Karlsruhe Institute of Technology

Recursive Surrogate-Modeling for Stochastic Search

Bachelor's Thesis
of

Klaus Philipp Theyssen

KIT Department of Informatics
Institute for Anthropomatics and Robotics (IAR)
Autonomous Learning Robots (ALR)

Referees: Prof. Dr. Techn. Gerhard Neumann
Prof. Dr. Ing. Tamim Asfour

Advisor: M.Sc. Maximilian Hüttenrauch

Duration: November 27th, 2020 — March 26th, 2021

Erklärung

Ich versichere hiermit, dass ich die Arbeit selbstständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht habe und die Satzung des Karlsruher Instituts für Technologie zur Sicherung guter wissenschaftlicher Praxis beachtet habe.

Karlsruhe, den 26. März 2021

Klaus Philipp Theyssen

Zusammenfassung

Roboter werden in den nächsten Jahrzehnten zunehmend unseren Alltag durchdringen. Sie werden ein breites Spektrum an Aufgaben wie Altenpflege, Such- und Rettungsaufgaben und allgemeine Assistenz im täglichen Leben übernehmen können. Derzeit sind die meisten Roboter darauf angewiesen von einem erfahrenen menschlichen Bediener programmiert zu werden. Damit Roboter Teil des täglichen Lebens werden können, müssen sie sich in verändernden Umgegebungen zurechtfinden und sich an neue Situationen anpassen. Daher werden selbstlernende Roboter zunehmend an Bedeutung gewinnen. Lernende Roboter ist beireits ein aktives Forschungfeld und wird weiter wachsen. Es besteht eine enge Beziehung zwischen Reinforcement Learning und Robotik, wobei sich beide Forschungsgebiete gegenseitig ergänzen. Eine wichtige Herausforderung ist es dateneffiziente Methoden für selbstlernende Roboter zu entwickeln, da Stichproben aus realen Interaktionen sehr kostspielig sind. Policy-Suchmethoden, ein Teilgebiet des Reinforcement Learning, können eingesetzt werden um verschiedene Aufgaben durch Versuch und Irrtum zu erlernen.

In dieser Arbeit versuchen wir, die Dateneffizienz eines Policy-Suchalgorithmus zu verbessern. Dafür benutzen wir rekursive Schätzmethoden wie den Kalman Filter, der aus einer Bayes'schen Perspektive eingeführt wird. Wir implementieren verschiedene Versionen von Filteralgorithmen und vergleichen sie mit bisherigen Methoden und testen unsere Algorithmen für Optimierungs-Testfunktionen und einfache planare Greifaufgaben.

Abstract

Robots will increasingly permeate our daily lifes over the next few decades. Fulfilling a wide range of tasks like elder care, search and rescue and general assistance in daily life. Today most robots rely on being taught and programmed by a skilled human operator. For robots to become part of daily life they need to cope with changing environments and adjust to new situations, therefore problem of autonomously learning robotics will become more important. Currently robot learning is already a very active field of research. There is a close relationship between Reinforcement learning and robotics, where both areas of research complement each other. A big challenge for robot learning is being sample efficiency, because real world samples are costly to obtain. Policy search methods, a sub-field of reinforcement learning, can be used to learn different task only through trial and error.

In this thesis we try to improve the sample efficiency of a policy search algorithm. This is done using classical recursive estimation techniques like the Kalman filter, introduced from a Bayesian perspective. We implement different versions of Filtering algorithms and compare them with previous methods and benchmark them on optimization test function and simple planar reaching tasks.

Table of Contents

Chapter 1.

Introduction

1.1. Motivation

Robots are already used extensively in industry to form production chains, where they perform the same task over and over. These robots are being programmed and fine tuned by a human engineer which requires experience and expertise.
Recently there has been a new development of robots becoming part of our daily life's, for example in the form of vacuum cleaner and lawn mowers. In the future areas like care giving and everyday assistance and household work may become successful application domains of robotics Schaal (2007). This poses a dramatic shift from the prior uses of robots in industry, where they mainly worked in isolated and predefined contexts. Especially countries like Japan, facing the problem of an aging population, put increasing effort to make robots viable in these areas.
In daily life the robots are confronted with different challenges, like adapting to different lightening conditions and objects being moved around. The robots which are currently on the market like vacuum cleaners and lawn mowers have either a "one size fits all" approach or need a special setup in software or hardware. To solve this issue machine learning and especially reinforcement learning will be key technologies to enable robots to adjust to dynamic and stochastic environments.

Robotics and reinforcement learning complement each other with robotics providing a real world testing ground, and reinforcement learning providing the framework for formulating problems and finding solution. The relationship may be similar to the one of math and physics Kober et al. (2013).
Compared to other domains in which reinforcement learning has been successfully employed

robotics poses unique challenges. Making it necessary to explore methods adjusted to the inherent challenges of Robotics, which include:

- high dimensional state and action space
- obtaining real world samples
- goal specification
- under-modeling and model uncertainty

Robots have potential to transform our society. By bringing robots from the factories into our homes there will be many possibilities for improvement, to reap the benefits of this development we will have to overcome many technical and ethical challenges alike.

1.2. Contribution

Policy search algorithms have shown promise as an alternative to value function-based reinforcement learning, especially for learning motor skills in robotics Deisenroth et al. (2013). The MORE algorithm as a policy search method based on information theoretic updates is introduced in Abdolmaleki et al. (2015). The key idea of MORE is to learn a surrogate Model of the objective function to efficiently computing the updates to the policy in closed form. One of the contributions of this thesis is to explore recursive estimation for learning the surrogate model of the objective function to increase sample efficiency. Besides that we aim to improve the runtime of the algorithm with this approach. We focus mainly on classical methods of parameter estimation and filtering like the Recursive Least Squares algorithm and the Kalman Filter, considering more advanced methods is part of future work. We benchmark our version of the MORE algorithm on optimization test functions and simple planar reaching tasks and compare them to previous results.

1.3. Structure of Thesis

The remainder of this thesis is structured as follows:

Chapter 2: In the fundamentals chapter we lay the foundation for the thesis, first introducing the basics of Reinforcement Learning and then focusing on the specifics of applying RL to robotic tasks. Then We focus on policy search as one method for solving the robot learning problem. Next we introduce the original MORE algorithm and finally we look at Bayesian filtering.

Chapter 3: Review of the related work in the field.

Chapter 4: In this chapter we motivate the idea of using recursive estimation for learning the surrogate model and then review methods used for data preprocessing. Finally we develop our approach of connecting the MORE algorithm with recursive parameter estimation.

Chapter 5: In the evaluation chapter we conduct experiments with the algorithms on several tests function and some simple reaching tasks for a planar robot. We compare our algorithm with original MORE and other stochastic search algorithms.

Chapter 6: We conclude the thesis with a summary of the achieved results and an outlook on future work.

Chapter 2.

Fundamentals

This chapter introduces basic concepts used throughout this thesis. First we discuss the basics of reinforcement learning and the problem of robot learning. Next we give an overview of policy search, a sub-field of reinforcement learning, as one method to solve the robot learning problem. The Kullback-Leibler divergence (KL) is introduced as an important information theoretic distance metric between probability distributions (Kullback and Leibler, 1951). Having discussed the underlying basics we can then introduce the MORE Algorithm. Finally we will look at Filtering from a Bayesian Estimation viewpoint and review the Kalman Filter for parameter estimation.

2.1. Reinforcement Learning

Reinforcement Learning (RL) is a subfield of Machine Learning concerned with agents learning to interact with their environment, enabling them to solve tasks. This is done through exploration and trial-and-error, trying to discover cause and effect relationship between actions. Compared to supervised learning and unsupervised learning it more closely resembles the way humans learn.
As Sutton and Barto (2018) puts it, the term reinforcement learning relates to a class of problems, solution methods and the field that studies these problems and solutions. Some famous examples include playing games like Go (Silver et al., 2016) and Atari games (Mnih et al., 2013) Generally RL is applicable to a large range of problems. Whereas in supervised learning the best action is presented to the system, the agent in a reinforcement learning setting receives a reward (or punishment). To gain information about the rewards the agent needs to explore previously unused actions. Dare to try new things or keep performing safe

well-known actions, this is the *exploration-exploitation tradeoff*. The agent should exploit actions he knows that give decent reward, but he first has to try different things to learn about these actions, and then he has to progressively focus in on them. The reward signal is typically a single scalar value, hence the amount of information the agent receives is minimal compared to supervised and unsupervised learning approaches.

