Regression from Linear Models to Neural Networks: Double Descent, Active Learning, and Sampling

Von der Fakultät Mathematik und Physik sowie dem Stuttgarter Zentrum für Simulationswissenschaften der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

vorgelegt von

David Holzmüller

aus Stuttgart

Hauptberichter:	Prof. Dr. Ingo Steinwart
Mitberichter:	Prof. Dr. Matthias Hein
	Prof. Dr. Francis Bach
Tag der mündlichen Prüfung:	25.07.2023

Institut für Stochastik und Anwendungen

Universität Stuttgart

2023

Acknowledgments

First, I would like to express my deepest gratitude to my supervisor Prof. Dr. Ingo Steinwart for providing me the opportunity to pursue this doctorate degree, giving me the freedom to work on different topics, and frequently providing helpful feedback and advice. I also appreciate the countless meetings with a relaxed and humorous atmosphere as well as all the other support throughout my journey. I am also thankful to Prof. Dr. Francis Bach for accepting me as a visiting researcher, taking the time to supervise me, and examining this dissertation. In addition, I would like to thank Prof. Dr. Matthias Hein for examining this dissertation and participating in my thesis advisory committee, Prof. Dr. Philipp Hennig for also participating in the latter, Prof. Dr. Mathias Niepert for being my second advisor for my milestone presentation, and Prof. Dr. Bernard Haasdonk for agreeing to be an additional examiner at my defense.

I could not have undertaken this journey without funding by the German Research Foundation through the Cluster of Excellence "Data-Integrated Simulation Science (SimTech)" under Germany's Excellence Strategy - EXC 2075 - 390740016. Moreover, I am thankful to the IMPRS-IS graduate school for further support.

I am also very grateful to my colleagues and collaborators for supporting me in my scientific endeavors and making them more enjoyable. In this regard, I want to mention Dr. Viktor Zaverkin, Max Schölpple, Moritz Haas, Prof. Dr. Dirk Pflüger, Tizian Wenzel, Dr. Felix Dangel, Dr. Hans Kersting, Daniel Winkle, Aleksandar Arsenijevic, Manuel Nonnenmacher, Dr. Simon Fischer, Dr. Thomas Hamm, Dr. Ingrid Blaschzyk, Max Thannheimer, Carolin Forster, Jun.-Prof. Dr. Marco Oesting, Dr. Anna Maria Aulbach, Dr. Jürgen Dippon, and Elke Maurer. Especially, I would like to express my deepest gratitude to Dr. Viktor Zaverkin for our very fruitful and enjoyable collaboration, of which one of the papers ended up in this thesis, and the resulting friendship. I would also like to thank Dr. Robin Merkle, Dr. Thomas Hamm, and Dr. Viktor Zaverkin for helping me with the formatting and structure of this thesis, to Max Schölpple, Bernd Holzmüller, and Prof. Dr. Ingo Steinwart for feedback on the introductory part, and to Prof. Dr. Dominik Göddeke and Elke Gangl for feedback on formal requirements.

Last but not least, I would like to mention my family and friends. In particular, I would like to thank my grandparents, my sister, and especially my parents, for their continuous and unconditional support in ways that are too numerous to list here. They have contributed significantly to my well-being during my Ph.D. time as well as before.

Abstract

Deep learning, that is, the training of deep neural networks, is a powerful and successful approach to machine learning, allowing to fit complex models to various forms of data. On the other hand, the flexibility of neural networks also poses great challenges of both theoretical and computational nature, some of which are addressed in this thesis. In particular, we will consider the task of regression, where for a given input, a continuous output variable should be predicted.

On the theoretical side, it is still only partially understood why neural network optimization works so well, why optimized neural networks perform well on new data, or why certain neural networks work better than others. Neural networks can be studied in an under-parameterized regime, where the number of parameters is much smaller than the number of training samples, but also in an over-parameterized regime, where it is much bigger than the number of training samples. In particular, researchers have been puzzled by the so-called double descent phenomenon, where the accuracy of neural networks deteriorates as they leave the under-parameterized regime, and improves again when they become more over-parameterized. The double descent phenomenon has been proven for linear regression models under various assumptions. In Chapter 5, we prove a lower bound for the expected test error of unregularized linear regression that exhibits the typical double descent peak between the over- and under-parameterized regimes while using much weaker assumptions than previous work. We show that these assumptions are satisfied by a large class of input distributions and feature maps, and in particular for certain random deep neural network feature maps. Our results therefore also imply a lower bound with the typical double descent shape for deep neural networks where only the last layer is trained.

On the practical side, the non-linear nature and the large number of parameters of neural networks renders accurate and efficient uncertainty estimation difficult. Good uncertainty estimates can help to make a neural network more trustworthy, but they are also relevant for active learning, where the uncertainty estimates can be used to generate more relevant training data for the neural network. In Chapter 6, we study efficient batch mode active learning methods for regression with neural networks and propose a framework in which such methods can be constructed by selecting a base kernel, applying kernel transformations to the base kernel, and then using the resulting kernel in a kernel-based selection method. We provide new components in our framework, allowing to leverage linearizations of the full network efficiently through sketching and facilitating efficient distribution-aware data selection through a novel clustering method. Our accompanying open source code implements our framework in an efficient and flexible manner. We also provide a benchmark consisting of 15 tabular data sets on which our new components achieve state-of-the-art performance.

While our work on active learning uses simple uncertainty estimation methods for efficiency, it is interesting in some cases to obtain more accurate Bayesian uncertainty estimates using sampling methods. For neural networks, this requires the use of sampling algorithms for non-log-concave Gibbs distributions, for which only weak guarantees have been known. In Chapter 7, we conduct an extensive study of convergence rates of such algorithms depending on the smoothness of the log-density and the magnitude of its derivatives. First, we study the information-based complexity of the problem, showing that algorithms only constrained by the number of function evaluations can achieve the same convergence rates as for function approximation, and in some regimes even better rates. A similar picture emerges for algorithms that estimate the log-normalization constant of an unnormalized Gibbs distribution. We then study computational reductions between these two problems and optimization, and we investigate the use of function approximations to obtain faster convergence rates. Finally, we prove convergence rate bounds on several simple algorithms as well as a recent variational approach.

Kurzfassung

Tiefes Lernen, also das Trainieren tiefer neuronaler Netze, ist ein mächtiger und erfolgreicher Ansatz für maschinelles Lernen, der es erlaubt, komplexe Modelle an verschiedene Formen von Daten anzupassen. Allerdings stellt die Flexibilität neuronaler Netze aber auch große Herausforderungen sowohl theoretischer als auch rechentechnischer Art, von denen einige in dieser Dissertation adressiert werden. Insbesondere werden wir das Regressionsproblem betrachten, bei dem für eine gegebene Eingabe eine kontinuierliche Ausgabevariable vorhergesagt werden soll.

Auf der theoretischen Seite ist es immer noch nur teilweise verstanden, warum die Optimierung neuronaler Netze so gut funktioniert, warum optimierte neuronale Netze sich auf neuen Daten gut verhalten oder warum gewisse neuronale Netze besser funktionieren als andere. Neuronale Netze können in einem unterparametrisierten Regime untersucht werden, in dem ihre Parameterzahl deutlich kleiner ist als die Anzahl der Trainingsdaten, aber auch in einem überparametrisierten Regime, in dem die Parameterzahl deutlich größer ist als die Anzahl der Trainingsdaten. Insbesondere haben Forscher über das sogenannte Double-Descent-Phänomen gerätselt, bei dem die Genauigkeit von neuronalen Netzen schlechter wird, wenn sie das unterparametrisierte Regime verlassen, und wieder besser wird, wenn sie stärker überparametrisiert werden. Das Double-Descent-Phänomen wurde für lineare Regressionsmodelle unter verschiedenen Annahmen bewiesen. In Kapitel 5 beweisen wir eine untere Schranke an den erwarteten Testfehler unregularisierter linearer Regression, welcher den typischen Double-Descent-Hügel zwischen den über- und unterparametrisierten Regimes aufweist, aber dabei deutlich schwächere Annahmen macht als vorangegangene Arbeiten. Wir zeigen, dass diese Annahmen von einer großen Klasse an Eingabeverteilungen und *feature maps* erfüllt sind, und insbesondere auch von solchen *feature maps*, die durch gewisse zufällige tiefe neuronalen Netze gegeben sind. Unsere Resultate implizieren daher auch eine untere Schranke mit der typischen Double-Descent-Form für tiefe neuronale Netze, bei denen nur die letzte Schicht trainiert wird.

Auf der praktischen Seite macht die nichtlineare Natur und die große Parameterzahl neuronaler Netze eine genaue und effiziente Unsicherheitsschätzung schwierig. Eine gute Unsicherheitsschätzung kann helfen, ein neuronales Netz vertrauenswürdiger zu machen, aber sie ist auch relevant für aktives Lernen, bei dem die Unsicherheitsschätzung genutzt werden kann, um relevantere Trainingdaten für das neuronale Netz zu generieren. In Kapitel 6 untersuchen wir effiziente aktive Lernmethoden für Regression mit neuronalen Netzen und führen ein Schema ein, in dem solche Methoden konstruiert werden können durch die Wahl eines Basiskerns, die Anwendung von Kerntransformationen auf den Basiskern und die Nutzung des resultierenden Kerns in einer kernbasierten Auswahlmethode. Wir stellen neue Komponenten für unser Schema bereit, die es erlauben, eine Linearisierung des vollen Netzes auf effiziente Art durch *Sketching* zu nutzen, und eine effiziente und verteilungsbewusste Datenauswahl durch eine neue Clustering-Methode zu treffen. Unser begleitender öffentlicher Programmcode implementiert unser Schema auf effiziente und flexible Art. Wir stellen darüber hinaus einen Benchmark bereit, der aus 15 tabellarischen Datensätzen besteht, auf denen unsere neuen Komponenten eine neue Bestmarke in der Qualität der resultierenden Netze erreichen.

Während unsere Arbeit zu aktivem Lernen aus Effizienzgründen einfache Unsicherheitsschätzungsmethoden verwendet, ist es in einigen Fällen interessant, genauere Bayesianische Unsicherheitsschätzungen mit Sampling-Methoden zu erhalten. Für neuronale Netze erfordert dies die Nutzung von Sampling-Algorithmen für nicht log-konkave Gibbs-Verteilungen, für welche nur schwache Garantien bekannt waren. In Kapitel 7 führen wir eine umfangreiche Untersuchung der Konvergenzraten solcher Algorithmen durch, abhängig von der Glattheit der log-Dichte und der Größe ihrer Ableitungen. Zuerst untersuchen wir die informationsbasierte Komplexität des Problems und zeigen dabei, dass Algorithmen, die nur durch die Anzahl der Funktionsauswertungen beschränkt sind, die gleichen Konvergenzraten wie Funktionsapproximation erreichen können, und in manchen Regimes sogar bessere Raten. Ein ähnliches Bild ergibt sich für Algorithmen, die die log-Normalisierungskonstante einer unnormalisierten Gibbs-Verteilung schätzen. Danach untersuchen wir Reduktionen zwischen diesen zwei Problemen und Optimierung und wir untersuchen die Nutzung von Funktionsapproximationen, um schnellere Konvergenzraten zu bekommen. Schließlich beweisen wir Schranken an die Konvergenzraten verschiedener einfacher Algorithmen sowie eines kürzlich publizierten variationellen Ansatzes.

Contents

Part I

Introduction

Chapter 1

Introduction and Contribution of this Thesis

1.1 Introduction and Motivation

Through the widespread adoption of computers, smartphones, and the internet, large amounts of data are generated, processed, and stored. Machine learning (ML) algorithms can leverage increasing amounts of data to improve their performance at various tasks. In recent years, ML has enabled significant progress in areas such as image classification (Cireşan et al., 2012; Krizhevsky et al., 2012), image generation (Goodfellow et al., 2014), game playing (Mnih et al., 2015; Silver et al., 2016), audio generation (Oord et al., 2016), machine translation (Vaswani et al., 2017), language understanding (Devlin et al., 2019), text generation (Brown et al., 2020), and protein structure prediction (Jumper et al., 2021).

While many of these most prominent successes stem from the areas of computer science and signal processing, simulation science is also undergoing a revolution through the integration of ML techniques (Lavin et al., 2021). For example, neural networks (NNs) are being employed for the (data-integrated) solution of partial differential equations (PDEs) (Raissi et al., 2019) including the electronic Schrödinger equation (Pfau et al., 2020; Hermann et al., 2020), learning PDE terms and equations from data (Raissi et al., 2019; Rackauckas et al., 2020; Kirkpatrick et al., 2021), and the solution of inverse problems (Cranmer et al., 2020).

An important application of ML in simulation science is to approximate functions $f^* : \mathcal{X} \to \mathcal{Y}$ that are computed by expensive simulations through a *surrogate model* $f : \mathcal{X} \to \mathcal{Y}$ that can be evaluated more efficiently. Prominent applications of surrogate models include the prediction of forces in atomistic simulations (see e.g. Behler and Parrinello, 2007; Deringer et al., 2019) and the prediction of PDE solutions given initial conditions (Li et al., 2021).

In this thesis, we will first study a branch of ML called *supervised learning*, where an output (or *label*) $y \in \mathcal{Y}$ should be predicted from an input $\boldsymbol{x} \in \mathcal{X}$. Given a *training data set* $D = ((\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n))$ of input-output pairs, a supervised learning method typically outputs a function $f_D : \mathcal{X} \to \mathcal{Y}$ or a probabilistic model $P(y|\boldsymbol{x}, D)$. In particular, supervised learning can also be used to create data-driven surrogate models. The two

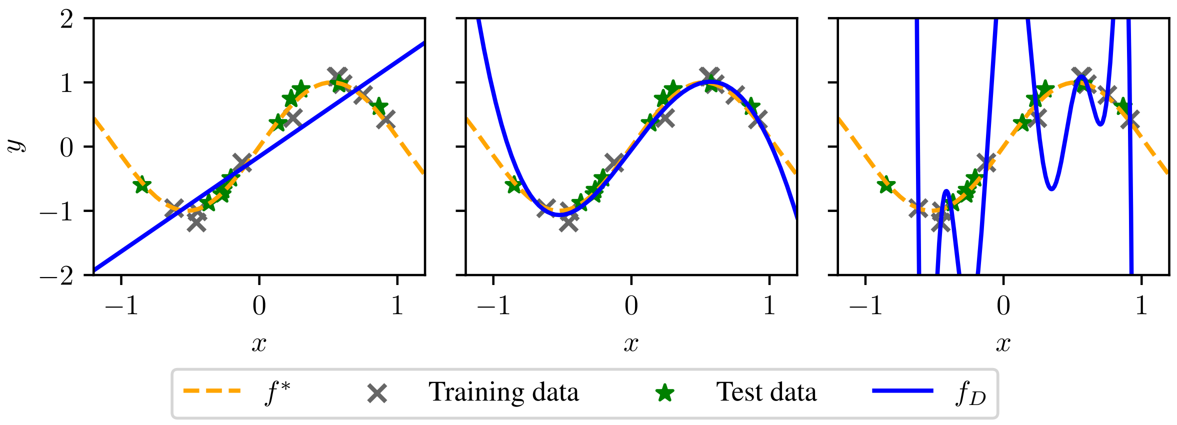


Figure 1.1: This figure shows training and test data generated from the function $f^*(x) = \sin(3x)$ with additive noise, as well as different functions f_D fitted to it. The function on the left does not generalize well due to underfitting. The function in the middle generalizes pretty well. The function on the right interpolates the training data, but generalizes poorly, which is an instance of overfitting.

most prominent problems in supervised learning are *classification*, where $\mathcal{Y}$ is discrete, and *regression*, where $\mathcal{Y}$ is continuous. In this thesis, we will focus on regression with $\mathcal{Y} = \mathbb{R}$.

To evaluate the learned function f_D, one can measure an average error on an independent test set $D_{\text{test}} = ((\tilde{\boldsymbol{x}}_1, \tilde{y}_1), \ldots, (\tilde{\boldsymbol{x}}_{n_{\text{test}}}, \tilde{y}_{n_{\text{test}}}))$. For example, a typical error measure for regression is the *mean squared error* (MSE) given by

$$\text{MSE} = \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} (\tilde{y}_i - f_D(\tilde{\boldsymbol{x}}_i))^2 ,$$

or the square root thereof, which is called *root mean squared error* (RMSE). Besides these *test error* measures, the average error on the training set can be measured as well. However, since the learned function f_D is often optimized to have low training error, the training error is not an unbiased estimate of the error that f_D will make on new data.

Informally speaking, if a learning method achieves low test error, it is said to *generalize*. Two complementary obstacles to generalization are *underfitting*, where the learning method does not achieve a low training error, and *overfitting*, where the learning method achieves a very low training error but a high test error. Underfitting typically occurs when a learning method chooses the function f_D from a limited set of functions that does not allow achieving low training error. Overfitting typically occurs when a learning method has too much freedom to choose the function f_D, such that the chosen function can fit the training data well but can exhibit irregular behavior outside of the training data. Underfitting and overfitting are visualized in Figure 1.1.

A particularly popular class of ML methods is given by deep learning methods, where the learned functions are represented by artificial (deep) NNs. NNs are parametric models, that is, functions $f_{\boldsymbol{\theta}}$ that depend on a vector of (learnable) parameters $\boldsymbol{\theta}$. They allow for great flexibility in specifying the functional form, also called *architecture*, of the learned function. For surrogate models, they also offer various possibilities to incorporate symmetries and other physical constraints. On the other hand, our theoretical understanding of why and under which assumptions NNs generalize is rather limited. For example, NNs often have many learnable parameters, such that one might expect them to overfit, but yet they often achieve a low test error in practice (Zhang et al., 2017). Moreover, many other questions are only partially understood, such as why the optimization of NNs works so well in practice, why NNs are so successful for high-dimensional data, or why and when certain architectures of NNs are better than others. We will discuss some aspects of the theory of NNs in Section 2.2.4 and contribute to this theory in Chapter 5.

To draw valid scientific conclusions, scientists need to be able to trust the output of a surrogate model. Besides theoretical guarantees and a low test error, a desirable property of a surrogate model is the ability to provide an estimate of the uncertainty of a prediction. This can be achieved by *Bayesian* ML methods, which typically do not provide a single function f_D but a distribution over functions f_D, from which uncertainty estimates can be derived. To obtain a prediction from a distribution of functions, one could try to sample some functions from the distribution using a sampling algorithm. Unfortunately, for Bayesian NNs, sampling is very computationally expensive, and practical convergence guarantees are lacking (Izmailov et al., 2021). We will discuss some aspects of Bayesian NNs in Section 2.2.4 as well as in Chapter 6 and take a closer look at the sampling problem in Chapter 7.

Another setting where uncertainty estimates are relevant is *active learning*. Active learning methods do not receive a fixed data set but are instead allowed to choose the inputs $\boldsymbol{x}_i$ for which the labels y_i are acquired. Active learning methods are also applicable to the construction of surrogate models since the simulated function f^* can typically be evaluated at arbitrary inputs $\boldsymbol{x}_i$. Compared to randomly sampling inputs $\boldsymbol{x}_i$, active learning methods can potentially enable reaching the same accuracy with fewer labels, and therefore fewer expensive evaluations of the simulation. Many active learning methods require uncertainty estimates to prioritize inputs or are even specifically designed for Bayesian models. We will study active learning for NNs in Chapter 6.

1.2 Contribution of this Thesis

This thesis consists of three papers. In the first paper (Holzmüller, 2021), we study the so-called double descent phenomenon, which describes that learning methods sometimes generalize badly when they have just enough parameters to perfectly fit the training data, but generalize better when they have even more parameters (Belkin et al., 2019). In particular, we study minimum-norm linear regression, a learning method that can be seen as a simple special case of neural network training, in the presence of noisy labels y_i. We show a lower bound for the expected test error that only depends on the noise variance, the number of samples n, and the input dimension. We show that our lower bound holds under non-degeneracy assumptions that are much weaker than the assumptions under which analytical formulas have been obtained. Especially, we show that our assumptions allow studying deep neural networks in a setting where only the last layer is trained. We also provide further theoretical and experimental results studying the sharpness of our lower bound.

In the second paper (Holzmüller et al., 2023), we study active learning for regression with NNs from a practical point of view. We study the batch active learning setting, where labels y_i for multiple inputs $\boldsymbol{x}_i$ are queried simultaneously. We introduce a kernel-based framework that allows building efficient batch active learning methods out of a variety of components. We provide open-source code containing efficient implementations of all components of our framework. Our framework allows us to reproduce existing methods and adapt classification methods to the regression setting. Moreover, we provide new components to the framework. We introduce a benchmark of 15 large tabular data sets on which we compare many different active learning methods. We find that a combination of our newly introduced components achieves the best average results on our benchmark.

In the third paper (Holzmüller and Bach, 2023), we study sampling methods on classes of probability distributions with smooth log-densities. Crucially, we do not assume that the log-density is convex, which is often not satisfied, e.g., for Bayesian NNs or equilibrium distributions of atomic configurations in molecular simulations. We first investigate the information-based complexity of sampling from such distributions, that is, the convergence rates that can be achieved when algorithms are only restricted by the number of points in which they can evaluate the log-density. For smooth log-densities, we show that sampling methods can in principle achieve good convergence rates through the use of surrogate models of the log-density. The same holds for methods that estimate the normalization constant of an unnormalized log-density. We study the relation between these two methods and optimization, obtaining various results for different metrics on probability distributions. Subsequently, we obtain various upper and lower bounds for the convergence rates of simple algorithms, such as sampling from piecewise constant approximations, approximating densities, simple Monte Carlo methods, and rejection sampling. We also study a variational approach towards estimating normalizing constants and show that all versions of it fail to come close to the information-based complexity rate for smooth log-densities.

Several papers by the author have not been included in this thesis. In particular, we have improved NN surrogate models for the prediction of potential energies and atomic forces of configurations of atoms (Zaverkin et al., 2021) and studied active learning (Zaverkin et al., 2022) and transfer learning (Zaverkin et al., 2023) for these surrogate models.

1.3 Overall Structure of the Work

The remainder of this thesis is structured as follows: We provide theoretical and methodological background for the rest of the thesis in Chapter 2. We then discuss the main results of this thesis in more detail in Chapter 3. In Chapter 4, we clarify the contributions of the author of this thesis to the results. Chapter 5 contains the first paper on double descent. Chapter 6 contains the second paper on active learning. Finally, Chapter 7 contains the third paper on sampling.

Chapter 2

Theoretical and Methodological Background

In many parts of this thesis, we are interested in the ability of learning algorithms to approximately reconstruct an unknown function $f : \mathcal{X} \to \mathcal{Y}$ from a data set $D = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)) \in (\mathcal{X} \times \mathcal{Y})^n$. Specifically, we will consider regression with $\mathcal{Y} = \mathbb{R}$. To study regression algorithms, we need to define how the reconstruction error will be measured and how we assume the data to be generated. A framework for modeling these aspects is given by statistical learning theory for supervised learning, which we will introduce in Section 2.1. We will then introduce specific methods for regression in Section 2.2, which are mainly relevant for Chapter 5 and Chapter 6. Finally, in Section 2.3, we will discuss Gibbs distributions and statistical distances, which will be relevant for Chapter 7.

2.1 Statistical Learning Theory

In this section, we will outline some basic assumptions and definitions from statistical learning theory that can be used to analyze learning methods. Our notation and exposition in this section roughly follow the one by Steinwart and Christmann (2008).

For supervised learning, we can consider the setting where we are given the following information:

- An input space $\mathcal{X}$, for example, $\mathcal{X} = \mathbb{R}^d$.
- An output space $\mathcal{Y}$, for example, $\mathcal{Y} = \mathbb{R}$.
- A loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$, for example, the square loss $L(y, t) = (y - t)^2$.
- A data set $D = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n))$, where $(\boldsymbol{x}_i, y_i)$ are i.i.d. samples from an unknown data-generating distribution P on $\mathcal{X} \times \mathcal{Y}$. We will sometimes use random variables $(X, Y) \sim P$ and we will denote the corresponding marginal distribution of X by P_X.

Technically, we need σ-Algebras on $\mathcal{X}$ and $\mathcal{Y}$ to define the measurability of functions $f : \mathcal{X} \to \mathcal{Y}$ and to define a probability space on $\mathcal{X} \times \mathcal{Y}$. In the following, measurability will be implicitly assumed as needed even if not mentioned explicitly, for details see the book by Steinwart and Christmann (2008).

Risk For a measurable function $f : \mathcal{X} \to \mathcal{Y}$, we define the *(population) risk* $\mathcal{R}_{L,P}(f)$ and the *empirical risk* $\mathcal{R}_{L,D}(f)$ by

$$\mathcal{R}_{L,P}(f) := \int L(y, f(\boldsymbol{x}))\,\mathrm{d}P(\boldsymbol{x}, y)\ , \qquad \mathcal{R}_{L,D}(f) := \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i))\ .$$

These two risks specify the average loss under P or D when predicting y with $f(\boldsymbol{x})$. The optimal achievable population risk is called *Bayes risk* and given by

$$\mathcal{R}^*_{L,P} := \inf_{f:\mathcal{X}\to\mathcal{Y}\ \text{measurable}} \mathcal{R}_{L,P}(f)\ .$$

If the infimum above is achieved by a unique (up to P_X-null sets) function f, we denote it by $f^*_{L,P}$. The *excess risk*

$$\mathcal{R}_{L,P}(f) - \mathcal{R}^*_{L,P}$$

quantifies the suboptimality of a given function f.

Example 2.1.1 (Binary classification)**.** In binary classification, we have two classes and want to predict which one of them an input belongs to. We can formalize the two classes as $\mathcal{Y} = \{-1, 1\}$ and define the classification error

$$L(y,t) := \begin{cases} 0 & , y = t \\ 1 & , y \neq t\ . \end{cases}$$

The population risk $\mathcal{R}_{L,P}(f)$ denotes the probability that a sample will be misclassified by f. Using the conditional distribution $P(Y = y|X = \boldsymbol{x})$, we can write the Bayes optimal classifier as

$$f^*_{L,P}(\boldsymbol{x}) = \operatorname*{argmax}_{y\in\{-1,1\}} P(Y = y \mid X = \boldsymbol{x})\ .$$

Of course, $f^*_{L,P}$ is only unique up to P_X-null sets if $P(Y = 1 \mid X = \boldsymbol{x}) \neq 1/2$ for P_X-almost all $\boldsymbol{x}$. The Bayes risk quantifies the average amount of noise on the labels and is given by

$$\mathcal{R}^*_{L,P} = \mathbb{E}_{\boldsymbol{x}\sim P_X} \min_{y\in\{-1,1\}} P(Y = y \mid X = \boldsymbol{x})\ .$$

Since the classification error is difficult to optimize using gradient-based optimization techniques, surrogate loss functions are often employed, see e.g. Steinwart and Christmann (2008). However, we will not need surrogate loss functions for regression. ◀

Example 2.1.2 (Regression with square loss)**.** Let L be the square loss $L(y,t) = (y-t)^2$ with $\mathcal{Y} = \mathbb{R}$. In this case, if $\mathbb{E}Y^2 < \infty$, it can be shown that the function $f^*_{L,P}$ exists and is given by the conditional expectation $f^*_{L,P}(\boldsymbol{x}) = \mathbb{E}_P[Y|X = \boldsymbol{x}]$. Moreover, an elementary calculation shows that the excess risk of a function f is given by

$$\mathcal{R}_{L,P}(f) - \mathcal{R}^*_{L,P} = \mathbb{E}_{\boldsymbol{x}\sim P_X}[(f(\boldsymbol{x}) - f^*_{L,P}(\boldsymbol{x}))^2] = \|f - f^*_{L,P}\|^2_{L_2(P_X)}\ .$$ ◀

Consistency In practice, the risk or the excess risk cannot be computed since the data-generating distribution P is unknown. However, if a test set D_{test} drawn from P independently of D is available, the population risk $\mathcal{R}_{L,P}(f)$ can be estimated by $\mathcal{R}_{L,D_{\text{test}}}(f)$. In the limit $n \to \infty$ of infinite training data, we would like the excess risk of a learning algorithm to converge to zero. This is formalized by the notion of *consistency*:

Definition 2.1.3 (Learning methods and (universal) consistency)**.** A *learning method* $\mathcal{L} = (\mathcal{L}_n)_{n \in \mathbb{N}_{\geq 1}}$ is a sequence of (measurable) functions $\mathcal{L}_n$ that map a data set $D \in (\mathcal{X} \times \mathcal{Y})^n$ to a (measurable) function $f_D : \mathcal{X} \to \mathcal{Y}$. A learning method is called *(L-risk) consistent* for a distribution P on $\mathcal{X} \times \mathcal{Y}$ if for all $\varepsilon > 0$, we have

$$\lim_{n \to \infty} P^n(\{D \in (\mathcal{X} \times \mathcal{Y})^n \mid \mathcal{R}_{L,P}(f_D) > \mathcal{R}^*_{L,P} + \varepsilon\}) = 0 \ ,$$

where P^n denotes the distribution of n i.i.d. random samples from P. In other words, this means that the excess risk on the random function f_D converges to zero in probability as we let the number n of samples go to infinity. A learning method is called *universally (L-risk) consistent* if it is (L-risk) consistent for every distribution P on $\mathcal{X} \times \mathcal{Y}$. ◀

Remark 2.1.4. The definition above considers deterministic learning methods, but it is straightforward to extend it to stochastic learning methods, where the mapping $D \mapsto f_D$ is not deterministic. As we will discuss in Section 2.2.4, this is relevant for neural networks, which often use randomized initialization and training methods.

Besides stochastic convergence of the excess risk, one can consider convergence in expectation or almost sure convergence. The corresponding notions of consistency are called *weak consistency* and *strong consistency*, respectively (Györfi et al., 2002).

Universal consistency has been first established for the k-nearest neighbors method (Stone, 1977) and has since been established for many more learning methods (see Devroye et al., 1996; Györfi et al., 2002; Steinwart and Christmann, 2008). Besides consistency, one might be interested in the rate of convergence of the excess risk, which is known as a *learning rate*. Unfortunately, obtaining non-trivial learning rates requires additional assumptions on the distribution P (Devroye, 1982; Devroye et al., 1996). ◀

Empirical risk minization A common approach towards constructing learning methods is called *empirical risk minimization* (ERM). While the risk $\mathcal{R}_{L,P}$ depends on the unknown distribution P and hence cannot be optimized directly, we can instead optimize the *empirical risk* $\mathcal{R}_{L,D}$. For example, for the square loss $L(y,t) = (y-t)^2$, $\mathcal{R}_{L,D}(f)$ is the mean squared error (MSE) of f on the training data D. After choosing a suitable set $\mathcal{F}$ of functions from $\mathcal{X}$ to $\mathcal{Y}$, we can then consider the empirical risk minimizer (ERM)

$$f_D = \operatorname*{arg\,min}_{f \in \mathcal{F}} \mathcal{R}_{L,D}(f) \ ,$$

provided that it (uniquely) exists. A common way to establish upper bounds on the excess risk for such methods is to show that

(a) there is some $f \in \mathcal{F}$ with low population risk $\mathcal{R}_{L,P}(f)$, (Approximation error bound)
(b) with high probability, for all $f \in \mathcal{F}$, the empirical risk $\mathcal{R}_{L,D}(f)$ is close to the population risk $\mathcal{R}_{L,P}(f)$. (Estimation error bound)

To achieve consistency, the approximation error bound (a) requires $\mathcal{F}$ to grow with n. On the other hand, the estimation error bound (b) requires the function class $\mathcal{F}$ to be not too large relative to n. Large function classes $\mathcal{F}$ can lead to overfitting, where the ERM f_D has much lower empirical risk than population risk. To prevent this, it is common to consider instead the *regularized ERM*

$$f_D = \operatorname*{arg\,min}_{f \in \mathcal{F}} \mathcal{R}_{L,D}(f) + \lambda \Omega(f) ~,$$

where $\Omega(f) \geq 0$ is some measure of the complexity of f and $\lambda > 0$ can be used to adjust the strength of the regularization. We will later see examples of such regularizers of the form $\Omega(f) = \|f\|^2$ for some Hilbert space norm $\|\cdot\|$.

An important class of learning methods is given by parametric methods, which consider classes of functions such as $\mathcal{F} = \{f_{\boldsymbol{\theta}} \mid \boldsymbol{\theta} \in \mathbb{R}^p\}$, where $\boldsymbol{\theta} \mapsto f_{\boldsymbol{\theta}}$ is a parametric function. The size of the function class is then related to the number of parameters p. We will refer to methods with $p \ll n$ as *under-parameterized* and methods with $p \gg n$ as *over-parameterized*. Typically, good estimation error bounds in the sense above for unregularized methods can only be given in the under-parameterized case, while guarantees for over-parameterized methods require the use of regularization. However, the necessity of regularization for over-parameterized methods has recently been questioned due to the successes of over-parameterized NNs (Zhang et al., 2017; Belkin et al., 2018). We will therefore also study unregularized methods in the over-parameterized regime in Section 2.2.1 and Chapter 5.

Bias-Variance decomposition We now investigate how the expected excess risk $\mathbb{E}_D \mathcal{R}_{L,P}(f_D) - \mathcal{R}^*_{L,P}$ can be further decomposed into non-negative parts. For the square loss, this can be done using a bias-variance decomposition (see e.g. Adlam and Pennington, 2020b), and we consider a straightforward generalization to convex loss functions using Jensen's inequality:

Theorem 2.1.5 (Jensen's inequality, see e.g. Theorem 7.11 in Klenke (2014))**.** *Let $G \subseteq \mathbb{R}^d$ be convex, let $\varphi : G \to \mathbb{R}$ be convex, and let X be an $\mathbb{R}^d$-valued random variable with $\mathbb{E}\|X\| < \infty$ and $P(X \in G) = 1$. Then,*

$$\mathbb{E}[\varphi(X)] \geq \varphi(\mathbb{E}[X]) ~.$$

Jensen's inequality is useful in many contexts, and it will also be one of our central tools in the proof of Theorem 5.4.3 in Chapter 5. In the following, we use $\mathbb{E}_D f_D$ to denote the point-wise expectation of the function f_D over the draw of the data set $D \sim P^n$. We consider the following decomposition of the expected excess risk:

$$\mathbb{E}_D \mathcal{R}_{L,P}(f_D) - \mathcal{R}^*_{L,P} = (\mathbb{E}_D[\mathcal{R}_{L,P}(f_D)] - \mathcal{R}_{L,P}(\mathbb{E}_D f_D)) + \left(\mathcal{R}_{L,P}(\mathbb{E}_D f_D) - \mathcal{R}^*_{L,P}\right) . \quad (2.1)$$

The last term $\mathcal{R}_{L,P}(\mathbb{E}_D f_D) - \mathcal{R}^*_{L,P}$ is always non-negative by definition of $\mathcal{R}^*_{L,P}$. Now suppose that the loss function $(y,t) \mapsto L(y,t)$ is convex in t for every y. If the risks are finite, we can use the Fubini-Tonelli theorem to exchange the order of integration and obtain

$$\mathbb{E}_D[\mathcal{R}_{L,P}(f_D)] - \mathcal{R}_{L,P}(\mathbb{E}_D f_D) = \int (\mathbb{E}_D[L(y, f_D(\boldsymbol{x}))] - L(y, \mathbb{E}_D f_D(\boldsymbol{x})))\, \mathrm{d}P(\boldsymbol{x}, y)$$

$$\geq \int 0 \,\mathrm{d}P(\boldsymbol{x}, y) = 0 \ ,$$

where we bounded the integrand using Jensen's inequality with $\varphi(t) = L(y, t)$.

For convex loss functions, the argument above shows that Eq. (2.1) decomposes the expected excess risk into two separate non-negative contributions. In the case of the square loss $L(y, t) = (y - t)^2$, we can further simplify

$$\begin{aligned}\mathbb{E}_D[L(y, f_D(\boldsymbol{x}))] - L(y, \mathbb{E}_D f_D(\boldsymbol{x})) &= \mathbb{E}_D[(y - f_D(\boldsymbol{x}))^2] - (\mathbb{E}_D[y - f_D(\boldsymbol{x})])^2 \\ &= \mathrm{Var}_D(y - f_D(\boldsymbol{x})) = \mathrm{Var}_D(f_D(\boldsymbol{x})) \ ,\end{aligned}$$

which yields

$$\mathbb{E}_D[\mathcal{R}_{L,P}(f_D)] - \mathcal{R}_{L,P}(\mathbb{E}_D f_D) = \mathbb{E}_{\boldsymbol{x} \sim P_X} \mathrm{Var}_D(f_D(\boldsymbol{x})) \ .$$

Hence, for the square loss, Eq. (2.1) yields the classical bias-variance decomposition.

When decomposing the data set $D = (\boldsymbol{X}, \boldsymbol{y})$ into the inputs $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ and the labels $(y_1, \ldots, y_n)$, we can further decompose the variance term as

$$\begin{aligned}\mathbb{E}_D[\mathcal{R}_{L,P}(f_D)] - \mathcal{R}_{L,P}(\mathbb{E}_D f_D) = &\big(\mathbb{E}_{\boldsymbol{X},\boldsymbol{y}}[\mathcal{R}_{L,P}(f_{(\boldsymbol{X},\boldsymbol{y})})] - \mathbb{E}_{\boldsymbol{X}}[\mathcal{R}_{L,P}(\mathbb{E}_{\boldsymbol{y}|\boldsymbol{X}} f_{(\boldsymbol{X},\boldsymbol{y})})]\big) \\ + \ &\big(\mathbb{E}_{\boldsymbol{X}}[\mathcal{R}_{L,P}(\mathbb{E}_{\boldsymbol{y}|\boldsymbol{X}} f_{(\boldsymbol{X},\boldsymbol{y})})] - \mathcal{R}_{L,P}(\mathbb{E}_{\boldsymbol{X},\boldsymbol{y}} f_{(\boldsymbol{X},\boldsymbol{y})})\big) \ .\end{aligned}$$

Again, both terms are non-negative for convex loss functions thanks to Jensen's inequality. In Chapter 5, we will prove a lower bound for the first term

$$\mathcal{E}_{\text{Noise}} := \mathbb{E}_{\boldsymbol{X},\boldsymbol{y}}[\mathcal{R}_{L,P}(f_{(\boldsymbol{X},\boldsymbol{y})})] - \mathbb{E}_{\boldsymbol{X}}[\mathcal{R}_{L,P}(\mathbb{E}_{\boldsymbol{y}|\boldsymbol{X}} f_{(\boldsymbol{X},\boldsymbol{y})})] \ ,$$

which can be seen as the contribution of the label noise to the expected excess risk. In the case where the learned function f_D depends on an additional source of randomness, for example, a random feature map or random initialization, one obtains a further term in the decomposition above. Such decompositions have been used, for example, by d'Ascoli et al. (2020) and Adlam and Pennington (2020b).

2.2 Specific Regression Methods

In the following, we will introduce some regression methods relevant to this thesis. We will start with plain linear regression in Section 2.2.1, introduce feature maps and kernels in Section 2.2.2, and explain Bayesian linear regression in Section 2.2.3. Finally, we will discuss neural networks in Section 2.2.4.

2.2.1 Linear Regression

In the following, we will introduce linear regression, which is one of the simplest regression methods. For alternative expositions of linear regression, we refer to standard textbooks (e.g. Hastie et al., 2009; Bishop, 2006). We consider the class of linear functions from $\mathcal{X} = \mathbb{R}^d$ to $\mathcal{Y} = \mathbb{R}$ given by

$$\mathcal{F}_{\text{lin}} := \{f : \mathcal{X} \to \mathcal{Y}, \boldsymbol{x} \mapsto \boldsymbol{x}^\top \boldsymbol{\theta} \mid \boldsymbol{\theta} \in \mathbb{R}^d\} \ .$$

We consider the square loss $L(y,t) = (y-t)^2$ and, for a function $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\theta}$, the ridge (or Tikhonov) regularization $\Omega(f_{\boldsymbol{\theta}}) \coloneqq \|\boldsymbol{\theta}\|_2^2$. By defining

$$\boldsymbol{X} \coloneqq \begin{pmatrix} \boldsymbol{x}_1^\top \\ \vdots \\ \boldsymbol{x}_n^\top \end{pmatrix} \in \mathbb{R}^{n\times d}, \qquad \boldsymbol{y} \coloneqq \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^n ,$$

we can then write the regularized empirical risk as follows, using λ/n instead of λ to obtain more convenient expressions later on:

$$\begin{aligned} \mathcal{L}_\lambda(\boldsymbol{\theta}) &\coloneqq \mathcal{R}_{L,D}(f_{\boldsymbol{\theta}}) + \frac{\lambda}{n}\Omega(f_{\boldsymbol{\theta}}) \\ &= \frac{1}{n}\sum_{i=1}^n (y_i - \boldsymbol{x}_i^\top \boldsymbol{\theta})^2 + \frac{\lambda}{n}\|\boldsymbol{\theta}\|_2^2 \\ &= \frac{1}{n}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}\|_2^2 + \frac{\lambda}{n}\|\boldsymbol{\theta}\|_2^2 . \end{aligned} \tag{2.2}$$

The expression above is quadratic in $\boldsymbol{\theta}$ and its gradient and Hessian are given by

$$\begin{aligned} \nabla\mathcal{L}_\lambda(\boldsymbol{\theta}) &= \frac{2}{n}\left(\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}) + \lambda\boldsymbol{\theta}\right) = \frac{2}{n}\left(\left(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I}\right)\boldsymbol{\theta} - \boldsymbol{X}^\top\boldsymbol{y}\right) , \\ H\mathcal{L}_\lambda(\boldsymbol{\theta}) &= \frac{2}{n}\left(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I}\right) , \end{aligned} \tag{2.3}$$

where $\boldsymbol{I}$ denotes an identity matrix of matching size.

Regularized solution If $\lambda > 0$, the Hessian is positive definite. By setting the gradient to zero, we can therefore obtain the minimizer

$$\boldsymbol{\theta}_\lambda^* \coloneqq (\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^\top\boldsymbol{y} .$$

Lemma 2.2.1 (Over-parameterized linear regression formula)**.** *We have the alternative representation*

$$\boldsymbol{\theta}_\lambda^* = \boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{X}^\top + \lambda\boldsymbol{I})^{-1}\boldsymbol{y} .$$

Proof. From the Woodbury matrix identity

$$(\boldsymbol{A} + \boldsymbol{U}\boldsymbol{C}\boldsymbol{V})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{U}(\boldsymbol{C}^{-1} + \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U})^{-1}\boldsymbol{V}\boldsymbol{A}^{-1} ,$$

we obtain

$$\begin{aligned} (\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I})^{-1} &= (\lambda\boldsymbol{I} + \boldsymbol{X}^\top\boldsymbol{I}\boldsymbol{X})^{-1} \\ &= \lambda^{-1}\boldsymbol{I} - (\lambda^{-1}\boldsymbol{I})\boldsymbol{X}^\top(\boldsymbol{I} + \boldsymbol{X}(\lambda^{-1}\boldsymbol{I})\boldsymbol{X}^\top)^{-1}\boldsymbol{X}(\lambda^{-1}\boldsymbol{I}) \\ &= \lambda^{-1}(\boldsymbol{I} - \boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{X}^\top + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}) . \end{aligned} \tag{2.4}$$

Using $\boldsymbol{I} = (\boldsymbol{X}\boldsymbol{X}^\top + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}\boldsymbol{X}^\top + \lambda(\boldsymbol{X}\boldsymbol{X}^\top + \lambda\boldsymbol{I})^{-1}$, we obtain

$$\begin{aligned} (\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^\top &= \lambda^{-1}\boldsymbol{X}^\top(\boldsymbol{I} - (\boldsymbol{X}\boldsymbol{X}^\top + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}\boldsymbol{X}^\top) \\ &= \boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{X}^\top + \lambda\boldsymbol{I})^{-1} . \end{aligned}$$

□

Unregularized solutions For $\lambda = 0$, the situation can be more difficult because there are multiple empirical risk minimizers if $\boldsymbol{X}^\top \boldsymbol{X}$ is not invertible. This situation will be relevant in Chapter 5 when studying over-parameterized models. We now show that, as mentioned by Hastie et al. (2022), several approaches towards defining a unique solution lead to the same parameters being selected. This is also relevant for the study of over-parameterized neural networks, where the global optimum of the empirical risk minimization problem is usually not unique and different global optima can have very different generalization properties (Mücke and Steinwart, 2019).

Theorem 2.2.2 (Equality of unregularized solutions)**.** *Let $\boldsymbol{X} \in \mathbb{R}^{n\times d}, \boldsymbol{y} \in \mathbb{R}^n$. Consider*

(a) the zero-regularization limit $\boldsymbol{\theta}_0^* := \lim_{\lambda \searrow 0} \boldsymbol{\theta}_\lambda^*$,
(b) the pseudoinverse solution $\boldsymbol{\theta}^+ = \boldsymbol{X}^+\boldsymbol{y}$, where $\boldsymbol{X}^+$ is the Moore-Penrose pseudoinverse of $\boldsymbol{X}$,
(c) the minimum-norm solution

$$\boldsymbol{\theta}^* := \operatorname*{arg\,min}_{\boldsymbol{\theta}\in B} \|\boldsymbol{\theta}\|_2 , \qquad B := \{\boldsymbol{\theta} \in \mathbb{R}^d \mid \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}\|_2^2 = \min_{\tilde{\boldsymbol{\theta}}\in\mathbb{R}^d} \|\boldsymbol{y} - \boldsymbol{X}\tilde{\boldsymbol{\theta}}\|_2^2\} ,$$

and
(d) the limit of zero-initialized gradient descent $\boldsymbol{\theta}_{\eta,\infty} := \lim_{k\to\infty} \boldsymbol{\theta}_k$, where

$$\boldsymbol{\theta}_{k+1} := \boldsymbol{\theta}_k - \eta \nabla \mathcal{L}(\boldsymbol{\theta}_k), \qquad \boldsymbol{\theta}_0 := \boldsymbol{0} \in \mathbb{R}^d ,$$

for $\eta \in (0, n/\|\boldsymbol{X}\|_2^2)$, where $\|\boldsymbol{X}\|_2$ is the largest singular value of $\boldsymbol{X}$.

Then, $\boldsymbol{\theta}_0^*, \boldsymbol{\theta}^+, \boldsymbol{\theta}^*$, and $\boldsymbol{\theta}_{\eta,\infty}$ are well-defined and $\boldsymbol{\theta}_0^* = \boldsymbol{\theta}^+ = \boldsymbol{\theta}^* = \boldsymbol{\theta}_{\eta,\infty}$.

Proof.

(a) To show that the limit is well-defined, we consider a singular value decomposition of $\boldsymbol{X}$:

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top,$$

where $\boldsymbol{U} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{V} \in \mathbb{R}^{d\times d}$ are orthogonal matrices and $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times d}$ is a diagonal matrix with non-negative diagonal elements $\sigma_1 \geq \ldots \geq \sigma_{\min\{n,d\}}$ called singular values. We can then write

$$\begin{aligned}(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^\top &= (\boldsymbol{V}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top + \lambda\boldsymbol{I})^{-1}\boldsymbol{V}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top \\ &= (\boldsymbol{V}(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma} + \lambda\boldsymbol{I})\boldsymbol{V}^\top)^{-1}\boldsymbol{V}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top \\ &= \boldsymbol{V}(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top .\end{aligned}$$

Here, the matrix $(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Sigma}^\top \in \mathbb{R}^{d\times n}$ is diagonal and its diagonal entries $\sigma_i/(\sigma_i^2 + \lambda)$ converge to $1/\sigma_i$ for $\lambda \searrow 0$, where we define $1/0 := 0$. Let $\boldsymbol{\Sigma}^+$ denote the limiting matrix. Then, the limit

$$\lim_{\lambda\searrow 0}(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^\top = \boldsymbol{V}\boldsymbol{\Sigma}^+\boldsymbol{U}^\top$$

exists, hence $\boldsymbol{\theta}_0^*$ is well-defined.

(b) The Moore-Penrose pseudoinverse can be defined as $\boldsymbol{X}^+ = \boldsymbol{V}\boldsymbol{\Sigma}^+\boldsymbol{U}^\top$ (see e.g. the proof of Theorem 5.2.1 in Wang et al., 2018), which shows $\boldsymbol{\theta}^+ = \boldsymbol{\theta}_0^*$.

(c) A proof that $\boldsymbol{\theta}^*$ is well-defined and equal to $\boldsymbol{\theta}^+$ is given in Theorem 1.1.6 in Wang et al. (2018); we give an alternative proof here. First, we show that $\boldsymbol{\theta}_0^* \in B$ by contradiction. Suppose that $\boldsymbol{\theta}_0^* \notin B$, such that there exists $\tilde{\boldsymbol{\theta}}$ with $\mathcal{L}_0(\tilde{\boldsymbol{\theta}}) < \mathcal{L}_0(\boldsymbol{\theta}_0^*)$. But then,

$$\lim_{\lambda \searrow 0} \mathcal{L}_\lambda(\tilde{\boldsymbol{\theta}}) = \mathcal{L}_0(\tilde{\boldsymbol{\theta}}) < \mathcal{L}_0(\boldsymbol{\theta}_0^*) = \lim_{\lambda \searrow 0} \mathcal{L}_\lambda(\boldsymbol{\theta}_\lambda^*) ,$$

which contradicts the optimality of $\boldsymbol{\theta}_\lambda^*$ for $\lambda > 0$. Hence, $\boldsymbol{\theta}_0^* \in B$.
It remains to show that $\boldsymbol{\theta}_0^*$ is the minimum-norm element in B. Identifying the matrix $\boldsymbol{X}$ with its associated linear map, we denote its range by $R(\boldsymbol{X})$, its null space by $N(\boldsymbol{X})$, and the orthogonal complement of $N(\boldsymbol{X})$ by $N(\boldsymbol{X})^\perp$. Since $R(\boldsymbol{X})$ is a linear space and $\boldsymbol{\theta}_0^* \in B$, $\boldsymbol{X}\boldsymbol{\theta}_0^*$ must be the orthogonal projection of $\boldsymbol{y}$ onto $R(\boldsymbol{X})$. Then, for any $\boldsymbol{\theta} \in \mathbb{R}^d$, Pythagoras' theorem yields $\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}\|_2^2 = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}_0^*\|_2^2 + \|\boldsymbol{X}\boldsymbol{\theta}_0^* - \boldsymbol{X}\boldsymbol{\theta}\|_2^2$, which implies

$$B = \{\boldsymbol{\theta} \in \mathbb{R}^d \mid \boldsymbol{X}\boldsymbol{\theta} = \boldsymbol{X}\boldsymbol{\theta}_0^*\} = \boldsymbol{\theta}_0^* + N(\boldsymbol{X}) .$$

On the other hand, for $\lambda > 0$, Lemma 2.2.1 yields

$$\boldsymbol{\theta}_\lambda^* = \boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{X}^\top + \lambda \boldsymbol{I})^{-1}\boldsymbol{y} \in R(\boldsymbol{X}^\top) = N(\boldsymbol{X})^\perp ,$$

which yields $\boldsymbol{\theta}_0^* \in N(\boldsymbol{X})^\perp$. This means that $\boldsymbol{\theta}_0^*$ is the minimum-norm element from B.

(d) Since $\boldsymbol{\theta}^*$ is optimal, we have $\boldsymbol{0} = \nabla\mathcal{L}(\boldsymbol{\theta}^*) = \frac{2}{n}(\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\theta}^* - \boldsymbol{X}^\top\boldsymbol{y})$. We can therefore rewrite the gradient descent update as

$$\boldsymbol{\theta}_{k+1} \coloneqq \boldsymbol{\theta}_k - \eta\nabla\mathcal{L}_\lambda(\boldsymbol{\theta}_k) = \boldsymbol{\theta}_k - \eta\frac{2}{n}\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{\theta}_k - \boldsymbol{y}) = \boldsymbol{\theta}_k - \eta\frac{2}{n}\boldsymbol{X}^\top\boldsymbol{X}(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*) .$$

Consequently, the difference to the optimum $\bar{\boldsymbol{\theta}}_k \coloneqq \boldsymbol{\theta}_k - \boldsymbol{\theta}^*$ satisfies

$$\bar{\boldsymbol{\theta}}_{k+1} = \bar{\boldsymbol{\theta}}_k - \eta\frac{2}{n}\boldsymbol{X}^\top\boldsymbol{X}\bar{\boldsymbol{\theta}}_k = \left(\boldsymbol{I} - \eta\frac{2}{n}\boldsymbol{X}^\top\boldsymbol{X}\right)\bar{\boldsymbol{\theta}}_k .$$

Using the singular value decomposition $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ from (a), we have the diagonalization $\boldsymbol{X}^\top\boldsymbol{X} = \boldsymbol{V}\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}\boldsymbol{V}^\top$ with eigenvalues

$$\lambda_1 = \sigma_1^2, \ldots, \lambda_{\min\{n,d\}} = \sigma_{\min\{n,d\}}^2, \lambda_{\min\{n,d\}+1} = 0, \ldots, \lambda_d = 0 .$$

Therefore,

$$\begin{aligned}\bar{\boldsymbol{\theta}}_k &= \left(\boldsymbol{I} - \eta\frac{2}{n}\boldsymbol{X}^\top\boldsymbol{X}\right)^k \bar{\boldsymbol{\theta}}_0 = \boldsymbol{V}\left(\boldsymbol{I} - \eta\frac{2}{n}\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}\right)^k \boldsymbol{V}^\top\bar{\boldsymbol{\theta}}_0 \\ &= \boldsymbol{V}\operatorname{diag}((1-\alpha\lambda_1)^k, \ldots, (1-\alpha\lambda_d)^k)\boldsymbol{V}^\top\bar{\boldsymbol{\theta}}_0 .\end{aligned}$$

Now, since we assumed $\eta < n/\lambda_1$, we have $1 - \alpha\lambda_i \in (-1, 1]$ for all i and $1 - \alpha\lambda_i = 1$ iff $i > r$, where r is the rank of $\boldsymbol{X}^\top\boldsymbol{X}$. Hence, we obtain in the limit

$$\lim_{k\to\infty} \bar{\boldsymbol{\theta}}_k = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^\top\bar{\boldsymbol{\theta}}_0, \qquad \boldsymbol{D} \coloneqq \operatorname{diag}(\underbrace{0, \ldots, 0}_{r \text{ zeros}}, 1, \ldots, 1) .$$

We can compute $N(\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^\top) = R(\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^\top)^\perp = N(\boldsymbol{X}^\top\boldsymbol{X})^\perp = N(\boldsymbol{X})^\perp$. From our proof of (c), it follows that $\bar{\boldsymbol{\theta}}_0 = \boldsymbol{0} - \boldsymbol{\theta}^* \in N(\boldsymbol{X})^\perp$. Therefore, we have $\lim_{k\to\infty} \boldsymbol{\theta}_k - \boldsymbol{\theta}^* = \lim_{k\to\infty} \bar{\boldsymbol{\theta}}_k = \boldsymbol{0}$. □

The Moore-Penrose pseudoinverse satisfies many properties, cf. Section 1.1.1 in Wang et al. (2018), for example

- $\boldsymbol{X}^+ = \boldsymbol{X}^{-1}$ whenever $\boldsymbol{X}^{-1}$ exists, and
- $\boldsymbol{X}^+ = (\boldsymbol{X}^\top\boldsymbol{X})^+\boldsymbol{X}^\top = \boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{X}^\top)^+$.

Hence, the following formula is valid in both cases $\lambda > 0$ and $\lambda = 0$:

$$\boldsymbol{\theta}^*_\lambda = (\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I})^+\boldsymbol{X}^\top\boldsymbol{y} = \boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{X}^\top + \lambda\boldsymbol{I})^+\boldsymbol{y}\ . \tag{2.5}$$

2.2.2 Feature Maps and Kernels

A standard trick to use linear regression to learn non-linear functions is to consider functions of the form $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \phi(\boldsymbol{x})^\top\boldsymbol{\theta}$, where $\phi : \mathbb{R}^d \to \mathbb{R}^p$ is a fixed (non-linear) function called *feature map*. All results of Section 2.2.1 can then be applied to the transformed data $\tilde{\boldsymbol{x}}_i \coloneqq \phi(\boldsymbol{x}_i)$. In particular, we can define the feature matrix

$$\phi(\boldsymbol{X}) \coloneqq \begin{pmatrix} \phi(\boldsymbol{x}_1)^\top \\ \vdots \\ \phi(\boldsymbol{x}_n)^\top \end{pmatrix}$$

and use Eq. (2.5) to obtain the following general formula for the regularized or minimum-norm ERM solution with $\lambda \geq 0$:

$$\boldsymbol{\theta}^*_\lambda = (\phi(\boldsymbol{X})^\top\phi(\boldsymbol{X}) + \lambda\boldsymbol{I})^+\phi(\boldsymbol{X})^\top\boldsymbol{y} = \phi(\boldsymbol{X})^\top(\phi(\boldsymbol{X})\phi(\boldsymbol{X})^\top + \lambda\boldsymbol{I})^+\boldsymbol{y}\ . \tag{2.6}$$

Example 2.2.3 (Polynomial regression)**.** Suppose that $\mathcal{X} = \mathbb{R}$. For a given dimension $p \in \mathbb{N}_{\geq 1}$, we can define the polynomial feature map $\phi : \mathcal{X} \to \mathbb{R}^p, x \mapsto (1, x, x^2, \ldots, x^{p-1})^\top$. The resulting functions $f_{\boldsymbol{\theta}}(x) = \phi(x)^\top\boldsymbol{\theta}$ are then exactly the polynomials of degree $p-1$. It is therefore possible to learn non-linear functions in x with a model that is still linear in $\boldsymbol{\theta}$ and therefore easy to optimize. It is also worth emphasizing that the feature map ϕ is not surjective, and even if P_X can be described by a Lebesgue density, the distribution of $\phi(X), X \sim P_X$, cannot be described by a Lebesgue density. Hence, the use of feature maps can complicate distributional assumptions, and we will revisit this problem in Section 5.4. ◀

Unregularized linear regression with feature maps is a central ingredient of the moving least squares method (Lancaster and Salkauskas, 1981), which we use in Section 7.3 as an example of a regression method that can achieve optimal convergence rates for smooth functions f^* when the labels y_i are the non-noisy function values $f(\boldsymbol{x}_i)$.

Kernel trick By using the last formulation in Eq. (2.6), we see that the resulting learned function only depends on the *kernel* $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ given by $k(\boldsymbol{x}, \boldsymbol{x}') := \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$:

$$\begin{aligned} f_{\boldsymbol{\theta}^*_\lambda}(\boldsymbol{x}) &= \phi(\boldsymbol{x})^\top \boldsymbol{\theta}^*_\lambda \\ &= \phi(\boldsymbol{x})^\top \phi(\boldsymbol{X})^\top (\phi(\boldsymbol{X})\phi(\boldsymbol{X})^\top + \lambda \boldsymbol{I})^+ \boldsymbol{y} \\ &= \begin{pmatrix} \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}_1) \\ \vdots \\ \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}_n) \end{pmatrix} \left(\begin{pmatrix} \phi(\boldsymbol{x}_1)^\top \phi(\boldsymbol{x}_1) & \dots & \phi(\boldsymbol{x}_1)^\top \phi(\boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ \phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_1) & \dots & \phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_n) \end{pmatrix} + \lambda \boldsymbol{I} \right)^+ \boldsymbol{y} \\ &= \begin{pmatrix} k(\boldsymbol{x}, \boldsymbol{x}_1) \\ \vdots \\ k(\boldsymbol{x}, \boldsymbol{x}_n) \end{pmatrix} \left(\begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{pmatrix} + \lambda \boldsymbol{I} \right)^+ \boldsymbol{y} \\ &=: k(\boldsymbol{x}, \boldsymbol{X})(k(\boldsymbol{X}, \boldsymbol{X}) + \lambda \boldsymbol{I})^+ \boldsymbol{y} \ . \end{aligned} \tag{2.7}$$

The general idea of replacing the explicit use of a feature map ϕ in a learning method by the kernel k is known as the *kernel trick*. It can be computationally beneficial if

(1) the kernel $k(\boldsymbol{x}, \boldsymbol{x}')$ can be evaluated much more efficiently than through the direct formula $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$, or
(2) the feature space dimension p is larger than the number of samples n, such that the matrix inverse

$$(k(\boldsymbol{X}, \boldsymbol{X}) + \lambda \boldsymbol{I})^+ = (\phi(\boldsymbol{X})\phi(\boldsymbol{X})^\top + \lambda \boldsymbol{I})^+ \in \mathbb{R}^{n \times n}$$

can be computed more efficiently than the matrix inverse

$$(\phi(\boldsymbol{X})^\top \phi(\boldsymbol{X}) + \lambda \boldsymbol{I})^+ \in \mathbb{R}^{p \times p} \ .$$

We will encounter the computational tradeoff between these two versions multiple times in Chapter 6. The kernel version of (regularized) linear regression is known as *kernel ridge regression*. For more details on the kernel trick and kernel methods, we refer to standard textbooks such as the ones by Schölkopf and Smola (2002) and Shawe-Taylor and Cristianini (2004).

Theory While the theory of kernels and kernel methods is not the subject of this thesis, we will shortly discuss some of the major results. For a broader overview, we refer to Steinwart and Christmann (2008). Formally, a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called kernel if there exists a Hilbert space $\mathcal{H}$ and a feature map $\phi : \mathcal{X} \to \mathcal{H}$ such that $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_{\mathcal{H}}$ for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$. Equivalently, kernels can be defined through the condition that matrices of the form $k(\boldsymbol{X}, \boldsymbol{X})$ as above (for arbitrary n) must always be symmetric and positive semidefinite. Note that the Hilbert space $\mathcal{H}$ in the first definition need not be finite-dimensional. For example, the Gaussian kernel given by

$$k_{\mathrm{Gauss}}(\boldsymbol{x}, \boldsymbol{x}') := \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}{2\gamma^2} \right) \tag{2.8}$$

with parameter $\gamma^2 > 0$ can only be represented by feature maps with infinite-dimensional feature space $\mathcal{H}$. Nonetheless, the Gaussian kernel can be evaluated efficiently and is therefore an excellent example for case (1) in which the kernel formulation is beneficial.

There is a one-to-one correspondence between kernels k and reproducing kernel Hilbert spaces (RKHSs) $\mathcal{H}_k$, which are Hilbert spaces of functions $f : \mathcal{X} \to \mathbb{R}$ in which all point evaluation functionals are continuous. The kernel k can be represented by the canonical feature map $\phi : \mathcal{X} \to \mathcal{H}_k$ with $\phi(\boldsymbol{x}) = k(\boldsymbol{x}, \cdot)$, and it is called reproducing kernel for $\mathcal{H}_k$ due to the important identity

$$\langle k(\boldsymbol{x}, \cdot), f \rangle_{\mathcal{H}_k} = f(\boldsymbol{x})$$

for all functions $f \in \mathcal{H}_k$. The span of all functions $k(\boldsymbol{x}, \cdot)$ for $\boldsymbol{x} \in \mathcal{X}$ is dense in $\mathcal{H}_k$, and especially $\mathcal{H}_k$ also contains the learned function $f_{\boldsymbol{\theta}^*_\lambda}$. For $\lambda > 0$, we can also obtain the kernel regression solution through an infinite-dimensional optimization problem in the RKHS $\mathcal{H}_k$:

$$f_{\boldsymbol{\theta}^*_\lambda} = \operatorname*{arg\,min}_{f \in \mathcal{H}_k} \mathcal{R}_{L,D}(f) + \frac{\lambda}{n} \|f\|^2_{\mathcal{H}_k} .$$

Versions of the famous representer theorem (see e.g. Schölkopf et al., 2001) state that even for a larger class of loss functions and regularizations based on $\|f\|_{\mathcal{H}_k}$, the optimum is still of the form $f(\boldsymbol{x}) = \sum_{i=1}^n \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x})$ for some $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$. In other words, infinite-dimensional regularized ERM problems over $\mathcal{H}_k$ can typically be reduced to n-dimensional optimization problems.

For kernel methods, there is a rich theory regarding consistency and learning rates (Steinwart and Christmann, 2008). In particular, universal consistency can be achieved for sufficiently expressive kernels such as the Gaussian kernel above, provided suitable regularization. Moreover, kernel methods can achieve further good properties such as the learning of derivatives of the target function (Fischer and Steinwart, 2020) and fast learning rates for distributions on submanifolds (Hamm and Steinwart, 2021).

Further considerations It should be noted that while the kernel is uniquely determined by the feature map ϕ, the converse is not true. Generally, if $\phi : \mathcal{X} \to \mathcal{H}$ is a feature map for k, then for every orthogonal linear operator $U : \mathcal{H} \to \mathcal{H}$, $U \circ \phi$ is also a feature map for k. The calculation in Eq. (2.7) above shows that such orthogonal transformations of the feature map do not alter the learned function $f_{\boldsymbol{\theta}^*_\lambda}$. In this sense, the use of the kernel k circumvents a rotational ambiguity in the feature map ϕ. On the other hand, regularization criteria based on other norms such as $\|\boldsymbol{\theta}\|_1$ are usually not invariant under orthogonal transformations and can therefore not be represented by kernels. The kernel formulation also allows comparing and taking limits of feature maps with different feature space dimensions p. We will take advantage of this in Chapter 6, where we use sketching to approximate a kernel by another kernel with a smaller feature space dimension p, which facilitates more efficient computations.

2.2.3 Bayesian Linear Regression

While linear regression and kernel ridge regression can exhibit good generalization, they do not provide an estimate of the uncertainty of their predictions. Uncertainty estimates can be important in some settings, in particular, if decisions should be made based on whether the output of the regression model can be trusted and where it needs to be improved. A principled approach to provide such uncertainties is the Bayesian paradigm. In the Bayesian approach, we model the data-generating distribution P not as a fixed unknown distribution but instead consider P itself to be random. Then, the distribution of P expresses our uncertainty about the truth, and we can reduce this uncertainty by conditioning this distribution on observed data.

Probabilistic model In the following, we will derive a Bayesian version of linear regression known as *Bayesian linear regression*, which also comes in a kernelized version known as Gaussian Process regression (GPR). For more details, we refer to the textbooks by Bishop (2006), Rasmussen and Williams (2005), and Murphy (2022). We first assume that the target function to be learned is of the form $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\theta}$, where the unknown parameter vector $\boldsymbol{\theta}$ is drawn from a *prior distribution.* We omit the feature map here to draw a closer analogy to Section 2.2.1, but all formulas generalize analogously to the use of a feature map. For reasons of convenience that will become clear later, we assume the prior distribution to be a normal distribution with given variance $\sigma^2\lambda^{-1}$:

$$\boldsymbol{\theta} \sim \mathcal{N}(0, \sigma^2\lambda^{-1}\boldsymbol{I}) \ .$$

We then model the data labels as $y_i = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + \varepsilon_i$ where $\varepsilon_1, \ldots, \varepsilon_n \sim \mathcal{N}(0, \sigma^2)$ are assumed to be independent of each other and of the $\boldsymbol{x}_i$. In a more informal Bayesian notation, which we will adopt in this section for simplicity, we can write this model in terms of probability densities as

$$\begin{aligned}
p(\boldsymbol{\theta}) &= \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{0}, \sigma^2\lambda^{-1}\boldsymbol{I}) \\
p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) &\coloneqq p(y_1, \ldots, y_n \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n; \boldsymbol{\theta}) = \prod_{i=1}^n p(y_i \mid \boldsymbol{x}_i; \boldsymbol{\theta}) \\
&= \prod_{i=1}^n \mathcal{N}(y_i \mid f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \sigma^2) \ .
\end{aligned}$$

The latter term $p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})$ is also commonly referred to as the *likelihood.* If we assume that the inputs $\boldsymbol{X}$ are independent of $\boldsymbol{\theta}$, we can apply Bayes' theorem to the probability densities without having to model the input distribution $p(\boldsymbol{X}) = p(\boldsymbol{X} \mid \boldsymbol{\theta})$:

$$\begin{aligned}
p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y}) &= \frac{p(\boldsymbol{X}, \boldsymbol{y} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{X}, \boldsymbol{y})} = \frac{p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})p(\boldsymbol{X} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\boldsymbol{y} \mid \boldsymbol{X}, \tilde{\boldsymbol{\theta}})p(\boldsymbol{X} \mid \tilde{\boldsymbol{\theta}})p(\tilde{\boldsymbol{\theta}}) \,\mathrm{d}\tilde{\boldsymbol{\theta}}} \\
&= \frac{p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\boldsymbol{y} \mid \boldsymbol{X}, \tilde{\boldsymbol{\theta}})p(\tilde{\boldsymbol{\theta}}) \,\mathrm{d}\tilde{\boldsymbol{\theta}}} \ .
\end{aligned}$$

The resulting distribution $p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y})$ is known as the *posterior distribution.* The normalization constant in the denominator is often referred to as the *evidence* and can be neglected in some calculations. Using C_1, C_2 to denote suitable constants independent of $\boldsymbol{\theta}$, we obtain in our Gaussian model above:

$$\begin{aligned}
-\log p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y}) \quad &= \quad C_1 - \log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) - \log p(\boldsymbol{\theta}) \\
&= \quad C_2 + \frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))^2 + \frac{\lambda}{2\sigma^2} \|\boldsymbol{\theta}\|_2^2 \\
&\overset{\text{Eq. (2.2)}}{=} \quad C_2 + \frac{n}{2\sigma^2} \mathcal{L}_\lambda(\boldsymbol{\theta}) \ .
\end{aligned}$$

Relation to ERM It turns out that the neg-log-posterior is a shifted and rescaled version of the classical regularized linear regression objective from Eq. (2.2). Hence, the mode of the posterior distribution, known as the *maximum a-posteriori* (MAP) estimate, is exactly the minimizer $\boldsymbol{\theta}^*_\lambda = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}^\top \boldsymbol{y}$ for ridge regression that we have seen in Section 2.2.1. The use of the quadratic loss function $L(y, t) = (y - t)^2$ arises in the Bayesian setting from the assumption that the label noise ε_i follows a normal distribution. However, as we have discussed in Section 2.1, the use of the quadratic loss function is well-motivated even with non-Gaussian label noise, as a consistent learning method will converge to the conditional expectation $f^*_{L,P}(\boldsymbol{x}) = \mathbb{E}_P[Y \mid X = \boldsymbol{x}]$. On the other hand, the assumption of Gaussian noise is essential for the posterior distribution, which will determine the uncertainty estimates about the parameters $\boldsymbol{\theta}$ and later also about the predictions y.

Analytic solution To derive a closed-form representation of the posterior distribution, we observe that the neg-log-posterior above is a quadratic function in $\boldsymbol{\theta}$ with a unique minimum at $\boldsymbol{\theta}^*_\lambda$. Therefore, the posterior distribution must be of the form $p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*_\lambda, \boldsymbol{\Sigma})$, where we need to determine $\boldsymbol{\Sigma}$. On the one hand, we can use the Hessian of $\mathcal{L}_\lambda$ computed in Eq. (2.3) on page 12 to obtain

$$H_{\boldsymbol{\theta}}(-\log p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y})) = H_{\boldsymbol{\theta}} \frac{n}{2\sigma^2}\mathcal{L}_\lambda(\boldsymbol{\theta}) = \frac{1}{\sigma^2}(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I}) \ .$$

On the other hand, we can use the formula for the density of a normal distribution to obtain

$$H_{\boldsymbol{\theta}}(-\log \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*_\lambda, \boldsymbol{\Sigma})) = \boldsymbol{\Sigma}^{-1} \ .$$

Therefore, we arrive at the posterior

$$p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*_\lambda, \boldsymbol{\Sigma}), \qquad \boldsymbol{\Sigma} = \sigma^2(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \ .$$

Using the posterior distribution, we can also compute the *posterior predictive distribution* for the label $\tilde{y} = f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}) + \tilde{\varepsilon}$ at a new test point $\tilde{\boldsymbol{x}}$. Indeed, from a standard identity on the behavior of normal distributions under linear transformations, we obtain that under the posterior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y})$, $f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}) = \tilde{\boldsymbol{x}}^\top \boldsymbol{\theta}$ is distributed as

$$f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}) \mid \boldsymbol{X}, \boldsymbol{y} \sim \mathcal{N}\left(\tilde{\boldsymbol{x}}^\top \boldsymbol{\theta}^*_\lambda, \sigma^2 \tilde{\boldsymbol{x}}^\top (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \tilde{\boldsymbol{x}}\right) \ . \tag{2.9}$$

Since $\tilde{\varepsilon} \sim \mathcal{N}(0, \sigma^2)$ is independent of $\tilde{\boldsymbol{x}}$, $\boldsymbol{X}$ and $\boldsymbol{y}$, it follows that

$$p(\tilde{y} \mid \tilde{\boldsymbol{x}}, \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}\left(\tilde{y} \mid \tilde{\boldsymbol{x}}^\top \boldsymbol{\theta}^*_\lambda, \sigma^2 + \sigma^2 \tilde{\boldsymbol{x}}^\top (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \tilde{\boldsymbol{x}}\right) \ .$$

In this model, the posterior predictive variance $\sigma^2 + \sigma^2 \tilde{\boldsymbol{x}}^\top (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \tilde{\boldsymbol{x}}$ hence decomposes into the *aleatoric uncertainty* σ^2, which cannot be reduced by collecting more data, and the *epistemic uncertainty* $\sigma^2 \tilde{\boldsymbol{x}}^\top (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \tilde{\boldsymbol{x}}$, which stems from the uncertainty about the true parameter vector $\boldsymbol{\theta}$.

In addition to single-input predictive distributions and their variance, we can also compute multi-input predictive distributions, which are always Gaussian. In particular, we are interested in the covariance of the predictions at different inputs, which can again be computed by applying a linear transformation to the posterior distribution in Eq. (2.9):

$$\mathrm{Cov}(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}), f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}') \mid \boldsymbol{X}, \boldsymbol{y}) = \sigma^2 \tilde{\boldsymbol{x}}^\top (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \tilde{\boldsymbol{x}}' \ .$$

Kernel version Again, all of the derivations above apply analogously to a prior distribution over functions of the form $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \phi(\boldsymbol{x})^\top \boldsymbol{\theta}$, where $\phi : \mathbb{R}^d \to \mathbb{R}^p$ is a given feature map. To obtain a kernel version corresponding to the feature map version, we note that we can write the posterior predictive mean as $\phi(\tilde{\boldsymbol{x}})^\top \boldsymbol{\theta}^*_\lambda = k(\tilde{\boldsymbol{x}}, \boldsymbol{X})(k(\boldsymbol{X}, \boldsymbol{X}) + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$, which we already derived in Section 2.2.2. Moreover, for the posterior predictive covariance, we can use the Woodbury matrix identity as in Eq. (2.4) on page 12 and obtain

$$\begin{aligned}&\operatorname{Cov}(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}), f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}') \mid \boldsymbol{X}, \boldsymbol{y}) \\ &= \sigma^2\lambda^{-1}k(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}') - \sigma^2\lambda^{-1}k(\tilde{\boldsymbol{x}}, \boldsymbol{X})(k(\boldsymbol{X}, \boldsymbol{X}) + \lambda\boldsymbol{I})^{-1}k(\boldsymbol{X}, \tilde{\boldsymbol{x}}') \ .\end{aligned}$$

To eliminate the occurrence of the factor $\sigma^2\lambda^{-1}$, one can instead consider the (prior) covariance kernel given by

$$\begin{aligned}k_{\text{cov}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}') &\coloneqq \operatorname{Cov}(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}), f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}')) = \mathbb{E}_{\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\lambda^{-1}\boldsymbol{I})} \phi(\tilde{\boldsymbol{x}})^\top \boldsymbol{\theta}\boldsymbol{\theta}^\top \phi(\tilde{\boldsymbol{x}}') \\ &= \sigma^2\lambda^{-1}\phi(\tilde{\boldsymbol{x}})^\top\phi(\tilde{\boldsymbol{x}}') = \sigma^2\lambda^{-1}k(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}') \ .\end{aligned}$$

The covariance kernel is a rescaled version of k with corresponding feature map $\phi_{\text{cov}}(\boldsymbol{x}) = \sqrt{\sigma^2\lambda^{-1}}\phi(\boldsymbol{x})$ and weight prior $\boldsymbol{\theta} \sim \mathcal{N}(0, \boldsymbol{I})$. Now, the posterior predictive covariance simplifies to

$$\begin{aligned}&\operatorname{Cov}(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}), f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}') \mid \boldsymbol{X}, \boldsymbol{y}) \\ &= \sigma^2 + k_{\text{cov}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}') - k_{\text{cov}}(\tilde{\boldsymbol{x}}, \boldsymbol{X})(k_{\text{cov}}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2\boldsymbol{I})^{-1}k_{\text{cov}}(\boldsymbol{X}, \tilde{\boldsymbol{x}}') \ , \end{aligned} \tag{2.10}$$

and the posterior predictive distribution simplifies to

$$\begin{aligned}p(\tilde{y} \mid \tilde{\boldsymbol{x}}, \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}\Big(&\tilde{y} \mid k_{\text{cov}}(\tilde{\boldsymbol{x}}, \boldsymbol{X})(k_{\text{cov}}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}, \\ &\sigma^2 + k_{\text{cov}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}) - k_{\text{cov}}(\tilde{\boldsymbol{x}}, \boldsymbol{X})(k_{\text{cov}}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2\boldsymbol{I})^{-1}k_{\text{cov}}(\boldsymbol{X}, \tilde{\boldsymbol{x}})\Big) \ .\end{aligned}$$

In Section 6.4.2, we use the fact that the posterior predictive covariance also defines a covariance kernel

$$k_{\text{cov}\to\text{post}(\boldsymbol{X})}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}') = \operatorname{Cov}(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}), f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}') \mid \boldsymbol{X}, \boldsymbol{y}) \ ,$$

and we will refer to the mapping $k_{\text{cov}} \mapsto k_{\text{cov}\to\text{post}(\boldsymbol{X})}$ as a kernel transformation.

2.2.4 Neural Networks

As we have seen above, linear regression models can be solved easily due to their linearity in $\boldsymbol{\theta}$, while they still allow learning nonlinear functions in $\boldsymbol{x}$ through the use of a nonlinear feature map ϕ. However, the specification of a good feature map ϕ can be very difficult. (Artificial) neural networks (NNs) allow learning the feature map ϕ from data, at the cost of making the ERM problem non-convex and therefore potentially much more challenging. To construct an NN, we have to choose a parametric feature map $\phi_{\tilde{\boldsymbol{\theta}}}$ with a parameter vector $\tilde{\boldsymbol{\theta}}$ that can then be optimized jointly with $\boldsymbol{\theta}$. Here, we will consider a particularly simple form of NNs called fully-connected neural network (FCNN) or multilayer perceptron (MLP). For a much broader introduction to neural networks, we refer to books such as the one by Goodfellow et al. (2016).

Network architecture To specify an FCNN model, we have to make a few choices:

- Choose a number $L \geq 2$ of layers, which will also be called the *depth* of the NN. More layers will correspond to more sequential processing steps, and the term *deep learning* refers to the use of (somewhat) deep neural networks.
- Choose an input dimension d_0 and an output dimension d_L. In the linear regression setting we considered before, this corresponds to $d_0 = d$ and $d_L = 1$.
- Choose *hidden layer widths* $d_1, \dots, d_{L-1} \in \mathbb{N}_{\geq 1}$. For example, one could choose $d_1 = \ldots = d_{L-1} = 256$.
- Choose a (non-linear) *activation function* $\varphi : \mathbb{R} \to \mathbb{R}$. A common activation function is the rectified linear unit (ReLU) activation function given by $\varphi(x) = \max\{0, x\}$.

Given an input $\boldsymbol{x} = \boldsymbol{x}^{(0)} \in \mathbb{R}^d$, an FCNN then computes an output $\boldsymbol{z}^{(L)}$ iteratively via

$$\begin{aligned} \boldsymbol{z}^{(l)} &:= \boldsymbol{W}^{(l)}\boldsymbol{x}^{(l-1)} + \boldsymbol{b}^{(l)} \in \mathbb{R}^{d_l} \;, \\ \boldsymbol{x}^{(l)} &:= \varphi(\boldsymbol{z}^{(l)}) \in \mathbb{R}^{d_l} \;, \end{aligned}$$

where the activation function φ is applied element-wise to the vector $\boldsymbol{z}^{(l)}$. The *weight matrices* $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ and the *bias vectors* $\boldsymbol{b}^{(l)} \in \mathbb{R}^{d_l}$ are learnable parameters of the NN. We will write the NN again as a parametric function $f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(0)}) = \boldsymbol{z}^{(L)}$, where the parameter vector $\boldsymbol{\theta}$ comprises all weight matrices and bias vectors:

$$\boldsymbol{\theta} = (\boldsymbol{W}^{(1)}, \boldsymbol{b}^{(1)}, \dots, \boldsymbol{W}^{(L)}, \boldsymbol{b}^{(L)}) \;.$$

For $L = 2$, we can write the NN explicitly as

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{b}^{(2)} + \boldsymbol{W}^{(2)}\varphi(\boldsymbol{b}^{(1)} + \boldsymbol{W}^{(1)}\boldsymbol{x}) \;.$$

The use of the activation function φ is crucial: Without it, the NN could only represent linear functions in $\boldsymbol{x}$. On the other hand, the activation function is deliberately not applied after the last layer of the neural network, since it could restrict the range of the learned function. Besides the ReLU activation, there are a few other popular choices such as the SiLU activation $\varphi(x) = \frac{x}{1+e^{-x}}$ (Elfwing et al., 2018). More activation functions are discussed in Section 5.5.

Training To train an NN, that is, optimize its parameters, simple first-order optimization methods are frequently employed. For example, NNs could be trained using *gradient descent* (GD) starting from some initial parameter vector $\boldsymbol{\theta}^{(0)}$ by iterating

$$\boldsymbol{\theta}^{(t+1)} := \boldsymbol{\theta}^{(t)} - \eta \nabla \mathcal{L}(\boldsymbol{\theta}^{(t)}) \;,$$

where $\eta > 0$ is a step-size parameter and $\mathcal{L}$ is a least-squares ERM objective similar to Section 2.2.1:

$$\mathcal{L}(\boldsymbol{\theta}) := \frac{1}{n}\sum_{i=1}^{n}(y_i - f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))^2 \;.$$

When the data set size n is large, computing the sum in the definition of $\mathcal{L}(\boldsymbol{\theta})$ at each gradient descent step can be prohibitively expensive. To circumvent this, we can rewrite the gradient of $\mathcal{L}$ as an expectation over sample gradients via

$$\nabla \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} (y_i - f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))^2 = \mathbb{E}_i \nabla_{\boldsymbol{\theta}} (y_i - f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))^2 ,$$

which can then be approximated using Monte Carlo sampling. Variants of this procedure are known as (mini-batch) *stochastic gradient descent* (SGD). The very popular Adam optimizer (Kingma and Ba, 2015), which we employ in Chapter 6, extends SGD with a momentum term and a normalization using moving averages of squared gradients. While none of these optimizers are guaranteed to find a global optimum of the objective $\mathcal{L}$, they are successfully applied in practice. We do not derive an analytical formula for $\nabla \mathcal{L}(\boldsymbol{\theta})$ here since this gradient can be automatically computed in modern deep learning frameworks by the use of automatic differentiation.

While we used an unregularized objective in our formulas above, neural networks can also be regularized using a norm penalty on the parameters. Moreover, popular regularization strategies include dropout (Srivastava et al., 2014), where some entries of intermediate vectors are randomly set to zero during training, and early stopping, where the optimization is stopped before convergence.

Initialization The first-order optimization methods discussed previously all require an initial parameter vector $\boldsymbol{\theta}^{(0)}$ to start the optimization. However, a trivial initialization $\boldsymbol{\theta}^{(0)} = \mathbf{0}$ is problematic because the gradient $\nabla \mathcal{L}(\mathbf{0})$ is zero, i.e., the zero vector is a saddle point of the optimization problem. Even an initialization with identical non-zero components will have permutational symmetries that cannot be broken by typical gradient-based optimizers. To break these symmetries, $\boldsymbol{\theta}^{(0)}$ is often initialized randomly. For example, a popular initialization method by He et al. (2015) initializes all components of the weight matrices independently as

$$W_{ij}^{(l)} \sim \mathcal{N}(0, 2/d_{l-1})$$

and initializes the biases with zero. Here, the factor $1/d_{l-1}$ in the variance ensures that the components of the pre-activation vector $\boldsymbol{z}^{(l)}$ have roughly the same magnitude as the components of $\boldsymbol{x}^{(l-1)}$, while the factor 2 compensates for the ReLU activation setting half of the values to zero.

Theory There are many approaches toward a theoretical analysis of NNs. A setting that has gained particular interest is the analysis of over-parameterized NNs, which are frequently employed in practice. Different theoretical results suggest that strongly over-parameterized NNs can allow finding global optima by gradient-based optimization methods (Chizat and Bach, 2018; Du et al., 2019; Allen-Zhu et al., 2019). An approach that is particularly amenable to classical tools is to consider linearizations of NNs w.r.t. their parameters. As a simple case, one can consider optimizing only over the last-layer parameters of an NN, in which the NN is already linear. In this case, one obtains linear regression with the rest of the NN acting as a feature map. As shown in Theorem 2.2.2, when initializing the last layer to zero, gradient descent training converges to the minimum-norm linear regression solution. We study this simplified setting in Chapter 5 and use Bayesian linear regression in the last layer in Chapter 6.

A more realistic way to analyze NNs is to linearize them in all parameters, motivated by the Taylor expansion

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = f_{\boldsymbol{\theta}_0}(\boldsymbol{x}) + (\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\boldsymbol{x}))^\top (\boldsymbol{\theta} - \boldsymbol{\theta}_0) + O(\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|^2) ~,$$

which holds if the activation function and therefore the NN is twice continuously differentiable. One can then define the corresponding feature map $\phi(\boldsymbol{x}) = \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\boldsymbol{x})$ and the corresponding kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$, which is also known as a finite-width *neural tangent kernel* (NTK) (Jacot et al., 2018). We use the finite-width NTK and an approximation thereof in Chapter 6 for Bayesian linear regression as an alternative to the last-layer-based approximation. Of course, the finite-width NTK depends on the linearization point $\boldsymbol{\theta}_0$, which could be a random initialization or the parameters at any point during training. To study over-parameterized NNs, it is tempting to take an infinite-width limit $d_1, \ldots, d_{L-1} \to \infty$ of this kernel. However, with our NN definition above, this limit would be degenerate since the gradients are not appropriately normalized. This can be fixed by using the so-called neural tangent parameterization (NTP) (Jacot et al., 2018; Lee et al., 2019):

$$\begin{aligned} \boldsymbol{z}^{(l)} &:= \sigma_w \sqrt{\frac{1}{d_{l-1}}} \boldsymbol{W}^{(l)} \boldsymbol{x}^{(l-1)} + \sigma_b \boldsymbol{b}^{(l)} \in \mathbb{R}^{d_l} \\ \boldsymbol{x}^{(l)} &:= \varphi(\boldsymbol{z}^{(l)}) \in \mathbb{R}^{d_l} ~. \end{aligned}$$

with suitable scaling factors $\sigma_w, \sigma_b \in \mathbb{R}$. In this case, it is possible to show that the finite-width NTKs, when linearized at parameters during training that depend on the training step and the random initialization, converge to an infinite-width NTK that does not depend on the training step and the random initialization (Jacot et al., 2018; Lee et al., 2019; Arora et al., 2019). Especially, the linearization is exact in the infinite-width limit and the NN training essentially corresponds to the training of a corresponding kernel method.

The neural tangent kernel correspondence has been used to prove universal consistency for certain heavily over-parameterized NNs with early stopping (Ji et al., 2021), and similar ideas have been used to prove the inconsistency of NNs in other cases (Holzmüller and Steinwart, 2022). However, it should be noted that the limiting behavior to infinite-width NTKs depends on the parameterization and does not capture certain practical observations (Yang and Hu, 2021; Bietti and Bach, 2021).

Bayesian NNs Besides providing uncertainties, a Bayesian treatment of NNs also promises better predictions, but approximating the posterior predictive distribution of NNs is computationally very difficult (Izmailov et al., 2021). In particular, due to the non-linear dependence of an NN on its parameters, the log-posterior $\log p(\boldsymbol{\theta} \mid D)$ is often multimodal and non-concave, which renders efficient sampling difficult. We investigate the non-log-concave sampling problem from a general standpoint in Chapter 7.

When cheaper approaches are needed, it is possible to resort to linearizations as discussed above, for example in the last layer or in all parameters, which allow turning an

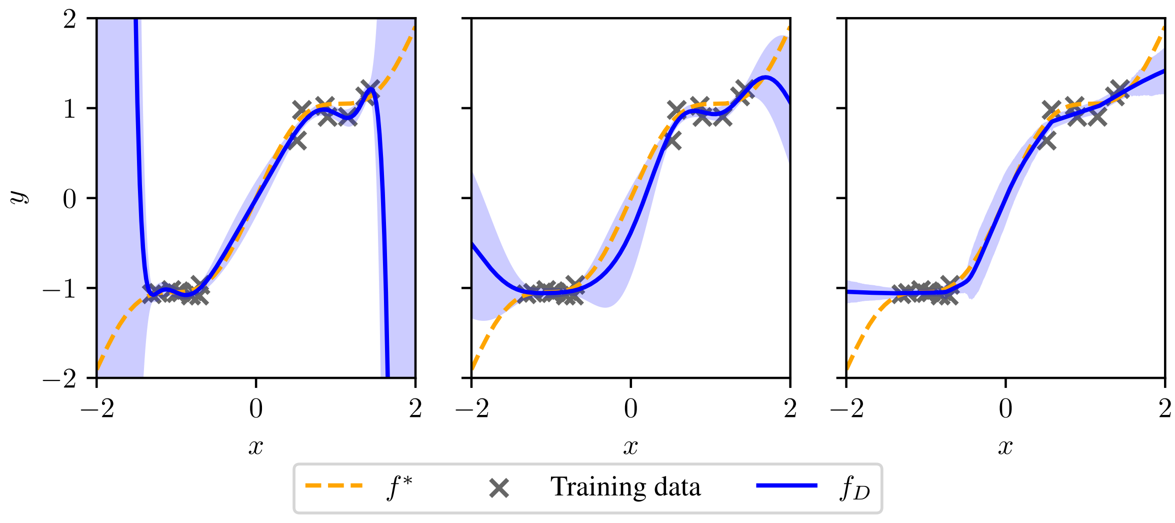


Figure 2.1: Posterior mean (f_D) and one posterior standard deviation around it (shaded area), excluding the aleatoric uncertainty term σ^2, for various Bayesian regression methods. Training data generation and all methods use a noise variance of $\sigma = 0.1$. Left: Bayesian linear regression with polynomial feature map as in Example 2.2.3 up to degree 14. Middle: Gaussian process regression with Gaussian kernel using $\gamma = 0.5$, cf. Eq. (2.8). Right: NN with ReLU activation function as in Chapter 6 with Bayesian uncertainty estimates given by linearization in all of the parameters, i.e., using $k_{\text{grad}\to\mathcal{X}_{\text{train}}}$ as described in Chapter 6.

NN into a Bayesian linear regression model. Crucially, the linearization is usually done around the trained parameters of the NN, such that the NN can still learn the feature map from the training data. For an overview of such approaches, we refer to Daxberger et al. (2021). We use linearization-based approaches in Chapter 6 for active learning with NNs. Moreover, it is also possible to study Bayesian NNs in an infinite-width limit. In this case, under suitable assumptions, the distribution of randomly initialized NNs converges to a Gaussian process whose covariance kernel is the so-called neural network Gaussian process (NNGP) kernel (Neal, 1994; Lee et al., 2018; Matthews et al., 2018). We also use this kernel as a comparison in Chapter 6. Goan and Fookes (2020) provide a broader overview of Bayesian NNs. Figure 2.1 shows a linearization-based approximate Bayesian NN in comparison with Bayesian linear regression and GP regression on a toy data set.

2.3 Gibbs Distributions and Statistical Distances

As we have discussed previously, unlike learning methods based on empirical risk minimization (ERM), Bayesian models naturally provide uncertainty estimates and are therefore also well-suited for active learning. In our work on active learning in Chapter 6, we employ Bayesian linear regression models based on linearizations of NNs, which allow computing the posterior distribution analytically. However, it can be desirable to get a more accurate posterior distribution for NNs and other non-linear models. A common approach to approximately represent such a posterior distribution is through sampling, i.e., drawing i.i.d. random samples from the posterior. This is in contrast to ERM, where optimization instead of sampling is used. However, the posterior may often be a multi-modal distribution, and sampling from multi-modal distributions can be challenging. Sampling from multi-modal distributions is also highly relevant to statistical mechanics, where the thermodynamic equilibrium distribution of atomic configurations is considered.

Gibbs distributions In both Bayesian ML and statistical mechanics, a convenient form to express the corresponding distribution is as a *Gibbs distribution* (or Boltzmann distribution) P_f given by the density

$$p_f(\boldsymbol{x}) \coloneqq \frac{\exp(f(\boldsymbol{x}))}{Z_f}, \qquad Z_f \coloneqq \int_{\mathcal{X}} \exp(f(\boldsymbol{x}))\,\mathrm{d}\boldsymbol{x}\ ,$$

for a given domain $\mathcal{X}$ and function $f : \mathcal{X} \to \mathbb{R}$. In statistical mechanics, the Gibbs distribution is used with $f(\boldsymbol{x}) = -V(\boldsymbol{x})/(k_B T)$, where $V(\boldsymbol{x})$ is the potential energy of state $\boldsymbol{x}$, k_B is Boltzmann's constant, and T is the temperature of the system (see e.g. Swendsen, 2020). Another relevant quantity is the log-normalization constant

$$L_f \coloneqq \log Z_f\ .$$

We follow the convention of statistical mechanics, where L_f is considered to be a function of additional parameters of f such as the temperature T, and call L_f the *log-partition function* corresponding to f. In statistical mechanics, the log-partition function is of great interest as it can describe relevant properties of a system (see e.g. Faulkner and Livingstone, 2022).

In Bayesian ML, the suitability of a formulation in terms of Gibbs distributions is not immediately obvious. Suppose that we have a Bayesian model specifying a prior $p(\boldsymbol{\theta})$ and a likelihood $p(D|\boldsymbol{\theta})$. Given a data set D, we can set $f(\boldsymbol{\theta}) \coloneqq \log(p(D|\boldsymbol{\theta})p(\boldsymbol{\theta}))$, such that the resulting Gibbs distribution is exactly the posterior distribution:

$$p_f(\boldsymbol{\theta}) = \frac{\exp(f(\boldsymbol{\theta}))}{Z_f} = \frac{p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(D|\tilde{\boldsymbol{\theta}})p(\tilde{\boldsymbol{\theta}})\,\mathrm{d}\tilde{\boldsymbol{\theta}}} = p(\boldsymbol{\theta}|D)\ .$$

Moreover, the log-partition function is equal to the log-evidence or log-marginal likelihood:

$$L_f = \log\left(\int p(D|\tilde{\boldsymbol{\theta}})p(\tilde{\boldsymbol{\theta}})\,\mathrm{d}\tilde{\boldsymbol{\theta}}\right) = \log p(D)\ .$$

It can be used to compare Bayesian models, as models with higher marginal likelihood $p(D)$ are often preferable (Robert, 2007). The considerations above show that Gibbs distributions can be used to express central quantities in Bayesian ML, but are they also numerically favorable? Typically, numerical algorithms converge faster for smooth functions with small (higher-order) derivatives. We have already seen in the case of Bayesian linear regression in Section 2.2.3 that the log-likelihood $\log p(D|\boldsymbol{\theta})$ and the log-prior $\log p(\boldsymbol{\theta})$ are quadratic functions, which have rather small (higher-order) derivatives. More generally, in the case of i.i.d. samples $D = (D_1, \ldots, D_N)$, the likelihood $p(D|\boldsymbol{\theta})$ is a product of individual likelihoods, which means that the log-likelihood is a sum of individual log-likelihoods. Hence, the Gibbs distribution formulation is suitable for Bayesian ML in the sense that the derivatives of f should approximately grow linearly with N, while the derivatives of p_f might grow much faster.

Distances for probability distributions Algorithms for sampling from Gibbs distributions P_f will usually produce samples from an approximate distribution $\tilde{P}_f$. Similarly, algorithms for computing the log-partition function L_f will usually only compute an approximation $\tilde{L}_f$. While we can easily compare L_f and $\tilde{L}_f$ through the absolute difference $|L_f - \tilde{L}_f|$, there are many different options for comparing P_f and $\tilde{P}_f$. We will introduce some of these options in the following. For a broader overview of such statistical distances and their properties and relations, we refer to Gibbs and Su (2002) and Section 2.4 in Tsybakov (2009).

For probability measures P, Q on $\mathcal{X} \subseteq \mathbb{R}^d$ that have densities p, q with respect to another measure μ, the *Kullback-Leibler divergence* (or relative entropy) is defined as

$$D_{\mathrm{KL}}(P \,\|\, Q) \coloneqq \int_{\mathcal{X}} p(\boldsymbol{x}) \log\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right) \mathrm{d}\mu(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{x}\sim P}\left[-\log\left(\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\right)\right] .$$

The definition of the KL divergence is independent of the choice of μ. While the KL divergence is not symmetric, it is non-negative. This follows from the inequality $\log(x) \leq x - 1$ for $x \in (0, \infty)$:

$$\begin{aligned} D_{\mathrm{KL}}(P \,\|\, Q) &= \mathbb{E}_{\boldsymbol{x}\sim P}\left[-\log\left(\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\right)\right] \\ &\geq \mathbb{E}_{\boldsymbol{x}\sim P}\left[1 - \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\right] = \int (p(\boldsymbol{x}) - q(\boldsymbol{x}))\,\mathrm{d}\mu(\boldsymbol{x}) = 1 - 1 = 0 . \end{aligned}$$

Moreover, since $\log(x) = x - 1$ only for $x = 1$, we have $D_{\mathrm{KL}}(P \,\|\, Q) = 0$ if and only if $p(\boldsymbol{x})/q(\boldsymbol{x}) = 1$ for P-almost all $\boldsymbol{x}$, which is equivalent to $P = Q$. The non-negativity of the KL divergence can also be proven by applying Jensen's inequality (see Theorem 2.1.5) to the convex function $\varphi = -\log$.

Another statistical distance is the *total variation* (TV) distance given by (see e.g. Gibbs and Su, 2002; Tsybakov, 2009)

$$\begin{aligned} D_{\mathrm{TV}}(P, Q) &\coloneqq \sup_{A \subseteq \mathcal{X} \text{ measurable}} |P(A) - Q(A)| = \frac{1}{2} \sup_{\|h\|_\infty \leq 1} \left| \int h \,\mathrm{d}P - \int h \,\mathrm{d}Q \right| \\ &= \frac{1}{2} \int |p(\boldsymbol{x}) - q(\boldsymbol{x})| \,\mathrm{d}\mu(\boldsymbol{x}) . \end{aligned}$$

Here, the first two formulas are also defined when P or Q do not have a density. Moreover, by defining $\Gamma(P, Q)$ as the set of measures on $\mathcal{X} \times \mathcal{X}$ with marginal distributions P and Q in the first and second component, we can define the 1-Wasserstein distance (or Kantorovich-Rubinstein distance) as

$$W_1(P, Q) \coloneqq \inf_{\mu \in \Gamma(P,Q)} \int \|\boldsymbol{x} - \boldsymbol{x}'\|_2 \,\mathrm{d}\mu(\boldsymbol{x}, \boldsymbol{x}') .$$

By the Kantorovich-Rubinstein theorem, we have the alternative characterization

$$W_1(P, Q) = \sup_{\text{1-Lipschitz functions } h} \left| \int h \,\mathrm{d}P - \int h \,\mathrm{d}Q \right| .$$

The TV distance and the 1-Wasserstein distance are metrics. For the TV distance, we have the trivial upper bound $D_{\mathrm{TV}}(P, Q) \leq 1$ for all probability distributions P, Q. Moreover, Pinsker's inequality states that

$$D_{\mathrm{TV}}(P, Q) \leq \sqrt{\frac{1}{2} D_{\mathrm{KL}}(P \parallel Q)} \ ,$$

and if $\mathcal{X}$ is bounded, we also have the inequality

$$W_1(P, Q) \leq \mathrm{diam}(\mathcal{X}) D_{\mathrm{TV}}(P, Q) \ ,$$

where $\mathrm{diam}(\mathcal{X}) \coloneqq \sup_{\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}'\|_2$. In Chapter 7, we also introduce another metric

$$D_{\text{sup-log}}(P, Q) \coloneqq \|\log(p/q)\|_\infty \ ,$$

which we call the sup-log distance. Here, we assume μ to be the Lebesgue measure on $\mathcal{X}$ for simplicity.

In Chapter 7, we will mainly compare probability distributions using the TV, 1-Wasserstein, and sup-log distances. From these distances, bounds on the KL divergence could be derived using Pinsker's inequality and the trivial fact that the sup-log distance is an upper bound for the KL divergence. The KL divergence will also play a different role in an optimization objective in Section 7.4.4.

Chapter 3

Main Results and Outlook

3.1 Main Results

In this section, we describe the main results of this thesis, which are contained in the following papers:

(1) David Holzmüller, *On the universality of the double descent peak in ridgeless regression*, published at the International Conference on Learning Representations, 2021.
Reference: Holzmüller (2021)
Link: `https://openreview.net/forum?id=0IO5VdnSAaH`
Link to arXiv version: `https://arxiv.org/abs/2010.01851`
This article (specifically version 8 on arXiv) is reproduced in Chapter 5.

(2) David Holzmüller, Viktor Zaverkin, Johannes Kästner, and Ingo Steinwart, *A framework and benchmark for deep batch active learning for regression*, published at the Journal of Machine Learning Research, 2023.
Reference: Holzmüller et al. (2023)
Link: `https://jmlr.org/papers/v24/22-0937.html`
Link to arXiv version: `https://arxiv.org/abs/2203.09410`
This article (specifically version 4 on arXiv) is reproduced in Chapter 6.

(3) David Holzmüller and Francis Bach, *Convergence rates for non-log-concave sampling and log-partition estimation*, preprint available on arXiv, 2023.
Reference: Holzmüller and Bach (2023)
Link: `https://arxiv.org/abs/2303.03237`
This article (specifically version 3 on arXiv) is reproduced in Chapter 7.

Chapter 4 contains a statement on the contributions of the author of this thesis to the articles above. The following three subsections will present the main results of the respective articles.

3.1.1 Double Descent

For many learning methods such as kernel methods, consistency guarantees have been established in regimes where these learning methods are sufficiently regularized such that the empirical risk is close to the population risk with high probability (Devroye et al., 1996;

Györfi et al., 2002; Steinwart and Christmann, 2008). However, NNs are often employed in settings where they are over-parameterized, i.e., having more parameters than the number n of training samples. Zhang et al. (2017) observed that such NNs can achieve low test errors on practical data sets despite being able to fit random noise. Belkin et al. (2018) found that kernel methods exhibit a similar phenomenon and showed that this could not be explained by existing generalization bounds at the time. It has been observed multiple times that learning methods can perform especially badly in a critical regime where they are just able to interpolate the training data, and their performance improves when they become more over-parameterized (Bös and Opper, 1997; Advani et al., 2020; Neal et al., 2019; Spigler et al., 2019; Belkin et al., 2019). This phenomenon is now known as "double descent" thanks to Belkin et al. (2019), and has been studied empirically for NNs in detail by Nakkiran et al. (2021a).

The theoretical understanding of double descent is mostly limited to unregularized linear regression models. Most of these investigations are limited to rather specific data distributions P_X or feature maps ϕ, and they consider a limit of $n, p \to \infty$, where p is the feature space dimension and n is the number of samples (Advani et al., 2020; Belkin et al., 2020; Hastie et al., 2022; Mei and Montanari, 2022; d'Ascoli et al., 2020). We prove a lower bound on the expected excess risk that does not require a limit and holds for a broad class of "non-degenerate" feature maps and data distributions P_X if the labels y_i are noisy. Note that a related probabilistic lower bound on the excess risk has been proven by Muthukumar et al. (2020), but requires stronger assumptions as discussed in Appendix 5.L. We first assume that the labels are corrupted by noise:

Assumption 3.1.1 (Label noise)**.** Let $\mathcal{X} = \mathbb{R}^d, \mathcal{Y} = \mathbb{R}$, let $L(y,t) = (y-t)^2$ be the square loss, and let P be a probability distribution on $\mathcal{X} \times \mathcal{Y}$. For $(X, Y) \sim P$, we assume

(INT) $\mathbb{E}[Y^2] < \infty$, which implies $\mathcal{R}_{L,P}(0) < \infty$ and hence $\mathcal{R}_{L,P}(f^*_{L,P}) < \infty$,
(NOI) $\operatorname{Var}(Y|X = \boldsymbol{x}) \geq \sigma^2$ for P_X-almost all $\boldsymbol{x} \in \mathcal{X}$, ◀

Our notion of non-degeneracy can be defined as follows:

Definition 3.1.2 (Non-degenerate feature map)**.** Let $\mathcal{X}, \mathcal{Y}, P$ be as in Assumption 3.1.1. We say that a feature map $\phi : \mathcal{X} \to \mathbb{R}^p$ is non-degenerate for P if the following conditions are satisfied for random variables $(X, Y) \sim P$ and $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ with n i.i.d. rows sampled from P:

(MOM) $\mathbb{E}\|\phi(X)\|_2^2 < \infty$, such that the moment matrix $\boldsymbol{\Sigma} := \mathbb{E}\phi(X)\phi(X)^\top$ is well-defined,
(COV) $\boldsymbol{\Sigma}$ is invertible,
(FRK) $\phi(\boldsymbol{X}) \in \mathbb{R}^{n \times p}$ almost surely has full rank, i.e., $\operatorname{rank} \phi(\boldsymbol{X}) = \min\{n, p\}$. ◀

Main result In the most general case, we consider a random feature map $\phi : \Omega \times \mathcal{X} \to \mathbb{R}^p, (\omega, \boldsymbol{x}) \mapsto \phi_\omega(\boldsymbol{x})$ that we assume to be measurable, where $(\Omega, \mathcal{A}, \tilde{P})$ is a given probability space. For example, ϕ_ω could be a neural network with random initial parameters ω, or it could simply be a deterministic feature map independent of ω. For a given data set $D = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)) \in (\mathcal{X} \times \mathcal{Y})^n$, we then define the minimum-norm linear regression function using the Moore-Penrose pseudoinverse as in Section 2.2.2:

$$f_{D,\omega}(\boldsymbol{x}) := \phi_\omega(\boldsymbol{x})^\top \phi_\omega(\boldsymbol{X})^+ \boldsymbol{y} \ .$$

We then prove the following main result, where the form provided here is a slightly weaker version of Corollary 5.4.4:[1]

Theorem 3.1.3 (Lower bound for minimum-norm linear regression)**.** *Let P be a probability distribution satisfying Assumption 3.1.1 and let ϕ be a random feature map as above. Let $L(y,t) = (y-t)^2$ be the square loss. Suppose that ϕ_ω is non-degenerate for P for $\tilde{P}$-almost all $\omega \in \Omega$. Then, the following lower bound for the expected excess risk holds:*

$$\mathbb{E}_{\omega\sim\tilde{P}}\mathbb{E}_{D\sim P^n}\mathcal{R}_{L,P}(f_{D,\omega}) - \mathcal{R}^*_{L,P} \geq \begin{cases} \sigma^2\frac{n}{p+1-n} & , \text{ if } p \geq n \\ \sigma^2\frac{p}{n+1-p} & , \text{ if } p \leq n \ . \end{cases}$$

Here, the lower bound for the under-parameterized case $p \leq n$ has been shown in a slightly different setting by Mourtada (2022), while the lower bound for the overparameterized case $p \geq n$ is new. Both cases are proven using Schur complements and Jensen's inequality, although in different ways. The lower bound assumes that ω is drawn independently from D, in particular, it assumes that ω is not learned from D. Since we consider the expected excess risk, we cannot draw conclusions about consistency as defined in Definition 2.1.3, but we could use it to disprove weak consistency of certain learning methods, cf. Remark 2.1.4.

Non-degeneracy results We then analyze the non-degeneracy condition in detail in Section 5.5, where we develop a theory to analyze the strongest assumptions (COV) and (FRK) for analytic feature maps based on the identity theorem for analytic functions. In particular, we prove in Theorem 5.5.6 that (COV) and (FRK) are satisfied for random NN feature maps with non-polynomial analytic activation functions, which extends a result by Nguyen and Hein (2017) to a larger class of activation functions and NNs without biases. Due to the correspondence of minimum-norm linear regression to gradient descent optimization discussed in Theorem 2.2.2, our results imply lower bounds for the expected excess risk of certain NNs where only the last layer is trained with gradient descent. If the last layer is not initialized to zero but randomly from a zero-mean distribution, a bias-variance decomposition could be used to show that the lower bound holds as well. Additionally, we also show (COV) and (FRK) for feature maps corresponding to polynomial kernels and random Fourier features.

Further results In addition to the major results above, the paper discusses further questions such as the (asymptotic) tightness of the lower bound and provides experimental results showing that the shape of the lower bound is approximately achieved for various feature maps.

3.1.2 Active Learning

In Section 3.1.1, we considered the case where the data set D consists of samples $(\boldsymbol{x}_i, y_i)$ that are drawn independently from a distribution P. However, in some settings, it is

[1]Compared to Corollary 5.4.4, Theorem 3.1.3 directly lower bounds the expected excess risk instead of the estimator variance. The bound for the expected excess risk follows from a bias-variance decomposition as in Eq. (5.3). Moreover, the noise assumption (NOI) used in this section conditions directly on $\boldsymbol{x}$, which implies the noise assumption conditioning on $\phi(\boldsymbol{x})$ in Section 5.4 thanks to Lemma 5.H.1.

possible for the inputs $\boldsymbol{x}_i$ to be chosen by the learning method itself, such that only the labels y_i are drawn from an unknown conditional distribution $P(y_i \mid \boldsymbol{x}_i)$. For example, given a function f^* that can be computed by executing a (slow) simulation method, one might want to approximate it using a learned function f_D that can be computed much more efficiently. In this case, a label $y_i = f^*(\boldsymbol{x}_i)$ can be computed for an arbitrary input $\boldsymbol{x}_i$, but requires an evaluation of the (slow) simulation method. Hence, optimizing the choice of inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ can potentially allow reducing the number of computed labels to reach a desired accuracy.

Active learning refers to the setting where the inputs $\boldsymbol{x}_i$ can be chosen sequentially based on the previously chosen inputs and labels $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_{i-1}, y_{i-1})$. For NNs, active learning can be computationally expensive if an NN needs to be re-trained after every acquired label. Therefore, we study batch active learning, where multiple inputs are chosen at once for labeling. Batch active learning can drastically reduce the number of times an NN has to be trained compared to standard active learning, and it can also allow using parallelized labeling methods. Batch active learning can be applied by alternating the following two steps until the data set is considered good enough:

(1) Given a labeled dataset D and an unlabeled finite dataset $\mathcal{X}_{\text{pool}} \subseteq \mathcal{X}$, use a batch active learning method to select a batch of unlabeled data $\mathcal{X}_{\text{batch}} \subseteq \mathcal{X}_{\text{pool}}$, which typically also involves training an NN on D.
(2) Label the batch $\mathcal{X}_{\text{batch}}$ and add it to the training set D.

Framework Our goal is to evaluate and improve different methods for step (1) in the regression setting. Batch active learning methods can be derived from different backgrounds such as core-set construction or a combination of Bayesian ML and information theory. We propose a kernel-based framework that allows to break down many batch active learning methods from different backgrounds into different computational components, which can then be recombined in a flexible manner. A batch active learning method in our framework consists of the following steps:

(a) Train an NN on the training set D.
(b) Construct a base kernel k that represents the trained NN in some sense.
(c) Possibly transform the base kernel k, for example, to make it more efficient or represent posteriors.
(d) Apply a selection method that uses k to select a batch $\mathcal{X}_{\text{batch}}$ of points from $\mathcal{X}_{\text{pool}}$.

We then study existing and novel options for (b)–(d). For example, it is possible to treat the last layer of the NN as a Bayesian linear regression model with the rest of the NN as a fixed feature map. Here, the fixed feature map corresponds to the base kernel used in (b) and the Bayesian linear regression uncertainty can be obtained by a posterior kernel transformation as in Eq. (2.10). This is used, for example, by ACS-FW (Pinsler et al., 2019) and BAIT (Ash et al., 2021), with different selection methods. We also adapt other methods such as BatchBALD (Kirsch et al., 2019) and BADGE (Ash et al., 2019) from the classification to the regression setting and include them in our framework.

New components We propose to replace the corresponding last-layer base kernel k_{ll} with the finite-width neural tangent kernel (NTK)

$$k_{\text{grad}}(\boldsymbol{x}, \boldsymbol{x}') = \langle \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}), \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}') \rangle \ ,$$

where $\boldsymbol{\theta}$ are the trained parameters of the NN. This kernel is motivated by a linearization of the NN around the trained parameters, while the last-layer kernel only linearizes in the last-layer parameters (where the model is already linear). To reduce the dimensionality of the feature space of k_{grad}, we propose to use a simple but efficient tensor sketching approach that exploits a product structure in the network gradients. Moreover, we introduce a novel clustering-inspired selection method, which we call largest cluster maximum distance (LCMD).

Benchmark We introduce a benchmark consisting of 15 large tabular regression data sets. We evaluate many combinations of base kernels, kernel transformations, and selection methods on our benchmark and analyze them with different metrics. We find that in terms of the geometric mean root mean squared error (RMSE) across data sets, all selection methods benefit from replacing the last-layer kernel with a sketched gradient kernel, with only little runtime overhead. Comparing selection methods, we find that our LCMD selection method achieves the best results on most error metrics such as RMSE, closely followed by the BAIT and k-means++ selection methods. When considering the maximum error to get an indication of out-of-distribution performance, other methods that sample the input space more uniformly perform better. Our experiments demonstrate that most of the considered batch active learning methods are relatively insensitive to how frequently the NN is re-trained. We also observe that the advantage of active learning over random sampling varies strongly between data sets, and we found that the amount of this advantage can be predicted rather well by considering the quotient of RMSE and mean absolute error (MAE) on the initial training set, i.e., before the first batch active learning step.

3.1.3 Sampling and Log-Partition Function Estimation

As we have discussed in Section 2.3, sampling from Gibbs distributions P_f with density

$$p_f(\boldsymbol{x}) = \frac{\exp(f(\boldsymbol{x}))}{Z_f}, \qquad Z_f = \int_{\mathcal{X}} \exp(f(\boldsymbol{x})) \,\mathrm{d}\boldsymbol{x}$$

and computing the log-partition function $L_f := \log Z_f$ are two highly relevant tasks, for example, for Bayesian ML and statistical mechanics.

In the paper reproduced in Chapter 7, we consider sampling algorithms and log-partition function estimation algorithms that are allowed to evaluate a function f in n points. We are then interested in how quickly their errors in the sampled distribution or the estimated log-partition function converge to zero, as a function of n. Specifically, we study Gibbs distributions on the cube $\mathcal{X} = [0, 1]^d$ and investigate the worst-case errors over a class $\mathcal{F}_{d,m,B}$ of m-times continuously differentiable functions whose derivatives are bounded by B. Our analysis is motivated by a connection of sampling to optimization in the limit of zero temperature. For optimization, it is known that without computational constraints on the algorithms, a worst-case convergence rate of $O_{m,d}(Bn^{-m/d})$ can be achieved, where the constant in $O_{m,d}$ can depend on m and d (Novak, 1988). Especially, this shows that the convergence rates can be good even in high dimensions if the functions have high smoothness m. Moreover, recently Rudi et al. (2020) have achieved convergence rates close to the optimal ones for classes of m-smooth functions with a runtime like $O(n^{3.5})$, which is polynomial in n and d. Consequently, our paper poses the question of whether similar rates can be (efficiently) achieved for the sampling and log-partition problems.

Information-based complexity First, we study the information-based complexity of the sampling and log-partition problems. Informally speaking, information-based complexity studies the optimal convergence rates that can be achieved when the algorithms are not constrained computationally but only by the number of function evaluations n. For the sampling problem, we consider different metrics to measure the distributional error, such as the total variation (TV) distance and the 1-Wasserstein distance. We show that when the function must be evaluated in deterministic points, the optimal rate is $\Theta(Bn^{-m/d})$ for log-partition estimation and $\Theta(\min\{1, Bn^{-m/d}\})$ for sampling in TV or 1-Wasserstein distances. This can be realized by optimal approximation of f, despite the exponential in the definition of the Gibbs distribution. When randomized evaluation points are allowed, we show that the optimal rate is the same in an optimization regime (roughly $Bn^{-m/d} \gg 1$) but better otherwise. Better rates outside of the optimization regime can be achieved by combining approximation with importance sampling or rejection sampling.

Relations between problems We study multiple ways to relate problems. First, we demonstrate how surrogates obtained by smooth optimal approximation methods allow trading runtime for better convergence rates, and turn algorithms with stochastic evaluation points into algorithms with deterministic evaluation points. We then study how convergence rates behave when sampling algorithms are used for log-partition estimation through thermodynamic integration, and how they behave when log-partition estimation algorithms are used for sampling through bisection sampling. Finally, we establish guarantees for how approximate optimization can be performed by approximate sampling.

Algorithms Subsequently, we turn to the study of specific, usually efficient, algorithms. We prove exact worst-case convergence rates for sampling and log-partition estimation through piecewise constant approximation. We then show that for functions $f \in \mathcal{F}_{d,m,B}$, the density satisfies $p_f \in \mathcal{F}_{d,m,\tilde{B}}$ in general only for $\tilde{B} = \Theta(\max\{1, B\}^{d+m})$, which is not exponential in B but still problematic for density-based approximation algorithms. We obtain further bounds for simple Monte Carlo sampling and log-partition estimation algorithms as well as rejection sampling with uniform proposal distribution, however, all bounds are far from the optimal rate. Our experiments confirm some interesting regime transitions found in the theoretical analysis. Finally, we study an approach by Bach (2022) towards log-partition estimation and show that it cannot exceed the rate $O_{m,d}(Bn^{-2/d})$ in an intermediate regime $B \sim n^{2/d}$.

Relation to regression As we have discussed above, sampling algorithms can be important for Bayesian regression problems. On the flip side, regression can also be used for sampling. We use function approximation to obtain upper bounds on the information-based complexity of the two problems, and we also analyze the possibility to apply algorithms to an approximant instead. However, there are some differences to the regression setting studied in the other papers:

- We consider the options to evaluate the function f in n fixed, adaptive, or even randomized points. This is similar to the setting of active learning, but not to the standard regression setting in supervised learning.
- The functions f considered here can usually be evaluated exactly (up to numerical precision), that is, without noise on the labels $y_i = f(\boldsymbol{x}_i)$.
- We measure the regression error by the L_∞ norm, i.e. $\|f-g\|_\infty = \sup_{x\in\mathcal{X}} |f(\boldsymbol{x})-g(\boldsymbol{x})|$. This is essential here since the function f is used in an exponential function, which is very sensitive to large approximation errors.

The function approximation setting considered in this paper is well-studied, we refer to Wendland (2004) for more details. However, in some cases, we also want the approximant to be as smooth as the original function, for which we use more recent results from Li (2016) and Mirzaei (2015) in Section 7.3.

3.2 Summary and Outlook

In the papers contained in this thesis, we have studied different machine learning problems, mostly related to regression and neural networks. Our first paper proves a lower bound for minimum-norm linear regression that applies to a large class of (random) feature maps, including random neural network feature maps. Our lower bound exhibits a double descent type behavior and extends the class of models where this behavior can be theoretically analyzed further in the direction of simplified neural networks. Recently, Ghosh and Belkin (2022) have found a lower bound similar to ours that also applies to overfitting but non-interpolating linear models. While double descent is rather well understood for linear models by now, a corresponding theory for practically relevant NNs is still lacking, and creating such a theory is an interesting albeit difficult avenue for further research. Beyond the double descent peak, generalization bounds for overfitted strongly over-parameterized NNs are still not satisfactory, and even linearized NNs are only partially understood in this regime (Liang et al., 2020; Mallinar et al., 2022; Lai et al., 2023). A further avenue for theoretical research on NNs is to study the generalization properties of different architectures like convolutional neural networks or Transformers, or to investigate other settings such as active learning, transfer learning, or unsupervised learning.

