Artificial Curiosity for Motor Skill Learning

Bewegungslernen mittels Artificial Curiosity

Bachelor-Thesis von Yannick Schröcker

Mai 2014

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Artificial Curiosity for Motor Skill Learning
Bewegungslernen mittels Artificial Curiosity

Vorgelegte Bachelor-Thesis von Yannick Schröcker

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Dr. Gerhard Neumann

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter 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.

Darmstadt, den 28.05.2014

(Yannick Schröcker)

Abstract

Reinforcement learning can be applied to episodic tasks in robotics in order to learn how to perform motor skills without the need for prior knowledge. Algorithms that can learn contextual tasks enable the robot to generalize his learned knowledge about a specific task to another, similar task and to solve this task without further specific training. This requires the robot to train on given contexts which are sampled from a distribution. Typically, a uniform distribution is used. However, the distribution has an effect on the learning process and can be learned in order to improve the final performance of the robot. To this end, we introduce Active Relative Entropy Policy Search (ActiveREPS). ActiveREPS is a modification of Relative Entropy Policy Search (REPS) that learns a context distribution by maximizing the improvement of the expected reward in order to train the robot where it improves most. We show that ActiveREPS can be used to improve the performance of the robot when executing motor skills when compared to reinforcement learning with uniformly sampled contexts. The obtained contexts are then used to learn a policy. To this end, we introduce Local Relative Entropy Policy Search (LocalREPS), an adaptation of REPS that is trained for each context locally. We show in this thesis that LocalREPS is capable of learning policies that depend non-linearly on the context and outperforms REPS on tasks with complex dependencies on the context.

Zusammenfassung

Reinforcement Learning erlaubt es Robotern komplexe, episodische Aufgaben zu lösen ohne auf vorhergehendes Wissen zurückzugreifen. Algorithmen die außerdem kontextuelle Aufgaben lösen können erlauben es Robotern von ihrem gelernten Wissen zu abstrahieren und ähnliche Aufgaben ohne gesondertes Training zu bewältigen. Diese Algorithmen setzen voraus, dass der Roboter auf einer Anzahl an Kontexten zu trainiert wird, üblicherweise werden dazu gleichverteilte Kontexte genommen. Alternativ ist es möglich die Kontextverteilung zu lernen um damit das Endresultat des Lernprozesses zu verbessern. Um dies zu erreichen stellen wir in dieser Thesis Active Relative Entropy Policy Search (ActiveREPS) vor. ActiveREPS ist eine Modifikation von Relative Entropy Policy Search (REPS), die die Kontextverteilung lernt, die den Unterschied im Erwartungswert des Rewards maximiert um den Roboter auf Kontexten zu trainieren, auf denen die größte Verbesserung des Endergebnisses erzielt wird. Wir zeigen in dieser Thesis, dass ein Roboter, der auf von ActiveREPS bestimmten Kontexten trainiert wurde, bessere Resultate beim Bewegungslernen erzielt als ein Roboter, der auf gleichverteilten Kontexten trainiert wurde. Die gelernte Kontextverteilung kann dann benutzt werden um eine Policy zu lernen. Zu diesem Zweck stellen wir in dieser Thesis außerdem Local Relative Entropy Policy Search (LocalREPS) vor. LocalREPS ist eine Variation von REPS, die für jeden Kontext lokal trainiert wird. Wir zeigen in dieser Thesis, dass LocalREPS dadurch policies lernen kann, die nicht-linear von dem Kontext abhängig sind und dass LocalREPS damit bessere Resultate erzielen kann als REPS.

Contents

1 Introduction **5**

2 Related Work **7**

3 Relative Entropy Policy Search **9**
3.1 Policy Search . . . 9
3.2 Relative Entropy Policy Search . . . 9
3.3 Local Relative Entropy Policy Search . . . 11

4 Active Relative Entropy Policy Search **12**
4.1 Active Relative Entropy Policy Search . . . 12
4.2 Using Active Relative Entropy Policy Search for learning the context distribution . . . 13
4.3 Managing the Training Set . . . 13

5 Evaluation **14**
5.1 Table tennis . . . 14
5.1.1 Dynamic Movement Primitives . . . 14
5.1.2 The Learning Setup . . . 14
5.1.3 Results . . . 15
5.2 5-Link Reaching Task . . . 19
5.2.1 The Learning Setup . . . 19
5.2.2 Results . . . 20

6 Conclusion **22**
6.1 Future Work . . . 22

List of Figures

1.1 Diagram of learning process with and without artificial curiosity . . . 5

3.1 Examples of policies . . . 11
a A Gaussian linear policy, learned by REPS . . . 11
b A Gaussian locally linear policy, learned by LocalREPS . . . 11

5.1 Results of the table tennis task . . . 15
a Picture of simulation . . . 15
b Reward obtained on uniform and fixed evaluation samples by REPS, ActiveREPS and SAGG-RIAC with REPS . . . 15
c Reward obtained on training samples by REPS, ActiveREPS and SAGG-RIAC with REPS . . . 15
d Comparison of reward obtained by ActiveREPS on training samples and on evaluation samples . . . 15
5.2 Context-learning process on table tennis . . . 17
a Untransformed reward after 50 iterations . . . 17
b Transformed reward after 50 iterations . . . 17
c Context distribution after 50 iterations . . . 17
d New samples after 50 iterations . . . 17
e Untransformed reward after 100 iterations . . . 17
f Context distribution after 100 iterations . . . 17
5.3 Sampling process of ActiveREPS on table tennis . . . 18
5.4 Planar reaching robot . . . 19
5.5 Results on reaching task . . . 20
a Reward obtained by ActiveREPS, LocalREPS and REPS . . . 20
b Reward obtained without mixture model . . . 20
c Reward obtained with 1d context . . . 20
d Reward obtained with low ϵ . . . 20
5.6 Sampling process of ActiveREPS on reaching task . . . 21

1 Introduction

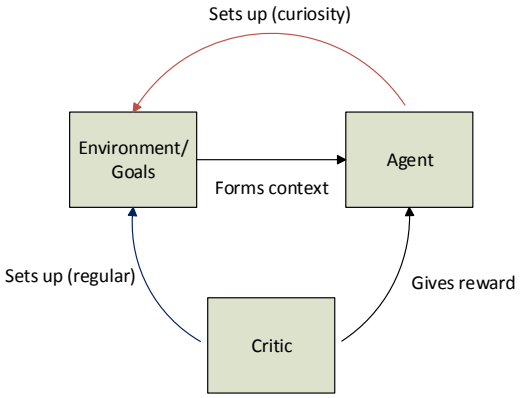


Figure 1.1: Episodic reinforcement learning with and without artificial curiosity

One of the key goals in robotics is and has been to provide robots with the ability to solve tasks without the need of hard coding detailed knowledge about the task and the robot into the robot's system. To this end, reinforcement learning methods have been utilized and have shown good results so far. However, using reinforcement learning to teach robots how to solve a single task is often impractical. Instead, the robot should be able to learn whole classes of tasks and generalize from its past experience. This generalization can be done based on parameters of the task which then form a so called context. A context provides the necessary information about the specific task and can include parameters that describe the state of the environment (which is not influenced by the actions of the robot) as well as the objectives of the robot. One example for a contextual reinforcement learning problem would be a ball-throwing task where a robot-arm has to throw a ball to a certain, freely chosen position: For example, a ball-throwing task might have a context that describes the target. In that case, the robot would have to learn how to aim at arbitrary targets and generalize over its training samples. Another example that we will later use for evaluation as well is a robot arm playing table tennis. In the table tennis task, the robot has to be able to handle different settings of the ball cannon. Another possible parameter would be the target position on the other side of the table. As a result, the context of this task could consist of the forces that the ball cannon applies as well as of the coordinates of the target. Generalization is key in all of these tasks and necessary to solve them efficiently.

While there are algorithms that are cab able of generalizing over different tasks and contexts, the training of these algorithms often uses the same conditions for training the robots as those that are encountered when actually performing the task. This means that the contexts are merely observed and not chosen by the learning process. However, in environments where the context can be controlled, it can be possible to take advantage of this circumstance for training. For example, a human might set a ball cannon in table tennis to aim at a specific corner in order to train that particular stroke more effectively. The reason why a human player might do this is that he has identified the need to improve particular strokes and thinks that he can improve his play best by training them directly. Artificial curiosity is based on this idea and aims to improve the learning process of the robot by giving it the ability to chose contexts during training. The intention is to chose contexts where the robot thinks it can improve its overall performance most. We will mention different approaches to artificial curiosity in Section 2 and introduce our own approach in the remainder of this thesis.

In this paper, we focus on episodic reinforcement learning, where the contexts as well as the actions are chosen in the beginning of each episode and then carried through without further decision making. For example when learning a motor skill, the robot would learn whole trajectories instead of learning each step separately. This is a special case of the more general MDP approach where the agents can execute multiple actions sequentially and reevaluate the next actions after an action has been executed and its effects have been observed. Specifically, our work is based on Episodic Relative Entropy Policy Search which is capable of learning a policy based on actions as well as of generalizing the learned policy based on observed contexts. While the policy is learned by REPS, the contexts for training are sampled based on a fixed distribution; We attempt to learn a better distribution over contexts instead. Figure 1.1 outlines the steps of an episodic learning setup with and without artificial curiosity.

The algorithm we propose in this paper is called Active Relative Entropy Policy Search (ActiveREPS). ActiveREPS is a modification of episodic REPS that learns a context distribution based on the past learning progress. Specifically, ActiveREPS is optimizing the gain in expected reward in comparison to the previous iteration. This is achieved by a modification of REPS' objective function that we will explain in detail in Section 4. The idea behind this metric is to advance the learning as quickly as possible by choosing contexts for training where the change in the expected reward is as high as possible.