The classical approach to formalizing problems in RL is through Markov Decision Processes (MDPs). MDPs are a mathematical framework for decision making in deterministic and stochastic environments. MDPs focus on only three aspects - sensation, action and goal, which are central for reinforcement learning problems. MDPs satisfy the Markov property (cite), which state that "the future is independent of the past given the present". In our case this means the next state s' and the reward only depend on the previous state s and action a Sutton et al. (1992). An MDP can be formally defined as a tuple (S, A, P, r):

- a set of states $s \in S$ that describe the environment.
- a set of actions $a \in A$ that can be performed by the agent in the environment
- a transition function $P(s_{t+1}|s_t, a_t)$ that gives the probability of a new state s_{t+1} after an action a_t has been taken in state s_t
- a reward function $r(s_t, a_t)$ that specifies the immediate reward after taking action a_t in state s_t

The Markov property can be expressed as

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1} \cdots) = P(s_{t+1}|s_t, a_t)$$

This recapitulates the notion of state - a state is a sufficient statistic for predicting the future, rendering previous observations irrelevant. In robotics we may only find some approximate notion of state.

For our setting we model the agent and its environment as a state $s \in S$. The agent may perform action $a \in A$ which can be either discrete or continuous. For every action step the agent receives a Reward R, which is a scalar value. The overarching goal is to find a mapping from states to actions, called policy π, that picks actions in a way that the reward is maximized. We can distinguish an *episodic* setting from an on-going task. In the episodic setting the task is restarted after the end of an episode and the goal is to maximize the total reward per episode. In on-going tasks the goal is to simply achieve high average reward over the whole life-time or one can use a formulation with a discounted return (weighting the future and past differently).

Generally the goal is to find an optimal policy π^*, a mapping from states to actions that maximizes the expected return J. Optimal behavior can be modeled in different ways, resulting in different definitions for expected return.

Finite-horizon model, for horizon H meaning the next H timesteps:

$$J = E\left\{\sum_{t=0}^{H} R_t\right\}$$

Discounted reward, discounting future rewards by a discount factor γ (with $0 \leq \gamma < 1$):

$$J = E\left\{\sum_{t=0}^{\infty} \gamma^t R_t\right\}$$

Reinforcement learning can also be seen as general case of optimal control as in Sutton et al. (1992), whereas optimal control assumes perfect knowledge, RL uses approximations and data-driven techniques.

2.2. Robot Learning

The sub-field where reinforcement learning and machine learning intersect with robotics is called *Robot Learning*. It aims to bridge the gap between programmed robots, with fine tuned controllers and fully autonomous robots. As proposed in Deisenroth et al. (2013), robot control can be modeled as a reinforcement learning problem.
The state space $\mathbf{x}$ in robotic tasks is high dimensional and made up of the internal state of the robot (e.g., joint position, body position, camera images) and external state (e.g. object locations, lighting). The true state is not observable and also not noise free. The Robot chooses its next motor control u according to a policy π. This policy π may be deterministic $a = \pi(s)$ or stochastic $a \sim \pi(s,a) = P(a|s)$. The motor commands u alter the state according to the probabilistic transition function $p(\mathbf{x}_{t+1}|x_t, u_t)$. This transition function is not known, in model-based policy search this function is learned from data and used to improve the policy. Collectively the states and actions of the robot form a *trajectory* $\tau = (x_0, u_0, x_1, u_1, ...)$ which is also called a *rollout* or a *path*. There has to be a numeric scoring system assessing the quality of the robots trajectory and returning a reward signal $R(\tau)$. For *episodic learning tasks* the task ends after a given number T of time steps. Then the accumulated reward $R(\tau)$ for a trajectory is given by

$$R(\tau) = r_T(x_T) + \sum_{t=0}^{T-1} r_t(x_t, u_t)$$

where r_t is an instantaneous reward function (e.g. a punishment for energy consumed) and r_T the final reward, when performing a reaching task this may take the form of a quadratic punishment term for deviation from the goal posture.

If we consider an infinite-horizon for an on-going task we get

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r(x_t, u_t)$$

where $\gamma \in [0,1)$ is a discount factor that discounts rewards further in the future.

Many tasks in robotics can thus be formulated as choosing an optimal control policy π^* that maximizes the expected accumulated reward

$$J_\pi = \mathbb{E}[R(\tau)|\pi] = \int R(\tau) p_\pi(\tau) d\tau$$

where $R(\tau)$ defines the objectives of the task, and $p_\pi(\tau)$ is the distribution over trajectories τ.

Formulating robotic tasks in this way allows us to use methods from reinforcement learning.

2.2.1. Challenges

Reinforcement learning is in general a difficult problem. One reason for this is that the reward signal may be given only occasionally and even then it may be unclear which of the agents actions were responsible for a certain reward signal.

Robotics is different compared to other fields where Reinforcement Learning is used extensively. First of all the states and actions of the robots in the real world are inherently continuous requiring us to deal with the resolution. In addition the state space can have a high dimensionality and working with real world systems on real hardware is costly and makes manual interventions necessary. Robots require algorithms to run in real-time and working with real sensors further introduces discrepancy between sensing and execution.

Most traditional methods from RL like TD-learning (Sutton and Barto, 2018) have been unfit for these particular requirements of robotic tasks. Stressing the importance of robotics as a testing ground for RL that demands new developments and innovative research. We will now discuss some problems encountered when applying RL to robotics, this treatment does only focus on a certain set of challenges and is not exhaustive.

Curse of Dimensionality

The term "Curse of Dimensionality" was coined by Bellman (1957) when he explored optimal control in higher dimensions and encountered an exponential explosion of the states and actions. For example, for the evaluation of our algorithm we run a simulation of a simple planar reaching task with a 5 link robot arm using Dynamic Movement Primitives (Ijspeert et al., 2002) for policy representation and get a 25 state dimension. Especially modern anthropomorphic robots tend to have many degrees of freedom.

The agent in RL generally needs to collect data throughout the entire state-space to ensure global optimization, for robotics dealing with three dimensional space makes this very difficult. To alleviate this problem a widespread idea is to use expert demonstration to get a good initialization for the agents policy, this eliminates the need to explore the entire search space, instead the agent can focus on locally optimizing the initial policy. This concept of imitation learning focuses on the problem of "learning from demonstrations" which plays an important role for robotics (Osa et al., 2018). Imitation learning will enable domain experts to teach motions and skills without special knowledge about robotics, which will be crucial when robots start making their way from factories into everyday life.

Curse of Real-world Samples

Robot hardware is expensive and suffers from wear and tear, making costly maintenance necessary. Hence, safe methods for real robots should avoid big jumps in policy updates, as such sudden changes may result in unpredictable movements and consequently damage the robot. This problem is commonly referred to as *safe exploration*. Whereas in traditional reinforcement learning safe exploration does not receive much attention, it has become a key issue for robot learning (Schneider, 1997). Different external factors like temperature may change the robot dynamics and additional uncertainty from the sensors makes it difficult to reproduce and compare results.

Most tasks also require a "human in the loop" who either supervises the robot or resets the setup after one episode. Even if this can be avoided the learning speed simply cannot be sped up. For a single robot the training time has a natural limit and in total only relatively few executions can be completed. One method for gathering more data is "collective robot learning" described in Kehoe et al. (2015). The idea is for multiple robots to share their data on trajectories, policies and outcomes. Currently this seems only viable for large corporations with significant capital. Thus, in robot learning the constraint of using only a small number of trials is given more weight than limiting memory consumption or computational complexity. Stressing the importance of developing sample efficient algorithms. Generally the state represented by sensors slightly lags behind the real state due to processing and communication delays. This is in stark contrast to most other reinforcement learning algorithms, which assume actions to take effect instantaneously.