In our second paper, we studied efficient batch active learning methods for regression with neural networks. We introduced a kernel-based framework with an efficient implementation in our open-source code, which allows us to combine both existing and new components to obtain a wide variety of batch active learning algorithms. We also provide a benchmark consisting of 15 tabular data sets, on which we show that a combination of our new components achieves state-of-the-art performance. We have subsequently applied some of the considered active learning methods to atomistic NNs (Zaverkin et al., 2022). Moreover, our framework has recently been extended to base kernels using predictions of ensembles (Kirsch, 2023). Our framework could be further extended by developing sketching methods that address other types of NN layers than fully-connected layers, such that the sketched gradient kernel could be applied to other types of NNs, and it would be interesting to see for which kinds of NNs such a sketched gradient kernel is particularly beneficial compared to the last-layer kernel. Moreover, an interesting direction for future research would be an extension to multi-output regression and classification, perhaps by using Fisher information similar to how it is used in BAIT (Ash et al., 2021). Similarly, an extension beyond pool-based active learning could be relevant. For example, in atomistic ML, a good pool set is often not known *a priori*, since sampling from the (Gibbs) distribution already requires the expensive computation of atomic force labels.

In our third paper, we investigated convergence rates of sampling and log-partition estimation algorithms for non-log-concave Gibbs distributions. For smooth log-densities, we find a strong discrepancy between known efficiently achievable rates, including the ones proven by us, and the optimal rates provided by our analysis of the information-based complexity of the sampling and log-partition estimation problems. We also provide new results on the relation between the problems of sampling, log-partition estimation, and optimization. Hopefully, our results can contribute to a new theory on non-log-concave sampling and inspire the development of better non-log-concave sampling algorithms. In particular, it would be interesting to see an analysis of more advanced sampling algorithms, for example of the Markov Chain Monte Carlo (MCMC) family, in the non-log-concave setting. On the complementary side, general lower bounds for achievable convergence rates outside of the optimization regime would be desirable. In addition, our research could be extended in many other directions, such as considering different statistical distances for probability distributions or studying alternatives to thermodynamic integration for the log-partition problem.

Part II

Cumulative part

Chapter 4

Declaration to the Cumulative Part

In the cumulative part, this thesis contains three research papers (Chapters 5 – 7). Their title and publication status are:

(1) David Holzmüller, *On the universality of the double descent peak in ridgeless regression*, published at the International Conference on Learning Representations, 2021.
Reference: Holzmüller (2021)
Link: `https://openreview.net/forum?id=0IO5VdnSAaH`
Link to arXiv version: `https://arxiv.org/abs/2010.01851`
This article (specifically version 8 on arXiv) is reproduced in Chapter 5.

(2) David Holzmüller, Viktor Zaverkin, Johannes Kästner, and Ingo Steinwart, *A framework and benchmark for deep batch active learning for regression*, published at the Journal of Machine Learning Research, 2023.
Reference: Holzmüller et al. (2023)
Link: `https://jmlr.org/papers/v24/22-0937.html`
Link to arXiv version: `https://arxiv.org/abs/2203.09410`
This article (specifically version 4 on arXiv) is reproduced in Chapter 6.

(3) David Holzmüller and Francis Bach, *Convergence rates for non-log-concave sampling and log-partition estimation*, preprint available on arXiv, 2023.
Reference: Holzmüller and Bach (2023)
Link: `https://arxiv.org/abs/2303.03237`
This article (specifically version 3 on arXiv) is reproduced in Chapter 7.

I, *David Holzmüller*, hereby declare that the co-author lists are complete and that I have not reproduced, without acknowledgement, the work of another. I declare that I majorly contributed to these articles, including in particular the phase of problem selection, literature research, and the derivation of theoretical and numerical results as well as the writing process. This applies to all three articles listed above.

Chapters 5 – 7 reproduce the original articles except for the following changes:

- for the first article, version 8 on arXiv is reproduced, which contains some minor corrections compared to the publication at the ICLR conference (version 5 on arXiv).

- for the second article, version 4 on arXiv is reproduced, which contains some minor corrections compared to the publication at JMLR.
- the articles have been formatted in the style of this thesis and the numbers of theorems, definitions, etc. have been adjusted accordingly.
- some typing errors have been corrected.
- the bibliography for the three articles is reproduced at the end of the thesis together with the bibliography for the introduction.

Except for these changes, Chapters 5 – 7 are reproductions of the corresponding papers listed above.

Place, Date	David Holzmüller

Chapter 5

On the Universality of the Double Descent Peak in Ridgeless Regression

David Holzmüller[1]

Published in International Conference on Learning Representations (2021)
Reference: Holzmüller (2021), link: `https://openreview.net/forum?id=0IO5VdnSAaH`

Abstract

We prove a non-asymptotic distribution-independent lower bound for the expected mean squared generalization error caused by label noise in ridgeless linear regression. Our lower bound generalizes a similar known result to the overparameterized (interpolating) regime. In contrast to most previous works, our analysis applies to a broad class of input distributions with almost surely full-rank feature matrices, which allows us to cover various types of deterministic or random feature maps. Our lower bound is asymptotically sharp and implies that in the presence of label noise, ridgeless linear regression does not perform well around the interpolation threshold for any of these feature maps. We analyze the imposed assumptions in detail and provide a theory for analytic (random) feature maps. Using this theory, we can show that our assumptions are satisfied for input distributions with a (Lebesgue) density and feature maps given by random deep neural networks with analytic activation functions like sigmoid, tanh, softplus, or GELU. As further examples, we show that feature maps from random Fourier features and polynomial kernels also satisfy our assumptions. We complement our theory with further experimental and analytic results.

5.1 Introduction

Seeking a better understanding of the successes of deep learning, Zhang et al. (2017) pointed out that deep neural networks can achieve very good performance despite being able to fit random noise, which sparked the interest of many researchers in studying the performance of interpolating learning methods. Belkin et al. (2018) made a similar

[1]ISA, University of Stuttgart, Stuttgart, Germany

observation for kernel methods and showed that classical generalization bounds are unable to explain this phenomenon. Belkin et al. (2019) observed a "double descent" phenomenon in various learning models, where the test error first decreases with increasing model complexity, then increases towards the "interpolation threshold" where the model is first able to fit the training data perfectly, and then decreases again in the "overparameterized" regime where the model capacity is larger than the training set. This phenomenon has also been discovered in several other works (Bös and Opper, 1997; Advani et al., 2020; Neal et al., 2019; Spigler et al., 2019). Nakkiran et al. (2021a) performed a large empirical study on deep neural networks and found that double descent can not only occur as a function of model capacity, but also as a function of the number of training epochs or as a function of the number of training samples.

Theoretical investigations of the double descent phenomenon have mostly focused on specific unregularized ("ridgeless") or weakly regularized linear regression models. These linear models can be described via i.i.d. samples $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d$, where the covariates $\boldsymbol{x}_i$ are mapped to feature representations $\boldsymbol{z}_i = \phi(\boldsymbol{x}_i) \in \mathbb{R}^p$ via a (potentially random) feature map ϕ, and (ridgeless) linear regression is then performed on the transformed samples $(\boldsymbol{z}_i, y_i)$. While linear regression with random features can be understood as a simplified model of fully trained neural networks, it is also interesting in its own right: For example, random Fourier features (Rahimi and Recht, 2008) and random neural network features (see e.g. Cao et al., 2018; Scardapane and Wang, 2017) have gained a notable amount of attention.

Unfortunately, existing theoretical investigations of double descent are usually limited in one or more of the following ways:

(1) They assume that the $\boldsymbol{z}_i$ (or a linear transformation thereof) have (centered) i.i.d. components. This assumption is made by Hastie et al. (2022), while Advani et al. (2020) and Belkin et al. (2020) even assume that the $\boldsymbol{z}_i$ follow a Gaussian distribution. While the assumption of i.i.d. components facilitates the application of some random matrix theory results, it excludes most feature maps: For feature maps ϕ with $d < p$, the $\boldsymbol{z}_i$ will usually be concentrated on a d-dimensional submanifold of $\mathbb{R}^p$, and will therefore usually *not* have i.i.d. components.

(2) They assume a (shallow) random feature model with a fixed distribution of the $\boldsymbol{x}_i$, e.g. an isotropic Gaussian distribution or a uniform distribution on a sphere. Examples of this are the single-layer random neural network feature models by Hastie et al. (2022) in the unregularized case and by Mei and Montanari (2022); d'Ascoli et al. (2020) in the regularized case. A simple Fourier model with $d = 1$ has been studied by Belkin et al. (2020). While these analyses provide insights for some practically relevant random feature models, the assumptions on the input distribution prevent them from applying to real-world data.

(3) Their analysis only applies in a high-dimensional limit where $n, p \to \infty$ and $n/p \to \gamma$, where $\gamma \in (0, \infty)$ is a constant. This applies to all works mentioned in (1) and (2) except the model by Belkin et al. (2020) where the $\boldsymbol{z}_i$ follow a standard Gaussian distribution.

In this paper, we provide an analysis under significantly weaker assumptions. We introduce the basic setting of our paper in Section 5.2 and Section 5.3. Our main contributions are:

- In Section 5.4, we show a non-asymptotic distribution-independent lower bound for the expected excess risk of ridgeless linear regression with (random) features. While the underparameterized bound is adapted from a minimax lower bound in Mourtada (2022), the overparameterized bound is new and perfectly complements the underparameterized version. The obtained general lower bound relies on significantly weaker assumptions than most previous works and shows that there is only limited potential to reduce the sensitivity of unregularized linear models to label noise via engineering better feature maps.
- In Section 5.5, we show that our lower bound applies to a large class of input distributions and feature maps including random deep neural networks, random Fourier features, and polynomial kernels. This analysis is also relevant for related work where similar assumptions are not investigated (e.g. Mourtada, 2022; Muthukumar et al., 2020). For random deep neural networks, our result requires weaker assumptions than a related result by Nguyen and Hein (2017).
- In Section 5.6 and Appendix 5.C, we compare our lower bound to new theoretical and experimental results for specific examples, including random neural network feature maps as well as finite-width Neural Tangent Kernels (Jacot et al., 2018). We also show that our lower bound is asymptotically sharp in the limit $n, p \to \infty$.

Similar to this paper, Muthukumar et al. (2020) study the "fundamental price of interpolation" in the overparameterized regime, providing a probabilistic lower bound for the generalization error under the assumption of subgaussian features or (suitably) bounded features. We explain the difference to our lower bound in detail in Appendix 5.L, showing that our overparameterized lower bound for the expected generalization error requires significantly weaker assumptions, that it is uniform across feature maps, and that it yields a more extreme interpolation peak.

Our lower bound also applies to a large class of kernels if they can be represented using a feature map with finite-dimensional feature space, i.e. $p < \infty$. For ridgeless regression with certain classes of kernels, lower or upper bounds have been derived (Liang and Rakhlin, 2020; Rakhlin and Zhai, 2019; Liang et al., 2020). However, as explained in more detail in Appendix 5.K, these analyses impose restrictions on the kernels that allow them to ignore "double descent" type phenomena in the feature space dimension p.

Beyond Double Descent, a series of papers have studied "Multiple Descent" phenomena theoretically and empirically, both with respect to the number of parameters p and the input dimension d. Adlam and Pennington (2020a) and d'Ascoli et al. (2020) theoretically investigate Triple Descent phenomena. Nakkiran et al. (2021b) argue that Double Descent can be mitigated by optimal regularization. They also empirically observe a form of Triple Descent in an unregularized model. Liang et al. (2020) prove an upper bound exhibiting infinitely many peaks and empirically observe Multiple Descent. Chen et al. (2021) show that in ridgeless linear regression, the feature distributions can be designed to control the locations of ascents and descents in the double descent curve for a "dimension-normalized" noise-induced generalization error. Our lower bound provides a fundamental limit to this "designability" of the generalization curve for methods that can interpolate with probability one in the overparameterized regime.

Proofs of our statements can be found in the appendix. We provide code to reproduce all of our experimental results at

`https://github.com/dholzmueller/universal_double_descent`

and, together with the computed data, at `https://doi.org/10.18419/darus-1771`.

5.2 Basic Setting and Notation

Following Györfi et al. (2002), we consider the scenario where the samples $(\boldsymbol{x}_i, y_i)$ of a data set $D = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)) \in (\mathbb{R}^d \times \mathbb{R})^n$ are sampled independently from a probability distribution P on $\mathbb{R}^d \times \mathbb{R}$, i.e. $D \sim P^n$.[2] We define

$$\boldsymbol{X} \coloneqq \begin{pmatrix} \boldsymbol{x}_1^\top \\ \vdots \\ \boldsymbol{x}_n^\top \end{pmatrix} \in \mathbb{R}^{n \times d}, \qquad \boldsymbol{y} \coloneqq \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^n .$$

We also consider random variables $(\boldsymbol{x}, y) \sim P$ that are independent of D and denote the distribution of $\boldsymbol{x}$ by P_X. The (least squares) *population risk* of a function $f : \mathbb{R}^d \to \mathbb{R}$ is defined as

$$R_P(f) \coloneqq \mathbb{E}_{\boldsymbol{x},y}(y - f(\boldsymbol{x}))^2 .$$

We assume $\mathbb{E}y^2 < \infty$. Then, R_P is minimized by the target function f_P^* given by

$$f_P^*(\boldsymbol{x}) = \mathbb{E}(y|\boldsymbol{x}) ,$$

we have $R_P(f_P^*) < \infty$, and the *excess risk* (a.k.a. generalization error) of a function f is

$$R_P(f) - R_P(f_P^*) = \mathbb{E}_{\boldsymbol{x}}(f(\boldsymbol{x}) - f_P^*(\boldsymbol{x}))^2 .$$

Notation For two symmetric matrices, we write $\boldsymbol{A} \succeq \boldsymbol{B}$ if $\boldsymbol{A} - \boldsymbol{B}$ is positive semidefinite and $\boldsymbol{A} \succ \boldsymbol{B}$ if $\boldsymbol{A} - \boldsymbol{B}$ is positive definite. For a symmetric matrix $\boldsymbol{S} \in \mathbb{R}^{n \times n}$, we let $\lambda_1(\boldsymbol{S}) \geq \ldots \geq \lambda_n(\boldsymbol{S})$ be its eigenvalues in descending order. We denote the trace of $\boldsymbol{A}$ by $\operatorname{tr}(\boldsymbol{A})$ and the Moore-Penrose pseudoinverse of $\boldsymbol{A}$ by $\boldsymbol{A}^+$. For $\phi : \mathbb{R}^d \to \mathbb{R}^p$ and $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, we let $\phi(\boldsymbol{X}) \in \mathbb{R}^{n \times p}$ be the matrix with ϕ applied to each of the *rows* of $\boldsymbol{X}$ individually. For a set $\mathcal{A}$, we denote its indicator function by $\mathbb{1}_{\mathcal{A}}$. For a random variable $\boldsymbol{x}$, we say that $\boldsymbol{x}$ has a Lebesgue density if P_X can be represented by a probability density function (w.r.t. the Lebesgue measure). We say that $\boldsymbol{x}$ is nonatomic if for all possible values $\widetilde{\boldsymbol{x}}$, $P(\boldsymbol{x} = \widetilde{\boldsymbol{x}}) = 0$.[3] We denote the uniform distribution on a set $\mathcal{A}$, e.g. the unit sphere $\mathbb{S}^{p-1} \subseteq \mathbb{R}^p$, by $\mathcal{U}(\mathcal{A})$. We denote the normal (Gaussian) distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ by $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. For $n \in \mathbb{N}$, we define $[n] \coloneqq \{1, \ldots, n\}$.

We review relevant matrix facts, e.g. concerning the Moore-Penrose pseudoinverse, in Appendix 5.B.

[2]Although many of our theorems apply to general domains $\boldsymbol{x}_i \in \mathcal{X}$ and not just $\mathcal{X} = \mathbb{R}^d$, we set $\mathcal{X} = \mathbb{R}^d$ for notational simplicity. We require $\mathcal{X} = \mathbb{R}^d$ whenever we assume that the distribution of the $\boldsymbol{x}_i$ has a Lebesgue density or work with analytic feature maps.

[3]For Borel measures as considered here, this is equivalent to the usual definition of non-atomic measures, see section IV in Knowles (1967).

5.3 Linear Regression With (Random) Features

The most general setting that we will consider in this paper is ridgeless linear regression in random features: Given a random variable $\boldsymbol{\theta}$ that is independent of the data set D and an associated random feature map $\phi_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^p$, we define the estimator

$$f_{\boldsymbol{X},\boldsymbol{y},\boldsymbol{\theta}}(\boldsymbol{x}) \coloneqq \phi_{\boldsymbol{\theta}}(\boldsymbol{x})^\top \phi_{\boldsymbol{\theta}}(\boldsymbol{X})^+ \boldsymbol{y} ,$$

which simply performs unregularized linear regression with random features. As a special case, the feature map $\phi_{\boldsymbol{\theta}}$ may be deterministic, in which case we drop the index $\boldsymbol{\theta}$. An even more specialized case is ordinary linear regression, where $d = p$ and $\phi_{\boldsymbol{\theta}} = \mathrm{id}$, yielding $f_{\boldsymbol{X},\boldsymbol{y}}(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{X}^+ \boldsymbol{y}$.

As described in Hastie et al. (2022), the ridgeless linear regression parameter $\widehat{\boldsymbol{\beta}} \coloneqq \phi_{\boldsymbol{\theta}}(\boldsymbol{X})^+ \boldsymbol{y}$

- has minimal Euclidean norm among all parameters $\boldsymbol{\beta}$ minimizing $\|\phi_{\boldsymbol{\theta}}(\boldsymbol{X})\boldsymbol{\beta} - \boldsymbol{y}\|_2^2$,
- is the limit of gradient descent with sufficiently small step size on $L(\boldsymbol{\beta}) \coloneqq \|\phi_{\boldsymbol{\theta}}(\boldsymbol{X})\boldsymbol{\beta} - \boldsymbol{y}\|_2^2$ with initialization $\boldsymbol{\beta}^{(0)} \coloneqq \boldsymbol{0}$, and
- is the limit of ridge regression with regularization $\lambda > 0$ for $\lambda \searrow 0$:

$$\widehat{\boldsymbol{\beta}} = \lim_{\lambda \searrow 0} \phi_{\boldsymbol{\theta}}(\boldsymbol{X})^\top (\phi_{\boldsymbol{\theta}}(\boldsymbol{X})\phi_{\boldsymbol{\theta}}(\boldsymbol{X})^\top + \lambda \boldsymbol{I}_n)^{-1} \boldsymbol{y} . \tag{5.1}$$

For a fixed feature map ϕ, the kernel trick provides a correspondence between ridgeless linear regression with ϕ and ridgeless kernel regression with the kernel $k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \coloneqq \phi(\boldsymbol{x})^\top \phi(\tilde{\boldsymbol{x}})$ via

$$f_{\boldsymbol{X},\boldsymbol{y}}(\boldsymbol{x}) = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{X})^+ \boldsymbol{y} = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{X})^\top (\phi(\boldsymbol{X})\phi(\boldsymbol{X})^\top)^+ \boldsymbol{y} = k(\boldsymbol{x}, \boldsymbol{X}) k(\boldsymbol{X}, \boldsymbol{X})^+ \boldsymbol{y} , \tag{5.2}$$

where

$$k(\boldsymbol{x}, \boldsymbol{X}) \coloneqq \begin{pmatrix} k(\boldsymbol{x}, \boldsymbol{x}_1) \\ \vdots \\ k(\boldsymbol{x}, \boldsymbol{x}_n) \end{pmatrix}, \qquad k(\boldsymbol{X}, \boldsymbol{X}) \coloneqq \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{pmatrix} .$$

5.4 A Lower Bound

In this section, we state our main theorem, which provides a non-asymptotic distribution-independent lower bound on the expected excess risk.

The expected excess risk $\mathbb{E}_{\boldsymbol{X},\boldsymbol{y},\boldsymbol{\theta}} R_P(f_{\boldsymbol{X},\boldsymbol{y},\boldsymbol{\theta}}) - R_P(f_P^*)$ can be decomposed into several different contributions (see e.g. d'Ascoli et al., 2020). In the following, we will focus on the contribution of label noise to the expected excess risk for the estimators $f_{\boldsymbol{X},\boldsymbol{y},\boldsymbol{\theta}}$ considered in Section 5.3. Using a bias-variance decomposition with respect to $\boldsymbol{y}$, it is not hard to show that the label-noise-induced error provides a lower bound for the expected excess risk:

$$\begin{aligned}\mathcal{E}_{\text{Noise}} &\coloneqq \mathbb{E}_{\boldsymbol{X},\boldsymbol{y},\boldsymbol{\theta},\boldsymbol{x}} \left(f_{\boldsymbol{X},\boldsymbol{y},\boldsymbol{\theta}}(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{y}|\boldsymbol{X}} f_{\boldsymbol{X},\boldsymbol{y},\boldsymbol{\theta}}(\boldsymbol{x})\right)^2 \\ &\leq \mathbb{E}_{\boldsymbol{X},\boldsymbol{\theta},\boldsymbol{x}} \left[\mathbb{E}_{\boldsymbol{y}|\boldsymbol{X}} \left(f_{\boldsymbol{X},\boldsymbol{y},\boldsymbol{\theta}}(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{y}|\boldsymbol{X}} f_{\boldsymbol{X},\boldsymbol{y},\boldsymbol{\theta}}(\boldsymbol{x})\right)^2 + \left(\mathbb{E}_{\boldsymbol{y}|\boldsymbol{X}} f_{\boldsymbol{X},\boldsymbol{y},\boldsymbol{\theta}}(\boldsymbol{x}) - f_P^*(\boldsymbol{x})\right)^2\right]\end{aligned}$$

$$\begin{aligned}&= \mathbb{E}_{\boldsymbol{X},\boldsymbol{y},\boldsymbol{\theta},\boldsymbol{x}} \left(f_{\boldsymbol{X},\boldsymbol{y},\boldsymbol{\theta}}(\boldsymbol{x}) - f_P^*(\boldsymbol{x})\right)^2 \\ &= \mathbb{E}_{\boldsymbol{X},\boldsymbol{y},\boldsymbol{\theta}}(R_P(f_{\boldsymbol{X},\boldsymbol{y},\boldsymbol{\theta}}) - R_P(f_P^*)) \ . \end{aligned} \tag{5.3}$$

For linear models as considered here, it is not hard to see that $\mathcal{E}_{\text{Noise}}$ does not depend on f_P^* and is equal to the expected excess risk in the special case $f_P^* \equiv 0$.

In the following, we first consider the setting where the feature map ϕ is deterministic. We will consider linear regression on $\boldsymbol{z} := \phi(\boldsymbol{x})$ and $\boldsymbol{Z} := \phi(\boldsymbol{X})$ and formulate our assumptions directly w.r.t. the distribution P_Z of $\boldsymbol{z}$, hiding the dependence on the feature map ϕ. While the distribution P_X of $\boldsymbol{x}$ is usually fixed and determined by the problem, the distribution P_Z can be actively influenced by choosing a suitable feature map ϕ. We will analyze in Section 5.5 how the assumptions on P_Z can be translated back to assumptions on P_X and assumptions on ϕ.

Remark 5.4.1. For typical feature maps, we have $p > d$, P_Z is concentrated on a d-dimensional submanifold of $\mathbb{R}^p$ and the components of $\boldsymbol{z}$ are not independent. A simple example (cf. Proposition 5.5.4) is a polynomial feature map $\phi : \mathbb{R}^1 \to \mathbb{R}^p, x \mapsto (1, x, x^2, \ldots, x^{p-1})$. The imposed assumptions on P_Z should hence allow for such distributions on submanifolds and *not* require independent components. ◀

Definition 5.4.2. Assuming that $\mathbb{E}\|\boldsymbol{z}\|_2^2 < \infty$, i.e. (MOM) in Theorem 5.4.3 holds, we can define the (positive semidefinite) second moment matrix

$$\boldsymbol{\Sigma} := \mathbb{E}_{\boldsymbol{z}\sim P_Z}\left[\boldsymbol{z}\boldsymbol{z}^\top\right] \in \mathbb{R}^{p\times p} \ .$$

If $\mathbb{E}\boldsymbol{z} = 0$, $\boldsymbol{\Sigma}$ is also the covariance matrix of $\boldsymbol{z}$. If $\boldsymbol{\Sigma}$ is invertible, i.e. (COV) in Theorem 5.4.3 holds, the rows $\boldsymbol{w}_i := \boldsymbol{\Sigma}^{-1/2}\boldsymbol{x}_i$ of the "whitened" data matrix $\boldsymbol{W} := \boldsymbol{Z}\boldsymbol{\Sigma}^{-1/2}$ satisfy $\mathbb{E}\boldsymbol{w}_i\boldsymbol{w}_i^\top = \boldsymbol{I}_p$. ◀

With these preparations, we can now state our main theorem. Its assumptions and the obtained lower bound will be discussed in Section 5.5 and Section 5.6, respectively.

Theorem 5.4.3 (Main result). *Let $n, p \geq 1$. Assume that P and ϕ satisfy:*

(INT) $\mathbb{E}y^2 < \infty$ *and hence* $R_P(f_P^*) < \infty$,
(NOI) $\operatorname{Var}(y|\boldsymbol{z}) \geq \sigma^2$ *almost surely over* $\boldsymbol{z}$,
(MOM) $\mathbb{E}\|\boldsymbol{z}\|_2^2 < \infty$, *i.e.* $\boldsymbol{\Sigma}$ *exists and is finite,*
(COV) $\boldsymbol{\Sigma}$ *is invertible,*
(FRK) $\boldsymbol{Z} \in \mathbb{R}^{n\times p}$ *almost surely has full rank, i.e.* $\operatorname{rank}\boldsymbol{Z} = \min\{n, p\}$.

Then, for the ridgeless linear regression estimator $f_{\boldsymbol{Z},\boldsymbol{y}}(\boldsymbol{z}) = \boldsymbol{z}^\top\boldsymbol{Z}^+\boldsymbol{y}$, the following holds:

$$\text{If } p \geq n, \qquad \mathcal{E}_{\text{Noise}} \overset{\text{(I)}}{\geq} \sigma^2\mathbb{E}_{\boldsymbol{Z}}\operatorname{tr}((\boldsymbol{Z}^+)^\top\boldsymbol{\Sigma}\boldsymbol{Z}^+) \overset{\text{(II)}}{\geq} \sigma^2\mathbb{E}_{\boldsymbol{Z}}\operatorname{tr}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1}) \overset{\text{(IV)}}{\geq} \sigma^2\frac{n}{p+1-n} \ .$$

$$\text{If } p \leq n, \qquad \mathcal{E}_{\text{Noise}} \overset{\text{(I)}}{\geq} \sigma^2\mathbb{E}_{\boldsymbol{Z}}\operatorname{tr}((\boldsymbol{Z}^+)^\top\boldsymbol{\Sigma}\boldsymbol{Z}^+) \overset{\text{(III)}}{=} \sigma^2\mathbb{E}_{\boldsymbol{Z}}\operatorname{tr}((\boldsymbol{W}^\top\boldsymbol{W})^{-1}) \overset{\text{(V)}}{\geq} \sigma^2\frac{p}{n+1-p} \ .$$

Here, the matrix inverses exist almost surely in the considered cases. Moreover, we have:

- *If (NOI) holds with equality, then* (I) *holds with equality.*
- *If $n = p$ or $\boldsymbol{\Sigma} = \lambda\boldsymbol{I}_p$ for some $\lambda > 0$, then* (II) *holds with equality.*

For a discussion on how $\boldsymbol{\Sigma}$ influences $\mathcal{E}_{\text{Noise}}$, we refer to Remark 5.G.1. We can extend Theorem 5.4.3 to random features if it holds for almost all of the random feature maps:

Corollary 5.4.4 (Random features). *Let $\boldsymbol{\theta} \sim P_\Theta$ be a random variable such that $\phi_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^p$ is a random feature map. Consider the random features regression estimator $f_{\boldsymbol{X},\boldsymbol{y},\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{z}_{\boldsymbol{\theta}}^\top \boldsymbol{Z}_{\boldsymbol{\theta}}^+ \boldsymbol{y}$ with $\boldsymbol{z}_{\boldsymbol{\theta}} := \phi_{\boldsymbol{\theta}}(\boldsymbol{x})$ and $\boldsymbol{Z}_{\boldsymbol{\theta}} := \phi_{\boldsymbol{\theta}}(\boldsymbol{X})$. If for P_Θ-almost all $\widetilde{\boldsymbol{\theta}}$, the assumptions of Theorem 5.4.3 are satisfied for $\boldsymbol{z} = \boldsymbol{z}_{\widetilde{\boldsymbol{\theta}}}$ and $\boldsymbol{Z} = \boldsymbol{Z}_{\widetilde{\boldsymbol{\theta}}}$ (with the corresponding matrix $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{\widetilde{\boldsymbol{\theta}}}$), then*

$$\mathcal{E}_{\text{Noise}} \geq \begin{cases} \sigma^2 \frac{n}{p+1-n} & \text{if } p \geq n, \\ \sigma^2 \frac{p}{n+1-p} & \text{if } p \leq n. \end{cases}$$

The main novelty in Theorem 5.4.3 is the explicit uniform lower bound (IV) for $p \geq n$: The lower bound (V) for $p \leq n$ follows by adapting Corollary 1 in Mourtada (2022). Statements similar to (I), (II) and (III) have also been proven, see e.g. Hastie et al. (2022) and Theorem 1 in Muthukumar et al. (2020). However, as discussed in Section 5.1, Hastie et al. (2022) make significantly stronger assumptions for computing the expectation. In Appendix 5.L, we explain in more detail that the probabilistic overparameterized lower bound of Muthukumar et al. (2020) is not distribution-independent and only applies to a smaller class of distributions than our lower bound. For a discussion on how Theorem 5.4.3 applies to kernels with finite-dimensional feature space, we refer to Appendix 5.K.

5.5 When are the Assumptions Satisfied?

In this section, we want to discuss the assumptions of Theorem 5.4.3 and provide different results helping to verify these assumptions for various input distributions and feature maps. The theory will be particularly nice for analytic feature maps, which we define now:

Definition 5.5.1 (Analytic function). A function $f : \mathbb{R}^d \to \mathbb{R}$ is called (real) analytic if for all $\boldsymbol{z} \in \mathbb{R}^d$, the Taylor series of f around $\boldsymbol{z}$ converges to f in a neighborhood of $\boldsymbol{z}$. A function $f : \mathbb{R}^d \to \mathbb{R}^p, \boldsymbol{z} \mapsto (f_1(\boldsymbol{z}), \ldots, f_p(\boldsymbol{z}))$ is called (real) analytic if $f_1, \ldots, f_p$ are analytic. ◀

Sums, products, and compositions of analytic functions are analytic, cf. e.g. Section 2.2 in Krantz and Parks (2002). We will discuss examples of analytic functions later in this section.

Proposition 5.5.2 (Characterization of (COV) and (FRK)). *Consider the setting of Theorem 5.4.3 and let* FRK(n) *be the statement that (FRK) holds for n. Then,*

(i) Let $n \geq 1$. Then, FRK(n) *iff $P(\boldsymbol{z} \in U) = 0$ for all linear subspaces $U \subseteq \mathbb{R}^p$ of dimension* $\min\{n, p\} - 1$.

(ii) Let (MOM) hold. Then, (COV) holds iff $P(\boldsymbol{z} \in U) < 1$ for all linear subspaces $U \subseteq \mathbb{R}^p$ of dimension $p - 1$.

Assuming that (MOM) holds such that (COV) is well-defined, consider the following statements:

(a) $\mathrm{FRK}(p)$ *holds.*
(b) $\mathrm{FRK}(n)$ *holds for all* $n \geq 1$.
(c) (COV) holds.
(d) There exists a fixed deterministic matrix $\widetilde{\boldsymbol{X}} \in \mathbb{R}^{p \times d}$ *such that* $\det(\phi(\widetilde{\boldsymbol{X}})) \neq 0$.

We have (a) $\Leftrightarrow$ *(b)* $\Rightarrow$ *(c)* $\Rightarrow$ *(d). Furthermore, if* $\boldsymbol{x} \in \mathbb{R}^d$ *has a Lebesgue density and* ϕ *is analytic, then (a) – (d) are equivalent.*

With this in mind, we can characterize the assumptions now:

- The assumption (INT) is standard (see e.g. Section 1.6 in Györfi et al., 2002) and guarantees $R_P(f_P^*) < \infty$, such that the excess risk is well-defined.
- The assumption (NOI) is required to ensure the existence of sufficient label noise. Importantly, Lemma 5.H.1 shows that (NOI), i.e. $\mathrm{Var}(y|\boldsymbol{z}) \geq \sigma^2$ almost surely over $\boldsymbol{z}$, holds if $\mathrm{Var}(y|\boldsymbol{x}) \geq \sigma^2$ almost surely over $\boldsymbol{x}$. All Double Descent papers from Section 5.1 make the stronger assumption that the distribution of $y - \mathbb{E}(y|\boldsymbol{x})$ is independent of $\boldsymbol{x}$ or even a fixed Gaussian.
- The assumption (MOM) can be reformulated as $\mathbb{E}\|\boldsymbol{z}\|_2^2 = \mathbb{E}\|\phi(\boldsymbol{x})\|_2^2 = \mathbb{E}k(\boldsymbol{x}, \boldsymbol{x}) < \infty$. For example, if k or equivalently ϕ are bounded, or if ϕ is continuous and $\|\boldsymbol{x}\|_2$ is bounded, then (MOM) is satisfied. Such assumptions are frequently imposed (see e.g. Chapters 6, 7, 8, 10 in Györfi et al., 2002). In this sense, (MOM) is a standard assumption.
- The assumptions (COV) and $\mathrm{FRK}(n)$ are implied by $\mathrm{FRK}(p)$ and are even equivalent to $\mathrm{FRK}(p)$ in the underparameterized case $p \leq n$. In the following, we will therefore focus on proving $\mathrm{FRK}(p)$ for various ϕ and P_X. In the case $p = n$, $\mathrm{FRK}(p)$ ensures that $f_{\boldsymbol{X},\boldsymbol{y}}$ almost surely interpolates the data, or equivalently that the kernel matrix $k(\boldsymbol{X}, \boldsymbol{X})$ almost surely has full rank. Importantly, assuming $\mathrm{FRK}(p)$ is weaker than assuming a strictly positive definite kernel since strictly positive definite kernels require $p = \infty$. Example 5.D.1 shows that the assumption (FRK) in Theorem 5.4.3 cannot be removed.

For the analytic function $\phi = \mathrm{id}$ with $d = p$, Proposition 5.5.2 yields a simple sufficient criterion: If $\boldsymbol{z}$ has a Lebesgue density, then (FRK) holds for all n. This assumption is already more realistic than assuming i.i.d. components. However, Proposition 5.5.2 is also very useful for other analytic feature maps, as we will see in the remainder of this section.

Remark 5.5.3. Suppose that $\phi \not\equiv 0$ is analytic, $\boldsymbol{x}$ has a Lebesgue density, and that (INT), (MOM), and (NOI) are satisfied. If (d) in Proposition 5.5.2 does not hold, there exists $\tilde{p} < p$ such that the lower bound from Theorem 5.4.3 holds with p replaced by $\tilde{p}$: Let $U := \mathrm{Span}\{\phi(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathbb{R}^d\}$. Since $\phi \not\equiv 0$, $\tilde{p} := \dim U \geq 1$. Moreover, (d) holds iff $\dim U = p$. Take any isometric isomorphism $\psi : U \to \mathbb{R}^{\tilde{p}}$ and define the feature map $\tilde{\phi} : \mathbb{R}^d \to \mathbb{R}^{\tilde{p}}, \boldsymbol{x} \mapsto \psi(\phi(\boldsymbol{x}))$. Then, $\tilde{\phi}$ is analytic since ψ is linear, and $\tilde{\phi}$ satisfies (d), hence Theorem 5.4.3 can be applied to $\tilde{\phi}$. However, ϕ and $\tilde{\phi}$ lead to the same kernel k since ψ is isometric, hence to the same estimator $f_{\boldsymbol{X},\boldsymbol{y}}$ by Eq. (5.2) and hence to the same $\mathcal{E}_{\mathrm{Noise}}$. ◀

Proposition 5.5.4 (Polynomial kernel)**.** *Let* $m, d \geq 1$ *and* $c > 0$*. For* $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathbb{R}^d$*, define the polynomial kernel* $k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) := (\boldsymbol{x}^\top \tilde{\boldsymbol{x}} + c)^m$*. Then, there exists a feature map* $\phi : \mathbb{R}^d \to \mathbb{R}^p$*,* $p := \binom{m+d}{m}$*, such that:*

(a) $k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \phi(\boldsymbol{x})^\top \phi(\tilde{\boldsymbol{x}})$ *for all* $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathbb{R}^d$*, and*

(b) if $\boldsymbol{x} \in \mathbb{R}^d$ *has a Lebesgue density and we use* $\boldsymbol{z} = \phi(\boldsymbol{x})$*, then (FRK) is satisfied for all* n*.*

Proposition 5.5.4 says that the lower bound from Theorem 5.4.3 holds for ridgeless kernel regression with the polynomial kernel with $p \coloneqq \binom{m+d}{m}$ if $\boldsymbol{x}$ has a Lebesgue density and $\mathbb{E}\|\boldsymbol{z}\|_2^2 = \mathbb{E}k(\boldsymbol{x}, \boldsymbol{x}) = \mathbb{E}(\|\boldsymbol{x}\|_2^2 + c)^m < \infty$. The proof of Proposition 5.5.4 can be extended to the case $c = 0$, where one needs to choose $p = \binom{m+d-1}{m}$. In general, we discuss in Appendix 5.K that Theorem 5.4.3 can often be applied to ridgeless kernel regression, where p needs to be chosen as the minimal feature space dimension for which k can still be represented.

We can also extend our theory to analytic random feature maps:

Proposition 5.5.5 (Random feature maps)**.** *Consider feature maps* $\phi_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^p$ *with (random) parameter* $\boldsymbol{\theta} \in \mathbb{R}^q$*. Suppose the map* $(\boldsymbol{\theta}, \boldsymbol{x}) \mapsto \phi_{\boldsymbol{\theta}}(\boldsymbol{x})$ *is analytic and that* $\boldsymbol{\theta}$ *and* $\boldsymbol{x}$ *are independent and have Lebesgue densities. If there exist fixed* $\widetilde{\boldsymbol{\theta}} \in \mathbb{R}^q, \widetilde{\boldsymbol{X}} \in \mathbb{R}^{p \times d}$ *with* $\det(\phi_{\widetilde{\boldsymbol{\theta}}}(\widetilde{\boldsymbol{X}})) \neq 0$*, then almost surely over* $\boldsymbol{\theta}$*, (FRK) holds for all* n *for* $\boldsymbol{z} = \phi_{\boldsymbol{\theta}}(\boldsymbol{x})$*.*

In Appendix 5.C, we demonstrate that Proposition 5.5.5 can be used to computationally verify (FRK) for analytic random feature maps.

Up until now, we have assumed that $\boldsymbol{x}$ has a Lebesgue density. It is desirable to weaken this assumption, such that $\boldsymbol{x}$ can, for example, be concentrated on a submanifold of $\mathbb{R}^d$. It is necessary for FRK(p) with $p \geq 2$ that $\boldsymbol{x}$ is nonatomic, such that the $\boldsymbol{x}_i$ are distinct with probability one. In general, this is not sufficient: For example, if $\phi = \mathrm{id}$ and $\boldsymbol{x}$ lives on a proper linear subspace of $\mathbb{R}^d$, FRK(p) is not satisfied. Perhaps surprisingly, we will show next that for random neural network feature maps, it is indeed sufficient that $\boldsymbol{x}$ is nonatomic.[4] Especially, our lower bound in Corollary 5.4.4 thus applies to a large class of feedforward neural networks where only the last layer is trained (and initialized to zero, such that gradient descent converges to the Moore-Penrose pseudoinverse).

Theorem 5.5.6 (Random neural networks)**.** *Let* $d, p, L \geq 1$*, let* $\sigma : \mathbb{R} \to \mathbb{R}$ *be analytic and let the layer sizes be* $d_0 = d$*,* $d_1, \ldots, d_{L-1} \geq 1$ *and* $d_L = p$*. Let* $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d_{l+1} \times d_l}$ *for* $l \in \{0, \ldots, L-1\}$ *be random variables and consider the two cases where*

(a) σ *is not a polynomial with less than* p *nonzero coefficients,* $\boldsymbol{\theta} \coloneqq (\boldsymbol{W}^{(0)}, \ldots, \boldsymbol{W}^{(L-1)})$ *and the random feature map* $\phi_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^p$ *is recursively defined by*

$$\phi(\boldsymbol{x}^{(0)}) \coloneqq \boldsymbol{x}^{(L)}, \quad \boldsymbol{x}^{(l+1)} \coloneqq \sigma(\boldsymbol{W}^{(l)} \boldsymbol{x}^{(l)}) \ .$$

(b) σ *is not a polynomial of degree* $< p - 1$*,* $\boldsymbol{\theta} \coloneqq (\boldsymbol{W}^{(0)}, \ldots, \boldsymbol{W}^{(L-1)}, \boldsymbol{b}^{(0)}, \ldots, \boldsymbol{b}^{(L-1)})$ *with random variables* $\boldsymbol{b}^{(l)} \in \mathbb{R}^{d_{l+1}}$ *for* $l \in \{0, \ldots, L-1\}$*, and the random feature map* $\phi_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^p$ *is recursively defined by*

$$\phi(\boldsymbol{x}^{(0)}) \coloneqq \boldsymbol{x}^{(L)}, \quad \boldsymbol{x}^{(l+1)} \coloneqq \sigma(\boldsymbol{W}^{(l)} \boldsymbol{x}^{(l)} + \boldsymbol{b}^{(l)}) \ .$$

[4]This appears to be a convenient consequence of randomizing the feature map: For each fixed feature map $\phi_{\widetilde{\boldsymbol{\theta}}}$, there may be an exceptional set $E_{\widetilde{\boldsymbol{\theta}}}$ of nonatomic input distributions P_X for which FRK(p) is not satisfied. However, Theorem 5.5.6 shows that for each nonatomic input distribution P_X, the set $\{\widetilde{\boldsymbol{\theta}} \mid P_X \in E_{\widetilde{\boldsymbol{\theta}}}\}$ is a Lebesgue null set. For (deterministic) feature maps where it is not possible to prove FRK(p) for all nonatomic P_X, there is a trick that works in certain cases: If $\boldsymbol{x}$ lives on a submanifold (e.g. a sphere), we might be able to write $\boldsymbol{x} = \psi(\boldsymbol{v})$, where $\boldsymbol{v}$ has a Lebesgue density and ψ is analytic (e.g. the stereographic projection for the sphere). Then, we can apply our theory to the analytic feature map $\phi_{\boldsymbol{\theta}} \circ \psi$.

In both cases, if $\boldsymbol{\theta}$ has a Lebesgue density and $\boldsymbol{x}$ is nonatomic, then (FRK) holds for all n and almost surely over $\boldsymbol{\theta}$.

The assumptions of Theorem 5.5.6 on $\boldsymbol{\theta}$ are satisfied by the classical initialization methods of He et al. (2015) and Glorot and Bengio (2010). Possible choices of σ are presented in Table 5.1. A statement similar to Theorem 5.5.6 has been proven in Lemma 4.4 in Nguyen and Hein (2017). However, their statement only applies to networks with bias and to a more restricted class of activation functions: For example, the activation functions RBF, GELU, SiLU/Swish, Mish, sin and cos are not covered by their assumptions. In Appendix 5.E, we explain that the proofs of many other theorems in the literature similar to Theorem 5.5.6 for single-layer networks are incorrect.

Table 5.1: Some examples of analytic activation functions that are not polynomials. The CDF of $\mathcal{N}(0,1)$ is denoted by Φ. Other non-polynomial analytic activation functions are sin, cos or erf.

Activation function	Equation
Sigmoid	$\operatorname{sigmoid}(x) = 1/(1+e^{-x})$
Hyperbolic Tangent	$\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$
Softplus	$\operatorname{softplus}(x) = \log(1+e^x)$
RBF	$\operatorname{RBF}(x) = \exp(-\beta x^2)$
GELU (Hendrycks and Gimpel, 2016)	$\operatorname{GELU}(x) = x\Phi(x)$
SiLU (Elfwing et al., 2018)	$\operatorname{SiLU}(x) = x \operatorname{sigmoid}(x)$
Swish (Ramachandran et al., 2017)	$\operatorname{Swish}(x) = x \operatorname{sigmoid}(\beta x)$
Mish (Misra, 2019)	$\operatorname{Mish}(x) = x \tanh(\operatorname{softplus}(x))$

Under the assumptions of Theorem 5.5.6, if $\|\boldsymbol{x}\|_2$ is bounded, (MOM) holds for all $\boldsymbol{\theta}$. This follows since any analytic function ϕ is also continuous, and continuous functions preserve boundedness. However, the activation functions from Table 5.1 even satisfy $|\sigma(x)| \leq a|x| + b$ for some $a, b \geq 0$ and all $x \in \mathbb{R}$. In this case, it is not hard to see that (MOM) already holds for all $\boldsymbol{\theta}$ if $\mathbb{E}\|\boldsymbol{x}\|_2^2 < \infty$.

Theorem 5.5.6 does not hold for ReLU, ELU (Clevert et al., 2015), SELU (Klambauer et al., 2017) or other activation functions with a perfectly linear part. To see this, observe that with nonzero probability, all weights and inputs are positive. In this case, the feature map acts as a linear map from $\mathbb{R}^d$ to $\mathbb{R}^p$, and if $d < p = n$, the output matrix $\phi(\boldsymbol{X})$ cannot be invertible.

In Appendix 5.I, we show that (FRK) is satisfied for random Fourier features if $\boldsymbol{x}$ is nonatomic and the frequency distribution (i.e. the Fourier transform of the shift-invariant kernel) has a Lebesgue density.

5.6 Quality of the Lower Bound

In this section, we discuss how sharp the lower bound from Theorem 5.4.3 is. To this end, we assume that $\operatorname{Var}(y|\boldsymbol{z}) = \sigma^2$ almost surely over $\boldsymbol{z}$.

In their Lemma 3, Hastie et al. (2022) consider the case where $\boldsymbol{z}$ has i.i.d. entries with zero mean, unit variance, and finite fourth moment. They then use the Marchenko-Pastur law to show in the limit $p, n \to \infty, p/n \to \gamma > 1$ that $\mathcal{E}_{\text{Noise}} \to \sigma^2 \frac{1}{\gamma - 1}$. In this limit, our lower bound shows

$$\mathcal{E}_{\text{Noise}} \geq \sigma^2 \frac{n}{p+1-n} = \sigma^2 \frac{1}{p/n + 1/n - 1} \to \sigma^2 \frac{1}{\gamma - 1} ,$$

hence, in this sense, our overparameterized bound is asymptotically sharp. An analogous argument shows that the underparameterized bound is also asymptotically sharp. To better understand to which extent our lower bound is non-asymptotically sharp in the over- and underparameterized regimes, we explicitly compute $\mathcal{E}_{\text{Noise}} = \sigma^2 \mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+)$ (cf. Theorem 5.4.3) for some distributions P_Z:

Theorem 5.6.1. *Let $P_Z = \mathcal{U}(\mathbb{S}^{p-1})$. Then, P_Z satisfies the assumptions (MOM), (COV), and (FRK) for all n with $\boldsymbol{\Sigma} = \frac{1}{p}\boldsymbol{I}_p$. Moreover, for $n \geq p = 1$ or $p \geq n \geq 1$, we can compute*

$$\mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+) = \begin{cases} \frac{1}{n} & \text{if } n \geq p = 1, \\ \frac{1}{p} & \text{if } p \geq n = 1, \\ \infty & \text{if } 2 \leq n \leq p \leq n+1, \\ \frac{n}{p-1-n} \cdot \frac{p-2}{p} & \text{if } 2 \leq n \leq n+2 \leq p. \end{cases}$$

The formulas in the next theorem have already been computed by Breiman and Freedman (1983) for $p \leq n-2$ and by Belkin et al. (2020) for general p. Our alternative proof circumvents a technical problem in their proof for $p \in \{n-1, n, n+1\}$, cf. Appendix 5.J.

Theorem 5.6.2. *Let $P_Z = \mathcal{N}(0, \boldsymbol{I}_p)$. Then, P_Z satisfies the assumptions (MOM), (COV), and (FRK) for all n with $\boldsymbol{\Sigma} = \boldsymbol{I}_p$. Moreover, for $n, p \geq 1$,*

$$\mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+) = \begin{cases} \frac{n}{p-1-n} & \text{if } p \geq n+2, \\ \infty & \text{if } p \in \{n-1, n, n+1\}, \\ \frac{p}{n-1-p} & \text{if } p \leq n-2. \end{cases}$$

For $P_Z = \mathcal{N}(0, \boldsymbol{I}_p)$ and the lower bound from Theorem 5.4.3, the formulas for the under- and overparameterized cases can be obtained from each other by switching the roles of n and p. Numerical data strongly suggests that this symmetry does *not* hold exactly for $P_Z = \mathcal{U}(\mathbb{S}^{p-1})$.

For $p \geq n+2$, we can relate our lower bound (Theorem 5.4.3), the result for the sphere (Theorem 5.6.1) and the result for the Gaussian distribution (Theorem 5.6.2) as follows:

$$\frac{n}{p+1-n} = \frac{n}{p-1-n} \cdot \frac{(p+1-n)-2}{p+1-n} \leq \frac{n}{p-1-n} \cdot \frac{p-2}{p} < \frac{n}{p-1-n} .$$

These values are also shown in Figure 5.1 together with empirical values of NN feature maps specifically optimized to minimize $\mathcal{E}_{\text{Noise}}$. Since $\boldsymbol{\Sigma}$ affects $\mathcal{E}_{\text{Noise}}$ only for $p > n$ (cf. Remark 5.G.1), it is not surprising that the feature map optimized for $n = 60 > 30 = p$ performs badly in the overparameterized regime. The results in Figure 5.1 support the hypothesis that among all P_Z satisfying (MOM), (COV) and (FRK), $P_Z = \mathcal{U}(\mathbb{S}^{p-1})$ minimizes $\mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+)$. The plausibility of this hypothesis is further discussed in

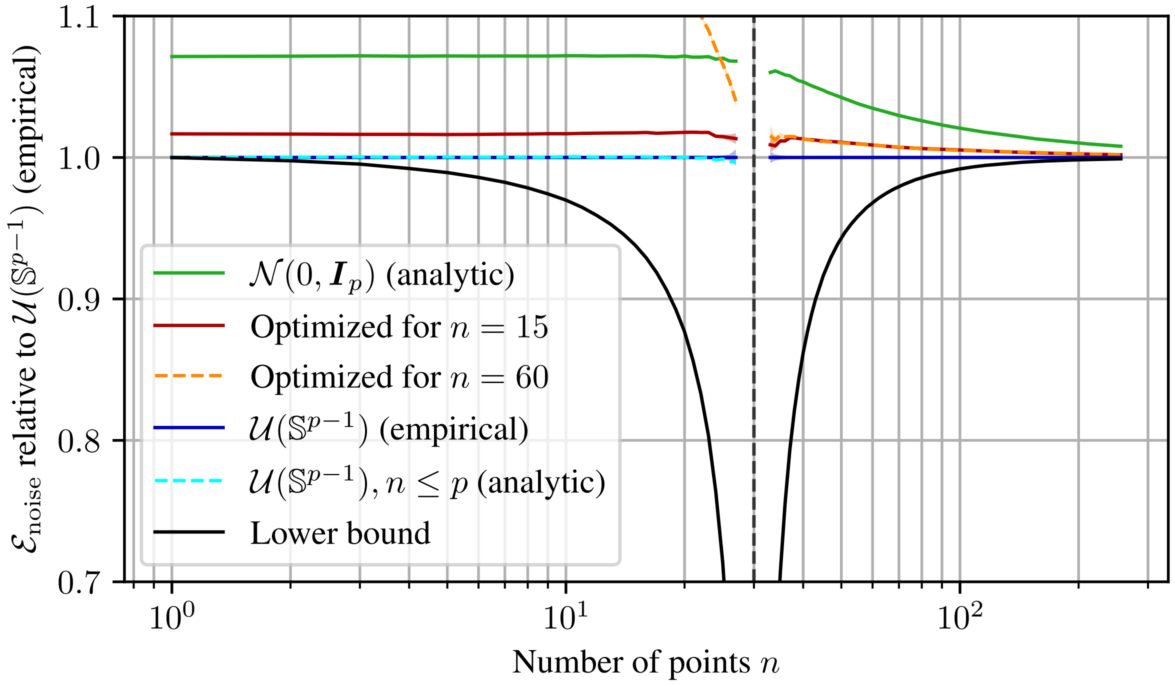


Figure 5.1: Various estimates and bounds for $\mathcal{E}_{\text{Noise}}$ relative to $\mathcal{E}_{\text{Noise}}$ for $P_Z = \mathcal{U}(\mathbb{S}^{p-1})$, using $\text{Var}(y|\boldsymbol{z}) = 1$ and $p = 30$. The optimized curves correspond to multi-layer NN feature maps whose parameters were trained to minimize $\mathcal{E}_{\text{Noise}}$ for $n = 15$ or $n = 60$. More experiments and details on the setup can be found in Appendix 5.C. We do not plot estimates for $n \in \{28, \ldots, 32\}$ since they have high estimation variances.

Remark 5.G.3. We can prove the hypothesis for $n = 1$ or $p = 1$ since in this case, the results from Theorem 5.6.1 are equal to the lower bound from Theorem 5.4.3.

When using a continuous feature map $\phi : \mathbb{R}^d \to \mathbb{R}^p$ with $d \leq p - 2$, it seems to be unclear at first whether the low $\mathcal{E}_{\text{Noise}}$ of $P_Z = \mathcal{U}(\mathbb{S}^{p-1})$ can even be achieved. We show in Proposition 5.J.2 that this is possible using space-filling curve feature maps.

The results in this section and in Appendix 5.C show that while our lower bound presumably does not fully capture the height of the interpolation peak at $p \approx n$, it is quite sharp in the practically relevant regimes $p \gg n$ and $p \ll n$ irrespective of d.

Acknowledgments The author would like to thank Ingo Steinwart for proofreading most of this paper and for providing helpful comments. Funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2075 - 390740016. The author thanks the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for support.

Appendix

5.A Overview

The appendix is structured as follows: In Appendix 5.B, we provide an overview of some matrix identities used throughout the paper. We provide additional numerical experiments for various (random) feature maps in Appendix 5.C. The counterexample given in Appendix 5.D shows that the assumption (FRK) cannot be omitted from Theorem 5.4.3. In Appendix 5.E, we provide an overview of full-rank theorems for random neural networks in the literature and explain why their proofs are incorrect. We then prove our main results in Appendix 5.F before discussing consequences in Appendix 5.G. In Appendix 5.H, we give proofs for the theorems and propositions from Section 5.5. As an addition, we prove (FRK) for random Fourier features in Appendix 5.I. Proofs for the statements from Section 5.6 are given in Appendix 5.J. In Appendix 5.K, we compare our results to some results for ridgeless kernel regression. Finally, in Appendix 5.L, we compare our result to the one by Muthukumar et al. (2020).

Whenever we prove theorems or propositions from the main paper (like Theorem 5.4.3) in the Appendix, we repeat their statement before the proof for convenience. In contrast, new theorems or propositions are numbered according to the section they are stated in, e.g. Proposition 5.I.1.

5.B Matrix Algebra

In the following, we will present some facts about matrices that are relevant to this paper. For a general reference, we refer to textbooks on the subject (e.g. Golub and Van Loan, 1989; Bhatia, 2013).

Let $n, p \geq 1$ and let $k \coloneqq \min\{n, p\}$. The singular value decomposition (SVD) of a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times p}$ is a decomposition $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$ into orthogonal matrices $\boldsymbol{U} \in \mathbb{R}^{n \times k}, \boldsymbol{V} \in \mathbb{R}^{p \times k}$ with $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{I}_k$ and a diagonal matrix $\boldsymbol{D} \in \mathbb{R}^{k \times k}$ with non-negative diagonal elements $s_1(\boldsymbol{A}) \geq \ldots \geq s_k(\boldsymbol{A})$ called *singular values.*

For a given symmetric square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ with eigenvalues $\lambda_1(\boldsymbol{A}) \geq \ldots \geq \lambda_n(\boldsymbol{A})$, the trace satisfies

$$\operatorname{tr}(\boldsymbol{A}) = \sum_{i=1}^n A_{ii} = \sum_{i=1}^n \lambda_i \ .$$

The trace is linear and invariant under cyclical permutations. We use this multiple times in arguments of the following type: If $\boldsymbol{v} \in \mathbb{R}^p$ and $\boldsymbol{A} \in \mathbb{R}^{p \times p}$ are stochastically independent (e.g. because $\boldsymbol{A}$ is constant), we can write

$$\mathbb{E}_{\boldsymbol{v}}\boldsymbol{v}^\top \boldsymbol{A}\boldsymbol{v} = \mathbb{E}_{\boldsymbol{v}} \operatorname{tr}(\boldsymbol{v}^\top \boldsymbol{A}\boldsymbol{v}) = \mathbb{E}_{\boldsymbol{v}} \operatorname{tr}(\boldsymbol{A}\boldsymbol{v}\boldsymbol{v}^\top) = \operatorname{tr}(\boldsymbol{A}\mathbb{E}_{\boldsymbol{v}}\boldsymbol{v}\boldsymbol{v}^\top) \ .$$

Moreover, if $\boldsymbol{A} \succ 0$, then for all $i \in [n]$, $\lambda_i(\boldsymbol{A}) > 0$ and we have

$$\lambda_{n+1-i}(\boldsymbol{A}^{-1}) = \frac{1}{\lambda_i(\boldsymbol{A})} \ .$$

For a positive definite matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{p\times p}$, the SVD and the eigendecomposition coincide as $\boldsymbol{\Sigma} = \boldsymbol{U}\operatorname{diag}(\lambda_1(\boldsymbol{\Sigma}), \dots, \lambda_p(\boldsymbol{\Sigma}))\boldsymbol{U}^\top$ and we can define the inverse square root as

$$\boldsymbol{\Sigma}^{-1/2} \coloneqq \boldsymbol{U}\operatorname{diag}(\lambda_1(\boldsymbol{\Sigma})^{-1/2}, \dots, \lambda_p(\boldsymbol{\Sigma})^{-1/2})\boldsymbol{U}^\top \succ 0 \ ,$$

which is the unique s.p.d. matrix satisfying $(\boldsymbol{\Sigma}^{-1/2})^2 = \boldsymbol{\Sigma}^{-1}$.

By the Courant-Fischer-Weyl theorem, two symmetric matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times n}$ with $\boldsymbol{A} \preceq \boldsymbol{B}$ satisfy

$$\lambda_i(\boldsymbol{A}) = \max_{\substack{\mathcal{V}\subseteq\mathbb{R}^d \text{ subspace} \\ \dim\mathcal{V}=i}} \min_{\substack{\boldsymbol{v}\in\mathcal{V} \\ \|\boldsymbol{v}\|_2=1}} \boldsymbol{v}^\top\boldsymbol{A}\boldsymbol{v} \leq \max_{\substack{\mathcal{V}\subseteq\mathbb{R}^d \text{ subspace} \\ \dim\mathcal{V}=i}} \min_{\substack{\boldsymbol{v}\in\mathcal{V} \\ \|\boldsymbol{v}\|_2=1}} \boldsymbol{v}^\top\boldsymbol{B}\boldsymbol{v} = \lambda_i(\boldsymbol{B}) \ .$$

Let $\boldsymbol{A} \in \mathbb{R}^{n\times p}$. The Moore-Penrose pseudoinverse $\boldsymbol{A}^+$ of $\boldsymbol{A}$ satisfies the following relations (see e.g. Section 1.1.1 in Wang et al., 2018):

- Suppose $\boldsymbol{A}$ has the SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$, where $\boldsymbol{D} = \operatorname{diag}(s_1, \dots, s_k)$, $k \coloneqq \min\{n, p\}$. Using the convention $1/0 \coloneqq 0$, we can write $\boldsymbol{A}^+ = \boldsymbol{V}\boldsymbol{D}^+\boldsymbol{U}^\top$, where $\boldsymbol{D}^+ \coloneqq \operatorname{diag}(1/s_1, \dots, 1/s_k)$.
- $\boldsymbol{A}^+ = (\boldsymbol{A}^\top\boldsymbol{A})^+\boldsymbol{A}^\top = \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^+$.
- If $\boldsymbol{A}$ is invertible, then $\boldsymbol{A}^+ = \boldsymbol{A}^{-1}$.
- $\boldsymbol{A}^+(\boldsymbol{A}^+)^\top = (\boldsymbol{A}^\top\boldsymbol{A})^+$.

We will also use a basic fact on the Schur complement that, for example, is outlined in Appendix A.5.5 in Boyd and Vandenberghe (2004): If

$$0 \prec \boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix} \in \mathbb{R}^{(m_1+m_2)\times(m_1+m_2)} \ ,$$

then $\boldsymbol{A}_{22} \succ 0$ and $\boldsymbol{A}_{11} - \boldsymbol{A}_{12}\boldsymbol{A}_{22}^{-1}\boldsymbol{A}_{21} \succ 0$ and the top-left $m_1 \times m_1$ block of $\boldsymbol{A}^{-1}$ is given by $(\boldsymbol{A}_{11} - \boldsymbol{A}_{12}\boldsymbol{A}_{22}^{-1}\boldsymbol{A}_{21})^{-1}$.

5.C Experiments

In the following, we experimentally investigate $\mathcal{E}_{\text{Noise}}$ for different (random) feature maps. We will first give an overview of the plots and then explain the details of how they were generated and some implications. More details can be found in the provided code. All empirical curves show one estimated standard deviation of the mean estimator as a shaded area around the estimated mean, but this standard deviation is sometimes too small to be visible. We assume $\operatorname{Var}(y|\boldsymbol{z}) = 1$ almost surely over $\boldsymbol{z}$ such that (I) in Theorem 5.4.3 holds with equality with $\sigma^2 = 1$.

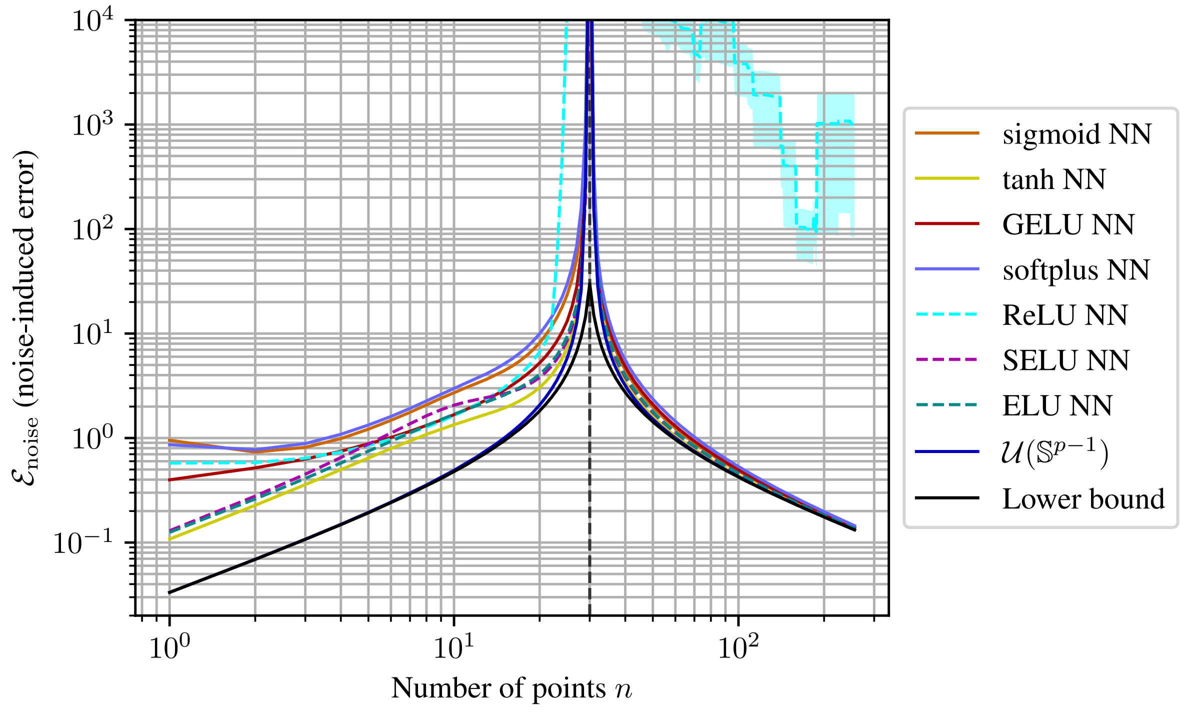


Figure 5.C.1: Estimated $\mathcal{E}_{\text{Noise}}$ for random neural network feature maps (cf. Theorem 5.5.6) with different activation functions and $d_0 = d = 10, d_1 = d_2 = 256, d_3 = p = 30$. We include the results from $P_Z = \mathcal{U}(\mathbb{S}^{d-1})$ as comparison, cf. Section 5.6.

Figure 5.C.1 shows $\mathcal{E}_{\text{Noise}}$ for random three-layer neural network feature maps with $p = 30$, different activation functions and varying n. Note that all neural networks produce higher $\mathcal{E}_{\text{Noise}}$ than $P_Z = \mathcal{U}(\mathbb{S}^{p-1})$. The effect of non-isotropic covariance matrices in the overparameterized regime can be clearly seen when comparing Figure 5.C.1 to Figure 5.C.2, where features have been whitened separately for each set of random parameters $\boldsymbol{\theta}$, cf. Remark 5.G.1. Figure 5.C.3 then shows $\mathcal{E}_{\text{Noise}}$ for $n = 30$ and varying p.

Note that double descent is usually plotted as a function of the "model complexity" p as in Belkin et al. (2019), but varying p is only possible when we have a (random) feature map $\phi_{\boldsymbol{\theta}}^{(p)} : \mathbb{R}^d \to \mathbb{R}^p$ for each value of p. For the following NTK and polynomial kernels, there is no canonical way to define such a sequence of feature maps. For this reason, we will plot their results only with varying n. Double descent as a function of the number of samples n has for example been pointed out by Nakkiran et al. (2021a) and Nakkiran (2019).

Figure 5.C.4 shows $\mathcal{E}_{\text{Noise}}$ for various random finite-width Neural Tangent Kernels (NTKs), cf. Jacot et al. (2018). These results mostly exhibit larger $\mathcal{E}_{\text{Noise}}$ than the random NN feature maps from Figure 5.C.1, perhaps because of correlations parameter gradients in different layers. However, this comparison is not really fair since the NTK feature map uses a much smaller underlying neural network and the input dimension d is smaller.

Figure 5.C.5 and Figure 5.C.6 show $\mathcal{E}_{\text{Noise}}$ for two variants of random Fourier features for two different scalings of the random parameters. Figure 5.C.6 shows that for random Gaussian parameters with large variance (corresponding to an approximated narrow Gaussian kernel), the values of $\mathcal{E}_{\text{Noise}}$ for random Fourier features are very close to the

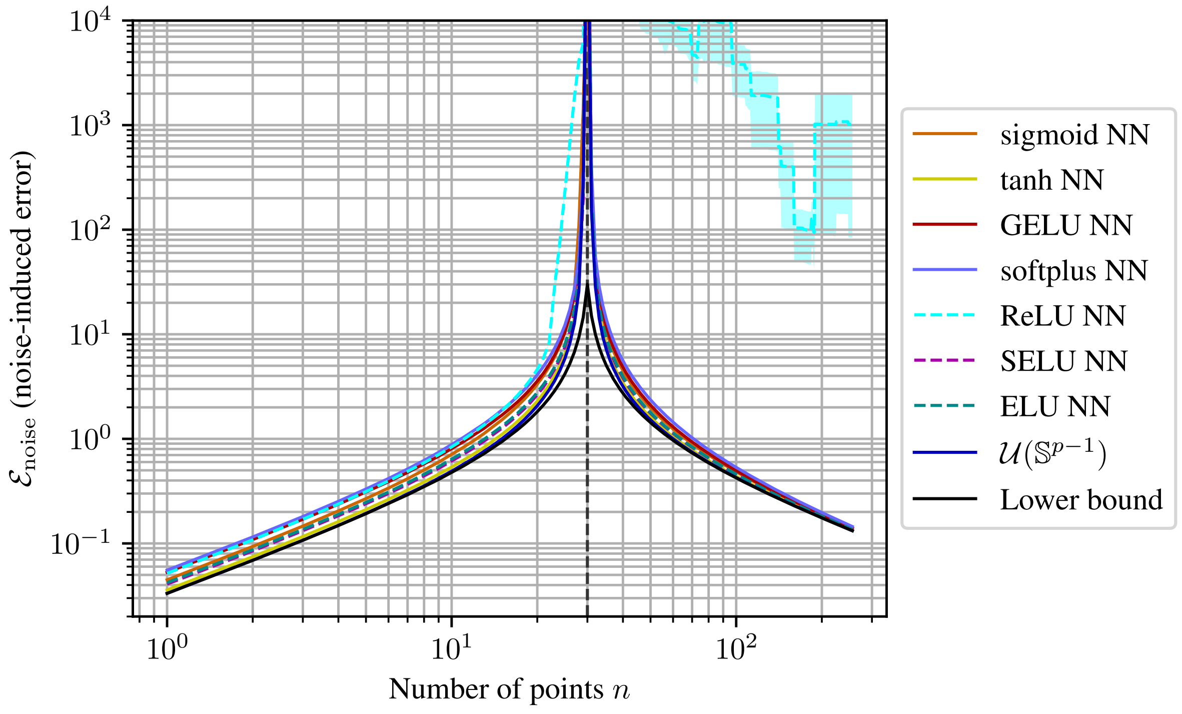


Figure 5.C.2: As in Figure 5.C.1 but with whitened features, i.e. using $\mathbb{E}\operatorname{tr}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1})$ in the overparameterized case $n \le p$, cf. Theorem 5.4.3 and Remark 5.G.1.

values for $P_Z = \mathcal{U}(\mathbb{S}^{p-1})$. We decided to plot these values relative to each other as in Figure 5.1 since the curves would overlap in a normal plot like Figure 5.C.5. Note that the the version of random Fourier features with sin and cos features automatically yields constant $\|\boldsymbol{z}\|_2$ like for $P_Z = \mathcal{U}(\mathbb{S}^{p-1})$.

Finally, Figure 5.C.7 shows that linear regression with the polynomial kernel is quite sensitive to label noise. We use $p = 35$ for the polynomial kernel since there are no particularly interesting polynomial kernels with $p = 30$.

Neural Network feature maps For Figures 5.C.1, 5.C.2 and 5.C.3, we use random neural network feature maps without biases as in Theorem 5.5.6 with $d_0 = d = 10, d_1 = d_2 = 256$ and $d_3 = p$. As the input distribution P_X, we use $\mathcal{N}(0, \boldsymbol{I}_d)$. We initialize the NN weights independently as $W_{ij}^{(l)} \sim \mathcal{N}(0, 1/V_l)$, where

$$V_l := \begin{cases} d_0 & \text{if } l = 0, \\ d_l \operatorname{Var}_{u\sim\mathcal{N}(0,1)}(\sigma(u)) & \text{if } l \ge 1. \end{cases}$$

Here, $\operatorname{Var}_{u\sim\mathcal{N}(0,1)}(\sigma(u))$ is approximated once by using 10^4 samples for u. This initialization is similar (and for the ReLU activation essentially equivalent) to the initialization method by He et al. (2015). The initialization variances are chosen such that for fixed input $\boldsymbol{x}$ with $\|\boldsymbol{x}\|_2 \approx 1$, the pre-activations in each layer are approximately $\mathcal{N}(0, 1)$-distributed.

NTK feature maps For Figure 5.C.4, we use an NTK parameterization similar to the original one proposed by Jacot et al. (2018), but again with activation-dependent scaling.

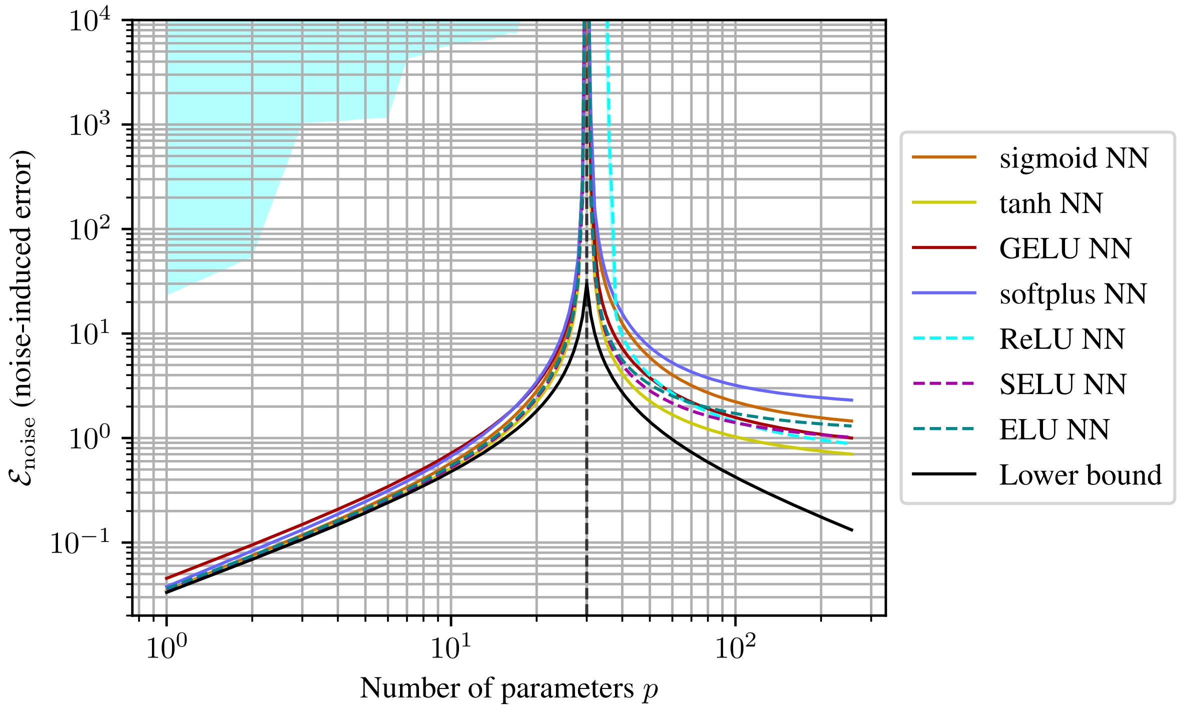


Figure 5.C.3: As in Figure 5.C.1 but with $n = 30$ fixed and varying $d_3 = p$.

Our neural network is given by[5]

$$\tilde{\phi}_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^1, \boldsymbol{x} \mapsto \frac{1}{\sqrt{V_1}} \boldsymbol{W}^{(1)} \sigma\left(\frac{1}{\sqrt{V_0}} \boldsymbol{W}^{(0)}\right)$$

with $d_0 = d = 4, d_1 = 6, d_2 = 1$ and $\boldsymbol{W}^{(l)}_{ij} \sim \mathcal{N}(0,1)$ i.i.d. Our input distribution is again $P_X = \mathcal{N}(0, \boldsymbol{I}_d)$. The NTK feature map is then defined as

$$\phi_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^p, \boldsymbol{x} \mapsto \frac{\partial}{\partial \boldsymbol{\theta}} \tilde{\phi}_{\boldsymbol{\theta}}(\boldsymbol{x}) \ ,$$

where $p = 6 \cdot 4 + 1 \cdot 6 = 30$ is the number of parameters in $\boldsymbol{\theta}$. Note that moving the variances V_l outside of $\boldsymbol{W}^{(l)}$ does not affect the forward pass but only the backward pass (i.e. the derivatives).

While we have not theoretically established the properties (FRK) and (COV) for such feature maps, we can do this experimentally for analytic activation functions like sigmoid, tanh, softplus and GELU: Since the random NTK feature map is a derivative of the analytic random NN feature map, it is also analytic. By Proposition 5.5.5, if (FRK) does not hold, then for every fixed $\widetilde{\boldsymbol{\theta}}$, the range of $\phi_{\widetilde{\boldsymbol{\theta}}}$ must be contained in a proper linear subspace of $\mathbb{R}^p$, and therefore $\boldsymbol{Z} = \phi_{\boldsymbol{\theta}}(\boldsymbol{X})$ never has full rank for $n \geq p$. In this case,

[5]For random NN feature maps as in Theorem 5.5.6, one can interpret the linear regression as being an extra layer on top of the neural network, and therefore the last layer of the feature map should contain an activation function. For NTK feature maps, one can instead interpret the linear regression as performing a "linearized" training of the whole NN, and the whole NN usually does not contain an activation function in the last layer.

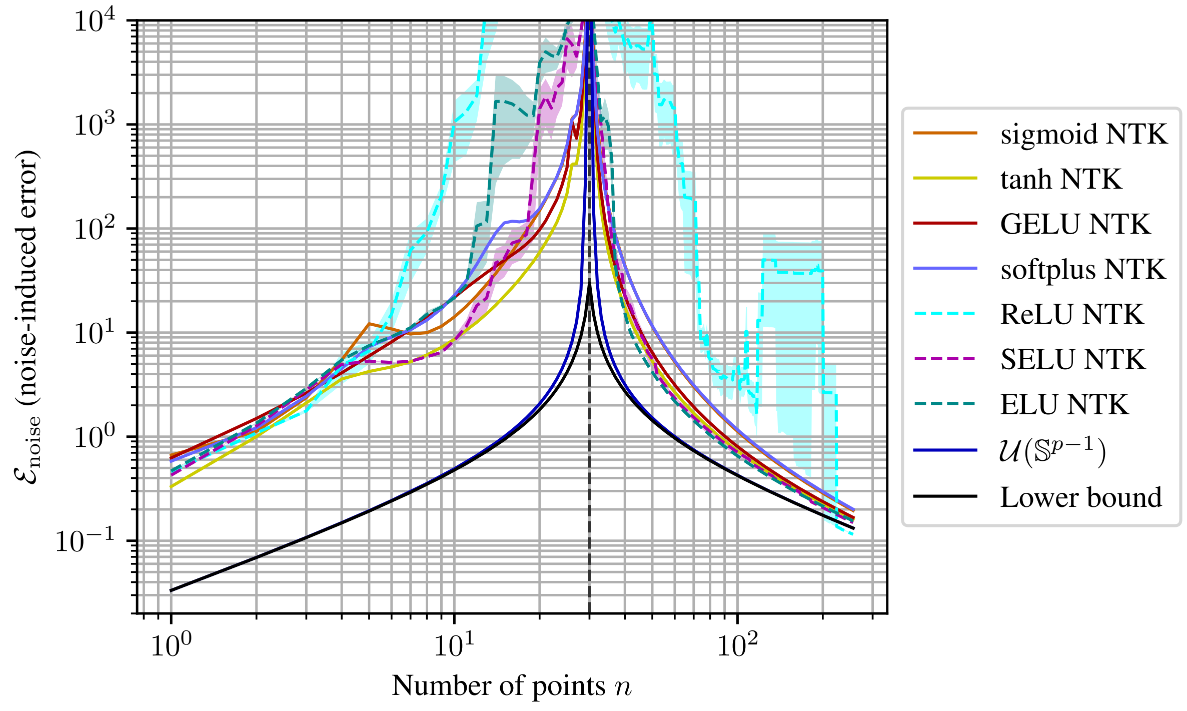


Figure 5.C.4: Estimated $\mathcal{E}_{\text{Noise}}$ for Neural Tangent Kernel (NTK) feature maps given by various random neural networks (cf. Theorem 5.5.6) with $d_0 = d = 4, d_1 = 6, d_3 = 1$, resulting in $p = 4 \cdot 6 + 6 \cdot 1 = 30$. We include the empirical results from $P_Z = \mathcal{U}(\mathbb{S}^{d-1})$ as comparison, cf. Section 5.6.

the singular value $s_p(\boldsymbol{Z})$ would be zero, and even accounting for numerical errors, the "inverse condition number" $\frac{s_p(\boldsymbol{Z})}{s_1(\boldsymbol{Z})}$ should at most be of the order of 64-bit float machine precision, i.e. around 10^{-16}. However, among 10^4 samples of $\boldsymbol{Z}$ for $n := 90 \geq 30 = p$, the maximum observed "inverse condition number" was greater than 10^{-3} for all of the activation functions sigmoid, tanh, softplus and GELU.[6] Hence, by Proposition 5.5.2 and Proposition 5.5.5, we can confidently conclude that (COV) and (FRK) hold for all n almost surely over $\boldsymbol{\theta}$ for this network size and these activation functions.

Estimation of $\mathcal{E}_{\text{Noise}}$ To estimate $\mathcal{E}_{\text{Noise}}$, we proceed as follows: Recall from Section 5.3 that ridgeless regression is the limit of ridge regression for $\lambda \searrow 0$. We use a small regularization of $\lambda = 10^{-12}$ to improve numerical stability. Also for numerical stability, we use a singular value decomposition (SVD) $\boldsymbol{Z} = \boldsymbol{U} \operatorname{diag}(s_1, \ldots, s_k)\boldsymbol{V}^\top$ with $k := \min\{n, p\}$ as in Appendix 5.B. The regularized approximation of $\boldsymbol{Z}^+$ is then

$$\boldsymbol{Z}^+ \approx (\boldsymbol{Z}^\top \boldsymbol{Z} + \lambda \boldsymbol{I}_p)^{-1}\boldsymbol{Z}^\top = \boldsymbol{V} \operatorname{diag}\left(\frac{s_1}{s_1^2+\lambda}, \ldots, \frac{s_k}{s_k^2+\lambda}\right) \boldsymbol{U}^\top . \tag{5.4}$$

We can then estimate

$$\operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+) = \operatorname{tr}(\boldsymbol{Z}^+(\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma}) \approx \operatorname{tr}\left(\boldsymbol{V} \operatorname{diag}\left(\frac{s_1^2}{(s_1^2+\lambda)^2}, \ldots, \frac{s_k^2}{(s_k^2+\lambda)^2}\right) \boldsymbol{V}^\top \boldsymbol{\Sigma}\right) . \tag{5.5}$$

[6]We use $n > p$ since this usually improves the "inverse condition number" of $\boldsymbol{Z}$.

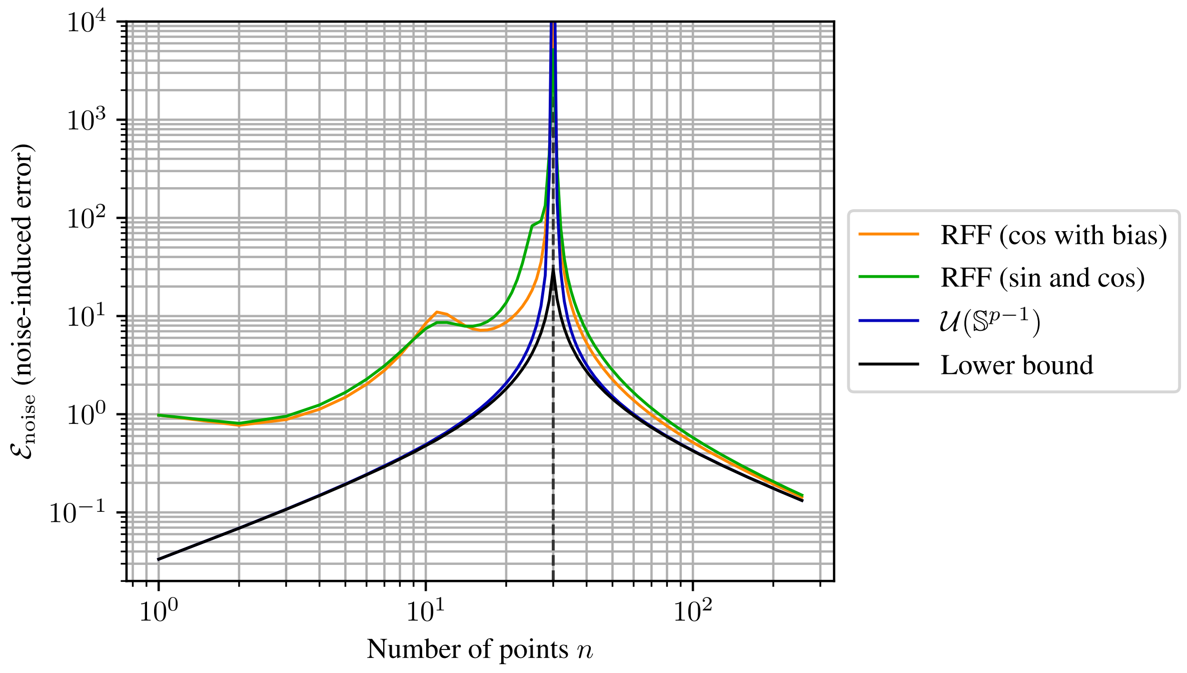


Figure 5.C.5: Estimated $\mathcal{E}_{\text{Noise}}$ for the two versions of random Fourier features described in Appendix 5.I. We use $d = 10$, $P_X = \mathcal{N}(0, \boldsymbol{I}_d)$, $p = 30$ and the weight vector distribution $P_k = \mathcal{N}(0, \frac{1}{p}\boldsymbol{I}_d)$ (cf. Appendix 5.I).

We can then obtain $m = 10^4$ (non-ReLU NNs, polynomial kernel, high-variance random Fourier features) or $m = 10^5$ (all other empirical results) sampled estimates for $\operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+)$ to obtain a Monte-Carlo estimate of $\mathbb{E}\operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+)$ by repeating the following procedure m times:

(1) Sample a random parameter $\boldsymbol{\theta}$.
(2) Sample random matrices $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ and $\tilde{\boldsymbol{X}} \in \mathbb{R}^{l \times d}$, $l = 10^4$, with i.i.d. P_X-distributed rows.
(3) Compute $\boldsymbol{Z} := \phi_{\boldsymbol{\theta}}(\boldsymbol{X})$ and $\tilde{\boldsymbol{Z}} := \phi_{\boldsymbol{\theta}}(\tilde{\boldsymbol{X}})$.
(4) Compute the estimate $\boldsymbol{\Sigma} := \frac{1}{l}\tilde{\boldsymbol{Z}}^\top \tilde{\boldsymbol{Z}}$.
(5) Compute a regularized estimate of $\operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+)$ using the SVD of $\boldsymbol{Z}$ and Eq. (5.5).

For performance reasons, we make the following modification of steps (2) and (5): Since we perform the computation for all $n \in [256]$, we sample $\boldsymbol{X} \in \mathbb{R}^{256 \times d}$ and then, for all $n \in [256]$, perform the computation for n using the first n rows of $\boldsymbol{Z}$. Hence, the estimates for different n are not independent. In Figure 5.C.3, we perform an analogous optimization for p by taking the first $p \in [256]$ of the $d_3 = 256$ output features in $\boldsymbol{Z}$.

Curious ReLU results The curves for the ReLU NNs and ReLU NTKs in the under-parameterized regime $p \leq n$ may seem odd. The locally almost constant "plateaus" are presumably an artifact of the non-independent estimates for different n as explained in the last paragraph. As discussed in Section 5.5, networks with ReLU, ELU, or SELU activations do not satisfy (FRK). It seems plausible that, since both "halves" of the ReLU function are linear, ReLU networks have a significantly higher chance than SELU or ELU

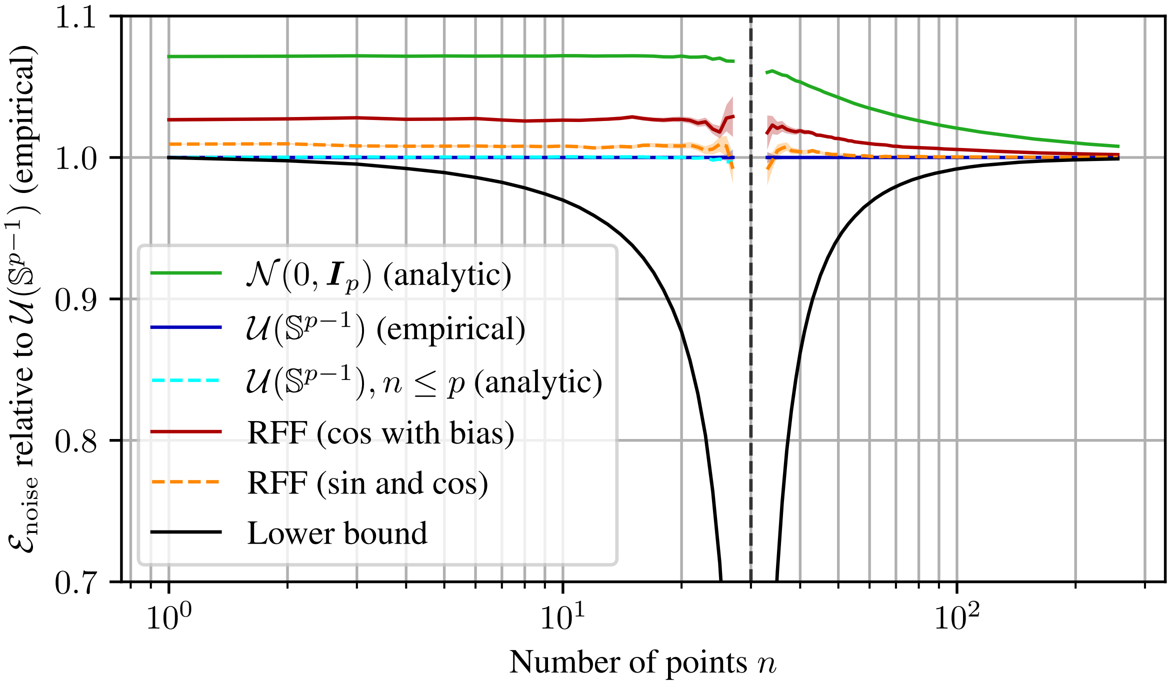


Figure 5.C.6: Estimated $\mathcal{E}_{\text{Noise}}$ for the two versions of random Fourier features described in Appendix 5.I relative to $P_Z = \mathcal{U}(\mathbb{S}^{p-1})$. We use $d = 10$, $P_X = \mathcal{N}(0, \boldsymbol{I}_d)$, $p = 30$ and the weight vector distribution $P_k = \mathcal{N}(0, \boldsymbol{I}_d)$ (cf. Appendix 5.I).

networks to be initialized with "bad" parameters that are likely to generate "degenerate" feature matrices $\boldsymbol{Z}$ that do not have full rank at $n = p$ and only become full rank for some $n > p$. When inspecting the data underlying the plots, the estimate of $\mathcal{E}_{\text{Noise}}$ for ReLU networks in the underparameterized regime seems to be dominated by few outliers. It seems that the distribution of $\operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+)$ for ReLU networks is often so heavy-tailed that computing more Monte Carlo samples does not significantly reduce the estimation uncertainty.

Whitening For computing $\mathbb{E}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1}) = \mathbb{E}((\boldsymbol{Z}\boldsymbol{\Sigma}^{-1}\boldsymbol{Z}^\top)^{-1})$ in the overparameterized case in Figure 5.C.2, we regularize both matrix inversions on the right-hand side as above: For a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{m \times m}$, we use a symmetric eigendecomposition $\boldsymbol{A} = \boldsymbol{U} \operatorname{diag}(s_1, \ldots, s_m)\boldsymbol{U}^\top$ and approximate

$$\boldsymbol{A}^{-1} \approx \boldsymbol{U} \operatorname{diag}\left(\frac{s_1}{s_1^2 + \lambda}, \ldots, \frac{s_m}{s_m^2 + \lambda}\right) \boldsymbol{U}^\top .$$

Optimization For our optimized feature maps in Figure 5.1 with $p = 30$, we use a neural network feature map with $d_0 = d = p = 30, d_1 = d_2 = 256, d_3 = p = 30$ and tanh activation function. We use NTK parameterization and zero-initialized biases, leading to

$$\phi_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma(\boldsymbol{b}^{(2)} + V_2^{-1/2}\boldsymbol{W}^{(2)}\sigma(\boldsymbol{b}^{(1)} + V_1^{-1/2}\boldsymbol{W}^{(1)}\sigma(\boldsymbol{b}^{(0)} + V_0^{-1/2}\boldsymbol{W}^{(0)}\boldsymbol{x})))$$

with independent initialization $W_{ij}^{(l)} \sim \mathcal{N}(0, 1)$, $b_i^{(l)} = 0$. As the input distribution, we use $P_X = \mathcal{N}(0, \boldsymbol{I}_d)$. For given $\boldsymbol{\theta}$, let $\boldsymbol{\Sigma}_{\boldsymbol{\theta}} := \mathbb{E}_{\boldsymbol{x}}\phi_{\boldsymbol{\theta}}(\boldsymbol{x})\phi_{\boldsymbol{\theta}}(\boldsymbol{x})^\top$, i.e. we define the second moment

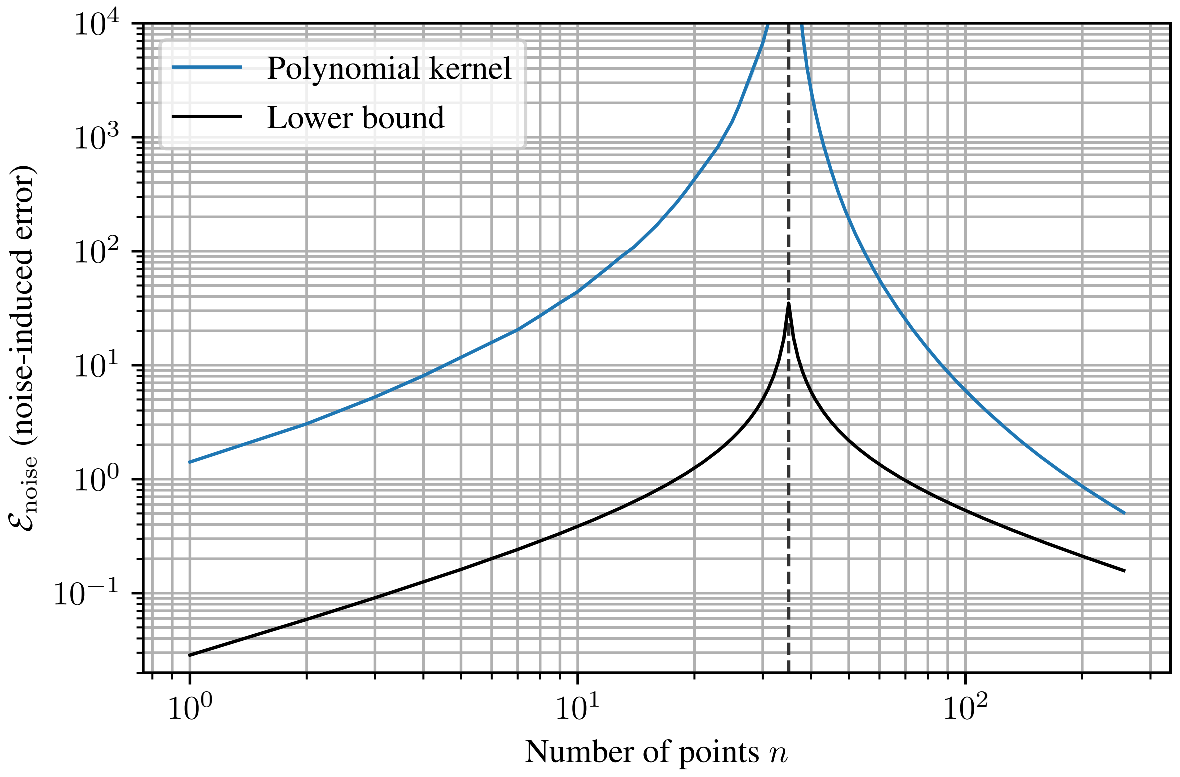


Figure 5.C.7: Estimated $\mathcal{E}_{\text{Noise}}$ for the polynomial kernel (cf. Proposition 5.5.4) with $d = 3$, $m = 4$, $c = 1$, $P_X = \mathcal{N}(0, \boldsymbol{I}_d)$, resulting in $p = \binom{m+d}{m} = \binom{7}{4} = 35$.

matrix $\boldsymbol{\Sigma}$ as depending on $\boldsymbol{\theta}$. We then optimize the loss function

$$L(\boldsymbol{\theta}) := \mathbb{E}_{\boldsymbol{X}} \operatorname{tr}((\phi_{\boldsymbol{\theta}}(\boldsymbol{X})^+)^\top \boldsymbol{\Sigma}_{\boldsymbol{\theta}} \phi_{\boldsymbol{\theta}}(\boldsymbol{X})^+)$$

using AMSGrad (Reddi et al., 2018) with a learning rate that linearly decays from 10^{-3} to 0 over 1000 iterations. To approximate $L(\boldsymbol{\theta})$ in each iteration, we approximate $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ using 1000 Monte Carlo points and we draw 1024 different realizations of $\boldsymbol{X}$ (this can be considered as using batch size 1024). We also use a regularized version as in Eq. (5.4), but we omit the SVD for reasons of differentiability.

5.D A Counterexample

Example 5.D.1. Let $p \geq 2$. Consider the uniform distribution P_Z on an orthonormal basis $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_p\} \subseteq \mathbb{R}^p$. Then, $\boldsymbol{\Sigma} = \frac{1}{p} \sum_{i=1}^p \boldsymbol{e}_i \boldsymbol{e}_i^\top = \frac{1}{p} \boldsymbol{I}_p$ and hence (COV) is satisfied. However, from Proposition 5.5.2 it is easy to see that for any $n \geq 2$, (FRK) is not satisfied. Indeed, if the vector $\boldsymbol{e}_i$ occurs m_i times among the samples $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$, then $\boldsymbol{Z}^\top \boldsymbol{Z} = \operatorname{diag}(m_1, \ldots, m_p)$. Assuming $\operatorname{Var}(y|\boldsymbol{z}) = \sigma^2 := 1$ for all $\boldsymbol{z}$, we then obtain from Theorem 5.4.3 with the convention $\frac{1}{0} := 0$:

$$\mathcal{E}_{\text{Noise}} = \mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+) = \frac{1}{p} \mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}(\boldsymbol{Z}^+ (\boldsymbol{Z}^+)^\top) = \frac{1}{p} \mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{Z}^\top \boldsymbol{Z})^+)$$

$$= \frac{1}{p} \sum_{i=1}^{p} \mathbb{E} \frac{1}{m_i} = \mathbb{E} \frac{1}{m_1} ,$$

where m_1 follows a binomial distribution with parameters n and $\frac{1}{p}$. Using $\mathbb{E}\frac{1}{m_1} \leq \mathbb{E}1 = 1$, it is easy to see that the lower bound is violated for some values of n. For example, Figure 5.D.1 shows that for $p = 30$, the lower bound is violated in a large region, especially in the overparameterized regime.[7] The underlying reason is that the function $x \mapsto \frac{1}{x}$ is convex on $(0, \infty)$, but not on $[0, \infty)$. If $\boldsymbol{Z}$ has singular values s_i, the pseudo-inverse $\boldsymbol{Z}^+$ has singular values $\frac{1}{s_i}$. If $s_i = 0$ is possible, we cannot use a convexity-based argument anymore. ◀

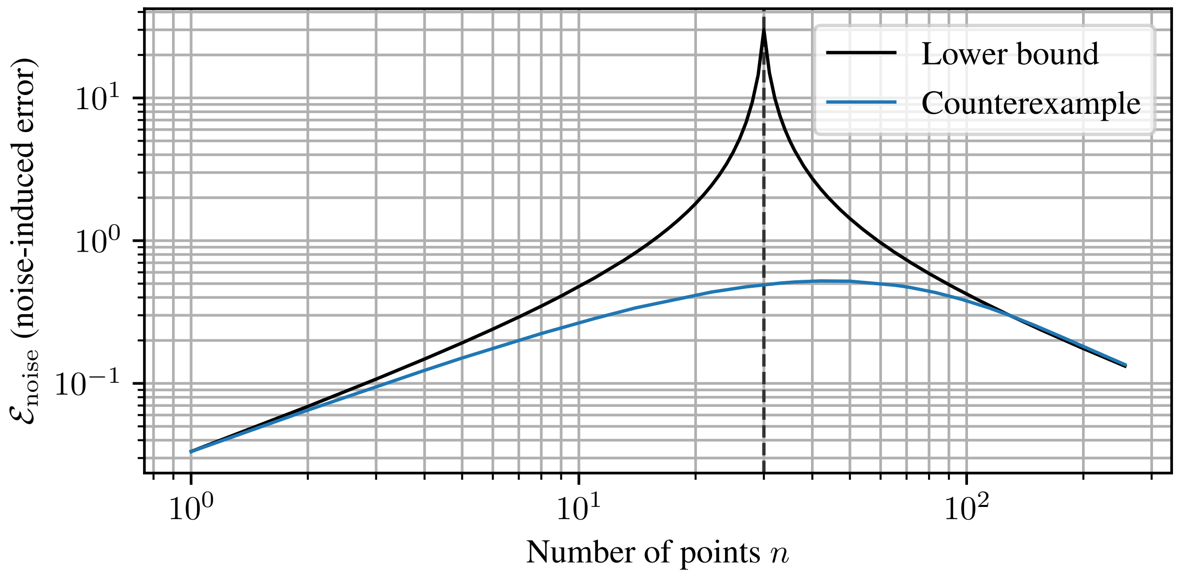


Figure 5.D.1: The counterexample given in Example 5.D.1 for $p = 30$ often has lower $\mathcal{E}_{\text{Noise}}$ than the lower bound from Theorem 5.4.3, but violates its assumption (FRK). For the counterexample, $\mathcal{E}_{\text{Noise}}$ was approximated as explained in Example 5.D.1 using 10^6 Monte Carlo samples for each n. We assume $\text{Var}(y|\boldsymbol{z}) = 1$ almost surely over $\boldsymbol{z}$.

Remark 5.D.2 (Histogram regression)**.** The distribution P_Z in Example 5.D.1 may seem contrived at first, but such a distribution can arise in histogram regression (cf e.g. Chapter 4 in Györfi et al., 2002). For example, suppose that P_X is supported on a domain $\mathcal{D} \subseteq \mathbb{R}^d$ and this domain is partitioned into disjoint sets $\mathcal{A}_1, \ldots, \mathcal{A}_p$. Then, performing histogram regression on this partition is equivalent to performing ridgeless linear regression with the feature map $\phi : \mathbb{R}^d \to \mathbb{R}^p$ with $\phi(\boldsymbol{x}) := \boldsymbol{e}_i$ if $\boldsymbol{x} \in \mathcal{A}_i$. If all partitions are equally likely, i.e. $P_X(\mathcal{A}_i) = 1/p$ for all $i \in \{1, \ldots, p\}$, then P_Z is the uniform distribution on $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_p\}$ as in Example 5.D.1. ◀

[7] For $n = 1$, (FRK) holds and it is easy to see that the lower bound holds exactly in this case.

5.E Full-rank Results for Random Weight Neural Networks

In the literature on neural networks with random weights (Schmidt et al., 1992) and the later virtually identical Extreme Learning Machine (ELM) (Huang et al., 2006), properties similar to (FRK) have been investigated for random neural network feature maps. In the following, we review the relevant approaches known to us and explain why most of them are flawed.

First, Sartori and Antsaklis (1991) state a result where the assumptions are not clearly specified, but which could look as follows:

Claim 1 (Sartori and Antsaklis (1991), informally discussed after Lemma 1)**.** *For $n = p$, consider a single-layer random neural network with weights $\boldsymbol{W}^{(0)} \in \mathbb{R}^{(n-1)\times d}$ and signum activation σ that appends a 1 to its output:*

$$\phi_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^p, \boldsymbol{x} \mapsto (\sigma(\boldsymbol{W}^{(0)}\boldsymbol{x}), 1) \ .$$

If $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ are distinct, then for almost all $\boldsymbol{W}^{(0)}$, $\phi_{\boldsymbol{\theta}}(\boldsymbol{X})$ is invertible.

This claim is evidently false for $n \geq 2$. For example, the following argument works for $n \geq 3$: For $\boldsymbol{\tau} \in \{-1, 0, 1\}^n$, let $\mathcal{U}_{\boldsymbol{\tau}} := \{\boldsymbol{w} \in \mathbb{R}^n \mid \sigma(\boldsymbol{w}^\top \boldsymbol{x}_j) = \tau_j \text{ for all } j\}$. Then, $\bigcup_{\boldsymbol{\tau} \in \{-1,0,1\}^n} \mathcal{U}_{\boldsymbol{\tau}} = \mathbb{R}^n$ and since $\{-1, 0, 1\}^n$ is finite, there exists $\boldsymbol{\tau}^*$ such that $\mathcal{U}_{\boldsymbol{\tau}^*}$ is not a Lebesgue null set. Hence the set $\mathcal{W} := \{\boldsymbol{W}^{(0)} \in \mathbb{R}^{(n-1)\times n} \mid \text{at least two rows of } \boldsymbol{W}^{(0)} \text{ are in } \mathcal{U}_{\boldsymbol{\tau}^*}\}$ is not a Lebesgue null set. But for all $\boldsymbol{W}^{(0)} \in \mathcal{W}$, the matrix $\phi_{\boldsymbol{\theta}}(\boldsymbol{X}) = (\sigma(\boldsymbol{X}(\boldsymbol{W}^{(0)})^\top) \mid \mathbf{1}_{n\times 1})$ has two columns equal to $\boldsymbol{\tau}^*$ and is therefore not invertible, contradicting Claim 1.

Second, Tamura and Tateishi (1997) state a result where the assumptions are not clearly formulated, but which could look as follows:

Claim 2 (Tamura and Tateishi (1997))**.** *For $n = p$, consider a single-layer random neural network with weights $\boldsymbol{W}^{(0)} \in \mathbb{R}^{(n-1)\times d}$, biases $\boldsymbol{b}^{(0)} \in \mathbb{R}^{n-1}$ and sigmoid activation σ that appends a 1 to its output:*

$$\phi_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^p, \boldsymbol{x} \mapsto (\sigma(\boldsymbol{W}^{(0)}\boldsymbol{x} + \boldsymbol{b}^{(0)}), 1) \ .$$

If $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ are distinct, then for almost all $\boldsymbol{W}^{(0)}$ and all $a < b$, there exists $\boldsymbol{b}^{(0)} \in [a, b]^{n-1}$ such that $\phi_{\boldsymbol{\theta}}(\boldsymbol{X})$ is invertible.

Tamura and Tateishi (1997) attempt to prove this claim via showing that the curve $c_i : [a, b] \to \mathbb{R}^n, b_i \mapsto \left(\sigma((\boldsymbol{w}_i^{(0)})^\top \boldsymbol{x}_j + b_i)\right)_{j\in[n]}$ is not contained in any $(n-1)$-dimensional subspace of $\mathbb{R}^n$, which would allow them to pick a suitable bias b_i for each curve c_i such that the $c_i(b_i)$ and $(1, \ldots, 1)^\top$ are linearly independent. Although this part is formulated confusingly, it should work out. The major problem is that under the counterassumption that c_i lies in an $(n-1)$-dimensional subspace, they construct an infinite (overdetermined) system of equations involving derivatives of the sigmoid function which they claim has no solution. However, in general, overdetermined systems can have solutions and they do not prove why this would not be the case in their situation. Indeed, the only properties of the sigmoid function they use is that it is C^∞ and not a polynomial, and it is not hard to see that the exponential function has these properties as well and leads to a solvable overdetermined system since its derivatives are all identical.

This leads us to the next claim:

Claim 3 (Huang (2003), Theorem 2.1)**.** *Consider the setting of Claim 2 but with $\boldsymbol{W}^{(0)} \in \mathbb{R}^{n \times d}$ instead of appending a 1 to all feature vectors. If $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ are distinct and $\boldsymbol{\theta}$ has a Lebesgue density, then $\phi_{\boldsymbol{\theta}}(\boldsymbol{X})$ is invertible almost surely over $\boldsymbol{\theta}$.*

Huang (2003) uses the "proof" by Tamura and Tateishi (1997) to show that from any nontrivial intervals one can choose $\boldsymbol{W}^{(0)}, \boldsymbol{b}^{(0)}$ such that $\phi_{\boldsymbol{\theta}}(\boldsymbol{X})$ is invertible, setting all rows of $\boldsymbol{W}^{(0)}$ to be equal. He then concludes without further reasoning that $\phi_{\boldsymbol{\theta}}(\boldsymbol{X})$ must then be invertible almost surely over $\boldsymbol{\theta}$. This major unexplained step cannot be fixed by a continuity-based argument since all $b_i^{(0)}$ were chosen from the same interval and similarly, all rows of $\boldsymbol{W}^{(0)}$ were chosen from the same interval. Since the sigmoid function is analytic, it would, however, be possible to prove this step using the multivariate identity theorem for analytic functions, Theorem 5.H.3, in a fashion similar to the proof of (d) $\Rightarrow$ (a) in Proposition 5.5.2. The approach pursued in Huang (2003) thus introduces an additional problem on top of the problem of Tamura and Tateishi (1997) although this additional problem is in principle fixable.

A similar strategy is reused in the next claim issued in a particularly popular paper:

Claim 4 (Huang et al. (2006), Theorem 2.1)**.** *Claim 3 holds for all C^∞ activation functions σ.*

Not only does the "proof" in Huang et al. (2006) inherit the problems from the previous "proofs", but now the result is obviously false as well: For $\sigma \equiv 0$, the matrix $\phi_{\boldsymbol{\theta}}(\boldsymbol{X})$ can never be invertible, and for $\sigma = \mathrm{id}$, it is also easy to find counterexamples. Moreover, it is not sufficient to exclude (low-order) polynomials σ: For example, the well-known construction (e.g. Remark 3.4 (d) in Chapter V.3 in Amann and Escher, 2005)

$$\sigma(x) := \begin{cases} 0 & , x \leq 0 \\ e^{-1/x} & , x > 0 \end{cases}$$

yields a C^∞ function σ that is zero on $(-\infty, 0]$ but not a polynomial. For this function, it is not hard to see that $\phi_{\boldsymbol{\theta}}(\boldsymbol{X})$ would be singular with nonzero probability since there is a chance that $\boldsymbol{W}^{(0)}\boldsymbol{x} + \boldsymbol{b}^{(0)}$ contains only negative components.

Despite these evident problems and the paper's popularity, Claim 4 is restated years later as Theorem 1 in a survey paper by the same author (Huang et al., 2015). In his Theorem 1, Guo (2018) attempts to disprove Claim 4. However, this "disproof" is also invalid: For certain saturating activation functions like tanh, Guo (2018) lets $\boldsymbol{\theta} \to \infty$ in a certain fashion, which causes $\phi_{\boldsymbol{\theta}}(\boldsymbol{X})$ to converge to a singular matrix. The problem with this approach is that just because the limiting matrix is singular, the matrices for finite $\boldsymbol{\theta}$ do not need to be singular. However, this limiting behavior might at least explain the empirical results of Henriquez and Ruz (2017), which find in a certain setting with sigmoid activation that numerically, $\phi_{\boldsymbol{\theta}}(\boldsymbol{X})$ is usually not a full-rank matrix. In contrast, Widrow et al. (2013) numerically reach the opposite conclusion. This is not a contradiction, considering that very small eigenvalues of $\phi_{\boldsymbol{\theta}}(\boldsymbol{X})$ may be numerically rounded to zero, and the probability of having such small eigenvalues depends on the chosen distributions of $\boldsymbol{x}$ and $\boldsymbol{\theta}$.

The only previously published correct result known to us is the following one:

Lemma 5.E.5 (Lemma 4.4 in Nguyen and Hein (2017)). *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be analytic, strictly increasing and let one of the following two conditions be satisfied:*

(a) σ is bounded, or
(b) there exist positive constants $\rho_1, \rho_2, \rho_3, \rho_4$ such that $|\sigma(t)| \leq \rho_1 e^{\rho_2 t}$ for $t < 0$ and $|\sigma(t)| \leq \rho_3 t + \rho_4$ for $t \geq 0$.

Let $\phi_{\boldsymbol{\theta}}$ be a random NN feature map with biases as in Theorem 5.5.6 (b) and let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ be distinct. If $p \geq n-1$, then the $n \times (p+1)$ feature matrix

$$\begin{pmatrix} \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_1)^\top & 1 \\ \vdots & \vdots \\ \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_n)^\top & 1 \end{pmatrix}$$

has rank n for (Lebesgue-) almost all $\boldsymbol{\theta}$.

In Theorem 5.H.9, we generalize Lemma 5.E.5 (without the appended ones in the feature matrix) to a substantially larger class of analytic activation functions.[8] As argued above, it is not possible to replace the assumption that σ is analytic with $\sigma \in C^\infty$ and as shown in Remark 5.H.17, our exclusion of certain polynomials for σ is in general necessary.

5.F Proofs for Section 5.4

In this section, we prove our main theorem and corollary from Section 5.4. Recall from Definition 5.4.2 that if $\mathbb{E}\|\boldsymbol{z}\|_2^2 < \infty$, we can define the second moment matrix

$$\boldsymbol{\Sigma} := \mathbb{E}_{\boldsymbol{z} \sim P_Z}\left[\boldsymbol{z}\boldsymbol{z}^\top\right] \in \mathbb{R}^{p \times p} ,$$

and if $\boldsymbol{\Sigma}$ is invertible, we can also define the "whitened" data matrix $\boldsymbol{W} := \boldsymbol{Z}\boldsymbol{\Sigma}^{-1/2}$, whose rows $\boldsymbol{w}_i = \boldsymbol{\Sigma}^{-1/2}$ satisfy $\mathbb{E}\boldsymbol{w}_i\boldsymbol{w}_i^\top = \boldsymbol{I}_p$.

Theorem 5.4.3 (Main result). *Let $n, p \geq 1$. Assume that P and ϕ satisfy:*

(INT) $\mathbb{E}y^2 < \infty$ and hence $R_P(f_P^*) < \infty$,
(NOI) $\operatorname{Var}(y|\boldsymbol{z}) \geq \sigma^2$ almost surely over $\boldsymbol{z}$,
(MOM) $\mathbb{E}\|\boldsymbol{z}\|_2^2 < \infty$, i.e. $\boldsymbol{\Sigma}$ exists and is finite,
(COV) $\boldsymbol{\Sigma}$ is invertible,
(FRK) $\boldsymbol{Z} \in \mathbb{R}^{n \times p}$ almost surely has full rank, i.e. $\operatorname{rank} \boldsymbol{Z} = \min\{n, p\}$.

[8]It is in principle possible to extend our results to the case with appended ones in the feature matrix by choosing the activation function for one of the output neurons to be $\sigma \equiv 1$. As discussed in Remark 5.H.16, our arguments have no problem handling different activation functions. In this case, we would only need to adapt the corresponding Taylor series coefficients in Lemma 5.H.8.

Then, for the ridgeless linear regression estimator $f_{\boldsymbol{Z},\boldsymbol{y}}(\boldsymbol{z}) = \boldsymbol{z}^\top \boldsymbol{Z}^+ \boldsymbol{y}$, the following holds:

$$\text{If } p \geq n, \qquad \mathcal{E}_{\text{Noise}} \overset{\text{(I)}}{\geq} \sigma^2 \mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+) \overset{\text{(II)}}{\geq} \sigma^2 \mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1}) \overset{\text{(IV)}}{\geq} \sigma^2 \frac{n}{p+1-n} .$$

$$\text{If } p \leq n, \qquad \mathcal{E}_{\text{Noise}} \overset{\text{(I)}}{\geq} \sigma^2 \mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+) \overset{\text{(III)}}{=} \sigma^2 \mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{W}^\top\boldsymbol{W})^{-1}) \overset{\text{(V)}}{\geq} \sigma^2 \frac{p}{n+1-p} .$$

Here, the matrix inverses exist almost surely in the considered cases. Moreover, we have:

- *If (NOI) holds with equality, then* (I) *holds with equality.*
- *If $n = p$ or $\boldsymbol{\Sigma} = \lambda \boldsymbol{I}_p$ for some $\lambda > 0$, then* (II) *holds with equality.*

Proof. By (FRK), we only consider the case where $\boldsymbol{Z}$ has full rank. For further details on some matrix computations, we refer to Appendix 5.B.

Step 1: Preparation. Since the pairs $(\boldsymbol{z}_1, y_1), \ldots, (\boldsymbol{z}_n, y_n)$ are independent, we have the conditional covariance matrix

$$\operatorname{Cov}(\boldsymbol{y}|\boldsymbol{Z}) = \begin{pmatrix} \operatorname{Var}(y_1|\boldsymbol{z}_1) & & \\ & \ddots & \\ & & \operatorname{Var}(y_n|\boldsymbol{z}_n) \end{pmatrix} \overset{\text{(I)}}{\succeq} \sigma^2 \boldsymbol{I}_n$$

with equality in (I) if (NOI) holds with equality. Therefore,

$$\begin{aligned}
\mathcal{E}_{\text{Noise}} &\overset{\text{def}}{=} \mathbb{E}_{\boldsymbol{Z},\boldsymbol{y},\boldsymbol{\theta},\boldsymbol{z}} \left(f_{\boldsymbol{Z},\boldsymbol{y},\boldsymbol{\theta}}(\boldsymbol{z}) - \mathbb{E}_{\boldsymbol{y}|\boldsymbol{Z}} f_{\boldsymbol{Z},\boldsymbol{y},\boldsymbol{\theta}}(\boldsymbol{z}) \right)^2 \\
&= \mathbb{E}_{\boldsymbol{Z},\boldsymbol{y},\boldsymbol{z}} \left(\boldsymbol{z}^\top \boldsymbol{Z}^+ \boldsymbol{y} - \mathbb{E}_{\boldsymbol{y}|\boldsymbol{Z}} \boldsymbol{z}^\top \boldsymbol{Z}^+ \boldsymbol{y} \right)^2 \\
&= \mathbb{E}_{\boldsymbol{Z},\boldsymbol{z}} \mathbb{E}_{\boldsymbol{y}|\boldsymbol{Z}} \boldsymbol{z}^\top \boldsymbol{Z}^+ (\boldsymbol{y} - \mathbb{E}_{\boldsymbol{y}|\boldsymbol{Z}} \boldsymbol{y})(\boldsymbol{y} - \mathbb{E}_{\boldsymbol{y}|\boldsymbol{Z}} \boldsymbol{y})^\top (\boldsymbol{Z}^+)^\top \boldsymbol{z} \\
&= \mathbb{E}_{\boldsymbol{Z},\boldsymbol{z}} \boldsymbol{z}^\top \boldsymbol{Z}^+ \operatorname{Cov}(\boldsymbol{y}|\boldsymbol{Z}) (\boldsymbol{Z}^+)^\top \boldsymbol{z} \\
&\overset{\text{(I)}}{\geq} \sigma^2 \mathbb{E}_{\boldsymbol{Z},\boldsymbol{z}} \boldsymbol{z}^\top \boldsymbol{Z}^+ (\boldsymbol{Z}^+)^\top \boldsymbol{z} \\
&= \sigma^2 \mathbb{E}_{\boldsymbol{Z},\boldsymbol{z}} \operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{z}\boldsymbol{z}^\top \boldsymbol{Z}^+) \\
&= \sigma^2 \mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} (\boldsymbol{Z}^+)^\top) .
\end{aligned}$$

Step 2.1: Reformulation, underparameterized case. In the case $p \leq n$, we have $\boldsymbol{Z}^+ = (\boldsymbol{Z}^\top \boldsymbol{Z})^{-1} \boldsymbol{Z}^\top$ thanks to (FRK) and thus

$$\operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+) = \operatorname{tr}(\boldsymbol{\Sigma}^{1/2} \boldsymbol{Z}^+ (\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma}^{1/2}) = \operatorname{tr}(\boldsymbol{\Sigma}^{1/2} (\boldsymbol{Z}^\top \boldsymbol{Z})^{-1} \boldsymbol{\Sigma}^{1/2}) = \operatorname{tr}((\boldsymbol{W}^\top \boldsymbol{W})^{-1}) .$$

Step 2.2: Reformulation, overparameterized case. In the case $p \geq n$, we have $\boldsymbol{Z}^+ = \boldsymbol{Z}^\top (\boldsymbol{Z}\boldsymbol{Z}^\top)^{-1}$ thanks to (FRK). For $\boldsymbol{\Sigma} = \lambda \boldsymbol{I}_p$, we can show $\operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+) = \operatorname{tr}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1})$ similar to Step 2.1. For general $\boldsymbol{\Sigma}$, we can obtain a lower bound by "removing a projection": First, let

$$\begin{aligned}
\boldsymbol{S} &:= (\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+ = (\boldsymbol{Z}\boldsymbol{Z}^\top)^{-1} \boldsymbol{Z} \boldsymbol{\Sigma} \boldsymbol{Z}^\top (\boldsymbol{Z}\boldsymbol{Z}^\top)^{-1} , \\
\boldsymbol{A} &:= \boldsymbol{\Sigma}^{1/2} \boldsymbol{Z}^\top .
\end{aligned}$$

Now, since $\boldsymbol{Z}$ and $\boldsymbol{\Sigma}$ have full rank, we can compute

$$\boldsymbol{S}^{-1} = \boldsymbol{Z}\boldsymbol{Z}^\top (\boldsymbol{Z}\boldsymbol{\Sigma}\boldsymbol{Z}^\top)^{-1} \boldsymbol{Z}\boldsymbol{Z}^\top = \boldsymbol{W}\boldsymbol{A}(\boldsymbol{A}^\top \boldsymbol{A})^{-1} \boldsymbol{A}^\top \boldsymbol{W}^\top .$$

Since $\boldsymbol{A}(\boldsymbol{A}^\top \boldsymbol{A})^{-1}\boldsymbol{A}^\top$ is the orthogonal projection onto the column space of $\boldsymbol{A}$, we have $\boldsymbol{S}^{-1} \preceq \boldsymbol{W}\boldsymbol{W}^\top$ and hence $\lambda_i(\boldsymbol{S}^{-1}) \leq \lambda_i(\boldsymbol{W}\boldsymbol{W}^\top)$ by the Courant-Fischer-Weyl theorem. This yields

$$\begin{aligned}\operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+) &= \operatorname{tr}(\boldsymbol{S}) = \sum_{i=1}^n \lambda_{n+1-i}(\boldsymbol{S}) = \sum_{i=1}^n \frac{1}{\lambda_i(\boldsymbol{S}^{-1})} \\ &\geq \sum_{i=1}^n \frac{1}{\lambda_i(\boldsymbol{W}\boldsymbol{W}^\top)} = \sum_{i=1}^n \lambda_{n+1-i}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1}) = \operatorname{tr}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1}) \ .\end{aligned}$$

For $n = p$, $\boldsymbol{A}(\boldsymbol{A}^\top \boldsymbol{A})^{-1}\boldsymbol{A}^\top$ projects onto a p-dimensional space and is therefore the identity matrix, yielding equality.