The core idea of artificial curiosity and thus the core idea behind ActiveREPS is to have the algorithm choose the context of a task in order to improve the learning process. This means that the algorithm exploits the effect that different choices for contexts have on the learning progress. As a consequence, the most interesting kind of task for artificial curiosity algorithms are tasks where the effect of the context is large. However, as we will show in Section 3, REPS has limitations specific to this kind of task. In order to solve this problem, we furthermore propose Local Relative Entropy Policy Search (LocalREPS). LocalREPS is a modification of REPS that is trained locally around a specified context. We will show that LocalREPS is capable to learn policies on tasks where the performance of regular REPS is limited by its assumptions over the influence of the context-space.

In Section 3, we will recapitulate REPS and introduce LocalREPS while we will present ActiveREPS in Section 4. In Section 5, we will then evaluate our new approach on the already mentioned table-tennis-task as well as on a two dimensional reaching task.

2 Related Work

Very closely related to the problem of choosing context is the field of exploration in reinforcement learning. Typically, this term is used when talking about a set-up where the agent is trying different actions in a certain state in order to explore how they affect the state of the robot and the environment. In an episodic set-up, on the other hand, we want to explore contexts that are not manipulated by the actions of the agent or rather can only be manipulated by synthetic actions which are made-up in order to have the agent control the environment artificially (which can be done to make the problem-definitions compatible). This difference is particularly obvious when considering the exploration-exploitation trade-off that is usually addressed by exploration-algorithms(e.g. E3[7] or R-Max[4]); Exploitation chooses actions that are optimal to accomplish the goal where exploration chooses actions to gain knowledge about them. However, in this thesis we want to choose contexts for training. An exploitation strategy when choosing contexts is impossible since the context defines the task and can only be chosen during training.

There are different approaches to exploration that have been used in the past. The easiest approach is random exploration which chooses actions randomly. For example, ϵ-greedy executes the optimal action with a set probability while sampling them uniformly, i.e. exploring, otherwise. Another example would be a soft-max policy which does not explore uniformly but uses the Q-value of executing an action in a state to determine a probability with which that action is executed. These probabilities are then used to randomly choose an action.

A more sophisticated approach can be used for model-based learning algorithms and is based on the idea that the solutions for model-based reinforcement learning algorithms get approximately optimal when the relevant states have been visited often enough. One of the most well-known algorithms based on this idea is E3[7]. E3 defines an explicit exploration and an explicit exploitation strategy where exploitation is done by solving the MDP only on the states that have been visited at least m times(the "known" states) to reach the goal state. To perform exploration, first the MDP is solved on the known states in order to reach an unknown state. When the algorithm reaches an unknown state it explores by performing "balanced wandering", meaning that the algorithm executes the action that has been executed the least for this state and follows this course of action until it hits a known state again. Here it can then follow either the exploitation or the exploration policy. Another popular algorithm in this class is $R-Max$[4], which makes the exploration-exploitation trade-off implicit by assigning the maximum reward to states that have not yet been visited m times, while solving the problem using a regular reinforcement-learning algorithm.

A third approach to exploration is to learn probabilistic policies which explore different actions naturally by sampling actions from a probability distribution. Examples where this approach has been used include Bayesian model-based learning, such as in value-function-methods which are intractable but can be approximated(BEB[9]), as well as REPS.

A special kind of exploration problem is called artificial curiosity. This kind of exploration is concerned with the exploration of tasks based on past feedback, i.e. rewards. As a result, it has more resemblance to our own problem. Early work in the field of artificial curiosity include the work of Schmidhuber et. al.[14] who were doing model-based reinforcement learning based on neural networks and added another, "intrinsic", reward function based on the difference between the actual performance of the agent and the outcome that was predicted by the internal model. A similar reward-term has been used in [15] which incorporated the same kind of intrinsic reward into the reward function of an option-learning algorithm that learns a whole database of skills instead of how to perform single tasks. The algorithm uses the curiosity induced by this reward function to learn a new task whenever it detects an interesting task in an unspecified manner.

Different kinds of intrinsic reward function have been developed as well, for example by Lopes et. al.[10] who proposed a reward similar to $R-max$, only that the reward is set to maximum based on the competence of the learning-algorithm on that state instead of conditioning the reward function on the amount of visits to the state in question. The competence-measure that is proposed in that paper is based on the change in the estimated log-likelihood of the model as well as the estimated variance. Lopes et. al. also proposed a similar modification of BEB which is using a reward function based on visitation-counts as well.

All the previously mentioned algorithms for artificial curiosity drive the learning process by defining an intrinsic reward function. Intrinsic reward-signals have also been found in the psychology of many animals. In [16], Singh et. al. provide a comparison of intrinsic reward functions used for artificial curiosity with intrinsic reward signals that are hypothesized in evolutionary psychology. They derive another intrinsic reward-function that is optimal based on a more general fitness-function. However, the fitness function has to be defined by the designer of the robot. It needs to evaluate the history of

an agent similarly to a reward function but can be can be derived from the task in a more direct manner, e.g. by counting the number of times that a goal has been reached.

The most similar approach that also uses artificial curiosity for contextual policy search has been presented by Baranes et. al.: In [3], they present the SAGG-RIAC framework, which is an artificial curiosity framework that learns where to sample contexts in an episodic setting. SAGG-RIAC divides the context-space into hyper-cubical regions from which the contexts are then sampled based on a strategy similar to ϵ-greedy. SAGG-RIAC is assigning an interest-value to each region that determine how often a context is sampled from that region. This interest is based on the amount of growth that the empirically collected reward-samples have seen in this particular region and is defined as the difference between the average reward of the last $\frac{\zeta}{2}$ samples and the $\frac{\zeta}{2}$ samples before these

$$I(G_k) = \frac{\sum_{j=|G_k|-\zeta}^{|G_k|-\frac{\zeta}{2}} R_{k,j} - \sum_{j=|G_k|-\frac{\zeta}{2}}^{|G_k|} R_{k,j}}{\zeta}, \tag{2.1}$$

where G_k denotes a region and $R_{k,j;j<|G_k|}$ are the rewards collected in G_i in chronological order. ζ is a parameter that denotes the number of samples that are to be considered. This metric is approximating the difference between value functions (expected reward) of older iterations and value functions of newer iterations. Between two iterations i and $i+1$ this can written as $V^{(i+1)}(s) - V^{(i)}(s)$ which we will use in this thesis.

In [12], Peters et. al. present Relative Entropy Policy Search, a policy search algorithm with the core idea to bound the distance between two policies in order to ensure a stable learning progress and to avoid premature convergence. We will use REPS both, for learning the policy as well as as the basis for our own ActiveREPS algorithm which will learn the context-distribution. REPS is capable of generalizing over different observed contexts[8] which makes it suitable for our purposes. We will describe REPS in detail in the next chapter. For a comparison of different policy search algorithms see [5].

We intend to use the same idea behind the metric used in SAGG-RIAC while alleviating some of its problems. Specifically, SAGG-RIAC is using hypercubical regions with hard boundaries for obtaining new samples. However, different levels of complexity are unlikely to have hard boundaries in the context-space and can seldomly be mapped to hypercubical regions. As a result, we attempt to learn a single, non-parametric probability distribution instead of discrete probabilities for hypercubical regions. Furthermore, SAGG-RIAC is using distinct algorithms for learning the context-distribution and for learning the policy. Instead, we intend to use a more unified approach by learning the context-distribution using a variation of the same learning algorithm that is used for learning the policy.

In order to learn the context-distribution, we will incorporate this metric into a modified objective function and solve the learning problem analogously to the optimization process used in REPS. We will furthermore show that the solution that we derive from this adaption is implementing its curiosity aspect by incorporating an intrinsic reward function into standard REPS and is therefore in line other approaches to artificial curiosity.

For a more detailed discussion of different approaches to exploration and curiosity see [11].

In Section , we introduce LocalREPS. LocalREPS is a modification of REPS that only aims to be correct in a small region of the context-space. In the past, locally weighted learning has often been used for supervised learning, for example local regression. Different methods used for supervised learning are discussed in [1]. Atkeson et. al. show that locally weighted learning can be used for model-based reinforcement learning[2]. LocalREPS combines locally weighted learning with model-free policy search using weightings based on a distance metric similar to the aforementioned methods.

3 Relative Entropy Policy Search

3.1 Policy Search

Reinforcement Learning aims to learn how to solve tasks solely by interacting with the environment without prior, task-specific knowledge. These tasks are often represented as a Markov Decision Process where the agent tries to navigate a space of states S using actions of an action space A based on a reward function $R : S \times A \to \mathbb{R}$ which denotes how good a certain state is as well as how good the action was that lead to that state. In robotics, the state-space S and the action-space A are typically continuous. However, in a contextual episodic learning setup, there is no state that could be modified by actions. Instead, the state space is replaced by a space of contexts S. Furthermore, A contains whole trajectories as an episode has to be solved without using multiple transitions.

Reinforcement learning then strives to find a deterministic policy $a = \pi(s)$ or a probabilistic policy $\pi(a|s)$ that solves the task by choosing actions that maximize the reward. One particular class of reinforcement learning algorithms is policy search. Policy search attempts to find a policy using optimization. A specific algorithm in this class is Relative Entropy Policy Search which we will explain in the following section.

