INSTITUT FÜR PARALLELE UND VERTEILTE SYSTEME

ABTEILUNG SCIENTIFIC COMPUTING

STUDIENGANG SIMULATION TECHNOLOGY

Bachelorarbeit

Vorgelegt an der Universität Stuttgart

Iterative Surrogate Modeling for Black-Box Optimization Tasks

Betreuer 1

M.Sc. Peter DOMANSKI

Institut für Parallele und Verteilte Systeme

Betreuer 2

Prof. Dr. Dirk PFLÜGER

Institut für Parallele und Verteilte Systeme

Vorgelegt von

Autor	Marc HINGAR
Matrikel-Nr.	3394580
SimTech-Nr.	170
Abgabetermin	November 2023

Kurzfassung

Surrogatmodellierung ist eine beliebte Methodik, um Kosten bei der Durchführung von Experimenten mittels teuren Simulationen zu sparen. Für das Training von Surrogaten werden in der Regel adaptive Samplingansätze verwendet, um ein möglichst effektives Training mit möglicht wenig Daten durchführen zu können.

Adaptive Samplingansätze können anhand ihrer verwendeten Exploitationsstrategie in vier Kategorien unterteilt werden. Methoden werden in der Regel immer mit anderen Methoden derselben Kategorie verglichen, deshalb vergleiche ich adaptive Samplingansätze der vier Kategorien miteinander hinsichtlich Sampel- und Samplingeffizienz. Die Ergebnisse zeigen, dass jede der Kategorien ihre eigenen Stärken und Schwächen mit sich bringt. Deshalb könnte man in Zukunft auch über 'ensemble'-Lernstrategien nachdenken, bei denen adaptive Samplingmethoden verschiedener Kategorien miteinander kombiniert werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Verzeichnis der Algorithmen

1 Einleitung

Simulationen kommen heutzutage in nahezu allen Forschungs-, Fach- und Anwendungsbereichen zum Einsatz. Angewendet auf komplexe Aufgabenstellungen erfordert ihre Berechnung allerdings enorme Rechenleistung, viel Zeit und damit teils gewaltige Kosten. Ein prominentes Beispiel aus der Literatur ist eine Crashtestsimulation von Ford MotorCompany. Für ein vollbesetztes Auto betrug die Rechenzeit für nur einen einzigen Datenpunkt zwischen 36 und 160 Stunden (vgl. [KIDT09]).

Um genauere Einblicke in das Verhalten komplexer, mittels Simulationen modellierter Systeme zu erhalten, sind mehrere Simulationsdurchläufe von Nöten. Um bei der Durchführung von Experimenten Zeit und Kosten zu sparen, bietet sich die Nutzung von sogenannten Surrogat- oder Metamodellen - beide Begriffe werden in der Literatur häufig synonym verwendet - an. Unter Surrogat- bzw. Metamodellierung versteht man das Erstellen und Trainieren rechenzeitgünstiger Hilfsmodelle, deren Aufgabe darin besteht, einen komplexen, rechenaufwendigen Prozess zu approximieren. Bei der Surrogatmodellierung kommen häufig Modelltypen aus dem Bereich des herkömmlichen maschinellen Lernens (shallow machine learning) wie Support-Vektor-Maschinen oder flache neuronale Netze zum Einsatz [FTVV14]. Ziel ist es, das Verhalten einer teuren Simulation mittels weniger komplexer Modelle zu approximieren und somit die Auswertungsgeschwindigkeit deutlich zu steigern.

Es wird zwischen zwei verschiedenen Anwendungsmöglichkeiten von Surrogatmodellen unterschieden. Globale Surrogate werden als vollwertiger Ersatz für teure Simulationen verwendet. Sobald ein globales Surrogatmodell erfolgreich trainiert ist und das Verhalten der teuren Simulation auf dem gesamten Eingabebereich akkurat wiedergibt, können alle zukünftigen Experimente mit Hilfe des Surrogats durchgeführt werden. Lokale Surrogatmodelle kommen hingegen vor allem bei Optimierungsaufgaben zum Einsatz, bei denen nur eingeschränkte Bereiche des Eingaberaums von Interesse sind. Mittels lokaler Modelle, die nur innerhalb einer Teilregion des gesamten Eingaberaums trainiert werden, soll der Optimierungsprozess in Richtung des globalen Optimums geleitet werden. Im Anschluss an den Optimierungsprozess werden diese Modelle allerdings i.d.R. wieder verworfen, da sie keinen weiteren Nutzen mit sich bringen [KIDT09].

Erfolgreiches Training von Surrogaten hängt von zwei Faktoren ab: einerseits von der Wahl des Modelltyps und damit einhergehend der Kapazität des Modells und andererseits von der Wahl der verwendeten Trainingsdaten. Während ein angemessener Modelltyp durch einfaches Ausprobieren und Vergleichen von verschiedenen Modellen gefunden werden kann, ist

das Erstellen des optimalen Trainingsdatensatzes eine größere Herausforderung. Ein Trainingsdatensatz besteht aus mehreren Paaren von Eingaben, denen jeweils die gewünschte Modellantwort - also ein Zielwert - zugeordnet wird. Während das Erzeugen von möglichen Eingabedaten mit keinerlei Aufwand verbunden ist, muss jeder der angestrebten Ausgabewerte durch einen Simulationslauf erkauft werden.

In der Regel werden Simulationen als sogenannte *Black-Box-Funktionen* betrachtet. Das bedeutet, dass die einzige Möglichkeit, an Erkenntnisse über das Funktionsverhalten zu gelangen, darin besteht, das Eingabe-Ausgabe-Verhalten der Simulation zu beobachten. Es können im Voraus weder Annahmen über den Wertebereich der Ausgaben noch über Stetigkeit oder andere Eigenschaften getroffen werden.

Deshalb besteht keine Möglichkeit, im Voraus die optimale Menge an Trainingsdaten zu bestimmen. Um unnötige Auswertungen der Simulation zu vermeiden, sollte der Trainingsdatensatz so klein wie möglich gehalten werden. Um zusätzlich zu garantieren, dass das Surrogatmodell beim Training auf Informationen aus allen Bereichen des betrachteten Eingaberaums zugreifen kann, bietet es sich intuitiv an, ein raumfüllendes Verfahren zur Lokalisierung der einzelnen Punkte zu verwenden.

Traditionelle Verfahren des maschinellen Lernens verwenden einen Datensatz mit fixer Größe, der vor Beginn des Modelltrainings erstellt wird. Falls die Anzahl der verwendeten Punkte zum Training des Modells zu klein gewählt wird und die Punkte mittels raumfüllender Verfahren verteilt wurden, wird das resultierende Surrogat nicht in der Lage sein, das Verhalten der Simulation zu imitieren. Damit ist der erstellte Datensatz fast wertlos und es muss eine neue Menge an Daten berechnet werden.

Aktives Lernen ist ein Teilbereich des maschinellen Lernens. Das Ziel ist es, die Anzahl an teuren Annotationen, d.h. Berechnung von Ausgabewerten zu gegebenen Eingabedaten, zu reduzieren - zu Lasten von erhöhten Kosten bei der Bestimmung geeigneter Eingabedatenpunkte. Statt mit einem fixen, im Voraus erstellten Trainingsdatensatz mit vorgegebener, fester Größe, wird beim aktiven Lernen das Modelltraining mit einem sehr kleinen, initialen Datensatz gestartet. Während jeder Iteration hat der Lernalgorithmus nun die Möglichkeit, den erwünschten Zielwert eines nicht-annotierten Datenpunktes aus dem Eingaberaum zu erfragen. Hierbei kann er auf Erkenntnisse, die während des Trainings gewonnen wurden, zurückgreifen. Beispielsweise können Regionen, in denen das Surrogatmodell große Schwierigkeiten mit der Approximation des Verhaltens der Simulation hat, mit zusätzlichen Datenpunkten dichter gefüllt werden als andere, leichter zu approximierende Regionen. Der initiale Datensatz wird also während jeder Trainingsiteration durch ein neues Paar an Eingaben und zugehörigen Ausgabewerten erweitert. Aus diesem Grund werden für aktives Lernen synonym auch die Begriffe 'iteratives Sampling', 'adaptives Sampling' oder in der Statistik 'optimales experimentelles Design' verwendet.

Verschiedenste Autoren stellten in den vergangen Jahren viele unterschiedliche adaptive Samplingansätze für die Erstellung globaler Surrogatmodelle zur Lösung der ein- und mehr-

dimensionalen Regressionsaufgabe vor. In ihrer 2017 erstellten Übersicht nehmen Liu et al. (2017) [HYJ17] eine grobe Kategorisierung der Methoden in vier Gruppen vor:

- Ausschussbasierte Verfahren
- Kreuzvalidierungsbasierte Verfahren
- Gradientenbasierte Verfahren
- Varianzbasierte Verfahren

Aufgrund der gewaltigen Menge an verschiedenen Methoden verzichten Liu et al. allerdings vollständig auf experimentelle Vergleiche. Im Zuge dieser Arbeit versuche ich, diese Lücke, speziell für eindimensionale, globale Surrogatmodellierung, ein wenig zu füllen. Ich werde aus jeder der oben genannten Kategorien eine existierende Samplingmethode auswählen und innerhalb des 'auto_tune'-Frameworks implementieren. Anschließend werde ich diese Methoden hinsichtlich ihrer globalen Güte - gemessen anhand des mittleren quadratischen Fehlers ('Mean-Squared-Error', MSE) und R^2-Scores auf einem Testdatensatz - und ihrer Sampeleffizienz vergleichen.

Die weitere Arbeit ist in folgender Weise gegliedert: In Kapitel 2 werde ich im Zuge des SimTech-Propaedeuticums eine Literaturrecherche zum Thema 'iteratives Sampling' durchführen. Kapitel 3 beschreibt die von mir ausgewählten iterativen Samplingstrategien, welche ich im 4. Kapitel anhand einiger deterministischer Benchmarkfunktionen aus dem Bereich der Optimierung miteinander vergleichen werde. Kapitel 5 bildet mit einer Diskussion über die erlangten Ergebnisse und einer kurzen Zusammenfassung den Abschluss meiner Arbeit.

2 Grundlagen

Das folgende Kapitel erhebt nicht den Anspruch, das gesamte Themengebiet des iterativen Samplings abzudecken. Ich konzentriere mich im Folgenden auf Methoden und Themen, die für meine Arbeit von Bedeutung sind und häufig in verwandter Literatur verwendet werden. Dies schließt beispielsweise Arbeiten, die adaptives Sampling für Klassifizierungsaufgaben oder tiefes aktives Lernen ('deep active learning') betrachten, aus. Für Letzteres möchte ich auf die Übersicht, erstellt von Ren et al. (2021) [PYX+21] verweisen.

2.1 Allgemeine Übersicht maschinelles Lernen

Das Ziel von maschinellem Lernen ist es, allgemein gültige Muster aus einer begrenzten Menge an verfügbaren Daten abzuleiten. Bei der Anwendung von Techniken aus dem Bereich des maschinellen Lernens wird hauptsächlich zwischen zwei Szenarien unterschieden: überwachtes und unüberwachtes Lernen.

2.1.1 Überwachtes Lernen

Beim überwachten Lernen geht es darum, aus einer Menge an Eingaben einen zugehörigen Ausgabewert vorherzusagen. Es gibt zwei verschiedene Aufgabentypen, die mittels überwachtem Lernen gelöst werden können. Bei der *Klassifizierung* nimmt die Ausgabe des Modells diskrete Werte, die sogenannten Labels, an. Ein Beispiel: Die Klassifizierung handgeschriebener Zahlen anhand einer Menge von Pixelwerten (vgl. MNIST Datensatz). Bei der *Regression* hingegen werden kontinuierliche Werte zurückgegeben, z.B. eine Schätzung des Gewichts einer Person, deren Körpergröße und Geschlecht bekannt sind.

Während des Lernprozesses können die Vorhersagen eines Modells mit den tatsächlichen Zielwerten verglichen werden. Somit kann eine Metrik $Loss(\hat{f})$ berechnet werden, die angibt, wie genau das Modell in der Lage ist, die Trainingsdaten wiederzugeben. Ziel des überwachten Trainingsvorgangs ist es, $Loss(\hat{f})$ über die Menge aller möglichen Eingaben $x \in \mathcal{X}$ zu minimieren. Allerdings ist es in der Praxis unmöglich, diese Metrik über $\mathcal{X}$ zu bestimmen. Stattdessen wird $Loss(\hat{f})$ über den verfügbaren Trainingsdatensatz minimiert. Um die Güte des Modells auf unbekannten Datenpunkten zu überprüfen und somit die Generalisierungsfähigkeit des Modells zu bestimmen, wird dessen Leistung immer wieder anhand eines Testdatensatzes

überprüft. Zusätzlich wird ein dritter, von Trainings- und Testdatensatz verschiederer Validierungsdatensatz benötigt, um die Abstimmung der Hyperparameter, im Falle eines neuronalen Netzes beispielsweise die Architektur, des Modells zu überprüfen.

2.1.2 Unüberwachtes Lernen

Beim unüberwachten Lernen fehlen die vorgegebenen Zielwerte. Damit liegen auch keine konkreten Informationen darüber vor, wie gut die Performanz des zu trainierenden Modells ist. Eine Methode des unüberwachten Lernens ist beispielsweise die Clusteranalyse. Ziel ist es hier, Gemeinsamkeiten verschiedener Punkte eines Datenbestandes aufzudecken und ähnliche Datenpunkte zu gruppieren.

