

English Title

Deutscher Titel

Bachelor-Thesis von John Doe aus Birthplace
Januar 2018



TECHNISCHE
UNIVERSITÄT
DARMSTADT



English Title
Deutscher Titel

Vorgelegte Bachelor-Thesis von John Doe aus Birthplace

1. Gutachten: Prof. Dr. N. N.
2. Gutachten: Prof. Dr. N. N.
3. Gutachten: Prof. Dr. N. N.

Tag der Einreichung:

Please cite this document with:

URN: urn:nbn:de:tuda-tuprints-38321

URL: <http://tuprints.ulb.tu-darmstadt.de/id/eprint/3832>

Dieses Dokument wird bereitgestellt von tuprints,

E-Publishing-Service der TU Darmstadt

<http://tuprints.ulb.tu-darmstadt.de>


tuprints@ulb.tu-darmstadt.de



This publication is licensed under the following Creative Commons License:

Attribution – NonCommercial – NoDerivatives 4.0 International

<http://creativecommons.org/licenses/by-nc-nd/4.0/>



For Thomas Hesse and Kevin Luck

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

In der abgegebenen Thesis stimmen die schriftliche und elektronische Fassung überein.

Darmstadt, den 23. Januar 2018

(John Doe)

Thesis Statement

I herewith formally declare that I have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.

In the submitted thesis the written copies and the electronic version are identical in content.

Darmstadt, January 23, 2018

(John Doe)

Abstract

Reinforcement learning relies on policy gradient but the gradient is known only in expectation and most of the time stochastic policies. This leaves some room for zero order methods and BO can combine solving the problem and the exploration strategy from deterministic policies. We investigate in this paper how to integrate efficient exploration strategies stemming from Bayesian optimization for solving high dimensional reinforcement learning problems. We propose a novel optimization algorithm that is able to scale Bayesian optimization to such high dimensional tasks by restricting the search to the local vicinity of a search distribution and by proposing kernels capturing similarity in behavior rather than parameter. We show in the experiments that our approach can be very useful for applications such as robotics.

Zusammenfassung

Das Ziel im bestärkten Lernen ist das Finden einer Strategie, welche die erhaltene Belohnung eines Agenten maximiert. Da der Suchraum für mögliche Strategien sehr groß sein kann, verwenden wir Bayesian optimization, um die Anzahl der Evaluierungen durch den Agenten zu minimieren. Das hat den Vorteil, dass zeit- und kostenaufwändige Abläufe, wie beispielsweise das Bewegen eines Roboterarms, reduziert werden. Die Effektivität der Suche wird maßgeblich von der Wahl des Kernels beeinflusst. Standardkernel in der Bayesian optimization vergleichen die Parameter von Strategien um eine Vorhersage über bisher nicht evaluierte Strategien zu treffen.

Der Trajectorykernel vergleicht statt der Parameter, die aus den jeweiligen Strategien resultierenden Verhaltensweisen. Dadurch werden unterschiedliche Strategien mit ähnlichem Resultat von der Suche weniger priorisiert.

Wir zeigen die Überlegenheit des verhaltensbasierten Kernels gegenüber dem parameterbasierten anhand von Roboters-terierungssimulationen.

Acknowledgments

Contents

1. Introduction	2
1.1. Getting started	2
1.2. Documentation	4
2. Motivation	6
3. Foundations	7
3.1. Global Bayesian optimization	7
3.2. Hyper parameter optimization	7
3.3. Markov decision process	7
3.4. Kernel for Gaussian process	8
3.5. Acquisition function	9
3.6. Gaussian Process Regression	10
4. Experiments	11
4.1. Implementation	11
4.2. Cart pole	13
5. Results	14
6. Discussion	15
7. Outlook	16
Bibliography	17
A. Some Appendix	19

Figures and Tables

List of Figures

1.1. The structure of the IAS Thesis L ^A T _E X-Framework illustrated.	4
---	---

List of Tables

Abbreviations, Symbols and Operators

1 Introduction

1.1 Getting started

1.1.1 Installing Glossaries

Note for Windows users: While the `makeglossaries` command is a perl script for Unix users, there is also a `.bat` version of the file for Windows users. However, I don't know how to set up MiKTeX or equivalent to use this package. Feel free to add a comment if you can add information about this step.