Curse of Goal Specification

Defining a good reward function in robot reinforcement learning is difficult and often needs a lot of domain knowledge and expertise. There are certain trade-offs to keep in mind, for example performing a powerful swing for a hitting task may yield high reward but may damage or shorten the life-time the robot. Reinforcement learning algorithms may solve tasks in unintended ways exploiting the reward function in an unforeseen style (Ng et al., 1999).

Inverse reinforcement learning (Russell, 1998) provides an alternative to specifying the reward function manually. The goal of inverse reinforcement learning is to recover the unknown reward function from the expert's trajectories.

Still, learning algorithms are rarely employed on robots for real daily usage. Also most algorithms are over fitted to a particular robot architecture and do not generalize to other robots easily. Even minor flaws or errors in the employed method can completely prevent success of learning. Nonetheless appropriately chosen algorithms and rewards functions can already achieve promising results, but as a large number of different methods exist there is no clear recipe for robot learning. Stressing that robot learning still has a lot open problems, and will continue to grow as a research field.

2.2.2. Policy Search

Many traditional methods in RL try to estimate the expected long-term reward of a policy for each state $\mathbf{x}$ and time step t, which leads to formulation of the value function $V_t^\pi(\mathbf{x})$. With the value function we can asses the quality of executing action $\mathbf{u}$ in state $\mathbf{x}$. This assessment is used to directly compute the policy by action selection or to update the policy π. As value function methods struggle with the challenges in reinforcement learning, the main approach for robotics has become policy search. Policy search methods opposed to value-based methods use parameterized policies π_θ and search directly in the parameter space Θ. This allows using RL with high-dimensional continuous action spaces encountered in robotics by reducing the search space of possible policies. Policy search further allows the usage of predefined task-appropriate policy representations like Dynamic Movement Primitives (Schaal et al., 2005), as well as easily integrating imitation learning for policy initialization.

Generally we can divide policy search into model-free and model-based and differentiate whether stochastic or deterministic trajectories are used. Model-free policy search uses trajectories from the robot directly for updating the policy. Model-based methods use the data from the robot to learn a model of the robot. This model is then used to generate trajectories that are used for policy updates. Due to their simplicity and by avoiding the need of learning a model, which further introduces the problem of under-modeling, model-free methods have been widely employed.

The most important concept is computing the policy updates. Both model-free and model-based policy search use policy gradients (PG), expectation-maximization (EM)-based updates, or information-theoretic insights (Inf.Th.).

In this thesis we will focus on model free policy search methods where the trajectories are generated by "sampling" from the robot. More specifically we will focus on stochastic search algorithms, which are general black-box optimizers. They are used in a wide range of fields like operations research, machine learning and also policy search. Since these algorithms do not use any knowledge about the objective function it is straightforward to apply them to policy search in the episode-based formulation.

Using stochastic search algorithms we keep an upper-level policy $\pi_\omega(\theta)$ which selects the parameters of the actual control policy $\pi_\theta(\mathbf{u}|\mathbf{x})$ of the robot. Instead of directly finding the parameters θ of the lower-level policy we want to find the parameter vector ω which defines a search distribution over θ. We can then use this search distribution to directly explore the parameter space.

2.3. Kullback-Leibler (KL) Divergence

Several Algorithms used for policy search like NES (Wierstra et al., 2014) and REPS (Peters et al., 2010) rely on the Kullback-Leibler Divergence, also known as the relative entropy, for controlling the difference between the old and updated policy. Working with real robots additionally requires to perform safe exploration. Big exploration steps may result in damaging the hardware. Specifically, it measures the Shannon entropy of one distribution relative to the other. The KL divergence from q to p is defined as

$$KL(p||q) = \int p(\theta) \log \left(\frac{p(\theta)}{q(\theta)} \right) d\theta$$

where p and q are continuous probability distributions. Note, that in general the relative entropy is not symmetric under interchange of the distributions p and q. In general $KL(p||q) \neq KL(q||p)$, therefore in a mathematically sense it is not strictly a distance.

2.4. MORE Algorithm

Model-Based Relative Entropy Stochastic Search (MORE) (Abdolmaleki et al., 2015) is a stochastic search algorithm, that can be used as a policy search method for episodic reinforcement learning tasks. The key idea is using information-theoretic policy updates by bounding the relative entropy (Kullback Leibler divergence) between two subsequent policies. As MORE uses no gradient information and requires only function evaluations of the objective function for the optimization. The essential difference of MORE compared to previous algorithms using the KL-bound like REPS (Peters et al., 2010) lies in utilizing a quadratic surrogate model of the objective function to satisfy the KL-bound in closed form without approximations. On top of that it introduces a lower bound constraint on the entropy of the new distribution to avoid premature convergence.

When using the MORE algorithm we want to maximize an objective function $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$. The goal is to find a parameter vector $\mathbf{x} \in \mathbb{R}^n$ with the highest possible objective value. For this the MORE algorithm iteratively samples from a search distribution, implemented as a multivariate Gaussian distribution, i.e. $\pi(\mathbf{x}) = \mathbf{N}(\mathbf{x}|\mu,\Sigma)$. This distribution corresponds to an upper-level policy parameterized by the mean and covariance matrix. In each iteration N samples are drawn from the search distribution. Each sample $\mathbf{x}$ is then evaluated on the objective function yielding the corresponding objective value r, this constitutes the data $\{\mathbf{x}^k, r^k\}_{k=1...N}$ which is used to compute a new search distribution. This iterative process is run until the algorithm converges.

2.4.1. MORE Framework

We can use constraint optimization to formulate an optimization problem Equation (2.1) for obtaining a new search distribution that maximizes the expected objective value while upper-bounding the KL-divergence and lower-bounding the entropy of the distribution.

$$\begin{aligned} \max_{\pi} \int \pi(\theta)\mathcal{R}_\theta d\theta, \\ \text{s.t. } \mathrm{KL}(\pi(\theta)||q(\theta)) \leq \varepsilon, \\ H(\pi) \geq \beta, \\ 1 = \int \pi(\theta) d\theta \end{aligned} \tag{2.1}$$

where $H(\pi) = -\int \pi(\theta)\log\pi(\theta)d\theta$ denotes the entropy of the search distribution. The parameters ε and β can be changed to control the exploration-exploitation trade-off of the algorithm. We will now proceed to derive the solution to the constraint optimization problem using the theory of Lagrange multipliers and duality.

Forming the Lagrangian for our optimization problem we get

$$\mathfrak{L}(\pi,\eta,\omega) = \int \pi(\theta)\mathcal{R}_\theta d\theta \,+\, \eta\left(\varepsilon - \int \pi(\theta)\,\log\frac{\pi(\theta)}{q(\theta)}d\theta\right) - \,\omega\left(\beta + \int \pi(\theta)\,\log(\pi(\theta))d\theta\right)$$

Optimizing the Lagrangian by computing the derivative with respect to the mean and covariance matrix yields

$$\pi(\theta) \propto q(\theta)^{\eta/(\eta+\omega)}\exp\left(\frac{\mathcal{R}_\theta}{\eta+\omega}\right)$$

the solution depends on the quadratic and linear term of the surrogate model and the Lagrangian multipliers η and ω. These in turn can be obtained by minimizing the dual function:

$$g(\eta,\omega) = \eta\varepsilon - \omega\beta + (\eta-\omega)\log\left(\int q(\theta)^{\frac{\eta}{\eta+\omega}}\exp\left(\frac{\mathcal{R}_\theta}{\eta+\omega}\right)d\theta\right)$$

For full derivation of the equations see Appendix A.1.