2.1.3 Trainingsdatensatz

Die Herangehensweise für beide Szenarien ist traditionell sehr ähnlich. In einem ersten Schritt müssen genügend Datenpunkte gesammelt und in einem Trainingsdatensatz zusammengefasst werden, bevor die entsprechenden Lernalgorithmen und Techniken des jeweiligen Gebiets zum Einsatz kommen. Diesen Vorgang nennt man statisches Lernen.

Wie viel sind 'genügend Datenpunkte'? Im Allgemeinen geht man davon aus, dass ein großer bis sehr großer Trainingsdatensatz die Gefahr von Überanpassung ('Overfitting') verringert. Von Überanpassung spricht man, wenn das verwendete Modell die verwendeten Trainingsdaten zu genau abbildet - sie im Grunde auswendig gelernt hat - und damit bekannten Werten einen nahezu perfekten Ausgabewert zuordnen kann, allerdings keine grundlegenden Muster innerhalb der Daten erkannt hat und deshalb keinerlei Generalisierungsfähigkeit für unbekannte Datenpunkte besitzt. Allerdings ist gerade im Fall von überwachtem Lernen das Sammeln vieler Trainingsdaten ein teurer und zeitintensiver Vorgang.

Ein Beispiel hierzu habe ich bereits in der Einleitung dargestellt: Eine Ford MotorCompany Crashtestsimulation, bei der die Rechenzeit der Simulation für einen einzigen Datenpunkt zwischen anderthalb und siebeneinhalb Tagen lag (vgl. [KIDT09]).

Deshalb ist es erstrebenswert, zum Training einen möglichst kleinen Datensatz zu verwenden. Die Literatur geht davon aus, dass alle Datenpunkte, die zum Modelltraining verwendet werden, einer Gleichverteilung über den gesamten Eingaberaum entsprechen und damit voneinander unabhängig und gleichverteilt sind ('independent and identically distributed', vgl. [GBC16]). Allerdings gibt es viele Situationen, in denen die Möglichkeit besteht, aktiv auf den Samplingprozess Einfluss zu nehmen. Falls beispielsweise eine Simulation die gewünschten Zielwerte zurückgibt, ist es möglich, der Funktion ganz gezielt Eingabewerte zur Annotation zu übergeben und somit bestimmte Regionen dichter mit Datenpunkte aufzufüllen als andere.

2.2 Sequentielles Sampling

Neben gezielter Auswahl der Eingabewerte des Trainingsdatensatzes ist es auch nicht zwangsweise erforderlich, bereits alle verwendeten Trainingsdaten vor Beginn des Trainingsvorgangs zu sammeln. Stattdessen besteht die Möglichkeit, während des Trainingsvorgangs, ausgehend von bisher getroffenen Erkenntnissen, neue Datenpunkte auszusuchen. Dies führt im Optimalfall zu möglichst guten Trainingsergebnissen unter Verwendung möglichst weniger Datenpunkte. Das Verfahren, den Samplingprozess während des Trainingsvorgangs fortzusetzen und ihn bewusst zu steuern, nennt man *sequentielles Sampling*.

Im Allgemeinen lässt sich der sequentielle Trainingsvorgang eines Modells $\hat{f}$ nach Crombecq et al. folgendermaßen beschreiben:

Algorithmus 2.1 Iteratives Sampling (vgl. [KDDT11])

Require:
Orakel $f : \mathcal{X} \to \mathcal{Y}$
Modell $\hat{f} : \mathcal{X} \to \mathcal{Y}$
$D = (X, y = f(X)) \subset \mathcal{X} \times \mathcal{Y}$

1: **while** ¬ Abbruchkriterium **do**
2: wähle $x_{neu} \in \bar{X}$
3: $y_{neu} \leftarrow f(x_{neu})$
4: $D \leftarrow (X \cup \{x_{neu}\}, y \cup \{y_{neu}\})$
5: trainiere Modell $\hat{f}(D)$
6: **end while**

Die Instanz, die für die Annotation neuer Datenpunkte zuständig ist, wird in der Literatur allgemein als Orakel bezeichnet und stellt die zu lernende Zielfunktion dar. Dies kann beispielsweise ein menschlicher Experte oder eine Computersimulation sein.

Der größte Vorteil iterativer Samplingmethoden gegenüber statischer Lernmethoden besteht darin, dass die benötigte Datensatzgröße automatisch während des Lernprozesses gefunden wird. Sobald ein vorgegebenes Abbruchkriterium, z.B. eine vorgegebene Modellgenauigkeit erfüllt ist, wird der Trainingsvorgang des Modells abgebrochen. Somit lässt sich die Gefahr für sogenanntes 'Oversampling', bei dem mehr teure Datenpunkte durch das Orakel ausgewertet werden müssen, als unbedingt notwendig, reduzieren. Allerdings ist hierfür ein passendes Abbruchkriterium von Nöten, sodass der Samplingprozess weder zu früh noch zu spät beendet wird. Auf explizite Abbruchkriterien werde ich im Folgenden noch etwas genauer eingehen.

2.3 Raumfüllendes und adaptives sequentielles Sampling

Zuerst werde ich mich mit der Fragestellung beschäftigen, wie neue Datenpunkte $x_{neu} \in \bar{X}$ ausgewählt werden können. Sequentielles Sampling lässt sich in zwei verschiedene Klassen unterteilen: raumfüllendes sequentielles Sampling und adaptives sequentielles Sampling, auch aktives Lernen genannt.

Ziel des raumfüllenden sequentiellen Samplingansatzes ist es, iterativ den gesamten Eingaberaum möglichst gleichmäßig mit Datenpunkten auszufüllen. Crombecq et al. (2009) [KIDT09] stellen eine iterative Methode vor, die während jeder Iteration die Voronoi-Zerlegung des Eingaberaums anhand der bisher bekannten Datenpunkten approximiert. Bei der Voronoi-Zerlegung wird der Raum in mehrere Zellen unterteilt, deren Zentrum jeweils ein Punkt aus der vorhandenen Trainingsdatenmenge bildet. Der Zelle $V_i \subset \mathcal{X}$ mit Zentrum $x_i \in X$ werden alle Punkte $x \in \bar{X}$ zugeordnet, die näher an x_i als allen anderen Punkte $x_j \in X$, $x_i \neq x_j$ liegen. Somit gilt: $\bigcup_{1 \leq i \leq m} V_i = \mathcal{X}$. Mittels Monte-Carlo-Simulation kann mit geringem Rechenaufwand das Volumen aller Voronoi-Zellen V_i approximiert werden. Je größer das Volumen einer Voronoi-Zelle V_i, desto weniger bekannte Datenpunkte liegen in der näheren Umgebung des Punktes x_i. Deshalb wählen Crombecq et al. die Stelle des Eingaberaums innerhalb der Voronoi-Zelle mit größtem Volumen als neuen Datenpunkt, die den größten minimalen Abstand zu allen Punkten $x \in X$ hat.

Adaptives sequentielles Sampling bzw. aktives Lernen basiert auf der Annahme, dass verschiedene Datenpunkte einer Datenmenge einen unterschiedlich großen Nutzen hinsichtlich der Aktualisierung eines zu trainierenden Modells haben. Im Gegensatz zu raumfüllenden Samplingansätzen verwenden adaptive sequentielle Methoden auch das Modell $\hat{f}$ und bekannte Ausgabewerte des Orakels f, um neue Datenpunkte $x_{neu} \in \bar{X}$ auszuwählen. Beispielsweise können komplizierte Regionen, in denen die Modellvorhersagen stark von den gewünschten Ausgabewerten des Orakels abweichen, dichter mit Datenpunkten gefüllt werden als Gegenden, die bereits hinreichend gut approximiert werden können.

Beim aktiven Lernen müssen also sogenannte *interessante Regionen* innerhalb des Eingaberaums identifiziert werden, für die eine höhere Datenpunktdichte zur adäquaten Approximation benötigt wird. Unter interessante Regionen fallen beispielsweise lokale Optima, Regionen mit sehr großen Gradienten oder Unstetigkeiten (vgl. Crombecq et al. (2009) [KIDT09]). Der Schritt, gezielt neue Datenpunkte innerhalb von bekannten interessanten Regionen beim Orakel anzufragen, wird *Exploitation* genannt.

Zahlreiche Autoren berichten allerdings davon, dass die Qualität von Modellen, deren Trainingsdatensatz iterativ ausschließlich mittels Exploitation erstellt wurde, sehr stark von der Qualität des initialen Datensatzes abhängig ist ([GSAA06], [RWA02], [VKO+13]). Rein exploitative Verfahren sind meist nicht in der Lage, neue, unbekannte interessante Regionen ausfindig zu machen. So bleiben einige Gebiete des Eingaberaums möglicherweise vollständig

unerkundet, während in anderen Regionen sehr viele Datenpunkte auf engstem Raum beisammenliegen. Dieses Phänomen wird Clusterbildung ('Clustering') genannt und führt zu einer schlechten globalen Vorhersagegenauigkeit des Modells.

Moderne und robuste adaptive Sampling Methoden müssen deshalb bei jeder Iteration zwischen lokaler Exploitation und globaler *Exploration* abwägen. Bei der Exploration wird der Eingaberaum nach neuen, unbekannten interessanten Regionen abgesucht. Dies geschieht durch gleichmäßiges Abdecken des Eingaberaums mittels raumfüllener iterativer Samplingmethoden.

Ein neuer Datenpunkt wird also durch Lösung der folgenden Optimierungsaufgabe bestimmt (vgl. [HYJ17]:

$$x_{neu} = \arg\max_{x \in \bar{X}} Score(x) \tag{2.1}$$

Die Score-Funktion weist jedem Punkt $x \in \bar{X}$ einen Wert für seinen Informationsgehalt zu. Dieser setzt sich zu seinem Beitrag zur Exploitation $exploit(x)$ und zur Exploration $explore(x)$ zusammen.

$$Score(x) = w_{exploit} * exploit(x) + w_{explore} * explore(x) \tag{2.2}$$

Das Abwägen zwischen Exploitation und Exploration geschieht unter Verwendung zweier Gewichtungsfaktoren $w_{explore}$ und $w_{exploit}$. Unter der Annahme, dass $w_{explore} + w_{exploit} = 1.0$, existieren verschiedene Ansätze, beiden Faktoren während jeder Iteration einen Wert zuzuweisen.

Crombecq et al. (2011a) [KDDT11] halten während des gesamten Samplingprozesses beide Faktoren konstant auf dem selben Wert: $w_{explore} = w_{exploit}$.

Im Gegensatz hierzu haben sich auch einige dynamische Verfahren etabliert, um die Größe der Gewichtungsfaktoren in Abhängigkeit von der Anzahl der durchlaufenen Iterationen zu bestimmen. Singh et al. (2013) [PDT13] erweitern den Explorations-Exploitations-Tradeoff durch einen *ϵ-greedy* Ansatz. Anstatt die Gewichtungsfaktoren konstant zu halten, wird in jeder Iteration entweder ausschließlich Exploration mit $w_{explore} = 1.0$ oder ausschließlich Exploitation mit $w_{exploit} = 1.0$ durchgeführt. Während jeder Iteration wird eine gleichverteilte Zufallszahl $\alpha \in [0, 1]$ gezogen. Für alle $\alpha < \epsilon$ wird ein neuer Datenpunkt ausschließlich explorativ, auf Basis der Bewertung $explore(x)$ gewählt. Für alle $\alpha \geq \epsilon$ ist allein Exploitation $exploit(x)$ für die Bewertung der potentiellen neuen Datenpunkte zuständig. Schwachpunkt dieser Variante ist allerdings, dass mit dem Parameter ϵ ein weiterer Hyperparameter eingeführt wird, der stark von den unbekannten Eigenschaften der Zielfunktion f abhängig ist und sich im Voraus nicht zuverlässig bestimmen lässt.

Ein weiterer Ansatz, um die Gewichtungsfaktoren $w_{explore}$ und $w_{exploit}$ zu bestimmen besteht darin, zu Beginn des aktiven Lernprozesses mit starkem Fokus auf Exploration zu starten, den Wert von $w_{explore}$ schrittweise während jeder Iteration ein klein wenig zu senken und im Gegenzug $w_{exploit}$ um den selben Betrag zu erhöhen, sodass am Ende des Samplingprozesses der Hauptfokus auf lokaler Exploitation liegt (vgl. [PDT13]).

Liu et al. (2016a) iterieren während des Samplingvorgangs wiederholt über einen festgelegten Plan, der mit starkem Fokus auf Exploration beginnt und mit einem großen Faktor $w_{exploit}$ endet.

Im Folgenden werde ich kurz einige Möglichkeiten aufzeigen, mittels derer lokale Exploitation bzw. globale Exploration realisiert werden können.

2.4 Abfrage-Strategien

2.4.1 Exploitation

Bei vielen der veröffentlichten Methoden zur lokalen Exploitation gehen die Autoren davon aus, dass das verwendete Modell $\hat{f}$ bereits im Voraus bekannt ist (z.B. [RH06]). Dieses Wissen wird verwendet, um das Modellverhalten auszunutzen und den Samplingprozess somit in die perfekte Richtung für den spezifischen verwendeten Modelltyp zu lenken. Dieser Ansatz wird auch *optimales sequentielles Design* genannt. Während optimales sequentielles Design das Erstellen von sehr effizienten Methoden erlaubt, können die auf diese Weise entwickelten Methoden nicht für andere Modelltypen eingesetzt werden, sollte das gewählte Modell nicht in der Lage sein, das Verhalten der Funktion f korrekt zu imitieren. Der resultierende Trainingsdatensatz ist genau auf den gewählten Modelltyp zugeschnitten. Die a priori Wahl des optimalen Modelltyps ist allerdings für Black-Box-Funktionen eine nicht-triviale Aufgabe.