1. **Get and unzip the glossaries package.** I downloaded it from [here](#). Though you can download the source and compile, I found it much easier to simply download the tex directory structure (tds) zip file. Unfortunately, the `texlive-latex-extra` package available on ubuntu or kubuntu does not contain the glossaries package – it only contains glossary and acronym. I unzipped the contents of the zip file into a directory called “texmf” in my home directory. You'll also want to run “`texhash /texmf/`” to update the latex database, according to the INSTALL instructions.
2. **(Optionally) get the xfor package.** If your system is like mine, after you've installed the glossaries package latex will complain that it doesn't have the xfor package (which also is not available via apt-get in Ubuntu). Download this package from [here](#).
3. **Open the glossaries zip as root in a nautilus window, terminal window, or equivalent.** You'll be copying the contents to various locations in the root directory structure, and will need root access to do this.
4. **Find the location of your root texmf directory.** In Karmic, this is `/usr/share/texmf/`, though it may be in another location on your system. Generally, you should have a local texmf folder, i.e. `/texmf/`, when receiving the IAS slide L^AT_EX template.
5. **Copy the contents of the tex and doc directories from the glossaries zip into the matching directory structure in your texmf directory.** For me, this meant copying the “doc/latex/glossaries” subdirectory in the zip file to “`/usr/share/texmf/doc/latex/`”, and the same for the tex directory (copy “tex/latex/glossaries” subdirectory in the zip file to “`/usr/share/texmf/tex/latex/`”). In theory, you can also copy the scripts/ directory in the same way, but I did step 6 instead, as this is what was suggested in the INSTALL document.
6. **Update the master latex database.** Simply run the command “`sudo mktexlsr`”
7. **Add the location of your scripts/glossaries directory to your \$PATH.** This gives programs access to `makeglossaries`, the perl script you will be using (if you're in linux/unix). If you followed my default instructions in step 1, this location will be “`/home/yourname/texmf/scripts/glossaries`”.
8. **Test the installation.** Change into the directory you created in step 1, into the “doc/latex/glossaries/samples/” subdirectory. There, run “`latex minimalgls`”. If you get an error about xfor, please see step 9. Otherwise, run “`makeglossaries`” and then “`latex minimalgls`” again. If everything works, the package is set up for command-line use. You may wish to modify your Kile setup to use glossaries – go to step 10 if this is the case.
9. **Set up the xfor package.** Run steps 3-6 again, but with the `xfor.tds.zip` file instead of the glossaries zip file. This package is simpler than glossaries, and does not contain a scripts/ subdirectory, so you will not need to do step 7. After installation, try running step 8 again: everything should work.

Source: [link](#)

1.1.2 Configure the Modification of the TU Design

You can find in the IAS Thesis folder a folder with the name *texmf*. This folder includes some modifications of the TUD design, for example an updated Thesis statement in english and german. After the installation of the TUD design (<http://exp1.fkp.physik.tu-darmstadt.de/tuddesign/>) you have to move the folder to your home folder. If the folder already exists, then move only the tud-files. Then you have to run the command *texhash ~/texmf* such that Latex can use the new files. Please note, that the *texmf* folder already includes some adaptations for the tud-beamer template. If you want to use the original TUD design again, rename the *texmf* folder and run *texhash* again.

If you have questions regarding the modifications of the TU design or suggestions, please let me know and send me an email to luck@ias.tu-darmstadt.de

1.2 Documentation

1.2.1 Structure of the IAS \LaTeX -Framework

The structure of this framework is illustrated in the following figure.