3.2 Relative Entropy Policy Search

In [12], Peters et. al. present Relative Entropy Policy Search (REPS). REPS is a probabilistic policy search algorithm that strives to iteratively find an optimal policy by maximizing the expected reward for a reward function $r(\mathbf{s}, \mathbf{a})$. The expected reward is defined as

$$J(\pi) = \sum_{\mathbf{s},\mathbf{a}} \mu^{\pi}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})r(\mathbf{s}, \mathbf{a}), \tag{3.1}$$

where $\mu^{\pi}(\mathbf{s})$ is the distribution of contexts. The core idea of REPS is to limit exploration of actions and contexts in order to stay close to the observed data of the last iteration. This keeps the algorithm from converging prematurely based on assumptions on unseen data and is achieved by putting a bound ϵ on the KL-divergence between the joint-distribution $p(\mathbf{s}, \mathbf{a}) = \mu^{\pi}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})$ that the optimization problem tries to find and the joint-distribution $q(\mathbf{s}, \mathbf{a})$ that has been found in the last iteration

$$\epsilon \geq \sum_{\mathbf{s},\mathbf{a}} \mu^{\pi}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s}) \log \frac{\mu^{\pi}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})}{q(\mathbf{s}, \mathbf{a})}. \tag{3.2}$$

As a result, the joint distribution of contexts and actions between two iterations cannot differ greatly. Furthermore, since we are working in a contextual setup and since we cannot freely choose the context distribution μ^{π}, it is necessary to fix the context-distribution to the observations. Since observations are only possible at samples, we cannot obtain the exact distribution. Instead, REPS matches the average of chosen features of the observed contexts with the average features of the desired context-distributions by adding a constraint

$$\sum_{\mathbf{s}} \mu^{\pi}(\mathbf{s})\Phi(\mathbf{s}) = \hat{\Phi}, \tag{3.3}$$

where Φ is the feature function and $\hat{\Phi}$ is the feature average of the observed contexts. For example, a possible feature function is $\Phi(\mathbf{s}) = (\mathbf{s}, \mathbf{s}^2)^T$, i.e. a feature function that matches mean and variance of both distributions.

As a consequence, the full optimization problem of Relative Entropy Policy Search reads as follows:

$$\max_{\pi,\mu^\pi} J(\pi) = \sum_{\mathbf{s},\mathbf{a}} \mu^\pi \pi(\mathbf{a}|\mathbf{s}) r(\mathbf{s},\mathbf{a}), \tag{3.4}$$

$$s.t. \epsilon \geq \sum_{\mathbf{s},\mathbf{a}} \mu^\pi(\mathbf{s})\pi(\mathbf{a}|\mathbf{s}) \log \frac{\mu^\pi(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})}{q(\mathbf{s},\mathbf{a})}, \tag{3.5}$$

$$\sum_{\mathbf{s}} \mu^\pi(\mathbf{s})\Phi(\mathbf{s}) = \hat{\Phi}, \tag{3.6}$$

$$1 = \sum_{\mathbf{s},\mathbf{a}} \mu^\pi(\mathbf{s})\pi(\mathbf{a}|\mathbf{s}). \tag{3.7}$$

This problem can be solved using the method of Lagrangian multipliers. Peters et. al. show that the new joint distributions can be obtained by putting weights on the old joint distribution where these weights depend on the rewards as well as the Lagrangian multipliers. The formula is

$$\mu^\pi(\mathbf{s})\pi(\mathbf{s}|\mathbf{a}) \propto q(\mathbf{s},\mathbf{a}) \exp\left(\frac{r(\mathbf{s},\mathbf{a}) - \theta^T\Phi(\mathbf{s})}{\eta}\right), \tag{3.8}$$

where η and θ are Lagrangian parameters. Peters et. al. furthermore show that $\theta^T\Phi(\mathbf{s})$ is an approximation of the value function $V^\pi(\mathbf{s}) + c$ for some c[12]. This approximated value function serves as a baseline and ensures that the weightings depend only on the quality of the actions and not on the quality of the contexts. This is also of particular interest when comparing REPS to ActiveREPS in Section 4. Without the constraint on the context-distribution, this term vanishes which we will also make use of in Section 4.

If we were to derive a policy directly from this solution we would need to evaluate the reward at every possible context-action pair $(\mathbf{s},\mathbf{a})$ which is impossible. However, we can obtain a usable policy using only samples by approximating it based on samples. This can be achieved by fitting a parametric policy $\pi'(\mathbf{a}|\mathbf{s},\omega)$ with parameters ω, e.g. a Gaussian linear model, so that it is as close to the real distribution as possible. In order to do this, we can minimize the KL-divergence of the parametric policy and the optimal policy. It can be shown that this can be calculated using equation 3.8 and a fixed amount of action-samples $\mathbf{a}^{(i)}$ at contexts $\mathbf{s}^{(i)}$:

$$\text{argmin}_\omega \text{KL}(\pi(\mathbf{a}|\mathbf{s})|\pi'(\mathbf{a}|\mathbf{s},\omega)) \tag{3.9}$$

$$= \text{argmin}_\omega \int \pi(\mathbf{a}|\mathbf{s}) \log \frac{\pi(\mathbf{a}|\mathbf{s})}{\pi'(\mathbf{a}|\mathbf{s},\omega)} d\mathbf{a} \tag{3.10}$$

$$= \text{argmin}_\omega \int\int \mu^\pi(\mathbf{s})\pi(\mathbf{a}|\mathbf{s}) \log \frac{\pi(\mathbf{a}|\mathbf{s})}{\pi'(\mathbf{a}|\mathbf{s},\omega)} d\mathbf{a} d\mathbf{s} \tag{3.11}$$

$$\approx \text{argmax}_\omega \sum_i \frac{\mu^\pi(\mathbf{s}^{(i)})\pi(\mathbf{a}^{(i)}|\mathbf{s}^{(i)})}{q(\mathbf{a}^{(i)},\mathbf{s}^{(i)}|\omega)} \log \pi'(\mathbf{a}^{(i)}|\mathbf{s}^{(i)},\omega), \tag{3.12}$$

where $w_i := \frac{p(\mathbf{s}^{(i)})\pi(\mathbf{a}^{(i)}|\mathbf{s}^{(i)})}{\pi'(\mathbf{a}^{(i)}|\mathbf{s}^{(i)},\omega)} \propto \exp \frac{r(\mathbf{s}^{(i)},\mathbf{a}^{(i)}) - \theta^T\Phi(\mathbf{s}^{(i)})}{\eta}$ can be derived by Eq. 3.8. This results in the weighted maximum likelihood problem

$$\omega^* = \text{argmax}_\omega \exp \frac{r(\mathbf{s}^{(i)},\mathbf{a}^{(i)}) - \theta^T\Phi(\mathbf{s}^{(i)})}{\eta} \log \pi'(\mathbf{a}^{(i)}|\mathbf{s},\omega), \tag{3.13}$$

which can be solved with respect to the parametric policies $\pi'(\mathbf{a}|\mathbf{s},\omega)$. For example, a linear Gaussian policy $\pi'(\mathbf{a}|\mathbf{s}) = N(\mu_\omega,\Sigma_\omega)$ can be used, where the mean depends linearly on the context. μ_ω and Σ_ω can be calculated by standard weighted linear regression using a weight-matrix $W = diag(w_1,w_2,...,w_n)$ based on the previously defined weights. The mean can then be calculated as $\mu_\omega = K\Phi(\mathbf{s}) + k$ where $\begin{pmatrix} k^T \\ K^T \end{pmatrix} = (\Phi(\mathbf{s})^T W \Phi(\mathbf{s}))^{-1}\Phi W\mathbf{a}$ and $\Sigma_\omega = \frac{\sum_i w_i(\mathbf{a}^{(i)} - \mu_\omega)^T(\mathbf{a}^{(i)} - \mu)}{\sum_i w_i}$.

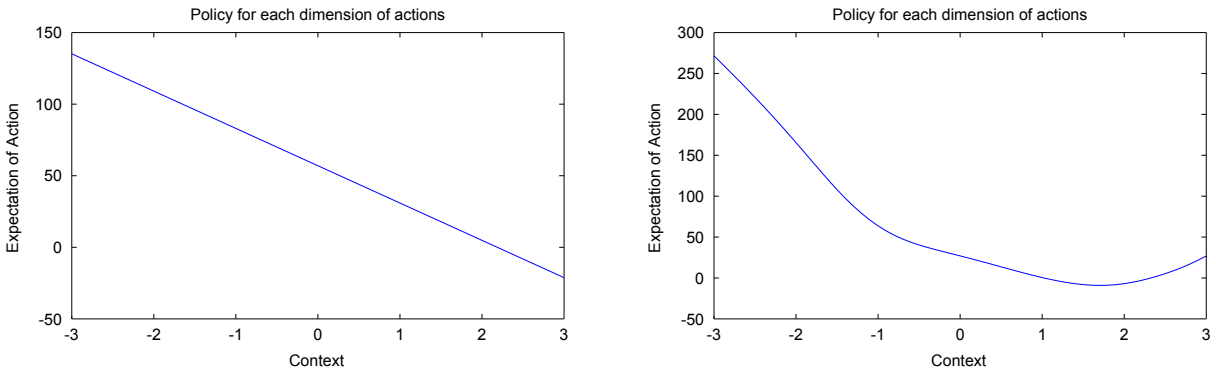


Figure 3.1: Examples for the expectation of policies(y-axis) dependent on context(x-axis). **a)** A Gaussian linear policy b) A Gaussian locally linear policy learned by LocalREPS

3.3 Local Relative Entropy Policy Search

In this thesis we are presenting a method to train the robot on areas of the context-space where the robot improves best. Naturally, this is more interesting on tasks where the context has a large influence on the learning process of the robot. However, this kind of task is also particularly difficult for REPS to learn. In order to retrieve a usable policy based on samples, REPS requires a parametric model. Typically, we are using models that depend linearly on the context, e.g. Gaussian linear models. This kind of model only works well if the optimal actions depend approximately linearly on the context, i.e. on tasks where the context-space has a particularly small influence on the learning process of the robot. As a result, we expect the performance on many of the tasks where ActiveREPS would yield improvements to the overall performance to be limited by the model that REPS is using. Furthermore, REPS only matches the average features of the internally used context-distribution and the observed samples. A complex context-space would require to choose a complex enough feature function as well.