Um Flexibilität bei der Modellwahl zu erhalten, existiert deshalb auch der Ansatz des *generischen sequentiellen Designs*, bei dem keine Annahmen über den verwendeten Modelltyp getroffen werden. Hierbei kann während des Samplingprozesses ausschließlich auf die bekannten Trainingsdaten bestehend aus Eingaben und Ausgaben des Orakels $(X, f(X))$ und erstellte Hilfsmodelle (siehe z.B. 2.4.1 oder 2.4.1) zurückgegriffen werden. Die Wahl des optimalen Modelltyps kann erst nach Zusammenstellung des Trainingsdatensatzes getroffen werden.

Vorhersagevarianz

Eine generische Exploitationsmethode ist die sogenannte Abfrage-mittels-Ausschuss-Methode ('*Query by Committee*', QBC). Der Lernalgorithmus unterhält hierbei einen sogenannten Ausschuss $\mathcal{C} = \{\hat{f}_i\}_{1 \leq i \leq t}$ an t Hilfsmodellen. Während jeder Iteration werden alle Modelle dieses Ausschusses mittels der verfügbaren Trainingsdatenmenge trainiert. Wenn der aktive Lernalgorithmus die Informativität eines neuen, unbekannen und nicht annotierten Punktes $exploit(x)$, $x \in \bar{X}$ bestimmen möchte, geschieht dies über die Unstimmigkeit in der Vorhersage der Mitglieder des Ausschusses. Jedes der Hilfsmodelle des Ausschusses versucht

hierbei, den tatsächlichen Ausgabewert vorherzusagen. Im Anschluss kann das Maß an Unsicherheit in den Vorhersagen wie bei Krogh und Vedelsby (1995) [AJ95] beispielsweise durch die Berechnung der Varianz der Modellvorhersagen bestimmt werden:

$$\hat{\sigma}^2_{QBC}(x) = \frac{1}{t}\sum_{i}^{t}(\hat{f}_i(x) - \bar{\hat{f}}(x))^2 \tag{2.3}$$

für $x \in \bar{X}$ und $\bar{\hat{f}}(x) = \frac{1}{t}\sum_i^t \hat{f}_i(x)$.

Bei großer Unstimmigkeit liegt der neue Punkt in einer Region des Eingaberaums, die mittels der verfügbaren Trainingsdaten nicht mit hinreichender Sicherheit approximiert werden kann. Somit bietet es sich an, diesen Punkt in den Trainingsdatensatz aufzunehmen.

Damit QBC funktionieren kann, wird ein gewisses Maß an Unstimmigkeit zwischen den Ausschussmodellen benötigt. Dies kann beispielsweise durch die Verwendung verschiedener Modelltypen innerhalb des Ausschusses sichergestellt werden (z.B. [FFN+12], [HYJ17]). Eine andere Möglichkeit besteht in der Nutzung des selben Modelltyps aller Hilfsmodelle des Ausschusses, wobei jedes einzelne Modell auf einer anderen Teilmenge des verfügbaren Trainingsdatensatzes trainiert wird. Hierfür wird der Trainingsdatensatz in t gleich große Teilmengen zerlegt. Jedes Modell wird auf der Vereinigung von $t-1$ dieser Teilmengen trainiert. Burbidge et al. (2007) [RJR07] haben aufgezeigt, dass hiermit gute Ergebnisse im Kontext von globaler Surrogatmodellierung für Regressionsaufgaben erzielt werden können, sofern der Bias des verwendeten Modelltyps klein ist. Der Bias beschreibt den Unterschied zwischen Modellvorhersage und tatsächlichem Wert (vgl. [AJ95]).

Kreuzvalidierung

Ein weiterer modellunabhängiger Ansatz zur lokalen Exploitation bieten kreuzvalidierungsbasierte Abfragestrategien. Das Kreuzvalidierungsverfahren kann verwendet werden, um mit Hilfe der bekannten Trainingsdaten den tatsächlichen Schätzfehler eines Modells zu approximieren. Dazu wird der Trainingsdatensatz in k verschiedene, gleich große Teilmengen T_i, $1 \leq i \leq k$ zerlegt. Nun können k neue Modelle trainiert werden, wobei für Modell $\hat{f}^{-i}$ Teilmenge T_i als Testdatenmenge zurückgehalten wird. Das Modelltraining von $\hat{f}^{-i}$ findet also auf der Datenmenge $D \setminus T_i$ statt. Der Kreuzvalidierungsfehler wird folgendermaßen definiert (z.B. [PLQ+15]):

$$\hat{e}(T_i) = |f(T_i) - \hat{f}^{-i}(T_i)|, \qquad 1 \leq i \leq m \tag{2.4}$$

Spezielle Beachtung im Bereich des adaptiven Samplings hat die sogenannte 'Leave-One-Out' (LOO) Kreuzvalidierungs-Methode gefunden. Hierbei werden für einen Trainingsdatensatz, bestehend aus m Eingabe-/Ausgabedatenpaaren, genau m Hilfsmodelle trainiert. Für jedes dieser Modelle $\hat{f}^{-i}$ besteht der Testdatensatz aus einem einzigen Paar an bekannten Daten $(x_i, f(x_i))$. Der gemittelte LOO-Kreuzvalidierungs-Fehler ist in erster Linie nicht als Metrik für die Modellgenauigkeit des Surrogatmodells zu betrachten. Viel eher beschreibt er die Sensibilität

des Surrogatmodells in Bezug auf den Verlust einzelner Datenpunkte. Falls $\hat{e}(x_i)$ klein ist, bedeutet das, dass sich in der Umgebung von x_i bereits genügend Datenpunkte mit bekanntem Ausgabewert befinden, um das Verhalten des Orakels trotz Verlust von x_i vorherzusagen. Falls allerdings $\hat{e}(x_i)$ groß ist, dann befinden sich in der Trainingsdatenmenge nicht genügend Punkte, um die Ausgabewerte des Orakels in der Umgebung von x_i verlässlich zu modellieren. Der LOO-Kreuzvalidierungs-Fehler kann also verwendet werden um interessante Regionen des Eingaberaums ausfindig zu machen.

Es ist wichtig hervorzuheben, dass dieses Verfahren nicht explizit für Punkte $x \in \bar{X}$ angewendet werden kann, da für diese Punkte kein angestrebter Zielwert zur Verfügung steht. Allerdings kann der Fehler in einem Punkt $x \in \bar{X}$ ohne bekannten Zielwert beispielsweise folgendermaßen berechnet werden (siehe [RWA02]):

$$\hat{e}(x) = \sqrt{\frac{1}{m}\sum_{i}^{m}(\hat{f}^{-i}(x) - \hat{f}(x))^2} \tag{2.5}$$

wobei $\hat{f}$ ein Surrogatmodell, trainiert auf dem kompletten Trainingsdatensatz X, repräsentiert.

Ein anderer Ansatz zur Bestimmung des Fehlers in $x \in \bar{X}$ liefern Li et al. (2010) [GVS10] oder auch Aute et al. (2013) [VKO+13]. Sie trainieren ein sogenanntes Fehlermodell $\hat{f}_e(x)$ mittels der Leave-One-Out Kreuzvalidierungsfehler $\hat{e}(x_i)$ an den bekannten Punkten und benutzen $\hat{f}_e$, um den Fehler in unbekannten Punkten vorherzusagen.

Je größer der geschätzte Fehler in unbekannten Punkten, desto höher ist deren Informationsgehalt.

Varianz

Dieses optimale sequentielle Design setzt die Verwendung von Gaußprozess bzw. Kriging-Modellen voraus. Beim Kriging wird der Ausgabewert eines unbekannten Datenpunktes $x \in \bar{X}$ anhand bekannter Ausgabewerte $f(X)$ und Abstand von den bekannten Datenpunkten geschätzt:

$$y(x) \sim GP(\hat{f}(x), \hat{\sigma}^2(x)) \tag{2.6}$$

wobei $\hat{f}(x)$ die Vorhersage und $\sigma^2(x)$ die Vorhersage-Varianz beschreibt (vgl. [HYJ17]).

Viele Autoren nutzen die Vorhersage-Varianz, die als Vorhersagefehler (MSE) betrachtet werden kann, um den Samplingprozess zu beeinflussen. Jin et al. (2002) [RWA02] wählen den Eingabepunkt aus, der die Varianz bzw. den mittleren quadratischen Fehler des Modells maximiert (MMSE-Methode):

$$x_{new} = \arg\max_{x \in \bar{X}} \hat{\sigma}^2 \tag{2.7}$$

Lin et al. (2004) [YFJ+04] haben gezeigt, dass die Funktion $R(x_i, x_j)$, die die Korrelation zwischen zwei beliebigen Punkten beschreibt, bei Kriging-Modellen ausschließlich vom Abstand beider Punkte abhängt. Um beim adaptiven Sampling zusätzlich Informationen über den Vorhersagefehler zu berücksichtigen, berechnen sie die Vorhersage-Varianz durch

$$\hat{\sigma}^2_{adj}(x_i, x_j) = \sigma^2 R_{adj}(x_i, x_j) = \sigma^2 \prod_{k=1}^{d} exp(-\eta_i \eta_j \theta_k |d_k|^2) \tag{2.8}$$

wobei θ_k und d_k Korrelation und euklidischen Abstand von x_i, x_j in k-ter Koordinatenrichtung beschreiben. Informationen über Vorhersagefehler sind in den Faktoren η_i und η_j enthalten. Liu et al. (2016) [HSY+16] verwenden beispielsweise den LOO-Kreuzvalidierungsfehler zur Bestimmung dieser Faktoren.

Einen weiteren Ansatz, den iterativen Samplingprozess zu leiten, liefert die sogenannte Maximum-Entropie-Methode. Hier wird derjenige Eingabewert, der die Determinante der Korrelationsmatrix im Bayesschen Framework maximiert, ausgewählt (Shewry and Wynn (1987) oder [HSX15]).

Gradienten

Im Allgemeinen ändern sich in Regionen mit großen Gradienten die Ausgabewerte einer Funktion durch kleine Änderungen der Eingabewerte drastisch. Mit wenigen verfügbaren Datenpunkten ist es hier sehr schwierig, ein Modell genau an das Ausgabeverhalten eines Orakels anzupassen. D.h. mittels Informationen über das Verhalten des Ausgabegradienten der Zielfunktion lassen sich Bereiche innerhalb des Eingaberaums identifizieren, in denen, verglichen mit Regionen mit kleinen Gradienten, eine größere Datenpunktdichte benötigt wird, um vernünftige Vorhersagen zu treffen.

Rumpfkeil et al. (2011) [MWM11] räsentieren in der Annahme, das Orakel stelle zusätzlich zu den Ausgaben $f(x)$ auch Informationen über die vorliegenden Gradienten $\nabla f(x)$ bereit, eine Methode, um ein gradienten-erweitertes Kriging-Modell zu trainieren.

Unter der allgemein verwendeten Annahme, dass das Orakel eine Black-Box-Funktion darstellt, können allerdings lediglich die Antworten $f(x)$ auf einen Eingabewert $x \in \mathcal{X}$ beobachtet werden. Somit liefert das Orakel keine Informationen über das Verhalten der Gradienten ∇f.

Eine Möglichkeit, trotzdem an die gewünschten Gradienteninformationen zu gelangen, besteht darin, die Gradienten des verwendeten Modells $\hat{f}$ zu nutzen. Da das Modell eine Approximation der Funktion f darstellt, sollten intuitiv auch die Gradienten $\nabla \hat{f}$ eine Approximation an die Gradienten des Orakels darstellen. Glücklicher Weise können die Gradienten eines Radialen-Basis-Funktionen-Modells (RBF) mit stetigen Kernelfunktionen an jedem Punkt kostengünstig berechnet werden (vgl. [HSY+16]). Unter Verwendung dieses Modelltyps ist es sogar möglich, zusätzlich die Krümmung der Orakelfunktion mittels der Hessematrix des RBF-Modells anzunähern. Mackman et al. (2013) [TCMK13] oder Wei et al. (2012) [XYL12] nutzen die Krümmung eines RBF-Surrogats, um den Vorhersagefehler des Modells möglichst schnell zu reduzieren.

Allerdings bringt die Verwendung der Gradienten bzw. der Krümmung des Modells einige Nachteile mit sich: Zum einen lassen sich mittels dieser Strategie keine generischen sequentiellen Samplingstrategien erstellen, da nicht für alle Modellarten ein Gradient definiert werden kann.

Ein Beispiel für einen Modelltypen ohne Gradienten wäre das K-nächste-Nachbarn ('K-Nearest-Neighbors', KNN) Modell. Während des Trainingsvorgangs speichert das KNN-Modell alle übergebenen Paare an Eingabe- und zugehörigen Ausgabedaten. Soll der Ausgabewert für einen unbekannten Punkt $x \in \mathcal{X}$ bestimmt werden, so sucht das Modell die $k \in \mathbf{N}$ ähnlichsten Punkte (basierend auf einer Distanzmetrik), vergleicht die Ausgabewerte dieser bekannten Punkte und liefert als Vorhersage beispielsweise den Mittelwert der Nachbarn. D.h. der Vorhersagewert aller Punkte, die sich die selben Nachbarn teilen, ist identisch. Somit ist hier der Gradient 0. Wird für die Berechnung des Ausgabewerts einer der Nachbarn durch einen anderen Punkt ersetzt, so springt der Vorhersagewert des KNN-Modells auf den neuen Mittelwert der bekannten Ausgaben der Nachbarn. Für Unstetigkeiten im Ausgabeverhalten einer Funktion sind keine Gradienten definiert.

