Journal of Machine Learning Research 15 (2014) 3371-3399 Submitted 12/13; Revised 6/14; Published 10/14

# Active Contextual Policy Search

**Alexander Fabisch** AFABISCH@INFORMATIK.UNI-BREMEN.DE
**Jan Hendrik Metzen** JHM@INFORMATIK.UNI-BREMEN.DE
*Robotics Research Group, University Bremen*
*Robert-Hooke-Str. 1, D-28359 Bremen, Germany*

**Editor:** Laurent Orseau

## Abstract

We consider the problem of learning skills that are versatilely applicable. One popular approach for learning such skills is contextual policy search in which the individual tasks are represented as context vectors. We are interested in settings in which the agent is able to *actively* select the tasks that it examines during the learning process. We argue that there is a better way than selecting each task equally often because some tasks might be easier to learn at the beginning and the knowledge that the agent can extract from these tasks can be transferred to similar but more difficult tasks. The methods that we propose for addressing the task-selection problem model the learning process as a non-stationary multi-armed bandit problem with custom intrinsic reward heuristics so that the estimated learning progress will be maximized. This approach does neither make any assumptions about the underlying contextual policy search algorithm nor about the policy representation. We present empirical results on an artificial benchmark problem and a ball throwing problem with a simulated Mitsubishi PA-10 robot arm which show that active context selection can improve the learning of skills considerably.

**Keywords:** reinforcement learning, policy search, movement primitives, active learning, multi-task learning

## 1. Introduction

Artificial agents like robots are deployed in increasingly complex environments where they have to fulfill a range of different tasks. Hard-coding the full set of behaviors required by an agent in such an environment before deployment becomes increasingly difficult. An alternative approach is to provide the agent with means for learning of novel behavior. By this, the agent might acquire additional behaviors when required. However, learning every behavior from scratch is cumbersome and impractical for real robots, where it is important to learn novel behavior within a small amount of trials. Learning a set of versatilely applicable skills is one way to make an agent *competent* (White, 1959) in an environment, which means that the agent can efficiently learn behaviors for a multitude of tasks imposed on him by building on top of the set of skills.

One means for acquiring reusable skills is reinforcement learning (RL). In RL, a skill is represented by a policy $\pi$, which selects actions based on the current state.[1] One popular means for learning such a policy in a robotic setting is *policy search* (Deisenroth et al., 2013).

1. A policy can be stochastic, that is, define a conditional probability distribution over the actions given the state, or be deterministic, that is, a function that maps from states to actions.

In policy search, policies are parameterized by a parameter vector $\boldsymbol{\theta}$ and the objective is to find $\boldsymbol{\theta}$ such that the expected return of the policy $\pi_{\boldsymbol{\theta}}$ is maximized. A problem with this formulation is that it is not easily extended to settings in which different tasks are imposed onto the agent, which result in different returns for the same policy $\pi_{\boldsymbol{\theta}}$. For instance, when an agent tries to throw a ball to different target positions (the tasks), the same policy might obtain very different returns depending on whether the respective target object is hit or not.

To address such multi-task settings, we consider the problem of learning policies for contextual RL problems. That is, we assume that similar tasks are distinguished by context vectors $\boldsymbol{s} \in S \subseteq \mathbb{R}^{n_s}$; that is, contexts are described by $n_s$-dimensional real vectors. We use the terms task and context mostly interchangeable in this paper. Instead of searching directly for $\boldsymbol{\theta}$ in the space of control policies $\pi_{\boldsymbol{\theta}}$, one can introduce an *upper-level policy* $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$, which is parameterized by $\boldsymbol{\omega}$ and defines a probability distribution over the parameters $\boldsymbol{\theta}$ of the actual control policy. The expected return of an upper-level policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$ in a parameterized RL problem is defined as

$$J(\boldsymbol{\omega}) = \int_{\boldsymbol{s}} p(\boldsymbol{s}) \int_{\boldsymbol{\theta}} \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s}) \mathcal{R}_{\boldsymbol{s},\boldsymbol{\theta}} \, \mathrm{d}\boldsymbol{\theta} \, \mathrm{d}\boldsymbol{s},$$

where $\mathcal{R}_{\boldsymbol{s},\boldsymbol{\theta}}$ is the expected return of policy $\pi_{\boldsymbol{\theta}}$ in context $\boldsymbol{s}$ and $p(\boldsymbol{s})$ is the probability density function of the context distribution. Searching for parameters $\boldsymbol{\omega}$ that maximize $J(\boldsymbol{\omega})$ is denoted as *contextual policy search* (Deisenroth et al., 2013). We do not make any further restricting assumptions about the upper-level policy, the control policy or the policy search algorithm in this paper.

Different methods have been proposed for contextual policy search: one class of methods learns first policies $\pi_{\boldsymbol{\theta}}$ for a set of tasks separately and uses thereupon regression algorithms to infer a deterministic function which generalizes the learned policy parameters over the entire context space (da Silva et al., 2012; Metzen et al., 2013). A second class of methods learns an upper-level policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$ directly without separating learning in an RL and a regression part (Peters and Schaal, 2007; Kober et al., 2012; Kupcsik et al., 2013).

A restriction of both class of methods is that they typically assume that $p(\boldsymbol{s})$ cannot be controlled by the agent. This is appropriate when the environment or a user imposes tasks onto the agent externally. However, an agent might set its goals by itself or select tasks in which it would like to increase its performance autonomously. For instance, when learning target-oriented throwing (Wirkus et al., 2012), the agent might self-select certain target positions which it cannot hit yet reliably and where it would like to improve its performance. Moreover, there might be certain problem domains, for example grasping a cup in environments with obstacles and many different but similar task configurations, where some configurations might be harder to learn than others. For example, grasping a cup is more difficult when it is surrounded by other cups or when the handle is on the far side. One way to approach such situations is to start learning with easy tasks and progress to more and more complex tasks over time, where knowledge acquired in prior tasks is transferred and reused. Thus, by autonomously selecting learning tasks (contexts), for example based on intrinsic motivation (Barto et al., 2004), an agent might increase its competence in an environment in a self-controlled manner.

In this paper, we consider how an agent can actively select contexts to make the best progress in learning $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$, that is in increasing its competence. We propose considering this *active task-selection problem* as a non-stationary bandit problem and using algorithms like D-UCB (Kocsis and Szepesvári, 2006), which are suited for this setting, as task-selection heuristic. Moreover, we examine different intrinsic motivation heuristics for rewarding the agent for selecting contexts during the learning process which are considered to increase the learning progress, that is, to reduce the number of trials that are required to master a given contextual problem. This is important on real robots where the number of trials required for learning a behavior needs to be as small as possible to reduce wear and tear of the robot.

The paper is structured as follows: in Section 2, we present related work in the fields of skill learning, active learning, and multi-armed bandits. Thereupon, we formalize the problem and present the proposed method and the intrinsic reward heuristics for active contextual policy search in Section 3 and demonstrate the underlying model of the contextual learning process in Section 4. We compare the heuristics empirically on two benchmarks, the first being a toy problem in which advantages and disadvantages of different heuristics are systematically studied (see Section 5.1) and the second being a robotic control problem based on a simulated Mitsubishi PA-10 robot. In this setting, an agent learns to throw balls at targets on the ground with two different reward functions which results in the "grid problem" and the "dartboard problem" (see Section 5.2). We conclude and provide an outlook in Section 6.

