Methods for Scalable and Safe Robot Learning

Olov Andersson

Linköping University
Department of Computer and Information Science
Artificial Intelligence and Integrated Computer Systems
SE-581 83 Linköping, Sweden

Linköping 2017
This is a Swedish Licentiate’s Thesis

Swedish postgraduate education leads to a doctor’s degree and/or a licentiate’s degree. A doctor’s degree comprises 240 ECTS credits (4 years of full-time studies). A licentiate’s degree comprises 120 ECTS credits.

Edition 1:1

© Olov Andersson, 2017
ISSN 0280-7971
URL http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-138398/

Published articles have been reprinted with permission from the respective copyright holder.
Typeset using \LaTeX

Printed by LiU-Tryck, Linköping 2017
ABSTRACT

Robots are increasingly expected to go beyond controlled environments in laboratories and factories, to enter real-world public spaces and homes. However, robot behavior is still usually engineered for narrowly defined scenarios. To manually encode robot behavior that works within complex real-world environments, such as busy work places or cluttered homes, can be a daunting task. In addition, such robots may require a high degree of autonomy to be practical, which imposes stringent requirements on safety and robustness.

The aim of this thesis is to examine methods for automatically learning safe robot behavior, lowering the costs of synthesizing behavior for complex real-world situations. To avoid task-specific assumptions, we approach this from a data-driven machine learning perspective. The strength of machine learning is its generality, given sufficient data it can learn to approximate any task. However, being embodied agents in the real-world, robots pose a number of difficulties for machine learning. These include real-time requirements with limited computational resources, the cost and effort of operating and collecting data with real robots, as well as safety issues for both the robot and human bystanders.

While machine learning is general by nature, overcoming the difficulties with real-world robots outlined above remains a challenge. In this thesis we look for a middle ground on robot learning, leveraging the strengths of both data-driven machine learning, as well as engineering techniques from robotics and control. This includes combining data-driven world models with fast techniques for planning motions under safety constraints, using machine learning to generalize such techniques to problems with high uncertainty, as well as using machine learning to find computationally efficient approximations for use on small embedded systems.

We demonstrate such behavior synthesis techniques with real robots, solving a class of difficult dynamic collision avoidance problems under uncertainty, such as induced by the presence of humans without prior coordination. Initially using online planning offloaded to a desktop CPU, and ultimately as a deep neural network policy embedded on board a 7 cm quadcopter.

This work has been supported by the Wallenberg Autonomous Systems and Software Program, the Swedish Foundation for Strategic Research (SSF) project Symbicloud and the ELLIIT Excellence Center at Linköping-Lund for Information Technology, in addition to those sources already acknowledged in the individual papers.
Acknowledgements

I am grateful for the support of my primary advisor Patrick Doherty, giving me the time and resources to explore interesting research directions. I am also grateful for the fruitful discussions with my co-advisor Mattias Villani. To all my colleagues in AIICS, a big thanks for the positive work environment. Special thanks to the UASTech lab in particular. Mariusz Wzorek, Piotr Rudol, Cyrille Berger, Tommy Persson and Karol Korwel, you have been a fount of knowledge on UAVs and field robotics. Finally, I am grateful to my parents and my late friend Thomas, this would not have happened without you.

Linköping, 16 June 2017
Olov Andersson
Contents

Abstract iii
Acknowledgments v
Contents vi
List of Figures viii
List of Tables x

1 Introduction 1
1.1 Motivation .......................................... 1
1.2 Methodology ......................................... 2
1.3 Problem Definition ................................. 2
1.4 Outline of Contributions ........................... 5
1.5 Publication List ...................................... 6
1.6 Thesis Outline ...................................... 6

2 Learning and Inference 7
2.1 Preliminaries ........................................ 7
2.2 Learning and Parameteric Models ................. 8
2.3 Gaussian Processes .................................. 9
2.4 Deep Learning ....................................... 11

3 Optimization-based Behavior and Control 15
3.1 Decision Making and Rational Agents ............ 15
3.2 Taxonomy of Sequential Decision Making ......... 18
3.3 Global Policy Approaches ......................... 19
3.4 Local Trajectory Approaches ...................... 22
3.5 Trajectory-Guided Policy Learning ............... 26

4 Discussion 29
4.1 Summary of Contributions ......................... 29
4.2 Limitations and Future Work ....................... 30
List of Figures

1.1 Two quadcopters used as test platforms during the research. . . . . 3
1.2 A rational agent interacting with its environment using an internal representation. .......................... 4

2.1 Sparse approximation to a $f : \mathbb{R}^1 \rightarrow \mathbb{R}^1$ Gaussian process inferred from data points in black. Predicted mean and 95% probability intervals in blue and cyan respectively. The red crosses represent the inducing points of the sparse approximation. ......................... 11
2.2 Architecture of a fully-connected neural network with two hidden layers for learning some function $f : \mathbb{R}^3 \rightarrow \mathbb{R}^2$. Artificial neurons are shown in white and their inputs are represented by arrows. .......................... 12

5.1 Forward simulation of planned actions for a quadcopter control agent told to reach position $x = 1$. While it plans in a 10-dimensional state-action space, only the target variable is shown. 49
5.2 The success rate per episode in terms of the percentage that completed without violating pole or track constraints, averaged over 50 runs. ......................................................... 51
5.3 Log plot of the final cart distance to the target position while keeping the pole balanced. The results are averaged over 50 runs. . . . 51
5.4 Positional control of quadcopter commanded to fly a rectangle pattern. Blue indicates acceleration, and red deceleration, by getting the quadcopter to tilt in or against the direction of movement. Green is constant velocity. ........................................... 53
5.5 Quadcopter $x,y$ position and velocity while flying the rectangle pattern. The dotted line represents the indoor domain constraint on velocity. ......................................................... 54

6.1 Human prediction safety margins. .............................. 64
6.2 Parametric safety margin. ....................................... 66
6.3 The LinkQuad Quadcopter. ........................................... 70
6.4 The intersection scenario, crossing a street. ......................... 71
6.5 Bayesian policy optimization of safety parameters for the intersection scenario. ........................................... 72
6.6 The warehouse scenario. ................................................... 73
6.7 Bayesian policy optimization of safety parameters for the warehouse scenario. ........................................... 74
6.8 Quadcopter avoiding one human obstacle. .......................... 76
6.9 Quadcopter avoiding two human obstacles. ......................... 76

7.1 Training procedure for deep neural network policy approximations. 87
7.2 The Bitcraze Crazyflie 2.0 nano-quadcopter. ......................... 90
7.3 Safety margin for stochastic collision avoidance. .................... 92
7.4 The warehouse scenario with three obstacles. ......................... 93
7.5 Computational cost of action selection. .............................. 94
7.6 On-board DNN quadcopter experiments. Top shows it dodging one human obstacle. Bottom shows it flying a rectangular pattern while avoiding one human obstacle. ................. 96
List of Tables

6.1 Results from intersection scenario for different algorithms and target safety levels. Actual safety level estimated over 12 h. . . . . 72
6.2 Warehouse scenario for different target safety levels. Actual safety level estimated over 12h. . . . . . . . . . . . . . . . . . . . . 74
6.3 Optimization results from warehouse scenario with different numbers of humans . . . . . . . . . . . . . . . . . . . . . . . . . . . 75
7.1 One non-cooperative moving obstacle. . . . . . . . . . . . . . . 93
7.2 Three non-cooperative moving obstacles. . . . . . . . . . . . . . 94
Introduction

Over the next few decades, robots are projected to go beyond the controlled environments of laboratories and factories, to increasingly enter real-world public spaces and homes. Several forecasts (Arntz and Zierahn 2016; Frey and Osborne 2017) claim that society is on the verge of a wider artificial intelligence (AI) driven wave of automation with wide-spread economic impact. However, robot behavior is still usually engineered for narrowly defined scenarios. Manually encoding robot behavior that works within complex real world environments, like busy work places or cluttered homes, can be a daunting task. In addition, such robots may require a high degree of autonomy to be practical, which imposes stringent requirements on safety and robustness.

1.1 Motivation

The aim of the research in this thesis is to find methods for automatically learning robot behavior, lowering the costs for synthesizing behavior for complex real-world situations. To avoid task-specific assumptions we approach this from a data-driven machine learning perspective.

Machine learning has been a major driving force behind the optimism for AI in recent years. By enabling machines to learn from data, the task of encoding correct behavior can itself be automated or greatly eased. With the aid of big data sets, this has resulted in rapid advances for software agents in domains such as internet advertising, recommender systems, as well as
1. INTRODUCTION

object, text and speech recognition. The strength of machine learning is its
generality, given sufficient data it can learn to approximate any task.

However, being embodied agents in the real-world, robots pose a number
of difficulties for machine learning. These include real-time requirements
with limited computational resources, the cost and effort of operating and
collecting data with real robots, as well as safety issues for both the robot
and human bystanders.

While machine learning is general by nature, overcoming the difficulties
with real-world robots outlined above remains a challenge. In this thesis
we look for a middle ground on robot learning, leveraging the strengths of
both data-driven machine learning, as well as engineering techniques from
robotics and control. This includes combing data-driven world models with
fast techniques for planning motions under safety constraints, using machine
learning to generalize such techniques to problems with high uncertainty, as
well as using machine learning to find computationally efficient approxima-
tions for use on small embedded systems.