Zum Anderen haben Deschrijver et al. (2011) [DKHT11] festgestellt, dass Modelle während des Trainingsvorgangs häufig nicht in der Lage sind, das Verhalten des Orakels korrekt zu immitieren. D.h. nicht nur die Vorhersagewerte des Modells sind fehlerbehaftet, sondern auch die vorhergesagten Gradienten. Dies kann im schlechtesten Fall dazu führen, dass der Samplingprozess fehlgeleitet wird und somit unnötig Ressourcen verschwendet werden.

Anstatt die Gradienten des Surrogatmodells zu verwenden, nutzen Crombecq et al. (2011) [KDDT11] bei ihrer vorgestellten generischen LOLA-Voronoi-Methode deshalb lineare Hilfsmodelle, die an lokale Nachbarschaften $N \subset X$ angepasst werden. Mittels der Abweichung der Zielwerte des Orakels in den bekannten Punkten der Nachbarschaft dieser lokalen linearen Approximationen kann die Nicht-Linearität einer Region bestimmt und somit die benötigte Datenpunktdichte angepasst werden.

2.4.2 Exploration

In den meisten Fällen wird Exploration mittels Abstandsbeschränkungen durchgeführt. So werden alle potentiellen neuen Datenpunkte $x_{neu} \in \bar{X}$, deren Abstand zu bereits bekannten Datenpunkten $x \in X$ kleiner als ein Mindestabstand d ist, sofort verworfen. In der Praxis ist es häufig sehr schwierig, einen passenden Wert für den Mindestabstand d zu bestimmen. Falls d zu klein gewählt wird, kann die lokale Anhäufung vieler Punkte nicht verhindert werden. Falls d allerdings zu groß gewählt wird, werden alle neuen Datenpunkte möglichst gleichmäßig im Eingaberaum verteilt und der aktive Lernalgorithmus verliert die Fähigkeit, neue Datenpunkte aus interessanten Regionen auszuwählen. Li et al. (2010) [GVS10] schlagen deshalb vor, den gemittelten minimalen Abstand aller bekannten Datenpunkte $x_i, x_j \in X$ zu verwenden. D.h.

$$d = \frac{1}{m} \sum_{i}^{m} min \, \|x_i - x_j\|_2 \qquad (x_i \neq x_j) \tag{2.9}$$

Aute et al. (2013) [VKO+13] wählen hingegen den halbierten Mindestabstand als größten minimalen Abstand aller bekannten Datenpunkte. Unter Verwendung der selben Notation wie oben bedeutet das:

$$d = \frac{1}{2} \max(\min \|x_i - x_j\|_2) \qquad (x_i \neq x_j) \tag{2.10}$$

2.5 Samplingansätze

Es gibt drei verschiedene Strategien, um an Kandidaten für neue Datenpunkte zu gelangen. Einerseits existiert die sogenannte '*membership query synthesis*'-Methode. Hierbei hat der Lernalgorithmus die Möglichkeit, Zielwerte beliebiger Datenpunkte aus dem Eingaberaum $x \in \bar{X}$ beim Orakel zu erfragen. Vor allem hat der Lernalgorithmus auch die Möglichkeit, selbstständig Daten zu generieren.

Dem gegenüber stehen zwei Methoden des selektiven Samplings. Beim strombasierten selektiven Sampling ('*stream-based selective sampling*') werden dem Lernalgorithmus nacheinander Datenpunkte aus dem Eingaberaum $x \in \bar{X}$ präsentiert. Der Lerner muss sich bei jedem dieser Sampel spontan entscheiden, ob er einen Zielwert vom Orakel abfragen möchte oder ob der Datenpunkt verworfen werden soll. Dieser Ansatz bietet sich vor allem auf rechenleistungsschwachen Systemen und solchen mit geringem Speicherplatz an. Beim poolbasierten selektiven Sampling ('*pool-based selective sampling*') wird dem Lernalgorithmus eine vorgegebene Menge an nicht annotierten Datenpunkten des Eingaberaums $M \subseteq \bar{X}$ vorgelegt. Der Algorithmus erstellt nun eine Rangfolge aller Punkte $x \in M$ und legt somit fest, welche neuen Punkte in den Trainingsdatensatz aufgenommen werden. Im Gegensatz zum strombasierten selektiven Samplingvorgang wird hierfür häufig eine große Rechenleistung und viel Speicher während jeder Iteration benötigt.

Neben diesen drei verschiedenen Samplingansätzen lässt sich eine weitere Kategorisierung von aktiven Lernansätzen vornehmen: sequentielles und Batch-Modus aktives Lernen. *Sequentielles aktives Lernen* fragt während jeder Iteration ausschließlich den Zielwert von einem einzigen Datenpunkt an. Beim *Batch-Modus aktiven Lernen* werden dem Trainingsdatensatz mehrere neue Datenpunkte pro Iteration hinzugefügt. Die meisten in der Literatur vorgestellten Ansätze implementieren den sequentiellen Ansatz des aktiven Lernens. Grund hierfür ist Gleichung (2.1). Wird zum Beispiel beim Pool-basierten Samplingansatz ein Ranking aller potentieller neuer Sampel erstellt, so werden Punkte, die sehr dicht beieinander liegen, häufig einen ähnlichen Score erhalten und deshalb auch im Ranking dicht beieinander liegen. Verfolgt man nun zur Erstellung eines Batches den naiven Ansatz, die n Datenpunkte mit höchstem Score auszuwählen, finden sich im Batch sehr häufig Eingabepunkte mit großer Ähnlichkeit zueinander wieder, was dazu führt, dass sich die jeweils gelieferten neuen Informationen häufig überlappen ('information ovelap'). Damit werden oftmals weniger teure, aber leider auch weniger nützliche Anfragen an das Orakel gesendet. Es muss zusätzlich eine weitere Metrik eingeführt werden, die Diversität unter den neuen Datenpunkten eines Batchs garantiert.

Crombecq et al. (2011) [KDDT11] wählen beispielsweise jeden Punkt eines Batches aus einer anderen Voronoizelle des Eingaberaums.

2.6 Abbruchkriterien

In der Literatur werden verschiedene Abbruchkriterien für den iterativen Samplingprozess vorgeschlagen. Universell wird eine obere Schranke für die maximale Datensatzgröße bzw. ein maximales Budget an Rechenleistung vorgegeben. Bei Überschreiten dieser Vorgaben kann der Samplingprozess abgebrochen werden.

Der große Vorteil von aktivem gegenüber passivem Lernen besteht darin, dass der Samplingprozess sofort beendet werden kann, sobald das Modell $\hat{f}$ gelernt hat, das Orakel f hinreichend genau zu approximieren. Hierfür werden von verschiedenen Autoren verschiedene Metriken verwendet. Beispielsweise verwenden Crombecq et al. (2009) [KIDT09] den mittleren euklidischen Fehler

$$MEE = \frac{1}{n} \sum_{i}^{n} \sqrt{(f(x_i) - \hat{f}(x_i))^2} \tag{2.11}$$

auf einer dichten Testdatenmenge. Westerman und Evins (2019) [PR19] nutzen das Bestimmtheitsmaß ('Coefficient of Determination') R^2, das auf einem Testdatensatz bestehend aus 100 zufällig gezogenen Punkten ausgewertet wird. Für n Modellvorhersagen $\hat{y}$ und zugehörige Zielwerte y ist der R^2-Score folgendermaßen definiert:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_i^n (y_i - \hat{y}_i)^2}{\sum_i^n (y_i - \bar{y}_i)^2} \tag{2.12}$$

wobei $\bar{y} = \frac{1}{n} \sum_i^n y_i$.

Der R^2-Score beschreibt den Anteil der Varianz von y, der durch die unabhängigen Variablen des Modells erklärt wird und bietet somit eine Metrik, mit der die Genauigkeit eines Modells bewertet werden kann. Für den Wertebereich von R^2 gilt: $R^2 \in (-\infty, 1.0]$. Sobald das Modell $\hat{f}$ einen Wert von mehr als 0.95 erreicht, endet der aktive Trainingsvorgang.

Allerdings scheint die Evaluation des Modells auf einem separaten Testdatensatz ein Ansatz zu sein, der sich ausschließlich für theoretische Benchmarktests eignet. Für praktische Anwendungen mit der Zielsetzung, so wenig teure Anfragen wie möglich an das Orakel stellen zu müssen, kann die Erstellung eines Testdatensatzes keine Lösung sein.

Wu [DCJ18] nutzt zusätzlich zur Beschränkung der Datensatzgröße einen angestrebten Wert bei der Auswertung von $\hat{f}$ mittels Kreuzvalidierungsverfahren auf dem Trainingsdatensatz, um den Samplings- und Lernprozess zu beenden.

Ein sehr ähnlicher Ansatz funktioniert über den generalisierten mittleren Kreuzvalidationsfehler ('*generalized mean squared cross-validation error*' GMSE) Mittels Leave-One-Out Kreuzvalidierungsfehler lässt sich dieser folgendermaßen bestimmen:

$$GMSE = \sqrt{\frac{1}{m}\sum_{i}^{m}\hat{e}(x_i)^2} \tag{2.13}$$

Viana et al. (2009) [FTVV14] und Liu et al. (2016) [HSY+16] zeigen auf, dass GMSE die Güte des verwendeten Metamodells $\hat{f}$ überbewertet, für genügend verfügbare Trainingsdaten allerdings eine gute Schätzung für die tatsächliche Genauigkeit der Modellvorhersage liefert.

Als weiteres beliebtes Abbruchkriterium wird die sukzessive relative Verbesserung (SRI) einer Fehlermetrik, beispielsweise des Kreuzvalidierungsfehlers oder des relativen absoluten Fehlers, betrachtet. Sofern sich über mehrere Iterationen kein Fortschritt zeigt, kann der Trainingsvorgang beendet werden.

2.7 Initialer Datensatz

Einige Autoren haben beobachtet, dass sich die Größe und Wahl des initialen Datensatzes zu Beginn des aktiven Lernvorgangs teils stark auf die Performanz des resultierenden Modells auswirken. Kim et al. (2009) [BYD09] zeigen auf, dass beispielsweise Methoden, die das Kreuzvalidierungsverfahren (vgl. [RWA02]) zur lokalen Exploitation benutzen, durch einen unpassenden und zu kleinen initialen Trainingsdatensatz möglicherweise sogar in falsche Richtungen gelenkt werden und dadurch neue Datenpunkte an eigentlich unwichtigen Stellen anfordern.

Die Frage nach einer angemessenen Größe für den initialen Trainingsdatensatz in Abhängigkeit von der Dimensionalität des Eingaberaums und/oder den verfügbaren Rechenleistungs-Ressourcen ist in der Literatur nicht abschließend geklärt. Während Loeppky et al. (2009) [JJW09] einen initialen Datensatz der Größe $N = 10d$, wobei d die Dimensionalität der Eingabedaten beschreibt, verwenden, verlassen sich neuere Arbeiten wie beispielsweise Liu et al. (2016) [HSY+16] auf eine Größe von $N = 5d$. Einen anderen Ansatz verfolgen Aute et al. (2013) [VKO+13]. Sie machen die Größe des initialen Datensatzes von der gewählten oberen Schranke der verwendeten Datensatzgröße abhängig: $N = 0.5 * N_{max}$. Allerdings sind diese Formeln ausschließlich das Resultat empirischer Datenerhebungen.

Zusätzlich existiert auch keine einheitliche Methodik, welche die Verteilung der initialen Punkte beschreibt. Eine raumfüllende Verteilung stellt eine größtmögliche Diversität der initialen Datenpunkte sicher. Allerdings ist in der praktischen Anwendung nicht immer garantiert, dass ein initialer Datensatz von Grund auf neu aufgebaut werden kann. Möglicherweise existiert bereits ein bestehender Datensatz, der keine den Eingaberaum füllenden Eigenschaften besitzt. Aus diesem Grund wählen viele Autoren, beispielsweise Wu (2018) [DCJ18], einen zufälligen initialen Datensatz und verlassen sich darauf, dass der verwendete sequentielle

Algorithmus mittels Abwägung von Exploration und Exploitation zu einem befriedigenden Ergebnis kommen wird.

3 Methoden

3.1 Verwendete Software

'auto_tune' ist ein in der Programmiersprache Python implementiertes Framework, mit dessen Hilfe automatische, datengetriebene Optimierung durchgeführt werden kann. Im Folgenden werde ich kurz die für meine Arbeit relevanten Funktionalitäten aufzeigen.

auto_tune bietet die Möglichkeite, Datensätze aus separaten Dateien einzulesen oder einen Datensatz mittels statischer Samplingmethoden wie zufälligem Sampling oder dem weit verbreiteten Latin Hypercube Sampling zu erstellen. Diese Datensätze können dann mittels Präbeziehungsweise Postprocessing Methoden bearbeitet werden.

Zusätzlich stehen viele verschiedene Modelltypen des maschinellen Lernens zum Training auf Datensätzen zur Verfügung. Für meinene Arbeit habe ich ausschließlich K-Nearest-Neighbor-Regressoren, Entscheidungsbaum-Regressoren, Gauß-Prozess-Regressoren und Multilayer-Perceptron-Regressoren aus der sklearn-Bibliothek verwendet.