## 2. Related Work

Our work is closely connected to two different fields of machine learning: learning general and broadly applicable skills with reinforcement learning or imitation learning and active learning for task selection. We summarize the related work of both fields in this section. In addition, we will briefly introduce the non-stationary multi-armed bandit problem which is related to our proposed method.

### 2.1 Skill Learning and Skill Transfer

In our setup, low-level policies are typically represented by dynamical movement primitives (DMPs; Ijspeert et al., 2013), although the proposed approaches are in no way restricted to this setting. DMPs encode arbitrarily shapeable, goal-directed trajectories. There are different variants of DMPs, which have in common that they use an internal time variable $z$ (called phase) which replaces explicit timing and allows arbitrary temporal scaling of the movement. Furthermore, the *transformation system* of a DMP is a spring-damper system that generates a goal-directed movement which is controlled by the phase variable $z$ and modified by a *forcing term* $f$. The shape can be adjusted by learning the weights $w_i$ of the forcing term through imitation or reinforcement learning. We use a variant of DMPs that has been developed by Mülling et al. (2011, 2013), which allows additionally specifying a target velocity at the end of the movement. In summary, we use a low-level policy

$$\boldsymbol{x}_{t+1} = \pi_{\boldsymbol{v},\boldsymbol{w}}(\boldsymbol{x}_t, t), \quad \boldsymbol{v} = (\boldsymbol{x}_0, \boldsymbol{g}, \dot{\boldsymbol{g}}, \tau)\,,$$

where $\boldsymbol{x}_t$ is the state (position, velocity, and acceleration) at time $t$, $\boldsymbol{w}$ are the weights of the forcing term and $\boldsymbol{v}$ are the following meta-parameters: $\boldsymbol{x}_0$ is the initial state, $\boldsymbol{g}$ is the final state, $\dot{\boldsymbol{g}}$ the desired final velocity, and $\tau$ is the duration of the movement. Note that this kind of policy defines a state trajectory and thus requires a low-level controller that generates the appropriate actions such that the state transition from $\boldsymbol{x}_t$ to $\boldsymbol{x}_{t+1}$ is performed, see Peters et al. (2012) for a discussion of this in a robotic setting.

The problem of learning broadly applicable skills has been approached from different perspectives. DMPs themselves have been designed to generalize over some meta-parameters such as the duration of the movement, the start position, and the final position. Additionally, some DMP variants are able to generalize over velocities at the end of the movement (Mülling et al., 2011, 2013). However, designing skills that generalize over more complex task parameterization is not straightforward. This is why machine learning techniques from the fields of imitation learning and reinforcement learning have been used to automatically infer upper-level policies.

Ude et al. (2010) used a form of self-imitation to generalize the task of throwing a ball at a desired target position. First, a number of throws with different policy parameterizations are generated and the final ball position is measured. That is, DMPs with different values of $\tau$, $\boldsymbol{g}$, $\boldsymbol{w}$, and release times of the ball have been executed. Then, a mixture of locally weighted regression (Cleveland and Devlin, 1988) and Gaussian process regression (Rasmussen and Williams, 2005) has been used to find mappings from the position that has been hit on the ground to the corresponding values of the meta-parameters that generated the throw. Thus, the problem has been reduced to a regression problem. The same approach has been used for reaching tasks and drumming. Kronander et al. (2011) proposed a similar method to play mini-golf. As underlying policy representation, stable estimators of dynamical systems (Khansari-Zadeh and Billard, 2011) have been used and another set of meta-parameters is learned with Gaussian process regression and Gaussian mixture regression. Both methods can be regarded as generalizing imitation learning algorithms.

The idea of da Silva et al. (2012) was to use the reinforcement learning algorithm *policy learning by weighting exploration with the returns* (PoWER; Kober and Peters, 2011) to generate a training set for the regression algorithm, which consists of nearly optimal parameters $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots\}$ for the low-level policies in different contexts $\{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots\}$. Thereupon, a regression algorithm is used to infer a deterministic policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$, the so-called *parameterized skill*, which generalizes over the entire context space. The authors considered the problem of dart throwing and observed that this domain has the additional challenge of discontinuities in the mapping from task parameters to meta-parameters of the policy. Such discontinuities are less likely when the examples have been generated by a human operator but appear especially if there are multiple solutions for a task with similar quality and if there are no constraints for the meta-parameters during reinforcement learning. To address this problem, the authors use Isomap (Tenenbaum et al., 2000) to extract manifolds in the meta-parameter space and learn different support vector regression models for these manifolds. An extension of parameterized skill denoted as *skill templates* has been proposed by Metzen et al. (2013), which learns not only the upper-level policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$ from the experience but also an estimate of the uncertainty in this generalization, which can be useful for subsequent adaptation of the policy.

An advantage of these approaches is that the upper level policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$, which maps from contexts to parameters $\boldsymbol{\theta}$ of the low-level policy can be an arbitrary function and hence, sophisticated regression methods can be used for generalization, which can deal for example with discontinuities. On the other hand, a functional relationship between context and control-policy parameters $\boldsymbol{\theta}$ as in a deterministic policy might not always be adequate; for instance, there might be multiple optima in the space of $\boldsymbol{\theta}$ for a single context (for example, think of fore- and backhand strokes). In addition, the approach requires to learn close-to-optimal parameters for the control policy in a context to generate just one training example for the regression algorithms. Consequently, all other experience collected during learning these close-to-optimal parameters is not being used for generalizing over the context space.

Several reinforcement-learning algorithms have been proposed that learn the upper-level policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$ without separating learning in an RL and a regression part (Peters and Schaal, 2007; Kober et al., 2012; Kupcsik et al., 2013). These approaches allow transferring experience between different contexts even if the behavior in these contexts is suboptimal. Thus, these approaches can make use of all collected experience and are thus typically more sample-efficient, that is, they can learn a close-to-optimal upper-level policy within less trials. Furthermore, the upper-level policy $\pi_{\boldsymbol{\omega}}$ can be stochastic, which means it can be implemented by a conditional probability distribution. This has the advantage that the agent's explorative behavior is explicitly modeled and can be adapted by the learning algorithm on the upper level. Furthermore, multiple optima in the space of the low-level policies can be represented by a multi-modal probability distribution on the upper level (Daniel et al., 2012).

Peters and Schaal (2007) applied reinforcement learning to learn contextual policies directly with reward weighted regression. Here, a stochastic upper-level policy similar to the mapping from task parameters to policy meta-parameters in the regression setting is learned directly. Kober et al. (2012) extended this to non-parametric policies in an approach denoted as cost-regularized kernel regression (CrKR). CrKR has been used to learn throwing movements as well as table tennis. Another reinforcement learning algorithm that can be used to learn the upper-level policy is contextual relative entropy policy search (C-REPS, Kupcsik et al., 2013). REPS is an information-theoretic approach to policy search, which aims in each iteration at maximizing the expected return of the new policy while bounding the Kullback-Leibler divergence between the old and new policy. Bounding this divergence enforces that the new policy is not too different from the old policy as large changes of the policy could for instance be dangerous in a robotic setting. C-REPS is an extension of this approach to the contextual policy search setting. We refer to Appendix A for more details on C-REPS.

The representation of the upper-level policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$ in these approaches is typically restricted: in CrKR, the upper level policy is modeled as a Gaussian process with the outputs being assumed to be independent (Deisenroth et al., 2013). C-REPS allows any policy which can be learned by weighted maximum likelihood. A frequently used variant is $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s}) = \mathcal{N}\left(\boldsymbol{\theta}|\boldsymbol{W}^T\varphi(\boldsymbol{s}), \boldsymbol{\Sigma}\right)$, where $\varphi(\boldsymbol{s})$ contains the linear and quadratic terms of the context vector $\boldsymbol{s}$, $\boldsymbol{W}$ is a weight matrix, and $\boldsymbol{\Sigma}$ the covariance of the stochastic policy.

A common assumption in contextual policy search is that the selection of the task in which the next trial will be performed is either not under the control of the agent or that

it does not matter in which order and in which frequency tasks are selected. In this paper, we study strategies for actively selecting the next task by means of active learning.

### 2.2 Active Learning and Artificial Curiosity

In machine learning it is typically desirable to require as few training examples as possible because it is often costly to acquire examples. For instance, in supervised learning it can be expensive to label data. Similarly, in reinforcement learning domains such as robotics, performing a trial is typically very costly. Hence, in general one would like to perform only those trials that maximize the learning progress. A research field that deals with selecting data or tasks from which one can learn the most is *active learning* (Settles, 2010). The goal of active learning is to label only data or examine only tasks that promise the greatest learning progress, that is: the most informative instances. Different query strategies to find the most informative instance are discussed by Settles (2010).

The idea to actively select tasks that accelerate the learning progress is related to *curriculum learning*: Bengio et al. (2009) and Gulcehre and Bengio (2013) assume that learning simple concepts first helps to learn more complex concepts that build upon those previously learned simple concepts in the context of supervised learning. We also try to exploit this hypothesis in our empirical evaluation. Another related work has been published by Ruvolo and Eaton (2013) in the context of lifelong multi-task learning, which compares several heuristics for actively selecting the task that will be learned. Among these heuristic are the following: select the task that maximizes the expected information gain (information maximization heuristic) and select the task on which the current model performs worst (diversity heuristic). Once a task is selected, all labels of data instances of this task are revealed to the agent.

It is not straightforward to transfer this approach to the reinforcement learning setting. A problem that occurs in reinforcement learning is that one does not get the correct solution, that is the optimal policy, of a queried task directly. Instead, one receives only an immediate reward and needs to optimize the solution by trial-and-error in order to maximize the long-term reward, which requires to select the same training tasks multiple times consecutively until a close-to-optimal policy is learned. This approach is followed by da Silva et al. (2014), where the parameterized skill approach discussed in Section 2.1 is extended to an active learning setting. For this, the authors propose a novel criterion for skill selection. In this criterion, the skill performance is modeled using Gaussian process regression with a spatiotemporal kernel which addresses the inherent non-stationarity of tracking the skill performance. Based on this estimate of the skill performance, the next task is chosen such that the maximum expected improvement in skill performance would be obtained if the outcome of learning this task is assumed to be an optimistic upper bound. A drawback of this approach is that it may not be the optimal (most sample-efficient) task-selection strategy to stick for many trials to the same task until a close-to-optimal policy for this task is learned. In this paper, we address how active task selection can be performed if a novel task is chosen after each trial which involves that typically no close-to-optimal policy has been learned in the last task.

A field related to active learning with applications in reinforcement learning is artificial curiosity (Oudeyer and Kaplan, 2004; Gottlieb et al., 2013). In particular, an architecture

called *self-adaptive goal generation - robust intelligent adaptive curiosity* (SAGG-RIAC) has been derived from the ideas of artificial curiosity by Baranes and Oudeyer (2013) and has been applied to learning of contextual problems. The goal self-generation and self-selection of SAGG-RIAC divides the context space into rectangular regions. These are split once the number of contexts that have been explored in these regions exceeds a threshold. For each region $R_i$, one can compute an interest value based on the derivative of the competence in that region

$$\text{interest}_i = \frac{1}{\zeta} \left| \left( \sum_{j=|R_i|-\zeta}^{|R_i|-\frac{\zeta}{2}} r(\boldsymbol{s}_j, \boldsymbol{\theta}_j) \right) - \left( \sum_{j=|R_i|-\frac{\zeta}{2}}^{|R_i|} r(\boldsymbol{s}_j, \boldsymbol{\theta}_j) \right) \right|,$$

where $|R_i|$ is the number of contexts that have been explored in region $R_i$, $\zeta$ is the size of a sliding window, and $r(\boldsymbol{s}_j, \boldsymbol{\theta}_j)$ is the return of a low-level policy with parameters $\boldsymbol{\theta}_j$ in context $\boldsymbol{s}_j$ (not to be confused with the expected return $\mathcal{R}_{\boldsymbol{s},\boldsymbol{\theta}} = \mathbb{E}\{r(\boldsymbol{s}, \boldsymbol{\theta})\}$). Essentially, this means regions with greater differences between recent and previous returns are considered to be more interesting. Hence, on the one hand SAGG-RIAC focuses on regions where the reward increases considerably over time. On the other hand, it also favors regions where the reward decreases. The intuition for this is that such a decrease might be due to a change in the environment in this region and, hence, it would make sense to explore this region more strongly. SAGG-RIAC selects goals either randomly from the whole context space or from a random region, where the probability of each region corresponds to its interest. For a selected region, the goal is either drawn from a uniform random distribution or from the vicinity of the context with the lowest previous return. Each of the three cases is again selected randomly with fixed probabilities.

One advantage of SAGG-RIAC is that it naturally copes with continuous context spaces. The method will most of the time concentrate on regions where it expects a high learning progress but does not converge to a single region of the context space. As an alternative to the SAGG-RIAC heuristic, we present an approach that is based on heuristic estimates of the learning progress and multi-armed bandit algorithms, which more naturally trades off exploration and exploitation of the noisy estimate of the learning progress. The corresponding algorithms will be discussed in the next section.

### 2.3 Non-Stationary Multi-Armed Bandit Problems

Multi-armed bandit problems (MABP: Robbins, 1952; Bubeck and Cesa-Bianchi, 2012) are situated between supervised and reinforcement learning. A MABP can be regarded as a one-step or one-state Markov decision process, in which an agent has to select one out of $K$ possible actions and obtains a reward afterwards that is typically assumed to be drawn independent and identically distributed for each action. The agent tries to minimize the regret, which is defined as the expected difference between its accumulated reward and the reward that would have been accumulated with the optimal but unknown stationary policy. One popular algorithm in this setting is upper confidence bound (UCB: Agrawal, 1995b). UCB learns a deterministic policy that always selects the action with the maximum upper bound on the confidence interval of the expected reward. This upper bound is constructed from the past rewards for the action and is based on their empirical mean and a padding function. The padding function summarizes the uncertainty in the estimate of the expected

reward. Since UCB always selects the action with the maximum upper bound, this results in choosing actions where the reward is either very uncertain (large padding) or expected to be high (large empirical mean). By this, UCB inherently trades off exploration and exploitation.

One crucial assumption of most algorithms like UCB is that the reward distributions do not change over time. This assumption will not be satisfied in our settings. There exist bandit algorithms that are designed to deal with non-stationary MABPs, in which the reward distributions might change over time, either abruptly, leading to sliding window UCB (Garivier and Moulines, 2011), or slowly but continuously, leading to discounted UCB (Kocsis and Szepesvári, 2006). In the next section, we show how we can use these kind of algorithms for active task selection by modeling the task-selection problem as a non-stationary multi-armed bandit problem.

## 3. Proposed Methods

In this paper, we consider contextual policy search problems. Thus, we try to learn $\boldsymbol{\omega}$ which maximizes $J(\boldsymbol{\omega}) = \int_{\boldsymbol{s}} p(\boldsymbol{s}) \int_{\boldsymbol{\theta}} \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s}) \mathcal{R}_{\boldsymbol{s},\boldsymbol{\theta}} \, \mathrm{d}\boldsymbol{\theta} \, \mathrm{d}\boldsymbol{s}$ by performing rollouts of the control policy with parameters $\boldsymbol{\theta}$ in contexts $\boldsymbol{s}$. However, in contrast to prior work, we consider the context distribution during learning not to be given by the environment but to be under the agent's control. While this assumption might not apply to all contextual policy search problems, there are sufficiently many settings, like for instance ball throwing with self-selecting goal positions, to make studying this setting worthwhile. Note that the context distribution $p(\boldsymbol{s})$ within $J(\boldsymbol{\omega})$ remains fixed during evaluation and cannot be modified by the agent.

We introduce the problem of active context selection and its objective, namely to maximize learning progress, formally in Section 3.1. We propose different intrinsic reward functions which can be considered as proxies for the learning progress objective (see Section 3.2). In Section 3.3, we discuss how a context selection policy can be learned using multi-armed bandit algorithms.

### 3.1 Active Context Selection

Formally, we introduce a context selection policy $\pi_\beta$, which selects the context in which the next trial will be performed. $\pi_\beta$ aims at optimizing the expected *learning progress* of a contextual policy search method like C-REPS and, hence, minimizing the required number of episodes to reach a desired level of performance. In contrast, the control policy $\pi_{\boldsymbol{\theta}}$ and the upper-level policy $\pi_{\boldsymbol{\omega}}$ are learned to maximize $J(\boldsymbol{\omega})$ directly. An illustration of the components of contextual policy search is given in Figure 1.

We define the learning progress at time $t$ when performing a trial in context $\boldsymbol{s}$, with the control policy parameters $\boldsymbol{\theta}$ selected according to $\pi_{\boldsymbol{\omega}}$, as $\Delta_{\boldsymbol{s}}(t) = J(\boldsymbol{\omega}_{t+k}) - J(\boldsymbol{\omega}_t)$, where $\boldsymbol{\omega}_t$ are the parameters learned by contextual policy search at time $t$. Note that contextual policy search methods like C-REPS typically do not update $\boldsymbol{\omega}$ after every rollout but only after a certain number of rollouts $k$. Hence, it is more appropriate to define the learning progress over the window $k$ than between successive values of $J(\boldsymbol{\omega}_t)$.

The learning task on the task-selection level can now be framed as finding $\pi_\beta$ such that $\pi_\beta$ selects contexts that maximize the expected learning progress $\mathbb{E}_{\pi_{\boldsymbol{\omega}}}\{\Delta_{\boldsymbol{s}}(t)\}$. The learning progress $\Delta_{\boldsymbol{s}}(t)$ is stochastic because the rollouts and the upper-level policy $\pi_{\boldsymbol{\omega}}$ are

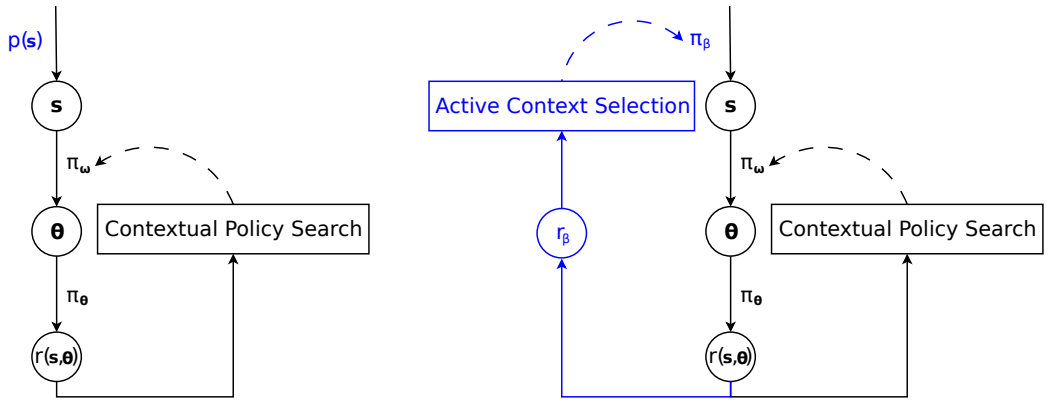


Figure 1: Contextual policy search (left): A context vector $\boldsymbol{s}$ is given by the environment according to some fixed distribution $p(\boldsymbol{s})$, the upper-level policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$ generates the parameters of the control policy for $\boldsymbol{s}$, the control policy is executed in the environment and a reward $r(\boldsymbol{s},\boldsymbol{\theta})$ is obtained. A contextual policy search component updates the upper-level policy based on the reward with the goal to maximize $J(\boldsymbol{\omega})$. Active contextual policy search (right): The context vector $\boldsymbol{s}$ is selected by the context selection policy $\pi_\beta$ which will be adjusted by an active context selection component based on the intrinsic reward $r_\beta$. $r_\beta$ is an estimate of the learning progress based on $r(\boldsymbol{s},\boldsymbol{\theta})$. Differing parts are marked in blue.

stochastic, hence the expectation. Furthermore, the learning progress is also non-stationary since it depends on the quality of $\boldsymbol{\omega}_t$: if $J(\boldsymbol{\omega}_t)$ is already close to the optimum, the typical learning progress is smaller than the learning progress at the beginning of learning when $\boldsymbol{\omega}_t$ is usually far from optimal.

We restrict ourselves to cases in which $\pi_\beta$ can only select among a finite number $K$ of predetermined contexts $\{\boldsymbol{s}_1,\ldots,\boldsymbol{s}_K\}$ during learning. These $K$ predetermined contexts are, however, elements of a continuous context space $S$ over which $\pi_{\boldsymbol{\omega}}$ shall generalize and where $J(\boldsymbol{\omega})$ is evaluated. The restriction to a discrete set of training contexts allows to apply well-established multi-armed bandit algorithms. Choosing the discrete set of training contexts can be based either on domain knowledge or on simple heuristics: for instance, for low-dimensional context spaces, an equally spaced grid over the context space can be used for training. In high-dimensional context spaces this would become infeasible due to the course of dimensionality. Hence, in this situation sampling the set of training contexts from a uniform random distribution over the continuous context space is more viable.

### 3.2 Intrinsic Reward Functions

Since $\mathbb{E}_{\pi_{\boldsymbol{\omega}}}\{\Delta_{\boldsymbol{s}}(t)\}$ is unknown to the agent and difficult to estimate because of its non-stationarity, we propose several heuristic but easily computable reward functions $r_\beta$ which reward the agent for certain proxy criteria. These proxy reward functions can be considered as means for intrinsic motivation of an agent (Barto et al., 2004), which drives it to engage in activity that increases its competence in task solving on a larger time scale. In the empirical

experiments, we evaluate which $r_\beta$ is a good proxy for the actual expected learning progress. Note that the heuristics that we propose are not exactly comparable to any query strategy from supervised learning since the learning process in reinforcement learning in a specific task is iterative unlike in typical supervised learning settings.

In a generalized form, we can write the proposed proxies of the learning progress as

$$r_\beta = f(r(\boldsymbol{s}_t, \boldsymbol{\theta}_t) - \hat{b}_{\boldsymbol{s}_t}),$$

where $r(\boldsymbol{s}_t, \boldsymbol{\theta}_t)$ is the return obtained in episode $t$ with policy $\pi_{\boldsymbol{\theta}_t}$ in context $\boldsymbol{s}_t$, $\hat{b}_{\boldsymbol{s}_t}$ is a baseline term, and $f$ is an operator. Note that the bandit algorithm addresses the stochasticity in $r(\boldsymbol{s}_t, \boldsymbol{\theta}_t)$ and, hence, the heuristics usually need not account for this stochasticity themselves.

#### 3.2.1 Best-Reward Heuristic

This heuristic directly uses the reward obtained by the control policy $\pi_{\boldsymbol{\theta}}$ in task $\boldsymbol{s}$ in rollout $t$ as the proxy for the learning progress, that is $r_\beta = r(\boldsymbol{s}, \boldsymbol{\theta}_t)$. Thus $\hat{b}_{\boldsymbol{s}_t} = 0$ and the operator $f$ is simply the identity. This heuristic corresponds to assuming that the agent makes the most progress when it focuses on improving its performance in tasks in which it is already performing well. The idea behind this heuristic is that we should first learn easy tasks perfectly because this can help us to learn similar but more difficult tasks later on.

A potential problem of this heuristic are settings in which high reward does not correspond to easy tasks, for instance when the maximum achievable reward differs in different context. Additionally, the best-reward heuristic requires that knowledge can be transferred very well between contexts. Otherwise it converges to the context with the highest rewards.

#### 3.2.2 Diversity Heuristic

Quite opposite to the best-reward heuristic, this heuristic encourages to select tasks in which the agent receives the worst reward. For this, the negative actual reward $r_\beta = -r(\boldsymbol{s}, \boldsymbol{\theta}_t)$ is used. Thus $\hat{b}_{\boldsymbol{s}_t} = 0$ and the operator $f$ is simply $f(x) = -x$. The intuition for this heuristic is that one should focus on the hardest tasks in which the current performance is worst since in these tasks the potential for large improvements is high. This heuristic bears similarities to the heuristic proposed by Ruvolo and Eaton (2013) for supervised learning and is hence called "diversity" heuristic.

Similar to the best-reward heuristic, this heuristic might have problems in settings in which the maximum achievable reward differs in different context and thus, the obtained rewards might not be comparable. Another disadvantage appears in cases with unlearnable contexts. The diversity heuristic would then focus on these unlearnable contexts.

#### 3.2.3 1-step Progress Heuristic

This heuristic uses the difference of the last two rewards obtained in the respective context as proxy for the actual learning progress, that is, $r_\beta = r(\boldsymbol{s}, \boldsymbol{\theta}_t) - r(\boldsymbol{s}, \boldsymbol{\theta}_{\bar{t}})$, where $\bar{t}$ is the index of the previous rollout in context $\boldsymbol{s}$. Thus the baseline is simply the last obtained reward $\hat{b}_{\boldsymbol{s}_t} = r(\boldsymbol{s}, \boldsymbol{\theta}_{\bar{t}})$ and the operator $f$ is the identity. The heuristic can be seen as the most straight-forward proxy for the learning progress as it replaces effectively the integration over

**Algorithm 1** Discounted Upper-Confidence Bound (D-UCB) (Kocsis and Szepesvári, 2006)

**Require:** $K$: number of tasks; $\gamma$: discounting factor; $\xi > 0$: some parameter controlling the strength of padding; $t$: number of rollouts; $i_1, \ldots, i_t$: previously selected tasks; $r_1, \ldots, r_t$: previous rewards

**if** $t < K$ **then**
  **return** $t$ # Sample each task at least once
**else**
  **for** $i \in \{1, \ldots, K\}$ **do**
    $n_i \leftarrow \sum_{t_j=1}^{t} \gamma^{t-t_j} \mathbb{1}_{\{i_{t_j}=i\}}$ # Discounted number of rollouts in task $i$
  **end for**
  $n \leftarrow \sum_{i=1}^{K} n_i$ # Discounted total number of rollouts
  **for** $i \in \{1, \ldots, K\}$ **do**
    $\overline{r}_i \leftarrow \frac{1}{n_i} \sum_{t_j=1}^{t} \gamma^{t-t_j} r_{t_j} \mathbb{1}_{\{i_{t_j}=i\}}$ # Discounted mean reward in task $i$
    $c_i \leftarrow 2B\sqrt{\frac{\xi \log n}{n_i}}$ # Padding function for task $i$
  **end for**
  **return** $\arg\max_{i \in \{1,\ldots,K\}} \overline{r}_i + c_i$ # Select task in which discounted UCB is maximal
**end if**

$\boldsymbol{s}$ and $\boldsymbol{\theta}$ in $J(\omega)$ by single samples and uses the reward sample $r(\boldsymbol{s}, \boldsymbol{\theta}_t)$ as estimate for $\mathcal{R}_{\boldsymbol{s},\boldsymbol{\theta}_t}$. As it is based solely on differences of rewards rather than absolute values, it should be able to cope better than the best-reward and diversity heuristic with situations, in which the maximum achievable reward differs in different contexts.

A drawback of the 1-step progress heuristic is that it generates reward signals $r_\beta$ with high variance. Variance in $r_\beta$ is inevitable because of the stochasticity of both the environment and the upper-level policy $\pi_{\boldsymbol{\omega}}$. However, since the baseline of this heuristic is the last obtained reward, this baseline has also a high variance, which need not be the case.

#### 3.2.4 Monotonic Progress Heuristic

Although bandit algorithms account for the stochasticity in the 1-step progress heuristic, it might be useful to reduce already the variance of the intrinsic reward. We can use more stable baselines such as the maximum reward of all previous rollouts in the context $\boldsymbol{s}$, that is: $\hat{b}_{\boldsymbol{s}} = \max_{t:\boldsymbol{s}_t=\boldsymbol{s}} r(\boldsymbol{s}, \boldsymbol{\theta}_t)$. Furthermore, since the learning progress is typically monotonically increasing, using the operator $f(x) = \max(0, x)$ to avoid negative $r_\beta$ appears to be reasonable. The resulting heuristic is denoted as "monotonic progress heuristic" and has the form $r_\beta = \max(0, r(\boldsymbol{s}, \boldsymbol{\theta}_t) - \max_{t:\boldsymbol{s}_t=\boldsymbol{s}} r(\boldsymbol{s}, \boldsymbol{\theta}_t))$.

The heuristic considers the learning progress to be monotonic and strictly positive. In comparison to the 1-step progress heuristic, the intrinsic reward $r_\beta$ will be more often zero and its baseline has less variance since unsuccessful explorative rollouts have no effect on its value.

### 3.3 Learning Context Selection Policies

Since the reward $r_\beta$ is stochastic, we propose to use a learning algorithm for determining $\pi_\beta$. As we consider only a finite number of predetermined contexts, the problem of learning $\pi_\beta$ can be framed as a MABP, where the contexts correspond to the "arms" and $r_\beta$ to the bandit's reward. Due to the non-stationarity of $r_\beta$, standard UCB-like algorithms are not sufficient as they expect the rewards to be drawn from a non-changing probability distribution. In contrast, we propose to use D-UCB (see Algorithm 1; Kocsis and Szepesvári, 2006) for active context selection since it explicitly addresses changing reward distributions. For this, D-UCB estimates the instantaneous expected reward by a weighted average of past rewards where higher weight is given to recent rewards. More specifically, a discount factor $\gamma \leq 1$ is introduced and the reward that has been obtained at time step $t_j$ is weighted with the factor $\gamma^{t-t_j}$ at time $t$. The central idea of D-UCB is that the discounting can compensate for continuously but slowly changing reward distributions.

The upper confidence bound of the D-UCB algorithm for arm $i$ takes the form $\overline{r}_i + c_i$ where $\overline{r}_i = \frac{1}{n_i} \sum_{t_j=1}^{t} \gamma^{t-t_j} r_{t_j} \mathbb{1}_{\{i_{t_j}=i\}}$ is the discounted mean and $c_i = 2B\sqrt{\frac{\xi \log n}{n_i}}$ is the padding function which controls the width of the confidence intervals. In this formulas, $n_i$ corresponds to the discounted number of draws of arm $i$ and $n$ to the total discounted number of draws (see Algorithm 1). Furthermore, $r_{t_j}$ is the reward obtained in the $t_j$-th rollout and $i_{t_j}$ is the arm played in this rollout. $\mathbb{1}_{\{i_{t_j}=i\}}$ is the Kronecker delta which is one if the equality holds and zero otherwise. $B$ and $\xi$ are parameters of the algorithm which control the width of the confidence intervals, where $B$ is an upper bound on the rewards and $\xi$ needs to be chosen appropriately.

## 4. Model of the Contextual Learning Problem

In this section, we show how contextual policy search algorithms can benefit from active context selection by means of a simple artificial model of the contextual learning problem. The model abstracts away the contextual policy search which is possible because our approach treats it as a black box (see Figure 1).

We assume that the context space $S = \{0, 1, \ldots, 9\}$ is discrete and associated to each context $\boldsymbol{s}$ is a hidden value $l_{\boldsymbol{s}} \in [0, 100]$ that indicates the agent's competence in $\boldsymbol{s}$, that is, how well $\boldsymbol{s}$ has been learned. Large values of $l_{\boldsymbol{s}}$ simulate that the current policy parameters in context $\boldsymbol{s}$ are close to the optimal policy parameters $\boldsymbol{\theta}^*(\boldsymbol{s})$. The true reward in context $\boldsymbol{s}$, which is given by

$$r(\boldsymbol{s}) = (1 + \exp(-0.1 l_{\boldsymbol{s}} + 4))^{-1} + b_{\boldsymbol{s}},$$

depends directly on $l_{\boldsymbol{s}}$. $r(\boldsymbol{s})$ corresponds to a scaled and shifted logistic function so that $r(\boldsymbol{s}) \approx b_{\boldsymbol{s}}$ if $l_{\boldsymbol{s}} \approx 0$ and $r(\boldsymbol{s}) \approx 1 + b_{\boldsymbol{s}}$ if $l_{\boldsymbol{s}} \geq 100$. In real learning problems, different tasks often have a different maximum reward. To simulate this, we artificially create a true reward baseline $b_{\boldsymbol{s}}$ for each context. The baseline is randomly sampled from a normal distribution with zero mean and standard deviation $\sigma_b$. The true reward is not observed directly. Instead, we add Gaussian noise with zero mean and standard deviation $\sigma_r$ to simulate trial and error of a learning agent.

We assume that each context has an intrinsic complexity which controls how much the agent learns in a single trial in this context. This complexity can change abruptly between

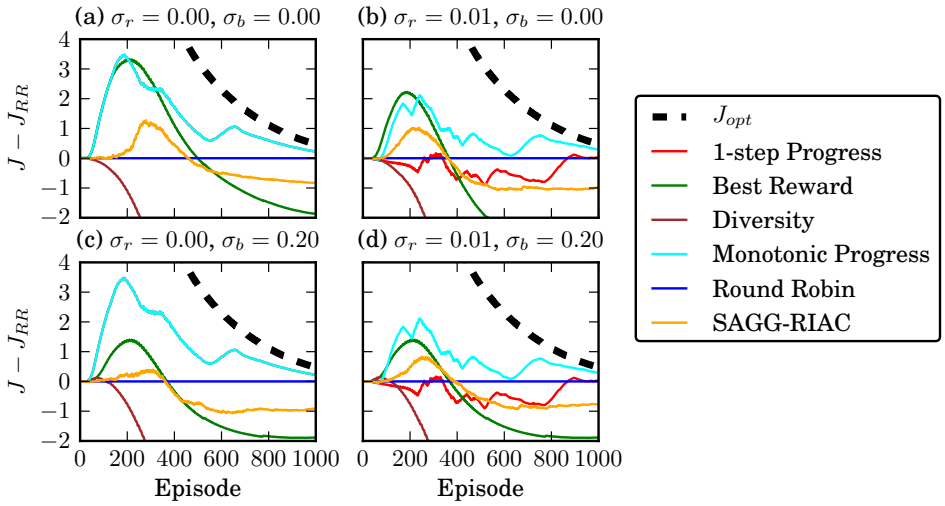


Figure 2: Relative learning curves of several active context selection methods. The performance of the round robin context selection baseline ($J_{RR}$) is subtracted from all learning curves. The values of the standard deviation for reward measurements $\sigma_r$ and of the standard deviation for the baseline $\sigma_b$ have been varied. After 1000 episodes the monotonic progress heuristic has approximately reached the upper bound of the performance $J_{opt}$.

neighboring contexts even in continuous domains. This model is motivated for instance by reaching tasks, in which it might happen that a slight modification of the goal position requires that the agent needs to avoid an obstacle that blocks the direct path. As a result, the slightly different context corresponding to the blocked goal would have a significantly worse expected learning progress than its neighbors and its solution cannot be transferred well. We model this behavior of learning progress and skill transfer by assigning a different learning progress factor $w_{\boldsymbol{s}} \in \{0.1^2, 0.2^2, \ldots, 1^2\}$ to each context randomly, where large $w_{\boldsymbol{s}}$ corresponds to a larger improvement of competence after one additional trial in $\boldsymbol{s}$.

Each time, the reward of a context $\boldsymbol{s}_t$ will be queried, the values $l_{\boldsymbol{s}}$ of each contexts $\boldsymbol{s}$ will be updated to simulate the learning progress according to the update rule

$$l_{\boldsymbol{s}}^{(t)} = l_{\boldsymbol{s}}^{(t-1)} + w_{\boldsymbol{s}_t} \cdot 0.5^{|\boldsymbol{s}-\boldsymbol{s}_t|},$$

which models that experience obtained in one context generalizes to other contexts based on their similarity (measured here using the euclidean distance). Contexts which are easier learnable (large $w_{\boldsymbol{s}}$) lead to higher overall learning progress at the beginning. However, it does not make sense to focus at the context with maximum $w_s$ indefinitely because $r(s)$ saturates once $l_s \geq 100$. Hence, the estimate of the learning progress has to be adaptive. Since we only have a discrete set of contexts, we can exactly compute $J = \sum_{\boldsymbol{s}} r(\boldsymbol{s})$ and the upper bound of $J$ is $J_{opt} = 10 + \sum_{\boldsymbol{s}} b_{\boldsymbol{s}}$.

In addition to our proposed methods, we examine the context generation and selection method from SAGG-RIAC and context selection in a fixed order (round robin). In order to

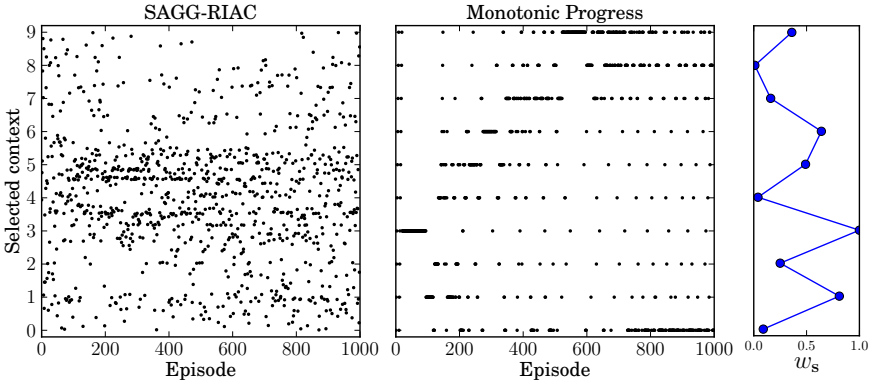


Figure 3: Selected contexts per episode with SAGG-RIAC and monotonic progress heuristic and D-UCB (with $\sigma_r = 0$). The learning progress factor $w_{\boldsymbol{s}}$ of each context is displayed on the right side.

show different properties of the active context selection methods, we display the number of episodes versus the relative $J$ in comparison to round robin selection in Figure 2. We have used the parameters $\gamma = 0.95$ and $\xi = 10^{-8}$ in all heuristics that are based on D-UCB. For the diversity and the best-reward heuristics we have used $B = 1$ and for all others $B = 0.25$. For SAGG-RIAC, we set the maximum size of samples per region before we split the region to 8 and the window size which is used to compute the interest value is 10.

We can see that despite different learning progress factors, round robin context selection is a good baseline. The best-reward heuristic that focuses on the best learnable context is a good heuristic at the beginning. However, when the best context approaches the optimum reward, the learning progress decreases and approaches zero. At this time it would be better to switch to a context in which greater learning progress can be achieved. This will be even more significant for larger context spaces because the transferability of knowledge decreases with the size of the context space and the complexity of the different contexts. In addition, an artificial baseline for each reward (see Figure 2 (c) and (d)) leads to severe degradation because the context with maximum $r(\boldsymbol{s})$ is not necessarily the context with maximum learning progress. The diversity heuristic does not work well either. This is because the differences of the expected learning progress are too large between contexts and it will select the worst learnable context.

The 1-step progress heuristic and monotonic progress heuristic essentially focus on the same context as the best-reward heuristic at the beginning. But they switch to other contexts with greater learning progress when the learning progress in the best learnable context decreases. Moreover, the 1-step progress and monotonic progress heuristics are invariant under different baselines. The heuristics behave identically when the reward is noise-free, that is $\sigma_r = 0$. If there is noise (which simulates the exploration of the agent), the monotonic progress heuristic is usually better (see Figure 2 (b) and (d)).

A context selection method that differs from all others is the context generation and selection method from SAGG-RIAC. As it has been described in Section 2.2, it is designed for continuous context spaces. However, it has a crucial disadvantage in our model of the

learning progress: we assume that the expected learning progress of neighboring contexts can change abruptly. This is a a problem for SAGG-RIAC because it focuses on *regions* of the context space that have a high reward derivative. Among these are not only regions with a high learning progress but also regions with abruptly changing learning progress. In Figure 3 we can see which contexts have been selected by SAGG-RIAC during the simulated learning process: it focuses most of the time on the region around the contexts 3, 4 and 5 because of the greatly varying learning progress factor $w_{\boldsymbol{s}}$ in this region, which results in a high competence derivative. Therefore, a significant part of the explored contexts are not informative because the context 4 has a very low learning progress.

The monotonic progress heuristic, in contrast, selects most of the time tasks with a high learning progress as we can see in Figure 3. At the beginning, it focuses on the context 3 which has the highest $w_{\boldsymbol{s}}$. After some time, when the learning progress in this context saturates, it concentrates on other contexts. At the end it concentrates on the contexts that have not been learned perfectly yet even though they have a low intrinsic learning progress factor.

## 5. Results

We provide an empirical evaluation of the proposed methods on an artificial contextual benchmark problem in Section 5.1 and in two ball throwing tasks with a simulated Mitsubishi PA-10 in Section 5.2.

### 5.1 Contextual Function Optimization

In this section, we evaluate the proposed approach on an artificial test problem, compare it to reasonable baseline methods, and analyze the effect of different intrinsic reward heuristics. The test problem is chosen such that some contexts are harder in the sense that the parameters $\boldsymbol{\theta}$ need to be chosen more precisely to reach the same level of return. By focusing on learning primarily the parameters $\boldsymbol{\theta}$ for these contexts, active context selection should be able to outperform a uniform random context selection.

#### 5.1.1 Problem Domain

The context is denoted by $\boldsymbol{s} \in S = [-1, 1]^{n_s}$. We use the objective function

$$f(\boldsymbol{\theta}, \boldsymbol{s}) = -||A\boldsymbol{\theta} - \boldsymbol{s}||_2 \cdot ||s||_2^2 + \sum_{i=0}^{n_s-1} \boldsymbol{s}_i,$$

where $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$ denotes the low-level parameters and the matrix $A \in [0, 1]^{n_\theta \times n_s}$ is chosen uniform randomly such that it has rank $n_s$ ($n_\theta > n_s$). The objective function consists of three terms: the *parameter error* $-||A\boldsymbol{\theta} - \boldsymbol{s}||_2$ which can be influenced by the agent's choice of $\boldsymbol{\theta}$, the *context complexity* $||s||_2^2$, which controls how strongly the agent's parameter error deteriorates the task performance, and the *baseline* $\sum_{i=0}^{n_s-1} \boldsymbol{s}_i$, which controls the maximum value in a context. Since A has rank $n_s$, $\boldsymbol{\theta}$ can always be chosen such that the parameter error becomes 0 and thus the optimal value $f^*(\boldsymbol{s})$ is equal to the baseline $\sum_{i=0}^{n_s-1} \boldsymbol{s}_i$. However, if the agent chooses $\boldsymbol{\theta}$ suboptimally, the same parameter error has different effects on the value of $f$ in different contexts: in contexts with high context complexity, the value of $f$ will