1.2 Methodology

To guide and validate this research, it was conducted in collaboration with
the field robotics group of the Artificial Intelligence and Integrated Com-
puter Systems (AIICS) division, specializing in autonomous unmanned
aerial vehicles (UAVs). Throughout the course of this work, significant effort
was put into evaluating the proposed methods with real robots, or initially,
reasonable simulations thereof. Two UAVs used for the experiments are pic-
tured in Figure 1.1. The LinkQuad is a quadcopter originally developed for
advanced autonomous outdoor scenarios, while the Bitcraze Crazyflie is a
simpler 7 cm platform that we found useful for testing flight behaviors. Be-
ing both fragile and of limited computational capacity, UAVs can be said to
embody the characteristic difficulties of robotics, posing a challenging do-
main for learning algorithms.

1.3 Problem Definition

The problem of learning behavior can be formalized as learning a policy
function $\pi(x) = u$, that given some world state vector $x$ comes up with a
suitable decision vector $u$. The world state $x$ includes the robot and environ-
ment states necessary to solve the problem, e.g. poses and dynamic state like
velocity. The decisions $u$ can include low-level control signals to actuators,
or higher level actions such as "turn left".

There are predominantly two types of approaches to learning a behavior
policy $\pi(x)$ in robotics, either it is learned directly from examples of correct
decisions $(x_1, u_1^\star), \ldots, (x_N, u_N^\star)$, or the robot is imbued with agency so that it
1.3. Problem Definition

(a) The LinkQuad Quadcopter.  
(b) The Bitcraze Crazyflie 2.0

Figure 1.1: Two quadcopters used as test platforms during the research.

can find a good solution on its own, only given feedback in terms of some performance metric.

The former case usually requires a teacher demonstrating examples of correct behavior which the learning algorithm imitates (Billard et al. 2008). This can be a simpler approach when an intuitive mapping from human to robot anatomy exists. However, since it lacks any understanding of the task beyond the examples, scaling to more complex behavior can require an impractical amount of input from a human teacher.

An advantage of instead imbuing the robot with agency is that a human only needs to encode the task objective, for example a cost based on distance to a goal state $x_g$. By making the robot a rational agent with the task encoded as its objective, it can leverage some internal representation of the world to come up with a suitable behavior on its own (Bertsekas 2005; Bertsekas 2012). While demonstrations can, when applicable, still be useful to guide the decisions of a rational agent, we argue that some degree of agency is highly beneficial to learning more complex behaviors.

To automate synthesis of new behaviors, in this thesis we make robots rational agents that are awarded some cost or reward when interacting with their environment. Furthermore, we will assume that it has some internal model of what the effect of its actions are, and by which it can plan a sequence of actions to satisfy its objective. A focus of this work is also on maintaining real-world safety requirements via inclusion of constraints on the agent’s decision process. A high-level overview of this can be seen in Figure 1.2.

Formally, this can be defined as solving the probabilistic sequential decision problem

$$\arg\min_{\pi(x)} \mathbb{E}_{x_{t:t+H}}[c(\tau_{t:t+H})]$$

subject to

$$\Pr(g(\tau_{t:t+H}) \geq 0) > p.$$  

(1.1)
1. INTRODUCTION

Starting from some world state \( p(x_t) \), the state is assumed to transition probabilistically according to some world model \( p(x_{t+1}|x_t, u_t) \). For example, in an autonomous driving context such a world model might describe the dynamics of the car on the road, but also other relevant objects in the world, such as the behavior of other cars or pedestrians. The next state depends both on the previous state \( x_t \) and the chosen action \( u_t \) of the agent at each time step. Any such sequence of actions \( u_t, ..., u_{t+H-1} \) will result in a sequence of states \( x_{t+1}, ..., x_{t+H}, \) which we together define as a trajectory \( \tau_{t:t+H} = [(x_{t+1}, u_t), ..., (x_{t+H}, u_{t+H-1})] \) through state-action space. An example of such a trajectory might be steering wheel angle and resulting position, orientation and angular velocity of an autonomous car. Solving such a problem entails learning a policy, a function that maps each state to an action \( \pi(x) = u \). The goal state of the task is encoded in a cost function \( c(\tau_{t:t+H}) \) for state and actions over some planning horizon \( H \), which could be a simple distance to a goal state. Due to the uncertain nature of the dynamics and state of real robots, such trajectories through state-action space are stochastic, and costs have to be evaluated as expected values \( E[c(\tau_{t:t+H})] \) over the trajectory distribution.

Although this definition makes no distinction between discrete and continuous state or actions, in this thesis we focus on continuous behavior and control that arises from robots acting in the real world.
In many real-world applications, task and safety constraints $g(\tau_{t,t+H}) \geq 0$ also need to be satisfied. These can include control saturation, speed limits, or geometric constraints for e.g. collision avoidance. Since a trajectory is probabilistic, constraints can similarly only be guaranteed with some probability $p$.

Sequential decision problems of the type above is studied in several overlapping disciplines, including robotics, control, operations research and machine learning. Reinforcement learning is one of the most general frameworks, additionally allowing the world model $p(x_t|x_{t-1}, u_{t-1})$ to be unknown, subsuming the decision problem in Equation 1.1. In practice, the world transition model will often be at least partially unknown with real robots. This introduces uncertainty in how the robot responds to control commands and what the results are in its environment. As we approach this from a machine learning perspective, reinforcement learning is the point from which we started, and this will be used as a general framework for the contributions in this thesis.

Learning $\pi(x)$ is generally a difficult problem for high-dimensional continuous domains, and real robots additionally have requirements on safety and real-time operation, that are rarely studied in the learning literature.

1.4 Outline of Contributions

All the contributions in this thesis can be seen as finding approximations to different parts of the constrained reinforcement learning problem in 1.1, attempting to make it more suitable for use with real robots. The contributions are briefly summarized below. Additional details and context is provided in the following chapters.

- We propose to use online constrained trajectory optimization for safe reinforcement learning, combining recent advances in MPC solvers and non-parametric model learning techniques.
- We propose a novel machine learning technique to convert probabilistic safety constraints in uncertain environments to computationally tractable deterministic constraints.
- We propose a resampling technique for learning deep neural network approximations of problems with safety constraints.
- We use the above techniques to demonstrate real-time behavior with real quadcopters for difficult dynamic collision avoidance problems under uncertainty. Initially using online planning offloaded to a desktop CPU, and ultimately as a deep neural network policy embedded on board a 7 cm nano-quadcopter.
1. INTRODUCTION

1.5 Publication List


1.6 Thesis Outline

The rest of the thesis is structured as follows. In Chapter 2 we briefly cover statistical inference and the supervised learning techniques used throughout the thesis. This includes learning models of robot dynamics, cost surfaces for Bayesian optimization as well as deep neural network policy approximations. In Chapter 3 we summarize the prerequisite background for the optimization techniques used to solve the decision problems in 1.1 and put them in the context of related work in the area. Finally, Chapter 4 concludes with a discussion of the results so far and explores promising avenues for future work. Part II of this thesis includes the relevant publications, slightly edited for presentation purposes.
2 Learning and Inference

This chapter will give a brief introduction to statistical learning and inference. As the literature on this subject is vast, this chapter is not intended as an exhaustive overview. The aim is only to provide context for the techniques used in this thesis.

2.1 Preliminaries

We take a probabilistic view of machine learning. This has the advantages of giving succinct definitions with the rigor of probability theory, and many other techniques can be seen as approximations to the full probabilistic machinery. Here we only present a high level overview, for more details we refer to Bishop et al. (2006) for a machine learning perspective, and Gelman et al. (2003) for a perspective from Bayesian statistics. In the following we assume some basic knowledge of probability theory. For brevity, we often casually refer to probability mass and density functions as probability distributions, and assume use of appropriate measure. We also make use of the common shorthand of writing \( p(x) \) instead of \( Pr(X = x) \).

In the probabilistic view, we assume there is some probability distribution \( p(x_1, ..., x_n) \) over all variables of interest \( X_1, ..., X_n \). This can include both robot and world state based on noisy sensor readings, but also any model parameters that need to be learned. Learning is then reduced to probabilistic inference in the model.
2. Learning and Inference

To facilitate inference we primarily rely on two rules derived from probability theory. Assume that we have some partitioning of the set of variables into two partitions \( a \) and \( b \), each representing one or more variables \( X_i \).

The **marginalization** rule takes a joint distribution \( p(a, b) \) and marginalizes out nuisance variables \( b \) that are not of interest, to produce a marginal distribution only over \( a \). Intuitively, this can be understood as summing or integrating over all outcomes of nuisance variables \( b \) for the remaining variables,

\[
p(a) = \int p(a, b) \, db. \tag{2.1}
\]

The **product rule**, sometimes called the chain rule of probability, allows you to factorize a joint probability distribution into a product of conditional distribution and marginal,

\[
p(a, b) = p(a|b)p(b). \tag{2.2}
\]

This is useful as defining a joint probability distribution directly can be difficult. Particularly for discrete variables, due to the joint outcome space being the Cartesian product of all included variables. Many parts of a model are often more naturally expressed as conditional distributions and simple prior marginal distributions. This also allows use of known conditional independences, such that \( p(a|b, c) = p(a|b) \), which can greatly simplify the model. For models with complicated structure, these independence assumptions can be made explicit by defining them via a probabilistic graphical model, such as a Bayesian network (Koller and Friedman 2009).