Ebenfalls sind viele der in ['https://www.sfu.ca/ ssurjano/optimization.html'] aufgelisteten Benchmarkfunktionen aus dem Bereich der Optimierung implementiert.

Die verschiedenen Funktionalitäten sind jeweils in entsprechenden Klassen gekapselt. Im Zuge meiner Arbeit werde ich Klassen für vier spezifische, iterative Samplingmethoden nach Vorbild der Klassen für statisches Sampling implementieren.

Für die Auswahl neuer potentieller Datenpunkte während jeder Iteration habe ich mich für den Pool-basierten, sequentiellen Ansatz entschieden. Das bedeutet in jeder neuen Iteration ein neuer Pool aus zufälligen, im Eingaberaum verteilten Datenpunkten erzeugt wird. Aus jedem iterativ erzeugten Pool wird genau ein Punkt ausgewählt, vom Orakel annotiert und dem Trainingsdatensatz hinzugefügt.

Des weiteren werde ich eine sogenannte 'IterativeLearner' Klasse erstellen, die mittels eines der von mir implementierten iterativen Samplingprozesse versucht, ein Surrogatmodell an die Ausgaben eines Orakels anzupassen.

3.2 Iterative Samplingmethoden

Liu et al. (2017) [HYJ17] teilen iterative Samplingmethoden basierend auf der verwendeten Exploitationsstrategie in vier verschiedene Kategorien, namentlich ausschussbasiertes, kreuzvalidierungsbasiertes, gradientenbasiertes und varianzbasiertes iteratives Sampling auf. Aus jeder dieser Kategorien werde ich in den folgenden Kapiteln jeweils eine Methode heraussuchen und innerhalb des 'auto_tune' Frameworks implementieren. Um die Flexibilität des 'auto_tune'-Frameworks bei der Wahl der verwendeten Modelle möglichst nicht einzuschränken, habe ich mich, außer beim varianzbasierten Ansatz, der speziell auf Gaußprozess-Modelle zugeschnitten ist, für generische, d.h. modellunabhängige Samplingstrategien, entschieden.

3.3 Zufälliger iterativer Sampler

Als erstes werde ich eine Klasse für einen zufällig agierenden iterativen Sampler erstellen. Dieser nutzt keinerlei Erkenntnisse über die aktuelle Verteilung der Trainingsdaten, unsicherheiten des verwendeten Surrogatmodells oder Ausgabewerte des Orakels. Falls iterativ genügend Datenpunkte gesammelt werden, liefert gleichverteiltes zufälliges Sampling annähernd eine gute, raumfüllende Datenpunktverteilung. Da die Berechnung von zufälligen Eingabewerten innerhalb des Eingaberaums $\mathcal{X}$ außerdem sehr günstig ist, spricht viel für die Wahl von zufälligem Sampling bei einer großen finalen Datenmenge. Allerdings ist diese Eigenschaft beim aktiven Lernen nicht hilfreich, da das Ziel schließlich die Erstellung eines kleinen Trainingsdatensatzes mit möglichst wenigen Auswertungen der teuren Zielfunktion ist. Für kleine Datensatzgrößen variiert die Qualität des trainierten Modells stark. Aus diesem Grund bietet sich die Verwendung von zufälligem iterativem Sampling zwar nicht für die praktische Verwendung an, dennoch werde ich für die späteren Experimente die Leistung eines Surrogatmodells, trainiert auf einem Datensatz der durch diese Methode erstellt wurde, als untere Schranke für die Leistung aller weiteren implementierten Samplingmethoden verwenden.

3.4 Ausschussbasierter iterativer Sampler

Im Folgenden implementiere ich zwei verschiedene Ausschussbasierte Verfahren. Eines realisiert lokale Exploitation mittels eines heterogenen Ausschusses, dessen Diversität durch verschiedene Modelltypen sichergestellt wird. Hierbei werden alle Modelle des Ausschusses auf allen verfügbaren Trainingsdaten trainiert.

Das zweite Verfahren verwendet stattdessen einen homogenen Ausschuss. Hierbei besitzen sind alle Ausschussmitglieder den selben Modelltyp. Um ein gewisses Maß an Uneinigkeit zwischen den Vorhersgen der einzelnen Modelle des Ausschusses zu garantieren, werden alle Modelle auf einer unterschiedlichen Teilmenge des verfügbaren Trainingsdatensatzes trainiert.

Dazu wird der Datensatz bei t Ausschussmitgliedern in t gleich große Teilmengen T_i zerlegt und das Training des Ausschussmodells $\hat{f}_i$ findet auf der Datenmenge $D_i = D \setminus T_i$ statt.

Um den Informationsgehalt eines nicht-annotierten Punktes aus dem vorliegenden Pool an möglichen neuen Datenpunkten zu bestimmen, verwende ich die Varianz zwischen den Vorhersagen der verschiedenen Auschussmitglieder. D.h.

$$exploit_{QBC}(x) = \frac{1}{t} \sum_{i}^{t} (\hat{f}_i(x) - \bar{f}(x))^2 \tag{3.1}$$

wobei $\bar{f}(x) = \frac{1}{t} \sum_i^t \hat{f}_i(x)$ gilt.

Li et al. (2009)[GSAA06] haben allerdings festgestellt, dass sich bei einem ausschließlich exploitativen Ausschussbasierten Ansatz ein Großteil der neuen Datenpunkte in Bereichen mit großer Uneinigkeit der einzelnen Hilfsmodelle, d.h. an Punkten großer Varianz in den Vorhersagen des Ausschusses, sammelt. Um lokale Clusterbildung zu vermeiden und ein gewisses Maß an Exploration sicherzustellen, wird die Auswahl eines neuen Datenpunktes nicht ausschließlich von der Größe der Varianz innerhalb der Vorhersagen der Ausschussmodelle abhängig gemacht. Eine eingeführte Abstandsbeschränkung stellt sicher, dass neue Datenpunkte einen Mindestabstand d von allen bisher bekannten Punkten X haben.

Mathematisch lässt sich das Auswahlverfahren für einen Pool P an möglichen Datenpunkten folgendermaßen beschreiben:

$$\begin{aligned} x_{neu} &= \arg\max_{x \in P} exploit_{QBC}(x) \\ \text{so dass: } &\|x - x_i\| \geq d \quad \forall x_i \in X \end{aligned} \tag{3.2}$$

Falls keiner der Datenpunkte innerhalb des Pools P die Abstandsbeschränkung erfüllen kann, so wird der neue Datenpunkt ausschließlich anhand des Kriteriums $exploit_{QBC}$ ausgewählt.

Der Mindestabstand kann über jeweils über einen von zwei verschiedenen Ansätzen bestimmt werden. Möglich ist entweder die Methode von Li et al. (2010) [GVS10], bei der der gemittelte minimale Abstand aller bekannten Datenpunkte $x_i, x_j \in X$ verwendet wird.

$$d = \frac{1}{m} \sum_{i}^{m} min \, \|x_i - x_j\|_2 \qquad (x_i \neq x_j) \tag{3.3}$$

Wahlweise lässt sich auch der halbierte größte minimale Abstand aller bekannten Datenpunkte als Abstandsbeschränkung nutzen (vgl. [VKO+13]). Unter Verwendung der selben Notation wie oben bedeutet das:

$$d = \frac{1}{2} \max(\min \|x_i - x_j\|_2) \qquad (x_i \neq x_j) \tag{3.4}$$

Ursprünglich beziehen sich Li et al. (2010) [GVS10] und Aute et al. (2013) [VKO+13] mittels ihrer Abstandsbeschränkungen auf kreuzvalidierungsbasierte aktive Lernalgorithmen. Experimente haben aber gezeigt, dass sich die Qualität der mittels QBC erstellten Surrogatmodelle durch Verwendung dieser Abstandsbeschränkungen deutlich gegenüber rein exploitativen Ausschussbasierten Ansätzen verbessert.

3.5 Kreuzvalidierungsbasierter iterativer Sampler

Bei der Implementierung des kreuzvalidierungsbasierten iterativen Samplingverfahrens habe ich mich an der Methodik von Aute et al. (2013) [VKO+13] orientiert.

Die Forschungsgruppe schlägt vor, anstatt des absoluten LOO-Kreuzvalidierungsfehlers eine modifizierte Variante zu berechnen. Aute et al. haben festgestellt, dass die Größe des LOO-Kreuzvalidierungsfehlers abhängig von der Größe des Zielwerts ist. Dieses Problem kann durch Normalisierung der Zielwerte und der Modellvorhersagen umgangen werden. Da allerdings im Voraus keine Kenntnisse über den Wertebereich der Zielfunktion vorliegen definieren sie folgende neue Gleichung zur Berechnung des Kreuzvalidierungsfehlers:

$$\hat{e}_r(x_i) = \left| \frac{f(x_i) - \hat{f}^{-i}(x_i)}{f(x_i)} \right| \tag{3.5}$$

Um den LOO-Kreuzvalidierungsfehler an neuen potentiellen Datenpunkten zu bestimmen, wird ein Kriging-Fehlermodell anhand der LOO-Kreuzvalidierungsfehler der bekannten Punkte trainiert.

Zur Vermeidung von Clusterbildung in Regionen mit großem Fehler habe ich die selben Abstandsbeschränkungn wie für die Ausschussbasierten Modelle implementiert.

3.6 Varianzbasierter iterativer Sampler

Auch 'Adaptive maximum entropy' (AME) - der varianzbasierte iterative Samplingansatz vorgeschlagen von Liu et al. (2016) [HSY+16] - nutzt unter anderem LOO-Kreuzvalidierung, um Fehlerinformationen des Modells in die Samplingstrategie zu integrieren.

AME baut auf der raumfüllenden 'maximum entropy'(ME)-Methode auf und erweitert diese durch einen exploitativen Komponenten, mittels dessen interessante Regionen gefunden und dichter mit Datenpunkten gefüllt werden können.

Die ME-Methode bestimmt einen neuen Datenpunkt durch Maximierung der Determinante der Kovarianzmatrix $\boldsymbol{\sigma}$:

$$x_{neu} = \arg\max_{x \in \bar{X}} det(\boldsymbol{\sigma}) \tag{3.6}$$

Die Kovarianzmatrix wird folgendermaßen bestimmt:

$$\boldsymbol{\sigma} = \sigma^2 \begin{pmatrix} \mathbf{R}_{XX} & \mathbf{R}_{XX_{neu}} \\ \mathbf{R}_{X_{neu}X} & 1 \end{pmatrix} \tag{3.7}$$

Der Faktor σ^2 bezeichnet hierbei die Varianz innerhalb der bekannten Eingabedaten. Unter einer stationären Annahme wird dieser Wert für alle Koordinatenrichtungen als konstant

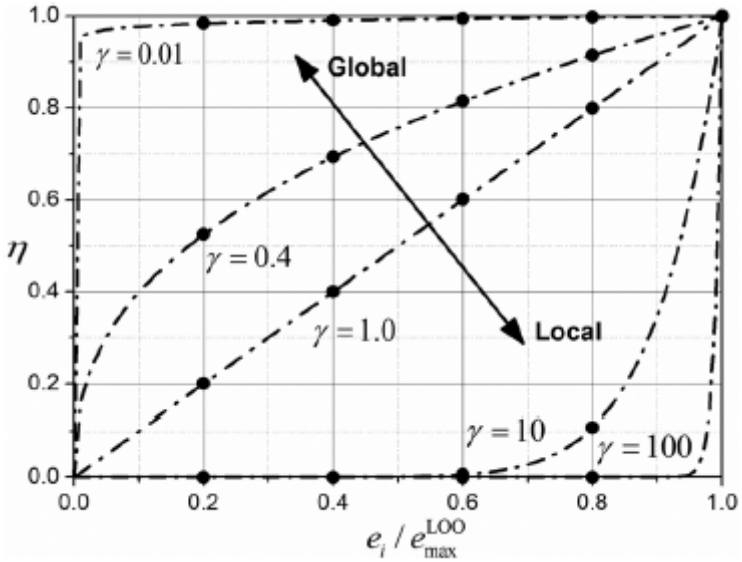

Abbildung 3.1: Plot des Parameters η für verschiedene Werte γ (vgl. [HSY+16])

betrachtet. $X_{neu} = X \bigcup \{x_{new}\}$ bezeichnet die Menge aller im Trainingsdatensatz enthaltenen Eingabewerte, erweitert durch einen neuen Eingabepunkt, für den noch keine bekannten Zielwerte durch das Orakel vorliegen.

R beschreibt die mittels Informationen über Vorhersagefehler angepasste Korrelationsfunktion zwischen zwei Punkten $x_i, x_j \in \mathcal{X}$ und wird von Liu et al folgendermaßen definiert:

$$\mathbf{R}_{adj}(x_i, x_j) = \prod_{k=1}^{d} exp(-\eta_i \eta_j \theta_k |d_k|^2) \tag{3.8}$$

θ_k beschreibt die Korrelation und d_k den Abstand zwischen x_i und x_j in Koordinatenrichtung k. Die Faktoren η_i bzw. η_j beschreiben zwei Anpassungsfaktoren für die Punkte x_i und x_j. Sie werden können über die LOO-Kreuzvalidierungsfehler bestimmt werden. Für LOO-Kreuzvalidierungsfehler in bekannten Datenpunkten $x_i \in X$ gilt:

$$\hat{e}(x_i) = \left| f(x_i) - \hat{f}^{-i}(x_i) \right| \tag{3.9}$$

Mit maximalem Kreuzvalidierungsfehler $\hat{e}_{max} = \max_{x_i \in X} \hat{e}(x_i)$ gilt nun für die Anpassungsfaktoren:

$$\eta_i = \left(\frac{\hat{e}(x_i)}{e_{max}} \right)^{\gamma} \tag{3.10}$$

Der Faktor γ ist zuständig für die Abwägung zwischen globaler Exploration und lokaler Exploitation.