Step 3.0: Random matrix bound, $p = 1$ or $n = 1$. If $n \geq p = 1$, $\boldsymbol{w}_i = w_i$ is a scalar and we have

$$\begin{aligned}\mathbb{E}\operatorname{tr}((\boldsymbol{W}^\top \boldsymbol{W})^{-1}) &= \mathbb{E}\frac{1}{\boldsymbol{W}^\top \boldsymbol{W}} = \mathbb{E}\frac{1}{\sum_{i=1}^n w_i^2} \geq \frac{1}{\mathbb{E}\sum_{i=1}^n w_i^2} = \frac{1}{\sum_{i=1}^n \operatorname{tr}(\mathbb{E} w_i w_i^\top)} = \frac{1}{n} \\ &= \frac{p}{n+1-p}\end{aligned}$$

by Jensen's inequality. Similarly, for $p \geq n = 1$, we obtain

$$\mathbb{E}\operatorname{tr}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1}) = \mathbb{E}\frac{1}{\boldsymbol{w}_1^\top \boldsymbol{w}_1} \geq \frac{1}{\mathbb{E}\boldsymbol{w}_1^\top \boldsymbol{w}_1} = \frac{1}{\operatorname{tr}(\mathbb{E}\boldsymbol{w}_1\boldsymbol{w}_1^\top)} = \frac{1}{p} = \frac{n}{p+1-n} \ .$$

Step 3.1: Random matrix bound, overparameterized case. We first consider the overparameterized case $p \geq n \geq 2$ and block-decompose

$$\boldsymbol{W} =: \begin{pmatrix} \boldsymbol{w}_1^\top \\ \boldsymbol{W}_2 \end{pmatrix} \in \mathbb{R}^{(1+(n-1))\times p} \quad \Rightarrow \quad \boldsymbol{W}\boldsymbol{W}^\top = \begin{pmatrix} \boldsymbol{w}_1^\top \boldsymbol{w}_1 & \boldsymbol{w}_1^\top \boldsymbol{W}_2^\top \\ \boldsymbol{W}_2 \boldsymbol{w}_1 & \boldsymbol{W}_2 \boldsymbol{W}_2^\top \end{pmatrix} \ .$$

Since $\boldsymbol{Z}$ has full rank, $\boldsymbol{W}$ has full rank. Because of $n \leq p$, it follows that $\boldsymbol{W}\boldsymbol{W}^\top \succ 0$. Therefore,

$$\left((\boldsymbol{W}\boldsymbol{W}^\top)^{-1}\right)_{11} = \left(\boldsymbol{w}_1^\top \boldsymbol{w}_1 - \boldsymbol{w}_1^\top \boldsymbol{W}_2^\top (\boldsymbol{W}_2 \boldsymbol{W}_2^\top)^{-1} \boldsymbol{W}_2 \boldsymbol{w}_1\right)^{-1} = \left(\boldsymbol{w}_1^\top (\boldsymbol{I}_p - \boldsymbol{P}_2)\boldsymbol{w}_1\right)^{-1} \ ,$$

where

$$\boldsymbol{P}_2 \coloneqq \boldsymbol{W}_2^\top \left(\boldsymbol{W}_2 \boldsymbol{W}_2^\top\right)^{-1} \boldsymbol{W}_2 \in \mathbb{R}^{p\times p}$$

is the orthogonal projection onto the column space of $\boldsymbol{W}_2^\top$. Thus, $\boldsymbol{P}_2$ has the eigenvalues 1 with multiplicity $n-1$ and 0 with multiplicity $p-(n-1)$, which yields $\operatorname{tr}(\boldsymbol{P}_2) = n-1$. Since the $\boldsymbol{z}_i$ are stochastically independent, $\boldsymbol{w}_1$ and $\boldsymbol{W}_2$ are also stochastically independent and we obtain

$$\mathbb{E}\boldsymbol{w}_1^\top (\boldsymbol{I}_p - \boldsymbol{P}_2)\boldsymbol{w}_1 = \mathbb{E}\operatorname{tr}((\boldsymbol{I}_p - \boldsymbol{P}_2)(\mathbb{E}_{\boldsymbol{w}_1} \boldsymbol{w}_1 \boldsymbol{w}_1^\top)) = \mathbb{E}\operatorname{tr}(\boldsymbol{I}_p - \boldsymbol{P}_2) = p+1-n \ .$$

Using Jensen's inequality with the convex function $(0,\infty) \to (0,\infty), x \mapsto 1/x$, we thus find that

$$\mathbb{E}\left((\boldsymbol{W}\boldsymbol{W}^\top)^{-1}\right)_{11} = \mathbb{E}\left(\boldsymbol{w}_1^\top (\boldsymbol{I}_p - \boldsymbol{P}_2)\boldsymbol{w}_1\right)^{-1} \geq \frac{1}{\mathbb{E}\boldsymbol{w}_1^\top (\boldsymbol{I}_p - \boldsymbol{P}_2)\boldsymbol{w}_1} = \frac{1}{p+1-n} \ .$$

Since $\operatorname{tr}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1}) = \sum_{i=1}^n ((\boldsymbol{W}\boldsymbol{W}^\top)^{-1})_{ii}$ and all diagonal entries can be treated in the same fashion (e.g. via permutation of the $\boldsymbol{w}_i$), it follows that

$$\mathbb{E}\operatorname{tr}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1}) \geq \frac{n}{p+1-n} \ .$$

Step 3.2: Random matrix bound, underparameterized case. In the case $n \geq p \geq 1$, we follow the proof of Corollary 1 in Mourtada (2022), which can be applied in our setting as follows: Introduce a new random variable $\boldsymbol{w}_{n+1}$ such that $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{n+1}$ are i.i.d. Then,

$$\mathbb{E}\operatorname{tr}((\boldsymbol{W}^\top\boldsymbol{W})^{-1}) = \mathbb{E}_{\boldsymbol{W}}\operatorname{tr}((\boldsymbol{W}^\top\boldsymbol{W})^{-1}\mathbb{E}_{\boldsymbol{w}_{n+1}}\boldsymbol{w}_{n+1}\boldsymbol{w}_{n+1}^\top) = \mathbb{E}\boldsymbol{w}_{n+1}^\top(\boldsymbol{W}^\top\boldsymbol{W})^{-1}\boldsymbol{w}_{n+1} \ .$$

Now, Lemma 1 in Mourtada (2022) uses the Sherman-Morrison formula to show that

$$\boldsymbol{w}_{n+1}^\top(\boldsymbol{W}^\top\boldsymbol{W})^{-1}\boldsymbol{w}_{n+1} = \frac{\hat{\ell}_{n+1}}{1-\hat{\ell}_{n+1}}$$

with the so-called leverage score

$$\hat{\ell}_{n+1} \coloneqq \boldsymbol{w}_{n+1}^\top(\boldsymbol{W}^\top\boldsymbol{W} + \boldsymbol{w}_{n+1}\boldsymbol{w}_{n+1}^\top)^{-1}\boldsymbol{w}_{n+1} \in [0, 1) \ .$$

Since $\boldsymbol{W}^\top\boldsymbol{W} + \boldsymbol{w}_{n+1}\boldsymbol{w}_{n+1}^\top = \sum_{i=1}^{n+1}\boldsymbol{w}_i\boldsymbol{w}_i^\top$ and since $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{n+1}$ are i.i.d., we obtain

$$\begin{aligned}
\mathbb{E}\hat{\ell}_{n+1} &= \mathbb{E}\operatorname{tr}\left(\left(\sum_{i=1}^{n+1}\boldsymbol{w}_i\boldsymbol{w}_i^\top\right)^{-1}\boldsymbol{w}_{n+1}\boldsymbol{w}_{n+1}^\top\right) \\
&= \frac{1}{n+1}\mathbb{E}\operatorname{tr}\left(\left(\sum_{i=1}^{n+1}\boldsymbol{w}_i\boldsymbol{w}_i^\top\right)^{-1}\sum_{i=1}^{n+1}\boldsymbol{w}_i\boldsymbol{w}_i^\top\right) \\
&= \frac{1}{n+1}\mathbb{E}\operatorname{tr}(\boldsymbol{I}_p) = \frac{p}{n+1} \ .
\end{aligned}$$

Finally, the function $[0, 1) \to (0, \infty), x \mapsto \frac{x}{1-x} = \frac{1}{1-x} - 1$ is convex, and Jensen's inequality yields

$$\mathbb{E}\operatorname{tr}((\boldsymbol{W}^\top\boldsymbol{W})^{-1}) = \mathbb{E}\frac{\hat{\ell}_{n+1}}{1-\hat{\ell}_{n+1}} \geq \frac{\mathbb{E}\hat{\ell}_{n+1}}{1-\mathbb{E}\hat{\ell}_{n+1}} = \frac{p}{n+1-p} \ . \qquad \square$$

Note that it is not possible to analyze Step 3.2 like Step 3.1 since the corresponding matrix blocks are not stochastically independent.

Corollary 5.4.4 (Random features)**.** *Let $\boldsymbol{\theta} \sim P_\Theta$ be a random variable such that $\phi_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^p$ is a random feature map. Consider the random features regression estimator $f_{\boldsymbol{X},\boldsymbol{y},\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{z}_{\boldsymbol{\theta}}^\top\boldsymbol{Z}_{\boldsymbol{\theta}}^+\boldsymbol{y}$ with $\boldsymbol{z}_{\boldsymbol{\theta}} \coloneqq \phi_{\boldsymbol{\theta}}(\boldsymbol{x})$ and $\boldsymbol{Z}_{\boldsymbol{\theta}} \coloneqq \phi_{\boldsymbol{\theta}}(\boldsymbol{X})$. If for P_Θ-almost all $\widetilde{\boldsymbol{\theta}}$, the assumptions of Theorem 5.4.3 are satisfied for $\boldsymbol{z} = \boldsymbol{z}_{\widetilde{\boldsymbol{\theta}}}$ and $\boldsymbol{Z} = \boldsymbol{Z}_{\widetilde{\boldsymbol{\theta}}}$ (with the corresponding matrix $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{\widetilde{\boldsymbol{\theta}}}$), then*

$$\mathcal{E}_{\text{Noise}} \geq \begin{cases} \sigma^2\frac{n}{p+1-n} & \text{if } p \geq n, \\ \sigma^2\frac{p}{n+1-p} & \text{if } p \leq n. \end{cases}$$

Proof. First, let $p \leq n$. Since $\boldsymbol{\theta}$ is independent from $\boldsymbol{X}, \boldsymbol{x}, \boldsymbol{y}$,

$$\begin{aligned}
\mathcal{E}_{\text{Noise}} &= \mathbb{E}_{\boldsymbol{X},\boldsymbol{y},\boldsymbol{\theta},\boldsymbol{x}} \left(\boldsymbol{z}_{\boldsymbol{\theta}}^\top \boldsymbol{Z}_{\boldsymbol{\theta}}^+ \boldsymbol{y} - \mathbb{E}_{\boldsymbol{y}|\boldsymbol{X}} \boldsymbol{z}_{\boldsymbol{\theta}}^\top \boldsymbol{Z}_{\boldsymbol{\theta}}^+ \boldsymbol{y}\right)^2 \\
&= \mathbb{E}_{\boldsymbol{\theta}} \left[\mathbb{E}_{\boldsymbol{X},\boldsymbol{y},\boldsymbol{x}} \left(\boldsymbol{z}_{\boldsymbol{\theta}}^\top \boldsymbol{Z}_{\boldsymbol{\theta}}^+ \boldsymbol{y} - \mathbb{E}_{\boldsymbol{y}|\boldsymbol{X}} \boldsymbol{z}_{\boldsymbol{\theta}}^\top \boldsymbol{Z}_{\boldsymbol{\theta}}^+ \boldsymbol{y}\right)^2\right] \\
&\overset{\text{Theorem 5.4.3}}{\geq} \mathbb{E}_{\boldsymbol{\theta}} \sigma^2 \frac{p}{n+1-p} = \sigma^2 \frac{p}{n+1-p} \ .
\end{aligned}$$

The case $p \geq n$ can be treated analogously. □

5.G Discussion of the Main Theorem

Remark 5.G.1 (Dependence on $\boldsymbol{\Sigma}$)**.** Hastie et al. (2022) discuss that $\boldsymbol{\Sigma}$ only influences the expected excess risk in the overparameterized regime. In the following, we will illustrate that Theorem 5.4.3 even implies that in the overparameterized case, $\boldsymbol{\Sigma} = \lambda \boldsymbol{I}_d$ yields the lowest $\mathcal{E}_{\text{Noise}}$. This fact is also discussed in Muthukumar et al. (2020) in a slightly different setting. Note that since P_X is unknown in general, $\boldsymbol{\Sigma}$ is also unknown in general.

Assume that (NOI) holds with equality such that we have $\mathcal{E}_{\text{Noise}} = \sigma^2 \mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+)$ by Theorem 5.4.3. Suppose that we perform linear regression on the whitened data $\tilde{\boldsymbol{z}} := \boldsymbol{\Sigma}^{-1/2} \boldsymbol{z}$. Then,

$$\begin{aligned}
\tilde{\boldsymbol{Z}} &= \boldsymbol{Z} \boldsymbol{\Sigma}^{-1/2} = \boldsymbol{W} \ , \\
\tilde{\boldsymbol{\Sigma}} &= \mathbb{E}_{\tilde{\boldsymbol{z}}} \tilde{\boldsymbol{z}} \tilde{\boldsymbol{z}}^\top = \boldsymbol{\Sigma}^{-1/2} \left[\mathbb{E}_{\boldsymbol{z}} \boldsymbol{z} \boldsymbol{z}^\top\right] \boldsymbol{\Sigma}^{-1/2} = \boldsymbol{I}_p \ , \\
\tilde{\boldsymbol{W}} &= \tilde{\boldsymbol{Z}} \tilde{\boldsymbol{\Sigma}}^{-1/2} = \tilde{\boldsymbol{Z}} = \boldsymbol{W} \ .
\end{aligned}$$

In the underparameterized case $p \leq n$, we then obtain

$$\mathbb{E}_{\tilde{\boldsymbol{Z}}} \operatorname{tr}((\tilde{\boldsymbol{Z}}^+)^\top \tilde{\boldsymbol{\Sigma}} \tilde{\boldsymbol{Z}}^+) = \mathbb{E}_{\tilde{\boldsymbol{Z}}} \operatorname{tr}((\tilde{\boldsymbol{W}}^\top \tilde{\boldsymbol{W}})^{-1}) = \mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{W}^\top \boldsymbol{W})^{-1}) = \mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+) \ .$$

Therefore, whitening the data does not make a difference if $p \leq n$. In contrast, for $p > n$, we only know

$$\mathbb{E}_{\tilde{\boldsymbol{Z}}} \operatorname{tr}((\tilde{\boldsymbol{Z}}^+)^\top \tilde{\boldsymbol{\Sigma}} \tilde{\boldsymbol{Z}}^+) \overset{\text{(II)}}{=} \mathbb{E}_{\tilde{\boldsymbol{Z}}} \operatorname{tr}((\tilde{\boldsymbol{W}} \tilde{\boldsymbol{W}}^\top)^{-1}) = \mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{W} \boldsymbol{W}^\top)^{-1}) \overset{\text{(II)}}{\leq} \mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+)$$

since (II) holds with equality for whitened features. From Step 2.1 in the proof of Theorem 5.4.3, it is obvious that (II) in general does not hold with equality. Hence, in the overparameterized case $p > n$, whitening the features often reduces and never increases $\mathcal{E}_{\text{Noise}}$. Since $\mathcal{E}_{\text{Noise}}$ is just a lower bound for the expected excess risk, whitening does not necessarily reduce the expected excess risk.

This phenomenon also has a different kernel-based interpretation: Under the assumptions (MOM) and (COV), we can choose an ONB $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p$ of eigenvectors of $\boldsymbol{\Sigma}$ with corresponding eigenvalues $\lambda_1, \ldots, \lambda_p > 0$. If $\boldsymbol{z} = \phi(\boldsymbol{x})$ and $k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}')$, we can write

$$k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^\top \left(\sum_{i=1}^n \boldsymbol{u}_i \boldsymbol{u}_i^\top\right) \phi(\boldsymbol{x}') = \sum_{i=1}^p \lambda_i \psi_i(\boldsymbol{x}) \psi_i(\boldsymbol{x}') \ , \tag{5.6}$$

where the functions $\psi_i : \mathbb{R}^d \to \mathbb{R}, \boldsymbol{x} \mapsto \frac{1}{\sqrt{\lambda_i}} \boldsymbol{u}_i^\top \phi(\boldsymbol{x})$ form an orthonormal system in $L_2(P_X)$:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}} \psi_i(\boldsymbol{x}) \psi_j(\boldsymbol{x}) &= \frac{1}{\sqrt{\lambda_i \lambda_j}} \boldsymbol{u}_i^\top \left[\mathbb{E}_{\boldsymbol{x}} \phi(\boldsymbol{x}) \phi(\boldsymbol{x})^\top\right] \boldsymbol{u}_j = \frac{1}{\sqrt{\lambda_i \lambda_j}} \boldsymbol{u}_i^\top \boldsymbol{\Sigma} \boldsymbol{u}_j = \frac{\lambda_j}{\sqrt{\lambda_i \lambda_j}} \boldsymbol{u}_i^\top \boldsymbol{u}_j \\ &= \delta_{ij} . \end{aligned} \tag{5.7}$$

Therefore, Eq. (5.6) is a Mercer representation of k and the eigenvalues λ_i of $\boldsymbol{\Sigma}$ are also the eigenvalues of the integral operator $T_k : L_2(P_X) \to L_2(P_X)$ associated with k:

$$(T_k f)(\boldsymbol{x}) := \int k(\boldsymbol{x}, \boldsymbol{x}') f(\boldsymbol{x}') \,\mathrm{d}P_X(\boldsymbol{x}') = \sum_{i=1}^n \lambda_i \langle \psi_i, f \rangle_{L_2(P_X)} \psi_i(\boldsymbol{x}) \ .$$

We can define a kernel $\tilde{k}$ with flattened eigenspectrum via

$$\tilde{k}(\boldsymbol{x}, \boldsymbol{x}') := \sum_{i=1}^p \psi_i(\boldsymbol{x}) \psi_i(\boldsymbol{x}') \ .$$

Its feature map $\boldsymbol{x} \mapsto (\psi_i(\boldsymbol{x}))_{i \in [n]}$ by Eq. (5.7) satisfies $\boldsymbol{\Sigma} = \boldsymbol{I}_p$. By the above discussion, $\mathcal{E}_{\text{Noise}}(\tilde{k}) \leq \mathcal{E}_{\text{Noise}}(k)$. However, this needs to be taken with caution: Assume for simplicity that for independent $\boldsymbol{x}, \tilde{\boldsymbol{x}}$, the feature map ϕ yields $\phi(\boldsymbol{x}), \phi(\tilde{\boldsymbol{x}}) \sim \mathcal{N}(0, \frac{1}{p}\boldsymbol{I}_p)$. Then, $k(\boldsymbol{x}, \boldsymbol{x}) = 1$ but $\mathbb{E} k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = 0$ and $\operatorname{Var} k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \frac{1}{p^2} \sum_{i=1}^p \mathbb{E}_{u,v \sim \mathcal{N}(0,1)} u^2 v^2 = \frac{1}{p}$. In this sense, for $p \to \infty$, k converges to the Dirac kernel $k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \delta_{\boldsymbol{x}, \tilde{\boldsymbol{x}}}$, which satisfies $\mathcal{E}_{\text{Noise}} = 0$ but provides bad interpolation performance if $f_P^* \not\equiv 0$. ◀

The next lemma and its proof are an adaptation of Lemma 4.14 in Bordenave and Chafaï (2012).

Lemma 5.G.2. *For $i \in [n]$, let $\mathcal{W}_{-i} := \operatorname{Span}\{\boldsymbol{w}_j \mid j \in [n] \setminus \{i\}\}$. Then, under the assumptions of Theorem 5.4.3, in the overparameterized case $p \geq n$,*

$$\operatorname{tr}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1}) = \sum_{i=1}^n \operatorname{dist}(\boldsymbol{w}_i, \mathcal{W}_{-i})^{-2} \ . \tag{5.8}$$

Proof. The case $n = 1$ is trivial, hence let $n \geq 2$. In Step 3.1 in the proof of Theorem 5.4.3, since $\boldsymbol{P}_2$ is the orthogonal projection onto $\mathcal{W}_{-1}$, we have

$$\boldsymbol{w}_1^\top (\boldsymbol{I}_p - \boldsymbol{P}_2) \boldsymbol{w}_1 = \|\boldsymbol{w}_1\|_2^2 - \|\boldsymbol{P}_2 \boldsymbol{w}_1\|_2^2 \overset{\text{Pythagoras}}{=} \operatorname{dist}(\boldsymbol{w}_1, \mathcal{W}_{-1})^2 \ ,$$

where $\operatorname{dist}(\boldsymbol{w}_1, \mathcal{W}_{-1})$ is the Euclidean distance between $\boldsymbol{w}_1$ and $\mathcal{W}_{-1}$. □

Remark 5.G.3 (Is $\mathcal{U}(\mathbb{S}^{p-1})$ optimal?)**.** From (I) in Theorem 5.4.3 it is clear that the best possible lower bound for $\mathcal{E}_{\text{Noise}}$ under the assumptions of Theorem 5.4.3 given $n, p \geq 1$ is

$$\mathcal{E}_{\text{Noise}} \geq \sigma^2 \inf_{\text{Distribution } P_Z \text{ on } \mathbb{R}^p \text{ satisfying (MOM), (COV), (FRK)}} \mathbb{E}_{\boldsymbol{Z} \sim P_Z^n} \operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+) \ . \tag{5.9}$$

Here, we want to discuss the hypothesis that the infimum in Eq. (5.9) is achieved (for example) by $P_Z = \mathcal{U}(\mathbb{S}^{p-1})$. Figure 5.1 shows that we were not able to obtain a lower $\mathcal{E}_{\text{Noise}}$ by optimizing a neural network feature map to minimize $\mathcal{E}_{\text{Noise}}$. In the following, we want to discuss some theoretical evidence as to why this is plausible in the overparameterized case $p \geq n$. Lemma 5.G.2 shows that Step 3.1 of the proof of Theorem 5.4.3 has a distance-based interpretation. In this interpretation, Step 3.1 then applies Jensen's inequality to the convex function $(0, \infty) \to (0, \infty), x \mapsto 1/x$ using

$$\mathbb{E}\,\mathrm{dist}(\boldsymbol{w}_i, \mathcal{W}_{-i})^2 = p + 1 - n \ . \tag{5.10}$$

We can use this perspective to gain insights on how distributions P_Z with small $\mathcal{E}_{\text{Noise}}$ in the overparameterized case $p \geq n$ look like. First of all, Remark 5.G.1 suggests that for minimizing $\mathcal{E}_{\text{Noise}}$ in the overparameterized case, $\boldsymbol{\Sigma}$ should be a multiple of $\boldsymbol{I}_p$, which is clearly satisfied for $\mathcal{U}(\mathbb{S}^{p-1})$ by Theorem 5.6.1. Since the lower bound obtained from (5.10) is independent of the distribution of $\boldsymbol{W}$, minimizing $\mathbb{E}\,\mathrm{tr}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1})$ amounts to minimizing the error made by Jensen's inequality, which essentially amounts to reducing the variance of the random variables $\mathrm{dist}(\boldsymbol{w}_i, \mathcal{W}_{-i})$. We can decompose $\mathrm{dist}(\boldsymbol{w}_i, \mathcal{W}_{-i}) = \|\boldsymbol{w}_i\|_2 \cdot \mathrm{dist}(\boldsymbol{w}_i/\|\boldsymbol{w}_i\|_2, \mathcal{W}_{-i})$, where $\mathrm{dist}(\boldsymbol{w}_i/\|\boldsymbol{w}_i\|_2, \mathcal{W}_{-i})$ only depends on the angular components $\boldsymbol{w}_j/\|\boldsymbol{w}_j\|_2$ for $j \in [n]$. This suggests that for a "good" distribution P_W,

- the radial component $\|\boldsymbol{w}_i\|_2$ should have low variance, and
- the distribution of the angular component $\boldsymbol{w}_i/\|\boldsymbol{w}_i\|_2$ should not contain "clusters", since clusters would increase the probability of $\mathrm{dist}(\boldsymbol{w}_i, \mathcal{W}_{-i})$ being very small.

Clearly, both points are perfectly achieved for $P_Z = \mathcal{U}(\mathbb{S}^{p-1})$. ◀

5.H Proofs for Section 5.5

In this section, we prove all theorems and propositions from Section 5.5 as well as some additional results.

5.H.1 Miscellaneous

First, we prove a statement about conditional variances.

Lemma 5.H.1. *In the setting of Theorem 5.4.3, we have*

$$\mathrm{Var}(y|\boldsymbol{z}) = \mathbb{E}(\mathrm{Var}(y|\boldsymbol{x})|\boldsymbol{z}) + \mathrm{Var}(\mathbb{E}(y|\boldsymbol{x})|\boldsymbol{z}) \ .$$

Hence, if $\mathrm{Var}(y|\boldsymbol{x}) \geq \sigma^2$ *almost surely over* $\boldsymbol{x}$*, then* $\mathrm{Var}(y|\boldsymbol{z}) \geq \sigma^2$ *almost surely over* $\boldsymbol{z}$*. The converse holds, for example, if* ϕ *is injective.*

Proof. For properties of conditional expectations, we refer to the literature, e.g. Chapter 4.1 in Durrett (2019). Since $\boldsymbol{z} = \phi(\boldsymbol{x})$ is a function of $\boldsymbol{x}$, we have $\mathbb{E}[\cdot|\boldsymbol{z}] = \mathbb{E}[\mathbb{E}(\cdot|\boldsymbol{x})|\boldsymbol{z}]$. Thus,

$$\begin{aligned}
\mathrm{Var}(y|\boldsymbol{z}) &= \mathbb{E}\left[(y - \mathbb{E}(y|\boldsymbol{z}))^2 \middle| \boldsymbol{z}\right] \\
&= \mathbb{E}\left[\Big((y - \mathbb{E}(y|\boldsymbol{x})) + (\mathbb{E}(y|\boldsymbol{x}) - \mathbb{E}(\mathbb{E}(y|\boldsymbol{x})|\boldsymbol{z}))\Big)^2 \middle| \boldsymbol{z}\right] \\
&= \mathbb{E}\left[\mathbb{E}\left((y - \mathbb{E}(y|\boldsymbol{x}))^2 | \boldsymbol{x}\right) \middle| \boldsymbol{z}\right]
\end{aligned}$$

$$\begin{aligned}&+ \mathbb{E}\left[\mathbb{E}\left((y - \mathbb{E}(y|\boldsymbol{x}))(\mathbb{E}(y|\boldsymbol{x}) - \mathbb{E}(\mathbb{E}(y|\boldsymbol{x})|\boldsymbol{z}))|\boldsymbol{x}\right)|\boldsymbol{z}\right] \\ &+ \mathbb{E}\left[\mathbb{E}\left((\mathbb{E}(y|\boldsymbol{x}) - \mathbb{E}(\mathbb{E}(y|\boldsymbol{x})|\boldsymbol{z}))^2|\boldsymbol{x}\right)\middle|\boldsymbol{z}\right] .\end{aligned}$$

For the first term, we have $\mathbb{E}\left((y - \mathbb{E}(y|\boldsymbol{x}))^2|\boldsymbol{x}\right) = \mathrm{Var}(y|\boldsymbol{x})$ by definition. The second term is zero: Since $\mathbb{E}(y|\boldsymbol{x}) - \mathbb{E}(\mathbb{E}(y|\boldsymbol{x})|\boldsymbol{z})$ is already a function of $\boldsymbol{x}$, we have

$$\begin{aligned}\mathbb{E}\left((y - \mathbb{E}(y|\boldsymbol{x}))(\mathbb{E}(y|\boldsymbol{x}) - \mathbb{E}(\mathbb{E}(y|\boldsymbol{x})|\boldsymbol{z}))|\boldsymbol{x}\right) &= \mathbb{E}\left((y - \mathbb{E}(y|\boldsymbol{x}))|\boldsymbol{x}\right)(\mathbb{E}(y|\boldsymbol{x}) - \mathbb{E}(\mathbb{E}(y|\boldsymbol{x})|\boldsymbol{z})) \\ &= (\mathbb{E}(y|\boldsymbol{x}) - \mathbb{E}(y|\boldsymbol{x}))(\mathbb{E}(y|\boldsymbol{x}) - \mathbb{E}(\mathbb{E}(y|\boldsymbol{x})|\boldsymbol{z})) \\ &= 0 .\end{aligned}$$

Finally, the third term is by definition equal to $\mathrm{Var}(\mathbb{E}(y|\boldsymbol{x})|\boldsymbol{z})$. Therefore,

$$\mathrm{Var}(y|\boldsymbol{z}) = \mathbb{E}(\mathrm{Var}(y|\boldsymbol{x})|\boldsymbol{z}) + \mathrm{Var}(\mathbb{E}(y|\boldsymbol{x})|\boldsymbol{z}) .$$

If ϕ is injective, then $\boldsymbol{x}$ is also a function of $\boldsymbol{z}$ and we obtain $\mathrm{Var}(y|\boldsymbol{z}) = \mathrm{Var}(y|\boldsymbol{x})$. □

Our main ingredient for analyzing analytic activation functions will be the univariate and multivariate identity theorems.

Theorem 5.H.2 (Identity theorem for univariate real analytic functions)**.** *Let $f : \mathbb{R} \to \mathbb{R}$ be analytic. If f is not the zero function, the set $\mathcal{N}(f) \coloneqq \{z \in \mathbb{R} \mid f(z) = 0\}$ has no accumulation point. In particular, $\mathcal{N}(f)$ is countable.*

Proof. See e.g. Corollary 1.2.7 in Krantz and Parks (2002) for the first statement. If $\mathcal{N}(f)$ is uncountable, there exists $k \in \mathbb{Z}$ such that $[k, k+1] \cap \mathcal{N}(f)$ is also uncountable, hence it contains a strictly increasing and bounded sequence of points, and the limit of this sequence is an accumulation point of $\mathcal{N}(f)$. □

Theorem 5.H.3 (Multivariate version of the identity theorem)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be analytic. If f is not the zero function, then $\mathcal{N}(f) \coloneqq \{\boldsymbol{x} \in \mathbb{R}^d \mid f(\boldsymbol{x}) = 0\}$ is a Lebesgue null set.*

Proof. Although less well-known than the univariate version, this multivariate version has been proven several times in the literature. For example, different proofs are given in Section 3.1.24 in Federer (1969), Lemma 1.2 in Nguyen (2015) and Proposition 0 in Mityagin (2015). More proof strategies have been hinted at in Lemma 5.22 in Kuchment (2015). Here, we provide an elementary proof following the proof strategy briefly mentioned at the beginning of Section 4.1 in Krantz and Parks (2002).

Let λ^d be the Lebesgue measure on $\mathbb{R}^d$ and let $\lambda \coloneqq \lambda^1$. We prove the statement by induction on $d \geq 1$. For $d = 1$, if $\lambda(\mathcal{N}(f)) > 0$, then $\mathcal{N}(f)$ is uncountable and hence $f \equiv 0$ by Theorem 5.H.2.

Now, let the statement hold for $d - 1 \geq 1$ and assume $\lambda^d(\mathcal{N}(f)) > 0$. For $a \in \mathbb{R}$, define the functions $f_a : \mathbb{R}^{d-1} \to \mathbb{R}, f_a(\boldsymbol{x}) = f(a, \boldsymbol{x})$. Then,

$$\begin{aligned}0 < \lambda^d(\mathcal{N}(f)) &= \int_{\mathbb{R}^p} \mathbb{1}_{\mathcal{N}(f)} \,\mathrm{d}\lambda^d = \int_{\mathbb{R}} \int_{\mathbb{R}^{d-1}} \mathbb{1}_{\mathcal{N}(f)}(a, \boldsymbol{x}) \,\mathrm{d}\lambda^{d-1}(\boldsymbol{x}) \,\mathrm{d}\lambda(a) \\ &= \int_{\mathbb{R}} \lambda^{d-1}(\mathcal{N}(f_a)) \,\mathrm{d}a .\end{aligned}$$

It follows that the set $U := \{x \in \mathbb{R} \mid \lambda^{d-1}(\mathcal{N}(f_x)) > 0\}$ satisfies $\lambda(U) > 0$. By induction, for all $x \in U$, we have $f_x \equiv 0$. Then, for all $\boldsymbol{x} \in \mathbb{R}^{d-1}$, we can conclude that the function $f_{\boldsymbol{x}} : \mathbb{R} \to \mathbb{R}, a \mapsto f(a, \boldsymbol{x})$ satisfies $\mathcal{N}(f_{\boldsymbol{x}}) \supseteq U$ and therefore $\lambda(\mathcal{N}(f_{\boldsymbol{x}})) \geq \lambda(U) > 0$. Using the case $d = 1$ again, it follows that $f_{\boldsymbol{x}} \equiv 0$ and therefore $f(a, \boldsymbol{x}) = 0$ for all $a \in \mathbb{R}$ and $\boldsymbol{x} \in \mathbb{R}^{d-1}$. □

The following lemma provides some intuition about null sets for readers less familiar with measure theory. Recall that a property Q holds almost surely with respect to a measure P on $\mathbb{R}^d$ if there exists a null set N, i.e. a measurable set with $P(N) = 0$, such that $Q(\boldsymbol{x})$ holds for all $\boldsymbol{x} \in \mathbb{R}^d \setminus N$.

Lemma 5.H.4. *Let λ^d be the Lebesgue measure on $\mathbb{R}^d$ and let P be a measure on $\mathbb{R}^d$ with a Lebesgue density function (i.e. a probability density function) p. Then, a null set with respect to λ^d is also a null set with respect to P. The converse holds if $p(\boldsymbol{x}) \neq 0$ for (almost) all $\boldsymbol{x}$.*

Proof. A well-known fact from measure and integration theory states that if a measure μ has a density with respect to a measure ν, then ν-null sets are also μ-null sets. Setting $\mu = P$ and $\nu = \lambda^d$ yields the first fact. If $p(\boldsymbol{x}) \neq 0$ for (almost) all $\boldsymbol{x}$, then $\mu := \lambda^d$ has density $1/p$ with respect to $\nu := P$, and hence the converse follows. □

Proposition 5.5.2 (Characterization of (COV) and (FRK))**.** *Consider the setting of Theorem 5.4.3 and let* FRK(n) *be the statement that (FRK) holds for n. Then,*

(i) Let $n \geq 1$. Then, FRK(n) *iff $P(\boldsymbol{z} \in U) = 0$ for all linear subspaces $U \subseteq \mathbb{R}^p$ of dimension* $\min\{n, p\} - 1$.
(ii) Let (MOM) hold. Then, (COV) holds iff $P(\boldsymbol{z} \in U) < 1$ for all linear subspaces $U \subseteq \mathbb{R}^p$ of dimension $p - 1$.

Assuming that (MOM) holds such that (COV) is well-defined, consider the following statements:

(a) FRK(p) *holds.*
(b) FRK(n) *holds for all $n \geq 1$.*
(c) (COV) holds.
(d) There exists a fixed deterministic matrix $\widetilde{\boldsymbol{X}} \in \mathbb{R}^{p \times d}$ such that $\det(\phi(\widetilde{\boldsymbol{X}})) \neq 0$.

We have (a) ⇔ (b) ⇒ (c) ⇒ (d). Furthermore, if $\boldsymbol{x} \in \mathbb{R}^d$ has a Lebesgue density and ϕ is analytic, then (a) – (d) are equivalent.

Proof. **Step 1: Prove (i) and (ii).**

(i) Denote the n (stochastically independent) rows of $\boldsymbol{Z}$ by $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$. First, assume $P(\boldsymbol{z} \in U) > 0$ for some subspace U of dimension $\min\{n, p\} - 1$. Then,

$$P\Big(\operatorname{rank}(\boldsymbol{Z}) \leq \min\{n, p\} - 1\Big) \geq P(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n \in U) = (P(\boldsymbol{z} \in U))^n > 0 \ .$$

For the converse, it suffices to consider the case $n \leq p$ since if $n > p$ and if an arbitrary $p \times p$ submatrix of $\boldsymbol{Z} \in \mathbb{R}^{n \times p}$ almost surely has full rank, then $\boldsymbol{Z}$ also almost surely has full rank. We prove the statement for $n \leq p$ by induction on n. For

$n = 1$, the claim is trivial. Thus, let $n > 1$ and let $\boldsymbol{z}_1, \dots, \boldsymbol{z}_n$ be the (stochastically independent) rows of $\boldsymbol{Z}$. Assume $P(\boldsymbol{z} \in U) = 0$ for all linear subspaces $U \subseteq \mathbb{R}^p$ of dimension $n-1$. Then, we also have $P(\boldsymbol{z} \in U) = 0$ for all linear subspaces $U \subseteq \mathbb{R}^p$ of dimension $n-2$ and by the induction hypothesis, we obtain that $\boldsymbol{z}_1, \dots, \boldsymbol{z}_{n-1}$ are almost surely linearly independent. Hence, almost surely, $\boldsymbol{z}_1, \dots, \boldsymbol{z}_n$ are linearly independent iff $\boldsymbol{z}_n \notin U_{n-1} \coloneqq \operatorname{Span}\{\boldsymbol{z}_1, \dots, \boldsymbol{z}_{n-1}\}$. Then, since $\boldsymbol{z}_n$ and U_{n-1} are stochastically independent,

$$\begin{aligned} P(\boldsymbol{Z} \in \mathbb{R}^{n \times p} \text{ has full rank}) &= P(\boldsymbol{z}_1, \dots, \boldsymbol{z}_n \text{ are linearly independent}) \\ &= P(\boldsymbol{z}_n \notin U_{n-1}) = \iint \mathbb{1}_{U_{n-1}^c}(\boldsymbol{z}_n) \, \mathrm{d}P_Z(\boldsymbol{z}_n) \, \mathrm{d}P(U_{n-1}) \\ &= \int P_Z(\boldsymbol{z}_n \notin U_{n-1}) \, \mathrm{d}P(U_{n-1}) = \int 1 \, \mathrm{d}P(U_{n-1}) = 1 \ . \end{aligned}$$

For the case $p \leq n$, (i) is also proven after Definition 1 in Mourtada (2022).

(ii) If (COV) does not hold, there exists a vector $0 \neq \boldsymbol{v} \in \mathbb{R}^p$ with $\boldsymbol{v}^\top \boldsymbol{\Sigma} \boldsymbol{v} = 0$, hence

$$0 = \boldsymbol{v}^\top \left(\mathbb{E} \boldsymbol{z} \boldsymbol{z}^\top \right) \boldsymbol{v} = \mathbb{E}(\boldsymbol{v}^\top \boldsymbol{z})^2 \ ,$$

and therefore $\boldsymbol{z}$ is almost surely orthogonal to $\boldsymbol{v}$. If U is the orthogonal complement of $\operatorname{Span}\{\boldsymbol{v}\}$ in $\mathbb{R}^p$, then $P(\boldsymbol{z} \in U) = 1$.

For the converse, if there exists a $(p-1)$-dimensional subspace U with $P(\boldsymbol{z} \in U) = 1$, then we can again take a vector $0 \neq \boldsymbol{v} \in \mathbb{R}^p$ that is orthogonal to U, reverse the above computation and obtain $\boldsymbol{v}^\top \boldsymbol{\Sigma} \boldsymbol{v} = 0$, hence (COV) does not hold.

Step 2: (a) $\Leftrightarrow$ (b). The implication (b) $\Rightarrow$ (a) is trivial and the implication (a) $\Rightarrow$ (b) follows immediately from (i).

Step 3: (b) $\Rightarrow$ (c). This also follows from (i) and (ii).

Step 4: (c) $\Rightarrow$ (d). We will prove by induction on n that for all $n \in \{0, \dots, p\}$, there exist $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^d$ such that $\phi(\boldsymbol{x}_1), \dots, \phi(\boldsymbol{x}_n)$ are linearly independent. For $n = 0$, the statement is trivial. Now assume that the statement holds for $\boldsymbol{x}_1, \dots, \boldsymbol{x}_{n-1}$, where $0 \leq n-1 \leq p-1$. Since (COV) holds, by (ii) the subspace $U \coloneqq \operatorname{Span}\{\phi(\boldsymbol{x}_1), \dots, \phi(\boldsymbol{x}_{n-1})\}$ satisfies $P(\phi(\boldsymbol{x}) \in U) < 1$, hence there is $\boldsymbol{x}_n$ such that $\boldsymbol{x}_n \notin U$ and for this choice, $\phi(\boldsymbol{x}_1), \dots, \phi(\boldsymbol{x}_n)$ are linearly independent.

Finally, the statement for $n = p$ yields the existence of $\boldsymbol{X} \in \mathbb{R}^{p \times d}$ such that $\phi(\boldsymbol{X})$ has linearly independent rows, which implies $\det(\phi(\boldsymbol{X})) \neq 0$.

Step 5: Analytic feature map. Assume that ϕ is analytic, $\boldsymbol{x}$ has a Lebesgue density and (d) holds. Let $n = p$. For $\widetilde{\boldsymbol{X}} \in \mathbb{R}^{p \times d}$, consider the analytic function $f(\widetilde{\boldsymbol{X}}) \coloneqq \det(\phi(\widetilde{\boldsymbol{X}}))$. (The determinant is analytic since it is a polynomial in the matrix entries.) Since (d) holds, Theorem 5.H.3 shows that $f(\widetilde{\boldsymbol{X}}) \neq 0$ for (Lebesgue-) almost all $\widetilde{\boldsymbol{X}}$. Since $\boldsymbol{x}$ has a Lebesgue density and $\boldsymbol{X}$ has independent $\boldsymbol{x}$-distributed rows, $\boldsymbol{X}$ has a Lebesgue density. Therefore, $f(\boldsymbol{X}) \neq 0$ almost surely over $\boldsymbol{X}$, hence (a) holds. □

Proposition 5.5.4 (Polynomial kernel). *Let $m, d \geq 1$ and $c > 0$. For $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathbb{R}^d$, define the polynomial kernel $k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \coloneqq (\boldsymbol{x}^\top \tilde{\boldsymbol{x}} + c)^m$. Then, there exists a feature map $\phi : \mathbb{R}^d \to \mathbb{R}^p$, $p \coloneqq \binom{m+d}{m}$, such that:*

(a) $k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \phi(\boldsymbol{x})^\top \phi(\tilde{\boldsymbol{x}})$ for all $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathbb{R}^d$, and

(b) if $\boldsymbol{x} \in \mathbb{R}^d$ has a Lebesgue density and we use $\boldsymbol{z} = \phi(\boldsymbol{x})$, then (FRK) is satisfied for all n.

Proof. Let $\mathcal{M} := \{(m_1, \ldots, m_{d+1}) \in \mathbb{N}_0^{d+1} \mid m_1 + \ldots + m_{d+1} = m\}$ and for $\boldsymbol{m} = (m_1, \ldots, m_{d+1}) \in \mathcal{M}$, let

$$C(\boldsymbol{m}) := \binom{m}{m_1 \; \ldots \; m_{d+1}}$$

be the corresponding multinomial coefficient. Then, $|\mathcal{M}| = \binom{m+d}{d} = p$. Define the feature map $\phi : \mathbb{R}^d \to \mathbb{R}^p$ by

$$\phi(\boldsymbol{x}) := \left(\sqrt{C(\boldsymbol{m})} z_1^{m_1} \cdots z_d^{m_d} \cdot (\sqrt{c})^{m_{d+1}}\right)_{\boldsymbol{m} \in \mathcal{M}} .$$

(a) We have

$$\begin{aligned} \phi(\boldsymbol{x})^\top \phi(\tilde{\boldsymbol{x}}) &= \sum_{\boldsymbol{m} \in \mathcal{M}} C(\boldsymbol{m}) (x_1 \tilde{x}_1)^{m_1} \cdots (x_d \tilde{x}_d)^{m_d} \cdot c^{m_{d+1}} \\ &= (x_1 \tilde{x}_1 + \ldots + x_d \tilde{x}_d + c)^m = k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) . \end{aligned}$$

(b) Assume that $\boldsymbol{x}$ has a Lebesgue density. Let U be an arbitrary $(p-1)$-dimensional linear subspace of $\mathbb{R}^p$. Then, there exists $0 \neq \boldsymbol{v} \in \mathbb{R}^p$ such that $U = (\operatorname{Span}\{\boldsymbol{v}\})^\perp$. Since the monomials $x_1^{m_1} \cdots x_d^{m_d}$ for $\boldsymbol{m} \in \mathcal{M}$ are all distinct, the polynomial

$$\boldsymbol{v}^\top \phi(\boldsymbol{x}) = \sum_{\boldsymbol{m} \in \mathcal{M}} \left(v_{\boldsymbol{m}} \sqrt{C(\boldsymbol{m}) c^{m_{d+1}}} x_1^{m_1} \cdots x_d^{m_d}\right)$$

is not the zero polynomial. By the identity theorem (Theorem 5.H.3), since $\boldsymbol{x}$ has a Lebesgue density and since polynomials are analytic,

$$P(\phi(\boldsymbol{x}) \in U) = P(\boldsymbol{v}^\top \phi(\boldsymbol{x}) = 0) = 0 .$$

Hence, Proposition 5.5.2 shows that (FRK) is satisfied for $n = p$ and hence for all n. □

We want to remark at this point that the proof strategy of Proposition 5.5.4, where the identity theorem is applied to the functions $\boldsymbol{v}^\top \phi(\boldsymbol{x})$ for all $0 \neq \boldsymbol{v} \in \mathbb{R}^p$, does not work for random feature maps: The statements

- For all $0 \neq \boldsymbol{v} \in \mathbb{R}^p$ for almost all $\boldsymbol{x}$ for almost all $\boldsymbol{\theta}$, $\boldsymbol{v}^\top \phi_{\boldsymbol{\theta}}(\boldsymbol{x}) \neq 0$
- For almost all $\boldsymbol{\theta}$ for all $0 \neq \boldsymbol{v} \in \mathbb{R}^p$ for almost all $\boldsymbol{x}$, $\boldsymbol{v}^\top \phi_{\boldsymbol{\theta}}(\boldsymbol{x}) \neq 0$

are not equivalent since in the first statement, the null set for $\boldsymbol{\theta}$ may depend on $\boldsymbol{v}$, and the union of the null sets for all $\boldsymbol{v}$ is an uncountable union. Perhaps the simplest counterexample is $n = p = 2$, $\boldsymbol{\theta} \sim \mathcal{N}(0, \boldsymbol{I}_2)$ and $\phi_{\boldsymbol{\theta}}(\boldsymbol{x}) := \boldsymbol{\theta}$, which satisfies the first but not the second statement. For our rescue, we can replace the uncountable analytic function family $(\boldsymbol{x} \mapsto \boldsymbol{v}^\top \phi_{\boldsymbol{\theta}}(\boldsymbol{x}))_{\boldsymbol{v} \in \mathbb{R}^p \setminus \{0\}}$ by the single analytic function $(\boldsymbol{\theta}, \boldsymbol{X}) \mapsto \det(\phi_{\boldsymbol{\theta}}(\boldsymbol{X}))$:

Proposition 5.5.5 (Random feature maps)**.** *Consider feature maps $\phi_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^p$ with (random) parameter $\boldsymbol{\theta} \in \mathbb{R}^q$. Suppose the map $(\boldsymbol{\theta}, \boldsymbol{x}) \mapsto \phi_{\boldsymbol{\theta}}(\boldsymbol{x})$ is analytic and that $\boldsymbol{\theta}$ and $\boldsymbol{x}$ are independent and have Lebesgue densities. If there exist fixed $\widetilde{\boldsymbol{\theta}} \in \mathbb{R}^q, \widetilde{\boldsymbol{X}} \in \mathbb{R}^{p \times d}$ with $\det(\phi_{\widetilde{\boldsymbol{\theta}}}(\widetilde{\boldsymbol{X}})) \neq 0$, then almost surely over $\boldsymbol{\theta}$, (FRK) holds for all n for $\boldsymbol{z} = \phi_{\boldsymbol{\theta}}(\boldsymbol{x})$.*

Proof. Consider the analytic map $(\boldsymbol{\theta}, \boldsymbol{X}) \mapsto \det(\phi_{\boldsymbol{\theta}}(\boldsymbol{X}))$. Suppose there exist $\widetilde{\boldsymbol{\theta}} \in \mathbb{R}^q$ and $\widetilde{\boldsymbol{X}} \in \mathbb{R}^{p \times d}$ with $\det(\phi_{\widetilde{\boldsymbol{\theta}}}(\widetilde{\boldsymbol{X}})) \neq 0$. Then, Theorem 5.H.3 tells us that $\det(\tilde{\phi}(\boldsymbol{\theta}, \boldsymbol{X})) \neq 0$ for almost all $(\boldsymbol{\theta}, \boldsymbol{X})$. This implies that for almost all $\boldsymbol{\theta}$, we have for almost all $\boldsymbol{X}$ that $\det(\phi_{\boldsymbol{\theta}}(\boldsymbol{X})) \neq 0$. Since by assumption, $\boldsymbol{\theta}$ has a density, this implies that almost surely over $\boldsymbol{\theta}$, there exists $\boldsymbol{X}$ such that $\det(\phi_{\boldsymbol{\theta}}(\boldsymbol{X})) \neq 0$. Since all $\phi_{\boldsymbol{\theta}}$ are analytic, the claim now follows using (d) $\Rightarrow$ (b) from Proposition 5.5.2. (The proof of (d) $\Rightarrow$ (a) $\Leftrightarrow$ (b) in Proposition 5.5.2 does not require (MOM).) $\square$

5.H.2 Random Networks with Biases

In order to prove (FRK) for random deep neural networks, we pursue slightly different approaches for networks with bias (this section) and without bias (Section 5.H.3). In both approaches, we consider a property related to the diversity of the $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ and proceed as follows:

(1) **Projections**: Ensure that random projections of the $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ almost surely preserve the diversity property, such that it is sufficient to consider the case $d = 1$.
(2) **Propagation**: Using the projection result from (1), prove that if the inputs to a layer have the diversity property, then the outputs also have the diversity property almost surely over the random parameters of the layer.
(3) **Independence**: Prove that if the inputs to the last layer have the diversity property and $n = p$, then the outputs of the last layer are almost surely linearly independent.

Our main tools will be the identity theorems for analytic functions (Theorem 5.H.2 and Theorem 5.H.3), expanding σ into its power series around a point and the Leibniz formula for the determinant of a $n \times n$ matrix, which is based on the permutation group S_n on $[n]$.

We consider two diversity properties:

(a) The first property is that $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are distinct. This is the weakest possible diversity property. However, it cannot always be used for networks without (random) biases: For example, if σ is an even function and $\boldsymbol{x}_i = -\boldsymbol{x}_j$ for some $i \neq j$, the propagation property is violated. As another example, if $\sigma(0) = 0$ and $\boldsymbol{x}_i = \boldsymbol{0}$ for some i, the independence property is violated.
(b) The second property, which works for networks with and without bias, is the property that $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are independent nonatomic random variables.

Using the first property yields shorter proofs and a slightly stronger theorem for networks with bias, Theorem 5.H.9. If we only care about probability distributions P_X on $\boldsymbol{x}$ that almost surely generate distinct $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, these probability distributions are exactly the nonatomic distributions. Hence from the viewpoint of probability distributions P_X on $\boldsymbol{x}$, which we take in the main part of the paper, the first property (a) does not provide a benefit over the second property (b) except for the shorter proofs.

The advantage of having biases is in being able to choose the point in which σ is Taylor-expanded. The following lemma shows that this choice enables us to make certain coefficients of the Taylor expansion nonzero:

Lemma 5.H.5. *Let $m \geq 1$. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be analytic and not a polynomial of degree less than m. Then, there exists $b \in \mathbb{R}$ such that the Taylor expansion*

$$\sigma(x) = \sum_{k=0}^{\infty} a_k (x-b)^k$$

of σ around b satisfies $a_0, \ldots, a_m \neq 0$.

Proof. Since σ is not a polynomial of degree less than m, neither of the derivatives $\sigma^{(0)}, \ldots, \sigma^{(m)}$ is the zero function. Since all of these derivatives are analytic, the set

$$\bigcup_{k=0}^{m} \{b \in \mathbb{R} \mid \sigma^{(k)}(b) = 0\}$$

is (by Theorem 5.H.3) a finite union of Lebesgue null sets and hence a Lebesgue null set. Hence, there exists $b \in \mathbb{R}$ such that $\sigma^{(0)}(b) \neq 0, \ldots, \sigma^{(m)}(b) \neq 0$. This implies that the corresponding coefficients $a_0, \ldots, a_m$ in the Taylor expansion around b are nonzero. □

We now prove our three-step program (1) – (3) from above for the distinctness property.

Lemma 5.H.6 (Projections of distinct variables)**.** *If $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ are distinct, there exists $\boldsymbol{u} \in \mathbb{R}^d$ such that $\boldsymbol{u}^\top \boldsymbol{x}_1, \ldots, \boldsymbol{u}^\top \boldsymbol{x}_d$ are also distinct.*

Proof. We essentially follow the corresponding proof in Lemma 4.3 in Nguyen and Hein (2017). By the identity theorem (Theorem 5.H.3), the functions

$$f_{ij} : \mathbb{R}^d \to \mathbb{R}, \boldsymbol{u} \mapsto \boldsymbol{u}^\top \boldsymbol{x}_i - \boldsymbol{u}^\top \boldsymbol{x}_j$$

for $i, j \in [n], i \neq j$ are nonzero almost everywhere, hence there exists $\boldsymbol{u} \in \mathbb{R}^p$ such that $\boldsymbol{u}^\top \boldsymbol{x}_1, \ldots, \boldsymbol{u}^\top \boldsymbol{x}_n$ are all distinct. □

Lemma 5.H.7 (Propagation of distinct variables)**.** *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be analytic and non-constant. If $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ are distinct, then for (Lebesgue-) almost all $(\boldsymbol{W}, \boldsymbol{b}) \in \mathbb{R}^{p \times d} \times \mathbb{R}^p$, the vectors $\sigma(\boldsymbol{W}\boldsymbol{x}_i + \boldsymbol{b})$ (with σ applied element-wise) are also distinct.*

Proof. **Step 1: Bias.** By assumption, σ is not a polynomial of degree less than $m = 1$. By Lemma 5.H.5, there exist $(a_k)_{k \geq 0}$, $b \in \mathbb{R}$ and $\varepsilon > 0$ such that $a_0, a_1 \neq 0$ and for all $x \in \mathbb{R}$ with $|x - b| < \varepsilon$,

$$\sigma(x) = \sum_{k=0}^{\infty} a_k (x-b)^k \ .$$

Step 2: Weight. By Lemma 5.H.6, we can choose $\boldsymbol{u} \in \mathbb{R}^d$ such that $\boldsymbol{u}^\top \boldsymbol{x}_1, \ldots, \boldsymbol{u}^\top \boldsymbol{x}_n$ are distinct. Now, consider $i, j \in [n]$ with $i \neq j$. The function

$$f_{ij} : \mathbb{R} \to \mathbb{R}, \lambda \mapsto \sigma(\lambda \boldsymbol{u}^\top \boldsymbol{x}_i + b) - \sigma(\lambda \boldsymbol{u}^\top \boldsymbol{x}_j + b)$$

satisfies

$$f_{ij}(\lambda) = \sum_{k=0}^{\infty} a_k ((\boldsymbol{u}^\top \boldsymbol{x}_i)^k - (\boldsymbol{u}^\top \boldsymbol{x}_j)^k) \lambda^k$$

for sufficiently small $|\lambda|$. Here, the coefficient $a_k((\boldsymbol{u}^\top \boldsymbol{x}_i)^k - (\boldsymbol{u}^\top \boldsymbol{x}_j)^k)$ is nonzero for $k = 1$, and hence f_{ij} is not the zero function. Using the identity theorem again (as in the proof of Lemma 5.H.6), we find that there exists $\lambda \in \mathbb{R}$ with $f_{ij}(\lambda) \neq 0$ for all $i, j \in [n]$ with $i \neq j$.

Step 3: Generalization. Now, choose

$$\boldsymbol{W} := \begin{pmatrix} \lambda \boldsymbol{u}^\top \\ \vdots \\ \lambda \boldsymbol{u}^\top \end{pmatrix} \in \mathbb{R}^{p \times d}, \qquad \boldsymbol{b} := \begin{pmatrix} b \\ \vdots \\ b \end{pmatrix} \in \mathbb{R}^p .$$

Then, by construction, the first components $(\sigma(\boldsymbol{W}\boldsymbol{x}_i + \boldsymbol{b}))_1$ for $i \in [n]$ are distinct. Hence, the analytic function

$$(\boldsymbol{W}, \boldsymbol{b}) \mapsto \prod_{\substack{i,j \in [n] \\ i \neq j}} ((\sigma(\boldsymbol{W}\boldsymbol{x}_i + \boldsymbol{b}))_1 - (\sigma(\boldsymbol{W}\boldsymbol{x}_j + \boldsymbol{b}))_1)$$

is not the zero function. By the identity theorem (Theorem 5.H.3), for (Lebesgue-) almost all $(\boldsymbol{W}, \boldsymbol{b})$, the first components of the vectors $\sigma(\boldsymbol{W}\boldsymbol{x}_i + \boldsymbol{b})$ are distinct, and therefore also the vectors themselves are distinct. □

Lemma 5.H.8 (Independence from distinct variables). *Let $p, d \geq 1$. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be analytic but not a polynomial of degree less than $p-1$. If $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p \in \mathbb{R}^d$ are distinct, then for (Lebesgue-) almost all $\boldsymbol{W} \in \mathbb{R}^{p \times d}$ and $\boldsymbol{b} \in \mathbb{R}^p$, the vectors $\sigma(\boldsymbol{W}\boldsymbol{x}_1 + \boldsymbol{b}), \ldots, \sigma(\boldsymbol{W}\boldsymbol{x}_p + \boldsymbol{b})$ are linearly independent.*

Proof. **Step 1: Preparation.** By Lemma 5.H.5, there exist $(a_k)_{k \geq 0}$, $b \in \mathbb{R}$ and $\varepsilon > 0$ such that $a_0, \ldots, a_{p-1} \neq 0$ and for all $x \in \mathbb{R}$ with $|x - b| < \varepsilon$,

$$\sigma(x) = \sum_{k=0}^{\infty} a_k (x - b)^k .$$

By Lemma 5.H.6, there exists $\boldsymbol{u} \in \mathbb{R}^d$ such that $x_i := \boldsymbol{u}^\top \boldsymbol{x}_i$ for $i \in [p]$ are all distinct. Using the fact that Vandermonde matrices of distinct x_i are invertible and using the Leibniz formula for the determinant (with the permutation group S_p on $[p]$), we obtain

$$D_x := \sum_{\pi \in S_p} \operatorname{sgn}(\pi) \prod_{i=1}^{p} x_{\pi(i)}^{i-1} = \det \begin{pmatrix} x_1^0 & \cdots & x_p^0 \\ \vdots & \ddots & \vdots \\ x_1^{p-1} & \cdots & x_p^{p-1} \end{pmatrix} \neq 0 . \tag{5.11}$$

Step 2: Determinant expansion. Define the analytic function

$$f : \mathbb{R}^p \to \mathbb{R}, \boldsymbol{w} \mapsto \det \begin{pmatrix} \sigma(w_1 x_1 + b) & \cdots & \sigma(w_p x_1 + b) \\ \vdots & \ddots & \vdots \\ \sigma(w_1 x_p + b) & \cdots & \sigma(w_p x_p + b) \end{pmatrix} .$$

For small enough $\|\boldsymbol{w}\|_2$, we can use the Leibniz formula for the determinant to write

$$f(\boldsymbol{w}) = \sum_{\pi \in S_p} \operatorname{sgn}(\pi) \prod_{i=1}^{p} \sum_{k=0}^{\infty} a_k (w_i x_{\pi(i)})^k$$

$$= \sum_{k_1,\dots,k_p \geq 0} \sum_{\pi \in S_p} \operatorname{sgn}(\pi) \prod_{i=1}^p a_{k_i} w_i^{k_i} x_{\pi(i)}^{k_i}$$

$$= \sum_{k_1,\dots,k_p \geq 0} \left(\prod_{i=1}^p a_{k_i}\right) \left(\sum_{\pi \in S_p} \operatorname{sgn}(\pi) \prod_{i=1}^p x_{\pi(i)}^{k_i}\right) w_1^{k_1} \cdots w_p^{k_p} .$$

For $k_i := i - 1$, we find the coefficient of $w_1^{k_1} \cdots w_p^{k_p}$ to be

$$\left(\prod_{i=1}^p a_{k_i}\right) \left(\sum_{\pi \in S_p} \operatorname{sgn}(\pi) \prod_{i=1}^p x_{\pi(i)}^{k_i}\right) \overset{(5.11)}{=} \left(\prod_{i=1}^p a_{k_i}\right) \cdot D_x \neq 0 ,$$

hence f is not the zero function and there exists $\boldsymbol{w} \in \mathbb{R}^p$ such that $f(\boldsymbol{w}) \neq 0$.

Step 3: Generalization. Consider the analytic function

$$g(\boldsymbol{W}, \boldsymbol{b}) := \det \begin{pmatrix} \sigma(\boldsymbol{W}\boldsymbol{x}_1 + \boldsymbol{b})^\top \\ \vdots \\ \sigma(\boldsymbol{W}\boldsymbol{x}_p + \boldsymbol{b})^\top \end{pmatrix} .$$

When setting $\boldsymbol{W} := \boldsymbol{w}\boldsymbol{u}^\top \in \mathbb{R}^{p \times d}$ and $\boldsymbol{b} := (b, \dots, b)^\top \in \mathbb{R}^p$, we obtain from Step 2 that $g(\boldsymbol{W}, \boldsymbol{b}) = f(\boldsymbol{w}) \neq 0$, hence g is not the zero function. But then, by the identity theorem (Theorem 5.H.3), g is nonzero for (Lebesgue-) almost all $(\boldsymbol{W}, \boldsymbol{b})$. □

Theorem 5.H.9. *Let $p, d \geq 1$, let $\sigma : \mathbb{R} \to \mathbb{R}$ be analytic but not a polynomial of degree less than $\max\{1, p-1\}$ and let $\boldsymbol{x}_1, \dots, \boldsymbol{x}_p \in \mathbb{R}^d$ be distinct. Let $L \geq 1$ and $d_0 := d, d_1, \dots, d_{L-1} \geq 1, d_L := p$. For $l \in \{0, \dots, l-1\}$, let $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d_{l+1} \times d_l}$ and $\boldsymbol{b}^{(l)} \in \mathbb{R}^{d_{l+1}}$ be random variables such that $\boldsymbol{\theta} := (\boldsymbol{W}^{(0)}, \dots, \boldsymbol{W}^{(L-1)}, \boldsymbol{b}^{(0)}, \dots, \boldsymbol{b}^{(L-1)})$ has a Lebesgue density. Consider the random feature map given by*

$$\phi_{\boldsymbol{\theta}}(\boldsymbol{x}^{(0)}) := \boldsymbol{x}^{(L)}, \quad \textit{where} \quad \boldsymbol{x}^{(l+1)} := \sigma\left(\boldsymbol{W}^{(l)}\boldsymbol{x}^{(l)} + \boldsymbol{b}^{(l)}\right) .$$

Then, almost surely over $\boldsymbol{\theta}$, $\phi_{\boldsymbol{\theta}}(\boldsymbol{x}_1), \dots, \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_p)$ are linearly independent.

Proof. By Lemma 5.H.4, it suffices to consider the case where $\boldsymbol{\theta}$ has a standard normal distribution, since a standard normal distribution has a nonzero probability density everywhere. Especially, in this case, all weights and biases are independent. Using Lemma 5.H.7 and that σ is non-constant, it follows by induction on $l \in \{0, \dots, L-1\}$ that $\boldsymbol{x}_1^{(l)}, \dots, \boldsymbol{x}_p^{(l)}$ are distinct almost surely over $\boldsymbol{\theta}$. But if $\boldsymbol{x}_1^{(L-1)}, \dots, \boldsymbol{x}_p^{(L-1)}$ are distinct, then $\boldsymbol{x}_1^{(L)}, \dots, \boldsymbol{x}_p^{(L)}$ are linearly independent almost surely over $\boldsymbol{\theta}$ by Lemma 5.H.8, which is what we wanted to show. □

5.H.3 Random Networks without Biases

As discussed at the beginning of Section 5.H.2, we will now consider the property of having independent nonatomic random variables $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$. We will again consider projection and propagation lemmas first.

Lemma 5.H.10 (Projections of nonatomic random variables)**.** *A random variable $\boldsymbol{x} \in \mathbb{R}^d$ is nonatomic iff for (Lebesgue-) almost all $\boldsymbol{u} \in \mathbb{R}^d$, $\boldsymbol{u}^\top \boldsymbol{x}$ is nonatomic.*

Proof. **Step 1: Decompose into subspace contributions.** For $k \in \{0, \ldots, d\}$, let

$$\mathcal{S}_k := \{\boldsymbol{w} + V \mid \boldsymbol{w} \in \mathbb{R}^d, V \text{ linear subspace of } \mathbb{R}^d, \dim V = k\}$$

be the set of k-dimensional affine subspaces of $\mathbb{R}^d$. Let $\mathcal{A}_0 := \{A \in \mathcal{S}_0 \mid P_X(A) > 0\}$, which corresponds to the set of atoms of P_X. We then recursively define "minimally contributing subspaces" for $k \in [d]$:

$$\mathcal{A}_k := \{A \in \mathcal{S}_k \mid P_X(A) > 0 \text{ and for all } \tilde{A} \in \mathcal{S}_{k-1} \text{ with } \tilde{A} \subseteq A,\ P_X(\tilde{A}) = 0\}\ .$$

We can define corresponding sets of "annihilating" vectors as follows: For $A = \boldsymbol{w} + V \in \mathcal{S}_k$, let $A^\perp := V^\perp = \{\boldsymbol{u} \in \mathbb{R}^d \mid \text{for all } \boldsymbol{v} \in V,\ \boldsymbol{u}^\top \boldsymbol{v} = 0\}$. This is well-defined since it is independent of the choice of $\boldsymbol{w}$. Define

$$\begin{aligned} \mathcal{U} &:= \bigcup_{k=0}^{d} \bigcup_{A \in \mathcal{A}_k} A^\perp \\ \mathcal{N} &:= \{\boldsymbol{u} \in \mathbb{R}^d \mid \boldsymbol{u}^\top \boldsymbol{x} \text{ is not nonatomic}\}\ . \end{aligned}$$

We want to show $\mathcal{U} = \mathcal{N}$.

Step 2: Show $\mathcal{U} \subseteq \mathcal{N}$. Let $A = \boldsymbol{w} + V \in \mathcal{A}_k$ for some $k \in \{0, \ldots, d\}$ and let $\boldsymbol{u} \in A^\perp$. Then,

$$0 < P_X(A) = P(\boldsymbol{x} \in A) \leq P(\boldsymbol{u}^\top \boldsymbol{x} \in \{\boldsymbol{u}^\top(\boldsymbol{w} + \boldsymbol{v}) \mid \boldsymbol{v} \in V\}) = P(\boldsymbol{u}^\top \boldsymbol{x} = \boldsymbol{u}^\top \boldsymbol{w})\ ,$$

which shows $\boldsymbol{u} \in \mathcal{N}$.

Step 3: Show $\mathcal{N} \subseteq \mathcal{U}$. Let $\boldsymbol{u} \in \mathcal{N}$, i.e. there exists $a \in \mathbb{R}$ such that $P(\boldsymbol{u}^\top \boldsymbol{x} = a) > 0$. Define $A := \{\boldsymbol{v} \in \mathbb{R}^d \mid \boldsymbol{u}^\top \boldsymbol{v} = a\}$. Then, A is an affine subspace of $\mathbb{R}^d$ and we have $P_X(A) > 0$ by construction of A. Among all affine subspaces $\tilde{A}$ of A with $P_X(\tilde{A} > 0)$, there exists one with minimal dimension. This subspace then satisfies $\tilde{A} \in \mathcal{A}_{\dim \tilde{A}}$ and it is not hard to show that $\boldsymbol{u} \in A^\perp \subseteq \tilde{A}^\perp$, hence $\boldsymbol{u} \in \mathcal{U}$.

Step 4: For all $k \in \{0, \ldots, d\}$, $\mathcal{A}_k$ is countable. To derive a contradiction, assume that $\mathcal{A}_k$ is uncountable. Then, there exists $\varepsilon > 0$ such that

$$\mathcal{A}_{k,\varepsilon} := \{A \in \mathcal{A}_k \mid P_X(A) \geq \varepsilon\}$$

is also uncountable. Pick an integer $n > 1/\varepsilon$ and pick n distinct sets $A_1, \ldots, A_n \in \mathcal{A}_{k,\varepsilon}$. We will show by induction on $l \in [n]$ that $\tilde{A}_l := A_1 \cup \ldots \cup A_l$ satisfies

$$P_X(\tilde{A}_l) = P_X(A_1) + \ldots + P_X(A_l)\ ,$$

which will then yield the contradiction

$$1 \geq P_X(\tilde{A}_l) = P_X(A_1) + \ldots + P_X(A_n) \geq n\varepsilon > 1\ .$$

Obviously, $P_X(\tilde{A}_1) = P_X(A_1)$. Assuming that the statement holds for $l \in [n-1]$, we first derive

$$P_X(\tilde{A}_{l+1}) = P_X(\tilde{A}_l \cup A_{l+1}) = P_X(\tilde{A}_l) + P_X(A_{l+1}) - P_X(\tilde{A}_l \cap A_{l+1})$$

$$= P_X(A_1) + \ldots + P_X(A_{l+1}) - P_X(\tilde{A}_l \cap A_{l+1}) \ .$$

For $i \neq j$, the intersection $A_i \cap A_j$ of two distinct k-dimensional affine subspaces is either empty or an affine subspace of dimension less than k, hence the definition of $\mathcal{A}_k$ yields $P_X(A_i \cap A_j) = 0$. Therefore,

$$\begin{aligned} P_X(\tilde{A}_l \cap A_{l+1}) &= P_X((A_1 \cap A_{l+1}) \cup \ldots \cup (A_l \cap A_{l+1})) \\ &\leq P_X(A_1 \cap A_{l+1}) + \ldots + P_X(A_l \cap A_{l+1}) \\ &= 0 + \ldots + 0 = 0 \ , \end{aligned}$$

which concludes the induction.

Step 5: Conclusion. Let $A \in \mathcal{S}_k$, then $A^\perp$ is a $(d-k)$-dimensional linear subspace of $\mathbb{R}^d$. If $\boldsymbol{x}$ is not nonatomic, the set $\mathcal{A}_0$ is not empty, and hence

$$\mathcal{N} = \mathcal{U} \supseteq \bigcup_{A \in \mathcal{A}_0} A^\perp = \mathbb{R}^d \ ,$$

which means that $\mathcal{N}$ is not a Lebesgue null set. Conversely, if $\boldsymbol{x}$ is nonatomic, the set $\mathcal{A}_0$ is empty, and therefore

$$\mathcal{N} = \mathcal{U} = \bigcup_{k=1}^{d} \bigcup_{A \in \mathcal{A}_k} A^\perp$$

is (by Step 4) a countable union of proper affine subspaces of $\mathbb{R}^d$, all of which are Lebesgue null sets. Therefore, $\mathcal{N}$ is a Lebesgue null set. □

Lemma 5.H.11 (Propagation of nonatomic random variables)**.** *Let the random variable $\boldsymbol{x} \in \mathbb{R}^d$ be nonatomic.*

(a) For all $p \geq 1$ and (Lebesgue-) almost all $\boldsymbol{W} \in \mathbb{R}^{p \times d}$, $\boldsymbol{W}\boldsymbol{x}$ is nonatomic.
(b) For all $\boldsymbol{b} \in \mathbb{R}^d$, $\boldsymbol{x} + \boldsymbol{b}$ is nonatomic.
(c) If $\sigma : \mathbb{R} \to \mathbb{R}$ is analytic and not constant, then $\sigma(\boldsymbol{x}) \in \mathbb{R}^d$ is nonatomic, where σ is applied element-wise.

Proof.

(a) By Lemma 5.H.10, the set $\mathcal{N} := \{\boldsymbol{u} \in \mathbb{R}^d \mid \boldsymbol{u}^\top \boldsymbol{x} \text{ is not nonatomic}\}$ is a Lebesgue null set. If $\boldsymbol{w}_1$ is the first row of $\boldsymbol{W}$, then clearly,

$$\{\boldsymbol{W} \in \mathbb{R}^{p \times d} \mid \boldsymbol{W}\boldsymbol{x} \text{ is not nonatomic}\} \subseteq \{\boldsymbol{W} \in \mathbb{R}^{p \times d} \mid \boldsymbol{w}_1 \in \mathcal{N}\} \ ,$$

where the right-hand side is a Lebesgue null set. This proves the claim.

(b) This is trivial.

(c) Let $\boldsymbol{z} \in \mathbb{R}^d$. Since the function $\mathbb{R} \to \mathbb{R}, x \mapsto \sigma(x) - z_i$ is analytic and not the zero function by assumption, its zero set $\sigma^{-1}(\{z_i\})$ is countable by the identity theorem, Theorem 5.H.2. Therefore, the set

$$\sigma^{-1}(\{\boldsymbol{z}\}) = \{\tilde{\boldsymbol{x}} \in \mathbb{R}^d \mid \sigma(\tilde{\boldsymbol{x}}) = \boldsymbol{z}\} = \sigma^{-1}(\{z_1\}) \times \ldots \times \sigma^{-1}(\{z_d\})$$

is also countable. Thus, since $\boldsymbol{x}$ is nonatomic, we obtain

$$P(\sigma(\boldsymbol{x}) = \boldsymbol{z}) = P(\boldsymbol{x} \in \sigma^{-1}(\{\boldsymbol{z}\})) = \sum_{\tilde{\boldsymbol{x}} \in \sigma^{-1}(\{\boldsymbol{z}\})} P(\boldsymbol{x} = \tilde{\boldsymbol{x}}) = \sum_{\tilde{\boldsymbol{x}} \in \sigma^{-1}(\{\boldsymbol{z}\})} 0 = 0 \ . \qquad \square$$

For proving independence from nonatomic random variables, we need some preparation. In the proof of the independence result for the case with biases (Lemma 5.H.8), a Vandermonde matrix appeared. In the case of networks without bias, we will not be able to use Lemma 5.H.5 to control the power series coefficients of σ, which requires us to treat Vandermonde matrices with more general exponents.

Lemma 5.H.12 (Random Vandermonde-type matrices)**.** *For $n \geq 1$, let $x_1, \ldots, x_n$ be independent nonatomic $\mathbb{R}$-valued random variables and let $k_1, \ldots, k_n$ be distinct non-negative integers. Then, the random Vandermonde-type matrix*

$$\boldsymbol{V} \coloneqq \boldsymbol{V}_{k_1,\ldots,k_n}(x_1, \ldots, x_n) \coloneqq \begin{pmatrix} x_1^{k_1} & \ldots & x_n^{k_1} \\ \vdots & \ddots & \vdots \\ x_1^{k_n} & \ldots & x_n^{k_n} \end{pmatrix}$$

is invertible with probability one.

Proof. By swapping rows of $\boldsymbol{V}$, we can assume without loss of generality that $0 \leq k_1 < k_2 < \ldots < k_n$. We prove the statement by induction on n. For $n = 1$, $\boldsymbol{V}$ is invertible whenever $x_1 \neq 0$, and this happens with probability one. Now assume that the statement holds for $n - 1 \geq 1$. For $i \in [n]$, define the submatrices

$$\widehat{\boldsymbol{V}}_i \coloneqq \boldsymbol{V}_{k_1,\ldots,k_{n-1}}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \ .$$

Since the statement holds for $n - 1$, $\widehat{\boldsymbol{V}}_n$ is invertible with probability one. Now, fix any such $x_1, \ldots, x_{n-1}$ where $\widehat{\boldsymbol{V}}_n$ is invertible. Especially, $C \coloneqq \det(\widehat{\boldsymbol{V}}_n)$ is a non-zero constant. We will show that $\boldsymbol{V}$ is then invertible almost surely over x_n. For this, we compute the determinant of $\boldsymbol{V}$ using the Laplace expansion with respect to the last row of $\boldsymbol{V}$ as

$$f(x_n) \coloneqq \det(\boldsymbol{V}) = \sum_{i=1}^{n} (-1)^{i+n} x_i^{k_n} \det(\widehat{\boldsymbol{V}}_i) = C x_n^{k_n} + \sum_{i=1}^{n-1} (-1)^{i+n} x_i^{k_n} \det(\widehat{\boldsymbol{V}}_i) \ .$$

By the Leibniz formula, for $i \in [n-1]$, $\det(\widehat{\boldsymbol{V}}_i)$ is a polynomial in x_n of degree $\leq k_{n-1} < k$. Hence, f is a nonzero polynomial in x_n of degree k_n and has at most k_n zeros. Since any finite set is a null set with respect to a nonatomic distribution, it follows that $\det(\boldsymbol{V}) \neq 0$ almost surely over x_n. Since the assumptions on $x_1, \ldots, x_{n-1}$ are also satisfied almost surely, the statement follows. □

As in the case of networks with biases, we will prove the independence result by first reducing it to the case $d = 1$. Since we also want to perform this reduction for random Fourier features in Proposition 5.I.1, we state the reduction to the $d = 1$ case as a separate result, Lemma 5.H.14, and define the $d = 1$ case in the following definition.

Definition 5.H.13 (Non-degenerate function)**.** Let $p, q \geq 1$. We call a function $f : \mathbb{R}^q \to \mathbb{R}^p$ *non-degenerate* if it is analytic and for arbitrary independent $\mathbb{R}$-valued nonatomic random variables $x_1, \ldots, x_p$, there almost surely exists $\boldsymbol{w} = \boldsymbol{w}_{x_1,\ldots,x_p} \in \mathbb{R}^q$ with

$$\det \begin{pmatrix} f(\boldsymbol{w}x_1)^\top \\ \vdots \\ f(\boldsymbol{w}x_p)^\top \end{pmatrix} \neq 0 \ . \qquad ◀$$

Lemma 5.H.14. *Let $f : \mathbb{R}^q \to \mathbb{R}^p$ be non-degenerate, let $\boldsymbol{W} \in \mathbb{R}^{q\times d}$ be a random variable with a Lebesgue density and let $\boldsymbol{x}_1, \dots, \boldsymbol{x}_p \in \mathbb{R}^d$ be independent nonatomic random variables. Then,*

$$\det \begin{pmatrix} f(\boldsymbol{W}\boldsymbol{x}_1)^\top \\ \vdots \\ f(\boldsymbol{W}\boldsymbol{x}_p)^\top \end{pmatrix} \neq 0$$

almost surely over $\boldsymbol{W}$ and $\boldsymbol{x}_1, \dots, \boldsymbol{x}_p$.

Proof. By Lemma 5.H.10, there exists $\boldsymbol{u} \in \mathbb{R}^d$ such that for all $i \in [p]$, $x_i \coloneqq \boldsymbol{u}^\top \boldsymbol{x}_i \in \mathbb{R}$ is nonatomic. Obviously, $x_1, \dots, x_p$ are independent. Fix $x_1, \dots, x_p$ such that $\boldsymbol{w} = \boldsymbol{w}_{x_1,\dots,x_p} \in \mathbb{R}^q$ as in Definition 5.H.13 exists, which is true with probability one since f is non-degenerate. Then, for $\widetilde{\boldsymbol{W}} \coloneqq \boldsymbol{w}\boldsymbol{u}^\top$, we have

$$g(\widetilde{\boldsymbol{W}}) \coloneqq \det \begin{pmatrix} f(\widetilde{\boldsymbol{W}}\boldsymbol{x}_1)^\top \\ \vdots \\ f(\widetilde{\boldsymbol{W}}\boldsymbol{x}_p)^\top \end{pmatrix} = \det \begin{pmatrix} f(\boldsymbol{w}x_1)^\top \\ \vdots \\ f(\boldsymbol{w}x_p)^\top \end{pmatrix} \neq 0 \ .$$

Since g is a non-zero analytic function, Theorem 5.H.3 shows that g is only zero on a Lebesgue null set, and this null set is also a null set with respect to the distribution of $\boldsymbol{W}$ since $\boldsymbol{W}$ has a Lebesgue density. Hence, $g(\boldsymbol{W}) \neq 0$ almost surely over $\boldsymbol{W}$. □

The following lemma proves the $d = 1$ version of the independence result, which can then be upgraded to the general case using Lemma 5.H.14.

Lemma 5.H.15 (Independence from nonatomic random variables)**.** *Let $p \geq 1$. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be analytic and not a polynomial with less than p nonzero coefficients. Then, the elementwise application function $f : \mathbb{R}^p \to \mathbb{R}^p, \boldsymbol{x} \mapsto (\sigma(x_1), \dots, \sigma(x_p))^\top$ is non-degenerate in the sense of Definition 5.H.13.*

Proof. Let $x_1, \dots, x_p$ be independent scalar nonatomic random variables.

Step 1: Power series coefficients. Since σ is analytic, there exists $\varepsilon > 0$ and coefficients $(a_k)_{k\geq 0}$ such that

$$\sigma(z) = \sum_{k=0}^{\infty} a_k z^k$$

for $z \in \mathbb{R}$ with $|z| < \varepsilon$. Let $K \coloneqq \{k \geq 0 \mid a_k \neq 0\}$. Then, $|K| \geq p$: Assume that $|K| \leq p - 1$, then the polynomial

$$h : \mathbb{R} \to \mathbb{R}, z \mapsto \sum_{k\in K} a_k z^k$$

equals σ on $(-\varepsilon, \varepsilon)$. Hence, the function $g \coloneqq \sigma - h$ is zero on $(-\varepsilon, \varepsilon)$. By Theorem 5.H.2, g is the zero function and hence $\sigma = h$ is a polynomial with less than p nonzero coefficients, which we assumed not to be the case.

Step 2: Condition on the x_i. By Step 1, we can choose indices $k_1 < k_2 < \ldots < k_p$ with $k_1, \ldots, k_p \in K$. Then, by Lemma 5.H.12 and the Leibniz formula for the determinant, we have

$$D_x := \sum_{\pi \in S_p} \operatorname{sgn}(\pi) x_{\pi(1)}^{k_1} \cdot \ldots \cdot x_{\pi(p)}^{k_p} = \det \begin{pmatrix} x_1^{k_1} & \ldots & x_p^{k_1} \\ \vdots & \ddots & \vdots \\ x_1^{k_p} & \ldots & x_p^{k_p} \end{pmatrix} \neq 0 \tag{5.12}$$

with probability one.

Step 3: Determinant power series. Now, fix a realization of $x_1, \ldots, x_p$ such that (5.12) holds. For $\boldsymbol{w} \in \mathbb{R}^p$ with sufficiently small $\|\boldsymbol{w}\|_\infty$, we can write

$$\begin{aligned} g(\boldsymbol{w}) &:= \det \begin{pmatrix} f(\boldsymbol{w}x_1)^\top \\ \vdots \\ f(\boldsymbol{w}x_p)^\top \end{pmatrix} \\ &= \det \begin{pmatrix} \sigma(w_1 x_1) & \ldots & \sigma(w_p x_1) \\ \vdots & \ddots & \vdots \\ \sigma(w_1 x_p) & \ldots & \sigma(w_p x_p) \end{pmatrix} \\ &= \sum_{\pi \in S_p} \operatorname{sgn}(\pi) \prod_{i=1}^p \sum_{k=0}^\infty a_k (w_i x_{\pi(i)})^k \\ &= \sum_{k_1, \ldots, k_p \geq 0} \sum_{\pi \in S_p} \operatorname{sgn}(\pi) \prod_{i=1}^p a_{k_i} w_i^{k_i} x_{\pi(i)}^{k_i} \\ &= \sum_{k_1, \ldots, k_p \geq 0} \left(\prod_{i=1}^p a_{k_i} \right) \left(\sum_{\pi \in S_p} \operatorname{sgn}(\pi) \prod_{i=1}^p x_{\pi(i)}^{k_i} \right) w_1^{k_1} \cdot \ldots \cdot w_p^{k_p} . \end{aligned} \tag{5.13}$$

Now, consider the special values $k_1, \ldots, k_p$ chosen in Step 2. Since $k_i \in K$, we have $a_{k_i} \neq 0$. The coefficient of the multivariate monomial $w_1^{k_1} \cdot \ldots \cdot w_p^{k_p}$ in Eq. (5.13) is

$$\left(\prod_{i=1}^p a_{k_i} \right) \left(\sum_{\pi \in S_p} \operatorname{sgn}(\pi) \prod_{i=1}^p x_{\pi(i)}^{k_i} \right) \overset{(5.12)}{=} a_{k_1} \cdot \ldots \cdot a_{k_p} \cdot D_x \neq 0 .$$

If g was the zero function, all derivatives of g would be zero and therefore the coefficients of all monomials would be zero, which is not the case. Hence, there exists $\boldsymbol{w} \in \mathbb{R}^p$ with $g(\boldsymbol{w}) \neq 0$. This shows that f is non-degenerate. □

5.H.4 Random Networks: Conclusion

In the following, we will prove Theorem 5.5.6 and discuss some possible extensions and limitations.

Theorem 5.5.6 (Random neural networks)**.** *Let $d, p, L \geq 1$, let $\sigma : \mathbb{R} \to \mathbb{R}$ be analytic and let the layer sizes be $d_0 = d$, $d_1, \ldots, d_{L-1} \geq 1$ and $d_L = p$. Let $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d_{l+1} \times d_l}$ for $l \in \{0, \ldots, L-1\}$ be random variables and consider the two cases where* *(a) σ is not a polynomial with less than p nonzero coefficients, $\boldsymbol{\theta} := (\boldsymbol{W}^{(0)}, \dots, \boldsymbol{W}^{(L-1)})$ and the random feature map $\phi_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^p$ is recursively defined by*

$$\phi(\boldsymbol{x}^{(0)}) := \boldsymbol{x}^{(L)}, \quad \boldsymbol{x}^{(l+1)} := \sigma(\boldsymbol{W}^{(l)}\boldsymbol{x}^{(l)}) .$$

(b) σ is not a polynomial of degree $< p-1$, $\boldsymbol{\theta} := (\boldsymbol{W}^{(0)}, \dots, \boldsymbol{W}^{(L-1)}, \boldsymbol{b}^{(0)}, \dots, \boldsymbol{b}^{(L-1)})$ with random variables $\boldsymbol{b}^{(l)} \in \mathbb{R}^{d_{l+1}}$ for $l \in \{0, \dots, L-1\}$, and the random feature map $\phi_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^p$ is recursively defined by

$$\phi(\boldsymbol{x}^{(0)}) := \boldsymbol{x}^{(L)}, \quad \boldsymbol{x}^{(l+1)} := \sigma(\boldsymbol{W}^{(l)}\boldsymbol{x}^{(l)} + \boldsymbol{b}^{(l)}) .$$

In both cases, if $\boldsymbol{\theta}$ has a Lebesgue density and $\boldsymbol{x}$ is nonatomic, then (FRK) holds for all n and almost surely over $\boldsymbol{\theta}$.

Proof. By Lemma 5.H.4, it suffices to consider the case where $\boldsymbol{\theta}$ has a standard normal distribution, since a standard normal distribution has a nonzero probability density everywhere. Especially, we can assume that all parameters in $\boldsymbol{\theta}$ are independent. By Proposition 5.5.2, we only need to prove (FRK) for $n = p$. Let $\boldsymbol{x}_1^{(0)}, \dots, \boldsymbol{x}_p^{(0)} \sim P_X$ be i.i.d. nonatomic random variables.

(a) If $p = 1$, σ is allowed to be a non-zero constant function. In this case, the feature map $\phi_{\boldsymbol{\theta}}$ is constant and non-zero with $p = 1$, which means that (FRK) holds. In the following, we thus assume that σ is non-constant. Let $\boldsymbol{x}^{(0)} \sim P_X$. Since $\boldsymbol{x}^{(0)}$ is nonatomic and σ is non-constant, an inductive application of Lemma 5.H.11 yields that almost surely over $\boldsymbol{\theta}$, $\boldsymbol{x}^{(L-1)}$ is also nonatomic. Hence, $\boldsymbol{x}_1^{(L-1)}, \dots, \boldsymbol{x}_p^{(L-1)}$ are independent and nonatomic almost surely over $\boldsymbol{\theta}$. But by Lemma 5.H.15, the elementwise application of σ is non-degenerate, and hence by Lemma 5.H.14, we almost surely have

$$\det \begin{pmatrix} \sigma(\boldsymbol{W}^{(L-1)}\boldsymbol{x}_1^{(L-1)})^\top \\ \vdots \\ \sigma(\boldsymbol{W}^{(L-1)}\boldsymbol{x}_p^{(L-1)})^\top \end{pmatrix} \neq 0 ,$$

which implies that (FRK) holds for $n = p$ almost surely over $\boldsymbol{\theta}$.

(b) If $p = 1$ and σ is a polynomial of degree $p - 1 = 0$, this means that σ is a non-zero constant function. Like in case (a), this implies that $\phi_{\boldsymbol{\theta}}$ is constant and non-zero with $p = 1$, which means that (FRK) holds. In the following, we thus assume again that σ is non-constant, i.e. not a polynomial of degree less than $\max\{1, p-1\}$. Since the distribution of the $\boldsymbol{x}_i^{(0)} := \boldsymbol{x}_i$ is non-atomic, they are distinct almost surely. By Theorem 5.H.9, $\boldsymbol{x}_1^{(L)}, \dots, \boldsymbol{x}_p^{(L)}$ are linearly independent almost surely over $\boldsymbol{\theta}$, which proves (FRK) for $n = p$ almost surely over $\boldsymbol{\theta}$. □

Remark 5.H.16 (Generalizations)**.** The proof technique used in Theorem 5.5.6 is quite robust and can be further generalized. For example, it is easy to incorporate different activation functions for different neurons, and in all layers but the last layer, the activation functions only need to be analytic and non-constant, as required by the corresponding propagation lemmas. It is also possible to treat fixed but nonzero biases using a combination of Lemma 5.H.11 (b) and using shifted activation functions $\tilde{\sigma}_i(x) = \sigma(x + b_i)$ in

Lemma 5.H.15. Also, the propagation lemmas, which are used for all layers except the last one, can be easily extended to DenseNet-like structures where the input of a layer is concatenated to the output. ◀

Remark 5.H.17 (Necessity of the assumptions)**.** For analytic σ, the assumptions on not being a too simple polynomial in Theorem 5.5.6 are necessary. For this, consider the case with $L = 1$ layer and $d = 1$.

(a) Assume that σ is a polynomial with less than p nonzero coefficients, i.e. $\sigma(x) = \sum_{k \in K} a_k x^k$ for $|K| \leq p - 1$. For arbitrary weights $\boldsymbol{w} \in \mathbb{R}^{p \times 1}$ and data points $x_1, \ldots, x_p \in \mathbb{R}$, we obtain the feature matrix $\phi_{\boldsymbol{w}}(\boldsymbol{X}) \in \mathbb{R}^{p \times p}$ with

$$\phi_{\boldsymbol{w}}(\boldsymbol{X})_{ij} = \sigma(w_j x_i) = \sum_{k \in K} a_k w_j^k x_i^k ,$$

which means that $\phi_{\boldsymbol{w}}(\boldsymbol{X})$ is the sum of the $|K| \leq p - 1$ matrices $(a_k w_j^k x_i^k)_{i,j \in [p]}$, which have at most rank 1. Hence, $\phi_{\boldsymbol{w}}(\boldsymbol{X})$ has at most rank $p - 1$ and is therefore not invertible.

(b) Assume that σ is a polynomial of degree less than $p - 1$, i.e. $\sigma = \sum_{k=0}^{p-2} a_k x^k$. For arbitrary weights $\boldsymbol{w} \in \mathbb{R}^{p \times 1}$, biases $\boldsymbol{b} \in \mathbb{R}^p$ and data points $x_1, \ldots, x_p \in \mathbb{R}$, we obtain the feature matrix $\phi_{\boldsymbol{w},\boldsymbol{b}}(\boldsymbol{X}) \in \mathbb{R}^{p \times p}$ with

$$\begin{aligned}\phi_{\boldsymbol{w},\boldsymbol{b}}(\boldsymbol{X})_{ij} = \sigma(w_j x_i + b_j) &= \sum_{k=0}^{p-2} a_k (w_j x_i + b_j)^k \\ &= \sum_{k=0}^{p-2} \sum_{l=0}^{k} a_k \binom{k}{l} b_j^{k-l} w_j^l x_i^l \\ &= \sum_{l=0}^{p-2} \left(\sum_{k=l}^{p-2} a_k \binom{k}{l} b_j^{k-l} w_j^l \right) x_i^l \\ &= \sum_{l=0}^{p-2} u_j^{(l)} x_i^l ,\end{aligned}$$

where $u_j^{(l)} \coloneqq \sum_{k=l}^{p-2} a_k \binom{k}{l} b_j^{k-l} w_j^l$ does not depend on i. Hence, $\phi_{\boldsymbol{w},\boldsymbol{b}}(\boldsymbol{X})$ is the sum of the $p - 1$ matrices $(u_j^{(l)} x_i^l)_{i,j \in [p]}$, each of which has rank at most 1. Hence, $\phi_{\boldsymbol{w},\boldsymbol{b}}(\boldsymbol{X})$ has at most rank $p - 1$ and is therefore not invertible. ◀

5.I Random Fourier Features

In a celebrated paper, Rahimi and Recht (2008) propose to approximate a shift-invariant positive definite kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \overline{k}(\boldsymbol{x} - \boldsymbol{x}')$ with a potentially infinite-dimensional feature map by a random finite-dimensional feature map, yielding so-called *random Fourier features*. If $\overline{k}$ is (up to scaling) the Fourier transform of a probability distribution P_k on $\mathbb{R}^d$, two versions of random Fourier features are proposed:

(1) One version uses $\phi_{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{x}) = \sqrt{2} \cos(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$, where the rows of $\boldsymbol{W} \in \mathbb{R}^{p \times d}$ are independently sampled from P_k and the entries of $\boldsymbol{b} \in \mathbb{R}^p$ are independently sampled

from the uniform distribution on $[0, 2\pi]$. This feature map is covered by Theorem 5.5.6 and hence, if P_k has a Lebesgue density and $\boldsymbol{x}$ is nonatomic, (FRK) is satisfied for all n. For example, if k is a Gaussian kernel, P_k is a Gaussian distribution and therefore has a Lebesgue density.

(2) The other version uses

$$\phi_{\boldsymbol{W}}(\boldsymbol{x}) = \begin{pmatrix} \sin(\boldsymbol{W}\boldsymbol{x}) \\ \cos(\boldsymbol{W}\boldsymbol{x}) \end{pmatrix}$$

with the same distribution over $\boldsymbol{W}$. It is not covered by Theorem 5.5.6 because of the different "activation functions" and the "weight sharing" between these activation functions. In the following proposition, we show that the proof of Theorem 5.5.6 can be adjusted to this setting and the conclusions still hold.

Proposition 5.I.1. *For $\boldsymbol{x} \in \mathbb{R}^d$, $\boldsymbol{W} \in \mathbb{R}^{q \times d}$ and $p := 2q$, define*

$$\phi_{\boldsymbol{W}}(\boldsymbol{x}) := \begin{pmatrix} \sin(\boldsymbol{W}\boldsymbol{x}) \\ \cos(\boldsymbol{W}\boldsymbol{x}) \end{pmatrix} \in \mathbb{R}^p .$$

If $\boldsymbol{W}$ has a Lebesgue density and $\boldsymbol{x}$ is nonatomic, then (FRK) holds for all n almost surely over $\boldsymbol{W}$.

Proof. **Step 1: Reduction.** According to Proposition 5.5.2, it suffices to consider the case $n = p$. By Lemma 5.H.14, it is then sufficient to prove that the function

$$f : \mathbb{R}^q \to \mathbb{R}^{2q}, \boldsymbol{x} \mapsto (\sin(\boldsymbol{x}), \cos(\boldsymbol{x}))$$

is non-degenerate in the sense of Definition 5.H.13.

Step 2: Condition on the x_i. We will proceed similar to Lemma 5.H.15. Let $x_1, \ldots, x_p$ be independent scalar nonatomic random variables. For $i \in [q]$, choose $k_i := 2i-1$ and $k_{q+i} := 2i-2$. Then, $k_1, \ldots, k_p$ are distinct non-negative integers, and by Lemma 5.H.12 and the Leibniz formula for the determinant, we have

$$D_x := \sum_{\pi \in S_p} \operatorname{sgn}(\pi) x_{\pi(1)}^{k_1} \cdot \ldots \cdot x_{\pi(p)}^{k_p} = \det \begin{pmatrix} x_1^{k_1} & \ldots & x_p^{k_1} \\ \vdots & \ddots & \vdots \\ x_1^{k_p} & \ldots & x_p^{k_p} \end{pmatrix} \neq 0$$

with probability one.

Step 3: Non-degeneracy. Now, suppose that we are indeed in the case where $D_x \neq 0$. Take the power series of sin and cos as

$$\sin(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!} =: \sum_{k=0}^{\infty} a_k x^k$$
$$\cos(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!} =: \sum_{k=0}^{\infty} b_k x^k .$$

Similar to the proof of Lemma 5.H.15, we can compute

$$g(\boldsymbol{w}) := \det \begin{pmatrix} f(\boldsymbol{w}x_1)^\top \\ \vdots \\ f(\boldsymbol{w}x_p)^\top \end{pmatrix}$$

$$
\begin{aligned}
&= \sum_{k_1,\dots,k_{2q}\geq 0} \sum_{\pi\in S_{2q}} \operatorname{sgn}(\pi) a_{k_1}\cdots a_{k_q} b_{k_{q+1}}\cdots b_{k_{2q}} w_1^{k_1+k_{q+1}}\cdots w_q^{k_q+k_{2q}} x_{\pi(1)}^{k_1}\cdots x_{\pi(2q)}^{k_{2q}} \\
&= \sum_{k_1,\dots,k_{2q}\geq 0} a_{k_1}\cdots a_{k_q} b_{k_{q+1}}\cdots b_{k_{2q}} \left(\sum_{\pi\in S_{2q}} \operatorname{sgn}(\pi) x_{\pi(1)}^{k_1}\cdots x_{\pi(2q)}^{k_{2q}}\right) \\
&\qquad \cdot w_1^{k_1+k_{q+1}}\cdots w_q^{k_q+k_{2q}} \qquad (5.14)
\end{aligned}
$$

Define the set $K := \{(k_1,\dots,k_{2q}) \in \mathbb{N}_0^{2q} \mid \text{for all } i\in[q],\ k_i + k_{q+i} = 4i-3\}$. Then, the coefficient c of the monomial $w_1^1 w_2^5 w_3^9 \cdot \ldots \cdot w_q^{4q-3}$ in (5.14) can be written as

$$
\begin{aligned}
c &:= \sum_{(k_1,\dots,k_{2q})\in K} c_{k_1,\dots,k_{2q}}\ , \\
c_{k_1,\dots,k_{2q}} &:= a_{k_1}\cdots a_{k_q} b_{k_{q+1}}\cdots b_{k_{2q}} \left(\sum_{\pi\in S_{2q}} \operatorname{sgn}(\pi) x_{\pi(1)}^{k_1}\cdots x_{\pi(2q)}^{k_{2q}}\right)\ .
\end{aligned}
$$

Now, consider $(k_1,\dots,k_{2q}) \in K$. Note that $a_k \neq 0$ iff k is odd and $b_k \neq 0$ iff k is even. For the choice $k_i := 2i-1, k_{q+i} := 2i-2$ for $i\in[q]$ from Step 2, we have $(k_1,\dots,k_{2q}) \in K$ and

$$
c_{k_1,\dots,k_{2q}} = a_{k_1}\cdots a_{k_q} b_{k_{q+1}}\cdots b_{k_{2q}} D_x \neq 0\ .
$$

In the following, we will show that $c_{k_1,\dots,k_{2q}} = 0$ for all other $(k_1,\dots,k_{2q}) \in K$, which implies $c \neq 0$ and therefore yields that g is not the zero function, which is what we want to show. If $k_i = k_j$ for some $i \neq j$, we have

$$
\sum_{\pi\in S_{2q}} \operatorname{sgn}(\pi) x_{\pi(1)}^{k_1}\cdots x_{\pi(2q)}^{k_{2q}} = \det\begin{pmatrix} x_1^{k_1} & \dots & x_p^{k_1} \\ \vdots & \ddots & \vdots \\ x_1^{k_p} & \dots & x_p^{k_p}\end{pmatrix} = 0\ ,
$$

since the i-th and j-th rows of the matrix are equal, and hence $c_{k_1,\dots,k_{2q}} = 0$. Now, suppose that $(k_1,\dots,k_{2q}) \in K$ with $c_{k_1,\dots,k_{2q}} \neq 0$. By induction, it is easy to show that $\{k_i, k_{q+i}\} = \{2i-1, 2i-2\}$ for all $i \in [q]$. But since

$$
a_{k_1}\cdots a_{k_q} b_{k_{q+1}}\cdots b_{k_{2q}} \neq 0
$$

and $a_k = 0$ for even k, we need to have $k_i = 2i-1, k_{q+i} = 2i-2$ for all $i\in[q]$. This shows the claim. □

5.J Proofs for Section 5.6

In this section, we first prove the analytic formulas from Section 5.6 before discussing the case of low input dimension d.

Theorem 5.6.1. *Let $P_Z = \mathcal{U}(\mathbb{S}^{p-1})$. Then, P_Z satisfies the assumptions (MOM), (COV), and (FRK) for all n with $\boldsymbol{\Sigma} = \frac{1}{p}\boldsymbol{I}_p$. Moreover, for $n \geq p = 1$ or $p \geq n \geq 1$, we can compute*

$$
\mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+) = \begin{cases} \frac{1}{n} & \text{if } n \geq p = 1, \\ \frac{1}{p} & \text{if } p \geq n = 1, \\ \infty & \text{if } 2 \leq n \leq p \leq n+1, \\ \frac{n}{p-1-n}\cdot\frac{p-2}{p} & \text{if } 2 \leq n \leq n+2 \leq p. \end{cases}
$$

Proof. **Step 1: Verify (MOM).** Let $\boldsymbol{x}_i \sim \mathcal{N}(0, \boldsymbol{I}_p)$ for $i \in [n]$ be independent. Then, $\boldsymbol{z}_i := \frac{\boldsymbol{x}_i}{\|\boldsymbol{x}_i\|_2} \sim \mathcal{U}(\mathbb{S}^{p-1})$. Since $\mathbb{E}\|\boldsymbol{z}_i\|_2^2 = \mathbb{E}1 = 1$, (MOM) is satisfied and thus, $\boldsymbol{\Sigma}$ is well-defined.

Step 2: Compute $\boldsymbol{\Sigma}$. We can use rotational invariance as follows: Let $\boldsymbol{V} \in \mathbb{R}^{p\times p}$ be an arbitrary fixed orthogonal matrix. Then, $\boldsymbol{V}\boldsymbol{x}_i \sim \mathcal{N}(0, \boldsymbol{V}\boldsymbol{V}^\top) = \mathcal{N}(0, \boldsymbol{I}_p)$ and hence $\boldsymbol{V}\boldsymbol{z}_i = \frac{\boldsymbol{V}\boldsymbol{x}_i}{\|\boldsymbol{x}_i\|_2} = \frac{\boldsymbol{V}\boldsymbol{x}_i}{\|\boldsymbol{V}\boldsymbol{x}_i\|_2} \sim \mathcal{U}(\mathbb{S}^{p-1})$. Therefore,

$$\boldsymbol{\Sigma} = \mathbb{E}\boldsymbol{z}_i\boldsymbol{z}_i^\top = \mathbb{E}\boldsymbol{V}\boldsymbol{z}_i\boldsymbol{z}_i^\top\boldsymbol{V}^\top = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^\top . \tag{5.15}$$

If $0 \neq \boldsymbol{v} \in \mathbb{R}^p$ is an eigenvector of $\boldsymbol{\Sigma}$ with eigenvalue λ, then $\boldsymbol{V}\boldsymbol{v}$ must by Eq. (5.15) also be an eigenvector of $\boldsymbol{\Sigma}$ with eigenvalue λ. But since $\boldsymbol{V}$ is an arbitrary orthogonal matrix, this means that $\boldsymbol{V}\boldsymbol{v}$ is an arbitrary rotation of $\boldsymbol{v}$. From this it is easy to conclude that $\boldsymbol{\Sigma} = \lambda\boldsymbol{I}_p$, and from

$$p\lambda = \operatorname{tr}(\boldsymbol{\Sigma}) = \mathbb{E}\operatorname{tr}(\boldsymbol{z}_i\boldsymbol{z}_i^\top) = \mathbb{E}\boldsymbol{z}_i^\top\boldsymbol{z}_i = \mathbb{E}1 = 1 ,$$

it follows that $\boldsymbol{\Sigma} = \frac{1}{p}\boldsymbol{I}_p$. Hence, (COV) is satisfied and $\boldsymbol{w}_i = \sqrt{p}\boldsymbol{z}_i$.

Step 3: Verify (FRK) for all n. By Proposition 5.5.2, it is sufficient to verify (FRK) for $n = p$. Therefore, let $n = p$. It is obvious from Proposition 5.5.2 with $\phi = \mathrm{id}$ that $\mathcal{N}(0, \boldsymbol{I}_p)$ satisfies (FRK). Hence, $\boldsymbol{X}$ almost surely has full rank. But then, since $\|\boldsymbol{x}_i\|_2 > 0$ almost surely,

$$\boldsymbol{Z} = \operatorname{diag}\left(\frac{1}{\|\boldsymbol{x}_1\|}, \ldots, \frac{1}{\|\boldsymbol{x}_n\|}\right)\boldsymbol{X}$$

almost surely has full rank as well, which proves (FRK).

Step 4.1: Computation for $n \geq p = 1$. In the underparameterized case $n \geq p = 1$, we can compute

$$\mathbb{E}_{\boldsymbol{Z}}\operatorname{tr}((\boldsymbol{Z}^+)^\top\boldsymbol{\Sigma}\boldsymbol{Z}^+) = \mathbb{E}_{\boldsymbol{Z}}\operatorname{tr}((\boldsymbol{W}^\top\boldsymbol{W})^{-1}) = \mathbb{E}_{\boldsymbol{Z}}\frac{1}{\sum_{i=1}^n w_i^2} = \mathbb{E}_{\boldsymbol{Z}}\frac{1}{n} = \frac{1}{n} .$$

Step 4.2: Computation for $p \geq n = 1$. In the overparameterized case $p \geq n = 1$, we can compute

$$\mathbb{E}_{\boldsymbol{Z}}\operatorname{tr}((\boldsymbol{Z}^+)^\top\boldsymbol{\Sigma}\boldsymbol{Z}^+) = \mathbb{E}_{\boldsymbol{Z}}\operatorname{tr}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1}) = \mathbb{E}_{\boldsymbol{Z}}\frac{1}{\boldsymbol{w}_1^\top\boldsymbol{w}_1} = \mathbb{E}_{\boldsymbol{Z}}\frac{1}{p} = \frac{1}{p} ,$$

where we used that since $\boldsymbol{\Sigma} = \frac{1}{p}\boldsymbol{I}_p$, $\boldsymbol{w}_1^\top\boldsymbol{w}_1 = \|\boldsymbol{w}_1\|_2^2 = \|\sqrt{p}\boldsymbol{z}_1\|_2^2 = p$.

Step 4.3: Computation for $p \geq n \geq 2$. Now, let $p \geq n \geq 2$. Since $\boldsymbol{\Sigma} = \frac{1}{p}\boldsymbol{I}_p$, we have

$$\mathbb{E}_{\boldsymbol{Z}}\operatorname{tr}((\boldsymbol{Z}^+)^\top\boldsymbol{\Sigma}\boldsymbol{Z}^+) = \mathbb{E}_{\boldsymbol{Z}}\operatorname{tr}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1})$$

by Theorem 5.4.3. Using that the $\boldsymbol{w}_i$ are i.i.d., we obtain from Lemma 5.G.2 that $\mathbb{E}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1}) = n\mathbb{E}\operatorname{dist}(\boldsymbol{w}_1, \mathcal{W}_{-1})^{-2}$, where $\mathcal{W}_{-1}$ is the space spanned by $\boldsymbol{w}_2, \ldots, \boldsymbol{w}_n$. Define the subspace $\mathcal{U}_n := \{\boldsymbol{z} \in \mathbb{R}^p \mid z_n = z_{n+1} = \ldots = z_p = 0\}$. By (FRK), we almost surely have $\dim(\mathcal{W}_{-1}) = n - 1$. Thus, there is an orthogonal matrix $\boldsymbol{U}_{-1}$ depending only on $\mathcal{W}_{-1}$ that rotates $\mathcal{W}_{-1}$ to $\mathcal{U}_n$:

$$\mathcal{U}_n = \boldsymbol{U}_{-1}\mathcal{W}_{-1} .$$

Because $\boldsymbol{w}_1$ is stochastically independent from $\mathcal{W}_{-1}$ and $\boldsymbol{U}_{-1}$ and its distribution is rotationally symmetric, we have the distributional equivalence (using the $\boldsymbol{z}_i$ and $\boldsymbol{x}_i$ from Step 1)

$$\begin{aligned}\operatorname{dist}(\boldsymbol{w}_1, \mathcal{W}_{-1})^2 &= \operatorname{dist}(\boldsymbol{U}_{-1}\boldsymbol{w}_1, \boldsymbol{U}_{-1}\mathcal{W}_{-1})^2 \overset{\text{distrib.}}{=} \operatorname{dist}(\boldsymbol{w}_1, \mathcal{U}_n)^2 = p(z_{1,n}^2 + \ldots + z_{1,p}^2) \\ &= p\frac{x_{1,n}^2 + \ldots + x_{1,p}^2}{\|\boldsymbol{x}_1\|_2^2} = p\frac{A}{A+B} ,\end{aligned}$$

where $A \coloneqq x_{1,n}^2 + x_{1,n+1}^2 + \ldots + x_{1,p}^2$ has a χ^2_{p+1-n} distribution and $B \coloneqq x_{1,1}^2 + \ldots + x_{1,n-1}^2$ has a χ^2_{n-1} distribution. Hence,

$$\mathbb{E}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1}) = n\mathbb{E}\operatorname{dist}(\boldsymbol{w}_1, \mathcal{W}_{-1})^{-2} = \frac{n}{p}\left(1 + \mathbb{E}\frac{B}{A}\right) .$$

Since $p \geq n \geq 2$, $n-1$ and $p+1-n$ are positive. Since A and B are independent, $\frac{B/(n-1)}{A/(p+1-n)}$ follows a Variance-Ratio F-distribution with parameters $n-1$ and $p+1-n$, whose mean is known (see e.g. Chapter 20 in Forbes et al., 2011):

$$\mathbb{E}\frac{B}{A} = \frac{n-1}{p+1-n}\mathbb{E}\frac{B/(n-1)}{A/(p+1-n)} = \begin{cases}\infty & \text{, if } p+1-n \leq 2, \\ \frac{n-1}{p+1-n}\frac{p+1-n}{(p+1-n)-2} = \frac{n-1}{p-1-n} & \text{, if } p+1-n > 2.\end{cases} \tag{5.16}$$

The infinite expectation for $p+1-n \leq 2$ is not explicitly specified in Forbes et al. (2011), but it is easy to obtain from the p.d.f. of the F-distribution: The p.d.f. $f(x)$ of the $F(a,b)$-distribution for $x \geq 0$ is

$$f(x) = C_{a,b}\frac{x^{(a-2)/2}}{(1+(a/b)x)^{(a+b)/2}} = \Theta(x^{-b/2-1}) \qquad (x \to \infty)$$

for some constant $C_{a,b}$ (cf. Chapter 20 in Forbes et al., 2011), and the expected value is therefore

$$\int_0^\infty xf(x)\,\mathrm{d}x = \int_0^\infty \Theta(x^{-b/2})\,\mathrm{d}x ,$$

which is infinite for $p+1-n = b \leq 2$. For $n \in \{p, p-1\}$, we therefore obtain $\mathbb{E}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1}) = \infty$. For $n \leq p-2$, we compute

$$\mathbb{E}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1}) = \frac{n}{p}\left(1 + \frac{n-1}{p-1-n}\right) = \frac{n}{p} \cdot \frac{p-2}{p-1-n} . \qquad \square$$

In the following, we will prove Theorem 5.6.2 using the same proof idea as for Theorem 5.6.1. The formulas in Theorem 5.6.2 have in principle already been computed by Breiman and Freedman (1983) for $p \leq n-2$ and by Belkin et al. (2020) for general p. However, our proof circumvents a technical problem in the proof of Belkin et al. (2020): Consider for example the case $p \leq n$. Belkin et al. (2020) mention that $(\boldsymbol{W}^\top\boldsymbol{W})^{-1}$ has an inverse Wishart distribution, which for $p \leq n-2$ has expectation $\frac{1}{n-1-p}\boldsymbol{I}_p$, and then use $\mathbb{E}\operatorname{tr}((\boldsymbol{W}^\top\boldsymbol{W})^{-1}) = \operatorname{tr}(\mathbb{E}(\boldsymbol{W}^\top\boldsymbol{W})^{-1})$. However, for $p \geq n-1$, the latter expectation is not specified in common literature[9] on the inverse Wishart distribution (Mardia et al., 1979; Press, 2005; von Rosen, 1988), presumably because it is ∞ for diagonal elements but is not well-defined for off-diagonal matrix elements.

[9]Belkin et al. (2020) do not cite any source on the inverse Wishart distribution.

Theorem 5.6.2. *Let $P_Z = \mathcal{N}(0, \boldsymbol{I}_p)$. Then, P_Z satisfies the assumptions (MOM), (COV), and (FRK) for all n with $\boldsymbol{\Sigma} = \boldsymbol{I}_p$. Moreover, for $n, p \geq 1$,*

$$\mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma} \boldsymbol{Z}^+) = \begin{cases} \frac{n}{p-1-n} & \text{if } p \geq n+2, \\ \infty & \text{if } p \in \{n-1, n, n+1\}, \\ \frac{p}{n-1-p} & \text{if } p \leq n-2. \end{cases}$$

Proof. **Step 1: Assumptions.** Verifying (MOM), (COV) and $\boldsymbol{\Sigma} = \boldsymbol{I}_p$ is trivial and (FRK) for all n follows from Proposition 5.5.2 with $\boldsymbol{x} = \boldsymbol{z}$ and $\phi = \mathrm{id}$.

Step 2: Overparameterized case. For the expectation, we first follow Step 4.3 in the proof of Theorem 5.6.1 in the overparameterized case $p \geq n \geq 1$, the main difference being that instead of $\boldsymbol{w}_i = \sqrt{p}\frac{\boldsymbol{x}_i}{\|\boldsymbol{x}_i\|_2}$, we now have $\boldsymbol{w}_i = \boldsymbol{x}_i$, which translates to the simpler equation

$$\operatorname{dist}(\boldsymbol{w}_1, \mathcal{W}_{-1})^2 \overset{\text{distrib.}}{=} A$$

with $A \sim \chi^2_{p+1-n}$. Let $B \sim \chi^2_1$ be independent of A, then we can compute similar to Eq. (5.16)

$$\begin{aligned} \mathbb{E} \operatorname{tr}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1}) &= n\mathbb{E} \operatorname{dist}(\boldsymbol{w}_1, \mathcal{W}_{-1})^{-2} = n\mathbb{E}\frac{1}{A} = n\,(\mathbb{E}B)\left(\mathbb{E}\frac{1}{A}\right) = n\mathbb{E}\frac{B}{A} \\ &= \frac{n}{p+1-n}\mathbb{E}\frac{B/1}{A/(p+1-n)} \\ &= \begin{cases} \infty & \text{if } p+1-n \leq 2, \\ \frac{n}{p+1-n}\frac{p+1-n}{(p+1-n)-2} = \frac{n}{p-1-n} & \text{if } p+1-n > 2. \end{cases} \end{aligned}$$

This proves the over-parameterized case.