In order to learn policies on complex context-spaces we therefore introduce Local Relative Entropy Policy Search. LocalREPS is an adaptation of REPS that is trained for each context locally around that context. The distance metric that we are using for LocalREPS is RBF-Kernels

$$k(\mathbf{s}, \mathbf{s_0}) = \exp -\frac{||\mathbf{s} - \mathbf{s_0}||^2}{2\sigma}. \tag{3.14}$$

For training LocalREPS at a context $\mathbf{s_0}$, we are therefore bounding the KL-divergence between the old joint distribution $q(\mathbf{s}, \mathbf{a})$ and the new joint distribution $\mu^\pi(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})$ only locally. This can be achieved by modifying the corresponding constraint on REPS' objective function so that the old joint distribution is multiplied with the RBF-Kernel

$$\epsilon \geq \sum_{\mathbf{s},\mathbf{a}} \mu^\pi(\mathbf{s})\pi(\mathbf{a}|\mathbf{s}) \log \frac{\mu^\pi(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})}{k(\mathbf{s}, \mathbf{s_0})q(\mathbf{s}, \mathbf{a})}. \tag{3.15}$$

Furthermore, we are matching the features only locally by matching the feature average with the weighted feature average of the observations $\hat{\Phi}_{\mathbf{s_0}}$. The observations are weighted by the distance as measured with the RBF-Kernel

$$\sum_{\mathbf{s}} \mu^\pi(\mathbf{s})\Phi(\mathbf{s}) = \hat{\Phi}_{\mathbf{s_0}}. \tag{3.16}$$

This results in the proportionality equation

$$\mu^\pi(\mathbf{s})\pi(\mathbf{s}|\mathbf{a}) \propto k(\mathbf{s}, \mathbf{s_0})q(\mathbf{s}, \mathbf{s_0}) \exp \frac{r(\mathbf{s}, \mathbf{a}) - \theta^T \Phi(\mathbf{s})}{\eta}, \tag{3.17}$$

which is a modified version of Equation 3.8. In order to train the parametric policy $\pi'(\mathbf{a}|\mathbf{s}, \omega)$ on samples we therefore need to multiply each weighting with its RBF-Kernel activation so that

$$\omega^* = \text{argmax}_\omega k(\mathbf{s}, \mathbf{s_0}) \exp \frac{r(\mathbf{s}^{(i)}, \mathbf{a}^{(i)}) - \theta^T \Phi(\mathbf{s}^{(i)})}{\eta} \log \pi'(\mathbf{a}^{(i)}|\mathbf{s}, \omega). \tag{3.18}$$

Using LocalREPS allows us to learn more complex policies as can be seen in Figure 3.1.

4 Active Relative Entropy Policy Search

4.1 Active Relative Entropy Policy Search

Contextual REPS as it was described in the last chapter is able to generalize over contexts but gathers the distribution of these contexts from observations and is therefore unable to take advantage of situations where we can choose a context distribution. In this paper we propose Active Relative Entropy Search, an adaptation of REPS that aims to learn the context distribution instead of the policy in order to train on contexts where the learning algorithm can improve the expected return the most. Since REPS already maximizes for both, the policy and the context-distribution, we mostly need to adapt the objective function. Instead of maximizing the expected reward, our intention is to maximize the expected interest, i.e. a metric that is high for contexts that are deemed "interesting" according to a specific interest metric. The interest metric that we are using for our approach is similar to the one used in SAGG-RIAC and is measuring the change of the expected reward (value function) for a given context. This can be written as

$$I(\mathbf{s}) = V^{(i+1)}(\mathbf{s}) - V^{(i)}(\mathbf{s}). \tag{4.1}$$

Similar to the interest function used by SAGG-RIACs, this interest function aims to choose contexts where the underlying policy-learning-algorithm is able to improve its performance the most. The modified objective function can be written as

$$\max_{\pi,\mu^\pi} J(\pi,\mu^\pi) = \sum_{\mathbf{s}} \mu^\pi (V^\pi(\mathbf{s}) - V_{old})(\mathbf{s}), \tag{4.2}$$

where V_{old} denotes the value function belonging to the old policy, i.e. the value function of the previous iteration.

As we want μ^π to be learnable, we need to relax the constraint given in Eq. 3.3 to allow REPS to learn the parts of the context that are learnable instead of fixing them to observations. The modified constraint looks as follows

$$\sum_{\mathbf{s}} \mu^\pi(\mathbf{s}) \Phi_{\mathbf{s}_{fix}} = \hat{\Phi_{fix}} \text{ with } \mathbf{s} = (\mathbf{s}_{fix}, \mathbf{s}_{free})^T, \tag{4.3}$$

where we split the context in two parts. $\mathbf{s}_{fix}$ is the part of the context that cannot be chosen whereas $\mathbf{s}_{free}$ is the part of the context that we can choose. For example, this situation could occur in a modification of the aforementioned table-tennis task where the incoming balls are thrown by a human instead of by a controllable ball cannon. In this scenario, the robot cannot choose the parameters of the incoming balls and as a result, they would be part of $\mathbf{s}_{fix}$. However, the robot would still be able to choose its target which makes it part of $\mathbf{s}_{free}$.

By expanding the value function and using a few transformations, this objective function can be reduced to a function similar to the objective function of the original REPS-algorithm

$$\begin{aligned} J(\pi,\mu^\pi) &= \sum_{\mathbf{s}} \mu^\pi (V^\pi(\mathbf{s}) - V_{old}(\mathbf{s})) && (4.4)\\ &= \sum_{\mathbf{s}} \mu^\pi \left(\sum_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) r(\mathbf{s},\mathbf{a}) - V_{old}(\mathbf{s}) \sum_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \right) && (4.5)\\ &= \sum_{\mathbf{s},\mathbf{a}} \mu^\pi(\mathbf{s}) \pi(\mathbf{a}|\mathbf{s}) (r(\mathbf{s},\mathbf{a}) - V_{old}(\mathbf{s})). && (4.6) \end{aligned}$$

The objective function is therefore equivalent to the one of regular REPS with a modified reward-function $r_{mod}(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) - V_{old}(\mathbf{s})$. Employing the regular (extrinsic) reward function would lead to contexts being chosen by the original

objective function without baseline, i.e. maximizing the expected reward. It follows that the curiosity aspect of this algorithm lies in the intrinsic reward contained in the modified reward function which is similar to the algorithms described in Section 2. The result of the modified REPS optimization problem is now given as

$$\mu^{\pi}(\mathbf{s})\pi(\mathbf{s}) \propto q(\mathbf{s},\mathbf{a}) \exp \frac{r_{mod}(\mathbf{s},\mathbf{a})}{\eta} = q(\mathbf{s},\mathbf{a}) \exp \frac{r(\mathbf{s},\mathbf{a}) - V_{old}(\mathbf{s})}{\eta}. \tag{4.7}$$

In order to evaluate this equation it is necessary to evaluate the value of the old value-function $V_{old}(\mathbf{s})$ at new contexts that haven't been used with the old policy. As V_{old} is not known, we have to learn the function based on the context-reward pairs that have been obtained in the last iteration. In order to learn V_{old}, we are using a Gaussian Process. Our experiments show that it is best to choose the hyper-parameters of the Gaussian Process based on Cross Validation as described in [13] instead of using Empirical Maximum Likelihood.

The Gaussian Process allows us to evaluate the modified reward function and to obtain a parametric context-distribution analogously to obtaining the parametric policy in standard REPS using weighted maximum likelihood. However, since our goal is to represent complex context-distributions, it is advisable to use a non-parametric model for the context-distribution as it is flexible enough to represent such complex distributions, for example Kernel Density Estimation.

4.2 Using Active Relative Entropy Policy Search for learning the context distribution

ActiveREPS can be used in two ways. The first way is to use it in order to learn a context-distribution and then to use this distribution for sampling the context while using another policy search algorithm such as regular REPS in order to learn a policy $\pi(\mathbf{a}|\mathbf{s})$. This method has the advantage that the model and the parameters of the policy-learning algorithm, such as the relative entropy bound ϵ, can be fine-tuned independently for learning the policy and the context-distribution.

The second possibility is to use ActiveREPS to learn both, a context-distribution as well as a policy. We have shown that ActiveREPS only introduces a different baseline to the REPS learning problem. As a consequence, ActiveREPS is still optimizing the policy as well and can be used to to learn one. This results in a simpler solution with a single learning-algorithm but requires the learning-algorithm and its parameters to be suitable for learning both distributions.

4.3 Managing the Training Set

Training regular REPS requires a set of state-action-reward tuples, sampled from the last policy. However, in order to reduce the amount of samples that are being collected, we can reuse samples from the previous iterations. For ActiveREPS, a more sophisticated strategy is necessary. Since the context-distribution might have most of its mass on certain regions, samples from other regions of the context-space become more unlikely. Hence, when the algorithm deletes old samples, it is forgetting all knowledge about unlikely areas of the context-space. Retaining older samples from all regions retains the support in these regions.