Abb. 3.1 Veranschaulicht die Auswirkungen des Exponenten γ auf den Koeffizienten η. Für sehr kleine γ nehmen die Koeffizenten η Werte sehr nahe an eins an. Für große Exponenten ist

allerdings ersichtlich, dass η selbst für relativ kleine LOO-Kreuzvalidierungsfehler sehr kleine Werte annimmt, was im Extremfall dazu führen kann, dass 3.8 für alle x_i, x_j ausschließlich den Wert 1 zurückgibt. Hierdurch ist die resultierende Matrix $\boldsymbol{\sigma}$ unterbestimmt ist und damit ergibt die Evaluation der Determinante für nahezu alle möglichen neuen Datenpunkte Null. D.h. falls γ zu groß gewählt wird, können Schwierigkeiten beim Exploitationsschritt auftreten.

Gleichung (3.10) kann nur für Punkte mit bereits bekanntem Zielwert bestimmt werden. Zur Berechnung der Korrelationsmatrix $\boldsymbol{\sigma}$ muss allerdings auch die Korrelationsfunktion (3.8) für einen Punkt $x_{neu} \in \bar{X}$ mit unbekannten Ausgabewerten bestimmt werden. Für diesen Fall wird für η_{neu} der Wert des Anpassungsfaktors des nächstgelegenen bekannten Datenpunkts gewählt. Für genaurere Angaben verweise ich auf das Paper von Liu et al. (2016) [HSY+16].

Der folgende Algorithmus verdeutlicht noch einmal die Funktionsweise des von mir implementierten varianzbasierten iterativen Samplers.

Algorithmus 3.1 Iteratives AME Sampling

Require:
Orakel $f : \mathcal{X} \to \mathcal{Y}$
Kriging-Modell $\hat{f} : \mathcal{X} \to \mathcal{Y}$
$D = (X, y = f(X)) \subset \mathcal{X} \times \mathcal{Y}$
$\boldsymbol{\gamma} = [\gamma_1, ..., \gamma_N]$

1: **while** ¬ Abbruchkriterium **do**
2: erstelle Pool $P \subset \bar{X}$
3: wähle γ aus $\boldsymbol{\gamma}$
4: berechne Kreuzvalidierungsfehler $\hat{e}(x), x \in X$
5: berechne alle Anpassungsfaktoren η
6: $x_{new} = \arg\max_{x \in P} \boldsymbol{\sigma}$
7: erweitere $D = (X \bigcup \{x_{neu}\}, f(X) \bigcup \{f(x_{neu}\})$
8: **end while**

3.7 Gradientenbasierter iterativer Sampler

Um die von auto_tune gebotene Flexibilität bezüglich der Surrogatmodellwahl zu erhalgen habe ich mich für das von Crombecq et al. (2009) [KDDT11] vorgestellte, modellunabhängige LOLA-Voronoi-Samplingverfahren entschieden.

Grundidee dieses Verfahrens ist es, Regionen des Eingaberaums ausfindig zu machen, in denen die Zielfunktion f ein nicht-lineares Verhalten zeigt. Zu diesem Zweck wird jedem Datenpunkt des Trainingsdatensatzes $x \in X$ ein hybrider Score $H(x) = V(x) + G(X)$ zugewiesen. $V(x)$ weist der Dichte der bereits bekannten Datenpunkte in der Umgebung von x einen Wert zu. Hierfür wird die Voronoi-Zerlegung des Eingaberaums anhand der Trainingsdatenpunkte mittels Monte-Carlo-Approximation bestimmt. Dies ist kostet deutlich weniger Rechenleistung

und skaliert besser mit höheren Dimensionen des Eingaberaums als explizite Berechnung der Voronoi-Zerlegung mittels Delaunay-Triangulation. Außerdem lassen sich die zufällig gezogenen Punkte der Monte-Carlo-Approximation als Pool potentieller neuer Datenpunkte wiederverwenden. $V(x)$ entspricht dem geschätzten, relativen Volumen der Voronoi-Zelle, deren Zentrum x bildet.

$G(x)$ stellt ein Maß für die Nicht-Linearität der Umgebung von x dar. Dieses Maß lässt sich anhand von Informationen über das Gradientenverhalten in x bestimmen. Der Gradient einer Funktion f an einem gegebenen Punkt $x_0 \in \mathcal{X}$ hat die Eigenschaft, dass er die best mögliche lokale, lineare Approximation dieser Funktion um x_0 darstellt.

$$f(x) = f(x_0) + \nabla f_{x_0} * (x - x_0) \tag{3.11}$$

Da der Gradient der Black-Box-Zielfunktion f nicht bekannt ist, wird dieser mittels lokalen-linearen Approximationen (LOLA) durch bekannte Ausgabewerte in benachbarten Punkten abgeschätzt. Ein sehr großer Fokus dieser Methode liegt darauf, geeignete Nachbarschaften zur Approximation der lokalen Gradienten zu finden. Das genaue Vorgehen hierfür wird ausführlichst von Crombecq et al. (2011) beschrieben.

Die Nachbarschaft von Punkt $x \in X$ wird folgendermaßen definiert: $N(x) = \{x_{n1}, ..., x_{nk}\}$ wobei $k = 2 * d$ der doppelten Anzahl an Dimensionen des Eingaberaums entspricht.

Der lokale Gradient g für x kann nun mit Hilfe von $N(x)$ durch Lösen des folgenden Gleichungssystems mittels 'Least-Squares'-Approximation bestimmt werden:

$$\begin{pmatrix} x_n 1^{(1)} - x^{(1)} & \dots & x_{n1}^{(d)} - x^{(d)} \\ \vdots & & \vdots \\ x_{nk}^{(1)} - x^{(1)} & \dots & x_{nk}^{(d)} - x^{(d)} \end{pmatrix} \begin{pmatrix} g^{(1)} \\ \vdots \\ g^{(d)} \end{pmatrix} = \begin{pmatrix} f(x_{n1}) \\ \vdots \\ f(x_{nk}) \end{pmatrix} \tag{3.12}$$

Allerdings hängt die Qualität der Gradientenschätzung von der Qualität der gewählten Nachbarschaft ab.

Das Maß der Nicht-Linearität in der Umgebung von x kann über die Abweichung der Ausgabewerte der Nachbarn von x von der lokalen-linearen Approximation (vgl. 3.11) durch x bestimmt werden.

$$G(x) = \sum_{i=1}^{k} |f(x_{ni}) - (f(x) + g * (x_{ni} - x))| \tag{3.13}$$

Da $V(x)$ das relative Volumen der einzelnen Voronoi-Zellen angibt liegen alle Werte innerhalb des Intervalls $[0, 1]$. Deshalb muss die Metrik $G(x)$ zur Berechnung des hybriden Scores $H(x)$ ebenfalls normiert werden. D.h.

$$H(x) = V(x) + \frac{G(x)}{\sum_j G(x_j)} \tag{3.14}$$

Sobald der hybride Score $H(x)$ für alle bekannten Trainingsdaten bestimmt ist, kann ein neuer Datenpunkt x_{neu} aus der Voronoi-Zelle um Punkt x mit größtem hybriden Score ausgewählt werden. Für die Auswahl eines neuen Datenpunktes bietet es sich an, die bei der Monte-Carlo-Simulation erzeugten Datenpunkte wieder zu verwenden. Der neue Datenpunkt sollte den größt möglichen Abstand von allen anderen bereits bekannten Datenpunkten besitzen.

Diese Methode ist gleichzeitig sehr einfach für Batch-Modus-Sampling erweiterbar. Um ein Batch der Größe n an neuen Datenpunkten zusammen zu stellen, kann je ein neuer Datenpunkt aus den n Voronoi-Zellen mit höchstem hybriden Score gezogen werden. Dies stellt die Diversität der Datenpunkte innerhalb des Batches sicher.

4 Ergebnisse

4.1 Rahmenbedingungen

Da das Verhalten der iterativen Samplingmethoden von der Wahl der spezifischen Hyperparameter abhängt stelle ich im Folgenden kurz die verwendeten Hyperparameterwerte vor, damit die Ergebnisse einfach reproduzierbar sind.

Alle verwendeten Benchmarkfunktionen sind auf den Bereich $\mathcal{X} = [-1, 1]^d$ skaliert.

Der initiale Trainingsdatensatz wird mittels Gleichverteilung zufällig erstellt und beinhaltet $m_{init} = \frac{m_{max}}{2}$ Datenpunkte. m_{max} repräsentiert hierbei die maximale Datensatzgröße.

Datenpunkte für den zufällig während jeder Iteration erstellten Datenpool werden mittels einer Gleichverteilung über den Eingaberaum verteilt. Die größe der während jeder Iteration zufällig erstellten Pools an potentiellen neuen Datenpunkten beträgt für die ausschussbasierten und kreuzvalidierungsbasierten Lerner 500 Datenpunkte. Das LOLA-Voronoi Verfahren verwertet die zufällig generierten Punkte der Monte-Carlo-Simulation zur Approximation der Volumina der einzelnen Voronoi-Zellen als Pool möglicher neuer Datenpunkte. D.h. die Poolgröße lässt sich folgendermaßen bestimmen: $|P| = 100 * |D|$.

Da die numerische Berechnung der Determinante großer Matrizen sehr teuer ist und für jeden Datenpunkt innerhalb des Datenpools durchgeführt werden muss, habe ich mich dazu entschieden, den Pool der AME Samplingmethode auf 100 Datenpunkte zu reduzieren.

Die meisten der verwendeten Hilfsmodelle, d.h. K-nächste-Nachbarn (KNeighborsRegressor), Entscheidungsbaum-Regressoren (DecisionTreeRegressor) und Gaußprozess-Modelle (GaussianProcessRegressor) nutzen die vorbelegten Standardeinstellungen der sk_learn python Bibliothek. Die verwendeten flachen neuronalen Netze (MLPRegressor) besitzen 32 verdeckte Neuronen und werden maximal über 8000 Iterationen trainiert. Standardmäßig wird das Training dieser Modelle nach 200 Trainingsiterationen abgebrochen. Bei anfänglichen Testläufen hat dies allerdings immer dazu geführt, dass das Training vor erreichter Konvergenz abgebrochen wurde und das trainierte Netz deshalb keine guten Vorhersagen liefern konnte.

Die beim homogenen und heterogenen ausschussbasierenden Samplingverfahren verwendeten Ausschüsse bestehen jeweils aus genau drei Mitgliedern. Während der homogene Ausschuss aus drei flachen neuronalen Netzen besteht, setzt sich der heterogene Ausschuss aus einem KNN-Modell, einem Entscheidungsbaum-Regressor und einem flachen neuronalen Netz zusammen.

Globale Exploration wird bei ausschuss- und kreuzvalidierungsbasierten Verfahren mittels der gemittelten minimalen Distanz (vgl. 3.3) gewährleistet.

Das verwendete AME Sampling Verfahren nutzt $\boldsymbol{\gamma} = \{0, 0.5, 1, 4\}$, um zwischen globaler Exploration und lokaler Exploitation abzuwägen. Liu et al. (2016) führen in ihrem Paper eine wesentlich aggressivere lokale Exploitationsstartegie mit einem Exponenten von 100 durch. Dieser Wert führt bei meiner Implementierung allerdings dazu, nahezu alle der Koeffizienten η verschwindend klein werden. Damit nimmt Gleichung 3.8 für fast alle Punkte x_i, x_j den Wert 1 an, was dazu führt, dass in 3.7 die Spalten von $\boldsymbol{\sigma}$ nicht mehr paarweise voneinander verschieden sind. Dies führt dazu, dass die Evaluation der Determinante für alle potentiellen neuen Datenpunkte 0 ergibt. Der Schritt, der eigentlich für lokale Exploitation sorgen sollte führt bei meiner Implementierung somit zur Auswahl eines, sehr teuren, zufälligen Datenpunktes. Dieses Problem tritt bei Liu et al. möglicherweise nicht auf, da bei ihrer Arbeit statt eines poolbasierten Samplingansatzes ein membership-query-synthesis Ansatz verwendet wird.

Als a-priori-Varianz σ^2 des initialen Datensatzes habe ich den Wert 10 verwendet.

4.2 Experimentelle Ergebnisse

Zur Berechnung des mittleren quadratischen Fehlers und des R^2-Scores verwende ich einen separaten Testdatensatz, der aus 500 gleichverteilten Datenpunkten innerhalb des skalierten Eingaberaums besteht. Alle Verfahren werden ausgeführt, bis sie 150 Trainingsdatenpunkte gesammelt haben.

4.2.1 Ackley-Funktion

Ackley-Funktionsgleichung:

$$f(x) = -a\, exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{d}\sum_{i=1}^{d} cos(cx_i)\right) + a + exp(1) \tag{4.1}$$

wobei $a = 20$, $b = 0.2$, $c = 2\pi$.

Abb. 4.1 stellt den Plot der zweidimensionalen Ackley-Funktion dar. Die Funktion hat eine trichterartige Form und besitzt ein globales Minimum im Ursprung. In den äußeren Bereichen des Eingaberaums ist sie sehr flach und weist sie eine Vielzahl lokaler Mini- und Maxima auf.