Step 3: Underparameterized case. Since the rows $\boldsymbol{w}_i$ of $\boldsymbol{W} \in \mathbb{R}^{n \times p}$ are independent and follow a $\mathcal{N}(0, \boldsymbol{I}_p)$ distribution, the rows of $\boldsymbol{W}^\top \in \mathbb{R}^{p \times n}$ are independent and follow a $\mathcal{N}(0, \boldsymbol{I}_n)$ distribution. Therefore, the underparameterized case $p \leq n$ follows from the overparameterized case $n \leq p$ by switching the roles of n and p. □

Remark 5.J.1. An alternative (and presumably similar) way to prove Theorem 5.6.2 is to use that the diagonal elements of a matrix with an inverse Wishart distribution follow an inverse Gamma distribution as specified in Example 5.2.2 in Press (2005). ◀

The next proposition shows that a small input dimension d does not necessarily provide a limitation:

Proposition 5.J.2. *Let $p, d \geq 1$. Then, there exists a probability distribution P_X on $\mathbb{R}^d$ (with bounded support) and a continuous feature map $\phi : \mathbb{R}^d \to \mathbb{R}^p$ such that for $\boldsymbol{x} \sim P_X$, $\phi(\boldsymbol{x}) \sim \mathcal{U}(\mathbb{S}^{p-1})$.*

Proof. For $p = 1$, the result is trivial, we will therefore assume $p \geq 2$. We will prove the result for any d by a reduction to the case $d = 1$, although substantially simpler constructions are possible for $d \geq p-1$. First, introduce the spaces

$$\mathbb{S}^{p-1}_+ := \{\boldsymbol{z} \in \mathbb{S}^{p-1} \mid z_p \geq 0\} \subseteq \mathbb{R}^p$$

$$\mathbb{B}^{p-1} \coloneqq \{\boldsymbol{z} \in \mathbb{R}^{p-1} \mid \|\boldsymbol{z}\|_2 \leq 1\}$$
$$\mathcal{X} \coloneqq [0,3] \times \{0\}^{d-1} \subseteq \mathbb{R}^d .$$

Step 1: Space-filling curve on the sphere. In this step, we show that there exists a continuous surjective map $\phi : \mathcal{X} \to \mathbb{S}^{p-1}$. First of all, let $f_1 : [0,1] \to [0,1]^{p-1}$ be continuous and surjective, e.g. a Hilbert or Peano curve (see e.g. Sagan, 2012). We define the following maps:

$$f_2 : [0,1]^{p-1} \to \mathbb{B}^{p-1}, \boldsymbol{u} \mapsto \begin{cases} \boldsymbol{0} & \text{if } \boldsymbol{u} = \boldsymbol{0} \\ \frac{\|\boldsymbol{u}\|_\infty}{\|\boldsymbol{u}\|_2}\boldsymbol{u} & \text{if } \boldsymbol{u} \neq \boldsymbol{0} \end{cases},$$
$$f_3 : \mathbb{B}^{p-1} \to \mathbb{S}^{p-1}_+, \boldsymbol{v} \mapsto \left(\boldsymbol{v}, \sqrt{1 - \|\boldsymbol{v}\|_2^2}\right) .$$

It is not hard to verify that f_2 and f_3 are continuous and surjective as well. For example, f_2 is continuous in $\boldsymbol{0}$ since $\|\boldsymbol{u}\|_\infty \leq \|\boldsymbol{u}\|_2$ for all $\boldsymbol{u} \in \mathbb{R}^{p-1}$. Thus, the map $f \coloneqq f_3 \circ f_2 \circ f_1 : [0,1] \to \mathbb{S}^{p-1}_+$ is continuous and surjective. Define the map

$$\tau : \mathbb{R}^p \to \mathbb{R}^p, \boldsymbol{z} \mapsto (z_1, \ldots, z_{p-1}, -z_p) .$$

Since we assumed $p \geq 2$ at the beginning of the proof, the sphere $\mathbb{S}^{p-1}$ is path-connected, hence there exists a path $h : [0,1] \to \mathbb{S}^{p-1}$ with $h(0) = f(1), h(1) = \tau(f(1))$. By the previous considerations, it is not hard to verify that the map

$$g : [0,3] \to \mathbb{S}^{p-1}, x \mapsto \begin{cases} f(x) & \text{if } x \in [0,1] \\ h(x-1) & \text{if } x \in [1,2] \\ \tau(f(3-x)) & \text{if } x \in [2,3] \end{cases}$$

is continuous and surjective. We can therefore define the continuous and surjective map $\phi : \mathcal{X} \to \mathbb{S}^{p-1}, \boldsymbol{x} \mapsto g(x_1)$.

Step 2: Existence of a pull-back measure. We consider the Borel σ-algebras $\mathcal{B}(\mathcal{X}), \mathcal{B}(\mathbb{S}^{p-1})$ on $\mathcal{X}$ and $\mathbb{S}^{p-1}$. The uniform distribution $P_Z = \mathcal{U}(\mathbb{S}^{p-1})$ on the sphere is defined with respect to $\mathcal{B}(\mathbb{S}^{p-1})$ and is therefore a Borel measure. Since ϕ is continuous, it is Borel measurable. Moreover, since $\mathcal{X}$ and $\mathbb{S}^{p-1}$ are complete separable metric spaces, they are also Souslin spaces, cf. Section 6.6 in Bogachev (2007). Since ϕ is surjective, Theorem 9.1.5 in Bogachev (2007) guarantees the existence of a measure P_X such that if $\boldsymbol{x} \sim P_X$, then $\phi(\boldsymbol{x}) \sim \mathcal{U}(\mathbb{S}^{p-1})$. Since $P_X(\mathcal{X}) = P_Z(\mathbb{S}^{p-1}) = 1$, P_X is a probability measure.

Step 3: Continuation. We can arbitrarily extend the mapping $\phi : \mathcal{X} \to \mathbb{S}^{p-1}$ to a continuous mapping $\phi : \mathbb{R}^d \to \mathbb{R}^p$. Moreover, the domain $\mathcal{X}$ of P_X can be extended to $\mathbb{R}^d$ via $P_X(A) \coloneqq P_X(A \cap \mathcal{X})$, the support of P_X is still bounded, and we still have $\phi(\boldsymbol{x}) \sim \mathcal{U}(\mathbb{S}^{p-1})$ if $\boldsymbol{x} \sim P_X$. □

Remark 5.J.3. The proof of Proposition 5.J.2 could be slightly shorter if we required $\phi(\boldsymbol{x}) \sim \mathcal{U}(\mathbb{S}^{p-1}_+)$ instead of $\phi(\boldsymbol{x}) \sim \mathcal{U}(\mathbb{S}^{p-1})$. This would be of similar interest since the uniform distribution $\mathcal{U}(\mathbb{S}^{p-1}_+)$ on the "half-sphere" leads to the same $\mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{Z}^+)^\top \boldsymbol{\Sigma}(\boldsymbol{Z}^+))$ as the uniform distribution $\mathcal{U}(\mathbb{S}^{p-1})$ on the full sphere: If $\boldsymbol{z}_i \sim \mathcal{U}(\mathbb{S}^{p-1}_+)$ and $\varepsilon_i \sim \mathcal{U}(\{-1,1\})$ are stochastically independent, then $\tilde{\boldsymbol{z}}_i \coloneqq \varepsilon_i \boldsymbol{z}_i \sim \mathcal{U}(\mathbb{S}^{p-1})$. Therefore, $\boldsymbol{\Sigma} = \tilde{\boldsymbol{\Sigma}}$, $\tilde{\boldsymbol{Z}} = \operatorname{diag}(\varepsilon_1, \ldots, \varepsilon_n)\boldsymbol{Z}$, and if $\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$ is a SVD of $\boldsymbol{Z}$, then $(\operatorname{diag}(\varepsilon_1, \ldots, \varepsilon_n)\boldsymbol{U})\boldsymbol{D}\boldsymbol{V}^\top$ is a

SVD of $\tilde{\boldsymbol{Z}}$. Therefore, $\boldsymbol{Z}$ and $\tilde{\boldsymbol{Z}}$ have the same singular values, hence $\boldsymbol{W}$ and $\tilde{\boldsymbol{W}}$ have the same singular values, hence $\operatorname{tr}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1}) = \operatorname{tr}((\tilde{\boldsymbol{W}}\tilde{\boldsymbol{W}}^\top)^{-1})$ for $p \geq n$ and $\operatorname{tr}((\boldsymbol{W}^\top\boldsymbol{W})^{-1}) = \operatorname{tr}((\tilde{\boldsymbol{W}}^\top\tilde{\boldsymbol{W}})^{-1})$ for $p \leq n$. ◀

Remark 5.J.4. One might ask whether it is possible in Proposition 5.J.2 to choose P_X as a "nice" distribution, like a uniform distribution on a cube or a Gaussian distribution. The answer to this question is affirmative if there exists an area-preserving space-filling curve $\phi : [0, \operatorname{volume}(\mathbb{S}^{p-1})] \to \mathbb{S}^{p-1}$. For $p = 3$, such a construction is informally described by Purser et al. (2009) and it seems plausible that such a construction is possible for all p. ◀

5.K Relation to Ridgeless Kernel Regression

In this section, we want to discuss the relation between this paper and recent work on ridgeless kernel regression. To this end, we need to introduce some terminology on representations of kernels with finite-dimensional feature maps.

Definition 5.K.1. Let $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a kernel, let p be an integer with $1 \leq p < \infty$ and let $\phi : \mathbb{R}^d \to \mathbb{R}^p$ be a (measurable) function. Then, (p, ϕ) is called a P_X-representation of k if

- $k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \phi(\boldsymbol{x})^\top \phi(\tilde{\boldsymbol{x}})$ almost surely for independent $\boldsymbol{x}, \tilde{\boldsymbol{x}} \sim P_X$, and
- $k(\boldsymbol{x}, \boldsymbol{x}) = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x})$ almost surely for $\boldsymbol{x} \sim P_X$.

If k has a P_X-representation, then we define

$$p_k := \min_{(p,\phi) \text{ is a } P_X\text{-representation of } k} p\ ,$$

i.e. p_k is the smallest p for which a P_X-representation exists. ◀

Usually, p_k corresponds to the dimension of the RKHS associated with the restriction of k to the support of P_X, but since the feature map ϕ only needs to represent the kernel P_X-almost surely, p_k may be smaller for pathological kernels. The following lemma states that Theorem 5.4.3, if applicable, should be applied to ridgeless kernel regression with $p = p_k$:

Lemma 5.K.2. *Let k be a kernel on $\mathbb{R}^d$ with $P_X(k(\boldsymbol{x}, \boldsymbol{x}) \neq 0) > 0$. Let (p, ϕ) be a P_X-representation of k.*

(a) Then, (COV) in Theorem 5.4.3 is satisfied iff $p = p_k$.
(b) The assumptions of Theorem 5.4.3 are satisfied for a P_X-representation (p_k, ϕ) of k if and only if they are satisfied for all such representations.
(c) If the assumptions of Theorem 5.4.3 are satisfied for any P_X-representation of k, then the lower bound from Theorem 5.4.3 also holds for ridgeless kernel regression with $p = p_k$.

Proof. In the notation of Section 5.3, the definition of P_X-representation implies that $k(\boldsymbol{X}, \boldsymbol{X}) = \phi(\boldsymbol{X})\phi(\boldsymbol{X})^\top$ and $k(\boldsymbol{x}, \boldsymbol{X}) = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{X})^\top$ almost surely.

(a) If (COV) is not satisfied and $p \geq 2$, it is possible to construct a P_X-representation with smaller p using the construction from Remark 5.5.3, hence $p > p_k$. If $p = 1$, (COV) is satisfied due to the assumption on k.
Conversely, assume $p > p_k$ and let $(p_k, \tilde{\phi})$ be another P_X-representation of k. Set $n = p$. Then, we almost surely have

$$\begin{aligned}\operatorname{rank} \phi(\boldsymbol{X}) &= \operatorname{rank}(\phi(\boldsymbol{X})\phi(\boldsymbol{X})^\top) = \operatorname{rank}(k(\boldsymbol{X}, \boldsymbol{X})) = \operatorname{rank}(\tilde{\phi}(\boldsymbol{X})\tilde{\phi}(\boldsymbol{X})^\top) \leq p_k \\ &< p ,\end{aligned}$$

hence $\phi(\boldsymbol{X})$ has full rank with probability zero. Since the rows $\phi(\boldsymbol{x}_i)$ of $\phi(\boldsymbol{X})$ are i.i.d., this means that there must be a proper linear subspace U of $\mathbb{R}^p$ such that $\phi(\boldsymbol{x}_i) \in U$ with probability one. But then, according to Proposition 5.5.2, (COV) is not satisfied.

(b) Let (p_k, ϕ) and $(p_k, \tilde{\phi})$ be two P_X-representations of k such that $\boldsymbol{z} = \phi(\boldsymbol{x})$ satisfies the assumptions of Theorem 5.4.3. We need to show that $\tilde{\boldsymbol{z}} = \tilde{\phi}(\boldsymbol{x})$ also satisfies the assumptions of Theorem 5.4.3: First of all, (INT) and (NOI) hold since they are independent of the feature map. Moreover, (COV) holds by (a). We find that (MOM) holds due to

$$\mathbb{E}\|\tilde{\boldsymbol{z}}\|_2^2 = \mathbb{E}\tilde{\phi}(\boldsymbol{x})^\top\tilde{\phi}(\boldsymbol{x}) = \mathbb{E}k(\boldsymbol{x}, \boldsymbol{x}) = \mathbb{E}\phi(\boldsymbol{x})^\top\phi(\boldsymbol{x}) = \mathbb{E}\|\boldsymbol{z}\|_2^2 < \infty$$

and (FRK) holds since, almost surely,

$$\begin{aligned}\operatorname{rank} \tilde{\phi}(\boldsymbol{X}) &= \operatorname{rank}(\tilde{\phi}(\boldsymbol{X})\tilde{\phi}(\boldsymbol{X})^\top) = \operatorname{rank}(k(\boldsymbol{X}, \boldsymbol{X})) = \operatorname{rank}(\phi(\boldsymbol{X})\phi(\boldsymbol{X})^\top) \\ &= \operatorname{rank} \phi(\boldsymbol{X}) = \min\{n, p\} .\end{aligned}$$

(c) Assume that there exists a P_X-representation (p, ϕ) of k that satisfies the assumptions of Theorem 5.4.3. By (a), we have $p = p_k$. By Eq. (5.2) in Section 5.3, the ridgeless kernel regression estimator and the linear regression estimator with the feature map ϕ are almost surely equivalent, hence they have the same $\mathcal{E}_{\text{Noise}}$. □

For kernels that cannot be represented with a finite-dimensional feature space, Theorem 5.4.3 cannot be applied. In fact, any distribution-independent lower bound for ridgeless kernel regression must be zero in this case: For example, the Kronecker delta kernel given by

$$k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \begin{cases} 1 & \text{if } \boldsymbol{x} = \tilde{\boldsymbol{x}} \\ 0 & \text{otherwise} \end{cases}$$

yields $\mathcal{E}_{\text{Noise}} = 0$ for any nonatomic input distribution P_X. Of course, this kernel is not well-suited for learning since the learned functions are zero almost everywhere. However, there exist results for ridgeless kernel regression with specific classes of kernels. For example, Rakhlin and Zhai (2019) show that in certain settings, ridgeless kernel regression with Laplace kernels is inconsistent because $\mathcal{E}_{\text{Noise}} = \Omega(1)$ as $n \to \infty$. Note that Laplace kernels in general do not allow for finite-dimensional feature map representations.

Liang and Rakhlin (2020) derive upper bounds for a certain class of kernels and input distributions with (linearly transformed) i.i.d. components. Their analysis focuses on the high-dimensional limit $d, n \to \infty$ with $0 < c \leq d/n \leq C < \infty$ and ignores the "effective dimension" p_k of the feature space. It appears that their analysis is not impacted by Double Descent w.r.t. p_k since their assumptions on the kernel imply either $p_k = \infty$ or $p_k/n \to \infty$ as $n, d \to \infty$: In particular, they consider kernels of the form

$$k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = h\left(\frac{1}{d}\langle \boldsymbol{x}, \tilde{\boldsymbol{x}}\rangle\right)$$

for a suitable smooth function h that is independent of d. Due to the factor $\frac{1}{d}$ and the limit $d \to \infty$, the kernel behaves essentially like a quadratic kernel

$$k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \approx a_0 + a_1\frac{1}{d}\langle \boldsymbol{x}, \tilde{\boldsymbol{x}}\rangle + a_2\left(\frac{1}{d}\langle \boldsymbol{x}, \tilde{\boldsymbol{x}}\rangle\right)^2 =: k_{\text{quad}}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \ ,$$

where the curvature a_2 should be positive in order to obtain good upper bounds on $\mathcal{E}_{\text{Noise}}$ (the variance term). For $a_2, a_1, a_0 > 0$, it is possible to represent this quadratic kernel with a feature map analogous to that of the polynomial kernel in Proposition 5.5.4 with feature space dimension $p = 1 + d + \frac{d(d+1)}{2}$. An argument similar to the proof of Proposition 5.5.4 shows that this feature map satisfies FRK(p) if $\boldsymbol{x}$ has a Lebesgue density, hence (COV) is satisfied. By Lemma 5.K.2 (a), we have $p_{k_{\text{quad}}} = p = \Theta(d^2) = \Theta(n^2)$, which shows $p_{k_{\text{quad}}}/n \to \infty$ as $n, d \to \infty$.

Liang et al. (2020) consider a similar setting with $d \sim n^\alpha, \alpha \in (0, 1)$, and find that $\mathcal{E}_{\text{Noise}}$ converges to zero under suitable assumptions as $d, n \to \infty$. Again, it appears that their assumptions on the kernel imply at least a strongly overparameterized regime with $p_k = \infty$ or $p_k/n \to \infty$ for $d, n \to \infty$, where our lower bound is vacuous.

5.L Novelty of the Overparameterized Bound

In their Corollary 1, Muthukumar et al. (2020) provide a lower bound in the case $p \geq n$ holding with high probability for $\boldsymbol{\varepsilon}^\top(\boldsymbol{W}\boldsymbol{W}^\top)^{-1}\boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \boldsymbol{I}_p)$ is a noise vector independent of $\boldsymbol{W}$. Since $\mathbb{E}\boldsymbol{\varepsilon}^\top(\boldsymbol{W}\boldsymbol{W}^\top)^{-1}\boldsymbol{\varepsilon} = \mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1}\mathbb{E}_{\boldsymbol{\varepsilon}}\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top) = \mathbb{E}_{\boldsymbol{Z}} \operatorname{tr}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1})$, their lower bound yields a lower bound for $\mathbb{E}\operatorname{tr}((\boldsymbol{W}\boldsymbol{W}^\top)^{-1})$. However, the resulting lower bound is weaker than ours and requires stronger assumptions:

(1) Assuming that the subgaussian norm $\|\boldsymbol{w}_i\|_{\psi_2} := \sup_{\boldsymbol{v}\in\mathbb{S}^{p-1}} \sup_{q\in\mathbb{N}_+} q^{-1/2}(\mathbb{E}|\boldsymbol{v}^\top\boldsymbol{w}_i|^q)^{1/q}$ (cf. Vershynin, 2010) is bounded by a constant $K < \infty$, they obtain a lower bound of the form $c_K\sigma^2\frac{n}{p}$ with a constant $c_K > 0$ that depends on K and is only explicitly specified for the case of centered Gaussian P_Z. They note that $\|\boldsymbol{w}_i\|_{\psi_2} \leq K$ holds, for example, if the components $w_{i,j}$ of $\boldsymbol{w}_i$ are independent and all satisfy $\|w_{i,j}\|_{\psi_2} \leq K$. However, as discussed in Remark 5.4.1, such independence assumptions are not realistic. In contrast, our lower bound is explicit, independent of constants like K and is larger: For example, at $n = p$, our lower bound is $\sigma^2 n$ and theirs is $\sigma^2 c_K$.

(2) Assuming $\|\boldsymbol{w}_i\|_2^2 \leq p$ almost surely, they obtain a lower bound of the form $c\sigma^2\frac{n}{p\log(n)}$. First of all, this lower bound converges to zero as $n = p \to \infty$. Moreover, since we always have $\mathbb{E}\|\boldsymbol{w}_i\|_2^2 = \mathbb{E}\operatorname{tr}(\boldsymbol{w}_i\boldsymbol{w}_i^\top) = \mathbb{E}\operatorname{tr}(\boldsymbol{I}_p) = p$, the assumption implies $\|\boldsymbol{w}_i\|_2^2 = p$ almost surely. Although we can sometimes guarantee constant $\|\boldsymbol{z}_i\|_2^2$, e.g. for certain random Fourier features, we cannot guarantee the same for $\boldsymbol{w}_i = \boldsymbol{\Sigma}^{-1/2}\boldsymbol{z}_i$ since $\boldsymbol{\Sigma}$ depends on the unknown input distribution P_X.

By inspecting the proof behind (1), one finds that $c_K \to 0$ as $K \to \infty$. Hence, lower bound of Muthukumar et al. (2020) might raise hope that it is possible to achieve low $\mathcal{E}_{\text{Noise}}$ by choosing features with a large (or even infinite) subgaussian norm. Our result shows that this is not possible: Essentially the only possibility to avoid a large $\mathcal{E}_{\text{Noise}}$ for ridgeless linear regression around $n \approx p$ is to violate the property (FRK) that guarantees the ability to interpolate the data in the overparameterized case, see Section 5.5. Otherwise, in order to achieve $\mathcal{E}_{\text{Noise}} < \varepsilon\sigma^2$, $\varepsilon \ll 1$, it is necessary to make the model either strongly underparameterized ($p < \varepsilon n$) or strongly overparameterized ($p > n/\varepsilon$).

Chapter 6

A Framework and Benchmark for Deep Batch Active Learning for Regression

David Holzmüller[1], Viktor Zaverkin[2], Johannes Kästner[2], and Ingo Steinwart[1]
Submitted to Journal of Machine Learning Research
Reference: Holzmüller et al. (2023), link: `https://arxiv.org/abs/2203.09410`

Abstract

The acquisition of labels for supervised learning can be expensive. To improve the sample efficiency of neural network regression, we study active learning methods that adaptively select batches of unlabeled data for labeling. We present a framework for constructing such methods out of (network-dependent) base kernels, kernel transformations, and selection methods. Our framework encompasses many existing Bayesian methods based on Gaussian process approximations of neural networks as well as non-Bayesian methods. Additionally, we propose to replace the commonly used last-layer features with sketched finite-width neural tangent kernels and to combine them with a novel clustering method. To evaluate different methods, we introduce an open-source benchmark consisting of 15 large tabular regression data sets. Our proposed method outperforms the state-of-the-art on our benchmark, scales to large data sets, and works out-of-the-box without adjusting the network architecture or training code. We provide open-source code that includes efficient implementations of all kernels, kernel transformations, and selection methods, and can be used for reproducing our results.

6.1 Introduction

While supervised machine learning (ML) has been successfully applied to many different problems, these successes often rely on the availability of large data sets for the problem at hand. In cases where labeling data is expensive, it is important to reduce the required number of labels. Such a reduction could be achieved through various means: First, finding

[1]ISA, University of Stuttgart, Stuttgart, Germany
[2]ITC, University of Stuttgart, Stuttgart, Germany

more sample-efficient supervised ML methods; second, applying data augmentation; third, leveraging information in unlabeled data via semi-supervised learning; fourth, leveraging information from related problems through transfer learning, meta-learning, or multi-task learning; and finally, appropriately selecting which data to label. Active learning (AL) takes the latter approach by using a trained model to choose the next data point to label (Settles, 2009). The need to retrain after every new label prohibits parallelized labeling methods and can be far too expensive, especially for neural networks (NNs), which are often slow to train. This problem can be resolved by batch mode active learning (BMAL) methods, which select multiple data points for labeling at once. When the supervised ML method is a deep NN, this is known as batch mode deep active learning (BMDAL) (Ren et al., 2021). Pool-based BMDAL refers to the setting where data points for labeling need to be chosen from a given finite set of points.

Supervised and unsupervised ML algorithms choose a model for given data. Multiple models can be compared on the same data using model selection techniques such as cross-validation. Such a comparison increases the training cost, but not the (potentially much larger) cost of labeling data. In contrast to supervised learning, AL is about choosing the data itself, with the goal to reduce labeling cost. However, different AL algorithms may choose different samples, and hence a comparison of N AL algorithms might increase labeling cost by a factor of up to N. Consequently, such a comparison is not sensible for applications where labeling is expensive. Instead, it is even more important to properly benchmark AL methods on tasks where labels are cheap to generate or a large number of labels is already available.

In the classification setting, NNs typically output uncertainties in the form of a vector of probabilities obtained through a softmax layer, while regression NNs typically output a scalar target without uncertainties. Therefore, many BMDAL algorithms only apply to one of the two settings. For classification, many BMDAL approaches have been proposed (Ren et al., 2021), and there exist at least some standard benchmark data sets like CIFAR-10 (Krizhevsky, 2009) on which methods are usually evaluated. On the other hand, the regression setting has been studied less frequently, and no common benchmark has been established to the best of our knowledge, except for a specialized benchmark in drug discovery (Mehrjou et al., 2021). We expect that the regression setting will gain popularity, not least due to the increasing interest in NNs for surrogate modeling (Behler, 2016; Kutz, 2017; Raissi et al., 2019; Mehrjou et al., 2021; Lavin et al., 2021).

6.1.1 Contributions

In this paper, we investigate pool-based BMDAL methods for regression. Our experiments use fully connected NNs on tabular data sets, but the considered methods can be generalized to different types of data and NN architectures. We limit our study to methods that do not require to modify the network architecture and training, as these are particularly easy to use and a fair comparison to other methods is difficult. We also focus on methods that scale to large amounts of (unlabeled) data and large acquisition batch sizes. Our contributions can be summarized as follows:

(1) We propose a framework for decomposing typical BM(D)AL algorithms into the choice of a kernel and a selection method. Here, the kernel can be constructed from a base kernel through a series of kernel transformations. The use of kernels as basic

building blocks allows for an efficient yet flexible and composable implementation of our framework, which we include in our open-source code. We also discuss how (regression variants of) many popular BM(D)AL algorithms can be represented in this framework and how they can efficiently be implemented. This gives us a variety of options for base kernels, kernel transformations, and selection methods to combine. Our framework encompasses both Bayesian methods based on Gaussian Processes and Laplace approximations as well as geometric methods.

(2) We discuss some alternative options to the ones arising from popular BM(D)AL algorithms: We introduce a novel selection method called LCMD; and we propose to combine the finite-width neural tangent kernel (NTK, Jacot et al., 2018) as a base kernel with sketching for efficient computation.

(3) We introduce an open-source benchmark for BMDAL involving 15 large tabular regression data sets. Using this benchmark, we compare different selection methods and evaluate the influence of the kernel, the acquisition batch size, and the target metric.

Our newly proposed selection method, LCMD, improves the state-of-the-art in our benchmark in terms of RMSE and MAE, while still exhibiting good performance for the maximum error. The NTK base kernel improves the benchmark accuracy for all selection methods, and the proposed sketching method can preserve this accuracy while leading to significant time gains. Figure 1 shows a comparison of our novel BMDAL algorithm against popular BMDAL algorithms from the literature, which are all implemented in our framework. The code for our framework and benchmark is based on PyTorch (Paszke et al., 2019) and is publicly available at

`https://github.com/dholzmueller/bmdal_reg`

and will be archived together with the generated data at `https://doi.org/10.18419/darus-3394`.

The rest of this paper is structured as follows: In Section 6.2, we introduce the basic problem setting of BMDAL for tabular regression with fully-connected NNs and introduce our framework for the construction of BMDAL algorithms. We discuss related work in Section 6.3. We then introduce options to build kernels from base kernels and kernel transformations in Section 6.4. Section 6.5 discusses various iterative kernel-based selection methods. Our experiments in Section 6.6 provide insights into the performance of different combinations of kernels and selection methods. Finally, we discuss limitations and open questions in Section 6.7. More details on the presented methods and experimental results are provided in the Appendix, whose structure is outlined in Appendix 6.A.

6.2 Problem Setting

In this section, we outline the problem of BMDAL for regression with fully-connected NNs. We first introduce the regression objective and fully-connected NNs. Subsequently, we introduce the basic setup of pool-based BMDAL as well as our proposed framework.

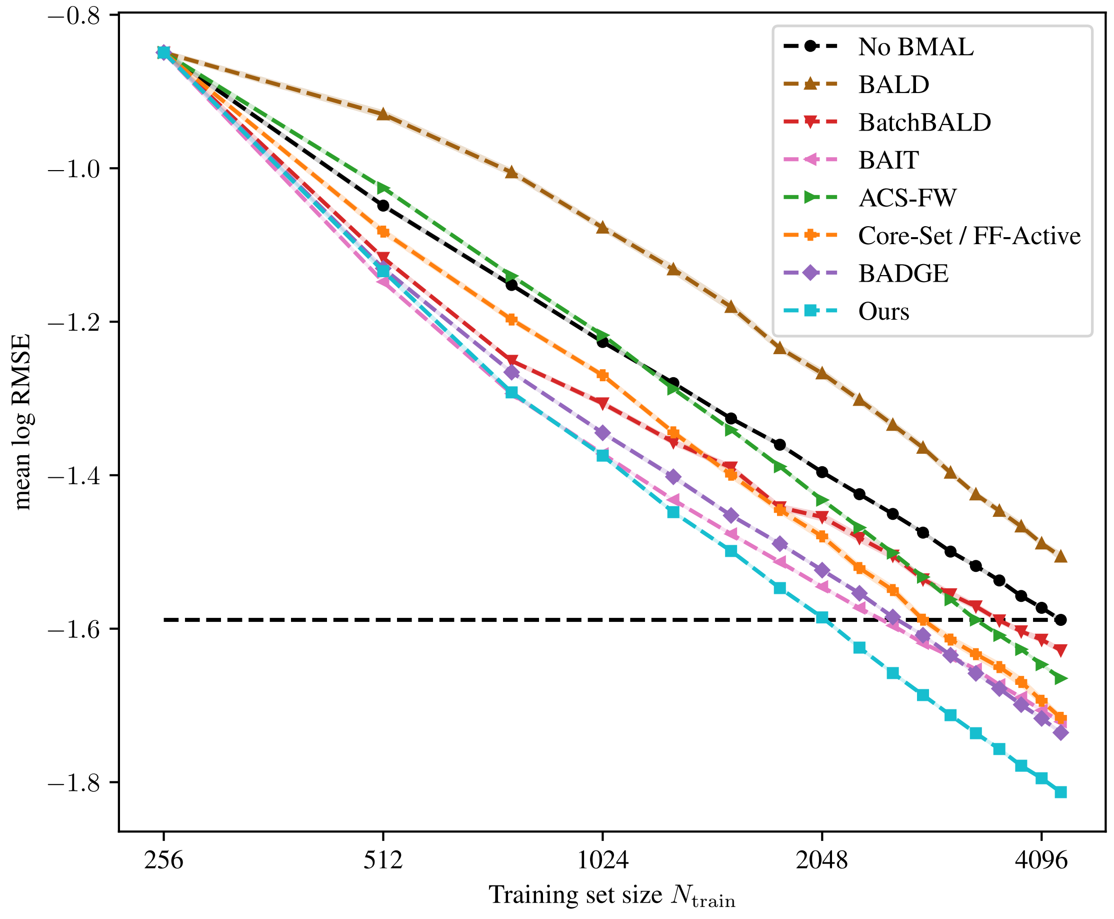


Figure 1: This figure shows how fast the averaged errors on our benchmark data sets decrease during BMAL for random selection (no BMAL), BALD (Houlsby et al., 2011), BatchBALD (Kirsch et al., 2019), BAIT (Ash et al., 2021), ACS-FW (Pinsler et al., 2019), Core-Set (Sener and Savarese, 2018), FF-Active (Geifman and El-Yaniv, 2017), BADGE (Ash et al., 2019), and our method. In Table 5 and Section 6.6, we specify how the compared methods are built from components explained in Section 6.4 and Section 6.5, and discuss further details such as modifications to apply them to regression. For the plot, we start with 256 random training samples and select 256 samples in each of 16 BMAL steps. The lines show the average of the logarithmic RMSE over all 15 benchmark data sets and 20 random splits between the BMAL steps. The shaded area, which is barely visible, corresponds to one estimated standard deviation of the mean estimator, cf. Section 6.E.4.

6.2.1 Regression with Fully-Connected Neural Networks

We consider multivariate regression, where the goal is to learn a function $f : \mathbb{R}^d \to \mathbb{R}$ from data $(\boldsymbol{x}, y) \in \mathcal{D}_{\text{train}} \subseteq \mathbb{R}^d \times \mathbb{R}$. In the case of NNs, we consider a parameterized function family $(f_{\boldsymbol{\theta}})_{\boldsymbol{\theta} \in \mathbb{R}^m}$ and try to minimize the mean squared loss on training data $\mathcal{D}_{\text{train}}$ with N_{train} samples:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N_{\text{train}}} \sum_{(\boldsymbol{x}, y) \in \mathcal{D}_{\text{train}}} (y - f_{\boldsymbol{\theta}}(\boldsymbol{x}))^2 .$$

We refer to the inputs and labels in $\mathcal{D}_{\text{train}}$ as $\mathcal{X}_{\text{train}}$ and $\mathcal{Y}_{\text{train}}$, respectively. Corresponding data sets are often referred to as *tabular data* or *structured data*. This is in contrast to data with a known spatiotemporal relationship between the input features, such as image or time-series data, where specialized NN architectures such as CNNs are more successful.

For our derivations and experiments, we consider an L-layer fully-connected NN $f_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}$ with parameter vector $\boldsymbol{\theta} = (\boldsymbol{W}^{(1)}, \boldsymbol{b}^{(1)}, \ldots, \boldsymbol{W}^{(L)}, \boldsymbol{b}^{(L)})$ and input size $d_0 = d$, hidden layer sizes $d_1, \ldots, d_{L-1}$, and output size $d_L = 1$. The value $\boldsymbol{z}_i^{(L)} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(0)})$ of the NN on the i-th input $\boldsymbol{x}_i^{(0)} \in \mathbb{R}^{d_0}$ is defined recursively by

$$\boldsymbol{x}_i^{(l+1)} = \varphi(\boldsymbol{z}_i^{(l+1)}) \in \mathbb{R}^{d_{l+1}}, \qquad \boldsymbol{z}_i^{(l+1)} = \frac{\sigma_w}{\sqrt{d_l}} \boldsymbol{W}^{(l+1)} \boldsymbol{x}_i^{(l)} + \sigma_b \boldsymbol{b}^{(l+1)} \in \mathbb{R}^{d_{l+1}} . \tag{6.1}$$

Here, the activation function $\varphi : \mathbb{R} \to \mathbb{R}$ is applied element-wise and $\sigma_w, \sigma_b > 0$ are constant factors. In our experiments, the weight matrices are initialized with independent standard normal entries and the biases are initialized to zero. The factors $\sigma_w/\sqrt{d_l}$ and σ_b stem from the neural tangent parametrization (NTP) (Jacot et al., 2018; Lee et al., 2019), which is theoretically motivated to define infinite-width limits of NNs and is also used in our applications. However, our derivations apply analogously to NNs without these factors. When considering different NN types such as CNNs, it is possible to apply our derivations only to the fully-connected part of the NN or to extend them to other layers as well.

6.2.2 Batch Mode Active Learning

In a single BMAL step, a BMAL algorithm selects a batch $\mathcal{X}_{\text{batch}} \subseteq \mathbb{R}^d$ with a given size $N_{\text{batch}} \in \mathbb{N}$. Subsequently, this batch is labeled and added to the training set. Here, we consider *pool-based* BMAL, where $\mathcal{X}_{\text{batch}}$ is to be selected from a given finite *pool set* $\mathcal{X}_{\text{pool}}$ of candidates. Other AL paradigms include membership query AL, where data points for labeling can be chosen freely, or stream-based AL, where data points arrive sequentially and must be immediately labeled or discarded. The pool set can potentially contain information about which regions of the input space are more important than others, especially if it is drawn from the same distribution as the test set. Moreover, pool-based BMAL allows for efficient benchmarking of BMAL methods on labeled data sets by reserving a large portion of the data set for the pool set, rendering the labeling part trivial.

When comparing and evaluating BMDAL methods, we are mainly interested in the following desirable properties:

(P1) The method should improve the sample efficiency of the underlying NN, even for large acquisition batch sizes N_{batch} and large pool set sizes N_{pool}, with respect to the

Algorithm 1 Basic pool-based BMDAL loop with initial labeled training set $\mathcal{D}_{\text{train}}$, unlabeled pool set $\mathcal{X}_{\text{pool}}$, BMDAL algorithm NextBatch (see Algorithm 2) and a list L_{batch} of batch sizes.

```
for AL batch size $N_{\text{batch}}$ in $L_{\text{batch}}$ do
    Train model $f_{\boldsymbol{\theta}}$ on $\mathcal{D}_{\text{train}}$
    Select batch $\mathcal{X}_{\text{batch}} \leftarrow \textsc{NextBatch}(f_{\boldsymbol{\theta}}, \mathcal{D}_{\text{train}}, \mathcal{X}_{\text{pool}}, N_{\text{batch}})$ with $|\mathcal{X}_{\text{batch}}| = N_{\text{batch}}$ and $\mathcal{X}_{\text{batch}} \subseteq \mathcal{X}_{\text{pool}}$
    Move $\mathcal{X}_{\text{batch}}$ from $\mathcal{X}_{\text{pool}}$ to $\mathcal{D}_{\text{train}}$ and acquire labels $\mathcal{Y}_{\text{batch}}$ for $\mathcal{X}_{\text{batch}}$
end for
Train final model $f_{\boldsymbol{\theta}}$ on $\mathcal{D}_{\text{train}}$
```

downstream application, which may or may not involve the same input distribution as training and pool data.

(P2) The method should scale to large pool sets, training sets, and batch sizes, in terms of both computation time and memory consumption.

(P3) The method should be applicable to a wide variety of NN architectures and training methods, such that it can be applied to different use cases.

(P4) The method should not require modifying the NN architecture and training method, for example by requiring to introduce Dropout, such that practitioners do not have to worry whether employing the method diminishes the accuracy of their trained NN.

(P5) The method should not require training multiple NNs for a single batch selection since this would deteriorate its runtime efficiency.[3]

(P6) The method should not require tuning hyperparameters on the downstream application since this would require labeling samples selected with suboptimal hyperparameters.

Property (P1) is central to motivate the use of BMDAL over random sampling of the data and is evaluated for our framework in detail in Section 6.6 and Appendix 6.E. We only evaluate methods with property (P2) since our benchmark involves large data sets. All methods considered here satisfy (P3) to a large extent. Indeed, although efficient computations are only studied for fully-connected layers here, the considered methods can be simply applied to the fully-connected part of a larger NN. All considered methods satisfy (P4), which also facilitates fair comparison in a benchmark. All our methods satisfy (P5), although our methods can incorporate ensembles of NNs. Although some of the considered methods have hyperparameters, we fix them to reasonable values independent of the data set in our experiments, such that (P6) is satisfied.

Algorithm 1 shows how BMDAL algorithms satisfying (P4) and (P5) can be used in a loop with training and labeling.

Wu (2018) formulates three criteria by which BMAL algorithms may select batch samples in order to improve the sample efficiency of a learning method:

(INF) The algorithm should favor inputs that are *informative* to the model. These could, for example, be those inputs where the model is most uncertain about the label.

[3]Technically, requiring multiple trained NNs would not be detrimental if it facilitated reaching the same accuracy with correspondingly larger N_{batch}.

Algorithm 2 Kernel-based batch construction framework

```
function KernelNextBatch($f_{\boldsymbol{\theta}}, \mathcal{D}_{\text{train}}, \mathcal{X}_{\text{pool}}, N_{\text{batch}}$)
    $k \leftarrow$ BaseKernel($f_{\boldsymbol{\theta}}$)
    $k \leftarrow$ TransformKernel($k, \mathcal{D}_{\text{train}}$)
    return Select($k, \mathcal{X}_{\text{train}}, \mathcal{X}_{\text{pool}}, N_{\text{batch}}$)
end function
```

(DIV) The algorithm should ensure that the batch contains *diverse* samples, i.e., samples in the batch should be sufficiently different from each other.

(REP) The algorithm should ensure *representativity* of the resulting training set, i.e., it should focus more strongly on regions where the pool data distribution has high density.

Note that (REP) might not be desirable if one expects a significant distribution shift between pool and test data. A challenge in trying to adapt non-batch AL methods to the batch setting is that some non-batch AL methods expect to immediately receive a label for every selected sample. It is usually possible to circumvent this by selecting the N_{batch} samples with the largest acquisition function scores at once, but this does not enforce (DIV) or (REP).

We propose a framework for assembling BMDAL algorithms that is shown in Algorithm 2 and consists of three components: First, a base kernel k needs to be chosen that should serve as a proxy for the trained network $f_{\boldsymbol{\theta}}$. Second, the kernel can be transformed using various transformations. These transformations can, for example, make the kernel represent posteriors or improve its evaluation efficiency. Third, a selection method is invoked that uses the transformed kernel as a measure of similarity between inputs. When using Gaussian Process regression with a given kernel k as a supervised learning method instead of an NN, the base kernel could simply be chosen as k. Note that Select does not observe the training labels directly, however, in the NN setting, these can be implicitly incorporated through kernels that depend on the trained NN.

Example 6.2.1. In Algorithm 2, the base kernel k could be of the form $k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \langle \phi(\boldsymbol{x}), \phi(\tilde{\boldsymbol{x}}) \rangle$, where ϕ represents the trained NN without the last layer. When interpreting k as the kernel of a Gaussian process, TransformKernel could then compute a transformed kernel $\tilde{k}$ that represents the posterior predictive uncertainty after observing the training data. Finally, Select could then choose the N_{batch} points $\boldsymbol{x} \in \mathcal{X}_{\text{pool}}$ with the largest uncertainty $\tilde{k}(\boldsymbol{x}, \boldsymbol{x})$. ◀

From a Bayesian perspective, our choice of kernel and kernel transformations can correspond to inference in a Bayesian approximation, as we discuss in Section 6.C.1, while the selection method can correspond to the optimization of an acquisition function. However, in our framework, the same "Bayesian" kernels can be used together with non-Bayesian selection methods and vice versa.

6.3 Related Work

The field of active learning, also known as query learning or sequential (optimal) experimental design (Fedorov, 1972; Chaloner and Verdinelli, 1995), has a long history dating back at least to the beginning of the 20th century (Smith, 1918). For an overview of the AL and BMDAL literature, we refer to Settles (2009); Kumar and Gupta (2020); Ren et al. (2021); Weng (2022).

We first review work relevant to the kernels in our framework, before discussing work more relevant to selection methods, and finally, data sets. More literature related to specific methods is also discussed in Section 6.4 and Section 6.5.

6.3.1 Uncertainty Measures and Kernel Approximations

A popular class of BMDAL methods is given by Bayesian methods since the Bayesian framework naturally provides uncertainties that can be used to assess informativeness. These methods require to use Bayesian NNs, or in other words, the calculation of an approximate posterior distribution over NN parameters. A simple option is to perform Bayesian inference only over the last layer of the NN (Lázaro-Gredilla and Figueiras-Vidal, 2010; Snoek et al., 2015; Ober and Rasmussen, 2019; Kristiadi et al., 2020). The Laplace approximation (Laplace, 1774; MacKay, 1992a) can provide a local posterior distribution around a local optimum of the loss landscape via a second-order Taylor approximation. An alternative local approach based on SGD iterates is called SWAG (Maddox et al., 2019). Ensembles of NNs (Hansen and Salamon, 1990; Lakshminarayanan et al., 2017) can be interpreted as a simple multi-modal posterior approximation and can be combined with local approximations to yield mixtures of Laplace approximations (Eschenhagen et al., 2021) or MultiSWAG (Wilson and Izmailov, 2020). Monte Carlo (MC) Dropout (Gal and Ghahramani, 2016) is an option to obtain ensemble predictions from a single NN, although it requires training with Dropout (Srivastava et al., 2014). Regarding uncertainty approximations, our considered algorithms are mainly related to exact last-layer methods and the Laplace approximation, as these do not require to modify the training process. Daxberger et al. (2021) give an overview of various methods to compute (approximate) Laplace approximations.

Some recent approaches also build on the neural tangent kernel (NTK) introduced by Jacot et al. (2018). Khan et al. (2019) show that certain Laplace approximations are related to the finite-width NTK. Wang et al. (2022) and Mohamadi et al. (2022) propose the use of finite-width NTKs for DAL for classification. Wang et al. (2021) use the finite-width NTK at initialization for the streaming setting of DAL for classification and theoretically analyze the resulting method. Aljundi et al. (2022) use a kernel related to the finite-width NTK for DAL. Shoham and Avron (2023), Borsos et al. (2020) and Borsos et al. (2021) use infinite-width NTKs for BMDAL and related tasks. Han et al. (2021) propose sketching for infinite-width NTKs and also evaluate it for DAL. In contrast to these papers, we propose sketching for finite-width NTKs and allow combining the resulting kernel with different selection methods.

6.3.2 Selection Methods

Besides a Bayesian NN model, a Bayesian BMDAL method needs to specify an acquisition function that decides how to prioritize the pool samples. Many simple acquisition functions for quantifying uncertainty have been proposed (Kumar and Gupta, 2020). Selecting the next sample where an ensemble disagrees most is known as Query-by-Committee (QbC)

(Seung et al., 1992). Krogh and Vedelsby (1994) employed QbC for DAL for regression. A more recent investigation of QbC to DAL for classification is performed by Beluch et al. (2018). Pop and Fulop (2018) combine ensembles with MC Dropout. Tsymbalov et al. (2018) use the predictive variance obtained by MC Dropout for DAL for regression. Zaverkin and Kästner (2021) use last-layer-based uncertainty in DAL for regression on atomistic data. Unlike the other approaches mentioned before, the approach by Zaverkin and Kästner (2021) can be applied to a single NN trained without Dropout.

Many uncertainty-based acquisition functions do not distinguish between epistemic uncertainty, i.e., lack of knowledge about the true functional relationship, and aleatoric uncertainty, i.e., inherent uncertainty due to label noise. Houlsby et al. (2011) propose the BALD acquisition function, which aims to quantify epistemic uncertainty only. Gal et al. (2017) apply BALD and other acquisition functions to BMDAL for classification with MC Dropout. To enforce diversity of the selected batch, Kirsch et al. (2019) propose BatchBALD and evaluate it on classification problems with MC Dropout. Ash et al. (2021) propose Bait, which also incorporates representativity through Fisher information based on last-layer features, and is evaluated on classification and regression data sets.

Another approach towards BMDAL is to find core-sets that represent $\mathcal{X}_{\text{pool}}$ in a geometric sense. Sener and Savarese (2018) and Geifman and El-Yaniv (2017) propose algorithms to cover the pool set with $\mathcal{X}_{\text{batch}} \cup \mathcal{X}_{\text{train}}$ in a last-layer feature space. Ash et al. (2019) propose BADGE, which applies clustering in a similar feature space, but includes uncertainty via gradients through the softmax layer for classification. ACS-FW (Pinsler et al., 2019) can be seen as a hybrid between core-set and Bayesian approaches, trying to approximate the expected log-posterior on the pool set with a core-set, also using last-layer-based Bayesian approximations. Besides Bait, ACS-FW is one of the few approaches that is designed and evaluated for both classification and regression. Our newly proposed selection method LCMD is clustering-based like the k-means++ method used in BADGE, but deterministic.

Many more approaches towards BMDAL exist, and they can be combined with additional steps such as pre-reduction of $\mathcal{X}_{\text{pool}}$ (Ghorbani et al., 2022) or re-weighting of selected instances (Farquhar et al., 2021). Most of these BMDAL methods are geared towards classification, and for a broader overview, we refer to Ren et al. (2021). For (image) regression, Ranganathan et al. (2020) introduce an auxiliary loss term on the pool set, which they use to perform DAL. It is unclear, though, to which extent their success is explained by implicitly performing semi-supervised learning.

Since we frequently consider Gaussian Processes (GPs) as approximations to Bayesian NNs in this paper, our work is also related to BMAL for GPs, although in our case the GPs are only used for selecting $\mathcal{X}_{\text{batch}}$ and not for the regression itself. Popular BMAL methods for GPs have been suggested for example by Seo et al. (2000) and Krause et al. (2008).

6.3.3 Data Sets

In terms of benchmark data sets for BM(D)AL for regression, Tsymbalov et al. (2018) use seven large tabular data sets, some of which we have included in our benchmark, cf. Section 6.E.1. Pinsler et al. (2019) use only one large tabular regression data set. Ash et al. (2021) use a small tabular regression data set and three image regression data sets,

Base kernel	Symbol	Feature map	Feature space dimension d_{feat}
Linear	k_{lin}	$\phi_{\text{lin}}(\boldsymbol{x}) = \boldsymbol{x}$	d
NNGP	k_{nngp}	not explicitly defined	∞
full gradient	k_{grad}	$\phi_{\text{grad}}(\boldsymbol{x}) = \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_T}(\boldsymbol{x})$	$\sum_{l=1}^{L} d_l(d_{l-1}+1)$
last-layer	k_{ll}	$\phi_{\text{ll}}(\boldsymbol{x}) = \nabla_{\tilde{\boldsymbol{W}}^{(L)}} f_{\boldsymbol{\theta}_T}(\boldsymbol{x})$	$d_L(d_{L-1}+1)$

Table 1: Overview of the introduced base kernels.

two of which are converted classification data sets. Wu (2018) benchmarks exclusively on small tabular data sets. Zaverkin and Kästner (2021) work with atomistic data sets, which require specialized NN architectures and longer training times, and are therefore less well-suited for a large-scale benchmark. Ranganathan et al. (2020) use CNNs on five image regression data sets. Recently, a benchmark for BMDAL for drug discovery has been proposed, which uses four counterfactual regression data sets (Mehrjou et al., 2021). In this paper, we provide an open-source benchmark on 15 large tabular data sets, which includes more baselines and evaluation criteria than evaluations in previous papers.

6.4 Kernels

In this section, we discuss a variety of base kernels yielding various approximations to a trained NN $f_{\boldsymbol{\theta}_T}$, as well as different kernel transformations that yield new kernels with different meanings or simply improved efficiency. In the following, we consider positive semi-definite kernels $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. For an introduction to kernels, we refer to the literature (e.g. Steinwart and Christmann, 2008). The kernels considered here can usually be represented by a feature map ϕ with finite-dimensional feature space, that is, $\phi : \mathbb{R}^d \to \mathbb{R}^{d_{\text{feat}}}$ with $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$. For a sequence $\mathcal{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ of inputs, which we sometimes treat like a set $\mathcal{X} \subseteq \mathbb{R}^d$ by a slight abuse of notation, we define the corresponding feature matrix

$$\phi(\mathcal{X}) = \begin{pmatrix} \phi(\boldsymbol{x}_1)^\top \\ \vdots \\ \phi(\boldsymbol{x}_n)^\top \end{pmatrix} \in \mathbb{R}^{n \times d_{\text{feat}}} \tag{6.2}$$

and kernel matrices $k(\boldsymbol{x}, \mathcal{X}) = (k(\boldsymbol{x}, \boldsymbol{x}_i))_i \in \mathbb{R}^{1 \times n}$, $k(\mathcal{X}, \mathcal{X}) = (k(\boldsymbol{x}_i, \boldsymbol{x}_j))_{ij} \in \mathbb{R}^{n \times n}$, $k(\mathcal{X}, \boldsymbol{x}) = (k(\boldsymbol{x}_i, \boldsymbol{x}))_i \in \mathbb{R}^{n \times 1}$.

6.4.1 Base Kernels

We first discuss various options for creating base kernels that induce some notion of similarity on the training and pool inputs. An overview of these base kernels can be found in Table 1.

Linear Kernel

A very simple baseline for other base kernels is the linear kernel $k_{\text{lin}}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \langle \boldsymbol{x}, \tilde{\boldsymbol{x}} \rangle$, corresponding to the identity feature map

$$\phi_{\text{lin}}(\boldsymbol{x}) \coloneqq \boldsymbol{x} \ .$$

It is usually very fast to evaluate but does not represent the behavior of an NN well. Moreover, its feature space dimension depends on the input dimension, and hence may not be suited for selection methods that depend on having high-dimensional representations of the data. A more accurate representation of the behavior of an NN is given by the next kernel:

Full gradient Kernel

If $\boldsymbol{\theta}_T$ is the parameter vector of the trained NN, we define

$$\phi_{\text{grad}}(\boldsymbol{x}) \coloneqq \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_T}(\boldsymbol{x}) \ .$$

This is motivated as follows: A linearization of the NN with respect to its parameters around $\boldsymbol{\theta}_T$ is given by the first-order Taylor expansion

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) \approx \tilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) \coloneqq f_{\boldsymbol{\theta}_T}(\boldsymbol{x}) + \langle \phi_{\text{grad}}(\boldsymbol{x}), \boldsymbol{\theta} - \boldsymbol{\theta}_T \rangle \ . \tag{6.3}$$

If we were to resume training from the parameters $\boldsymbol{\theta}_T$ after labeling the next batch $\mathcal{X}_{\text{batch}}$, the result of training on the extended data could hence be approximated by the function $f_{\boldsymbol{\theta}_T} + f_\Delta$, where f_Δ is the result of linear regression with feature map ϕ_{grad} on the data residuals $(\boldsymbol{x}_i, y_i - f_{\boldsymbol{\theta}_T}(\boldsymbol{x}_i))$ for $(\boldsymbol{x}_i, y_i) \in \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{batch}}$.

The kernel k_{grad} is also known as the (empirical / finite-width) *neural tangent kernel* (NTK). It depends on the linearization point $\boldsymbol{\theta}_T$, but can for certain training settings converge to a fixed kernel as the hidden layer widths go to infinity (Jacot et al., 2018; Lee et al., 2019; Arora et al., 2019). In practical settings, however, it has been observed that k_{grad} often “improves” during training (Fort et al., 2020; Long, 2021; Shan and Bordelon, 2021; Atanasov et al., 2021), especially in the beginning of training. This agrees with our observations in Section 6.6 and suggests that shorter training might already yield a gradient kernel that allows selecting a good $\mathcal{X}_{\text{batch}}$. Indeed, Coleman et al. (2019) found that shorter training and even smaller models can already be sufficient to select good batches for BMDAL for classification.

For fully-connected layers, we will now show that the feature map ϕ_{grad} has an additional product structure that can be exploited to reduce the runtime and memory consumption of a kernel evaluation. For notational simplicity, we rewrite Eq. (6.1) as

$$\begin{aligned} \boldsymbol{z}_i^{(l+1)} &= \tilde{\boldsymbol{W}}^{(l+1)} \tilde{\boldsymbol{x}}_i^{(l)}, \\ \tilde{\boldsymbol{W}}^{(l+1)} &\coloneqq \begin{pmatrix} \boldsymbol{W}^{(l+1)} & \boldsymbol{b}^{(l+1)} \end{pmatrix} \in \mathbb{R}^{d_{l+1} \times (d_l + 1)}, \quad \tilde{\boldsymbol{x}}_i^{(l)} \coloneqq \begin{pmatrix} \frac{\sigma_w}{\sqrt{d_l}} \boldsymbol{x}_i^{(l)} \\ \sigma_b \end{pmatrix} \in \mathbb{R}^{d_l + 1} \ , \end{aligned} \tag{6.4}$$

with parameters $\boldsymbol{\theta} = (\tilde{\boldsymbol{W}}^{(1)}, \ldots, \tilde{\boldsymbol{W}}^{(L)})$. Using the notation from Eq. (6.4), we can write

$$\phi_{\text{grad}}(\boldsymbol{x}_i^{(0)}) = \left(\frac{\mathrm{d}\boldsymbol{z}_i^{(L)}}{\mathrm{d}\tilde{\boldsymbol{W}}^{(1)}}, \ldots, \frac{\mathrm{d}\boldsymbol{z}_i^{(L)}}{\mathrm{d}\tilde{\boldsymbol{W}}^{(L)}} \right) = \left(\frac{\mathrm{d}\boldsymbol{z}_i^{(L)}}{\mathrm{d}\boldsymbol{z}_i^{(1)}} (\tilde{\boldsymbol{x}}_i^{(0)})^\top, \ldots, \frac{\mathrm{d}\boldsymbol{z}_i^{(L)}}{\mathrm{d}\boldsymbol{z}_i^{(L)}} (\tilde{\boldsymbol{x}}_i^{(L-1)})^\top \right) \ . \tag{6.5}$$

For a kernel evaluation, the factorization of the weight matrix derivatives can be exploited via

$$\begin{aligned}
k_{\text{grad}}(\boldsymbol{x}_i^{(0)}, \boldsymbol{x}_j^{(0)}) &= \sum_{l=1}^{L} \left\langle \frac{\mathrm{d}\boldsymbol{z}_i^{(L)}}{\mathrm{d}\boldsymbol{z}_i^{(l)}} (\tilde{\boldsymbol{x}}_i^{(l-1)})^\top, \frac{\mathrm{d}\boldsymbol{z}_j^{(L)}}{\mathrm{d}\boldsymbol{z}_j^{(l)}} (\tilde{\boldsymbol{x}}_j^{(l-1)})^\top \right\rangle_F \\
&= \sum_{l=1}^{L} \underbrace{\left\langle \tilde{\boldsymbol{x}}_i^{(l-1)}, \tilde{\boldsymbol{x}}_j^{(l-1)} \right\rangle}_{=:k_{\text{in}}^{(l)}(\boldsymbol{x}_i^{(0)}, \boldsymbol{x}_j^{(0)})} \cdot \underbrace{\left\langle \frac{\mathrm{d}\boldsymbol{z}_i^{(L)}}{\mathrm{d}\boldsymbol{z}_i^{(l)}}, \frac{\mathrm{d}\boldsymbol{z}_j^{(L)}}{\mathrm{d}\boldsymbol{z}_j^{(l)}} \right\rangle}_{=:k_{\text{out}}^{(l)}(\boldsymbol{x}_i^{(0)}, \boldsymbol{x}_j^{(0)})}, \qquad (6.6)
\end{aligned}$$

since $\langle \boldsymbol{a}\boldsymbol{b}^\top, \boldsymbol{c}\boldsymbol{d}^\top \rangle_F = \operatorname{tr}(\boldsymbol{b}\boldsymbol{a}^\top \boldsymbol{c}\boldsymbol{d}^\top) = \operatorname{tr}(\boldsymbol{a}^\top \boldsymbol{c}\boldsymbol{d}^\top \boldsymbol{b}) = \boldsymbol{a}^\top \boldsymbol{c}\boldsymbol{d}^\top \boldsymbol{b} = \langle \boldsymbol{a}, \boldsymbol{c} \rangle \cdot \langle \boldsymbol{b}, \boldsymbol{d} \rangle$. This means that k_{grad} can be decomposed into sums of products of kernels with smaller feature space dimension:[4]

$$k_{\text{grad}}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \sum_{l=1}^{L} k_{\text{in}}^{(l)}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \cdot k_{\text{out}}^{(l)}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \qquad (6.7)$$

When using Eq. (6.6), the full gradients $\frac{\mathrm{d}\boldsymbol{z}_i^{(L)}}{\mathrm{d}\tilde{\boldsymbol{W}}^{(l)}}$ never have to be computed or stored explicitly. If $\frac{\mathrm{d}\boldsymbol{z}_i^{(L)}}{\mathrm{d}\boldsymbol{z}_i^{(l)}}$ and $\tilde{\boldsymbol{x}}_i^{(l-1)}$ are already computed and the hidden layers contain $m = d_1 = \ldots = d_{L-1}$ neurons each, Eq. (6.6) reduces the runtime complexity of a kernel evaluation from $\Theta(m^2 L)$ to $\Theta(mL)$, and similarly for the memory complexity of pre-computed features. In Section 6.4.2, we will see how to further accelerate this kernel computation using sketching. Efficient computations of k_{grad} for more general types of layers and multiple output neurons are discussed by Novak et al. (2022).

Since k_{grad} consists of gradient contributions from multiple layers, it is potentially important that the magnitudes of the gradients in different layers are balanced. We achieve this, at least at initialization, through the use of the neural tangent parameterization (Jacot et al., 2018). For other NN architectures, however, it might be desirable to re-weight gradient magnitudes from different layers to improve the results obtained with k_{grad}.

Last-layer Kernel

A simple and rough approximation to the full-gradient kernel is given by only considering the gradient with respect to the parameters in the last layer:

$$\phi_{\text{ll}}(\boldsymbol{x}) := \nabla_{\tilde{\boldsymbol{W}}^{(L)}} f_{\boldsymbol{\theta}_T}(\boldsymbol{x}) \ .$$

From Eq. (6.5), it is evident that in the single-output regression case that we are considering, $\phi_{\text{ll}}(\boldsymbol{x}_i^{(0)})$ is simply the input $\tilde{\boldsymbol{x}}_i^{(L-1)}$ to the last layer of the NN. The latter formulation can also be used in the multi-output setting, and versions of it (with $\boldsymbol{x}_i^{(L-1)}$ instead of $\tilde{\boldsymbol{x}}_i^{(L-1)}$) have been frequently used for BMDAL (Sener and Savarese, 2018; Geifman and El-Yaniv, 2017; Pinsler et al., 2019; Ash et al., 2019; Zaverkin and Kästner, 2021; Ash et al., 2021).

[4]For the sketching method defined later, we may exploit that $k_{\text{out}}^{(L)}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = 1$, hence $k_{\text{out}}^{(L)}$ can be omitted.

Infinite-width NNGP

It has been shown that as the widths $d_1, \dots, d_{L-1}$ of the hidden NN layers converge to infinity, the distribution of the initial function $f_{\boldsymbol{\theta}_0}$ converges to a Gaussian Process with mean zero and a covariance kernel k_{nngp} called the neural network Gaussian process (NNGP) kernel (Neal, 1994; Lee et al., 2018; Matthews et al., 2018). This kernel depends on the network depth, the used activation function, and details such as the initialization variance and scaling factors like σ_w. In our experiments, we use the NNGP kernel corresponding to the employed NN setup, for which the formulas are given in Section 6.B.1.

As mentioned above, there exists an infinite-width limit of k_{grad}, the so-called neural tangent kernel (Jacot et al., 2018). We decided to omit it from our experiments in Appendix 6.E after preliminary experiments showed similarly bad performance as for the NNGP.

6.4.2 Kernel Transformations

The base kernels introduced in Section 6.4.1 are constructed such that kernel regression with these kernels serves as a proxy for regression with the corresponding NN. By using kernels, we can model interactions $k(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ between two inputs, which is crucial to incorporate diversity (DIV) into the selection methods. However, this is not always sufficient to apply a selection method. For example, sometimes we want the kernel to represent uncertainties of the NN after observing the data, or we want to reduce the feature space dimension to render selection more efficient. Therefore, we introduce various ways to transform kernels in this section. When applying transformations $T_1, \dots, T_n$ in this order to a base kernel k_{base}, we denote the transformed kernel by $k_{\mathrm{base} \to T_1 \to T_2 \to \dots \to T_n}$. Of course, we can only cover selected transformations relevant to our applications, and other transformations such as sums or products of kernels are possible as well.

Scaling

For a given kernel k with feature map ϕ and scaling factor $\lambda \in \mathbb{R}$, we can construct the kernel $\lambda^2 k$ with feature map $\lambda\phi$. This scaling can make a difference if we subsequently consider a Gaussian Process (GP) with covariance function $\lambda^2 k$. In this case, $\lambda^2 k(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ describes the covariance between $f(\boldsymbol{x})$ and $f(\tilde{\boldsymbol{x}})$ under the prior distribution over functions f. Since we train with normalized labels, $N_{\mathrm{train}}^{-1} \sum_{y \in \mathcal{Y}_{\mathrm{train}}} y_i^2 \approx 1$, we would like to choose the scaling factor λ such that $N_{\mathrm{train}}^{-1} \sum_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{train}}} \lambda^2 k(\boldsymbol{x}, \boldsymbol{x}) = 1$. Therefore, we propose the automatic scale normalization

$$k_{\to \mathrm{scale}(\mathcal{X}_{\mathrm{train}})}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \coloneqq \lambda^2 k(\boldsymbol{x}, \tilde{\boldsymbol{x}}), \qquad \lambda \coloneqq \left(\frac{1}{N_{\mathrm{train}}} \sum_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{train}}} k(\boldsymbol{x}, \boldsymbol{x}) \right)^{-1/2} .$$

Gaussian Process Posterior Transformation

For a given kernel k with corresponding feature map ϕ, we can consider a Gaussian Process (GP) with kernel k, which is equivalent to a Bayesian linear regression model with feature map ϕ: In feature space, we model our observations as $y_i = \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) + \varepsilon_i$ with weight prior

Notation	Description	d_{pre}	d_{post}	Configurable σ^2?
$k_{\to\text{scale}(\mathcal{X})}$	Rescale kernel to normalize mean $k(\boldsymbol{x}, \boldsymbol{x})$ on $\mathcal{X}$	any	d_{pre}	no
$k_{\to\text{post}(\mathcal{X},\sigma^2)}$	GP posterior covariance after observing $\mathcal{X}$	any	d_{pre}	yes
$k_{\to\mathcal{X}}$	Short for $k_{\to\text{scale}(\mathcal{X})\to\text{post}(\mathcal{X},\sigma^2)}$	any	d_{pre}	yes
$k_{\to\text{sketch}(p)}$	Sketching with p features	$< \infty$	p	no
$k_{\to\text{ens}(N_{\text{ens}})}$	Sum of kernels for N_{ens} ensembled networks	any	$N_{\text{ens}} d_{\text{pre}}$	no
$k_{\to\text{acs-grad}}$	Gradient-based kernel from Pinsler et al. (2019)	any	d_{pre}^2	yes
$k_{\to\text{acs-rf}(p)}$	Kernel from Pinsler et al. (2019) with p random features	$< \infty$	p	yes
$k_{\to\text{acs-rf-hyper}(p)}$	Kernel from Pinsler et al. (2019) with p random features and hyperprior on σ^2	$< \infty$	p	no

Table 2: Overview of our considered kernel transformations that can be applied to a kernel k. Here, d_{pre} refers to the feature space dimension of k and d_{post} refers to the feature space dimension after the transformation. Moreover, σ^2 refers to the assumed noise variance in the GP model.

$\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and i.i.d. observation noise $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. The random function $f(\boldsymbol{x}_i) \coloneqq \boldsymbol{w}^\top \phi(\boldsymbol{x}_i)$ now has the covariance function $\operatorname{Cov}(f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)) = \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

It is well-known, see e.g. Section 2.1 and 2.2 in Bishop (2006), that the posterior distribution of a Gaussian process after observing the training data $\mathcal{D}_{\text{train}}$ with inputs $\mathcal{X}_{\text{train}}$ is also a Gaussian process with kernel

$$\begin{aligned}
& k_{\to\text{post}(\mathcal{X}_{\text{train}},\sigma^2)}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \\
\coloneqq\ & \operatorname{Cov}(f(\boldsymbol{x}), f(\tilde{\boldsymbol{x}}) \mid \mathcal{X}_{\text{train}}, \mathcal{Y}_{\text{train}}) \\
=\ & k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) - k(\boldsymbol{x}, \mathcal{X}_{\text{train}})(k(\mathcal{X}_{\text{train}}, \mathcal{X}_{\text{train}}) + \sigma^2 \boldsymbol{I})^{-1} k(\mathcal{X}_{\text{train}}, \tilde{\boldsymbol{x}}) && (6.8) \\
\overset{\text{see below}}{=}\ & \phi(\boldsymbol{x})^\top (\sigma^{-2} \phi(\mathcal{X}_{\text{train}})^\top \phi(\mathcal{X}_{\text{train}}) + \boldsymbol{I})^{-1} \phi(\tilde{\boldsymbol{x}}) && (6.9) \\
=\ & \sigma^2 \phi(\boldsymbol{x})^\top (\phi(\mathcal{X}_{\text{train}})^\top \phi(\mathcal{X}_{\text{train}}) + \sigma^2 \boldsymbol{I})^{-1} \phi(\tilde{\boldsymbol{x}})\ . && (6.10)
\end{aligned}$$

Here, the equivalence between Eq. (6.8) and Eq. (6.9) for $\sigma^2 > 0$ can be obtained using the Woodbury matrix identity. In our implementation, we use the feature map version, Eq. (6.9), whenever $d_{\text{feat}} \leq \max\{1024, 3|\mathcal{X}_{\text{train}}|\}$. An explicit feature map can be obtained from Eq. (6.10) as

$$\phi_{\to\text{post}(\mathcal{X}_{\text{train}},\sigma^2)}(\boldsymbol{x}) = \sigma(\phi(\mathcal{X}_{\text{train}})^\top \phi(\mathcal{X}_{\text{train}}) + \sigma^2 \boldsymbol{I})^{-1/2} \phi(\boldsymbol{x})\ .$$

If the posterior with respect to two disjoint sets of inputs $\mathcal{X}_1, \mathcal{X}_2 \subseteq \mathbb{R}^d$ is sought, it is equivalent to condition first on $\mathcal{X}_1$ and then on $\mathcal{X}_2$:

$$k_{\to\text{post}(\mathcal{X}_1 \cup \mathcal{X}_2,\sigma^2)}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = k_{\to\text{post}(\mathcal{X}_1,\sigma^2)\to\text{post}(\mathcal{X}_2,\sigma^2)}(\boldsymbol{x}, \tilde{\boldsymbol{x}})\ . \qquad (6.11)$$

In our experiments, we rescale kernels before applying the posterior transformation, which we abbreviate by

$$k_{\to\mathcal{X}_{\text{train}}}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \coloneqq k_{\to\text{scale}(\mathcal{X}_{\text{train}})\to\text{post}(\mathcal{X}_{\text{train}},\sigma^2)}(\boldsymbol{x}, \tilde{\boldsymbol{x}})\ .$$

The application of the posterior transformation to network-dependent kernels can be seen as an instance of approximate inference with Bayesian NNs. Specifically, we show in Section 6.C.1 that $k_{\mathrm{ll}\to\mathrm{post}(\mathcal{X}_{\mathrm{train}},\sigma^2)}$ and $k_{\mathrm{grad}\to\mathrm{post}(\mathcal{X}_{\mathrm{train}},\sigma^2)}$ correspond to last-layer and generalized Gauss-Newton (GGN) approximations to the Hessian in a Laplace approximation (Laplace, 1774; MacKay, 1992a) for Bayesian NNs, see also Khan et al. (2019). Moreover, in the case of the last-layer kernel, this procedure is equivalent to interpreting the last layer of the NN as a Bayesian linear regression model.

Sketching

Sketching methods, which allow approximating matrices like $\phi(\mathcal{X})$ with smaller matrices in some sense, can be used to approximate a kernel k with high-dimensional feature space by a kernel with a lower-dimensional feature space (see e.g. Woodruff, 2014). For example, k_{grad} and kernels resulting from the ACS gradient transformation introduced in Section 6.4.2 involve product kernels with very high-dimensional feature spaces ($d_{\mathrm{feat}} > 250,000$). In our experiments, we apply sketching mainly to these kernels. This is especially useful for methods such as the posterior transformation discussed previously and the FrankWolfe and Bait selection methods explained in Section 6.5.2, which are not very efficient in the kernel formulation.

We sketch finite-dimensional feature maps as follows:

(1) **Generic finite-dimensional feature maps:** Consider a generic kernel k with finite-dimensional feature map $\phi : \mathbb{R}^{d_0} \to \mathbb{R}^{d_{\mathrm{pre}}}$. For a random vector $\boldsymbol{u} \sim \mathcal{N}(0, \boldsymbol{I}_{d_{\mathrm{pre}}})$, a single random feature is given by the feature map $\phi_{\boldsymbol{u}}(\boldsymbol{x}) := \boldsymbol{u}^\top \phi(\boldsymbol{x})$, which yields an unbiased estimate of the kernel since $\mathbb{E}_{\boldsymbol{u}}\langle \phi_{\boldsymbol{u}}(\boldsymbol{x}), \phi_{\boldsymbol{u}}(\tilde{\boldsymbol{x}})\rangle = k(\boldsymbol{x}, \tilde{\boldsymbol{x}})$. By combining multiple such random features, the accuracy of the kernel approximation can be improved. For p random features, we obtain the random feature map

$$\phi_{\to\mathrm{sketch}(p)}(\boldsymbol{x}) := \frac{1}{\sqrt{p}} \boldsymbol{U}\phi(\boldsymbol{x}) \in \mathbb{R}^p , \tag{6.12}$$

where $\boldsymbol{U} \in \mathbb{R}^{p\times d_{\mathrm{pre}}}$ is a random matrix with i.i.d. standard normal entries. This is also known as a Gaussian sketch.

In terms of the kernel distance

$$d_k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) := \|\phi(\boldsymbol{x}) - \phi(\tilde{\boldsymbol{x}})\|_2 = \sqrt{k(\boldsymbol{x}, \boldsymbol{x}) + k(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}) - 2k(\boldsymbol{x}, \tilde{\boldsymbol{x}})} , \tag{6.13}$$

the approximation quality of the sketched kernel can be analyzed using variants of the celebrated Johnson-Lindenstrauss lemma (Johnson and Lindenstrauss, 1984). For example, the following variant is proved in Section 6.C.2 based on a result by Arriaga and Vempala (1999).

Theorem 6.4.1 (Variant of the Johnson-Lindenstrauss Lemma)**.** *Let $\varepsilon, \delta \in (0, 1)$ and let $\mathcal{X} \subseteq \mathbb{R}^d$ be finite. If*

$$p \geq 8 \log(|\mathcal{X}|^2/\delta)/\varepsilon^2 , \tag{6.14}$$

then the following bound on all pairwise distances holds with probability $\geq 1 - \delta$ for the Gaussian sketch in Eq. (6.12):

$$\forall \boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathcal{X} : (1 - \varepsilon)d_k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \leq d_{k_{\to\mathrm{sketch}(p)}}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \leq (1 + \varepsilon)d_k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) . \tag{6.15}$$

Note that, counterintuitively, the lower bound in Eq. (6.14) does not depend on the feature space dimension d_{pre} of k.

(2) **Sums of kernels:** Consider the sum kernel $k = k_1 + k_2$ of kernels k_1, k_2 with finite-dimensional feature maps ϕ_1, ϕ_2. A feature map for k is given by $\phi(\boldsymbol{x}) := \begin{pmatrix} \phi_1(\boldsymbol{x}) \\ \phi_2(\boldsymbol{x}) \end{pmatrix}$. We can apply sketching as

$$\phi_{\to\text{sketch}(p)}(\boldsymbol{x}) := \phi_{1\to\text{sketch}(p)}(\boldsymbol{x}) + \phi_{2\to\text{sketch}(p)}(\boldsymbol{x}) \ . \tag{6.16}$$

This again yields an unbiased estimate of the kernel k. If $\phi_{1\to\text{sketch}(p)}$ and $\phi_{2\to\text{sketch}(p)}$ are sketched as in Eq. (6.12), then Eq. (6.16) is equivalent to sketching ϕ with Eq. (6.12) directly.

(3) **Products of kernels:** Consider the product kernel $k = k_1 \cdot k_2$ of kernels k_1, k_2 with finite-dimensional feature maps ϕ_1, ϕ_2. A feature map for k is given by $\phi(\boldsymbol{x}) := \phi_1(\boldsymbol{x}) \otimes \phi_2(\boldsymbol{x})$, where $\otimes$ is the tensor product. Hence, if the feature spaces of ϕ_1 and ϕ_2 have dimensions p_1 and p_2, respectively, the feature space of ϕ has dimension $p_1 p_2$. While this dimension is still finite, using Eq. (6.12) for sketching would potentially require a large amount of memory for storing $\boldsymbol{U}$ as well as a large runtime for the matrix-vector product. Therefore, we sketch product kernels more efficiently as

$$\phi_{\to\text{sketch}(p)}(\boldsymbol{x}) := \sqrt{p}\phi_{1\to\text{sketch}(p)}(\boldsymbol{x}) \odot \phi_{2\to\text{sketch}(p)}(\boldsymbol{x}) \ , \tag{6.17}$$

where $\odot$ denotes the element-wise product (or Hadamard product). This again yields an unbiased estimator of k without the need to perform computations in the $p_1 p_2$-dimensional feature space. While this simple tensor sketching method works sufficiently well for our purposes, its approximation properties are suboptimal and can be improved with a more complicated sketching method (Ahle et al., 2020).

In the kernel $k_{\text{grad}\to\text{sketch}(p)\to\text{post}(\mathcal{X}_{\text{train}},\sigma^2)}$, the inclusion of sketching can be considered a further approximation to the posterior predictive distribution for Bayesian NNs. In this context, a different sketching method has been proposed by Sharma et al. (2021). It is also possible to apply sketching to kernels with infinite-dimensional feature space (see e.g. Kar and Karnick, 2012; Zandieh et al., 2021; Han et al., 2021), but such kernels are less relevant in our case.

Ensembling

Ensembles of NNs have been demonstrated to yield good uncertainty estimates for DAL (Beluch et al., 2018) and can improve the uncertainty estimates of MC Dropout (Pop and Fulop, 2018). This motivates the study of ensembled kernels. When multiple NNs are trained on the same data and a kernel $k^{(i)}$ is computed for each model $i \in \{1, \dots, N_{\text{ens}}\}$ via a base kernel and a list of transformations, these kernels can be ensembled simply by adding them together:

$$k_{\to\text{ens}(N_{\text{ens}})} = k^{(1)} + \dots + k^{(N_{\text{ens}})} \ .$$

In the context of Bayesian NNs, ensembling of posterior kernels is related to a mixture of Laplace approximations (Eschenhagen et al., 2021), cf. Appendix 6.C.

ACS Random Features Transformation

In the following two paragraphs, we will briefly introduce multiple kernel transformations corresponding to various alternative ways of applying the ACS-FW method by Pinsler et al. (2019) to GP regression. For a more complete description, we refer to the original publication. ACS-FW seeks to approximate the expected complete data log posterior, $f_{\text{pool}}(\boldsymbol{\theta}) = \mathbb{E}_{\mathcal{Y}_{\text{pool}} \sim P(\mathcal{Y}_{\text{pool}} | \mathcal{X}_{\text{pool}}, \mathcal{D}_{\text{train}})} \log p(\boldsymbol{\theta} \mid \mathcal{D}_{\text{train}}, \mathcal{X}_{\text{pool}}, \mathcal{Y}_{\text{pool}})$, with the expected log posterior of the train data and the next batch, $f_{\text{batch}}(\boldsymbol{\theta}) = \mathbb{E}_{\mathcal{Y}_{\text{batch}} \sim P(\mathcal{Y}_{\text{batch}} | \mathcal{X}_{\text{batch}}, \mathcal{D}_{\text{train}})} \log p(\boldsymbol{\theta} \mid \mathcal{D}_{\text{train}}, \mathcal{X}_{\text{batch}}, \mathcal{Y}_{\text{batch}})$. Here, the labels $\mathcal{Y}_{\text{pool}}$ and $\mathcal{Y}_{\text{batch}}$ are drawn from the posterior distribution after observing $\mathcal{D}_{\text{train}}$. For a given Bayesian model, they then define different kernels resulting from this objective. As a Bayesian model, we use the same Gaussian process model as for the posterior transformation above, with kernel $k_{\to \text{scale}(\mathcal{X}_{\text{train}})}$. In this case, as shown in Section 6.C.3, we have $f_{\text{pool}}(\boldsymbol{\theta}) - f_{\text{batch}}(\boldsymbol{\theta}) = \sum_{\boldsymbol{x} \in \mathcal{X}_{\text{pool}} \setminus \mathcal{X}_{\text{batch}}} f_{\text{acs}}(\boldsymbol{x}, \boldsymbol{\theta})$ with

$$f_{\text{acs}}(\boldsymbol{x}, \boldsymbol{\theta}) \coloneqq \frac{1}{2} \log \left(1 + \frac{k_{\to \mathcal{X}_{\text{train}}}(\boldsymbol{x}, \boldsymbol{x})}{\sigma^2}\right) - \frac{(\boldsymbol{\theta}^\top \phi_{\to \text{scale}(\mathcal{X}_{\text{train}})}(\boldsymbol{x}))^2 + k_{\to \mathcal{X}_{\text{train}}}(\boldsymbol{x}, \boldsymbol{x})}{2\sigma^2} \tag{6.18}$$

The weighted inner product by (Pinsler et al., 2019) can then be written as

$$k_{\to \text{acs}}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \coloneqq \mathbb{E}_{\boldsymbol{\theta} \sim P(\boldsymbol{\theta} | \mathcal{D}_{\text{train}})} [f_{\text{acs}}(\boldsymbol{x}, \boldsymbol{\theta}) f_{\text{acs}}(\tilde{\boldsymbol{x}}, \boldsymbol{\theta})] \ . \tag{6.19}$$

The expectation in Eq. (6.19) can be approximated using Monte Carlo quadrature as

$$k_{\to \text{acs}}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \approx k_{\to \text{acs-rf}(d_{\text{post}})} \coloneqq \frac{1}{d_{\text{post}}} \sum_{i=1}^{d_{\text{post}}} [f_{\text{acs}}(\boldsymbol{x}, \boldsymbol{\theta}^{(i)}) f_{\text{acs}}(\tilde{\boldsymbol{x}}, \boldsymbol{\theta}^{(i)})] \ ,$$

where $\boldsymbol{\theta}^{(i)} \sim P(\boldsymbol{\theta} \mid \mathcal{D}_{\text{train}})$ are i.i.d. parameter samples from the posterior. This corresponds to the random features approximation proposed by Pinsler et al. (2019), given by

$$\phi_{\to \text{acs-rf}(d_{\text{post}})}(\boldsymbol{x}) \coloneqq d_{\text{post}}^{-1/2} (f_{\text{acs}}(\boldsymbol{x}, \boldsymbol{\theta}^{(1)}), \ldots, f_{\text{acs}}(\boldsymbol{x}, \boldsymbol{\theta}^{(d_{\text{post}})}))^\top \ .$$

In their regression experiments, Pinsler et al. (2019) use a slightly different feature map which we denote by $\phi_{\to \text{acs-rf-hyper}(d_{\text{post}})}$ in our experiments. They state that this has been derived from a GP model with a hyperprior on σ^2, although no hints on its derivation are provided in their paper, so we directly use their source code for our implementation.

ACS Gradient Transformation

As an alternative to the random features approximation in the previous paragraph, Pinsler et al. (2019) proposed the *weighted Fisher inner product* given by

$$k_{\to \text{acs-grad}}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \coloneqq \mathbb{E}_{\boldsymbol{\theta} \sim P(\boldsymbol{\theta} | \mathcal{D}_{\text{train}})} [\langle \nabla_{\boldsymbol{\theta}} f_{\text{acs}}(\boldsymbol{x}, \boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}} f_{\text{acs}}(\tilde{\boldsymbol{x}}, \boldsymbol{\theta}) \rangle] \ . \tag{6.20}$$

Under the Gaussian process model, they showed that an explicit formula for $k_{\to \text{acs-grad}}$ is given by

$$k_{\to \text{acs-grad}}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \frac{1}{\sigma^4} k_{\to \text{scale}(\mathcal{X}_{\text{train}})}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) k_{\to \mathcal{X}_{\text{train}}}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \ .$$

Since this product kernel can have a high-dimensional feature space, they used $k_{\to \text{acs-grad}}$ only on small data sets. In our experiments, we apply sketching to this kernel to scale it to large data sets. Again, they specified that they included a hyperprior on σ^2 in their experiments, but their corresponding implementation appears to be equivalent to $k_{\to \text{acs-grad}}$ for our purposes.

6.4.3 Discussion

Out of the base kernels and kernel transformations considered above, the training labels $\mathcal{Y}_{\text{train}}$ only influence the base kernels k_{ll} and k_{grad} through the trained parameters $\boldsymbol{\theta}_T$, and to some extent the transformation $k_{\to\text{acs-rf-hyper}(p)}$. Using k_{lin} and k_{nngp} with the selection methods below thus leads to passive learning or experimental design, where the entire set of training inputs $\mathcal{X}_{\text{train}}$ is selected before any of the labels $\mathcal{Y}_{\text{train}}$ are computed. This can be much cheaper because no NN retraining is required, but is also potentially less accurate.

Another consideration to be made when selecting a kernel is the feature space dimension d_{feat}. While a low d_{feat} is usually beneficial for runtime purposes, larger d_{feat} might allow for a more accurate representation of over-parameterized NNs. For k_{lin}, d_{feat} depends on the data set but is often rather small. For k_{ll}, d_{feat} can be reduced using sketching, but an effective increase of d_{feat} requires increasing the width of the last hidden layer, which might not always be desirable. For k_{grad}, d_{feat} is typically very large, and can be flexibly adjusted by using sketching. For k_{nngp}, we have $d_{\text{feat}} = \infty$, but sketching may also be applicable, see Han et al. (2021) for a similar application to infinite-width NTKs.

6.5 Selection Methods

In the following, we will discuss a variety of kernel-based selection methods. We first introduce the general iterative scheme that all evaluated methods (except Bait-FB) use, with its two variants called P (for pool) and TP (for train+pool). Subsequently, we explain specific selection methods.

6.5.1 Iterative Selection Methods

A natural approach towards selecting $\mathcal{X}_{\text{batch}}$ is to formulate an acquisition function a which scores an entire batch, such that $a(\mathcal{X}_{\text{batch}})$ should be maximized over all $\mathcal{X}_{\text{batch}} \subseteq \mathcal{X}_{\text{pool}}$ of size N_{batch}. However, the corresponding optimization problem is often intractable (Gonzalez, 1985; Civril and Magdon-Ismail, 2013). Many BMAL methods thus select points in a greedy/iterative fashion. To favor samples with high informativeness in an iterative selection scheme that tries to enforce diversity of the selected batch, two approaches can be used:

(P) Informativeness can be incorporated through the kernel. For example, $k_{\to\mathcal{X}_{\text{train}}}(\boldsymbol{x}, \boldsymbol{x})$ represents the posterior variance at $\boldsymbol{x}$ of a GP with scaled kernel $k_{\to\text{scale}(\mathcal{X}_{\text{train}})}$.

(TP) Informativeness can be incorporated implicitly by enforcing diversity of $\mathcal{X}_{\text{train}} \cup \mathcal{X}_{\text{batch}}$ instead of only enforcing diversity of $\mathcal{X}_{\text{batch}}$. In other words, a batch that is sufficiently different from the training set typically necessarily contains new information.

An iterative selection template with the two variants P and TP is shown in Algorithm 3, where different choices of NextBatch lead to different selection methods as discussed in Section 6.5.2. For simplicity of notation, Algorithm 3 does not reuse information in subsequent calls to NextBatch, which however is necessary to make the selection methods more efficient. We provide efficiency-focused pseudocode, which is also used for our implementation, and an analysis of runtime and memory complexities in Appendix 6.D.

Algorithm 3 Iterative selection algorithm template with customizable function NextSample, for which different options will be discussed in Section 6.5.2.

```
function Select($k$, $\mathcal{X}_{\text{train}}$, $\mathcal{X}_{\text{pool}}$, $N_{\text{batch}}$, mode $\in \{\text{P}, \text{TP}\}$)
    $\mathcal{X}_{\text{batch}} \leftarrow \emptyset$
    $\mathcal{X}_{\text{mode}} \leftarrow \mathcal{X}_{\text{train}}$ if mode = TP else $\emptyset$        ▷ Points considered as “selected”
    for $i$ from 1 to $N_{\text{batch}}$ do
        $\mathcal{X}_{\text{sel}} \leftarrow \mathcal{X}_{\text{mode}} \cup \mathcal{X}_{\text{batch}}$        ▷ Currently “selected” points
        $\mathcal{X}_{\text{rem}} \leftarrow \mathcal{X}_{\text{pool}} \setminus \mathcal{X}_{\text{batch}}$        ▷ Currently unselected points
        $\mathcal{X}_{\text{batch}} \leftarrow \mathcal{X}_{\text{batch}} \cup \{\text{NextSample}(k, \mathcal{X}_{\text{sel}}, \mathcal{X}_{\text{rem}})\}$
    end for
    return $\mathcal{X}_{\text{batch}}$
end function
```

Additionally, our implementation usually accelerates kernel computations through suitable precomputations, often by precomputing the features $\phi(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{X}_{\text{train}} \cup \mathcal{X}_{\text{pool}}$. In our notation, we treat $\mathcal{X}_{\text{train}}$ and $\mathcal{X}_{\text{pool}}$ as sets, assuming that all values are distinct. In practice, if multiple identical $\boldsymbol{x}_i$ are contained in $\mathcal{X}_{\text{train}}$ and/or $\mathcal{X}_{\text{pool}}$, they should still be treated as distinct.

6.5.2 Specific Methods

In the following, we will discuss a variety of choices for NextSample in Algorithm 3, leading to different selection methods. An overview of the resulting selection methods is given in Table 3. Table 4 shows how BMAL methods from the literature relate to the presented selection methods and kernels.

Random Selection

A simple baseline for comparison to other selection methods, denoted as Random, is to select $\mathcal{X}_{\text{batch}}$ randomly. We can formally express this as

$$\text{NextSample}(k, \mathcal{X}_{\text{sel}}, \mathcal{X}_{\text{rem}}) \sim \mathcal{U}(\mathcal{X}_{\text{rem}}) \ ,$$

where $\mathcal{U}(\mathcal{X}_{\text{rem}})$ is the uniform distribution over $\mathcal{X}_{\text{rem}}$. Since NextSample does not use $\mathcal{X}_{\text{sel}}$, the P and TP versions of Random are equivalent.

Naive Active Learning

If $k(\boldsymbol{x}, \boldsymbol{x})$ is interpreted as a measure for the uncertainty of the model at $\boldsymbol{x}$, naive active learning can simply be formalized as

$$\text{NextSample}(k, \mathcal{X}_{\text{sel}}, \mathcal{X}_{\text{rem}}) = \operatorname*{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\text{rem}}} k(\boldsymbol{x}, \boldsymbol{x}) \ . \tag{6.21}$$

Since NextSample only considers the diagonal of the kernel matrix $k(\mathcal{X}_{\text{rem}}, \mathcal{X}_{\text{rem}})$, we call the corresponding selection method MaxDiag. Similar to Random, the P and TP versions of MaxDiag are equivalent. If $k = \tilde{k}_{\to \mathcal{X}_{\text{train}}}$, $k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2$ represents the posterior predictive variance of a GP with kernel $\tilde{k}_{\to \text{scale}(\mathcal{X}_{\text{train}})}$ at $\boldsymbol{x}$. Unlike in the classification

Selection method	Description	Runtime complexity
Random	Random selection	$\mathcal{O}(N_{\text{pool}} \log N_{\text{pool}})$
MaxDiag	Naive active learning, picking largest diagonal entries	$\mathcal{O}(N_{\text{pool}}(T_k + \log N_{\text{pool}}))$
MaxDet	Greedy determinant maximization	$\mathcal{O}(N_{\text{cand}} N_{\text{sel}}(T_k + N_{\text{sel}}))$ or $\mathcal{O}(N_{\text{cand}} N_{\text{sel}} d_{\text{feat}})$
Bait-F	Forward-Greedy total uncertainty minimization	$\mathcal{O}(N_{\text{cand}} N_{\text{sel}} d_{\text{feat}} + (N_{\text{train}} + N_{\text{pool}}) d_{\text{feat}}^2)$
Bait-FB	Forward-Backward-Greedy total uncertainty minimization	$\mathcal{O}(N_{\text{cand}} N_{\text{sel}} d_{\text{feat}} + (N_{\text{train}} + N_{\text{pool}}) d_{\text{feat}}^2)$
FrankWolfe	Approximate kernel mean embedding using Frank-Wolfe	$\mathcal{O}((N_{\text{cand}} + N_{\text{pool}} N_{\text{batch}}) d_{\text{feat}})$ or $\mathcal{O}(N_{\text{cand}}^2 (T_k + 1))$
MaxDist	Greedy distance maximization	$\mathcal{O}(N_{\text{pool}} N_{\text{sel}}(T_k + 1))$
KMeansPP	Next point probability proportional to squared distance	$\mathcal{O}(N_{\text{pool}} N_{\text{sel}}(T_k + 1))$
LCMD (ours)	Greedy distance maximization in largest cluster	$\mathcal{O}(N_{\text{pool}} N_{\text{sel}}(T_k + 1))$

Table 3: Selection methods presented in this paper and their runtime complexities. For the runtime notation, we let T_k denote the runtime of a kernel evaluation and d_{feat} the dimensionality of its (pre-computed) features. Moreover, we write $N_{\text{cand}} := N_{\text{pool}} + |\mathcal{X}_{\text{mode}}|$ and $N_{\text{sel}} := N_{\text{batch}} + |\mathcal{X}_{\text{mode}}|$, with $\mathcal{X}_{\text{mode}}$ as in Algorithm 3. The runtime complexities are derived in Appendix 6.D. Further refinements of the runtime complexities for Random and MaxDiag are possible but not practically relevant to us, as these methods are already very efficient.

setting, the noise distribution $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ in the GP model is independent of $\boldsymbol{x}$, which renders different acquisition functions like maximum entropy (Shannon, 1948; MacKay, 1992b) or BALD (Houlsby et al., 2011) equivalent to Eq. (6.21). The active learning approach proposed by Zaverkin and Kästner (2021) corresponds to applying MaxDiag to $k_{\text{ll}\to\mathcal{X}_{\text{train}}}$ in the limit $\sigma^2 \to 0$. Out of the three objectives presented in Section 6.2.2, MaxDiag satisfies (INF), but not (DIV) and (REP). Indeed, if the pool set contains (almost-) duplicates, MaxDiag may select a batch consisting of (almost) identical inputs.

Greedy Determinant Maximization

To take account of the inputs $\mathcal{X}_{\text{sel}}$ that have already been selected, it is possible to also condition the GP on the selected values $\mathcal{X}_{\text{sel}}$, since the posterior variance of a GP does not depend on the unknown labels for $\mathcal{X}_{\text{sel}}$. Picking the input with maximal uncertainty after conditioning is equivalent to maximizing a determinant, as we show in Section 6.D.4:

$$\begin{aligned} \textsc{NextSample}(k, \mathcal{X}_{\text{sel}}, \mathcal{X}_{\text{rem}}) &= \operatorname*{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\text{rem}}} k_{\to \text{post}(\mathcal{X}_{\text{sel}}, \sigma^2)}(\boldsymbol{x}, \boldsymbol{x}) \\ &= \operatorname*{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\text{rem}}} \det(k(\mathcal{X}_{\text{sel}} \cup \{\boldsymbol{x}\}, \mathcal{X}_{\text{sel}} \cup \{\boldsymbol{x}\}) + \sigma^2 \boldsymbol{I}) , \quad (6.22) \end{aligned}$$

Known as	Selection method	Kernel	Remark
BALD (Houlsby et al., 2011)	MaxDiag	$k_{\to\mathrm{post}(\mathcal{X}_{\mathrm{train}},\sigma^2)}$	for GP with kernel k
BatchBALD (Kirsch et al., 2019)	MaxDet-P	$k_{\to\mathrm{post}(\mathcal{X}_{\mathrm{train}},\sigma^2)}$	for GP with kernel k, proposed for classification
Bait (Ash et al., 2021)	Bait-FB-P	$k_{\mathrm{ll}\to\mathrm{post}(\mathcal{X}_{\mathrm{train}},\sigma^2)}$	
ACS-FW (Pinsler et al., 2019)	FrankWolfe-P	$k_{\mathrm{ll}\to\mathrm{acs\text{-}rf}(p)}$ or $k_{\mathrm{ll}\to\mathrm{acs\text{-}grad}}$ or $k_{\mathrm{ll}\to\mathrm{acs\text{-}rf\text{-}hyper}(p)}$	
Core-Set (Sener and Savarese, 2018)	MaxDist-TP*	similar to k_{ll}	proposed for classification
FF-Active (Geifman and El-Yaniv, 2017)	MaxDist-TP	similar to k_{ll}	proposed for classification
BADGE (Ash et al., 2019)	KMeansPP-P	similar to k_{ll}	proposed for classification

* This refers to their simpler k-center-greedy selection method.

Table 4: Some (regression adaptations of) BM(D)AL methods from the literature and their corresponding selection methods and kernels.

We call the corresponding selection method MaxDet. It is equivalent to performing non-batch mode active learning on the GP with kernel k, and has been applied to GPs by Seo et al. (2000). Moreover, as we show in Section 6.D.4, it is also equivalent to applying BatchBALD (Kirsch et al., 2019) to the GP with kernel k. If $\sigma^2 = 0$, it is equivalent to the P-greedy method for kernel interpolation (De Marchi et al., 2005), and it is also related to the greedy algorithm for D-optimal design (Wynn, 1970). In comparison to a naive implementation that computes each determinant separately, the runtime complexity of the determinant computation in MaxDet can be reduced by a factor of $\mathcal{O}(N_{\mathrm{sel}}^2)$ to $\mathcal{O}(N_{\mathrm{cand}}N_{\mathrm{sel}}(T_k + N_{\mathrm{sel}}))$ when implementing MaxDet via a partial pivoted matrix-free Cholesky decomposition as suggested in Pazouki and Schaback (2011). For the case $N_{\mathrm{sel}} \gg d_{\mathrm{feat}}$, we show in Section 6.D.4 how the runtime complexity can be reduced further. For given $\sigma^2 > 0$ in Eq. (6.22), it follows from Eq. (6.11) that applying MaxDet-TP to a kernel k is equivalent to applying MaxDet-P to $k_{\to\mathrm{post}(\mathcal{X}_{\mathrm{train}},\sigma^2)}$ (with the same σ^2).

Greedy total uncertainty minimization

While MaxDet satisfies (INF) and (DIV), it does not satisfy (REP) since it does not incorporate the pool set distribution. To fix this, Ash et al. (2021) propose Bait. The regression version of Bait tries to minimize the sum of the GP posterior variances on the training and pool set.[5] In other words, Bait aims to minimize the acquisition function

$$a(\mathcal{X}_{\mathrm{sel}}) := \sum_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\mathrm{train}} \cup \mathcal{X}_{\mathrm{pool}}} k_{\to\mathrm{post}(\mathcal{X}_{\mathrm{sel}},\sigma^2)}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}) \ . \tag{6.23}$$

[5]There is an "unregularized" version of Bait that extends to classification, but we use the "regularized" version with $\sigma^2 > 0$ here since it is more natural for GPs and avoids numerical issues.

In Section 6.D.5, we show that Eq. (6.23) is equivalent to the original Bait formulation. This is also known as (Bayesian) V-optimal design (Montgomery, 2017) and a similar method has been studied for NNs by Cohn (1996). Ash et al. (2021) propose two alternative methods for efficient approximate optimization of this acquisition function. The first one, which we call Bait-F, is to simply use greedy selection as for the other methods in this section. The second alternative, which we call Bait-FB, greedily selects $2N_{\mathrm{batch}}$ points (forward step) and then greedily removes N_{batch} points (backward step). In our framework, we can define Bait-F by

$$\textsc{NextSample}(k, \mathcal{X}_{\mathrm{sel}}, \mathcal{X}_{\mathrm{rem}}) := \operatorname*{arg\,min}_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{rem}}} \sum_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\mathrm{train}} \cup \mathcal{X}_{\mathrm{pool}}} k_{\to \mathrm{post}(\mathcal{X}_{\mathrm{sel}} \cup \{\boldsymbol{x}\}, \sigma^2)}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}) ~.$$

Details on Bait-F and Bait-FB are given in Section 6.D.5. Like for MaxDet, applying Bait-F-TP or Bait-FB-TP to a kernel k is equivalent to applying Bait-F-P or Bait-FB-P to $k_{\to \mathrm{post}(\mathcal{X}_{\mathrm{train}}, \sigma^2)}$ (with the same σ^2).

Frank-Wolfe optimization

In order to make $\mathcal{X}_{\mathrm{batch}}$ representative of the pool set, Pinsler et al. (2019) suggest to choose $\mathcal{X}_{\mathrm{batch}}$ such that $\sum_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{pool}}} \phi(\boldsymbol{x})$ is well-approximated by $\sum_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{batch}}} w_{\boldsymbol{x}} \phi(\boldsymbol{x})$, where $w_{\boldsymbol{x}}$ are non-negative weights. Specifically, they propose to apply the Frank-Wolfe optimization algorithm to a corresponding optimization problem, which automatically selects elements of $\mathcal{X}_{\mathrm{batch}}$ iteratively. This can be seen as an attempt to represent the distribution of $\mathcal{X}_{\mathrm{pool}}$ with $\mathcal{X}_{\mathrm{batch}}$ by approximating the empirical kernel mean embedding $N_{\mathrm{pool}}^{-1} \sum_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{pool}}} k(\boldsymbol{x}, \cdot)$ using $\mathcal{X}_{\mathrm{batch}}$. The corresponding selection method can be implemented in kernel space or feature space. Since the kernel space version scales quadratically with N_{pool}, Pinsler et al. (2019) use the feature space version for large pool sets. We also use the feature space version for our experiments and show the pseudocode in Section 6.D.6. While the version by Pinsler et al. (2019) allows to select the same $\boldsymbol{x} \in \mathcal{X}_{\mathrm{pool}}$ multiple times, we prohibit this as it would allow to select smaller batches and thus prevent a fair comparison to other methods.

Greedy distance maximization

A simple strategy to enforce the diversity of a set of points is to greedily select points with maximum distance to all previously selected points. The resulting algorithm has been frequently proposed in the literature under different names, see Section 6.D.7. In our case, the kernel k gives rise to a distance measure $d_k(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ as in Eq. (6.13). With this distance measure, the MaxDist selection method is specified by

$$\textsc{NextSample}(k, \mathcal{X}_{\mathrm{sel}}, \mathcal{X}_{\mathrm{rem}}) = \operatorname*{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{rem}}} \min_{\boldsymbol{x}' \in \mathcal{X}_{\mathrm{sel}}} d_k(\boldsymbol{x}, \boldsymbol{x}') ~.$$

If the argmax is not unique, an arbitrary maximizer is chosen. If $\mathcal{X}_{\mathrm{sel}}$ is empty, we choose $\operatorname{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{rem}}} k(\boldsymbol{x}, \boldsymbol{x})$.

The use of MaxDist-TP with k_{lin} for BMAL has been suggested by Yu and Kim (2010), and with a kernel similar to k_{ll} for BMDAL by Sener and Savarese (2018) and Geifman and El-Yaniv (2017). Sener and Savarese (2018) also alternatively propose a more involved discrete optimization algorithm. In their experiments, MaxDist yielded only slightly worse results than the more involved optimization algorithm while being significantly faster and easier to implement. They note that the batch selected by MaxDist is suboptimal with respect to a covering objective by at most a factor of two. In Section 6.D.7, we show that a similar guarantee can be given when applying MaxDist to a sketched approximation of the desired kernel. The use of dimensionality reduction for MaxDist has also been analyzed by Eppstein et al. (2020). Inspired by the reasoning of Sener and Savarese (2018), we interpret the distances as uncertainty estimates: If both the optimal regression function f_* and the learned regression function $f_{\boldsymbol{\theta}_T}$ are L-Lipschitz with respect to d_k, and we have $y_i = f_*(\boldsymbol{x}_i) = f_{\boldsymbol{\theta}_T}(\boldsymbol{x}_i)$ on the training set, then we have the worst-case bound

$$|f_{\boldsymbol{\theta}_T}(\boldsymbol{x}) - f_*(\boldsymbol{x})| \leq 2L \min_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\mathrm{train}}} d_k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \ . \tag{6.24}$$

Of course, the Lipschitz constant L might itself depend on $\mathcal{X}_{\mathrm{train}}$, so this should only be interpreted as a crude heuristic. Wenzel et al. (2021) show that for kernel interpolation with Sobolev kernels, MaxDist and MaxDet with $\sigma^2 = 0$ yield asymptotically equivalent convergence rates.

k-means++ seeding

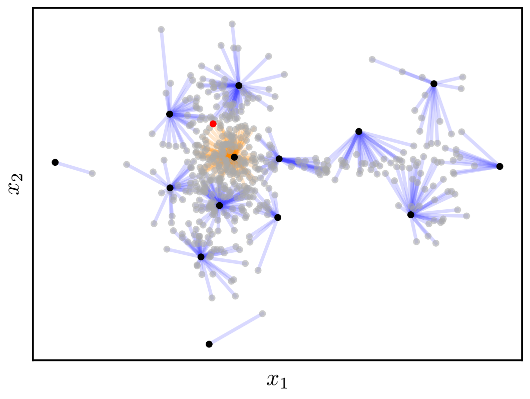
Similar to MaxDet, MaxDist enforces (INF) and (DIV) but not (REP). To incorporate (REP), i.e., sample more points from regions with higher pool set density, we can view batch selection as a clustering problem: For example, if the distance-based uncertainty estimate in Eq. (6.24) holds, we could try to minimize the corresponding upper bound on the pool set MSE $\frac{1}{N_{\mathrm{pool}}} \sum_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{pool}}} |f_{\boldsymbol{\theta}_T}(\boldsymbol{x}) - f_*(\boldsymbol{x})|^2$ after adding $\mathcal{X}_{\mathrm{batch}}$:

$$\mathcal{X}_{\mathrm{batch}} = \operatorname*{arg\,min}_{\mathcal{X}_{\mathrm{batch}} \subseteq \mathcal{X}_{\mathrm{pool}}, |\mathcal{X}_{\mathrm{batch}}| = N_{\mathrm{batch}}} \frac{1}{N_{\mathrm{pool}}} \sum_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{pool}}} \min_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\mathrm{mode}} \cup \mathcal{X}_{\mathrm{batch}}} d_k(\boldsymbol{x}, \tilde{\boldsymbol{x}})^2 \ . \tag{6.25}$$

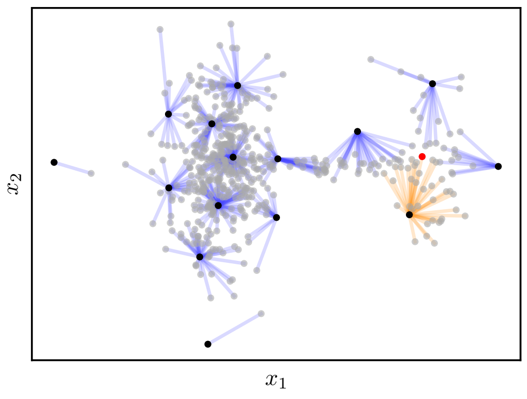
This optimization problem is essentially the k-medoids problem (Kaufman and Rousseeuw, 1990), which combines the objective for the k-means clustering algorithm (Lloyd, 1982) with the constraint that the cluster centers must be chosen from within the data to be clustered. For large pool sets, common k-medoids algorithms can be computationally infeasible. An efficient approximate k-medoids solution can be computed using the seeding method of the k-means++ algorithm (Arthur and Vassilvitskii, 2007; Ostrovsky et al., 2006), which simply chooses the next batch element randomly via the distribution

$$\forall \boldsymbol{x} \in \mathcal{X}_{\mathrm{rem}} : P(\textsc{NextSample}(k, \mathcal{X}_{\mathrm{sel}}, \mathcal{X}_{\mathrm{rem}}) = \boldsymbol{x}) = \frac{\min_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\mathrm{sel}}} d_k(\boldsymbol{x}, \tilde{\boldsymbol{x}})^2}{\sum_{\boldsymbol{x}' \in \mathcal{X}_{\mathrm{rem}}} \min_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\mathrm{sel}}} d_k(\boldsymbol{x}', \tilde{\boldsymbol{x}})^2} \ ,$$

and if $\mathcal{X}_{\mathrm{sel}}$ is empty, it selects $\textsc{NextSample}(k, \mathcal{X}_{\mathrm{sel}}, \mathcal{X}_{\mathrm{rem}})$ uniformly at random from $\mathcal{X}_{\mathrm{rem}}$. We refer to the corresponding selection method as KMeansPP. For the case $\mathcal{X}_{\mathrm{mode}} = \emptyset$, Arthur and Vassilvitskii (2007) showed that with respect to the objective in Eq. (6.25), the batch selected by KMeansPP is suboptimal by a factor of at most $16 + 8 \log(N_{\mathrm{batch}})$ in expectation. The use of KMeansPP-P for BMDAL has been proposed in the so-called BADGE method by Ash et al. (2019). In contrast to our setting, BADGE is designed for classification and introduces an uncertainty estimate into k_{ll} not through a posterior transformation but through the influence of the softmax output layer on the magnitude of the gradients.



(a) One LCMD step.



(b) Subsequent LCMD step.

Figure 2: Two steps of LCMD selection for points $\boldsymbol{x} \in \mathbb{R}^2$ with linear feature map $\phi(\boldsymbol{x}) = \boldsymbol{x}$. The black points represent the already selected points $\mathcal{X}_{\text{sel}}$ and the gray points represent the remaining points $\mathcal{X}_{\text{rem}}$. The lines associate each remaining point to the closest selected point, forming clusters. The orange lines represent the cluster with the largest sum of squared distances. The red point, which is the remaining point with the largest distance to the cluster center within the orange cluster, is selected next. In the left plot, it can be seen that the smaller-radius cluster on the left is preferred over the larger-radius clusters on the right due to the higher point density on the left. After adding a point from the cluster, its size shrinks and a cluster on the right becomes dominant, which is shown in the right plot.

Largest cluster maximum distance

As an alternative to the randomized KMeansPP method, we propose a novel deterministic method that is inspired by the same objective (Eq. (6.25)). This method, which we call LCMD (largest cluster maximum distance), is visualized in Figure 2. Intuitively, we enforce (REP) by limiting the selection to the largest cluster, while we also enforce (DIV) by picking the maximum distance point within this cluster. LCMD can be formally defined as follows: We interpret points $\tilde{\boldsymbol{x}} \in \mathcal{X}_{\text{sel}}$ as cluster centers. For each point $\boldsymbol{x} \in \mathcal{X}_{\text{rem}}$, we define its associated center $c(\boldsymbol{x}) \in \mathcal{X}_{\text{sel}}$ as

$$c(\boldsymbol{x}) := \operatorname*{arg\,min}_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\text{sel}}} d_k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \ .$$

If there are multiple minimizers, we pick an arbitrary one of them. Then, for each center $\tilde{\boldsymbol{x}} \in \mathcal{X}_{\text{sel}}$, we define the cluster size

$$s(\tilde{\boldsymbol{x}}) := \sum_{\boldsymbol{x} \in \mathcal{X}_{\text{rem}} : c(\boldsymbol{x}) = \tilde{\boldsymbol{x}}} d_k(\boldsymbol{x}, \tilde{\boldsymbol{x}})^2 \ .$$

We then pick the maximum-distance point from the cluster with maximum size:

$$\textsc{NextSample}(k, \mathcal{X}_{\text{sel}}, \mathcal{X}_{\text{rem}}) = \operatorname*{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\text{rem}} : s(c(\boldsymbol{x})) = \max_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\text{sel}}} s(\tilde{\boldsymbol{x}})} d_k(\boldsymbol{x}, c(\boldsymbol{x})) \ .$$

As for MaxDist, if $\mathcal{X}_{\text{sel}}$ is empty, we choose $\operatorname{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\text{rem}}} k(\boldsymbol{x}, \boldsymbol{x})$ instead. If the selection of pool points should be adapted to the distribution of another set $\mathcal{X}$ instead of $\mathcal{X}_{\text{pool}}$, one may simply compute the cluster sizes based on $\mathcal{X}$ instead. Importantly, like KMeansPP but unlike some other k-medoids methods, LCMD can be implemented with a runtime complexity that is linear in N_{pool}, as discussed in Section 6.D.9.

Other options

As mentioned above, MaxDist can be interpreted as a greedy optimization algorithm for a covering objective that is NP-hard to (approximately) optimize (Gonzalez, 1985; Feder and Greene, 1988), but for which other approximate optimization algorithms have been proposed (Sener and Savarese, 2018). Similarly, MaxDet-P attempts to greedily maximize $\det(k(\mathcal{X}_{\text{batch}}, \mathcal{X}_{\text{batch}}) + \sigma^2 \boldsymbol{I})$, which is NP-hard to (approximately) optimize (Civril and Magdon-Ismail, 2013), but for which other approximate optimization algorithms have been proposed (Bıyık et al., 2019). We do not investigate these advanced optimization algorithms here as they come with greatly increased runtime cost, and MaxDet and MaxDist already enjoy some approximation guarantees, as discussed in Appendix 6.D.

Yu and Kim (2010), Wu (2018), and Zhdanov (2019) suggest less scalable clustering-based approaches for BM(D)AL. Alternatively, it might be interesting to try the greedy k-means++ algorithm (Celebi et al., 2013), which provides a slightly less efficient alternative to the k-means++ algorithm.

Caselton and Zidek (1984) propose to optimize mutual information between the batch samples and the remaining pool samples, which is analyzed for GPs by Krause et al. (2008), but does not scale well with the pool set size for GPs, at least in a general kernel-space formulation. Another option is to remove the non-negativity constraint on the weights $w_{\boldsymbol{x}}$ used in FrankWolfe. This setting is also treated in a generalized fashion in Santin et al. (2021). An investigation of this method is left to future work.

6.5.3 Discussion

When considering the design criteria from Section 6.2.2, we can say that MaxDiag only satisfies (INF), while MaxDet and MaxDist also satisfy (DIV). Arguably, Bait, KMeansPP, and LCMD satisfy all three properties (INF), (DIV) and (REP). The FrankWolfe method is only designed to satisfy (REP), based on which one could argue that (INF) and (DIV) are also satisfied to some extent.

In terms of runtime complexity, as can be seen in Table 3, all considered selection methods are well-behaved for moderate feature space dimension d_{feat}. When considering kernels such as k_{grad}, whose evaluation is tractable despite having very high feature space dimension, distance-based selection methods are still efficient while Bait and FrankWolfe can become intractable for large pool set sizes, and MaxDet exhibits worse scaling with respect to N_{batch}. Moreover, if $\phi(\mathcal{X}_{\text{train}})$ has full rank and $d_{\text{feat}} \leq N_{\text{train}}$, it follows from Eq. (6.10) that in the limit $\sigma^2 \to 0$, the GP posterior uncertainty becomes zero everywhere. Hence, Bayesian posterior-based methods like MaxDet, Bait and FrankWolfe might require $d_{\text{feat}} \gtrsim N_{\text{train}}$ for good performance, which in turn deteriorates their runtime.

In the non-batch active learning setting, that is, for $N_{\text{batch}} = 1$, some selection methods become equivalent: LCMD-P, MaxDist-P, and MaxDet-P become equivalent to MaxDiag-P; moreover, KMeansPP-P becomes equivalent to Random. This suggests that for $N_{\text{batch}} = 1$, TP-mode is necessary for KMeansPP and LCMD to ensure (REP).

6.6 Experiments

To evaluate a variety of combinations of kernels, kernel transformations, and selection methods, we introduce a new open-source benchmark for BMDAL for regression. Our benchmark uses 15 large tabular regression data sets, with input dimensions ranging between two and 379, that are selected mostly from the UCI and OpenML repositories, cf. the detailed description in Section 6.E.1. The initial pool set size N_{pool} for these data sets lies between 31335 and 198720. As an NN model, we use a three-layer fully connected NN with 512 neurons in both hidden layers, parameterized as in Section 6.2.1. We train the NN using Adam (Kingma and Ba, 2015) for 256 epochs with batch size 256, using early stopping based on a validation set. Results shown here are for the ReLU activation function, but we also re-ran most of our experiments for the SiLU (a.k.a. swish) activation (Elfwing et al., 2018), and unless indicated otherwise, our insights discussed below apply to results for both activation functions. We manually optimized the parameters σ_w, σ_b and the learning rate separately for both activation functions to optimize the average logarithmic RMSE of Random selection. Details on the NN architecture and training are described in Section 6.E.3. For the hyperparameter σ^2, which occurs in MaxDet, Bait, and various posterior-based transformations, we found that smaller values typically yield better average results but may cause numerical instabilities. As a compromise, we chose $\sigma^2 = 10^{-6}$ in our experiments and use 64-bit floats for computations involving σ^2.

In our evaluation, we start with $N_{\text{train}} = 256$ and then acquire 16 batches with $N_{\text{batch}} = 256$ samples each using the respective BMAL method. We repeat this 20 times with different seeds for NN initialization and different splits of the data into training, validation, pool, and test sets. We measure the mean absolute error (MAE), root mean squared error (RMSE), 95% and 99% quantiles, and the maximum error (MAXE) on the test set after each BMAL step. For each of those five error metrics, we average the logarithms of the metric over the 20 repetitions, and, depending on the experiment, over the 16 steps and/or the 15 data sets. Note that a difference of δ between two logarithmic values corresponds to a ratio $e^\delta \approx 1 + \delta$ between the values; for example, a reduction by $\delta = 0.1$ corresponds to a reduction of the geometric mean error by about 10%. Our most important metric is the RMSE, but we will also put some focus on MAXE since it can be interpreted as a measure of robustness to distribution shifts. Generally, RMSE is more affected by rare but large errors than MAE, while the quantiles and MAXE exclusively focus on rare but large errors.

In the following, we will discuss some of the benchmark results. More detailed results can be found in Section 6.E.4.

6.6.1 Comparison to Existing Methods

Based on our detailed evaluation in Table 6.E.5 and Table 6.E.6, we propose a new BMDAL algorithm as the combination of the LCMD-TP selection method and the kernel $k_{\text{grad}\to\text{sketch}(512)}$. Figure 1 and Table 5 show that our proposed combination clearly outperforms other methods from the literature in terms of averaged logarithmic RMSE over our benchmark data sets and random splits. We incorporate the methods from the literature into our framework as shown in Table 5, which involves the following modifications:

- The BALD (Houlsby et al., 2011) and BatchBALD (Kirsch et al., 2019) acquisition functions are applied to a last-layer Gaussian Process model.
- For BAIT, we rescale k_{ll} based on the training set before applying the posterior transformation, see Section 6.4.2, and we apply regularization by using a small $\sigma^2 > 0$.
- For ACS-FW (Pinsler et al., 2019), we use the FrankWolfe selection method with $k_{\text{ll}\to\text{acs-rf-hyper}(512)}$. Compared to the experiments by Pinsler et al. (2019), there are several differences: First, we do not permit FrankWolfe to select smaller batches by selecting the same point multiple times. Second, we use 512 random features instead of 10. Third, our acs-rf-hyper transformation first rescales k_{ll} based on the training set, which we found to improve performance. Fourth, our k_{ll} kernel incorporates the last-layer bias and not only the weights.
- By Core-Set, we refer to the k-center-greedy method of Sener and Savarese (2018) applied to k_{ll}, which is also equivalent to FF-Active (Geifman and El-Yaniv, 2017).
- For BADGE (Ash et al., 2019), which originally incorporates uncertainties into ϕ_{ll} through softmax gradients, we use $\phi_{\text{ll}\to\mathcal{X}_{\text{train}}}$ instead of ϕ_{ll}.

As argued in Section 6.2.2, we do not compare to methods that require training with ensembles (Krogh and Vedelsby, 1994), since ensembles are more expensive to train and these methods are typically designed for non-batch active learning. Moreover, we do not compare to methods that require training with Dropout (Tsymbalov et al., 2018) or custom loss functions (Ranganathan et al., 2020), since these methods can change the error for the underlying NN, which makes them difficult to compare fairly and more inconvenient to use.

6.6.2 Evaluated Combinations

Our framework allows us to obtain a vast number of BMDAL algorithms via combinations of base kernels, kernel transformations, and the P and TP modes of different selection methods. Table 6.E.5 and Table 6.E.6 in Section 6.E.4 show a large number of such combinations for ReLU and SiLU activations, respectively. These combinations have been selected according to the following principles:

- Kernels for P-mode selection use posterior-based transformations, while kernels for TP-mode selection do not (see Section 6.5.1).
- Sketching and random features always use 512 target features. Similar to the hidden layer size of 512, this number has been selected to be a bit larger than the usually employed $N_{\text{batch}} = 256$. Note that due to the bias in the last layer, k_{ll} has a 513-dimensional feature space.
- FrankWolfe is only run in P mode (as proposed in Pinsler et al. (2019)), since in TP mode, kernel mean embeddings in 512-dimensional feature space would be approximated using more than 512 samples. Note that Pinsler et al. (2019) only use 10 instead of 512 random features in their experiments, leading to a worse approximation.