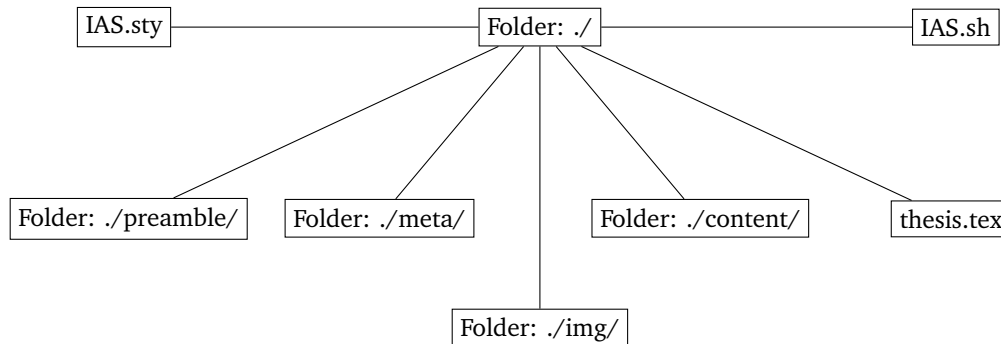


Figure 1.1.: The structure of the IAS Thesis \LaTeX -Framework illustrated.

The **./preamble/** folder should contain content that needs to be processed in the preamble section, i.e. before `\begin{document}`, as the name suggests.

The **./meta/** folder should contain content that is not directly related to the topic of the thesis or summarizes content of the thesis, i.e. `abstract.tex` or `acknowledgements.tex`.

The **./img/** folder should solely contain images, e.g. `png`, `eps`, etc.

The **./content/** folder is the most important for the user ever since you stuff in all your content related files / chapters in here. You can `\input{yourFile}` afterwards in the `./thesis.tex` where the content variable is defined.

1.2.2 Commands & shortcuts

There are several shortcuts and commands you should memorize!

- Wrapper command for vectors

`\cvec{}`

Example: **v**

- Wrapper command for matrices

`\cmat{}`

Example: **M**

- Shortcut for `\textbf{}`

`\bf{}`

Example: **bold font**

- Shortcut for `\textit{}`

`\it{}`

Example: *italic font*

- Shortcut for `\underline{}`

`\ul{}`

Example: underline

- Shortcut for `\mathcal{}`

`\mc{}`

Example: $\mathcal{N}(\mu, \Sigma)$

1.2.3 Getting started with Glossaries

For a comprehensive guide to glossaries, you should read this article. There is also sample code given in `./preamble/glossary.tex` which you should have a look at as well!



2 Motivation

TODO

3 Foundations

- Bayesian optimization
- Global optimization
- Local optimization
- Gaussian Process Regression
- Hyper Parameter optimization
- Expected Improvement
- Thompson Sampling
- Standard kernel
- Trajectory kernel

3.1 Global Bayesian optimization

```
init = 10
X = init uniformly random samples from the search space
Y = evaluations of the objective at the points X
for n = init to 200 + init do
    compute K(X, X)
    Optimize hyper parameters (optional)
    xn+1 = point at the optimum of the acquisition function over the Gaussian Process
    yn+1 = evaluation of the objective at the point xn+1
    X = {X, xn+1}
    Y = {Y, yn+1}
    n = n + 1
```

During acquisition function optimization $K(X, X)$ from the Gaussian process does not change, so it is precomputed. The values for the initial sample count and the iteration count are not fixed, but for simplification set here.

3.2 Hyper parameter optimization

Selecting proper hyper parameters for the Gaussian process regression enhances the efficiency of our learning algorithm. Also it helps avoiding numerical problems. To find an optimum for the signal variance hyperparameter σ_f and the length scale hyperparameter σ_l we maximize the log marginal likelihood function

$$\log p(y|x, \sigma_f, \sigma_l) = -\frac{1}{2}y^\top K_n^{-1}y - \frac{1}{2}\log|K_n| - \frac{n}{2}\log 2\pi, \quad (3.1)$$

from our Gaussian process. Where n is the number of observations, x is the $d \times n$ dataset of inputs and K_n is the covariance matrix for the noisy target y .