Abb. 4.2 und 4.3 zeigen deutlich, dass alle der verwendeten adaptiven Samplingstrategien für zweidimensionale Eingabewerte einem zufälligen Design in Hinblick auf Sampeleffizienz deutlich überlegen sind - was selbstverständlich auch zu Erwarten ist. Allerdings benötigt das Ausschussbasierte Verfahren deutlich mehr Datenpunkte, um den Fehler zu senken als

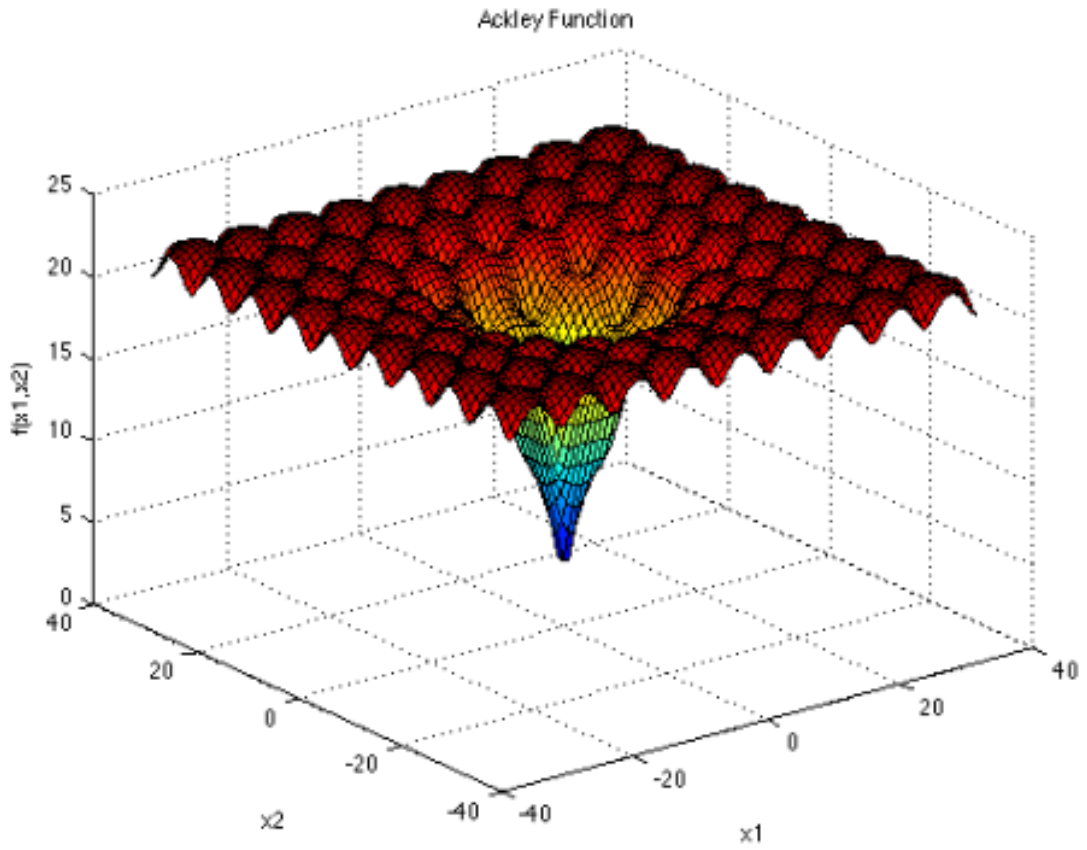

Abbildung 4.1: Ackley-Funktion in 2D (https://www.sfu.ca/ ssurjano/ackley.html)

LOLA-Voronoi, AME und kreuzvalidierungsbasiertes Sampling. Letzteren beiden Methoden gelingt es am schnellsten den Vorhersagefehler auf dem Testdatensatz zu reduzieren.

In der Literatur werden die meisten Experimente mit sehr niedrigdimensionalen Eingabedaten durchgeführt. Aus diesem Grund habe ich bei meinem zweiten Experiment die Dimensionalität des Eingaberaums deutlich erhöht. Es ist zu beobachten, dass für höherdimensionale Daten die Ausschussbasierten Verfahren Modelle mit höherer Qualität trainieren, als AME und Kreuzvalidierung. LOLA-Voronoi gelingt es sowohl für niedrig-dimensionale, als auch für höher-dimensionale Eingaberäume die Modellvorhersagequalität schnell zu verbessern.

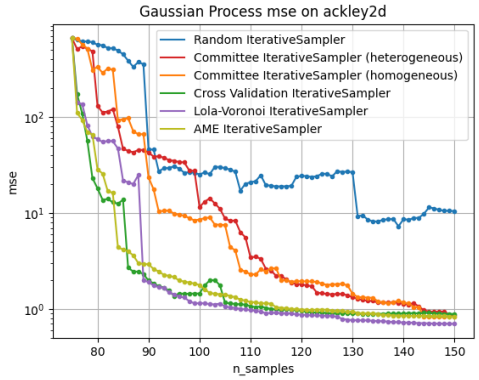

Abbildung 4.2: MSE-Verläufe auf 2D Ackley-Funktion

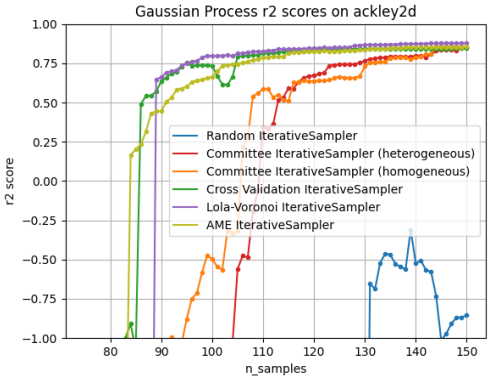

Abbildung 4.3: R^2-Verläufe auf 2D Ackley-Funktion

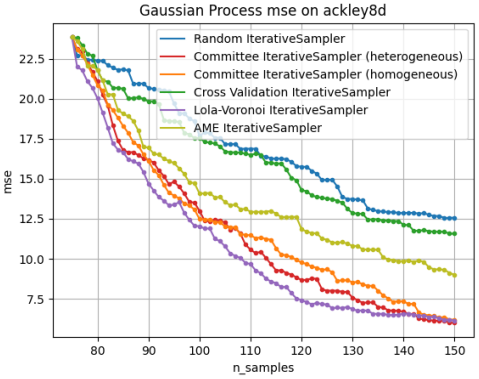

Abbildung 4.4: MSE-Verläufe auf 8D Ackley-Funktion

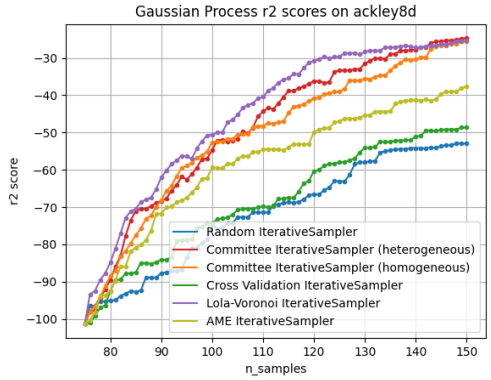

Abbildung 4.5: R^2-Verläufe auf 8D Ackley-Funktion

4.2.2 Rosenbrock-Funktion

Die Rosenbrock-Funktion oder auch Rosenbrock-Banane wird durch folgende Funktionsgleichung beschrieben.

$$f(x) = \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \tag{4.2}$$

Bei der Rosenbrock-Funktion lassen sich vor allem im zwei-dimensionalen Fall einige interessante Dinge beobachten. Selbst mit 150 verfügbaren Trainingsdatenpunkten ist der mittlere quadratische Vorhersagefehler aller Methoden auf dem Testdatensatz sehr groß. Beim homogenen ausschussbasierten Ansatz verschlechtert sich sogar die Vorhersageleistung des Modells innerhalb der ersten 30 Iterationen deutlich. Gemittelt über 10 Durchläufe liefert das Verfahren anfangs sogar schlechtere Modelle als das rein zufällige Sammeln von neuen Eingabedaten. Auch der heterogene ausschussbasierte Ansatz hat zwischenzeitlich große Probleme, seinen

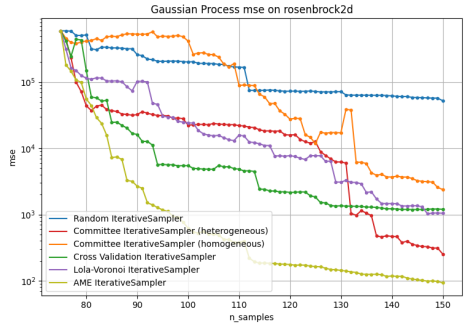

Abbildung 4.6: MSE-Verläufe auf 2D Rosenbrock-Funktion

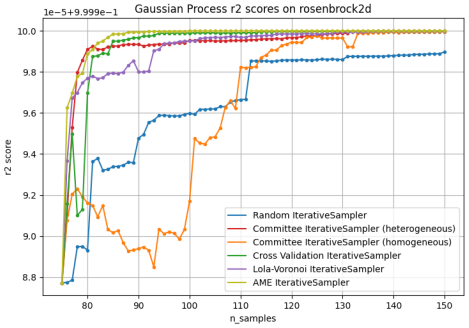

Abbildung 4.7: R^2-Verläufe auf 2D Rosenbrock-Funktion

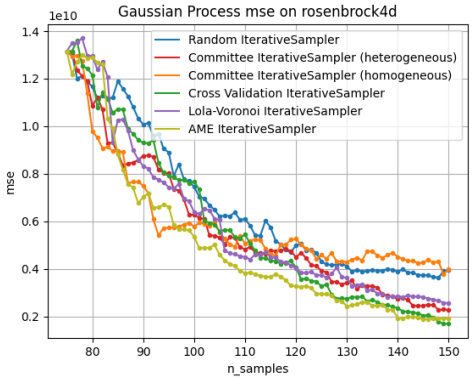

Abbildung 4.8: MSE-Verläufe auf 4D Rosenbrock-Funktion

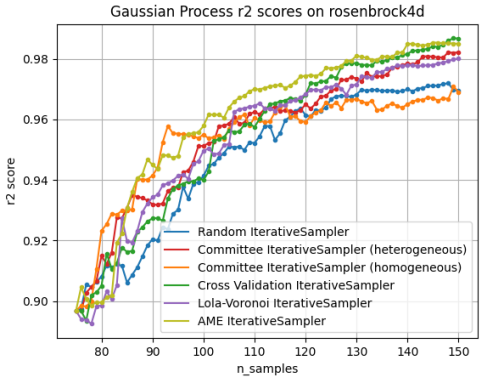

Abbildung 4.9: R^2-Verläufe auf 4D Rosenbrock-Funktion

MSE auf dem Testdatensatz zu verringern, wobei der R^2-Score des Testdatensatzes sehr nahe vor eins, dem Höchstwert, liegt. Das einzige Verfahren, dem es gelingt konstant einen Vorhersagefehler zu verbessern ist AME. Bei allen anderen Verfahren lässt sich ab einem bestimmten Punkt Stagnation beobachten.

Im vier-dimensionalen Beispiel (siehe Abb. 4.8 und 4.9) lässt sich die Diskrepanz zwischen gewaltigem mittleren quadratischen Fehler auf den Testdaten und nahezu perfektem R^2-Score noch besser beobachten.

Im Mittel gelingt es dem homogenen ausschussbasierten Sampler sehr schnell, den Fehler seines Modells deutlich zu reduzieren, aber ab einer Datensatzgröße von knapp über 90 Datenpunkten beginnt er allerdings stark zu stagnieren, so dass die Leistung des Modells am Ende vergleichber mit der des zufällig agierenden Samplers ist.

4.3 Sampling-Zeitaufwand

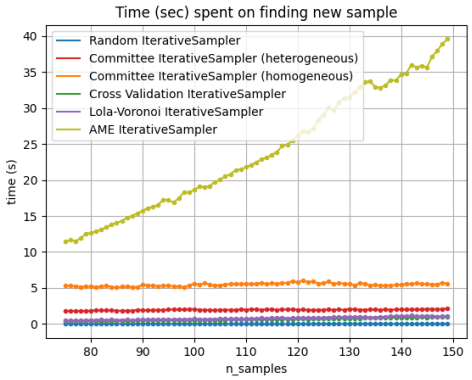

Abbildung 4.10: Zeitaufwand pro neuem Datenpunkt

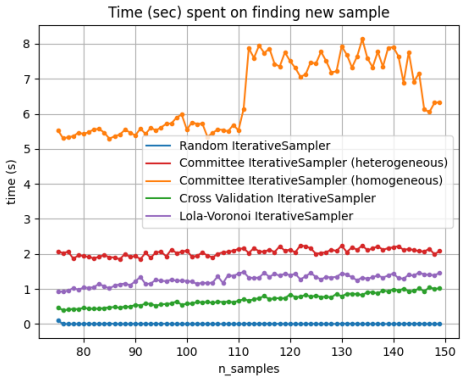

Abbildung 4.11: Zeitaufwand pro neuem Datenpunkt ohne AME

Die Zeitmessung wurde auf 2x Intel Xenon Silver 4116 mit 188 GB RAM durchgeführt.

In Abb. 4.10 ist deutlich erkennbar, dass die AME-Methode aufgrund von verwendetem LOO-Kreuzvalidierungsverfahren und teurer numerischen Besitmmung der Determinante der Korrelationsmatrix nichtlinear mit der Anzahl an bekannten Trainingsdaten ansteigt.