In order to retain old samples from old regions we are replacing the global bound based on the number of samples with a bound on the local activation of RBF-kernels. Specifically, when the training set consists of N chronologically ordered samples with contexts $\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_N$, a sample with a context $\mathbf{s}_i$ is being deleted if

$$K > \sum_{k=i+1}^{j} \exp -\frac{||\mathbf{s}_i - \mathbf{s}_k||}{2\sigma_d} \tag{4.8}$$

where K is the bound on the local activation and σ_d is a parameter denoting the bandwidth of the used kernel. Following this strategy leads to samples being deleted if it is the oldest sample in a region where the kernel-activation exceeds a set threshold. This can be seen as keeping a set of the newest K samples locally as opposed to the global limit used for regular REPS.

5 Evaluation

We have conducted two experiments to evaluate ActiveREPS as well as LocalREPS. The first experiment is a simulated table tennis task that is being solved with a combination of reinforcement and demonstration learning. We will show that in this experiment, ActiveREPS leads to a better performance of the robot after the training has been completed. The second experiment is a two dimensional simulation where the robot has to reach to a certain point. We will show that in this experiment, LocalREPS leads to a large improvement over regular REPS. We will furthermore utilize this experiment to discuss in which situations ActiveREPS does not lead to an improved performance when compared to regular (Local)REPS.

5.1 Table tennis

In the table tennis task a simulated robot arm has to return differently incoming balls to a specified position using a table tennis racket. The robot arm has 9 degrees of freedom and is mounted on rails that allow it to move horizontally in the x and y directions.

The goal of the robot is to first hit the ball with the racket and then to return it to a desired position. How the ball is shot at the robot is determined by the context of this particular learning problem. The context is 2-dimensional and consists of the initial velocities of the ball in the x and y direction whereas the velocity in the z direction is fixed.

5.1.1 Dynamic Movement Primitives

In order to reduce the dimensionality of actions, we employ Dynamic Movement Primitives(DMP)[6]. The DMPs are trained by a demonstration trajectory, in our case a forehand stroke. They are capable of adapting to different end positions of the trajectory while retaining the shape of the trajectory. Hence, we can use the end-positions of the DMPs as actions for the reinforcement learning scenario of the movement instead of encoding whole trajectories which would be much harder. DMPs are dynamical systems which track a goal attractor g by following a shape that is defined by a forcing function $f_{\mathbf{w}}(z)$ where z is a phase variable that represents time

$$\ddot{y} = \tau^2\alpha(\beta(g-y) - \frac{\dot{y}}{\tau}) + \tau^2 f_{\mathbf{w}}(z), \tag{5.1}$$

$$\dot{z} = -\tau\alpha_z z, \tag{5.2}$$

where α and β are fixed parameters. In order to learn the shape of the DMP by demonstration we need to learn the forcing function $f_{\mathbf{w}}(z)$. This is done by linear ridge regression using a fixed amount of features $\psi(z)$ based on radial basis functions

$$f_{\mathbf{w}}(z) = \psi^T(z)\mathbf{w}. \tag{5.3}$$

Learning the parameters $\mathbf{w}$ allows us to learn the shape of the trajectory by demonstration so that we only need reinforcement learning to adapt the end position g. However, as the shape of the trajectory is fixed, we can only learn forehand strokes which may require a larger amount of movement to reach all incoming balls.

5.1.2 The Learning Setup

As baseline, we use standard REPS with a reward function based on the minimal distance from ball to racket as well as the distance from the ball to the goal position on the table

$$r(\mathbf{s}) = -\min(|s_{ball} - s_{racket}|) + I(c - |s_{ball,i} - s_{goal}|), \tag{5.4}$$

where s_{ball} is the trajectory of the ball, s_{racket} is the trajectory of the racket, $s_{ball,i}$ is the position of the ball when it hits the table and s_{goal} is the desired target position on the table. The variable c is a constant and I is an indicator that is 1 if the racket hit the ball and 0 otherwise. This reward function will give gradually higher rewards when the robot gets

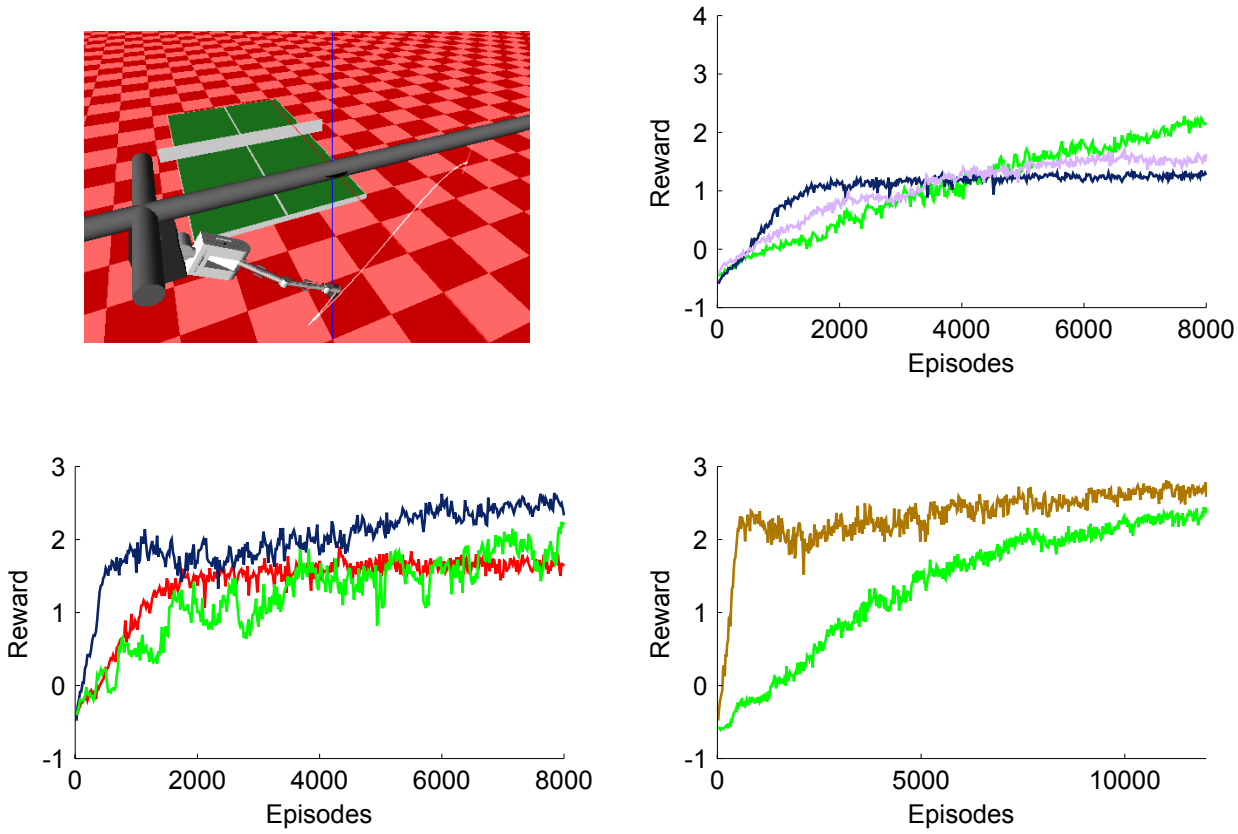


Figure 5.1: a) The simulated robot arm used in the table-tennis-environment. **b)** Learning curves of the table tennis task on fixed, uniformly sampled states. Regular REPS is shown in blue, REPS with SAGG-RIAC in pink, ActiveREPS in green. **c)** Learning curves of the table tennis task on the states sampled for training, colors as in b). **d** Evaluation of the reward on uniformly sampled, fixed states in green and the reward on training samples in brown

closer to hitting the ball. When the robot hits the ball, it gets gradually higher rewards when the ball is hitting the table closer to the target position. The fixed reward c that is given for hitting the ball encodes the maximum distance of the returned ball to the target position at which the trajectory is still considered to be better than one that barely misses to hit the ball.

We evaluated this task for 400 iterations using 20 samples for each iteration, while we also keep 400 samples of older iterations for training REPS. As mentioned in Section 3, the policy that is learned by REPS has to be approximated with a parametric policy using a weighted ML estimate; For this task, we are using a Gaussian linear model. This is sufficient since the end-position of DMPs generalizes well enough using a linear model. Furthermore, we are using a relative-entropy bound of $\epsilon = 2$. Furthermore, contextual REPS requires a feature function to be chosen. We use radial basis functions as features as we obtained the best results with this representation.

In this experiment, we compare this baseline to Active Relative Entropy Policy Search. We use ActiveREPS to learn both, the context-distribution and the policy simultaneously. In addition to the parametric model for the policy we also require a parametric model for the context-distribution. We have chosen Kernel Density Estimation as model for the context-distribution using hyperparameters acquired by 5-fold cross validation.