BMDAL method	Selection method	Kernel	mean log RMSE (↓)	
			ReLU	SiLU
Supervised learning	RANDOM	—	-1.401	-1.406
BALD (Houlsby et al., 2011) with last-layer GP	MAXDIAG	$k_{\mathrm{ll}\to\mathcal{X}_{\mathrm{train}}}$	-1.285	-1.300
BatchBALD (Kirsch et al., 2019) with last-layer GP	MAXDET-P	$k_{\mathrm{ll}\to\mathcal{X}_{\mathrm{train}}}$	-1.463	-1.467
BAIT (Ash et al., 2021)	BAIT-FB-P	$k_{\mathrm{ll}\to\mathcal{X}_{\mathrm{train}}}$	-1.541	-1.522
ACS-FW (Pinsler et al., 2019)	FRANKWOLFE-P	$k_{\mathrm{ll}\to\mathrm{acs\text{-}rf\text{-}hyper}(512)}$	-1.439	-1.437
Core-Set* (Sener and Savarese, 2018), FF-Active (Geifman and El-Yaniv, 2017)	MAXDIST-TP	k_{ll}	-1.491	-1.515
BADGE (Ash et al., 2019) with last-layer GP uncertainty	KMEANSPP-P	$k_{\mathrm{ll}\to\mathcal{X}_{\mathrm{train}}}$	-1.530	-1.484
Ours	LCMD-TP	$k_{\mathrm{grad}\to\mathrm{sketch}(512)}$	**-1.590**	**-1.597**

* This refers to their simpler k-center-greedy selection method.

Table 5: Comparison of our BMDAL method against other methods from the literature (cf. Table 4). The mean log RMSE is averaged over all data sets, repetitions, and BMAL steps for the respective experiments with ReLU or SiLU activation function. We make small adjustments to the literature methods as described in Section 6.6.

- Due to the equivalence between P mode with posterior transformation and TP mode without posterior transformation mentioned in Section 6.5.2, BAIT is always run in P mode, and MAXDET is mostly run in P mode except for k_{grad} and k_{nngp} due to their high-dimensional feature space.

In general, we observe the following trends in our results across selection methods:

- The network-dependent base kernels k_{ll} and k_{grad} clearly outperform the network-independent base kernels k_{lin} and k_{nngp} across different selection methods, modes and kernel transformations.
- Out of the network-dependent base kernels, k_{grad} typically outperforms k_{ll}, at least for NN hyperparameters optimized for RANDOM (cf. Section 6.E.3). It should be noted that in our ReLU experiments, these optimized hyperparameters typically lead to many dead neurons in the last hidden layer, which may affect k_{ll} by reducing the effective feature space dimension.[6] We define the effective feature space dimension of the pool set for a kernel k as

$$d_{\mathrm{eff}} := \frac{\mathrm{tr}(k(\mathcal{X}_{\mathrm{pool}}, \mathcal{X}_{\mathrm{pool}}))}{\|k(\mathcal{X}_{\mathrm{pool}}, \mathcal{X}_{\mathrm{pool}})\|_2} = \frac{\lambda_1 + \ldots + \lambda_{d_{\mathrm{feat}}}}{\lambda_1} ,$$

[6]In the extreme case where all neurons in the last hidden layer are dead, the network-dependent base kernels become degenerate, which can cause numerical problems in selection methods. Once a selection method suggests an invalid (e.g. already selected) sample for the batch, we fill up the rest of the batch with random samples. Out of the 597900 BMDAL steps in our ReLU experiments, such invalid samples were suggested in just 4 steps in total.

where $\lambda_1 \geq \ldots \geq \lambda_{d_{\text{feat}}}$ are the eigenvalues of the feature covariance matrix

$$\phi(\mathcal{X}_{\text{pool}})^\top \phi(\mathcal{X}_{\text{pool}}) \in \mathbb{R}^{d_{\text{feat}} \times d_{\text{feat}}} .$$

With this definition, d_{eff} is indeed typically much larger for $k_{\text{grad}\to\text{sketch}(512)}$ than for k_{ll} in our experiments.[7] Another difference is that, for the ReLU activation function, k_{grad} is discontinuous while k_{ll} is not.

- For k_{grad}, applying sketching does not strongly affect the resulting accuracy while leading to considerably faster runtimes.
- When evaluating the use of ensembled NN kernels, we want to differentiate between the effect of ensembling on the accuracy of supervised learning and the effect of ensembling on the quality of the selected batches $\mathcal{X}_{\text{batch}}$. To eliminate the former effect, we only consider the averaged errors of the individual ensemble members and not the error of their averaged predictions. With this method of evaluation, we find that ensembling of network-dependent kernels only leads to small improvements in the error, at least for the ensembling configurations we tested. This is in contrast to other papers where the uncertainty of the ensemble predictions turned out to be more beneficial (Beluch et al., 2018; Pop and Fulop, 2018). Perhaps ensembling is less useful in our case because our non-ensembled kernels already provide good uncertainty measures.
- Out of the acs-grad, acs-rf and acs-rf-hyper transformations, acs-rf often performs best, except for FrankWolfe-P with base kernel k_{grad}, where acs-rf-hyper performs best.
- In contrast to Ash et al. (2021), we find that Bait-FB does not perform better than Bait-F.
- The relative gains for BMDAL methods compared to Random selection are typically largest on metrics such as MAXE or 99% quantile, and worst on MAE.
- All investigated BMDAL methods only take a few seconds to select a batch in our experiments on our NVIDIA RTX 3090 GPUs (cf. Appendix 6.E), which is typically faster than the time for training the corresponding NN. Hence, we expect all investigated BMDAL methods to be much faster than the time for labeling in most scenarios where BMDAL is desirable. Note that the runtime of TP-mode selection methods is comparable to those of P-mode selection methods only because we typically run P-mode selection with 64-bit floats to avoid numerical issues for posteriors. TP-mode selection methods need to consider $N_{\text{train}} + N_{\text{batch}}$ instead of N_{batch} selected points, which can be significantly slower than P-mode if $N_{\text{train}} \gg N_{\text{batch}}$. Especially for TP-mode selection methods, it may therefore be desirable to let N_{batch} grow proportionally to N_{train}.

[7]Specifically, averaged over all corresponding ReLU experiments with $N_{\text{batch}} = 256$ and over all BMAL steps, $k_{\text{grad}\to\text{sketch}(512)}$ leads to an average d_{eff} of about 5.5, while k_{ll} leads to an average d_{eff} of about 1.7. On corresponding BMAL steps, the effective dimension is larger for $k_{\text{grad}\to\text{sketch}(512)}$ about 95% of the time. For SiLU, the results are slightly less extreme, with effective dimensions of 4 and 2.3, and the effective dimension of $k_{\text{grad}\to\text{sketch}(512)}$ being larger about 90% of the time.

Selection method	Sel. mode	Selected kernel	mean log RMSE	Avg. time [s]
RANDOM	—	—	-1.401	0.001
MAXDIAG	—	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf}(512)}$	-1.370	0.650
MAXDET	P	$k_{\text{grad}\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-1.512	0.770
BAIT	F-P	$k_{\text{grad}\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-1.585	1.508
FRANKWOLFE	P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf-hyper}(512)}$	-1.542	0.823
MAXDIST	P	$k_{\text{grad}\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-1.514	0.713
KMEANSPP	P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf}(512)}$	-1.569	0.836
LCMD	TP	$k_{\text{grad}\to\text{sketch}(512)}$	**-1.590**	0.981

Table 6: Selected kernels and modes per selection method that are shown in our plots. The mean log RMSE is averaged over all data sets, repetitions, and BMAL steps. The average time for the batch selection is averaged over all data sets and BMAL steps, measured at one repetition with only one process running on each NVIDIA RTX 3090 GPU. An overview of all results can be found in Table 6.E.5.

6.6.3 Best Kernels and Modes for each Selection Method

Our base kernels, kernel transformations, and selection modes yield numerous ways to apply each selection method. To compare selection methods, we choose for each selection method the best-performing combination according to the averaged logarithmic RMSE, excluding kernels with ensembling and k_{grad} without sketching since they considerably increase computational cost while providing comparable accuracy to their more efficient counterparts. The selected combinations are shown in Table 6. Compared to the literature methods in Table 5, we see that optimizing the kernel and mode can yield a considerable difference in performance. Note that the relative performance between the configurations for SiLU is slightly different, as can be seen in Table 6.E.6. If we selected combinations according to the SiLU results, the combinations for MAXDIST and KMEANSPP in Table 6 would use TP-mode and $k_{\text{grad}\to\text{sketch}(512)}$ instead. When considering only kernels based on k_{ll}, the results from Table 6.E.5 and Table 6.E.6 show that a comparison of selection methods would look qualitatively similar.

6.6.4 Comparison of Selection Methods

We compare the selected configurations from Table 6 in several aspects. Figure 3 shows the evolution of the mean logarithmic MAE, RMSE, MAXE, 95% quantile, and 99% quantile over the BMAL steps, for a batch size of $N_{\text{batch}} = 256$. This demonstrates that the best considered BMDAL methods can match the average performance of RANDOM selection with about half of the samples for RMSE, and even fewer samples for MAXE. On individual data sets, this may differ, as shown in Figure 4. From this figure, it is apparent that when considering RMSE, LCMD-TP outperforms other methods not only in terms of average performance but across the majority of the data sets. Specifically, Table 6.E.3 shows that LCMD-TP matches or exceeds the performance of the other selected BMDAL methods on 8 out of the 15 data sets in terms of RMSE. Figure 4 also shows that the selected MAXDET-P and MAXDIST-P configurations yield very similar performance on all data sets.

Figure 5 shows the influence of the chosen batch size N_{batch} on the final performance at $N_{\text{train}} = 4352$ training samples. As expected, the naive active learning scheme MAXDIAG

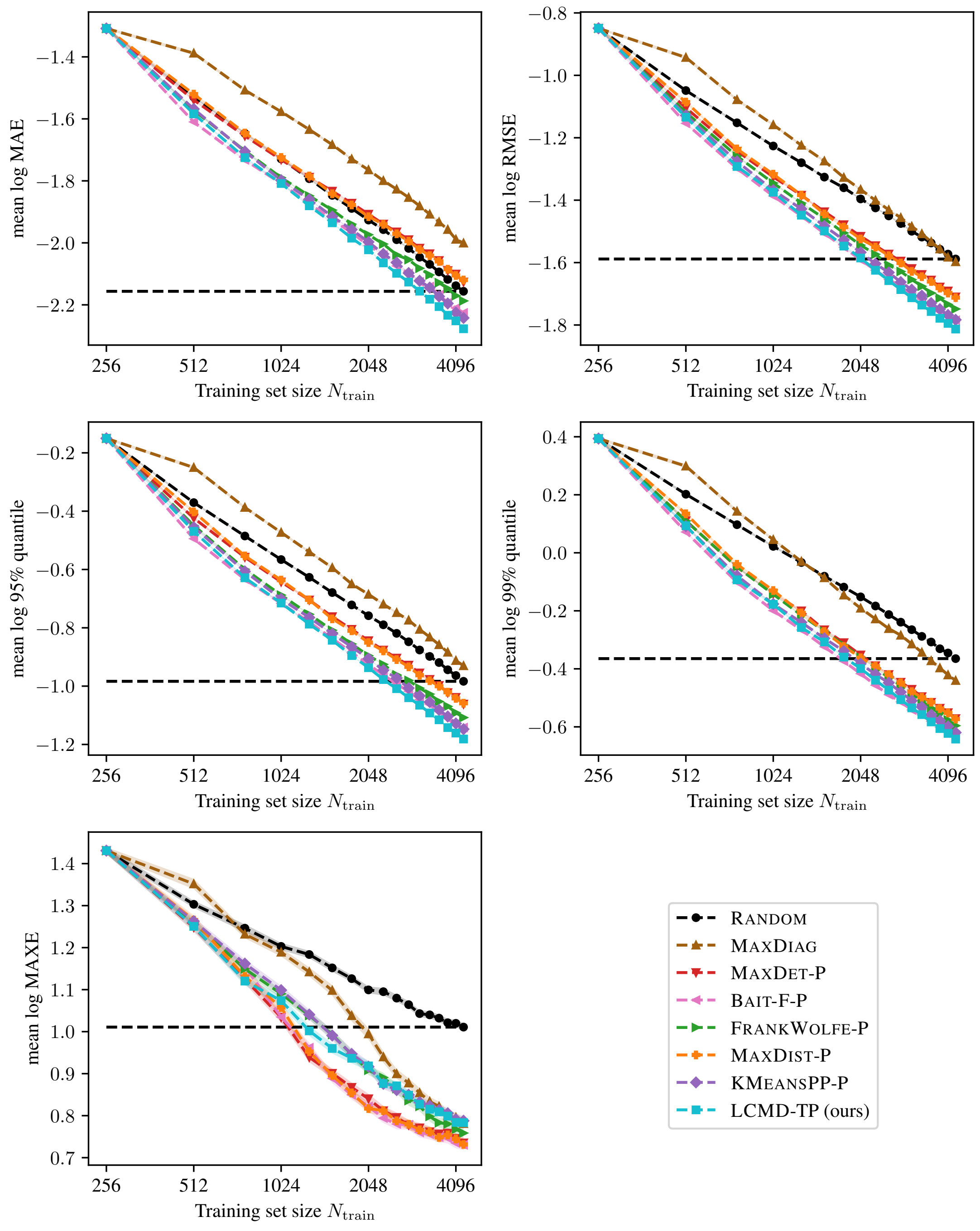


Figure 3: This figure shows how fast the errors decrease during BMAL for different selection methods and their corresponding kernels from Table 6. Specifically, for each of the five error metrics, the corresponding plot shows the logarithmic error metric between each BMAL step for $N_{\text{batch}} = 256$, averaged over all repetitions and data sets. The performance of Random can be interpreted as the performance of supervised learning without active learning. The black horizontal dashed line corresponds to the final performance of Random at $N_{\text{train}} = 4352$. The shaded area, which is nearly invisible for all metrics except MAXE, corresponds to one estimated standard deviation of the mean estimator, cf. Section 6.E.4.

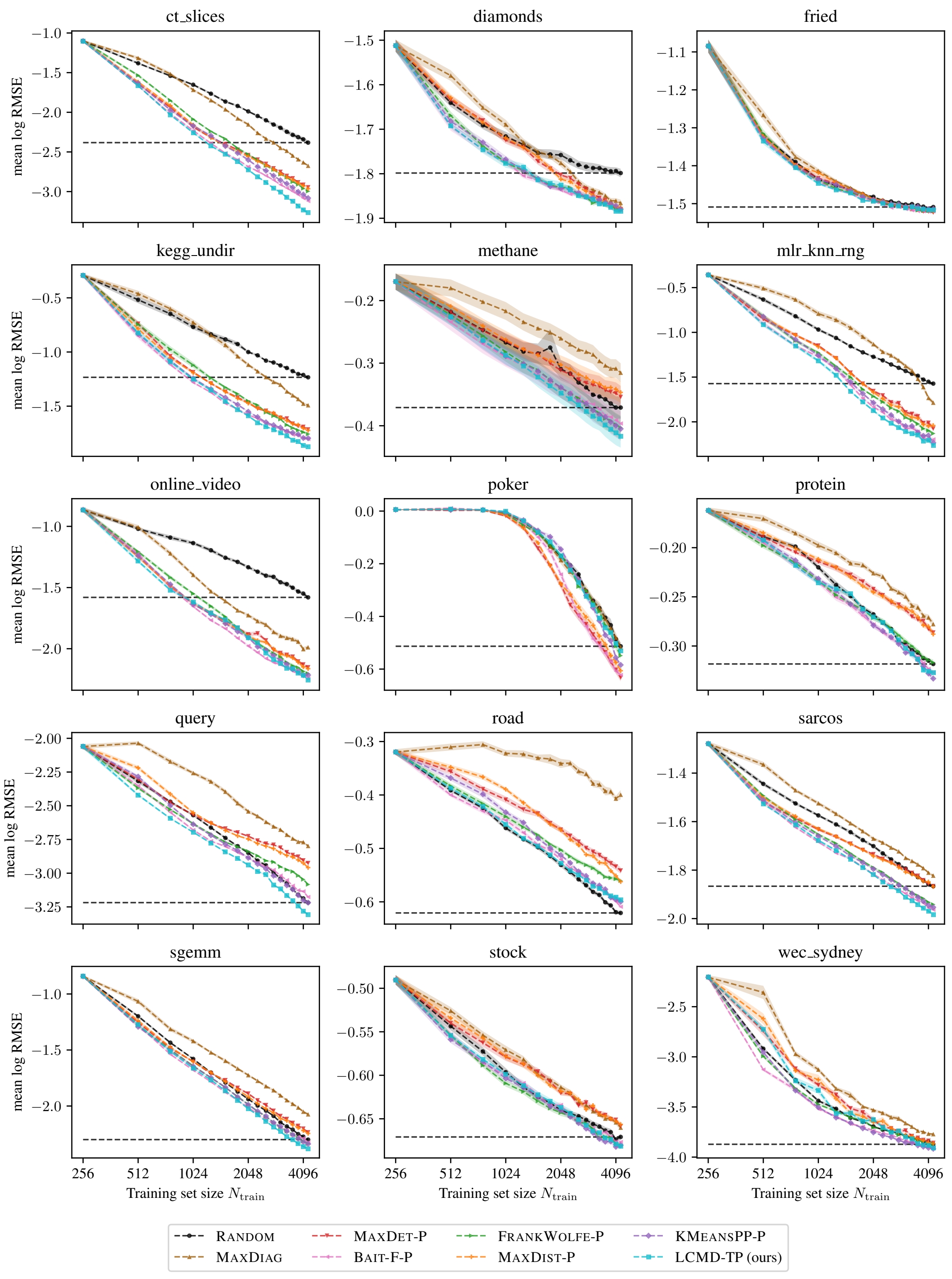


Figure 4: This figure shows how fast the RMSE decreases during BMAL on the individual benchmark data sets for different selection methods and their corresponding kernels from Table 6. Specifically, the plots above show the logarithmic RMSE between each BMAL step for $N_{\text{batch}} = 256$, averaged over all repetitions. The black horizontal dashed line corresponds to the final performance of RANDOM at $N_{\text{train}} = 4352$. The shaded area corresponds to one estimated standard deviation of the mean estimator, cf. Section 6.E.4.

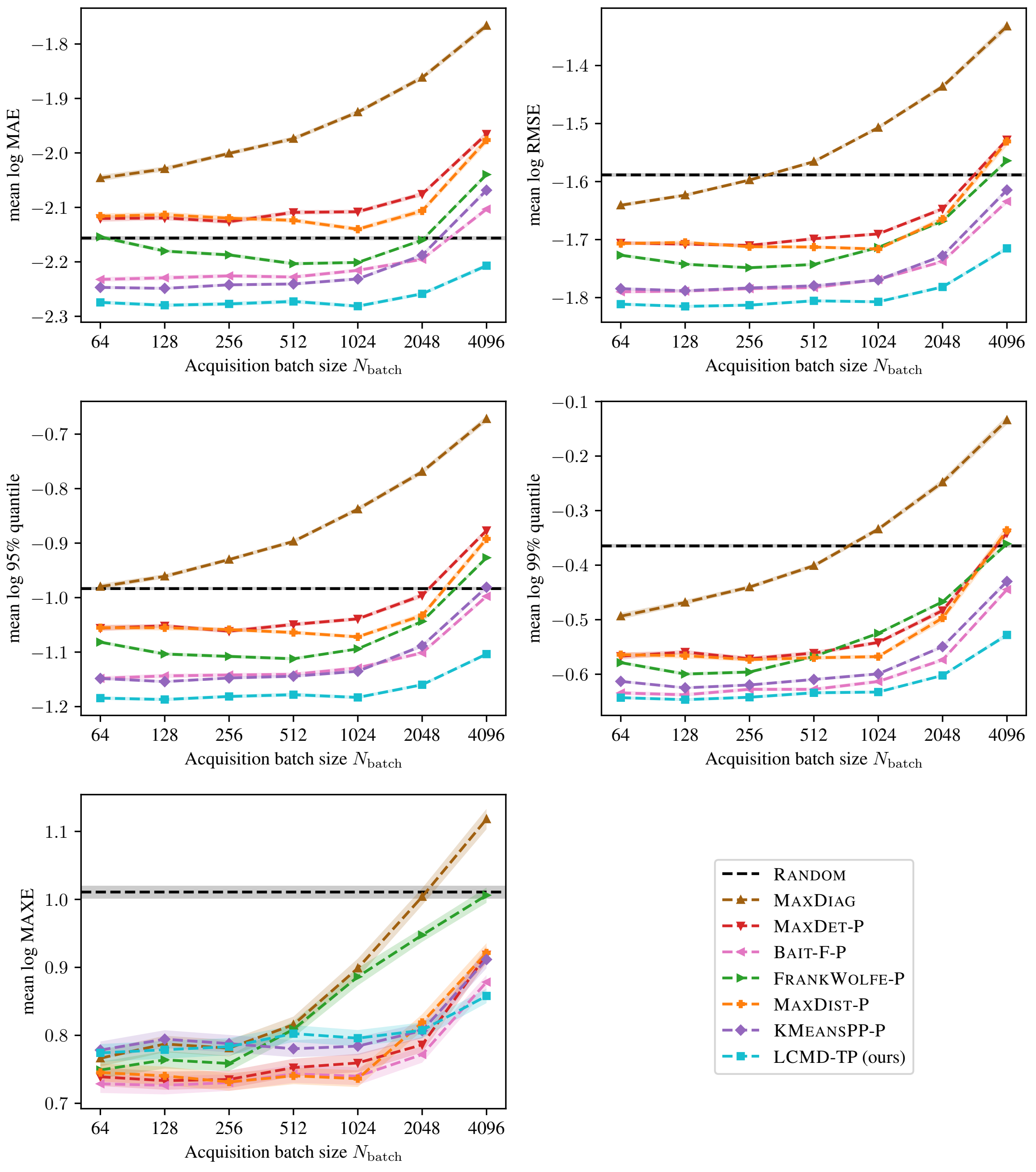


Figure 5: This figure shows how much the final accuracy of different BMDAL methods deteriorates when fewer BMAL steps with larger batch sizes are used. Specifically, we use different selection methods with the corresponding kernels from Table 6, starting with $N_{\text{train}} = 256$ and then performing 2^m BMAL steps with batch size $N_{\text{batch}} = 2^{12-m}$ for $m \in \{0, \dots, 6\}$, such that the final training set size is 4352 in each case. For each of the five error metrics, the corresponding plot shows the final logarithmic error metric, averaged over all data sets and repetitions. Note that the performance of RANDOM selection does not depend on N_{batch} but only on the final training set size, hence it is shown as a constant line here. The shaded area, which is nearly invisible for all metrics except MAXE, corresponds to one estimated standard deviation of the mean estimator, cf. Section 6.E.4.

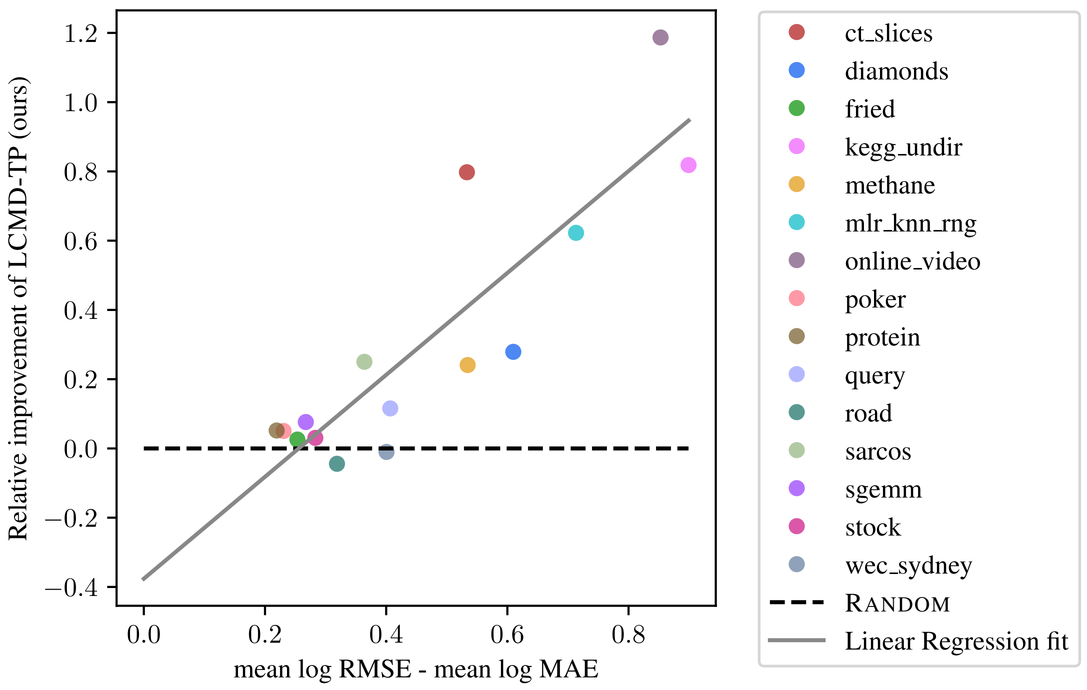


Figure 6: This figure shows, for each data set, the improvement in sample efficiency of LCMD-TP over Random (on the y-axis) versus the variation of the error distribution (on the x-axis). The variation is measured as mean log RMSE − mean log MAE on the initial training set ($N_{\text{train}} = 256$). The improvement in sample efficiency is measured by $\frac{\text{mean log RMSE(LCMD-TP)}-\text{mean log RMSE(R}\textsc{andom})}{\text{mean log RMSE(R}\textsc{andom})-(\text{mean log RMSE at } N_{\text{train}}=256)}$. Here, the means are taken over all 20 repetitions and, unless indicated otherwise, over the trained networks after each of the 16 BMDAL steps. The Pearson correlation coefficient for the plotted data is $R \approx 0.88$.

is particularly sensitive to the batch size. The other selection methods are less sensitive to changes in N_{batch} and exhibit almost no degradation in performance up to $N_{\text{batch}} = 1024$. Note that this "threshold" might depend on the initial and final training set sizes as well as the feature-space dimension d_{feat}.

Overall, the discussed figures and the detailed results in Table 6.E.5 show that LCMD-TP performs best in terms of MAE, RMSE, and 95% quantile, followed by Bait-F-P and KMeansPP. For MAXE, MaxDist, MaxDet and Bait-F-P exhibit the best performances. Since MaxDet and MaxDist are motivated by worst-case considerations, it is not surprising that they perform well on MAXE, while the strong performance of Bait-F-P on MAXE is unexpected. Moreover, it is perhaps surprising that in Figure 3, the relative performances for the 99% quantile are arguably more similar to the RMSE performances than to the MAXE performances. For the 95% quantile, the relative performances for MaxDet, MaxDist and MaxDiag are even worse than for the RMSE. Thus, the use of MaxDist instead of LCMD may only be advisable if one expects strong distribution shifts between pool and test sets.

6.6.5 When should BMDAL be Applied?

While LCMD-TP achieves excellent average performance, its benefits over Random selection vary strongly between data sets, as is evident from Figure 4. Overall, LCMD-TP with $k_{\mathrm{grad}\to\mathrm{sketch}(512)}$ outperforms random selection in terms of RMSE on 13 out of the 15 data sets and barely performs worse on the other two. Nonetheless, an *a priori* estimate of the benefits of LCMD-TP over Random selection on individual data sets could be useful to inform practitioners on whether they should be interested in applying BMDAL. Figure 6 shows that on our 15 benchmark data sets, the variation of test errors after training on the initial training set ($N_{\mathrm{train}} = 256$), measured by the quotient $\frac{\mathrm{RMSE}}{\mathrm{MAE}}$, is strongly correlated with the improvement in sample efficiency through LCMD-TP over Random. In other words, the larger $\frac{\mathrm{RMSE}}{\mathrm{MAE}}$ on the initial training set, the more benefit we can expect from LCMD-TP with $k_{\mathrm{grad}\to\mathrm{sketch}(512)}$ over random selection.

6.7 Conclusion

In this paper, we introduced a framework to compose BMDAL algorithms out of base kernels, kernel transformations, and selection methods. We then evaluated different combinations of these components on a new benchmark consisting of 15 large tabular regression data sets. In our benchmark results, for all considered selection methods, replacing the wide-spread last-layer kernel k_{ll} by a sketched finite-width neural tangent kernel $k_{\mathrm{grad}\to\mathrm{sketch}(p)}$ leads to accuracy improvements at similar runtime and memory cost. Moreover, our novel LCMD selection method sets new state-of-the-art results in our benchmark in terms of RMSE and MAE.

6.7.1 Limitations

The BMDAL methods in our framework are very attractive for practitioners using NNs for regression since they are scalable to large data sets and can be applied to a wide variety of NN architectures and training methods without requiring modifications to the NN. However, while our benchmark contains many large data sets, it cannot cover all possible application scenarios that the considered BMDAL methods could be applied to. For example, it is unclear whether our insights can be transferred to applications like drug discovery (Mehrjou et al., 2021) or atomistic ML (Zaverkin and Kästner, 2021), where other types of data and other NNs are employed. Even in the tabular data setting, the relevance of our results for smaller data sets or recently proposed NN architectures (e.g. Gorishniy et al., 2021; Somepalli et al., 2022; Kadra et al., 2021) is unclear. Moreover, the current benchmark does not involve distribution shifts between pool and test data, which would be interesting for some practical applications.

6.7.2 Remaining Questions

Our results give rise to some interesting questions for future research, of which we list some in the following: Can k_{grad} be adapted to incorporate effects of optimizers such as Adam? How can it be efficiently evaluated and sketched for other types of layers? How can we decide which method to use on a data set, beyond just using the one with the best average performance across the benchmark? Are there some characteristics of data sets or training setups that can be used to predict which method will perform best? Are clustering-based methods like LCMD-TP also superior to other methods for non-batch AL? How much better performance could be attained if the methods had access to the pool labels, and what kinds of batches would such a method select? Would it have similar properties as Zhou et al. (2021) found for classification? How can our framework be generalized to classification or multi-output regression?

Our framework is formulated for the pool-based AL setting, where samples should be selected from a pool of unlabeled samples. By using a different kind of selection methods with the same kernels and kernel transformations, our framework could be adapted to the streaming AL setting, where unlabeled samples arrive sequentially and one has to decide immediately whether to label them or not. For the membership-query AL setting, where unlabeled samples can be chosen arbitrarily, the situation is more difficult: Since the feature map ϕ is typically not surjective, samples cannot be chosen in the feature space directly, and a direct choice in the input space might require differentiating the kernel for efficient optimization. Nonetheless, extending our methods to other AL settings could be an interesting avenue for further research.

Acknowledgments We want to thank Philipp Hennig, Tizian Wenzel, Daniel Winkle, Benjamin Unger, and Paul Bürkner for helpful comments. Funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2075 – 390740016. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting David Holzmüller. Viktor Zaverkin acknowledges the financial support received in the form of a Ph.D. scholarship from the Studienstiftung des Deutschen Volkes (German National Academic Foundation).

Appendix

6.A Overview

The appendix structure mirrors the structure of the main paper, providing more details on the corresponding sections of the main paper: We discuss further details on base kernels in Appendix 6.B, on kernel transformations in Appendix 6.C, and on selection methods in Appendix 6.D. The latter section includes efficiency-focused pseudocode as well as discussions on relations to the literature. Finally, we provide details on the setup and results of our experiments in Appendix 6.E.

6.B Details on Base Kernels

In the following, we will provide details for the infinite-width NNGP kernel.

6.B.1 NNGP Kernel

For the fully-connected NN model considered in Section 6.2.1 with a ReLU activation function, the NNGP kernel is given by

$$k_{\mathrm{nngp}}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \coloneqq k_{\mathrm{nngp}}^{(L)}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) ,$$

where we roughly follow Lee et al. (2019) and recursively define

$$\begin{aligned}
k_{\mathrm{nngp}}^{(1)}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) &\coloneqq \frac{\sigma_w^2}{d}\langle \boldsymbol{x}, \tilde{\boldsymbol{x}}\rangle \\
k_{\mathrm{nngp}}^{(l+1)}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) &\coloneqq \sigma_w^2 f(k_{\mathrm{nngp}}^{(l)}(\boldsymbol{x}, \boldsymbol{x}), k_{\mathrm{nngp}}^{(l)}(\boldsymbol{x}, \tilde{\boldsymbol{x}}), k_{\mathrm{nngp}}^{(l)}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}})) \\
f(a, b, c) &\coloneqq \frac{\sqrt{ac}}{2\pi}\left(\sqrt{1-u^2} + u(\pi - \arccos(u))\right) \text{ with } u \coloneqq \frac{b}{\sqrt{ac}} .
\end{aligned}$$

Note that we do not include the σ_b terms here since we initialize the biases to zero unlike Lee et al. (2019).

6.C Details on Kernel Transformations

In the following, we will discuss additional aspects of various kernel transformations.

6.C.1 Gaussian Process Posterior Transformation

Khan et al. (2019) showed that certain posterior approximation methods for NNs turn them into Gaussian processes with the finite-width NTK k_{grad}. In the following, we will present a self-contained derivation of this relationship in our framework, including the last-layer kernel k_{ll}. In our exposition, we roughly follow Daxberger et al. (2021). For a Bayesian NN, we impose a prior $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta} \mid \mathbf{0}, \lambda^2\boldsymbol{I})$ on the NN parameters $\boldsymbol{\theta}$. Using an observation model $y_i = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + \varepsilon_i, \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ i.i.d., the negative log-likelihood of the data is given by

$$-\log p(\mathcal{Y}_{\mathrm{train}} \mid \mathcal{X}_{\mathrm{train}}, \boldsymbol{\theta}) = C + \frac{1}{2\sigma^2}\sum_{i=1}^{N_{\mathrm{train}}} (y_i - f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))^2 = C_1 + \frac{1}{2\sigma^2} N_{\mathrm{train}}\mathcal{L}(\boldsymbol{\theta})$$

for some constant $C_1 \in \mathbb{R}$. The negative log-posterior is hence given by

$$\begin{aligned}\tilde{\mathcal{L}}(\boldsymbol{\theta}) &\coloneqq -\log p(\boldsymbol{\theta} \mid \mathcal{Y}_{\mathrm{train}}, \mathcal{X}_{\mathrm{train}}) \\ &= \log(Z) - \log p(\mathcal{Y}_{\mathrm{train}} \mid \mathcal{X}_{\mathrm{train}}, \boldsymbol{\theta}) - \log p(\boldsymbol{\theta}) \\ &= C_2 + \frac{1}{2\sigma^2} N_{\mathrm{train}}\mathcal{L}(\boldsymbol{\theta}) + \frac{1}{2\lambda^2}\|\boldsymbol{\theta}\|_2^2 ,\end{aligned}$$

If $\boldsymbol{\theta}^*$ minimizes $\tilde{\mathcal{L}}$, that is, if $\boldsymbol{\theta}^*$ is a maximum *a posteriori* (MAP) estimate, we obtain the second-order Taylor approximation

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}) \approx \tilde{\mathcal{L}}(\boldsymbol{\theta}^*) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^\top \boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{\theta}^*), \quad \boldsymbol{H} \coloneqq \nabla_{\boldsymbol{\theta}}^2 \tilde{\mathcal{L}}(\boldsymbol{\theta}^*) ,$$

which yields a Gaussian approximation to the posterior, the so-called Laplace approximation (Laplace, 1774; MacKay, 1992a):

$$p(\boldsymbol{\theta} \mid \mathcal{Y}_{\mathrm{train}}, \mathcal{X}_{\mathrm{train}}) \approx \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*, \boldsymbol{H}^{-1}) .$$

The Hessian is given by

$$\begin{aligned}\boldsymbol{H} &= \frac{1}{\lambda^2}\boldsymbol{I} + \frac{1}{2\sigma^2}\sum_{(\boldsymbol{x},y)\in\mathcal{D}_{\mathrm{train}}} \nabla_{\boldsymbol{\theta}}^2 (y - f_{\boldsymbol{\theta}^*}(\boldsymbol{x}))^2 \\ &= \lambda^{-2}\boldsymbol{I} + \sigma^{-2}\sum_{(\boldsymbol{x},y)\in\mathcal{D}_{\mathrm{train}}} \big((\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}^*}(\boldsymbol{x}))(\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}^*}(\boldsymbol{x}))^\top + (f_{\boldsymbol{\theta}^*}(\boldsymbol{x}) - y)\nabla_{\boldsymbol{\theta}}^2 f_{\boldsymbol{\theta}^*}(\boldsymbol{x})\big) .\end{aligned}$$

By ignoring the terms $(f_{\boldsymbol{\theta}^*}(\boldsymbol{x}) - y)\nabla_{\boldsymbol{\theta}}^2 f_{\boldsymbol{\theta}^*}(\boldsymbol{x})$, which are expected to be small since the first factor is small, we arrive at the generalized Gauss-Newton (GGN) approximation to the Hessian (Schraudolph, 2002):

$$\boldsymbol{H}_{\mathrm{GGN}} = \lambda^{-2}\boldsymbol{I} + \sigma^{-2}\sum_{(\boldsymbol{x},y)\in\mathcal{D}_{\mathrm{train}}} (\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}^*}(\boldsymbol{x}))(\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}^*}(\boldsymbol{x}))^\top .$$

By pretending that $\boldsymbol{\theta}^* = \boldsymbol{\theta}_T$, i.e. that the parameters at the end of training are the minimizer of $\tilde{\mathcal{L}}$, we can relate this to ϕ_{grad}:

$$\boldsymbol{H}_{\mathrm{GGN}} = \lambda^{-2}\boldsymbol{I} + \sigma^{-2}\sum_{(\boldsymbol{x},y)\in\mathcal{D}_{\mathrm{train}}} \phi_{\mathrm{grad}}(\boldsymbol{x})\phi_{\mathrm{grad}}(\boldsymbol{x})^\top$$

$$= \lambda^{-2}\boldsymbol{I} + \sigma^{-2}\phi_{\mathrm{grad}}(\mathcal{X}_{\mathrm{train}})^\top \phi_{\mathrm{grad}}(\mathcal{X}_{\mathrm{train}}) \ .$$

If we want to compute the predictive distribution of $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ for $\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid \mathcal{X}_{\mathrm{train}})$, we can further use the linearization

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) \approx f_{\boldsymbol{\theta}^*}(\boldsymbol{x}) + \langle \boldsymbol{\theta} - \boldsymbol{\theta}^*, \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}^*}(\boldsymbol{x}) \rangle = f_{\boldsymbol{\theta}^*}(\boldsymbol{x}) + \langle \boldsymbol{\theta} - \boldsymbol{\theta}^*, \phi_{\mathrm{grad}}(\boldsymbol{x}) \rangle \ ,$$

which, according to Immer et al. (2021), improves the results for the predictive distribution. The predictive distribution can then be approximated as

$$p(\mathcal{Y} \mid \mathcal{X}, \mathcal{D}_{\mathrm{train}}) \approx \mathcal{N}(\mathcal{Y} \mid f_{\boldsymbol{\theta}^*}(\mathcal{X}), \phi_{\mathrm{grad}}(\mathcal{X}) \boldsymbol{H}_{\mathrm{GGN}}^{-1} \phi_{\mathrm{grad}}(\mathcal{X})^\top) \ ,$$

with the covariance matrix

$$\begin{aligned}
&\phi_{\mathrm{grad}}(\mathcal{X}) \boldsymbol{H}_{\mathrm{GGN}}^{-1} \phi_{\mathrm{grad}}(\mathcal{X})^\top \\
&= \phi_{\mathrm{grad}}(\mathcal{X})(\lambda^{-2}\boldsymbol{I} + \sigma^{-2}\phi_{\mathrm{grad}}(\mathcal{X}_{\mathrm{train}})^\top \phi_{\mathrm{grad}}(\mathcal{X}_{\mathrm{train}}))^{-1} \phi_{\mathrm{grad}}(\mathcal{X})^\top \\
&= \sigma^2 \lambda \phi_{\mathrm{grad}}(\mathcal{X})(\lambda\phi_{\mathrm{grad}}(\mathcal{X}_{\mathrm{train}})^\top \lambda\phi_{\mathrm{grad}}(\mathcal{X}_{\mathrm{train}}) + \sigma^2 \boldsymbol{I})^{-1} \lambda\phi_{\mathrm{grad}}(\mathcal{X})^\top \\
&= (\lambda^2 k_{\mathrm{grad}})_{\to \mathrm{post}(\mathcal{X}_{\mathrm{train}}, \sigma^2)}(\mathcal{X}, \mathcal{X}) \ .
\end{aligned}$$

This demonstrates that $(\lambda^2 k_{\mathrm{grad}})_{\to \mathrm{post}(\mathcal{X}_{\mathrm{train}}, \sigma^2)}(\mathcal{X}, \mathcal{X})$ yields the posterior predictive covariance on $\mathcal{X}$ of a Bayesian NN under the following approximations:

(1) The parameter posterior is approximated using the Laplace approximation,
(2) The Hessian matrix in the Laplace approximation is approximated using the GGN approximation,
(3) The predictive posterior is further approximated using a linearization of the NN, and
(4) The MAP estimate is approximated by the trained parameters of the NN.

If we perform Bayesian inference only over the last-layer parameters (i.e., $\boldsymbol{\theta} = \tilde{\boldsymbol{W}}^{(L)}$), the derivation above shows instead that the posterior predictive covariance is approximated by $(\lambda^2 k_{\mathrm{ll}})_{\to \mathrm{post}(\mathcal{X}_{\mathrm{train}}, \sigma^2)}(\mathcal{X}, \mathcal{X})$. Moreover, for the last-layer parameters, the approximations (1) – (3) are exact, since the NN is affine linear in the last-layer parameters, and the approximation (4) does not change the resulting kernel. Such last-layer Bayesian models have been employed, for example, by Lázaro-Gredilla and Figueiras-Vidal (2010); Snoek et al. (2015); Ober and Rasmussen (2019); Kristiadi et al. (2020), and are also known as neural linear models (Ober and Rasmussen, 2019).

Eschenhagen et al. (2021) experimentally demonstrated that taking a mixture of multiple Laplace approximations around different local minima of the loss function can improve uncertainty predictions for Bayesian NNs. If we consider N_{ens} local minima $\boldsymbol{\theta}^{(i)}$ with corresponding base kernels $k^{(i)}$ and combine the Laplace approximations with uniform weights, we obtain the posterior distributions

$$p(\mathcal{Y} \mid \mathcal{X}, \mathcal{D}_{\mathrm{train}}) \approx \frac{1}{N_{\mathrm{ens}}} \sum_{i=1}^{N_{\mathrm{ens}}} \mathcal{N}(\mathcal{Y} \mid f_{\boldsymbol{\theta}^{(i)}}(\mathcal{X}), (\lambda k^{(i)})_{\to \mathrm{post}(\mathcal{X}_{\mathrm{train}}, \sigma^2)}(\mathcal{X}, \mathcal{X})) \ .$$

By the law of total covariance, we have

$$\mathrm{Cov}(y_1, y_2 \mid \boldsymbol{x}_1, \boldsymbol{x}_2, \mathcal{D}_{\mathrm{train}}) \approx \mathbb{E}_{i \sim \mathcal{U}\{1, \ldots, N_{\mathrm{ens}}\}}[(\lambda^2 k^{(i)})_{\to \mathrm{post}(\mathcal{X}_{\mathrm{train}}, \sigma^2)}(\boldsymbol{x}_1, \boldsymbol{x}_2)]$$

$$\begin{aligned}
&+ \operatorname{Cov}_{i\sim\mathcal{U}\{1,\dots,N_{\mathrm{ens}}\}}(f_{\boldsymbol{\theta}^{(i)}}(\boldsymbol{x}_i), f_{\boldsymbol{\theta}^{(i)}}(\boldsymbol{x}_j)) \\
= &(\lambda^2 k)_{\to\mathrm{post}(\mathcal{X}_{\mathrm{train}},\sigma^2)\to\mathrm{ens}(N_{\mathrm{ens}})}(\boldsymbol{x}_1, \boldsymbol{x}_2) \\
&+ \operatorname{Cov}_{i\sim\mathcal{U}\{1,\dots,N_{\mathrm{ens}}\}}(f_{\boldsymbol{\theta}^{(i)}}(\boldsymbol{x}_i), f_{\boldsymbol{\theta}^{(i)}}(\boldsymbol{x}_j)) .
\end{aligned}$$

Hence, the predictive covariance for a mixture of Laplace approximations is approximately given by an ensembled posterior kernel plus the covariance of the ensemble predictions. Note that due to the summation, it is important that the (posterior) kernel is scaled correctly. We leave an experimental evaluation of this approach to future work.

6.C.2 Sketching

In the following, we prove the variant of the Johnson-Lindenstrauss theorem mentioned in Section 6.4.2.

Theorem 6.4.1 (Variant of the Johnson-Lindenstrauss Lemma). *Let $\varepsilon, \delta \in (0,1)$ and let $\mathcal{X} \subseteq \mathbb{R}^d$ be finite. If*

$$p \geq 8\log(|\mathcal{X}|^2/\delta)/\varepsilon^2 , \tag{6.14}$$

then the following bound on all pairwise distances holds with probability $\geq 1-\delta$ for the Gaussian sketch in Eq. (6.12):

$$\forall \boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathcal{X} : (1-\varepsilon)d_k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \leq d_{k_{\to\mathrm{sketch}(p)}}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \leq (1+\varepsilon)d_k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) . \tag{6.15}$$

Proof. By Theorem 1 from Arriaga and Vempala (1999), the following bound holds for fixed $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathcal{X}$ with probability[8] $\geq 1 - 2e^{-\varepsilon^2 p/8}$:

$$(1-\varepsilon)d_k(\boldsymbol{x}, \tilde{\boldsymbol{x}})^2 \leq d_{k_{\to\mathrm{sketch}(p)}}(\boldsymbol{x}, \tilde{\boldsymbol{x}})^2 \leq (1+\varepsilon)d_k(\boldsymbol{x}, \tilde{\boldsymbol{x}})^2 . \tag{6.26}$$

Now, Eq. (6.26) implies Eq. (6.15) for a single pair $\boldsymbol{x}, \tilde{\boldsymbol{x}}$ because of

$$(1-\varepsilon)^2 \leq (1-\varepsilon) \leq (1+\varepsilon) \leq (1+\varepsilon)^2 .$$

To obtain the bound for all pairs $(\boldsymbol{x}, \tilde{\boldsymbol{x}})$, we note that the bound is trivial for pairs $(\boldsymbol{x}, \boldsymbol{x})$ and it is equivalent for the pairs $(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ and $(\tilde{\boldsymbol{x}}, \boldsymbol{x})$. Hence, we only have to consider less than $\frac{|\mathcal{X}|^2}{2}$ pairs. By the union bound, the probability that Eq. (6.15) holds for all pairs $(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ is at least

$$1 - 2\frac{|\mathcal{X}|^2}{2}e^{-\varepsilon^2 p/8} \overset{\text{Eq. (6.14)}}{\geq} 1-\delta . \qquad \square$$

6.C.3 ACS Random Features Transformation

We will now derive the form of $k_{\to\mathrm{acs}}$ presented in Section 6.4.2. Partially following the notation of Pinsler et al. (2019), we want to compute

$$k_{\to\mathrm{acs}}(\boldsymbol{x}_n, \boldsymbol{x}_m) := \langle \mathcal{L}_n, \mathcal{L}_m \rangle = \mathbb{E}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta}|\mathcal{D}_{\mathrm{train}})}[\mathcal{L}_n(\boldsymbol{\theta})\mathcal{L}_m(\boldsymbol{\theta})] ,$$

where

$$\mathcal{L}_m(\boldsymbol{\theta}) = \mathbb{E}_{y_m\sim p(\cdot|\boldsymbol{x}_m,\mathcal{D}_{\mathrm{train}})}[\log p(y_m \mid \boldsymbol{x}_m, \boldsymbol{\theta})] + \mathbb{H}[y_m \mid \boldsymbol{x}_m, \mathcal{D}_{\mathrm{train}}]$$

[8]We inserted the factor 2 in front of $e^{-\varepsilon^2 p/8}$ that has been forgotten in their Theorem 1.

with $\mathbb{H}[y_m \mid \boldsymbol{x}_m, \mathcal{D}_{\text{train}}]$ denoting the conditional entropy of y_m given $\boldsymbol{x}_m$ and $\mathcal{D}_{\text{train}}$.

In a GP model without hyper-prior on σ^2, we have

$$\begin{aligned} p(y_m \mid \boldsymbol{x}_m, \boldsymbol{\theta}) &= \mathcal{N}(y_m \mid f_{\boldsymbol{\theta}_T}(\boldsymbol{x}_m) + \boldsymbol{\theta}^\top \phi_{\to\text{scale}(\mathcal{X}_{\text{train}})}(\boldsymbol{x}_m), \sigma^2) \\ p(y_m \mid \boldsymbol{x}_m, \mathcal{D}_{\text{train}}) &= \mathcal{N}(y_m \mid f_{\boldsymbol{\theta}_T}(\boldsymbol{x}_m), k_{\to\mathcal{X}_{\text{train}}}(\boldsymbol{x}_m, \boldsymbol{x}_m) + \sigma^2) \\ \mathbb{H}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})) &= \frac{n}{2}\ln(2\pi e) + \frac{1}{2}\ln(\det(\boldsymbol{\Sigma})) \quad \text{for } \boldsymbol{\mu} \in \mathbb{R}^n, \boldsymbol{\Sigma} \in \mathbb{R}^{n\times n} , \end{aligned}$$

which allows us to derive

$$\mathbb{H}[y_m \mid \boldsymbol{x}_m, \mathcal{D}_{\text{train}}] = \frac{1}{2} + \frac{1}{2}\log(2\pi(k_{\to\mathcal{X}_{\text{train}}}(\boldsymbol{x}_m, \boldsymbol{x}_m) + \sigma^2)) .$$

By shifting the prior and the log-likelihood by $f_{\boldsymbol{\theta}_T}(\boldsymbol{x}_m)$, we obtain

$$\begin{aligned} &\mathbb{E}_{y_m \sim p(\cdot \mid \boldsymbol{x}_m, \mathcal{D}_0)}[\log p(y_m \mid \boldsymbol{x}_m, \boldsymbol{\theta})] \\ &= \mathbb{E}_{y_m \sim \mathcal{N}(0, k_{\to\mathcal{X}_{\text{train}}}(\boldsymbol{x}_m,\boldsymbol{x}_m)+\sigma^2)}[\log \mathcal{N}(y_m \mid \boldsymbol{\theta}^\top \phi_{\to\text{scale}(\mathcal{X}_{\text{train}})}(\boldsymbol{x}_m), \sigma^2)] \\ &= -\frac{1}{2}\log(2\pi\sigma^2) - \mathbb{E}_{y_m \sim \mathcal{N}(0, k_{\to\mathcal{X}_{\text{train}}}(\boldsymbol{x}_m,\boldsymbol{x}_m)+\sigma^2)} \frac{(y_m - \boldsymbol{\theta}^\top \phi_{\to\text{scale}(\mathcal{X}_{\text{train}})}(\boldsymbol{x}_m))^2}{2\sigma^2} \\ &= -\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\left((\boldsymbol{\theta}^\top \phi_{\to\text{scale}(\mathcal{X}_{\text{train}})}(\boldsymbol{x}_m))^2 + k_{\to\mathcal{X}_{\text{train}}}(\boldsymbol{x}_m, \boldsymbol{x}_m) + \sigma^2\right) . \end{aligned}$$

Therefore, we have $k_{\to\text{acs}}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \coloneqq \mathbb{E}_{\boldsymbol{\theta} \sim P(\boldsymbol{\theta} \mid \mathcal{D}_{\text{train}})}[f_{\text{acs}}(\boldsymbol{x}, \boldsymbol{\theta}) f_{\text{acs}}(\tilde{\boldsymbol{x}}, \boldsymbol{\theta})]$ with

$$\begin{aligned} f_{\text{acs}}(\boldsymbol{x}_m, \boldsymbol{\theta}) \coloneqq \mathcal{L}_m(\boldsymbol{\theta}) = \frac{1}{2}\log\left(1 + \frac{k_{\to\mathcal{X}_{\text{train}}}(\boldsymbol{x}_m, \boldsymbol{x}_m)}{\sigma^2}\right) \\ - \frac{(\boldsymbol{\theta}^\top \phi_{\to\text{scale}(\mathcal{X}_{\text{train}})}(\boldsymbol{x}_m))^2 + k_{\to\mathcal{X}_{\text{train}}}(\boldsymbol{x}_m, \boldsymbol{x}_m)}{2\sigma^2} . \end{aligned}$$

Moreover, it follows from Eq. (4) in Pinsler et al. (2019) that

$$f_{\text{pool}}(\boldsymbol{\theta}) - f_{\text{batch}}(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}_m \in \mathcal{X}_{\text{pool}} \setminus \mathcal{X}_{\text{batch}}} \mathcal{L}_m(\boldsymbol{\theta}) = \sum_{\boldsymbol{x} \in \mathcal{X}_{\text{pool}} \setminus \mathcal{X}_{\text{batch}}} f_{\text{acs}}(\boldsymbol{x}, \boldsymbol{\theta}) .$$

6.D Details on Selection Methods

In this section, we will provide efficiency-focused pseudocode for all selection methods and analyze its runtime and memory complexity. Hereby, we will neglect that the required integer bit size for indexing elements of $\mathcal{X}_{\text{pool}}$ and $\mathcal{X}_{\text{train}}$ grows logarithmically with $N_{\text{pool}} + N_{\text{train}}$. For some selection methods, we will additionally discuss relations to the literature and theoretical properties. Section 6.D.1 will first provide a pseudocode structure for iterative selection, where the missing components are then specified for the respective selection methods in the subsequent sections. The following notation will be used throughout this section:

We allow having vectors and matrices indexed by points $\boldsymbol{x}$ instead of indices $i \in \mathbb{N}$, which we write with square brackets as $\boldsymbol{v}[\boldsymbol{x}]$ or $\boldsymbol{M}[\boldsymbol{x}, \tilde{\boldsymbol{x}}]$. In a practical implementation, where the points $\boldsymbol{x} \in \mathcal{X}_{\text{pool}}$ are for example numbered as $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_{\text{pool}}}$, one may simply use $\boldsymbol{v}[i]$ instead of $\boldsymbol{v}[\boldsymbol{x}_i]$. Again, we assume in pseudocode that all $\boldsymbol{x}$ are distinct, such that we can use set notation, but identical copies of $\boldsymbol{x}$ should be treated as distinct. This problem also disappears when using indices. We denote by $\boldsymbol{u} \odot \boldsymbol{v}$ the element-wise (Hadamard) product of the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$. Whenever an argmax is not unique, we leave the choice of the maximizer to the implementation.

Algorithm 4 Iterative selection algorithm template involving three customizable functions Init, Add and Next that are allowed to have side effects (i.e., read/write variables in Select).

```
function Select($k$, $\mathcal{X}_{\mathrm{train}}$, $\mathcal{X}_{\mathrm{pool}}$, $N_{\mathrm{batch}}$, mode $\in$ {P, TP})
    $\mathcal{X}_{\mathrm{mode}} \leftarrow \mathcal{X}_{\mathrm{train}}$ if mode = TP else $\emptyset$
    $\mathcal{X}_{\mathrm{cand}} \coloneqq \mathcal{X}_{\mathrm{mode}} \cup \mathcal{X}_{\mathrm{pool}}$
    $\mathcal{X}_{\mathrm{batch}} \leftarrow \emptyset$
    Init
    for $\boldsymbol{x}$ in $\mathcal{X}_{\mathrm{mode}}$ do
        Add($\boldsymbol{x}$)
    end for
    for $i$ from 1 to $N_{\mathrm{batch}}$ do
        $\boldsymbol{x} \leftarrow$Next
        if $\boldsymbol{x} \in \mathcal{X}_{\mathrm{batch}} \cup \mathcal{X}_{\mathrm{train}}$ (failed selection) then
            fill up $\mathcal{X}_{\mathrm{batch}}$ with $N_{\mathrm{batch}} - |\mathcal{X}_{\mathrm{batch}}|$ random samples from $\mathcal{X}_{\mathrm{pool}} \setminus \mathcal{X}_{\mathrm{batch}}$ and return $\mathcal{X}_{\mathrm{batch}}$
        end if
        $\mathcal{X}_{\mathrm{batch}} \leftarrow \mathcal{X}_{\mathrm{batch}} \cup \{\boldsymbol{x}\}$
        Add($\boldsymbol{x}$)
    end for
    return $\mathcal{X}_{\mathrm{batch}}$
end function
```

6.D.1 Iterative Selection Scheme

While our iterative selection template shown in Algorithm 3 is sufficient for a high-level understanding, it is not well-suited for an efficient implementation. To this end, we present a more detailed iterative selection template in Algorithm 3, which closely matches our open-source implementation. The template in Algorithm 3 involves three methods called Init, Add, and Next, which are allowed to have side effects, i.e. access and modify common variables. For each selection method except Random and MaxDiag, our implementation directly mirrors this structure, containing a class providing the three methods together with Select from Algorithm 3.

6.D.2 Random

We implement Random by taking the first N_{batch} indices out of a random permutation of $\{1, \dots, N_{\mathrm{pool}}\}$. Since there are $N_{\mathrm{pool}}!$ possible random permutations, this requires at least $\log_2(N_{\mathrm{pool}}!) = \mathcal{O}(N_{\mathrm{pool}} \log N_{\mathrm{pool}})$ random bits, so the runtime for this suboptimal implementation is, in theory, "only" $\mathcal{O}(N_{\mathrm{pool}} \log N_{\mathrm{pool}})$, which is still extremely fast in practice. The memory complexity for our suboptimal implementation is $\mathcal{O}(N_{\mathrm{pool}})$.

6.D.3 MaxDiag

A simple implementation of MaxDiag is shown in Algorithm 5. The runtime of this implementation is $\mathcal{O}(N_{\mathrm{pool}} \log N_{\mathrm{pool}})$ due to sorting. While other algorithms might be faster

Algorithm 5 MaxDiag pseudocode implementation using Algorithm 4.

```
function Init
    Sort elements in $\mathcal{X}_{\text{pool}}$ as $\tilde{\boldsymbol{x}}_1, \dots, \tilde{\boldsymbol{x}}_{N_{\text{pool}}}$ such that $k(\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_1) \geq \dots \geq k(\tilde{\boldsymbol{x}}_{N_{\text{pool}}}, \tilde{\boldsymbol{x}}_{N_{\text{pool}}})$
end function

function Add($\boldsymbol{x}$)
end function

function Next
    return $\tilde{\boldsymbol{x}}_i$
end function
```

for $N_{\text{batch}} \ll N_{\text{pool}}$, the runtime is already very fast in practice. The memory complexity is $\mathcal{O}(N_{\text{pool}})$.

6.D.4 MaxDet

Equivalence of MaxDet to Non-batch Mode Active Learning With Fixed Kernel

Using the Schur determinant formula

$$\det \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix} = \det(\boldsymbol{A}) \det(\boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}) \ ,$$

we can compute

$$\begin{aligned}
&\det(k(\mathcal{X} \cup \{\boldsymbol{x}\}, \mathcal{X} \cup \{\boldsymbol{x}\}) + \sigma^2 \boldsymbol{I}) \\
&= \det \begin{pmatrix} k(\mathcal{X}, \mathcal{X}) + \sigma^2 \boldsymbol{I} & k(\mathcal{X}, \boldsymbol{x}) \\ k(\boldsymbol{x}, \mathcal{X}) & k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2 \end{pmatrix} \\
&= \det(k(\mathcal{X}, \mathcal{X}) + \sigma^2 \boldsymbol{I}) \det(k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2 - k(\boldsymbol{x}, \mathcal{X})(k(\mathcal{X}, \mathcal{X}) + \sigma^2 \boldsymbol{I})^{-1} k(\mathcal{X}, \boldsymbol{x})) \\
&= \det(k(\mathcal{X}, \mathcal{X}) + \sigma^2 \boldsymbol{I}) \cdot (\sigma^2 + k_{\to \text{post}(\mathcal{X}, \sigma^2)}(\boldsymbol{x}, \boldsymbol{x})) \ . \qquad (6.27)
\end{aligned}$$

This shows that

$$\operatorname*{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\text{rem}}} k_{\to \text{post}(\mathcal{X}_{\text{sel}}, \sigma^2)}(\boldsymbol{x}, \boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\text{rem}}} \det(k(\mathcal{X}_{\text{sel}} \cup \{\boldsymbol{x}\}, \mathcal{X}_{\text{sel}} \cup \{\boldsymbol{x}\}) + \sigma^2 \boldsymbol{I}) \ .$$

Equivalence of MaxDet to BatchBALD on a GP

As in Section 6.4.2, we consider a GP model in feature space, given by $y_i = \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) + \varepsilon_i$ with weight prior $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and i.i.d. observation noise $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. The objective of BatchBALD (Kirsch et al., 2019) is to maximize the mutual information

$$a(\mathcal{X}_{\text{batch}}) := \mathbb{H}(\mathcal{Y}_{\text{batch}} \mid \mathcal{X}_{\text{batch}}, \mathcal{D}_{\text{train}}) - \mathbb{E}_{\boldsymbol{w} \sim p(\boldsymbol{w} \mid \mathcal{D}_{\text{train}})} \mathbb{H}(\mathcal{Y}_{\text{batch}} \mid \mathcal{X}_{\text{batch}}, \mathcal{D}_{\text{train}}, \boldsymbol{w}) \ ,$$

where $\mathbb{H}$ refers to the (conditional) entropy. Writing $\mathcal{Y}^*_{\text{batch}} := \mathbb{E}[\mathcal{Y}_{\text{batch}} \mid \mathcal{X}_{\text{batch}}, \mathcal{D}_{\text{train}}]$, we have

$$p(\mathcal{Y}_{\text{batch}} \mid \mathcal{X}_{\text{batch}}, \mathcal{D}_{\text{train}}) = \mathcal{N}(\mathcal{Y}_{\text{batch}} \mid \mathcal{Y}^*_{\text{batch}}, k(\mathcal{X}_{\text{batch}}, \mathcal{X}_{\text{batch}}) + \sigma^2 \boldsymbol{I})$$

$$p(\mathcal{Y}_{\text{batch}} \mid \mathcal{X}_{\text{batch}}, \mathcal{D}_{\text{train}}, \boldsymbol{w}) = \mathcal{N}(\mathcal{Y}_{\text{batch}} \mid \phi(\mathcal{X}_{\text{train}})^\top \boldsymbol{w}, \sigma^2 \boldsymbol{I})$$
$$\mathbb{H}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})) = \frac{n}{2}\ln(2\pi e) + \frac{1}{2}\ln(\det(\boldsymbol{\Sigma})) \quad \text{for } \boldsymbol{\mu} \in \mathbb{R}^n, \boldsymbol{\Sigma} \in \mathbb{R}^{n\times n} .$$

Hence, we can compute

$$a(\mathcal{X}_{\text{batch}}) = \frac{1}{2}\ln(\det(k(\mathcal{X}_{\text{batch}}, \mathcal{X}_{\text{batch}}) + \sigma^2 \boldsymbol{I})) - N_{\text{batch}} \ln(\sigma) . \tag{6.28}$$

This shows that greedy maximization of the BatchBALD acquisition function, as proposed by Kirsch et al. (2019), is equivalent to MaxDet. Kirsch et al. (2019) showed that for any Bayesian model, greedy optimization of a is suboptimal by a factor of at most $(1 - 1/e)$. By applying this result to the GP model, we obtain that the same suboptimality bound applies to MaxDet for the form of a given in Eq. (6.28).

Equivalence of MaxDet to the P-greedy Algorithm

In our notation, the P-greedy algorithm (De Marchi et al., 2005) can be written as

$$\text{NextSample}(k, \mathcal{X}_{\text{sel}}, \mathcal{X}_{\text{rem}}) = \operatorname*{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\text{rem}}} P_{k,\mathcal{X}_{\text{sel}}}(\boldsymbol{x}) ,$$

where by Lemma 4.1 in De Marchi et al. (2005), the non-negative power function $P_{k,\mathcal{X}_{\text{sel}}}$ can be written as

$$P_{k,\mathcal{X}_{\text{sel}}}(\boldsymbol{x})^2 = k(\boldsymbol{x}, \boldsymbol{x}) - k(\boldsymbol{x}, \mathcal{X}_{\text{sel}}) k(\mathcal{X}_{\text{sel}}, \mathcal{X}_{\text{sel}})^{-1} k(\mathcal{X}_{\text{sel}}, \boldsymbol{x}) .$$

A calculation analogous to Eq. (6.27) therefore shows that P-greedy is equivalent to MaxDet with $\sigma^2 = 0$.

Relation to the Greedy Algorithm for D-optimal Design

In our notation, the D-optimal design problem (Wald, 1943) is to maximize the determinant $\det(\phi(\mathcal{X}_{\text{sel}})^\top \phi(\mathcal{X}_{\text{sel}}))$, which can only be nonzero in the case $N_{\text{sel}} \geq d_{\text{feat}}$. It can be seen as the $\sigma \to 0$ limit for $N_{\text{sel}} \geq d_{\text{feat}}$ of the determinant-maximization objective that motivates MaxDet since an eigenvalue-based argument shows that

$$\det(\phi(\mathcal{X}_{\text{sel}})^\top \phi(\mathcal{X}_{\text{sel}}) + \sigma^2 \boldsymbol{I}) = \sigma^{d_{\text{feat}} - N_{\text{sel}}} \det(k(\mathcal{X}_{\text{sel}}, \mathcal{X}_{\text{sel}}) + \sigma^2 \boldsymbol{I}) .$$

In this sense, the corresponding greedy algorithm (Wynn, 1970) is the underparameterized ($d_{\text{feat}} \leq N_{\text{sel}}$) analog to the P-greedy algorithm, since the latter can only be well-defined in the overparameterized regime ($d_{\text{feat}} \geq N_{\text{sel}}$). Some guarantees for this greedy algorithm are given by Wynn (1970) and Madan et al. (2019). While the classical D-optimal design uses $\sigma = 0$, Bayesian D-optimal design uses $\sigma > 0$ and is thus even more directly related to MaxDet (Chaloner and Verdinelli, 1995).

Kernel-space Implementation of MaxDet

In the following, we want to derive an efficient kernel-space implementation of MaxDet. Let $\mathcal{X}_{\text{cand}} \coloneqq \mathcal{X}_{\text{mode}} \cup \mathcal{X}_{\text{pool}}$. We perform a partial pivoted Cholesky decomposition of the matrix $\boldsymbol{M} = k(\mathcal{X}_{\text{cand}}, \mathcal{X}_{\text{cand}}) + \sigma^2 \boldsymbol{I}$, which has been suggested for P-greedy by Pazouki and Schaback (2011) and in the context of determinantal point processes by Chen et al. (2018). We denote submatrices of $\boldsymbol{M}$ for example by $\boldsymbol{M}[\mathcal{X}, \boldsymbol{x}] := (M_{\tilde{\boldsymbol{x}},\boldsymbol{x}})_{\tilde{\boldsymbol{x}} \in \mathcal{X}}$.

Suppose that at the current step, the points $\mathcal{X} := \mathcal{X}_{\text{mode}} \cup \mathcal{X}_{\text{batch}}$ have already been added. Consider the Cholesky decomposition $\boldsymbol{M}[\mathcal{X}, \mathcal{X}] = \boldsymbol{L}(\mathcal{X})\boldsymbol{L}(\mathcal{X})^\top$. Then, the Cholesky decomposition for $\mathcal{X} \cup \{\boldsymbol{x}\}$ is of the form

$$\begin{aligned} \begin{pmatrix} \boldsymbol{M}[\mathcal{X},\mathcal{X}] & \boldsymbol{M}[\mathcal{X},\boldsymbol{x}] \\ \boldsymbol{M}[\boldsymbol{x},\mathcal{X}] & \boldsymbol{M}[\boldsymbol{x},\boldsymbol{x}] \end{pmatrix} &= \boldsymbol{L}(\mathcal{X} \cup \{\boldsymbol{x}\})\boldsymbol{L}(\mathcal{X} \cup \{\boldsymbol{x}\})^\top \\ &= \begin{pmatrix} \boldsymbol{L}(\mathcal{X}) & \boldsymbol{0} \\ \boldsymbol{b}(\mathcal{X},\boldsymbol{x})^\top & \sqrt{c(\mathcal{X},\boldsymbol{x})} \end{pmatrix} \begin{pmatrix} \boldsymbol{L}(\mathcal{X}) & \boldsymbol{0} \\ \boldsymbol{b}(\mathcal{X},\boldsymbol{x})^\top & \sqrt{c(\mathcal{X},\boldsymbol{x})} \end{pmatrix}^\top , \end{aligned} \tag{6.29}$$

which implies $\boldsymbol{b}(\mathcal{X}, \boldsymbol{x}) = \boldsymbol{L}(\mathcal{X})^{-1}\boldsymbol{M}[\mathcal{X}, \boldsymbol{x}]$ and $c(\mathcal{X}, \boldsymbol{x}) = \boldsymbol{M}[\boldsymbol{x}, \boldsymbol{x}] - \|\boldsymbol{b}(\mathcal{X}, \boldsymbol{x})\|_2^2$. Using the general inversion formula for block-triangular matrices given by

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{B} & \boldsymbol{C} \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{A}^{-1} & \boldsymbol{0} \\ -\boldsymbol{C}^{-1}\boldsymbol{B}\boldsymbol{A}^{-1} & \boldsymbol{C}^{-1} \end{pmatrix} ,$$

we obtain

$$\begin{aligned} \boldsymbol{b}(\mathcal{X} \cup \{\boldsymbol{x}\}, \tilde{\boldsymbol{x}}) &= \boldsymbol{L}(\mathcal{X} \cup \{\boldsymbol{x}\})^{-1}\boldsymbol{M}[\mathcal{X} \cup \{\boldsymbol{x}\}, \tilde{\boldsymbol{x}}] \\ &= \begin{pmatrix} \boldsymbol{L}(\mathcal{X})^{-1} & \boldsymbol{0} \\ -c(\mathcal{X},\boldsymbol{x})^{-1/2}\boldsymbol{b}(\mathcal{X},\boldsymbol{x})^\top \boldsymbol{L}(\mathcal{X})^{-1} & c(\mathcal{X},\boldsymbol{x})^{-1/2} \end{pmatrix} \begin{pmatrix} \boldsymbol{M}[\mathcal{X},\tilde{\boldsymbol{x}}] \\ \boldsymbol{M}[\boldsymbol{x},\tilde{\boldsymbol{x}}] \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{b}(\mathcal{X},\tilde{\boldsymbol{x}}) \\ c(\mathcal{X},\boldsymbol{x})^{-1/2}(\boldsymbol{M}[\boldsymbol{x},\tilde{\boldsymbol{x}}] - \boldsymbol{b}(\mathcal{X},\boldsymbol{x})^\top \boldsymbol{b}(\mathcal{X},\tilde{\boldsymbol{x}})) \end{pmatrix} \end{aligned} \tag{6.30}$$

and therefore

$$c(\mathcal{X} \cup \{\boldsymbol{x}\}, \tilde{\boldsymbol{x}}) = c(\mathcal{X}, \tilde{\boldsymbol{x}}) - c(\mathcal{X}, \boldsymbol{x})^{-1}(\boldsymbol{M}[\boldsymbol{x}, \tilde{\boldsymbol{x}}] - \boldsymbol{b}(\mathcal{X}, \boldsymbol{x})^\top \boldsymbol{b}(\mathcal{X}, \tilde{\boldsymbol{x}}))^2 . \tag{6.31}$$

With the above considerations, we can implement MaxDet as follows, which is given as pseudocode in Algorithm 6:

- For Init, we initialize $\boldsymbol{b}(\emptyset, \boldsymbol{x})$ to an empty vector and $\boldsymbol{c}(\emptyset, \boldsymbol{x}) = M[\boldsymbol{x}, \boldsymbol{x}]$. We do not precompute $\boldsymbol{M}$ since not all entries of $\boldsymbol{M}$ will be used, otherwise, the runtime complexity would be quadratic in N_{cand}.
- For Next, note that by Eq. (6.29),

  $$\det \boldsymbol{M}[\mathcal{X} \cup \{\boldsymbol{x}\}, \mathcal{X} \cup \{\boldsymbol{x}\}] = c(\mathcal{X}, \boldsymbol{x}) \cdot \det(\boldsymbol{L}(\mathcal{X}))^2 ,$$

  hence

  $$\operatorname*{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\text{pool}} \setminus \mathcal{X}_{\text{batch}}} \det \boldsymbol{M}[\mathcal{X} \cup \{\boldsymbol{x}\}, \mathcal{X} \cup \{\boldsymbol{x}\}] = \operatorname*{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\text{pool}} \setminus \mathcal{X}_{\text{batch}}} c(\mathcal{X}, \boldsymbol{x}) .$$

- For Add, we update the $\boldsymbol{b}$ and c values as in Eq. (6.30) and Eq. (6.31).

Regarding the runtime complexity, the Add step is clearly the most expensive one, requiring $\mathcal{O}(N_{\text{cand}} N_{\text{sel}})$ operations for computing $\boldsymbol{B}^\top \boldsymbol{B}[\cdot, \boldsymbol{x}]$ and $\mathcal{O}(N_{\text{cand}} T_k)$ operations for computing $k(\mathcal{X}_{\text{cand}}, \boldsymbol{x})$. Since Add is called N_{sel} times, the total runtime complexity is therefore $\mathcal{O}(N_{\text{cand}} N_{\text{sel}}(T_k + N_{\text{sel}}))$. The total memory complexity is $\mathcal{O}(N_{\text{cand}} N_{\text{sel}})$, required for storing $\boldsymbol{B}$.

Algorithm 6 MaxDet pseudocode implementation in kernel space for given $\sigma^2 \geq 0$ using Algorithm 4.

```
function Init
    $\boldsymbol{B} \leftarrow$ empty $0 \times \mathcal{X}_{\mathrm{cand}}$ matrix
    $\boldsymbol{c} \leftarrow (k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2)_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{cand}}}$
end function

function Add($\boldsymbol{x}$)
    $\boldsymbol{v} \leftarrow \sqrt{\boldsymbol{c}} \odot ((k(\mathcal{X}_{\mathrm{cand}}, \boldsymbol{x}) + \sigma^2 \boldsymbol{I}[\mathcal{X}_{\mathrm{cand}}, \boldsymbol{x}] - \boldsymbol{B}^\top \boldsymbol{B}[\cdot, \boldsymbol{x}])$    $\triangleright$ $\sqrt{\boldsymbol{c}}$ should be understood element-wise
    $\boldsymbol{B} \leftarrow \begin{pmatrix} \boldsymbol{B} \\ \boldsymbol{v}^\top \end{pmatrix}$
    $\boldsymbol{c} \leftarrow \boldsymbol{c} - \boldsymbol{v} \odot \boldsymbol{v}$
end function

function Next
    return $\operatorname{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{pool}} \setminus \mathcal{X}_{\mathrm{batch}}} c[\boldsymbol{x}]$
end function
```

Feature-space Implementation of MaxDet

In cases where the feature space dimension d_{feat} of ϕ is smaller than the number N_{sel} of added samples, it can be beneficial to implement MaxDet in feature space instead. For $\mathcal{X}_{\mathrm{sel}} := \mathcal{X}_{\mathrm{mode}} \cup \mathcal{X}_{\mathrm{batch}}$, we want to compute

$$\underset{\boldsymbol{x} \in \mathcal{X}_{\mathrm{pool}} \setminus \mathcal{X}_{\mathrm{batch}}}{\operatorname{argmax}} \quad k_{\to \mathrm{post}(\mathcal{X}_{\mathrm{sel}}, \sigma^2)}(\boldsymbol{x}, \boldsymbol{x}) \ .$$

Now, from Eq. (6.9), we know that

$$k_{\to \mathrm{post}(\mathcal{X}_{\mathrm{sel}}, \sigma^2)}(\boldsymbol{x}, \boldsymbol{x}) = \phi(\boldsymbol{x})^\top \hat{\boldsymbol{\Sigma}}_{\mathcal{X}_{\mathrm{sel}}}^{-1} \phi(\boldsymbol{x}), \qquad \hat{\boldsymbol{\Sigma}}_{\mathcal{X}_{\mathrm{sel}}} := \sigma^{-2} \phi(\mathcal{X}_{\mathrm{sel}})^\top \phi(\mathcal{X}_{\mathrm{sel}}) + \boldsymbol{I} \ .$$

Adding a point to $\mathcal{X}_{\mathrm{sel}}$ leads to a rank-1 update of $\hat{\boldsymbol{\Sigma}}_{\mathcal{X}_{\mathrm{sel}}}$, and the corresponding update of $\hat{\boldsymbol{\Sigma}}_{\mathcal{X}_{\mathrm{sel}}}^{-1}$ can be computed using the Sherman-Morrison formula. This gives rise to three approaches towards implementing MaxDet in feature space:

(1) Keep track of $\phi(\boldsymbol{x})$ and $\hat{\boldsymbol{\Sigma}}_{\mathcal{X}_{\mathrm{sel}}}^{-1}$. Compute $k_{\to \mathrm{post}(\mathcal{X}_{\mathrm{sel}}, \sigma^2)}(\boldsymbol{x}, \boldsymbol{x}) = \phi(\boldsymbol{x})^\top \hat{\boldsymbol{\Sigma}}_{\mathcal{X}_{\mathrm{sel}}}^{-1} \phi(\boldsymbol{x})$ in each step.

(2) Keep track of $\phi(\boldsymbol{x})$ and $\psi(\boldsymbol{x}) := \hat{\boldsymbol{\Sigma}}_{\mathcal{X}_{\mathrm{sel}}}^{-1} \phi(\boldsymbol{x})$. Compute

$$k_{\to \mathrm{post}(\mathcal{X}_{\mathrm{sel}}, \sigma^2)}(\boldsymbol{x}, \boldsymbol{x}) = \phi(\boldsymbol{x})^\top \psi(\boldsymbol{x})$$

in each step.

(3) Keep track of $\phi_{\to \mathrm{post}(\mathcal{X}_{\mathrm{sel}}, \sigma^2)}(\boldsymbol{x})$, one possible realization of which is $\phi_{\to \mathrm{post}(\mathcal{X}_{\mathrm{sel}}, \sigma^2)}(\boldsymbol{x}) = \hat{\boldsymbol{\Sigma}}_{\mathcal{X}_{\mathrm{sel}}}^{-1/2} \phi(\boldsymbol{x})$. Compute $k_{\to \mathrm{post}(\mathcal{X}_{\mathrm{sel}}, \sigma^2)}(\boldsymbol{x}, \boldsymbol{x}) = \phi_{\to \mathrm{post}(\mathcal{X}_{\mathrm{sel}}, \sigma^2)}(\boldsymbol{x})^\top \phi_{\to \mathrm{post}(\mathcal{X}_{\mathrm{sel}}, \sigma^2)}(\boldsymbol{x})$ in each step.

Option (1) is less computationally efficient than (2), (3) since one needs to compute matrix-vector products instead of only inner products. Version (2) and (3) are similar, but here we favor (3) since it only requires storing one vector instead of two for each $\boldsymbol{x}$. Since

$$\phi_{\to\mathrm{post}(\mathcal{X}_{\mathrm{sel}}\cup\{\boldsymbol{x}\},\sigma^2)} = \phi_{\to\mathrm{post}(\mathcal{X}_{\mathrm{sel}},\sigma^2)\to\mathrm{post}(\{\boldsymbol{x}\},\sigma^2)}~,$$

we will now consider how to efficiently compute a single posterior update $\phi_{\to\mathrm{post}(\{\boldsymbol{x}\},\sigma^2)}$. To this end, we first consider how to compute matrix square roots of specific rank-1 updates:

Lemma 6.D.1. *Let $\boldsymbol{v} \in \mathbb{R}^p$ and let $c \geq -\frac{1}{\boldsymbol{v}^\top\boldsymbol{v}}$. Then,*

$$\boldsymbol{I} + c\boldsymbol{v}\boldsymbol{v}^\top = \left(\boldsymbol{I} + \frac{c}{1+\sqrt{1+c\boldsymbol{v}^\top\boldsymbol{v}}}\boldsymbol{v}\boldsymbol{v}^\top\right)^2~.$$

Proof. Due to the condition on c, the square root is well-defined. We have

$$\left(\boldsymbol{I} + \frac{c}{1+\sqrt{1+c\boldsymbol{v}^\top\boldsymbol{v}}}\boldsymbol{v}\boldsymbol{v}^\top\right)^2 = \boldsymbol{I} + C\boldsymbol{v}\boldsymbol{v}^\top~,$$

where

$$C = \frac{2c}{1+\sqrt{1+c\boldsymbol{v}^\top\boldsymbol{v}}} + \frac{c^2\boldsymbol{v}^\top\boldsymbol{v}}{(1+\sqrt{1+c\boldsymbol{v}^\top\boldsymbol{v}})^2} = \frac{2c(1+\sqrt{1+c\boldsymbol{v}^\top\boldsymbol{v}}) + c^2\boldsymbol{v}^\top\boldsymbol{v}}{2+2\sqrt{1+c\boldsymbol{v}^\top\boldsymbol{v}}+c\boldsymbol{v}^\top\boldsymbol{v}} = c~. \qquad \square$$

The following proposition shows how to update the posterior feature map after observing a point $\boldsymbol{x}$:

Proposition 6.D.2 (Forward update)**.** *Let $\sigma^2 > 0$, let k be a kernel and let $\tilde{k} := k_{\to\mathrm{post}(\{\boldsymbol{x}\},\sigma^2)}$. Then,*

$$\tilde{k}(\boldsymbol{x}', \boldsymbol{x}'') = k(\boldsymbol{x}', \boldsymbol{x}'') - k(\boldsymbol{x}', \boldsymbol{x})(k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2\boldsymbol{I})^{-1}k(\boldsymbol{x}, \boldsymbol{x}'')$$

Consequently, if ϕ is a feature map for k, then

$$\tilde{\phi}(\boldsymbol{x}') := \left(\boldsymbol{I} - \frac{\phi(\boldsymbol{x})\phi(\boldsymbol{x})^\top}{\sigma^2+\phi(\boldsymbol{x})^\top\phi(\boldsymbol{x})}\right)^{1/2}\phi(\boldsymbol{x}') = \left(\boldsymbol{I} - \beta\phi(\boldsymbol{x})\phi(\boldsymbol{x})^\top\right)\phi(\boldsymbol{x}')$$

is a feature map for $\tilde{k}$, where

$$\beta := \frac{1}{\sqrt{\sigma^2+\phi(\boldsymbol{x})^\top\phi(\boldsymbol{x})}\left(\sqrt{\sigma^2+\phi(\boldsymbol{x})^\top\phi(\boldsymbol{x})}+\sigma\right)}~.$$

Proof. The kernel update equation follows directly from Eq. (6.8).

Step 1: Feature map. The specified feature map $\tilde{\phi}$ satisfies

$$\begin{aligned}\tilde{\phi}(\boldsymbol{x}')^\top\tilde{\phi}(\boldsymbol{x}'') &= \phi(\boldsymbol{x}')^\top\left(\boldsymbol{I} - \frac{\phi(\boldsymbol{x})\phi(\boldsymbol{x})^\top}{\sigma^2+\phi(\boldsymbol{x})^\top\phi(\boldsymbol{x})}\right)\phi(\boldsymbol{x}'') \\ &= k(\boldsymbol{x}', \boldsymbol{x}'') - \tilde{k}(\boldsymbol{x}', \boldsymbol{x})(\sigma^2 + k(\boldsymbol{x}, \boldsymbol{x}))^{-1}k(\boldsymbol{x}, \boldsymbol{x}'') \\ &= \tilde{k}(\boldsymbol{x}', \boldsymbol{x}'')~,\end{aligned}$$

hence it is a feature map for $\tilde{k}$.

Step 2: Square root. According to Lemma 6.D.1, we have

$$\begin{aligned}
&\left(\boldsymbol{I} - \frac{\phi(\boldsymbol{x})\phi(\boldsymbol{x})^\top}{\sigma^2 + \phi(\boldsymbol{x})^\top\phi(\boldsymbol{x})}\right)^{1/2} \\
&= \boldsymbol{I} - \frac{1}{(\sigma^2 + \phi(\boldsymbol{x})^\top\phi(\boldsymbol{x}))\left(1 + \sqrt{1 - \phi(\boldsymbol{x})^\top\phi(\boldsymbol{x})/(\sigma^2 + \phi(\boldsymbol{x})^\top\phi(\boldsymbol{x}))}\right)}\phi(\boldsymbol{x})\phi(\boldsymbol{x})^\top \\
&= \boldsymbol{I} - \frac{1}{(\sigma^2 + \phi(\boldsymbol{x})^\top\phi(\boldsymbol{x}))(1 + \sigma(\sigma^2 + \phi(\boldsymbol{x})^\top\phi(\boldsymbol{x}))^{-1/2})}\phi(\boldsymbol{x})\phi(\boldsymbol{x})^\top \\
&= \boldsymbol{I} - \frac{1}{\sqrt{\sigma^2 + \phi(\boldsymbol{x})^\top\phi(\boldsymbol{x})}(\sqrt{\sigma^2 + \phi(\boldsymbol{x})^\top\phi(\boldsymbol{x})} + \sigma)}\phi(\boldsymbol{x})\phi(\boldsymbol{x})^\top . \qquad \square
\end{aligned}$$

In our implementation, we keep track of the following quantities:

$$\begin{aligned}
\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}] &\coloneqq \phi_{\to\mathrm{post}(\mathcal{X},\sigma^2)}(\boldsymbol{x}) \\
c_{\mathcal{X}}[\boldsymbol{x}] &\coloneqq \boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top \boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}] .
\end{aligned}$$

Note that $\phi_{\to\mathrm{post}(\mathcal{X},\sigma^2)}$ is not uniquely defined, since any rotation of $\phi_{\to\mathrm{post}(\mathcal{X},\sigma^2)}$ leads to the same kernel. When computing $\boldsymbol{\Phi}$ as defined above, we do not care which version of $\phi_{\to\mathrm{post}(\mathcal{X},\sigma^2)}$ it corresponds to, as long as the same version of $\phi_{\to\mathrm{post}(\mathcal{X},\sigma^2)}$ is used for all $\boldsymbol{x}$.

Following Proposition 6.D.2, $\boldsymbol{\Phi}$ and c can be updated with a new observation $\boldsymbol{x} \notin \mathcal{X}$ as follows:

$$\begin{aligned}
\boldsymbol{\Phi}_{\mathcal{X}\cup\{\boldsymbol{x}\}}[\boldsymbol{x}'] &= \boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}'] - \beta_{\mathcal{X}}(\boldsymbol{x})\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\langle \boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}], \boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}'] \rangle \\
\boldsymbol{c}_{\mathcal{X}\cup\{\boldsymbol{x}\}}[\boldsymbol{x}'] &= \boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}']^\top \left(\boldsymbol{I} - \frac{\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top}{\sigma^2 + \boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]}\right)\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}'] \\
&= \boldsymbol{c}_{\mathcal{X}}[\boldsymbol{x}'] - \gamma_{\mathcal{X}}(\boldsymbol{x})^{-2}\langle \Phi_{\mathcal{X}}[\boldsymbol{x}], \Phi_{\mathcal{X}}[\boldsymbol{x}'] \rangle^2 ,
\end{aligned}$$

where

$$\begin{aligned}
\gamma_{\mathcal{X}}(\boldsymbol{x}) &\coloneqq \sqrt{\sigma^2 + \boldsymbol{c}_{\mathcal{X}}[\boldsymbol{x}]} \\
\beta_{\mathcal{X}}(\boldsymbol{x}) &\coloneqq \frac{1}{\gamma_{\mathcal{X}}(\boldsymbol{x})(\gamma_{\mathcal{X}}(\boldsymbol{x}) + \sigma)} .
\end{aligned}$$

Together, these considerations lead to a feature-space implementation of MaxDet, presented in Algorithm 7.

For the complexity analysis of Algorithm 7, we exclude the computation of the feature matrix $\phi(\mathcal{X}_{\mathrm{cand}})$. As for the kernel-space version of MaxDet, the runtime is then dominated by the runtime of Add, which has a runtime complexity of $\mathcal{O}(N_{\mathrm{cand}}d_{\mathrm{feat}})$. Since it is called N_{sel} times, the total runtime complexity of Algorithm 7 is $\mathcal{O}(N_{\mathrm{cand}}N_{\mathrm{sel}}d_{\mathrm{feat}})$. If the kernel k is evaluated by an inner product of the pre-computed features, the runtime of a kernel evaluation scales as $T_k = \Theta(d_{\mathrm{feat}})$. In this case, the runtime of the kernel-space version in Algorithm 6 has a runtime complexity of $\mathcal{O}(N_{\mathrm{cand}}N_{\mathrm{sel}}(d_{\mathrm{feat}} + N_{\mathrm{sel}}))$, which is not asymptotically better than the one for Algorithm 7. However, in our implementation, we observe that the kernel-space version typically runs faster for $N_{\mathrm{sel}} \lesssim 3d_{\mathrm{feat}}$, which can be attributed to the smaller constant in the runtime of Add. It is easily verified that the memory complexity of Algorithm 7 scales as $\mathcal{O}(N_{\mathrm{cand}}d_{\mathrm{feat}})$.

Algorithm 7 MaxDet pseudocode implementation in feature space for given $\sigma^2 \geq 0$ using Algorithm 4.

```
function Init
    $\boldsymbol{\Phi} \leftarrow \phi(\mathcal{X}_{\mathrm{cand}}) \in \mathbb{R}^{N_{\mathrm{cand}} \times d_{\mathrm{feat}}}$                    ▷ feature matrix
    $\boldsymbol{c} \leftarrow (\langle \boldsymbol{\Phi}[\boldsymbol{x}, \cdot]^\top, \boldsymbol{\Phi}[\boldsymbol{x}, \cdot]^\top \rangle)_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{cand}}}$     ▷ vector containing the kernel diagonal
end function

function Add($\boldsymbol{x}$)
    $\gamma \leftarrow \sqrt{\sigma^2 + \boldsymbol{c}[\boldsymbol{x}]}$
    $\beta \leftarrow (\gamma(\gamma + \sigma))^{-1}$
    $\boldsymbol{u} \leftarrow \boldsymbol{\Phi}\boldsymbol{\Phi}[\boldsymbol{x}, \cdot]^\top$
    $\boldsymbol{c} \leftarrow \boldsymbol{c} - \gamma^{-2}(\boldsymbol{u} \odot \boldsymbol{u})$
    $\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi} - \beta \boldsymbol{u} \boldsymbol{\Phi}[\boldsymbol{x}, \cdot]$
end function

function Next
    return $\operatorname{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{pool}} \setminus \mathcal{X}_{\mathrm{batch}}} \boldsymbol{c}[\boldsymbol{x}]$
end function
```

6.D.5 Bait

In this section, we write $\mathcal{X}_{\mathrm{tp}} \coloneqq \mathcal{X}_{\mathrm{train}} \cup \mathcal{X}_{\mathrm{pool}}$.

Connection Between Kernel and Feature Map Formulations

We can rewrite Bait's acquisition function from Eq. (6.23) as

$$\begin{aligned} a(\mathcal{X}) &\coloneqq \sum_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\mathrm{tp}}} k_{\to \mathrm{post}(\mathcal{X}, \sigma^2)}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}) \overset{\text{Eq. (6.10)}}{=} \sigma^2 \sum_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\mathrm{tp}}} \phi(\tilde{\boldsymbol{x}})^\top (\phi(\mathcal{X})^\top \phi(\mathcal{X}) + \sigma^2 \boldsymbol{I})^{-1} \phi(\tilde{\boldsymbol{x}}) \\ &= \sigma^2 \sum_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\mathrm{tp}}} \operatorname{tr}\left(\phi(\tilde{\boldsymbol{x}})^\top (\phi(\mathcal{X})^\top \phi(\mathcal{X}) + \sigma^2 \boldsymbol{I})^{-1} \phi(\tilde{\boldsymbol{x}})\right) \\ &= \sigma^2 \sum_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\mathrm{tp}}} \operatorname{tr}\left((\phi(\mathcal{X})^\top \phi(\mathcal{X}) + \sigma^2 \boldsymbol{I})^{-1} \phi(\tilde{\boldsymbol{x}}) \phi(\tilde{\boldsymbol{x}})^\top\right) \\ &= \sigma^2 \operatorname{tr}\left((\phi(\mathcal{X})^\top \phi(\mathcal{X}) + \sigma^2 \boldsymbol{I})^{-1} \phi(\mathcal{X}_{\mathrm{tp}})^\top \phi(\mathcal{X}_{\mathrm{tp}})\right) \\ &= \sigma^2 \operatorname{tr}\left((\boldsymbol{A}_{\mathcal{X}} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{A}_{\mathcal{X}_{\mathrm{tp}}}\right) , \end{aligned}$$

where

$$\boldsymbol{A}_{\mathcal{X}} \coloneqq \phi(\mathcal{X})^\top \phi(\mathcal{X}) .$$

The latter trace-based formulation corresponds to the formulation by Ash et al. (2021).

Forward Version

We will first derive an efficient implementation of Bait-F in feature space, which builds on our derivation of MaxDet in feature space and serves as a basis for Bait-FB in feature space. As for the feature space version of MaxDet, we choose to update the features using square roots of rank-1 updates. We will not derive a kernel space version of Bait here since it appears that a kernel space version would not scale to large data sets. In the following, we will assume $\sigma^2 > 0$.

In a single iteration of the Bait-F selection method, we want to find $\boldsymbol{x} \in \mathcal{X}_{\mathrm{pool}} \setminus \mathcal{X}_{\mathrm{batch}}$ *maximizing*

$$\begin{aligned} & a(\mathcal{X}) - a(\mathcal{X} \cup \{\boldsymbol{x}\}) \\ \overset{\text{Proposition 6.D.2}}{=} & \sum_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\mathrm{tp}}} \frac{k_{\to\mathrm{post}(\mathcal{X},\sigma^2)}(\tilde{\boldsymbol{x}}, \boldsymbol{x})^2}{k_{\to\mathrm{post}(\mathcal{X},\sigma^2)}(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2} \\ = & \frac{\phi_{\to\mathrm{post}(\mathcal{X},\sigma^2)}(\boldsymbol{x})^\top \phi_{\to\mathrm{post}(\mathcal{X},\sigma^2)}(\mathcal{X}_{\mathrm{tp}})^\top \phi_{\to\mathrm{post}(\mathcal{X},\sigma^2)}(\mathcal{X}_{\mathrm{tp}}) \phi_{\to\mathrm{post}(\mathcal{X},\sigma^2)}(\boldsymbol{x})}{\phi_{\to\mathrm{post}(\mathcal{X},\sigma^2)}(\boldsymbol{x})^\top \phi_{\to\mathrm{post}(\mathcal{X},\sigma^2)}(\boldsymbol{x}) + \sigma^2}~. \end{aligned}$$

While the inner product in the denominator of the last expression corresponds to the quantity $\boldsymbol{c}_{\mathcal{X}}[\boldsymbol{x}]$ from the MaxDet feature space implementation, we still need a way to efficiently compute the numerator. Therefore, in addition to the quantities $\boldsymbol{\Phi}_{\mathcal{X}}$ and $\boldsymbol{c}_{\mathcal{X}}$ tracked in the feature-space implementation of MaxDet, we track the following quantities for Bait:

$$\begin{aligned} \boldsymbol{\Sigma}_{\mathcal{X}} &\coloneqq \phi_{\to\mathrm{post}(\mathcal{X},\sigma^2)}(\mathcal{X}_{\mathrm{tp}})^\top \phi_{\to\mathrm{post}(\mathcal{X},\sigma^2)}(\mathcal{X}_{\mathrm{tp}}) \\ \boldsymbol{v}_{\mathcal{X}}[\boldsymbol{x}] &\coloneqq \Phi_{\mathcal{X}}[\boldsymbol{x}]^\top \boldsymbol{\Sigma}_{\mathcal{X}} \Phi_{\mathcal{X}}[\boldsymbol{x}]~. \end{aligned}$$

Using these quantities, we can write

$$a(\mathcal{X}) - a(\mathcal{X} \cup \{\boldsymbol{x}\}) = \frac{\boldsymbol{v}_{\mathcal{X}}[\boldsymbol{x}]}{\boldsymbol{c}_{\mathcal{X}}[\boldsymbol{x}] + \sigma^2}~.$$

Following Proposition 6.D.2, we obtain the following update equations:

$$\begin{aligned} \boldsymbol{\Sigma}_{\mathcal{X}\cup\{\boldsymbol{x}\}} &= (\boldsymbol{I} - \beta_{\mathcal{X}}(\boldsymbol{x})\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top)\boldsymbol{\Sigma}_{\mathcal{X}}(\boldsymbol{I} - \beta_{\mathcal{X}}(\boldsymbol{x})\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top) \\ &= \boldsymbol{\Sigma}_{\mathcal{X}} - \beta_{\mathcal{X}}(\boldsymbol{x})\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top\boldsymbol{\Sigma}_{\mathcal{X}} - \beta_{\mathcal{X}}(\boldsymbol{x})\boldsymbol{\Sigma}_{\mathcal{X}}\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top \\ &\quad + \beta_{\mathcal{X}}(\boldsymbol{x})^2\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top\boldsymbol{\Sigma}_{\mathcal{X}}\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top \\ \boldsymbol{v}_{\mathcal{X}\cup\{\boldsymbol{x}\}}[\boldsymbol{x}'] &= \boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}']^\top(\boldsymbol{I} - \gamma_{\mathcal{X}}^{-2}(\boldsymbol{x})\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top)\boldsymbol{\Sigma}_{\mathcal{X}}(\boldsymbol{I} - \gamma_{\mathcal{X}}(\boldsymbol{x})^{-2}\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top)\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}'] \\ &= \boldsymbol{v}_{\mathcal{X}}[\boldsymbol{x}'] - 2\gamma_{\mathcal{X}}(\boldsymbol{x})^{-2}\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}']^\top\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top\boldsymbol{\Sigma}_{\mathcal{X}}\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}'] \\ &\quad + \gamma_{\mathcal{X}}(\boldsymbol{x})^{-4}\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}']^\top\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top\boldsymbol{\Sigma}_{\mathcal{X}}\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}']~. \end{aligned}$$

This leads to the pseudocode in Algorithm 8. For the runtime analysis, we neglect the time for evaluating ϕ as usual. Then, the runtime of Init is $\mathcal{O}((N_{\mathrm{train}} + N_{\mathrm{pool}})d_{\mathrm{feat}}^2)$, the runtime of Add is $\mathcal{O}((N_{\mathrm{cand}} + d_{\mathrm{feat}})d_{\mathrm{feat}})$ and the runtime of Next is $\mathcal{O}(N_{\mathrm{pool}})$. Hence, the overall runtime of Bait-F is $\mathcal{O}(N_{\mathrm{cand}}N_{\mathrm{sel}}d_{\mathrm{feat}} + (N_{\mathrm{train}} + N_{\mathrm{pool}})d_{\mathrm{feat}}^2)$. The memory complexity is $\mathcal{O}((N_{\mathrm{cand}} + d_{\mathrm{feat}})d_{\mathrm{feat}})$.

Forward-Backward Version

To fit Bait-FB into our framework, we first extend our iterative selection template to include a backward selection step. The extended template is shown in Algorithm 9. Here,

Algorithm 8 BAIT-F pseudocode implementation in feature space for given $\sigma^2 > 0$ using Algorithm 4.

```
function INIT
    $\boldsymbol{\Phi} \leftarrow \phi(\mathcal{X}_{\text{cand}}) \in \mathbb{R}^{N_{\text{cand}} \times d_{\text{feat}}}$    ▷ feature matrix
    $\boldsymbol{c} \leftarrow (\langle \boldsymbol{\Phi}[\boldsymbol{x}, \cdot]^\top, \boldsymbol{\Phi}[\boldsymbol{x}, \cdot]^\top \rangle)_{\boldsymbol{x} \in \mathcal{X}_{\text{cand}}}$    ▷ vector containing the kernel diagonal
    $\boldsymbol{\Sigma} \leftarrow \phi(\mathcal{X}_{\text{tp}})^\top \phi(\mathcal{X}_{\text{tp}})$    ▷ Train and pool second moment matrix
    $\boldsymbol{v} \leftarrow (\langle \boldsymbol{\Phi}[\boldsymbol{x}, \cdot]^\top, \boldsymbol{\Sigma}\boldsymbol{\Phi}[\boldsymbol{x}, \cdot]^\top \rangle)_{\boldsymbol{x} \in \mathcal{X}_{\text{cand}}}$    ▷ Numerator of the acquisition function
end function

function ADD($\boldsymbol{x}$)
    $\gamma \leftarrow \sqrt{\sigma^2 + \boldsymbol{c}[\boldsymbol{x}]}$
    $\beta \leftarrow (\gamma(\gamma + \sigma))^{-1}$
    $\boldsymbol{u} \leftarrow \boldsymbol{\Phi}\boldsymbol{\Phi}[\boldsymbol{x}, \cdot]^\top$
    $\tilde{\boldsymbol{u}} \leftarrow \boldsymbol{u} \odot \boldsymbol{u}$
    $\boldsymbol{w} \leftarrow \boldsymbol{\Sigma}\boldsymbol{\Phi}[\boldsymbol{x}, \cdot]^\top$
    $\tilde{v} \leftarrow \boldsymbol{v}[\boldsymbol{x}]$
    $\boldsymbol{v} \leftarrow \boldsymbol{v} - 2\gamma^{-2}(\boldsymbol{\Phi}\boldsymbol{w}) \odot \boldsymbol{u} + \gamma^{-4}\tilde{v}\tilde{\boldsymbol{u}}$
    $\boldsymbol{A} \leftarrow \boldsymbol{w}\Phi[\boldsymbol{x}, \cdot]$
    $\boldsymbol{\Sigma} \leftarrow \boldsymbol{\Sigma} - \beta(\boldsymbol{A} + \boldsymbol{A}^\top) + \beta^2\tilde{v}\Phi[\boldsymbol{x}, \cdot]^\top\Phi[\boldsymbol{x}, \cdot]$
    $\boldsymbol{c} \leftarrow \boldsymbol{c} - \gamma^{-2}\tilde{\boldsymbol{u}}$
    $\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi} - \beta\boldsymbol{u}\boldsymbol{\Phi}[\boldsymbol{x}, \cdot]$
end function

function NEXT
    return $\operatorname{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\text{pool}} \setminus \mathcal{X}_{\text{batch}}} \frac{\boldsymbol{v}[\boldsymbol{x}]}{\sigma^2 + \boldsymbol{c}[\boldsymbol{x}]}$
end function
```

we have an additional parameter N_{extra} specifying how many additional batch elements will be selected in the forward step and then removed in the backward step. Following Ash et al. (2021), we set $N_{\text{extra}} \coloneqq \min\{N_{\text{batch}}, N_{\text{pool}} - N_{\text{batch}}\}$ in our experiments.

To implement BAIT-FB within Algorithm 9, we can reuse the methods INIT, ADD and NEXT from BAIT-F. In addition, we need to implement the methods REMOVE and NEXTBACKWARD for the backward step. To this end, the following proposition shows how the kernel and feature map update when *removing* a point $\boldsymbol{x}$ from the set $\mathcal{X}$ of observed points:

Proposition 6.D.3 (Backward update)**.** *For a kernel k and $\sigma^2 > 0$, let $\tilde{k} \coloneqq k_{\to\text{post}(\{\boldsymbol{x}\},\sigma^2)}$. Then,*

$$k(\boldsymbol{x}', \boldsymbol{x}'') = \tilde{k}(\boldsymbol{x}', \boldsymbol{x}'') + \tilde{k}(\boldsymbol{x}', \boldsymbol{x})(\sigma^2 - \tilde{k}(\boldsymbol{x}, \boldsymbol{x}))^{-1}\tilde{k}(\boldsymbol{x}, \boldsymbol{x}'') \ .$$

Consequently, if $\tilde{\phi}$ is a feature map for $\tilde{k}$, then

$$\phi(\boldsymbol{x}') \coloneqq \left(\boldsymbol{I} + \frac{\tilde{\phi}(\boldsymbol{x})\tilde{\phi}(\boldsymbol{x})^\top}{\sigma^2 - \tilde{\phi}(\boldsymbol{x})^\top\tilde{\phi}(\boldsymbol{x})}\right)^{1/2} \tilde{\phi}(\boldsymbol{x}') = \left(\boldsymbol{I} + \tilde{\beta}\tilde{\phi}(\boldsymbol{x})\tilde{\phi}(\boldsymbol{x})^\top\right) \tilde{\phi}(\boldsymbol{x}')$$

Algorithm 9 Forward-backward selection algorithm template involving five customizable functions INIT, ADD, NEXT, REMOVE, NEXTBACKWARD that are allowed to have side effects (i.e., read/write variables in SELECT).

```
function SELECT($k$, $\mathcal{X}_{\text{train}}$, $\mathcal{X}_{\text{pool}}$, $N_{\text{batch}}$, $N_{\text{extra}}$, mode $\in \{\text{P}, \text{TP}\}$)
    $\mathcal{X}_{\text{mode}} \leftarrow \mathcal{X}_{\text{train}}$ if mode = TP else $\emptyset$
    $\mathcal{X}_{\text{cand}} \coloneqq \mathcal{X}_{\text{mode}} \cup \mathcal{X}_{\text{pool}}$
    $\mathcal{X}_{\text{batch}} \leftarrow \emptyset$
    INIT
    for $\boldsymbol{x}$ in $\mathcal{X}_{\text{mode}}$ do
        ADD($\boldsymbol{x}$)
    end for
    for $i$ from 1 to $N_{\text{batch}} + N_{\text{extra}}$ do
        $\boldsymbol{x} \leftarrow$NEXT
        if $\boldsymbol{x} \in \mathcal{X}_{\text{batch}} \cup \mathcal{X}_{\text{train}}$ (failed selection) then
            ensure $|\mathcal{X}_{\text{batch}}| = N_{\text{batch}}$ by removing the latest samples or filling up with random samples
            return $\mathcal{X}_{\text{batch}}$
        end if
        $\mathcal{X}_{\text{batch}} \leftarrow \mathcal{X}_{\text{batch}} \cup \{\boldsymbol{x}\}$
        ADD($\boldsymbol{x}$)
    end for
    for $i$ from 1 to $N_{\text{extra}}$ do
        $\boldsymbol{x} \leftarrow$NEXTBACKWARD
        if $\boldsymbol{x} \notin \mathcal{X}_{\text{batch}} \cup \mathcal{X}_{\text{train}}$ (failed selection) then
            ensure $|\mathcal{X}_{\text{batch}}| = N_{\text{batch}}$ by removing the latest samples
            return $\mathcal{X}_{\text{batch}}$
        end if
        $\mathcal{X}_{\text{batch}} \leftarrow \mathcal{X}_{\text{batch}} \setminus \{\boldsymbol{x}\}$
        REMOVE($\boldsymbol{x}$)
    end for
    return $\mathcal{X}_{\text{batch}}$
end function
```

is a feature map for k, where

$$\tilde{\beta} := \frac{1}{\sqrt{\sigma^2 - \tilde{\phi}(\boldsymbol{x})^\top \tilde{\phi}(\boldsymbol{x})}\left(\sqrt{\sigma^2 - \tilde{\phi}(\boldsymbol{x})^\top \tilde{\phi}(\boldsymbol{x})} + \sigma\right)} .$$

Proof. **Step 1: Finding $k(\boldsymbol{x}, \boldsymbol{x})$.** We have

$$\tilde{k}(\boldsymbol{x}, \boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) - \frac{k(\boldsymbol{x}, \boldsymbol{x})^2}{k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2} = \frac{k(\boldsymbol{x}, \boldsymbol{x})\sigma^2}{k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2} < \sigma^2 .$$

Hence,

$$\sigma^2 - \tilde{k}(\boldsymbol{x}, \boldsymbol{x}) = \frac{\sigma^2(k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2)}{k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2} - \frac{k(\boldsymbol{x}, \boldsymbol{x})\sigma^2}{k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2} = \frac{\sigma^4}{k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2} ,$$

which yields

$$\frac{\sigma^2}{\sigma^2 - \tilde{k}(\boldsymbol{x}, \boldsymbol{x})} = \frac{k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2}{\sigma^2} .$$

Step 2: Finding $k(\boldsymbol{x}', \boldsymbol{x})$. Now, we compute

$$\tilde{k}(\boldsymbol{x}', \boldsymbol{x}) = k(\boldsymbol{x}', \boldsymbol{x}) - k(\boldsymbol{x}', \boldsymbol{x})\frac{k(\boldsymbol{x}, \boldsymbol{x})}{k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2} = k(\boldsymbol{x}', \boldsymbol{x})\frac{\sigma^2}{k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2} ,$$

which yields

$$k(\boldsymbol{x}', \boldsymbol{x}) = \tilde{k}(\boldsymbol{x}', \boldsymbol{x})\frac{k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2}{\sigma^2} = \tilde{k}(\boldsymbol{x}', \boldsymbol{x})\frac{\sigma^2}{\sigma^2 - \tilde{k}(\boldsymbol{x}, \boldsymbol{x})} .$$

Step 3: Finding $k(\boldsymbol{x}', \boldsymbol{x}'')$. Finally, we have

$$\tilde{k}(\boldsymbol{x}', \boldsymbol{x}'') = k(\boldsymbol{x}', \boldsymbol{x}'') - \frac{k(\boldsymbol{x}', \boldsymbol{x})k(\boldsymbol{x}, \boldsymbol{x}'')}{k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2} ,$$

which yields

$$\begin{aligned}
k(\boldsymbol{x}', \boldsymbol{x}'') &= \tilde{k}(\boldsymbol{x}', \boldsymbol{x}'') + k(\boldsymbol{x}', \boldsymbol{x}) \cdot \frac{1}{k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2} \cdot k(\boldsymbol{x}, \boldsymbol{x}'') \\
&= \tilde{k}(\boldsymbol{x}', \boldsymbol{x}'') + \tilde{k}(\boldsymbol{x}', \boldsymbol{x})\frac{k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2}{\sigma^2} \cdot \frac{1}{k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2} \cdot \tilde{k}(\boldsymbol{x}, \boldsymbol{x}'')\frac{\sigma^2}{\sigma^2 - \tilde{k}(\boldsymbol{x}, \boldsymbol{x})} \\
&= \tilde{k}(\boldsymbol{x}', \boldsymbol{x}'') + \frac{\tilde{k}(\boldsymbol{x}', \boldsymbol{x})\tilde{k}(\boldsymbol{x}, \boldsymbol{x}'')}{\sigma^2 - \tilde{k}(\boldsymbol{x}, \boldsymbol{x})} .
\end{aligned}$$

Step 4: Feature map. The specified feature map ϕ satisfies

$$\begin{aligned}
\phi(\boldsymbol{x}')^\top \phi(\boldsymbol{x}'') &= \tilde{\phi}(\boldsymbol{x}')^\top \left(\boldsymbol{I} + \frac{\tilde{\phi}(\boldsymbol{x})\tilde{\phi}(\boldsymbol{x})^\top}{\sigma^2 - \tilde{\phi}(\boldsymbol{x})^\top \tilde{\phi}(\boldsymbol{x})}\right) \tilde{\phi}(\boldsymbol{x}'') \\
&= \tilde{k}(\boldsymbol{x}', \boldsymbol{x}'') + \tilde{k}(\boldsymbol{x}', \boldsymbol{x})(\sigma^2 - \tilde{k}(\boldsymbol{x}, \boldsymbol{x}))^{-1}\tilde{k}(\boldsymbol{x}, \boldsymbol{x}'') \\
&= k(\boldsymbol{x}', \boldsymbol{x}'') ,
\end{aligned}$$

hence it is a feature map for k.

Step 5: Square root. According to Lemma 6.D.1, we have

$$\begin{aligned}
&\left(\boldsymbol{I} + \frac{\tilde{\phi}(\boldsymbol{x})\tilde{\phi}(\boldsymbol{x})^\top}{\sigma^2 - \tilde{\phi}(\boldsymbol{x})^\top\tilde{\phi}(\boldsymbol{x})}\right)^{1/2} \\
&= \boldsymbol{I} + \frac{1}{(\sigma^2 - \tilde{\phi}(\boldsymbol{x})^\top\tilde{\phi}(\boldsymbol{x}))\left(1 + \sqrt{1 + \tilde{\phi}(\boldsymbol{x})^\top\tilde{\phi}(\boldsymbol{x})/(\sigma^2 - \tilde{\phi}(\boldsymbol{x})^\top\tilde{\phi}(\boldsymbol{x}))}\right)}\tilde{\phi}(\boldsymbol{x})\tilde{\phi}(\boldsymbol{x})^\top \\
&= \boldsymbol{I} + \frac{1}{(\sigma^2 - \tilde{\phi}(\boldsymbol{x})^\top\tilde{\phi}(\boldsymbol{x}))(1 + \sigma(\sigma^2 - \tilde{\phi}(\boldsymbol{x})^\top\tilde{\phi}(\boldsymbol{x}))^{-1/2})}\tilde{\phi}(\boldsymbol{x})\tilde{\phi}(\boldsymbol{x})^\top \\
&= \boldsymbol{I} + \frac{1}{\sqrt{\sigma^2 - \tilde{\phi}(\boldsymbol{x})^\top\tilde{\phi}(\boldsymbol{x})}(\sqrt{\sigma^2 - \tilde{\phi}(\boldsymbol{x})^\top\tilde{\phi}(\boldsymbol{x})} + \sigma)}\tilde{\phi}(\boldsymbol{x})\tilde{\phi}(\boldsymbol{x})^\top . \qquad \square
\end{aligned}$$

To formulate the backward update for our tracked quantities $\boldsymbol{\Phi}, \boldsymbol{c}, \boldsymbol{\Sigma}$ and $\boldsymbol{v}$, we define for $\boldsymbol{x} \in \mathcal{X}$ the scalars

$$\begin{aligned}
\tilde{\gamma}_{\mathcal{X}}(\boldsymbol{x}) &:= \sqrt{\sigma^2 - \boldsymbol{c}[\boldsymbol{x}]} \\
\tilde{\beta}_{\mathcal{X}}(\boldsymbol{x}) &:= \frac{1}{\gamma(\gamma + \sigma)} .
\end{aligned}$$

Then, using Proposition 6.D.3, we can compute the backwards update as

$$\begin{aligned}
\boldsymbol{\Phi}_{\mathcal{X}\setminus\{\boldsymbol{x}\}}[\boldsymbol{x}'] &= \boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}'] + \tilde{\beta}_{\mathcal{X}}(\boldsymbol{x})\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\langle\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}], \boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}']\rangle \\
\boldsymbol{c}_{\mathcal{X}\setminus\{\boldsymbol{x}\}}[\boldsymbol{x}'] &= \boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}']^\top\left(\boldsymbol{I} + \frac{\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top}{\sigma^2 - \boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]}\right)\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}'] \\
&= \boldsymbol{c}_{\mathcal{X}}[\boldsymbol{x}'] + \tilde{\gamma}_{\mathcal{X}}(\boldsymbol{x})^{-2}\langle\Phi_{\mathcal{X}}[\boldsymbol{x}], \Phi_{\mathcal{X}}[\boldsymbol{x}']\rangle^2 \\
\boldsymbol{\Sigma}_{\mathcal{X}\setminus\{\boldsymbol{x}\}} &= (\boldsymbol{I} - \beta_{\mathcal{X}}(\boldsymbol{x})\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top)\boldsymbol{\Sigma}_{\mathcal{X}}(\boldsymbol{I} - \beta_{\mathcal{X}}(\boldsymbol{x})\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top) \\
&= \boldsymbol{\Sigma}_{\mathcal{X}} + \tilde{\beta}_{\mathcal{X}}(\boldsymbol{x})\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top\boldsymbol{\Sigma}_{\mathcal{X}} + \tilde{\beta}_{\mathcal{X}}(\boldsymbol{x})\boldsymbol{\Sigma}_{\mathcal{X}}\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top \\
&\quad + \beta_{\mathcal{X}}(\boldsymbol{x})^2\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top\boldsymbol{\Sigma}_{\mathcal{X}}\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top \\
\boldsymbol{v}_{\mathcal{X}\setminus\{\boldsymbol{x}\}}[\boldsymbol{x}'] &= \boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}']^\top(\boldsymbol{I} + \tilde{\gamma}_{\mathcal{X}}^{-2}(\boldsymbol{x})\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top)\boldsymbol{\Sigma}_{\mathcal{X}}(\boldsymbol{I} + \tilde{\gamma}_{\mathcal{X}}(\boldsymbol{x})^{-2}\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top)\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}'] \\
&= \boldsymbol{v}_{\mathcal{X}}[\boldsymbol{x}'] + 2\tilde{\gamma}_{\mathcal{X}}(\boldsymbol{x})^{-2}\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}']^\top\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top\boldsymbol{\Sigma}_{\mathcal{X}}\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}'] \\
&\quad + \tilde{\gamma}_{\mathcal{X}}(\boldsymbol{x})^{-4}\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}']^\top\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top\boldsymbol{\Sigma}_{\mathcal{X}}\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}]^\top\boldsymbol{\Phi}_{\mathcal{X}}[\boldsymbol{x}'] .
\end{aligned}$$

For the backward step, we want to find $\boldsymbol{x} \in \mathcal{X}$ *minimizing*

$$\begin{aligned}
&a(\mathcal{X} \setminus \{\boldsymbol{x}\}) - a(\mathcal{X}) \\
\overset{\text{Proposition 6.D.3}}{=} &\sum_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\text{tp}}} \frac{k_{\to\text{post}(\mathcal{X},\sigma^2)}(\tilde{\boldsymbol{x}}, \boldsymbol{x})^2}{\sigma^2 - k_{\to\text{post}(\mathcal{X},\sigma^2)}(\boldsymbol{x}, \boldsymbol{x})} \\
= &\frac{\phi_{\to\text{post}(\mathcal{X},\sigma^2)}(\boldsymbol{x})^\top\phi_{\to\text{post}(\mathcal{X},\sigma^2)}(\mathcal{X}_{\text{tp}})^\top\phi_{\to\text{post}(\mathcal{X},\sigma^2)}(\mathcal{X}_{\text{tp}})\phi_{\to\text{post}(\mathcal{X},\sigma^2)}(\boldsymbol{x})}{\sigma^2 - \phi_{\to\text{post}(\mathcal{X},\sigma^2)}(\boldsymbol{x})^\top\phi_{\to\text{post}(\mathcal{X},\sigma^2)}(\boldsymbol{x})} \\
= &\frac{\boldsymbol{v}_{\mathcal{X}}[\boldsymbol{x}]}{\sigma^2 - \boldsymbol{c}_{\mathcal{X}}[\boldsymbol{x}]} .
\end{aligned}$$

The corresponding implementation of Remove and NextBackward is given in Algorithm 10, completing the implementation of Bait-FB. The runtimes of Remove

Algorithm 10 Functions REMOVE and NEXTBACKWARD that, together with Algorithm 8 and Algorithm 9, yield a pseudocode implementation of BAIT-FB in feature space for given $\sigma^2 > 0$.

```
function REMOVE($\boldsymbol{x}$)
    $\tilde{\gamma} \leftarrow \sqrt{\sigma^2 - \boldsymbol{c}[\boldsymbol{x}]}$
    $\tilde{\beta} \leftarrow (\gamma(\gamma + \sigma))^{-1}$
    $\boldsymbol{u} \leftarrow \boldsymbol{\Phi}\boldsymbol{\Phi}[\boldsymbol{x}, \cdot]^\top$
    $\tilde{\boldsymbol{u}} \leftarrow \boldsymbol{u} \odot \boldsymbol{u}$
    $\boldsymbol{w} \leftarrow \boldsymbol{\Sigma}\boldsymbol{\Phi}[\boldsymbol{x}, \cdot]^\top$
    $\tilde{v} \leftarrow \boldsymbol{v}[\boldsymbol{x}]$
    $\boldsymbol{v} \leftarrow \boldsymbol{v} + 2\tilde{\gamma}^{-2}(\boldsymbol{\Phi}\boldsymbol{w}) \odot \boldsymbol{u} + \tilde{\gamma}^{-4}\tilde{v}\tilde{\boldsymbol{u}}$
    $\boldsymbol{A} \leftarrow \boldsymbol{w}\Phi[\boldsymbol{x}, \cdot]$
    $\boldsymbol{\Sigma} \leftarrow \boldsymbol{\Sigma} + \tilde{\beta}(\boldsymbol{A} + \boldsymbol{A}^\top) + \tilde{\beta}^2\tilde{v}\Phi[\boldsymbol{x}, \cdot]^\top\Phi[\boldsymbol{x}, \cdot]$
    $\boldsymbol{c} \leftarrow \boldsymbol{c} + \tilde{\gamma}^{-2}\tilde{\boldsymbol{u}}$
    $\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi} + \tilde{\beta}\boldsymbol{u}\boldsymbol{\Phi}[\boldsymbol{x}, \cdot]$
end function

function NEXTBACKWARD
    return $\arg\min_{\boldsymbol{x} \in \mathcal{X}_{\text{batch}}} \frac{\boldsymbol{v}[\boldsymbol{x}]}{\sigma^2 - \boldsymbol{c}[\boldsymbol{x}]}$
end function
```

and NEXTBACKWARD are equivalent to those of ADD and NEXT, respectively, since the implementation is almost identical. Hence, the runtime complexity of BAIT-FB is given by $\mathcal{O}(N_{\text{cand}}(N_{\text{sel}} + 2N_{\text{extra}})d_{\text{feat}} + (N_{\text{train}} + N_{\text{pool}})d_{\text{feat}}^2)$. The memory complexity is again $\mathcal{O}((N_{\text{cand}} + d_{\text{feat}})d_{\text{feat}})$.