Entropy constraint

The bound β is defined such that relative difference between the entropy of the policy $H(\pi)$ and a minimum exploration policy $H(\pi_0)$ is decreased for a certain percentage:

$$H(\pi) - H(\pi_0) \geq \gamma(H(q) - H(\pi_0)) \rightarrow \beta = \gamma(H(q) - H(\pi_0) + H(\pi_0)$$

2.4.2. Surrogate Model

The key idea of the MORE algorithm is using a surrogate model of the objective function for satisfying the bound on the KL-divergence. It has proven to be sufficient to use a quadratic model, since the exponent of Gaussian distribution is also quadratic, it is not possible to exploit the information of a more complex surrogate model. The surrogate model has an quadratic amount of parameters, estimating these parameters poses a bottle-neck in terms of sample efficiency for the algorithm. The original approach of MORE employs a Bayesian dimensionality reduction method and linear regression.

2.4.3. Analytic Solution for Dual-Function and Policy

Assuming we are given a quadratic surrogate model

$$\mathcal{R}_\theta \approx \theta^T \mathbf{R} \theta + \theta^T \mathbf{r} + r_0$$

we can solve the dual function in closed form.

$$g(\eta, \omega) = \eta\varepsilon - \beta\omega + \frac{1}{2}\left(\mathbf{f}^T \mathbf{F} \mathbf{f} - \eta \mathbf{b}^T \mathbf{Q}^{-1} \mathbf{b} - \eta \log|2\pi\mathbf{Q}|p + (\eta + \omega)\log|2\pi(\eta + \omega)\mathbf{F}|\right)$$

with $\mathbf{F} = (\eta\Sigma^{-1} - 2\mathbf{R})^{-1}$ and $\mathbf{f} = \eta\Sigma^{-1}\mu + \mathbf{r}$

With the assumption of π being Gaussian we get the solution Equation (2.2) for updating the mean and covariance matrix of the search distribution.

$$\begin{aligned} \pi_{t+1} &= \mathrm{N}(\mu_{t+1}, \Sigma_{t+1}) \\ \mu_{t+1} &= (\eta\Sigma_t^{-1}\mu_t + \mathbf{r})/(\eta + \omega) \\ \Sigma_{t+1} &= (\eta\Sigma_t^{-1} + \mathbf{R})/(\eta + \omega) \end{aligned} \tag{2.2}$$

With these equations we can iteratively update the search distribution.

2.5. Bayesian Filtering

Optimal filtering is concerned with estimating the state of a time-varying system which is indirectly observed through noisy measurements. This section will focus on optimal filtering from a Bayesian perspective and is largely based on Särkkä (2013).

The term "Bayesian" refers to inference methods that represent "degrees of certainty" using probability theory, based on applying Bayes' rule to update the degree of certainty given data. More generally as Gelman et al. (2013) puts its, Bayesian inference is the process of fitting a probability model to a set of data and summarizing the result by a probability distribution on the parameters of the model and on unobserved quantities such as predictions for new observations.

Filtering methods are widely used in robotics to deal with noisy sensor measurements. This includes tasks like object tracking, robot control and robot localization (Chen, 2011). Since robots need to make decisions based on relatively small amounts of data, it is common to adopt an Bayesian perspective when using filtering methods and for reasoning about the environment in general (Thrun, 2002).

2.5.1. Bayesian Parameter Estimation

In general when using Bayesian models for estimating *unknown parameters* θ the following probability distributions are used.

- **Prior Distribution:** Encodes the information on parameter θ before seeing any observations. When we are uncertain about our prior information we can choose a high variance of the distribution or use a non-informative prior (which imposes the minimal amount of structure on the data).

$$p(\theta) = \text{information on parameter } \theta \text{before seeing any observations}$$

- **Measurement Model:** Models the relationship between true parameters and the measurements.

$$p(y|\theta) = \text{distribution of observation } y \text{given the parameters} \theta$$

- **Posterior Distribution** The conditional distribution of the parameters given the observations. It represents the updated belief about the parameters after obtaining the measurements. It can be computed by using Bayes' rule.

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} \propto p(y|\theta)p(\theta)$$

2.5.2. Optimal filtering as Bayesian inference

The goal of optimal filtering can be seen as solving a statistical inversion problem where the unknown quantity is a potentially vector valued time series $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, ...\}$ which is observed through a set of noisy measurements $\{\mathbf{y}_1, \mathbf{y}_2, ...\}$, as depicted in figure 2.1.

$$\begin{tikzcd} \text{hidden:} \arrow[r, dashed] & \mathbf{x}_{k-1} \arrow[r] \arrow[d] & \mathbf{x}_k \arrow[r] \arrow[d] & \mathbf{x}_{k+1} \arrow[r, dashed] \arrow[d] & {} \\ \text{observed:} & \mathbf{y}_{k-1} & \mathbf{y}_k & \mathbf{y}_{k+1} & \end{tikzcd}$$

Figure 2.1.: A sequence of hidden states $\mathbf{x}_k$ is indirectly observed through noisy measurements. Drawing recreated from Särkkä (2013)

We want to estimate the hidden states from the observed measurements. Solving this problem in a Bayesian sense means we need to compute the joint posterior distribution of all states given all the measurements, for this we can simply use Bayes' rule. This results in batch solution to the statistical estimation problem, with the following steps.

$$p(\mathbf{x}_{0:T}|y_{1:T}) = \frac{p(y_{1:T}|\mathbf{x}_{0:T})p(\mathbf{x}_{0:T})}{p(y_{1:T})}$$

where $p(\mathbf{x}_{0:T})$ is the prior distribution defined by the dynamic model, $p(y_{1:T}|\mathbf{x}_{0:T})$ is the likelihood model for the measurements and $p(y_{1:T})$ is the normalization constant defines as

$$p(y_{1:T}) = \int p(y_{1:T}|\mathbf{x}_{0:T})p(\mathbf{x}_{0:T})d\mathbf{x}_{0:T}.$$

This formulation is unfit for dynamic estimation task where we receive measurements one at a time, since at each time step we would have to recompute the full posterior distribution. As the number of time steps increases, also the dimensionality of the full posterior would increase making computations intractable. If we instead only compute selected marginal distributions we can make computation feasible again. For this we need to restrict our dynamic models to probabilistic Markov sequences, with a transition probability distribution $p(\mathbf{x}_k|\mathbf{x}_{k-1})$.

In *Bayesian filtering and smoothing* the following marginal distributions are considered

- *Filtering distributions* computed by the Bayesian filter are the marginal distributions of the current state $\mathbf{x}_k$ given the current and previous measurements $y_{1:k} = \{y_1, ..., y_k\}$

$$p(\mathbf{x}_k|y_{1:k}), \quad k = 1, ..., T.$$

- *Prediction distributions* which can be computed with the prediction step of the Bayesian filter are the marginal distributions of the future state $\mathbf{x}_{k+n}$, with n steps after the current time step

$$p(\mathbf{x}_{k+n}|y_{1:k}), \quad k = 1,...,T \quad n = 1,2,...$$

- *Smoothing distributions* computed by the Bayesian smoother are the marginal distributions of the state $\mathbf{x}_k$ given a certain interval $y_{1:T} = \{y_1,...,y_T\}$ of measurements with $T > k$

$$p(\mathbf{x}_k|y_{1:T}), \quad k = 1,...,T$$

For our surrogate model estimation problem in the iterative MORE algorithm we use the filtering distribution. By limiting us to the filtering distribution we can now formulate the recursive Bayesian solution to the statistical inversion problem as follow:

1. The distribution of measurements is modeled by the likelihood function $p(y_k|\mathbf{x}_k)$ and the measurements are assumed to be conditionally independent

2. In the beginning at time step 0 all information about $\mathbf{x}$ is contained in the prior distribution $p(\mathbf{x}_0)$

3. The measurements are assumed to be obtained one at a time, first y_1 then y_2 and so on. The key idea is to use the posterior distribution from the previous time step as the current prior distribution

$$\begin{aligned}
p(\mathbf{x}_1|y_1) &= \frac{1}{Z_1} p(y_1|\mathbf{x}_1)p(\mathbf{x}_0) \\
p(\mathbf{x}_2|y_{1:2}) &= \frac{1}{Z_2} p(y_2|\mathbf{x}_2)p(\mathbf{x}_1|y_1) \\
&\vdots \\
p(\mathbf{x}_k|y_{1:k}) &= \frac{1}{Z_k} p(y_k|\mathbf{x}_k)p(\mathbf{x}_k|y_{1:k-1}) \\
&\vdots \\
p(\mathbf{x}_T|y_{1:T}) &= \frac{1}{Z_T} p(y_T|\mathbf{x}_T)p(\mathbf{x}_{T-1}|y_{1:T-1})
\end{aligned}$$

   With the normalization term $Z_k = \int p(y_k|\mathbf{x}_k)p(\mathbf{x}_k|y_{1:k-1})d\mathbf{x}_k$.