3.3 Markov decision process

In our reinforcement learning problem we use a Markov decision process model to describe possible decisions as probabilities. Our model consists of a tuple (S, A, P, P_0, R) , holding all states $s \in S$, all actions $a \in A$, all state transitioning probabilities, and all corresponding rewards. Assume an agent which executes a policy x for T time steps receiving a final reward $\bar{R}(\xi)$. This reward depending on a policy is what we want to maximize.

We formulate the conditional probability of observing trajectory ξ given policy x by

$$P(\xi|x) = P_0(s_0) \prod_{t=1}^T P(s_t|s_{t-1}, a_{t-1}) P_\pi(a_{t-1}|s_{t-1}, x)$$

in which trajectory $\xi = (s_0, a_0, \dots, s_{T-1}, a_{T-1}, s_T)$ contains the sequence of state, action tuples and $x \in \mathbb{R}^d$ a set of d policy parameters. $P_0(s_0)$ is the probability of starting in the initial state s_0 . $P(s_t|s_{t-1}, a_{t-1})$ is the probability of transitioning from state s_{t-1} to s_t when action a_{t-1} is executed. The stochastic mapping $P_\pi(a_{t-1}|s_{t-1}, x)$ is the probability for selecting the action a_{t-1} when in state s_{t-1} and executing the parametric policy x . So we receive a final reward for a sampled trajectory,

$$\bar{R}(\xi) = \sum_{t=1}^T R(s_{t-1}, a_{t-1}, s_t),$$

which is the sum of all immediate rewards, given by an rewarding function $R(s_{t-1}, a_{t-1}, s_t)$. This rewarding function depends on the given environment. For example it rewards a state we want to achieve by returning a value greater than zero and can penalize states we do not want our agent to be in by returning negative values.

3.4 Kernel for Gaussian process

3.4.1 Squared exponential kernel

$$D(x_i, x_j) = x_i - x_j$$

$$K(x_i, x_j, \sigma) = \sigma_f^2 \exp\left(-\frac{D^2(x_i, x_j)}{2\sigma_l^2}\right)$$

3.4.2 Matern 5/2 kernel

$$D(x_i, x_j) = x_i - x_j$$

$$K(x_i, x_j, \sigma) = \sigma_f^2 \exp\left(-\frac{D^2(x_i, x_j)}{2\sigma_l^2}\right)$$

3.4.3 Trajectory kernel

Standard kernels like the squared exponential kernel, relate policies by measuring the difference between policy parameter values. Therefore policies with similar behavior but different parameters are not compared adequately. The behavior based Trajectory kernel fixes this, by relating policies to their resulting behavior. This makes our policy search more efficient, since we avoid redundant search of different policies with similar behaviour. To examine the difference between two policies x_i and x_j the discrete Kullback Leibler divergence

$$D_{\text{KL}}(P(\xi|x_i)||P(\xi|x_j)) = \sum_i P(\xi|x_i) \log \frac{P(\xi|x_i)}{P(\xi|x_j)}.$$

is applied to the respective policy-trajectory mappings $P(\xi|x_i)$ and $P(\xi|x_j)$. It measures how the two probability distributions diverge from another.

In general $D_{\text{KL}}(P(\xi|x_i)||P(\xi|x_j))$ is not equal to $D_{\text{KL}}(P(\xi|x_j)||P(\xi|x_i))$. But we need a symmetric distance measure. So we sum up the two divergences

$$D(x_i, x_j) = D_{\text{KL}}(P(\xi|x_i)||P(\xi|x_j)) + D_{\text{KL}}(P(\xi|x_j)||P(\xi|x_i)),$$

to achieve that $D(x_i, x_j) = D(x_j, x_i)$. An additional requirement for the kernel is the resulting matrix to be positive semi-definite and scalable[1]. Therefore we exponentiate the negative of our distance matrix D . We also apply the hyper parameters σ_f to compensate for the signal variance and σ_l to adjust for signal scale. This gives us the covariance matrix

$$K(x_i, x_j, \sigma) = \sigma_f \exp(-\sigma_l D(x_i, x_j)), \quad (3.2)$$

for the gaussian process.