In addition, when we want to solve decision problems where the cost of a choice depends on unknown variables, it is convenient to work with expected values. These represent the average cost over all possible outcomes of the unknown variables, weighted by their respective probability

\[
\mathbb{E}_{p(a)}[g(a)] = \int g(a)p(a) \, da, \tag{2.3}
\]

where \( g(a) \) is allowed to be an arbitrary function of the unknown variables.

2.2 Learning and Parameteric Models

Probabilistic models are typically used to make some inferences from data. In robotics the data is usually collected from noisy sensor readings and we may be interested in making inferences about the surrounding environment or the robot itself. A model is constructed as a joint probability distribution, suitably factorized using the product rule, where some variables may be observed and others are unknown. Assuming that variables in partition \( b \) are observed we want to compute \( p(a|b) \), set the observed variables in the joint
distribution to their data values and marginalize out any remaining unobserved variables from $a$ that we are not interested in.

Even when we do not need to marginalize out nuisance variables, we may still need marginalization to normalize the posterior $p(a|b)$. To see this, consider a simple supervised learning example where we want to approximate some function $y = f_\theta(x)$ parameterized by $\theta$, given examples of inputs $x$ and noisy outputs $y$. It is convenient to model the noisy $y$ as a conditional probability given $x$ and $\theta$, with a prior on $\theta$, resulting in $p(y|x, \theta)p(\theta)$. However, what we want is the posterior distribution of $\theta$ conditioned on the data. For this we can use the eponymous Bayes’ rule of Bayesian statistics,

$$p(\theta|y, x) = \frac{p(y|x, \theta)p(\theta)}{p(y)}.$$  (2.4)

It is easy to see that Bayes’ rule is just an application of the product rule, choosing another way to factorize the distribution $p(y, \theta|x)$. As noted, the normalizing factor $p(y)$ is unknown here and requires marginalization to compute.

The integrals of marginalization in Equation (2.1) or expectations in Equation (2.3) often lack a closed-form solution in practice. Approximations generally have to be used when these are non-linear or non-Gaussian. In addition, while the dimensionality of robot state $x$ is often in the range of a dozen, to learn advanced models it is not uncommon to require inference over millions of model parameters. High precision approximations such as numerical quadrature are intractable for high-dimensional learning problems, and even sampling can quickly get infeasible for the real-time requirements of robots. Variational approximations can sometimes be applicable, but this is still very much an active research topic.

The simple maximum-a-posteriori (MAP) approximation instead optimizes on the posterior distribution and uses a mode as point estimate $\hat{\theta}$. These are efficient to compute but optimistic due to disregard of parameter uncertainty. Care must be taken to avoid negative side effects such as overfitting. By foregoing use of prior, this also reduces to the classical maximum likelihood estimate (MLE). These are ubiquitous in practice, and throughout this thesis we use both of these for parts of models where probabilistic inference lacks closed-form solution.

### 2.3 Gaussian Processes

Parametric models easily allow you to incorporate prior knowledge, for example in supervised learning of $y = f(x)$ we may know the mathematical structure of the function $f$ and only need to estimate a few parameters. However, many real-world phenomena are difficult or too complicated to derive from prior knowledge. In that case we may want to learn
a function with a minimum of assumptions. This is the case in the classical view of reinforcement learning, where the world is treated as unknown. One class of completely data-driven models are Gaussian processes. These are non-parametric in the sense that the complexity can increase with the amount of data. Gaussian processes are a popular choice of probabilistic non-parametric models where both the mean and uncertainty of $f$ can be computed in closed form under some assumptions.

Formally, given a training set of $n$ observations on input variables $X \in \mathbb{R}^{d \times n}$ and outputs $y \in \mathbb{R}^d$ where $y$ is corrupted by additive noise $y = f(x) + \epsilon$, one can put a Gaussian process prior on the latent function $f(x)$ and attempt to learn $f(x)$ from data.

A Gaussian process (GP) is defined as a set of random variables, any finite number of which have a joint Gaussian distribution (Rasmussen and Williams 2006). The process is completely specified by a mean function $m(x)$ and a covariance function $k(x, x')$ that are functions of the input variables. For clarity we assume that all data is standardized with mean zero, turning the covariance function into

$$k(x, x') = \mathbb{E}[f(x)f(x')].$$ (2.5)

This defines the covariance of function values at pairs of input points, such that the distribution of any points on $f(x)$ is completely specified by a joint multivariate Gaussian.

A common covariance function also used throughout this thesis is the squared-exponential, which decays with the distance between the points

$$k_f(x, x') = \sigma_f^2 \exp \left[ -\frac{1}{2} \sum_{i=1}^d \left( \frac{x_i - x'_i}{\ell_i} \right)^2 \right].$$ (2.6)

The parameters $\sigma_f^2$ and $\ell_i$ represent signal variance and per-dimension length-scales respectively, which together make up the hyperparameters $\theta_f$. The hyperparameter part of the model lacks closed-form solution and MAP inference is used instead. The length-scales can adapt to attenuate unneeded input dimensions from the problem, known as automatic relevance determination.

The covariance function is used to define matrices of covariances between function values for the input points $X$, any set of prediction points $X_*$ as well as their cross-covariances. We denote these as $\Sigma_{X,X}$, $\Sigma_{X_*}X_* \Sigma_{X,X_*}$ and $\Sigma_{X_*}X_*$ respectively. As points on $p(f(x)|X, \theta_f)$ are multivariate Gaussian and the noise is Gaussian with standard deviation $\sigma_n^2$, the data likelihood $p(y|X, \theta_f, \sigma_n^2)$ is jointly Gaussian with covariance $\Sigma_{X,X} + \sigma_n^2 I$, and the predictive distribution at any new points $x_*$ admit closed form solution.

The non-parametric nature of Gaussian processes, coupled with the closed-form inference on $f(x)$, makes them very flexible models while remaining resistant to overfitting even for small data sets. Data efficiency is
highly desirable for robotics due to the cost of operating and maintaining real hardware. Unfortunately, the non-parametric nature of GPs means that computational complexity of training and prediction increases with the number of data points, being \( O(n^3) \) and \( O(n) \) respectively. This quickly gets infeasible even on a desktop computer, and real robots tend to have more limited resources. Much effort has been invested in finding cheaper sparse approximations, and this is still an active research area. Such sparse processes, visualized in Figure 2.1, use a small number of inducing inputs as anchor points for an approximation while still aiming to retain reasonable accuracy.

Most of our contributions are in the context of approximations to the decision problem in Equation 1.1, and will be further described in Chapter 3. Briefly, in Paper I we suggest a novel constrained reinforcement learning method using sparse GPs to efficiently learn world models. We use the FITC approximation (Snelson and Ghahramani 2006) which has been shown to work well with the square-exponential kernel in applications, and has a computational complexity of \( O(nm^2) \) and \( O(m) \) for training and prediction respectively. In Paper II we also propose a novel technique for learning agent behavior under probabilistic safety constraints, relying on Gaussian process based Bayesian optimization.

### 2.4 Deep Learning

While Gaussian processes are highly accurate for small data sets, and sparse approximations can handle larger ones, training with very large data sets
2. Learning and Inference

Figure 2.2: Architecture of a fully-connected neural network with two hidden layers for learning some function \( f : \mathbb{R}^3 \rightarrow \mathbb{R}^2 \). Artificial neurons are shown in white and their inputs are represented by arrows.

can still be a problem. Even more importantly, the generalization ability of "black-box" covariance functions like the popular square-exponential scale poorly with input dimension. Unless automatic relevance determination can find only a moderate number of needed input variables, learning a black-box function is simply not practical. Deep learning attempts to address this problem by learning hierarchical, so called deep, representations. The by-far most popular models are multi-layered neural networks (Bengio, Goodfellow, and Courville 2015). These are composed of layers of parameterized artificial neurons, each a linear combination of its inputs passed through a non-linear activation function. Network structure can range from generic fully-connected feed-forward networks exemplified by Figure 2.2, to variants imposing additional structural assumptions tailored to a particular domain. The idea is that a hierarchical representation should be able to naturally learn abstractions to better exploit existing structure in the task, and therefore decompose the problem into a composition of smaller ones. Mathematically, this can be seen as learning a composition of functions \( f(g(h(x))) \), where hopefully no component needs to be as complicated as in a "flat" representation \( f(x) \).

While technically parametric, neural networks are very flexible black-box models like Gaussian processes. Parameter inference for such deep architectures usually lack a closed-form solution, and point estimates typically have to be found via optimization. A great advantage of deep learning is the existence of efficient software for offloading training to a GPU, allowing for larger data sets and models than may have previously been practical. We use Tensorflow (Abadi et al. 2016), a recent graph-based language for numerical computation from Google.
In Paper III we use deep learning to learn approximations to behavior policies $\pi_\theta(x)$ for the agent decision problem in Equation (1.1). Our approach allows us to generate large quantities of training data, which was a natural fit for GPU-based training. We use fully-connected feed-forward deep neural networks (DNN), with the parameter vector $\theta$ representing the weights $W_i$ and biases $b_i$ of neurons for each layer $i$.

Each DNN layer $i$ was defined by

$$y_{i+1} = h_i(W_i y_i + b_i)$$

(2.7)

with network input $y_1 = x$, output $y_N = \hat{\pi}(x)$, and $h_i(x)$ is the (vector) activation function for layer $i$. We use the popular ReLU activation function, which for each neuron $j$ is the scalar function

$$h_{i,j}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

(2.8)

for all hidden layers $i$, except the output layer $y_N$, which is linear.

Deep neural network training is currently an area under intense research. Most rely on stochastic gradient approximations from mini-batches. We found adaptive momentum (Kingma and Ba 2015) useful, and used dropout (Srivastava et al. 2014) to mitigate overfitting. While the training was done offline on a GPU, we managed to implement the resulting network for real-time use on a nano-quadcopter microcontroller. Due to the rising trend of GPU miniaturization for embedded systems, we expect such methods to be of increasing viability for robotics.
Here we provide a brief overview of different approaches to optimization-based behavior and control and their application to robotics. As noted in Section 1.3, there is a large body of work on solving sequential decision problems, crossing over into several different fields. We do not aim to give a complete account of this body of work here, but only to provide context for the publications in this thesis. We refer the interested reader to Thrun, Burgard, and Fox (2005) and Deisenroth, Neumann, Peters, et al. (2013) for a robotics perspective, Bertsekas (2005) and Bertsekas (2012) for control, Powell (2007) from operations research. A classical introduction to reinforcement learning in discrete domains can be found in Sutton and Barto (1998).

The rest of this chapter is structured as follows. First we provide an overview of the AI perspective on decisions problems, and suitable delimitations for applications in robotics. Then we provide a taxonomy of the relevant methods from a robot learning perspective, ultimately leading up our contributions being positioned with regard to related work in sections 3.4 and 3.5.

3.1 Decision Making and Rational Agents

Decision problems under uncertainty are usually formalized as finding some action $u$ minimizing the expected cost over uncertain problem variables $x$,

$$\arg\min_u \mathbb{E}_x [c(x, u)]. \quad (3.1)$$
An agent obeying the above decision rule also satisfies common axioms of rationality, and is therefore a *rational agent*. However, many problems exhibit sequential dependence between earlier decisions and the state variables \( x \). For example, the location of an autonomous car can only be indirectly manipulated through a sequence of steering, acceleration and braking actions. As an agent interacts with the world, each action will change the world state, but the agent needs to *plan* a sequence of actions to fulfill its objective. It therefore has to solve a harder sequential decision problem,

\[
\arg \min_{\pi(x)} \mathbb{E}_{x_{t:t+H}} \left[ c(\tau_{t:t+H}) \right].
\]  

(3.2)

Here \( \tau_{t:t+H} \) represents a trajectory through state-action space \( \{(x_{t+1}, u_t), \ldots, (x_{t+H}, u_{t+H-1})\} \), from the current state \( x_t \) over some planning horizon \( H \). As any continuous-time problem can be transcribed into discrete time via numerical integration, we only consider discrete-time problems. The world state therefore evolves in discrete time steps \( x_t \) according to some probability distribution \( p(x_{t+1} | x_t, u_t) \). The uncertainty nature of this world transition model can result from imperfect models of robot actuation or other parts of the environment, including human behavior. In addition, all current and future state \( p(x_{t+1:t+H} | o_{0:t+H}) \) carry uncertainty, as they have to be estimated from noisy sensor observations \( o_t \). Taken together, this results in a probability distribution over future state based on the current action choice. The sequential nature of the problem implies that the best decision in the current step depends on decisions made in subsequent states, which now also have to be considered over a multitude of possible outcomes, and so on. Due to this branching of the event space, *probabilistically optimal* decisions usually have to be sought jointly as a function of the state. For brevity, we here make the assumption that such a decision *policy* is invariant of time, \( \pi(x) \). Finding an optimal policy \( \pi(x) \) for non-trivial state and action spaces can be very difficult.

In addition we want to solve this problem under constraints. This may include arbitrary task requirements, but due to the fragility and dangers involved with real robots, the main focus here is on safety of the robot and human bystanders. Due to state uncertainty, we can only satisfy such constraints with high probability. This finally results in the probabilistically constrained, or chance-constrained, sequential decision problem

\[
\arg \min_{\pi(x)} \mathbb{E}_{x_{t:t+H}} \left[ c(\tau_{t:t+H}) \right] \quad \text{subject to} \\
\Pr(g(\tau_{t:t+H}) \geq 0) > p.
\]  

(3.3)

A particular class of such problems that we study in this thesis is avoiding collisions with humans working in close proximity, such as in warehouse,
office, home or even autonomous outdoor scenarios. The uncertainty here can also stem from the difficulty of predicting human behavior, and we want to satisfy constraints with high probability.

**Delimitations**

We here make a distinction between *optimal* behavior, and the *optimization-based* methods in this thesis. Our aim is to find approximate techniques for robot behavior, and we approach this from an AI and machine learning perspective. By contrast, optimal control often puts more focus on optimality and provable guarantees, traditionally with regard to time or fuel consumption. While this can translate to robot behavior, real-word robots often have several hard-to-define aspects which can only be approximately encoded. For example, exactly by which criteria should a future household robot tasked with doing the dishes or cooking be judged? Subtle issues can intrude even when planning simple motions. A time-optimal humanoid robot navigating an office would go for maximum acceleration and de-acceleration, running through the corridors. Humans rarely do this even when in a hurry, as even simple motion is governed by complex factors such as social norms. One solution may be to learn the social norms through the cost or reward function, presumably like humans do. Many classical reinforcement learning techniques also assume unknown reward function (Sutton and Barto 1998), but this is rarely used with real robots due to the increased data requirements this implies. We leave such considerations for future work and assume we have some reasonable but approximate objective.

The second reason for not chasing optimality is that we assume both state estimates and internal models of the environment are imperfect, and therefore any predictions are imperfect. This includes the estimated uncertainty of such predictions and therefore even a probabilistically optimal plan or policy will not be optimal in practice. Furthermore, due to computational limitations of real robots, there is little hope of even finding exact solutions for non-trivial problems. In consideration of these real-world issues, we do not expect to actually find or prove optimality, and it is not a major focus of the techniques in this thesis.

While this may seem like a strong limitation, agent behavior only needs to be adequate for its real world task. Consider that humans sometimes make demonstrably poor decisions, and therefore are not optimal either. While humans are often modeled as rational agents in the social sciences, it has long been known that human rationality is bounded (Simon 1955), although the exact mechanics appear difficult to pin down. While biological plausibility is not an objective of this thesis, this is a strong indication that optimality can be seen as more of an ambition rather than a strict requirement.
3. Optimization-based Behavior and Control

3.2 Taxonomy of Sequential Decision Making

Sequential decision problems can be stated in several different ways, but many differ only in terminology.

For example, in classical reinforcement learning it is customary to define the objective function of the agent in terms of maximizing rewards or utility instead of minimizing costs, but these are equivalent. Likewise, classical reinforcement learning originated with discrete problems, whereas states \( s \in S \) and actions \( a \in A \) are discrete. Robot sensors and actuators are better represented by metric spaces, and therefore it is common to define these as real vectors for state \( x \), and actions as control inputs \( u \) instead. Throughout this thesis we sometimes use these interchangeably. Reinforcement learning traditionally focused on Markov Decision Processes (MDP). Such decision processes are assumed to satisfy a Markov property, in that state transitions are conditionally independent of all earlier state \( p(x_{t+1}|u_t, x_t, \ldots, u_1, x_0) = p(x_{t+1}|x_t, u_t) \). In the following we also make the Markov assumption, such that the world state for each \( t \) factors into conditional distributions \( p(x_{t+1}|x_t, u_t) \) from some initial prior \( p(x_0) \).

An assumption of perfect observability is also common. Since sensors on real robots are noisy, this never holds exactly in robotics, but it can still work well as long as state uncertainty is reasonably invariant to future action choices, an assumption we make here. A difficult exploration versus exploitation problem arises when this does not hold. In that case an agent has to weigh the value of information gathering actions against trying to solve its task and possibly failing due to high uncertainty. Such problems are called Partially Observable MDPs (POMDPs), and are even more difficult to compute. Although MDP methods can be used to solve POMDPs via e.g. state augmentation (Thrun, Burgard, and Fox 2005), it is not considered in this thesis. The world model \( p(x_{t+1}|x_t, u_t) \) is often considered unknown in reinforcement learning, and learning that does introduce an exploration problem. However, instead of attempting an optimal POMDP solution, it is common to reduce it to an MDP with randomized or sub-optimal exploration actions. In the randomized case, the policy can also be seen as a probability distribution conditioned on the state \( \pi(u|x) \), rather than a deterministic function of it.

Functional differences in sequential decision problems include which parts of the problem are treated as probabilistic, which parts are treated as unknown, and the parameterization of unknown parts. Another distinction is if constraints are included, ignored, or emulated via costs in the objective function.

In robotics applications, the world model tends to be at least partially unknown. This can include the dynamics of the robot itself, but also its environment. While e.g. classical control usually focuses on problems where the model can be mostly derived from physics, classical reinforcement learn-
3.3. Global Policy Approaches

(Sutton and Barto 1998) traditionally takes a *tabula rasa* view, assuming no prior knowledge about the environment at all. Both have their merit, as deriving models from physics is time consuming and liable to making incorrect assumptions, while learning models is expensive in terms of computation and data. We take a machine learning perspective, and the first major distinction in such approaches is how they approach the model problem. Either the problem is decomposed into separate model learning and planning problems, or a *model-free* method is used that circumvents an explicit representation of the model. As stated in the introduction, our main interest is in *model-based* methods as they tend to be more data efficient, a key desiderata for real robots. However, learned or derived models can still be used as simulators with model-free methods, which is not uncommon in practice.

A second delineation can be made based on planning horizon. An underlying problem can be *episodic* with a finite duration or terminal state, or it can be a persistent non-episodic task. Non-episodic tasks are of more interest for behavior in autonomous robots. Yet, even if the task is non-episodic, the planning horizon of the agent, as defined in Equation (3.3), only needs to be sufficiently long to solve the underlying tasks. The objective can be defined either in terms of a *fixed* time horizon $H$ or an *infinite* horizon $H \rightarrow \infty$, where the latter requires geometric discounting for the cost to be finite, or use of an average cost.

The third and perhaps most important distinction can be made between different approaches to solving the resulting decision problem. Do we directly search for a *global policy* $\pi(x)$, or do we use dynamic programming to compute a *value function* $V(x)$ as proxy for the policy? Another alternative is to instead look for *local policies* $\pi_t(x)$ valid only around a trajectory $x_{t:t+H}$, which would then have to be recomputed online if the robot diverges. Finally, we could also use some combination of the above.

As an exhaustive overview of reinforcement learning is beyond the scope of this thesis, we will focus on the methods currently popular for robot learning and contrast them to our contributions.

3.3 Global Policy Approaches

The most straightforward approach to solving Equation (3.3) is *policy search*, optimizing directly on the policy $\pi(x)$. If the problem is discrete, the policy can be enumerated in a table. If it is continuous, the optimal policy can be approximated by a family of policies $\pi_\theta(x)$ parameterized by parameter vector $\theta$. Non-trivial instances of this problem lack closed-form solution and global optima guarantees. Even optimizing on $\pi_\theta(x)$ to find a local optima is difficult for non-trivial horizons. The expected cost results in difficult integrals, and for the policy to be powerful enough to represent a good solution, one may have to use a black-box approximation such as a neural
network. Most work on policy search (Kober and Peters 2012; Deisenroth, Neumann, Peters, et al. 2013) in robotics are model-free reinforcement learning methods. Model-free methods can sidestep both the issue of estimating a world model \( p(x_{t+1}|x_t, u) \), as well as some complexity in the objective, by numerically evaluating policy performance on sampled state transitions and rewards from the robot in the world. These can also be used with simulations if a world model is available, which can be beneficial in robotics due to the large number of samples these approaches typically need.

Perhaps the simplest approach is to use finite-difference numerical differentiation or derivative-free optimization techniques on the policy parameters \( \theta \). Over the last two decades a large number of more advanced approaches have been proposed, many trying to exploit some structure in the problem. The classical REINFORCE algorithm (Williams 1992) exploits that under a stochastic policy, the gradient of policy parameters can be rewritten as an expectation that can be estimated via model-free Monte Carlo simulations. This results in an optimization problem over (parameterized) probability distributions \( \pi_\theta(u|x) \),

\[
\arg \min_{\pi_\theta(u|x)} \mathbb{E}_{\tau_{t:t+H}|\pi_\theta} \left[ c(\tau_{t:t+H}) \right].
\] (3.4)

Such stochastic policies can serve as an integration of exploration strategy into the policy search problem, usually realized by adding local Gaussian noise on a deterministic policy function like a neural network. Much work builds on this, for example to better leverage the sequential structure (Deisenroth, Neumann, Peters, et al. 2013). Modern extensions often better exploit the geometry of the problem and regulate step size (Kakade 2002; Schulman et al. 2015).

As model-free policy search requires a large number of interactions with the system, most successes for use with real robots have relied on domain knowledge to craft suitable policies with a moderate number of parameters (Kober and Peters 2012) and reasonable initial values. Model-based approaches do not need to interact with the system to update its policy, and are known to be much more data efficient (c.f. sec. 6, Kober and Peters (2012). As noted, the objective is unfortunately difficult to compute in the general case. A special case is exploited by PILCO (Deisenroth and Rasmussen 2011), which under some moderately restrictive assumptions on state, reward function and Gaussian process model, actually can compute policy gradients analytically.

Another way to solve the decision problem is via dynamic programming. While this approach has been dominant in classical reinforcement learning, it has seen less use in robotics due to the difficulties of continuous state and actions. Consider the infinite-horizon case of \( H \to \infty \) from \( t = 0 \) in an MDP
with $\lambda$-discounted sum rewards,

$$\arg\min_{\pi(x)} \mathbb{E}_{x_0:x} \left[ \sum_{t=0}^{\infty} \lambda^t c(x_t, \pi(x_t)) \right].$$  \hfill (3.5)

The reward sequence under a fixed policy from any state $x$ converges to a value as a geometric sum for $\lambda < 1$. Under MDP assumptions, the value of each state can be represented by a time-invariant global value function $V_\theta(x)$. Further, the value of the current state can be decomposed into an immediate cost plus the expected value of the next state, $V_\theta(x_t) = \mathbb{E}_{x_{t+1}|x_t} [c(x_t, \pi(x_t)) + \lambda V_\theta(x_{t+1})]$. Under a known world transition model $p(x_{t+1}|x_t, u_t)$, this can be used to compute the value function for a particular policy, policy evaluation, via a fixed-point iteration. Value iteration extends this to directly compute the value function $V_\theta^*(x)$ of the, similarly time-invariant, optimal policy $\pi^*(x)$ via the fixed-point iterations $i$,

$$V_{\theta}^{i+1}(x_t) = \min_{u_t} \mathbb{E}_{x_{t+1}} \left[ c(x_t, u_t) + \lambda V_{\theta}^i(x_{t+1}) \right].$$  \hfill (3.6)

The optimal policy itself can then be recovered by local one-step choices on the optimal value function. Solving for the value function can also be viewed as solving a dual to the original policy optimization problem (Kober and Peters 2012).

Most value function approaches in reinforcement learning are model-free. Instead of computing the expected value of the next state under an uncertain world model, they sample the world transition by interacting with the world, either in simulation or with a real system. A famous example is Q-learning (Watkins and Dayan 1992), where the eponymous Q-function is a value function augmented with action, $Q_\theta(x, u)$ to facilitate solving for the minimum action $u$ in a sampling framework

$$Q_{\theta}^{i+1}(x_{t-1}, u_{t-1}) = \min_{u_t} c(x_{t-1}, u_{t-1}) + \lambda Q_{\theta}^i(x_t, u_t).$$  \hfill (3.7)

The Q-function is incrementally updated after each sampled transition $(x_{t-1}, u_{t-1}, x_t)$, as if the best action will be chosen in the current step. Q-learning is therefore an off-policy algorithm. It is not required to actually follow the optimal policy and can easily incorporate exploratory moves. Sufficient exploration is generally of great importance for value-function approaches to converge in practice.

A fundamental bottleneck with value-function approaches is in the representation of the value function itself. An exhaustive enumeration of all state is often infeasible, as a naive discretization of continuous state spaces suffer from the curse of dimensionality. A major focus of classical reinforcement learning has been to find approximations to such value functions while maintaining favorable convergence properties of the fixed-point iterations.
This has shown itself to be difficult for the general case, and more powerful approximations such as neural networks are known to exhibit convergence issues, see Kober and Peters (2012) for a discussion. While dynamic programming is globally optimal, these approximations can also introduce local minima. With the rise of deep learning there has been a resurgence of interest in value function approaches, notably using deep neural networks for Q-functions (Mnih et al. 2015) to learn video game behavior directly from image inputs. Robotics usually requires continuous action spaces, which introduces additional difficulties, such as for minimizing $u$. Recently, (Lillicrap et al. 2015) combined deep neural-network value-functions with policy gradients for continuous actions spaces. It has also been augmented with local linear approximations to the world models (Gu et al. 2016), somewhat mitigating the high data requirements of model-free reinforcement learning.

Such deep neural-network approaches to value-function approximation is a promising direction, which currently enjoys considerable research activity, but they have so far not had as much success in robotics. A recent benchmarking study (Duan et al. 2016) found that while the DDPG algorithm of (Lillicrap et al. 2015) initially improved faster than direct policy gradient approaches, they suffered from erratic convergence and sometimes fell behind their more traditional counterparts. In addition, it curiously found that both policy gradient and value function approaches had trouble converging to useful behavior on a navigation task. This task included non-static goal states in addition to areas to avoid, which are ubiquitous in robotics, and reminiscent of the collision avoidance problem we study in Paper II. We suspect that this highlights an underappreciated problem with comparing simulation results of policy learning techniques. As the difficulty of the task depends on the dimensionality of the data distribution $p(\tau)$, the problem gets easier if the data is confined to a small sub-space. Under fixed policy $\pi(x)$, simulating a robot without noise from a fixed starting position will always result in the same trajectory, a one-dimensional data distribution. However, a real robot would quickly diverge from the trajectory due to modelling error alone. In essence, it may just have learned an open-loop trajectory. Autonomous robots may additionally have to be able to deal with a wider range of possible situations than many simulation benchmarks assume.