6.D.6 FRANKWOLFE

Pinsler et al. (2019) proposed to apply the Frank-Wolfe constrained optimization algorithm (Frank and Wolfe, 1956) to the problem of sparsely approximating the empirical kernel mean embedding of $\mathcal{X}_{\text{pool}}$ with non-negative weights. Like MAXDET, the resulting FRANKWOLFE method allows for a kernel-space and a feature-space implementation. Pinsler et al. (2019) used both versions in their experiments and presented the kernel-space version as pseudocode. Algorithm 11 is an optimized adaptation of their kernel-space version to our framework. A difference between our version and theirs is that in our version, NEXT does not allow choosing a previously selected point. Hence, our version prevents the possibility of generating smaller batches by selecting the same point multiple times. Moreover, our version reduces the runtime complexity from $\mathcal{O}(N_{\text{cand}}^2(T_k + N_{\text{sel}}))$ to $\mathcal{O}(N_{\text{cand}}^2(T_k + 1))$ by reusing previously computed quantities in ADD. The memory complexity of Algorithm 11 is $\mathcal{O}(N_{\text{cand}}^2)$, which can be reduced to $\mathcal{O}(N_{\text{cand}})$ by not storing the kernel matrix $\boldsymbol{K}$, at the cost of having to recompute some kernel values in ADD.

The quadratic complexity in N_{cand} for the kernel-space version of FRANKWOLFE shown in Algorithm 11 makes it infeasible for large N_{pool}, such as in our experiments. However, FRANKWOLFE can be realized much more efficiently in moderate-dimensional feature spaces, as shown in Algorithm 12 and implemented in our code and (less efficiently)

Algorithm 11 FrankWolfe pseudocode implementation in kernel space using Algorithm 4, following Pinsler et al. (2019).

```
function Init
    $\boldsymbol{K} \leftarrow (k(\boldsymbol{x}, \tilde{\boldsymbol{x}}))_{\boldsymbol{x},\tilde{\boldsymbol{x}} \in \mathcal{X}_{\mathrm{cand}}}$        ▷ Corresponds to $\langle \mathcal{L}_n, \mathcal{L}_n \rangle$
    $\boldsymbol{c} \leftarrow (\sqrt{\boldsymbol{K}[\boldsymbol{x}, \boldsymbol{x}]})_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{cand}}}$        ▷ Corresponds to $\sigma_n$
    $r \leftarrow \sum_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{cand}}} \boldsymbol{c}[\boldsymbol{x}]$        ▷ Corresponds to $\sigma$
    $\boldsymbol{u} \leftarrow (\sum_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\mathrm{cand}}} \boldsymbol{K}[\boldsymbol{x}, \tilde{\boldsymbol{x}}])_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{cand}}}$        ▷ Corresponds to $\langle \mathcal{L}, \mathcal{L}_n \rangle$
    $\boldsymbol{v} \leftarrow (0)_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{cand}}}$        ▷ Corresponds to $\langle \mathcal{L}(\boldsymbol{w}), \mathcal{L}_n \rangle$
    $s \leftarrow 0$        ▷ Corresponds to $\langle \mathcal{L}(\boldsymbol{w}), \mathcal{L}(\boldsymbol{w}) \rangle$
    $t \leftarrow 0$        ▷ Corresponds to $\langle \mathcal{L}(\boldsymbol{w}), \mathcal{L} \rangle$
end function

function Add($\boldsymbol{x}$)
    $\gamma \leftarrow \frac{r\boldsymbol{c}[\boldsymbol{x}]^{-1}(\boldsymbol{u}[\boldsymbol{x}] - \boldsymbol{v}[\boldsymbol{x}]) + s - t}{r^2 - 2r\boldsymbol{c}[\boldsymbol{x}]^{-1}\boldsymbol{v}[\boldsymbol{x}] + s}$
    $s \leftarrow (1-\gamma)^2 s + 2(1-\gamma)\gamma r \boldsymbol{c}[\boldsymbol{x}]^{-1}\boldsymbol{v}[\boldsymbol{x}] + \gamma^2 r^2$
    $t \leftarrow (1-\gamma)t + \gamma r \boldsymbol{c}[\boldsymbol{x}]^{-1}\boldsymbol{u}[\boldsymbol{x}]$
    $\boldsymbol{v} \leftarrow (1-\gamma)\boldsymbol{v} + \gamma r \boldsymbol{c}[\boldsymbol{x}]^{-1}\boldsymbol{K}[\boldsymbol{x}, \cdot]$
end function

function Next
    return $\operatorname{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{pool}} \setminus \mathcal{X}_{\mathrm{batch}}} \boldsymbol{c}[\boldsymbol{x}]^{-1}(\boldsymbol{u}[\boldsymbol{x}] - \boldsymbol{v}[\boldsymbol{x}])$
end function
```

in the code of Pinsler et al. (2019). In Algorithm 12, when ignoring the computation of $\boldsymbol{\Phi}$, the runtime complexity of Init is $\mathcal{O}(N_{\mathrm{cand}} d_{\mathrm{feat}})$, the runtime complexity of Add is $\mathcal{O}(d_{\mathrm{feat}})$, and the runtime of Next is $\mathcal{O}(N_{\mathrm{pool}} d_{\mathrm{feat}})$. In total, we obtain a runtime complexity of $\mathcal{O}((N_{\mathrm{cand}} + N_{\mathrm{pool}} N_{\mathrm{batch}} + N_{\mathrm{sel}}) d_{\mathrm{feat}}) = \mathcal{O}((N_{\mathrm{cand}} + N_{\mathrm{pool}} N_{\mathrm{batch}}) d_{\mathrm{feat}})$. The memory complexity of Algorithm 12 is $\mathcal{O}(N_{\mathrm{cand}} d_{\mathrm{feat}})$.

6.D.7 MaxDist

The MaxDist selection method has been proposed various times in the literature under many different names. Up to the selection of the first two points, it is equivalent to the Kennard-Stone algorithm (Kennard and Stone, 1969) proposed for experimental design. Rosenkrantz et al. (1977) proposed it under the name *farthest insertion* to generate an insertion order for constructing an approximate TSP solution. Later, Gonzalez (1985) proposed it as an approximation algorithm for a clustering problem. In this context, it is also known as *farthest-point clustering* (Bern and Eppstein, 1996) or *k-center greedy* (Sener and Savarese, 2018). Moreover, it has been proposed as an initialization method for k-means clustering (Katsavounidis et al., 1994). MaxDist is also equivalent to the *geometric greedy* algorithm for kernel interpolation (De Marchi et al., 2005). When used with $N_{\mathrm{batch}} = N_{\mathrm{pool}}$ to construct an ordering of the points, it is known as *farthest-first traversal* or *greedy permutation* of a finite metric space (Eppstein et al., 2020).

Algorithm 13 shows a pseudocode implementation of MaxDist. Since Add has a runtime of $\mathcal{O}(N_{\mathrm{pool}}(T_k + 1))$, the runtime of Algorithm 13 is $\mathcal{O}(N_{\mathrm{pool}} N_{\mathrm{sel}}(T_k + 1))$. The

Algorithm 12 FrankWolfe pseudocode implementation in feature space using Algorithm 4.

```
function Init
    $\boldsymbol{\Phi} \leftarrow \phi(\mathcal{X}_{\text{cand}}) \in \mathbb{R}^{N_{\text{cand}} \times d_{\text{feat}}}$    ▷ Corresponds to $\mathcal{L}_n$
    $\boldsymbol{c} \leftarrow (\|\boldsymbol{\Phi}[\boldsymbol{x}, \cdot]\|_2)_{\boldsymbol{x} \in \mathcal{X}_{\text{cand}}}$    ▷ Corresponds to $\sigma_n$
    $r \leftarrow \sum_{\boldsymbol{x} \in \mathcal{X}_{\text{cand}}} \boldsymbol{c}[\boldsymbol{x}]$    ▷ Corresponds to $\sigma$
    $\tilde{\boldsymbol{\Phi}} \leftarrow (\boldsymbol{c}[\boldsymbol{x}]^{-1} \boldsymbol{\Phi}[\boldsymbol{x}, i])_{\boldsymbol{x} \in \mathcal{X}_{\text{cand}}, i \in \{1, \ldots, d_{\text{feat}}\}} \in \mathbb{R}^{N_{\text{cand}} \times d_{\text{feat}}}$    ▷ Corresponds to $\frac{1}{\sigma_n}\mathcal{L}_n$
    $\boldsymbol{u} \leftarrow \sum_{\boldsymbol{x} \in \mathcal{X}_{\text{cand}}} \boldsymbol{\Phi}[\boldsymbol{x}, \cdot]$    ▷ Corresponds to $\mathcal{L}$
    $\boldsymbol{v} \leftarrow \boldsymbol{0} \in \mathbb{R}^{d_{\text{feat}}}$    ▷ Corresponds to $\mathcal{L}(\boldsymbol{w})$
end function

function Add($\boldsymbol{x}$)
    $\gamma \leftarrow \frac{\langle r\tilde{\boldsymbol{\Phi}}[\boldsymbol{x}, \cdot] - \boldsymbol{v}, \boldsymbol{u} - \boldsymbol{v} \rangle}{\langle r\tilde{\boldsymbol{\Phi}}[\boldsymbol{x}, \cdot] - \boldsymbol{v}, r\tilde{\boldsymbol{\Phi}}[\boldsymbol{x}, \cdot] - \boldsymbol{v} \rangle}$
    $\boldsymbol{v} \leftarrow (1 - \gamma)\boldsymbol{v} + \gamma r \tilde{\boldsymbol{\Phi}}[\boldsymbol{x}, \cdot]$
end function

function Next
    return $\operatorname{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\text{pool}} \setminus \mathcal{X}_{\text{batch}}} \langle \tilde{\boldsymbol{\Phi}}[\boldsymbol{x}, \cdot], \boldsymbol{u} - \boldsymbol{v} \rangle$
end function
```

memory complexity is $\mathcal{O}(N_{\text{pool}})$.

We will now investigate approximation guarantees for MaxDist with respect to a covering objective, called *minmax radius clustering* or *euclidean k-center problem* (Bern and Eppstein, 1996). Approximation guarantees can also be given for a related objective called *minmax diameter clustering* (Gonzalez, 1985; Bern and Eppstein, 1996), which will not be discussed here. The following notation will help to define the minmax radius clustering problem:

Definition 6.D.4. For a given pseudometric d (i.e., a metric except that $d(\boldsymbol{x}, \boldsymbol{x}') = 0$ is allowed for $\boldsymbol{x} \neq \boldsymbol{x}'$), batch size $N_{\text{batch}} \in \mathbb{N}$ and batch $\mathcal{X}_{\text{batch}} \subseteq \mathcal{X}_{\text{pool}}$, we define

$$\begin{aligned}\Delta_d(\mathcal{X}_{\text{batch}}) &\coloneqq \max_{\boldsymbol{x} \in \mathcal{X}_{\text{pool}}} \min_{\boldsymbol{x}' \in \mathcal{X}_{\text{mode}} \cup \mathcal{X}_{\text{batch}}} d(\boldsymbol{x}, \boldsymbol{x}')\ , \\ \Delta_d^{N_{\text{batch}}} &\coloneqq \min_{\mathcal{X}_{\text{batch}} \subseteq \mathcal{X}_{\text{pool}}, |\mathcal{X}_{\text{batch}}| = N_{\text{batch}}} \Delta_d(\mathcal{X}_{\text{batch}})\ . \qquad \blacktriangleleft\end{aligned}$$

The minmax radius clustering problem is defined as finding a batch $\mathcal{X}_{\text{batch}} \subseteq \mathcal{X}_{\text{pool}}$ such that $\Delta_d(\mathcal{X}_{\text{batch}})$ is close to $\Delta_d^{N_{\text{batch}}}$. The following lemma asserts that MaxDist yields a 2-approximation to this problem, which is in general (close to) the best possible approximation ratio for any polynomial-time algorithm unless P = NP (Feder and Greene, 1988).

Lemma 6.D.5. *Let $\mathcal{X}_{\text{batch}}$ be the batch selected by* MaxDist *applied to k. Then,*

$$\Delta_{d_k}(\mathcal{X}_{\text{batch}}) \leq 2\Delta_{d_k}^{N_{\text{batch}}}\ .$$

Proof. For $\mathcal{X}_{\text{mode}} = \emptyset$, this has been proven for example in Bern and Eppstein (1996). Sener and Savarese (2018) mentioned the result for general $\mathcal{X}_{\text{mode}}$ but it is unclear where this is proven. Therefore, we give a proof sketch here.

Algorithm 13 MaxDist pseudocode implementation using Algorithm 4.

```
function Init
    $\boldsymbol{c} \leftarrow (k(\boldsymbol{x}, \boldsymbol{x}))_{\boldsymbol{x} \in \mathcal{X}_{\text{cand}}}$
    $\boldsymbol{d} \leftarrow (\infty)_{\boldsymbol{x} \in \mathcal{X}_{\text{pool}}}$                ▷ Minimum squared distances
end function

function Add($\boldsymbol{x}$)
    $\widetilde{\boldsymbol{d}} \leftarrow (\boldsymbol{c}[\boldsymbol{x}] + \boldsymbol{c}[\tilde{\boldsymbol{x}}] - 2k(\boldsymbol{x}, \tilde{\boldsymbol{x}}))_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\text{pool}}}$   ▷ Compute squared kernel distances $d_k(\boldsymbol{x}, \tilde{\boldsymbol{x}})^2$
    $\boldsymbol{d} \leftarrow \min(\boldsymbol{d}, \widetilde{\boldsymbol{d}})$                ▷ element-wise minimum
end function

function Next
    if no point has been added yet then
        return $\operatorname{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\text{pool}} \setminus \mathcal{X}_{\text{batch}}} \boldsymbol{c}[\boldsymbol{x}]$
    end if
    return $\operatorname{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\text{pool}} \setminus \mathcal{X}_{\text{batch}}} \boldsymbol{d}[\boldsymbol{x}]$
end function
```

Let $d \coloneqq d_k$. Let D be the distance of the last selected point in $\mathcal{X}_{\text{batch}}$ to the remaining points in $\mathcal{X}_{\text{batch}} \cup \mathcal{X}_{\text{mode}}$. Then, $\Delta_d(\mathcal{X}_{\text{batch}}) \leq D$, because otherwise another point with a larger distance would have been chosen instead. At the same time, all points in $\mathcal{X}_{\text{batch}}$ are at least a distance of D apart from any other point in $\mathcal{X}_{\text{batch}} \cup \mathcal{X}_{\text{mode}}$, since otherwise the last point would have been chosen already in an earlier step.

Now, consider a set $\tilde{\mathcal{X}}_{\text{batch}} \subseteq \mathcal{X}_{\text{pool}}$ with $|\tilde{\mathcal{X}}_{\text{batch}}| = N_{\text{batch}}$ such that $\Delta_d(\tilde{\mathcal{X}}_{\text{batch}}) = \Delta_d^{N_{\text{batch}}}$. To derive a contradiction, assume $\Delta_d^{N_{\text{batch}}} < \Delta_d(\mathcal{X}_{\text{batch}})/2$. Then, for every $\boldsymbol{x} \in \mathcal{X}_{\text{batch}}$, there must be $\tilde{\boldsymbol{x}} \in \mathcal{X}_{\text{mode}} \cup \tilde{\mathcal{X}}_{\text{batch}}$ such that $d(\boldsymbol{x}, \tilde{\boldsymbol{x}}) < \Delta_d(\mathcal{X}_{\text{batch}})/2$. By our previous considerations, $\tilde{\boldsymbol{x}}$ cannot be in $\mathcal{X}_{\text{mode}}$, so it must be in $\tilde{\mathcal{X}}_{\text{batch}}$. Moreover, because points in $\mathcal{X}_{\text{batch}}$ are at least D apart, no two of them can be closer than $\Delta_d(\mathcal{X}_{\text{batch}})/2$ to the same point in $\tilde{\mathcal{X}}_{\text{batch}}$, hence by the pigeonhole principle every point in $\tilde{\mathcal{X}}_{\text{batch}}$ must have a point in $\mathcal{X}_{\text{batch}}$ that is closer to it than $\Delta_d(\mathcal{X}_{\text{batch}})/2$. Now, let $\boldsymbol{x}'$ be an arbitrary point in $\mathcal{X}_{\text{pool}}$. Then, there is $\tilde{\boldsymbol{x}} \in \mathcal{X}_{\text{mode}} \cup \tilde{\mathcal{X}}_{\text{batch}}$ that is closer than $\Delta_d(\mathcal{X}_{\text{batch}})/2$ to $\boldsymbol{x}'$. Moreover, there is $\boldsymbol{x}$ in $\mathcal{X}_{\text{mode}} \cup \mathcal{X}_{\text{batch}}$ that is closer than $\Delta_d(\mathcal{X}_{\text{batch}})/2$ to $\tilde{\boldsymbol{x}}$. Hence, the triangle inequality yields $d(\boldsymbol{x}', \boldsymbol{x}) < \Delta_d(\mathcal{X}_{\text{batch}})$, and since $\boldsymbol{x}'$ was arbitrary, this is a contradiction. □

The following simple result will be helpful to prove an approximation guarantee when using sketching:

Lemma 6.D.6. *Let $\mathcal{X}_{\text{pool}}$ be a finite set and let $\alpha > 0$. Moreover, let d_1, d_2 be pseudometrics on a set $\mathcal{X}$ such that $d_1(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \leq \alpha d_2(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ for all $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathcal{X}_{\text{pool}}$. Then,*

$$\begin{aligned} \forall \mathcal{X}_{\text{batch}} \subseteq \mathcal{X}_{\text{pool}} &: \Delta_{d_1}(\mathcal{X}_{\text{batch}}) \leq \alpha \Delta_{d_2}(\mathcal{X}_{\text{batch}}) \;, \\ \forall N_{\text{batch}} \in \{1, \ldots, |\mathcal{X}_{\text{pool}}|\} &: \Delta_{d_1}^{N_{\text{batch}}} \leq \alpha \Delta_{d_2}^{N_{\text{batch}}} \;. \end{aligned}$$

Proof. Let $\mathcal{X}_{\text{batch}} \subseteq \mathcal{X}_{\text{pool}}$ and let $\boldsymbol{x} \in \mathcal{X}_{\text{pool}}$. For the element $\boldsymbol{x}'' \in \mathcal{X}_{\text{mode}} \cup \mathcal{X}_{\text{batch}}$ minimizing $d_2(\boldsymbol{x}, \boldsymbol{x}'')$, we have

$$\min_{\boldsymbol{x}' \in \mathcal{X}_{\text{mode}} \cup \mathcal{X}_{\text{batch}}} d_1(\boldsymbol{x}, \boldsymbol{x}') \leq d_1(\boldsymbol{x}, \boldsymbol{x}'') \leq \alpha d_2(\boldsymbol{x}, \boldsymbol{x}'') = \alpha \min_{\boldsymbol{x}' \in \mathcal{X}_{\text{train}} \cup \mathcal{X}_{\text{batch}}} d_2(\boldsymbol{x}, \boldsymbol{x}') \;. \tag{6.32}$$

Algorithm 14 KMeansPP pseudocode implementation using Algorithm 4.

```
function Init
    $\boldsymbol{c} \leftarrow (k(\boldsymbol{x}, \boldsymbol{x}))_{\boldsymbol{x} \in \mathcal{X}_{\text{cand}}}$
    $\boldsymbol{d} \leftarrow (\infty)_{\boldsymbol{x} \in \mathcal{X}_{\text{pool}}}$                    ▷ Minimum squared distances
end function

function Add($\boldsymbol{x}$)
    $\widetilde{\boldsymbol{d}} \leftarrow (\boldsymbol{c}[\boldsymbol{x}] + \boldsymbol{c}[\tilde{\boldsymbol{x}}] - 2k(\boldsymbol{x}, \tilde{\boldsymbol{x}}))_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\text{pool}}}$  ▷ Compute squared kernel distances $d_k(\boldsymbol{x}, \tilde{\boldsymbol{x}})^2$
    $\boldsymbol{d} \leftarrow \min(\boldsymbol{d}, \widetilde{\boldsymbol{d}})$                    ▷ element-wise minimum
end function

function Next
    if no point has been added yet then
        return uniform random sample from $\mathcal{X}_{\text{pool}} \setminus \mathcal{X}_{\text{batch}}$
    end if
    return sample $\boldsymbol{x} \in \mathcal{X}_{\text{pool}} \setminus \mathcal{X}_{\text{batch}}$ with probability proportional to $\boldsymbol{d}[\boldsymbol{x}]$
end function
```

Applying an analogous argument to $\boldsymbol{x}$ shows that $\Delta_{d_1}(\mathcal{X}_{\text{batch}}) \leq \alpha \Delta_{d_2}(\mathcal{X}_{\text{batch}})$. Another application to $\mathcal{X}_{\text{batch}}$ then shows that $\Delta_{d_1}^{N_{\text{batch}}} \leq \alpha \Delta_{d_2}^{N_{\text{batch}}}$. □

Finally, we obtain the following approximation guarantee with sketching:

Theorem 6.D.7. *Suppose that Eq. (6.15) holds. Then, the batch $\mathcal{X}_{\text{batch}}$ with size N_{batch} computed by* MaxDist *applied to $k_{\to \text{sketch}(p)}$ satisfies*

$$\Delta_{d_k}(\mathcal{X}_{\text{batch}}) \leq 2\frac{1+\varepsilon}{1-\varepsilon}\Delta_{d_k}^{N_{\text{batch}}} .$$

Proof. Let $d_1 := d_k$ and $d_2 := d_{k_{\to \text{sketch}(p)}}$. Then,

$$\begin{aligned} \Delta_{d_1}(\mathcal{X}_{\text{batch}}) &\overset{\text{Lemma 6.D.6}}{\leq} \frac{1}{1-\varepsilon}\Delta_{d_2}(\mathcal{X}_{\text{batch}}) \overset{\text{Lemma 6.D.5}}{\leq} 2\frac{1}{1-\varepsilon}\Delta_{d_2}^{N_{\text{batch}}} \\ &\overset{\text{Lemma 6.D.6}}{\leq} 2\frac{1+\varepsilon}{1-\varepsilon}\Delta_{d_1}^{N_{\text{batch}}} . \end{aligned}$$

□

6.D.8 KMeansPP

Algorithm 14 shows pseudocode for KMeansPP. Like for MaxDist, we obtain a runtime complexity of $\mathcal{O}(N_{\text{pool}}N_{\text{sel}}(T_k + 1))$ and a memory complexity of $\mathcal{O}(N_{\text{pool}})$.

6.D.9 LCMD

A pseudocode implementation of our newly proposed LCMD method is shown in Algorithm 15. Here, the Add method has a runtime complexity of $\mathcal{O}(N_{\text{pool}}(T_k + 1))$ and the Next method has a runtime complexity of $\mathcal{O}(N_{\text{pool}} + N_{\text{sel}})$. The overall runtime complexity is therefore $\mathcal{O}(N_{\text{pool}}N_{\text{sel}}(T_k + 1) + N_{\text{batch}}(N_{\text{pool}} + N_{\text{sel}})) = \mathcal{O}(N_{\text{pool}}N_{\text{sel}}(T_k + 1))$. The memory complexity of Algorithm 15 is $\mathcal{O}(N_{\text{cand}})$.

Algorithm 15 LCMD pseudocode implementation using Algorithm 4.

```
function Init
    $\boldsymbol{c} \leftarrow (k(\boldsymbol{x}, \boldsymbol{x}))_{\boldsymbol{x} \in \mathcal{X}_{\text{cand}}}$
    $\boldsymbol{d} \leftarrow (\infty)_{\boldsymbol{x} \in \mathcal{X}_{\text{pool}}}$                    ▷ Minimum squared distances
    $\boldsymbol{v} \leftarrow (0)_{\boldsymbol{x} \in \mathcal{X}_{\text{pool}}}$          ▷ Associated cluster centers; dummy initialization
end function

function Add($\boldsymbol{x}$)
    $\widetilde{\boldsymbol{d}} \leftarrow (\boldsymbol{c}[\boldsymbol{x}] + \boldsymbol{c}[\tilde{\boldsymbol{x}}] - 2k(\boldsymbol{x}, \tilde{\boldsymbol{x}}))_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\text{pool}}}$   ▷ Compute squared kernel distances $d_k(\boldsymbol{x}, \tilde{\boldsymbol{x}})^2$
    $\boldsymbol{v} \leftarrow (\boldsymbol{v}[\tilde{\boldsymbol{x}}] \text{ if } \boldsymbol{d}[\tilde{\boldsymbol{x}}] \leq \widetilde{\boldsymbol{d}}[\tilde{\boldsymbol{x}}] \text{ else } \boldsymbol{x})_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\text{pool}}}$         ▷ Update associated cluster centers
    $\boldsymbol{d} \leftarrow \min(\boldsymbol{d}, \widetilde{\boldsymbol{d}})$                                  ▷ element-wise minimum
end function

function Next
    if no point has been added yet then
        return $\operatorname{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\text{pool}} \setminus \mathcal{X}_{\text{batch}}} \boldsymbol{c}[\boldsymbol{x}]$
    end if
    $\boldsymbol{s} \leftarrow (\sum_{\tilde{\boldsymbol{x}} \in \mathcal{X}_{\text{pool}} \setminus \mathcal{X}_{\text{batch}} : \boldsymbol{v}[\tilde{\boldsymbol{x}}] = \boldsymbol{x}} \boldsymbol{d}[\tilde{\boldsymbol{x}}])_{\boldsymbol{x} \in \mathcal{X}_{\text{mode}} \cup \mathcal{X}_{\text{batch}}}$          ▷ Compute cluster sizes
    $s_{\max} = \max_{\boldsymbol{x} \in \mathcal{X}_{\text{mode}} \cup \mathcal{X}_{\text{batch}}} \boldsymbol{s}[\boldsymbol{x}]$                       ▷ Maximum cluster size
    return $\operatorname{argmax}_{\boldsymbol{x} \in \mathcal{X}_{\text{pool}} \setminus \mathcal{X}_{\text{batch}} : \boldsymbol{s}[\boldsymbol{x}] = s_{\max}} \boldsymbol{d}[\boldsymbol{x}]$
end function
```

6.E Details on Experiments

In the following, we provide a more detailed description of our experimental setup and our results. All NN computations were performed with 32-bit floating-point precision, but we switched to 64-bit floating-point precision whenever posterior transformations (computations involving σ^2) were involved. All MaxDet computations used the kernel-space implementation and all FrankWolfe computations used the feature-space implementation. All experiments were run on a workstation with four NVIDIA RTX 3090 GPUs and an AMD Ryzen Threadripper PRO 3975WX CPU with 256 GB RAM.

6.E.1 Data Sets

We selected 15 tabular regression data sets from different sources, roughly using the following criteria:

(a) The data set should be sufficiently large after removing rows with missing values (at least 40000 samples).
(b) The data set should be in a format suitable to perform regression, e.g., not consist of a few long time series.
(c) The data set should not have too many categorical or text columns with many categories.
(d) The test RMSE for randomly sampled training sets should drop substantially when going from $N_{\text{train}} = 256$ to $N_{\text{train}} = 17 \cdot 256$. In our case, the selected 15 data sets

differ from the other tested data sets in that the RMSE dropped at least by 14%. With this criterion, we want to exclude data sets that would not significantly affect the benchmark results, e.g., because they are too easy to learn or because they are too noisy.

An overview of the selected data sets can be found in Table 6.E.1 and Table 6.E.2. Our main data sources are the UCI and OpenML repositories (Dua and Graff, 2017; Vanschoren et al., 2013). The sgemm and ct_slices data sets have also been used by Tsymbalov et al. (2018). In contrast to Tsymbalov et al. (2018), we use the undirected and not the directed version of the kegg data set since it contains more samples and the RMSE drops more strongly between $N_{\text{train}} = 256$ and $N_{\text{train}} = 17 \cdot 256$. Concerning the other four data sets used in Tsymbalov et al. (2018), we omitted the BlogFeedback, YearPredictionMSD, and Online News Popularity data sets due to criterion (d), and could not find the Rosenbrock 2000D data set online. Although the poker data set is originally a multi-class classification data set, we include it since it is noise-free and sufficiently difficult to learn.

Our accompanying code allows automatically downloading and processing all data sets.

6.E.2 Preprocessing

We preprocess the data sets in the following way: On some data sets, we remove unwanted columns such as identifier columns, see the details in our code. We then remove rows with NaN (missing) values. Next, if necessary, we randomly subsample the data set such that it contains at most 500000 samples. We use 80% of the data set, but at most 200000 samples, for training, validation, and pool data. Of those, we initially use $N_{\text{train}} = 256$ and $N_{\text{valid}} = 1024$ and reserve the rest for the pool set. Subsequently, we remove columns with only a single value. We one-hot encode small categorical columns, allowing at most 300 new continuous columns, and discard larger categorical columns. On some data sets, we transform the labels y, for example by applying a logarithm or by taking the median of multiple target values, we refer to our code for further details. We standardize the labels y such that they have mean 0 and variance 1.[9] We preprocess the inputs $\boldsymbol{x} \in \mathbb{R}^d$ as

$$x_j^{\text{processed}} := 5 \tanh\left(\frac{1}{5} \cdot \frac{x_j - \hat{\mu}_j}{\hat{\sigma}_j}\right) ,$$

where we compute

$$\begin{aligned} \hat{\mu}_j &:= \frac{1}{N_{\text{train}} + N_{\text{pool}}} \sum_{\boldsymbol{x} \in \mathcal{X}_{\text{train}} \cup \mathcal{X}_{\text{pool}}} x_j \\ \hat{\sigma}_j^2 &:= \frac{1}{N_{\text{train}} + N_{\text{pool}}} \sum_{\boldsymbol{x} \in \mathcal{X}_{\text{train}} \cup \mathcal{X}_{\text{pool}}} (x_j - \hat{\mu}_j)^2 . \end{aligned}$$

The motivation for the tanh function is to reduce the impact of outliers by soft-clipping the coordinates to the interval $(-5, 5)$.

For the sarcos data set, we only used the training data since the test data on the GPML web page (see Table 6.E.2) is already contained in the training data. For the poker

[9]This only leaks a negligible amount of information from the test set and in turn allows us to better compare the errors across data sets.

Short name	Initial pool set size	Test set size	Number of features
sgemm	192000	48320	14
wec_sydney	56320	14400	48
ct_slices	41520	10700	379
kegg_undir	50407	12921	27
online_video	53748	13756	26
query	158720	40000	4
poker	198720	300000	95
road	198720	234874	2
mlr_knn_rng	88123	22350	132
fried	31335	8153	10
diamonds	41872	10788	29
methane	198720	300000	33
stock	45960	11809	9
protein	35304	9146	9
sarcos	34308	8896	21

Table 6.E.1: Data set characteristics.

data set, we only used the test data since it contains roughly a million samples, while the training set contains around 25000 samples.

The resulting characteristics of the processed data sets can be found in Table 6.E.1. The full names, links, and citations are contained in Table 6.E.2.

6.E.3 Neural Network Configuration

We use a fully-connected NN with two hidden layers with 512 neurons each ($L = 3$, $d_1 = d_2 = 512$). We employ the neural tangent parametrization as discussed in Section 6.2.1 with the ReLU activation function. We initialize biases to zero and weights i.i.d. from $\mathcal{N}(0, 1)$. For optimization, we use the Adam (Kingma and Ba, 2015) optimizer with its default parameters $\beta_1 = 0.9, \beta_2 = 0.999$, and let the learning rate (see below) decay linearly to zero over training. We use a mini-batch size of 256 and train for 256 epochs. After each epoch, we measure the validation RMSE on a validation set with 1024 samples. After training, we set the trained model parameters $\boldsymbol{\theta}_T$ to the parameters from the end of the epoch where the lowest validation RMSE was attained. While the use of a large validation set might not be realistic for many data-scarce BMAL scenarios, we see this as a simple proxy for more complicated cross-validation or refitting strategies.

While the reasoning of forward variance preservation as in the well-known Kaiming initialization (He et al., 2015) suggests to set $\sigma_w = \sqrt{2}$, we find that smaller values of σ_w can substantially improve the RMSE of the trained models. Possible explanations for this phenomenon might be that large σ_w increases the scale of the disturbance by the random initial function of the network (Nonnenmacher et al., 2021) or brings the NN more towards a “lazy training” regime (Chizat et al., 2019). Therefore, we manually tuned σ_w, σ_b and the initial learning rate to optimize the mean log RMSE across all data sets for Random selection. We arrived at $\sigma_w = \sigma_b = 0.2$ and an initial learning rate of 0.375.

To assess whether our insights apply to other NN configurations, we also run experiments for a fully-connected NN with the SiLU (a.k.a. Swish) activation function (Elfwing et al., 2018). Again, we use optimized hyperparameters for SiLU, specifically an initial learning rate of 0.15 as well as $\sigma_w = 0.5, \sigma_b = 1.0$.

6.E.4 Results

Table 6.E.5 shows averaged logarithmic error metrics and runtimes for a wide variety of configurations. Some conclusions from these results are discussed in Section 6.6. Table 6.E.6 shows analogous results for our NN configuration with the SiLU instead of the ReLU activation function. Note that the results for k_{nngp} still use the NNGP for ReLU, though, since we do not know of an analytic expression of the NNGP for SiLU. Results on individual data sets for selected methods are shown in Table 6.E.3 and Table 6.E.4.

In this section, we also provide more plots complementing the figures from the main part of the paper. Figure 6.E.1 shows batch size plots on individual data sets for RMSE and Figure 6.E.2 shows learning curve plots on individual data sets for the 99% quantile. Moreover, Figure 6.E.3 and Figure 6.E.4 allow comparing two methods across data sets on RMSE and MAXE, respectively.

The estimated standard deviations of the mean estimators in Figure 1, Figure 3, Figure 4, Figure 5, Figure 6.E.1 and Figure 6.E.2 are computed as follows: Consider random variables X_{ij} representing the log metric values on repetition i and data set j. Then, it is well-known that for the mean estimator $\hat{\mu}_j := \frac{1}{20}\sum_{i=1}^{20} X_{ij}$, an unbiased estimator of its variance is given by

$$\hat{\sigma}_j^2 := \frac{1}{20-1}\sum_{i=1}^{20}(X_{ij} - \hat{\mu}_j)^2$$

Since all mean estimators $\hat{\mu}_j$ are independent, the variance of the total mean estimator $\hat{\mu} := \frac{1}{15}\sum_{j=1}^{15}\hat{\mu}_j$ can be estimated as

$$\hat{\sigma}^2 := \frac{1}{15^2}\sum_{j=1}^{15}\hat{\sigma}_j^2 \ .$$

Our plots hence show $\hat{\sigma}$ as the estimated standard deviation of the mean estimator $\hat{\mu}$.

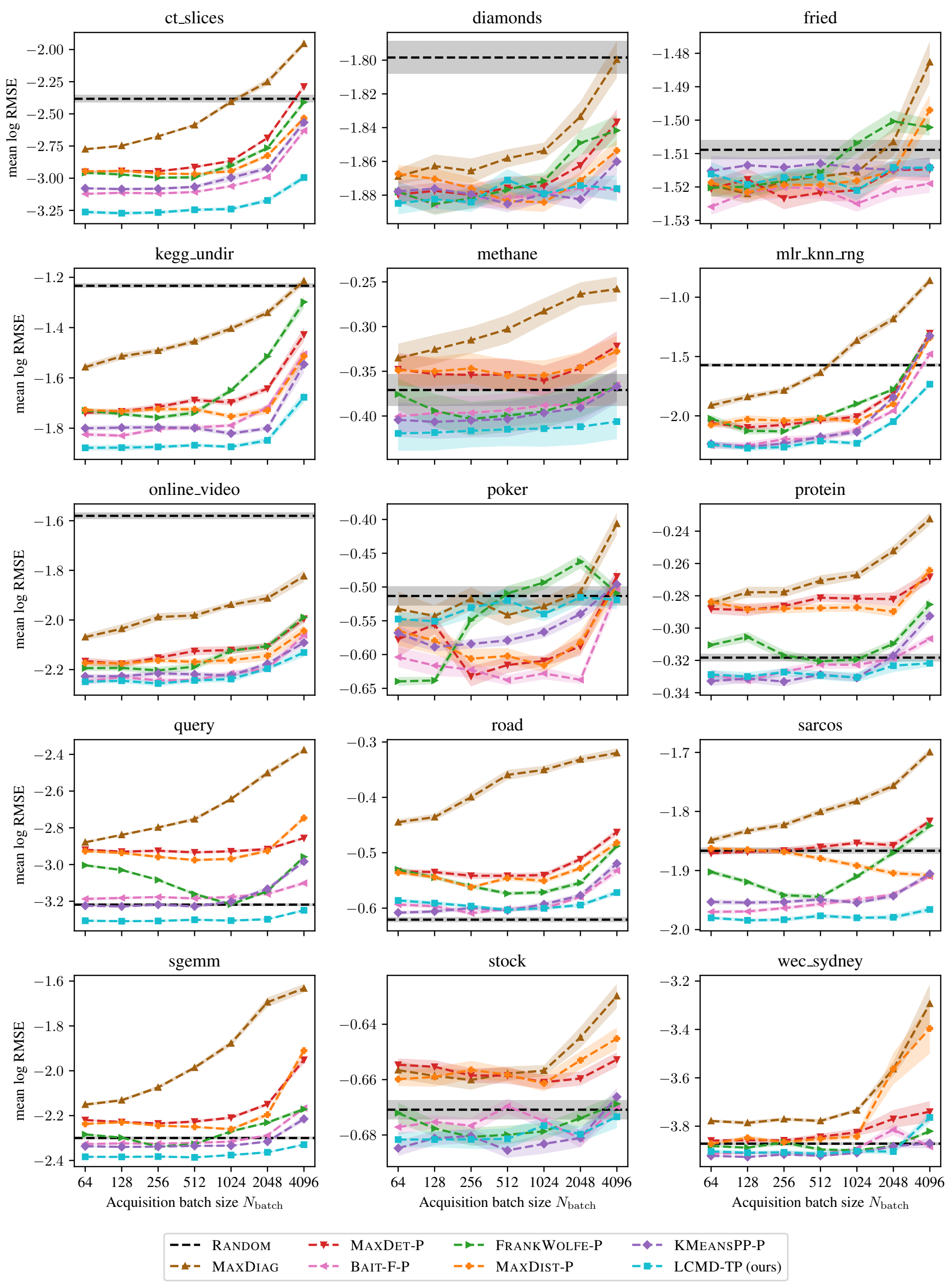


Figure 6.E.1: This figure shows how much the final accuracy of different BMDAL methods deteriorates on individual data sets when fewer BMAL steps with larger batch sizes are used. Specifically, we use different selection methods with the corresponding kernels from Table 6, starting with $N_{\text{train}} = 256$ and then performing 2^m BMAL steps with batch size $N_{\text{batch}} = 2^{12-m}$ for $m \in \{0, \dots, 6\}$, such that the final training set size is 4352 in each case. The plots show the final logarithmic error metric, averaged over all repetitions. Note that the performance of RANDOM selection does not depend on N_{batch} but only on the final training set size, hence it is shown as a constant line here. The shaded area corresponds to one estimated standard deviation of the mean estimator, cf. Section 6.E.4.

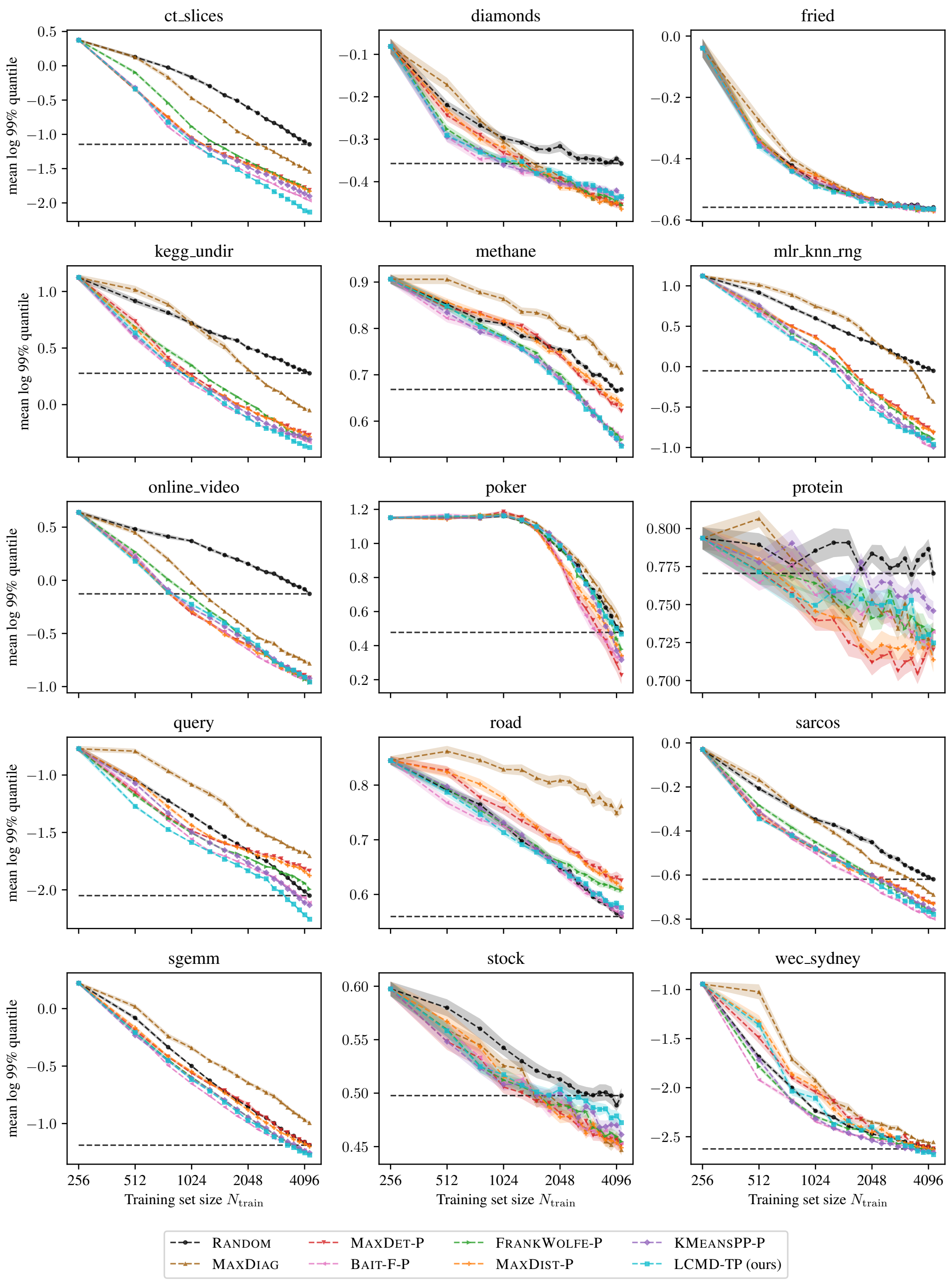


Figure 6.E.2: This figure shows how fast the 99% quantile decreases during BMAL on the individual benchmark data sets for different selection methods and their corresponding kernels from Table 6. Specifically, the plots above show the logarithmic 99% quantile between each BMAL step for $N_{\text{batch}} = 256$, averaged over all repetitions. The black horizontal dashed line corresponds to the final performance of RANDOM at $N_{\text{train}} = 4352$. The shaded area corresponds to one estimated standard deviation of the mean estimator, cf. Section 6.E.4.

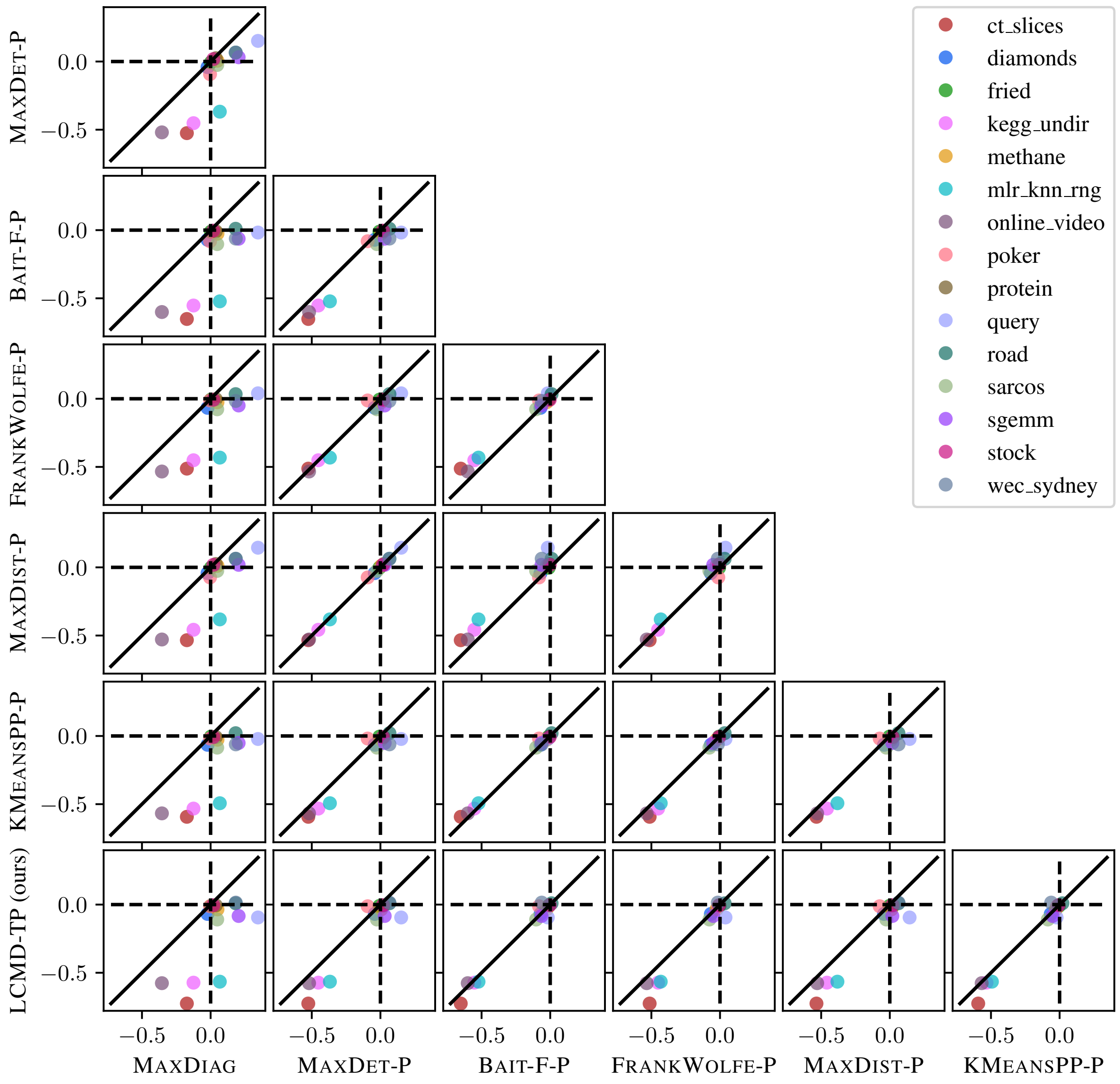


Figure 6.E.3: Each subplot shows the errors of two selection methods and their corresponding selected kernels from Table 6. Specifically, the coordinates correspond to the mean log RMSE of the method on the data set minus the mean log RMSE of Random selection on the same data set. Hence, the method on the x axis has a lower mean log RMSE than Random on a data set if the corresponding point is left of the vertical dashed line, and it has a lower mean log RMSE than the method on the y axis if the corresponding point is left of the diagonal line. Similarly, the method on the y axis has a lower mean log RMSE than Random on a data set if the corresponding point is below the horizontal dashed line, and it has a lower mean log RMSE than the method on the x axis if the corresponding point is below the diagonal line.

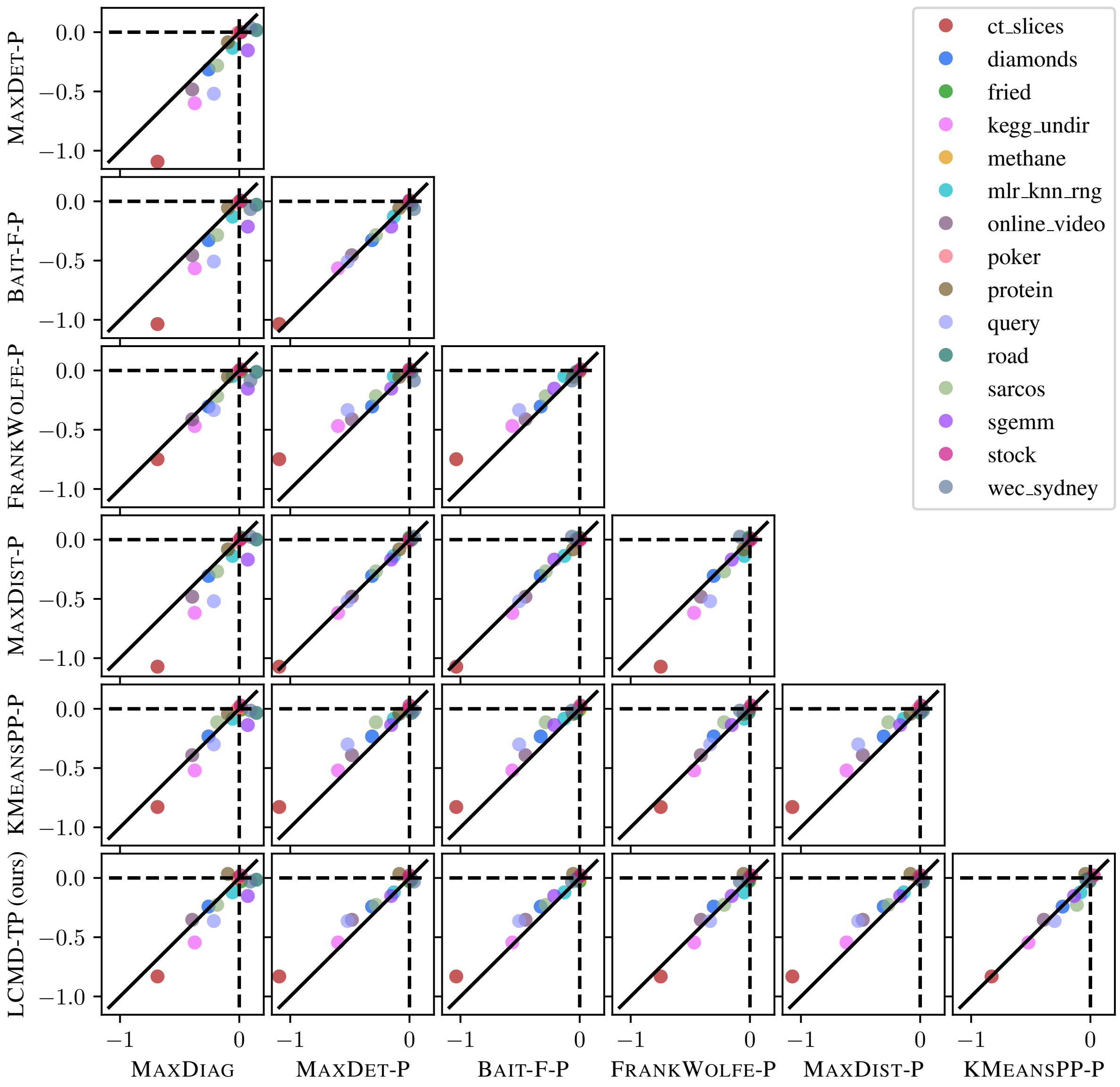


Figure 6.E.4: Each subplot shows the errors of two selection methods and their corresponding selected kernels from Table 6. Specifically, the coordinates correspond to the mean log MAXE of the method on the data set minus the mean log MAXE of Random selection on the same data set. Hence, the method on the x axis has a lower mean log MAXE than Random on a data set if the corresponding point is left of the vertical dashed line, and it has a lower mean log MAXE than the method on the y axis if the corresponding point is left of the diagonal line.

Short name	Source	OpenML ID	Full name	Citation
sgemm	UCI		SGEMM GPU kernel performance	Ballester-Ripoll et al. (2019)
wec_sydney	UCI		Wave Energy Converters	Neshat et al. (2018)
ct_slices	UCI		Relative location of CT slices on axial axis	Graf et al. (2011)
kegg_undir	UCI		KEGG Metabolic Reaction Network (Undirected)	Shannon et al. (2003)
online_video	UCI		Online Video Characteristics and Transcoding Time	Deneke et al. (2014)
query	UCI		Query Analytics Workloads	Anagnostopoulos et al. (2018), Savva et al. (2018)
poker	UCI		Poker Hand	—
road	UCI		3D Road Network (North Jutland, Denmark)	Kaul et al. (2013)
mlr_knn_rng	OpenML	42454	mlr_knn_rng	—
fried	OpenML	564	fried	Friedman (1991)
diamonds	OpenML	42225	diamonds	—
methane	OpenML	42701	Methane	Ślęzak et al. (2018)
stock	OpenML	1200	BNG(stock)	—
protein	OpenML	42903	physicochemical-protein	—
sarcos	GPML		SARCOS data	Vijayakumar and Schaal (2000)

Table 6.E.2: Overview of used data sets. The second column entries are hyperlinks to the respective web pages.

Data set	Random	MaxDiag	MaxDet-P	Bait-F-P	FrankWolfe-P	MaxDist-P	KMeansPP-P	LCMD-TP (ours)
ct_slices	0.141	0.123	0.085	0.076	0.088	0.085	0.081	**0.072**
diamonds	0.173	0.169	0.166	**0.161**	0.162	0.166	0.162	**0.161**
fried	0.230	0.231	0.228	**0.227**	0.228	0.229	0.229	0.228
kegg_undir	0.380	0.346	0.245	0.222	0.248	0.243	0.227	**0.220**
methane	0.733	0.770	0.736	0.714	0.714	0.740	0.713	**0.708**
mlr_knn_rng	0.294	0.326	0.211	0.183	0.199	0.209	0.190	**0.176**
online_video	0.263	0.190	0.159	**0.149**	0.159	0.158	0.153	0.152
poker	0.806	0.803	**0.742**	0.751	0.797	0.754	0.794	0.797
protein	0.763	0.793	0.782	**0.757**	0.761	0.782	**0.757**	0.759
query	0.058	0.082	0.066	0.057	0.060	0.066	0.057	**0.053**
road	**0.586**	0.702	0.625	0.592	0.606	0.624	0.598	0.591
sarcos	0.181	0.190	0.176	0.164	0.168	0.176	0.167	**0.163**
sgemm	0.152	0.185	0.155	0.142	0.144	0.153	0.144	**0.140**
stock	0.531	0.541	0.540	0.529	0.527	0.540	**0.526**	0.528
wec_sydney	0.027	0.034	0.030	**0.025**	0.027	0.030	0.026	0.028

Table 6.E.3: This table shows the averaged (non-logarithmic) RMSEs per data set, averaged over all repetitions, and BMAL steps, for each of the selection methods with kernels as in Table 6.

Data set	Random	MaxDiag	MaxDet-P	Bait-F-P	FrankWolfe-P	MaxDist-P	KMeansPP-P	LCMD-TP (ours)
ct_slices	0.093	0.069	0.053	0.044	0.050	0.052	0.046	**0.038**
diamonds	0.166	0.155	0.153	**0.152**	**0.152**	0.153	0.153	**0.152**
fried	0.221	0.219	**0.218**	0.219	0.219	0.219	0.220	0.219
kegg_undir	0.291	0.225	0.180	0.165	0.173	0.179	0.166	**0.154**
methane	0.692	0.731	0.704	0.675	0.670	0.709	0.669	**0.661**
mlr_knn_rng	0.208	0.168	0.126	0.112	0.119	0.131	0.108	**0.105**
online_video	0.206	0.137	0.116	0.106	0.111	0.115	0.109	**0.105**
poker	0.600	0.598	**0.533**	0.537	0.578	0.547	0.558	0.589
protein	0.727	0.758	0.751	0.721	0.729	0.750	**0.717**	0.721
query	0.040	0.061	0.054	0.042	0.046	0.052	0.040	**0.037**
road	**0.538**	0.671	0.582	0.544	0.571	0.570	0.549	0.551
sarcos	0.155	0.162	0.155	0.140	0.143	0.154	0.142	**0.138**
sgemm	0.100	0.126	0.107	0.098	0.097	0.106	0.096	**0.092**
stock	0.511	0.517	0.518	0.508	**0.506**	0.519	**0.506**	**0.506**
wec_sydney	0.021	0.023	0.021	**0.020**	0.021	0.021	**0.020**	**0.020**

Table 6.E.4: This table shows the (non-logarithmic) RMSEs after the last BMAL step per data set, averaged over all repetitions, for each of the selection methods with kernels as in Table 6.

Selection method	Kernel	MAE	RMSE	95%	99%	MAXE	avg. time [s]
Random	—	-1.934	-1.401	-0.766	-0.163	1.107	0.001
MaxDiag	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf}(512)}$	-1.777	-1.370	-0.690	-0.189	0.978	0.650
MaxDiag	$k_{\text{grad}\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-1.777	-1.369	-0.690	-0.186	0.986	0.551
MaxDiag	$k_{\text{grad}\to\text{ens}(3)\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-1.768	-1.366	-0.685	-0.188	0.970	1.392
MaxDiag	$k_{\text{grad}\to\mathcal{X}_{\text{train}}}$	-1.766	-1.355	-0.675	-0.168	0.996	4.108
MaxDiag	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-grad}}$	-1.751	-1.345	-0.664	-0.163	1.007	0.553
MaxDiag	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf-hyper}(512)}$	-1.743	-1.334	-0.656	-0.148	1.021	0.650
MaxDiag	$k_{\text{ll}\to\text{acs-rf}(512)}$	-1.713	-1.286	-0.606	-0.084	1.052	0.030
MaxDiag	$k_{\text{ll}\to\mathcal{X}_{\text{train}}}$	-1.722	-1.285	-0.614	-0.077	1.080	0.142
MaxDiag	$k_{\text{nngp}\to\mathcal{X}_{\text{train}}}$	-1.754	-1.272	-0.606	-0.044	1.088	2.592
MaxDiag	$k_{\text{lin}\to\mathcal{X}_{\text{train}}}$	-1.585	-1.103	-0.426	0.143	1.222	0.007
MaxDet-TP	$k_{\text{grad}\to\text{scale}(\mathcal{X}_{\text{train}})}$	-1.930	-1.522	-0.855	-0.356	0.856	8.883
MaxDet-P	$k_{\text{grad}\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-1.915	-1.512	-0.844	-0.350	0.867	0.770
MaxDet-P	$k_{\text{ll}\to\text{ens}(3)\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-1.931	-1.504	-0.849	-0.337	0.873	0.673
MaxDet-P	$k_{\text{grad}\to\text{ens}(3)\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-1.895	-1.500	-0.830	-0.344	0.868	1.608
MaxDet-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf}(512)}$	-1.893	-1.492	-0.820	-0.323	0.886	0.869
MaxDet-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-grad}}$	-1.872	-1.475	-0.802	-0.311	0.878	0.893
MaxDet-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf-hyper}(512)}$	-1.872	-1.475	-0.803	-0.310	0.888	0.872
MaxDet-P	$k_{\text{ll}\to\mathcal{X}_{\text{train}}}$	-1.895	-1.463	-0.808	-0.288	0.916	0.370
MaxDet-P	$k_{\text{ll}\to\text{acs-rf}(512)}$	-1.876	-1.443	-0.792	-0.264	0.961	0.518
MaxDet-TP	$k_{\text{nngp}\to\text{scale}(\mathcal{X}_{\text{train}})}$	-1.848	-1.358	-0.702	-0.133	1.028	5.449
MaxDet-P	$k_{\text{lin}\to\mathcal{X}_{\text{train}}}$	-1.682	-1.187	-0.524	0.055	1.191	0.170
Bait-F-P	$k_{\text{grad}\to\text{ens}(3)\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-2.011	-1.587	-0.927	**-0.419**	0.859	2.346
Bait-F-P	$k_{\text{grad}\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-2.013	-1.585	-0.926	-0.412	0.862	1.508
Bait-FB-P	$k_{\text{grad}\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-2.007	-1.584	-0.921	-0.411	**0.852**	3.050
Bait-F-P	$k_{\text{ll}\to\text{ens}(3)\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	**-2.041**	-1.583	-0.940	-0.400	0.880	1.408
Bait-F-P	$k_{\text{ll}\to\mathcal{X}_{\text{train}}}$	-2.003	-1.545	-0.900	-0.362	0.891	1.149
Bait-FB-P	$k_{\text{ll}\to\mathcal{X}_{\text{train}}}$	-1.998	-1.541	-0.895	-0.357	0.888	2.731
Bait-F-P	$k_{\text{lin}\to\mathcal{X}_{\text{train}}}$	-1.721	-1.220	-0.562	0.022	1.162	0.232
FrankWolfe-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf-hyper}(512)}$	-1.977	-1.542	-0.892	-0.362	0.918	0.823
FrankWolfe-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-grad}\to\text{sketch}(512)}$	-1.999	-1.520	-0.879	-0.317	1.015	0.914
FrankWolfe-P	$k_{\text{ll}\to\text{acs-rf}(512)}$	-1.992	-1.519	-0.883	-0.321	1.022	0.421
FrankWolfe-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf}(512)}$	-1.995	-1.499	-0.864	-0.287	1.055	0.825
FrankWolfe-P	$k_{\text{ll}\to\text{acs-grad}\to\text{sketch}(512)}$	-1.943	-1.446	-0.807	-0.226	1.085	0.517
FrankWolfe-P	$k_{\text{ll}\to\text{acs-rf-hyper}(512)}$	-1.937	-1.439	-0.793	-0.225	1.016	0.421
FrankWolfe-P	$k_{\text{grad}\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-1.924	-1.410	-0.765	-0.178	1.134	0.723
FrankWolfe-P	$k_{\text{ll}\to\mathcal{X}_{\text{train}}}$	-1.896	-1.391	-0.752	-0.158	1.139	0.326
MaxDist-TP	$k_{\text{ll}\to\text{ens}(3)\to\text{sketch}(512)}$	-1.948	-1.518	-0.860	-0.348	0.905	0.655
MaxDist-P	$k_{\text{grad}\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-1.916	-1.514	-0.845	-0.351	0.866	0.713
MaxDist-TP	$k_{\text{grad}\to\text{sketch}(512)}$	-1.899	-1.506	-0.838	-0.347	0.868	0.653
MaxDist-TP	k_{grad}	-1.894	-1.503	-0.834	-0.342	0.866	2.347
MaxDist-TP	$k_{\text{grad}\to\text{ens}(3)\to\text{sketch}(512)}$	-1.889	-1.498	-0.831	-0.342	0.871	0.743
MaxDist-TP	k_{ll}	-1.924	-1.491	-0.832	-0.307	0.927	0.621
MaxDist-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf}(512)}$	-1.893	-1.491	-0.819	-0.322	0.891	0.810
MaxDist-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-grad}}$	-1.873	-1.477	-0.802	-0.312	0.888	0.832
MaxDist-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf-hyper}(512)}$	-1.867	-1.472	-0.798	-0.306	0.895	0.811
MaxDist-P	$k_{\text{ll}\to\mathcal{X}_{\text{train}}}$	-1.889	-1.459	-0.807	-0.283	0.947	0.309
MaxDist-P	$k_{\text{ll}\to\text{acs-rf}(512)}$	-1.863	-1.430	-0.777	-0.247	0.980	0.410
MaxDist-TP	k_{lin}	-1.888	-1.398	-0.749	-0.172	1.034	0.242
MaxDist-TP	k_{nngp}	-1.876	-1.386	-0.735	-0.159	1.038	1.315
KMeansPP-TP	k_{grad}	-2.025	-1.569	-0.927	-0.378	0.966	2.357
KMeansPP-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf}(512)}$	-2.006	-1.569	-0.912	-0.385	0.929	0.836
KMeansPP-TP	$k_{\text{grad}\to\text{ens}(3)\to\text{sketch}(512)}$	-2.025	-1.569	-0.926	-0.377	0.967	0.754
KMeansPP-TP	$k_{\text{grad}\to\text{sketch}(512)}$	-2.023	-1.567	-0.925	-0.376	0.967	0.663
KMeansPP-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-grad}}$	-2.008	-1.558	-0.905	-0.366	0.979	0.859
KMeansPP-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf-hyper}(512)}$	-1.994	-1.554	-0.899	-0.370	0.957	0.836
KMeansPP-P	$k_{\text{grad}\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-2.020	-1.549	-0.905	-0.348	0.997	0.738
KMeansPP-P	$k_{\text{ll}\to\mathcal{X}_{\text{train}}}$	-2.007	-1.530	-0.895	-0.329	1.008	0.335
KMeansPP-P	$k_{\text{ll}\to\text{acs-rf}(512)}$	-1.986	-1.529	-0.889	-0.339	0.977	0.435
KMeansPP-TP	$k_{\text{ll}\to\text{ens}(3)\to\text{sketch}(512)}$	-2.020	-1.522	-0.890	-0.317	1.014	0.666
KMeansPP-TP	k_{ll}	-2.015	-1.521	-0.887	-0.316	1.014	0.632
KMeansPP-TP	k_{nngp}	-1.969	-1.446	-0.817	-0.215	1.072	1.319
KMeansPP-TP	k_{lin}	-1.968	-1.441	-0.816	-0.212	1.077	0.252
LCMD-TP (ours)	$k_{\text{grad}\to\text{ens}(3)\to\text{sketch}(512)}$	-2.040	**-1.594**	**-0.947**	-0.408	0.920	1.073
LCMD-TP (ours)	k_{grad}	-2.038	**-1.594**	-0.946	-0.408	0.908	2.714
LCMD-TP (ours)	$k_{\text{grad}\to\text{sketch}(512)}$	-2.033	-1.590	-0.941	-0.404	0.917	0.981
LCMD-TP (ours)	$k_{\text{ll}\to\text{ens}(3)\to\text{sketch}(512)}$	-2.026	-1.555	-0.912	-0.358	0.952	0.984
LCMD-P (ours)	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf}(512)}$	-1.992	-1.547	-0.905	-0.369	0.974	0.875
LCMD-TP (ours)	k_{ll}	-2.017	-1.544	-0.902	-0.346	0.961	0.948
LCMD-P (ours)	$k_{\text{grad}\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-1.940	-1.534	-0.869	-0.371	0.864	0.774
LCMD-P (ours)	$k_{\text{ll}\to\text{acs-rf}(512)}$	-1.969	-1.513	-0.885	-0.331	1.010	0.473
LCMD-P (ours)	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf-hyper}(512)}$	-1.898	-1.501	-0.827	-0.336	0.878	0.876
LCMD-P (ours)	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-grad}}$	-1.896	-1.496	-0.827	-0.334	0.880	0.899
LCMD-P (ours)	$k_{\text{ll}\to\mathcal{X}_{\text{train}}}$	-1.920	-1.485	-0.840	-0.311	0.935	0.372
LCMD-TP (ours)	k_{nngp}	-1.960	-1.447	-0.811	-0.216	1.043	1.587
LCMD-TP (ours)	k_{lin}	-1.958	-1.445	-0.810	-0.215	1.036	0.525

Table 6.E.5: This table shows the performance and runtime of different combinations of selection methods and kernels. The columns labeled "MAE" to "MAXE" contain averaged logarithmic values of the corresponding metrics, averaged over all data sets, repetitions, and BMAL steps. For ensembled kernels, the metrics of the individual ensemble members were averaged to isolate the effect of ensembling on the batch selection. Runtimes were measured at one of the 20 repetitions where only one process was started per GPU, and are averaged over all BMAL steps and data sets. The employed hardware is described in Appendix 6.E.

Selection method	Kernel	MAE	RMSE	95%	99%	MAXE	avg. time $[s]$
Random	—	-1.923	-1.406	-0.774	-0.178	1.119	0.001
MaxDiag	$k_{\text{grad}\to\text{ens}(3)\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-1.753	-1.358	-0.690	-0.205	0.956	1.393
MaxDiag	$k_{\text{grad}\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-1.751	-1.351	-0.682	-0.192	0.961	0.551
MaxDiag	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf}(512)}$	-1.749	-1.350	-0.680	-0.189	0.962	0.651
MaxDiag	$k_{\text{grad}\to\mathcal{X}_{\text{train}}}$	-1.749	-1.346	-0.677	-0.182	0.967	4.107
MaxDiag	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-grad}}$	-1.735	-1.342	-0.671	-0.184	0.961	0.553
MaxDiag	$k_{\text{nngp}\to\mathcal{X}_{\text{train}}}$	-1.768	-1.301	-0.634	-0.095	1.054	2.584
MaxDiag	$k_{\text{ll}\to\mathcal{X}_{\text{train}}}$	-1.744	-1.300	-0.631	-0.100	1.037	0.142
MaxDiag	$k_{\text{ll}\to\text{acs-rf}(512)}$	-1.713	-1.286	-0.606	-0.084	1.052	0.031
MaxDiag	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf-hyper}(512)}$	-1.675	-1.279	-0.604	-0.111	1.007	0.649
MaxDiag	$k_{\text{lin}\to\mathcal{X}_{\text{train}}}$	-1.622	-1.158	-0.483	0.068	1.156	0.007
MaxDet-TP	$k_{\text{grad}\to\text{scale}(\mathcal{X}_{\text{train}})}$	-1.955	-1.546	-0.895	-0.400	0.834	8.914
MaxDet-P	$k_{\text{grad}\to\text{ens}(3)\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-1.913	-1.523	-0.867	-0.390	0.843	1.607
MaxDet-P	$k_{\text{grad}\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-1.915	-1.520	-0.864	-0.381	0.846	0.770
MaxDet-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf}(512)}$	-1.910	-1.514	-0.857	-0.372	0.855	0.868
MaxDet-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-grad}}$	-1.886	-1.503	-0.840	-0.370	0.846	0.890
MaxDet-P	$k_{\text{ll}\to\text{ens}(3)\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-1.921	-1.485	-0.823	-0.304	0.894	0.675
MaxDet-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf-hyper}(512)}$	-1.847	-1.468	-0.804	-0.335	0.863	0.871
MaxDet-P	$k_{\text{ll}\to\mathcal{X}_{\text{train}}}$	-1.910	-1.467	-0.807	-0.281	0.913	0.371
MaxDet-P	$k_{\text{ll}\to\text{acs-rf}(512)}$	-1.905	-1.459	-0.801	-0.272	0.924	0.470
MaxDet-TP	$k_{\text{nngp}\to\text{scale}(\mathcal{X}_{\text{train}})}$	-1.869	-1.401	-0.737	-0.198	0.980	5.456
MaxDet-P	$k_{\text{lin}\to\mathcal{X}_{\text{train}}}$	-1.714	-1.239	-0.574	-0.020	1.118	0.170
Bait-F-P	$k_{\text{grad}\to\text{ens}(3)\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-2.025	-1.594	-0.948	**-0.436**	0.855	2.342
Bait-F-P	$k_{\text{grad}\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-2.023	-1.588	-0.943	-0.429	0.859	1.505
Bait-FB-P	$k_{\text{grad}\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-2.019	-1.585	-0.938	-0.426	0.853	3.051
Bait-F-P	$k_{\text{ll}\to\text{ens}(3)\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-2.002	-1.541	-0.885	-0.345	0.894	1.407
Bait-F-P	$k_{\text{ll}\to\mathcal{X}_{\text{train}}}$	-1.989	-1.524	-0.869	-0.324	0.913	1.147
Bait-FB-P	$k_{\text{ll}\to\mathcal{X}_{\text{train}}}$	-1.989	-1.522	-0.868	-0.323	0.917	2.733
Bait-F-P	$k_{\text{lin}\to\mathcal{X}_{\text{train}}}$	-1.742	-1.264	-0.601	-0.045	1.110	0.233
FrankWolfe-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf-hyper}(512)}$	-1.961	-1.529	-0.882	-0.363	0.922	0.822
FrankWolfe-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf}(512)}$	-1.977	-1.502	-0.864	-0.299	1.015	0.824
FrankWolfe-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-grad}\to\text{sketch}(512)}$	-1.960	-1.481	-0.840	-0.279	1.049	0.915
FrankWolfe-P	$k_{\text{ll}\to\text{acs-rf}(512)}$	-1.959	-1.467	-0.830	-0.255	1.031	0.422
FrankWolfe-P	$k_{\text{ll}\to\text{acs-rf-hyper}(512)}$	-1.923	-1.437	-0.797	-0.227	1.016	0.420
FrankWolfe-P	$k_{\text{ll}\to\text{acs-grad}\to\text{sketch}(512)}$	-1.911	-1.413	-0.771	-0.191	1.092	0.519
FrankWolfe-P	$k_{\text{grad}\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-1.881	-1.384	-0.736	-0.157	1.129	0.725
FrankWolfe-P	$k_{\text{ll}\to\mathcal{X}_{\text{train}}}$	-1.859	-1.358	-0.711	-0.128	1.143	0.324
MaxDist-TP	$k_{\text{grad}\to\text{ens}(3)\to\text{sketch}(512)}$	-1.919	-1.540	-0.889	-0.416	**0.818**	0.744
MaxDist-TP	k_{grad}	-1.919	-1.539	-0.888	-0.413	0.823	2.351
MaxDist-TP	$k_{\text{grad}\to\text{sketch}(512)}$	-1.918	-1.536	-0.886	-0.409	0.820	0.652
MaxDist-TP	$k_{\text{ll}\to\text{ens}(3)\to\text{sketch}(512)}$	-1.958	-1.526	-0.879	-0.360	0.876	0.655
MaxDist-P	$k_{\text{grad}\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-1.913	-1.518	-0.861	-0.378	0.849	0.712
MaxDist-TP	k_{ll}	-1.951	-1.515	-0.870	-0.346	0.888	0.621
MaxDist-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf}(512)}$	-1.906	-1.508	-0.851	-0.364	0.865	0.811
MaxDist-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-grad}}$	-1.890	-1.507	-0.844	-0.373	0.843	0.833
MaxDist-P	$k_{\text{ll}\to\mathcal{X}_{\text{train}}}$	-1.913	-1.468	-0.810	-0.282	0.917	0.311
MaxDist-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf-hyper}(512)}$	-1.846	-1.462	-0.802	-0.327	0.868	0.810
MaxDist-P	$k_{\text{ll}\to\text{acs-rf}(512)}$	-1.899	-1.452	-0.794	-0.264	0.937	0.409
MaxDist-TP	k_{lin}	-1.905	-1.435	-0.777	-0.233	0.982	0.238
MaxDist-TP	k_{nngp}	-1.897	-1.426	-0.768	-0.224	0.989	1.286
KMeansPP-TP	$k_{\text{grad}\to\text{ens}(3)\to\text{sketch}(512)}$	-2.022	-1.566	-0.931	-0.382	0.971	0.755
KMeansPP-TP	k_{grad}	-2.023	-1.566	-0.932	-0.381	0.969	2.360
KMeansPP-TP	$k_{\text{grad}\to\text{sketch}(512)}$	-2.022	-1.566	-0.932	-0.381	0.969	0.662
KMeansPP-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf}(512)}$	-2.001	-1.558	-0.914	-0.381	0.920	0.837
KMeansPP-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-grad}}$	-2.006	-1.542	-0.902	-0.351	0.972	0.858
KMeansPP-P	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf-hyper}(512)}$	-1.987	-1.541	-0.898	-0.363	0.972	0.836
KMeansPP-P	$k_{\text{grad}\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-2.006	-1.531	-0.895	-0.334	0.996	0.736
KMeansPP-TP	$k_{\text{ll}\to\text{ens}(3)\to\text{sketch}(512)}$	-2.011	-1.525	-0.894	-0.325	1.000	0.665
KMeansPP-TP	k_{ll}	-2.007	-1.523	-0.889	-0.320	1.010	0.632
KMeansPP-P	$k_{\text{ll}\to\text{acs-rf}(512)}$	-1.982	-1.501	-0.861	-0.299	0.985	0.434
KMeansPP-P	$k_{\text{ll}\to\mathcal{X}_{\text{train}}}$	-1.980	-1.484	-0.849	-0.275	1.020	0.335
KMeansPP-TP	k_{nngp}	-1.968	-1.465	-0.829	-0.246	1.051	1.298
KMeansPP-TP	k_{lin}	-1.966	-1.464	-0.827	-0.248	1.057	0.249
LCMD-TP (ours)	$k_{\text{grad}\to\text{ens}(3)\to\text{sketch}(512)}$	**-2.041**	**-1.600**	**-0.961**	-0.425	0.901	1.071
LCMD-TP (ours)	k_{grad}	-2.038	-1.598	-0.958	-0.423	0.899	2.712
LCMD-TP (ours)	$k_{\text{grad}\to\text{sketch}(512)}$	-2.038	-1.597	-0.957	-0.422	0.898	0.977
LCMD-TP (ours)	$k_{\text{ll}\to\text{ens}(3)\to\text{sketch}(512)}$	-2.027	-1.554	-0.916	-0.359	0.941	0.979
LCMD-TP (ours)	k_{ll}	-2.022	-1.550	-0.911	-0.357	0.944	0.946
LCMD-P (ours)	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf}(512)}$	-1.982	-1.532	-0.889	-0.348	0.945	0.874
LCMD-P (ours)	$k_{\text{grad}\to\text{sketch}(512)\to\mathcal{X}_{\text{train}}}$	-1.930	-1.531	-0.880	-0.392	0.847	0.774
LCMD-P (ours)	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-grad}}$	-1.907	-1.520	-0.862	-0.389	0.849	0.898
LCMD-P (ours)	$k_{\text{grad}\to\text{sketch}(512)\to\text{acs-rf-hyper}(512)}$	-1.905	-1.508	-0.856	-0.371	0.870	0.877
LCMD-TP (ours)	k_{lin}	-1.971	-1.484	-0.836	-0.274	0.991	0.517
LCMD-TP (ours)	k_{nngp}	-1.970	-1.482	-0.834	-0.271	0.996	1.560
LCMD-P (ours)	$k_{\text{ll}\to\mathcal{X}_{\text{train}}}$	-1.928	-1.481	-0.825	-0.296	0.919	0.370
LCMD-P (ours)	$k_{\text{ll}\to\text{acs-rf}(512)}$	-1.963	-1.476	-0.840	-0.272	0.986	0.475

Table 6.E.6: This table shows the performance and runtime of different combinations of selection methods and kernels for the SiLU activation function. The columns labeled "MAE" to "MAXE" contain averaged logarithmic values of the corresponding metrics, averaged over all data sets, repetitions and BMAL steps. For ensembled kernels, the metrics of the individual ensemble members were averaged to isolate the effect of ensembling on the batch selection. Runtimes were measured at one of the 20 repetitions where only one process was started per GPU, and are averaged over all BMAL steps and data sets. The employed hardware is described in Appendix 6.E.

Chapter 7

Convergence Rates for Non-Log-Concave Sampling and Log-Partition Estimation

David Holzmüller[1] and Francis Bach[2]

Preprint available on arXiv
Reference: Holzmüller and Bach (2023), link: `https://arxiv.org/abs/2303.03237`

Abstract

Sampling from Gibbs distributions $p(x) \propto \exp(-V(x)/\varepsilon)$ and computing their log-partition function are fundamental tasks in statistics, machine learning, and statistical physics. However, while efficient algorithms are known for convex potentials V, the situation is much more difficult in the non-convex case, where algorithms necessarily suffer from the curse of dimensionality in the worst case. For optimization, which can be seen as a low-temperature limit of sampling, it is known that smooth functions V allow faster convergence rates. Specifically, for m-times differentiable functions in d dimensions, the optimal rate for algorithms with n function evaluations is known to be $O(n^{-m/d})$, where the constant can potentially depend on m, d and the function to be optimized. Hence, the curse of dimensionality can be alleviated for smooth functions at least in terms of the convergence rate. Recently, it has been shown that similarly fast rates can also be achieved with polynomial runtime $O(n^{3.5})$, where the exponent 3.5 is independent of m or d. Hence, it is natural to ask whether similar rates for sampling and log-partition computation are possible, and whether they can be realized in polynomial time with an exponent independent of m and d. We show that the optimal rates for sampling and log-partition computation are sometimes equal and sometimes faster than for optimization. We then analyze various polynomial-time sampling algorithms, including an extension of a recent promising optimization approach, and find that they sometimes exhibit interesting behavior but no near-optimal rates. Our results also give further insights on the relation between sampling, log-partition, and optimization problems.

[1]ISA, University of Stuttgart, Stuttgart, Germany
[2]SIERRA, INRIA, Paris, France

7.1 Introduction

The tasks of sampling from a Gibbs distribution with density $p(x) \propto \exp(-V(x)/\varepsilon)$ and computing the corresponding normalization constant are important problems in many computational fields such as machine learning, (Bayesian) statistics and statistical physics. Specifically, we are interested in the following setting:

Definition 7.1.1 (Sampling and log-partition problems)**.** Let $d \geq 1$, let $\mathcal{X} = [0,1]^d$ and let $f : \mathcal{X} \to \mathbb{R}$ be bounded and measurable.[3] The *sampling problem* is to draw samples from the distribution P_f on $\mathcal{X}$ with density

$$p_f(x) \coloneqq \frac{\exp(f(x))}{Z_f} ,$$

where $Z_f \coloneqq \int_{\mathcal{X}} \exp(f(x)) \,\mathrm{d}x$ is the normalization constant or *partition function.* The *log-partition problem* is to compute the *log-partition function*

$$L_f \coloneqq \log Z_f = \log\left(\int_{\mathcal{X}} \exp(f(x)) \,\mathrm{d}x\right) . \qquad \blacktriangleleft$$

Distributions of the form P_f are known as *Gibbs distributions*, *Gibbs measures*, or *Boltzmann distributions.* They arise for example in statistical physics with $f(x) = -V(x)/\varepsilon$, where $V(x)$ denotes the (potential) energy of state x, and ε is (proportional to) the temperature of the system.[4] Instead of the temperature, sometimes the inverse temperature (or coldness / thermodynamic beta) $\beta = 1/\varepsilon$ is used. The log-partition problem is also related to the computation of the free energy $-\varepsilon L_{-V/\varepsilon}$. In energy-based models in ML, f could be learned. In a Bayesian statistical or ML model with parameters θ and data D, we could set $f(\theta) \coloneqq \log p(D|\theta) + \log p(\theta)$ to sample from the posterior distribution $p_f(\theta) = p(\theta|D)$ or compute the log-evidence $L_f = \log p(D) = \log\left(\int_{\mathcal{X}} p(D|\theta)p(\theta)\,\mathrm{d}\theta\right)$. In some contexts, L_f is also called the log-marginal likelihood, which is useful for model selection (Robert, 2007).

While exact sampling and log-partition computation are possible for some simple functions f like linear functions, we can usually only expect algorithms to obtain approximations within a limited runtime. Therefore, we are interested in how fast this approximation converges to the true P_f or L_f in terms of the number n of times that the algorithm is allowed to evaluate f. To study this, we need to make some assumptions on f. While efficient sampling algorithms for suitable classes of concave f are known, at least with access to gradients of f (Dwivedi et al., 2018; Mangoubi and Vishnoi, 2018; Chewi et al., 2021; Altschuler and Talwar, 2022), we are interested in larger classes of non-concave functions, which are defined in the following:

Definition 7.1.2 (Further notation)**.** For measurable functions $f : \mathcal{X} \to \mathbb{R}$, we use the notation $\|f\|_\infty \coloneqq \operatorname{ess\,sup}_{x \in \mathcal{X}} |f(x)|$. We define function spaces of m-times continuously

[3]We choose $\mathcal{X}$ as the unit cube for convenience: It is compact, has unit volume, does not have too sharp corners, there are well-studied approximation results, and it allows to investigate algorithms for periodic functions. However, many of our results could be generalized to other domains.

[4]Technically, $\varepsilon = k_B T$, where k_B is Boltzmann's constant and T is the temperature.

differentiable functions[5] whose derivatives are bounded by some constant $B \geq 0$:

$$\mathcal{F}_{d,m,B} \coloneqq \{f \in C^m(\mathcal{X}), \|f\|_{C^m} \leq B\}, \qquad \|f\|_{C^m} \coloneqq \sup_{\alpha \in \mathbb{N}_0^d : |\alpha|_1 \leq m} \|\partial_\alpha f\|_\infty .$$

Here, we use the notation $|\alpha|_1 \coloneqq \alpha_1 + \cdots + \alpha_d$ and $\partial_\alpha f = \frac{\partial^{|\alpha|_1} f}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}}$. Moreover, if f is Lipschitz, we denote its minimal Lipschitz constant by $|f|_1$. If f is bounded, we denote its maximum by M_f. We define $\bar{f} \coloneqq f - L_f$, such that $L_{\bar{f}} = 0$ and $P_{\bar{f}} = P_f$. Finally, we denote the uniform distribution on $\mathcal{X}$ by $\mathcal{U}(\mathcal{X})$. ◀

We study the worst-case error of algorithms over the function class $\mathcal{F}_{d,m,B}$, which is formally defined in Section 7.2. This error depends on the variables (B, n, d, m). We study the asymptotic behavior in terms of n and B while ignoring constants depending only on m and d for simplicity. Depending on the definition of the norm, such constants are often necessarily exponential in d and represent the part of the curse of dimensionality that cannot be overcome in this setting (Novak and Woźniakowski, 2009). For example, typical convergence rates for function approximation are of the form $O_{m,d}(Bn^{-m/d})$, which we sometimes also write as $O_{m,d}(\|f\|_{C^m} n^{-m/d})$ (Novak, 1988; Wendland, 2004). As we will see later, the dependence on B is not always linear, and tracking the dependence on B is important since the function f appears inside an exponential. When using asymptotic notation like $O_{m,d}$, we mean that the corresponding inequality should hold for *all* values of $n \in \mathbb{N}_{\geq 1}$ and $B > 0$, not only large enough values.[6]

We express bounds on the error E achieved for n function evaluations, such as $E = O_{m,d}(Bn^{-m/d})$. Some authors prefer to express rates in terms of the number of function evaluations needed to reach an error E or lower, which would then for example be $n = O_{m,d}((B/E)^{d/m})$.

Sometimes, we explicitly include a temperature $\varepsilon > 0$ and formulate our theorems in terms of f/ε instead of f. It is well-known that in the limit of low temperatures ($\varepsilon \searrow 0$), sampling becomes essentially equivalent to optimization. Here, we give a quantitative version of this statement:

Lemma 7.1.3 (Optimization limit). *Let $f : \mathcal{X} \to \mathbb{R}$ be Lipschitz-continuous with Lipschitz constant $|f|_1 < \infty$. Then, for any temperature $\varepsilon > 0$, the maximum $M_f = \max_{x \in \mathcal{X}} f(x)$ satisfies*

$$|M_f - \varepsilon L_{f/\varepsilon}| \leq \varepsilon d \log(1 + 3d^{-1/2}|f|_1/\varepsilon) \longrightarrow 0 \quad \text{for } \varepsilon \searrow 0 . \tag{7.1}$$

Moreover, for any bounded and measurable $f : \mathcal{X} \to \mathbb{R}$ and any $\delta \in (0, 1]$, we have

$$P_{f/\varepsilon}(\{x \in \mathcal{X} \mid f(x) < \varepsilon L_{f/\varepsilon} - \varepsilon \log(1/\delta)\}) \leq \delta .$$

[5] It would also be possible to replace $C^m(\mathcal{X})$ with the slightly larger Sobolev space $W^{m,\infty}(\mathcal{X})$.

[6] Specifically, we use the notation $g(B, n, d, m) \leq O_{m,d}(h(B, n, d, m))$ to mean

$$\forall d, m \in \mathbb{N}_{\geq 1} \exists C_{m,d} > 0 \forall B > 0, n \in \mathbb{N}_{\geq 1} : g(B, n, d, m) \leq C_{m,d} h(B, n, d, m) ,$$

and we similarly write $g \geq \Omega_{m,d}(h)$ for $h \leq O_{m,d}(g)$, as well as $g = \Theta_{m,d}(h)$ for $g \leq O_{m,d}(h)$ and $h \leq O_{m,d}(g)$.

Lemma 7.1.3 is proven in Appendix 7.B and can be related to our function classes by using $|f|_1 \leq d^{1/2}\|f\|_{C^1}$, cf. Lemma 7.B.1. Note that such a convergence result does not hold for general bounded functions, as can be seen for the characteristic function $f = \mathbb{1}_{\{0\}}$, which satisfies $M_f = 1$ but $\varepsilon L_{f/\varepsilon} = 0$ for all $\varepsilon > 0$. Eq. (7.1) is related to Corollary 1 of Ma et al. (2019), which shows that in order to achieve an optimization error of E through sampling, it is necessary that $1/\varepsilon = \widetilde{\Omega}(d/E)$.

Hwang (1980) uses Laplace's method to show that under certain assumptions on the Hessians at the maximizers, $P_{f/\varepsilon}$ converges weakly to a distribution on the maximizers. In contrast to optimization, where it typically does not matter which global maximum is found, the low-temperature limit of sampling often yields a unique distribution on the maximizers. Talwar (2019) uses this to show that optimization can be easier than sampling for very particular classes of functions. It is known that the optimal worst-case convergence rate for optimization on $\mathcal{F}_{d,m,B}$ is the same as for approximation, i.e., $O_{m,d}(Bn^{-m/d})$ (see Novak, 1988, and references therein). Recently, Rudi et al. (2020) and Woodworth et al. (2022) have shown polynomial-time optimization algorithms (in n and d) achieving rates close to the optimal rate under relatively mild additional assumptions.

In the high-temperature limit $\varepsilon \to \infty$, $P_{f/\varepsilon}$ converges to a uniform distribution, and Ma et al. (2019) showed that the Metropolis-adjusted Langevin algorithm (MALA) can achieve exponentially fast convergence rates in n, although with exponential dependence on the Lipschitz constant of ∇f. While MALA theoretically does not fit in our framework since it uses gradient information of f, this could be emulated using numerical differentiation, and we show in Section 7.4.2 that in our setting, if B is known, a fixed-budget version of rejection sampling can also achieve similar rates (see also Talwar, 2019).

Since we know that polynomial-time algorithms with fast convergence rates are possible for the high-temperature case and for optimization, which is essentially the low-temperature limit, this poses the question of whether we can find such algorithms for the general sampling and log-partition problems. Ideally, such an algorithm should have the following properties:

- A convergence rate close to the optimal convergence rate $O_{m,d}(\|f\|_{C^m} n^{-m/d})$ for approximation, at least up to $m \geq \Omega(d)$, such that the exponent in the rate does not approach zero for large d,
- Polynomial runtime $O(n^k)$ for some k independent of d and m. Especially, the runtime should be polynomial in d and m. Moreover, the runtime should not depend on $\|f\|$.
- Adaptivity: The algorithm should achieve these rates without knowing m and $\|f\|$. This is not investigated here, and for sampling and log-partition estimation, m can often be known.

Regarding the implications of achieving near-optimal rates in polynomial time, consider the following example:

Example 7.1.4. Consider a Bayesian model where the data set $D = (D_1, \dots, D_N)$ consists of N observed samples that are assumed to be drawn in an i.i.d. fashion. Then, we can again model

$$f(\theta) := \log p(\theta, D) = \log p(D_1 \mid \theta) + \cdots + \log p(D_N \mid \theta) + \log p(\theta) ,$$

which is a sum of $N + 1$ functions. Hence, we would expect that $\|f\|_{C^m}$ scales like $\Theta(N)$.

(a) Suppose that we have a log-partition method with rate $\Theta_{m,d}(\|f\|_{C^m} n^{-m/d})$ and polynomial runtime $O_{m,d}(n^k)$. To achieve an error of $O(1)$ for the log-evidence L_f, this method would need $n = \Theta_{m,d}(N^{d/m})$ function evaluations and hence a runtime of $O_{m,d}(N^{kd/m})$. If $m = d$, the exponent kd/m is independent of the dimension d.

(b) Now, suppose instead that the runtime is of the form $O_{m,d}(n^m)$ or the rates are of the form $O_{m,d}(\|f\|_{C^m} n^{-1/d})$ or $O_{m,d}(\|f\|_{C^m}^m n^{-m/d})$. In each case, to achieve an error of $O(1)$ for the log-evidence L_f, the resulting runtime would be polynomial in N, but the exponent would be proportional to d. ◀

7.1.1 Contribution

Our contributions are as follows:

(1) We analyze the information-based complexity of the sampling and log-partition problems, i.e., the worst-case optimal rates without computational constraints, in Section 7.2. For algorithms that evaluate f at a deterministic set of points, we show that the optimal rate for the log-partition problem is $\Theta_{m,d}(Bn^{-m/d})$, i.e., the same as for approximation. For the bounded total variation and 1-Wasserstein metrics, the optimal rate for sampling is $\Theta_{m,d}(\min\{1, Bn^{-m/d}\})$. For algorithms that are allowed to evaluate f at a stochastic set of points, we show that the optimal rates are the same in an optimization regime but can be faster in the high-temperature regime.

(2) We show reductions between different problems. For example, we analyze how log-partition algorithms can be employed for sampling and vice versa, and analyze the resulting guarantees for the rates. We also discuss how approximate sampling algorithms can be employed for optimization. Moreover, we show how function approximation yields reductions between different runtime complexities, convergence rates, and from stochastic to deterministic evaluation points.

(3) We analyze bounds on the convergence rates for different algorithms. For example, we show that it is possible to achieve the rate $O_{m,d}(Bn^{-m/d})$, but with runtime $O(n^m)$, which is polynomial but still involves the curse of dimensionality since we need $m = \Theta(d)$ to beat the curse of dimensionality in the convergence rate. We show that other simple and efficient algorithms also fail to achieve the optimal rates in different ways, sometimes with multi-regime behavior. Finally, we study an approach toward the log-partition problem by Bach (2022), whose optimization limit has been used by Woodworth et al. (2022) to obtain near-optimal optimization rates in polynomial-time. We show that all versions of this approach necessarily fail to exceed the rate $O_{m,d}(Bn^{-2/d})$ in an intermediate temperature regime $\varepsilon \sim n^{-2/d}$ (corresponding to $B \sim n^{2/d}$).

7.1.2 Related Work

The analysis of sampling algorithms has received considerable attention in recent years. In the case where p_f is (strongly) log-concave, that is, if f is (strongly) concave, convergence rates of Markov chain Monte Carlo (MCMC) sampling algorithms have been studied extensively. For example, good convergence rates in terms of the dimension d have been established for versions of the Langevin algorithm (Chewi et al., 2021; Altschuler and Talwar, 2022) and Hamiltonian Monte Carlo (Mangoubi and Vishnoi, 2018). Chewi et al.

(2022) establish an algorithm with optimal convergence rate for the case $d = 1$, while not much is known about algorithm-independent lower bounds in other cases.

For sampling from more general non-log-concave distributions, convergence rates have been established for versions of the Langevin algorithm. Bou-Rabee and Hairer (2013) showed an essentially geometric convergence result in TV distance for a class of non-log-concave Gibbs distributions, but without clear dependence of the constants on f. Mangoubi and Vishnoi (2019) and Zou et al. (2021) prove convergence rates that are polynomial in d but additionally depend on properties of f through the Cheeger constant. The analysis of Ma et al. (2019) and Cheng et al. (2018) is closer to our setting, and their convergence rate is polynomial in d as well, but their rate exhibits an exponential dependence on the Lipschitz constant of ∇f and the radius of the domain where f is non-log-convex. Bou-Rabee et al. (2020) obtain similar results for Hamiltonian Monte Carlo. Balasubramanian et al. (2022) show that even for non-log-concave distributions, averaged Langevin Monte Carlo converges quickly to a distribution with low relative Fisher information to the target distribution, although this does not imply that the distribution is close to the target distribution with respect to other measures such as the total variation distance. Chewi et al. (2023) prove corresponding lower bounds. Woodard et al. (2009) show that the mixing time of parallel and simulated tempering for certain distributions can scale exponentially with d, but in a setting different from ours. Achddou et al. (2019) propose and analyze an adaptive rejection sampling algorithm using a piecewise constant approximation of the density. Their setting is significantly different from ours as well, and they only consider functions of low (Hölder) smoothness and regimes with large n. Marteau-Ferey et al. (2022) propose an approximation-based sampling algorithm with a rate similar to $O_{m,d,B}(n^{-m/d})$ but without analyzing the dependence on B.

Another related line of work studies the relation of sampling to optimization. Through their analysis of Langevin algorithms in the non-log-concave setting, Ma et al. (2019) show that there are settings where sampling is easier than optimization. Talwar (2019) provides a simpler argument and shows that the converse can also occur for special function classes. The relation between sampling and optimization is also exploited in simulated annealing (Kirkpatrick et al., 1983). A different connection between sampling and optimization stems from Jordan et al. (1998), who showed that Langevin-type sampling can be interpreted as a gradient flow over distributions for the Wasserstein metric. For an overview of connections between sampling and optimization, we also refer to Cheng (2020).

The log-partition problem is often addressed through sampling algorithms, for example via thermodynamic integration (Kirkwood, 1935). For an overview of thermodynamic integration and other methods for the log-partition problem, we refer to Gelman and Meng (1998) and Friel and Wyse (2012). Ge et al. (2020) analyze an annealing algorithm combined with multilevel Monte Carlo sampling for the log-partition problem in the log-concave setting, and also give an information-based lower bound on the achievable convergence rate. Another popular approach is the Laplace approximation (Laplace, 1774), whose log-partition function however does not converge to the true log-partition function as $n \to \infty$. Well-tempered metadynamics (Barducci et al., 2008) is a popular approach towards the log-partition problem in molecular dynamics simulations, although it relies on a well-chosen low-dimensional collective variable representation. Recently, Marteau-Ferey et al. (2022) have suggested an approach that performs sampling via estimating the log-partition function. Bach (2023) and Bach (2022) suggest further approaches toward solving the log-partition problem.

To analyze possible convergence rates for the sampling and log-partition problem without computational constraints, we use the framework of information-based complexity. Here, we refer to Novak (1988) and Traub (2003) for an overview of this topic. In particular, our work is motivated by the works of Rudi et al. (2020) and Woodworth et al. (2022), who demonstrated that for optimization, convergence rates close to the optimal rates from information-based complexity can be achieved in polynomial time.

The rest of our paper is organized as follows: In Section 7.2, we study upper and lower bounds for the information-based complexity of different variants of the sampling and log-partition problems. In Section 7.3, we study relations and reductions between different variants of the sampling, log-partition, and optimization problems. We then study convergence rates of different algorithms in Section 7.4. We compare some of these algorithms experimentally in Section 7.5 before concluding in Section 7.6. All proofs are provided in the appendix, which is structured analogously to the main part of this paper, and whose structure is overviewed in Appendix 7.A.

7.2 Information-based Complexity

In this section, we look at the log-partition and sampling problems from the viewpoint of (worst-case) *information-based complexity*, where one is interested in what is possible if one is not constrained computationally but only by the number n of function evaluations of the unknown function f. We adopt the general setting of Novak (1988), where one is given a function space $\mathcal{F}$ (such as $\mathcal{F}_{d,m,B}$) of functions $f : \mathcal{X} \to \mathbb{R}$ and wishes to approximate a map $S : \mathcal{F} \to \mathcal{M}$, with the approximation error on $\mathcal{M}$ measured by a metric D. For example, the following problems are considered by Novak (1988):

- *Approximation*: $S_{\mathrm{app}}(f) := f$ and $D_\infty(f, g) := \|f - g\|_\infty$.
- *Optimization*: $S_{\mathrm{opt}^*}(f) := \sup_{x\in\mathcal{X}} f(x)$ and $D_{\mathrm{abs}}(a, b) := |a - b|$.
- *Integration*: $S_{\mathrm{int}}(f) := \int_{\mathcal{X}} f(x)\,\mathrm{d}x$ and $D_{\mathrm{abs}}(a, b) = |a - b|$.

We can define our sampling and log-partition problems in this context as follows:

- *Log-partition*: $S_L(f) = L_f$ and $D_{\mathrm{abs}}(a, b) = |a - b|$.
- *Sampling*: While a sampling algorithm produces samples, we do not want to compare errors of individual samples but the error of the distribution of the samples. Therefore, we set $S_{\mathrm{samp}}(f) := P_f$. For $D(P, Q)$, we can use different metrics or divergences on probability distributions, which will be discussed in Section 7.2.1.

7.2.1 Deterministic Evaluation Points

To consider minimax optimal convergence rates, we still need to define a space $\mathcal{A}$ of admissible maps $\tilde{S} : \mathcal{F} \to \mathcal{M}$. Here, we will first consider maps that evaluate functions in a deterministic set of points, before considering stochastic points in Section 7.2.2. For example, we define

$$\mathcal{A}_n := \left\{\tilde{S} = \phi \circ N \mid N(f) = (f(x_1), \ldots, f(x_n)) \text{ for some } x_1, \ldots, x_n \in \mathcal{X}\right\} ,$$

the set of maps that only evaluate f in n deterministic and non-adaptive points. We can also allow adaptive points by defining

$$\mathcal{A}_n^{\mathrm{ad}} := \left\{\tilde{S} = \phi \circ N \mid N(f) = (f(x_1), f(x_2(f(x_1))), \ldots, f(x_n(f(x_1), \ldots, f(x_{n-1}))))\right\} ,$$

where evaluation points may be chosen depending on previous function values. We are interested in the (non-adaptive/adaptive) minimax optimal error

$$e_n(\mathcal{F}, S, D) := \inf_{\tilde{S} \in \mathcal{A}_n} \sup_{f \in \mathcal{F}} D(S(f), \tilde{S}(f)), \qquad e_n^{\mathrm{ad}}(\mathcal{F}, S, D) := \inf_{\tilde{S} \in \mathcal{A}_n^{\mathrm{ad}}} \sup_{f \in \mathcal{F}} D(S(f), \tilde{S}(f)) .$$

The sets $\mathcal{A}_n$ and $\mathcal{A}_n^{\mathrm{ad}}$ can be interpreted as classes of "black-box algorithms" that are only constrained in the number evaluations of f but not in terms of computational efficiency or computability. The minimax-optimal errors e_n and e_n^{ad} thus give lower bounds to what can be achieved by computationally efficient algorithms.

For the case of sampling, maps $\tilde{S} \in \mathcal{A}_n$ (or $\mathcal{A}_n^{\mathrm{ad}}$) produce distributions based on n function evaluations of a function f. They correspond to idealized sampling algorithms in the following sense: We consider an idealized sampling algorithm to take some source of randomness ω sampled from a distribution P_Ω independent of f, and then output a random sample $X_f(\omega) = \tilde{\phi}(N(f), \omega)$. For example, ω could be a sequence of i.i.d. random variables from the uniform distribution $\mathcal{U}([0,1])$ on the interval $[0,1]$. The maps $\tilde{S} \in \mathcal{A}_n$ (or $\mathcal{A}_n^{\mathrm{ad}}$) then correspond to the distributions produced by such sampling algorithms, i.e.,

$$\tilde{S}(f) = \text{ distribution of } X_f(\omega) \text{ for } \omega \sim P_\Omega.$$

The following theorem, which is proven in Section 7.C.1, adapts known results on minimax optimal rates to our considered function spaces.

Theorem 7.2.1 (adapted from Novak (1988)). *We have*

$$\begin{aligned}
e_n(\mathcal{F}_{d,m,B}, S_{\mathrm{app}}, D_\infty) &= \Theta_{m,d}(Bn^{-m/d}), & e_n^{\mathrm{ad}}(\mathcal{F}_{d,m,B}, S_{\mathrm{app}}, D_\infty) &= \Theta_{m,d}(Bn^{-m/d}), \\
e_n(\mathcal{F}_{d,m,B}, S_{\mathrm{opt}^*}, D_{\mathrm{abs}}) &= \Theta_{m,d}(Bn^{-m/d}), & e_n^{\mathrm{ad}}(\mathcal{F}_{d,m,B}, S_{\mathrm{opt}^*}, D_{\mathrm{abs}}) &= \Theta_{m,d}(Bn^{-m/d}), \\
e_n(\mathcal{F}_{d,m,B}, S_{\mathrm{int}}, D_{\mathrm{abs}}) &= \Theta_{m,d}(Bn^{-m/d}), & e_n^{\mathrm{ad}}(\mathcal{F}_{d,m,B}, S_{\mathrm{int}}, D_{\mathrm{abs}}) &= \Theta_{m,d}(Bn^{-m/d}) .
\end{aligned}$$

Novak (1988) gives these results in a form similar to $e_n(\mathcal{F}_{d,m,1}, S_{\mathrm{app}}, D_\infty) = \Theta_{m,d}(n^{-m/d})$. This implies the rates for general $B \geq 0$ in the theorem above since S_{app} and D_∞ are positively homogeneous, which leads to $D_\infty(S_{\mathrm{app}}(Bf), Bg) = BD_\infty(S_{\mathrm{app}}(f), g)$. The same holds for optimization and integration, but not for log-partition estimation and sampling. Hence, for our considered problems, it is important to explicitly study the dependence on B, since it is not necessarily linear. The optimal rates for approximation can be achieved, for example, using piecewise polynomial interpolation, local polynomial reproductions, or moving least squares (Wendland, 2004), see also Theorem 7.3.1. The optimal rates for optimization and integration can be achieved by optimizing or integrating a corresponding approximation.

We use the following distance measures for probability distributions P, Q on $\mathcal{X}$:

- The sup-log distance $D_{\text{sup-log}}(P, Q) := \left\|\log\left(\frac{\mathrm{d}P}{\mathrm{d}Q}\right)\right\|_\infty$, where $\|\cdot\|_\infty$ is taken over $\mathcal{X}$, and $D_{\text{sup-log}}(P, Q) = \infty$ whenever P and Q are not both absolutely continuous

with respect to each other. The sup-log distance is a symmetrized version of the max-divergence $D_\infty(P||Q)$, which is used in differential privacy (Dwork et al., 2010) and is a special case of Rényi divergences for $\alpha = \infty$ (cf. Van Erven and Harremos, 2014). The sup-log distance is particularly well-suited to our setting, thanks to its relation to uniform approximation.
- The total variation distance $D_{\mathrm{TV}}(P, Q) \coloneqq \sup_{A \subseteq \mathcal{X} \text{ measurable}} |P(A) - Q(A)|$.
- The 1-Wasserstein distance $W_1(P, Q) \coloneqq \inf_{X \sim P, Y \sim Q} \mathbb{E}\|X - Y\|_2$, also known as Kantorovich–Rubinstein or earth mover distance.

The sup-log, total variation, and 1-Wasserstein distances are metrics.[7] We first show that these quantities can be bounded in terms of the approximation error:

Proposition 7.2.2 (Upper bounds via approximation). *For bounded and measurable $f, g : \mathcal{X} \to \mathbb{R}$, we have*

(a) $|L_f - L_g| \leq \|f - g\|_\infty$.
(b) $d^{-1/2} W_1(P_f, P_g) \leq D_{\mathrm{TV}}(P_f, P_g) \leq D_{\text{sup-log}}(P_f, P_g) \leq 2\|f - g\|_\infty$.

Proposition 7.2.2 is proven in Section 7.C.1. For the KL divergence, which we will not study further, we can leverage the results of Proposition 7.2.2 by using the trivial bound $D_{\mathrm{KL}}(P \parallel Q) \leq D_{\text{sup-log}}(P, Q)$ as well as the inequality $D_{\mathrm{KL}}(P \parallel Q) \leq D_{\text{sup-log}}(P, Q)(e^{D_{\text{sup-log}}(P,Q)} - 1)$ from Lemma III.2 of Dwork et al. (2010).

Theorem 7.2.1 and Proposition 7.2.2 lead to upper bounds on the minimax optimal rates. Combined with the trivial upper bound $D_{\mathrm{TV}}(P, Q) \leq 1$, these are optimal for the deterministic point setting:

Theorem 7.2.3 (Information-based complexity of sampling and log-partition with deterministic evaluation points). *We have*

$$\begin{aligned} e_n(\mathcal{F}_{d,m,B}, S_L, D_{\mathrm{abs}}) &= \Theta_{m,d}(Bn^{-m/d}), \\ e_n(\mathcal{F}_{d,m,B}, S_{\mathrm{samp}}, D_{\text{sup-log}}) &= \Theta_{m,d}(Bn^{-m/d}), \\ e_n(\mathcal{F}_{d,m,B}, S_{\mathrm{samp}}, D_{\mathrm{TV}}) &= \Theta_{m,d}(\min\{1, Bn^{-m/d}\}), \\ e_n(\mathcal{F}_{d,m,B}, S_{\mathrm{samp}}, W_1) &= \Theta_{m,d}(\min\{1, Bn^{-m/d}\}), \end{aligned}$$

and the same rates hold for adaptive points.

Theorem 7.2.3 is proven in Section 7.C.1. The minimax optimal rates for optimization can be related to those for approximation on a very general class of function spaces (Novak, 1988). For sampling, such a general relationship does not hold: For example, the set $\mathcal{F} \coloneqq \{f : \mathcal{X} \to \mathbb{R} \mid \|f\|_\infty \leq 1, \{x \mid f(x) \neq 0\} \text{ is finite}\}$ satisfies $e_n(\mathcal{F}, S_{\mathrm{app}}, D_\infty) = 1$ for all $n \in \mathbb{N}$, but all functions $f \in \mathcal{F}$ have the same distribution and the same log-partition function. However, our proofs for the lower bounds in Theorem 7.2.3 follow the general idea that underlies many lower bounds for Sobolev-type functions: place bumps with small support in regions that the algorithm does not query.

[7]For the sup-log distance, the triangle inequality follows from $\log\left(\frac{\mathrm{d}P}{\mathrm{d}R}\right) = \log\left(\frac{\mathrm{d}P}{\mathrm{d}Q}\right) + \log\left(\frac{\mathrm{d}Q}{\mathrm{d}R}\right)$.

7.2.2 Stochastic Evaluation Points

We also want to consider methods that are allowed to choose the points x_i stochastically, such as Monte-Carlo type methods (Metropolis and Ulam, 1949; Brooks et al., 2011). For the log-partition problem, we again follow Novak (1988) and define the set ${}^*C(\mathcal{A}_n^{\mathrm{ad}})$ of random variables $\tilde{S} : \Omega \to \mathcal{A}_n^{\mathrm{ad}}$ with given base distribution P_Ω with associated minimax optimal error[8]

$$ {}^*\sigma_n^{\mathrm{ad}}(\mathcal{F}, S, D) \coloneqq \inf_{(\tilde{S}, P_\Omega) \in {}^*C(\mathcal{A}_n^{\mathrm{ad}})} \sup_{f \in \mathcal{F}} \mathbb{E}_{\omega \sim P_\Omega} D(S(f), \tilde{S}(\omega)(f)) ~. $$

When applying this definition to sampling, a map $\tilde{S}$ would output a *random* distribution. However, the random samples produced by a sampling algorithm typically still follow a *fixed* distribution, regardless of whether the function f is evaluated in deterministically or randomly chosen points. Hence, the model in Eq. (7.2) is inadequate for sampling. Instead, we consider again idealized sampling algorithms using some randomness $\omega \sim \Omega$, but this time, we allow the function to be evaluated in randomly and adaptively chosen points, by considering random samples of the form $X_f(\omega) = \tilde{\phi}(N(f, \omega), \omega)$. We then denote the corresponding map from f to P_{X_f} by $\tilde{S}$ and define the set $\mathcal{A}_n^{\text{ad-stoch}}$ of all $\tilde{S}$ that can be realized in this fashion using n function evaluations. We then define

$$ e_n^{\text{ad-stoch}}(\mathcal{F}, S_{\mathrm{samp}}, D) \coloneqq \inf_{\tilde{S} \in \mathcal{A}_n^{\text{ad-stoch}}} \sup_{f \in \mathcal{F}} D(S_{\mathrm{samp}}(f), \tilde{S}(f)) ~. $$

Unlike the deterministic points setting, the stochastic points setting potentially requires to evaluate f at n different points for every generated sample. This has the unintuitive consequence that for a map $\tilde{S} \in \mathcal{A}_n^{\text{ad-stoch}}$, the distribution $\tilde{S}(f)$ typically depends on the values of f at infinitely many points, but a sample from $\tilde{S}(f)$ can be drawn by only evaluating f at n (stochastic) points.

Again, results for approximation, optimization, and integration are known and can be adapted to our function classes:

Theorem 7.2.4 (adapted from Novak (1988)). *We have*

$$ \begin{aligned} {}^*\sigma_n^{\mathrm{ad}}(\mathcal{F}_{d,m,B}, S_{\mathrm{app}}, D_\infty) &= \Theta_{m,d}(Bn^{-m/d}), \\ {}^*\sigma_n^{\mathrm{ad}}(\mathcal{F}_{d,m,B}, S_{\mathrm{opt}^*}, D_{\mathrm{abs}}) &= \Theta_{m,d}(Bn^{-m/d}), \\ {}^*\sigma_n^{\mathrm{ad}}(\mathcal{F}_{d,m,B}, S_{\mathrm{int}}, D_{\mathrm{abs}}) &= \Theta_{m,d}(Bn^{-1/2-m/d}) ~. \end{aligned} $$

For a proof sketch, we refer to Section 7.C.2. The faster rate for integration can be achieved by spending half of the n points for approximating f with g and spending the other half of the points on Monte Carlo quadrature to estimate the error (Novak, 1988)

$$ \int f(x) \,\mathrm{d}x - \int g(x) \,\mathrm{d}x = \mathbb{E}_{x \sim \mathcal{U}(\mathcal{X})}[f(x) - g(x)] ~. $$

For a more practical algorithm, we refer to Chopin and Gerber (2022). For log-partition estimation, we can similarly use an importance sampling formulation

$$ L_f - L_g = \log\left(\mathbb{E}_{x \sim P_g}\left[\exp(f(x) - g(x))\right]\right) ~. $$

[8]Novak (1988) defines further variants, for example with $L_2(P_\Omega)$ instead of $L_1(P_\Omega)$ convergence or more limited stochastic resources, which we will not discuss here for simplicity.

Theorem 7.2.5 (Upper bound for stochastic log-partition)**.** *There exists a constant $C_{m,d} > 0$ depending only on m and d such that*

$$^*\sigma_n^{\text{ad}}(\mathcal{F}_{d,m,B}, S_L, D_{\text{abs}}) \leq O_{m,d}\left(\min\left\{Bn^{-m/d}, \exp(C_{m,d}Bn^{-m/d})Bn^{-1/2-m/d}\right\}\right) \ .$$

The upper bound above, which is proven in Section 7.C.2, exhibits a fast transition between the rates $n^{-m/d}$ and $n^{-1/2-m/d}$. This is necessary, as we can exploit the relation of the log-partition problem to optimization to show that the rate $n^{-m/d}$ is optimal in an optimization regime:

Proposition 7.2.6 (Lower bound for stochastic log-partition)**.** *For $m \geq 1$, we have*

$$^*\sigma_n^{\text{ad}}(\mathcal{F}_{d,m,B}, S_L, D_{\text{abs}}) \geq \Omega_{m,d}(Bn^{-m/d}) - d\log(1+3B) \ .$$

Proposition 7.2.6 is proven in Section 7.C.2. We leave a lower bound outside of the optimization regime as an open problem, however, we conjecture that the rate $Bn^{-1/2-m/d}$ from the upper bound in Theorem 7.2.5 cannot be improved. For a certain class of strongly concave f with Lipschitz gradient, Theorem 5.1 by Ge et al. (2020) contains a lower bound which, in our setting, could be roughly expressed as $\Omega_{d,B}(n^{-1/(2-c/d)})$ for some constant c. A simple Taylor expansion shows $-1/(2-c/d) \leq -1/2-(c/4)/d$, hence this rate is compatible with our upper bound for $m = 2$ if $c \geq 8$.

Algorithm 16 Rejection sampling with proposal distribution P_g limited to n function evaluations.

```
function REJECTIONSAMPLING($f$, $g$, number of steps $n$)
    for $i$ from 1 to $n$ do
        Sample $x \sim P_g$ and $u \sim \mathcal{U}([0,1])$
        Return $x$ if $ue^{g(x)} \leq e^{f(x)}$
    end for
    return Sample from $P_g$
end function
```

To achieve better rates for sampling in the stochastic points setting, we combine approximation with a budget-limited version of rejection sampling defined in Algorithm 16. If g is shifted appropriately such that it upper-bounds f, we obtain the following convergence rate bound:

Lemma 7.2.7 (General rejection sampling bound)**.** *Suppose that $f, g : \mathcal{X} \to \mathbb{R}$ are bounded and measurable with $f(x) \leq g(x)$ for all $x \in \mathcal{X}$. Then, the distribution $\tilde{P}_f$ of* REJECTIONSAMPLING(f, g, n) *satisfies*

$$\begin{aligned} \tilde{P}_f &= (1-p_R)P_f + p_R P_g \qquad (7.2) \\ D_{\text{sup-log}}(P_f, \tilde{P}_f) &\leq \min\left\{D_{\text{sup-log}}(P_f, P_g), p_R(\exp(D_{\text{sup-log}}(P_f, P_g)) - 1)\right\} \\ D_{\text{TV}}(P_f, \tilde{P}_f) &= p_R D_{\text{TV}}(P_f, P_g) \\ W_1(P_f, \tilde{P}_f) &= p_R W_1(P_f, P_g) \ , \end{aligned}$$

where $p_R = (1 - Z_f/Z_g)^n \leq \exp(-nZ_f/Z_g)$ is the probability of overall rejection.

The proof can be found in Section 7.C.2. Due to the early stopping after n rejections, rejection sampling may significantly oversample regions where p_f is very small. Since $D_{\text{sup-log}}$ is very sensitive to this behavior, the corresponding bound is worse than for D_{TV} and W_1.

By using half of the n points to create an approximation g and then using a shifted version of g for rejection sampling with the other half of the n points, we obtain the following upper bound on the minimax optimal error:

Theorem 7.2.8 (Upper bound for sampling with stochastic evaluation points)**.** *There exists a constant $C_{m,d} > 0$ such that*

$$e_n^{\text{ad-stoch}}(\mathcal{F}_{d,m,B}, S_{\text{samp}}, D_{\text{sup-log}}) \leq \begin{cases} O_{m,d}(Bn^{-m/d}) & , C_{m,d}Bn^{-m/d} > 1 \\ O_{m,d}((C_{m,d}Bn^{-m/d})^{n/2+1}) & , C_{m,d}Bn^{-m/d} \leq 1 \ . \end{cases}$$

Theorem 7.2.8 is proven in Section 7.C.2. Combinations of approximation and rejection sampling have also been used, for example, by Achddou et al. (2019) and Chewi et al. (2022). For $C_{m,d}Bn^{-m/d} \leq 1$, the upper bound above decays faster than exponential in n. The bound is not tight, as the exponent $n/2 + 1$ can at least be improved close to n at the cost of increasing the constant $C_{m,d}$. However, for the optimization regime, the bound is tight:

Theorem 7.2.9 (Lower bound for sampling with stochastic evaluation points)**.** *There exists a constant $c_{m,d} > 0$ such that for $B > 0$ and $n \in \mathbb{N}$ with $Bn^{-m/d} \geq c_{m,d}(1 + \log(n))$, we have*

$$\begin{aligned} e_n^{\text{ad-stoch}}(\mathcal{F}_{d,m,B}, S_{\text{samp}}, D_{\text{sup-log}}) &\geq \Omega_{m,d}(Bn^{-m/d}) \\ e_n^{\text{ad-stoch}}(\mathcal{F}_{d,m,B}, S_{\text{samp}}, D_{\text{TV}}) &\geq \Omega_{m,d}(1) \\ e_n^{\text{ad-stoch}}(\mathcal{F}_{d,m,B}, S_{\text{samp}}, W_1) &\geq \Omega_{m,d}(1) \ . \end{aligned}$$

The proof of Theorem 7.2.9 in Section 7.C.2 uses the classical approach of hiding a bump, although explicitly exploiting the relation to optimization via Proposition 7.3.6 might also work. Proving lower bounds for sampling with stochastic points outside of the optimization regime seems difficult. Indeed, when restricting the function class a bit further, we can even achieve zero error:

Proposition 7.2.10. *Let $\mathcal{F} \coloneqq \{f \in C(\mathcal{X}) \mid \|f\|_\infty \leq \log(3/2), L_f = 0\}$. Then,*

$$e_n^{\text{ad-stoch}}(\mathcal{F}, S_{\text{samp}}, D_{\text{sup-log}}) = 0$$

for all $n \geq 1$.