This recursive formulation of Bayesian estimation has several useful properties. First of all this can be considered as the *online learning* solution to the Bayesian learning problem. As each step in the recursive estimation is a full Bayesian update step, *batch* Bayesian inference is a special case of recursive Bayesian inference. Due to the sequential nature of the estimation we can model what happens to the unknown quantity $\mathbf{x}$. This turns out to be the basis of filtering theory, where time behavior is modeled by assuming the quantity to be a time-dependent stochastic process $\mathbf{x}(t)$.

2.5.3. Least squares

To ease the application of Bayesian Filtering to our specific problem at a later stage we will now discuss a simple regression problem. First we are going to derive the least squares solution, as we use an least squares approach for comparison in our experiments.

We consider a simple linear regression problem,

$$y_k = x_1 + x_2 r_k + \varepsilon_k \tag{2.3}$$

where we assume that the measurement noise is zero mean Gaussian with a given variance $\varepsilon_k \sim \mathrm{N}(0, \sigma^2)$ and that the prior distribution of the parameters $\mathbf{x} = (x_1\ x_2)^T$ is Gaussian with known mean and covariance $\mathbf{x} \sim \mathrm{N}(\mathbf{m}_0, \mathbf{P}_0)$. Note that in this sections the parameters $\mathbf{x}$ are assumed to stay constant. We now want to estimate the parameters $\mathbf{x}$ from a set of measurement data $\mathcal{D} = \{(t_1, y_1), ..., (t_T, y_T)\}$.

In compact in probabilistic terms we can write this as

$$\begin{aligned} p(y_k|\mathbf{x}) &= \mathrm{N}(y_k|\mathbf{H}_k\mathbf{x}, \sigma^2) \\ p(\mathbf{x}) &= \mathrm{N}(\mathbf{x}|\mathrm{m}_0, \mathrm{P}_0) \end{aligned} \tag{2.4}$$

Here $H_k = (1\ t_k)$ is the design matrix and contains the regressors and $\mathrm{N}(\cdot)$ denotes the Gaussian probability density function (see Appendix A.2). The row vector $\mathbf{H}_k$ is denoted in matrix notation, to avoid using different notation for scalar and vector measurements. The batch solution can then be easily obtained by application of Bayes'rule

$$\begin{aligned} p(\mathbf{x}|y_{1:T}) &\propto p(\mathbf{x}) \prod_{k=1}^{T} p(y_k|\mathbf{x}) \\ &= N(\mathbf{x}|m_0, P_0) \prod_{k=1}^{T} N(y_k|\mathbf{H}_k\mathbf{x}, \sigma^2) \end{aligned}$$

Because the prior and likelihood are Gaussian, the *posterior distribution* in turn is also Gaussian

$$p(\mathbf{x}|y_{1:T}) = \mathrm{N}(\mathbf{x}|m_t, P_t)$$

By completing the quadratic form in the exponent we get the equations (2.5) for the mean and covariance of the posterior distribution.

$$\begin{aligned} \mathbf{m}_T &= \left[\mathbf{P}_0^{-1} + \frac{1}{\sigma^2}\mathbf{H}^T\mathbf{H}\right]^{-1} \left[\frac{1}{\sigma^2}\mathbf{H}^T\mathbf{y} + \mathbf{P}_0^{-1}\mathbf{m}_0\right] \\ \mathbf{P}_T &= \left[\mathbf{P}_0^{-1} + \frac{1}{\sigma^2}\mathbf{H}^T\mathbf{H}\right]^{-1} \end{aligned} \tag{2.5}$$

where $\mathbf{H}_k = (1\ t_k)$ and

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_1 \\ \vdots \\ \mathbf{H}_T \end{pmatrix} = \begin{pmatrix} 1 & t_1 \\ \vdots & \vdots \\ 1 & t_t \end{pmatrix}, \qquad y = \begin{pmatrix} y_1 \\ \vdots \\ y_T \end{pmatrix}$$

2.5.4. Kalman Filter

For the least squares solution the parameters $\mathbf{x} = (x_1\ x_2)$ of the regression model (2.3) are assumed to stay constant. Now we assume the parameters are allowed to perform a Gaussian random walk between measurements.

$$\begin{aligned} p(y_k|\mathbf{x}_k) &= \mathrm{N}(y_k|\mathbf{H}_k\mathbf{x}_k, \sigma^2) \\ p(\mathbf{x}_k|\mathbf{x}_{k-1}) &= \mathrm{N}(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{Q}) \\ p(\mathbf{x}_0) &= \mathrm{N}(\mathbf{x}_0|\mathbf{m}_0, \mathbf{P}_0) \end{aligned}$$

where $\mathbf{Q}$ is the covariance of the random walk.
We will now formulate the linear regression model as a time-invariant model by avoiding explicit covariates t_k. Which has the advantage that the model is not dependent on the absolute time, but only on the relative positions of states and measurements in time. We denote the time difference between consecutive times as $\Delta t_{k-1} = t_k - t_{k-1}$. The idea is that if the underlying phenomenon (signal, state, parameter) x_k was exactly linear, the difference between adjacent time points could be written exactly as

$$x_k - x_{k-1} = \dot{x}\,\Delta t_{k-1}$$

where $\dot{x}$ is the derivative, which is constant in the linear case.

$$\begin{aligned} x_{1,k} &= x_{1,k-1} + \Delta t_{k-1} x_{2,k-1} + q_{1,k-1} \\ x_{2,k} &= x_{2,k-1} + q_{2,k-1} \\ y_k &= x_{1,k} + s_k \end{aligned}$$

where the signal is the first component of the state ($x_{1,k} = x_k$) and the derivative is the second ($x_{2,k} = \dot{x_k}$). The noises are $s_k \sim \mathrm{N}(0, \sigma^2)$ and $(q_{2,k-1}, q_{2,k-1}) \sim \mathrm{N}(0, \mathbf{Q})$. Now the linear regression model (2.4) can be written in the form

$$\begin{aligned} p(y_k|\mathbf{x}_k) &= \mathrm{N}(y_k|\mathbf{H}\mathbf{x}_k, \sigma^2) \\ p(\mathbf{x}_k|\mathbf{x}_{k-1}) &= \mathrm{N}(\mathbf{x}_k|\mathbf{A}_{k-1}\mathbf{x}_{k-1}, \mathbf{Q}) \end{aligned}$$

where

$$\mathbf{A}_{k-1} = \begin{pmatrix} 1 & \Delta t_{k-1} \\ 0 & 1 \end{pmatrix}, \qquad \mathbf{H} = (1 \quad 0)$$

Which makes it a special case of generic linear Gaussian models of the form,

$$p(y_k|\mathbf{x}_k) = \mathrm{N}(y_k|\mathbf{H}_k\mathbf{x}_k, \mathbf{R}_k)$$
$$p(\mathbf{x}_k|\mathbf{x}_{k-1}) = \mathrm{N}(\mathbf{x}_k|\mathbf{A}_{k-1}\mathbf{x}_{k-1}, \mathbf{Q}_{k-1})$$

for which the Kalman Filter (Kalman, 1960) is the optimal recursive solution (for the full derivation see appendix A.3).

The Kalman filter equations can be expressed as prediction and update steps as follows.