Abb. 4.11 zeigt, dass bei den ausschussbasierten Verfahren die benötigte Zeit pro Datenpunkt im Trend nahezu konstant bleibt. Die Samplingdauer des homogenen ausschussbasierten Samplers ist sehr stark mit Rauschen belegt. Dies liegt sehr wahrscheinlich am Training der verwendeten flachen Neuronalen Netzen der sk_learn-Bibliothek. Das Training der Netze wird über maximal 8000 Iterationen ausgeführt, allerdings kann bei Konvergenz das Training schon deutlich früher beendet werden.

Die Berechnungsdauer eines neuen Datenpunktes bei LOLA-Voronoi und dem kreuzvalidierungsbasierten Verfahren scheint annähernd linear mit der Trainingsdatensatzgröße zu skalieren.

4.4 Diskussion

Alles in allem scheint das AME-Verfahren für niedrig-dimensionale Eingabedaten sehr erfolgsversprechend zu sein. Abgesehen vom achtdimensionalen Eingaberaum der Ackley-Funktion gelingt es diesem Verfahren stets den MSE-Testfehler mittels weniger neuer Datenpunkte stark zu verringern.

Dieses Resultat ist allerdings auch zu erwarten, da AME als einziges Verfahren ein optimales sequentielles Design verwendet. D.h. das Verfahren ist genau auf die Nutzung eines

Gaußprozess-Modells zugeschnitten und kann bekannte Charakteristiken des Modells ausnützen.

Da die anderen Verfahren allesamt ein generisches sequentielles Design verwenden, können sie die Struktur des verwendeten Modells nicht verwenden, um den Samplingprozess zu optimieren.

Allerdings liefert das LOLA-Voronoi Verfahren in höher-dimensionalen Eingaberäumen die besten Ergebnisse. Dies liegt möglicherweise daran, dass das Verfahren im Gegensatz zu den ausschussbasierten und dem kreuzvalidierungsbasierten Verfahren auf einem robusten explorationsverfahren beruht.

Das deutlich schlechtere Abschneiden der ausschussbasierten Modelle könnte auf das Ausbleiben von Hyperparameteroptimierung zustande gekommen sein. Während LOLA-Voronoi und das kreuzvalidierungsverfahren keine wichtigen Hyperparameter besitzen, nimmt der Modell-Ausschuss beim ausschussbasierten Sampling die zentrale Rolle ein. Falls die Modelle des Ausschusses nicht in der Lage sind, das Verhalten der Zielfunktion annähernd immitieren zu können, wird der Samplingprozess fehlgeleitet.

Da es für die Hyperparameteroptimierung allerdings ebenfalls teure Simulationsauswertungen benötigt, sind Verfahren ohne oder mit wenigen hyperparametern vorzuziehen.

Besonders interessant ist das Verhalten von R^2-Score und MSE des Testdatensatzes bei der zweidimensionalen Rosenbrock-Funktion im Hinblick auf mögliche Abbruchkriterien des iterativen Samplingvorgangs. Trotz sehr hoher R^2 Werte früh im Training kann der mittlere quadratische Fehler beim AME-Ansatz auf der Testdatenmenge noch deutlich reduziert werden.

Auch die Verwendung einer Testdatenmenge zur Überprüfung des Abbruchkriteriums muss in praktischen Anwendungsfällen hinterfragt werden. Da sich viele bestehende Verfahren das Kreuzvalidierungsverfahren zu Nutze machen, um den Samplingprozess zu leiten, wäre es auch denkbar, diese bereits durchgeführten Berechnungen weiter zu verwenden und das Abbruchkriterium beispielsweise mittels Kreuzvalidierung zu überprüfen.

In Bezug auf die Sampeleffizienz lässt sich feststellen, dass das AME-Verfahren zwar bei den meisten der Experimenten die besten Modelle zurückliefert, allerdings ist dies mit einem deutlich erhöhtem Rechenaufwand gegenüber der anderen Verfahren verbunden. Je nach praktischem Anwendungsfall, beispielsweise wenn die Auswertung der Zielfuntkion Stunden oder gar Tage dauert, sind diese erhöhten Kosten vernachlässigbar.

Da allerdings der Aufwand, die Position eines neuen Datenpunktes zu bestimmen bei AME exponentiell ansteigt, wäre es denkbar, den Trainingsprozess auf zwei oder mehrere verschiedene adaptive Samplingverfahren aufzuteilen. So könnten die ersten Schritte mittels AME durchgeführt werden, im Anschluss könnte vielleicht ein ausschuss- oder kreuzvalidierungsbasiertes Verfahren übernehmen.

5 Zusammenfassung und Ausblick

Im Zuge dieser Arbeit habe ich vier verschiedene iterative Samplingmethoden in das auto_tune-Framework integriert: ein ausschussbasiertes iteratives Verfahren, ein kreuzvalidierungsbasiertes Verfahren, das LOLA-Voronoi-Verfahren und das iterative AME-Verfahren.

Diese Samplingansätze habe ich mittels der Ackley- und Rosenbrock- Benchmarkfunktionen aus dem Bereich der Optimierung hinsichtlich Sampeleffizienz (wie viele Sampel werden benötigt, um eine vorgegebene Güte zu erreichen) und Samplingeffizienz (wie lange dauert es, bis ein neuer Datenpunkt ausgewählt ist) untersucht.

Hinsichtlich der Sampeleffizienz konnte ich feststellen, dass die Qualität der verschiedenen Methoden stark von den Eigenschaften der Zielfunktion abhängig ist. Ein universell anwendbares, bestes Verfahren hat sich nicht für die verwendeten Benchmarkfunktionen heraus kristallisiert.

In Bezug auf Samplingeffizienz hat sich ergeben, dass die ausschussbasierten Verfahren für sehr kleine Datensatzgrößen zwar etwas schlechter abschneiden als das LOLA-Voronoi bzw. das kreuzvalidierungsbasierte Verfahren, allerdings hinsichtlich größerer Trainingsdatensätze deutlich besser skalieren. D.h. die von ihnen benötigte Zeit um einen neuen Datenpunkt zu bestimmen wächst nur sehr langsam mit steigender Datensatzgröße.

Je nach verfügbarem Budget eignen sich ausschussbasierte Verfahren also vergleichsweise gut zur iterativen Erweiterung von etwas größeren Datensätzen. So wäre beispielsweise auch die Kombination zweier adaptiver Samplingverfahren denkbar. Um Flexibilität bei der Modellwahl zu erhalten wäre es möglich, den Samplingvorgang mittels LOLA-Voronoi zu starten und, sobald bei dieser Methode die Samplingkosten überhand nehmen, auf ein ausschussbasiertes Verfahren umzuschwenken.

Die Verwendung von MSE und R^2-Scores als Abbruchkriterium des Trainingsvorgangs hat sich als problematisch herausgestellt. Am Beispiel der Rosenbrockfunktion hat sich gezeigt, dass ein sehr guter R^2-Score auf dem Testdatensatz nicht gleichbedeutend mit einem geringen MSE ist.

Zusätzlich ist, wie bereits von vielen Autoren bestätigt, deutlich geworden, dass höherdimensionale Eingabedaten adaptiven Samplingmethoden häufig Schwierigkeiten bereiten. Vielleicht bietet tiefes aktives Lernen eine Lösung für höherdimensionale Problemstellungen.

Literaturverzeichnis

[AJ95] K. A, V. J. „Neural network ensembles, cross validation, and active learning“. In: *Advances in Neural Information Processing Systems* (1995), S. 231–238 (zitiert auf S. 11).

[BYD09] K. B, L. Y, C. D-H. „Construction of the radial basis function based on a sequential sampling approach using cross-validation“. In: *Mech Sci Technol 23* 12 (2009), S. 3357–3365 (zitiert auf S. 17).

[DCJ18] W. D, L. C-T, H. J. „Active Learning for Regression Using Greedy Sampling“. In: (2018) (zitiert auf S. 16, 17).

[DKHT11] D. D, C. K, N. HM, D. T. „Adaptive sampling algorithm for macromodeling of parameterized-parameter responses“. In: *IEEE Trans Microwave Theory Tech* (2011), S. 39–45 (zitiert auf S. 14).

[FFN+12] D. F, M. F, A. N, P. E, B. Y, B. N. „Active learning for spectroscopic data regression“. In: *J Chemom* 26.7 (2012), S. 374–383 (zitiert auf S. 11).

[FTVV14] V. FA, S. TW, B. V, T. V. „Metamodeling in multidisciplinary design optimization: how far have we really come?“ In: *AIAA* (Feb. 2014), S. 670–690 (zitiert auf S. 1, 17).

[GBC16] I. Goodfellow, Y. Bengio, A. Courville. *Deep Learning*. http : / / www . deeplearningbook.org. MIT Press, 2016 (zitiert auf S. 6).

[GSAA06] L. G, A. S, F.-M. A, D. A. „Approximation of multiresponse deterministic engineering simulations: a dependent metamodeling approach“. In: *Struct Multidiscip Optim* 31 (2006), S. 260–269 (zitiert auf S. 8, 21).

[GVS10] L. G, A. V, A. S. „An accumulative error based adaptive design of experiments for offline metamodeling“. In: *Struct Multidiscip Optim* 40.1 (2010), S. 137–155 (zitiert auf S. 12, 14, 21).

[HSX15] L. H, X. S, W. X. „Sequential sampling designs based on space reduction“. In: *Eng Optim* 47.7 (2015), S. 867–884 (zitiert auf S. 13).

[HSY+16] L. H, X. S, M. Y, C. X, W. X. „An adaptive Bayesian sequential sampling approach for global metamodeling“. In: *JMech Des* (2016) (zitiert auf S. 13, 17, 22–24).

[HYJ17] L. H, O. Y-S, C. J. „A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design“. In: *Struct Multidisc Optim* (2017), S. 393–416 (zitiert auf S. 3, 9, 11, 12, 20).

[JJW09] L. JL, S. J, W. WJ. „Choosing the sample size of a computer experiment: a practical guide“. In: *Technometrics* (2009), S. 366–376 (zitiert auf S. 17).

[KDDT11] C. K, G. D, D. D, D. T. „A novel hybrid sequential design strategy for global surrogate modeling of computer experiments“. In: *International Journal of Information Management* (2011), S. 1948–1974 (zitiert auf S. 7, 9, 14, 16, 24).

[KIDT09] C. K, C. I, G. D, D. T. „Space-Filling Sequential Design Strategies for Adaptive Surrogate Modelling“. In: *International Journal of Information Management* (2009) (zitiert auf S. 1, 6, 8, 16).

[MWM11] R. M, Y. W, D. M. „Adynamic sampling method for kriging and cokriging surrogate models“. In: *AIAA* (2011), S. 883 (zitiert auf S. 13).

[PDT13] S. P, D. D, D. T. „A balanced sequential design strategy for global surrogate modeling“. In: *IEEE* 36.1 (2013), S. 2172–2179 (zitiert auf S. 9).

[PLQ+15] J. P, S. L, Z. Q, Z. H, S. X, X. J. „A novel sequential exploration-exploitation sampling strategy for globalmetamodeling“. In: *IFAC-PapersOnLine* 48.28 (2015), S. 532–537 (zitiert auf S. 11).

[PR19] W. P, E. R. „Adaptive Sampling For Building Simulation Surrogate Model Derivation Using The LOLA-Voronoi Algorithm“. In: *International Journal of Information Management* (2019) (zitiert auf S. 16).

[PYX+21] R. P, X. Y, C. X, H. P-Y, L. Z, G. B, C. X, W. X. „A Survey of Deep Active Learning“. In: *International Journal of Information Management* (2021) (zitiert auf S. 5).

[RH06] G. RB, L. HK. „Adaptive design of supercomputer experiments“. In: *International Journal of Information Management* (2006) (zitiert auf S. 10).

[RJR07] B. R, R. JJ, K. RD. „Active learning for regression based on query by committee“. In: *International Conference on Intelligent Data Engineering and Automated Learning* (2007), S. 209–218 (zitiert auf S. 11).

[RWA02] J. R, C. W, S. A. „On sequential sampling for global metamodeling in engineering design“. In: *ASME 2002 International design engineering technical Conferences and Computers and information in engineering conference, Montreal, Canada* (2002), S. 539–548 (zitiert auf S. 8, 12, 17).

[TCMK13] M. T, A. C, G. M, B. K. „Comparison of adaptive sampling methods for generation of surrogate aerodynamic models“. In: *AIAA* 51.4 (2013), S. 797–808 (zitiert auf S. 13).

[VKO+13] A. V, S. K, A. O, A. S, R. R. „Crossvalidation based single response adaptive design of experiments for kriging metamodeling of deterministic computer simulations“. In: *Struct Multidiscip Optim* 48.3 (2013), S. 581–605 (zitiert auf S. 8, 12, 15, 17, 21, 22).

[XYL12] W. X, W. Y-Z, C. L-P. „A new sequential optimal sampling method for radial basis functions“. In: *International Journal of Information Management* (2012) (zitiert auf S. 13).

[YFJ+04] L. Y, M. F, A. JK, T. K-L, C. VC. „A sequential exploratory experimental design method: development of appropriate empirical models in design“. In: *ASME 2004 International design engineering technical Conferences and Computers and information in engineering conference, Salt Lake City, Utah, USA* (2004), S. 1021–1025 (zitiert auf S. 13).

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift