Information Driven Exploration in Robotics

Von der Fakultät 5: Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von
Johannes Kulick
aus Hildesheim

Hauptberichter: Prof. Dr. Marc Toussaint
Mitberichter: Prof. Dr. Miriam Mehl

Tag der mündlichen Prüfung: 24.08.2016

Institut für Parallele und Verteilte Systeme der Universität Stuttgart
2016

Für Hartmut.

Summary

Imagine an intelligent robot entering an unknown room. It starts interacting with its new surroundings to understand what properties the new objects have and how they interact with each other. Finally, he gathered enough information to skillfully perform various tasks in the new environment. This is the vision behind our research towards intelligent robots. An important role in the described behavior is the ability to chose actions in order to learn new things. This ability we call exploration. It enables the robot to quickly learn about the properties of the objects. Surprisingly autonomous exploration has been mostly neglected by robotics research so far, because many fundamental problems like motor control and perception were still not satisfactory solved. The developments of recent years have, however, overcome this hurdle. State of the art methods enable us now to conduct research on exploration in robotics.

On the other hand the machine learning and statistics community has developed methods and the theoretical background to lead learning algorithms to the most promising data. Under the terms *active learning* and *experimental design* many methods have been developed to improve the learning rate with fewer training data.

In this thesis we combine results from both fields to develop a framework of exploration in robotics. We base our framework on the notion of information and information gain, developed in the field of information theory. And although we show that optimal exploration is a computational hard problem, we develop efficient exploration strategies using information gain as measure and Bayesian experimental design as foundation.

To test the explorative behavior generated by our strategies we introduce

the *Physical Exploration Challenge*. It formalizes the desired behavior as exploration of *external* degrees of freedom. External degrees of freedom are those the robot can not articulate directly but only by interacting with the environment. We present how we can model different exploration tasks of external degree of freedom: Exploring the meaning of geometric symbols by moving objects, exploring the existence of joints and their properties, and exploring how different joints in the environment are interdependent. Different robots show these exploration tasks in both simulated and real world experiments.

Zusammenfassung

Wie würde sich ein intelligenter Roboter verhalten, der einen ihm unbekannten Raum betritt? Vermutlich würde er anfangen all die Dinge um sich herum zu untersuchen, um sich ein Bild darüber zu verschaffen, welche Eigenschaften die Objekte ausmachen und wie sie miteinander zusammenhängen. Dieses Wissen würde es ihm dann ermöglichen verschiedenste Aufgaben in der neuen Umgebung zu erledigen. Eine zentrale Rolle bei diesem Verhalten spielt die Fähigkeit eigenständig zu entschieden, was es zu untersuchen gilt. Diese Fähigkeit nennt man Exploration. Erstaunlicherweise wurde autonome Exploration bisher in der Robotik vernachlässigt. Der Grund liegt darin, dass grundlegendere Fähigkeiten, wie zum Beispiel die Erzeugung von Bewegung oder die Wahrnehmung, die Wissenschaft bisher vor große Probleme stellten. Die Entwickelungen der letzten Jahre in diesen Bereichen ermöglichen uns aber nun Exploration der Umwelt mit Robotern zu untersuchen.

Auf der anderen Seite wurden im Bereich des Maschinellen Lernens und der Statistik Methoden und theoretische Grundlagen entwickelt, die Lernalgorithmen in die Lage versetzen, selber ihre Trainingdaten zu sammeln. Dadurch kann die Lernrate mit möglichst wenig Trainingsdaten verbessert werden. Diese Methoden werden unter den Begriffen *Aktives Lernen* und *Experimentelles Design* zusammengefasst.

In dieser Arbeit kombinieren wir die Resultate aus den beiden vorge-

nannten Feldern. Wir entwickeln damit die Grundlagen für autonome Exploration in der Robotik. Wir leiten diese Grundlagen von der Informationstheorie ab, die eine formale Definition von den Größen Information und Informationsgewinn entwickelt hat. Und obwohl wir zeigen, dass optimale Exploration nicht effizient berechenbar ist, können wir basierend auf dem Informationsgewinn Heuristiken entwicklen, die zu effizienten Explorationsstrategien führen.

Um das Explorationsverhalten, dass sich aus diesen Strategien entwickelt, zu testen, führen wir die *Physical Exploration Challenge* ein, das Problem der physikalischen Exploration. Es formalisiert unsere Vision eines intelligenten, explorierenden Roboters als Problem der Exploration von *externen* Freiheitsgraden. Externe Freiheitsgrade sind solche, die der Roboter nicht direkt beeinflussen kann, sondern nur durch Interaktion mit der Umwelt. Schlussendlich modellieren wir verschiedene Explorationsaufgaben von externen Freiheitsgraden und zeigen mit verschiedenen Robotern, simulierten wie auch echten, wie diese Aufgaben gelöst werden können. Die Aufgaben umfassen dabei das Erkunden der Bedeutung von Symbolen, die geometrische Zusammenhänge widerspiegeln, die Exploration von Existenz und Eigenschaften von Gelenken in der Umwelt und wie die Stellung von Gelenken entscheidend für die Beweglichkeit andere Gelenke sein kann.

Acknowledgements

I am grateful for the support of several people and organizations

I am most thankful for Marc. I can not imagine a better supervisor.

I would like to thank my co-authors Stefan, Tobias, Manuel, Robert and Oliver for doing research with me. I have had much fun finding out new things with you.

I am also very happy to have had such a nice group to work in. Thanks to all the other MLR-lers: Stanimir, Nikolay, Dmitry, Andrea, Peter, Vien, Matt, Marion and Carola. Special thanks go to Stefan for all the hours spent together upon various robots or papers.

I did not have one group, but two. Thanks to Oliver and the whole RBO team for their hospitality, especially Rico, Sebastian, Roberto and Manuel.

I want to thank Sebastian, Robert, Andrea, Matt, Peter, and Rico for proof-reading my manuscript, and Alice Auersperg for providing the nice picture of the cockatoo opening the lockbox on page 21.

I would like to thank my friends and family. You made the time away from robots and formulas great. Especially I would like to thank my mother, Gabi, and finally the two most important people for me, Emil and Franzi.

My research was funded by the German Science Foundation (DFG) under the grant numbers TO-409/9-1 and TO-409/7-1, both within the priority program *Autonomous Learning*.

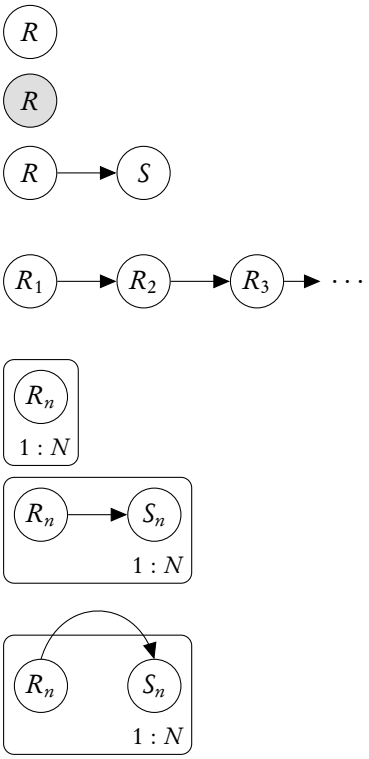
Contents

List of Figures

List of Tables

Graphical Model Notation

This notation resembles the notation of Bayesian networks with some additional syntax. For an introduction to this notation see Barber (2012).



A random variable R, R is latent.

A random variable R, R is observable.

Two random variables R and S, S depends on R.

A dynamic Bayesian network, defining a chain of random variables $R_1, R_2, R_3, \ldots$, each dependent on its predecessor.

A plate, defining $N - 1$ random variables $R_1, \ldots, R_N$.

A plate defining that each random variable S_i is dependent on the random variables R_i.

A plate defining that each random variable S_i is dependent on all random variables $R_1, \ldots, R_N$.

1 Introduction

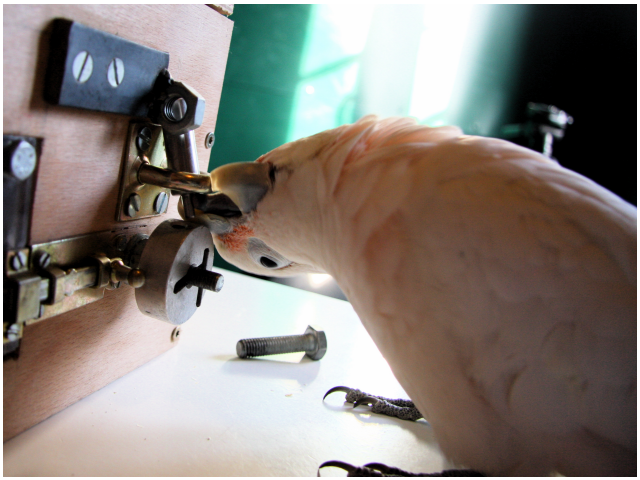


Figure 1.1: Cockatoo 'Muppet' is confronted with a complex locking mechanism and achieves success by exploring it. Picture courtesy of Alice Auersperg.

A cockatoo is confronted with a complicated box, locked by several mechanisms blocking the way to a small amount of grain behind a translucent door, as shown in Fig. 1.1. The bird sees the food and tries to reach it, it picks the door, then starts picking the various mechanism. Its first attempts are unsuccessful, but after a while it is able to unlock the first mechanism using its claws and sprout. Slowly it progresses by trying out the other mechanisms, exploring each until it understands how to

> unlock it. Slowly, one after one, they open, until eventually the cockatoo can reach the grain behind the door. Happily, it receives its reward.

This scene is an excerpt from a video of an experiment conducted by Auersperg et al. (2013). Watching the bird instantly makes the observer think how intelligent these animals must be to be able to solve such a complicated task. What are the viewers astonished about? It is most likely not the ability of the cockatoo to see the food, and not the ability to open the complex mechanisms with its claws. Such intuitive abilities are deemed easy, although Moravec's paradox tells us that those things are surprisingly hard to achieve (Moravec 1988). The viewer is astonished about the cockatoo's ability to *explore* the mechanisms, and the ability to *learn* how to solve the puzzle.

Exploration and learning are two important building blocks of many intelligent systems. Exploration is the ability to actively gather information about new situations. As such it crucially needs learning. Learning describes any method to improve the own abilities. Without it exploration would be useless, because the gathered experiences would not affect the behavior of an exploring agent. But augmented with learning, exploration can be a powerful mechanism to guide the actions of an agent. Through exploration agents can gather information to cut off branches of possibilities to find out the truth. Learning tells them how to incorporate that information to become better.

The cockatoo also learns to improve its ability to open the lockbox in the long run: In repeated experiments the bird opens the door much faster, even if the mechanisms are shuffled (i.e., the locks are in different order). It has learned how to operate the mechanism and the execution is not a problem anymore. It seems like it has built a mental model of the mechanism and only needs to navigate through that model to unlock the mechanism again. In this thesis we will primarily focus on exploration for model-based learning. Building a precise model that predicts how an environment reacts is a way to transfer experiences to new situations. To gather such experience we could rely on randomly chosen actions, or simply observe the environment,

but actively gathering them can help us to understand the dynamics of an environment faster as well as getting a more precise understanding.

Summarizing, exploration is a key ability to deal with new situations and thus a key ability for intelligent behavior. Without exploration strategies, finding new solutions would be notoriously hard, new situations would overburden us, and any of our greater achievements as human race would not have been possible. To be clear, the ability to explore is not sufficient to build intelligent behavior, but it is a necessary means.

When the new scientific field of artificial intelligence emerged during the second half of the last century, the goal was to build artifacts that resemble the human intelligence and autonomy at least partly. If we still want to achieve this goal, there is no question the systems we build will need ways to explore their environment. And while we are still far away from human-like intelligence, the experiment of Auersperg et al. (2013) shows us that also other systems, which we acknowledge to be intelligent, are able to autonomously explore their environment. In this thesis we will develop methods that equip robots with this ability.

1.1 Active Learning as General Principle of Data Gathering

For any learning process we need data to train the algorithm with. These data sets can have different forms. In supervised classification, for example, they consist of input data and labels of a classes these data samples belong to. In reinforcement learning they are typically trajectories through the space of possible states augmented with reward signals that encode how much value each of these states has. And in clustering tasks there are only the input data samples, from which the structure of the data distribution should be learned. From these data sets various learning algorithms try to improve the solution of the given task: Learning a classifier tries to improve the prediction of a class label for new data samples, a reinforcement learner seeks to find a better policy to act in a given scenario and a clustering algorithm tries to segment the data set.

Many of these learning algorithms rely on large, pre-sampled data sets. Exploration on the other hand actively gathers data about the environment. It is the process of learning while choosing data for the training data set. This method of generating data sets is also often called active learning (Settles 2012). The principle behind active learning is very general: The learning agent is not presented with training data, but it is equipped with a method to choose the data it needs for learning by itself. This method is two-fold: first it needs means to create samples or must be provided with access to already generated samples and second a way to measure how interesting a sample is. Based on this measure it can choose the best fitting sample for the learning task.

We will use this principle to build exploration strategies for robots. A major difference to active learning approaches in the literature is that when exploring real environments, a robot needs to manipulate objects in the real world by means of its embodiment. It thus needs to interact with objects. Active learning methods in the machine learning literature, in contrast, sample purely within software.

Reasoning about configurations of an environment also introduces the problem of physical reasoning. In the robotic context a data sample resembles a physical configuration of an environment, e.g., it consists of positions and orientations of all objects in the scene. When sampling data points from such distributions, they must obey physical laws. If not carefully constructed some samples might be infeasible: Objects might float in the air, penetrate each other, or otherwise disobey physics. This resembles the problem of Lang and Baum (1992). In a handwritten digit recognition task samples generated by their active learning method could not reasonable be classified, because they did not resemble any digit. We will show how we can generate data samples that are physically feasible or in other settings how to avoid the problem entirely by finding useful parametrizations.

Summarizing, active learning in its generality is powerful and has proven to outperform passive methods (Tomanek and Olsson 2009). It is therefore a good candidate for a framework of exploration and we will use it as basis throughout this thesis.

1.2 Shannon Entropy as Formalized Driving Force

With active learning as the general formulation of exploration, the question remains, what criterion the agent should follow, when actively choosing new data points.

When learning a model of the environment, we want each new data point to reduce the uncertainty over the model we have so far. Reducing the uncertainty most would improve our model most. To represent the state of the environment together with our uncertainty we rely on the Bayesian interpretation of probabilities, where uncertainty is represented as probability distribution. We call distributions that describe our own belief over the state of the world, including the uncertainty of it, belief distributions (Kaelbling, Littman, and Cassandra 1998).

When measuring the uncertainty over the belief distribution the natural measure is the Shannon entropy (Shannon 1948). It captures how much information a new sample of a distribution contains in expectation. Thus if it is low, we can predict a new sample better, or from a different viewpoint we have more information about the underlying distribution. By reducing the uncertainty over our belief of the world we improved our ability to predict future samples of the environment. We have learned to predict better.

As such exploration is gathering the most information with the fewest exploration steps. And since Shannon entropy captures all this in a well developed framework, we will use it as the main criterion to drive our exploration.

We will however see, that we can not directly optimize that criterion because of computational barriers. We thus develop heuristics that indirectly minimize the entropy of the distribution we are interested in. Those are still based on information theoretic measures but will work step wise, so keeping the computational effort at an affordable level.

1.3 Thesis Outline

This thesis is structured as follows: The next section is about the theory of active learning. It starts with a review of the active learning and exploration

literature (Sec. 2.1). Then we recall results from Wolpert (1996) that there is no algorithm that can perform better in active learning than random exploration (Sec. 2.2), but only if we do not have any assumption about the distribution to learn. Following that, we show that with priors there is a Bayes optimal solution (Sec. 2.4), but that it is still too expensive to compute. Chapter 3 develops and analyzes a maximum cross entropy strategy for exploration, that is well suited for situations where strong priors are in place but should not mislead the exploration.

The following chapters are about exploration in robotic environments. We first give background on related robotics research (Sec. 4.1). We then define the physical exploration challenge, which is the formalization of the desired robotic behavior (Sec. 4.2). Following, the rigid world assumption (Sec. 4.2.2) limits the set of possible scenarios to a reasonable size, by restricting the environments to consist only of rigid bodies connected by joints.

After this chapter on background in robotics we present various experiments on exploration in robotics. We show a robotic exploration task for learning a symbolic representation, that is in turn used to plan in a reinforcement learning task (Sec. 5). These experiments already contain the full exploration stack, but in a somewhat limited scenario. The scenario is widened by including the most important degrees of freedom in rigid worlds: joints. We show how to explore environments to discover new joints, their parameters, and finally their dependency structure (Sec. 6).

The thesis closes with a analysis of the results and discusses its limitations and the future work they imply (Sec. 7).

1.4 Published Parts of the Thesis

This thesis is based on already published publications. It enhances these publications by broader theoretical background, deeper analysis and wider discussion. The publications are:

- J. Kulick, M. Toussaint, et al. (2013). "Active learning for teaching a robot grounded relational symbols". In: *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 1451–1457

- S. Otte et al. (2014). "Entropy Based Strategies for Physical Exploration of the Environment's Degrees of Freedom". In: *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 615–622
- J. Kulick, R. Lieck, et al. (2015). "The Advantage of Cross Entropy over Entropy in Iterative Information Gathering". In: *arXiv e-prints* 1409.7552v2 (stat.ML)
- J. Kulick et al. (2015a). "Active Exploration of Joint Dependency Structures". In: *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pp. 2598–2604
- J. Kulick et al. (2015b). "Robots Solving Serial Means-Means-End Problems". In: *RSS Workshop on Combining AI Reasoning and Cognitive Science with Robotics*

2 Theory on Active Learning

A learning system consists of two major parts: A learning algorithm, which improves the current behavior, and data. Every learning algorithm needs data to be trained on, the so called training set.

In classical machine learning the training data of the algorithm is chosen not by the algorithm itself, but by an independent source. This can include hand chosen data sets by human experimenters or randomly sampled sets. While the learning algorithms are able to improve with more data, they are not the driving force in generating the training data.

Active learning, in contrast, is a concept that gives the learning agent the control over the data set used for training (Settles 2012). It is also known by the name *experimental design*. To choose training data the agent needs a notion of informativeness of the data points. This can, for instance, be driven by a measure of novelty of new data points, or how likely new data would surprise the agent. The way how an agent learns from new data might also influence how it measures the relevance. Knowing the algorithm that is used to learn can affect the choice of training data immensely. But also the knowledge of a particular task might influence the notion of importance of data points.

Another important aspect of active learning is that generating training data might be expensive. Although in the time of Big Data large data sets are available more easily, there are still areas where acquiring data is not easy. Running a whole experiment, for instance, with a complicated apparatus in physics or a genetic experiment over a few generations of a species, is still expensive. Automating the process of choosing useful parameter sets for experiments is the task of experimental design, as coined by the statistical research community (Fedorov 1972). But also when considering an exploring robot, a new sample consists of moving somewhere, inspecting objects, and

interacting with them, which takes time. Reducing these costs is a major benefit of using active learning algorithms to generate data sets, as only the most important data points need to be classified or evaluated.

Active learning has proven to be beneficial for learning complex tasks (Tomanek and Hahn 2009). It was, however, used mostly in the area of machine learning to learn parameters of predictive systems. Using it to generate explorative behavior on robots is not yet widely applied. We leveraged the general active learning idea to enable robots explore their environment. We need to develop a good notion on what explorative behavior should optimize. After reviewing related work in the field of active learning we will argue that strategies based on the theory of information as introduced by Shannon (1948) are well suited. Theory of information is a well developed field and many active learning algorithms are based on it already.

After discussing the related work we will derive the optimal solution to the active learning problem. This derivation will give an understanding of the computational difficulties that arise when attempting to implement it.

Because of these computational barriers, various heuristics have been developed to find good active learning strategies. We will analyze a collection of them and discuss their strengths and weaknesses for generating explorative behaviors. Namely, they will be classical Bayesian experimental design, uncertainty sampling, and Query-by-committee (QBC), each of which is an example of a different category of active learners. From that analysis we develop a novel method to generate behavior that overcomes the main weaknesses for the use case of robotic agents exploring their environment. Its major benefit will be that it tries to challenge its current belief, until it has enough evidence to be certain about it. Informally speaking, it resembles the scientific method, trying to find experiments that possibly falsify its own hypothesis, instead of searching only for supporting evidence.

We will give experimental evidence that this novel strategy performs better in inference tasks, i.e., where information is gathered to infer the distribution of latent variables of the model, whereas it is inferior when information is gathered for prediction tasks, i.e., predicting the outcome of further experiments on the same variable. We will end the chapter with an analysis of the reasons for this dual outcome.

2.1 Related Work in Active Learning

Settles (2010, 2012) gives a broad overview over the area of active learning. He categorizes the algorithms in two different ways. The first distinction is how new samples can be acquired by the algorithm. He distinguishes between three variants: *Query Synthesis*, *Stream-based Sampling*, and *Pool-based Sampling*. The other distinction is how the active learning algorithms weight the importance of the different samples. The three major approaches are *Uncertainty Sampling*, *Version Space* methods and *Optimal Experimental Design*.

2.1.1 Sample Acquisition

Query synthesis might seem the most intuitive way to generate samples: the agent can query the whole input space and thus synthesize a new sample on its own and ask for its label or evaluation. Some of the earliest attempts to active learning were of this kind (Angluin 1988; Cohn, Ghahramani, et al. 1996). It is, however, often the case that defining the input space is not a trivial task. If we apply too wide boundaries to it, some samples may have no meaning or generate undefined behavior. This was the case, for example, with actively learning handwritten characters (Lang and Baum 1992), where a human oracle was not able to identify any actual character in the generated queries. The same difficulties arise with physical configurations of objects supposed to be build by a robot that does not obey the rules of physics, e.g., objects floating in free space. Those desired samples are not possible to generate, whereas query synthesis assumes that all samples in the sample space are feasible. To use query synthesis the sample space must be sufficiently small to include only feasible samples, where it should not exclude any useful sample. Defining the sample space sufficiently might not be trivial or not even possible. For the particular problem of defining physically feasible samples we will explain solutions in section 4.3.

To avoid this problem stream-based and pool-based sampling are often used. Stream-based sampling generates a continuous stream of samples of the actual distribution of data, which is assumed to be cheap and the active

learner only has to decide whether to ask for the label or evaluation, respectively, of the current sample (Cohn, Atlas, et al. 1994). Pool-based sampling generates a large pool of samples from the distribution, which is again assumed to be free or inexpensive. The active learner now chooses the most promising sample from that pool to get a label or evaluation. Pool-based sampling is often applied throughout many fields of machine learning. Settles (2012, p. 9) gives many examples.

Algorithm 1 shows an overview of the three different methods of acquisition. While in theory these sample acquisition strategies differ, in practice the problem of infeasible or meaningless samples often still persists. Only if a black box sampler of the real distribution is available for low cost, stream- or pool-based sampling offer benefits. If the labeling process is the expensive part and querying the input space is cheap, those techniques are valuable. If sampling from the real distribution is, however, expensive, those techniques offer little benefit.

2.1.2 Uncertainty sampling

The second distinction between active learning algorithms according to Settle is the way to choose samples. Uncertainty sampling, as coined by Lewis and Catlett (1994), samples in areas where the learner is uncertain about its own prediction accuracy. Uncertainty can be measured differently. An easy example of measuring uncertainty is the following: Consider a linear binary classifier. Now measure the distance of each sample to the decision boundary. The sample closest to the decision boundary is the most uncertain one. For a probabilistic binary classification model, this translates to the sample whose class probability is closest to 0.5 (Lewis and Gale 1994). For multiclass classification problems, Culotta and McCallum (2005) expanded this idea to sampling the least confident sample, which they defined to be the sample with the lowest probability for the most likely class:

$$x^* = \operatorname*{argmin}_x p(\hat{y} \mid x) \tag{2.1}$$

where $\hat{y} = \operatorname{argmax}_y p(y \mid x)$ is the most likely class. Another variant is to measure the margin between the most likely and the second most likely

Algorithm 1 Three different approaches to sample acquisition in active learning

function QuerySynthesisActLearning(ImportanceMeasure)
  $D \leftarrow \emptyset$
  repeat
    $x^* \leftarrow \text{argmax}_{x \in \mathcal{D}}$ ImportanceMeasure(x)
    $y^* \leftarrow$ get label for x^* from orcale
    $D \leftarrow D \cup \{(x^*, y^*)\}$
    train learner on D
  until some convergence criterion
end function

function PoolBasedActiveLearning(ImportanceMeasure)
  $X \leftarrow \emptyset$
  repeat
    $x \leftarrow$ draw sample from $\mathcal{D}$
    $X \leftarrow X \cup \{x\}$
  until sufficient number of samples drawn
  $D \leftarrow \emptyset$
  repeat
    $x^* \leftarrow \text{argmax}_{x \in X}$ ImportanceMeasure(x)
    $y^* \leftarrow$ get label for x^* from orcale
    $D \leftarrow D \cup \{(x^*, y^*)\}$
    $X \leftarrow X \setminus x$
    train learner on D
  until some convergence criterion
end function

Algorithm 2 Active learning sample acquisition (continued)

```
function StreamBasedActiveLearning(ImportanceMeasure,
SampleCriterion)
    D ← ∅
    repeat
        x̃ ← draw sample from 𝒟
        if SampleCriterion(ImportanceMeasure(x̃)) then
            x* ← x̃
            y* ← get label for x* from orcale
            D ← D ∪ {(x*, y*)}
            train learner on D
        end if
    until some convergence criterion
end function
```

class, where large margins are considered less important, because the learner is already quite certain about the outcome (Scheffer et al. 2001). It, however, ignores again all other class probabilities.

A more general uncertainty measure is the entropy of a distribution (Shannon 1948), denoted as $H[p(x)]$, which measures the expected number of bits of information a random event contains:

$$x^* = \operatorname*{argmin}_x H[p(y \mid x)] \tag{2.2}$$

$$= \operatorname*{argmin}_x \left(-\sum_y p(y \mid x) \log p(y \mid x) \right). \tag{2.3}$$

Generally, entropy is a measurement of how uncertain we are about the outcome of a draw of a given distribution. For learning a prediction we want to be as certain as possible about the next outcome, and hence minimize the entropy.

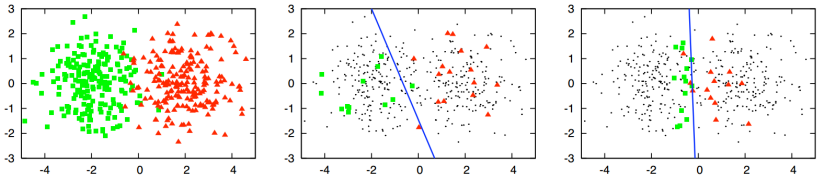


Figure 2.1: A toy example of pool-based uncertainty sampling. On the left there are all samples in the pool classified. In the center we have linear classifier trained on a random selection of samples (Accuracy 70%). On the right uncertainty sampling is used to choose the samples (Accuracy 90%). Figure from Settles (2010).

It is used throughout the literature of active learning as measure of uncertainty (e.g., Ratnaparkhi et al. (1994), Hwa (2004), Körner and Wrobel (2006), Settles and Craven (2008), and Kulick, Toussaint, et al. (2013)), and generally performs well. There are, however, examples showing that for particular problems it is worse than random sampling (Schütze et al. 2006; Tomanek and Hahn 2009; Wallace et al. 2010). In Sec. 2.2 we will show theoretical findings whether active learning methods are generally better than passive ones.

2.1.3 Version Space Methods

Another view on learning algorithms is the idea of version spaces proposed by Mitchell (1977, 1982) and its implications. The version space is defined as the set of hypotheses that conform with the training data. E.g., in a linear separable classification task, all linear separators that perfectly separate the training examples. Learning is now the reduction of the version space until only the true hypothesis remains.

Based on this view on learning Cohn, Atlas, et al. (1994) proposed an algorithm, which Settles (2012) calls Query-By-Disagreement (QBD). It is a stream-based active learning algorithm for binary classification that explicitly maintains the version space. Whenever two hypotheses of the version space disagree on the label of a new sample arriving, the true label is queried

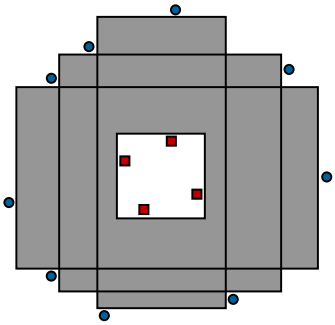


Figure 2.2: An example of Query-by-Disagreement. Each blue circle and each red square denotes a sample of one of two classes. Each hypothesis is a rectangular decision boundary between the two classes. If a new sample falls within the gray area not all hypotheses of the version space agree on the label and the true label is queried. If a sample falls outside the gray area all hypotheses agree on the label and the version space does not need to be adapted. Figure based on Cohn, Atlas, et al. (1994).

and the version space is adapted accordingly. If all hypotheses agree on the label the new sample does not carry new information and it is unnecessary to query the true label. An illustrative example of QBD is shown in Figure 2.2.

There are a few drawbacks of QBD. Maintaining the version space explicitly might be prohibitively expensive. With noisy labels or overlapping class areas there might be no hypothesis in the version space. And for non stream based situations it might be useful to measure the amount of disagreement between the hypotheses for each sample to decide which one to choose, especially if there are more than two classes to be labeled.

The first issue is tackled by an algorithm called Query-by-Committee, proposed by Freund, Seung, et al. (1997) and Seung et al. (1992). Query-by-committee does not maintain the whole version space but a finite set of committee members as representations of hypotheses. These members are learning algorithms that can be trained. Whenever they disagree on a label the true label is queried and the algorithms are retrained on the extended

training set. Thus the committee members represent hypotheses from the version space and are updated to fit in the version space after each query.

McCallum and Nigam (1998) extended this algorithm further to use it in pool-based active learning scenarios. They do not simply decide whether two committee members disagree or not, but measure the amount of disagreement. Thus they can use the algorithm for regression tasks as well as multiple class classification tasks. It is also trifling that all committee members conform to all training data, allowing for noisy data. Their measure of disagreement is the sum of all Kullback-Leibler (KL) divergences from each committee member's predictive distribution to the mean predictive distribution of all committee members:

$$\frac{1}{\| \theta \|} \sum_{\theta} D_{KL} \left[p(y \mid \theta, D, x) \;\middle\|\; \frac{\sum_{\theta} p(y \mid \theta, D, x)}{\| \theta \|} \right]. \tag{2.4}$$

This measures how different the prediction of each committee member $p(y \mid \theta, D, x)$ is to the mean prediction of all committee members together. The mean prediction assigns a uniform prior over committee members. It can easily be extended to a fully Bayesian version by incorporating a belief over committee members $p(\theta \mid D)$:

$$\sum_{\theta} p(\theta \mid D) \cdot D_{KL} \left[p(y \mid \theta, D, x) \;\middle\|\; \sum_{\theta} p(\theta \mid D) p(y \mid \theta, D, x) \right]. \tag{2.5}$$

Further versions of the algorithms were proposed by Abe and Hiroshi (1998), based on the ideas of boosting (Freund and Schapire 1997) and bagging (Breiman 1996).

2.1.4 Optimal Experimental Design

All the previously described methods give heuristics on which sample is the most informative for the learning algorithm. We are, however, interested in maximizing some measurable performance of the algorithm. Thus we

would like to know, how a given sample would change that performance if sampled. We would than sample at positions that increase our performance most. We would sample optimally with respect to our performance measure. Unfortunately the change of performance is dependent on the actual evaluation of the sample (i.e., its actual label or value). We therefore have to integrate over all possible outcomes for all possible hypotheses and use the expected utility of the sample. This method is called *Optimal Experimental Design* (Chaloner and Verdinelli 1995; Fedorov 1972) or, when we use a prior belief over the possible outcomes, *Bayesian Experimental Design*. The general formulation of Bayesian experimental design is

$$x^* = \operatorname*{argmax}_x \int_y \max_{d \in \mathcal{D}} \int_\Theta U(d, \theta, x, y) p(\theta \mid y, x) p(y \mid x) d\theta dy, \tag{2.6}$$

where U is the utility measure, dependent on a decision d taken based on the outcome y from sampling x with hypothesis θ (Chaloner and Verdinelli 1995). A decision is two-fold: the choice of the design of the experiment x and a final decision after the experiment, which can have different utilities as well. In practice this final decision is often ignored. With this formulation, Bayesian experimental design maximizes the expected utility of a sample w.r.t. the hypothesis and the outcome.

The most common utility function is the information gain, for example, how much information a distribution has gained by adding a sample. Information gain is defined as the Kullback-Leiber divergence (Kullback and Leibler 1951) between the posterior and the prior distribution:

$$U(d, \theta, x, y) = \mathrm{D}_{\mathrm{KL}} \left[p(\theta \mid y, x) \parallel p(\theta) \right] \tag{2.7}$$

$$= \int p(\theta \mid y, x) \log \frac{p(\theta \mid y, x)}{p(\theta)} d\theta dy. \tag{2.8}$$

Experimental design is mainly investigated within the field of statistics, but the machine learning community has had similar ideas. Roy and McCallum (2001) proposed an optimal active learning algorithm, that optimizes

the expected loss function of the learner. To make the computation of the expectation tractable they compute a Monte Carlo estimate.

2.1.5 Exploration in Reinforcement Learning

Another track of research that has to deal with information gathering is reinforcement learning (Kaelbling, Littman, and Moore 1996; Sutton and Barto 1998). Put general, reinforcement learning are methods that learn a strategy from interaction with the environment through trial-and-error. The value of a state is rated by a reward function, which might be arbitrarily shaped. It does not directly evaluate an action, but only the resulting state. An agent hence needs to assign the reward to the correct action or action sequence on its own.

Typically each state (or each state-action pair) is assigned a value computed from the rewards gathered in the experiences. If the value function captures the reward function sufficiently, an agent can greedily follow the values of reachable states to maximize its reward. However, the value function is not known, but the agent needs to infer it.

To gather experiences an exploration policy is needed. Often ad-hoc methods are used to specify the exploration strategy. Examples are ε-greedy, that performs random actions in a fraction of the actions (defined by the parameter ε, hence the name) or Boltzmann exploration, which uses the expected value to perform actions (Kaelbling, Littman, and Moore 1996).

Other strategies use the paradigm of *optimism in face of uncertainty*, i.e., they are more likely to perform actions about whose outcome they are uncertain. Examples are exploration bonus algorithms (Kolter and Ng 2009; Sutton 1990) and R_{max} (Brafman and Tennenholtz 2002).

Explicit Explore and Exploit (E^3) from Kearns and Singh (2002) explicitly computes policies for exploration of new states, and exploitation of the value function with a fixed horizon. Both policies alone are optimal for their task, but combined the algorithm is only near-optimal with polynomial bounds, as is R_{max}. Based on E^3 and R_{max} Lang, Toussaint, and Kersting (2012) develop the REX (relational exploration) framework for exploration in relational reinforcement learning (see Sec. 5.2 for a short introduc-

tion to relational reinforcement learning).

These exploration strategies use the reinforcement learning formulation as Markov Decision Problem (MDP) and leverage this formulation to define an optimal solution. The actual derivation of the optimal solution is topic of Section 2.4.

Another idea for exploration in reinforcement learning is to use artificial curiosity as intrinsic motivation of the agent. A central driving force of exploration in animals and humans is their curiosity. The concept of artificial curiosity has been introduced to the reinforcement learning literature by Schmidhuber (1991a,b). It models the curiosity found in animals and humans as desire to build an exact world model, i.e., a model that can predict effects of actions in an environment. While we can not predict the outcome of an interaction with the environment, it is interesting to examine this interaction. It becomes boring when we understood the effect. The attention is then drawn to effects we do not understand yet.

2.2 A No-Free-Lunch Theorem for Active Learning

Given the various forms of active learning, it is interesting to ask whether any of the given algorithms generally performs better than the others. Wolpert and Macready (1995, 1997) showed that for *optimization* the so called No-free-lunch theorem holds, which states that any two optimization algorithms are equally good, if averaged over all possible functions. The task of optimization is to find the point at which a function has its highest value.

To formalize the No-free-lunch theorem we use a Bayesian formulation of optimization and learning, based on the extended Bayesian formalism of Wolpert (1996, 2001) and Wolpert and Macready (1995, 1997). Let a_1, a_2, f, m and D be random variables. a_1 and a_2 define the query algorithms to choose the x-values of the training set $D = \{(x_1, y_1), \ldots, (x_m, y_m)\}$. f is the target function, and for the training set $y_i = f(x_i)$. m is the size of the training set. We write D_m^x and D_m^y to name the sets of x and y values of the training set respectively. With this formalism the No-free-lunch theorem for optimization reads:

Theorem 1 (No-free-lunch for optimization).

$$\forall a_1, a_2, m, D_m^y : \sum_f p(D_m^y \mid f, m, a_1) = \sum_f p(D_m^y \mid f, m, a_2)$$

The proof is given by Wolpert and Macready (1997). The intuition behind the proof is that when summarizing over all functions, every function value is equally likely, because we do not assume any shape of a function to be more likely. Thus every set of function values is always equally likely regardless of the inputs. Hence, it is independent of the optimization algorithm. From this theorem it directly follows that for any performance measure $\Phi(D_m^y)$, only dependent on the data D_m^y, the performance of any two query algorithms a_1, a_2 is the same if averaged over all f:

$$\begin{aligned} &\forall a_1, a_2, m, \Phi(\cdot), D_m^y : \\ &\sum_f p(\Phi(D_m^y) \mid f, m, a_1) = \sum_f p(\Phi(D_m^y) \mid f, m, a_2) \end{aligned} \tag{2.9}$$

When considering learning, one is typically not interested in optimizing the function that is learned, but in the performance when predicting further values of the given function. Are there learning algorithms whose prediction performance is generally better than others? To analyze this, we need a measure of the prediction performance of a learning algorithm. This performance can be measured by means of a loss functions $c = L(f(x), l(x))$ that measures how well a prediction of a learner $l(x)$ covers the actual value of the function $f(x)$. Typical examples of loss functions include quadratic loss and zero-one loss functions. These loss functions are, however, not only dependent on D_m^y but also on f, thus the No-free-lunch theorem of optimization does not directly apply.

But for the set of *homogeneous* loss functions Wolpert (1996, 2001) showed that any two learning algorithms have the same performance if averaged over all supervised learning problems. A loss function is called homogeneous if $\sum_{f(x)} \delta[c, L(f(x), l(x))]$ (with δ being the Kronecker delta) is independent on $l(x)$. Loosely speaking this ensures that if we have no information about

$f(x)$ a priori all hypotheses are equally good. Quadratic loss and zero-one loss are again examples of loss functions that are homogeneous.

Theorem 2 (No-free-lunch for supervised learning).

$$\forall l_1, l_2, m, c : \sum_f p(c \mid f, m, l_1) = \sum_f p(c \mid f, m, l_2)$$

The proof is given by Wolpert (1996). A not so widely known corollary of this proof is that the cost of a learning algorithm is not only independent of the learner but also from the dataset used to train the algorithm, as long as the query algorithm is independent of all function values it has not queried yet (see Wolpert 1996, Sec. 5.4). Thus we can formulate a No-free-lunch theorem for active learning.

Theorem 3 (No-free-lunch for active learning).

$$\forall a_1, a_2, l_1, l_2, m, c, : \sum_f p(c \mid f, m, a_1, l_1) = \sum_f p(c \mid f, m, a_2, l_2)$$

The proof is again given by Wolpert (1996). This states, that any two active learners (i.e., a combination of an algorithm a that queries a training set and a learning algorithm l that learns the target function based on this training set) performs equally well if averaged over all learning problems.

While this result seems discouraging at the first sight, it only states that averaged over *all learning problems* there is no possible gain for an algorithm. Naturally many of the possible learning problems are not interesting for us, as they are problems without any structure to exploit. Usually we are not interested in learning such functions, but want to learn functions that have structure to some extend.

While the No-free-lunch theorems have been sharpened by Schumacher et al. (2001) to apply to sets of functions if and only if the set is closed under permutation and further to non-uniform distributions over f by Igel and Toussaint (2005), they also showed that for the large subset of input spaces with a non-trivial neighborhood relation (i.e., not all or no points are neighbors) the no-free-lunch theorems do not apply.

Thus we should consider the outcome of the theorem not as a negative one, but as a positive one: For function spaces with sufficient structure it is possible to perform better than random guesses. Or taken differently: Given a structured prior belief over the problem space we can formulate an optimal solution. Section 2.4 will cover this optimal solution.

2.3 Active Learning Problem Definition

As shown in the overview of the current state of the field (Sec. 2.1), there is no single problem statement for the problem of learning actively in an optimal fashion. There are even results suggesting, that active learning is on par with all other sampling strategies (see Sec. 2.2), but this is only true for sets of learning problems that have no exploitable structure. For real world problem sets such a structure can be used for finding optimal strategies. In this chapter we will give a definition for active learning, which is quite general and which we will use throughout the theses. It is based on self-information (Shannon 1948).

Self-information is defined as the number of bits (i.e., the smallest pieces of information possible to transmit) needed to transmit the state of a random variable. It can be computed as

$$I(x) = -\log\left(p(x)\right). \tag{2.10}$$

One interpretation of which is the amount of information that is missing to accurately determine the state of the random variable. Consider the extreme case of one outcome having the probability of 1. No information is needed to determine the state of that random variable, since before drawing from the distribution it is already clear what result will occur. Hence, the information of such an event is 0.

If framed as a probabilistic model all information in the active learning framework can only be contained in the state of a random variable. Thus the self-information of the random variable holding the desired knowledge is a well suited criterion. We want to minimize the amount of information that is missing to know the state of the random variable. However, the self-information only covers the information that is contained in one sample. If

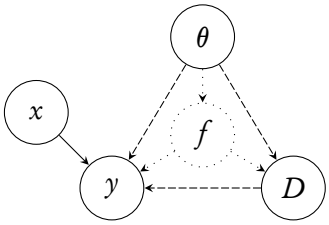


Figure 2.3: A generic generative model of data as Bayesian network. The already known data D as well as the new query x and label y are drawn from a distribution f. Dashed arcs describe the dependencies after marginalizing out f, dotted arcs describe the dependencies before that. θ subsumes hyperparameters or different models that can generate the data. Figure from Kulick, Lieck, et al. (2015).

we want to measure the information of a distribution, we need the expected information of a draw from that distribution, the entropy:

$$H[p(x)] = E[I(x)]_x \tag{2.11}$$

$$= -\int_x p(x)\log(p(x)), \tag{2.12}$$

This is the expected amount of information contained in a sample from the distribution (which is thought of as message, since Shannon worked on theory of communication). There is a distinction between the entropy of a discrete and a continuous distribution. The latter is called differential entropy. We will not make the distinction between the two whenever they are interchangeable and call both simply entropy. We will explicitly call them discrete and differential entropy when necessary.

We now define the necessary parts of the model, that are needed to derive the optimal solution.

Let f, x, y, θ and $D_N = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ be random variables. Their dependency structure is depicted in 2.3. (The notation of graphical

models follows the Bayesian network conventions. It is shortly summed up in the notation overview at the beginning of this thesis.) f is a data generating function, x and x_i are input data of the learner, y_i observed outputs and y a predicted output. Normally we marginalize out f, as we are not interested in the distribution of data generating functions. θ are potential hyper parameters or different model classes leading to different data generating functions.

We call $p(\theta \mid D, x, y)$ the *posterior* hypotheses belief after observing an additional data point (x, y). Accordingly, we call $p(\theta \mid D)$ the *prior* hypotheses belief, even though it is already conditioned on observed data. It does, however, play the role of a Bayesian prior in computing the posterior hypotheses belief. We call $p(y \mid \theta, D, x)$ the predictive distribution, as it is the distribution to predict further data from the underlying function. Typically, machine learning algorithms try to learn this distribution.

Within the probabilistic model, active learning can be framed as optimization problem of the expected information of y, given already observed data x and D_N. This resembles the typical desire to learn a predictive distribution, that can predict further samples of the data distribution:

$$(x_1^*, \ldots, x_N^*) = \underset{(x_1,\ldots,x_N)}{\operatorname{argmin}} E\left[-\log p(y \mid x, D_N)\right]_{x,y,D_N}. \tag{2.13}$$

Naturally, the formulation of active learning as minimizing the expected information can be reframed as minimizing the expected entropy of the distribution as follows, making the two notions of information for the case of active learning equivalent (note the subscripts of the expectation, which shows over which random variables the expectation is taken):

$$(x_1^*, \ldots, x_N^*) = \underset{(x_1,\ldots,x_N)}{\operatorname{argmin}} E\left[-\log p(y \mid x, D_N)\right]_{x,y,D_N} \tag{2.14}$$

$$= \underset{(x_1,\ldots,x_N)}{\operatorname{argmin}} E\left[-\int_y p(y \mid x, D_N) \log p(y \mid x, D_N)\right]_{x,D_N} \tag{2.15}$$

$$= \underset{(x_1,\ldots,x_N)}{\operatorname{argmin}} E\big[H\left[p(y \mid x, D_N)\right]\big]_{x,D_N}. \tag{2.16}$$

But learning the predictive distribution is only one view on active learning. So far we have marginalized θ, the hyperparameters of the model. But we might also be interested in other unobservable factors, that lead to the data. Thus we can also define active learning of the hyperparameters:

$$(x_1^*, \ldots, x_N^*) = \underset{(x_1,\ldots,x_N)}{\operatorname{argmin}} E\big[H\left[p(\theta \mid y, x, D_N)\right]\big]_{y,x,D_N}. \tag{2.17}$$

This is the view of experimental design, that is not necessarily interested in the actual value of an experiment, but in an underlying parameter of the model the experiment might unveil.

Throughout this thesis we will refer to the active learning problem as minimizing the expected entropy of any distribution, w.r.t. all observed data.

2.4 Bayes Optimal Solution to the Active Learning Problem

When considering the different approaches to active learning it is insightful to know that there exists a Bayes optimal solution to the problem that active learning tries to solve. We will outline the optimal solution and show that it is computationally intractable.

2.4.1 Multi-armed Bandits

To derive the optimal solution to the general active learning problem we start with a simple multi-armed bandit setting. Consider a gambling hall with n bandits, that return a reward when pulled. The task of an agent is now to repeatedly pull bandits to receive maximum reward. The agent, however, does not know what each bandit will return and each return itself may be in fact stochastic. Thus the agent can only maintain a belief over the return it will receive when pulling a bandit.

Each action has two effects: First, the agent gets a reward, i.e., the bandit returns money. But second, the agent gathers information about the bandit it played, and it is worthwhile to include this information in its model of the bandit, for it might help the agent to choose a better bandit to play in the next round.

Formally, this setting can be modeled as follows. Let $a_t \in \{1, \ldots, n\}$ be actions the agent can perform (i.e., machines it can play) and $r_t \in \mathbb{R}$ the rewards received at each time step. Find a policy

$$\pi : ((a_1, r_1), (a_2, r_2), \ldots, (a_{t-1}, r_{t-1})) \mapsto a_t \tag{2.18}$$

that maximizes[1]

$$\max E\left[\sum_{i=1}^{T} r_i\right]. \tag{2.19}$$

We now represent the agent's knowledge of the history

$$h_t = ((a_1, r_1), (a_2, r_2), \ldots, (a_{t-1}, r_{t-1})) \tag{2.20}$$

as belief over latent parameters of each machine $\theta = (\theta_1, \ldots, \theta_n)$ that determine the stochastic reward $p(r_i \mid \theta_i)$ each bandit returns. This belief is formulated as probability distribution $p(\theta \mid h_t)$.

The history about the action and rewards contain all information about the process. We call such a set of observations a *sufficient statistic*, if and only if no other set of observations contains more information about a parameters (Fisher 1922). A sufficient statistic is not necessarily unique. Thus we could exchange the history with another sufficient statistic of the bandits.

For a binary bandit scenario where each bandit returns a binary reward $r \in \{0, 1\}$ the belief distribution might for example be modeled as follows. As the n bandits are independent of each other the belief factorizes into

$$p(\theta \mid h_t) = \prod_{i=1}^{n} p(\theta_i \mid h_t). \tag{2.21}$$

[1] There are other possible objectives, like $\max r_T$, but for the sake of brevity we focus on one single objective.

Now each θ_i can be Beta distributed with α_i and β_i counting how often each bandit has returned a reward or not.

$$\theta_i \sim \text{Beta}(p_i \mid \alpha_i, \beta_i). \tag{2.22}$$

How can we build strategies to chose actions such that we gain as much reward as possible? We need to have a look into decision theory.

2.4.2 Markov Decision Processes

These kinds of sequential decision making problems are usually modeled as Markov decision process (MDP) as introduced by Bellman (1957a). The definition and introduction of MDPs given here follows Kaelbling, Littman, and Cassandra (1998).

An MDP is a tuple $(\mathcal{S}, \mathcal{A}, T, R)$, with

- $\mathcal{S}$ a set of states,
- $\mathcal{A}$ a set of actions,
- $T : \mathcal{S} \times \mathcal{A} \to \Pi(\mathcal{S})$ (with $\Pi(\cdot)$ being a probability space over a set) a state-transition function giving a probability distribution over states, which state is reached by performing an action in a state,
- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ a reward function, giving the immediate reward of an action in a state.

The joint distribution of an MDP is depicted in the graphical model shown in Fig. 2.4. As can be seen in the figure, the state at time t only depends on the state and action at time $t-1$ and not on any earlier state or action. Thus the following equation, called the *Markov property*, holds for the stochastic process:

$$p(s_t \mid s_{t-1}, a_{t-1}, \ldots, s_1, a_1) = p(s_t \mid s_{t-1}, a_{t-1}). \tag{2.23}$$

To find the optimal policy $\pi^*(s)$, i.e., the policy that maximizes the expected return, we compute the value of each state $V_{\pi,t}(s)$ as expected sum of future

rewards given the policy. If we have a finite time horizon T, this can be done by the following recursion:

$$V_{t-1}(s) = \max_{\pi} E\left[\sum_{i=t}^{T} R(s_i, \pi(s_i))\right] \tag{2.24}$$

$$= \max_{\pi}\left(R(s_t, \pi(s_t)) + E\left[\sum_{i=t+1}^{T} R(s_i, \pi(s_i)\right]\right) \tag{2.25}$$

$$= \max_{\pi}\left(R(s_t, \pi(s_t)) + \sum_{s'} p(s' \mid s_t, \pi(s_t))\, V_t(s')\right). \tag{2.26}$$

This is the well-known Bellman equation for the value function (see, for example, Bellman (1957a,b), Kaelbling, Littman, and Cassandra (1998), and Sutton and Barto (1998)). A policy that is greedy with respect to the value function is optimal and exists (Sutton and Barto 1998).

For an infinite horizon, i.e., T is infinite, the expected return may also be infinite. We can, however, use a discount factor $0 < \gamma < 1$, and define the value function as the expected discounted future return.[2]

$$V_{t-1}(s) = \max_{\pi} E\left[\sum_{i=t}^{\infty} \gamma^{i-t} R(s_i, \pi(s_i))\right] \tag{2.27}$$

$$= \max_{\pi}\left(R(s_t, \pi(s_t)) + \gamma \cdot E\left[\sum_{i=t+1}^{\infty} \gamma^{i-t-1} R(s_i, \pi(s_i))\right]\right) \tag{2.28}$$

$$= \max_{\pi}\left(R(s_t, \pi(s_t)) + \gamma \sum_{s'} p(s' \mid s_t, \pi(s_t))\, V_t(s')\right) \tag{2.29}$$

There exist various algorithms, such as policy iteration and value iteration (Sutton and Barto 1998), that can compute the optimal policy using

[2] We will cover only the finite time case for the further reading, although everything applies to the infinite time horizon case as well.

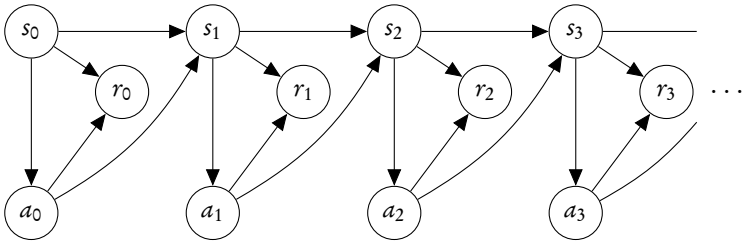


Figure 2.4: The joint distribution of a MDP depicted as DBN. As the state is Markovian the actions can depend only on the current state.

dynamic programming. And although they are quite efficient – only polynomial time in the number of states and actions – the state space alone might be prohibitively large, as we will see.

2.4.3 Partial Observability and Belief-MDPs

In the bandit scenario the state of the environment, θ, is not observable, i.e., an agent can not directly observe the value of each machine's parameter. Only the return of each slot machine is observable after it is played. While the actual state is still Markovian, the agent does not know it precisely. Such a process is called a partially observable Markov decision process (POMDP). The description of POMDPs also follows Kaelbling, Littman, and Cassandra (1998).

A POMDP consists of

- An MDP $(\mathcal{S}, \mathcal{A}, T, R)$,
- Ω, a finite set of observations and
- $O : \mathcal{S} \times \mathcal{A} \to \Pi(\Omega)$, an observation function, giving the probability distribution over observations given a state and action.

The joint distribution of a POMDP is depicted in Fig. 2.5. Instead of directly accessing the state, the agent in a POMDP only has access to *observations*, which depend on the state and the agent's action. These observations

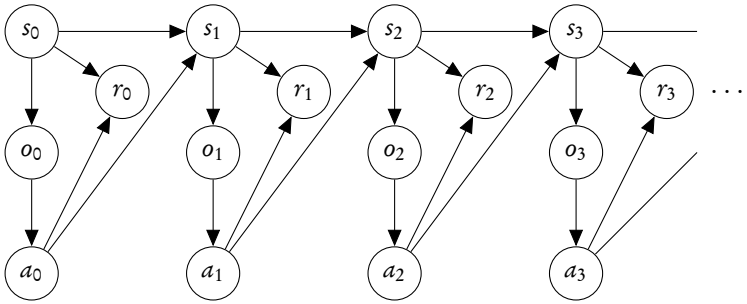


Figure 2.5: The joint distribution of a POMDP depicted as DBN. The policy in this example is naively generated without any memory of former observations.

are not guaranteed to follow the Markov property, given only the former observations. Thus relying only on the current observations might lead to sub-optimal policies. Different states, for example, might create the same observations, but need different actions for the optimal policy. Thus the agent needs to maintain a sufficient statistics, e.g., the history of observations.

Given the history of observations, the agent can build a belief $b_t \in \Pi(S)$ over the true state. This belief is formulated as probability distribution over the state space to account for the uncertainty over the true state in a typical Bayesian manner. To compute a belief for the next time step, we recursively compute the probability over states given the current belief b_t, the current observation o_t and the current action a_t:

$$p(s_{t+1} \mid b_t, o_t, a_t) = \frac{p(o_t \mid s_{t+1}, b_t, a_t)p(s_{t+1} \mid b_t, a_t)}{p(o_t \mid b_t, a_t)} \tag{2.30}$$

$$= \frac{p(o_t \mid s_{t+1}, b_t, a_t) \sum_{s_t \in S} p(s_{t+1} \mid s_t, b_t, a_t)p(s_t \mid b_t, a_t)}{p(o_t \mid b_t, a_t)} \tag{2.31}$$

All these quantities—except for the normalization factor in the denom-

inator, which only normalizes the distribution to sum up to one—are defined in the POMDP model. We call this distribution the state estimator SE of the POMDP.

Based on this belief we can build a so-called *belief MDP* (see Fig.2.6. This is a continuous state MDP over the belief distribution over the true state. It is defined as tuple $(\mathcal{B}, \mathcal{A}, \tau, \rho)$ with

- $\mathcal{B} \subseteq \Pi(\mathcal{S})$ the set of belief states, being probability distributions over states,
- $\mathcal{A}$, the same actions as in the POMDP,
- $\tau(b', a, b)$, the transition function, which is defined as

$$\tau(b', a, b) = p(b' \mid a, b) = \sum_{o \in O} p(o \mid a, b) p(b' \mid a, b, o) \tag{2.32}$$

- $\rho(b, a)$ the reward function defined as

$$\rho(b, a) = \sum_{s \in \mathcal{S}} p(s \mid a, b) R(s, a). \tag{2.33}$$

Note that b and b' are probability distributions and the transition function thus is a distribution over distributions. However, the distribution $p(b' \mid a, b, o)$ can be defined in terms of the state estimator of the POMDP as

$$p(b' \mid a, b, o) = \begin{cases} 1 \text{ if } SE(a, b, o) = b' \\ 0 \text{ else} \end{cases} \tag{2.34}$$

We now have again an MDP, where we can compute the value function. Since the history we base the belief MDP on is a sufficient statistic, a policy greedy w.r.t. $V_t(b)$ will be an optimal policy in the POMDP (Kaelbling, Littman, and Cassandra 1998). Using the belief state to gain a optimal policy for a POMDP is called planning in belief space or belief planning (Kaelbling

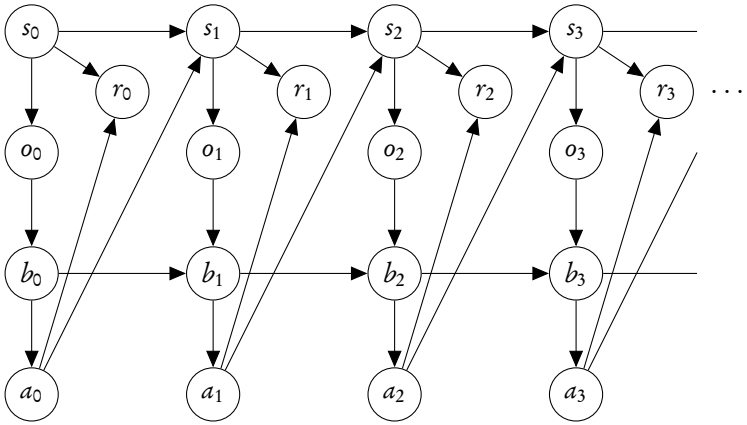


Figure 2.6: The joint distribution of a belief MDP depicted as DBN. It can be seen that the underlying structure of a belief MDP is a POMDP. Additionally a memory over the observations is held, the so called belief.

and Lozano-Pérez 2013). To compute the value function we apply the value iteration algorithm to the belief MDP and gain:

$$V_{t-1}(b) = \max_{\pi} E\left[\sum_{i=t}^{T} \rho(b_i, \pi(b_i))\right] \tag{2.35}$$

$$= \max_{\pi}\left(\rho(b_t, \pi(b_t)) + E\left[\sum_{i=t+1}^{T} \rho(b_i, \pi(b_i))\right]\right) \tag{2.36}$$

$$= \max_{\pi}\left(\rho(b_t, \pi(b_t)) + \sum_{b' \in \mathcal{B}} p(b' \mid b_t, \pi(b_t))\, V_t(b')\right) \tag{2.37}$$

$$= \max_{\pi} \left(\sum_{s \in \mathcal{S}} p(s \mid b_t, \pi(b_t)) R(s, \pi(b_t)) + \sum_{b' \in \mathcal{B}} \sum_{o \in O} p(o \mid b_t, \pi(b_t)) p(b' \mid b_t, \pi(b_t), o) V_t(b') \right) \tag{2.38}$$

The belief state space is exponential number of states of the underlying MDP. Thus the polynomial time of dynamic programming algorithms turns out to be too expensive. As Howard (1960) and later in more detail Kaelbling, Littman, and Cassandra (1998) showed, solving a POMDP generally is *NP*-complete. Finding good policies for a concrete POMDP is therefore a hard problem.

2.4.4 Global Optimization: Infinite Bandits

So far we considered the bandit problem for decision making. To cover active learning we need to extend the formalism further. While the bandit scenario deals with a discrete set of decisions, in active learning the agent might has to choose samples $(x_1, \ldots, x_N)$ from a continuous space. The continuous version of bandits, *continuous global optimization*, extends the bandit setting to continuous observations.

Global optimization is a framework, where a global optimum of a function should be found. Its continuous formulation can be considered as a version of the bandit setting, where there are infinitely many bandits, one at each position of the sample space, while the reward is the actual value of the function (since it is to be maximized).

The problem of continuous global optimization is defined by Močkus (1975) as follows. Let $f : \mathcal{D} \to \mathbb{R}$ be a real-valued function and let $x^* = \operatorname{argmax}_x f(x)$ be the maximum of f. We now define a decision function $d_n : \mathcal{D}^n \times \mathbb{R}^n \to \mathcal{D}$ which maps the observed data $x_1, \ldots, x_n$ and their function evaluation $f(x_1), \ldots, f(x_n)$ to a new data point x_{n+1} to query.

$$x_{n+1} = d_n(x_1, \ldots, x_n, f(x_1), \ldots, f(x_n)) \tag{2.39}$$

The task is to choose $N \in \mathbb{N}$ points $x_1, \ldots, x_N \in \mathcal{D}$, such that they minimize $|x^* - \max\{x_1, \ldots, x_N\}|$, i.e., to minimize the distance of the maximum element in the trajectory to the actual maximum of the function.

Global optimization in general does not necessarily consider stochastic functions or noisy observations of function values. If f has stochastic outcomes, it is useful to maintain a belief over the possible outcomes. And even with non-stochastic outcomes it is useful to have a probabilistic belief over not yet discovered areas. Jonas Močkus wrote a series of articles about the Bayesian method towards solving global optimization, which maintains a probabilistic belief over the function optimize (Močkus 1972; Močkus 1975; Močkus 1989). They coined the term *Bayesian optimization*. In Bayesian optimization we have a prior $p(f)$ over the function space. With each observation (x_i, y_i) we compute a posterior $p(f \mid (x_1, y_1) \ldots, (x_i, y_i))$, our belief b_i over the possible states.

With this belief we can, analogous to the previous section, form a POMDP and perform belief planning. The belief space $\mathcal{B}$ is the probability space of all functions over $\mathcal{D}$. An action is to choose an input point for the function and since f is defined over the space $\mathcal{D}$ the action space is $\mathcal{A} = \mathcal{D}$. The reward function is defined as

$$\rho(b, a) = \int_f p(f \mid a, b) f(a) \,, \tag{2.40}$$

since we want to maximize the actual function we have a belief over. The transition function becomes

$$\tau(b', a, b) = p(b' \mid a, b) = \int_{o \in O} p(o \mid a, b) p(b' \mid a, b, o). \tag{2.41}$$

Based on this belief MDP we can define the value function recursively exactly same way as in the discrete state. The only change happening is that sums now become integrals.

$$V_{t-1}(b) = \max_{\pi} E\left[\sum_{i=t}^{T} \rho(b_i, \pi(b_i))\right] \tag{2.42}$$

$$= \max_{\pi} \left(\rho(b_t, \pi(b_t)) + E\left[\sum_{i=t+1}^{T} \rho(b_i, \pi(b_i)) \right] \right) \tag{2.43}$$

$$= \max_{\pi} \left(\rho(b_t, \pi(b_t)) + \int_{b' \in \mathcal{B}} p(b' \mid b_t, \pi(b_t)) V_t(b') \right) \tag{2.44}$$

$$= \max_{\pi} \left(\int_{s \in \mathcal{S}} p(s \mid b_t, \pi(b_t)) s(\pi(b_t)) + \iint_{b' \in \mathcal{B}, o \in O} p(o \mid b_t, \pi(b_t)) p(b' \mid b_t, \pi(b_t), o) V_t(b') \right). \tag{2.45}$$

$V_{t-1}(b)$ is the optimal optimizer for the global optimization problem. There is, however, no known feasible way to solve this recursive equation, even for minimalistic examples.

2.4.5 Active Learning as Global Optimization of Information

The active learning problem as stated in Section 2.3 is to sample N samples from a (potentially) continuous space in order to minimize the expected entropy. Recall the active learning definition of Eq. 2.16 and Eq. 2.17:

$$(x_1^*, \ldots, x_N^*) = \underset{(x_1, \ldots, x_N)}{\operatorname{argmin}} E\big[H\left[p(y \mid x, D_N)\right]\big]_{x, D_N}, \tag{2.16 rev.}$$

$$(x_1^*, \ldots, x_N^*) = \underset{(x_1, \ldots, x_N)}{\operatorname{argmin}} E\big[H\left[p(\theta \mid y, x, D_N)\right]\big]_{y, x, D_N y, x, D_N}. \tag{2.17 rev.}$$

We can reframe that problem as a global optimization problem and apply belief planning to that problem. As in global optimization, we can have a belief over the function f, which generates the data points y from the inputs x, at any time t

$$b_t(f) = p(f \mid (x_1, y_1), \ldots, (x_t, y_t)). \tag{2.46}$$

Now we are not interested in directly maximize that function, but instead we want to maximize our information about it. Thus the reward is the expected neg.-entropy [3] of the distribution we are learning:

$$\rho(b_t, a) = -E\big[H\left[p(y \mid x, f, D_t)\right]\big]_{x,f,D_t}, \tag{2.47}$$

$$\rho(b_t, a) = -E\big[H\left[p(\theta \mid x, y, f, D_t)\right]\big]_{x,y,f,D_t}. \tag{2.48}$$

Again, we could use belief planning to find the optimal policy for active learning:

$$V_{t-1}(b) = \max_{\pi} \left(\rho(b_t, a) + \int_{b' \in \mathcal{B}} p(b' \mid b_t, \pi(b_t)) V_t(b') \right). \tag{2.49}$$

As with global optimization it is infeasible to compute the optimal solution to the active learning problem. While incorporating knowledge of all possible future states via the recursion leads to the optimal solution, for almost every case it is prohibitively expensive to actually compute such a expected future reward.

But we will see that we can still can build efficient heuristic algorithms. In the next section we will develop such an algorithm for environments where we have strong priors over parameters.

[3] *Negative* entropy because we maximize the reward.

3 Maximum Cross-Entropy Strategy

Parts of this chapter have already been published as:

J. Kulick, R. Lieck, et al. (2015). "The Advantage of Cross Entropy over Entropy in Iterative Information Gathering". In: *arXiv e-prints* 1409.7552v2 (stat.ML)

We have seen that we need to incorporate prior beliefs over the environment to find optimal solutions to exploration. Especially in real world exploration tasks we can encode many aspects of the world as prior knowledge, since the properties of the environment are inherently properties of the real world. For instance, physics always applies. We also know that environments are made *by* but also—and even more important—*for* humans. This knowledge is not limited to a single environment, but holds for many situations robots might face. If we encode this knowledge as prior belief in the learning algorithm, exploration can focus on those cases where the priors do *not* hold or are vague. Unsurprisingly the most interesting parts of the world are those which are special, and we want exploration methods to focus on these specialties. Active learning methods for real world exploration should thus include a notion of what is regular in real world scenarios to infer what is interesting.

In the Bayesian framework developed so far prior knowledge is naturally defined as prior belief distributions. Consider the graphical model in Fig. 2.3. We want to gather information about the underlying model of the environment, i.e., about hyperparameters θ that define the behavior of the environment. In real world scenarios such a hyperparameter can be as abstract as whether two objects are connected by a joint (this is exactly the case in the

experiments in Sec.4.2). Typically, we have strong beliefs about such parameters (e.g., not many objects are connected by a joint, but if, that is interesting), which would be encoded by a high a priori probability for each two objects that they are not connected. But as discussed, exploration should quickly uncover if areas do not follow the prior belief. We would like to find out the interesting areas of the state, that do not follow our prior belief and update them accordingly, e.g., we want to find out, which objects are connected by a joint. In this chapter we will show that active learning methods so far are not well suited in situations where we have a strong prior belief. We will develop an algorithm that closes this gap.

3.1 Bayesian Experimental Design for Exploration

Maximizing information about the hyperparameters θ is typically the task of Bayesian experimental design. Since directly optimizing over the whole trajectory of samples is computationally too expensive (see Section 2.4), we need to handle the problem iteratively and do not consider all possible future states. It is intuitive to minimize the one-step expected hypotheses entropy[1]. This is a common utility function for Bayesian experimental design (Chaloner and Verdinelli 1995). We use negative entropy (NE) for clarity. That way we can maximize all criteria.

$$x_{NE} = \operatorname*{argmax}_x \int_y -p(y \mid x, D) H\left[p(\theta \mid D, x, y)\right] \,. \tag{3.1}$$

It is instructive to rewrite this same criterion in various ways. We can, for instance, subtract $H\left[p(\theta \mid D)\right]$, as it is a constant offset to the maximizing operator (see App. A.2 for the detailed transformations):

$$\operatorname*{argmax}_x \int_y -p(y \mid x, D) H\left[p(\theta \mid D, x, y)\right] \tag{3.2}$$

[1] Note that this is often called the *conditional entropy* and written $H\left[\theta \mid y\right]$. To avoid confusion, we will not use this shorthand notation but explicitly state the distribution we take the entropy of and over what random variable we will take the expectation.

$$= \operatorname*{argmax}_x - \int_y p(y \mid x, D)\, H[p(\theta \mid y, x, D)] - H[p(\theta \mid D)] \tag{3.3}$$

$$= \operatorname*{argmax}_x \int_y p(y \mid x, D)\, \mathrm{D}_{\mathrm{KL}}\left[p(\theta \mid y, x, D) \parallel p(\theta \mid D)\right] . \tag{3.4}$$

These rewritings of expected entropy establish the direct relation to utility functions of Bayesian experimental design (Eq. (3) and (4) in Chaloner and Verdinelli 1995). We find that x_{NE} can be interpreted both as maximizing the expected neg. entropy, as in Eq. (3.1), or maximizing the expected KL divergence, as in Eq. (3.4).

Minimizing the expected model entropy is one way of maximizing information gain about θ. However, in the iterative setup we will empirically show that this criterion can get stuck in local optima: Depending on the stochastic sample D, the hypotheses posterior $p(\theta \mid D)$ may be "mislead", that is, having low entropy while giving high probability to a wrong choice of θ. The same situation arises when having a strong prior belief over θ. As detailed below, the attempt to further minimize the entropy of $p(\theta \mid D, x, y)$ in such a situation may lead to suboptimal choices of x_{NE} that confirm the current belief instead of challenging it.

This is obviously undesirable. Instead we want to have a robust belief that cannot be changed much by future observations, not because the change is avoided, but because all necessary evidence is already incorporated in the belief. We therefore want to induce the biggest change possible with every added observation. In that way, we avoid local minima that occur if a belief is wrongly biased. While minimizing the entropy would in this situation avoid observations that change the belief, measuring the change of the belief regards an increase of entropy as a desirable outcome.

3.2 Maximum Cross-Entropy

One approach to maximize the expected change of the entropy would be to maximize

$$x_{CH} = \operatorname*{argmax}_x \int_y p(y \mid x, D) \left| H[p(\theta \mid y, x, D)] - H[p(\theta \mid D)] \right| . \tag{3.5}$$

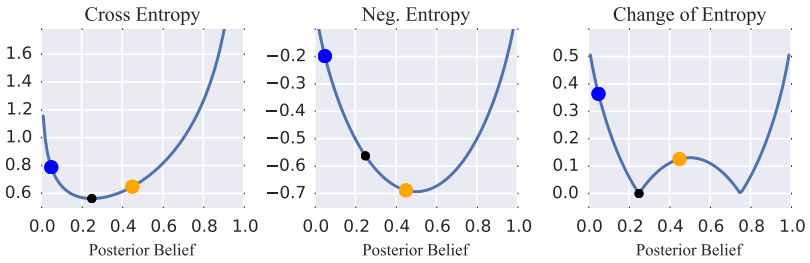


Figure 3.1: Characteristics of three different criteria for choosing samples: Cross entropy, neg. entropy, and change of entropy. The belief is over a binary variable, i.e., a point on the x-axis encodes the whole belief. The black dot indicates the prior belief of 0.25, the blue and yellow dot indicate the posterior after having seen two different observations. These observations are not explicitly shown. The blue graph shows how the different criteria assess different expected posterior beliefs. *Cross entropy* regards a change in any direction as an improvement. *Neg. entropy*, in contrast, prefers changes that support the current belief over those that challenge it – unless the posterior belief flips to having an even lower entropy than the prior. *Change of entropy* is similar to cross entropy in that for small changes it regards any direction as an improvement. For larger changes, however, is has a local optimum for a flat posterior of 0.5 and a local minimum for a flipped posterior with the same entropy as the prior. (Figure from Kulick, Lieck, et al. (2015).)

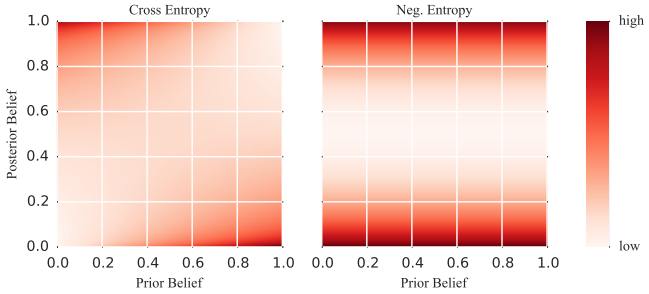


Figure 3.2: Cross entropy and neg. entropy as a function of prior and posterior belief of two possible hypotheses. Values are normalized and the axes show the probability of one of the two hypotheses. Maximizing the cross entropy prefers a high entropy only if it *changes* the belief while maximizing the neg. entropy ignores the prior belief. (Figure from Kulick, Lieck, et al. (2015).)

This criterion has two undesirable pathologies: (a) it always has a local maximum for a flat posterior belief with maximum entropy—unless the prior is already flat—and (b) changing a strong belief, say 0.25/0.75 for a binary hyperparameter, to the equally strong but contradictory belief of 0.75/0.25 is one of the global minima with zero change of entropy (see Figure 3.1).

Another criterion that measures the change of the belief is the cross entropy between the current and the expected belief.

$$H[p(x); q(x)] = -\int_x p(x) \log q(x). \tag{3.6}$$

The differences between the neg. entropy and cross-entropy can be seen in Fig. 3.2. Whereas the neg. entropy is the same for all prior beliefs, the cross entropy is high, when prior and posterior belief disagree. Intuitively neg. entropy actually does not measure the *change* of distributions, but only the information of the posterior belief $p(\theta \mid D, x, y)$.

When facing strong but misleading priors the neg. entropy will not value

change that does not tilt the belief toward the opposite, i.e., a different outcome on a discrete random variable is more likely than the most likely outcome in the prior. We will see this in detail soon. For situations were strong priors are desirable we therefore propose the *MaxCE* strategy which maximizes the expected *cross entropy* between the prior hypotheses belief $p(\theta \mid D)$ and the posterior hypotheses belief $p(\theta \mid D, x, y)$. This, again, can be transformed to maximizing the KL-divergence, but now with switched arguments (see again App. A.2 for details).

$$x_{CE} = \underset{x}{\operatorname{argmax}} \int_y p(y \mid x, D)\, H[p(\theta \mid D); p(\theta \mid D, x, y)] \tag{3.7}$$

$$= \underset{x}{\operatorname{argmax}} \int_y p(y \mid x, D)\, \mathrm{D}_{\mathrm{KL}}\left[p(\theta \mid D) \parallel p(\theta \mid D, x, y)\right]. \tag{3.8}$$

The KL-divergence $\mathrm{D}_{\mathrm{KL}}\left[p(\theta \mid D) \parallel p(\theta \mid D, x, y)\right]$ quantifies the additional information captured in $p(\theta \mid D, x, y)$ relative to the previous knowledge $p(\theta \mid D)$.

This does not necessarily require the entropy to decrease: the expected divergence $\mathrm{D}_{\mathrm{KL}}\left[p(\theta \mid D) \parallel p(\theta \mid D, x, y)\right]$ can be high even if the expected entropy of the distribution $p(\theta \mid D, x, y)$ is higher than $H[p(\theta \mid D)]$—so this criterion is not the same as minimizing expected model entropy. The following example and the later quantitative experiments will demonstrate the effect of this difference.

3.2.1 An Illustrative Example

Bayesian experimental design suggests to minimize the expected entropy of the model distribution Eq. (3.3). As we stated earlier, this may lead to getting stuck in local optima in an iterative scenario. We now explicitly show an example of such a situation. Assume a regression scenario where two Gaussian Processes (GP, see for instance Rasmussen and Williams (2006) for an introduction to GPs) hypotheses should approximate a ground truth function. Both GPs use a squared exponential kernel, but have a different length

scale hyperparameter. The ground truth function is a sample from one of the GPs.

Consider now a case where the first two observations by chance support the wrong hypothesis. This may happen due to the fact that the ground truth function is itself only a random sample from the prior over all functions described by the underlying GP and as such might be a relatively unlikely one. Furthermore observations may be noisy, which may lead to a similar effect. Such a scenario is shown in Figure 3.3. After two observations the probability for the wrong model in this scenario is already high. If we now compute the expected neg. entropy from Equation 3.3 it has its maximum close to the samples we already got. This is due to the fact that samples possibly supporting the other—the correct—model would temporarily decrease the neg. entropy. It would only increase again if the augmented posterior actually flipped and the probability, such that it is higher than the prior belief for the wrong hypothesis.

The MaxCE approach of maximizing cross entropy (see Equation 3.7) on the other hand favors *changes* of the hypothesis posterior in any direction, not only to lower entropy, and therefore recovers much faster from the misleading first samples. Figure 3.3 shows both objectives for this explicit example.

3.2.2 The Conditional (Posterior) Hypotheses Entropy is not Submodular

At the first glance, the former findings might contradict the fact that the entropy is submodular (Fujishige 1978) and optimizing submodular functions can be done efficiently (Iwata et al. 2001; Nemhauser et al. 1978). But it is to notice that the hypotheses entropy we are trying to minimize is a different entropy: The submodular entropy function is a set function on set of random variables, where the entropy of the joint distribution of all variables in the set is computed. Formally if $\Omega = \{V_1, \ldots, V_n\}$ is a set of random variables, then for any $S \subseteq \Omega$ the entropy of this subset $H[p(S)]$ is submodular. In contrast, we compute the entropy of the distribution of a fixed random variable, conditioned on a set of random variables (see Eq. (3.1)). As

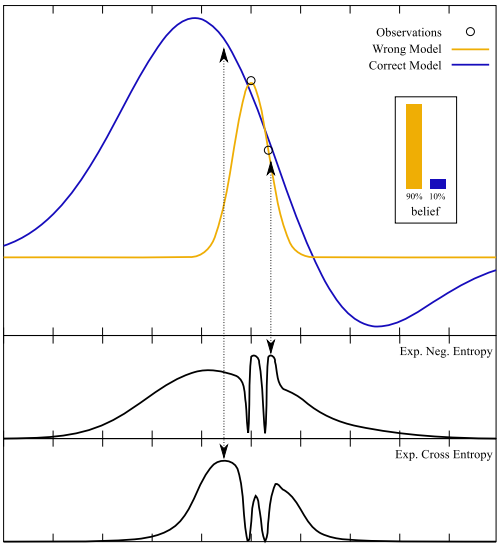


Figure 3.3: The top graph shows two competing hypotheses where one corresponds to the correct model (the Gaussian process the data are actually drawn from) and the other is wrong (a Gaussian process with a narrower kernel). For the two observations seen so far, the current belief is biased towards the wrong model because it is the more flexible one, i.e., the underlying Gaussian process can adopt to functions with higher frequencies. The two curves below correspond to the expected neg. entropy (Eq. 3.3) and the expected cross entropy (Eq. 3.4) of the belief, conditioned on a query at the corresponding location. The arrows indicate the query location following each of the two objectives. Figure from Kulick, Lieck, et al. (2015).

noted earlier this is the conditional entropy. See Sec. A.1 for a proof that the conditional hypotheses entropy is not submodular.

3.2.3 Differences to Traditional Active Learning Methods

As opposed to most existing active learning methods, Bayesian experimental design with the traditional utility function, as well as our MaxCE criterion, defines the objective directly in terms of the hypotheses belief $p(\theta \mid D)$ instead of the predictive belief $p(y \mid x, D)$.

As active learning is typically used to improve machine learning procedures it utilizes the predictive belief to measure sampling performance. It bears however close resemblance to Bayesian experimental design. Minimizing the expected entropy on the predictive belief for example is the direct translation from Bayesian experimental design to the predictive belief:

$$x_{SI} = \operatorname*{argmax}_x \int_y p(y \mid x, \theta, D) H\left[p(y \mid x, \theta, D)\right] \tag{3.9}$$

In the case $p(y \mid x, \theta, D)$ assumes the form of a Gaussian, this is the same as minimizing the expected variance of the predictive belief, since

$$H\left[\mathcal{N}(\mu, \sigma^2)\right] = \frac{1}{2} \log(2\pi e \sigma^2), \tag{3.10}$$

which is a strictly monotonically increasing function on σ^2. Minimizing the expected mean variance over the whole predictive space is introduced as active learning criterion by Cohn, Ghahramani, et al. (1996).

A mix between these two worlds is Query-by-Committee (QBC) (McCallum and Nigam 1998; Seung et al. 1992). While aiming at discriminating between different hypotheses, it uses the predictive belief for measurements.

Note that finding the correct hypotheses does not directly lead to good predictive performance. While samples from one area of the input space might clearly discriminate between two competing hypotheses, they do not necessarily tell anything about the other areas of the input space. So we

might still predict poorly, although we are certain, which hypothesis is the correct one. On the other hand good prediction performance does not necessarily lead to discrimination between hypotheses, since hypotheses might agree on the prediction in large areas and only disagree only in a small fraction of cases.

3.3 Quantitative Experiments

Tasks that occur in real world scenarios can often be classified as either regression (predicting a function value) or classification (predicting a class label) tasks. We tested both task classes first on synthetic data. We also tested the regression scenario on a real world data set. Usually machine learning scientists are interested in prediction performance, whereas classical statisticians might value that finding the correct hypothesis might help for the task as well as generalizing to further situations. We tested both in our experiments. As ad-hoc solution for prediction we combined uncertainty sampling with the MaxCE in a linear fashion.

Further results of the exploration strategy in more complex scenarios can be found in Sec. 6.2. All earlier robotic experiments used other measures, because the MaxCE method was not yet developed.

3.3.1 Compared Methods

We compared six different strategies: MaxCE, classical Bayesian experimental design, which we define to minimizes the expected entropy, query-by-committee, uncertainty sampling, and random sampling, which randomly chooses the next sample point.

Normally, uncertainty sampling is used to train a single model, i.e., only one hypothesis is assumed, while for comparing it to our methods we have to consider a set of hypotheses. The most natural way to handle the set of models is as a mixture model and then minimize the variance of this mixture model

$$x_{US} = \operatorname*{argmin}_x E\left[p(y \mid x, \theta, D)^2\right]_\theta - E\left[p(y \mid x, \theta, D)\right]_\theta^2. \tag{3.11}$$

Additionally we tested a mixture of MaxCE and uncertainty sampling

$$x_{mix} = \underset{x}{\operatorname{argmax}}\ \alpha \cdot f_{CE}(x) + (1-\alpha) \cdot f_{US}(x)\,, \tag{3.12}$$

where f_{CE} and f_{US} are the objective functions maximizing the expected cross entropy and the entropy of the predictive distribution respectively:

$$f_{CE}(x) = \int_y p(y \mid x, D)\, H[p(\theta \mid D); p(\theta \mid D, x, y)] \tag{3.13}$$

$$f_{US}(x) = -E\left[p(y \mid x, \theta, D)^2\right]_\theta + E\left[p(y \mid x, \theta, D)\right]_\theta^2 . \tag{3.14}$$

The mixing coefficient, which was found by a series of trial runs, was $\alpha = 0.5$ for both synthetic data sets and $\alpha = 0.3$ for the CT slices data set.

3.3.2 Measures

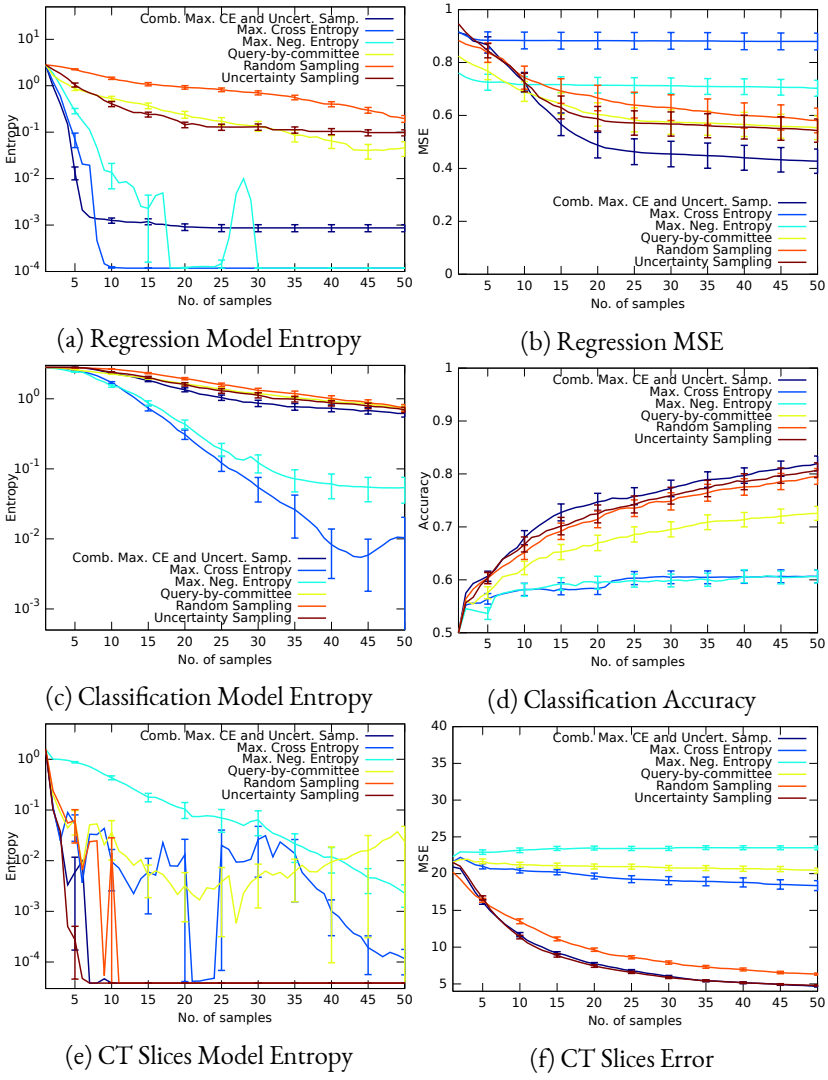
To measure progress in discriminating between hypotheses we compute the entropy of the posterior hypotheses belief for each method. To measure progress in the predictive performance we plot the classification accuracy and the mean squared error for classification and regression, respectively. To compute an overall predictive performance for a method we took the weighted average over the different models, with the posterior probabilities as weights. This corresponds to the maximum a posteriori estimate of the marginal prediction

$$p(y \mid D, x) = \sum_\theta p(\theta \mid D)\, p(y \mid \theta, D, x)\,. \tag{3.15}$$

Fig. 3.4 shows these measures for all our experiments.

3.3.3 Synthetic Data

We test our method in both a 3d-regression and a 3d-classification task. The setup for both experiments was essentially the same: A ground truth Gaussian Process (GP) was used to generate data. The kernel of the ground truth



(a) Regression Model Entropy

(b) Regression MSE

(c) Classification Model Entropy

(d) Classification Accuracy

(e) CT Slices Model Entropy

(f) CT Slices Error

Figure 3.4: The mean performance of the different exploration methods for the classification and regression tasks. Figure from Kulick, Lieck, et al. (2015).

GP was randomly chosen to depend either on all three dimensions (x, y, z), only a subset of two dimensions (x, y), (y, z) or (x, z), or on only one dimension (x), (y) or (z). Finding the correct hypothesis in this case corresponds to a feature selection problem: uncovering on which features the unknown true GP depends on. The latent variable θ, to be uncovered by the active learning strategies, enumerates exactly those seven possibilities. One run consisted of each method independently choosing fifty queries one-by-one from the same ground truth model. After each query the corresponding candidate GP was updated and the hypotheses posterior was computed.

Fig. 3.4a, 3.4b, 3.4c and 3.4d show the mean performance over 100 runs of the synthetic classification and regression tasks, respectively. Both hypotheses belief entropy and accuracy/mean squared error are shown.

On this synthetic data MaxCE significantly outperforms all other tested methods in terms of entropy, followed by Bayesian experimental design, and the mixture of MaxCE and uncertainty sampling (Fig. 3.4a and 3.4c). As expected, in terms of classification accuracy and predictive error both MaxCE and Bayesian experimental design perform poorly. This is because their objectives are not designed for prediction but for hypothesis discrimination. However, the mixture of MaxCE and uncertainty sampling, performs best (Fig. 3.4b and 3.4d), which is presumably due to its capability to uncover the correct hypothesis quickly and to trade prediction accuracy for improved chances of finding the true hypothesis.

3.3.4 CT-Slice Data

We also test our methods on a 384-dimensional real world data set from the machine learning repository of the University of California, Irvine (Bache and Lichman 2013). The task on this set is to find the relative position of a computer tomography (CT) slice in the human body based on two histograms measuring the position of bone (240 dimensions) and gas (144 dimensions). We used three GPs with three different kernels: a γ-exponential kernel with $\gamma = 0.4$, an exponential kernel, and a squared exponential kernel. Although obviously none of these processes generated the data, we try to find the best matching process alongside with a good regression result.

Fig. 3.4e and 3.4f show the mean performance over 40 runs on the CT slice data set.

In the CT slice data set neither MaxCE nor Bayesian experimental design minimize the entropy quickly (Fig. 3.4e). This may be a consequence of the true model *not* being among the available alternatives. As a consequence both methods continuously challenge the belief thereby preventing it from converging. QBC may be subject to the same struggle, here even resulting in an increase of entropy after the first 25 samples. In contrast, the entropy converges reliably for uncertainty sampling, the mixture method, and random sampling. Concerning the predictive performance MaxCE, Bayesian experimental design, and QBC do not improve noticeably over time (cf. explanation above). Again uncertainty sampling and the mixture method perform much better, while here the difference between them is not significant.

3.4 Conclusion

Exploration strategies typically try to improve the prediction of a learning algorithm. In exploring the environment with a robotic agent we are more interested in gathering information about the environment instead of directly predicting its future state. As discussed discrimination between hypotheses does not directly lead to good prediction performance, because it might not cover the input space sufficiently and still know the correct hypothesis. This information, however, generalizes better about the environment and is easily transferable to new tasks, eventually leading to better prediction performance due to better models.

Bayesian experimental design aims exactly at gathering information about hypotheses. We have shown that traditional measures from Bayesian experimental design are not necessarily suitable for the nature of exploration, where we want to incorporate strong prior beliefs to quicken the exploration and focus on particularities of the current environment. In such a situation maximizing the neg.-entropy of the belief tries to prove the strong belief instead of acknowledging that other regions of the input space might contradict the current best belief. We developed a method without this short-

coming. The MaxCE method tries to implement the scientific method: it values experiments that falsify a hypotheses equally to those strengthening it. Thus the interesting parts of the environment, which do not follow the prior belief, are uncovered quickly. We will show examples of this behavior in the subsequent chapters.

Using strong prior beliefs over the environment is an important technique to speed up lengthy exploration runs and let the learning focus on the relevant parts of the environment. Having exploration strategies that incorporate this knowledge is thus a major step towards robots autonomy.

4 Background on Exploration in Robotics

The Physical Exploration Challenge (see Sec. 4.2) and the Rigid World Assumption (see Sec. 4.2.2) were first mentioned in the following publication and are defined in greater detail in this thesis.

S. Otte et al. (2014). "Entropy Based Strategies for Physical Exploration of the Environment's Degrees of Freedom". In: *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 615–622

Exploring the environment is important for both animals and humans to learn about their surroundings. From infants playing with various objects to learn about their properties (e.g.j Baldwin et al. (1993)) to scientists exploring their area of research with experiments, new knowledge is gathered by means of exploration. The cockatoo experiment, mentioned already in the introduction of this thesis, is an impressive presentation of exploratory behavior in birds, and showed how cockatoos were able to open a complex mechanism to reach food after exploring the mechanism with various actions (Auersperg et al. 2013). It shows that to solve problems in new environments techniques for exploration are crucial.

Given its importance in animals and humans, exploration should be a very important subfield of robotics. Also the typical usage of robots suggests that autonomous exploration is important in the robotics research community: The mars rover explores the distant planet (Crisp et al. 2003), robotic search teams enter hazardous environments (Ackerman 2011; Guizzo 2011)

and the DARPA Robotics Challenge 2015 showed state-of-the-art robots in typical search-and-rescue situations (DARPA Robotics Chellange Organisation Team 2015). Robots in all these examples apparently need to explore their environment. They need to act in unknown environments and need to be fault tolerant to previously unknown scenarios. However, these robots do not autonomously explore their environment. Instead they are remotely controlled, without much autonomy. The exploration strategy is outsourced to a human operator and the robots are only remote avatars of a human being. For these specific projects this is a reasonable decision, given the current state of autonomy in robotics. But if robots should be widely deployed in the future, remote controlled systems are not an option. The problem of exploration has to be addressed. And while there are some examples of fully autonomously exploring robots (Thrun et al. 2004), in those examples the robots only passively map their environment and do not truly interact with it.

Surprisingly, autonomous exploration of the environment or parts of it only recently found interest within the robotics community. A major reason is that tough challenges were tackled developing the various sub-systems of robots, like motion planning, navigation, or core vision techniques. But today the fundamental abilities for robots exploring the environment as well as the computational power of computers, are developed well enough to open the room for conducting research on bringing autonomous exploration to robots.

Autonomous exploration of the environment is an important ability for animals and humans, and so it is for artificial intelligent agents. In any realistic scenario it is impossible to model the environment exhaustively. Instead, robots, supposed to act in a wide variety of applications, need to build those models on their own by interacting with the environment. It is thus important to model the interaction strategies for the agents instead of the specific environment.

As we have shown in the last chapter, active learning strategies are good candidates for generating exploration strategies. They perform well on machine learning problems and are theoretically sound.

In this chapter, we will give background on work on active learning in

robotics. We will first examine the literature on this topic, showing that so far exploration in robotics is mainly focused on exploring internal degrees of freedom, i.e., degrees of freedom that can directly be altered by the robot, typically through motors. We also discuss the recent developments on exploration on external degrees of freedom, degrees of freedom that are only changeable through interaction with the environment. Then we will present work we base parts of our robotic system in the experiments on, e.g., the perception system of our robot. After the related work we will define the *Physical Exploration Challenge*, which will define the problem of exploration in robotics rigorously and frame the setting of the further experiments. To focus on the problems arising from the exploration of external degrees of freedom context, we restrict the scenarios to a relatively easily modeled set, namely we will assume the environments to consist of rigid bodies and joints only. We will end this chapter by a brief discussion of the problem of *physical reasoning* and possible solutions. Physical reasoning is the process of reasoning about physical configurations of an environment. Many of theses configurations might be infeasible due to physical laws. We will discuss how we can ensure that a robot does not try to explore such configurations.

4.1 Related Work

4.1.1 Exploration of Internal Degrees of Freedom

Using active learning in robotics has a long tradition. Already the early active learning papers from Cohn, Ghahramani, et al. (1996) mentions learning the dynamics of a robotic arm as application for active learning and even earlier Thrun (1992) used active learning to train robot navigation. However, active learning in robotics is mainly concerned with exploration of internal degrees of freedom, which we define as DOF directly controllable by the robot (see Sec.4.2.1). We will review the literature on active learning on internal DOF and then show that recently exploration on external DOF draw attention to the community.

Exploration is central in the field of developmental robotics (Cangelosi

and Smith 2015; Lungarella et al. 2003). In developmental robotics the key question is how to enable robots to develop skills from a small set of built-in capabilities, similar to the early development of human children. To extend their capabilities they need to learn from novel situations. Exploration is needed to confront robots with those new situations. Hester et al. (2013) and Lopes et al. (2012) define exploration for that purpose in the reinforcement learning framework. The described problems are however not full robotic environments, but abstract away the actual robot. The same is true for intrinsically motivated reinforcement learning described by Singh et al. (2010) where an internal reward signal is used to generate explorative behavior.

An often applied procedure used in developmental robotics is motor babbling. Here the robot constantly changes motor commands, which the robot should reach with an endeffector (Rolf et al. 2010). With those movements the robot typically learns its inverse kinematic model. Usually the motor position is randomly changed, hence the name babbling. It comes from observations of human infants that seem to randomly move their bodies in order to understand how it reacts to specific muscle commands (Meltzoff and Moore 1997). A more sophisticated approach is to actively choose goals, both directly in motor space as well as in goal space (Moulin-Frier and Oudeyer 2013a,b). These approaches use a measure of learning accuracy, which they define in terms of the distance of the prediction to the actual outcome of a motor command. Thus the learner tries to actively choose motor commands that improve the prediction of the kinematic model of the robot.

Another area of robotic research, where active learning techniques are applied, is self-localization and mapping (SLAM). While already only localization benefits from the active choice of exploration actions (Burgard et al. 1997; Fox et al. 1998), SLAM poses the following problem: a robot has to simultaneously draw a map of the area it is acting in and localize itself on that map. The active choice of movements to enhance the precision of both the map and the localization is called active SLAM (Leung et al. 2006, 2008; Sim and Roy 2005). Various criteria for measuring the uncertainty in active SLAM have been proposed and typically resemble the criteria in active learning and optimal design such as A-optimality and D-optimality, usual

measures in the frequentists optimal design literature (Carrillo et al. 2012). But also Bayesian techniques are used to actively drive the exploration in SLAM (Martinez-Cantin et al. 2007). Stachniss et al. (2005) use information theoretic measures to decide on the next action to take. However, all active SLAM algorithms are limited to move the robot to a desired position and do not handle interactions with the environment.

Although grasping is inherently concerned with manipulating external degrees of freedom, active learning in the grasping literature has mainly applied to internal DOF: Active learning is used to sample grasp directions or positions (Kroemer et al. 2010; Montesano and Lopes 2012; Salganicoff et al. 1996) to determine the best grasping strategy—approach direction or pre-grasp pose of the hand—for a given object. It is also used to sample approach trajectories to unknown objects to use the tactile feedback to model the surface of the object (Dragiev et al. 2011, 2013).

Active learning algorithms are also used within the field of human-robot-interaction (HRI). When interacting with human beings, it is crucial to ask meaningful questions and gain as much insight as possible with each, to not annoy the human operators. In the work of Chao et al. (2010) a human teacher is supposed to teach a robot symbols made from paper cut-outs. Here the teacher is asked to generate new samples by speech interaction, the robot does not manipulate the state on its own. In the work of Cakmak and Thomaz (2012) a robot learns a pouring task, where it generates new trajectories which are then judged from a human teacher. Here it only explores internal degrees of freedom.

4.1.2 Exploration of External Degrees of Freedom

All active learning and exploration methods discussed so far are concerned with the *internal* degrees of freedom of a system, i.e., all degrees of freedoms that are directly controllable by the system, such as joint angles of servo motors. None of the work is directed towards exploration of external degrees of freedom, which can be only controlled through interaction with the environment with directly controllable degrees of freedom, e.g., the position of an object, which is only controllable when grasped. From the 596 papers

published at *Robotics: Science and Systems* within the period of its existence so far (since 2005) only 34 (less than 6%) are concerned with manipulating objects at all, whereas almost 25% are concerned with trajectory generation (e.g., motion planning, and control)[1]. This is certainly explainable by the computational difficulty of the problems involved in manipulation tasks and also by the difficulty in performing the tasks on real robotics hardware. Manipulating external degrees of freedom also bears the risk of damage to both the robot and the environment and is thus often avoided. Given that only such a small percentage of research papers deals with manipulation at all, it is no wonder that autonomous exploration through interaction is still not thoroughly analyzed. In recent years, however, the topic got attention.

One of the first publications using active learning on external degrees of freedom is our paper about teaching robots grounded symbols (Kulick, Toussaint, et al. 2013). It uses active learning to ask a human teacher for labels about spatial relations of objects. The detailed findings are part of this thesis (see Sec. 5).

van Hoof, Kroemer, and Peters (2014) and van Hoof, Kroemer, Ben Amor, et al. (2012) conducted research on autonomous exploration within cluttered environments for segmentation, also called interactive perception. A robot pushes objects around on a cluttered tabletop scenario to understand which parts on the sensory input belong to the same object. It uses a probabilistic forward model to estimate the effects and quantifies the information obtained by the action. In a similar fashion, Bersch et al. (2012) and Hausman, Balint-Benczedi, et al. (2013) segment objects through motion induced from a robotic arm. They, however, use the concavity of regions as heuristic to guide actions instead of a full probabilistic model of the actions.

[1] Measuring whether a paper is concerned with such a broad topic is subjective. Of course any manipulation has to be done by some means of trajectory generation, so arguably any trajectory generation paper has potential influence on manipulation. In our count, however, we only included all papers which are directly concerned with manipulating the state of an object not directly controllable, e.g., relocation of objects or cutting objects. We decided upon whether a paper is concerned with manipulation by manually judging the title and in the case of uncertain titles judging the abstract. The numbers are, however, not necessarily sharp. They give nevertheless a good estimate of the amount of research done in a given direction.

The principles of artificial curiosity—building a predictive world model as intrinsic motivation—were also applied to robotics. Ngo et al. (2012) and Ngo et al. (2013) used artificial curiosity to learn skills such as building block stacks from playing with blocks. In this work the focus is on learning action skills as byproduct of learning a world model. All previously discussed research does not consider more complex environments with constrained degrees of freedom, i.e., joints.

Very close in spirit to our work is the work of Hausman, Niekum, et al. (2015). They employ a recursive state estimator to classify types of joints and estimate their parameters, while keeping a distribution over joint hypothesis available within a particle filter. This enables them to actively choose actions by means of information theoretic measures. An action in their setting is a force applied to an object. They also use a Bayesian change point detection (Niekum et al. 2015) to find end points of joints. We employ a change point detection to more generally find hints in the dynamics of a joint, that lead to joint dependency detection (see Sec. 6.2). Their work is, however, only on the level of exploring one particular joint. We used similar strategies for higher level exploration strategies (i.e., answering the question of which object in a complex environment we should explore first).

In work on interactively identifying joints by Barragán et al. (2014) an input-output hidden Markov is used to model articulated mechanisms. They use Bayesian inference to uncover the latent variables, especially the type of the mechanism. To make the computations tractable they limit themselves to a discrete sets of mechanisms and a discrete set of actions. The robot has then to infer which joint type it is acting on, based on the observation if the desired motion could be executed or not. Our experiments are going further in that they consider more than one joint in the environment. We also address the question of existence of joints and how to account for the entropy of non-existing distributions.

Another quite different approach is that of Höfer et al. (2014). They pose the exploration problem as relational reinforcement learning problem, where reward is given when a dependency between different parts of furniture is uncovered, e.g., learning that a given handle opens a door. A ε-greedy strategy is used for exploration. Note that exploration in this context is two-

fold: at the one hand they describe the problem as reinforcement learning problem and thus need exploration in a reinforcement learning sense, on the other hand the policy learned by the reinforcement learner implements an exploration strategy in the sense of exploration defined earlier in this thesis.

4.1.3 Planning in Belief Space

Beside these explicit exploration heuristics, Kaelbling and Lozano-Pérez (2013) model robotic tasks as POMDP and plan in belief space, as described in Sec. 2.4. Thus they act optimally without explicitly incorporating exploration strategies in their work. That way they, however, have to model uncertainty over everything explicitly—so far the belief representation is relatively naive and does not scale well. They use a Gaussian distribution over the pose of each objects, which will be intractable when the number of objects in a scene raises. It is also laborious to hand-build the representations for a new domain of tasks.

4.1.4 Interactive Perception

In our experiments we want to learn a model of the underlying kinematics of the environment (see Sec. 4.2.2 for details of the assumption of a rigid world). To learn a model of such an environment, we need algorithms that are able to identify parameters of joints from sensory input. Various methods have been developed. They are subsumed under the term *interactive perception*, since they need movement of objects and thus interaction with objects. In the experiments in the further chapter we use such systems to perceive type and parameter of the joints.

Sturm et al. (2011) propose a probabilistic method, that identifies the type of a joint from a time series of pose estimates of objects. It is able to distinguish between rigid, revolute and prismatic joints. For other types of connected movements it uses a non-parametric Bayesian model. It leverages a graphical model to obtain the most probable (maximum a posteriori probability) kinematic model for each link between objects. It is, however, necessary to obtain trajectories from object parts. Thus it uses a marker based

object detection.

The work of Brock's group, in contrast, acts directly on the camera image. They published a series of papers on identifying joint types from sensory input and interaction with the objects (Katz, Orthey, et al. 2014; Katz, Pyuro, et al. 2008; Martín-Martín and Brock 2014). They use RGB features (SURF features, see Bay et al. (2008)) which they enhance for the use on RGB-D data. From the movement of these features they model the rigid body motion. From this motion the kinematic structure is identified. With this structure a forward model is inferred to predict the further movement and finally generate a recursive state estimator.

Since our work assumes objects to be identified, we use the library from Sturm to estimate the joint type and parameters.

Despite the term interactive perception, the motions in the work of both groups are still pre-scripted by a human experimenter and not autonomously inferred. These algorithms are thus very good tools for the use with exploration strategies but do not address the question of exploration.

4.1.5 Dynamics of Joints

When exploring degrees of freedom models of joints are helpful in a variety of use cases. They define joint parameters and can give a prediction of movement. Endres et al. (2013) analyzed the movement of doors and built a analytic dynamic model of them, which enables them to estimate the door parameters, such as the center of rotation of the hinge, and the deceleration parameter (which subsumes physical effects such as friction or inertia) directly by solving a system of linear equations. As input they need the trajectory of a point on the door and its normal, which they observe with a depth sensor. They are then able to predict movements of the door given forces acting on it. With their compliant robotic arm they can move the door to desired positions. In our work we leverage a similar, but simpler, dynamic model of a joint (a) to infer joint parameters with sampling methods (see Sec. 6.1) and (b) to pre-process force-torque sensor data (see Sec. 6.2).

4.2 The Physical Exploration Challenge

To assess the exploration performance of a robot, we need to define a goal behavior we want to achieve. Loosely speaking this behavior can be described as follows.

> If a robot enters a new environment it should start interacting with it until it has acquired a sufficiently precise model of it.

The *Physical Exploration Challenge* is the formalization of this behavior. This is a important problem for robotics, as it is a central behavior to be applied to unknown environments. It enables the agent to find out the specialties of a scenario to be able to use the objects in its environment. The physical exploration challenge consists of three central parts:

1. *Perception:* The agent needs to be able to perceive the state of the environment and its various parts. In the environments we will investigate this will be mostly the perception of joints. We will use the work of Sturm et al. (2011) for this purpose.

2. *Motor Skills:* The agent needs to be able to interact with the world by means of its motor skills. This is a central research area of robotics. We will use state of the art operational space controllers to enable the agent for interaction (Toussaint et al. 2010). Learning these skills has been, as mentioned before, the topic of developmental robotics (Moulin-Frier and Oudeyer 2013b).

3. *Exploration Strategies:* The agent needs strategies that tell him how to explore environments. This essentially is the question of which action the robot should take next. To answer that question we will show how to transfer insights from active learning research to exploration of external degrees of freedom of environments in robotic tasks. Developing successful and efficient exploration strategies on external degrees of freedom is the main contribution of the following chapters.

We will now rigorously define the problem of the physical exploration challenge.

4.2.1 Environments, and Internal and External Degrees of Freedom

Many definitions of degrees of freedom haven been proposed over time (Pennestri et al. 2005). We will use a definition based on the formulation of general dynamical systems (Mazzola and Giunti 2012).

Definition 1. *Let $E = (T, M, \Phi)$ be a dynamical system on a monoid T, with state space $M \subseteq R^m$ being a set and evolution function $\Phi : T \times M \mapsto M$. We call E the environment. Every dimension of the state space M we call a degree of freedom.*

We do not want the robot and its strategy be defined by the evolution function, since we want to use an agent in multiple environments and want to clearly distinct between the environment and the robot. Thus we need the agent to be able to interact with the environment. We first define the agent as its policy.

Definition 2. *Let A be a set of actions. Then an agent is defined by its policy $\pi : T \times M \mapsto A$.*

Now we define the interface for the agent to interact with the environment. First we distinguish between internal and external degrees of freedom. Although the following definition alone does not explain the difference between internal and external degrees of freedom we need it for further definitions.

Definition 3. *The state space M of an environment is the Cartesian product of the two sets $M = M_I \times M_E$. We call M_I internal degrees of freedom, and M_E external degrees of freedom.*

The agent needs a way to interact with the environment through the controllable degrees of freedom. Thus we define the evolution function as concatenation of the agent's policy, the robot's model, and the environment.

Definition 4. *A robot is defined by its internal evolution function* $\psi : T \times M_I \times A \mapsto M_I$. *The external evolution function* $\phi : T \times M_I \times M_E \mapsto M$ *defines the behavior of the environment. Let* $m = (m_I, m_E) \in M_I \times M_E$ *and* $t \in T$. *Then the complete evolution function* Φ *of the environment is then defined as*

$$\Phi(t, m) = \phi\Big(t, \overbrace{\psi\Big(t, m_I, \underbrace{\pi(t, m)}_{\text{agent's policy}}\Big)}^{\text{robot's evolution function}}, m_E\Big). \tag{4.1}$$

Note, that an agent can only interact with the environment by changing the internal degrees of freedom. Knowledge over ϕ (and ψ) is essential to do meaningful things in an environment. This is the prior knowledge we need to incorporate in our agent to efficiently explore its environment. For robotic tasks, knowledge about the environment includes, for instance, knowledge about physics and kinematics. All these effects are incorporated in ϕ. Thus we need to know the behavior of ϕ well enough to manipulate external degrees of freedom by means of ϕ and ψ, e.g., pushing a door uses the knowledge of Newton's laws of motion to change the pose of the door—external degrees of freedom—by moving ones own joints—internal degrees of freedom.

4.2.2 Rigid World Assumption

Our world consists of many degrees of freedom which are very hard to model and manipulate. To focus our research on exploration, we limit the scenarios we analyse. For all following experiments we assume the world to consist only of rigid bodies connected by joints. The only change of state can be imposed by exerting forces to objects. We call this the *rigid world assumption*.

This intentionally excludes many scenarios from the research: folding towels, handling liquids, or squishing deformable objects are not possible within these limited scenarios. All these are interesting areas of research, also when conducting experiments on exploration. But since we want to focus

on the problems that arise from exploration itself, we keep the scenarios simpler to magnify those problems. The rigid world assumption still includes a wide variety of manipulation and exploration tasks: picking and placing of objects, handling furniture, or investigating the parameters of doors are some examples which are possible within the rigid world assumption.

By using the rigid world assumption the only degrees of freedom in an environment become the pose of objects, possibly constrained by joints. Thus the degrees of freedom are greatly reduced and it is computational tractable to model them within our probabilistic framework.

4.2.3 Exploration in Rigid Environments

As we have stated the exploration challenge is about learning a model of the environment. The behavior of the model is defined by ϕ, which can be potentially stochastic. Thus we want to minimize the expected information of the probability distribution of outcomes of ϕ given experience in the environment so far. This is what the active learning definition (see Eq. 2.16 and Eq. 2.17, page 46) is about:

$$(x_1, \ldots, x_n) = E\big[H\big[p(\phi(t,x) \mid x, D)\big]\big]_{x \in M, D}\,, \tag{4.2}$$

with t the current time. And if ϕ is a parameterized function, where we just want to learn the parameters

$$(x_1, \ldots, x_n) = E\big[H\big[p(\theta \mid \phi(t,x), x, D)\big]\big]_{x \in M, \phi(t,x), D}\,. \tag{4.3}$$

Thus we can see that exploration of external degrees of freedom is active learning on the evolutionary function of the environment. We use our internal degrees of freedom to maneuver to regions of the environment's state space which have the biggest expected information gain.

In Sec. 5 and Sec. 6 we will present various experiments that support the thesis that active learning can be used to drive exploration of external degrees of freedom in robotic tasks and environments in a meaningful way.

4.3 Physical Reasoning

When dealing with external degrees of freedom one particular challenge is to create configurations that are feasible. Consider the situation where a sample might be the position of an object. If the agent can sample all possible positions, many of them do not obey physical rules. For instance, an object might penetrate another one or might be floating in the air, not following the rule of gravity. This is an important difference to exploring internal degrees of freedom, where limits are either known before or easy to obtain.

Often pool-based sampling is used to avoid situations that are infeasible. However, we still need to create a pool of physically feasible samples. Thus we need to reason over the physical feasibility of samples. This process we call *physical reasoning*.

Closely connected to physical reasoning is the *controllability* of objects. While some samples might be physically feasible, we have no means to build theses samples, because we can not control some degrees of freedom. While internal degrees of freedom are always controllable, external degrees might not be changeable due to different reasons, the most common being that the agent is not in contact with an object and thus can not move it. When reasoning over samples to explore, they must be reachable as well as feasible.

Toussaint (2015) proposed a mathematical programming approach, called Logic-Geometric Programming, which alternates between symbolic planning and subsymbolic optimization. It uses symbolic actions together with trajectory planning methods to construct possible situations instead of sampling and deciding whether a situation is feasible or not. This has the advantage of generating the necessary actions to build a state alongside with that state. Another benefit is that it can only build reachable samples, because it inherently uses the abilities of the agent to generate samples.

We used a simpler approach in the experiments of Sec. 5, since Logic-Geometric Programming was not introduced yet. We use a physical simulator (*NVidia PhysX*) within the rejection criterion for an rejection sampling step. Rejection sampling is a method to generate samples from distributions we can normally not sample from. We sample from an adversarial distribution and reject samples proportionally to the likelihood, that this

sample would not appear in the desired distribution (Von Neumann 1951). For physical reasoning we generate samples of physical configurations of the environment from a very broad distribution. From that sample a scene in the physic simulator can be instantiated and the simulator is run. If no unexpected movement is noted, e.g., from penetrations or gravity, the situation is considered feasible. That way we can create a pool of feasible samples, which we can in turn rate by means of exploration measures.

Often it is the case that the given problem is relatively easy to parametrize, such that boundaries of the space are easier to define, as for example when uncovering joint dependency structures in Sec. 6.2. Then we can rely on those boundaries and do not need physical reasoning. When this opportunity exists it is worthwhile to build the parametrization, since physical reasoning techniques are computationally heavy.

The following two chapters will bring the theoretical background which we have presented in this chapter into practice. We have conducted several experiments that show how to explore external degrees of freedom. First we present work on unconstrained DOF (Sec. 5), followed by experiments on exploration of joints (Sec. 6.1) and their dependencies (Sec. 6.2).

5 Active Symbol Grounding

Parts of this section have already been published as:

J. Kulick, M. Toussaint, et al. (2013). "Active learning for teaching a robot grounded relational symbols". In: *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 1451–1457

Many real world tasks consist of manipulating the state of external degrees of freedom. Objects must be picked up and placed somewhere else, furniture must be opened to reach the content, or tools must be used to achieve a desired effect. To explore new environments it is important that robots interact with surrounding objects to learn how these objects work. They need to alter the state of the objects to find functional states and to learn how to operate mechanisms.

Complex object manipulation tasks require both motion generation on the geometric level as well as sequential composition and reasoning on more abstract, e.g., symbolic relational representations (Katz, Pyuro, et al. 2008; Lemaignan et al. 2011). In existing systems that incorporate both levels it is usually assumed that a set of grounded action symbols (motion/manipulation primitives) as well as state symbols are predefined (Beetz et al. 2010). In manipulation scenarios where the world is composed of objects and the state is naturally described by properties and relations of objects, relational representations based on conjunctions of logical predicates are well-suited (Džeroski et al. 2001). They generalize well over the actual object instance and transfer knowledge to similar objects and new environments. This led to the recent development of efficient methods for relational reinforcement learning (RL) and planning, which combine probabilistic learning and planning with relational representations (Lang and Toussaint 2010). However,

learning appropriate action and state symbols themselves remains a fundamental challenge.

Instead of predefining a set of symbols by the system designer it is desirable that a robot learns new symbols that enable it to reason about new situations. In this experiment we will use the following setting. A human teacher labels a few examples of situations that are descriptive for a new symbol and considered to be useful for a new task. From this the robot learns a first model of the symbol, which it actively aims to refine. The exploration in this setting is to move objects to interesting configurations and ask the teacher whether a given symbol holds or not. Learning the model of a symbol is called the symbol grounding problem. We will test the accuracy of the learned symbol groundings as well as their usefulness in reinforcement learning tasks.

5.1 The Symbol Grounding Problem

In any symbolic systems, symbols are meaningless syntactic elements in the beginning. They are connected to a semantic by a grounding. This grounding is specific to an agent and maps parts of states of it sensory input to a symbol, giving the symbol its meaning. To communicate with these symbols it is necessary that agents share the meanings of the used symbols. That does not mean they need to have exactly the same mapping. But for the situation that is communicated the mapping from both communication partners should map to the same symbols. The image of that mapping should be sufficiently overlapping. The symbol grounding problem is to find such a mapping.

There exist two famous philosophical papers about the symbol grounding problem. The first is from Searle (1980), which introduces the problem as the "Chinese Room" problem. It is an intellectual game putting the reader in a situation where she communicates with the world only over a terminal with Chinese symbols (where she normally does not speak Chinese) and a Chinese-Chinese dictionary. Searle now asks if the reader will ever be able to understand the meaning of the symbols and draws an analogy to the

symbol grounding in an artificial intelligence, which can also never truly understand the meaning of the symbols it subsumes. Harnad (1990, §8 – §10) extends that line of reason and asks:

> Suppose you had to learn Chinese as a first language and the only source of information you had was a Chinese/Chinese dictionary! This is more like the actual task faced by a purely symbolic model of the mind: How can you ever get off the symbol/symbol merry-go-round? How is symbol meaning to be grounded in something other than just more meaningless symbols? This is the symbol grounding problem.

Here he questions the possibility of a symbolic AI at large. Our approach will not disprove Harnad's intuition and argument. Instead we will ground the symbols needed for a relational reinforcement learning task in sub-symbolic sensory input in an active manner. We do not assume a fully symbolic AI, but add a sub-symbolic state, based on sensory input – an embodiment. The symbols are directly given by a human teacher, the robot should only learn its grounding. This process is sometimes called *symbol anchoring* (Coradeschi and Saffiotti 2000, 2003) to contrast it with the symbol grounding problem. As such symbol anchoring is classification. Each symbol is connected to a classifier, that determines whether a given symbol holds for a state or not.

For learning or anchoring symbols from a human teacher we need intelligent exploration to i) ask meaningful questions to the human teacher and ii) speed up learning to avoid annoyance of the human teacher. Cakmak and Thomaz (2012) showed in their research that asking good questions is crucial for human-robot interfaces and that using active learning methods is a suitable means to that end.

5.2 Relational Reinforcement Learning

In the following experiments the symbol groundings are used within the framework of relational reinforcement learning. We briefly describe this

framework here.

Reinforcement Learning is a way to find good policies in MDPs. As described before (see Sec. 2.4.2) an MDP consists of states, actions, transition function and reward function. While in simple environments states and actions can be enumerated, state representations becomes more complicated for more realistic settings. A more sophisticated approach is to use a so called relational state and action space (De Raedt 1997). The resulting MDP is called a relational MDP and reinforcement learning algorithms on such relational MDPs are called relational reinforcement learning, accordingly (Džeroski et al. 2001).

A relational state is compounded of objects O, predicates P and functions F. It is the conjunction of (potentially negated) predicates and functions over the objects, which are true for a given state. It can be seen an enumeration of all attributes of the state and relations between objects in the state (hence the name). For a detailed description see Pasula et al. (2007).

To define the transition function we need to define the outcome of actions. For the actions in the experiment we use noisy, indeterministic, deictic (NID-) rules as defined by Pasula et al. (2007). An example of these rules is:

$$\begin{aligned} &pickup(X) : \{Y : block(Y)\} \\ &block(X), on(X, Y) \\ &\quad \rightarrow \begin{cases} 0.6 : \neg on(X, Y), inhand(X) \\ 0.3 : \text{no change} \\ 0.1 : \text{noise} \end{cases} \end{aligned} \tag{5.1}$$

A NID-rule describes two things: the outcome of an action and constraints when an action can be applied. The rule first shows the name of the action and the list of parameters in parentheses ($pickup(X)$). All parameters are variables. Variables can refer to any object in the scene. This property allows NID rules to generalize over object instances. A rule applies to all objects if they satisfy the given constraints. Variables are written with upper letters, in this case X and Y. When applying the rule to an actual state,

i.e., performing the action, these variables have to be assigned to an actual object within the constraints of the rule. This assignment is ambiguously called *grounding* in the relational reinforcement learning literature, and not to be confused with the symbol grounding. If no grounding is possible the action can not be performed. We will shortly explain how to ground the variables but first describe the constraints of the rule.

The constraints of the rule are split in two parts: the deictic and the context block. After the colon follows the list of deictic constraints in curly brackets. Deictic variables are variables important to describe the context of the rule, although they do not appear in the parameters of the rule. The deictic constraints define the constraints on the deictic variables. The context block ($block(X), on(X, Y)$) follows in the next line. It describes contextual constraints that must be satisfied to apply the rule.

A grounding assigns actual objects to all variables of the rule. A grounding is legal if all predicates in the deictic and context block evaluate to true. In this example the function parameter X must be assigned to an object that is a block ($block(X)$ must be true), which must be on another block Y ($on(X, Y)$ and $block(Y)$ must be true).

Then follows the list of possible outcomes and their stochastic distribution. Each line shows first the probability of the outcome and then the outcome. The outcome includes all predicates that become true or false after the action has performed. The rules can have a noise outcome, which captures all outcomes not precisely modeled by the action. An action in real world can have many unpredictable or highly unlikely outcomes. The noise outcome subsumes those outcomes to generalize better. In the given rule everything around the block the agent picked up might break down because the block was somewhere in the middle of a structure. Many predicates in the state would change in an unpredictable manner.

Pasula et al. (2007) showed how these rules can be learned from experiences and Lang and Toussaint (2010) showed how planning can be done in such relational domains with their PRADA algorithm. They also investigated what exploration in such relational domains could look like (Lang, Toussaint, and Kersting 2012). The relational symbols, predicates and functions, however, are always handcrafted. We now show that those symbols

can also be learned and how active learning as exploration principle can guide the selection of training data.

5.3 Learning Symbol Groundings

We assume that a scene s is composed of objects. For each object i we have a geometric feature vector $x_i \in \mathbb{R}^m$ describing geometric or other physical properties of the object (like radius, height, absolute position, color). Likewise, for each pair of objects (ij) we have features $x_{ij} \in \mathbb{R}^M$ describing the objects' geometric relation (like the difference $x_i - x_j$, distance, relative position, etc.).

We define a grounded relational symbol $\sigma = (\mathfrak{p}, f)$ as a tuple of a first order predicate $\mathfrak{p}$ and a discriminative function f grounding the symbol. f determines the probability that the predicate $\mathfrak{p}$ holds for objects $\mathbf{o} = (o_i, o_j)$ given their features $x_\mathbf{o} = (x_i, x_j, x_{i,j})^T$ (or $\mathbf{o} = o_i$ and $x_\mathbf{o} = x_i$ for the unary case) in state s,

$$p(\mathfrak{p}(\mathbf{o}) \mid x_\mathbf{o}) = \mathrm{sig}(f(x_\mathbf{o})). \tag{5.2}$$

where sig is the logistic function $\mathrm{sig}(z) = \frac{1}{e^{-z}+1}$. For each continuous state s we can now define a symbolic state $t \in \{0, 1\}^v$. For all objects and combinations of objects $\mathbf{o}$ and symbols $\sigma = (\mathfrak{p}, f)$, t has an entry t_i which is 1 if and only if $\mathrm{sig}(f(x_\mathbf{o})) > 0.5$ and 0 otherwise. t describes which symbolic predicates are considered to be true in state s.

Learning a discriminative function is the classical task for classification in machine learning. We use Gaussian Process Classification (Rasmussen and Williams 2006) to learn the discriminative function f. Gaussian Process Classification (GPC) has the benefit of estimating an uncertainty about every prediction in addition to the prediction itself. In fact it learns a complete distribution, namely a Gaussian, for all input data samples. We will use that for the active learning strategy. The active learning algorithm can query an oracle for $y_{\mathfrak{p},\mathbf{o}}$, which holds the truth value of whether $\mathfrak{p}$ holds for objects $\mathbf{o}$. For the sake of clarity we drop the use of the subscripts and assume that all

following equations refer to the same predicate $\mathfrak{p}$ and set of objects $\mathbf{o}$. Thus in the remainder of this section x means $x_{\mathbf{o}}$ and y means $y_{\mathfrak{p},\mathbf{o}}$.

A benefit of using a GPC is, that computing the entropy of a Gaussian is easy and so we can define the global information gain as the integral of the entropy over the whole space of the classifier, comparable to what Cohn, Ghahramani, et al. (1996) suggested.

$$\mathcal{H}(D) := \int_{\mathbf{x}} H\left[p(y \mid x, D)\right] . \tag{5.3}$$

For active learning, the system chooses the sample $x^*_{\mathcal{H}}$ with

$$x^*_{\mathcal{H}} = \underset{x'}{\operatorname{argmax}} \;\; \mathcal{H}(D) - E\big[\mathcal{H}(D \cup \{(x', y')\})\big]_{y'} . \tag{5.4}$$

Typically, the integral in equation (5.3) as well as the maximization to gain $x^*_{\mathcal{H}}$ cannot be computed analytically. To approximate the integral (5.3) we perform a Monte-Carlo integration, by sampling k physically feasible reference configurations. The optimization in (5.4) is approximated using a pool of physical feasible situations, where the best configuration is queried (see Sec. 5.4).

While this approximation leads to good results it is computationally expensive, since $H\left[p(y \mid x, D)\right]$ has to be computed at all reference points for all tested samples. This is especially problematic in situations where time for an oracle is considered expensive, such as in experiments with a human teacher.

Therefore, we compare this objective to a local estimation of the uncertainty. For this purpose we use the entropy of the predictive distribution directly as optimization criterion and the robot chooses the sample x^*_{Loc} with

$$x^*_{Loc} = \underset{x'}{\operatorname{argmax}} \;\; H\left[p(y \mid x', D)\right] . \tag{5.5}$$

This criterion scales well to high dimensional spaces, since no Monte-Carlo integration (with very high k) is needed to approximate an integral. We optimize again via the pooling approach.

5.4 Physical Reasoning: Sampling Physically Feasible and Informative Situations

Given a classifier f we can estimate the reduction of predictive uncertainty for any new input x. However, the inputs x to a classifier have to be features of a real physical situations. Not every feature vector can be generated by a physically feasible situation and the feature map from sensory inputs to input vectors cannot be inverted to retrieve a physical situation that generates a given feature. The physical feasibility can be viewed as a structural constraint of the classifier's input domain which is not present in typical active learning approaches and which we have to explicitly take into account. Thus we have to only sample from this subspace during learning the groundings. To cover this subspace we use a physical simulator to generate a large set of physically feasible situations, which are steady. The robot now can generate the features of the situations from the simulator and compute the expected reduction of the predictive uncertainty from this simulated situation. I have discussed the problem in detail in Sec. 4.3.

This approach is a form of pool-based active learning (see Sec. 2.1). By using the simulator we do not need to actually build the situations in real world, but can generate them fast in simulation and still have a notion of steadiness and feasibility. The actual query is then evaluated in real-world.

5.5 Robot manipulation to generate informative situations

To interact with a human teacher in the given active learning scenario the agent needs to manipulate objects in the real world and literally generate informative situations (see Fig. 5.1). Because we compute the features from simulated situations we have access to the positions of the objects and build the most interesting scene in real world. But since the sensorimotor loop of a robot is subject of many sources of noise, we are not able to precisely generate the same situation as we had in the simulation. While the normal pool based active learning assumes that the samples can directly be labeled by the

teacher, the robot-human scenario introduces another step in the processing and the labeled sample x' is different from the originally generated sample x. It does not necessarily lead to the same information gain. While our system uses GPC, which explicitly handles noisy observations and is able to generalize to close by samples, we do not handle the difference between the desired and the actual sample. We do, however, evaluate the current situation and label the actual sample and not blindly the desired one.

To place objects into desired poses we generate robot pick-and-place motions using standard robot trajectory optimization methods. Concerning the object picking, the pre-grasp (hand and finger poses before closing fingers) and reaching trajectory to the pre-grasp are jointly optimized; the objective function includes cost terms for trajectory length and collision and joint limit avoidance during the trajectory as well as cost terms for the relative wrist-to-object pose, finger tips to surface distances, and opposedness of finger tips of the final pre-grasp. The optimization is done with a fast sequential 2nd order optimizer that exploits the sequential structure of the trajectory in the (banded) matrix inversion. The placing trajectories are similarly optimized, with cost terms reflecting the desired final placement of the object. See Toussaint et al. (2010) for details.

5.6 Evaluation of Active Symbol Grounding

We investigate whether (a) our active learning method permits faster learning than passive learning, (b) our formal model of symbol grounding allows to ground relational symbols in the physical world, and (c) the learned symbols enable abstract robot learning and reasoning, opening the door to the techniques developed in symbolic artificial intelligence, such as relational RL planners.

5.6.1 Experimental Setup

We examine our symbol learning approach in a robot manipulation scenario where a robot manipulates balls, cubes and cylinders on a table. The robot can execute three motor primitives: *closeHandAround(X)* grabs an

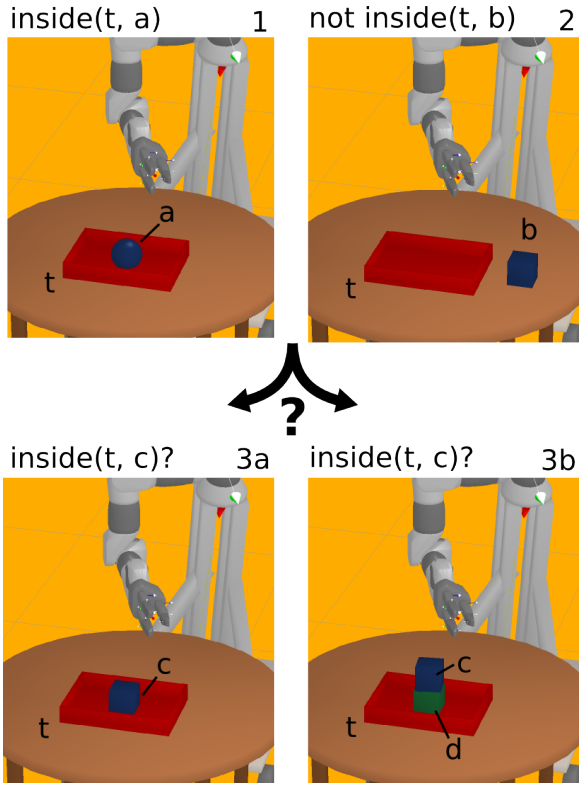


Figure 5.1: In active learning of grounded relational symbols, the robot generates situations in which it is uncertain about the symbol grounding. After having seen the examples in (1) and (2), the robot can decide whether it wants to see (3a) or (3b). An actively learning robot takes its current knowledge into account and prefers to see the more novel (3b). Figure from Kulick, Toussaint, et al. (2013).

object; *openHandOver(X)* opens the robot's hand above some other object and *openHandAt(X)* opens the hand at a specific point in space. While opening the hand always succeeds, grabbing objects might fail (both in simulation and real world).

The object observations of the robot are described by continuous feature vectors x_i comprising the pose of the object (position of the center and orientation of the object) and the size of object o_i. The object features are used to construct features $x_{ij} = \phi(x_i, x_j)$ describing the relationship between objects, namely the distance and size differences and the sine of the angle between the main axes of the objects.

We performed experiments both in simulation (see Fig. 5.4) and on a real robot (see Fig. 5.3). In both scenarios the robot consists of a Schunk Light Weight arm with 7 DoF, a Schunk Dextrous Hand with 7 DoF, 6×14 tactile arrays on each of the 6 finger segments and a Microsoft Kinect depth sensor. A point cloud based vision system directly computes the positions and sizes of the objects.

5.6.2 Experiment 1: Quantitative results in simulation

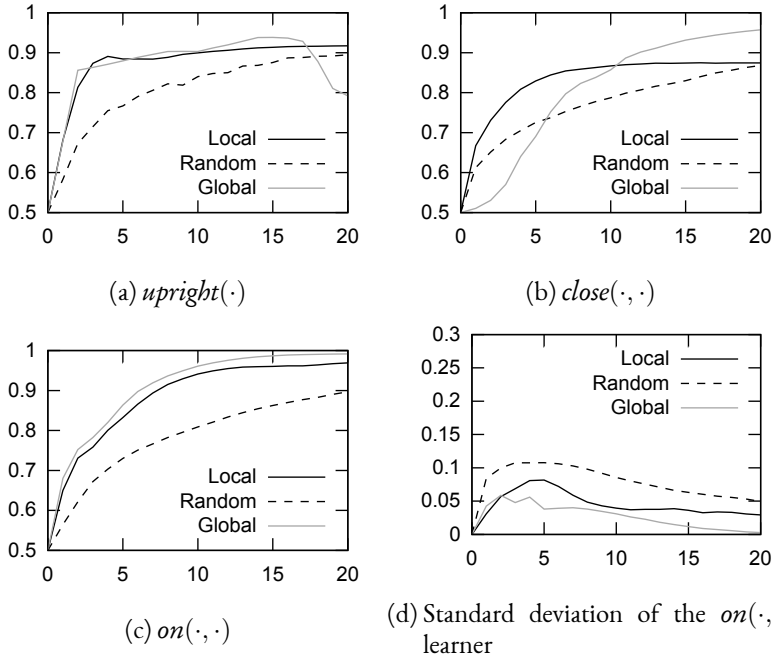
Specific setup In the first experiment we let an agent learn unary and binary spatial relations of objects in a simulated environment. For each symbol the agent is provided with an example where the relation underlying the symbol holds.

The unary predicate, the robot should learn, is *upright(X)* while the binary predicates in this experiment are *on(X,Y)* and *close(X,Y)*.

The meaning of the symbols should be the following:

- $upright(X)$, the object's main axis a is within ε degrees of the z-axis of the world coordinate frame
- $on(X, Y)$, the distance Δ_{pos_z} along the world coordinate frame's z-axis of the center positions of both objects is

$$0 < \Delta_{pos_z} < \frac{size_z(X) + size_z(Y)}{2} + \varepsilon, \tag{5.6}$$



(a) *upright*$(\cdot)$

(b) *close*$(\cdot,\cdot)$

(c) *on*$(\cdot,\cdot)$

(d) Standard deviation of the *on*$(\cdot,\cdot)$ learner

Figure 5.2: *Experiment 1: Comparison of the proposed active learning approach with passive learning.* The results for learning unary and binary grounded relational symbols in simulation show that active learning outperforms passive learning. (d) shows the deviation of the learner *not* the deviation of the mean estimator. This deviation is very small due to the high number of experiments ($n = 1000$). Figures from Kulick, Toussaint, et al. (2013).

(with $size_z(X)$ being the size of object X in z-direction) the distance in x and y direction is smaller than ε,

- $close(X, Y)$, the distance of the closest surface points of object X and Y is smaller than ε.

The agent should now learn a predictive model for the given symbol by querying an teacher with new situations. The teacher in these experiments is a handcrafted oracle, that computes whether a relation holds or not.[1] After each query the agent can recompute its model and then choose the next point. Each test is performed 1000 times for both active learning criteria (i.e., the local and the global) and a passive learner (i.e., a learner choosing random samples).

To evaluate the learning rate the classifier is tested with 5000 random samples after each query and the classification rate is computed.

Results and Discussion In figure 5.2 we show the results of these tests and the standard deviation of one learner. The standard deviation of the mean estimator is very small (< 0.01 for all points), due to the high number of experiments. Hence we do not show it in the graphs.

It can be seen that in all cases the active learning approach outperforms the passive learning. Also the standard deviation is smaller. When comparing the different information gain measurements, one can see that the learning progress is slower during the start of the *close* learner when using the global criterion. Qualitative experiments suggest that this is an artifact of the small size of the reference set at which the variance is evaluated.

Another interesting investigation is the learning decay of the *upright* learner after 17 samples. The relation is only depending on the angle of the object. Since we use the sine of the angle, we only have a usable feature range of $[-1, 1]$ to learn this relation. The global criterion starts over-fitting here, since the Gaussian process has a kernel that is too wide to perform better.

[1] To acquire quantitative data we do these experiments without a human in the loop, although similar results can be expected, since the oracle simply provides fast access to labeled data.

Overall the local criterion leads to similar results, although the global criterion outperforms it in all experiments. However, the computing time is much shorter for the local criterion and one might choose it over the global one, if computation time is expensive.

5.6.3 Experiment 2: Real-world robot

Specific Setup To show the possibility of real world applications we performed a complete active learning procedure on a real-world robot as specified above with a human in the loop. The robot is – similar to the first experiment – in the begining provided with a situation where the relation underlying the symbol holds. It then queries the human teacher by actively manipulating the objects with pick and place actions to generate an interesting situation. After generating the situation the robot senses the real position of the objects and computes the features. It then updates its classificator with the new data. By generating data with object manipulation, noise is introduced in the sample generating process, since grasping and putting objects does not always work perfectly on a real robot. The trajectory generator also tries to avoid collisions, hence the real positions differ from the desired. The Kinect perception system introduces another source of noise.

After each query the agent is tested with 5000 simulated situations to test its learning performance.

Results and Discussion The classification rate of an example trial is shown in figure 5.3.

The experiment shows that learning on an abstract symbolic level can be done by interaction with a human teacher. The method is robust against noise in the sample generation, such as control noise, although the learning rate decreases.

We discovered that the rather big embodiment led to problems while generating samples with cylinders very close to each other. The Schunk Dexterous Hand was not able to put the cylinders directly next to each other and the robot therefore tried to generate a very similar sample several times without success. Thus the learning rate decreases after the 7th sample and

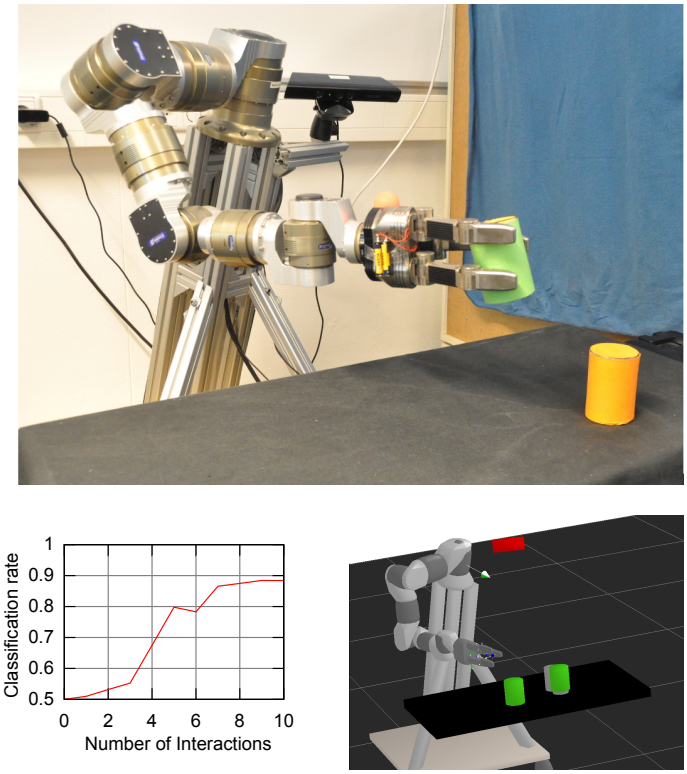


Figure 5.3: *Experiment 2: Learning on a real world robot.* A qualitative example of learning the symbol *on* with a real-world robot. The robot builds interesting situations (top) according to a sample it generated with active learning (bottom right). After each query it integrates the result to improve its understanding of the predicate. The classificaton rate is shown at the bottom left. Figure from Kulick, Toussaint, et al. (2013).

the classification rate improves less, because the optimal sample could not be built. This is an interesting aspect of physical reasoning. Since we did not include the actual process of positioning the object with the given embodiment, we included samples in the pool that were not physically feasible, i.e., we did not specify our sampling distribution correctly.

5.6.4 Experiment 3: Full-fledged relational RL

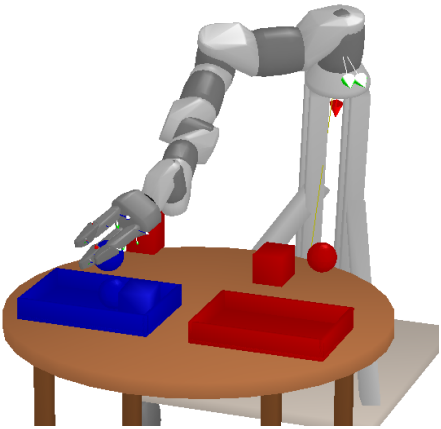


Figure 5.4: A simulated robot plays with boxes and balls scattered on a table. The task is to put balls in the blue tray and boxes in the red tray, regardless of their color. Figure from Kulick, Toussaint, et al. (2013).

The symbols we learned in the previous experiments have not been used for any task so far. We learn such relational symbols to use them in relational Reinforcement Learning scenarios. While relational symbols in the literature are hand-crafted and thus not noisy, learned symbols might include wrong information. In this experiment we tested whether the learned symbols give us the ability to plan in complex real-world scenarios.

Specific Setup The task for the robot is to clean up a table with scattered objects. It should put objects of different shape in different trays, until no objects are left on the table (see Fig. 5.4).

To measure the performance the robot receives a reward for every object within the correct tray. To make faster solutions more profitable the reward is discounted over time, such that every further step decreases the reward.

The actual experiment consists of three stages:

At the first stage we learned the relation *inside(X, Y)*. The symbol should denote whether an object lies in a tray. This relation is the most important one for the task, since it can show the robot how it performs. Note that the robot does not receive reward from the real world, but from his *belief state*, i.e., the state might include wrong symbols or miss symbols, since the learned symbol might predict the state wrongly.

The second stage is to learn the probabilistic transition model of the action set. For this purpose the robot performs random actions. From the perceived states, which may include the grounded symbol, it learns noisy, indeterministic, deictic rules (NID rules) from the sequences as described by Pasula et al. 2007 (see Sec. 5.2). They are used to provide a relational planner with a stochastic model of action outcomes.

Eventually, we use the PRADA planning algorithm from Lang and Toussaint 2010 to actually plan action sequences and record the reward earned. Here the real reward is computed, not the reward of the belief state, to actually evaluate the real behavior.

We performed the experiment six times. Three times the symbol was learned with the local criterion and three times it was learned by a passive learner. Each experiment consists of 30 complete runs.

Results and Discussion Fig. 5.5 shows the results of the experiments. It is apparent that learning grounded symbols in an active manner leads to significantly higher rewards than passive learning. Adding more samples improves both, the active and the passive approach, but the gain is bigger when using actively chosen queries.

The difference to the optimal behavior can partly be explained by non

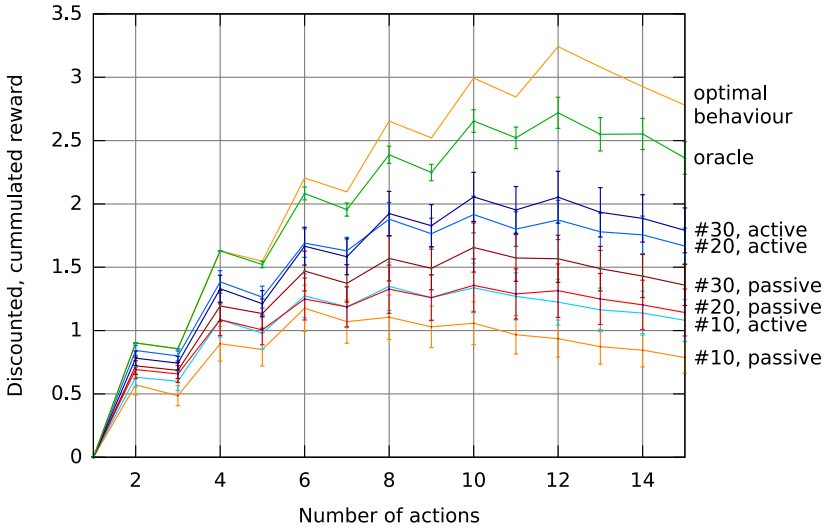


Figure 5.5: *Experiment 3: Full relational reinforcement learning scenario.* The active learner requires significantly fewer samples to learn grounded relational symbols which permit an autonomous agent to plan for high-reward states. The step like shape of the curves are artifacts from the task: grasping an object can not lead to reward, but only putting things into trays. Thus at best only every second action can generate reward. Figure from Kulick, Toussaint, et al. (2013).

optimal behavior of the planner and noisy outcomes of actions (e.g. a ball falling outside a tray), but also partly by non optimal grounding. The behavior with optimal grounding (done by the handcrafted oracle routine) suffers from the noisy control, but outperforms the learned symbol. The action outcomes of the optimal behavior shown are always perfect. It must therefore be considered merely theoretical and has no deviation.

5.7 Discussion

To enable robots to reason on an abstract level and generalize, symbolic relational representations are a well suited, but handcrafting the groundings is inflexible, time consuming and needs expertise in programming. Teaching robots new symbols in an easy way is thus desirable. We propose an active learning approach to teach robots the groundings of relational symbols with a human teacher. The method is shown to learn spatial relations with few training samples. Hence, it is applicable in interaction with human beings where time is considered expensive. Using active learning methods sped up the process. We have also successfully used the learned symbols in solving a task. Unsurprisingly the agent performed better with better grounded symbols. We showed that active exploration led to better symbol groundings and thus better performance in solving the clean up task.

Although this is itself an important achievement the insight we gain from the experiments goes further. It is active learning on external degrees of freedom for the first time. The robot has to manipulate the state of its surrounding environment to sample new queries. Thus a reasoning about the physical feasibility of proposed states is needed. The proposed physical reasoning approach using a physical simulator for rejection sampling has shown to work well despite its simplicity. However, we have also seen its limitations. It ignores the embodiment of the robot and generates samples the robot is not able to build. Simulating the actions from the robot would solve this problem, but would further increase the computational costs. It is also questionable if the simple rejection sampling approach scales to more complex scenarios. While in this scenario many samples could be accepted,

more complex physical configurations are more likely to be unstable and will be rejected. Thus we will need to sample a huge amount of configurations to generate a pool of sufficient size.

The experiments show the complete pipeline of an autonomously exploring robot in a spotlight. While all parts of that pipeline are present, many of them are tailored to the task at hand. For example perception and action primitives are relatively strong bound to the pick and place tasks. Especially, the symbols that should be learned were given from a human teacher. Thus the prior knowledge of the environment—which symbols are necessary—was out-sourced to a human. The following experiments will focus on how we can encode such knowledge to let the robot decide more autonomously on what to explore.

6 Joint Exploration

Parts of this chapter have already been published as:

S. Otte et al. (2014). "Entropy Based Strategies for Physical Exploration of the Environment's Degrees of Freedom". In: *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 615–622

J. Kulick et al. (2015a). "Active Exploration of Joint Dependency Structures". In: *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pp. 2598–2604

In (Otte et al. 2014) I contributed the belief representation of the existence of random variables and was involved in the design of the experiments. I will explicitly refer to the paper for the additional parts.

When considering the rigid world assumption (see Sec. 4.2.2), the most interesting degrees of freedom are external joints. They are not directly controllable by the agent and still are degrees of freedom that do not model simply the pose of an object freely movable in space, but they are constrained by the joint parameters such as axis and limits. Joints are designed by humans to structure the environment. They are where functional benefits arise from changing the state of degrees of freedom in the world: turning handles to open windows, pushing buttons to switch on or off the light, or turning a key to lock a door.

In this chapter we show scenarios where a robot needs to explore environments with joints in various places. First, we show how to explore the existence of joints, which entails the difficulty to model existence in a proba-

bilistic model. Later we focus on structures where one joint changes the behavior of another joint, e.g., a handle that locks or opens a window. We call such structures *joint dependency structures*. Robots need to uncover such dependency structures to solve a wide variety of tasks.

6.1 Exploring the Existence of Joints

The exploration challenge states that a robot entering a new environment should start building a world model. With the rigid world assumption, that is to collect information about (a) the objects in the world, and (b) the joints connecting them.

In this section we will investigate the following scenario: A robot enters a new environment with several objects. It should explore if and how these objects are connected. Additionally to discovering the degrees of freedom, it should model their behavior by finding the parameters of the joint models.

For this task, we assume that the robot has access to a segmentation of the scene into rigid bodies (objects). The robot can perceive (without physical interaction) their shapes and poses, but the kinematic structure and the dynamic properties of the joints are initially unknown.

We propose the following minimum set of properties that a robot should know to accomplish generic manipulation tasks in a rigid-body world:

- *Object type:* An object can be either *movable* or *static*.
- *Joint type:* If there exists a joint between two rigid bodies, the joint can be either *rotational* or *prismatic*.
- *Joint limits:* Each joint has mechanical limits, captured by a minimum and maximum value of the joint variable.
- *Friction coefficient:* Friction slows down the movement of a shape attached to a joint. Although different physical quantities act together (friction forces, inertia etc.), we summarize them in a single parameter.

We therefore define random variables for each property of an object i: object type O^i, the joint type J^i, and we group the joint properties—upper and lower joint limit and friction coefficient—in one RV for each joint type (θ_r^i for the rotational and θ_p^i for the prismatic joint).

6.1.1 Belief Representation

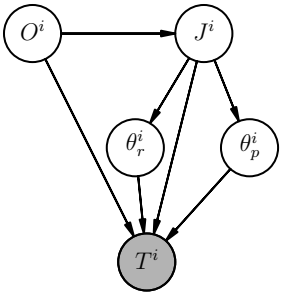


Figure 6.1: Graphical model that represents the belief associated to a single object i. Object type O^i, joint type J^i, joint parameters for the rotational joint θ_r^i, joint parameters for the prismatic joint θ_p^i, the observed trajectory T^i of the object. Figure from Otte et al. (2014).

We define a structured belief over the properties for each object (see graphical model in Fig. 6.1). Each object in the scene graph is augmented with such a joint distribution.

Let the index i denote the object i. In the following, i is omitted for readability. Let O and J be discrete random variables and θ_r and θ_p be sets of continuous random variables. The *object type* O indicates if the object is *movable* or *static*. Its discrete probability distribution is calculated from the number of times an object is observed to be *movable* or *static*. J represents the *joint type* which can be *rotational* or *prismatic*. To represent the dependent

nature of the world, we also add a pseudo value *nil*, which we will explain in the next section. The probability distribution is estimated by counting the number of observations for each value. θ_r and θ_p are the parameters of the joints, consisting of upper and lower joint limit and a friction factor. These variables are assumed to be Gaussian distributed. Dependent on all theses variables is the trajectory T of the object. The trajectory is observable.

6.1.2 The nil Value

In exploration scenarios often the task itself is to find out about new things. Thus it is inherently important to model the uncertainty over the existence of objects. When exploring degrees of freedom in a rigid world, this applies as well to joints. When starting an exploration, we might not know if two objects are connected by a joint and therefore want to discover whether a joint exists or not. Thus an agent only has a belief over the existence of a random variable. To apply exploration strategies, we need to model this existence belief properly.

Normally, the existence of random variables is not in question. During inference no random variables appear or disappear and the structure of the probabilistic model remain constant. But consider a case where we have a belief over certain aspects of an object, e.g., its position, size and weight. In a probabilistic formulation those aspects would typically become random variables. Now, if we are not certain if the object really exists, we would also capture this fact by a random variable. But now the existence of the random variables capturing the properties of an object depend on the actual state of the random variable capturing the existence of said object. Only if the object exists, does it have a position, size, and weight. And when considering exploration based on entropy: how can we compute the entropy of distributions that might not exist?

Not much research has dealt with the problem of existence in probabilistic models. Ngo et al. (2013) used the probabilistic outcomes of NID rules (as described by Pasula et al. (2007), see Sec. 5.2) to model objects that can come into existence and plan in such domains. A deterministic planner that can handle uncertainty of the objects in existence is from Srivastava et al.

(2009) using three valued logic (Sagiv et al. 2002). A more general approach is the Bayesian Logic (BLOG) as described by Milch et al. (2007). They all, however, do not deal with the actual distributions nor with entropies over possibly non-existent distributions. However, we need to compute the entropy to measure the expected information gain if want to explore such environments.

We propose the following solution: We introduce a pseudo value for distributions which we call *nil* value. It captures the situation when the existence of the variable is unknown. Consider the simplified case of object type O and the dependent joint type J. If we have a probability $p(O = \text{static}) = 0.3$, we only have a 0.7 belief that there is a joint type involved in the model. So we have a deterministic dependency between O and J, stating that the marginal is $p(J = \text{nil}) = 0.3$. We call J *nil-dependent* on O being static. To avoid corrupting the entropy by possible existence or non-existence of a RV we have to handle the *nil* value case for discrete and continuous RV separately.

Definition 5. *If X is a discrete random variable nil-dependent on another discrete random variable Y having the value v the conditional probability distribution is*

$$p(X = \text{nil} \mid Y) = \begin{cases} 1 \text{ if } Y = v \\ 0 \text{ else} \end{cases} . \tag{6.1}$$

With these pseudo values, we can compute meaningful entropies over the distributions. Using this formalization the random variable X will have minimal entropy if Y assumes the value v. In the joint example this means that if we encounter O to be static the distribution $p(J \mid O)$ becomes one for the *nil* value and zero everywhere else, thus having a entropy of zero.