- The prediction step

$$\begin{aligned} \mathbf{m}_k^- &= \mathbf{A}_{k-1}\mathbf{m}_{k-1} \\ \mathbf{P}_k^- &= \mathbf{A}_{k-1}\mathbf{P}_{k-1}\mathbf{A}_{k-1}^T + \mathbf{Q}_{k-1} \end{aligned} \tag{2.6}$$

- The update step

$$\begin{aligned} v_k &= y_k - \mathbf{H}_k\mathbf{m}_k^- \\ \mathbf{S}_k &= \mathbf{H}_k\mathbf{P}_k^T\mathbf{H}_k^T + \mathbf{R}_k \\ \mathbf{K}_k &= \mathbf{P}_k^-\mathbf{H}_k^T\mathbf{S}_k^{-1} \\ \mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k v_k \\ \mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T \end{aligned} \tag{2.7}$$

We did not find a good state transition model for our problem of estimating the parameters of the quadratic model of the objective function. We tried some momentum based approaches during our evaluation but did not receive any promising results. Therefore we limit our approach to adding model noise in the prediction step and using an Identity matrix as the state transition matrix **A**. Hence, our approach can be more fittingly described as a recursive least squares algorithm with a drift model instead of a Kalman filter.

Chapter 3.

Related Work

The MORE algorithm can be used as a model-free policy search algorithm with an information theoretic approach for updating the policy. Information-theoretic policy algorithms use the Kullback-Leibler Divergence in a constrained optimization problem to bound the distance of the old policy to the new one. Whereas MORE uses a surrogate model to compute the new search distribution in closed form other methods like the relative entropy policy search algorithm (REPS) proposed in Peters et al. (2010) uses a sample-based approximation for the KL-divergence. While in Peters et al. (2010) a step-based policy search algorithm is proposed the episode-based version of REPS is presented in Kupcsik et al. (2013) which is equivalent to stochastic search. Using Taylor approximations of the KL-divergence leads to the natural evolutionary strategies (NES) presented in Wierstra et al. (2014). NES uses the concept of the natural gradient, where the update is performed according to the standard gradient while simultaneously satisfying the KL-bound between subsequent search distributions.

The Covariance Matrix Adaptation-Evolutionary Strategy (CMA-ES) (Hansen, 2006) is a widely used stochastic search algorithm. That is based on heuristics to update the search distribution. In our experiments we try to beat benchmark for sample efficiency set by this algorithm.

A contextual version of MORE has been explored in Tangkaratt et al. (2017), where variables that do not change for a given task but vary from one task to another give the notion of context. Enabling learning from high-dimensional context variables like camera images.

Model-based methods for approximating the objective function with a local surrogate have been used in derivative free optimization Nocedal and Wright (2006). Also, local surrogate models have been used in trust region methods like Powell (2009). These methods maintain a point estimate and a trust region around this point instead of a search distribution. The point estimate is updated by optimizing the surrogate and staying in the trust region.

Recursive estimation with methods like the Kalman Filter is extensively used in robotics (Chen, 2011), mainly to estimate the state and environment of the robot from noisy sensor measurements.

Chapter 4.

Recursive Surrogate-Modeling for MORE

In this chapter we will first motivate the idea of using recursive filtering for estimating the surrogate model. Then we formulate the surrogate estimation as a regression problem and derive the recursive least squares approach for the drift model. Then we formulate our version of the MORE algorithm with recursive estimation of the surrogate model. Next we discuss various data preprocessing techniques we use or explored.

4.1. Motivation

So far, the surrogate model has been estimated from scratch in each iteration using the samples and objective values. Real world samples are very costly we want to be sample efficient. Further subsequent models are correlated (TODO: add Figure of surrogate model) due to the locality of the data, because the KL-divergence is used to bound the distance between subsequent search distributions. Therefore recursive estimation techniques may be used to exploit the information of previous surrogate models and thus reduce the total amount of samples needed.

Previously estimating the surrogate model for the MORE algorithm has been solely based on samples and corresponding rewards. There was a sample pool used to be more sample efficient. By using an recursive filtering approach we want to utilize the information of previous surrogate models instead of using only samples and rewards for estimation.

Since the MORE algorithm uses the KL-bound to limit the distance from the previous search distribution the subsequent surrogate models are also locally correlated.

The surrogate model has the form

$$\mathbf{M}(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^T A\mathbf{x} + \mathbf{x}^T b + c \tag{4.1}$$

With a quadratic term A, a spd(semi positive definite) matrix, a linear term b and a scalar c.

The learned quadratic models are locally correlated and thus we can use recursive filtering algorithms to exploit the information of previous models at the current time step, this has the potential to result in using less samples and also reducing the overall runtime of the algorithm.

4.2. Surrogate Model Estimation

4.2.1. Regression problem

If we formulate this as a regression problem we get:

$$y_k = \mathbf{x}^T \mathbf{A}\mathbf{x} + \mathbf{x}^T \mathbf{b} + c + \varepsilon_k$$

So we want to estimate the surrogate model parameters $\theta = (\mathbf{A}, \mathbf{b}, c)$ from the samples $\mathbf{x}$ and the rewards y. We also assume that the measurement noise ε_k is zero mean Gaussian $\varepsilon_k \sim N(0, \sigma^2)$ distributed.

In our algorithm we solve this problem by using the Recursive Least squares equations **??**.

4.3. Recursive Least Squares

Recursive least squares with drift model for estimated parameters as special case of Kalman Filter

$$p(\theta_k | y_{1:k-1}) = \mathrm{N}(\theta_k | m_k^-, P_k^-)$$

with

$$m_k^- = m_{k-1} \tag{4.2}$$

$$P_k^- = P_{k-1} + Q \tag{4.3}$$

Now we simply replace P_{k-1} with P_k^- in our update equations:

$$
\begin{aligned}
S_k &= \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \sigma^2 \\
\mathbf{K}_k &= \mathbf{P}_k^- \mathbf{H}_k^T S_k^{-1} \\
\mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k [y_k - \mathbf{H}_k \mathbf{m}_k^-] \\
\mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k S_k \mathbf{K}_k^T
\end{aligned}
$$

4.4. Data Preprocessing Techniques

TODO: move this whole section into evaluation/experiments?

To improve the performance of the algorithm we examined several data processing techniques. Here talk theoretically about the methods and in evaluation part show experiments about different techniques (for example show whitened model parameters)

4.4.1. Whitening

Whitening is common data preprocessing method in statistical analysis to transform a correlated random vector into an uncorrelated one Kessy et al. (2018).

We employ whitening for our algorithm, because uncorrelated random variables often greatly simplify multivariate data analysis (cite).

Whitening is a linear transformation that converts a d-dimensional random vector $\mathbf{x} = (x_1, ..., x_d)^T$ with mean $\mathrm{E}(\mathbf{x}) = \mu = (\mu_1, ..., \mu_d)^T$ and positive definite $d \times d$ covariance matrix $\mathrm{var}(\mathbf{x}) = \Sigma$ into a new random vector

$$\mathbf{z} = (z_1, ..., z_d)^T = \mathbf{W}\mathbf{x} \tag{4.4}$$

of the same dimension d and with unit diagonal "white" covariance $\mathrm{var}(\mathbf{z}) = \mathbf{I}$. The square $d \times d$ matrix $\mathbf{W}$ is called the whitening matrix.

The whitening transformation defined in Equation Equation (4.4) requires the choice of a suitable whitening matrix W. Since $\mathrm{var}(z) = \mathbf{I}$ it follows that $\mathbf{W}\Sigma\mathbf{W}^T = \mathbf{I}$ and thus
$mathbfW(\Sigma\mathbf{W}^T$
$mathbfW) = \mathbf{W}$, which is fulfilled if $\mathbf{W}$ satisfies the condition

$$\mathbf{W}^T\mathbf{W} = \Sigma^{-1}$$

This constrain does not uniquely determine the whitening matrix $\mathbf{W}$, instead given Σ there are infinitely many possible matrices $\mathbf{W}$, because it allows for rotational freedom.

We used *Cholesky whitening* which is based on Cholesky factorization of the precision matrix $\mathbf{L}\mathbf{L}^T = \Sigma^{-1}$, which leads to the whitening matrix:

$$\mathbf{W}^{\text{Chol}} = \mathbf{L}^T$$

.

If the cholesky whitening is unsuccessful we fall back to standardizing the random vector meaning $\text{var}(\mathbf{z}) = 1$ but the correlations are not removed.

- TODO: include example samples before and after whitening

4.4.2. Sample pool

The original MORE algorithm and other policy search algorithms (REPS) we use a sample pool.

Theoretically we would want to avoid using a sample pool and instead use the information form past samples in the form of the previously predicted model parameters. But with our algorithm using no sample pool led to unsatisfactory estimation results.

We increased the model noise for older samples, encoding the fact that older samples should have a higher covariance, meaning we are less certain about them. We implemented this by simple adding a constant amount to the covariance of the RLS equations for each time a sample is used. For higher dimensional task this is around 5-10 times. Still the use of a sample pool is theoretically unsound and problematic, instead a accurate state transition model could be introduced.

4.4.3. Higher model noise for older samples

Using the recursive least squares algorithm we had to maintain a sample pool, because otherwise the resulting estimation was not satisfactory. From a theoretic perspective this sloppy and ideally we would refrain from using the sample pool.

We introduced a simple counter for each sample, indicating how many times the sample has been used. Then for older samples (higher counter) we increased the model noise proportionally. We tried a linear relationship between the counter and the increased covariance for the model parameters.

$$new\_cov = constant\_cov\_matrix + w * \text{counter} * \mathbf{I}$$

- TODO: try different weightings: exponential

This improved our results on the rosenbrock test function compared to the RLS version with constant model noise for all samples. In each MORE iteration first the oldest samples from the sample pool are used, meaning the recent samples with lowest model noise are processed last by RLS.

4.4.4. Normalization

Original MORE approach uses standard score normalization, doing it in a batch way at each iteration for all samples in the sample pool.

This is not fit for recursive estimation. - different types of reward functions difficulties: - rosenbrock (high range of values) - sharp spikes, in rewards (punishing term for reaching task)

- simply online calculation of mean and var for normalization is not a good fit (include plots and results of investigation)

- instead use moving average to calculate mean and var only of window (discard old data)

- describe moving average computation of normalization

4.5. MORE with Recursive Surrogate-Modeling

Algorithmus 1 : MORE Algorithm with Recursive Surrogate-Modeling

Input : Parameters ε and β, initial distribution,
K number of iterations, N samples per iteration

```
for k = 1,...,K do
    for n = 1,...,N do
        Draw parameters θ_n ~ π
        Execute task with θ_n and receive R(θ_n)
    end
    begin Learn quadratic model with Recursive Least Squares
        for n = 1,...,N do
            Whitening: Wθ_n
            Increase model noise for older samples
            Compute surrogate model parameters with RLS
        end
    end
    Minimize dual function g(η, ω) using Eq.
    Update search distribution π using Equation (2.2)
end
```

Chapter 5.

Evaluation

This chapter will introduce the setup for the experiments and the tasks.

5.1. Setup

All algorithms were implemented in python, the optimization algorithm for MORE is Low-storage BFGS from nlopt[1]. The clusterwork2 (cw2) framework, developed at the ALR group, was used for running experiments. The experiments were conducted on a machine with two 8-core AMD Ryzen 2700X processores clocked at 2.6 GHz and 31GB of RAM.

- code is uploaded to gitlab/github (archive repo)

5.1.1. Hyperparameter search

We did a mixture of manual hyperparameter tuning and grid search with the cw2 framework. (name some number of different parameter sets tested)

[1] https://nlopt.readthedocs.io/en/latest/

5.2. Experiments

We now evaluate the performance of our algorithms on various problems.

5.2.1. Test Functions for Optimization

Based on Molga and Smutnicki (2005).

Figure 5.1 shows the rosenbrock function, a uni-model optimization function. The global minimum $f(\mathbf{x}) = 0$. In the experiments the mean of the initial distribution has been chosen randomly.

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

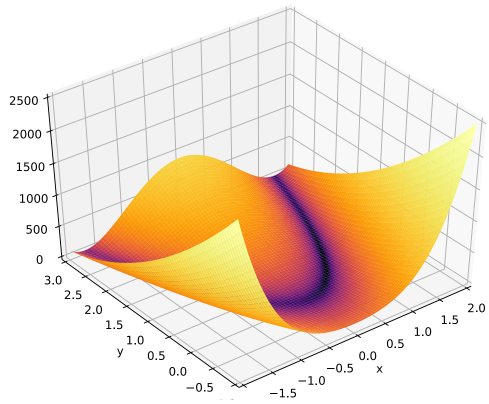

Figure 5.1.: rosenbrock function

In figures Figure 5.2 and Figure 5.3 we show the best results achieved on 5 dimensional and 15 dimensional rosenbrock. We observe that the our RLS approach uses the least amount of samples. For 5 dimensional rosenbrock the BLR approach is ommitted due to considerably worse performance. One observation to make is that RLS converges only to around 1e-9, while the LS approach manages to

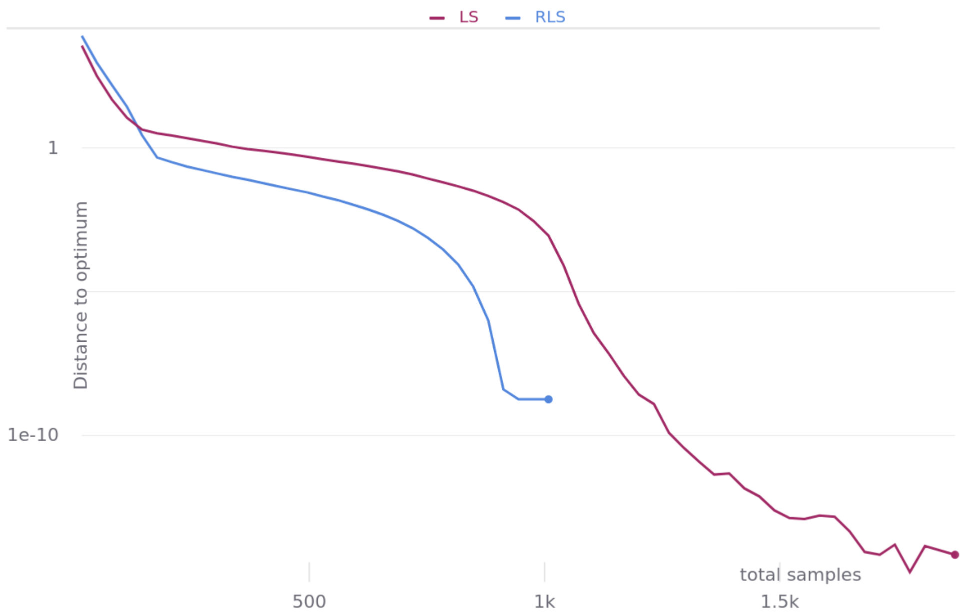

Figure 5.2.: This figure shows the median of (10 runs) for RLS and LS on 5 dimensional Rosenbrock