The proof idea, executed in Section 7.C.2, is to use RejectionSampling$(\tilde{f}, g, 1)$, where $\tilde{f}(x) \coloneqq \log(2\exp(f(x)) - 1)$ and $g(x) = \log(2)$ are constructed such that the resulting distribution is exactly P_f. The assumption that L_f is known is necessary to exactly control the acceptance probability in the rejection sampling step.

7.3 Relations Between Different Problems

In this section, we study how different problems such as sampling, log-partition estimation, and optimization are related, especially also in terms of reductions between algorithms, their runtime complexities, and their convergence rates. Again, certain bounds can be established via the connection to function approximation. For this, we need an efficient approximation method achieving optimal convergence rates while producing a smooth approximation. This is possible using the moving least squares method (Lancaster and Salkauskas, 1981), which produces an approximant $g(x) = g_x(x)$, where g_x is a local polynomial regression function fitted using a smooth local weight function $w(x_i, x)$. The following theorem shows that the moving least squares method achieves the desired properties:

Theorem 7.3.1 (adapted from Li (2016) and Mirzaei (2015))**.** *Let $m, d \in \mathbb{N}_{\geq 1}$. Using the moving least squares method, it is possible to construct an approximation f_n of $f \in C^m(\mathcal{X})$ using n deterministic non-adaptive function evaluations such that*

(a) $\|f - f_n\|_{C^k} \leq O_{m,d}(\|f\|_{C^m} n^{-(m-k)/d})$ for $k \in \{0, 1, \ldots, m\}$,
(b) the runtime of constructing f_n is zero (construction takes place on-the-fly during evaluation), and
(c) the runtime of evaluating f_n at a point $x \in \mathcal{X}$ is $O_{m,d}(1)$.

We prove Theorem 7.3.1 in Appendix 7.D.

7.3.1 Runtime-Accuracy Trade-off

When investigating sampling and log-partition algorithms, we study their convergence rate and their runtime complexity both in terms of the number n of required function evaluations. Here, we show that these two quantities can be traded off against each other to some extent. Improving the computational complexity at the cost of worse convergence rates is easy by increasing n without using the additional function values:

Example 7.3.2 (Trading convergence rates for better runtime complexity)**.** Suppose we have an algorithm A for the sampling or log-partition problems with convergence rate $\Theta_{m,d}(\|f\|_{C^m} n^{-\alpha_{m,d}})$ and runtime $\Theta_{m,d}(n^{\beta_{m,d}})$. We can then evaluate f in n points but only use $N \leq n$ of these points for A. If $N = \Theta_{m,d}(n^\gamma)$, $\gamma \in (0, 1]$, we obtain a convergence rate of $\Theta_{m,d}(\|f\|_{C^m} N^{-\alpha_{m,d}}) = \Theta_{m,d}(\|f\|_{C^m} n^{-\gamma\alpha_{m,d}})$ and a runtime of $\Theta_{m,d}(n + N^{\beta_{m,d}}) = \Theta_{m,d}(n^{\max\{1,\gamma\beta_{m,d}\}})$. A similar construction could be used to move constants $C_{m,d} \geq 1$ or potential factors $\|f\|_{C^m}^k \geq 1$ from the runtime to the convergence rate. ◀

Of course, the construction in Example 7.3.2 does not improve the runtime needed to reach a desired error level, but it shows that some combinations of runtime complexity and convergence rates are not better than others. To trade runtime complexity for better convergence rates, an analogous construction is not possible, since it would need to use $N > n$ function evaluations, which would contradict the definition of n. However, we can instead use N evaluations of an approximant created using n function evaluations:

Example 7.3.3 (Trading runtime complexity for better convergence rates)**.** Suppose again that we have an algorithm A for the sampling or log-partition problems with convergence rate $\Theta_{m,d}(\|f\|_{C^m} n^{-\alpha_{m,d}})$ and runtime $\Theta_{m,d}(n^{\beta_{m,d}})$. We consider an algorithm resulting from the following construction:

(1) Use an approximation algorithm as in Theorem 7.3.1 to create an approximation f_n of f using n (deterministic) function evaluations.
(2) Run algorithm A on $N = \Theta_{m,d}(n^\gamma)$ function evaluations of f_n, $\gamma \in (0, \infty)$.

By Theorem 7.3.1, we have $\|f_n - f\|_\infty \leq O_{m,d}(\|f\|_{C^m} n^{-m/d})$, and by Proposition 7.2.2, this rate also applies to the considered distances of L_{f_n} to L_f or P_{f_n} to P_f. By the triangle inequality, the resulting algorithm has a convergence rate of

$$O_{m,d}(\|f\|_{C^m} n^{-m/d} + \|f_n\|_{C^m} N^{-\alpha_{m,d}}) = O_{m,d}(\|f\|_{C^m} n^{-\min\{m/d, \gamma\alpha_{m,d}\}}) ,$$

where we used $\|f_n\|_{C^m} \leq O_{m,d}(\|f\|_{C^m})$ due to Theorem 7.3.1 (a) with $k = m$, and runtime complexity

$$O_{m,d}(n^{\gamma\beta_{m,d}}) . \qquad \blacktriangleleft$$

While the construction in Example 7.3.3 also does not improve the runtime complexity needed to reach a desired error level, it can still be useful if evaluations of the approximant (or surrogate model) f_n are much cheaper than evaluations of f. This principle is used for example in computational chemistry, where expensive direct simulations f are approximated with machine-learned interatomic potentials f_n (Deringer et al., 2019).

7.3.2 Relation between stochastic and deterministic evaluation points

When the construction in Example 7.3.3 is applied to a sampling algorithm with stochastic evaluation points, it yields a sampling algorithm with deterministic evaluation points. This can be advantageous since the latter only needs n function evaluations to draw an arbitrary number of samples, while the former may require n new function evaluations for every drawn sample. On the other hand, this construction limits the convergence rate of the sampling algorithm to $\Omega_{m,d}(Bn^{-m/d})$, a rate which can be improved by sampling algorithms with stochastic evaluation points outside of the optimization regime (cf. Theorem 7.2.8).

Applying the construction in Example 7.3.3 to a log-partition algorithm with stochastic evaluation points yields a stochastic log-partition algorithm with deterministic evaluation points. We did not consider such algorithms separately in Section 7.2.2. However, such an algorithm is never better than its median or expected output, which is a deterministic log-partition method with deterministic evaluation points. Hence, it follows from Theorem 7.2.3 that the convergence rate of the construction in Example 7.3.3 is limited to $\Omega_{m,d}(Bn^{-m/d})$, and this rate can be improved by log-partition algorithms with stochastic evaluation points outside of the optimization regime (cf. Theorem 7.2.5).

7.3.3 Relation Between Sampling and Log-partition Estimation

A natural question is whether efficient sampling algorithms can be used to obtain efficient log-partition estimators and vice versa. We study both of these directions in the following. In fact, sampling algorithms are frequently employed for log-partition estimation in computational statistical physics and other fields (Frenkel and Smit, 2001; Friel and Wyse, 2012). One method to achieve this is thermodynamic integration (Kirkwood, 1935), of which we present a particularly simple version here. By integrating the derivative of $L(\beta) := L_{\beta f}$, it is possible to derive the following formula (Gelman and Meng, 1998; Friel and Wyse, 2012):

$$L_f = \int_0^1 \mathbb{E}_{x\sim P_{\beta f}}[f(x)] \,\mathrm{d}\beta = \mathbb{E}_{\beta\sim\mathcal{U}([0,1])}\mathbb{E}_{x\sim P_{\beta f}} f(x) \ .$$

Thermodynamic integration can be used more generally to estimate a difference $L_f - L_g$ by integrating along a path between f and g. In practice, the inner expectation is typically evaluated by Monte Carlo methods using sampling algorithms to sample from $P_{\beta f}$, while the outer integral is typically approximated with a suitable (deterministic) quadrature rule. For convenience of analysis, we will consider the case where both expectations are approximated using Monte Carlo quadrature:

Theorem 7.3.4 (Convergence of thermodynamic integration)**.** *Given $N \in \mathbb{N}_{\geq 1}$ and a sampling algorithm producing samples from approximate distributions $\tilde{P}_{\beta f}$, consider the following algorithm:*

- *Sample $\beta_1, \ldots, \beta_N \sim \mathcal{U}([0,1])$ independently.*
- *Draw $X_i \sim \tilde{P}_{\beta_i f}$ independently.*
- *Output $\tilde{L}_f := \frac{1}{N}\sum_{i=1}^N f(X_i)$.*

Then, for $\delta > 0$, we have

$$|L_f - \tilde{L}_f| \leq |L_f - \mathbb{E}\tilde{L}_f| + 2\|f\|_\infty \sqrt{\frac{\log(2/\delta)}{2N}}$$

with probability $\geq 1 - \delta$, where

$$|L_f - \mathbb{E}\tilde{L}_f| \leq 2\|f\|_\infty \sup_{\beta\in[0,1]} D_{\mathrm{TV}}(P_{\beta f}, \tilde{P}_{\beta f}),$$
$$|L_f - \mathbb{E}\tilde{L}_f| \leq |f|_1 \sup_{\beta\in[0,1]} W_1(P_{\beta f}, \tilde{P}_{\beta f}).$$

Theorem 7.3.4 is proven in Section 7.D.1. In the upper bounds above, we obtain additional factors $\|f\|_\infty$ or $|f|_1$, which deteriorate the convergence rate. While it appears that these factors are in general necessary for the TV and 1-Wasserstein distances, we explain in Remark 7.D.2 that better bounds in terms of $D_{\text{sup-log}}$ seem plausible but appear to be more difficult to prove. When considering the runtime complexity and convergence rate of the construction in Theorem 7.3.4, it is important to set them in relation to the total number n of function evaluations used. For example, if sampling from $P_{\beta_i f}$ uses $\tilde{n}$ function evaluations, then in general $n = (\tilde{n}+1)N$. If the employed sampling algorithm is non-adaptive with deterministic evaluation points, we only need $n = \tilde{n} + N$ function evaluations. Still, due to the Monte Carlo nature of thermodynamic integration, the convergence rate is at least limited to $\Omega_{m,d,f}(n^{-1/2})$, which is not optimal as we showed in Theorem 7.2.5. Of course, thermodynamic integration can be performed on top of an approximation of f instead, similar to Example 7.3.3.

Algorithm 17 Bisection sampling algorithm using a log-partition algorithm $\tilde{L}$.

```
function BisectionSampling(Function $f : \mathcal{X} \to \mathbb{R}^d$, Log-partition algorithm $\tilde{L}$, Number $M \in \mathbb{N}_0$ of bisection steps per dimension)
    For a hyperrectangle $\mathcal{Z} = \bigtimes_{i=1}^d [z_i, z_i + h_i]$, define $f_{\mathcal{Z}} : \mathcal{X} \to \mathbb{R}$ by $f_{\mathcal{Z}}(x) := f(z_1 + h_1 x_1, \dots, z_d + h_d x_d)$
    $\mathcal{Z} \leftarrow \mathcal{X}$
    for $i$ from 1 to $M$ do
        for $j$ from 1 to $d$ do
            Split $\mathcal{Z}$ along dimension $j$ into two equal-sized hyperrectangles $\mathcal{Z}_1$ and $\mathcal{Z}_2$
            Compute $p_1 := \sigma(\tilde{L}_{f_{\mathcal{Z}_1}} - \tilde{L}_{f_{\mathcal{Z}_2}})$, where $\sigma(u) = (1 + \exp(-u))^{-1}$ is the sigmoid function
            Sample $k = 1$ with probability $p_1$ and $k = 2$ otherwise
            $\mathcal{Z} \leftarrow \mathcal{Z}_k$
        end for
    end for
    return sample from the uniform distribution $\mathcal{U}(\mathcal{Z})$
end function
```

Now, we ask the converse question: Can an efficient log-partition algorithm be used for efficient sampling? To achieve such a reduction, we note that we can apply a log-partition algorithm not only to the target function f but also for example to multiple shifted and rescaled versions of f, which amounts to computing the log-partition function on subsets of the cube $\mathcal{X}$. This is exploited in Algorithm 17, which we refer to as bisection sampling. Bisection sampling has been studied for example by Marteau-Ferey et al. (2022). We give an upper bound on its error in the sup-log distance:

Theorem 7.3.5 (Convergence of bisection sampling)**.** *Let $m \geq 1, B \geq 0$ and $M \in \mathbb{N}_0$. Let $f \in \mathcal{F}_{d,m,B}$ and let $\tilde{L}$ be a log-partition estimator with worst-case error $E \geq 0$ on $\mathcal{F}_{d,m,B}$. Let $f \in C^m(\mathcal{X})$ and let $\tilde{P}_f$ be the distribution of samples produced by* BisectionSampling$(f, \tilde{L}, M)$ *in Algorithm 17. Then,*

$$D_{\text{sup-log}}(P_f, \tilde{P}_f) \leq 2MdE + 2^{-M} d \|f\|_{C^1} \ .$$

Of course, Theorem 7.3.5, which is proven in Section 7.D.1, also implies bounds on the TV and 1-Wasserstein distances using Proposition 7.2.2. The first term in the upper bound grows with M, which stems from the possibility to make an error of order $2E$ per loop iteration. However, when the resulting error decays quickly enough in the loop, it is possible to make the first term independent of M. For example, this could arise because the log-partition algorithm achieves smaller errors for smoother functions. It is also possible if we consider the 1-Wasserstein distance, which provides better error bounds on smaller hyperrectangles.

In order to analyze the resulting convergence rates, suppose that the log-partition algorithm $\tilde{L}$ uses N evaluation points. Ignoring rounding issues, we can set $M = \log_2(N^{m/d})$ and obtain the rate

$$D_{\text{sup-log}}(P_f, \tilde{P}_f) \leq O_{m,d}(E \log(N) + \|f\|_{C^1} N^{-m/d}) \ ,$$

where BISECTIONSAMPLING uses up to $n := 2MdN = O_{m,d}(N \log(N))$ function evaluations. Hence, we typically only lose polylogarithmic terms in the convergence rate, unlike for thermodynamic integration. Even for log-partition algorithms with deterministic evaluation points, the resulting sampling algorithm uses stochastic evaluation points. Again, bisection sampling can be performed on top of an approximation of f instead, similar to Example 7.3.3.

7.3.4 Relation to Optimization

Due to the relationship between sampling and optimization, a natural question is in which sense approximate sampling algorithms can perform approximate optimization. Actually, we can consider two kinds of optimization problems, similar to Novak (1988):

(OPT) The problem of outputting $x \in \mathcal{X}$ such that $|M_f - f(x)|$ is small can be seen as the low-temperature limit of the sampling problem.
(OPT*) The problem of outputting an estimate $\tilde{M}_f$ such that $|M_f - \tilde{M}_f|$ is small can be seen as the low-temperature limit of the log-partition problem.

As special cases of the reductions between sampling and log-partition estimation in Section 7.3.3, we can obtain reductions between (OPT) and (OPT*): For (OPT*), we can simply evaluate f at the estimate x obtained from (OPT), which can be seen as a simple special case of thermodynamic integration. On the other hand, for (OPT), we can recursively use (OPT*) to see whether the optimum is contained in a subdomain of $\mathcal{X}$, which corresponds to the low-temperature limit of bisection sampling.

To obtain a bound for approximate (OPT*) via approximate log-partition estimation $\tilde{L}$, we note that Lemma 7.1.3 directly yields

$$|M_f - \varepsilon \tilde{L}_{f/\varepsilon}| \leq |M_f - \varepsilon L_{f/\varepsilon}| + \varepsilon |L_{f/\varepsilon} - \tilde{L}_{f/\varepsilon}| \leq \varepsilon d \log(1 + 3d^{-1/2}\varepsilon |f|_1) + \varepsilon |L_{f/\varepsilon} - \tilde{L}_{f/\varepsilon}|$$

for temperatures $\varepsilon > 0$.

When performing approximate (OPT) via sampling from an approximate distribution $Q = \tilde{P}_f$, the result depends on the employed distance metric. Since the result of sampling is of course stochastic, we will upper-bound probabilities of the form $Q(\{x \in \mathcal{X} \mid f(x) \leq \alpha\})$ of obtaining a function value $f(x) \leq \alpha$ when drawing x from Q.

Proposition 7.3.6 (Optimization by approximate sampling)**.** *Let Q be a probability distribution on $\mathcal{X}$. Then, for any $\delta \in (0, 1]$ and $\varepsilon > 0$,*

(a) $Q(\{x \in \mathcal{X} \mid f(x) \leq \varepsilon L_{f/\varepsilon} - \varepsilon \log(1/\delta) - \varepsilon D_{\text{sup-log}}(P_{f/\varepsilon}, Q)\}) \leq \delta$,
(b) $Q(\{x \in \mathcal{X} \mid f(x) \leq \varepsilon L_{f/\varepsilon} - \varepsilon \log(1/\delta)\}) \leq \delta + D_{\text{TV}}(P_{f/\varepsilon}, Q)$,
(c) $Q(\{x \in \mathcal{X} \mid f(x) < \varepsilon L_{f/\varepsilon} - \varepsilon \log(2/\delta) - 2\delta^{-1}|f|_1 W_1(P_{f/\varepsilon}, Q)\}) \leq \delta$.

Proposition 7.3.6 is proven in Section 7.D.2. If $Q = P_{g/\varepsilon}$ for some bounded $g : \mathcal{X} \to \mathbb{R}$, the bound in ($a$) recovers the known optimization bound $f(\operatorname{argmax} g) \geq M_f - 2\|f - g\|_\infty$ in the limit $\varepsilon \searrow 0$ using Lemma 7.1.3 and Proposition 7.2.2. We can deduce from (a) that a sampling algorithm achieving the optimal rate $O_{m,d}(\|f\|_{C^m} n^{-m/d})$ in terms of $D_{\text{sup-log}}$ can be used (with sufficiently small ε) to achieve the optimal rate for (OPT) as well. On the other hand, the bounds (b) and (c) are much weaker. For example, (a) still gives a good bound for $D_{\text{sup-log}}(P_{f/\varepsilon}, Q) = 1/2$, but (b) only gives a low-probability bound for $D_{\text{TV}}(P_{f/\varepsilon}, Q) = 1/2$ and (c) is trivial for $W_1(P_{f/\varepsilon}, Q) = 1/2$. Note that an argument analogous to (b) has been used in Corollary 1 by Ma et al. (2019) to analyze the convergence of Langevin algorithms for approximate optimization.

7.4 Algorithms

In this section, we investigate the convergence rates of different algorithmic approaches toward the sampling and log-partition problems.

7.4.1 Approximation-based Algorithms

First, we study approximation-based algorithms. In Section 7.2.1, we have seen that in principle, approximation-based methods can achieve the optimal rates for the sampling and log-partition problems with deterministic points. However, for most approximations g, it is unclear how to sample from P_g or compute L_g. In the following, we will consider a few cases where this is possible:

Piecewise Constant Approximation

A very simple approximation method is piecewise constant approximation. Here, we study the convenient setting where $n = N^d$ for some $N \in \mathbb{N}$:

- Divide $\mathcal{X}$ into N^d equally-sized cubes $\mathcal{X}_1, \ldots, \mathcal{X}_n$ by dividing $[0, 1]$ into N intervals.
- Output the function $g_{f,n}$ that is piecewise constant on each cube and interpolates f at the center $x^{(i)}$ of the cube $\mathcal{X}_i$. Boundary points can be assigned to an arbitrary adjacent cube.

Given a piecewise constant function $g_{f,n}$, we can easily compute $L_{g_{f,n}} = \log\left(\frac{1}{n}\sum_{i=1}^n e^{f(x^{(i)})}\right)$ in time $O_{m,d}(n)$. Similarly, we can sample from $g_{f,n}$ in time $O_{m,d}(n)$ by first sampling a subcube $\mathcal{X}_i$ with probability $p_i = e^{f(x^{(i)}) - L_{g_{f,n}}}$ and then drawing a uniform random sample from $\mathcal{X}_i$.[9] However, the convergence rate is bad, as we prove in Section 7.E.1:

Theorem 7.4.1 (Convergence rate of piecewise constant approximation)**.** *Let $m \geq 1$ and $n = N^d$ as above. If $g_{f,n}$ is a piecewise constant interpolant as above, we have*

$$\sup_{f \in \mathcal{F}_{d,m,B}} |L_f - L_{g_{f,n}}| = \begin{cases} \Theta_{m,d}(Bn^{-1/d}) & , \textit{ if } m = 1 \textit{ or } Bn^{-1/d} > 1 \\ \Theta_{m,d}(\max\{B, B^2\}n^{-2/d}) & , \textit{ otherwise.} \end{cases}$$
$$\sup_{f \in \mathcal{F}_{d,m,B}} D_{\text{sup-log}}(P_f, P_{g_{f,n}}) = \Theta_{m,d}(Bn^{-1/d}) \ .$$

The rates of piecewise constant approximation are thus optimal for $m = 1$, but not for $m > 1$. For $m > 1$, using a combination with higher-order function approximation as

[9] This could be improved to $O_{m,d}(\log n)$ by precomputing partial sums of the p_i once in $O(n)$. Then, for drawing a sample, we can sample $u \sim \mathcal{U}([0, 1])$ and use binary search to find the correct "bucket" k with $\sum_{i=1}^{k-1} p_i \leq u < \sum_{i=1}^k p_i$.

in Example 7.3.3, it is possible to achieve the rate $O_{m,d}(Bn^{-m/d})$ with runtime $O_{m,d}(n^m)$. The result above also shows that the piecewise constant log-partition method can achieve the faster rate $O(n^{-2/d})$ of midpoint quadrature only outside of the optimization regime. We leave it as an open problem whether such faster rates are also achieved for sampling with D_{TV} or W_1. Achddou et al. (2019) analyze a combination of piecewise constant approximation with rejection sampling, but in a setting incomparable to ours. They also note that piecewise constant approximation achieves optimal rates for Hölder classes of functions.

Beyond piecewise constant approximations, piecewise linear approximations also allow for efficient sampling and log-partition estimation, and they should allow to achieve convergence rates of $O_{m,d}(Bn^{-2/d})$. We leave a precise analysis of this approach as an open problem.

Density-based Approximation

Another option to obtain tractable sampling and log-partition algorithms is to directly approximate the unnormalized density $p(x) = e^{f(x)}$. Since probability distributions are normalized, approximating λp with λq yields the same sampling and log-partition errors as approximating p with q, but the approximation error $\|\lambda p - \lambda q\|_\infty$ depends on $\lambda > 0$. To obtain a scale-invariant bound for the sampling and log-partition errors, we need to divide the approximation bound by a normalization constant:

Proposition 7.4.2 (Density approximation bounds). *Let $p, q : \mathbb{R} \to [0, \infty)$ be bounded and measurable such that $I_p, I_q > 0$, where $I_p := \int_{\mathcal{X}} p(x)\,\mathrm{d}x$. Define probability distributions P, Q with densities p/I_p and q/I_q, respectively. Then,*

$$\begin{aligned} |\log I_p - \log I_q| &\leq \log\left(\frac{1}{1 - \|p-q\|_\infty / I_p}\right) \quad \text{if } \|p-q\|_\infty < I_p, \\ D_{\mathrm{TV}}(P, Q) &\leq \frac{\|p-q\|_\infty}{\max\{I_p, I_q\}} \leq \frac{\|p-q\|_\infty}{I_p} \;. \end{aligned}$$

We prove Proposition 7.4.2 in Section 7.E.1. While $\|p\|_\infty = \|e^f\|_\infty$ can be exponential in $\|f\|_\infty$, Proposition 7.4.2 demonstrates that we need to incorporate the normalization constant to obtain reasonable estimates for sampling and log-partition computation. After incorporating the normalization constant, we arrive at terms of the form $\|p\|/I_p = \|e^f\|/Z_f = \|e^f/Z_f\| = \|p_f\|$. Hence, the norm of the (normalized) density plays an important role for convergence rates of density-based approximation approaches. As it turns out, $\|p_f\|$ does not scale exponentially in $\|f\|_\infty$, but still badly:

Theorem 7.4.3 (Density norm). *For $m \geq 1$, we have*

$$\sup_{f \in \mathcal{F}_{d,m,B}} \|p_f\|_{C^m} = \Theta_{m,d}\left(\max\{1, B\}^{m+d}\right)$$

and this asymptotic rate is attained by $f_{d,m,B}(x) = Bd^{-1}(x_1 + \cdots + x_d)$.

Theorem 7.4.3 is proven in Section 7.E.1. Suppose that $f \in C^m(\mathcal{X})$ and we can approximate $p(x) = e^{f(x)}$ with a non-negative function q with optimal[10] rate $O_{m,d}(\|p\|_{C^m} n^{-m/d})$.

[10]This rate is worst-case optimal if we only know $\|p\|_{C^m}$. However, it might not be optimal over the smaller class of functions of the form e^f with small $\|f\|_{C^m}$.

By combining Proposition 7.4.2 and Theorem 7.4.3, the distribution Q associated with the unnormalized density q then satisfies

$$D_{\mathrm{TV}}(P_f, Q) \leq O_{m,d}(\max\{1, \|f\|_{C^m}\}^{m+d} n^{-m/d}) \ . \tag{7.3}$$

Although this rate is optimal in terms of n for deterministic evaluation points, it is bad in terms of $\|f\|_{C^m}$, cf. also Example 7.1.4.

Marteau-Ferey et al. (2022) propose a sampling algorithm based on approximating the density with a (non-negative) sum-of-squares model. Specifically, for a Gibbs distribution, they suggest approximating $\sqrt{p}$ with q and then using q^2 as an unnormalized density. They achieve a rate of $O_{m,d,f}(n^{-m/d})$ in polynomial time without explicitly stating the dependence on $\|f\|$, but we conjecture that the dependence on $\|f\|$ is similar to Eq. (7.3).

7.4.2 Simple Stochastic Algorithms

We now analyze the convergence rates for some simple stochastic algorithms.

Rejection Sampling With Uniform Proposal Distribution

A simple stochastic algorithm is rejection sampling with a uniform proposal distribution. The following proposition shows that this can achieve better rates in terms of the TV distance than the density-based approximation rates in Eq. (7.3) if the maximum M_f of f is known:

Proposition 7.4.4 (Convergence of rejection sampling)**.** *Let $m \geq 1$ and let $f \in C^1(\mathcal{X})$. Then, the distribution $\tilde{P}_f$ produced by* RejectionSampling*(f, M_f, n) (see Algorithm 16) satisfies*

$$\begin{aligned} D_{\text{sup-log}}(P_f, \tilde{P}_f) &\leq \min\left\{2\|f\|_\infty, \exp\left(2\|f\|_\infty - n/\|p_f\|_\infty\right)\right\} \\ D_{\mathrm{TV}}(P_f, \tilde{P}_f) &\leq \min\{1, 2\|f\|_\infty\} \exp(-n/\|p_f\|_\infty) \\ &\leq O_{m,d}(\min\{1, \|f\|_\infty\} \max\{1, \|f\|_{C^1}\}^m n^{-m/d}) \ . \end{aligned}$$

A proof can be found in Section 7.E.2. Lower bounds for the convergence of rejection sampling could be obtained using Lemma 7.2.7, but the resulting formula would not be easy to interpret. In any case, an argument similar to the one in Section 7.4.2 and Section 7.E.2 can be made to show that RejectionSampling(f, M_f, n) cannot achieve the rate $O_{m,d}(Bn^{-m/d})$. We leave it as an open question whether similar rates to Proposition 7.4.4 can be achieved when M_f is approximately known or when a guess for M_f is used that slowly increases with n. Note that Talwar (2019) studies a similar setting where rejection sampling is not stopped after n rejections.

Monte Carlo Log-partition

Since the log-partition problem involves an integral, it is natural to approximate the integral by Monte Carlo (MC) quadrature. The following theorem gives an upper bound on the convergence rate:

Theorem 7.4.5 (Upper bounds for MC log-partition)**.** *Let $f : \mathcal{X} \to \mathbb{R}$ be Lipschitz, let $X_1, \ldots, X_n \sim \mathcal{U}(\mathcal{X})$ be independent and let*

$$\tilde{L}_n \coloneqq \log S_n, \qquad S_n \coloneqq \frac{1}{n}\sum_{i=1}^{n} \exp(f(X_i)).$$

Then, for any $\delta \in (0, 1]$, the following convergence rates hold:

(a) ***Optimization regime:*** *If $n \leq 4\log(2/\delta)(1 + 3d^{-1/2}|f|_1)^d$, we have*

$$|\tilde{L}_n - L_f| \leq d^{1/2}(\log(1/\delta))^{1/d}|f|_1 n^{-1/d} + \log(4\log(2/\delta)) + d\log(1 + 3d^{-1/2}|f|_1)$$

with probability $\geq 1 - \delta$.

(b) ***Quadrature regime:*** *If $n \geq 4\log(2/\delta)(1 + 3d^{-1/2}|f|_1)^d$, we have*

$$|\tilde{L}_n - L_f| \leq 4\log(2/\delta)^{1/2}(1 + 3d^{-1/2}|f|_1)^{d/2} n^{-1/2}$$

with probability $\geq 1 - \delta$.

We prove Theorem 7.4.5 in Section 7.E.2. Roughly speaking, the rates in the theorem above behave like $|f|_1 n^{-1/d}$ until an error of $O(1)$ is reached, and then they change to $|f|_1^{d/2} n^{-1/2}$. Figure 1 and our experiments later in Figure 2 show that this reflects the qualitative behavior of the error on linear f in practice.

Monte Carlo Sampling

We can also consider a sampling version of the Monte Carlo log-partition method considered in Section 7.4.2. The following theorem shows that it cannot achieve good rates in the optimization regime either:

Theorem 7.4.6 (Lower bound for MC sampling)**.** *Let $f : \mathcal{X} \to \mathbb{R}$ be bounded and measurable. Let $X_1, \ldots, X_n \sim \mathcal{U}(\mathcal{X})$ and let the random index $I \in \{1, \ldots, n\}$ be distributed as*

$$P(I = i) = \frac{\exp(f(X_i))}{\sum_{j=1}^{n} \exp(f(X_j))}.$$

Consider the distribution $\tilde{P}_f$ of the random sample X_I. Then, for all $B > 0$ and $n \geq 1$ with $Bn^{-1/d} \geq 4d\log(4d)$,

$$\sup_{f \in \mathcal{F}_{d,m,B}} D_{\mathrm{TV}}(P_f, \tilde{P}_f) \geq \frac{1}{2}\ .$$

The lower bounds in Theorem 7.4.6, proven in Section 7.E.2, show that $n \geq \Omega_{m,d}(B^d)$ points are required to achieve an error below $O(1)$, which is significantly worse than the around $\Theta_{m,d}(B^{d/m})$ points required by a method with the optimal rate for deterministic evaluation points. The proof only uses that the density $\tilde{p}_f$ is upper-bounded by n, and would apply analogously (using $n + 1$ instead of n) to rejection sampling with uniform proposal distribution as considered in Section 7.4.2.

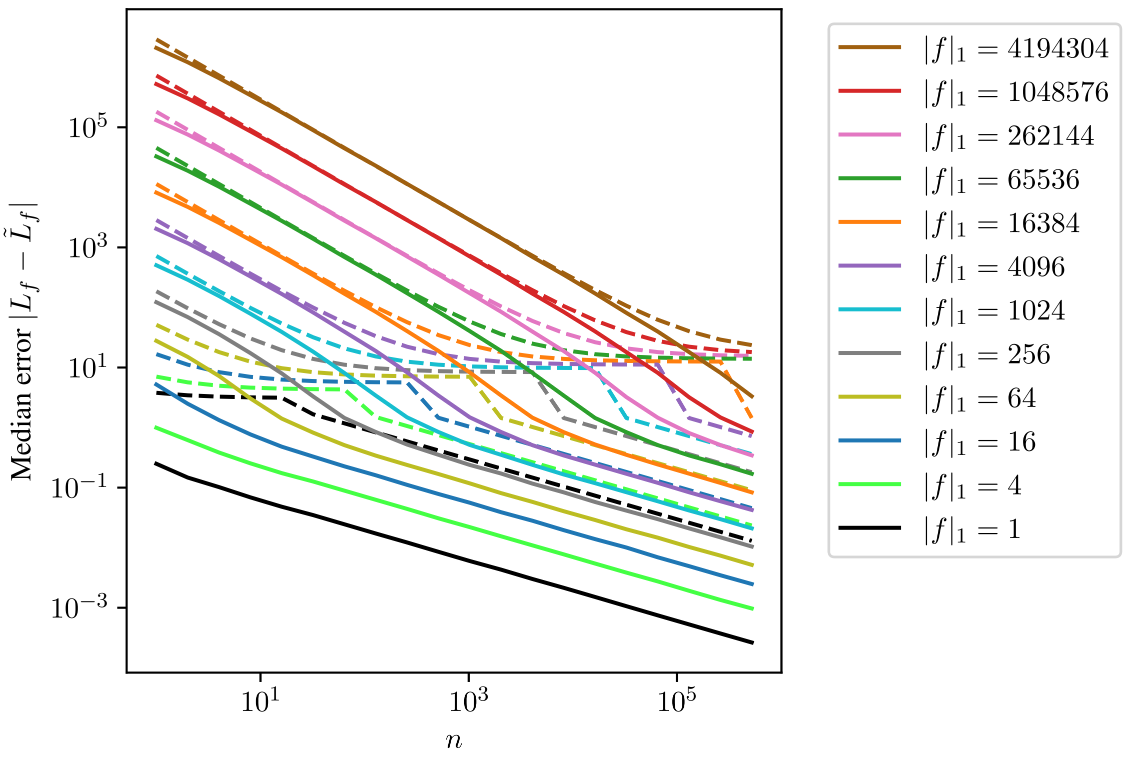


Figure 1: Median error of MC log-partition for the function $f : [0,1] \to \mathbb{R}, x \mapsto \beta x$ (with $d = 1$), for varying numbers of points n and values of $\beta = |f|_1 > 0$. Medians were computed out of 10001 repetitions. The dashed lines show the corresponding upper bounds from Theorem 7.4.5 for the median ($\delta = 1/2$).

7.4.3 Markov Chain Monte Carlo Algorithms

Markov Chain Monte Carlo (MCMC) methods are a very popular class of sampling algorithms. In particular, gradient-based MCMC algorithms such as versions of Langevin MCMC and Hamiltonian Monte Carlo (Duane et al., 1987) have been studied intensively in recent years. While most theoretical guarantees only consider the case of concave f, there have been a few extensions where f is allowed to be non-concave in a compact region of the domain. For example, Ma et al. (2019) study a certain class of functions whose gradient is L-Lipschitz and that are non-concave in a region with radius R but α-strongly convex outside of it. For the Metropolis-adjusted Langevin algorithm (MALA) to reach a TV distance error $E > 0$, they obtain the mixing time bound

$$n \leq O\left(\frac{e^{40LR^2}}{\alpha}(L/\alpha)^{3/2}d^{1/2}(d\ln(L/\alpha) + \ln(1/E))^{3/2}\right) . \tag{7.4}$$

We have used n here for the mixing time since it corresponds to the number of gradient evaluations, which are potentially more informative than the function evaluations normally allowed in our setting but can be approximated using $d + 1$ function evaluations. The dependence of the upper bound in Section 7.4.3 on L, which is related to $\|f\|_{C^2}$ in our setting, is exponential. We are not aware of a lower bound, but conjecture that a tight lower bound will also have an exponential dependence on $\|f\|$ in some fashion. This indicates that Langevin MCMC could perform worse than rejection sampling in our setting.

Beyond Langevin MCMC, there are many other popular MCMC methods, for example, variants of Hamiltonian Monte Carlo, parallel tempering (or replica exchange MCMC), and simulated tempering. Obtaining convergence rates for these methods on function classes like $\mathcal{F}_{d,m,B}$ is an interesting problem but left open in this paper. While Woodard et al. (2009) prove torpid (slow) mixing for parallel and simulated tempering in some settings, they show an exponential dependency on d for certain mixtures of Gaussians, which does not appear to imply suboptimal rates in our setting.

7.4.4 Variational Formulation for Log-Partition Estimation

In the following, we will introduce the variational approach to the log-partition problem by Bach (2022). We will first start with the simpler optimization setting. Let $\mathcal{P}(\mathcal{X})$ be the space of probability measures on $\mathcal{X}$. We start with the formulation

$$M_f = \max_{x \in \mathcal{X}} f(x) = \sup_{P \in \mathcal{P}(\mathcal{X})} \int f(x) \, \mathrm{d}P(x) \ , \tag{7.5}$$

which converts a finite-dimensional non-concave optimization problem into an infinite-dimensional convex optimization problem. To apply the approach by Bach (2022), we need to approximate the function f by a model of the form

$$g(x) = \varphi(x)^* H \varphi(x),$$

where H is a Hermitian matrix and $\varphi : \mathcal{X} \to \mathbb{C}^N$ is a suitable feature map. For example, for $d = 1$, if f is periodic and we use Fourier features $\varphi(x) = (1, e^{ix}, \ldots, e^{(N-1)ix})^\top$, then H can be determined by trigonometric interpolation, see also Woodworth et al. (2022).

For a probability distribution $P \in \mathcal{P}(\mathcal{X})$, we define the moment matrix

$$\Sigma_P \coloneqq \int_{\mathcal{X}} \varphi(x)\varphi(x)^* \, \mathrm{d}P(x) \ .$$

Because of

$$\int_{\mathcal{X}} g(x) \, \mathrm{d}P(x) = \int_{\mathcal{X}} \mathrm{tr}[\varphi(x)^* H \varphi(x)] \, \mathrm{d}P(x) = \int_{\mathcal{X}} \mathrm{tr}[H \varphi(x)\varphi(x)^*] \, \mathrm{d}P(x) = \mathrm{tr}[H\Sigma_P] \ ,$$

we then obtain

$$M_g = \sup_{P \in \mathcal{P}(\mathcal{X})} \mathrm{tr}[H\Sigma_P] = \sup_{\Sigma \in \mathcal{K}} \mathrm{tr}[H\Sigma] \ ,$$

where $\mathcal{K}$ is the (convex) set of all possible values of Σ_P. This reduces the infinite-dimensional convex optimization problem in Eq. (7.5) to a finite-dimensional convex optimization problem, and at least for certain feature maps, the set $\mathcal{K}$ has a sufficiently nice structure for optimization.

To extend this approach to the log-partition problem, Bach (2022) uses the following variational formulation by Donsker and Varadhan (1983) for general base distributions Q, where $D_{\mathrm{KL}}(P \parallel Q) = \int \log\left(\frac{\mathrm{d}P}{\mathrm{d}Q}\right) \mathrm{d}P$ is the KL divergence:

$$L_f(Q) \coloneqq \log \int_{\mathcal{X}} e^{f(x)} \, \mathrm{d}Q(x) = \sup_{P \in \mathcal{P}(\mathcal{X})} \int_{\mathcal{X}} f(x) \, \mathrm{d}P(x) - D_{\mathrm{KL}}(P \parallel Q) \ .$$

Again, after approximating f by g, we can replace the integral by $\operatorname{tr}[H\Sigma_P]$. However, to obtain a finite-dimensional optimization problem, we also need to replace the KL divergence with something that only depends on Σ_P instead of P. Since this is not possible exactly, Bach (2022) proposes multiple lower bounds, of which the tightest one (and most difficult to compute) is

$$D_{\mathrm{KL}}^{\mathrm{OPT}}(\Sigma_P \parallel \Sigma_Q) \coloneqq \inf_{\tilde{P},\tilde{Q}\in\mathcal{P}(\mathcal{X}):\Sigma_P=\Sigma_{\tilde{P}},\Sigma_Q=\Sigma_{\tilde{Q}}} D_{\mathrm{KL}}(\tilde{P} \parallel \tilde{Q}) \ .$$

This yields the following upper bound on the log-partition function:

$$\begin{aligned} L_g^{\mathrm{OPT}}(Q) &\coloneqq \sup_{P\in\mathcal{P}(\mathcal{X})} \int_{\mathcal{X}} g(x)\,\mathrm{d}P(x) - D_{\mathrm{KL}}^{\mathrm{OPT}}(\Sigma_P \parallel \Sigma_Q) \\ &= \sup_{P\in\mathcal{P}(\mathcal{X})} \operatorname{tr}[H\Sigma_P] - D_{\mathrm{KL}}^{\mathrm{OPT}}(\Sigma_P \parallel \Sigma_Q) = \sup_{\Sigma\in\mathcal{K}} \operatorname{tr}[H\Sigma] - D_{\mathrm{KL}}^{\mathrm{OPT}}(\Sigma \parallel \Sigma_Q) \\ &\geq L_g(Q) \ . \end{aligned} \tag{7.6}$$

Our investigation begins here: After inserting the definition of $D_{\mathrm{KL}}^{\mathrm{OPT}}$, a simple calculation shows that due to the minus sign, the infimum over $\tilde{P}$ merges with the supremum over P, and the infimum over $\tilde{Q}$ turns into a supremum:

Lemma 7.4.7. *For a model of the form $g(x) = \varphi(x)^*H\varphi(x)$ as above, we have*

$$L_g^{\mathrm{OPT}}(Q) = \sup_{\tilde{Q}\in\mathcal{P}(\mathcal{X}):\Sigma_{\tilde{Q}}=\Sigma_Q} L_g(\tilde{Q}) \ .$$

This formulation allows us to show a lower bound on the achievable convergence rate. The basic idea is as follows: Since Q is only known through finitely many moments Σ_Q, we can find a discrete distribution $\tilde{Q}$ with the same moments. We then choose f such that it attains its maximum at one of the discrete points, and if g approximates f sufficiently well, we show that the discrete distribution places too much weight on the maximum.

Theorem 7.4.8 (Lower bound for OPT relaxation)**.** *Let $\varphi : \mathcal{X} \to \mathbb{C}^N$ be continuous. Let*

$$n \coloneqq \dim_{\mathbb{C}} \mathcal{V}_{\mathrm{lin}}, \qquad \mathcal{V}_{\mathrm{lin}} \coloneqq \operatorname{Span}_{\mathbb{C}} \{\varphi(x)\varphi(x)^* \mid x \in \mathcal{X}\} \subseteq \mathbb{C}^{N\times N} \ .$$

In other words, n is the number of effective degrees-of-freedom of the model $g(x) = \varphi(x)^*H\varphi(x)$, and hence corresponds to the maximum number of points where such a model can interpolate arbitrary function values. Then, there exists a point $z \in \mathcal{X}$ depending only on φ, such that the periodic and analytic function

$$f : \mathcal{X} \to \mathbb{R}, x \mapsto \sum_{i=1}^{d} \cos(2\pi(x_i - z_i))$$

satisfies

$$|L_g^{\mathrm{OPT}}(\mathcal{U}([0,1])) - L_{\beta f}(\mathcal{U}([0,1]))| \geq \log\left(\frac{\beta^{d/2}}{2n+1}\right) - \|g - \beta f\|_\infty \tag{7.7}$$

for any model $g(x) = \varphi(x)^*H\varphi(x)$ and any $\beta > 0$.

What are the implications of Theorem 7.4.8, which is proven in Section 7.E.3, on convergence rates? To answer this question, we need to consider the limit $n \to \infty$, which means that N, φ, f, g in general depend on n, and we will denote them by N_n, φ_n, f_n, g_n, respectively. We also consider an inverse temperature $\beta_n := (e(2n+1))^{2/d}$. Since f_n is analytic, an approximation method with optimal rate should achieve the rate $\|g_n - \beta_n f_n\|_\infty \leq O_{m,d}(\|\beta_n f_n\|_{C^m} n^{-m/d}) = O_{m,d}(n^{-(m-2)/d})$ for every $m \in \mathbb{N}$. Suppose that this is at least achieved for $m = 3$, such that $\lim_{n\to\infty} \|g_n - \beta_n f_n\|_\infty = 0$. Then,

$$\begin{aligned} |L_{g_n}^{\mathrm{OPT}}(\mathcal{U}([0,1])) - L_{\beta_n f_n}(\mathcal{U}([0,1]))| &\geq \log\left(\frac{\beta_n^{d/2}}{2n+1}\right) - \|g_n - \beta_n f_n\|_\infty \\ &\geq 1 - O_{m,d}(n^{-(m-2)/d}) \\ &= \Omega_{m,d}(\beta_n n^{-2/d}) \text{ for sufficiently large } n. \end{aligned}$$

In other words, the approximation error and the log-partition error of the OPT relaxation in Eq. (7.6) cannot both achieve a rate strictly better than $O_{m,d}(\|f\| n^{-2/d})$ even for infinitely smooth functions, no matter which (continuous) feature map φ is chosen.

7.5 Experiments

To further investigate the convergence behavior of some simple algorithms, we study them numerically on functions of the form $f : [0,1]^3 \to \mathbb{R}, x \mapsto \beta(x_1 + x_2 + x_3)$. While these functions are simple (and concave), they pose a challenge to some general algorithms as they have a large range in relation to their Lipschitz constant. The dimension $d = 3$ has been chosen for visualization purposes, to be able to distinguish the convergence rates $n^{-1/d}$ and $n^{-2/d}$ from the typical MC convergence rate of $n^{-1/2}$. Our plots can be reproduced using the code at

`github.com/dholzmueller/sampling_experiments`

7.5.1 Log-partition Estimation

For the log-partition problem, we consider the following algorithms:

- **PC**: Compute the log-partition function of a piecewise constant approximation as in Section 7.4.1.
- **MC**: Monte carlo log-partition estimation as in Section 7.4.2.
- **PC+MC**: We use importance sampling, specifically MC quadrature on top of a piecewise constant approximation as described in Section 7.2.2: We use $n/2$ function evaluations to compute a piecewise constant approximation g of f and then use the other $n/2$ function evaluations for an MC approximation of the right-hand side in

$$L_f = L_g + \log\left(\mathbb{E}_{x\sim P_g}\left[\exp(f(x) - g(x))\right]\right) .$$

Note that PC and MC run in linear time $O_{m,d}(n)$ while PC+MC can require a runtime of $O_{m,d}(n^2)$ or $O_{m,d}(n \log n)$ depending on the implementation.

Figure 2 shows the convergence of these methods for $\beta \in \{0.1, 40, 10000\}$. For $\beta = 10000$, the methods are in an optimization regime, where PC and MC follow the

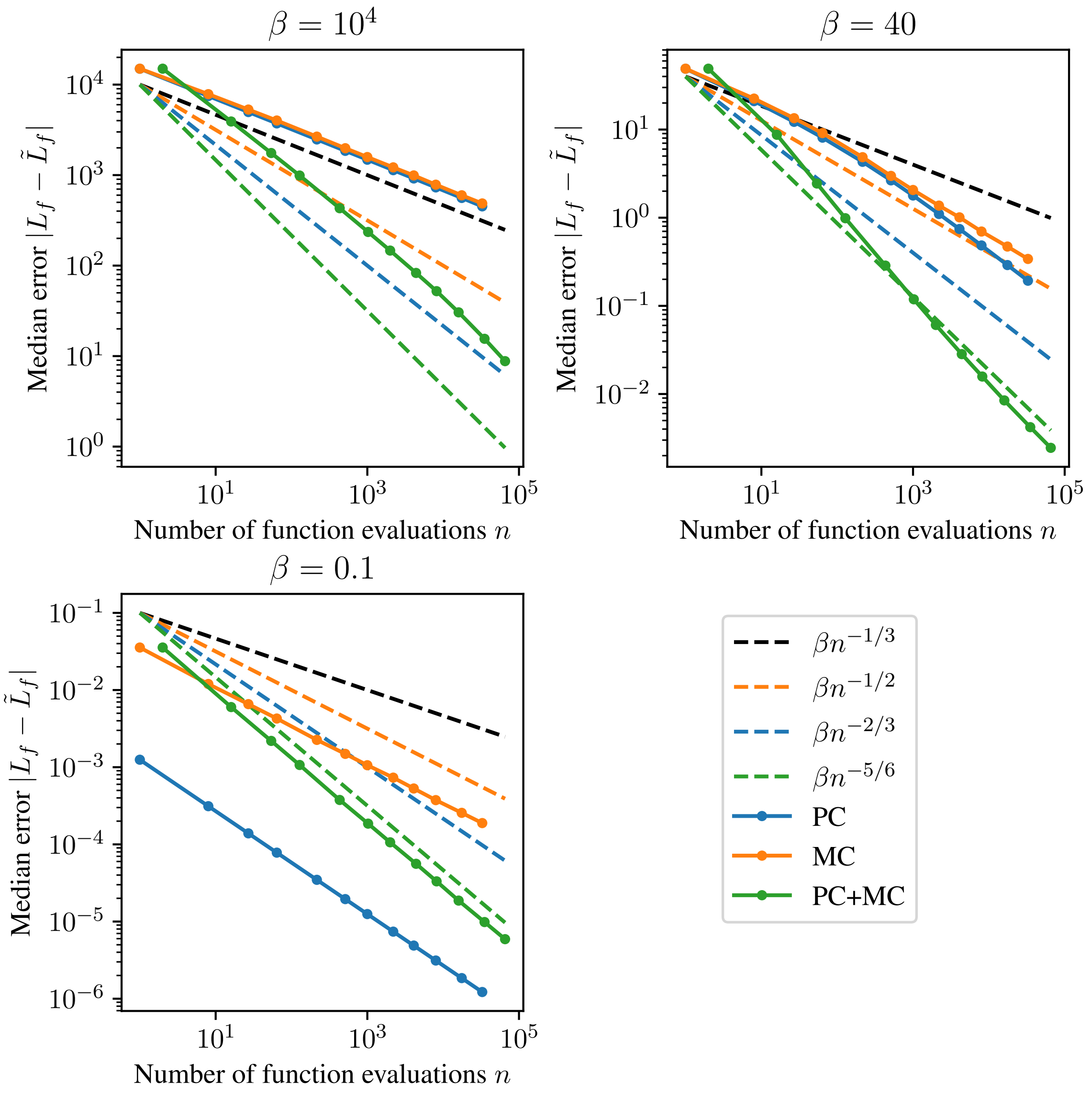


Figure 2: Convergence of the (median) error $|L_f - \tilde{L}_f|$ for different values of $\beta \in \{0.1, 40, 10000\}$. For the stochastic methods MC and PC+MC, the median is taken over 10001 independent runs.

rate $O(n^{-1/3})$ of the corresponding upper bounds in Theorem 7.4.1 and Theorem 7.4.5. Meanwhile, PC+MC follows the rate $O(n^{-2/3})$. This can be understood intuitively by noting that due to the linear nature of f, the PC proposal distribution will mostly propose points close to the optimum, such that the MC component can get much closer to the optimum than with a uniform proposal distribution.

For $\beta = 30$, we observe a transition between an optimization regime and a quadrature regime. In the quadrature regime, the convergence rate of MC is the classical MC quadrature rate $O(n^{-1/2})$, matching the upper bound in Theorem 7.4.5. Meanwhile, the convergence rate of PC transitions to $O(n^{-2/3})$, matching the worst-case bound in Theorem 7.4.1, whose proof uses a linear f for the lower bound. The combination PC+MC approaches a convergence rate around $O(n^{-5/6})$. This can be understood as the MC rate $O(n^{-1/2})$ combined with the *approximation* rate (not log-partition rate) of PC, which is $O(n^{-1/3})$. The rate $O(n^{-5/6})$ can be proven formally using arguments analogous to the proof of Theorem 7.2.5 in Section 7.C.2.

For $\beta = 0.1$, we see the same quadrature regime rates as for $\beta = 30$, except that now the constant in the rate for PC is smaller than those of MC and PC+MC. This can be explained by an observation in the proof of Theorem 7.4.1 in Section 7.E.1: Since PC performs midpoint quadrature, its error depends on the curvature of $\exp(f)$. Since f is linear, the curvature of $\exp(f)$ is significantly smaller than the worst-case curvature when $\beta \ll 1$. On the other hand, the convergence rate of PC+MC depends on the *approximation* rate of PC, which does not exhibit this phenomenon.

7.5.2 Sampling

To study convergence rates for sampling, we need a way to estimate distances between probability distributions through samples. While this can be achieved for the Wasserstein distance, and more efficiently for the related Sinkhorn distances, an even more efficient and easy-to-compute measure is the energy distance (see e.g. Székely and Rizzo, 2013) given by

$$D_{\text{energy}}(P, Q)^2 = 2\mathbb{E}_{x \sim P, x' \sim Q}\|x - x'\|_2 - \mathbb{E}_{x \sim P, x' \sim P}\|x - x'\|_2 - \mathbb{E}_{x \sim Q, x' \sim Q}\|x - x'\|_2 \ .$$

We estimate the energy distance $D_{\text{energy}}(P_f, \tilde{P}_f)$ by sampling a finite number of samples $x_1, \ldots, x_N \sim P_f$ and $\tilde{x}_1, \ldots, \tilde{x}_N \sim \tilde{P}_f)$ and then computing the energy distance $D_{\text{energy}}(Q, \tilde{Q})$ of the empirical distributions

$$Q \coloneqq \frac{1}{N}\sum_{i=1}^{N} \delta_{x_i}, \qquad \tilde{Q} \coloneqq \frac{1}{N}\sum_{i=1}^{N} \delta_{\tilde{x}_i} \ ,$$

where δ_x is the Dirac distribution at x. We compare the following sampling algorithms:

- **PC**: Sampling from a piecewise constant approximation as in Section 7.4.1.
- **MC**: Monte carlo sampling as defined in Section 7.4.2.
- **RS**: We return $\textsc{RejectionSampling}(f, M_f, n)$ as defined in Algorithm 16 and investigated in Section 7.4.2. Here, we know M_f explicitly due to the simple nature of f.

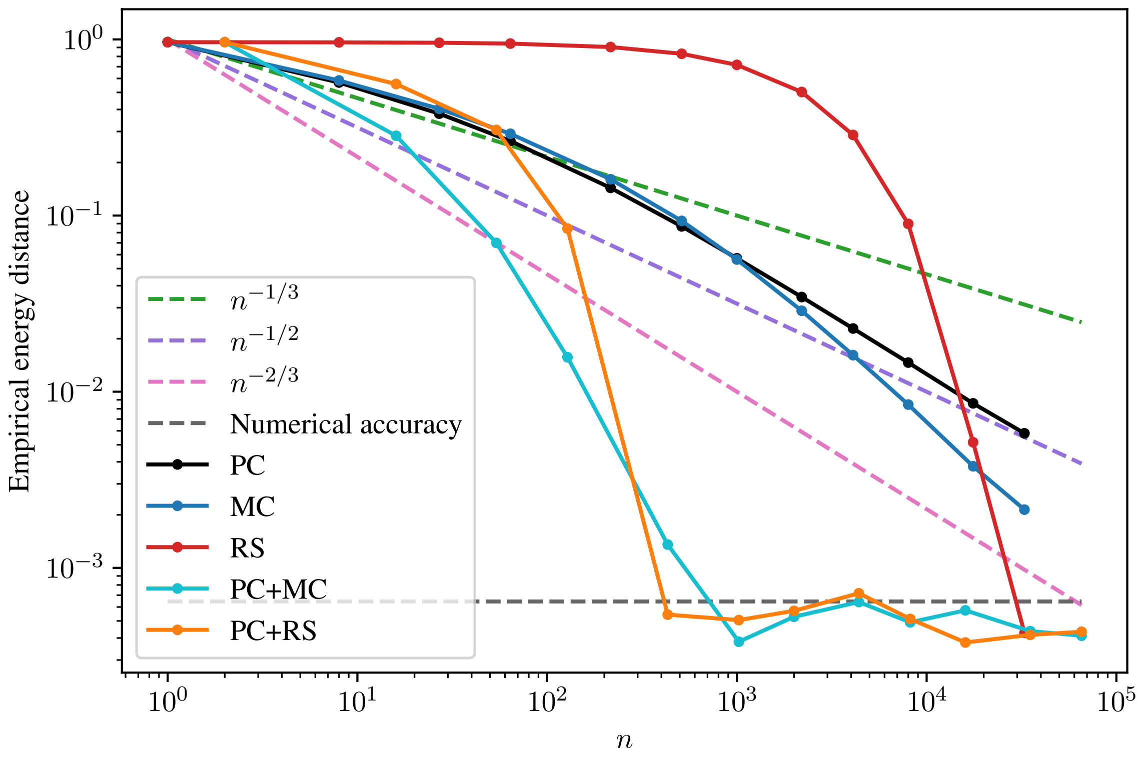


Figure 3: Convergence of different sampling methods in terms of the empirical energy distance, computed using $N = 10^6$ samples for each distribution, to the true distribution P_f for $\beta = 15$. Here, n denotes the number of function evaluations used for drawing a sample, where PC uses the same function evaluations for each sample while MC and RS need new function evaluations for every drawn sample. The gray dashed line corresponds to the maximum empirical energy distance of two sets of $N = 10^6$ samples both drawn from P_f, where the maximum is taken over three random draws.

- **PC+MC**: Performing MC sampling on top of a piecewise constant proposal distribution: We compute a piecewise constant approximant g of f with $n/2$ points, then draw samples $X_1, \dots, X_{n/2} \sim P_g$ and output X_I, where

$$P(I = i) = \frac{\exp(f(X_i) - g(X_i))}{\sum_{j=1}^{n/2} \exp(f(X_j) - g(X_j))} .$$

- **PC+RS**: We use $n/2$ points to compute a piecewise constant approximation g of f and then return $\text{R}\small{\text{EJECTION}}\text{S}\small{\text{AMPLING}}(f, g + M_{f-g}, n/2)$ as defined in Algorithm 16. Here, we know M_{f-g} explicitly due to the simple nature of f.

Note that PC, MC, and RS run in linear time $O_{m,d}(n)$ while PC+MC and PC+RS can require $O_{m,d}(n^2)$ or $O_{m,d}(n \log n)$ depending on the implementation.

For the sampling algorithms in Figure 3, the behavior in terms of convergence rates is less clear than for the log-partition algorithms. Nonetheless, it is evident that combining approximation-based and stochastic methods performs better than either of the two in isolation. Moreover, it is noticeable that RS, our budget-limited variant of rejection sampling, initially performs poorly while reaching fast convergence for larger values of n, when the probability of overall rejection becomes small.

7.6 Conclusion

In this paper, we studied the convergence rates of sampling and log-partition estimation methods on classes of m-smooth functions on the d-dimensional unit cube $\mathcal{X} = [0,1]^d$. In Section 7.2, we showed that without computational constraints, the optimal achievable convergence rates are of the form $O_{m,d}(Bn^{-m/d})$ or even better depending on the setting. We then investigated several computational reductions between problems in Section 7.3, showing that several problems are similarly hard. In Section 7.4, we studied convergence rates of specific algorithms, which are however far from being optimal unless one is willing to spend a computational effort on the order of $O(n^m)$, i.e., exponential in the smoothness m for which the optimal rate should be achieved. Our experimental study nonetheless confirms practical differences between the convergence rates of some of the investigated efficient algorithms.

Our work poses the central question of whether near-optimal convergence rates for smooth functions can be achieved with runtimes that are of fixed polynomial order $O_{m,d}(n^k)$, i.e., where k does not depend on m or d. Moreover, for many sampling algorithms, it is unclear which convergence rates they can achieve in our setting. For example, variants of parallel tempering are often employed for non-log-concave problems, and diffusion models might prove to be relevant if the score function can be approximated efficiently (Chen et al., 2022). An analysis of (mixtures of) Laplace approximations might also be interesting in this context. Beyond specific algorithms, proving lower bounds outside of the optimization regime is still an open question except for some special cases (Chewi et al., 2022), and other probability distance measures such as the KL divergence could be considered as well.

Acknowledgments We thank Alessandro Rudi, Viktor Zaverkin, Hans Kersting, Ingo Steinwart, Davoud Mirzaei, Marc Lambert, and Eric Moulines for helpful discussions. Funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2075 – 390740016. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting David Holzmüller.

Appendix

7.A Overview

The structure of the appendix matches that of the main part of this paper. We give proofs for Section 7.1 in Appendix 7.B, for Section 7.2 in Appendix 7.C, for Section 7.3 in Appendix 7.D, and for Section 7.4 in Appendix 7.E.

7.B Proofs for Introduction

Lemma 7.1.3 (Optimization limit). *Let $f : \mathcal{X} \to \mathbb{R}$ be Lipschitz-continuous with Lipschitz constant $|f|_1 < \infty$. Then, for any temperature $\varepsilon > 0$, the maximum $M_f = \max_{x \in \mathcal{X}} f(x)$ satisfies*

$$|M_f - \varepsilon L_{f/\varepsilon}| \leq \varepsilon d \log(1 + 3d^{-1/2}|f|_1/\varepsilon) \longrightarrow 0 \quad \textit{for } \varepsilon \searrow 0 \ . \tag{7.1}$$

Moreover, for any bounded and measurable $f : \mathcal{X} \to \mathbb{R}$ and any $\delta \in (0, 1]$, we have

$$P_{f/\varepsilon}(\{x \in \mathcal{X} \mid f(x) < \varepsilon L_{f/\varepsilon} - \varepsilon \log(1/\delta)\}) \leq \delta \ .$$

Proof. For the first part of the lemma, let $x^* \in \mathcal{X}$ be a maximizer of f. Without loss of generality, assume that f is shifted such that $f(x^*) = 0$.

Step 1: Upper bound. We have

$$L_f = \log \int_{\mathcal{X}} e^{f(x)} \, \mathrm{d}x \leq \log \int_{\mathcal{X}} 1 \, \mathrm{d}x = \log 1 = 0 = M_f \ .$$

Step 2: Lower bound. To show a lower bound on L_f, define the side length $R := (\max\{1, d^{-1/2}|f|_1\})^{-1}$. Since $R \leq 1$, $\mathcal{X}$ contains an axis-aligned subcube $\tilde{\mathcal{X}}$ of side length R containing x^*. Each point $x \in \tilde{\mathcal{X}}$ has distance at most $\sqrt{d}R$ from x^*, and hence by Lipschitzness, we have

$$f(x) \geq f(0) - |f|_1 \sqrt{d} R = -|f|_1 \sqrt{d} R \ .$$

We consider two cases:

(a) **Case 1: $d^{-1/2}|f|_1 \leq 1$.** In this case, we have $R = 1$ and hence

$$L_f = \log \int_{\mathcal{X}} e^{f(x)} \, \mathrm{d}x \geq \log \int_{\mathcal{X}} e^{-|f|_1 \sqrt{d}} \, \mathrm{d}x = -|f|_1 \sqrt{d} = -d(d^{-1/2}|f|_1) \ .$$

The function $h(x) := \log(1 + 3x) - x$ is concave and $h(0), h(1) \geq 0$, which shows $h(x) \geq 0$ for $x \in [0, 1]$. Hence,

$$L_f \geq -d \log(1 + 3d^{-1/2}|f|_1) \ .$$

(b) **Case 2: $d^{-1/2}|f|_1 > 1$.** In this case, we have $R = (d^{-1/2}|f|_1)^{-1}$ and hence $f(x) \geq -d$ for $x \in \tilde{\mathcal{X}}$. This yields

$$\begin{aligned} L_f &= \log \int_{\mathcal{X}} e^{f(x)} \,\mathrm{d}x \geq \log \int_{\tilde{\mathcal{X}}} e^{f(x)} \,\mathrm{d}x \\ &\geq \log \int_{\tilde{\mathcal{X}}} e^{-d} \,\mathrm{d}x = -d + d\log(R) = -d - d\log(d^{-1/2}|f|_1) \\ &= -d\log(ed^{-1/2}|f|_1) \geq -d\log(1 + 3d^{-1/2}|f|_1) \ . \end{aligned}$$

Step 3: Including the temperature. By replacing f with f/ε, we obtain

$$|M_f - \varepsilon L_{f/\varepsilon}| = |\varepsilon M_{f/\varepsilon} - \varepsilon L_{f/\varepsilon}| = \varepsilon |M_{f/\varepsilon} - L_{f/\varepsilon}| \leq \varepsilon d \log(1 + 3d^{-1/2}|f|_1/\varepsilon) \ .$$

Step 4: Probabilistic bound. We have

$$\begin{aligned} P_{f/\varepsilon}(x : f(x) \leq \varepsilon L_{f/\varepsilon} - \varepsilon\log(1/\delta)) &\leq \int_{\{x \in \mathcal{X} : f(x) \leq \varepsilon L_{f/\varepsilon} - \varepsilon \log(1/\delta)\}} \exp(f(x)/\varepsilon - L_{f/\varepsilon}) \,\mathrm{d}x \\ &\leq \int_{\mathcal{X}} \exp(-\log(1/\delta)) \,\mathrm{d}x = \delta \ . \end{aligned}$$ □

The following lemma will be useful to deal with Lipschitz constants:

Lemma 7.B.1. *Let $f \in C^m(\mathcal{X}), m \geq 1$. Then, $|f|_1 \leq d^{1/2}\|f\|_{C^m}$.*

Proof. We have

$$|f|_1 = \sup_{x \in \mathcal{X}} \|\nabla f(x)\|_2 \leq \sup_{x \in \mathcal{X}} d^{1/2}\|\nabla f(x)\|_\infty \leq d^{1/2}\|f\|_{C^1} \leq d^{1/2}\|f\|_{C^m} \ .$$ □

7.C Proofs for Information-based Complexity

Most of our lower bounds rely on the common strategy of hiding smooth functions with small support somewhere in the domain (see e.g. Novak, 1988). We will consider the following bump functions:

Definition 7.C.1 (Bump functions)**.** We define the template one-dimensional bump function

$$\tilde{b} : \mathbb{R} \to \mathbb{R}, x \mapsto \begin{cases} \exp(4 - (1-x)^{-1} - (x+1)^{-1}) & , \text{ if } x \in (-1, 1) \\ 0 & , \text{ otherwise} \end{cases}$$

and, for given dimension d, the template multi-dimensional bump function

$$b : \mathbb{R}^d \to \mathbb{R}, x \mapsto \tilde{b}(x_1) \cdots \tilde{b}(x_d) \ .$$

for $z \in \mathbb{R}^d$ and $\delta > 0$, the shifted and scaled bump functions

$$b_{z,\delta} : \mathbb{R}^d \to \mathbb{R}, x \mapsto b\left(\frac{x - z}{\delta}\right) \ .$$

Moreover, we define the open cube

$$B_\infty(x, \delta) \coloneqq \{z \in \mathbb{R}^d \mid \|z - x\|_\infty < \delta\} \ .$$ ◀

The following lemma illustrates some important properties of these bump functions:

Lemma 7.C.2 (Bump functions)**.** *The bump functions $b_{z,\delta}$ from Definition 7.C.1 satisfy*

(a) $b_{z,\delta}$ is zero outside of $B_\infty(z, \delta)$.
(b) $b_{z,\delta}$ is infinitely often continuously differentiable and all of its derivatives are bounded,
(c) there exists a constant $C_{m,d} > 0$ independent of z and δ such that for all $z \in \mathbb{R}^d$ and $\delta > 0$,

$$\|b_{z,\delta}\|_{C^m(\mathbb{R}^d)} \leq C_{m,d} \max\{1, \delta^{-m}\} \ .$$

(d) For $x \in B_\infty(z, \delta/2)$, we have $b_{z,\delta}(x) \geq 1$.

Proof.

(a) This is easy to verify from the definition.
(b) It is well-known, see e.g. Remark 3.4 (d) in Chapter V.3 of Amann and Escher (2005), that the function

$$\hat{b} : \mathbb{R} \to \mathbb{R}, x \mapsto \begin{cases} 0 & , x \leq 0 \\ \exp(-1/x) & , x > 0 \end{cases}$$

is C^∞. Since $\tilde{b}(x) = e^4 \hat{b}(1 - x) \cdot \hat{b}(x + 1)$, $\tilde{b}$ is also C^∞, and so must be b and $b_{z,\delta}$. Moreover, since $b_{z,\delta}$ has compact support, all the derivatives are bounded.
(c) Let $C_{m,d} := \|b\|_{C^m(\mathbb{R}^d)}$. We have

$$\sup_{x \in \mathbb{R}^d} \left| \frac{\partial^\alpha b_{z,\delta}(x)}{\partial x^\alpha} \right| = \delta^{-|\alpha|_1} \sup_{x \in \mathbb{R}^d} \left| \frac{\partial^\alpha b(x)}{\partial x^\alpha} \right| \ .$$

Therefore, by definition of the C^m-norm, we have

$$\|b_{z,\delta}\|_{C^m(\mathbb{R}^d)} \leq C_{m,d} \max\{1, \delta^{-m}\} \ .$$

(d) It is easy to verify that $\tilde{b}(x) \geq 1$ for $|x| \leq 1/2$. For $x \in B_\infty(z, \delta/2)$, we have

$$\|(x - z)/\delta\|_\infty \leq 1/2 \ ,$$

hence

$$b_{z,\delta}(x) = b\left(\frac{x - z}{\delta}\right) \geq 1 \cdots 1 = 1 \ . \qquad \square$$

The following lemma is useful to bound the number of bump functions that we can hide in a domain:

Lemma 7.C.3. *For $k \in \mathbb{N}_{\geq 1}$, a third-slice $\tilde{\mathcal{X}} := [0, 1/3] \times [0, 1]^{d-1}$ of the cube $\mathcal{X}$ contains at least k disjoint open cubes $B_\infty(z_1, r_k), \ldots, B_\infty(z_k, r_k)$ with radius*

$$r_k = \frac{k^{-1/d}}{12} \ .$$

Proof. Choose $N := \lceil k^{1/d} \rceil$. We can divide $\tilde{\mathcal{X}}$ into $N \cdot (3N)^{d-1} \geq N^d \geq k$ cubes of side length $(3N)^{-1}$ and radius

$$r = \frac{1}{6N} \geq \frac{1}{6(k^{1/d} + 1)} \geq \frac{1}{12k^{1/d}} = \frac{k^{-1/d}}{12} = r_k \ . \qquad \square$$

7.C.1 Deterministic Evaluation Points

We first adapt some results from Novak (1988) to our setting.

Theorem 7.2.1 (adapted from Novak (1988)). *We have*

$$\begin{aligned} e_n(\mathcal{F}_{d,m,B}, S_{\text{app}}, D_\infty) &= \Theta_{m,d}(Bn^{-m/d}), & e_n^{\text{ad}}(\mathcal{F}_{d,m,B}, S_{\text{app}}, D_\infty) &= \Theta_{m,d}(Bn^{-m/d}), \\ e_n(\mathcal{F}_{d,m,B}, S_{\text{opt}^*}, D_{\text{abs}}) &= \Theta_{m,d}(Bn^{-m/d}), & e_n^{\text{ad}}(\mathcal{F}_{d,m,B}, S_{\text{opt}^*}, D_{\text{abs}}) &= \Theta_{m,d}(Bn^{-m/d}), \\ e_n(\mathcal{F}_{d,m,B}, S_{\text{int}}, D_{\text{abs}}) &= \Theta_{m,d}(Bn^{-m/d}), & e_n^{\text{ad}}(\mathcal{F}_{d,m,B}, S_{\text{int}}, D_{\text{abs}}) &= \Theta_{m,d}(Bn^{-m/d}) . \end{aligned}$$

Proof. **Step 1: Upper bounds.** For $S \in \{S_{\text{app}}, S_{\text{opt}^*}, S_{\text{int}}\}$, Novak (1988) states upper bounds of the form $O_{m,d}(n^{-m/d})$ for bounded classes of functions in the Sobolev space $W_\infty^{m,d}$, which contain $\mathcal{F}_{d,m,B_{m,d}}$ for some $B_{m,d} > 0$ (see Section 1.3.11 and 1.3.12 in Novak (1988)). Hence, for the corresponding metric D, we have

$$e_n(\mathcal{F}_{d,m,B_{m,d}}, S, D) \leq O_{m,d}(n^{-m/d}) .$$

For another value of B, we can then take a near-optimal $\tilde{S} \in \mathcal{A}_n$ for $\mathcal{F}_{d,m,B_{m,d}}$ and define

$$\hat{\tilde{S}}(f) := \frac{B}{B_{m,d}} \tilde{S}\left(\frac{B_{m,d}}{B} f\right) ,$$

and by positive homogeneity of S and D, this then achieves the rate $O_{m,d}(Bn^{-m/d})$.

Step 2: Lower bounds. For lower bounds, it is again sufficient to consider $\mathcal{F}_{d,m,B_{m,d}}$ for a single $B_{m,d} > 0$. Novak (1988) uses bump functions created by rescaling and shifting the template bump function

$$\Phi(x) = \begin{cases} a \prod_{i=1}^d (1 - x_i^2)^m & , x \in [-1,1]^d \\ 0 & , \text{ otherwise} \end{cases}$$

for some appropriate constant $a > 0$. This function is in $W_\infty^{m,d}$ but not all of its weak m-th derivatives are continuous. Hence, the constructed counterexamples do not directly apply to $\mathcal{F}_{d,m,B_{m,d}}$. However, it is possible to replace Φ by the C^∞ bump function b from Definition 7.C.1 since the norms of the derivatives behave in the same fashion for scaled and shifted versions of b, as shown in Lemma 7.C.2. Hence, the same lower bounds still apply to $\mathcal{F}_{d,m,B_{m,d}}$. □

We can now turn to our upper bounds through approximation:

Proposition 7.2.2 (Upper bounds via approximation). *For bounded and measurable $f, g : \mathcal{X} \to \mathbb{R}$, we have*

(a) $|L_f - L_g| \leq \|f - g\|_\infty$.
(b) $d^{-1/2} W_1(P_f, P_g) \leq D_{\text{TV}}(P_f, P_g) \leq D_{\text{sup-log}}(P_f, P_g) \leq 2\|f - g\|_\infty$.

Proof. Since L_f and P_f are not influenced by changing f on null sets, we will ignore exceptional null sets in the essential supremum in the definition of $\| \cdot \|_\infty$ in the following.

(a) We have

$$\begin{aligned} L_g &= \log \int_{\mathcal{X}} e^{g(x)} \,\mathrm{d}x \\ &\leq \log \int_{\mathcal{X}} e^{f(x)+\|f-g\|_\infty} \,\mathrm{d}x = \log \left(e^{\|f-g\|_\infty} \cdot \int_{\mathcal{X}} e^{f(x)} \,\mathrm{d}x \right) \\ &= \left(\log \int_{\mathcal{X}} e^{f(x)} \,\mathrm{d}x \right) + \|f-g\|_\infty = L_f + \|f-g\|_\infty \;, \end{aligned}$$

and the other inequality follows analogously.

(b) We have $p_f(x) = \exp(f(x) - L_f)$ and $p_g(x) = \exp(g(x) - L_g)$, hence

$$\begin{aligned} D_{\text{sup-log}}(P_f, P_g) &= \left\| \log \left(\frac{p_f}{p_g} \right) \right\|_\infty = \|(f - L_f) - (g - L_g)\|_\infty \\ &\leq \|f-g\|_\infty + |L_f - L_g| \\ &\overset{\text{(a)}}{\leq} 2\|f-g\|_\infty \;. \end{aligned}$$

Let $\bar{f} := f - L_f$. By a well-known property of the TV distance (see e.g. Lemma 2.1 in Tsybakov, 2009),

$$D_{\mathrm{TV}}(P_f, P_g) = D_{\mathrm{TV}}(P_{\bar{f}}, P_{\bar{g}}) = \frac{1}{2} \int_{\mathcal{X}} |e^{\bar{f}(x)} - e^{\bar{g}(x)}| \,\mathrm{d}x \;.$$

Now, consider a fixed $x \in \mathcal{X}$. Without loss of generality, assume $\bar{f}(x) \leq \bar{g}(x)$. Then,

$$e^{\bar{g}(x) - \|\bar{f}-\bar{g}\|_\infty} \leq e^{\bar{f}(x)} \leq e^{\bar{g}(x)} \;,$$

which yields

$$\begin{aligned} |e^{\bar{f}(x)} - e^{\bar{g}(x)}| &\leq (1 - e^{-\|\bar{f}-\bar{g}\|_\infty}) e^{\bar{g}(x)} \leq (1 - e^{-\|\bar{f}-\bar{g}\|_\infty})(e^{\bar{f}(x)} + e^{\bar{g}(x)}) \\ &\leq \|\bar{f} - \bar{g}\|_\infty (e^{\bar{f}(x)} + e^{\bar{g}(x)}) \;. \end{aligned}$$

Therefore,

$$D_{\mathrm{TV}}(P_f, P_g) \leq \frac{1}{2} \int_{\mathcal{X}} \|\bar{f} - \bar{g}\|_\infty (e^{\bar{f}(x)} + e^{\bar{g}(x)}) \,\mathrm{d}\mu(x) = \|\bar{f} - \bar{g}\|_\infty = D_{\text{sup-log}}(P_f, P_g) \;.$$

The bound $W_1(P_f, P_g) \leq \operatorname{diam}(\mathcal{X}) D_{\mathrm{TV}}(P_f, P_g) = d^{1/2} D_{\mathrm{TV}}(P_f, P_g)$ for the 1-Wasserstein distance, where $\operatorname{diam}(\mathcal{X})$ is the diameter of $\mathcal{X}$, is well-known (see e.g. Gibbs and Su, 2002). □

The following technical lemmas will be used for the lower bound afterward.

Lemma 7.C.4. *Let $a, b > 0$. Then,*

$$\frac{a}{a+b} \geq \frac{1}{2} \min \left\{ 1, \frac{a}{b} \right\} \;.$$

Proof. If $a \leq b$, we have

$$\frac{a}{a+b} \geq \frac{a}{2b} \geq \frac{1}{2}\min\left\{1, \frac{a}{b}\right\} .$$

Similarly, if $a \geq b$, we have

$$\frac{a}{a+b} \geq \frac{a}{2a} = \frac{1}{2} \geq \frac{1}{2}\min\left\{1, \frac{a}{b}\right\} . \qquad \square$$

Lemma 7.C.5. *Let $c \in (0, 1]$. Then, the function*

$$h : [0, \infty) \to \mathbb{R}, x \mapsto \log(1 + c(e^x - 1))$$

satisfies $h(x) \geq cx$ for all $x \geq 0$.

Proof. For all $x \geq 0$, we have

$$h'(x) = \frac{ce^x}{1 + c(e^x - 1)} = \frac{c}{c + (1 - c)e^{-x}} \geq \frac{c}{c + (1 - c)} = c .$$

Therefore,

$$h(x) = h(0) + \int_0^x h(u)\,\mathrm{d}u \geq \int_0^x c\,\mathrm{d}u = cx . \qquad \square$$

Now, we are ready to prove the exact minimax optimal rates. The main technical difficulty is that for the lower bound in the 1-Wasserstein distance, we need to hide many bumps that are far apart, and we need to bound the resulting Wasserstein distance.

Theorem 7.2.3 (Information-based complexity of sampling and log-partition with deterministic evaluation points)**.** *We have*

$$\begin{aligned} e_n(\mathcal{F}_{d,m,B}, S_L, D_{\text{abs}}) &= \Theta_{m,d}(Bn^{-m/d}), \\ e_n(\mathcal{F}_{d,m,B}, S_{\text{samp}}, D_{\text{sup-log}}) &= \Theta_{m,d}(Bn^{-m/d}), \\ e_n(\mathcal{F}_{d,m,B}, S_{\text{samp}}, D_{\text{TV}}) &= \Theta_{m,d}(\min\{1, Bn^{-m/d}\}), \\ e_n(\mathcal{F}_{d,m,B}, S_{\text{samp}}, W_1) &= \Theta_{m,d}(\min\{1, Bn^{-m/d}\}), \end{aligned}$$

and the same rates hold for adaptive points.

Proof. **Step 0: Upper bounds.** We know from Theorem 7.2.1 that the rate $O_{m,d}(Bn^{-m/d})$ can be achieved for approximation with non-adaptive deterministic evaluation points, and we know from Proposition 7.2.2 that this rate can therefore also be achieved for the log-partition problem and the sampling problem with $D_{\text{sup-log}}$, D_{TV}, and W_1. Moreover, since $D_{\text{TV}}(P, Q) \leq 1$ for all distributions P, Q, we obtain an upper bound of $O_{m,d}(\max\{1, Bn^{-m/d}\})$ for D_{TV}. Similarly, since $\mathcal{X}$ has diameter $d^{1/2}$, W_1 is upper bounded by $d^{1/2} = O_{m,d}(1)$, and hence we also obtain an upper bound of $O_{m,d}(\max\{1, Bn^{-m/d}\})$ for W_1. The upper bounds also hold for the adaptive setting since it is more permissive.

In the following, we will derive matching asymptotic lower bounds for the adaptive setting, which then also hold for the non-adaptive setting. To this end, let $\tilde{S} \in \mathcal{A}_n^{\text{ad}}$ for the log-partition or sampling problem on the function class $\mathcal{F}_{d,m,B}$.

Step 1: Defining grids in the cube. We can cut the cube $\mathcal{X}$ along one axis into three equally shaped slices $\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2$:

$$\mathcal{C}_k \coloneqq [k/3, (k+1)/3] \times [0,1]^{d-1}, \qquad k \in \{0,1,2\} \ .$$

Then, by Lemma 7.C.3, we can find a finite set of points $\mathcal{G}_k \subseteq \mathcal{C}_k$ with $|\mathcal{G}_k| = 2n$ such that the open cubes $B_\infty(x, \delta_n)$ for $x \in \mathcal{G}_k$ and radius

$$\delta_n = \frac{(2n)^{-1/d}}{12} \geq \frac{n^{-1/d}}{24}$$

are contained in $\mathcal{C}_k$ and disjoint.

Step 2: Removing points close to queried points. Let $\mathcal{X}_n$ denote the $\leq n$ points where $\tilde{S}$ queries the zero function. For fixed $k \in \{0, 2\}$, the $2n$ cubes $(B_\infty(x, \delta_n))_{x \in \mathcal{G}_k}$ are disjoint. Hence there must be a subset $\tilde{\mathcal{G}}_k \subseteq \mathcal{G}_k$ containing n points whose corresponding cubes do not contain any point from $\mathcal{X}_n$.

Step 3: Two different functions. Now, for $k \in \{0, 2\}$ and $C_{m,d}$ as in Lemma 7.C.2, define the functions

$$f_k(x) \coloneqq BC_{m,d}^{-1} \delta_n^m \sum_{z \in \tilde{\mathcal{G}}_k} b_{z,\delta_n}(x) \ .$$

We have $\delta_n \leq 1$ and hence $\|b_{z,\delta_n}\|_{C^m} \leq C_{m,d}\delta_n^{-m}$ by Lemma 7.C.2. Because the support of the bump functions does not overlap, we have $\|f_k\|_{C^m} \leq B$ by Lemma 7.C.2 and hence $f_k \in \mathcal{F}_{d,m,B}$. By the construction of $\tilde{\mathcal{G}}_k$, f_0 and f_2 are zero on $\mathcal{X}_n$. Hence, even an adaptive $\tilde{S}$ must also query f_k at the points in $\mathcal{X}_n$, and since both are equal at those points, we must have

$$\tilde{S}(f_0) = \tilde{S}(f_2) \ .$$

Step 4: Wasserstein distance of both functions. Because f_0 and f_2 use the same number of equally wide bump functions whose support is fully contained in $\mathcal{X}$, we have

$$L_{f_0} = L_{f_2} \ . \tag{7.8}$$

To lower-bound the 1-Wasserstein distance, we use its dual formulation and choose the 1-Lipschitz function $\varphi(x) \coloneqq x_1 - 1/3$. This yields

$$\begin{aligned} W_1(P_{f_0}, P_{f_2}) \quad &\geq \quad \mathbb{E}_{x \sim P_{f_2}} \varphi(x) - \mathbb{E}_{x \sim P_{f_0}} \varphi(x) = \int_{\mathcal{X}} \varphi(x)(e^{\overline{f}_2(x)} - e^{\overline{f}_0(x)}) \,\mathrm{d}x \\ &\overset{\text{Eq. (7.8)}}{=} e^{-L_{f_0}} \int_{\mathcal{X}} \varphi(x)(e^{f_2(x)} - e^{f_0(x)}) \,\mathrm{d}x \ . \end{aligned} \tag{7.9}$$

Step 5: Lower-bounding the normalization constant. We first define the "bump integral"

$$I_n \coloneqq \int_{B_\infty(z,\delta_n)} (e^{BC_{m,d}^{-1}\delta_n^m b_{z,\delta_n}(x)} - 1) \,\mathrm{d}x \ ,$$

which is independent of z. Then, we have

$$I_n \overset{\text{Lemma 7.C.2}}{\geq} \int_{B_\infty(z,\delta_n/2)} (e^{BC_{m,d}^{-1}\delta_n^m} - 1) \,\mathrm{d}x$$

$$= \qquad \delta_n^d (e^{BC_{m,d}^{-1}\delta_n^m} - 1) \ . \tag{7.10}$$

We then obtain

$$\begin{aligned} e^{L_{f_0}} &= \int_{\mathcal{X}} e^0 \,\mathrm{d}x + \sum_{z \in \tilde{\mathcal{G}}_0} \int_{B_\infty(z,\delta_n)} (e^{BC_{m,d}^{-1}\delta_n^m b_{z,\delta_n}(x)} - e^0) \,\mathrm{d}x \\ &= 1 + nI_n \ . \end{aligned} \tag{7.11}$$

Step 6: Lower-bounding the integral. By construction of the functions φ, f_0, and f_2, we know that

$$\int_{\mathcal{X}} \varphi(x)(e^{f_2(x)} - e^{f_0(x)}) \,\mathrm{d}x \geq \int_{\mathcal{C}_2} \varphi(x)(e^{f_2(x)} - e^{f_0(x)}) \,\mathrm{d}x \ . \tag{7.12}$$

Using $\varphi(x) \geq 1/3$ and $f_2(x) \geq f_0(x)$ for $x \in \mathcal{C}_2$, we can lower-bound the latter integral as

$$\begin{aligned} \int_{\mathcal{C}_2} \varphi(x)(e^{f_2(x)} - e^{f_0(x)}) \,\mathrm{d}x &\geq \sum_{z \in \tilde{\mathcal{G}}_2} \int_{B_\infty(z,\delta_n)} \frac{1}{3}(e^{BC_{m,d}^{-1}\delta_n^m b_{z,\delta_n}(x)} - 1) \,\mathrm{d}x \\ &= \frac{1}{3} nI_n \ . \end{aligned}$$

Step 7: Wasserstein distance lower bound. By combining the previous lower bounds with Eq. (7.12), Eq. (7.11) and Eq. (7.9), we arrive at

$$W_1(P_{f_0}, P_{f_2}) \geq \frac{(1/3)nI_n}{1 + nI_n} = \frac{1}{3} \frac{I_n}{I_n + n^{-1}} \ .$$

We can then apply Lemma 7.C.4 and Eq. (7.10) to obtain, for a suitable constant $c_{m,d} > 0$,

$$\begin{aligned} W_1(P_{f_0}, P_{f_2}) &\geq \frac{1}{6} \min\left\{1, \frac{I_n}{n^{-1}}\right\} \\ &\geq \frac{1}{6} \min\left\{1, \frac{\delta_n^d (e^{BC_{m,d}^{-1}\delta_n^m} - 1)}{n^{-1}}\right\} \\ &\geq \frac{1}{6} \min\left\{1, n\delta_n^d BC_{m,d}^{-1}\delta_n^m\right\} \\ &\geq \frac{1}{6} \min\left\{1, c_{m,d} B n^{-m/d}\right\} \ . \end{aligned}$$

Step 8: Wasserstein minimax rate lower bound. Suppose that we are considering the sampling problem. As argued before, we have $\tilde{S}(f_0) = \tilde{S}(f_2)$. Hence, by an application of the triangle inequality, we must have $k \in \{0, 2\}$ such that

$$W_1(P_{f_k}, \tilde{S}(f_k)) \geq \frac{1}{12} \min\left\{1, c_{m,d} B n^{-m/d}\right\} \ .$$

The Wasserstein minimax lower bound then follows by setting $f := f_k$.

Step 9: TV distance minimax lower bound. Since

$$D_{\mathrm{TV}}(P_f, \tilde{S}(f)) \geq d^{-1/2} W_1(P_f, \tilde{S}(f))$$

(see e.g. Gibbs and Su, 2002), we obtain the same asymptotic lower bound for the TV distance.

Step 10: Sup-log minimax lower bound. We have

$$\begin{aligned} D_{\text{sup-log}}(P_{f_0}, P_{f_2}) &= \|(f_0 - L_{f_0}) - (f_2 - L_{f_2})\|_\infty = \|f_0 - f_2\|_\infty \\ &\geq \Omega_{m,d}(BC_{m,d}^{-1}\delta_n^m) = \Omega_{m,d}(Bn^{-m/d}) \ . \end{aligned}$$

Since $\tilde{S}(f_0) = \tilde{S}(f_2)$, by the triangle inequality, there must hence exist $k \in \{0, 2\}$ such that

$$D_{\text{sup-log}}(P_{f_k}, \tilde{S}(f_k)) \geq \Omega_{m,d}(Bn^{-m/d}) \ .$$

Step 11: Log-partition minimax lower bound. Suppose that we instead consider the log-partition problem. Setting $c_d = 24^{-d}$, we obtain

$$\begin{aligned} L_{f_2} &= \log(1 + nI_n) \geq \log(1 + n\delta_n^d(e^{BC_{m,d}^{-1}\delta_n^m} - 1)) \\ &\geq \log\left(1 + c_d(e^{BC_{m,d}^{-1}\delta_n^m} - 1)\right) \\ &\overset{\text{Lemma 7.C.5}}{\geq} c_d BC_{m,d}^{-1}\delta_n^m \\ &\geq \Omega_{m,d}(Bn^{-m/d}) \ . \end{aligned}$$

Since $\tilde{S}$ cannot distinguish the zero function $f \equiv 0$ and f_2, we must have

$$\max\{|L_f - \tilde{S}(f)|, |L_{f_2} - \tilde{S}(f_2)|\} \geq \Omega_{m,d}(Bn^{-m/d}) \ . \qquad \square$$

7.C.2 Stochastic Evaluation Points

Again, we first adapt some related results from Novak (1988) to our setting.

Theorem 7.2.4 (adapted from Novak (1988)). *We have*

$$\begin{aligned} {}^*\sigma_n^{\text{ad}}(\mathcal{F}_{d,m,B}, S_{\text{app}}, D_\infty) &= \Theta_{m,d}(Bn^{-m/d}), \\ {}^*\sigma_n^{\text{ad}}(\mathcal{F}_{d,m,B}, S_{\text{opt}^*}, D_{\text{abs}}) &= \Theta_{m,d}(Bn^{-m/d}), \\ {}^*\sigma_n^{\text{ad}}(\mathcal{F}_{d,m,B}, S_{\text{int}}, D_{\text{abs}}) &= \Theta_{m,d}(Bn^{-1/2-m/d}) \ . \end{aligned}$$

Proof. Analogous to the proof of Theorem 7.2.1 in Section 7.C.1, this can be shown using the positive homogeneity of $S \in \{S_{\text{app}}, S_{\text{opt}^*}, S_{\text{int}}\}$ and $D \in \{D_\infty, D_{\text{abs}}\}$, and by replacing the bump functions in the lower bound by the C^∞ bump functions from Definition 7.C.1. $\square$

We now prove our upper bound for log-partition estimation with stochastic evaluation points through approximation and importance sampling:

Theorem 7.2.5 (Upper bound for stochastic log-partition). *There exists a constant $C_{m,d} > 0$ depending only on m and d such that*

$${}^*\sigma_n^{\text{ad}}(\mathcal{F}_{d,m,B}, S_L, D_{\text{abs}}) \leq O_{m,d}\left(\min\left\{Bn^{-m/d}, \exp(C_{m,d}Bn^{-m/d})Bn^{-1/2-m/d}\right\}\right) \ .$$

Proof. The bound $O_{m,d}(Bn^{-m/d})$ can be achieved even through methods with deterministic evaluation points, as proven in Theorem 7.2.3, hence we only need to show the other bound. Since the first bound is always better for $n = 1$, we can in the following assume $n \geq 2$.

Let $\tilde{S}_n \in \mathcal{A}_n$ be a sequence of methods for which the worst-case errors

$$e_n \coloneqq \sup_{f \in \mathcal{F}_{d,m,B}} D_\infty(S_{\mathrm{app}}(f), \tilde{S}_n(f))$$

achieve the optimal rate $O_{m,d}(Bn^{-m/d})$ for the approximation problem on $\mathcal{F}_{d,m,B}$.

Set $N \coloneqq \lfloor n/2 \rfloor$, such that $N = \Omega(n)$ (since we assumed $n \geq 2$) and $2N \leq n$. Set $g \coloneqq \tilde{S}_N(f)$. For N i.i.d. random variables $X_1, \ldots, X_N \sim P_g$, set

$$\begin{aligned} \mu_N &\coloneqq \frac{1}{N} \sum_{i=1}^N \exp(f(X_i) - g(X_i)) \\ \tilde{S}_L(f) &\coloneqq L_g + \log \mu_N \ . \end{aligned}$$

Then, $\tilde{S}_L$ only uses $2N \leq n$ function evalutaions of f, hence $\tilde{S}_L \in {}^*C(\mathcal{A}_n^{\mathrm{ad}})$.

Since $\|f - g\|_\infty \leq e_N$ and since exp is $\exp(e_N)$-Lipschitz on $(-\infty, e_N)$, we have

$$1 - e_N \leq \exp(-e_N) \leq \exp(f(X_i) - g(X_i)) \leq \exp(e_N) \leq 1 + e_N \exp(e_N) \ .$$

Hence, $|\exp(f(X_i) - g(X_i)) - \mathbb{E}\mu_N| \leq (\exp(e_N) + 1)e_N$, which implies

$$\operatorname{Var} \mu_N \leq N^{-1}((\exp(e_N) + 1)e_N)^2 \ .$$

Additionally,

$$\log \mathbb{E}\mu_N = \log \int_{\mathcal{X}} e^{f(x)-g(x)} e^{g(x)-L_g} \,\mathrm{d}x = L_f - L_g \ .$$

Moreover, we have $e^{f(X_i)-g(X_i)} \in [\exp(-e_N), \exp(e_N)]$ and hence $\mu_N \in [\exp(-e_N), \exp(e_N)]$. Since log is $\exp(e_N)$-Lipschitz on $[\exp(-e_N), \exp(e_N)]$, we obtain

$$\begin{aligned} \mathbb{E}|L_f - \tilde{S}_L(f)| &= \mathbb{E}|\log(\mathbb{E}\mu_N) - \log(\mu_N)| \leq \mathbb{E} \exp(e_N)|\mu_N - \mathbb{E}\mu_N| \\ &\leq \exp(e_N)\sqrt{\mathbb{E}[(\mu_N - \mathbb{E}\mu_N)^2]} = \exp(e_N)\sqrt{\operatorname{Var} \mu_N} \\ &\leq N^{-1/2} e_N \exp(e_N)(\exp(e_N) + 1) \leq 2\exp(2e_N) N^{-1/2} e_N \\ &\leq O_{m,d}(\exp(C_{m,d} B n^{-m/d}) B n^{-1/2-m/d}) \end{aligned}$$

for a suitable constant $C_{m,d} > 0$. □

In the optimization regime, we can directly exploit the relation to optimization to get a lower bound:

Proposition 7.2.6 (Lower bound for stochastic log-partition). *For $m \geq 1$, we have*

$${}^*\sigma_n^{\mathrm{ad}}(\mathcal{F}_{d,m,B}, S_L, D_{\mathrm{abs}}) \geq \Omega_{m,d}(Bn^{-m/d}) - d \log(1 + 3B) \ .$$

Proof. Take any stochastic log-partition method $\tilde{S} \in {}^*C(\mathcal{A}_n^{\mathrm{ad}})$. We can also interpret this as a stochastic optimization method. Hence, we know from an adaptation of the corresponding lower bound by Novak (1988) that there exists a constant $c_{m,d} > 0$ and a function $f \in \mathcal{F}_{d,m,B}$ such that $\mathbb{E}D_{\mathrm{abs}}(\tilde{S}(f), S_{\mathrm{opt}^*}(f)) \geq c_{m,d}Bn^{-m/d}$. But then, using $|f|_1 \leq d^{1/2}\|f\|_{C^1} \leq d^{1/2}B$ from Lemma 7.B.1, we obtain

$$\begin{aligned} \mathbb{E}|\tilde{S}(f) - L_f| &\geq \mathbb{E}|\tilde{S}(f) - M_f| - |M_f - L_f| \\ &= \mathbb{E}D_{\mathrm{abs}}(\tilde{S}(f), S_{\mathrm{opt}^*}(f)) - |M_f - L_f| \\ &\overset{\text{Lemma 7.1.3}}{\geq} c_{m,d}Bn^{-m/d} - d\log(1+3B) \;. \qquad \square \end{aligned}$$

The following lemma will be useful to obtain a bound for rejection sampling in the sup-log distance:

Lemma 7.C.6. *Let $p \in [0,1]$ and $c \geq 0$. Then, for any $a \in [-c, c]$, we have*

$$|\log(1 + p(e^a - 1))| \leq \min\{c, p(e^c - 1)\} \;.$$

Proof. For an upper bound, we use $\log(1+x) \leq x$ to obtain

$$\begin{aligned} \log(1 + p(e^a - 1)) &\leq \log(1 + p(e^c - 1)) \leq p(e^c - 1) \;, \\ \log(1 + p(e^a - 1)) &\leq \log(1 + (e^c - 1)) = c \;. \end{aligned}$$

For lower bounds, we note that

$$1 + p(e^a - 1) \geq 1 + p(e^{-c} - 1) \geq 1 + (e^{-c} - 1) = e^{-c} \;.$$

This immediately yields $\log(1 + p(e^a - 1)) \geq -c$. Moreover, because log is e^c-Lipschitz on $[e^{-c}, \infty)$, we have

$$\begin{aligned} \log(1 + p(e^a - 1)) &\geq \log(1 + p(e^{-c} - 1)) = \log(1 + p(e^{-c} - 1)) - \log(1) \\ &\geq -e^c|p(e^{-c} - 1)| = -p(e^c - 1) \;. \qquad \square \end{aligned}$$

Now, we can prove upper bounds for rejection sampling:

Lemma 7.2.7 (General rejection sampling bound)**.** *Suppose that $f, g : \mathcal{X} \to \mathbb{R}$ are bounded and measurable with $f(x) \leq g(x)$ for all $x \in \mathcal{X}$. Then, the distribution $\tilde{P}_f$ of* RejectionSampling(f, g, n) *satisfies*

$$\begin{aligned} \tilde{P}_f &= (1 - p_R)P_f + p_R P_g \qquad (7.2) \\ D_{\text{sup-log}}(P_f, \tilde{P}_f) &\leq \min\{D_{\text{sup-log}}(P_f, P_g), p_R(\exp(D_{\text{sup-log}}(P_f, P_g)) - 1)\} \\ D_{\mathrm{TV}}(P_f, \tilde{P}_f) &= p_R D_{\mathrm{TV}}(P_f, P_g) \\ W_1(P_f, \tilde{P}_f) &= p_R W_1(P_f, P_g) \;, \end{aligned}$$

where $p_R = (1 - Z_f/Z_g)^n \leq \exp(-nZ_f/Z_g)$ is the probability of overall rejection.

Proof. **Step 1: Exact distribution.** We prove Eq. (7.2) via induction on n. For $n = 0$, this is clear. Now, suppose the statement is true for $n \in \mathbb{N}_0$. Denote by A the event that RejectionSampling$(f, g, n+1)$ accepts in the first iteration. Then, we have

$$P(A) = \mathbb{E}_{x \sim P_g}\mathbb{E}_{u \sim \mathcal{U}([0,1])}\mathbb{1}[ue^{g(x)} \leq e^{f(x)}] \overset{f \leq g}{=} \mathbb{E}_{x \sim P_g}\frac{e^{f(x)}}{e^{g(x)}} = \int_{\mathcal{X}} e^{f(x) - g(x)}\frac{e^{g(x)}}{Z_g}\,\mathrm{d}x = \frac{Z_f}{Z_g} \;.$$

The density of $x \in \mathcal{X}$ conditional on acceptance is

$$p(x|A) \propto p(A|x)p(x) = e^{f(x)-g(x)} \frac{e^{g(x)}}{Z_g} \propto p_f(x) \ ,$$

hence $P(x|A) = P_f$. On the other hand, the distribution $P(x|A^c)$, i.e. the distribution of x conditioned on not-acceptance is the distribution for RejectionSampling(f, g, n), which we know from Eq. (7.2) by the induction hypothesis. Hence, the distribution $\tilde{P}_f$ for RejectionSampling$(f, g, n+1)$ is

$$\begin{aligned}\tilde{P}_f &= P(A)P(\cdot|A) + P(A^c)P(\cdot|A^c) = \frac{Z_f}{Z_g}P_f + \left(1 - \frac{Z_f}{Z_g}\right)\left(P_f + \left(1 - \frac{Z_f}{Z_g}\right)^n (P_g - P_f)\right) \\ &= P_f + \left(1 - \frac{Z_f}{Z_g}\right)^{n+1} (P_g - P_f) = (1 - p_R)P_f + p_R P_g \ .\end{aligned}$$

The argument above also shows that the overall rejection probability is $(1 - P(A))(1 - Z_f/Z_g)^n = (1 - Z_f/Z_g)^{n+1}$. Moreover, the bound $(1 - Z_f/Z_g)^n \leq \exp(-nZ_f/Z_g)$ follows from $1 - x \leq \exp(-x)$ for $x \geq 0$.

Step 2: Sup-log distance. From step 1, we see that

$$\begin{aligned} D_{\text{sup-log}}(P_f, \tilde{P}_f) &= \left\|\log\left((1 - p_R)e^{\bar{f}} + p_R e^{\bar{g}}\right) - \bar{f}\right\|_\infty \\ &= \left\|\log\left((1 - p_R) + p_R e^{\bar{g}-\bar{f}}\right)\right\|_\infty \\ &= \left\|\log\left(1 + p_R(e^{\bar{g}-\bar{f}} - 1)\right)\right\|_\infty \\ &\overset{\text{Lemma 7.C.6}}{\leq} \min\left\{\|\bar{g} - \bar{f}\|_\infty, p_R\left(e^{\|\bar{g}-\bar{f}\|_\infty} - 1\right)\right\} \\ &= \min\left\{D_{\text{sup-log}}(P_g, P_f), p_R(\exp(D_{\text{sup-log}}(P_g, P_f)) - 1)\right\} \ . \end{aligned}$$

Step 3: TV distance. Using Eq. (7.2), we obtain for the TV distance:

$$\begin{aligned} D_{\text{TV}}(P_f, \tilde{P}_f) &= \sup_{A \subseteq \mathcal{X} \text{ measurable}} |P_f(A) - \tilde{P}_f(A)| = \sup_{A \subseteq \mathcal{X} \text{ measurable}} |p_R P_f(A) - p_R P_g(A)| \\ &= p_R D_{\text{TV}}(P_f, P_g) \ . \end{aligned}$$

Step 4: 1-Wasserstein distance. Using Eq. (7.2), we obtain for the 1-Wasserstein distance:

$$\begin{aligned} W_1(P_f, \tilde{P}_f) &= \sup_{\varphi \text{ 1-Lipschitz}} \left(\int \varphi(x)\,\mathrm{d}P_f(x) - \int \varphi(x)\,\mathrm{d}\tilde{P}_f(x)\right) \\ &= p_R \sup_{\varphi \text{ 1-Lipschitz}} \left(\int \varphi(x)\,\mathrm{d}P_f(x) - \int \varphi(x)\,\mathrm{d}P_g(x)\right) \\ &= p_R W_1(P_f, P_g) \ . \end{aligned}$$

□

With the upper bounds for rejection sampling proven above, we can analyze a combination of approximation and rejection sampling to prove the following upper bound:

Theorem 7.2.8 (Upper bound for sampling with stochastic evaluation points)**.** *There exists a constant $C_{m,d} > 0$ such that*

$$e_n^{\text{ad-stoch}}(\mathcal{F}_{d,m,B}, S_{\text{samp}}, D_{\text{sup-log}}) \leq \begin{cases} O_{m,d}(Bn^{-m/d}) & , C_{m,d}Bn^{-m/d} > 1 \\ O_{m,d}((C_{m,d}Bn^{-m/d})^{n/2+1}) & , C_{m,d}Bn^{-m/d} \leq 1 \ . \end{cases}$$

Proof. **Step 1: Sampling method definition.** We consider the following sampling method:

(1) Use $\lfloor n/2 \rfloor$ function evaluations to create an approximation g of f, using a near-optimal approximation method such that the worst-case sup-log error is $E_n \leq O_{m,d}(Bn^{-m/d})$.
(2) Return a sample using RejectionSampling$(f, g + e_n, \lceil n/2 \rceil)$.

For step (1) we note that we have $\lfloor n/2 \rfloor \geq \Omega(n)$ except if $n = 1$. However, in the case $n = 1$, we can use the approximation $g = 0$ with worst-case error $E_n = B \leq O_{m,d}(Bn^{-m/d})$. Thus, it is indeed possible to achieve the bound in step (1).

Step 2: Upper bound. Denote by $C_{m,d} > 0$ a constant such that $E_n \leq C_{m,d}Bn^{-m/d}/2$. Moreover, denote by $\tilde{P}_f$ the distribution produced by the sampling method defined in step 1. By Lemma 7.2.7, we have for $\tilde{g} \coloneqq g + E_n$:

$$\begin{aligned} &D_{\text{sup-log}}(P_f, \tilde{P}_f) \\ &\leq \min\left\{D_{\text{sup-log}}(P_f, P_g), \left(1 - \frac{Z_f}{Z_{\tilde{g}}}\right)^{\lceil n/2 \rceil} (\exp(D_{\text{sup-log}}(P_f, P_g)) - 1)\right\} . \end{aligned} \tag{7.13}$$

The first bound $D_{\text{sup-log}}(P_f, P_g)$ already yields the desired bound for $C_{m,d}Bn^{-m/d} > 1$. Now, consider the case $C_{m,d}Bn^{-m/d} \leq 1$. We have

$$Z_f = \int_{\mathcal{X}} e^{f(x)} \,\mathrm{d}x \geq \int_{\mathcal{X}} e^{\tilde{g}(x) - 2E_n} \,\mathrm{d}x \geq \exp(-C_{m,d}Bn^{-m/d}) Z_{\tilde{g}} .$$

Now, the second bound in Eq. (7.13) yields

$$\begin{aligned} D_{\text{sup-log}}(P_f, \tilde{P}_f) &\leq \left(1 - \exp(-C_{m,d}Bn^{-m/d})\right)^{n/2} (\exp(C_{m,d}Bn^{-m/d}) - 1) \\ &\leq \exp(C_{m,d}Bn^{-m/d}) \left(1 - \exp(-C_{m,d}Bn^{-m/d})\right)^{n/2+1} \\ &\leq e \cdot (C_{m,d}Bn^{-m/d})^{n/2+1} \leq \Omega_{m,d}((C_{m,d}Bn^{-m/d})^{n/2+1}) . \end{aligned}$$ □

Next, we prove corresponding lower bounds in the optimization regime, again using bump functions:

Theorem 7.2.9 (Lower bound for sampling with stochastic evaluation points)**.** *There exists a constant $c_{m,d} > 0$ such that for $B > 0$ and $n \in \mathbb{N}$ with $Bn^{-m/d} \geq c_{m,d}(1 + \log(n))$, we have*

$$\begin{aligned} e_n^{\text{ad-stoch}}(\mathcal{F}_{d,m,B}, S_{\text{samp}}, D_{\text{sup-log}}) &\geq \Omega_{m,d}(Bn^{-m/d}) \\ e_n^{\text{ad-stoch}}(\mathcal{F}_{d,m,B}, S_{\text{samp}}, D_{\text{TV}}) &\geq \Omega_{m,d}(1) \\ e_n^{\text{ad-stoch}}(\mathcal{F}_{d,m,B}, S_{\text{samp}}, W_1) &\geq \Omega_{m,d}(1) . \end{aligned}$$

Proof. We re-use some results from the proof of Theorem 7.2.3 in Section 7.C.1. We consider again the decomposition of the cube $\mathcal{X}$ into three slices

$$\mathcal{C}_k \coloneqq [k/3, (k+1)/3] \times [0,1]^{d-1}, \qquad k \in \{0, 1, 2\} .$$

Consider a sampling algorithm $\tilde{S} \in \mathcal{A}_n^{\text{ad-stoch}}$ with stochastic evaluation points and consider a corresponding random sample $X_f = \phi(N(f, \omega), \omega)$ as defined in Section 7.2.2.

Step 1.1: Candidate functions for the sup-log distance. By Lemma 7.C.3, $\mathcal{C}_0$ contains $4(n+1)$ disjoint open balls $B_\infty(z_i, \delta_n)$ with radius

$$\delta_n := r_{4(n+1)} = \frac{(4(n+1))^{-1/d}}{12} \geq \frac{n^{-1/d}}{96} .$$

Let $f_0 \equiv 0$ be the zero function. Consider the set $\mathcal{Q}(\omega)$ containing the n random points where $N(f_0, \omega)$ queries f_0 and the one random point $\phi(N(f_0, \omega), \omega)$ that the sampling method outputs. We can pick an $i \in \{1, \ldots, 4(n+1)\}$ such that the cube $B_\infty(z_i, \delta_n)$ contains a point from $\mathcal{Q}(\omega)$ only with probability $\leq 1/4$. With $C_{m,d}$ as in Lemma 7.C.2, we define

$$f_1(x) := C_{m,d}^{-1} B \delta_n^{-1} b_{z_i,\delta_n}(x) ,$$

which satisfies $f_1 \in \mathcal{F}_{d,m,B}$. Using analogous arguments to the proof of Theorem 7.2.3 in Section 7.C.1, we obtain

$$\begin{aligned} L_{f_1} &= \log(1 + I_n) \geq \log(1 + \delta_n^d(e^{C_{m,d}^{-1} B \delta_n^m} - 1)) \geq C_{m,d}^{-1} B \delta_n^m + \log(\delta_n^d) \\ &\geq \tilde{c}_{m,d} B n^{-m/d} - \log(n) - d \log(96) \end{aligned}$$

for a suitable constant $\tilde{c}_{m,d} > 0$.

Step 1.2: Bounding the distribution on f_1. Now, the probability of the event $\phi(N(f_0, \omega), \omega) \in B_\infty(z_i, \delta_n)$ is at most $1/4$ by construction. Moreover, the probability of $N(f_0, \omega)$ querying $B_\infty(z_i, \delta_n)$ is also at most $1/4$, hence the probability of $N(f_1, \omega)$ querying $B_\infty(z_i, \delta_n)$ is also at most $1/4$. By the union bound, the probability that $\phi(N(f_1, \omega), \omega) \in B_\infty(z_i, \delta_n)$ is at most $1/2$. Now, to have $D_{\text{sup-log}}(\tilde{S}(f_1), P_{f_1}) < \infty$, $\tilde{S}(f_1)$ must be of the form P_g for some function $g : \mathcal{X} \to \mathbb{R}$. Without loss of generality, we can assume $L_g = 0$. Then, since the set $\hat{\mathcal{X}} := \mathcal{X} \setminus B_\infty(z_i, \delta_n)$ satisfies $P_g(\hat{\mathcal{X}}) \geq 1/2$, there exists $x \in \hat{\mathcal{X}}$ with $p_g(x) \geq 1/2$, implying $g(x) \geq \log(1/2)$. But then,

$$\begin{aligned} D_{\text{sup-log}}(\tilde{S}(f_1), P_{f_1}) &\geq |(g(x) - L_g) - (f_1(x) - L_{f_1})| = |g(x) + L_{f_1}| \\ &\geq \tilde{c}_{m,d} B n^{-m/d} - \log(n) - d \log(96) - \log(2) \\ &\geq \tilde{c}_{m,d} B n^{-m/d} - \log(n) - 6d . \end{aligned}$$

Especially, for $B n^{-m/d} \geq 12 d \tilde{c}_{m,d}^{-1}(1 + \log(n))$, we have

$$D_{\text{sup-log}}(\tilde{S}(f_1), P_{f_1}) \geq \tilde{c}_{m,d} B n^{-m/d} - \frac{1}{2} \tilde{c}_{m,d} B n^{-m/d} = \Omega_{m,d}(B n^{-m/d}) .$$

Step 2.1: Candidate functions for the Wasserstein distance. By Lemma 7.C.3, for $M \in \mathbb{N}$ to be determined later, we can place Mn subcubes each in $\mathcal{C}_0$ and $\mathcal{C}_2$ with radius

$$\delta_n := r_{Mn} := \frac{(Mn)^{-1/d}}{12} .$$

By an analogous argument to Step 1.1, we can find subcubes $B_\infty(z_0, \delta_n)$ and $B_\infty(z_2, \delta_n)$ of $\mathcal{C}_0$ and $\mathcal{C}_2$ such that the probability of one of them being queried for f_0 is at most $2/M$. Following Lemma 7.C.2, we construct the functions

$$f_k(x) := C_{m,d}^{-1} B \delta_n^m b_{z_k,\delta_n}(x), \qquad k \in \{0, 2\} ,$$

which are contained in $\mathcal{F}_{d,m,B}$.

Step 2.2: Bounding the Wasserstein distance. We set $M \coloneqq 20d$. Since the two subcubes are only queried with probability at most $2/M$, we know that

$$W_1(\tilde{S}(f_0), \tilde{S}(f_2)) \leq d^{1/2} D_{\mathrm{TV}}(\tilde{S}(f_0), \tilde{S}(f_2)) \leq d^{1/2} \frac{2}{M} \leq \frac{1}{10} .$$

With an argument analogous to the proof of Theorem 7.2.3 in Section 7.C.1, we obtain

$$W_1(P_{f_0}, P_{f_2}) \geq \frac{1}{3} \frac{I_n}{1 + I_n} \overset{\text{Lemma 7.C.4}}{\geq} \frac{1}{6} \min\{1, I_n\} ,$$

and

$$I_n \geq \delta_n^d (e^{BC_{m,d}^{-1} \delta_n^m} - 1) \geq \tilde{c}_{m,d} n^{-1} (e^{\tilde{c}_{m,d} B n^{-m/d}} - 1)$$

for a suitable constant $\tilde{c}_{m,d} \in (0, 1)$. Now, suppose that

$$B n^{-m/d} \geq \tilde{c}_{m,d}^{-1} (1 + \log(\tilde{c}_{m,d}^{-1}))(1 + \log(n)) .$$

We obtain

$$\tilde{c}_{m,d} B n^{-m/d} \geq 1 + \log(\tilde{c}_{m,d}^{-1}) + \log(n) \geq \log(\tilde{c}_{m,d}^{-1} n + 1)$$

and therefore

$$W_1(P_{f_0}, P_{f_2}) \geq \frac{1}{6} \min\{1, I_n\} \geq \frac{1}{6} \min\{1, 1\} = \frac{1}{6} .$$

Since W_1 satisfies the triangle inequality, there must exist $k \in \{0, 2\}$ with

$$W_1(\tilde{S}(f_k), P_{f_k}) \geq \frac{1}{2} \left(\frac{1}{6} - \frac{1}{10} \right) = \frac{1}{30} = \Omega_{m,d}(1) .$$

Step 3: TV lower bound. The corresponding lower bound for the TV distance follows from the inequality $W_1(P, Q) \leq d^{1/2} D_{\mathrm{TV}}(P, Q)$. ☐

Finally, we prove our auxiliary result on the complexity of sampling when the log-partition function is known:

Proposition 7.2.10. *Let $\mathcal{F} \coloneqq \{f \in C(\mathcal{X}) \mid \|f\|_\infty \leq \log(3/2), L_f = 0\}$. Then,*

$$e_n^{\text{ad-stoch}}(\mathcal{F}, S_{\text{samp}}, D_{\text{sup-log}}) = 0$$

for all $n \geq 1$.

Proof. For $f \in \mathcal{F}$, define $\tilde{f}(x) \coloneqq \log(2\exp(f(x)) - 1)$ and $g(x) \coloneqq \log(2)$. We have

$$\begin{aligned} Z_{\tilde{f}} &= \int_{\mathcal{X}} (2\exp(f(x)) - 1) \,\mathrm{d}x = 2Z_f - 1 = 1 , \\ Z_g &= \int_{\mathcal{X}} e^{\log(2)} \,\mathrm{d}x = 2 . \end{aligned}$$

Let $\tilde{P}_f$ be the distribution of RejectionSampling$(\tilde{f}, g, 1)$, which only uses one evaluation of $\tilde{f}$ and therefore only one evaluation of f. By Lemma 7.2.7, we have

$$\tilde{P}_f = \frac{Z_{\tilde{f}}}{Z_g} P_{\tilde{f}} + \left(1 - \frac{Z_{\tilde{f}}}{Z_g}\right) P_g = \frac{1}{2} P_{\tilde{f}} + \frac{1}{2} P_g = P_f . \qquad ☐$$

7.D Proofs for Relations Between Different Problems

The proof of the following theorem adapts results from the literature, showing that they apply to our setting:

Theorem 7.3.1 (adapted from Li (2016) and Mirzaei (2015)). *Let $m, d \in \mathbb{N}_{\geq 1}$. Using the moving least squares method, it is possible to construct an approximation f_n of $f \in C^m(\mathcal{X})$ using n deterministic non-adaptive function evaluations such that*

(a) $\|f - f_n\|_{C^k} \leq O_{m,d}(\|f\|_{C^m} n^{-(m-k)/d})$ for $k \in \{0, 1, \dots, m\}$,
(b) the runtime of constructing f_n is zero (construction takes place on-the-fly during evaluation), and
(c) the runtime of evaluating f_n at a point $x \in \mathcal{X}$ is $O_{m,d}(1)$.

Proof. **Step 1: The method.** The idea of the moving least squares method (Lancaster and Salkauskas, 1981) is to obtain an approximation $g(x) = g_x(x)$ of $f(x)$ at an evaluation point x by determining g_x as the solution to a polynomial least-squares regression problem with data $(x_i, f(x_i))$, weighted with weights $\Phi(x, x_i)$ that (smoothly) vanish for large $\|x - x_i\|$. We will not state the exact method here but refer to the publications by Li (2016) and Mirzaei (2015), whose analysis we are using here. While Theorem 4.1 of Li (2016) essentially directly provides the result (a), it is unclear to us if the corresponding constants are independent of the evaluation points in the way that we need. Thus, in the following, we will try to verify the slightly stronger conditions of Theorem 3.12 of Mirzaei (2015) and explain how it can be adapted to our setting with minor modifications.

Step 2: Verifying the assumptions. Now, we list the major assumptions of Theorem 3.12 of Mirzaei (2015) and show that they are satisfied for a suitable choice of evaluation points and weighting function. The assumptions on smoothness are deferred until Step 3, where we will show how to adapt them to our setting. We define the number $N := \lfloor n^{1/d} \rfloor$ of grid points along each axis. By dividing each axis into N equal intervals, we obtain a partition of $\mathcal{X}$ into N^d cubes. Let X be the set of midpoints of these cubes. Hence, $|X| = N^d = \lfloor n^{1/d} \rfloor^d \geq (n^{1/d}/2)^d = \Omega_{m,d}(n)$.