The same problem of uncertainty over existence arises with continuous distributions, but it is not possible to inject a discrete pseudo-value here. We therefore use a Dirac delta as pseudo-value. The Dirac delta function has the minimal entropy (i.e., $-\infty$). Intuitively it can be thought of as an infinitesimal *nil* value. The Dirac delta simply formalizes in a continuous space that everything is known about a variable and no uncertainty is in play. Thus if

we would know that a random variable does not exist, we could set its distribution to a Dirac delta, because there is no uncertainty over non-existent random variables. The only question would be where to position the Dirac delta, since every actual value makes equally little sense—a non-existent variable does not have any value. We could introduce a pseudo *nil* position, but it is unclear where this position should be and how a marginal or a expectation over the existence of the random variable would then be computed. But since we use this pseudo value only for computing the entropy, the actual position of the Dirac delta is not important as long as it does not bias the marginal distribution. Since the entropy of the nil-depended distribution should decrease with increasing probability $p(Y = v)$, we center the Dirac delta at the mean of the nil-dependent distribution conditioned on $Y \neq v$. Thus we do not introduce any bias by the pseudo value.

Definition 6. *If X is a continuous random variable nil-dependent on a discrete random variable Y having the value v, the conditional probability distribution is given by:*

$$p(X \mid Y = v) = \delta_{E[p(X|Y \neq v)]_X}(X), \tag{6.2}$$

with $\delta_x(\cdot)$ being the Dirac delta function at position x.

Although this is the clear translation of the discrete to the continuous case, it is not possible to calculate the marginal of a random variable if a Dirac delta function is involved in a mixture. We therefore approximate the Dirac with a very narrow Gaussian distribution. This is particularly useful since the joint parameters naturally have Gaussian distributions and the entropies become easily comparable.

6.1.3 Calculating the Entropy

Now, for each object i and its parameters $O^i, J^i, \theta_p^i, \theta_r^i$ we could calculate the entropy assuming independence of its parameters

$$H\left[O^i, J^i, \theta_r^i, \theta_p^i\right] = H\left[O^i\right] + H\left[J^i\right] + H\left[\theta_p^i\right] + H\left[\theta_r^i\right]. \tag{6.3}$$

However, the distributions are not independent. We define an entropy measure, $\hat{H}$, which weights the entropy of each distribution according to its likelihood and takes the nil-dependence into account:

$$\hat{H}\left[O^i, J^i, \theta_r^i, \theta_p^i\right] = H\left[O^i\right] \tag{6.4}$$

$$+ p(O^i = \text{movable})\, H\left[J^i\right] \tag{6.5}$$

$$+ p(J^i = \text{prismatic})\, H\left[\theta_p^i\right] \tag{6.6}$$

$$+ p(J^i = \text{rotational})\, H\left[\theta_r^i\right]. \tag{6.7}$$

Note that $\hat{H}$ can be computed analytically because each distribution is either categorical or Gaussian.

6.1.4 Experimental setup

To test if the modeling of joint existence leads to useful exploration strategies, we conduct two experiments with the structured belief. The robot enters an unknown environment, where the task of the robot is to learn the joint positions and existence in the new environment. It can perceive the position of objects in the world. Once it interacts with an object it perceives a 3d trajectory of the object. The object type O^i is updated with a *static* observation if there was no movement, or with a *moving* observation if movement was observed. For learning the joint type J^i and pose, we use the *articulation library* by Sturm et al. (2011). The distribution for the joint type J^i is updated accordingly.

Because the articulation library does not infer joint properties such as joint limits and a friction coefficient, we employ a graphical model to infer these parameters based on the physical laws of motion. We project the perceived 3d trajectories to the 1d joint space (i.e., rotation angle or position of the prismatic joint) and model the motion there. For inference we used the Markov Chain Monte-Carlo python library PyMC (Patil et al. 2010). For details of the parameter estimation see our paper (Otte et al. 2014).

We conduct two experiments: A synthetic experiment in a toy scenario and a physically simulated experiment in a realistic robotic simulation. Both experiments are equal in all parts, but the actuation of objects. While in the synthetic experiment forces can directly be applied to objects, in the physical simulation experiment actual trajectories of a robot are computed, that in turn apply forces to objects. Through the simulation more sources of noise are introduced: The controller does not perfectly follow the trajectories, and the physics simulation models some dynamics wrongly and has numerical instabilities.

The action that is performed is a simple "push the object" action. We assume that a push is strong enough to move any object that is not static (and that we do not by chance exactly push in the direction of an axis etc.). This is implemented as a fixed force applied to the object at a random surface point towards the center of mass of the object. The set of actions is therefore equal to the set of objects and the terms are used interchangeably.

Based on this setup we test four exploration strategies:

1. *Random:* The agent chooses one of the objects in the world uniformly at random. This simple strategy serves as a baseline for our evaluation.

2. *Round robin:* The agent selects objects sequentially. Although this strategy seems to be a very simple, one should note that for certain worlds—such as worlds that only consist of the same objects and thus return the same reward/reduction of entropy—the round robin strategy yields optimal results.

3. *Expected change of entropy:* The agent computes the expected change of entropy for each object in the belief and chooses the object that minimizes this criterion.

4. *Max Entropy:* The agent computes the current entropy for each object. It chooses the one with the highest entropy. The assumption is that objects with high entropy are not yet modeled properly and thus a large reduction in entropy can be expected from exploring it.

This heuristic is successful in many kinds of problems, however, it pays unjustified attention to actions with random outcome.

6.1.5 Synthetic Experiment

Our first set of experiments is in a purely synthetic scenario to show the basic characteristics of a set of strategies. A set of two objects is given. One of them is static, one of them is attached to the static world by a prismatic joint. This is the minimal example that can lead to interesting behavior. We expect an agent to focus on the movable object and ignore the static object after only very few interactions. In the scenario the agent—a purely algorithmic one with no physical representation—chooses an object to explore. The agent then observes whether the object is static or movable, which type of joint it is attached to (if at all), and the value of the continuous parameters of the joint.

In Fig. 6.2, we show the situation after exploring each object five times. We show both discrete distributions along with their entropies and the expected change of entropy. The first two plots show the distributions of object type and joint type for each object, the third and forth show the discrete and differential entropies for each distribution and the last plot shows the expected change of entropy.

One can see that the first object is most likely a wall or another fixed object and the second object is probably movable along a prismatic joint. Although both objects have been explored the same amount of times, the expected change of entropy is *higher* for the movable object (see the last plot). This is due to the fact that the static object with high probability has a nonexistent joint type and joint properties (a *nil* value). Thus the entropy of those distributions is small (see the third and fourth plot). Furthermore, the probability of change is small, since we are already certain of the object being static. Consequently, the expected change of entropy is also small.

An explorer maximizing the expected change of entropy or choosing the object with maximal entropy would choose the object with the prismatic joint over the static object. This is an interesting behavioral observation. Intuitively, it is a reasonable decision, since drawers and doors seem more

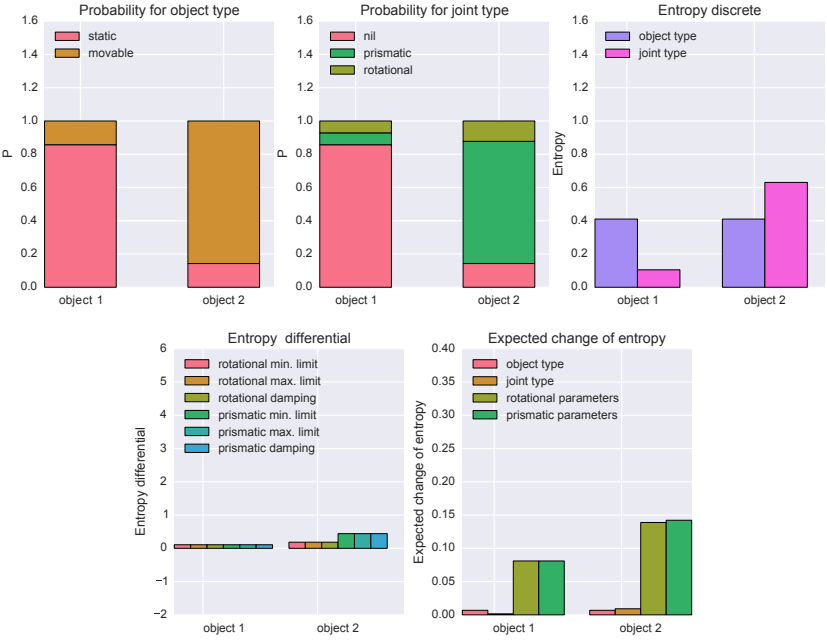


Figure 6.2: An example of a belief after ten exploration steps: Both objects have been explored five times. The first two plots show the distribution of the object type and the joint type, the third and forth plot show the entropies of the various distributions—either discrete or continuous—and the last plot shows the expected change of entropy of each distribution. Figure after Otte et al. (2014).

interesting to us. With our belief representation, we were able to catch this intuition by formally deducing higher self-information from those objects with larger parameter sets.

Fig. 6.3 provides further support for this statement. It shows the reduction of entropy achieved by the different strategies. Maximizing the expected change of entropy outperforms the heuristic strategies, although the difference in such a simple scenario is small. We will see a more complex scenario in the physical simulation experiment.

6.1.6 Physical Simulation

To test and compare the different strategies in a more realistic scenario, we set up a simulation of an environment with several DOF. The agent in this scenario is a PR2 robot with two 7-DOF arms, a telescopic spine, two grippers and a omni-directional base. We generate the full body motions with a rapidly exploring random tree (RRT) (LaValle and Kuffner 2000).

As shown in Fig. 6.4, in a more realistic scenario the difference between strategies is more pronounced. Although our observations are noise-free, we still have various sources of uncertainty. The physics simulation is not very precise and leads to unrealistic movements. Our 1d point mass model may not capture all of these effects. Also the joint pose inferred by articulation introduces a source of noise.

However, we can see that the strategies driven from information theory lead to better and faster uncertainty reduction. Also, the round robin and random strategies are still successful. Since we investigate the complete model, this surprising result is reasonable. We have no specific task but to learn a precise model of the world. So the properties of *all* objects are equally important. Thus only the fact that static objects lack certain properties makes them less interesting. Still, each exploration leads to more certainty that they are static and thus to a reduction of uncertainty.

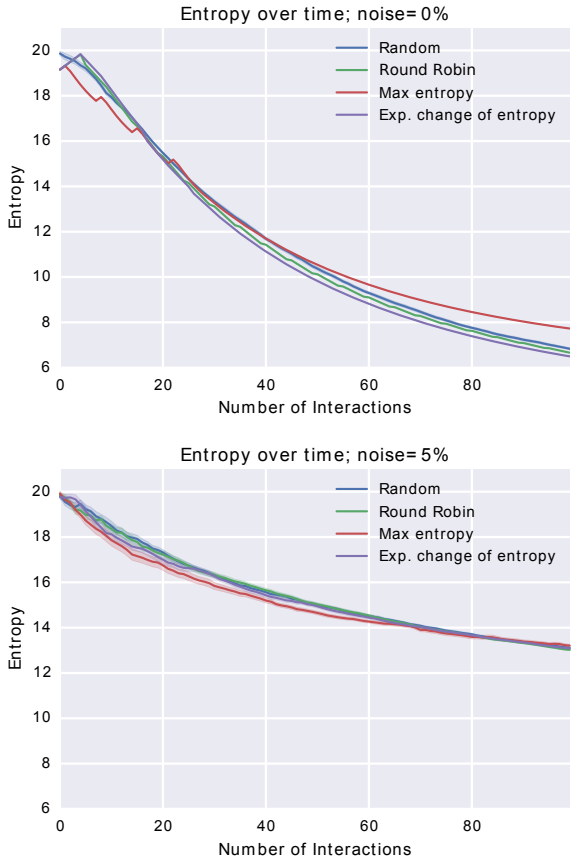


Figure 6.3: The performance in minimizing the belief entropy of different strategies in the toy world (without noise and with 5% noise). 20 runs were performed. Figure from Otte et al. (2014).

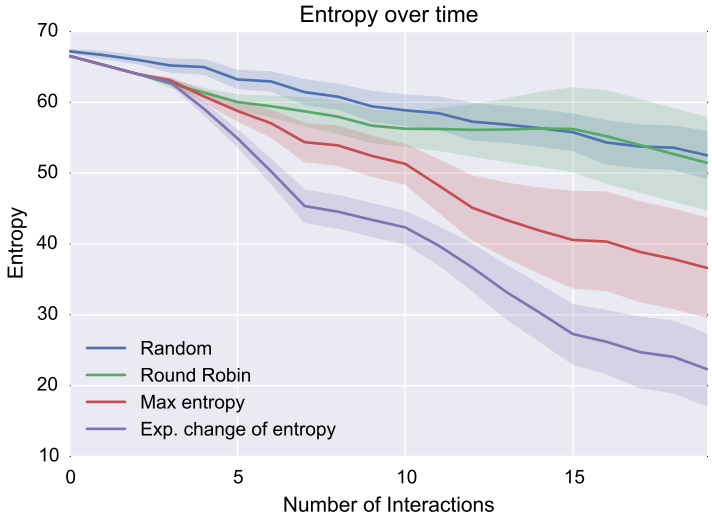


Figure 6.4: The performance in minimizing the belief entropy of different strategies in the physical simulation experiment. 19 runs were performed. Figure from Otte et al. (2014).

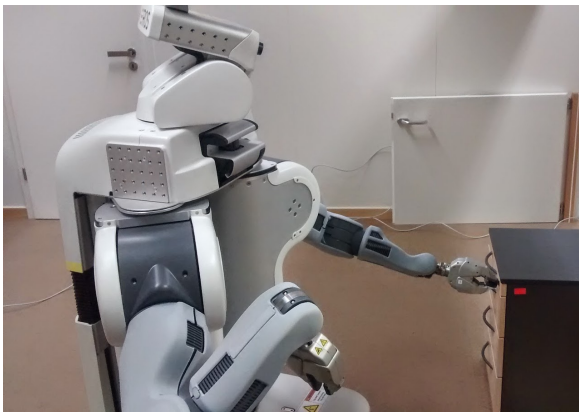


Figure 6.5: Real world experiment: our PR2 is exploring the joint dependency structure of a cabinet. After only two observations, the robot learned that in order to open the drawer, it must first unlock it by turning the key (compare to Fig. 6.10). Figure from Kulick et al. (2015a).

6.2 Dependent Joint Exploration

As shown in the previous section, learning a world model is still relatively easy and straight forward when all joints are directly movable with very simple actions. This changes, however, if every day objects like furniture come into play. Those objects have a dependency structure between the joints they consist of. A key, for example, opens a door when in the right position. When we assume every dependency structure to be equally likely, i.e., the position of each joint could possibly lock every other joint, we face a huge amount of dependency structures. Every joint has a continuous state from which all other joints might be dependent. There are too many possible dependencies to draw any sensible conclusion. Fortunately, the world is not an unstructured environment, but instead formed by human beings. We build things to lead our actions to functional changes. For example, many handles of doors open and close them at one end of their possible range, or snap into important states to guide the operator. Also, dependencies are more likely between joints close to each other, whereas dependencies between remote joints are rarely found. All these clues should lead our search for joint dependencies.

In this chapter we will use the priors about how joint dependency structures occur in the world to build fast and efficient inference methods for those structures, which in turn can then be used to explore those structures efficiently. With the rigid world assumption, joint dependencies are one of the more complex functional structures. Exploring them efficiently will lead to a much greater range of possible tasks, that can be solved.

Research on joints has not touched on dependency structures, but has focused on (a) handling and controlling known mechanisms (Klingbeil et al. 2010; Nagatani and Yuta 1995; Peterson et al. 2000), (b) estimating joint types and parameters (Martín-Martín and Brock 2014; Sturm et al. 2011) from given data, or (c) autonomous exploration to distinguish between predefined models (Barragán et al. 2014). In contrast, our work focuses on autonomous exploration of mechanisms with complex joint dependency structures, i.e., mechanisms where certain parts can only be articulated if the joints are in a specific configuration, not known beforehand. For this we as-

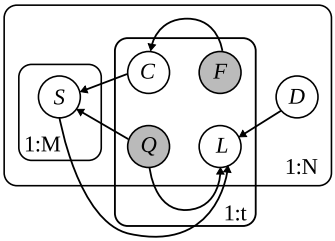


Figure 6.6: The probabilistic graphical model for joint dependency structures. Tab. 6.1 explains the random variables. Arrows leaving a plate and entering it again denote dependencies from all source RVs to all target RVs. Figure from Kulick et al. (2015a).

sume the control and the parameter estimation of mechanisms as given.

6.3 Probabilistic Modeling of Joint Dependency Structures

Joint dependency structures are structures where the state of one joint depends on the state of another joint. Consider environments (of rigid worlds), where each joint can be *locked* or *unlocked*. We call this the *locking state* of the joint. If a joint is *locked* no movement is possible (e. g., if the door handle is not turned the door cannot be opened), if a joint is *unlocked* movement within the normal constraints of the joint (e. g., axis limits or friction) is possible.

The locking state of one joint can change depending on the state of other joints. We call the joint whose position determines the locking state of another joint the *master* and the joint which locking state depends on the master's position the *slave*. We can divide the master's joint configuration into segments which put the slave into the *locked* or *unlocked* locking state.

With arbitrary possible dependencies between any two joints in the world, the search space for dependencies grows exponentially and can not efficiently be searched even for a small number of joints. This is especially problematic

Table 6.1: Summary of used symbols.

Symbol	*Description*	*Domain*
N	Number of joints	$\mathbb{N}$
M^j	Maximum joint angle of joint j	$\mathbb{R}$
t, s, u, v	Indices for time	$\mathbb{N}$
j	Index for joints	$\{1, \ldots, N\}$
D^j	RV, dependency of joint j	$\{1, \ldots, N+1\}$
L_t^j	RV, locking state of joint j	$\{0, 1\}$
Q_t^j	RV, joint state/position of joint j at time t	$\mathbb{R}$
F_t^j	RV, force/torque measurements of joint j at time t	$\mathbb{R}$
C_t^j	RV, change points of joint j at time t	$\{0, 1\}$
S_p^j	RV, segment borders of joint j at position p	$\{0, 1\}$

if a robot is to uncover the joint dependency structure of real world mechanisms. We overcome this problem by using the sensor clues the mechanisms offer. During the manipulation of a joint we measure its force/torque (F/T) feedback. Change points in the F/T measurements should indicate the borders between the locking state segments, e.g., in one segment the master locks the slave, and in other segments the master does not lock the slave.

6.3.1 Modeling the Joint Dependency Structure

The robot has to infer the joint dependency structure from its actions and the F/T measurements. For this we model the joint dependency structure as a probabilistic model (depicted in Fig. 6.6). We introduce random variables $D^{1:N}$, $L_{1:t}^{1:N}$, $Q_{1:t}^{1:N}$, $F_{1:t}^{1:N}$, $C_{1:t}^{1:N}$, and $S_M^{1:N}$, summarized in Tab. 6.1, which we explain in detail below. N is the number of joints to be modeled and t is the time index for the time dependent random variables. M^j is the maximum reachable joint position of joint j and we assume without loss of generality that the minimum joint position is 0.

D^j is a discrete random variable with the domain $\{1, \ldots, N+1\} \setminus \{j\}$. The first $N-1$ states indicate which other joints joint j depends on. The last state indicates if joint j is independent of all other joints. This choice of D^j limits our model to one-to-one joint dependencies. It is easy to extend this to more complex dependencies by extending the domain of D^j. That, however, enlarges the space of possible dependency structures significantly.

L_t^j is the locking state of joint j at time step t. It is a binary variable stating whether joint j is *locked* or *unlocked*. We could incorporate further locking states by increasing the cardinality of L_t^j. Thus we could encode states where a joint is, for example, disassembled or broken.

Q_t^j is the joint state of joint j at time t, i. e., the angle for rotational joints and the prismatic extension for prismatic joints.

F_t^j are potentially pre-processed force/torque (F/T) sensor measurements observed at time t for joint j. The measurements can be mapped to a particular joint since the agent knows which joint it is actuating. In these experiments only F/T measurements are used; however, different modalities, such as sound, could be incorporated in a similar fashion.

C_t^j is a binary variable that states whether at time t a change point in the F/T measurements was detected. Again, this can be mapped to a joint since the agent knows which joint it is actuating.

S_q^j is a binary variable stating whether there is a segment border at position q in joint state space of joint j. A segment border divides the continuous joint space into discrete functional units of the space. Since the joint space is a continuous space, strictly speaking S^j is a random field over the joint space. However, we simplified this by finely discretizing the joint space in our implementation such that S^j becomes a set of random variables.

The agent observes the joint state $Q_t^{1:N}$ at a given time t and the F/T measurements F_t^j. To query the locking state L_t^j, an oracle can be asked. Alternatively, an action could be performed (e. g., pushing, pulling or rotating the joint) to compute the locking state from the observations.

6.3.2 MaxCE Exploration Strategy

We formalize the goal to uncover the joint dependency structure as to minimize the uncertainty over D^j. This can be achieved by minimizing the entropy over D^j. Since we want to include many strong priors about joint dependencies (such as F/T clues or distance between joints), we leverage the MaxCE method developed in Sec. 3.

For this purpose, we compute the expected one-step cross-entropy between the current joint dependency structure distribution $P_{D_t^j}$ and the expected joint dependency structure distribution one time step ahead $P_{D_{t+1}^j}$. We maximize this expectation to get the optimal next sample position $Q_{t+1}^{0:N*}$:

$$(Q_{t+1}^{1:N*}, j) = \underset{(Q_{t+1}^{1:N}, j)}{\operatorname{argmax}} \sum_{L_{t+1}^j} p\left(L_{t+1}^j \mid Q_{t+1}^{1:N}, S^{1:N}\right) \cdot H\left[P_{D_t^j}; P_{D_{t+1}^j}\right] \tag{6.8}$$

with

$$p_{D_t^j} = p\left(D^j \mid L_{1:t}^j, Q_{1:t}^{1:N}, S^{1:N}\right) \tag{6.9}$$

$$p_{D_{t+1}^j} = p\left(D^j \mid L_{1:t+1}^j, Q_{1:t+1}^{1:N}, S^{1:N}\right). \tag{6.10}$$

6.3.3 Likelihood

The posterior of the joint dependency structure is dependent on the likelihood of the queries $p\left(L_{t+1}^j \mid D^j, Q_{t+1}^{1:N}, S^{1:N}\right)$. Since the locking state is a discrete variable, the usual choice for its probability is a Dirichlet prior. The likelihood is then:

$$p(L_{t+1}^j \mid D^j, Q_{t+1}^{1:N}, S^{1:N}) = \frac{\Gamma(\sum_k \alpha_k)}{\Gamma(O + \sum_k \alpha_k)} \prod_{k=0}^{K} \frac{\Gamma(c_k + \alpha_k)}{\Gamma(\alpha_k)}, \tag{6.11}$$

where O is the number of queries already made, K is the cardinality of L_{t+1}^j, α_k are the Dirichlet hyper-parameters, c_k are the query counts (i.e., counts

for each locking state, we will define it in depth shortly), and Γ is the gamma function.

In principle, we could use counts of how often each locking state of joint j occurred in the queries. We call this the unweighted query counts $\hat{c_k}$. Let $l_i^j \in \{0, \dots, K\}$ be the observed locking state of joint j in i-th query, where $0 \leq i < O$. Then the unweighted query count of locking state k is

$$\hat{c_k} = \sum_{i=0}^{O-1} \delta(l_i^j, k), \tag{6.12}$$

with $\delta(\cdot, \cdot)$ being the Kronecker delta

$$\delta(a, b) = \begin{cases} 1 \text{ if } a = b \\ 0 \text{ else} \end{cases} . \tag{6.13}$$

We will now explain how we weight the query counts to account for the fact that the locking state can change depending on the segment the joint is in. Let $\tau(i)$ be the time step of the i-th query. We weight each query by the probability that it was observed in the same segment as master joint h is at time step $t + 1$, i. e., the probability that there is no segment border in joint space between the joint positions of master joint h between $Q^h_{\tau(i)}$ and Q^h_{t+1} (cf. Eq. 6.15). The master joint is defined by the dependency structure D^j.

$$c_k = \sum_{i=0}^{O-1} \left(\delta(l_i^j, k) \prod_{q=Q^h_{\tau(i)}}^{Q^h_{t+1}} \left(1 - p\left(S_q^h \mid Q^h_{1:t}, C^h_{1:t}, F^l_{1:t} \right) \right) \right) . \tag{6.14}$$

How we can compute the probability of a segment border will be subject of Sec. 6.3.4.

6.3.4 Change Points

As described in the last section, we depend upon the probability of a segment border between two points in joint space that describes whether these

two points are in the same functional segment. While we could use typical distance functions like radial basis functions, we leverage the knowledge that state changes can often be inferred from feedback. To model this, we use the F/T sensor measurements to generate a probability distribution over segment borders, based on detected change points in the F/T measurements.

We assume that the segment borders are independent from each other. The probability of no state change between two positions Q^j_u and Q^j_v at time u and v of joint j is then

$$\prod_{q=Q^j_u}^{Q^j_v} \left(1 - p\left(S^j_q \mid Q^j_{1:t}, C^j_{1:t}, F^j_{1:t}\right)\right). \tag{6.15}$$

As depicted in the graphical model (see Fig. 6.6), S^j_q is dependent on $Q^j_{1:t}$ and $C^j_{1:t}$. We model this dependency as the probability of a segment border being the mean probability of a change point in the force measurements at a particular joint position. Thus we can infer S^j_q by

$$p\left(S^j_q \mid Q^j_{1:t}, C^j_{1:t}, F^j_{1:t}\right) = \frac{\sum_{s=1}^{t} \delta(Q^j_s, q) p(C^j_s \mid F^j_{1:t})}{\sum_{s=1}^{t} \delta(Q^j_s, q)} \tag{6.16}$$

with $\delta(\cdot,\cdot)$ the Kroenecker delta.

To infer the change point probability from the sensor measurements we leverage the Bayesian change point detection (BCPD) introduced by Fearnhead (2006).[1]

BCPD computes the posterior $p(C^j_s \mid F^j_{1:t})$ – the probability of a change point at a time s, given the sensor input.

It models the sensor data as m segments of independent constant data with Gaussian noise. Thus the data likelihood of two points F^j_t, F^j_s, conditioning them on being in the same segment, is

[1] The code for computing Bayesian change point detection is published as open-source python library at http://www.github.com/hildensia/bayesian_changepoint_detection.

$$p(t,s) = \int_{\mu} \prod_{i=t}^{s} T(F_i^j \mid \mu) \cdot \pi(\mu), \tag{6.17}$$

where $T(\cdot \mid \mu)$ is the Student's t distribution with location parameter μ and π is a prior distribution over the possible means. The Student's t distribution arises because the variance of the Gaussian noise is unknown, and we place a conjugate prior on it (see Murphy (2007) for details).

With this model we can recursively compute the exact probability of each time step being a change point between segments, even marginalizing out the number of change points m. The detailed derivation can be found in (Fearnhead 2006). Although the force torque measurement are not well described by a piecewise constant model, we can pre-process them, such that they obey such dynamics (see Sec. 6.4.1).

6.4 Experiments

6.4.1 General Setup

We test our dependency model in (a) a physical simulation and (b) on a real robot. Here we describe the parts of the experiments that are shared. The details specific to each experiment are described in the corresponding sections. The goal of the robot in both experiments is to uncover the joint dependency structure D of a complex mechanism.

Assumptions

We assume the agent initially has a kinematic model of the environment, including the existance of joints and their parameters, but not their dependencies.

For simplicity we also assume the F/T profile of each joint is known, i. e., the F/T recording of each joint of the entire joint space.

Actions

An action in our experiments is setting all joints to a desired state and querying the locking state of a joint. The exploration strategy (see Eq. 6.8) provides us with the desired target configurations of all joints $Q_{t+1}^{1:N*}$ and the joint j of which the locking state is supposed to be checked. Given the model of the mechanism, a controller can bring the mechanism to the desired configuration if possible. The locking state of joint j is supplied via the simulation or via a human oracle respectively.

Prior

For the probabilistic model, we must choose priors. We want to incorporate the knowledge that most joints in the world are independent of other joints, e.g., most drawers and cabinet doors can be opened directly without unlocking them and they do not lock each other. Therefore, for the dependency prior $p(D)$, we set the probability of a joint being independent to 0.7. We also think that proximity is a good indicator of the dependency of joints. Joints that are close to each other are more likely to depend on each other than joints that are far apart. For the dependency prior $p(D)$, we set the probability of being dependent on another joint proportional to $\frac{1}{d(i,j)}$, with $d(i,j)$ being the Euclidean distance between joint i and j. The probability of a joint locking itself is set to zero. More formally:

$$p(D^j = i) = \begin{cases} 0 & \text{if } i = j \quad \text{(self-dependent)} \\ .7 & \text{if } i = N+1 \quad \text{(independent)} \\ \frac{1}{d(i,j)c_N} & \text{else} \quad (j \text{ depends on } i) \end{cases} \tag{6.18}$$

with c_N being a normalization constant.

Data Pre-processing

The second graph in Fig. 6.7 shows the F/T sensor readings while actuating a joint. These readings are a non-linear time series. To efficiently compute the change point probabilities we, assume a piecewise constant model plus

Gaussian noise. To account for that assumption, we pre-process the F/T measurements as follows.

The measured force is the force needed to induce a certain velocity change on the joint, which can be calculated using physical laws as

$$\Delta v = \sqrt{v^2 - c_f v \Delta t}, \tag{6.19}$$

with v the velocity, Δt the change in time and c_f a virtual friction constant incorporating the mass of the object, the contact force and actual friction coefficient. From the actual F/T measurements we can thus compute the virtual friction constant which changes if there is feedback in the joint mechanism. As can be seen in the third graph of Fig. 6.7, the virtual friction constant is piecewise constant up to sensor noise. We can feed it to the Bayesian change point detection and compute change point probabilities (fourth graph of Fig. 6.7).

We map (cf. Eq. 6.16) the change point probabilities, which are in time space (bottom plot in Fig. 6.7), into joint space and get the probabilities over segment borders (top plot in Fig. 6.8). With this data we can then start the actual exploration of the mechanism.

The Exploration Sequence

The data pre-processing yields the segment border probabilities as depicted in the first graph of Fig. 6.8. From the segment border probabilities and all observation data we can compute the next query point/action (see Sec. 6.3 and Sec. 6.4.1). Then, the robot performs the given action, i.e., it brings all joints into the goal configuration $Q_{t+1}^{1:N*}$ and checks the locking state of joint j. The graphical model is updated with the resulting observation and the process is repeated. Fig. 6.8 shows some exploration steps of this procedure.[2]

[2] The code for performing the experiments is published as open-source software at http://www.github.com/hildensia/joint_dependency.