3.5 Acquisition function

The core of the Bayesian optimization consists of selecting the next point, in our search space, to evaluate. We get that point by executing a so called acquisition function. It depends on the incumbent model of the true objective and returns that next point. In our case a point means a set of policy parameters. We choose expected improvement during the global bayesian optimization and in the local optimization we use Thompson sampling. Both acquisition functions take the mean und the variance from the gaussian process model into account to guide the exploration process. The difficulty lies in avoiding excessive exploration or exploitation. Exploration looks for points with a high variance and exploitation selects points with a high mean instead. The latter one would result in a local optimum whereas too much exploration may not improve at all. (To not be confused, bla, local optimum != optimum in local optimization)

3.5.1 Thompson sampling

(refactor full covariance matrix Σ , since σ represents deviation and σ^2 variance) For the Thompson sampling we sample one function from the Gaussian process posterior,

$$f \sim \text{GP}(0, K(X, X_*)),$$

where X is the $n \times d$ dataset of already evaluated points, and X_* is a randomly Gaussian distributed set of points with mean and variance given by the local optimizer. To draw values from the distribution we need the mean vector μ and the full covariance matrix V from the gaussian process model. First we take the lower Cholesky decomposite of V such that $AA^T = V$. Then we compute a vector z , which consists of independent Gaussian distributed valus with zero mean and unit variance. Finally we get a vector f of sampled values:

$$f(x_*) = \mu + Az.$$

We take the one with the highest value such that,

$$x_{n+1} = \arg \max_{x_*} f(x_*),$$

to return the next evaluation point.

3.5.2 Expected improvement

To calculate an expected improvement function value we need the mean value $\mu(x_*)$, and the standard deviation value $\sigma(x_*)$ at the test point x_* from the gaussian process. Also we need the maximum of all observations y_{max} and a trade-off parameter τ . For the cumulative distribution function and the probability density function from the Gaussian distribution we write $\Phi(\cdot)$ and $\phi(\cdot)$ respectively. They both have zero mean and unit variance. We adopt the expected improvement function

$$EI(x_*) = \begin{cases} (\mu(x_*) - y_{max} - \tau)\Phi(Z(x_*)) + \sigma(x_*)\phi(Z(x_*)) & \text{if } \sigma(x_*) > 0 \\ 0 & \text{if } \sigma(x_*) = 0 \end{cases}$$

where

$$Z(x_*) = \begin{cases} \frac{(\mu(x_*) - y_{max} - \tau)}{\sigma(x_*)} & \text{if } \sigma(x_*) > 0 \\ 0 & \text{if } \sigma(x) = 0 \end{cases}$$

as suggested in [2]. The trade-off parameter τ is set to 0.01 accordingly. The expected improvement function is then optimized over the whole search space to give us the next evaluation point

$$x_{n+1} = \arg \max_{x_*} EI(x_*).$$

3.6 Gaussian Process Regression

Since we want to estimate an objective function in a machine learning environment we elect to use Gaussian process regression. It fits a multivariate gaussian distribution over our prior data. From the regression we get a posterior mean and variance which describe our model of the objective function. The mean represents a prediction of the true objective at a given point and the variance represents the uncertainty at that point. The more points our model incorporates the smaller the variance, and the preciser the predictions, in the vicinity around prior points.

In real world applications we always have some noise in our objective observations. Therefore a Gaussian distributed error term,

$$\epsilon \sim \mathcal{N}(0, \sigma_n^2),$$

with zero mean and σ_n^2 variance is added to the function value. The observed target

$$y_n = f(x) + \epsilon$$

regards this noise. Before doing regression we transform our observations to zero mean and uniform variance:

$$y = \frac{y_n - \mu_{y_n}}{\sigma_{y_n}}.$$

The knowledge our training data provides is represented by the kernel matrix $K(x, x)$. With a vector of test points x_* we get the joint distribution of the normalized target values and the function values at the test locations:

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(x, x) + \sigma_n^2 I & K(x, x_*) \\ K(x_*, x) & K(x_*, x_*) \end{bmatrix}\right)$$

Now we can calculate the posterior mean and variance at given test points.