### 3.4 Local Trajectory Approaches

Since automatically learning a good global policy or value function approximation is difficult, it seems reasonable to consider local approximations. It is well-known that the decision problem in Equation (3.2) can be solved analytically for the special case of sum-quadratic costs and linear world model with Gaussian noise. This is known as the linear-quadratic or linear-quadratic-Gaussian regulator (LQR/LQG). The resulting time-variant value
functions $V_t(x)$ and policies $\pi_t(x)$ are quadratic and linear respectively (Bertsekas 2005), and dynamic programming can be used to compute each $\pi_t(x)$ via a backwards recursion from $V_{t+T}(x)$. Differential Dynamic Programming (DDP) is a classical trajectory optimization approach for non-linear sum-quadratic systems by means of iterative quadratic model approximations (Mayne 1966). This results in the optimization problem

$$\arg \min_{\pi_t(x), \ldots, \pi_{t+H}(x)} \mathbb{E}_{x_{t+1:t+H}} \left[ \sum_{i=1}^{t+H} c(x_i, \pi_i(x_i)) \right]. \quad (3.8)$$

The popular iterative LQR (iLQR) instead makes linear model approximations and is computationally cheaper at $O(d^3 H)$ per iteration (Todorov and Li 2004), where $d$ is the dimension of the state space. These can also be used in model-based reinforcement learning, which separates the problem into model learning and trajectory planning, through the full state space, to satisfy the agent’s objective. This is attractive due to the combined data efficiency of model-based approaches, and the comparatively low computational complexity of local trajectory approaches compared to learning a global policy. The downside is that they will need to be recomputed online unless you only need to follow canned trajectories in static environments. Sometimes this is sufficient, and has been used to great effect for mimicking aerobatics maneuvers (Abbeel, Coates, and Ng 2010).

Another problem is that when computing behavior for real robots we want to incorporate safety constraints, for example to avoid falling over, or colliding with human bystanders. Usually there is some state we want the robot to reach, and regions of the state space we want it to avoid. Unfortunately, such constraints generally ruin the simple quadratic value function above. In addition, as the iLQR cost function has to be well approximated by a positive-definite quadratic form, constraints are difficult to effectively emulate as penalties on that. A linear cost is possible, but can result in very erratic convergence and requires careful line-search. Indeed, iLQR is known to perform a type of Gauss-Newton on the objective, which is problematic with non-quadratic objectives.

In applied, or numerical, optimal control, it is common to solve deterministic problems with constraints. The optimization community has invested much effort into reliable general-purpose constrained solvers (Nocedal and Wright 2006) that are also applicable to constrained trajectory optimization. While classical optimal control mainly focused on offline generation of trajectories, model-predictive control (MPC) aims to solve these deterministic
constrained trajectory problems under a sliding fixed horizon as in our definition.

\[
\arg \min_{\tau_{t+H}} \ c(\tau_{t+H})
\]
subject to
\[
\begin{align*}
x_{i+1} &= f(x_i, u_i), \forall i. \\
x_t &= x_{\text{init}} \\
g(\tau_{t+H}) &\geq 0.
\end{align*}
\] (3.9)

For certain problem classes, mostly quadratic problems with linear constraints, the control community has also devised fast low-latency solvers for MPC.

Turning to the contributions in this thesis, our initial contribution in Paper I is to leverage these new solvers for the reinforcement learning task under safety constraints. We divide the problem into learning non-linear models using data efficient and computationally attractive sparse Gaussian processes (c.f. Chapter 2.3), and use a recent fast interior-point QP solver (Domahidi et al. 2012) with linear time complexity in planning horizon \(H\). A QP solver can be used to solve arbitrary non-linear problems via sequential QP iterations (Nocedal and Wright 2006). SQP, and non-linear programming in general, are large topics on their own and will not be covered here. Although the general non-linear case can involve many intricacies, we found a simple variant sufficient for our problem. While constrained trajectory optimization has historically been prohibitively expensive, recent advances in solvers and hardware allows real-time operation on a desktop CPU even with \(H = 100\). To the best of our knowledge, real-time constrained trajectory optimization has not been previously explored by the reinforcement learning community. Handling safety and risk in general has only started to attract interest recently (García and Fernández 2015), and then usually in the form of risk-averse objectives and discrete MDPs rather than the explicit safety constraints in continuous domains that are desirable for robotics.

As Gaussian process models are resistant to overfitting even for small data sets, they are unusually data efficient for being non-parametric models, which has previously been used to great effect with policy search in PILCO (Deisenroth and Rasmussen 2011). Although the idea of using trajectory optimization for model-based reinforcement learning was raised some time ago in an often overlooked paper by Atkeson (1998), there has recently been a surge of interest in this area. Two other papers simultaneous to our work also combined Gaussian processes with trajectory optimization (Boedecker et al. 2014; Pan and Theodorou 2014), but did not consider constraints.

A remaining issue is the impact on optimality for determinizing the problem. On unconstrained problems where the dynamic programming approach is applicable, the LQG actually converges to the same solution. It
3.4. Local Trajectory Approaches

is well known that under the separation principle, a linear quadratic Gaussian problem can be reduced to a deterministic LQR problem. Additionally, the iterative LQR can be seen as doing Gauss-Newton optimization on a deterministic objective. This implies that as long as the noise is reasonably small, the model is reasonably linear and the cost reasonably quadratic locally around the solution, a determinized problem can be a good approximation to the full stochastic problem in Equation (3.3). Although reinforcement learning has traditionally focused on stochastic properties, MPC, like much of numerical optimal control, is usually explicitly deterministic. As this is also seeing increased use in real applications, one may conclude that this assumption is often not a big problem in practice. One issue that we did encounter is that of constraints. Seen as an objective penalty, these are very non-quadratic, and a determinized problem will be optimistic with regard to constraints. This can be disastrous to safety constraints, and in Paper I we just suggested manually adding a safety margin to mitigate this.

In Paper II we examine in greater detail such deterministic approximations of safety constraints for stochastic scenarios. We focus on collision avoidance scenarios with hard-to-predict moving obstacles like human bystanders. Such collision avoidance is considered a hard problem due to the dynamics and uncertainty involved. Both robot and human dynamics need to be taken into account, and without making assumptions on prior coordination, the robot has to guarantee safety under possibly unexpected behavior from the humans. Collision avoidance is a high-risk scenario and such safety constraints need to hold with high accuracy, raising the need for a more sophisticated approach to handling constraints in stochastic environments. For linear-Gaussian problems, or problems that can be approximated as such, uncertainty can be propagated in closed-form. This also allows easier approximations for some probabilistic constraints, e.g., Blackmore and Ono (2009) and Vitus and Tomlin (2011). These typically use the predictive distribution, or incorporate linear feedback laws. We found that the predictive distribution is infeasibly pessimistic in scenarios with moving obstacles. Some kind of model of controller recourse also needs to be included, e.g., if a pedestrian at some point suddenly turns towards the robot, it has a chance to react to this event and adjust its course. While recourse in the form of linear control policies can admit closed-form solution for linear-Gaussian system, this is unsuitable for the non-convex and constrained case. In addition to obstacle constraints, a real-world robot often has saturation constraints on actuators and task level constraints such as maximum velocity.

To satisfy both probabilistic safety constraints and dynamically feasible trajectory planning, within real-time computational requirements, in Paper II we propose a novel combination of deterministic trajectory optimization and policy search. We introduce a parameterization of a soft-constrained MPC program such that it can be seen as a policy πθ(x), and whose parameters θ can be learned by policy search. By leveraging recent advances
in constrained Bayesian optimization (Gelbart, Snoek, and Adams 2014; Hernández-Lobato et al. 2015), such a policy can automatically be tuned to satisfy probabilistic safety constraints \( \Pr(g(\tau_{t+H}) \geq 0) > p \) with high probability. The resulting program is then a determined approximation to the stochastic problem. While sub-optimal in objective, it more importantly satisfies the safety constraints and allows real-time operation. We demonstrate this in simulation as well as with the LinkQuad quadcopter. To the best of our knowledge, this is the first success with tackling the full stochastic decision problem of Equation (3.3) in real-time for non-cooperative moving obstacles.

3.5 Trajectory-Guided Policy Learning

In this thesis we set out to find automatic and scalable techniques for robot learning, characterized by the probabilistic decision problem in Equation (3.3). These techniques need to be efficient in terms of computation and data requirements, as well as allow inclusion of safety constraints for use with real robots. While the parameterized trajectory-solver approximations in Paper II allowed real-time solutions to difficult collision avoidance scenarios under uncertainty, this required a powerful CPU. Even though such constrained trajectory optimization can be linear in the time horizon, it is typically at least cubic in the number of state dimensions. For many complex behaviors, or smaller embedded platforms, real-time performance appears difficult to attain without relying on domain-specific simplifications.

An advantage of traditional global policy approaches outlined in Section 3.3, is that evaluating the policy can be very fast once it has been learned. For example, if the policy is represented by a deep neural network, choosing an action for a particular state only requires feed-forward evaluation, which computationally is linear in the number of weights. As noted earlier, the major difficulty with global policy approaches is that convergence to a good policy can be both difficult and time consuming.