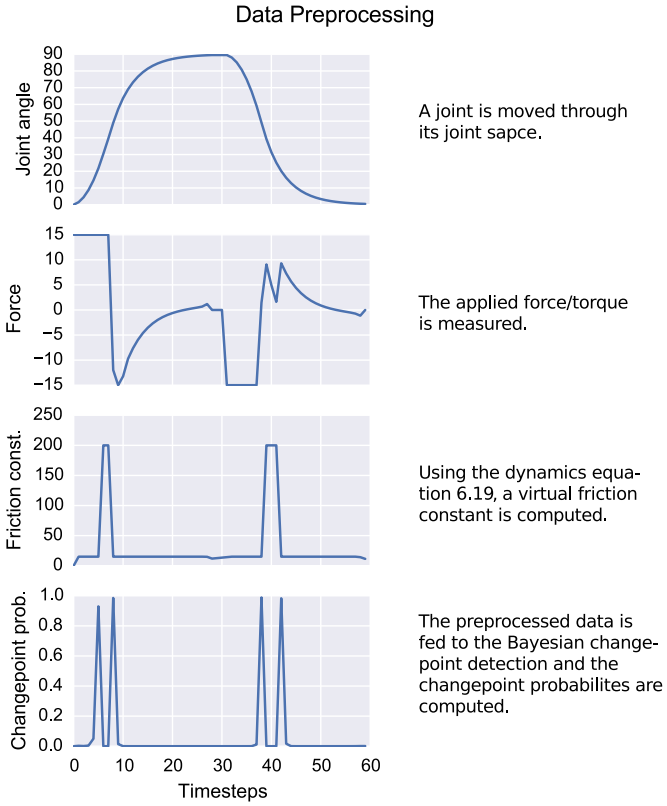


Figure 6.7: Data gathered from exploring a (simulated) cupboard, which has a key that locks the door between 20 and 50 degrees. All graphs show data from the key. The pre-processing of the F/T data shown here is done once once before the exploration. Figure after Kulick et al. (2015a).

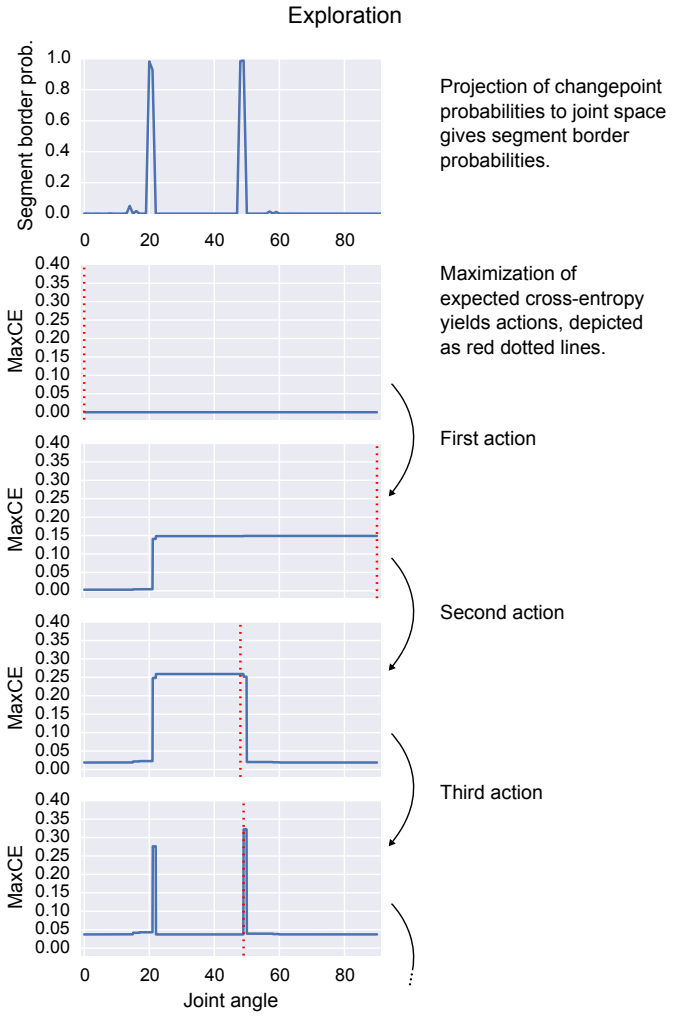


Figure 6.8: The exploration sequence of the same simulated cupboard as depicted in Fig. 6.7. (Caption continued on next page.)

Figure 6.8: (Previous page.) Again, only data from the key is shown. Each action moves the key to the desired configuration shown in red. After each action, the robot also observes the locking state of one joint. In this particular example, it explores the key of the cupboard and always queries the locking state of the door. Note that in the first step no observations are present and thus all actions are equally interesting. Tiebreaking is done randomly. After three actions, the probability of the key being the master of the door is already greater than 70% (not shown). Figure from Kulick et al. (2015a).

6.4.2 Physical Simulation: Setup

First we test our method in a physical simulation. We simulate the dynamics of the joints to determine change points in the movement profile. The agent applies forces to the joints to move them to desired positions using a PD-controller. The agent senses the applied forces and the position of the joint and can use this information to infer the change points and segments.

We test three different strategies to uncover joint dependency structures:

1. the *baseline strategy* selects actions randomly,
2. the *expected entropy strategy* minimizes the expected entropy of the distribution of D^j, and
3. the *MaxCE strategy* maximizes the cross-entropy as described in Sec. 3.

Additionally, we test the influence of the change point detection on the overall performance. We test all the strategies with *(a)* the change point detection enabled and *(b)* using a common exponential distance function as likelihood of two experiences belonging to the same joint state, instead of using the segments detected by the change point detection.

Each strategy is evaluated in 50 different environments. An environment consists of furniture with different joint dependency structures and different parameters of the locking state. An environment is generated by ran-

Table 6.2: Furniture used in the simulation.

Name	*Description*	*Locking mechanism*
Cupboard with handle	A cupboard with a door and a handle attached to it.	The handle must be at upper or lower limit to unlock the door
Cupboard with lock	A cupboard with a door and a key in a lock	The key must be in a particular position to unlock the door
Drawer with handle	A drawer with a movable handle	The handle must be at upper or lower limit to unlock the drawer
Drawer with lock	A drawer with a key in a lock	The key must be in a particular position to unlock the drawer

domly choosing three different objects from Tab. 6.2 and randomly choosing at which joint positions the furniture is locked. E.g., one instance of an environment might consist of a cupboard with a key (which is unlocked if the key is, e.g., between 73 and 93 degree), a cupboard with a handle (which opens, e.g., at its upper limit), and a drawer with a key (which is unlocked if the key is, e.g., between 122 and 142 degree). In an exploration sequence, the agent can perform 30 actions.

6.4.3 Physical Simulation: Results and Discussion

Fig. 6.9 shows the performances of the different strategies with and without change point detection. The upper plot shows the average correctly classified joint dependencies over time. We define correctly classified as recognizing the correct joint dependency with a probability of ≥ 0.5. Note that due to the prior, three dependencies are always classified correctly, namely the independent joints. Here one can see that the MaxCE method is able to classify significantly more dependencies correctly as well as faster when

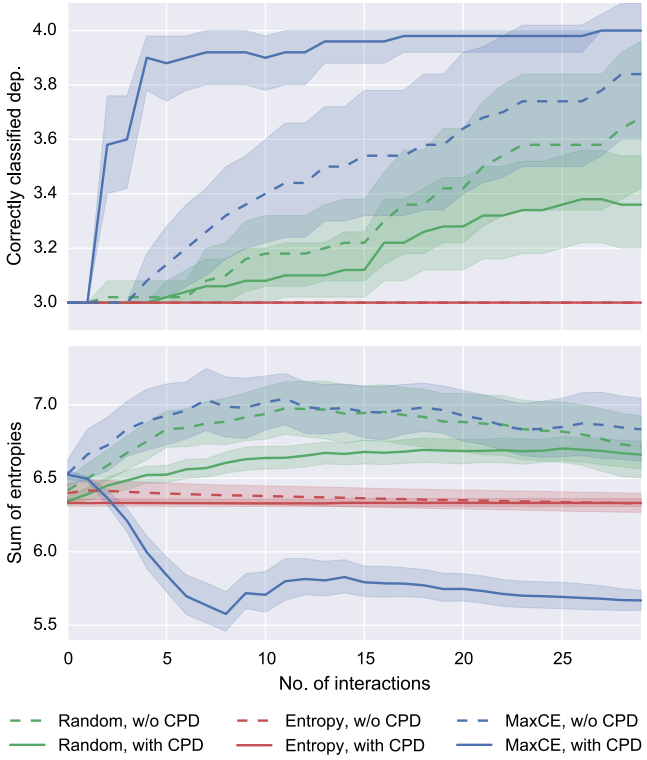


Figure 6.9: Results of the simulation experiment. We show the sum of entropies of all D^j random variables. The error bars reflect a 99% confidence interval of the mean estimator. Figure from Kulick et al. (2015a).

compared to all other strategies. One can also see that the change point detection increases the performance of the strategies. Note that the entropy method is not able to find out anything apart from the three independent assumptions, which are already correct due to the prior. The entropy is already quite low and cannot easily be reduced. Querying configurations that lower the independence belief would increase the entropy which is undesired although it would improve the belief in the long run.

In the lower graph we show the sum of entropies of all D^j over the actions. First, we can see that the MaxCE strategy with change point detection is the only strategy which is able to lower the entropies significantly. Again the entropy method does not lower the entropy, because the prior is too strong. All other strategies raise the entropy. This is due to the strong prior that joints are independent of each other. Thus lowering the belief of independence raises the entropy.

The MaxCE strategy that use the change point detection reduce the uncertainty faster than its counterpart without change point detection, because with change point detection an observation yields information for the entire segment. With the exponential distance function on the other hand fewer observations contribute to the dependency structure belief, because they are deemed uninformative.

6.4.4 Real World Experiment: Setup

We also test our dependency model using a real PR2 and a typical office cabinet. The cabinet has a drawer, which can be locked/unlocked by a key. The key also works as handle to open the drawer once the drawer is unlocked. Again, the robot knows the physical model of the cabinet, its dynamics and its joints (see Sec 6.4.1). The configuration of the furniture is measured through proprioception of the robot. The robot's wrist can measure the joint angle of the key. The extension of the arm corresponds to the joint value/extension of the drawer. We manually position the gripper of the robot around the key. No re-grasping is required during the experiment. The MaxCE strategy is used to explore the cabinet. Fig 6.5 shows our PR2 exploring the joint dependency structure of a cabinet drawer.

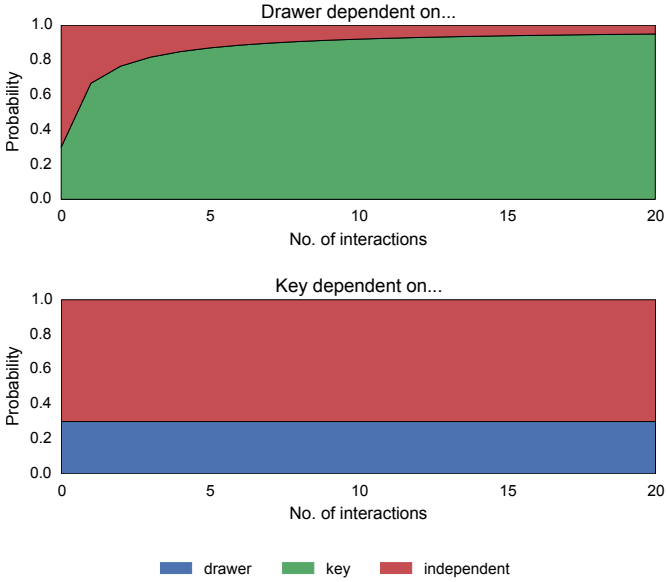


Figure 6.10: The results of the real world experiment. We show the probability distribution of the two involved D^j random variables over the number of interactions the robot made. We can see that the dependency of the key as master over the slave drawer is uncovered very early, whereas the probability of the independence of the key is not significantly raised during the interactions. The self-dependency is zero throughout the whole experiment as it has a hard zero prior. Figure from Kulick et al. (2015a).

As additional sensory clues the robot uses F/T feedback of the drawer, measured by a sensor located in the wrists of the PR2. The measurements consist of the three dimensional force vector and three dimensional torque vector acting on the wrist of the robot. We sum up these values and preprocess them as stated in Sec. 6.4.1.

6.4.5 Real World Experiment: Results and Discussion

Fig. 6.10 shows the results of the conducted experiment. The main result is that the robot is able to uncover the dependency of the drawer from the key position. Even with several sources of noise—proprioception, F/T sensor, and motion generation—it took only very few actions to uncover the correct dependency structure for the drawer. We can thus conclude that the proposed method works well even in real world scenarios.

Looking at the exploration sequence in more detail, we see that the robot was not able to increase the probability that the key was an independent joint. This results from the fact that there are no direct observations which imply independence and therefore would increase the probability for the joint being independent. During the experiment we only observed that the key was in the "unlocked" locking state, but this does not increase the probability of the key being an independent joint. Being always in the unlocked stated can either mean that no master was found so far, thus we could not lock the state yet, or that the joint is independent indeed. To guarantee independence we would have to observe the "unlocked" state for the entire configuration space, i.e., the joint space of *all joints*. This could only be accomplished by exhaustively searching the configuration space.

6.4.6 Conclusion

We developed an active learning method to explore complex joint dependency structures. The method leveraged the sensory clues from the mechanisms (F/T measurements were used) to segment the joint space into meaningful discrete clusters. While in this experiment we limited clues to F/T clues, it is relatively straight forward to add more data sources, such as sound.

The MaxCE strategy proved to be an efficient strategy to explore joint dependency structures whereas the common active learning strategy, expected entropy, did not succeed. We demonstrated our method in simulation and on a real PR2.

One major limitation of our approach is the fact that only one-to-one dependencies can be modeled, since the D^j random variable only gives the dependency to exactly one other joint. While this could easily be extended to multiple dependencies, the exploration strategies would have to cope with $O(N^2)$ different models already for two dependencies of each joint, up to exponentially many models for arbitrary dependencies. Arbitrary joint dependencies are, however, rare. Using a prior that limits the set to a useful size, e.g., only few master joints for a slave, would reduce the amount of models drastically.

7 Discussion and Conclusion

We have seen that exploration is a necessary ability for robots to solve many problems. In this thesis we developed a general framework to solve exploration problems. While its results are convincing, there are of course limitations to the approach described. Also every research raises new questions to be answered. In this chapter we will first discuss competing methods, that are candidates for also solving the exploration problem. Then we discuss limitations of our work and further work to overcome these limitations. We will close the thesis with a final discussion of the results.

7.1 Discussion on Alternative Approaches

7.1.1 Could Big Data Solve the Exploration Problem?

Big Data and coupled with it the term *Deep Learning* are the latest trends in the machine learning community. Deep Learning techniques together with immense databases, aggregated through the ascend of ubiquitous online devices, have achieved tremendous results in various topics of learning (e.g., image classification (Russakovsky et al. 2015), face recognition (Taigman et al. 2014), or mastering the game of Go (Silver et al. 2016)). With these achievements in sight, it is natural to ask, whether using enormous amounts of data could render specialized exploration strategies, as those developed in this thesis, useless. Could the use of Big Data together with Deep Learning solve exploration as well?

Deep Learning, as pioneered by Bengio et al. (2007), Hinton, Osindero, et al. (2006), and Poultney et al. (2006), refers to training an artificial neural network with more than one hidden layer. Deep neural networks are mainly very powerful function approximators. To solve the exploration problem

one could train such a net to approximate a function from sensor inputs to actions. These actions should in turn maximize information gain and thus implement an exploration strategy. It would be necessary to incorporate a notion of information gain into the training process of the net, such that it would be able to represent sensor inputs in a way that novelty of new observations could be measured. Although it is not clear how such a net would look like nor how it would be trained, it is not unlikely that research toward such a net would generate valuable insights and results. Such an *end-to-end* learning would implement a model-free exploration strategy, as it does not require a model of the environment's dynamics. See the next section for further discussion about model-free exploration. Neural networks are, however, notorious to need a large amount of data to be trained. This is the main reason why it recently became successful: the ubiquity of digitally processed data made huge training sets available. Deep Learning needed Big Data to prove its potential.

But to make use of Big Data the data has to be available in the first place. In robotics, generation of data samples is still often expensive. On the one hand working with a robot is tedious and the hardware is often expensive in terms of money. Running a real robot also bears the risk to break it, such that the operation time of a robot is limited. On the other hand, there is no standardized format to archive data gathered by one robot and even if there was, the translation of actions from one robot to another is hard. Nevertheless, robots become more and more available and affordable. This leads to much more data being generated and could eventually lead to big data sets for learning in robotics. With these data sets, we could generate good approximations of the underlying distributions in the world, which often would mitigate the need of exploration. Whenever the environment would follow this distribution the robot could operate instantly. And while this is desirable and will be very useful in the future, exploration is exactly about the outliers of these distributions. Even if we have many data sets from which we can extract prior knowledge, we still want to find the specialties of the environment at hand. And in every new environment we only have little data. We then need efficient strategies to find the interesting parts of the environment that make it special. Even if the environment follows the

prior distribution, we might still be interested in the actual value of properties. They enable us to perform well in the environment.

MaxCE is an algorithm which is exactly about finding outliers and specialties quickly. We have shown (e.g., in Sec. 6) that especially in the presence of strong priors MaxCE can be used for finding good explorative actions. When available, Big Data will give us exactly those strong priors on a much more reliable basis than using common sense and rule-of-thumb approximations as we did. However, the exploration strategy could stay the same.

7.1.2 Model Free Exploration

All the exploration strategies developed in this thesis are model-based algorithms. That means they use a predictive distribution to calculate expected effects of actions. One could ask if there are model-free alternatives to this. Model-free methods directly compute values of state-action pairs without using a forward model to predict the outcome of an action. The values are learned from experiences without first learning a model. Exploration in such a setting would mean to choose actions that improve the approximation of the values. We could use the same underlying principle as in the other examples in this thesis: we could model a distribution over the state-action pair values and try to minimize the entropy of that distribution. But since we do not have a forward model we could not use for example the MaxCE strategy or traditional Bayesian experimental design.

Since an agent with a model-free exploration strategy does not have a notion about the outcome of an action, it cannot rely on the possible changes that occur in the state to measure novel experiences. Instead, in a purely model-free scenario the agent has only counts of state-action pairs together with the reward earned. With only these data we can not transfer knowledge of one state to another state. Thus an agent needs to explore large parts of the state space to learn a good policy. We could tackle this problem by developing similarity measures between states, but this would be similar to building a model of the state, since it would need to encode how different states would react to actions. Although not directly being a model of the dynamics of the environment, it would be a first step in that direction.

With a model-based approach on the other hand we are able to transfer knowledge gathered in one part of the state space to different states, because we learn the dynamics of the environment. In a purely exploration scenario, learning a model also has the benefit of generalizing to different tasks. Using the model of the dynamics, we are able to plan towards different goals without relearning a value function.

7.2 Limitations and Future Work

7.2.1 Relaxing the Rigid World Assumption

Relaxing the strict assumption of a world consisting only of rigid bodies is an obvious path for further research. This would clearly widen the area where our methods and framework would be applicable. We could in principle widen the spectrum until modeling the behavior of human beings as degrees of freedom in the environment. While this would enlarge the problem-set greatly, our expectation is that many of the underlying principles described in this thesis would still be applicable. Information gain as driving force behind exploration and probabilistic models as representations are by no means limited to modeling rigid environments. Also MaxCE will be applicable on much more complex models. The reason for the rigid world assumption is to have a set of environments and tasks that are easier to operate with a robot of today's capabilities. Within these boundaries we can focus on the general principles behind exploration using well understood algorithms for perception and movement generation.

That said, it is clear that building efficient inference algorithms on very complex models is hard. Many of the assumptions made within the various experiments in this thesis do not hold for non-rigid bodies like cloth or liquids. However, for many tasks one can not model the non-rigidity at all and just assume things to be rigid. For instance, the detailed movement of fabric of cloth is seldom important to solve a task. If one wanted to explore the possible movements for folding shirts, however, it is, requiring more detailed models. But we can hope that we could still use the probabilistic exploration framework, as long as we can find useful representations

of the fabric. The rigid world assumption does not restrict the generality of the principles we have investigated, it allows us to focus on the structure of exploration.

7.2.2 Automatically Modeling Exploration Tasks

In this thesis each exploration task (e.g., learning joint dependency structures or learning joint existence) is defined by hand. We build graphical models to capture the tasks and derive inference algorithms from the models manually. The ultimate autonomy would be to automatically model those tasks. To accomplish that a robot would need to learn the representation of each task while performing it. Representation learning is often defined as dimensionality reduction process, as for example with auto-encoders (Hinton and Salakhutdinov 2006). Auto-encoders compress input data to a lower dimensional space. The objective for the compression is to be able to reproduce the data with minimal corruption from the compressed representation. Jonschkowski and Brock (2015) added more semantics to that setting by incorporating general prior knowledge about the environment in the representation learning process, like a notion of Newton's law of motion. The representations enabled their robot to learn navigation tasks in a reinforcement learning setting.

Still, these representations do not catch the variety of structured graphical models. There is interesting research on learning Bayesian networks from data (De Campos and Ji 2011). However, the techniques are still not well enough understood, that automatically modeling a robotic exploration task will be possible soon. For still some time, we will need to build the models by hand.

7.2.3 Planning to Explore

The exploration strategies used in this thesis have one major limitation: They only look one step ahead. This is reasonable when optimizing *submodular* functions, since it is guaranteed to perform near optimal. But in our formulation of active learning there are no guarantees for convergence or near op-

timal behavior. As shown in Sec. 2.4, optimal exploration requires to compute an optimal policy in a POMDP. Thus for choosing the optimal action we need to take all possible future action outcomes into account. Instead we only take the possible outcomes of *one* action into account. This can lead to suboptimal performance. However, taking a possibly infinite amount of future states into account is too expensive. But we still could look more steps ahead, using tree-search algorithms.

For our probabilistic approach Monte-Carlo-Tree-Search (MCTS) algorithms are especially interesting (Browne et al. 2012). While classical tree-search algorithms build a search tree up to a particular depth and evaluate the leaf notes by some heuristic, MCTS samples the model further to estimate the reward that could be reached from that state. This estimate can be propagated back to the current decision of which action one should take next. Through sampling the model, we get an estimate of the reward without the need of an expert building a heuristic. MCTS normally does not incorporate explicit exploration strategies. For complex tasks, however, exploring first could enhance the result by giving a better understanding of the underlying model. Using the MaxCE measure as intrinsic motivation could be a driving force towards such explicit exploration behavior.

7.2.4 Physical Reasoning

The methods for physical reasoning in this thesis are very rudimentary. The joint exploration experiments are purely parametrized. Physical reasoning in these experiments only consists of setting proper limits to the parameters. The active symbol grounding (Sec. 5) needs slightly more sophisticated methods. Still, the rejection sampling approach is very basic and will need too many samples to generate useful distributions in more complex scenarios. Toussaint (2015) gives interesting directions how physical reasoning could be driven by the abilities of an agent to build up possible future configurations. However, reasoning about the physical state of an environment needs much more attention of the scientific community if robots are to be enabled to act in such environments.

7.2.5 Safe Exploration

An important question for robotic exploration is whether the chosen actions are safe to perform. In traditional active learning scenarios sampling does not come with severe punishments more than computational costs. When robots act in real environments they can damage both itself as well as the environment or even harm human beings. When leaving the lab environment, it is important to model such costs and perform safe exploration. While restricting the inner degrees of freedom to reachable limits is easy, doing so for external degrees of freedom is hard. Limits are not known, it is not clear how easily things may break and how much force one can exert on them.

All those questions are not yet modeled nor answered within the developed framework. Safe exploration in MDPs and reinforcement learning has gotten some attention (Hans et al. 2008; Moldovan and Abbeel 2012). For real-world exploration more research is needed. Especially the physical reasoning becomes more involved if damage and injuries are supposed to be integrated in the reasoning process.

7.2.6 Object Assumption

Another limitation of all exploration experiments in this thesis is the assumption of solved object recognition, i.e., all methods are based on the assumption that we already have the visual input segmented into rigid bodies. While Martín-Martín and Brock (2014) or Sturm et al. (2011) provide methods to recognize joints and objects, using them still means to decouple the recognition from the exploration behavior. While this is a reasonable factorization of the problem, it might not be the best one. It would be interesting to feed information from the exploration back into the recognition algorithms to enhance their results.

7.3 Conclusion

Exploration is an important ability for any intelligent being, be it biological or artificial. As we have seen in the cockatoo experiment described in the beginning of the thesis, exploration enables agents to operate in totally unknown environments. Exploring the external degrees of freedom of an environment is a way how we can enable robots to solve tasks in many unknown environments, as degrees of freedom are a major functional unit in the world. Exploring their existence and their function is necessary to do many useful things, if we do not want to manually model all degrees of freedom. If we want robots and artificial systems to understand the world and be able to interact with it in unknown scenarios and environments, we undoubtedly need a good framework for exploration.

So far, researchers tend to neglect pure autonomous exploration in favor of structuring the environment. When deployed to unstructured environments they are often remote-controlled, not using exploration strategies as well. The areas where exploration was used in robotics so far were either about the internal degrees of freedom or about mapping the environment.

In this thesis we developed a framework for exploring degrees of freedom based on information theory, that enables robots to handle new situations by leading them to explorative actions that very generally lead to information gain. The notion of informativeness drives the robot to uncover as much knowledge about its environment as possible. The MaxCE strategy, developed within this thesis, let it handle strong priors efficiently, by handling outliers as interesting points, rather than undesired mischief-maker. While similar Bayesian experimental design and active learning techniques have been studied for many years, their use for robotic exploration is a recent development.

We have shown these theoretic foundations to be useful in various scenarios, all obeying the rigid world assumption. Although the rigid world assumption covers a whole range of useful environments, many are omitted. However, the general principle of information gain as driving principle directly translates to other environments and will be still applicable way beyond the rigid world assumption. How models in such environments will

look like is still open to research. But information gain and MaxCE will be applicable, as long as the models are within the probabilistic formalism.

With the information theoretic foundation it is a natural choice to model environments within the framework of Bayesian statistics. However, Bayesian methods are notorious for being computationally very expensive. We have, however, shown that using the right structural belief we can efficiently compute them—even exactly. Nevertheless, we might discover situations where no such structure is apparent and we have to resolve to general inference techniques, that are computationally hard. On the other side, using probabilistic models enables us to easily incorporate prior knowledge, leading to faster exploration.

We also discussed limitations to our approach. The first major limitation is that we have to manually model every new exploration task (exploring object existence, exploring joint existence and parameters, exploring joint dependency structures). While these tasks are relatively broad and translate well to different environments, it still involves the process of a human researcher defining the model and the exact inference methods. It is an interesting question if we will be able to unify the different exploration task developed in this thesis to one general model of exploration.

The second major limitation is that we can not solve the exploration problem optimally, but have to rely on one-step look-ahead heuristics. We have, however, shown that our heuristics are successful in a variety of tasks. We also have discussed a possible way to incorporate the information theoretic measures into planning algorithms. With the farther planning horizon of these algorithms they should improve over the one-step look-ahead.

Concluding, we are optimistic that the methods developed in this thesis will enable robots to solve a wider range of tasks in unknown environments. Moving the investigation of environments from the researcher to the robot will be a huge step to more autonomous, more intelligent systems, which will perform well in a broad area of tasks rather than in narrow specialized fields. If we are to build truly autonomous systems, exploring their environment will always be one of their important tasks, as it has for any intelligent being. Hence, understanding the process driving the exploration is important.

A Appendices

A.1 Proof: The Conditional Hypotheses Entropy is not Submodular

We proof that the conditional hypotheses entropy is not submodular by contradiction.

Definition 7. *For a set Ω, the set function $f\colon 2^{\Omega} \to \mathbb{R}$ is submodular if and only if*

$$f(D \cup \{y_1\}) + f(D \cup \{y_2\}) \geq f(D \cup \{y_1, y_2\}) + f(D) \tag{A.1}$$

with $D \subset \Omega$ and $y_1, y_2 \in \Omega \setminus D$.

Definition 8. *For a random variable θ and a set of random variables Y,*

$$H[\theta|Y] = \sum_{Y} p(Y) H[p(\theta|Y)] \tag{A.2}$$

is the conditional entropy.