$$K_n(x, x) = K(x, x) + \sigma_n^2 I \tag{3.3}$$

$$\mu(x_*) = K(x_*, x) K_n(x, x)^{-1} y \tag{3.4}$$

$$\sigma^2(x_*) = K(x_*, x_*) - K(x_*, x) K_n(x, x)^{-1} K(x, x_*)$$

4 Experiments

4.1 Implementation

4.1.1 Normalization constant

4.1.2 Numerical stability

in the end -> 1. explain what we want to achieve (no log of small values for example) 2. cholesky 3. hyp param matrix

4.1.3 Bayesian Optimizaion algorithm

4.1.4 Cholesky decomposition

(GP, TS)

4.1.5 OpenAI Gym in Python

4.1.6 Gaussian Process Regression

(difference between full covariance and cov vector)

Instead of calculating the inverse of K_n in (3.4) we use the lower Cholesky decomposed matrix:

$$LL^T = K_n$$

This is considered faster and numerically more stable [3]. $K_n^{-1}y$ then transforms to:

$$K_n^{-1}y = (LL^T)^{-1}y = (L^{-T}L^{-1})y = L^{-T}(L^{-1}y) = L^T \setminus (L \setminus y). \quad (4.1)$$

The backslash operator denotes the matrix left division, so the solution $x = A \setminus b$ satisfies the system of linear equations $Ax = b$. Matrix K_n must be positive definite for the cholesky decomposition. So we double the noise variance hyperparameter σ_n^2 from (3.3) until positive definiteness is achieved.

4.1.7 Hyper Parameter Optimization

Especially for the trajectory kernel we want a hyper parameter optimization, because all the values of the distance matrix D may get very big. When dividing by a well tuned hyper parameter σ_l before applying the exponential function (3.2), we avoid getting a zero kernel matrix K .

When calculating $\log(|K_n|)$ for the hyperparameter optimization (3.1), again we use the Cholesky decomposition of K . Thus the determinant transforms to

$$|K_n| = |LL^T| = |L||L^T| = |L||L| = |L|^2.$$

Since the determinant of the Cholesky decomposed matrix,

$$|L| = \prod_i L_{ii},$$

is the product of its diagonal elements, we can transform this into a numerically more stable version:

$$\log(|K_n|) = \log(|L|^2) = 2 \log(|L|) = 2 \log(\prod_i L_{ii}) = 2 \sum_i \log(L_{ii}).$$

The computation of $K_y^{-1}y$ in (3.1) is done by the same method we already use in the Gaussian process (4.1).

4.1.8 Estimation of Trajectory Kernel Function Values

We estimate the divergence values, because the computational effort will be greatly reduced[1]. We use a Monte-Carlo estimate for the approximation

$$\hat{D}(x_i, x_j) = \sum_{\xi \in \xi_i} \log \left(\frac{P(\xi|x_i)}{P(\xi|x_j)} \right) + \frac{1}{N} \sum_{\xi \in \xi_j} \log \left(\frac{P(\xi|x_j)}{P(\xi|x_i)} \right)$$

of the divergences between policies with already sampled trajectories. Here ξ_i is the set of trajectories generated by policy x_i . For our gaussian process regression we also need a distance measure between a policy with known trajectories and new policies with unknown trajectories. Since there is no closed form solution to this we use the importance sampled divergence

$$\hat{D}(x_{new}, x_j) = \sum_{\xi \in \xi_j} \left[\frac{P(\xi|x_{new})}{P(\xi|x_j)} \log \left(\frac{P(\xi|x_{new})}{P(\xi|x_j)} \right) + \log \left(\frac{P(\xi|x_j)}{P(\xi|x_{new})} \right) \right]$$

to estimate the divergence between the new policy x_{new} and the policy x_j with already sampled trajectories ξ_j . Since we only have a ratio of transitioning probabilities present in our trajectory kernel we can reduce the logarithmic term to:

$$\begin{aligned} \log \left(\frac{P(\xi|x_i)}{P(\xi|x_j)} \right) &= \log \left(\frac{P_0(s_0) \prod_{t=1}^T P_s(s_t|s_{t-1}, a_{t-1}) P_\pi(a_{t-1}|s_{t-1}, x_i)}{P_0(s_0) \prod_{t=1}^T P_s(s_t|s_{t-1}, a_{t-1}) P_\pi(a_{t-1}|s_{t-1}, x_j)} \right) \\ &= \log \left(\prod_{t=1}^T \frac{P_\pi(a_{t-1}|s_{t-1}, x_i)}{P_\pi(a_{t-1}|s_{t-1}, x_j)} \right) \\ &= \sum_{t=1}^T \log \left(\frac{P_\pi(a_{t-1}|s_{t-1}, x_i)}{P_\pi(a_{t-1}|s_{t-1}, x_j)} \right) \end{aligned}$$

Summing up the logarithms in the end is also numerically more stable than taking the logarithm of the whole product.

4.1.9 Action selection

In continuous action space we use a linear policy to action mapping

$$a = f_s(s)^\top x + \epsilon_a,$$

with a small gaussian noise ϵ_a needed for stochastic policies. So the actions are Gaussian distributed:

$$a \sim \mathcal{N}(f_s(s)x, \epsilon_a^2).$$

Therefore the resulting probability density of the action selection,

$$P_\pi(a|s, x) = \frac{1}{\sqrt{2\pi\epsilon_a^2}} \exp \left(-\frac{(a - f_s(s)x)^2}{2\epsilon_a^2} \right),$$

enables us to do the computations of the logarithm of the probability ratios in the trajectory kernel more efficient:

$$\begin{aligned} \sum_{t=0}^{T-1} \log \left(\frac{P_\pi(a_t|s_t, x_i)}{P_\pi(a_t|s_t, x_j)} \right) &= \sum_{t=0}^{T-1} \log \left(\frac{\frac{1}{\sqrt{2\pi\epsilon_a^2}} \exp \left(-\frac{(a_t - f_s(s_t)x_i)^2}{2\epsilon_a^2} \right)}{\frac{1}{\sqrt{2\pi\epsilon_a^2}} \exp \left(-\frac{(a_t - f_s(s_t)x_j)^2}{2\epsilon_a^2} \right)} \right) \\ &= \sum_{t=0}^{T-1} \log \left(\exp \left(-\frac{(a_t - f_s(s_t)x_i)^2}{2\epsilon_a^2} - \left(-\frac{(a_t - f_s(s_t)x_j)^2}{2\epsilon_a^2} \right) \right) \right) \\ &= \frac{1}{2\epsilon_a^2} \sum_{t=0}^{T-1} ((a_t - f_s(s_t)x_j)^2 - (a_t - f_s(s_t)x_i)^2). \end{aligned}$$

The function $f_s(s)$, depending on the state, computes our d dimensional state feature vector. In discrete action space environments we use a parametric soft-max action selection policy:

$$P(a|s) = \frac{\exp(f_s(s)^\top x_a)}{\sum_{i \in A} \exp(f_s(s)^\top x_i)}.$$

Again it consists of the linear mapping $f_s(s)^\top x$ and A holds all possible actions. The resulting action is then sampled from the probability of action a given state s .

4.2 Cart pole

In the cart pole experiment for example it is simply returning the four state values. During the cart pole experiments it has been found useful to threshold the resulting action to $[-1, 1]$ to prevent unrealistic behaviour in the simulation.

5 Results

TODO

6 Discussion

TODO



7 Outlook

TODO

Bibliography

- [1] A. Wilson, A. Fern, and P. Tadepalli, “Using trajectory data to improve bayesian optimization for reinforcement learning,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 253–282, 2014.
- [2] E. Brochu, V. M. Cora, and N. De Freitas, “A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning,” *arXiv preprint arXiv:1012.2599*, 2010.
- [3] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*, vol. 1. MIT press Cambridge, 2006.
- [4] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, “Taking the human out of the loop: A review of bayesian optimization,” *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.



A Some Appendix

Use letters instead of numbers for the chapters.