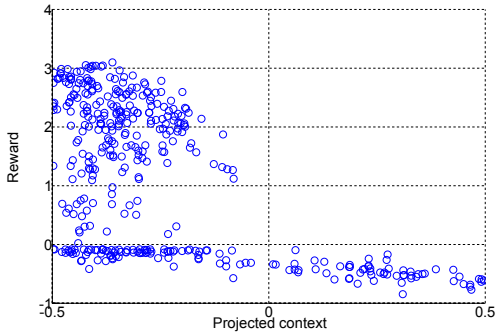
5.1.3 Results

We averaged each experiment over 8 trials and compute the performance by averaging the reward obtained after each iteration on 20, uniformly sampled contexts. We are comparing this to the performance on the contexts used for training which are sampled from the context-distribution that is learned by ActiveREPS and concentrates on the easier parts of the context-space first. The results show that ActiveREPS succeeds to learn the task faster than regular REPS on the

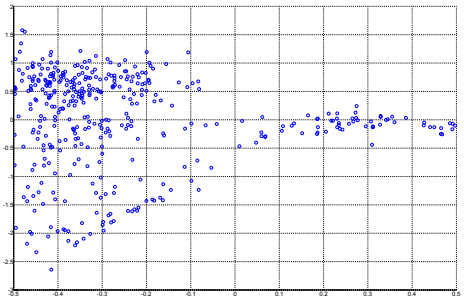
training-samples (Figure 5.1 c). However, on the fixed contexts the obtained reward is initially lower, because early on, the fastest progress can be obtained in a small region of the context space while the majority of the fixed contexts lie in different regions. Therefore, the performance of the robot at these points is not greatly influenced by the training samples which concentrate where the fastest progress can be obtained. However, in later iterations, ActiveREPS samples in different regions as can be seen in Figure 5.1 b) and the performance on the fixed states approaches the performance on the training-samples, and, thereby surpasses the performance of the baseline. One might expect the learning progress on the uniformly sampled states to be faster as well since we are optimizing the expected improvement over the complete context-space. However, since this can only be evaluated on the samples, the algorithm does not necessarily optimize the expected improvement outside of the areas of the context-space where samples have been obtained.

Figure 5.2 shows the sampling process of contexts during learning. We can see that after 50 iterations, the contexts concentrate on a particular region in the context space with a few outliers. After 100 iterations, we can already see another region emerging. In subsequent iterations, we see that contexts have been sampled in almost all parts of the context space. We can also see that very few contexts have been sampled in the center of the context-space during the first 400 iterations, which makes sense as the samples on the border are most difficult to learn. They are also the most informative to learn as the policy depends only linearly on the context. It is noteworthy that the reward function gives a large penalty for not hitting the ball, therefore, learning to hit the ball robustly results in the largest improvement of the average reward. Figure 5.3 shows the learning process. At 50 iterations most samples are in a specific area of the context-space, however, the results in this area are still mixed(Figure 5.3 a) and many of the newer samples obtain higher rewards than the older samples in the training-set. Therefore, the modified reward function gives a high reward on these contexts (Figure 5.3 b). Consequently, REPS learns a context-distribution, which is high in a particular region (Figure 5.3 c) and therefore creates samples foremost in that particular region (Figure 5.3 d). After 50 more iterations, the rewards in this region are almost consistently good (Figure 5.3 e) and the context-distribution shifts to another region (Figure 5.3 f).

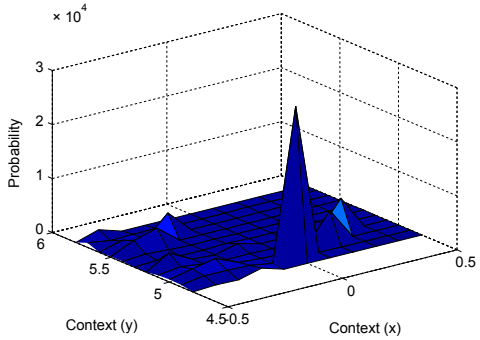
We furthermore compared ActiveREPS to SAGG-RIAC for sampling the contexts, using a region size of 150 and $\zeta - = 15$. In the results, we can still see the same tendencies of faster learning on training samples which results in a better learning progress on fixed samples. However, the observed effect isn't nearly as large as with ActiveREPS. The results of this comparison can also be seen in Figure 5.1 b) and c).



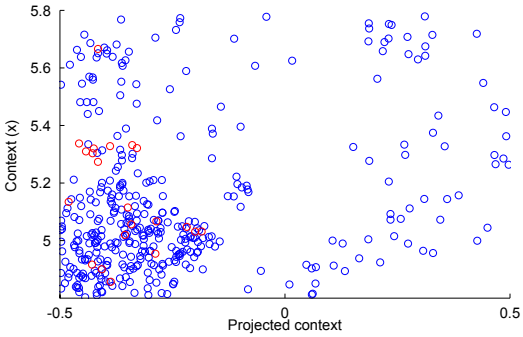
Untransformed(extrinsic) reward on training samples after 50 iterations (only x-dimension of context shown for readability)



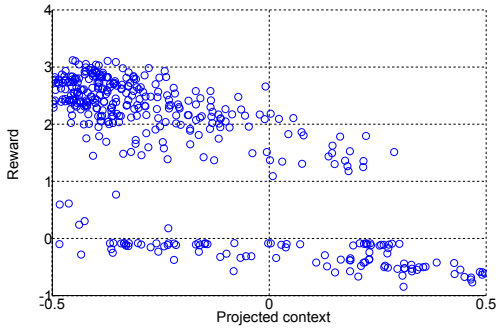
Transformed(in- & extrinsic) reward on training samples after 50 iterations (only x-dimension of context shown for readability)



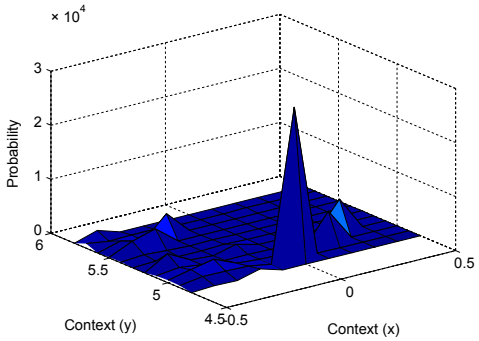
Learned context-distribution at iteration 50



Training samples after iteration 50 (red are newly sampled in that iteration



Untransformed(extrinsic) reward on training samples after 100 iterations (only x-dimension of context shown for readability)



Learned context-distribution at iteration 100

Figure 5.2: The context-learning process. After 50 iterations most samples are in a particular region, the rewards in this region still differ greatly (a) which, after subtracting the old value function, leads to a modified reward (r_{mod}) that is still high in this region (b). As a result, the context-distribution has most mass in this region (c) which leads to new samples being sampled there (d). After 50 more iterations the rewards gathered in this region are more homogeneous (e) which leads to a shift in the context-distribution (f).

i

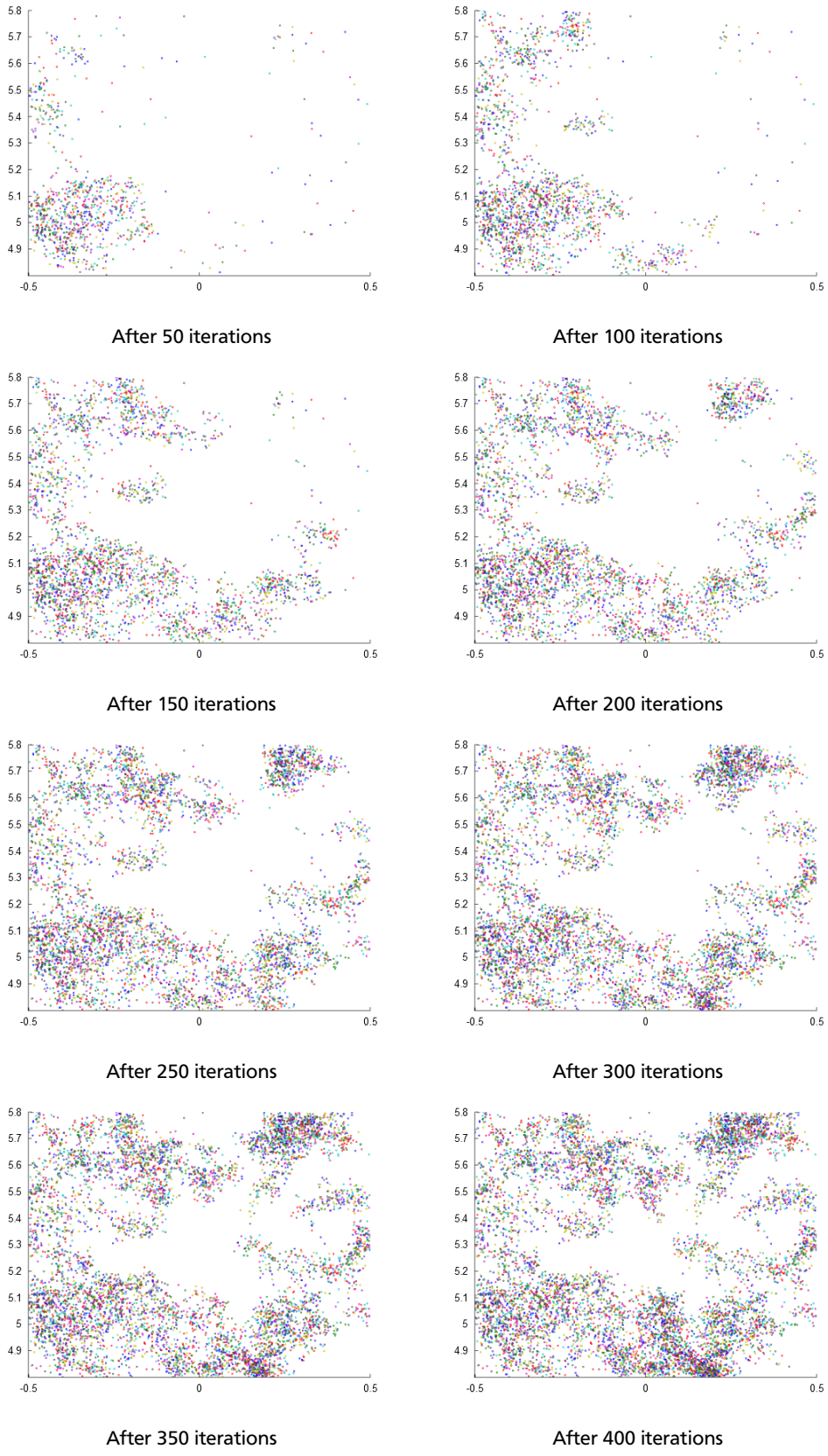


Figure 5.3: All contexts sampled during training on the table tennis task. Each point represents one sample, the axes describe both dimensions of the context. New contexts are mostly sampled from a narrow area that shifts once the performance on the previous area is good enough. After 50 iterations almost all samples come from a small region in the context space, after 400 iterations most of the space is covered.