In robotics, there has recently been a surge of interest in using a local trajectory optimizer to aid learning of global policies represented by deep neural networks. Guided policy search (GPS) introduced the concept by speeding up sampling-based policy search via importance sampling, where the proposal was found by a local trajectory approach (Levine and Koltun 2013a). As is common in policy search (c.f. Section 3.3), this takes a probabilistic perspective of the global policy \( \pi_{\text{NN}}(u|x) \), and the local trajectory policy \( \pi_{\tau}(u|x) \) was derived from approximate local LQG policies \( \pi_{\tau}(x) \). This evolved to a variational solution, where expected cost from a trajectory distribution and a Kullback-Leibler divergence against a global policy are both minimized (Levine and Koltun 2013b). Such formulations result in objectives
3.5. Trajectory-Guided Policy Learning

that jointly optimize trajectories and approximate them with a policy,

\[
\arg \min_{\pi_{\tau}, \pi_{NN}} \mathbb{E}_{\tau_{t:t+H}/\pi_t} \left[ c(\tau_{t:t+H}) \right] + \lambda \, D_{KL}(\pi_t \| \pi_{NN}).
\]  

(3.10)

A range of variations have been proposed, like learning unknown dynamics models (Levine and Abbeel 2014) and to end-to-end learning directly from images (Levine, Finn, et al. 2016). Characteristic for most GPS approaches is that the trajectories and global policy are jointly optimized in an alternating fashion with a penalty on discrepancies.

A deterministic variant was proposed by Mordatch and Todorov (2014). They also noted a connection to directly solving the deterministic trajectory optimization problem in Equation (3.9). If hard constraints are used instead of just penalizing the difference of trajectory actions against the global policy, it reduces to direct deterministic policy optimization over a time horizon.

\[
\arg \min_{\pi_{NN}(x)} c(\tau_{t:t+H})
\]

subject to

\[
\mathbf{u}_t = \pi_{NN}(\mathbf{x}_t), \forall t.
\]  

(3.11)

We are not aware of any published success with such a direct approach. Learning neural networks is currently highly heuristic even in a supervised learning setting. Presumably this could introduce difficulties as part of the more complex problem above. GPS approaches can be somewhat sub-optimal in that they use a simplified objective, penalties instead of a hard constraint, or sometimes both a penalty method and a KL-divergence. Soft constraints with a sufficiently large penalty can of course converge to hard constraints, but choice of penalty appear heuristic in the literature even when convergence could be possible. We are not aware of any published evaluation of the effects of this, but such approaches appear to converge to good policies in practice and can be seen as an approximation. Unfortunately, GPS approaches also tend to come with a number of limitations. First, probabilistic GPS generally makes a number of assumptions for tractability. They rely heavily on linear-quadratic-Gaussian assumptions to compute trajectory distributions via the LQG algorithm, which as previously discussed is problematic with safety constraints. Second, even though the policy is globally defined, they are generally trajectory-centric in that it is jointly optimized with a set of trajectories. For practical reasons, these trajectories have to be reasonably short and drawn from a small number of regions. This is not a problem if the task is to learn e.g. an episodic arm movement from a small set of possible problem configurations. For an autonomous robot in an uncontrolled environment, a policy has to generalize over the wide distribution of states that it can encounter. Unfortunately, the actual state distribution depends on the agent’s behavior in turn, making them difficult to disentangle. This is known
as a data mismatch problem. While the idea of the joint trajectory-policy objective of GPS will afford some protection for the duration of the trajectories, it is unclear how to select representative trajectory starting points for use on the non-episodic problems of autonomous robots. Finally, due to tight coupling between the trajectory and policy learning problems, each GPS variant tends to require its own custom solver.

In Paper III we propose a trajectory-guided policy learning approach that focuses problems with safety constraints. It is a coarser but less restrictive supervised learning transformation than the aforementioned methods, and can easily leverage big data sets sampled from an online trajectory optimizer (MPC) teacher interacting with the world through simulations from transition and sensing distributions. Such long simulation runs allow large amounts of state-action data to be generated on typical stationary non-episodic tasks.

The simplest possible approach would be to learn a policy directly via supervised learning, minimizing a loss function on the observed state action pairs. In theory, a key advantage of the joint optimization of trajectories and policy in GPS is that the policy approximation is made with some regard to the task objective, instead of just minimizing the average difference in action choice. Discrepancies in some regions of the state space may be less desirable than others, for example near safety constraints. However, due to the reliance on LQG, current GPS solvers do not support probabilistic safety constraints, or state constraints in general.

We instead propose a supervised learning transformation that addresses constrained problems directly. We use a risk-aware resampling step to reduce the problem to supervised learning, where the loss function near constraints is amplified to reduce the risk of dangerous constraint violations. This reduces the problem into distinct phases of constrained trajectory optimization and supervised learning. Being agnostic of solvers for either, it allows use of either mature general-purpose software, or custom-tailored solvers when desired. We use the approximation from Paper II to generate trajectory distributions satisfying probabilistic safety constraints. To efficiently learn deep neural network policies with the large data sets generated by our simulation approach, we chose Google’s Tensorflow (Abadi et al. 2016) with GPU acceleration, a recent data flow language for deep learning. The resampling reduction to regular supervised learning from long simulation runs also sidesteps the trajectory-centric data distributions of GPS, and can leverage iterative schemes like DAgGER (Ross, Gordon, and Bagnell 2011) to mitigate remaining data mismatch problems. This approach allows us to reduce the computational costs of the collision avoidance behavior from Paper II by factors of 50 to 500, while retaining safety requirements. We demonstrate the approach in a simulated collision avoidance scenario, as well as by implementing it on-board the microcontroller of a 7 cm nano-quadcopter.
4 Discussion

The aim of this thesis was to examine scalable techniques for safe robot behavior learning. We formalized this as a sequential decision problem for a constrained rational agent, drawing upon techniques from reinforcement learning, robotics and control to find feasible approximations. In this chapter we summarize the contributions and discuss their limitations, as well as potential avenues for future work.

4.1 Summary of Contributions

As noted in the introduction, all of the contributions in this thesis can be seen as finding approximations to different parts of the constrained reinforcement learning problem in 1.1, attempting to make it more suitable for use with real robots. The papers included in this thesis, and their contributions, are briefly summarized below.

- In Paper I we considered approximations to the full reinforcement learning problem with safety constraints. We learned state transitions $p(x_{t+1}|x_t, u_t)$ entirely from data with sparse Gaussian processes and proposed approximating the policy $\pi(x)$ with recent fast constrained trajectory solvers, allowing the inclusion of safety constraints $g(\tau_{t:t+H}) \geq 0$ on the robot. Trajectory optimization is attractive as contrary to global policy approaches, it is guaranteed to converge in time polynomial in the state dimension. With advances in constrained solvers and com-
4. Discussion

putational power, it is increasingly feasible solve these online. Con-
strained online trajectory optimization does not appear to have been
previously considered in the reinforcement learning community. We
evaluated it on a simulated RL cart-pole benchmark and a quadcopter
domain.

- Applying such constrained trajectory solvers requires determinizing the
problem, including the constraints. While solving the full probabilistically
constrained problem is generally intractable, deterministic safety
constraints would need some manual safety margin to not be vio-
lated under uncertainty. In Paper II we addressed this by drawing on
Bayesian optimization and policy search, a traditional RL technique, to
learn parameterized deterministic soft-constraints satisfying the origi-
nal probabilistic constraints with desired confidence, \( \Pr(g(\tau_{t+H}) \geq 0) > p \). We focused on the difficult problem of collision avoidance, under dy-
namics and uncertainty, with non-cooperative moving obstacles. We
demonstrated a safe real-time capable MPC approximation to the full
collision avoidance problem with real quadcopter flights.

- In Paper III we examined using deep learning with a trajectory solver
as teacher to learn faster policy approximations \( \hat{\pi}(x) \). While training a
neural-network can be computationally intensive, evaluating it is only
linear in the number of weights. This is particularly useful for computationally constrained applications such as small robots and other
embedded systems. However, approximating a policy via supervised
learning can result in unsafe behavior due to the recursive structure
of dynamical systems introducing accumulating errors. Strategies like
guided policy search (GPS) aim to reduce these through the use of
tightly coupled learning and trajectory optimization architectures. We
instead proposed an active learning strategy via risk-aware resampling
combined with sampled big data sets to directly reduce error in dan-
gerous regions, prioritizing robot safety requirements while being ag-
nostic of solver for both trajectories and deep learning. This allows use
of mature software for e.g. learning with big data sets such as Google
Tensorflow, and we could use the probabilistically safe solver of Pa-
per II. We demonstrated the efficacy of the approach on the collision
avoidance domain by embedding a deep neural network policy on the
microcontroller of a 7cm nano-quadcopter.