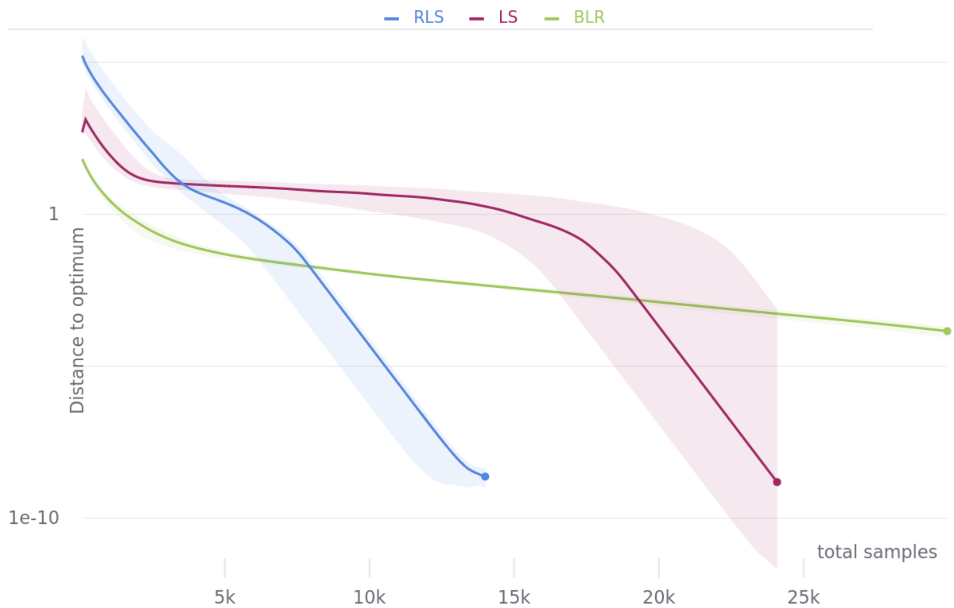

Figure 5.3.: This figure shows the median of (10 runs) for RLS, LS and BLR (original MORE) on 15 dimensional Rosenbrock

5.2.2. Planar Reaching Tasks

Dynamic Movement Primitives

- dynamic system-based motor primitives based on work [Ijspeert 2002, Schaal 2003,2007] improved imitation and reinforcementlearning

- represent both discrete point-to-point movements as well as rhythmic motion with motor primitives

- first system canonical system (phase variable)

- we can build complex motor skills from basic primitives, its based on dynamical systems

- discrete movement primitives using only a single first order system

- key advantage the second system is linear in the shape parameters and can be efficiently learned

- initialized with imitation learning

- DMPs as policy representation

5.2.3. Via Point Reaching Task

- TODO: create figure to explain task

We used a 5 link robot, similar to MORE setup. Therefore 25 dimensions of the problem, which should be at viapoint (1,1) at time step 100 and at viapoint (5,0) at timestep 200. In Figure 5.4 we see the task solved by the original MORE algorithm.

Problem of 351 parameters for recursive least squares

5.2.4. Hole Reaching Task

5 link robot, that has to reach into hole at [2,0] without collision with ground or walls.

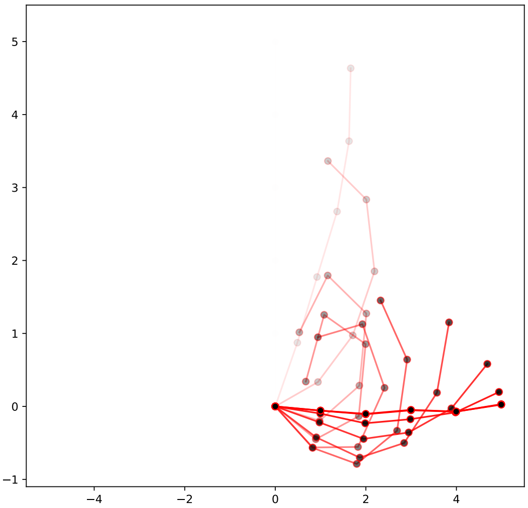

Figure 5.4.: This figure shows the movement resulting from the policy learned by BLR for the viapoint reaching task. The postures of the resulting motion are shown as overlay, where darker postures indicate a posture which is close in time to the hole.

5.2.5. Tests on preprocessing techniques

Here I want to include some tests on different versions of final algorithm and what we learned form experiments. Like using weighted model noise on older samples, and normalization but also different KL-bounds etc.

Tests on data preprocessing methods, different normalization techniques. Whitening very important, making it possible to track the model parameters more easily.

- compare normalization techniques (moving average, batch normalization)

- best performance comparison (more iterations)

- best performance comparison of different methods (total samples)

- test influence of KL bound

- test influence of whitening, normalization?

- test influence of noise of drift model

5.3. Evaluation

- some theoretical problems: sample pool, model noise

- improve sample efficency of rosenbrock and reaching task, better than original MORE and beating CMA-ES benchmark

Chapter 6.

Conclusion and Future Work

6.1. Conclusion

- managed to improve sample efficiency on simple test functions

We achieved some good results on the rosenbrock test function considerably improving the sample efficiency. Nonetheless for the planar reaching tasks we did not manage to achieve a significant improvement. Also our proposed algorithm did not work with multi modal and noisy objective functions.

- One main problem seems to be tuning the recursive estimation for one specific task.

The data seems to be difficult to Still

- inexperience of author with kalman filter (setting up matrices, choosing good state transition model)

6.2. Future Work

- remove sample pool

- usage of state transition model (kalman filter prediction step)

- make learning more stable for reaching tasks (safe exploration)

- learning the model noise: run original MORE approach and do backpropagation with pytorch over the weights

Bibliography

A. Abdolmaleki, R. Lioutikov, J. R. Peters, N. Lau, L. Pualo Reis, and G. Neumann. Model-based relative entropy stochastic search. *Advances in Neural Information Processing Systems*, 28:3537–3545, 2015.

R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

S. Chen. Kalman filter for robot vision: a survey. *IEEE Transactions on Industrial Electronics*, 59(11):4409–4420, 2011.

M. P. Deisenroth, G. Neumann, and J. Peters. *A survey on policy search for robotics*. now publishers, 2013.

A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian data analysis*. CRC press, 2013.

N. Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation*, pages 75–102, 2006.

A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. *Advances in neural information processing systems*, 15:1547–1554, 2002.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, Mar. 1960.

B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg. A survey of research on cloud robotics and automation. *IEEE Transactions on automation science and engineering*, 12(2):398–409, 2015.

A. Kessy, A. Lewin, and K. Strimmer. Optimal whitening and decorrelation. *The American Statistician*, 72(4):309–314, 2018.

J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

A. G. Kupcsik, M. P. Deisenroth, J. Peters, G. Neumann, et al. Data-efficient generalization of robot skills with contextual policy search. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence, AAAI 2013*, pages 1401–1407, 2013.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

M. Molga and C. Smutnicki. Test functions for optimization needs. *Test functions for optimization needs*, 101:48, 2005.

A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287, 1999.

J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

T. Osa, J. Pajarinen, G. Neumann, J. Bagnell, P. Abbeel, and J. Peters. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, 7(1-2):1–179, Mar. 2018.

J. Peters, K. Mülling, and Y. Altun. Relative entropy policy search. In *AAAI*, volume 10, pages 1607–1612. Atlanta, 2010.

M. J. Powell. The bobyqa algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, pages 26–46, 2009.

S. Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103, 1998.

S. Särkkä. *Bayesian filtering and smoothing*, volume 3. Cambridge University Press, 2013.

S. Schaal. The new robotics—towards human-centered machines. *HFSP journal*, 1(2): 115–126, 2007.

S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. In *Robotics research. the eleventh international symposium*, pages 561–572. Springer, 2005.

J. G. Schneider. Exploiting model uncertainty estimates for safe dynamic control learning. *Advances in neural information processing systems*, pages 1047–1053, 1997.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

R. S. Sutton, A. G. Barto, and R. J. Williams. Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems Magazine*, 12(2):19–22, 1992.

V. Tangkaratt, H. van Hoof, S. Parisi, G. Neumann, J. Peters, and M. Sugiyama. Policy search with high-dimensional context variables. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

S. Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

Appendix A.

Appendix

A.1. MORE: Equations

- add derivation of lagragian function for MORE with clear steps

A.2. Gaussian probability function

A random variable $x \in \mathbb{R}^n$ has a Gaussian distribution with mean $m \in \mathbb{R}^n$ and covariance $P \in \mathbb{R}^{n \times n}$ if its probability density has the form

$$\mathrm{N}(x|m,P) = \frac{1}{(2\pi)^{n/2}|P|^{1/2}} \exp\left(-\frac{1}{2}(x-m)^T P^{-1}(x-m)\right)$$

where $|P|$ is the determinant of the matrix P.

A.3. Kalman Filter Derivation