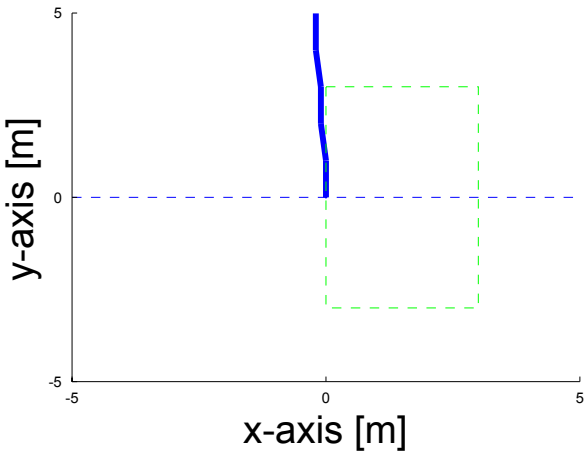


Figure 5.4: Simulated 5-link robot(blue) and the goal-space(green)

5.2 5-Link Reaching Task

For our second evaluation, we simulated a planar robot arm with 5 links (Figure 5.4). The task of the robot is to reach a certain point in the 2d-plane with its end effector, where the target position is determined by the context. To be more specific, the context space consists of the area between $(0, -3)$ and $(3, 3)$. The robot has 5 links with a length of 1 and its base is placed at the position $(0, 0)$. The difficulty in this task in comparison to the previous evaluation comes from the lack of demonstrations. The actions in this setup consist of the weightings for the DMP, i.e. factors for radial basis functions that form a forcing function and determine the shape of the DMP. As a result, the dimensionality of the action-space is considerably higher and linear generalization is more difficult as a consequence. Especially in the part of the context-space that lies close to the base of the robot.

5.2.1 The Learning Setup

Similar to the table tennis task, we first evaluated the task with regular REPS and then compared it with ActiveREPS. However, as we will show later, regular REPS cannot learn this problem well enough which is why we also evaluate LocalREPS. In order to train the learning algorithm, we use a reward function that is based on the distance of the end effector to the goal at a certain point in time as well as a penalty based on the actions:

$$r(\mathbf{s}, \mathbf{a}) = -h\mathbf{a}^T\mathbf{a} - (s_{ee,i} - s_{goal})^2 \tag{5.5}$$

where $s_{ee,i}$ is the position of the effector at the relevant time step, s_{goal} is the position of the goal and h is a parameter that determines how much influence the actions have on the reward.

We used a relative entropy bound of $\epsilon = 0.8$. As a feature function, we are using $\Phi(\mathbf{x}) = (\mathbf{x}, \mathbf{x}^2)^T$ (moment matching) for LocalREPS and a feature function based on RBF-Kernels for regular REPS as this one needs to handle a larger and more heterogeneous space of contexts. Furthermore, we set the bandwidth of the locality weighting used in LocalREPS to 1.5. Finally, a Gaussian model for policies isn't sufficient(see fig. 5.5 b) since REPS finds multiple solutions for part of the context-space (the robot arm can go left and right) over which a Gaussian policy would average. Instead, we are using a Gaussian mixture model with 10 mixture components which we train using 10 iterations of the EM-algorithm.

We compared the results of REPS and LocalREPS with a combination of ActiveREPS and LocalREPS. For this, we use ActiveREPS to learn the context-distribution while using a separate instance of LocalREPS to learn a policy using the contexts sampled from the learned context-distribution. This also shows the ability of ActiveREPS to work in combination with different algorithms for learning a policy.

For the parametrization of ActiveREPS, we are again using a Kernel Density Estimation to approximate the context distribution. However, we found that for this task a fixed bandwidth of 0.5 works generally better for the overall task than a bandwidth learned with Cross Validation. Furthermore, we are using the same bandwidth of 0.5 for the locality measure used for deleting old samples.

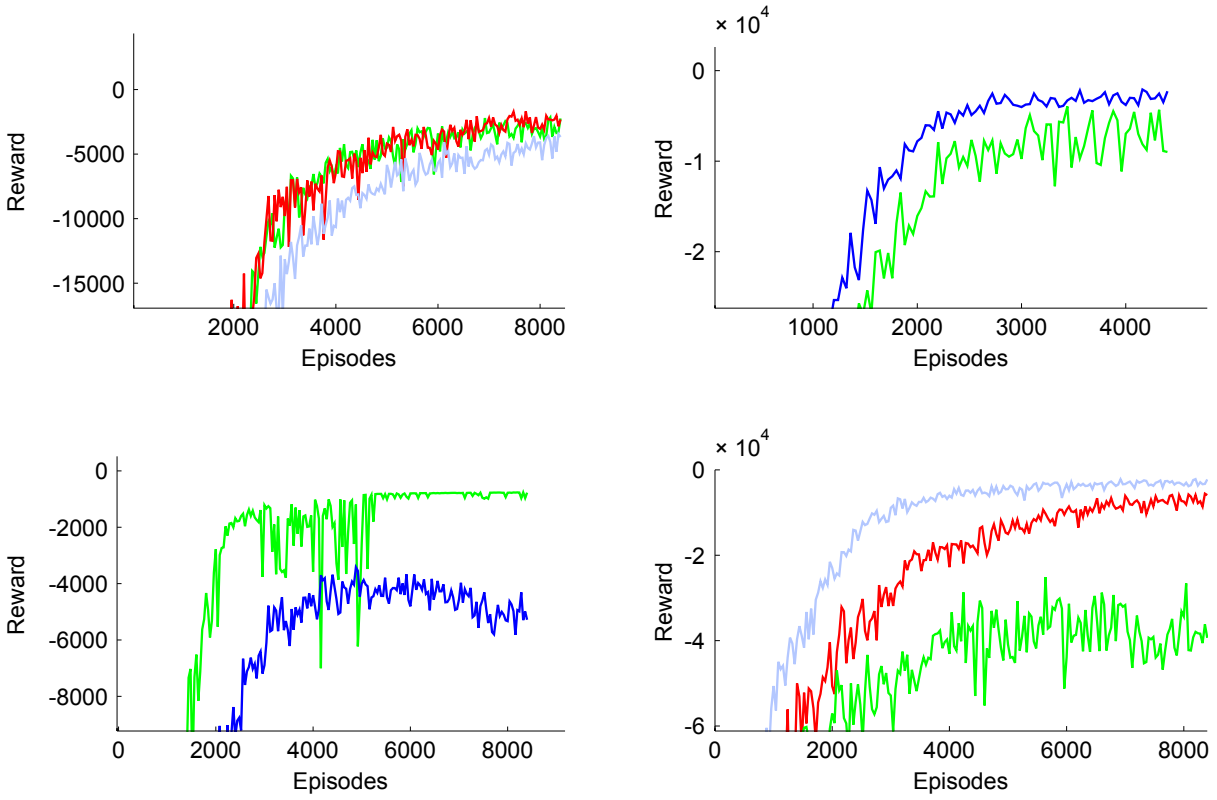


Figure 5.5: Reward on uniformly sampled, fixed states. **a)** 2D-context and mixture model. Regular REPS without ActiveREPS and with uniformly sampled contexts (blue) is outperformed by LocalREPS with uniformly sampled contexts (red). No further improvement by employing ActiveREPS (green) **b)** 1D-context and Gaussian model (no mixture model). LocalREPS with uniformly sampled contexts (blue) performs better than LocalREPS with contexts sampled from ActiveREPS (green). **c)** 1D-context and mixture model and uniformly sampled contexts. LocalREPS (green) greatly outperforms regular REPS (blue) **d)** LocalREPS with ActiveREPS and $\epsilon = 0.4$ outperforms regular LocalREPS without ActiveREPS and $\epsilon = 0.4$ by a large margin. However, LocalREPS with ActiveREPS and $\epsilon = 0.8$ is still better.

5.2.2 Results

Without employing ActiveREPS, we can see in Figure 5.5. a) that LocalREPS outperforms regular REPS by a significant margin. This is due to the non-linear dependency of the actions from the context as the actions consist of whole trajectories. Figure 5.5. c) shows that this margin gets even larger on a smaller context-space where the x-coordinate of the goal-position is always set to 1.5 since LocalREPS manages to improve further while regular REPS performs as poorly as on the large area. We also applied ActiveREPS, however, we could not see a significant improvement of the performance. By analyzing the generated contexts, we can see (Figure 5.6) that the samples are more uniformly distributed than in the table tennis task. The reason for that is the squared form of the reward function which makes early improvements on the task have more influence than improvements later in the learning progress. Subsequently, learning concentrates on a smaller part of the context-space much later than in the table tennis task. However, the policy does not improve anymore since REPS has already converged to a solution. We can change this behavior by manipulating the relative entropy bound. The results for that case can be seen in Figure 5.5. and show that ActiveREPS outperforms regular REPS. However, the change to ϵ also leads to a generally worse learning performance and is therefore undesirable. The evaluation of this task with a linear Gaussian model for approximating the policy (as well as a smaller, one-dimensional, subspace of the context-space) shows that ActiveREPS can even be harmful (Figure 5.5 b). This behavior results from ActiveREPS taking its measure of improvement directly from the rewards and does not take the limitations of the used model into account. As a result, ActiveREPS predicts large improvements in an area where the model cannot accommodate them and keeps sampling in this region.