4.2 Limitations and Future Work

We have found novel approximations to the decision problem in Equa-
tion (1.1) that enabled us to learn safe behavior for computationally con-
strained robotic platforms, demonstrated by going from simulation, to mid-
4.2. Limitations and Future Work

sized quadcopter, down to onboard implementation on a nano-quadcopter. A theme throughout this research has been a focus on close integration with real robotic systems such as the LinkQuad and Bitcraze Crazyflie quadcopter platforms. Conceptually, none of the proposed techniques are limited to quadcopters, although some aspects of our implementations are. These can essentially be traced to assumptions on the world transition model $p(x_{t+1}|x_t, u_t)$, or the trajectory optimization we use to approximate the decision problem in Equation (1.1). We started from the reinforcement learning tabula-rasa perspective in Paper I and used Gaussian process models, which is a non-parametric model for initially unknown functions. The sparse approximation we used is also reasonably scalable, but such black box assumptions can still require an impractical amount of data and computation for more difficult problems. In practice, parts of the models can often be derived from physical insight, in the best case requiring only a small number of parameters to be learned. A pragmatic middle ground, that we hope to explore more in future work, is use of semi-parametric models that can incorporate such physical insight into black-box techniques such as Gaussian processes or deep neural networks. For autonomous robots, this should ideally be sufficiently scalable to run online in real-time, which is still a non-trivial problem.

The constrained trajectory optimization problems that we solve online in Paper I and Paper II are the main computational bottlenecks. We used highly optimized MPC solvers from the control community. These predominantly focus on convex sequential QP problems, but can also be used to find local solutions to problems with non-linear models or constraints via SQP iterations (Nocedal and Wright 2006). Unfortunately, this adds to the computational cost, which we found challenging to fit in the limited real-time budget for our applications. To meet our requirements we did not implement the full SQP machinery, which would have added extra computation and engineering complexity. These simplifications worked for our problems, but if performance permits, a mature SQP solver is advisable to reduce the need for domain-specific tuning and verification. For example, in Paper I we used a heuristic constraint relaxation strategy and ran a smaller number of interior-point iterations for the QP solver than may be prudent in general. In retrospect, lowering the decision frequency would have been a better choice, as the anticipatory nature of MPC reduces the need for fast reactive change. In Paper II we additionally needed to add non-convex obstacle constraints, and the performance gain from a linear model assumption seemed to be a worthwhile trade-off for our problem. Real quadcopter flights can be dominated by other sources of error, such as sensing noise and unmodeled effects, as well as latencies due to computation, and possibly wireless communication if either sensing or computation is off-loaded.

In Paper III we avoided this bottleneck by approximating the trajectory optimizer with a deep neural-network control policy. The robot then only
4. Discussion

needs to evaluate the DNN control policy online, removing the real-time requirements from the trajectory optimizer. This opens up the possibility of using either off-the-shelf solvers, or custom-tailored solutions like our probabilistically constrained solver from Paper II. As most work on such deep policy approximations are still research prototypes, we expect that software integration into a mature framework for behavior synthesis could be of wider interest.

A downside of this approach is of course that in our quest for scalable online execution of behavior, we have gone from online trajectory planning to deep policy approximations that need to be computed in batch learning phases. If robot behavior needs to change in response to an event not encountered before, the robot would need to pause and recompute the policy. Online adaptation of behavior policies would therefore also be of great interest for truly autonomous systems in environments that change over time.


Bibliography


BIBLIOGRAPHY


36
Bibliography


Papers

The articles associated with this thesis have been removed for copyright reasons. For more details about these see:

http://urn.kb.se/resolve?urn:nbn:se:liu:diva-138398
Department of Computer and Information Science
Linköpings universitet
Licentiate Theses
Linköpings Studies in Science and Technology
Faculty of Arts and Sciences

No 17  Vojin Plavsic: Interleaved Processing of Non-Numerical Data Stored on a Cyclic Memory. (Available at: FOA, Box 1165, S-581 11 Linköping, Sweden. FOA Report B30062E)
No 73  Ola Strömfora: A Structure Editor for Documents and Programs, 1986.
No 177  Peter Åberg: Design of a Multiple View Presentation and Interaction Manager, 1989.
No 181  Henrik Eriksson: A Study in Domain-Oriented Tool Support for Knowledge Acquisition, 1989.
No 380 Johan Ringström: Compiler Generation for Parallel Languages from Denotational Specifications, 1993.
No 488 Eva Toller: Contributions to Parallel Multiparadigm Languages: Combining Object-Oriented and Rule-Based Programming, 1995.
FHIS 8/95 Dan Fristedt: Metoder i användning - mot förbättring av systemutveckling genom situationell metodkunskap och metodanalys, 1995.
No 538 Staffan Flodin: Efficient Management of Object-Oriented Queries with Late Binding, 1996.
No 546 Magnus Werner: Multidatabase Integration using Polymorphic Queries and Views, 1996.
FH-a 1/96 Mikael Lund: Affärsprocessinriktad förändringsanalys - utveckling och tillämpning av synsätt och metod, 1996.
No 558 Patrik Nordling: The Simulation of Rolling Bearing Dynamics on Parallel Computers, 1996.
No 563 Niclas Andersson: Compilation of Mathematical Models to Parallel Code, 1996.
No 587 Jörgen Lindström: Chefs användning av kommunikationsteknik, 1996.
No 589 Esa Falkenröth: Data Management in Control Applications - A Proposal Based on Active Database Systems, 1996.
No 627 Fredrik Eklund: Declarational Error Diagnosis of GAPLog Programs, 1997.
No 653 Mats Gustafsson: Bringing Role-Based Access Control to Distributed Systems, 1998.
FiF-a 13 Boris Karbonson: Metodanalys för förståelse och utveckling av systemutvecklingsverksamhet. Analys och värdering av systemutvecklingsmodeller och dess användning, 1997.
No 668 Per-Ove Zetterlund: Normering av svensk redovisning - En studie av tillkomsten av Redovisningsrådets rekommendation om koncernredovisning (RR01:91), 1998.
FiF-a 16 Marie-Thérèse Christiansson: Inter-organisatorisk verksamhetsutveckling - metoder som stöd vid utveckling av partnerskap och informationsmarknad, 1998.
No 712 Christina Wenneckstam: Information om immateriella resurser. Investeringar i forskning och utveckling samt i personal inom skogsindustrin, 1999.
No 741 Pawel Pietrzak: Static Incorrectness Diagnosis of CLP (FD), 1999.


Charlotte Björkergen: Learning for the next project - Bearers and barriers in knowledge transfer within an organisation, 1999.


Jean Paul Meynard: Control of industrial robots through high-level task programming, 2000.


Fredrik Elg: Ett dynamiskt perspektiv på individuella skillnader av heuristisk kompetens, intelligens, mentala modeller, mål och konflikter i kontroll av mikrovärlden Mero, 2002.


Pavel Bubnov: Debugging and Structural Analysis of Declarative Equation-Based Languages, 2002.


No 1084 Charlotte Stoitz: Calling for Call Centres - A Study of Call Centre Locations in a Swedish Rural Region, 2004.
FiF-a 74 Björn Johansson: Deciding on Using Application Service Provision in SMEs, 2004.
No 1095 Ulf Johansson: Rule Extraction - the Key to Accurate and Comprehensible Data Mining Models, 2004.
FiF-a 77 Ulf Larsson: Designarbete i dialog - karaktärisering av interaktionen mellan användare och utvecklare i en systemutvecklingsprocess, 2004.
No 1127 Per-Ola Kristensson: Large Vocabulary Shorthand Writing on Stylus Keyboard, 2004.
No 1149 Vaida Jakoniené: A Study in Integrating Multiple Biological Data Sources, 2005.
No 1156 Abdal Rashid Mohamed: High-Level Techniques for Built-In Self-Test Resources Optimization, 2005.
No 1162 Adrian Pop: Contributions to Meta-Modeling Tools and Methods, 2005.
FiF-a 84 Jenny Lagsten: Verksamhetsutvecklande utvärdering i informationssystemprojekt, 2005.
No 1167 Christina Keller: Virtual Learning Environments in higher education. A study of students’ acceptance of educational technology, 2005.
FiF-a 85 Anders Forsman: Standardisering som grund för informationssamverkan och tjänster - En fallstudie baserad på trafikinformationstjänster RDS-TMC, 2005.
FiF-a 86 Jan Olausson: Att modellera uppdrag - grunder för förståelser i processinriktade informationssystem i transaktionsintensiva verksamheter, 2005.
No 1190 David Dinka: Role and Identity - Experience of technology in professional settings, 2005.
Andreas Hansson: Increasing the Storage Capacity of Recursive Auto-associative Memory by Segmenting Data, 2005.
Andreas Käll: Översättningar av en managementmodell - En studie av införandet av Balanced Scorecard i ett landsting, 2005.
He Tan: Aligning and Merging Biomedical Ontologies, 2006.
No 1481 Anna Vapen: Contributions to Web Authentication for Untrusted Computers, 2011.
No 1506 David Landén: Complex Task Allocation for Delegation: From Theory to Practice, 2011.
No 1507 Kristian Ståvärk: Contributions to Parallel Simulation of Equation-Based Models on Graphics Processing Units, 2011.
No 1606 Karl Hammar: Towards an Ontology Design Pattern Quality Model, 2013.
No 1647 Dag Sonntag: A Study of Chain Graph Interpretations, 2014.
No 1683 Zlatan Dragić: Completing the Is-A Structure in Description Logics Ontologies, 2014.
FiF 118 Camilla Kirkegaard: Adding Challenge to a Teachable Agent in a Virtual Learning Environment, 2016.
No 1758 Venkatanathan Krishnamoorthy: Efficient and Scalable Content Delivery of Linear and Interactive Branched Videos, 2016.