- The considered domain Ω is a bounded set with Lipschitz boundary. We want to consider $\Omega := \mathcal{X}$, which is bounded and has a Lipschitz boundary.
- The maximum degree m of the polynomial basis satisfies $m \geq 1$. While m denotes the (known) smoothness of the target function f in our context, we will assume that the maximum degree of the polynomial basis is also m. While a maximum degree of $m - 1$ should be sufficient for our purposes (as it is in Li (2016)), using a maximum degree of m avoids notational confusion and simplifies the adaptation of the arguments of Mirzaei (2015).
- The fill distance $h_{X,\Omega} = \sup_{x \in \Omega} \min_{x' \in X} \|x - x'\|_2$ satisfies $h_{X,\Omega} \leq \min\{h_0, 1\}$ for some given constant $h_0 > 0$. In our case, the fill distance is $h_{X,\Omega} = \sqrt{d}/(2N) = \Theta_{m,d}(n^{-1/d})$, which satisfies the assumption for large enough values of n. The errors for smaller n do not affect the asymptotic rate.
- The weight function is defined through a radial function $\phi : [0, \infty) \to \mathbb{R}$, which is supported in $[0, 1]$ and its even extension belongs to $C^m(\mathbb{R})$. For this, we can just use the even and C^∞-smooth bump function b from Definition 7.C.1 and set $\phi(x) := b(x)$.

- The point set X is quasi-uniform with constant independent of f and n. This means that the separation distance

$$q_{X,\Omega} := \frac{1}{2} \min_{x,x' \in X : x \neq x'} \|x - x'\|_2$$

satisfies $q_{X,\Omega} \leq h_{X,\Omega} \leq c_{\mathrm{qu}} q_{X,\Omega}$ for a constant c_{qu} independent of f and n. In our case, we have $q_{X,\Omega} = 1/(2N)$, and hence we can set $c_{\mathrm{qu}} := \sqrt{d}$.

Step 3: Adapting the argument of Mirzaei (2015). Let f_n be the moving least squares approximation of f with evaluation points X. By Corollary 4.5 in Wendland (2004), f_n is in C^m since the weight function is also in C^m. Hence, the norms $\|f - f_n\|_{C^m}$ and $\|f - f_n\|_{W^m_\infty}$ are equivalent, where $W^m_p(\Omega)$ is the Sobolev space of smoothness m with the p-norm applied to the (weak) derivatives. Theorem 3.12 in Mirzaei (2015) shows that

$$\|f - f_n\|_{W_q^{|\alpha|}(\Omega)} \leq C h_{X,\Omega}^{m+s-|\alpha|-d\max\{0,1/p-1/q\}} \|f\|_{W_p^{m+s}(\Omega)}$$

for $p \in [1, \infty), q \in [1, \infty], s \in [0, 1)$ and a multi-index α satisfying $m > |\alpha| + d/p$. We would obtain (a) by setting $s = 0$, $p = q = \infty$, and $|\alpha| = k$. However, setting $p = \infty$ is not allowed by the assumptions of the theorem, and setting $|\alpha| = m$ for $m = k$ is also not allowed. Hence, we need to show that the theorem can be extended to $p = \infty$ and $|\alpha| = m$ in the special case $s = 0$ and $q = \infty$. The assumption $p < \infty$ is used for the Sobolev extension operator, but it is noted in the proof that $p = \infty$ is allowed for $s = 0$. The only other point where $p < \infty$ and $|\alpha| < m$ are required is in the invocation of Eq. (3.4) in Lemma 3.3 of Mirzaei (2015). However, for the special case $s = 0$, $p = q = \infty$ and $|\alpha| \leq m$, the statement of Lemma 3.3 also holds, as is shown by the Bramble-Hilbert lemma (cf. Lemma (4.3.8) in Brenner et al., 2008), which has also been employed by (Li, 2016) for the same purpose.

Step 4: Runtime bound. For (b) and (c), we note that due to the local support of the weight function, evaluating the moving least squares approximation at a point $x \in \mathcal{X}$ mainly requires the solution of a regression problem with $O_{m,d}(1)$ variables and evaluation points. This is shown, for example, above Lemma 3.6 in Mirzaei (2015). The moving least squares method does not require any pre-computations. □

7.D.1 Proofs for Relation between Sampling and Log-Partition Estimation

For analyzing thermodynamic integration, we are going to use Hoeffding's inequality in the form stated and proved in Theorem 6.10 in Steinwart and Christmann (2008).

Theorem 7.D.1 (Hoeffding's inequality). *Let $(\Omega, \mathcal{A}, P)$ be a probability space, $a < b$ be two real numbers, $n \geq 1$ be an integer, and $\xi_1, \ldots, \xi_n : \Omega \to [a, b]$ be independent random variables. Then, for all $\tau > 0$, we have*

$$P\left(\frac{1}{n}\sum_{i=1}^n (\xi_i - \mathbb{E}_P \xi_i) \geq (b - a)\sqrt{\frac{\tau}{2n}}\right) \leq e^{-\tau} .$$

Theorem 7.3.4 (Convergence of thermodynamic integration)**.** *Given $N \in \mathbb{N}_{\geq 1}$ and a sampling algorithm producing samples from approximate distributions $\tilde{P}_{\beta f}$, consider the following algorithm:*

- *Sample $\beta_1, \ldots, \beta_N \sim \mathcal{U}([0,1])$ independently.*
- *Draw $X_i \sim \tilde{P}_{\beta_i f}$ independently.*
- *Output $\tilde{L}_f := \frac{1}{N}\sum_{i=1}^N f(X_i)$.*

Then, for $\delta > 0$, we have

$$|L_f - \tilde{L}_f| \leq |L_f - \mathbb{E}\tilde{L}_f| + 2\|f\|_\infty \sqrt{\frac{\log(2/\delta)}{2N}}$$

with probability $\geq 1 - \delta$, where

$$|L_f - \mathbb{E}\tilde{L}_f| \leq 2\|f\|_\infty \sup_{\beta \in [0,1]} D_{\mathrm{TV}}(P_{\beta f}, \tilde{P}_{\beta f}),$$
$$|L_f - \mathbb{E}\tilde{L}_f| \leq |f|_1 \sup_{\beta \in [0,1]} W_1(P_{\beta f}, \tilde{P}_{\beta f}).$$

Proof. Obviously, we have

$$|\tilde{L}_f - L_f| \leq |L_f - \mathbb{E}\tilde{L}_f| + |\tilde{L}_f - \mathbb{E}\tilde{L}_f| \ .$$

Step 1: Bounding the second term. For bounding the second term, we use Hoeffding's inequality (Theorem 7.D.1) with $\xi_i := f(X_i)$, $n = N$ and $\tau := \log(2/\delta)$. Ignoring null sets, we can choose $b = \|f\|_\infty$ and $a = -\|f\|_\infty$. We then obtain

$$\tilde{L}_f - \mathbb{E}\tilde{L}_f \geq 2\|f\|_\infty \sqrt{\frac{\log(2/\delta)}{2N}}$$

with probability $\leq \exp(-\log(2/\delta)) = \delta/2$. By applying the same argument to $\xi_i = -f(X_i)$ and applying the union bound, we obtain

$$|\tilde{L}_f - \mathbb{E}\tilde{L}_f| \leq 2\|f\|_\infty \sqrt{\frac{\log(2/\delta)}{2N}}$$

with probability $\geq 1 - \delta$.

Step 2: Bounding the first term. We use

$$\begin{aligned}|L_f - \mathbb{E}\tilde{L}_f| &= \left|\mathbb{E}_{\beta \sim \mathcal{U}([0,1])}\mathbb{E}_{x \sim P_{\beta f}}[f(x)] - \mathbb{E}_{\beta \sim \mathcal{U}([0,1])}\mathbb{E}_{x \sim \tilde{P}_{\beta f}}[f(x)]\right| \\ &\leq \sup_{\beta \in \mathcal{U}([0,1])} \left|\mathbb{E}_{x \sim P_{\beta f}}[f(x)] - \mathbb{E}_{x \sim \tilde{P}_{\beta f}}[f(x)]\right| \ . \end{aligned} \tag{7.14}$$

We can assume that $|f|_1 \neq 0$ since the bound is clear otherwise. Using the dual formulation of the 1-Wasserstein distance and that $f/|f|_1$ is 1-Lipschitz, we directly obtain

$$\begin{aligned}\left|\mathbb{E}_{x \sim P_{\beta f}}[f(x)] - \mathbb{E}_{x \sim \tilde{P}_{\beta f}}[f(x)]\right| &= |f|_1 \left|\mathbb{E}_{x \sim P_{\beta f}}[f(x)/|f|_1] - \mathbb{E}_{x \sim \tilde{P}_{\beta f}}[f(x)/|f|_1]\right| \\ &\leq |f|_1 W_1(\tilde{P}_{\beta f}, P_{\beta f}) \ .\end{aligned}$$

Similarly, the bound on the TV distance follows from an alternative formulation of the TV distance (see e.g. Gibbs and Su, 2002) given by

$$D_{\mathrm{TV}}(P, Q) = \frac{1}{2} \sup_{g: \|g\|_\infty \leq 1} \left|\int g \, \mathrm{d}P - \int g \, \mathrm{d}Q\right| \ . \qquad \square$$

Remark 7.D.2. In Theorem 7.3.4, we can hope for a better bound in terms of the sup-log distance. For example, suppose that $\tilde{P}_{\beta f} = P_{\beta g}$, where g is an approximation of f that is independent of β. Since g is only determined up to a constant shift, we can assume that $L_g = L_f$. Then,

$$\begin{aligned}
|L_f - \mathbb{E}\tilde{L}_f| &= \left|\mathbb{E}_{\beta\sim\mathcal{U}([0,1])}\mathbb{E}_{x\sim P_{\beta f}}[f(x)] - \mathbb{E}_{\beta\sim\mathcal{U}([0,1])}\mathbb{E}_{x\sim P_{\beta g}}[f(x)]\right| \\
&\leq \left|\mathbb{E}_{\beta\sim\mathcal{U}([0,1])}\mathbb{E}_{x\sim P_{\beta f}}[f(x)] - \mathbb{E}_{\beta\sim\mathcal{U}([0,1])}\mathbb{E}_{x\sim P_{\beta g}}[g(x)]\right| \\
&\quad + \left|\mathbb{E}_{\beta\sim\mathcal{U}([0,1])}\mathbb{E}_{x\sim P_{\beta g}}[g(x)] - \mathbb{E}_{\beta\sim\mathcal{U}([0,1])}\mathbb{E}_{x\sim P_{\beta g}}[f(x)]\right| \\
&\leq |L_f - L_g| + \|g - f\|_\infty = 0 + D_{\text{sup-log}}(P_f, P_g) = D_{\text{sup-log}}(P_f, P_g) \ .
\end{aligned}$$

However, in the general case, the approach in Eq. (7.14) of taking the supremum over β cannot yield such a good bound. This can be seen by considering indicator functions $f = a\mathbb{1}_A$ and $g = (a+\delta)\mathbb{1}_A$. Instead, it appears that it would be necessary to obtain bounds depending on β and f and show that their integral over $\beta \in [0,1]$ is sufficiently small for all f. ◀

Theorem 7.3.5 (Convergence of bisection sampling)**.** *Let $m \geq 1, B \geq 0$ and $M \in \mathbb{N}_0$. Let $f \in \mathcal{F}_{d,m,B}$ and let $\tilde{L}$ be a log-partition estimator with worst-case error $E \geq 0$ on $\mathcal{F}_{d,m,B}$. Let $f \in C^m(\mathcal{X})$ and let $\tilde{P}_f$ be the distribution of samples produced by* BisectionSampling*$(f, \tilde{L}, M)$ in Algorithm 17. Then,*

$$D_{\text{sup-log}}(P_f, \tilde{P}_f) \leq 2MdE + 2^{-M}d\|f\|_{C^1} \ .$$

Proof. **Step 1: Log-density analysis.** We want to show that $\tilde{P}_f$ has a density $\tilde{p}_f$ and bound $\|\log\tilde{p}_f - \log p_f\|_\infty$. Partition $\mathcal{X}$ into 2^{Md} cubes of side length 2^{-M}. Since a density is only defined up to a null set, it suffices to consider an arbitrary x in the interior of one of these cubes, which we fix in the following. We denote the corresponding cube by $\mathcal{Z}^{(Md)}$. We can then find exactly one sequence $\mathcal{Z}^{(0)} = \mathcal{X}, \mathcal{Z}^{(1)}, \ldots, \mathcal{Z}^{(Md)}$ of hyperrectangles which could have been visited during the execution of Algorithm 17 to obtain x. Since Algorithm 17 samples uniformly from $\mathcal{Z}^{(Md)}$ and the volume of $\mathcal{Z}^{(Md)}$ is 2^{-Md}, we have the density

$$\tilde{p}_f(x) = 2^{Md}\tilde{P}_f(\mathcal{Z}^{(Md)}) \ .$$

On the other hand, a simple integration argument shows that the target density satisfies

$$\inf_{x'\in\mathcal{Z}^{(Md)}} p_f(x') \leq 2^{Md}P_f(\mathcal{Z}^{(Md)}) \leq \sup_{x'\in\mathcal{Z}^{(Md)}} p_f(x') \ .$$

This yields

$$\begin{aligned}
|\log\tilde{p}_f(x) - \log p_f(x)| &\leq \left|\log(2^{Md}\tilde{P}_f(\mathcal{Z}^{(Md)})) - \log(2^{Md}P_f(\mathcal{Z}^{(Md)}))\right| \\
&\quad + \left|\sup_{x'\in\mathcal{Z}^{(Md)}} \log p_f(x') - \inf_{x''\in\mathcal{Z}^{(Md)}} \log p_f(x'')\right| \ .
\end{aligned}$$

Step 2: Bounding the second term. Since $\mathcal{Z}^{(Md)}$ is an axis-aligned cube with side length 2^{-M}, we have for $x', x'' \in \mathcal{Z}^{(Md)}$:

$$|f(x') - f(x'')| \leq \sum_{i=1}^d \|\partial^i f\|_\infty 2^{-M} \leq 2^{-M}d\|f\|_{C^1} \ .$$

Step 3: Bounding the first term. We can simplify

$$\left|\log(2^{Md}\tilde{P}_f(\mathcal{Z}^{(Md)})) - \log(2^{Md}P_f(\mathcal{Z}^{(Md)}))\right| = \left|\log(\tilde{P}_f(\mathcal{Z}^{(Md)})) - \log(P_f(\mathcal{Z}^{(Md)}))\right| .$$

We want to show by induction over $k \in \{0, \ldots, Md\}$ that

$$\left|\log(\tilde{P}_f(\mathcal{Z}^{(k)})) - \log(P_f(\mathcal{Z}^{(k)}))\right| \leq 2kE ,$$

which will then yield the desired error bound for $k = Md$. This is obviously true for $k = 0$. Now, suppose that it is true for some $k \in \{0, \ldots, Md - 1\}$. Consider a partition of $\mathcal{Z}^{(k)}$ into two equal-sized sub-hyperrectangles $\mathcal{Z}_1$ and $\mathcal{Z}_2$ as in Algorithm 17 such that $\mathcal{Z}^{(k+1)} = \mathcal{Z}_i$ for some $i \in \{1, 2\}$. Then,

$$\tilde{P}_f(\mathcal{Z}^{(k+1)}) = \sigma(\tilde{L}_{f_{\mathcal{Z}_i}} - \tilde{L}_{f_{\mathcal{Z}_{3-i}}})\tilde{P}_f(\mathcal{Z}^{(k)}) ,$$

which also holds for $i = 2$ since the sigmoid function σ satisfies $\sigma(-u) = 1 - \sigma(u)$ for all $u \in \mathbb{R}$. Moreover, we have

$$\begin{aligned} P_f(\mathcal{Z}^{(k+1)}) &= \frac{P_f(\mathcal{Z}_i)}{P_f(\mathcal{Z}_i) + P_f(\mathcal{Z}_{3-i})} P_f(\mathcal{Z}^{(k)}) = \frac{e^{L_{f_{\mathcal{Z}_i}}}}{e^{L_{f_{\mathcal{Z}_i}}} + e^{L_{f_{\mathcal{Z}_{3-i}}}}} P_f(\mathcal{Z}^{(k)}) \\ &= \sigma(L_{f_{\mathcal{Z}_i}} - L_{f_{\mathcal{Z}_{3-i}}})P_f(\mathcal{Z}^{(k)}) . \end{aligned}$$

By definition of the functions $f_{\mathcal{Z}_{i'}}$, $i' \in \{1, 2\}$ in Algorithm 17, since the side-lengths h_j of $\mathcal{Z}_{i'}$ satisfy $h_j \leq 1$, we have $\|f_{\mathcal{Z}_{i'}}\|_{C^m} \leq \|f\|_{C^m} \leq B$, which means $f_{\mathcal{Z}_{i'}} \in \mathcal{F}_{d,m,B}$. Hence, by assumption, the log-partition error is

$$|\tilde{L}_{f_{\mathcal{Z}_{i'}}} - L_{f_{\mathcal{Z}_{i'}}}| \leq \varepsilon .$$

Now, the log-sigmoid function $h(u) := \log \sigma(u)$ satisfies $h'(u) = \frac{\sigma(u)(1-\sigma(u))}{\sigma(u)} = 1 - \sigma(u) \in (0, 1)$ and is therefore 1-Lipschitz. Hence,

$$\begin{aligned} \left|\log\left(\tilde{P}_f(\mathcal{Z}^{(k+1)})\right) - \log\left(P_f(\mathcal{Z}^{(k+1)})\right)\right| &\leq \left|\log\left(\tilde{P}_f(\mathcal{Z}^{(k)})\right) - \log\left(P_f(\mathcal{Z}^{(k)})\right)\right| \\ &\quad + \left|h(\tilde{L}_{f_{\mathcal{Z}_i}} - \tilde{L}_{f_{\mathcal{Z}_{3-i}}}) - h(L_{f_{\mathcal{Z}_i}} - L_{f_{\mathcal{Z}_{3-i}}})\right| \\ &\leq 2kE + 2E = 2(k+1)E , \end{aligned}$$

which completes the induction. □

7.D.2 Proofs for Relation to Optimization

Proposition 7.3.6 (Optimization by approximate sampling). *Let Q be a probability distribution on $\mathcal{X}$. Then, for any $\delta \in (0, 1]$ and $\varepsilon > 0$,*

(a) $Q(\{x \in \mathcal{X} \mid f(x) \leq \varepsilon L_{f/\varepsilon} - \varepsilon \log(1/\delta) - \varepsilon D_{\text{sup-log}}(P_{f/\varepsilon}, Q)\}) \leq \delta$,
(b) $Q(\{x \in \mathcal{X} \mid f(x) \leq \varepsilon L_{f/\varepsilon} - \varepsilon \log(1/\delta)\}) \leq \delta + D_{\text{TV}}(P_{f/\varepsilon}, Q)$,
(c) $Q(\{x \in \mathcal{X} \mid f(x) < \varepsilon L_{f/\varepsilon} - \varepsilon \log(2/\delta) - 2\delta^{-1}|f|_1 W_1(P_{f/\varepsilon}, Q)\}) \leq \delta$.

Proof.

(a) Suppose $D_{\text{sup-log}}(P_f, Q) < \infty$. Then, $Q = P_g$ for some g. For almost every $x \in \mathcal{X}$, we have the implications

$$\begin{aligned} f(x) \leq L_f - \log(1/\delta) - D_{\text{sup-log}}(P_f, Q) &\Leftrightarrow \bar{f}(x) \leq L_{\bar{f}} - \log(1/\delta) - \|\bar{f} - \bar{g}\|_\infty \\ &\Rightarrow \bar{g}(x) \leq L_{\bar{f}} - \log(1/\delta) \\ &\Leftrightarrow \bar{g}(x) \leq L_{\bar{g}} - \log(1/\delta) \ . \end{aligned}$$

Hence,

$$\begin{aligned} &Q(\{x \in \mathcal{X} \mid f(x) \leq L_f - \log(1/\delta) - D_{\text{sup-log}}(P_f, Q)\}) \\ &\leq Q(\{x \in \mathcal{X} \mid \bar{g}(x) \leq L_{\bar{g}} - \log(1/\delta)\}) \\ &= P_{\bar{g}}(\{x \in \mathcal{X} \mid \bar{g}(x) \leq L_{\bar{g}} - \log(1/\delta)\}) \overset{\text{Lemma 7.1.3}}{\leq} \delta \ . \end{aligned}$$

By using f/ε instead of f and multiplying both sides of the inequality by ε, we obtain

$$Q(\{x \in \mathcal{X} \mid f(x) \leq \varepsilon L_{f/\varepsilon} - \varepsilon \log(1/\delta) - \varepsilon D_{\text{sup-log}}(P_{f/\varepsilon}, Q)\}) \leq \delta \ .$$

(b) The TV norm bound follows from Lemma 7.1.3 because for the considered event A,

$$Q(A) \leq P_{f/\varepsilon}(A) + \sup_{A'} |Q(A) - P_{f/\varepsilon}(A')| = P_{f/\varepsilon}(A) + D_{\text{TV}}(P_{f/\varepsilon}, Q) \ .$$

(c) Let $\tilde{\varepsilon} > 0$. By definition of the Wasserstein distance, there exist random variables $X \sim P_{f/\varepsilon}$ and $Y \sim Q$ on a common probability space $(\Omega, \mathcal{F}, P_\Omega)$ such that $\mathbb{E}\|X - Y\|_2 \leq W_1(P_{f/\varepsilon}, Q) + \tilde{\varepsilon}$. By the Markov inequality, we then have

$$\|X - Y\|_2 \leq 2(W_1(P_{f/\varepsilon}, Q) + \tilde{\varepsilon})/\delta$$

with probability $\geq 1 - \delta/2$. Moreover, by Lemma 7.1.3, we have

$$f(X) > \varepsilon L_{f/\varepsilon} - \varepsilon \log(2/\delta)$$

with probability $\geq 1 - \delta/2$. By the union bound, we hence have

$$f(Y) > f(X) - |f|_1 \|X - Y\|_2 > \varepsilon L_{f/\varepsilon} - \varepsilon \log(2/\delta) - 2\delta^{-1}|f|_1(W_1(P_{f/\varepsilon}, Q) + \tilde{\varepsilon})$$

with probability $\geq 1 - \delta$. Since $\tilde{\varepsilon} > 0$ was arbitrary, the claim follows. □

7.E Proofs for Algorithms

7.E.1 Proofs for Approximation-based Algorithms

Proofs for Piecewise Constant Approximation

To study the error of piecewise constant approximation, we study the log-partition function of linear functions f. A first step is achieved using the following lemma:

Lemma 7.E.1. *Let*

$$r : \mathbb{R} \to \mathbb{R}, t \mapsto \log\left(\int_{-1/2}^{1/2} \exp(tu)\,\mathrm{d}u\right) = \begin{cases} 0 & , t = 0 \\ \log\left(t^{-1}(\exp(t/2) - \exp(-t/2))\right) & , t \neq 0 \ . \end{cases}$$

Then, r is even and $(1/2)$-Lipschitz with $r(t) \geq 0$ for all t and we have more generally for $a > 0$ and $t_1, \ldots, t_d \in \mathbb{R}$:

$$\log\left(\int_{[-a/2,a/2]^d} \exp\left(\sum_{k=1}^d t_k u_k\right)\,\mathrm{d}u\right) = d\log(a) + \sum_{k=1}^d r(at_k) \ .$$

Proof. It follows from a simple symmetry argument that r is even. We have

$$\int_{-1/2}^{1/2} \exp(tu)\,\mathrm{d}u \geq \int_{-1/2}^{1/2} (1 + tu)\,\mathrm{d}u = 1 \ ,$$

which shows $r(t) \geq 0$. Moreover, $\exp(hu) \in [\exp(-h/2), \exp(h/2)]$ for $h > 0$ and $u \in [-1/2, 1/2]$. Using the mean value theorem of integration, we obtain

$$\begin{aligned} r(t+h) - r(t) &= \log\left(\int_{-1/2}^{1/2} \exp(tu)\exp(hu)\,\mathrm{d}u\right) - \log\left(\int_{-1/2}^{1/2} \exp(tu)\,\mathrm{d}u\right) \\ &\in [-h/2, h/2] \ , \end{aligned}$$

which shows that r is 1/2-Lipschitz.

For the more general integral, we use that the integrand is a product of one-dimensional functions to decompose

$$\begin{aligned} \log\left(\int_{[-a/2,a/2]^d} \exp\left(\sum_{k=1}^d t_k u_k\right)\,\mathrm{d}u\right) &= \log\left(\prod_{k=1}^d \int_{-a/2}^{a/2} \exp\left(t_k u_k\right)\,\mathrm{d}u_k\right) \\ &\overset{\text{Subst. } u_k = av_k}{=} \log\left(\prod_{k=1}^d \int_{-1/2}^{1/2} \exp\left(t_k a v_k\right) a\,\mathrm{d}v_k\right) \\ &= d\log(a) + \sum_{k=1}^d r(at_k) \ . \end{aligned}$$ □

Another ingredient for the analysis of piecewise constant approximation is to analyze the global error through the errors on individual subcubes:

Lemma 7.E.2. *Let $f, g : \mathcal{X} \to \mathbb{R}$ be bounded and measurable. Let $\mathcal{X}_i$ be a partition of $\mathcal{X}$. Let $L_f(\mathcal{X}_i) := \log\left(\int_{\mathcal{X}_i} \exp(f(x))\,\mathrm{d}x\right)$. Then,*

$$\inf_i [L_f(\mathcal{X}_i) - L_g(\mathcal{X}_i)] \leq L_f - L_g \leq \sup_i [L_f(\mathcal{X}_i) - L_g(\mathcal{X}_i)] \ .$$

Proof. We prove the second inequality here, the first one follows analogously. Let $s := \sup_i [L_f(\mathcal{X}_i) - L_g(\mathcal{X}_i)]$. Then,

$$L_f = \log\left(\sum_i \exp(L_f(\mathcal{X}_i))\right) \leq \log\left(\sum_i \exp(s)\exp(L_g(\mathcal{X}_i))\right) = L_g + s \ .$$ □

We now prove convergence rates for piecewise constant approximation, using a combination of different approaches:

Theorem 7.4.1 (Convergence rate of piecewise constant approximation)**.** *Let $m \geq 1$ and $n = N^d$ as above. If $g_{f,n}$ is a piecewise constant interpolant as above, we have*

$$\sup_{f \in \mathcal{F}_{d,m,B}} |L_f - L_{g_{f,n}}| = \begin{cases} \Theta_{m,d}(Bn^{-1/d}) & , \text{ if } m = 1 \text{ or } Bn^{-1/d} > 1 \\ \Theta_{m,d}(\max\{B, B^2\}n^{-2/d}) & , \text{ otherwise.} \end{cases}$$
$$\sup_{f \in \mathcal{F}_{d,m,B}} D_{\text{sup-log}}(P_f, P_{g_{f,n}}) = \Theta_{m,d}(Bn^{-1/d}) \ .$$

Proof. Recall from Section 7.4.1 that $g_{f,n}$ is piecewise constant on the cubes $\mathcal{X}_i$, interpolating f in the cube centers $x^{(i)}$.

Step 1: Lipschitz-type upper bounds. Let $f \in \mathcal{F}_{d,m,B}$. Since f is $Bd^{1/2}$-Lipschitz by Lemma 7.B.1, it is easy to see that $\|f - g_{f,n}\|_\infty \leq O_{m,d}(B/N) = O_{m,d}(Bn^{-1/d})$. Then, it follows directly from Proposition 7.2.2 that $|L_f - L_{g_{f,n}}| \leq \|f - g_{f,n}\|_\infty \leq O_{m,d}(Bn^{-1/d})$ and $D_{\text{sup-log}}(P_f, P_{g_{f,n}}) \leq 2\|f - g_{f,n}\|_\infty \leq O_{m,d}(Bn^{-1/d})$.

Step 2: Lipschitz-type lower bound for log-partition with $m = 1$. If $m = 1$, it follows that

$$\sup_{f \in \mathcal{F}_{d,m,B}} |L_f - L_{g_{f,n}}| \geq e_n(\mathcal{F}_{d,m,B}, S_L, D_{\text{abs}}) \overset{\text{Theorem 7.2.3}}{\geq} \Omega_{m,d}(Bn^{-1/d}) \ .$$

Step 3: Lipschitz-type lower bound for sampling. Take $f(x) = \beta(x_1 + \cdots + x_d)$, where $\beta = B/d$, such that $f \in \mathcal{F}_{d,m,B}$. Pick the cube $\mathcal{X}_1 = (0, 1/N)^d$. Then, we have

$$D_{\text{sup-log}}(P_f, P_{g_{f,n}}) = \|\bar{f} - \bar{g}_{f,n}\|_\infty \geq \frac{1}{2}\left(\sup_{x \in \mathcal{X}_1} f(x) - \inf_{x \in \mathcal{X}_1} f(x)\right) = \frac{1}{2}B/N = \frac{1}{2}Bn^{-1/d} \ .$$

Step 4: Lower bound for log-partition with $m \geq 2$. As in Step 3, take $f(x) = \beta(x_1 + \cdots + x_d)$, where $\beta = B/d$, such that $f \in \mathcal{F}_{d,m,B}$. To prove a lower bound on $L_f - L_{g_{f,n}}$, we follow Lemma 7.E.2 and lower-bound the errors $L_f(\mathcal{X}_i) - L_{g_{f,n}}(\mathcal{X}_i)$ on individual subcubes $\mathcal{X}_i$.

Step 4.1: First lower bound. Fix a subcube $\mathcal{X}_i$ and set $a = 1/N$ and $t := \frac{\partial f}{\partial x_k}(x^{(i)}) = (\beta, \ldots, \beta)$. Denote the volume of $\mathcal{X}_i$ by $V_n = 1/n = a^d$. Since $g_{f,n}$ is constant on $\mathcal{X}_i$, we have

$$\begin{aligned} & L_f(\mathcal{X}_i) - L_{g_{f,n}}(\mathcal{X}_i) \\ = \quad & \log\left(\exp(f(x^{(i)})) \int_{\mathcal{X}_i} \exp(\langle t, x - x^{(i)} \rangle) \,\mathrm{d}x\right) - \log\left(V_n \exp(f(x^{(i)}))\right) \\ \overset{\text{Lemma 7.E.1}}{=} \quad & d\log(a) + \left(\sum_{k=1}^d r(at_k)\right) - \log(V_n) = \sum_{k=1}^d r(at_k) = dr(\beta/N) \\ \overset{\text{Lemma 7.E.4}}{\geq} \quad & cd\min\{|\beta/N|, |\beta/N|^2\} \geq \Omega_{m,d}(\min\{Bn^{-1/d}, B^2n^{-2/d}\}) \ , \end{aligned}$$

This lower bound is independent of i, hence by Lemma 7.E.2, we obtain

$$L_f - L_{g_{f,n}} \geq \Omega_{m,d}(\min\{Bn^{-1/d}, B^2n^{-2/d}\}) \ .$$

Step 4.2: Second lower bound. The lower bound above implicitly uses the strong convexity of $\exp(f)$. However, all of the curvature in $\exp(f)$ comes from exp and none from f. This is not sufficient in the case $B \ll 1$, where the quadratic dependency on B in $B^2 n^{-2/d}$ can be improved. For the case $B < 1$, we put the curvature into f by setting $f(x) := \beta \sum_{k=1}^d x_k^2$. We then have $\partial_k f(x) = 2\beta x_k$ and $\partial_k^2 f(x) = 2\beta$. Hence, we set $\beta := B/(2d)$ to ensure that $f \in \mathcal{F}_{d,m,B}$. We now again consider a subcube $\mathcal{X}_i$, for which we compute

$$\begin{aligned}
\int_{\mathcal{X}_i} \exp(g_{f,n}(x)) \,\mathrm{d}x &= V_n \exp(f(x^{(i)})) \;, \\
\int_{\mathcal{X}_i} \exp(f(x)) \,\mathrm{d}x &= \exp(f(x^{(i)})) \int_{\mathcal{X}_i} \exp(f(x) - f(x^{(i)})) \,\mathrm{d}x \\
&\geq \exp(f(x^{(i)})) \int_{\mathcal{X}_i} \big(1 + (f(x) - f(x^{(i)}))\big) \,\mathrm{d}x \\
&= \exp(f(x^{(i)})) \left(V_n + \int_{\mathcal{X}_i} \left(\sum_{k=1}^d \beta (x_k - x_k^{(i)})^2 \right) \mathrm{d}x \right) \\
&= \exp(f(x^{(i)})) \left(V_n + d a^{d-1} \beta \left[\frac{1}{3} u^3 \right]_{-a/2}^{a/2} \right) \\
&= \exp(f(x^{(i)})) V_n \left(1 + d \frac{2}{3 \cdot 2^3} \beta a^2 \right) \;.
\end{aligned}$$

Hence, we have

$$L_f(\mathcal{X}_i) - L_g(\mathcal{X}_i) \geq \log\left(1 + \frac{d}{12} \beta n^{-2/d}\right) = \log\left(1 + \frac{1}{24} B n^{-2/d}\right) \geq \frac{1}{48} B n^{-2/d} \;,$$

where we used $\log'(x) = 1/x \geq 1/2$ for $x \in [1, 2]$ in the last step.

Step 5: Better upper bound for log-partition for $m \geq 2$. Let $m \geq 2$, $f \in \mathcal{F}_{d,m,B}$ and $B n^{-1/d} \leq 1$. We define the piece-wise first-order approximant h_n, where for x in the interior of $\mathcal{X}_i$, we set

$$h_n(x) = f(x^{(i)}) + \langle \nabla f(x^{(i)}), x - x^{(i)} \rangle \;.$$

Our goal is to use

$$|L_f - L_{g_{f,n}}| \leq |L_f - L_{h_n}| + |L_{h_n} - L_g| \;.$$

Step 5.1: Bounding the first term. To bound the first term, we will bound $\|f - h_n\|_\infty$. Let $\delta(t) = f(x^{(i)} + t(x - x^{(i)})) - h_n(x^{(i)} + t(x - x^{(i)}))$. Then, we use Taylor's theorem to bound

$$f(x) - h_n(x) = \delta(1) = \delta(0) + 1 \cdot \delta'(0) + \frac{1^2}{2} \cdot \delta''(\xi) \;,$$

where $\xi \in (0, 1)$. Since h_n is constructed such that $\delta(0) = \delta'(0) = 0$, we have

$$|f(x) - h_n(x)| = \frac{1}{2} |\delta''(\xi)| = \frac{1}{2} (x - x^{(i)})^\top [\nabla^2 f(x^{(i)} + \xi(x - x^{(i)}))] (x - x^{(i)})$$

$$\leq d^2 \|x - x^{(i)}\|_\infty^2 \|f\|_{C^2} \leq \frac{d^2 B}{(2N)^2} = O_{m,d}(BN^{-2}) = O_{m,d}(Bn^{-2/d}) \ .$$

This shows $\|f - h_n\|_\infty \leq O_{m,d}(Bn^{-2/d})$ and therefore $|L_f - L_{h_n}| \leq O_{m,d}(Bn^{-2/d})$.

Step 5.2: Bounding the second term. To bound $|L_{h_n} - L_{g_{f,n}}|$, we follow Lemma 7.E.2 and bound the errors $|L_{h_n}(\mathcal{X}_i) - L_{g_{f,n}}(\mathcal{X}_i)|$ on individual subcubes $\mathcal{X}_i$. Fix a subcube $\mathcal{X}_i$ and set $a = 1/N$ and $t := \frac{\partial f}{\partial x_k}(x^{(i)})$. Denote the volume of $\mathcal{X}_i$ by $V_n = 1/n = a^d$. Since $g_{f,n}$ is constant on $\mathcal{X}_i$, we have

$$\begin{aligned}
& |L_{h_n}(\mathcal{X}_i) - L_{g_{f,n}}(\mathcal{X}_i)| \\
= \quad & \left|\log\left(\exp(f(x^{(i)})) \int_{\mathcal{X}_i} \exp(\langle t, x - x^{(i)}\rangle)\,\mathrm{d}x\right) - \log\left(V_n \exp(f(x^{(i)}))\right)\right| \\
\overset{\text{Lemma 7.E.1}}{=} \quad & \left| d\log(a) + \left(\sum_{k=1}^d r(at_k)\right) - \log(V_n)\right| = \left|\sum_{k=1}^d r(at_k)\right| \\
\overset{\text{Lemma 7.E.4}}{\leq} \quad & \sum_{k=1}^d C \min\{|at_k|, |at_k|^2\} \leq Cd \min\{Bn^{-1/d}, B^2 n^{-2/d}\} \ ,
\end{aligned}$$

where we used $|t_k| \leq \|f\|_{C^m}$ and $a = 1/N = n^{-1/d}$ in the last step. Using Lemma 7.E.2, we now obtain

$$|L_{h_n} - L_{g_{f,n}}| \leq Cd \min\{Bn^{-1/d}, B^2 n^{-2/d}\} \ ,$$

which concludes the upper bound. □

The following two lemmas provide some additional bounds that have been used in the previous proof:

Lemma 7.E.3. *Let*

$$H : \mathbb{R} \to \mathbb{R}, t \mapsto \sum_{k=0}^\infty \frac{t^k}{(k+1)!} = \begin{cases} \frac{\exp(t)-1}{t} & , t \neq 0 \\ 1 & , t = 0 \ . \end{cases}$$

$$h : \mathbb{R} \to \mathbb{R}, t \mapsto \sum_{k=0}^\infty \frac{t^k}{(k+2)!} = \begin{cases} \frac{H(t)-1}{t} & , t \neq 0 \\ 1/2 & , t = 0 \ . \end{cases}$$

Then, H and h are C^∞. Moreover,

- *H and H' are increasing on $[0, \infty)$ with $H(0) = 1$, $H'(0) = 1/2$, and $H(t) > 0$ for all $t \in (0, \infty)$.*
- *h and h' are increasing on $[0, \infty)$ with $h(0) = 1/2$, $h'(0) = 1/6$.*

Proof. Using the series representation, it follows that H and h are C^∞. We also directly obtain $H(0) = 1$ and $H'(0) = 1/2$ as well as $h(0) = 1/2$ and $h'(0) = 1/6$. Moreover, it follows that $H', H'' \geq 0$ on $[0, \infty]$, which implies that H' and H are increasing on $[0, \infty)$. The inequality $H(t) > 0$ follows from the non-series representation together with $H(0) = 1$. The results for h can be derived analogously. □

Lemma 7.E.4. *Consider the function r from Lemma 7.E.1. Then, there exist constants $c, C > 0$ such that*

$$c \min\{|t|, |t|^2\} \leq r(t) \leq C \min\{|t|, |t|^2\}$$

for all $t \in \mathbb{R}$.

Proof. Since r is an even function, it suffices to prove the inequalities for $t \geq 0$.

Step 1: Simplifying the derivative. First, we compute the derivative of r for $t \neq 0$:

$$\begin{aligned} r'(t) &= \frac{t}{\exp(t/2) - \exp(-t/2)} \cdot \left(\frac{\exp(t/2) + \exp(-t/2)}{2t} - \frac{\exp(t/2) - \exp(-t/2)}{t^2} \right) \\ &= \frac{1}{2} \frac{\exp(t) + 1}{\exp(t) - 1} - \frac{1}{t} = \frac{1}{2} + \frac{1}{\exp(t) - 1} - \frac{1}{t} \\ &= \frac{1}{2} + \frac{1}{t} \left(\frac{1}{\left(\frac{\exp(t) - 1}{t} \right)} - 1 \right) = \frac{1}{2} + \frac{1}{t} \left(\frac{1}{H(t)} - 1 \right) = \frac{1}{2} + \frac{1}{t} \cdot \frac{1 - H(t)}{H(t)} \\ &= \frac{1}{2} - \frac{h(t)}{H(t)} , \end{aligned} \tag{7.15}$$

where we used the functions h and H from Lemma 7.E.3. Since h and H are also continuous in $t = 0$, the equation

$$r'(t) = \frac{1}{2} - \frac{h(t)}{H(t)}$$

holds for all $t \in \mathbb{R}$.

Step 2: Upper bound. Since r is 1/2-Lipschitz, we obtain $r(t) = r(t) - r(0) \leq t/2$ for $t \geq 0$. For $t \in [0, 1]$, we can use $r'(0) = 0$ to obtain

$$r(t) = r(0) + tr'(0) + \frac{t^2}{2} r''(\xi) \leq \frac{t^2}{2} \sup_{u \in [0,1]} r''(u) \leq Ct^2$$

for some $\xi \in [0, t]$ and the constant $C = \frac{1}{2} \sup_{u \in [0,1]} r''(u) > 0$. This shows $r(t) \leq O(\min\{t, t^2\})$.

Step 3: Lower bound. We can now use Lemma 7.E.3 to further simplify for $t \geq 0$

$$r'(t) = \frac{1}{2} - \frac{h(t)}{H(t)} \geq \frac{1}{2} - \frac{1/2}{H(t)} = \frac{1}{2} \left(1 - \frac{1}{H(t)} \right) =: \tilde{r}(t) .$$

We find that

$$\tilde{r}'(t) = \frac{H'(t)}{2H(t)^2}$$

satisfies $\tilde{r}'(0) = 1/4$ and $\tilde{r}'(t) > 0$ for all $t \in [0, \infty)$.

Set $\tilde{c} := \inf_{t \in [0,1]} \tilde{r}'(t) > 0$. Since $H(0) = 1$, we have $\tilde{r}(0) = 0$. For $t \in [0, 1]$, this yields

$$\tilde{r}(t) = r(0) + \int_0^t \tilde{r}(u) \, \mathrm{d}u \geq \tilde{c}t .$$

For $t > 1$, since $\tilde{r}$ is increasing, we obtain $\tilde{r}(t) \geq \tilde{c}$. In total, this yields

$$\tilde{r}(t) \geq \tilde{c} \min\{1, t\} \qquad \text{for } t \in [0, \infty) \ .$$

Now, we obtain for $t \in [0, 1]$

$$r(t) = r(0) + \int_0^t r'(u) \, \mathrm{d}u \geq \int_0^t \tilde{r}(u) \, \mathrm{d}u \geq \int_0^t \tilde{c} \min\{1, u\} \, \mathrm{d}u = \frac{\tilde{c}}{2} t^2$$

and for $t > 1$

$$\begin{aligned} r(t) &= r(1) + \int_1^t r'(u) \, \mathrm{d}u \geq r(1) + \int_1^t \tilde{r}(u) \, \mathrm{d}u \geq r(1) + \int_1^t \tilde{c} \min\{1, u\} \, \mathrm{d}u \\ &\geq \frac{\tilde{c}}{2} + \tilde{c}(t-1) \geq \frac{\tilde{c}}{2} t \ . \end{aligned}$$

□

Proofs for Density-based Approximation

In the following, we analyze how log-partition and sampling errors can be bounded in terms of the underlying unnormalized densities:

Proposition 7.4.2 (Density approximation bounds)**.** *Let $p, q : \mathbb{R} \to [0, \infty)$ be bounded and measurable such that $I_p, I_q > 0$, where $I_p := \int_{\mathcal{X}} p(x) \, \mathrm{d}x$. Define probability distributions P, Q with densities p/I_p and q/I_q, respectively. Then,*

$$\begin{aligned} |\log I_p - \log I_q| &\leq \log\left(\frac{1}{1 - \|p - q\|_\infty / I_p}\right) \quad \textit{if } \|p - q\|_\infty < I_p, \\ D_{\mathrm{TV}}(P, Q) &\leq \frac{\|p - q\|_\infty}{\max\{I_p, I_q\}} \leq \frac{\|p - q\|_\infty}{I_p} \ . \end{aligned}$$

Proof. **Step 1: Log-partition function.** We have

$$\begin{aligned} \log I_p - \log I_q &\leq \log I_p - \log\left(\int_{\mathcal{X}} (p(x) - \|p - q\|_\infty) \, \mathrm{d}x\right) = \log\left(\frac{I_p}{I_p - \|p - q\|_\infty}\right) \\ &= \log\left(\frac{1}{1 - \|p - q\|_\infty / I_p}\right) \\ \log I_q - \log I_p &\leq \log\left(\int_{\mathcal{X}} (p(x) + \|p - q\|_\infty) \, \mathrm{d}x\right) - \log I_p = \log\left(\frac{I_p + \|p - q\|_\infty}{I_p}\right) \\ &= \log(1 + \|p - q\|_\infty / I_p) \\ &\leq \log\left(\frac{1}{1 - \|p - q\|_\infty / I_p}\right) . \end{aligned}$$

Step 2: Total variation distance. Without loss of generality, assume that $I_q \leq I_p = 1$, such that $\max\{I_p, I_q\} = 1$. Define the normalized density function $\tilde{q} := q/I_q$. Then,

$$\begin{aligned} D_{\mathrm{TV}}(P, Q) &= \frac{1}{2} \int_{\mathcal{X}} |p(x) - \tilde{q}(x)| \, \mathrm{d}x \leq \frac{1}{2} \int_{\mathcal{X}} (|p(x) - q(x)| + q(x)|(1 - 1/I_q)|) \, \mathrm{d}x \\ &\leq \frac{1}{2} \|p - q\|_\infty + \frac{1}{2} I_q |1 - 1/I_q| \leq \|p - q\|_\infty \ , \end{aligned}$$

where we have used $I_q|1 - 1/I_q| = |I_q - 1| = |I_q - I_p| \leq \|p - q\|_\infty$ in the last step. □

Next, we turn to bounding $\|p_f\|$ in terms of $\|f\|$. Our first result will provide a bound for the sup-norm:

Lemma 7.E.5. *For $f \in C^1(\mathcal{X})$, we have*

$$\|p_f\|_\infty \leq O_{m,d}(\max\{1, \|f\|_{C^1}\}^d) \ .$$

Proof. By Lemma 7.B.1, we have $|f|_1 \leq d^{1/2}\|f\|_{C^1}$, and hence

$$\begin{aligned} \|p_f\|_\infty \quad &= \quad \frac{\exp(M_f)}{Z_f} = \exp(M_f - L_f) \\ &\overset{\text{Lemma 7.1.3}}{\leq} \exp(d \log(1 + 3d^{-1/2}|f|_1)) = (1 + 3d^{-1/2}|f|_1)^d \\ &\leq \quad O_{m,d}(\max\{1, \|f\|_{C^1}\}^d) \ . \end{aligned}$$ □

The following lemma helps to bound the norms of products, which occur in the derivatives of $\exp(f)$:

Lemma 7.E.6. *Let $f, g \in C^m(\mathcal{X})$ for $m \geq 0$. Then, $\|fg\|_{C^m} \leq 2^m \|f\|_{C^m}\|g\|_{C^m}$.*

Proof. We use induction on m. This claim is obviously true for $m = 0$. Now suppose it is true for m and that $f, g \in C^{m+1}(\mathcal{X})$. Take any $\alpha \in \mathbb{N}_0^d$ with $|\alpha|_1 = m + 1$. Then, we can write $\partial_\alpha = \partial_\beta\partial_j$ for some $j \in \{1, \ldots, d\}$ and $\beta \in \mathbb{N}_0^d$ with $|\beta| = m$. We then have

$$\begin{aligned} \|\partial_\alpha(fg)\|_\infty &= \|\partial_\beta\partial_j(fg)\|_\infty = \|\partial_\beta((\partial_j f)g + f(\partial_j g))\|_\infty \\ &\leq \|(\partial_j f)g + f(\partial_j g)\|_{C^m} \leq 2^m \left(\|\partial_j f\|_{C^m}\|g\|_{C^m} + \|f\|_{C^m}\|\partial_j g\|_{C^m}\right) \\ &\leq 2^{m+1}\|f\|_{C^{m+1}}\|g\|_{C^{m+1}} \ . \end{aligned}$$

Moreover, for any $\alpha \in \mathbb{N}_0^d$ with $|\alpha|_1 \leq m$, we have

$$\|\partial_\alpha(fg)\|_\infty \leq \|fg\|_{C^m} \leq 2^m\|f\|_{C^m}\|g\|_{C^m} \leq 2^{m+1}\|f\|_{C^{m+1}}\|g\|_{C^{m+1}} \ .$$

This completes the proof of the induction step. □

Now, we can indeed bound higher-order norms of $\exp(f)$:

Lemma 7.E.7. *Let $f \in C^m(\mathcal{X})$ for $m \geq 0$. Then,*

$$\|e^f\|_{C^m} \leq 2^{m(m-1)/2} \max\{1, \|f\|_{C^m}\}^m \|e^f\|_\infty \ .$$

Proof. We prove this by induction on m. The claim is obviously true for $m = 0$. Now, suppose the claim is true for some $m \geq 0$ and let $f \in C^{m+1}(\mathcal{X})$. Take any $\alpha \in \mathbb{N}_0^d$ with $|\alpha|_1 = m + 1$. Then, we can write $\partial_\alpha = \partial_\beta\partial_j$ for some $j \in \{1, \ldots, d\}$ and $\beta \in \mathbb{N}_0^d$ with $|\beta| = m$. Thus,

$$\begin{aligned} \|\partial_\alpha e^f\|_\infty &= \|\partial_\beta\partial_j e^f\|_\infty = \|\partial_\beta(\partial_j f)e^f\|_\infty \\ &\leq \|(\partial_j f)e^f\|_{C^m} \overset{\text{Lemma 7.E.6}}{\leq} 2^m\|\partial_j f\|_{C^m}\|e^f\|_{C^m} \\ &\leq 2^m\|f\|_{C^{m+1}} 2^{m(m-1)/2} \max\{1, \|f\|_{C^m}\}^m\|e^f\|_\infty \\ &\leq 2^{(m+1)m/2} \max\{1, \|f\|_{C^{m+1}}\}^{m+1}\|e^f\|_\infty \ . \end{aligned}$$

Moreover, for any $\alpha \in \mathbb{N}_0^d$ with $|\alpha|_1 \leq m$, we have by the induction hypothesis

$$\begin{aligned}\|\partial_\alpha e^f\|_\infty \leq \|e^f\|_{C^m} &\leq 2^{m(m-1)/2} \max\{1, \|f\|_{C^m}\}^m \|e^f\|_\infty \\ &\leq 2^{(m+1)m/2} \max\{1, \|f\|_{C^{m+1}}\}^{m+1} \|e^f\|_\infty \ .\end{aligned}$$

This completes the proof of the induction step. □

It might be possible to improve the dependence on m in the previous lemma; we ignored this since we do not study the dependence on m. Combining the previous lemmas, we arrive at a higher-order norm bound for the density:

Theorem 7.4.3 (Density norm). *For $m \geq 1$, we have*

$$\sup_{f \in \mathcal{F}_{d,m,B}} \|p_f\|_{C^m} = \Theta_{m,d}\left(\max\{1, B\}^{m+d}\right)$$

and this asymptotic rate is attained by $f_{d,m,B}(x) = Bd^{-1}(x_1 + \cdots + x_d)$.

Proof. **Step 1: Upper bound.** For $f \in \mathcal{F}_{d,m,B}$, we have

$$\begin{aligned}\|p_f\|_{C^m} &= \|e^f/Z_f\|_{C^m} = \frac{1}{Z_f}\|e^f\|_{C^m} \\ &\overset{\text{Lemma 7.E.7}}{\leq} 2^{m(m-1)/2} \max\{1, \|f\|_{C^m}\}^m \|e^f/Z_f\|_\infty \\ &\overset{\text{Lemma 7.E.5}}{\leq} 2^{m(m-1)/2} \max\{1, \|f\|_{C^m}\}^m O_d(\max\{1, \|f\|_{C^1}\}^d) \\ &\leq O_{m,d}(\max\{1, \|f\|_{C^m}\}^{d+m}) \ .\end{aligned}$$

Step 2: Lower bound. Consider $f : \mathcal{X} \to \mathbb{R}, x \mapsto Bd^{-1}(x_1 + \cdots + x_d)$. A simple calculation yields $\|f\|_{C^m} = B$, hence $f \in \mathcal{F}_{d,m,B}$. Moreover, we have $M_f = f(1, \ldots, 1) = B$. We can also calculate

$$\begin{aligned}Z_f &= \int_{\mathcal{X}} \left(\prod_{i=1}^d \exp(Bd^{-1}x_i)\right) \mathrm{d}x = \left(\int_0^1 \exp(Bd^{-1}x_1)\, \mathrm{d}x_1\right)^d \\ &= (B^{-1}d[\exp(Bd^{-1}) - 1])^d \leq (B^{-1}d\exp(Bd^{-1}))^d = (B^{-1}d)^d \exp(B) \ .\end{aligned}$$

Finally, we compute

$$\partial_{x_1}^m p_f(x)|_{x=(1,\ldots,1)} = \partial_{x_1}^m \frac{\exp(f(x))}{Z_f}\bigg|_{x=(1,\ldots,1)} = (Bd^{-1})^m \frac{f(1,\ldots,1)}{Z_f} \geq (Bd^{-1})^{d+m} \ .$$

We know that $\sup_{x\in\mathcal{X}} p_f(x) \geq 1$ because p_f is a density on a unit-volume domain. Hence, we conclude $\|p_f\|_{C^m} \geq \Omega_{m,d}(\max\{1, B\}^{d+m})$. □

7.E.2 Proofs for Simple Stochastic Algorithms

Proofs for Rejection Sampling

The following bound for rejection sampling with uniform proposal distribution is a consequence of the general rejection sampling bounds in Lemma 7.2.7.

Proposition 7.4.4 (Convergence of rejection sampling)**.** *Let $m \geq 1$ and let $f \in C^1(\mathcal{X})$. Then, the distribution $\tilde{P}_f$ produced by* RejectionSampling*(f, M_f, n) (see Algorithm 16) satisfies*

$$\begin{aligned} D_{\text{sup-log}}(P_f, \tilde{P}_f) &\leq \min\left\{2\|f\|_\infty, \exp\left(2\|f\|_\infty - n/\|p_f\|_\infty\right)\right\} \\ D_{\text{TV}}(P_f, \tilde{P}_f) &\leq \min\{1, 2\|f\|_\infty\} \exp(-n/\|p_f\|_\infty) \\ &\leq O_{m,d}(\min\{1, \|f\|_\infty\} \max\{1, \|f\|_{C^1}\}^m n^{-m/d}) \ . \end{aligned}$$

Proof. **Step 1: Rejection probability.** Set $g(x) \coloneqq M_f$. Since

$$\|p_f\|_\infty = \frac{\exp(M_f)}{Z_f} = \frac{Z_g}{Z_f} \ ,$$

the overall rejection probability p_R from Lemma 7.2.7 satisfies

$$p_R \overset{\text{Lemma 7.2.7}}{\leq} \exp(-nZ_f/Z_g) = \exp(-n/\|p_f\|_\infty) \ .$$

Step 2: Sup-log distance. We have $D_{\text{sup-log}}(P_f, P_g) = \|\bar{f} - \bar{g}\|_\infty = \|\bar{f}\|_\infty \leq 2\|f\|_\infty$. We then obtain from Lemma 7.2.7 that

$$\begin{aligned} D_{\text{sup-log}}(P_f, \tilde{P}_f) &\leq \min\{D_{\text{sup-log}}(P_f, P_g), p_R(\exp(D_{\text{sup-log}}(P_f, P_g)) - 1)\} \\ &\leq \min\left\{2\|f\|_\infty, \exp\left(2\|f\|_\infty - n/\|p_f\|_\infty\right)\right\} \ . \end{aligned}$$

Step 3: TV distance. For the TV distance, we compute

$$D_{\text{TV}}(P_f, P_g) \leq D_{\text{sup-log}}(P_f, P_g) \leq 2\|f\|_\infty$$

and also employ the trivial bound $D_{\text{TV}}(P_f, P_g) \leq 1$. From Lemma 7.2.7, we obtain

$$D_{\text{TV}}(P_f, \tilde{P}_f) = p_R D_{\text{TV}}(P_f, P_g) \leq \exp(-n/\|p_f\|_\infty) \min\{1, 2\|f\|_\infty\} \ .$$

Using that $\exp(-x) \leq O_{m,d}(x^{-m/d})$ for $x > 0$, we obtain

$$\exp(-n/\|p_f\|_\infty) \leq \|p_f\|_\infty^{m/d} n^{-m/d} \overset{\text{Theorem 7.4.3}}{\leq} O_{m,d}\left(\max\{1, \|f\|_{C^1}\}^m n^{-m/d}\right) \ . \qquad \square$$

Proofs for Monte Carlo Log-partition

For analyzing the Monte Carlo log-partition estimator, we are going to use Bernstein's inequality in the form stated and proved in Theorem 6.10 in Steinwart and Christmann (2008):

Theorem 7.E.8 (Bernstein's inequality)**.** *Let $(\Omega, \mathcal{A}, P)$ be a probability space, $B > 0, \sigma > 0, n \geq 1$. Moreover, let $\xi_1, \ldots, \xi_n : \Omega \to \mathbb{R}$ be independent random variables with*

- $\mathbb{E}_P \xi_i = 0$
- $\|\xi_i\|_\infty \leq B$
- $\mathbb{E}_P \xi_i^2 \leq \sigma^2$

for all $i \in \{1, \dots, n\}$. Then,

$$P\left(\frac{1}{n}\sum_{i=1}^{n}\xi_i \geq \sqrt{\frac{2\sigma^2\tau}{n}} + \frac{2B\tau}{3n}\right) \leq e^{-\tau}$$

for all $\tau > 0$.

We will use Bernstein's inequality since it yields better bounds than Hoeffding's inequality (Theorem 7.D.1) when n is large and σ is significantly smaller than B, which will be the case in the following proof.

Theorem 7.4.5 (Upper bounds for MC log-partition)**.** *Let $f : \mathcal{X} \to \mathbb{R}$ be Lipschitz, let $X_1, \dots, X_n \sim \mathcal{U}(\mathcal{X})$ be independent and let*

$$\tilde{L}_n \coloneqq \log S_n, \qquad S_n \coloneqq \frac{1}{n}\sum_{i=1}^{n}\exp(f(X_i)).$$

Then, for any $\delta \in (0, 1]$, the following convergence rates hold:

(a) ***Optimization regime:*** *If $n \leq 4\log(2/\delta)(1 + 3d^{-1/2}|f|_1)^d$, we have*

$$|\tilde{L}_n - L_f| \leq d^{1/2}(\log(1/\delta))^{1/d}|f|_1 n^{-1/d} + \log(4\log(2/\delta)) + d\log(1 + 3d^{-1/2}|f|_1)$$

with probability $\geq 1 - \delta$.

(b) ***Quadrature regime:*** *If $n \geq 4\log(2/\delta)(1 + 3d^{-1/2}|f|_1)^d$, we have*

$$|\tilde{L}_n - L_f| \leq 4\log(2/\delta)^{1/2}(1 + 3d^{-1/2}|f|_1)^{d/2}n^{-1/2}$$

with probability $\geq 1 - \delta$.

Proof. Without loss of generality, we assume that f is shifted such that $M_f = 0$.

(a) **Step A.1: Simple one-sided bound.** We have

$$\tilde{L}_n - L_f \leq 0 - L_f \overset{\text{Lemma 7.1.3}}{\leq} d\log(1 + 3d^{-1/2}|f|_1) \ .$$

Step A.2: Bounding the other side. Define the empirical maximum

$$M_n \coloneqq \max_{i \in \{1, \dots, n\}} f(X_i) \ .$$

Since $\tilde{L}_n = \log(\sum_{i=1}^n \exp(f(X_i))) - \log(n) \geq M_n - \log(n)$, we obtain

$$L_f - \tilde{L}_n \leq 0 - \tilde{L}_n \leq \log(n) - M_n \leq \log(4\log(2/\delta)) + d\log(1 + 3d^{-1/2}|f|_1) - M_n \ .$$

It remains to provide a lower bound on M_n. Define $R \coloneqq (\log(1/\delta))^{1/d}n^{-1/d}$.

- **Case 1:** $R \geq 1$. In this case, for all $x \in \mathcal{X}$, we have

$$f(x) = f(x) - M_f \geq -d^{1/2}|f|_1 \geq -d^{1/2}|f|_1 R = -d^{1/2}(\log(1/\delta))^{1/d}|f|_1 n^{-1/d} \ ,$$

which implies that

$$-M_n \leq d^{1/2}(\log(1/\delta))^{1/d}|f|_1 n^{-1/d}$$

with probability 1.

- **Case 2:** $R \le 1$**.** Let x^* be a maximizer of f. In this case, there exists an axis-aligned subcube $\tilde{\mathcal{X}}$ of $\mathcal{X}$ with side length R containing x^*. For each $x \in \tilde{\mathcal{X}}$, we have $f(x) = f(x) - f(x^*) \ge -|f|_1 d^{1/2} R$. Moreover,

$$\begin{aligned} P(M_n \le -|f|_1 d^{1/2} R) &= P(f(X_1) \le -|f|_1 d^{1/2} R)^n \\ &\le P(X_1 \notin \tilde{\mathcal{X}})^n = (1 - P(X_1 \in \tilde{\mathcal{X}}))^n = (1 - R^d)^n \\ &\le e^{-R^d n} = \delta \ . \end{aligned}$$

(b) Define $\tau := \log(2/\delta)$. To apply Bernstein's inequality to $\xi_i := e^{f(X_i)} - e^{L_f}$, we need a sup-bound and a variance bound.
Step B.1: Sup-bound. Since $e^{f(\mathcal{X})} \subseteq [0, 1]$ by assumption on f, we have $\|\xi_i\|_\infty \le 1$.
Step B.2: Variance bound. Since $e^{f(\mathcal{X})} \subseteq [0, 1]$, we have:

$$\operatorname{Var}(\xi_i) = \operatorname{Var} e^{f(X_i)} \le \mathbb{E}[(e^{f(X_i)})^2] \le \mathbb{E} e^{f(X_i)} = e^{L_f} \ .$$

Step B.3: Non-logarithmic concentration. By Bernstein's inequality, we obtain for $\tau \in (0, 1)$:

$$P\left(|S_n - e^{L_f}| \ge \sqrt{\frac{2e^{L_f}\tau}{n}} + \frac{2\tau}{3n}\right) \le 2e^{-\tau} = \delta \ .$$

Step B.4: Lower bound on expectation. From Lemma 7.1.3, we obtain

$$e^{L_f} = e^{L_f - M_f} \ge e^{-d \log(1 + 3d^{-1/2}|f|_1)} = (1 + 3d^{-1/2}|f|_1)^{-d} \ .$$

Step B.5: Logarithmic concentration. By combining the previous steps, we obtain with probability $\ge 1 - 2e^{-\tau}$:

$$\left|\frac{S_n - e^{L_f}}{e^{L_f}}\right| \le \sqrt{\frac{2\tau}{n(1 + 3d^{-1/2}|f|_1)^{-d}}} + \frac{2\tau}{3n(1 + 3d^{-1/2}|f|_1)^{-d}} \ .$$

Since we assumed $n \ge 4\log(2/\delta)(1 + 3d^{-1/2}|f|_1)^d = 4\tau(1 + 3d^{-1/2}|f|_1)^d$, the right-hand-side is less than $1/2$. Since the logarithm is 2-Lipschitz on $[1/2, 3/2]$, we obtain

$$\begin{aligned} |\tilde{L}_n - L_f| &= \left|\log\left(\frac{S_n}{e^{L_f}}\right)\right| \\ &= \left|\log\left(1 + \frac{S_n - e^{L_f}}{e^{L_f}}\right) - \log(1)\right| \\ &\le 2\left|\frac{S_n - e^{L_f}}{e^{L_f}}\right| \\ &\le 2\left(\sqrt{\frac{2\tau}{n(1 + 3d^{-1/2}|f|_1)^{-d}}} + \frac{2\tau}{3n(1 + 3d^{-1/2}|f|_1)^{-d}}\right) \end{aligned}$$

By using the assumption $n \ge 4\log(2/\delta)(1 + 3d^{-1/2}|f|_1)^d = 4\tau(1 + 3d^{-1/2}|f|_1)^d$ from (b), we can further bound

$$|\tilde{L}_n - L_f| \le 2\left(\sqrt{\frac{2\tau}{n(1 + 3d^{-1/2}|f|_1)^{-d}}}\right.$$

$$
\begin{aligned}
&\quad + \frac{2\tau}{3n^{1/2}(4\tau(1+3d^{-1/2}|f|_1)^d)^{1/2}(1+3d^{-1/2}|f|_1)^{-d}}\Bigg) \\
&= (\sqrt{8}+2/3)\sqrt{\frac{\tau}{n(1+3d^{-1/2}|f|_1)^{-d}}} \\
&\leq 4(1+3d^{-1/2}|f|_1)^{d/2}\tau^{1/2}n^{-1/2} . \qquad \square
\end{aligned}
$$

Proofs for Monte Carlo Sampling

We now prove a simple lower bound for a simple Monte Carlo sampling algorithm. We use the TV distance, but the general approach could also be used to prove lower bounds for the sup-log and 1-Wasserstein distances.

Theorem 7.4.6 (Lower bound for MC sampling)**.** *Let $f : \mathcal{X} \to \mathbb{R}$ be bounded and measurable. Let $X_1, \ldots, X_n \sim \mathcal{U}(\mathcal{X})$ and let the random index $I \in \{1, \ldots, n\}$ be distributed as*

$$P(I = i) = \frac{\exp(f(X_i))}{\sum_{j=1}^n \exp(f(X_j))}.$$

Consider the distribution $\tilde{P}_f$ of the random sample X_I. Then, for all $B > 0$ and $n \geq 1$ with $Bn^{-1/d} \geq 4d\log(4d)$,

$$\sup_{f \in \mathcal{F}_{d,m,B}} D_{\mathrm{TV}}(P_f, \tilde{P}_f) \geq \frac{1}{2} .$$

Proof. Set $f(x) \coloneqq -\beta(x_1 + \cdots + x_d)$, where $\beta \coloneqq Bd^{-1}$, such that $f \in \mathcal{F}_{d,m,B}$. For $\delta \in (0,1]$, set $\mathcal{X}_\delta \coloneqq [0,\delta]^d$. Then, similar as in Lemma 7.E.1, we can compute

$$Z(\delta) \coloneqq \int_{\mathcal{X}_\delta} \exp(f(x)) \, \mathrm{d}x = \left(\frac{1-\exp(-\beta\delta)}{\beta}\right)^d .$$

We then obtain

$$P_f(\mathcal{X}_\delta) = \frac{Z(\delta)}{Z(1)} = \left(\frac{1-\exp(-\beta\delta)}{1-\exp(-\beta)}\right)^d \geq (1-\exp(-\beta\delta))^d \geq 1 - d\exp(-\beta\delta) ,$$

where we used Bernoulli's inequality in the last step. Setting $\delta \coloneqq \log(4d)/\beta$, we obtain

$$P_f(\mathcal{X}_\delta) \geq 1 - d\exp(-\beta\delta) \geq \frac{3}{4} .$$

On the other hand, we have

$$\tilde{P}_f(\mathcal{X}_\delta) = P(X_I \in \mathcal{X}_\delta) \leq \sum_{j=1}^n P(X_j \in \mathcal{X}_\delta) = n\delta^d = n\frac{(\log(4d))^d}{\beta^d} = n\frac{(d\log(4d))^d}{B^d} .$$

Hence, if $Bn^{-1/d} \geq 4d\log(4d)$, then $\tilde{P}_f(\mathcal{X}_\delta) \leq 4^{-d}$, which implies

$$D_{\mathrm{TV}}(P_f, \tilde{P}_f) \geq P_f(\mathcal{X}_\delta) - \tilde{P}_f(\mathcal{X}_\delta) \geq \frac{3}{4} - 4^{-d} \geq \frac{1}{2} . \qquad \square$$

7.E.3 Proofs for Variational Formulation

The following simple lemma is central to our lower bound for the variational formulation:

Lemma 7.4.7. *For a model of the form $g(x) = \varphi(x)^* H \varphi(x)$ as above, we have*

$$L_g^{\mathrm{OPT}}(Q) = \sup_{\tilde{Q} \in \mathcal{P}(\mathcal{X}) : \Sigma_{\tilde{Q}} = \Sigma_Q} L_g(\tilde{Q}) \ .$$

Proof. We have

$$\begin{aligned}
L_g^{\mathrm{OPT}}(Q) &= \sup_{P \in \mathcal{P}(\mathcal{X})} \mathrm{tr}[H\Sigma_P] - \inf_{\tilde{P}, \tilde{Q} \in \mathcal{P}(\mathcal{X}) : \Sigma_{\tilde{P}} = \Sigma_P, \Sigma_{\tilde{Q}} = \Sigma_Q} D_{\mathrm{KL}}(\tilde{P} \parallel \tilde{Q}) \\
&= \sup_{\Sigma \in \mathcal{K}} \mathrm{tr}[H\Sigma] - \inf_{\tilde{P}, \tilde{Q} \in \mathcal{P}(\mathcal{X}) : \Sigma_{\tilde{P}} = \Sigma, \Sigma_{\tilde{Q}} = \Sigma_Q} D_{\mathrm{KL}}(\tilde{P} \parallel \tilde{Q}) \\
&= \sup_{\Sigma \in \mathcal{K}} \sup_{\tilde{P}, \tilde{Q} \in \mathcal{P}(\mathcal{X}) : \Sigma_{\tilde{P}} = \Sigma, \Sigma_{\tilde{Q}} = \Sigma_Q} \mathrm{tr}[H\Sigma] - D_{\mathrm{KL}}(\tilde{P} \parallel \tilde{Q}) \\
&= \sup_{\tilde{Q} \in \mathcal{P}(\mathcal{X}) : \Sigma_{\tilde{Q}} = \Sigma_Q} \sup_{\tilde{P} \in \mathcal{P}(\mathcal{X})} \mathrm{tr}[H\Sigma_P] - D_{\mathrm{KL}}(\tilde{P} \parallel \tilde{Q}) \\
&= \sup_{\tilde{Q} \in \mathcal{P}(\mathcal{X}) : \Sigma_{\tilde{Q}} = \Sigma_Q} L_g(\tilde{Q}). \qquad \square
\end{aligned}$$

Before proving the lower bound, we prove a Taylor-based bound on the cosine function, which is then used in the subsequent lemma to bound an integral of the form $\int_{\mathcal{X}} \exp(\beta \cos(x - z)) \, \mathrm{d}x$.

Lemma 7.E.9. *For all $x \in [-1/4, 1/4]$, we have*

$$\cos(2\pi x) \leq 1 - \frac{8}{5}\pi^2 x^2 \ .$$

Proof. For $f(x) := \cos(2\pi x)$, we have

$$\begin{aligned}
f'(x) &= -2\pi \sin(2\pi x) \\
f''(x) &= -4\pi^2 \cos(2\pi x) \\
f'''(x) &= 8\pi^3 \sin(2\pi x) \\
f^{(4)}(x) &= 16\pi^4 \cos(2\pi x)
\end{aligned}$$

Applying Taylor's theorem with the Lagrange form of the remainder for $x \in [-1/4, 1/4]$ around $x_0 = 0$, we obtain for some $\xi \in [-1/4, 1/4]$:

$$f(x) = 1 - 2\pi^2 x^2 + \frac{2}{3}\pi^4 \cos(2\pi\xi) x^4 \ .$$

Therefore, we obtain for $x \in [-1/4, 1/4]$:

$$f(x) \leq 1 - 2\pi^2 x^2 + \left(\frac{2}{3}\pi^4 (1/4)^2\right) \cdot x^2 \leq 1 - 2\pi^2(1 - 1/5)x^2 = 1 - \frac{8}{5}\pi^2 x^2 \ . \qquad \square$$

We can now bound the normalizing constant of a rescaled cosine function, which will be used in the lower bound for the variational formulation:

Lemma 7.E.10. *For any $z \in \mathbb{R}^d$, define*

$$g_z : \mathcal{X} \to \mathbb{R}, x \mapsto \sum_{i=1}^{d} \cos(2\pi(x_i - z_i)) \ .$$

Then, for $\beta > 0$,

$$Z_{\beta g_z} \leq \beta^{-d/2} e^{\beta d} \ .$$

Proof. Since g_z is 1-periodic, $\exp(\beta g_z)$ is 1-periodic. Moreover, we have $\mathcal{X} = [0,1]^d$ and hence

$$\begin{aligned} Z_{\beta g_z} &= \int_{\mathcal{X}} \exp(\beta g_z(x))\, \mathrm{d}x = \int_{\mathcal{X}} \exp(\beta g_0(x))\, \mathrm{d}x \\ &= \int_{\mathcal{X}} \exp(\beta \cos(2\pi x_1)) \cdots \exp(\beta \cos(2\pi x_d))\, \mathrm{d}x \\ &= \left(\int_0^1 \exp(\beta \cos(2\pi x_1))\, \mathrm{d}x_1 \right)^d . \end{aligned} \tag{7.16}$$

From expanding the inequality $(1 - \sqrt{\beta})^2 \geq 0$, we obtain

$$e^\beta \geq 1 + \beta \geq 2\sqrt{\beta} \ . \tag{7.17}$$

This allows us to upper-bound the one-dimensional integral in Eq. (7.16) as

$$\begin{aligned} \int_0^1 e^{\beta \cos(2\pi x_1)}\, \mathrm{d}x_1 \quad &= \quad \int_{-1/4}^{3/4} e^{\beta \cos(2\pi x)}\, \mathrm{d}x \\ &\leq \quad \int_{[-1/4,1/4]} e^{\beta \cos(2\pi x)}\, \mathrm{d}x + \int_{[1/4,3/4]} e^{\beta \cos(2\pi x)}\, \mathrm{d}x \\ &\overset{\text{Lemma 7.E.9}}{\leq} \quad \int_{[-1/4,1/4]} e^{\beta(1 - \frac{8}{5}\pi^2 x^2)}\, \mathrm{d}x + \int_{[1/4,3/4]} 1\, \mathrm{d}x \\ &\leq \quad \frac{1}{2} + e^\beta \int_{-\infty}^{\infty} \exp\left(-\frac{x^2}{2 \cdot \frac{5}{16}\pi^{-2}\beta^{-1}} \right) \\ &= \quad \frac{1}{2} + e^\beta \sqrt{2\pi \frac{5}{16} \pi^{-2} \beta^{-1}} \\ &= \quad \frac{1}{2} + e^\beta \sqrt{\frac{5\beta^{-1}}{8\pi}} \\ &\overset{\text{Eq. (7.17)}}{\leq} \quad e^\beta \left(\frac{\sqrt{\beta^{-1}}}{4} + \sqrt{\frac{5\beta^{-1}}{8\pi}} \right) \\ &\leq \quad \beta^{-1/2} e^\beta \ . \end{aligned}$$

□

Finally, we can use some elementary convex geometry to find a lower bound for the error of the variational formulation.

Theorem 7.4.8 (Lower bound for OPT relaxation)**.** *Let $\varphi : \mathcal{X} \to \mathbb{C}^N$ be continuous. Let*

$$n \coloneqq \dim_{\mathbb{C}} \mathcal{V}_{\text{lin}}, \qquad \mathcal{V}_{\text{lin}} \coloneqq \operatorname{Span}_{\mathbb{C}} \{\varphi(x)\varphi(x)^* \mid x \in \mathcal{X}\} \subseteq \mathbb{C}^{N \times N} \ .$$

In other words, n is the number of effective degrees-of-freedom of the model $g(x) = \varphi(x)^* H \varphi(x)$, and hence corresponds to the maximum number of points where such a model can interpolate arbitrary function values. Then, there exists a point $z \in \mathcal{X}$ depending only on φ, such that the periodic and analytic function

$$f : \mathcal{X} \to \mathbb{R}, x \mapsto \sum_{i=1}^{d} \cos(2\pi(x_i - z_i))$$

satisfies

$$|L_g^{\mathrm{OPT}}(\mathcal{U}([0,1])) - L_{\beta f}(\mathcal{U}([0,1]))| \geq \log\left(\frac{\beta^{d/2}}{2n+1}\right) - \|g - \beta f\|_\infty \tag{7.7}$$

for any model $g(x) = \varphi(x)^* H \varphi(x)$ and any $\beta > 0$.

Proof. **Step 1: Representability by discrete distributions.** Let $Q \coloneqq \mathcal{U}(\mathcal{X})$. We want to find a discrete distribution $\tilde{Q} = \sum_{k=1}^{M} \lambda^{(k)} \delta_{x^{(k)}}$ with $\Sigma_{\tilde{Q}} = \Sigma_Q$. Using the feature map $\Phi : \mathcal{X} \to \mathbb{C}^{k \times k}, x \mapsto \varphi(x)\varphi(x)^*$, we can write

$$\Sigma_{\tilde{Q}} = \int_{\mathcal{X}} \varphi(x)\varphi(x)^* \,\mathrm{d}\tilde{Q}(x) = \sum_{k=1}^{M} \lambda^{(k)} \Phi(x^{(k)}) \ ,$$

which shows that the matrices $\Sigma_{\tilde{Q}}$ are exactly those in the convex hull $\operatorname{conv}(\Phi(\mathcal{X}))$ of $\Phi(\mathcal{X})$. Since $\Sigma_Q \in \mathcal{K}$ by definition of $\mathcal{K}$, we first want to show that $\mathcal{K} = \operatorname{conv}(\Phi(\mathcal{X}))$. As we have just demonstrated, the inclusion $\mathcal{K} \supseteq \operatorname{conv}(\Phi(\mathcal{X}))$ is simple. Moreover, because the integral $\Sigma_Q = \int_{\mathcal{X}} \Phi(x) \,\mathrm{d}Q(x)$ is a limit of finite sums, we obtain $\mathcal{K} \subseteq \overline{\operatorname{conv}(\Phi(\mathcal{X}))}$. Since Φ is continuous and $\mathcal{X}$ is compact, $\Phi(\mathcal{X})$ is also compact. Hence, since we are in finite dimension, $\operatorname{conv}(\Phi(\mathcal{X}))$ is compact (see e.g. Proposition 2.3 in Gallier, 2008), which means that $\operatorname{conv}(\Phi(\mathcal{X})) \subseteq \mathcal{K} \subseteq \overline{\operatorname{conv}(\Phi(\mathcal{X}))} = \operatorname{conv}(\Phi(\mathcal{X}))$.

Step 2: Bounding the number of discrete points. By definition, $\mathcal{V}_{\mathrm{lin}}$ is the $\mathbb{C}$-linear span of $\Phi(\mathcal{X})$. Using Step 1, we conclude $\mathcal{K} \subseteq \mathcal{V}_{\mathrm{lin}}$. Hence, $\mathcal{K}$ is contained in the space $\mathcal{V}_{\mathrm{lin}}$ with $\dim_{\mathbb{R}} \mathcal{V}_{\mathrm{lin}} \leq 2n$. By Carathéodory's theorem (see e.g. Theorem 2.2 in Gallier, 2008), the matrix $\Sigma_Q \in \mathcal{K}$ is hence representable as a convex combination of $2n+1$ points:

$$\Sigma_Q = \sum_{k=1}^{2n+1} \lambda^{(k)} \Phi(x^{(k)}),$$

with $\lambda^{(k)} \geq 0, \sum_k \lambda^{(k)} = 1$. By setting $\tilde{Q} \coloneqq \sum_{k=1}^{2n+1} \lambda^{(k)} \delta_{x^{(k)}}$, we obtain $\Sigma_{\tilde{Q}} = \Sigma_Q$.

Step 3: Determining z. Choose an arbitrary index $k^* \in \{1, \ldots, 2n+1\}$ such that $\lambda^{(k^*)} \geq (2n+1)^{-1}$, which always exists. For such an index, we set $z \coloneqq x^{(k^*)}$.

Step 4: Lower-bounding the approximate log-partition function. Using Lemma 7.4.7, we conclude

$$\begin{aligned} L_g^{\mathrm{OPT}}(\mathcal{U}([0,1])) &\geq L_g(\tilde{Q}) = \log\left(\sum_{k=1}^{2n+1} \lambda^{(k)} \exp(g(x^{(k)}))\right) \geq \log\left(\lambda^{(k^*)} \exp(g(x^{(k^*)}))\right) \\ &\geq \log\left((2n+1)^{-1} \exp(\beta f(x^{(k^*)}) - \|g - \beta f\|_\infty)\right) \end{aligned}$$

$$= \beta d - \log(2n+1) - \|g - \beta f\|_\infty \ .$$

Step 5: Upper-bounding the true log-partition function. We have

$$L_{\beta f}(\mathcal{U}([0,1])) = \log Z_{\beta f} \overset{\text{Lemma 7.E.10}}{\leq} \beta d - \log(\beta^{d/2}) \ .$$

Step 6: Putting it together. The previous two steps immediately yield the desired bound

$$\begin{aligned} |L_g^{\mathrm{OPT}}(\mathcal{U}([0,1])) - L_{\beta f}(\mathcal{U}([0,1]))| &\geq L_g^{\mathrm{OPT}}(\mathcal{U}([0,1])) - L_{\beta f}(\mathcal{U}([0,1])) \\ &\geq \log\left(\frac{\beta^{d/2}}{2n+1}\right) - \|g - \beta f\|_\infty \ . \end{aligned}$$

□

Bibliography

Juliette Achddou, Joseph Lam-Weil, Alexandra Carpentier, and Gilles Blanchard. A minimax near-optimal algorithm for adaptive rejection sampling. In *Algorithmic Learning Theory*, 2019.

Ben Adlam and Jeffrey Pennington. The neural tangent kernel in high dimensions: Triple descent and a multi-scale theory of generalization. In *International Conference on Machine Learning*, pages 74–84. PMLR, 2020a.

Ben Adlam and Jeffrey Pennington. Understanding double descent requires a fine-grained bias-variance decomposition. In *Advances in neural information processing systems*, 2020b.

Madhu S. Advani, Andrew M. Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020. Publisher: Elsevier.

Thomas D. Ahle, Michael Kapralov, Jakob BT Knudsen, Rasmus Pagh, Ameya Velingker, David P. Woodruff, and Amir Zandieh. Oblivious sketching of high-degree polynomial kernels. In *ACM-SIAM Symposium on Discrete Algorithms*, 2020.

Rahaf Aljundi, Nikolay Chumerin, and Daniel Olmeda Reino. Identifying wrongly predicted samples: A method for active learning. In *Winter Conference on Applications of Computer Vision*, 2022.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, 2019.

Jason M. Altschuler and Kunal Talwar. Resolving the mixing time of the Langevin algorithm to its stationary distribution for log-concave sampling. *arXiv:2210.08448*, 2022.

Herbert Amann and Joachim Escher. *Analysis.* Springer, 2005.

Christos Anagnostopoulos, Fotis Savva, and Peter Triantafillou. Scalable aggregation predictive analytics. *Applied Intelligence*, 48(9):2546–2567, 2018.

Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Russ R. Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Neural Information Processing Systems*, 2019.

Rosa I. Arriaga and Santosh Vempala. Algorithmic theories of learning. In *Foundations of Computer Science*, 1999.

David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *ACM-SIAM Symposium on Discrete Algorithms*, 2007.

Jordan Ash, Surbhi Goel, Akshay Krishnamurthy, and Sham Kakade. Gone fishing: Neural active learning with Fisher embeddings. In *Neural Information Processing Systems*, 2021.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*, 2019.

Alexander Atanasov, Blake Bordelon, and Cengiz Pehlevan. Neural networks as kernel learners: The silent alignment effect. In *International Conference on Learning Representations*, 2021.

Francis Bach. Sum-of-squares relaxations for information theory and variational inference. *arXiv:2206.13285*, 2022.

Francis Bach. Information theory with kernel methods. *IEEE Transactions on Information Theory*, 69(2):752–775, 2023.

Krishna Balasubramanian, Sinho Chewi, Murat A. Erdogdu, Adil Salim, and Shunshi Zhang. Towards a theory of non-log-concave sampling: first-order stationarity guarantees for Langevin Monte Carlo. In *Conference on Learning Theory*, 2022.

Rafael Ballester-Ripoll, Enrique G. Paredes, and Renato Pajarola. Sobol tensor trains for global sensitivity analysis. *Reliability Engineering & System Safety*, 183:311–322, 2019.

Alessandro Barducci, Giovanni Bussi, and Michele Parrinello. Well-tempered metadynamics: a smoothly converging and tunable free-energy method. *Physical review letters*, 100(2):020603, 2008.

Jörg Behler. Perspective: Machine learning potentials for atomistic simulations. *Journal of Chemical Physics*, 145(17):170901, 2016.

Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical Review Letters*, 98(14):146401, 2007.

Mikhail Belkin, Siyuan Ma, and Soumik Mandal. To understand deep learning we need to understand kernel learning. In *International Conference on Machine Learning*, pages 541–549. PMLR, 2018.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Mikhail Belkin, Daniel Hsu, and Ji Xu. Two models of double descent for weak features. *SIAM Journal on Mathematics of Data Science*, 2(4):1167–1180, 2020. Publisher: SIAM.

William H. Beluch, Tim Genewein, Andreas Nürnberger, and Jan M. Köhler. The power of ensembles for active learning in image classification. In *Conference on Computer Vision and Pattern Recognition*, 2018.

Marshall Bern and David Eppstein. Approximation algorithms for geometric problems. In *Approximation Algorithms for NP-hard Problems*, pages 296–345. PWS Publishing Company, 1996.

Rajendra Bhatia. *Matrix Analysis*, volume 169. Springer Science & Business Media, 2013.

Alberto Bietti and Francis Bach. Deep equals shallow for ReLU networks in kernel regimes. In *International Conference on Learning Representations*, 2021.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Vladimir I. Bogachev. *Measure Theory, Volume 2*, volume 1. Springer Science & Business Media, 2007.

Charles Bordenave and Djalil Chafaï. Around the circular law. *Probability surveys*, 9:1–89, 2012.

Zalán Borsos, Mojmir Mutny, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. In *Neural Information Processing Systems*, 2020.

Zalán Borsos, Marco Tagliasacchi, and Andreas Krause. Semi-supervised batch active learning via bilevel optimization. In *Conference on Acoustics, Speech and Signal Processing*, 2021.

Nawaf Bou-Rabee and Martin Hairer. Nonasymptotic mixing of the MALA algorithm. *IMA Journal of Numerical Analysis*, 33(1):80–110, 2013.

Nawaf Bou-Rabee, Andreas Eberle, and Raphael Zimmer. Coupling and convergence for Hamiltonian monte carlo. *The Annals of Applied Probability*, 30(3):1209–1250, 2020.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Leo Breiman and David Freedman. How many variables should be entered in a regression equation? *Journal of the American Statistical Association*, 78(381):131–136, 1983.

Susanne C. Brenner, L. Ridgway Scott, and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*, volume 3. Springer, 2008.

Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, and Amanda Askell. Language models are few-shot learners. In *Neural Information Processing Systems*, 2020.

Siegfried Bös and Manfred Opper. Dynamics of training. In *Advances in Neural Information Processing Systems*, pages 141–147, 1997.

Erdem Bıyık, Kenneth Wang, Nima Anari, and Dorsa Sadigh. Batch active learning using determinantal point processes. *arXiv:1906.07975*, 2019.

Weipeng Cao, Xizhao Wang, Zhong Ming, and Jinzhu Gao. A review on neural networks with random weights. *Neurocomputing*, 275:278–287, 2018.

William F. Caselton and James V. Zidek. Optimal monitoring network designs. *Statistics & Probability Letters*, 2(4):223–227, 1984.

M. Emre Celebi, Hassan A. Kingravi, and Patricio A. Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, 40(1):200–210, 2013.

Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, pages 273–304, 1995.

Laming Chen, Guoxin Zhang, and Eric Zhou. Fast greedy map inference for determinantal point process to improve recommendation diversity. In *Neural Information Processing Systems*, 2018.

Lin Chen, Yifei Min, Mikhail Belkin, and Amin Karbasi. Multiple descent: Design your own generalization curve. In *Neural Information Processing Systems*, 2021.

Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru R. Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. In *NeurIPS 2022 Workshop on Score-Based Methods*, 2022.

Xiang Cheng. *The Interplay between Sampling and Optimization.* PhD thesis, University of California, Berkeley, 2020.

Xiang Cheng, Niladri S. Chatterji, Yasin Abbasi-Yadkori, Peter L. Bartlett, and Michael I. Jordan. Sharp convergence rates for Langevin dynamics in the nonconvex setting. *arXiv:1805.01648*, 2018.

Sinho Chewi, Chen Lu, Kwangjun Ahn, Xiang Cheng, Thibaut Le Gouic, and Philippe Rigollet. Optimal dimension dependence of the metropolis-adjusted langevin algorithm. In *Conference on Learning Theory*, 2021.

Sinho Chewi, Patrik Gerber, Chen Lu, Thibaut Le Gouic, and Philippe Rigollet. The query complexity of sampling from strongly log-concave distributions in one dimension. In *Conference on Learning Theory*, 2022.

Sinho Chewi, Patrik Gerber, Holden Lee, and Chen Lu. Fisher information lower bounds for sampling. In *International Conference on Algorithmic Learning Theory*, 2023.

Lenaic Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In *Neural Information Processing Systems*, 2018.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In *Neural Information Processing Systems*, 2019.

Nicolas Chopin and Mathieu Gerber. Higher-order stochastic integration through cubic stratification. *arXiv:2210.01554*, 2022.

Dan Cireşan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition.* IEEE, 2012.

Ali Civril and Malik Magdon-Ismail. Exponential inapproximability of selecting a maximum volume sub-matrix. *Algorithmica*, 65(1):159–176, 2013.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv:1511.07289*, 2015.

David A. Cohn. Neural network exploration using optimal experiment design. *Neural Networks*, 9(6):1071–1083, 1996.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations*, 2019.

Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020. Publisher: National Acad Sciences.

Stéphane d'Ascoli, Levent Sagun, and Giulio Biroli. Triple descent and the two kinds of overfitting: Where & why do they appear? In *Neural Information Processing Systems*, 2020.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace Redux-Effortless Bayesian Deep Learning. In *Neural Information Processing Systems*, 2021.

Stefano De Marchi, Robert Schaback, and Holger Wendland. Near-optimal data-independent point locations for radial basis function interpolation. *Advances in Computational Mathematics*, 23(3):317–330, 2005.

Tewodors Deneke, Habtegebreil Haile, Sébastien Lafond, and Johan Lilius. Video transcoding time prediction for proactive load balancing. In *International Conference on Multimedia and Expo*, 2014.

Volker L. Deringer, Miguel A. Caro, and Gábor Csányi. Machine learning interatomic potentials as emerging tools for materials science. *Advanced Materials*, 31(46):1902765, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Luc Devroye. Any discrimination rule can have an arbitrarily bad probability of error for finite sample size. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(2): 154–157, 1982. doi: 10.1109/TPAMI.1982.4767222.

Luc Devroye, László Györfi, and Gabor Lugosi. *A Probabilistic Theory of Pattern Recognition.* Springer Science & Business Media, 1996.

Monroe D. Donsker and SR Srinivasa Varadhan. Asymptotic evaluation of certain Markov process expectations for large time. IV. *Communications on Pure and Applied Mathematics*, 36(2): 183–212, 1983.

Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

Dheeru Dua and Casey Graff. UCI Machine Learning Repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Simon Duane, Anthony D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

Rick Durrett. *Probability: theory and examples, 5th edition*, volume 49. Cambridge university press, 2019.

Raaz Dwivedi, Yuansi Chen, Martin J. Wainwright, and Bin Yu. Log-concave sampling: Metropolis-Hastings algorithms are fast! In *Conference on Learning Theory*, 2018.

Cynthia Dwork, Guy N. Rothblum, and Salil Vadhan. Boosting and differential privacy. In *Symposium on Foundations of Computer Science*, 2010.

Stéphane d'Ascoli, Maria Refinetti, Giulio Biroli, and Florent Krzakala. Double trouble in double descent: Bias and variance (s) in the lazy regime. In *International Conference on Machine Learning*, 2020.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

David Eppstein, Sariel Har-Peled, and Anastasios Sidiropoulos. Approximate greedy clustering and distance selection for graph metrics. *Journal of Computational Geometry*, 11(1):629–652, 2020.

Runa Eschenhagen, Erik Daxberger, Philipp Hennig, and Agustinus Kristiadi. Mixtures of Laplace approximations for improved post-hoc uncertainty in deep learning. In *NeurIPS 2021 Workshop on Bayesian Deep Learning*, 2021.

Sebastian Farquhar, Yarin Gal, and Tom Rainforth. On statistical bias in active learning: How and when to fix it. In *International Conference on Learning Representations*, 2021.

Michael F. Faulkner and Samuel Livingstone. Sampling algorithms in statistical physics: a guide for statistics and machine learning. *arXiv:2208.04751*, 2022.

Tomás Feder and Daniel Greene. Optimal algorithms for approximate clustering. In *ACM Symposium on Theory of Computing*, 1988.

Herbert Federer. *Geometric Measure Theory*. Springer, 1969.

Valerii V. Fedorov. *Theory of Optimal Experiments*. Academic Press, New York, 1972.

Simon Fischer and Ingo Steinwart. Sobolev norm learning rates for regularized least-squares algorithms. *The Journal of Machine Learning Research*, 21(1):8464–8501, 2020. Publisher: JMLRORG.

Catherine Forbes, Merran Evans, Nicholas Hastings, and Brian Peacock. *Statistical Distributions*. John Wiley & Sons, 2011.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M. Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. In *Neural Information Processing Systems*, 2020.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

Daan Frenkel and Berend Smit. *Understanding Molecular Simulation: From Algorithms to Applications*. Elsevier, 2001.

Jerome H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, pages 1–67, 1991.

Nial Friel and Jason Wyse. Estimating the evidence–a review. *Statistica Neerlandica*, 66(3): 288–308, 2012.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, 2016.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, 2017.

Jean Gallier. Notes on convex sets, polytopes, polyhedra, combinatorial topology, Voronoi diagrams and Delaunay triangulations. *arXiv:0805.0292*, 2008.

Rong Ge, Holden Lee, and Jianfeng Lu. Estimating normalizing constants for log-concave distributions: Algorithms and lower bounds. In *Symposium on Theory of Computing*, 2020.

Yonatan Geifman and Ran El-Yaniv. Deep active learning over the long tail. *arXiv:1711.00941*, 2017.

Andrew Gelman and Xiao-Li Meng. Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical Science*, pages 163–185, 1998.

Amirata Ghorbani, James Zou, and Andre Esteva. Data shapley valuation for efficient batch active learning. In *Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2022.

Nikhil Ghosh and Mikhail Belkin. A Universal Trade-off Between the Model Size, Test Loss, and Training Loss of Linear Predictors. *arXiv:2207.11621*, 2022.

Alison L. Gibbs and Francis Edward Su. On choosing and bounding probability metrics. *International Statistical Review*, 70(3):419–435, 2002.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2010.

Ethan Goan and Clinton Fookes. Bayesian neural networks: An introduction and survey. *Case Studies in Applied Bayesian Data Science: CIRM Jean-Morlet Chair, Fall 2018*, pages 45–87, 2020.

Gene H. Golub and Charles F. Van Loan. Matrix Computations. *The Johns Hopkins University Press, Baltimore, USA*, 1989.

Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Neural Information Processing Systems*, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT press, 2016.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *Neural Information Processing Systems*, 2021.

Franz Graf, Hans-Peter Kriegel, Matthias Schubert, Sebastian Pölsterl, and Alexander Cavallaro. 2D image registration in ct images using radial image descriptors. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2011.

Ping Guo. A vest of the pseudoinverse learning algorithm. *arXiv:1805.07828*, 2018.

László Györfi, Michael Kohler, Adam Krzyzak, and Harro Walk. *A Distribution-Free Theory of Nonparametric Regression*. Springer Science & Business Media, 2002.

Thomas Hamm and Ingo Steinwart. Adaptive learning rates for support vector machines working on data with low intrinsic dimension. *The Annals of Statistics*, 49(6):3153–3180, 2021. Publisher: Institute of Mathematical Statistics.

Insu Han, Haim Avron, Neta Shoham, Chaewon Kim, and Jinwoo Shin. Random features for the neural tangent kernel. *arXiv:2104.01351*, 2021.

Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J. Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *Annals of statistics*, 50(2):949, 2022. Publisher: NIH Public Access.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *IEEE Conference on Computer Vision*, 2015.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv:1606.08415*, 2016.

Pablo A. Henriquez and Gonzalo A. Ruz. An empirical study of the hidden matrix rank for neural networks with random weights. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 883–888. IEEE, 2017.

Jan Hermann, Zeno Schätzle, and Frank Noé. Deep-neural-network solution of the electronic Schrödinger equation. *Nature Chemistry*, 12(10):891–897, 2020.

David Holzmüller. On the universality of the double descent peak in ridgeless regression. In *International Conference on Learning Representations*, 2021.

David Holzmüller and Francis Bach. Convergence rates for non-log-concave sampling and log-partition estimation. *arXiv:2303.03237*, 2023.

David Holzmüller and Ingo Steinwart. Training two-layer ReLU networks with gradient descent is inconsistent. *Journal of Machine Learning Research*, 23(181):1–82, 2022. URL `http://jmlr.org/papers/v23/20-830.html`.

David Holzmüller, Viktor Zaverkin, Johannes Kästner, and Ingo Steinwart. A framework and benchmark for deep batch active learning for regression. *Journal of Machine Learning Research*, 24(164):1–81, 2023.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv:1112.5745*, 2011.

Gao Huang, Guang-Bin Huang, Shiji Song, and Keyou You. Trends in extreme learning machines: A review. *Neural Networks*, 61:32–48, 2015.

Guang-Bin Huang. Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE Transactions on Neural Networks*, 14(2):274–281, 2003.

Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

Chii-Ruey Hwang. Laplace's method revisited: weak convergence of probability measures. *The Annals of Probability*, pages 1177–1182, 1980. Publisher: JSTOR.

Alexander Immer, Maciej Korzepa, and Matthias Bauer. Improving predictions of Bayesian neural nets via local linearization. In *International Conference on Artificial Intelligence and Statistics*, 2021.

Pavel Izmailov, Sharad Vikram, Matthew D. Hoffman, and Andrew Gordon Gordon Wilson. What are Bayesian neural network posteriors really like? In *International Conference on Machine Learning*, 2021.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural Tangent Kernel: Convergence and generalization in neural networks. In *Neural Information Processing Systems*, 2018.

Ziwei Ji, Justin Li, and Matus Telgarsky. Early-stopped neural networks are consistent. *Advances in Neural Information Processing Systems*, 34:1805–1817, 2021.

William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26, 1984.

Richard Jordan, David Kinderlehrer, and Felix Otto. The variational formulation of the Fokker–Planck equation. *SIAM Journal on Mathematical Analysis*, 29(1):1–17, 1998.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, and Anna Potapenko. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. Well-tuned simple nets excel on tabular datasets. In *Neural Information Processing Systems*, 2021.

Purushottam Kar and Harish Karnick. Random feature maps for dot product kernels. In *Artificial Intelligence and Statistics*, 2012.

Ioannis Katsavounidis, C.-C. Jay Kuo, and Zhen Zhang. A new initialization technique for generalized Lloyd iteration. *IEEE Signal Processing Letters*, 1(10):144–146, 1994.

Leonard Kaufman and Peter J. Rousseeuw. Finding groups in data: an introduction to cluster analysis. *Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics*, 1990.

Manohar Kaul, Bin Yang, and Christian S. Jensen. Building accurate 3d spatial networks to enable next generation intelligent transportation systems. In *International Conference on Mobile Data Management*, 2013.

Ronald W. Kennard and Larry A. Stone. Computer aided design of experiments. *Technometrics*, 11(1):137–148, 1969.

Mohammad Emtiyaz E. Khan, Alexander Immer, Ehsan Abedi, and Maciej Korzepa. Approximate inference turns deep networks into gaussian processes. In *Neural Information Processing Systems*, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

James Kirkpatrick, Brendan McMorrow, David HP Turban, Alexander L. Gaunt, James S. Spencer, Alexander GDG Matthews, Annette Obika, Louis Thiry, Meire Fortunato, and David Pfau. Pushing the frontiers of density functionals by solving the fractional electron problem. *Science*, 374(6573):1385–1389, 2021.

Scott Kirkpatrick, C. Daniel Gelatt Jr, and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

John G. Kirkwood. Statistical mechanics of fluid mixtures. *The Journal of Chemical Physics*, 3 (5):300–313, 1935.

Andreas Kirsch. Black-Box Batch Active Learning for Regression. *Transactions on Machine Learning Research*, 2023.

Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. BatchBALD: Efficient and diverse batch acquisition for deep Bayesian active learning. In *Neural Information Processing Systems*, 2019.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980, 2017.

Achim Klenke. *Probability Theory: A Comprehensive Course.* Springer, 2014.

J. D. Knowles. On the existence of non-atomic measures. *Mathematika*, 14(1):62–67, 1967. Publisher: London Mathematical Society.

Steven G. Krantz and Harold R. Parks. *A Primer of Real Analytic Functions.* Springer Science & Business Media, 2002.

Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(2), 2008.

Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being Bayesian, even just a bit, fixes overconfidence in relu networks. In *International Conference on Machine Learning*, 2020.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Neural Information Processing Systems*, 2012.

Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In *Neural Information Processing Systems*, 1994.

Peter Kuchment. An overview of periodic elliptic operators. *arXiv:1510.00971*, 2015.

Punit Kumar and Atul Gupta. Active learning query strategies for classification, regression, and clustering: a survey. *Journal of Computer Science and Technology*, 35(4):913–945, 2020.

J. Nathan Kutz. Deep learning in fluid dynamics. *Journal of Fluid Mechanics*, 814:1–4, 2017.

Jianfa Lai, Manyun Xu, Rui Chen, and Qian Lin. Generalization ability of wide neural networks on $\mathbb{R}$. *arXiv:2302.05933*, 2023.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Neural Information Processing Systems*, volume 30, 2017.

Peter Lancaster and Kes Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of Computation*, 37(155):141–158, 1981.

Pierre Simon Laplace. Mémoire sur la probabilité de causes par les évènements. *Mémoires de Mathématique et de Physique, Presentés à l'Académie Royale des Sciences, par divers Savants & lus dans ses Assemblées. Tome Sixième*, pages 621–656, 1774.

Alexander Lavin, Hector Zenil, Brooks Paige, David Krakauer, Justin Gottschlich, Tim Mattson, Anima Anandkumar, Sanjay Choudry, Kamil Rocki, Atılım Güneş Baydin, and others. Simulation intelligence: Towards a new generation of scientific methods. *arXiv:2112.03235*, 2021.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as Gaussian processes. In *International Conference on Learning Representations*, 2018.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in neural information processing systems*, pages 8570–8581, 2019.

Xiaolin Li. Error estimates for the moving least-square approximation and the element-free Galerkin method in n-dimensional spaces. *Applied Numerical Mathematics*, 99:77–97, 2016.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

Tengyuan Liang and Alexander Rakhlin. Just interpolate: Kernel "ridgeless" regression can generalize. *Annals of Statistics*, 48(3):1329–1347, 2020.

Tengyuan Liang, Alexander Rakhlin, and Xiyu Zhai. On the multiple descent of minimum-norm interpolants and restricted lower isometry of kernels. In *Conference on Learning Theory*, 2020.

Stuart Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

Philip M. Long. Properties of the after kernel. *arXiv:2105.10585*, 2021.

Miguel Lázaro-Gredilla and Aníbal R. Figueiras-Vidal. Marginalized neural network mixtures for large-scale regression. *IEEE Transactions on Neural Networks*, 21(8):1345–1351, 2010.

Yi-An Ma, Yuansi Chen, Chi Jin, Nicolas Flammarion, and Michael I. Jordan. Sampling can be faster than optimization. *Proceedings of the National Academy of Sciences*, 116(42):20881–20885, 2019.

David JC MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992a.

David JC MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992b.

Vivek Madan, Mohit Singh, Uthaipon Tantipongpipat, and Weijun Xie. Combinatorial algorithms for optimal design. In *Conference on Learning Theory*, 2019.

Wesley J. Maddox, Pavel Izmailov, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Neural Information Processing Systems*, 2019.

Neil Mallinar, James Simon, Amirhesam Abedsoltan, Parthe Pandit, Misha Belkin, and Preetum Nakkiran. Benign, tempered, or catastrophic: toward a refined taxonomy of overfitting. In *Neural Information Processing Systems*, 2022.

Oren Mangoubi and Nisheeth Vishnoi. Dimensionally tight bounds for second-order Hamiltonian Monte Carlo. In *Neural Information Processing Systems*, volume 31, 2018.

Oren Mangoubi and Nisheeth K. Vishnoi. Nonconvex sampling with the Metropolis-adjusted Langevin algorithm. In *Conference on Learning Theory*, 2019.

K. V. Mardia, J. T. Kent, and J. M. Bibby. Multivariate analysis. *Probability and mathematical statistics. Academic Press Inc.*, 1979.

Ulysse Marteau-Ferey, Francis Bach, and Alessandro Rudi. Sampling from arbitrary functions via psd models. In *Artificial Intelligence and Statistics*, 2022.

Alexander G. de G. Matthews, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 2018.

Arash Mehrjou, Ashkan Soleymani, Andrew Jesson, Pascal Notin, Yarin Gal, Stefan Bauer, and Patrick Schwab. GeneDisco: A benchmark for experimental design in drug discovery. In *International Conference on Learning Representations*, 2021.

Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 75(4):667–766, 2022. Publisher: Wiley Online Library.

Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

Davoud Mirzaei. Analysis of moving least squares approximation revisited. *Journal of Computational and Applied Mathematics*, 282:237–250, 2015.

Diganta Misra. Mish: A self regularized non-monotonic neural activation function. *arXiv:1908.08681*, 2019.

Boris Mityagin. The zero set of a real analytic function. *arXiv:1512.07276*, 2015.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, and Georg Ostrovski. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Mohamad Amin Mohamadi, Wonho Bae, and Danica J. Sutherland. Making look-ahead active learning strategies feasible with Neural Tangent Kernels. In *Advances in Neural Information Processing Systems*, 2022.

Douglas C. Montgomery. *Design and Analysis of Experiments.* John Wiley & Sons, 2017.

Jaouad Mourtada. Exact minimax risk for linear least squares, and the lower tail of sample covariance matrices. *The Annals of Statistics*, 50(4):2157–2178, 2022. Publisher: Institute of Mathematical Statistics.

Kevin P. Murphy. *Probabilistic machine learning: an introduction.* MIT press, 2022.

Vidya Muthukumar, Kailas Vodrahalli, Vignesh Subramanian, and Anant Sahai. Harmless interpolation of noisy data in regression. *IEEE Journal on Selected Areas in Information Theory*, 2020.

Nicole Mücke and Ingo Steinwart. Global minima of DNNs: The plenty pantry. *arXiv:1905.10686*, 2019.

Preetum Nakkiran. More data can hurt for linear regression: Sample-wise double descent. *arXiv:1912.07242*, 2019.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021a. Publisher: IOP Publishing.

Preetum Nakkiran, Prayaag Venkat, Sham M. Kakade, and Tengyu Ma. Optimal regularization can mitigate double descent. In *International Conference on Learning Representations*, 2021b.

Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien, and Ioannis Mitliagkas. A modern take on the bias-variance tradeoff in neural networks. In *ICML 2019 Workshop on Identifying and Understanding Deep Learning Phenomena*, 2019.

Radford M. Neal. Priors for Infinite Networks. Technical Report CRG-TR-94-1, Dept. of Computer Science, University of Toronto, 1994.

Mehdi Neshat, Bradley Alexander, Markus Wagner, and Yuanzhong Xia. A detailed comparison of meta-heuristic methods for optimising wave energy converter placements. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018.

Quynh Nguyen and Matthias Hein. The loss surface of deep and wide neural networks. In *International Conference on Machine Learning*, pages 2603–2612, 2017.

Viet Dang Nguyen. Complex powers of analytic functions and meromorphic renormalization in QFT. *arXiv:1503.00995*, 2015.

Manuel Nonnenmacher, David Reeb, and Ingo Steinwart. Which minimizer does my neural network converge to? In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2021.

Erich Novak. *Deterministic and Stochastic Error Bounds in Numerical Analysis*, volume 1349. Springer, 1988.

Erich Novak and Henryk Woźniakowski. Approximation of infinitely differentiable multivariate functions is intractable. *Journal of Complexity*, 25(4):398–404, 2009.

Roman Novak, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Fast finite width neural tangent kernel. In *International Conference on Machine Learning*, 2022.

Sebastian W. Ober and Carl Edward Rasmussen. Benchmarking the neural linear model for regression. In *Symposium on Advances in Approximate Bayesian Inference*, 2019.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv:1609.03499*, 2016.

Rafail Ostrovsky, Yuval Rabani, Leonard J. Schulman, and Chaitanya Swamy. The effectiveness of Lloyd-type methods for the k-means problem. In *IEEE Symposium on Foundations of Computer Science*, 2006.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, and Luca Antiga. Pytorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems*, 2019.

Maryam Pazouki and Robert Schaback. Bases for kernel-based spaces. *Journal of Computational and Applied Mathematics*, 236(4):575–588, 2011.

David Pfau, James S. Spencer, Alexander GDG Matthews, and W. Matthew C. Foulkes. Ab initio solution of the many-electron Schrödinger equation with deep neural networks. *Physical Review Research*, 2(3):033429, 2020.

Robert Pinsler, Jonathan Gordon, Eric Nalisnick, and José Miguel Hernández-Lobato. Bayesian batch active learning as sparse subset approximation. In *Neural Information Processing Systems*, 2019.

Remus Pop and Patric Fulop. Deep ensemble bayesian active learning: Addressing the mode collapse issue in monte carlo dropout via ensembles. *arXiv:1811.03897*, 2018.

S. James Press. *Applied Multivariate Analysis: Using Bayesian and Frequentist Methods of Inference.* Courier Corporation, 2005.

Robert James Purser, Manuel de Pondeca, and Sei-Young Park. Construction of a Hilbert curve on the sphere with an isometric parameterization of area. *Office Note*, 2009.

Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning. *arXiv:2001.04385*, 2020.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Alexander Rakhlin and Xiyu Zhai. Consistency of interpolation with laplace kernels is a high-dimensional phenomenon. In *Conference on Learning Theory*, 2019.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *arXiv:1710.05941*, 2017.

Hiranmayi Ranganathan, Hemanth Venkateswara, Shayok Chakraborty, and Sethuraman Panchanathan. Deep active learning for image regression. In *Deep Learning Applications*, pages 113–135. Springer, Singapore, 2020.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005. ISBN 978-0-262-25683-4. URL `https://doi.org/10.7551/mitpress/3206.001.0001`.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the Convergence of Adam and Beyond. In *International Conference on Learning Representations*, 2018.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM Computing Surveys*, 54(9):1–40, 2021.

Christian P. Robert. *The Bayesian Choice: From Decision-theoretic Foundations to Computational Implementation*. Springer, 2007.

Daniel J. Rosenkrantz, Richard E. Stearns, and Philip M. Lewis. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563–581, 1977.

Alessandro Rudi, Ulysse Marteau-Ferey, and Francis Bach. Finding global minima via kernel approximations. *arXiv:2012.11978*, 2020.

Hans Sagan. *Space-filling curves*. Springer Science & Business Media, 2012.

Gabriele Santin, Toni Karvonen, and Bernard Haasdonk. Sampling based approximation of linear functionals in reproducing kernel Hilbert spaces. *BIT Numerical Mathematics*, pages 1–32, 2021. Publisher: Springer.

Michael A. Sartori and Panos J. Antsaklis. A simple method to derive bounds on the size and to train multilayer neural networks. *IEEE transactions on neural networks*, 2(4):467–471, 1991.

Fotis Savva, Christos Anagnostopoulos, and Peter Triantafillou. Explaining aggregates for exploratory analytics. In *International Conference on Big Data*, 2018.

Simone Scardapane and Dianhui Wang. Randomness in neural networks: An overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(2), 2017.

Wouter F. Schmidt, Martin A. Kraaijveld, and Robert PW Duin. Feed forward neural networks with random weights. In *International Conference on Pattern Recognition*, pages 1–1. IEEE COMPUTER SOCIETY PRESS, 1992.

Nicol N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002.

Bernhard Schölkopf and Alexander J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2002.

Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In *Conference on Computational Learning Theory*, pages 416–426. Springer, 2001.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018.

Sambu Seo, Marko Wallat, Thore Graepel, and Klaus Obermayer. Gaussian process regression: Active data selection and test point rejection. In *Mustererkennung 2000*, pages 27–34. Springer, 2000.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

H. Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Workshop on Computational Learning Theory*, 1992.

Haozhe Shan and Blake Bordelon. A theory of neural tangent kernel alignment and its influence on training. *arXiv:2105.14301*, 2021.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S. Baliga, Jonathan T. Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11):2498–2504, 2003.

Apoorva Sharma, Navid Azizan, and Marco Pavone. Sketching curvature for efficient out-of-distribution detection for deep neural networks. In *Uncertainty in Artificial Intelligence*, 2021.

John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis.* Cambridge university press, 2004.

Neta Shoham and Haim Avron. Experimental design for overparameterized learning with application to single shot deep active learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. Publisher: IEEE.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, and Marc Lanctot. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587): 484–489, 2016. Publisher: Nature Publishing Group.

Kirstine Smith. On the standard deviations of adjusted and interpolated values of an observed polynomial function and its constants and the guidance they give towards a proper choice of the distribution of observations. *Biometrika*, 12(1/2):1–85, 1918.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International Conference on Machine Learning*, 2015.

Gowthami Somepalli, Avi Schwarzschild, Micah Goldblum, C. Bayan Bruss, and Tom Goldstein. SAINT: Improved neural networks for tabular data via row attention and contrastive pre-training. In *NeurIPS 2022 Table Representation Workshop*, 2022.

Stefano Spigler, Mario Geiger, Stéphane d'Ascoli, Levent Sagun, Giulio Biroli, and Matthieu Wyart. A jamming transition from under-to over-parametrization affects generalization in deep learning. *Journal of Physics A: Mathematical and Theoretical*, 52(47):474001, 2019.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Ingo Steinwart and Andreas Christmann. *Support vector machines.* Springer Science & Business Media, 2008.

Charles J. Stone. Consistent nonparametric regression. *The Annals of Statistics*, pages 595–620, 1977.

Robert Swendsen. *An Introduction to Statistical Mechanics and Thermodynamics.* Oxford University Press, 2020.

Gábor J. Székely and Maria L. Rizzo. Energy statistics: A class of statistics based on distances. *Journal of Statistical Planning and Inference*, 143(8):1249–1272, 2013.

Kunal Talwar. Computational separations between sampling and optimization. In *Neural Information Processing Systems*, volume 32, 2019.

Shin'ichi Tamura and Masahiko Tateishi. Capabilities of a four-layered feedforward neural network: four layers versus three. *IEEE Transactions on Neural Networks*, 8(2):251–255, 1997.

Joseph F. Traub. Information-based complexity. In *Encyclopedia of Computer Science*, pages 850–854. John Wiley and Sons Ltd., GBR, 2003. ISBN 0-470-86412-5.

Alexandre B. Tsybakov. *Introduction to Nonparametric Estimation.* Springer Series in Statistics. Springer, 2009. doi: 10.1007/b13794.

Evgenii Tsymbalov, Maxim Panov, and Alexander Shapeev. Dropout-based active learning for regression. In *International Conference on Analysis of Images, Social Networks and Texts*, 2018.

Tim Van Erven and Peter Harremos. Rényi divergence and Kullback-Leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, 2014.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017.

Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv:1011.3027*, 2010.

Sethu Vijayakumar and Stefan Schaal. Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space. In *International Conference on Machine Learning*, 2000.

Dietrich von Rosen. Moments for the inverted Wishart distribution. *Scandinavian Journal of Statistics*, pages 97–109, 1988.

Abraham Wald. On the efficient design of statistical investigations. *The Annals of Mathematical Statistics*, 14(2):134–140, 1943.

Guorong Wang, Yimin Wei, Sanzheng Qiao, Peng Lin, and Yuzhuo Chen. *Generalized Inverses: Theory and Computations*, volume 53. Springer, 2018.

Haonan Wang, Wei Huang, Ziwei Wu, Hanghang Tong, Andrew J. Margenot, and Jingrui He. Deep active learning by leveraging training dynamics. In *Advances in Neural Information Processing Systems*, 2022.

Zhilei Wang, Pranjal Awasthi, Christoph Dann, Ayush Sekhari, and Claudio Gentile. Neural active learning with performance guarantees. In *Neural Information Processing Systems*, 2021.

Holger Wendland. *Scattered Data Approximation*. Cambridge University Press, 2004.

Lilian Weng. Learning with not enough data part 2: Active learning. *lilianweng.github.io/lil-log*, 2022. URL `https://lilianweng.github.io/lil-log/2022/02/20/active-learning.html`.

Tizian Wenzel, Gabriele Santin, and Bernard Haasdonk. A novel class of stabilized greedy kernel approximation algorithms: Convergence, stability and uniform point distribution. *Journal of Approximation Theory*, 262:105508, 2021.

Bernard Widrow, Aaron Greenblatt, Youngsik Kim, and Dookun Park. The No-Prop algorithm: A new learning algorithm for multilayer neural networks. *Neural Networks*, 37:182–188, 2013.

Andrew G. Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. In *Neural Information Processing Systems*, 2020.

Dawn Woodard, Scott Schmidler, and Mark Huber. Sufficient conditions for torpid mixing of parallel and simulated tempering. *Electronic Journal of Probability*, 14:780–804, 2009.

David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1–2):1–157, 2014.

Blake Woodworth, Francis Bach, and Alessandro Rudi. Non-convex optimization with certificates and fast rates through kernel sums of squares. In *Conference on Learning Theory*, 2022.

Dongrui Wu. Pool-based sequential active learning for regression. *IEEE Transactions on Neural Networks and Learning Systems*, 30(5):1348–1359, 2018.

Henry P. Wynn. The sequential generation of D-optimum experimental designs. *The Annals of Mathematical Statistics*, 41(5):1655–1664, 1970.

Greg Yang and Edward J. Hu. Tensor programs IV: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning*, pages 11727–11737. PMLR, 2021.

Hwanjo Yu and Sungchul Kim. Passive sampling for regression. In *International Conference on Data Mining*, 2010.

Amir Zandieh, Insu Han, Haim Avron, Neta Shoham, Chaewon Kim, and Jinwoo Shin. Scaling neural tangent kernels via sketching and random features. In *Neural Information Processing Systems*, 2021.

Viktor Zaverkin and Johannes Kästner. Exploration of transferable and uniformly accurate neural network interatomic potentials using optimal experimental design. *Machine Learning: Science and Technology*, 2(3):035009, 2021.

Viktor Zaverkin, David Holzmüller, Ingo Steinwart, and Johannes Kästner. Fast and Sample-Efficient Interatomic Neural Network Potentials for Molecules and Materials Based on Gaussian Moments. *Journal of Chemical Theory and Computation*, 17(10):6658–6670, 2021.

Viktor Zaverkin, David Holzmüller, Ingo Steinwart, and Johannes Kästner. Exploring chemical and conformational spaces by batch mode deep active learning. *Digital Discovery*, 1(5):605–620, 2022.

Viktor Zaverkin, David Holzmüller, Luca Bonfirraro, and Johannes Kästner. Transfer learning for chemically accurate interatomic neural network potentials. *Physical Chemistry Chemical Physics*, 25(7):5383–5396, 2023.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

Fedor Zhdanov. Diverse mini-batch active learning. *arXiv:1901.05954*, 2019.

Yilun Zhou, Adithya Renduchintala, Xian Li, Sida Wang, Yashar Mehdad, and Asish Ghoshal. Towards understanding the behaviors of optimal deep active learning algorithms. In *International Conference on Artificial Intelligence and Statistics*, 2021.

Difan Zou, Pan Xu, and Quanquan Gu. Faster convergence of stochastic gradient langevin dynamics for non-log-concave sampling. In *Uncertainty in Artificial Intelligence*, 2021.

Dominik Ślęzak, Marek Grzegorowski, Andrzej Janusz, Michał Kozielski, Sinh Hoa Nguyen, Marek Sikora, Sebastian Stawicki, and Łukasz Wróbel. A framework for learning and embedding multi-sensor forecasting models into a decision support system: A case study of methane concentration in coal mines. *Information Sciences*, 451:112–133, 2018.