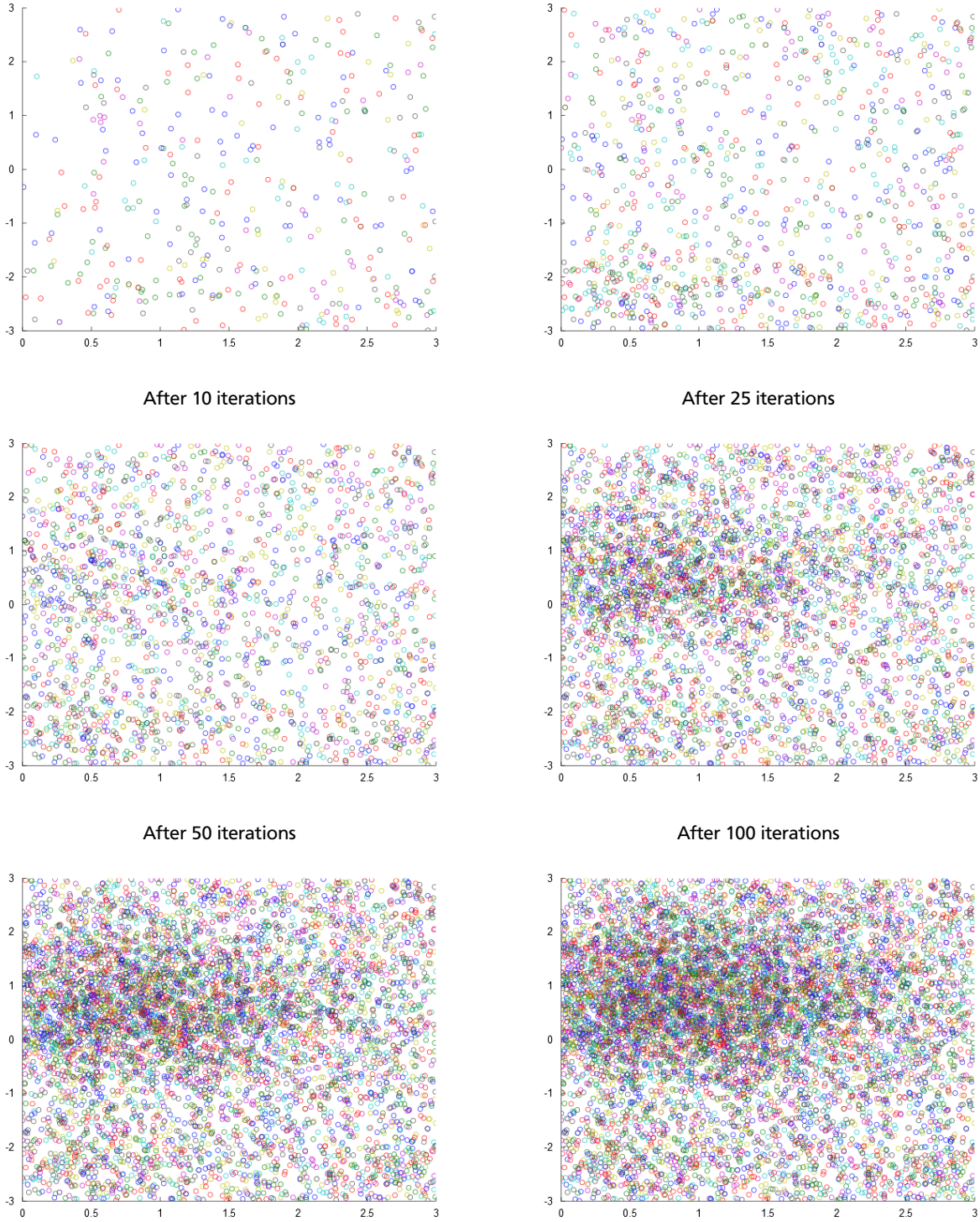


Figure 5.6: All contexts sampled during training on the reaching task. Each point represents one sample, the axes describe both dimensions of the context. New contexts are sampled much more uniformly than in Figure 5.2. New samples concentrate on a particular region of the context-space only after 100 iterations.

6 Conclusion

In this thesis, we introduced ActiveREPS, an algorithm that incorporates artificial curiosity into contextual, episodic REPS in order to guide the learning process by training on contexts where the overall performance can improve best. We have shown that it is possible to integrate the improvement of the learning performance directly into the objective function of REPS for learning a context-distribution. Furthermore, we have shown that this modification corresponds to including a intrinsic reward based on the learnable value function of past iterations into the reward function. We showed that, using ActiveREPS, we can improve the end-results of the learner where the extent of the improvement depends on the task as well as the chosen models.

Furthermore, we have introduced LocalREPS, a variation of REPS that is trained locally. We have shown that LocalREPS can improve the performance for tasks where the actions depend non-linearly on the context.

6.1 Future Work

In future work we will further analyze the behavior of ActiveREPS in combination with different parameterizations of REPS and the used models. Furthermore, we will analyze for which kind of tasks ActiveREPS yields the best improvements over regular REPS. We will use this knowledge in order to improve the performance on the presented tasks and to further expand the context-space. Secondly, we are going to evaluate the effect of incorporating exploration algorithms into ActiveREPS as the table tennis task shows that the context distribution can get very narrow. This makes it difficult to evaluate the improvement of the expected reward in other areas as there are very few newer samples. We hope that an exploration algorithm such as ϵ-greedy will alleviate this problem and lead to a globally faster learning progress. Moreover, we will compare the performance of different intrinsic reward functions in a model-free learning algorithm such as REPS and compare the results to our own approach. Finally, we want to apply the ActiveREPS algorithm in a non-episodic setup.

We are confident that ActiveREPS can be used to solve tasks that couldn't be solved in a satisfactory way using random samples on standard REPS.

Bibliography

[1] Christopher G Atkeson, Andrew W Moore, and Stefan Schaal. “Locally Weighted Learning”. In: *Artificial Intelligence Review* 11.1-5 (1997), pp. 11–73.

[2] Christopher G Atkeson, Andrew W Moore, and Stefan Schaal. “Locally Weighted Learning for Control”. In: *Artificial Intelligence Review* 11 (1997), pp. 75–113.

[3] Adrien Baranes, Pierre-yves Oudeyer Inria, and France January. “Active Learning of Inverse Models with Intrinsically Motivated Goal Exploration in Robots”. In: *Robotics and Autonomous Systems* 61.2012 (2013), pp. 49–73. arXiv:arXiv:1301.4862v1.

[4] RI Brafman and M Tennenholtz. “R-max-a general polynomial time algorithm for near-optimal reinforcement learning”. In: *Journal of Machine Learning Research 3:* 3 (2002), pp. 213–231.

[5] Marc Peter Deisenroth. “A Survey on Policy Search for Robotics”. In: *Foundations and Trends in Robotics* 2.1-2 (2011), pp. 1–142.

[6] AJ Ijspeert, Jun Nakanishi, and Stefan Schaal. “Learning attractor landscapes for learning motor primitives”. In: *Proceedings of Advances in Neural Information Processing Systems 15*. 2002, pp. 1547–1554.

[7] Michael Kearns and Satinder Singh. “Near Optimal Reinforcement Learning in Polynomial Time”. In: *Proceedings of the 15th International Conference on Machine Learning*. 1998, pp. 260–268.

[8] Jens Kober, Erhan Oztop, and Jan Peters. “Reinforcement learning to adjust robot movements to new situations”. In: *Proceedings of the International Joint Conference on Artificial Intelligence*. 2011.

[9] J. Zico Kolter and Andrew Y. Ng. “Near-Bayesian exploration in polynomial time”. In: *Proceedings of the 26th Annual International Conference on Machine Learning - ICML ’09*. New York, New York, USA: ACM Press, 2009, pp. 1–8.

[10] Manuel Lopes et al. “Exploration in Model-based Reinforcement Learning by Empirically Estimating Learning Progress.” In: *Proceedings of Advances in Neural Information Processing Systems*. i. 2012, pp. 1–9.

[11] Luis Montesano. “Active Learning for Autonomous Intelligent Agents :” 2014.

[12] Jan Peters and Yasemin Altun. “Relative Entropy Policy Search”. In: *Proceedings of the Twenty-Fourth National Conference on ArtiïňĄcial Intelligence (AAAI), Physically Grounded AI Track*. 2010.

[13] Carl Edward Rasmussen and Chris Williams. *Gaussian processes for machine learning*. Vol. 14. 2. Apr. 2006.

[14] Jürgen Schmidhuber. “A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers”. In: *Simulation of Adaptive Behavior: From Animals to Animats*. Vol. 1. 1991, pp. 222–227.

[15] Satinder Singh, AG Barto, and N Chentanez. “Intrinsically motivated reinforcement learning”. In: *Proceedings of Advances in Neural Information Processing Systems 17*. 2005, pp. 1281–1288.

[16] Satinder Singh, RL Lewis, and AG Barto. “Where do rewards come from”. In: *Proceedings of the Annual Conference of the Cognitive Science Society*. 2009, pp. 2601–2606.