Lemma 1. *$f\colon 2^{\Omega} \to \mathbb{R}$ with $f(Y) = H[\theta|Y]$ is not submodular.*

Proof. We proof by contradiction, giving an example that violates

$$H[\theta|\emptyset \cup \{y_1\}] + H[\theta|\emptyset \cup \{y_2\}] \geq H[\theta|\emptyset \cup \{y_1, y_2\}] + H[\theta|\emptyset] \ . \tag{A.3}$$

Let θ be a binary random variable, and y_1 and y_2 be identically distributed binary random variables with

$$\text{prior:} \qquad p(\theta) = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \tag{A.4}$$

$$\text{likelihood:} \quad p(y|\theta) = \begin{pmatrix} 0.1 & 0.9 \\ 0.9 & 0.1 \end{pmatrix} \tag{A.5}$$

$$\text{marginal:} \quad p(y) = \sum_{\theta} p(y|\theta)p(\theta) = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \tag{A.6}$$

$$\text{posterior:} \quad p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} = \begin{pmatrix} 0.1 & 0.9 \\ 0.9 & 0.1 \end{pmatrix}, \tag{A.7}$$

so that

$$\begin{aligned} H[\theta|y_1] &= H[\theta|y_2] = H[\theta|y] \\ &= -\sum_{y} p(y) \sum_{\theta} p(\theta|y) \log p(\theta|y) = 0.325, \end{aligned} \tag{A.8}$$

$$H[\theta|\emptyset] = H[\theta] = \sum_{\theta} p(\theta) \log p(\theta) = 0.693 \tag{A.9}$$

and

$$\begin{aligned} H[\theta|y_1] + H[\theta|y_2] &= 2 \cdot 0.325 < 0.693 \\ &= H[\theta] \leq H[\theta|y_1, y_2] + H[\theta] \ , \end{aligned} \tag{A.10}$$

which contradicts Eq. (A.3). □

A.2 Expected Kullback-Leibler divergence transformations

The KL-divergence, the entropy, and the cross-entropy of two distributions p and q are closely related (rows in Eq. (A.11)) and can be rewritten as expectation values (columns in Eq. (A.11))

$$\begin{aligned}
\mathrm{D}_{\mathrm{KL}}\big[p(x) \parallel q(x)\big] &= H\big[p(x), q(x)\big] &&- H\big[p(x)\big] \\
\| & \| && \| \\
\int p(x) \log \frac{p(x)}{q(x)} dx &= -\int p(x) \log q(x) &&+ \int p(x) \log p(x) \\
\| & \| && \| \\
E\Big[\log \frac{p(x)}{q(x)}\Big]_{p(x)} &= -E\Big[\log q(x)\Big]_{p(x)} &&+ E\Big[\log p(x)\Big]_{p(x)} .
\end{aligned} \tag{A.11}$$

When taking the expectation of the KL-divergence over $p(y|x, D)$, depending on the direction of the KL-divergence, either the entropy or the cross-entropy term is constant with respect to x (and therefore drops out when taking the argmax_x)

$$\begin{aligned}
&E\Big[\mathrm{D}_{\mathrm{KL}}\big[p(\theta|y, x, D) \parallel p(\theta|D)\big]\Big]_{p(y|x,D)} \\
&= -E\Big[H\big[p(\theta|y, x, D)\big]\Big]_{p(y|x,D)} + E\Big[H\big[p(\theta|y, x, D), p(\theta|D)\big]\Big]_{p(y|x,D)} && \text{(A.12)} \\
&= -E\Big[H\big[p(\theta|y, x, D)\big]\Big]_{p(y|x,D)} - E\Big[\log p(\theta|D)\Big]_{p(\theta|y,x,D),p(y|x,D)} && \text{(A.13)} \\
&= -E\Big[H\big[p(\theta|y, x, D)\big]\Big]_{p(y|x,D)} - E\Big[\log p(\theta|D)\Big]_{p(\theta|D)} && \text{(A.14)} \\
&= -E\Big[H\big[p(\theta|y, x, D)\big]\Big]_{p(y|x,D)} - \underbrace{H\big[p(\theta|D)\big]}_{const.} && \text{(A.15)}
\end{aligned}$$

$$
\begin{aligned}
&E\Big[\mathrm{D}_{\mathrm{KL}}\big[p(\theta|D) \parallel p(\theta|y,x,D)\big]\Big]_{p(y|x,D)} \\
&= E\Big[H\big[p(\theta|D), p(\theta|y,x,D)\big]\Big]_{p(y|x,D)} - E\Big[H\big[p(\theta|D)\big]\Big]_{p(y|x,D)} && \text{(A.16)} \\
&= E\Big[H\big[p(\theta|D), p(\theta|y,x,D)\big]\Big]_{p(y|x,D)} - \underbrace{H\big[p(\theta|D)\big]}_{const.} . && \text{(A.17)}
\end{aligned}
$$

For the step from Eq. (A.13) to Eq. (A.14), note that $p(\theta|x,D) = p(\theta|D)$ since θ is independent of x so that for any function $f(\theta, D)$ that depends only on θ and D, such as $\log p(\theta|D)$ above, an expectation over $p(\theta|y,x,D)$ and $p(y|x,D)$ is equal to an expectation over just $p(\theta|D)$

$$
\begin{aligned}
E\Big[f(\theta,D)\Big]_{p(\theta|y,x,D)\,p(y|x,D)} &= \iint_{y,\theta} f(\theta,D)\, p(\theta|y,x,D)\, p(y|x,D)\, dy\, d\theta && \text{(A.18)} \\
&= \int_{\theta} f(\theta,D) \left[\int_{y} p(\theta,y|x,D)\, dy \right] d\theta && \text{(A.19)} \\
&= \int_{\theta} f(\theta,D)\, p(\theta|x,D)\, d\theta && \text{(A.20)} \\
&= \int_{\theta} f(\theta,D)\, p(\theta|D)\, d\theta && \text{(A.21)} \\
&= E\Big[f(\theta,D)\Big]_{p(\theta|D)} . && \text{(A.22)}
\end{aligned}
$$

References

Abe, N. and M. Hiroshi (1998). "Query Learning Strategies Using Boosting and Bagging." In: *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 1–9.

Ackerman, E. (2011). "Japan Earthquake: iRobot Sending Packbots and Warriors to Fukushima Dai-1 Nuclear Plant". In: *IEEE Spectrum*.

Angluin, D. (1988). "Queries and concept learning". In: *Machine Learning Journal* 2.4, pp. 319–342.

Auersperg, A. M., A. Kacelnik, and A. M. von Bayern (2013). "Explorative Learning and Functional Inferences on a Five-Step Means-Means-End Problem in Goffin's Cockatoos (Cacatua goffini)". In: *PLoS ONE* 8.7.

Bache, K. and M. Lichman (2013). *UCI Machine Learning Repository*. Web archive of machine learning problems. URL: http://archive.ics.uci.edu/ml. University of California, Irvine, School of Information and Computer Sciences.

Baldwin, D. A., E. M. Markman, and R. L. Melartin (1993). "Infants' Ability to Draw Inferences about Nonobvious Object Properties: Evidence from Exploratory Play". In: *Child Development* 64.3, pp. 711–728.

Barber, D. (2012). *Bayesian Reasoning and Machine Learning*. Cambridge, UK: Cambridge University Press.

Barragán, P. R., L. P. Kaelbling, and T. Lozano-Pérez (2014). "Interactive Bayesian Identification of Kinematic Mechanisms". In: *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pp. 2013–2020.

Bay, H., A. Ess, T. Tuytelaars, and L. Van Gool (2008). "Speeded-Up Robust Features (SURF)". In: *Computer Vision and Image Understanding* 110.3, pp. 346–359.

Beetz, M., L. Mosenlechner, and M. Tenorth (2010). "CRAM - A Cognitive Robot Abstract Machine for everyday manipulation in human environments". In: *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 1012–1017.

Bellman, R. (1957a). "A Markovian Decision Process". In: *Journal of Mathematics and Mechanics* 6, pp. 679–684.

Bellman, R. (1957b). *Dynamic Programming*. Princeton, NJ: Princeton University Press.

Bengio, Y., P. Lamblin, D. Popovici, H. Larochelle, et al. (2007). "Greedy layer-wise training of deep networks". In: *Advances in Neural Information Processing Systems (NIPS)* 19, pp. 153–160.

Bersch, C., D. Pangercic, S. Osentoski, K. Hausman, Z.-C. Marton, R. Ueda, K. Okada, and M. Beetz (2012). "Segmentation of cluttered scenes through interactive perception". In: *RSS Workshop on Robots in Clutter: Manipulation, Perception and Navigation in Human Environments*.

Brafman, R. I. and M. Tennenholtz (2002). "R-MAX - A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning". In: *Journal of Machine Learning Research (JMLR)* 3, pp. 213–231.

Breiman, L. (1996). "Bagging Predictors". In: *Machine Learning Journal* 24.2, pp. 123–140.

Browne, C. B., E. Powley, D. Whitehouse, S. M. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, S. Colton, et al. (2012). "A survey of monte carlo tree search methods". In: *Trans. on Computational Intelligence and AI in Games* 4.1, pp. 1–43.

Burgard, W., D. Fox, and S. Thrun (1997). "Active mobile robot localization by entropy minimization". In: *Proc. of Workshop on Advanced Mobile Robots*, pp. 155–162.

Cakmak, M. and A. L. Thomaz (2012). "Designing robot learners that ask good questions". In: *Proc. of the Conf. on Human-Robot Interaction (HRI)*, pp. 17–24.

Cangelosi, A. and L. B. Smith (2015). *Developmental Robotics*. Cambridge, Mass.: MIT Press.

Carrillo, H., I. Reid, J. Castellanos, et al. (2012). "On the comparison of uncertainty criteria for active SLAM". In: *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pp. 2080–2087.

Chaloner, K. and I. Verdinelli (1995). "Bayesian Experimental Design: A Review". In: *Statistical Science* 10.3, pp. 273–304.

Chao, C., M. Cakmak, and A. Thomaz (2010). "Transparent active learning for robots". In: *Proc. of the Conf. on Human-Robot Interaction (HRI)*, pp. 317–324.

Cohn, D. A., L. Atlas, and R. Ladner (1994). "Improving generalization with active learning". In: *Machine Learning Journal* 15.2, pp. 201–221.

Cohn, D. A., Z. Ghahramani, and M. I. Jordan (1996). "Active learning with statistical models". In: *Journal of Artificial Intelligence Research (JAIR)* 4.1, pp. 129–145.

Coradeschi, S. and A. Saffiotti (2000). "Anchoring symbols to sensor data: preliminary report". In: *Proc. of the Nat. Conf. on Artificial Intelligence (AAAI)*, pp. 129–135.

Coradeschi, S. and A. Saffiotti (2003). "An introduction to the anchoring problem". In: *Robotics and Autonomous Systems* 43.2, pp. 85–96.

Crisp, J. A., M. Adler, J. R. Matijevic, S. W. Squyres, R. E. Arvidson, and D. M. Kass (2003). "Mars Exploration Rover mission". In: *Journal of Geophysical Research: Planets* 108 (E12), p. 8061.

Culotta, A. and A. McCallum (2005). "Reducing labeling effort for structured prediction tasks". In: *Proc. of the Nat. Conf. on Artificial Intelligence (AAAI)*. AAAI, pp. 746–751.

DARPA Robotics Chellange Organisation Team (2015). *DRC Finals Rule Book*. DISTAR Case 24388. DARPA.

De Campos, C. P. and Q. Ji (2011). "Efficient structure learning of Bayesian networks using constraints". In: *Journal of Machine Learning Research (JMLR)* 12, pp. 663–689.

De Raedt, L. (1997). "Logical settings for concept-learning". In: *Artificial Intelligence Journal* 95.1, pp. 187–201.

Dragiev, S., M. Toussaint, and M. Gienger (2011). "Gaussian process implicit surfaces for shape estimation and grasping". In: *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pp. 2845–2850.

Dragiev, S., M. Toussaint, and M. Gienger (2013). "Uncertainty aware grasping and tactile exploration". In: *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pp. 113–119.

Džeroski, S., L. De Raedt, and K. Driessens (2001). "Relational reinforcement learning". In: *Machine Learning Journal* 43.1-2, pp. 7–52.

Endres, F., J. Trinkle, and W. Burgard (2013). "Learning the Dynamics of Doors for Robotic Menipulation". In: *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 3543–3549.

Fearnhead, P. (2006). "Exact and Efficient Bayesian Inference for Multiple Changepoint Problems". In: *Statistics and Computing* 16.2, pp. 203–213.

Fedorov, V. V. (1972). *Theory of optimal experiments*. London, Oxford, Boston, New York and San Diego: Academic Press.

Fisher, R. A. (1922). "On the Mathematical Foundations of Theoretical Statistics." In: *Phil. Trans. of the Royal Society* 222, pp. 309–368.

Fox, D., W. Burgard, and S. Thrun (1998). "Active Markov localization for mobile robots". In: *Robotics and Autonomous Systems*. Autonomous Mobile Robots 25.3–4, pp. 195–207.

Freund, Y. and R. E. Schapire (1997). "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". In: *Journal of Computer and System Sciences* 55.1, pp. 119–139.

Freund, Y., H. S. Seung, E. Shamir, and N. Tishby (1997). "Selective Sampling Using the Query by Committee Algorithm". In: *Machine Learning Journal* 28.2-3, pp. 133–168.

Fujishige, S. (1978). "Polymatroidal dependence structure of a set of random variables". In: *Information and Control* 39.1, pp. 55–72.

Guizzo, E. (2011). "Fukushima Robot Operator Writes Tell-All Blog". In: *IEEE Spectrum*.

Hans, A., D. Schneegaß, A. M. Schäfer, and S. Udluft (2008). "Safe exploration for reinforcement learning." In: *Proc. of the Europ. Symp. on Artificial Neural Networks*, pp. 143–148.

Harnad, S. (1990). "The symbol grounding problem". In: *Physica D: Nonlinear Phenomena* 42.1, pp. 335–346.

Hausman, K., F. Balint-Benczedi, D. Pangercic, Z.-C. Marton, R. Ueda, K. Okada, and M. Beetz (2013). "Tracking-based interactive

segmentation of textureless objects". In: *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pp. 1122–1129.
Hausman, K., S. Niekum, S. Osentoski, and G. S. Sukhatme (2015). "Active Articulation Model Estimation through Interactive Perception". In: *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pp. 3305–3312.
Hester, T., M. Lopes, and P. Stone (2013). "Learning exploration strategies in model-based reinforcement learning". In: *Proc. of the Int. Conf. on Autonomous Agents and Multi-Agent Systems*, pp. 1069–1076.
Hinton, G. E. and R. R. Salakhutdinov (2006). "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786, pp. 504–507.
Hinton, G. E., S. Osindero, and Y.-W. Teh (2006). "A Fast Learning Algorithm for Deep Belief Nets". In: *Neural Computation* 18.7, pp. 1527–1554.
Höfer, S., T. Lang, and O. Brock (2014). "Extracting Kinematic Background Knowledge from Interactions Using Task-Sensitive Relational Learning". In: *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pp. 4342–4347.
Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. Cambridge, Mass.: The MIT Press.
Hwa, R. (2004). "Sample Selection for Statistical Parsing". In: *Computational Linguistics* 30.3, pp. 253–276.
Igel, C. and M. Toussaint (2005). "A No-Free-Lunch theorem for non-uniform distributions of target functions". In: *Journal of Mathematical Modelling and Algorithms* 3.4, pp. 313–322.
Iwata, S., L. Fleischer, and S. Fujishige (2001). "A combinatorial strongly polynomial algorithm for minimizing submodular functions". In: *Journal of the ACM (JACM)* 48.4, pp. 761–777.
Jonschkowski, R. and O. Brock (2015). "Learning state representations with robotic priors". In: *Autonomous Robots* 39.3, pp. 407–428.
Kaelbling, L. P. and T. Lozano-Pérez (2013). "Integrated Task and Motion Planning in Belief Space". In: *Int. Journal of Robotics Research* 32.9-10, pp. 1194–1227.

Kaelbling, L. P., M. Littman, and A. R. Cassandra (1998). "Planning and acting in partially observable stochastic domains". In: *Artificial Intelligence Journal* 101, pp. 99–134.

Kaelbling, L. P., M. L. Littman, and A. W. Moore (1996). "Reinforcement learning: A survey". In: *Journal of Artificial Intelligence Research (JAIR)*, pp. 237–285.

Katz, D., A. Orthey, and O. Brock (2014). "Interactive Perception of Articulated Objects". In: *Experimental Robotics*. Ed. by O. Khatib and G. S. V. Kumar. Vol. 79. Springer Tracts in Advanced Robotics. Berlin and Heidelberg: Springer, pp. 301–315.

Katz, D., Y. Pyuro, and O. Brock (2008). "Learning to manipulate articulated objects in unstructured environments using a grounded relational representation". In: *Proc. of the Int. Conf. on Robotics: Science and Systems (R:SS)*, pp. 254–261.

Kearns, M. and S. Singh (2002). "Near-Optimal Reinforcement Learning in Polynomial Time". In: *Machine Learning Journal* 49.2-3, pp. 209–232.

Klingbeil, E., A. Saxena, and A. Ng (2010). "Learning to Open New Doors". In: *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 2751–2757.

Kolter, J. Z. and A. Ng (2009). "Near-Bayesian Exploration in Polynomial Time". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 513–520.

Körner, C. and S. Wrobel (2006). "Multi-class Ensemble-Based Active Learning". In: *Proc. of the European Conf. on Machine Learning (ECML)*. Ed. by J. Fürnkranz, T. Scheffer, and M. Spiliopoulou. Lecture Notes in Computer Science. Berlin and Heidelberg: Springer, pp. 687–694.

Kroemer, O. B., R. Detry, J. Piater, and J. Peters (2010). "Combining active learning and reactive control for robot grasping". In: *Robotics and Autonomous Systems*. Hybrid Control for Autonomous Systems 58.9, pp. 1105–1116.

Kulick, J., R. Lieck, and M. Toussaint (2015). "The Advantage of Cross Entropy over Entropy in Iterative Information Gathering". In: *arXiv e-prints* 1409.7552v2 (stat.ML).

Kulick, J., S. Otte, and M. Toussaint (2015a). "Active Exploration of Joint Dependency Structures". In: *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pp. 2598–2604.

Kulick, J., S. Otte, and M. Toussaint (2015b). "Robots Solving Serial Means-Means-End Problems". In: *RSS Workshop on Combining AI Reasoning and Cognitive Science with Robotics*.

Kulick, J., M. Toussaint, T. Lang, and M. Lopes (2013). "Active learning for teaching a robot grounded relational symbols". In: *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 1451–1457.

Kullback, S. and R. A. Leibler (1951). "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1, pp. 79–86.

Lang, K. J. and E. B. Baum (1992). "Query learning can work poorly when a human oracle is used". In: *Proc. of the Int. Joint Conf. on Neural Networks (IJCNN)*, pp. 335–340.

Lang, T. and M. Toussaint (2010). "Planning with noisy probabilistic relational rules". In: *Journal of Artificial Intelligence Research (JAIR)* 39.1, pp. 1–49.

Lang, T., M. Toussaint, and K. Kersting (2012). "Exploration in Relational Domains for Model-based Reinforcement Learning". In: *Journal of Machine Learning Research (JMLR)* 13.1, pp. 3725–3768.

LaValle, S. M. and J. J. Kuffner (2000). "Rapidly-exploring random trees: Progress and prospects". In: *Proc. of Workshop on the Algorithmic Foundations of Robotics*.

Lemaignan, S., R. Ros, E. A. Sisbot, R. Alami, and M. Beetz (2011). "Grounding the Interaction: Anchoring Situated Discourse in Everyday Human-Robot Interaction". In: *Int. Journal of Social Robotics* 4.2, pp. 181–199.

Leung, C., S. Huang, and G. Dissanayake (2006). "Active SLAM using model predictive control and attractor based exploration". In: *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 5026–5031.

Leung, C., S. Huang, and G. Dissanayake (2008). "Active SLAM in structured environments". In: *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pp. 1898–1903.

Lewis, D. D. and J. Catlett (1994). "Heterogeneous uncertainty sampling for supervised learning". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 148–156.

Lewis, D. D. and W. A. Gale (1994). "A Sequential Algorithm for Training Text Classifiers". In: *Proc. of the Ann. Int. Conf. on Research and Development in Information Retrieval*, pp. 3–12.

Lopes, M., T. Lang, M. Toussaint, and P.-Y. Oudeyer (2012). "Exploration in model-based reinforcement learning by empirically estimating learning progress". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 206–214.

Lungarella, M., G. Metta, R. Pfeifer, and G. Sandini (2003). "Developmental robotics: a survey". In: *Connection Science* 15, pp. 151–190.

Martinez-Cantin, R., N. de Freitas, A. Doucet, and J. A. Castellanos (2007). "Active Policy Learning for Robot Planning and Exploration under Uncertainty." In: *Proc. of the Int. Conf. on Robotics: Science and Systems (R:SS)*, pp. 321–328.

Martín-Martín, R. and O. Brock (2014). "Online Interactive Perception of Articulated Objects with Multi-Level Recursive Estimation Based on Task-Specific Priors". In: *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 2494–2501.

Mazzola, C. and M. Giunti (2012). "Dynamical Systems on Monoids: Toward a General Theory of Deterministic Systems and Motion". In: *Methods, Models, Simulations and Approaches Towards a General Theory of Change*, pp. 173–185.

McCallum, A. and K. Nigam (1998). "Employing EM in pool-based active learning for text classification". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 359–367.

Meltzoff, A. N. and M. K. Moore (1997). "Explaining facial imitation: A theoretical model". In: *Early Development & Parenting* 6.3-4, pp. 179–192.

Milch, B., B. Marthi, S. Russell, D. Sontag, D. L. Ong, and A. Kolobov (2007). "BLOG: Probabilistic Models with Unknown Objects". In: *Introduction to Statistical Relational Learning*, pp. 373–426.

Mitchell, T. M. (1977). "Version Spaces: A Candidate Elimination Approach to Rule Learning". In: *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 305–310.

Mitchell, T. M. (1982). "Generalization as search". In: *Artificial Intelligence Journal* 18.2, pp. 203–226.

Močkus, J. (1972). "On a Bayesian method for seeking an extremum". In: *Automatika i vychislitelnaja tekhnika. (Russian)* 3.

Močkus, J. (1975). "On Bayesian methods for seeking the extremum". In: *Tech. Conf. on Optimization Techniques*. Springer, pp. 400–404.

Močkus, J. (1989). "The Bayesian Approach to Local Optimization". In: *Bayesian Approach to Global Optimization*. Mathematics and Its Applications 37. Springer, pp. 125–156.

Moldovan, T. M. and P. Abbeel (2012). "Safe exploration in markov decision processes". In: *arXiv e-prints* 1205.4810 (cs.LG).

Montesano, L. and M. Lopes (2012). "Active learning of visual descriptors for grasping using non-parametric smoothed beta distributions". In: *Robotics and Autonomous Systems* 60.3, pp. 452–462.

Moravec, H. (1988). *Mind Children: The Future of Robot and Human Intelligence*. Harvard University Press. 228 pp.

Moulin-Frier, C. and P.-Y. Oudeyer (2013a). "Exploration strategies in developmental robotics: a unified probabilistic framework". In: *Joint Int. Conf. on Development and Learning and Epigenetic Robotics (ICDL)*. IEEE, pp. 1–6.

Moulin-Frier, C. and P.-Y. Oudeyer (2013b). "Learning how to reach various goals by autonomous interaction with the environment: unification and comparison of exploration strategies". In: *Multidisciplinary Conf. on Reinforcement Learning and Decision Making (RLDM)*.

Murphy, K. P. (2007). *Conjugate Bayesian analysis of the Gaussian distribution*. Technical Report. University of British Columbia.

Nagatani, K. and S. Yuta (1995). "An Experiment on Opening-Door-Behavior by an Autonomous Mobile Robot with a Manipulator". In: *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 45–50.

Nemhauser, G. L., L. A. Wolsey, and M. L. Fisher (1978). "An analysis of approximations for maximizing submodular set functions". In: *Mathematical Programming* 14.1, pp. 265–294.

Ngo, H. Q., M. D. Luciw, A. Förster, and J. Schmidhuber (2012). "Learning skills from play: Artificial curiosity on a Katana robot arm." In: *Proc. of the Int. Joint Conf. on Neural Networks (IJCNN)*, pp. 1–8.

Ngo, H., M. D. Luciw, A. Förster, and J. Schmidhuber (2013). "Confidence-based progress-driven self-generated goals for skill acquisition in developmental robots". In: *Frontiers in Psychology* 4, pp. 188–206.

Niekum, S., S. Osentoski, C. G. Atkeson, and A. G. Barto (2015). "Online Bayesian Changepoint Detection for Articulated Motion Models". In: *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pp. 1468–1475.

Otte, S., J. Kulick, M. Toussaint, and O. Brock (2014). "Entropy Based Strategies for Physical Exploration of the Environment's Degrees of Freedom". In: *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 615–622.

Pasula, H. M., L. S. Zettlemoyer, and L. P. Kaelbling (2007). "Learning symbolic models of stochastic domains". In: *Journal of Artificial Intelligence Research (JAIR)*, pp. 309–352.

Patil, A., D. Huard, and C. J. Fonnesbeck (2010). "PyMC: Bayesian stochastic modelling in Python". In: *Journal of Statistical Software* 35.4, pp. 1–81.

Pennestri, E., M. Cavacece, and L. Vita (2005). "On the computation of degrees-of-freedom: a didactic perspective". In: *Proc. of Int. Design Engineering Technical Conf. and Computers and Information in Engineering Conf.* American Society of Mechanical Engineers, pp. 1733–1741.

Peterson, L., D. Austin, and D. Kragic (2000). "High-level control of a mobile manipulator for door opening". In: *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 2333–2338.

Poultney, C., S. Chopra, Y. L. Cun, et al. (2006). "Efficient learning of sparse representations with an energy-based model". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 1137–1144.

Rasmussen, C. and C. Williams (2006). *Gaussian processes for machine learning*. MIT Press.

Ratnaparkhi, A., J. Reynar, and S. Roukos (1994). "A Maximum Entropy Model for Prepositional Phrase Attachment". In: *Proc. of the Workshop on Human Language Technology*. Stroudsburg, PA, USA, pp. 250–255.

Rolf, M., J. Steil, and M. Gienger (2010). "Goal Babbling Permits Direct Learning of Inverse Kinematics". In: *Trans. on Autonomous Mental Development* 2.3, pp. 216–229.

Roy, N. and A. McCallum (2001). "Toward Optimal Active Learning through Monte Carlo Estimation of Error Reduction". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 441–448.

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. (2015). "Imagenet large scale visual recognition challenge". In: *Int. Journal of Computer Vision* 115.3, pp. 211–252.

Sagiv, M., T. Reps, and R. Wilhelm (2002). "Parametric shape analysis via 3-valued logic". In: *Trans. on Programming Languages and Systems* 24.3, pp. 217–298.

Salganicoff, M., L. H. Ungar, and R. Bajcsy (1996). "Active Learning for Vision-Based Robot Grasping". In: *Machine Learning Journal* 23.2-3, pp. 251–278.

Scheffer, T., C. Decomain, and S. Wrobel (2001). "Active Hidden Markov Models for Information Extraction". In: *Advances in Intelligent Data Analysis*. Ed. by F. Hoffmann, D. J. Hand, N. Adams, D. Fisher, and G. Guimaraes. Lecture Notes in Computer Science 2189. Berlin and Heidelberg: Springer, pp. 309–318.

Schmidhuber, J. (1991a). "A possibility for implementing curiosity and boredom in model-building neural controllers". In: *From Animals to*

Animats: Proc. of the Int. Conf. on Simulation of Adaptive Behavior, pp. 222–227.

Schmidhuber, J. (1991b). "Curious model-building control systems". In: *Proc. of the Int. Joint Conf. on Neural Networks (IJCNN)*, pp. 1458–1463.

Schumacher, C., M. D. Vose, and L. D. Whitley (2001). "The No Free Lunch and Problem Description Length". In: *Proc. of the Genetic and Evolutionary Computation Conf*. Pp. 565–570.

Schütze, H., E. Velipasaoglu, and J. O. Pedersen (2006). "Performance Thresholding in Practical Text Classification". In: *Proc. of the Int. Conf. on Information and Knowledge Management*. ACM, pp. 662–671.

Searle, J. R. (1980). "Minds, brains, and programs". In: *Behavioral and Brain Sciences* 3.03, pp. 417–424.

Settles, B. (2010). *Active Learning Literature Survey*. Computer Sciences Technical Report 1648. University of Wisconsin–Madison.

Settles, B. (2012). *Active Learning*. Ed. by R. Brachman, W. Cohen, and T. Dietterich. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool.

Settles, B. and M. Craven (2008). "An Analysis of Active Learning Strategies for Sequence Labeling Tasks". In: *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, pp. 1070–1079.

Seung, H. S., M. Opper, and H. Sompolinsky (1992). "Query by Committee". In: *Proc. of the Annual Conf. on Computational Learning Theory*, pp. 287–294.

Shannon, C. E. (1948). "A Mathematical Theory of Communication". In: *The Bell System Technical Journal* 27, pp. 379–423.

Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis (2016). "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587, pp. 484–489.

Sim, R. and N. Roy (2005). "Global A-Optimal Robot Exploration in SLAM". In: *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pp. 661–666.

Singh, S., R. Lewis, A. Barto, and J. Sorg (2010). "Intrinsically Motivated Reinforcement Learning: An Evolutionary Perspective". In: *Trans. on Autonomous Mental Development* 2.2, pp. 70–82.

Srivastava, S., N. Immerman, and S. Zilberstein (2009). "Abstract Planning with Unknown Object Quantities and Properties." In: *Proc. of the Symp. on Abstraction, Reformulation and Approximation (SARA)*, pp. 143–150.

Stachniss, C., G. Grisetti, and W. Burgard (2005). "Information Gain-based Exploration Using Rao-Blackwellized Particle Filters." In: *Proc. of the Int. Conf. on Robotics: Science and Systems (R:SS)*, pp. 65–72.

Sturm, J., C. Stachniss, and W. Burgard (2011). "A Probabilistic Framework for Learning Kinematic Models of Articulated Objects". In: *Journal of Artificial Intelligence Research (JAIR)* 41.2, pp. 477–526.

Sutton, R. S. (1990). "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 216–224.

Sutton, R. S. and A. G. Barto (1998). *Reinforcement Learning: An Introduction*. Cambridge, Mass.: MIT Press.

Taigman, Y., M. Yang, M. Ranzato, and L. Wolf (2014). "Deepface: Closing the gap to human-level performance in face verification". In: *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 1701–1708.

Thrun, S., S. Thayer, W. Whittaker, C. Baker, W. Burgard, D. Ferguson, D. Hahnel, D. Montemerlo, A. Morris, Z. Omohundro, C. Reverte, and W. W (2004). "Autonomous exploration and mapping of abandoned mines". In: *Robotics Automation Magazine* 11.4, pp. 79–91.

Thrun, S. B. (1992). *Efficient Exploration In Reinforcement Learning*. Technical Report. Carnegie Mellon University.

Tomanek, K. and U. Hahn (2009). "Semi-supervised Active Learning for Sequence Labeling". In: *Proc. of the Joint Conf. of the Ann. Meeting of*

the ACL and the Int. Joint Conf. on Natural Language Processing of the AFNLP, pp. 1039–1047.

Tomanek, K. and F. Olsson (2009). "A web survey on the use of active learning to support annotation of text data". In: *Proc. of the Workshop on Active Learning for Natural Language Processing*, pp. 45–48.

Toussaint, M. (2015). "Logic-Geometric Programming: An Optimization-Based Approach to Combined Task and Motion Planning". In: *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 1930–1936.

Toussaint, M., N. Plath, T. Lang, and N. Jetchev (2010). "Integrated motor control, planning, grasping and high-level reasoning in a blocks world using probabilistic inference". In: *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pp. 385–391.

Van Hoof, H., O. Kroemer, and J. Peters (2014). "Probabilistic Segmentation and Targeted Exploration of Objects in Cluttered Environments". In: *Trans. on Robotics* 5, pp. 1198–1209.

Van Hoof, H., O. Kroemer, H. Ben Amor, and J. Peters (2012). "Maximally informative interaction learning for scene exploration". In: *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 5152–5158.

Von Neumann, J. (1951). "Various Techniques Used in Connection With Random Digits". In: *Journal of Research of the National Bureau of Standards* 3, pp. 36–38.

Wallace, B. C., K. Small, C. E. Brodley, and T. A. Trikalinos (2010). "Active Learning for Biomedical Citation Screening". In: *Proc. of the Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pp. 173–182.

Wolpert, D. H. (1996). "The Lack of a Priori Distinctions Between Learning Algorithms". In: *Neural Computation* 8.7, pp. 1341–1390.

Wolpert, D. H. (2001). "The supervised learning no-free-lunch Theorems". In: *In Proc. of Online World Conference on Soft Computing in Industrial Applications*, pp. 25–42.

Wolpert, D. H. and W. G. Macready (1995). *No Free Lunch Theorems for Search*. Technical Report SFI-TR-95-02-010. Santa Fe Institute.

Wolpert, D. H. and W. G. Macready (1997). "No free lunch theorems for optimization". In: *Trans. on Evolutionary Computation* 1.1, pp. 67–82.

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Dissertation selbstständig und ausschließlich auf der Grundlage der in der Arbeit angegebenen Hilfsmittel und Hilfen verfasst habe.

Johannes Kulick
Berlin, den 2. April 2016