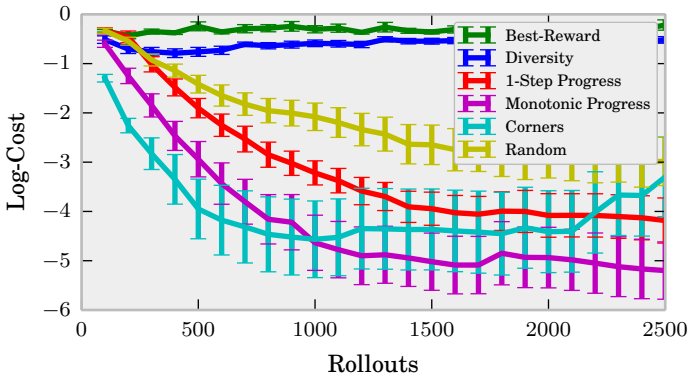


Figure 4: Learning Curves of Different Task Selection Heuristics. A contextual upper-level policy has been learned using C-REPS for different active task-selection strategies. The logarithm of the cost $|f(\boldsymbol{\theta}, \boldsymbol{s}) - f^*(\boldsymbol{s})|$ averaged over 100 test contexts is used as performance measure. Shown are mean and standard error of the mean for 20 runs of 2500 rollouts.

be considerably smaller than $f^*$, while the difference will be less pronounced in contexts with lower context complexity. Most extremely, for $\boldsymbol{s} = \mathbf{0}$, the choice of $\boldsymbol{\theta}$ is arbitrary since $f$ will always be equal to $f^*$. Thus, an agent should focus on learning $\boldsymbol{\theta}$ in the contexts with high complexity if its objective is to minimize $|f(\boldsymbol{\theta}, \boldsymbol{s}) - f^*(\boldsymbol{s})|$.

#### 5.1.2 Comparison of Task Selection Heuristics

In a first experiment, we compare task selection with D-UCB for different intrinsic reward heuristics to two baseline methods. In this experiment, training takes place on 25 contexts placed on an equidistant grid over a context space with $n_s = 2$ dimensions; that is, the set of contexts that will be used for training is $S_{\text{train}} = [-1, -\frac{1}{2}, 0, \frac{1}{2}, 1]^2$. The objective of learning, however, is to generalize $\pi_{\boldsymbol{\omega}}$ over the entire context space $S$, that is, to choose $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$ such that $f(\boldsymbol{\theta}, \boldsymbol{s})$ is maximized. As baseline, we use a "Random" task-selection heuristic, which selects uniform randomly among the training contexts. Moreover, we use a "Corner" task-selection heuristic, which selects the four contexts, where the context complexity is maximal, that is $\boldsymbol{s} = (\pm 1, \pm 1)$, in a round-robin fashion.

For the D-UCB, we have used $B = 1.0$, $\gamma = 0.99$, and $\xi = 10^{-8}$. The external reward $r(\boldsymbol{s}, \boldsymbol{\theta})$, based on which the intrinsic reward $r_\beta$ is computed, is set to $r(\boldsymbol{s}, \boldsymbol{\theta}) = f(\boldsymbol{\theta}, \boldsymbol{s})$ where $\boldsymbol{\theta} = \pi_{\boldsymbol{\omega}}(\boldsymbol{s})$ is sampled from the upper-level policy for the given context $\boldsymbol{s}$. Note that the rewards in $\boldsymbol{s}$ have high variance because of the agent's explorative behavior and are also non-stationary since they depend on the current upper level policy $\pi_{\boldsymbol{\omega}}$. Contextual policy search was conducted with C-REPS with $\epsilon = 2.0$, $N = 50$, and performing an update every 25 rollouts. The evaluation criterion is the expected value of $|f(\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s}), \boldsymbol{s}) - f^*(\boldsymbol{s})|$ of the learned contextual policy $\pi_{\boldsymbol{\omega}}$ over the context space $S$, where exploration of $\pi_{\boldsymbol{\omega}}$ is disabled. We approximate this quantity by computing the average return of $\pi_{\boldsymbol{\omega}}$ on 100 test contexts sampled uniform randomly from $S$.

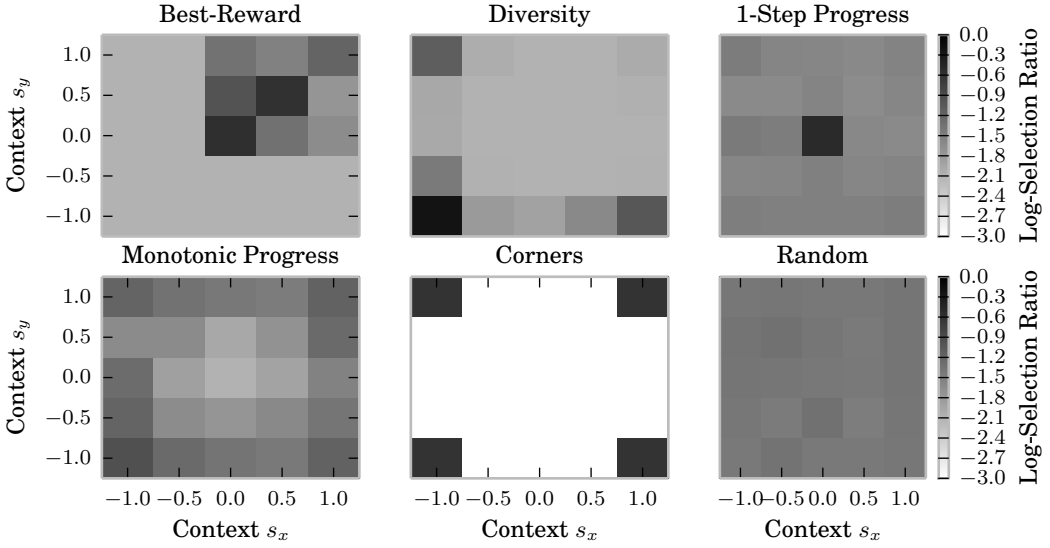


Figure 5: Task preferences of different task-selection strategies during the first 250 rollouts of training. Shown is the logarithm of the mean selection ratio.

Figure 4 shows the learning curves for different task-selection strategies. Figure 5 shows which contexts (tasks) are selected by the different strategies during the first 250 rollouts of training. D-UCB with the "Best-Reward" and the "Diversity" intrinsic reward performs significantly worse than a uniform random task selection. The reason for this is that "Best-Reward" focuses mostly on tasks where the baseline term is large (upper-right area in Figure 5) or where the task complexity is small (central area). Conversely, "Diversity" focuses on areas where the baseline term is small. Both strategies are too imbalanced if the baseline term's contribution is not negligible.

D-UCB with the "1-step Progress" intrinsic reward performs equally bad during the first 250 rollouts. The reason for the low initial progress is that the "1-step Progress" intrinsic reward not only rewards progress but also penalizes regression. However, regression is inevitable during the initial explorative phase. Because of this, this intrinsic reward heuristic focuses initially on contexts with small context complexity where the parameter error and thus the explorative behavior have only a small effect on the actual reward. After the initial explorative phase, this intrinsic reward gets more informative and the corresponding active task selection outperforms uniform random task selection in the long run.

D-UCB with the "Monotonic Progress" intrinsic reward performs considerably better than both D-UCB with the other heuristics and uniform random task selection. The reason for this is that it initially favors complex contexts (the outer areas in Figure 5 with $||s||_2 \gg 0$), where the potential reward improvement is large, without any preference for tasks with small of large baseline value. This task-selection strategy works well and results in a large and stable learning progress. Based on this, we have tested a second baseline denoted as "Corners", which selects the four contexts with maximum context complexity in a round-robin fashion. While this resulted in a very rapid learning progress initially, it is slightly

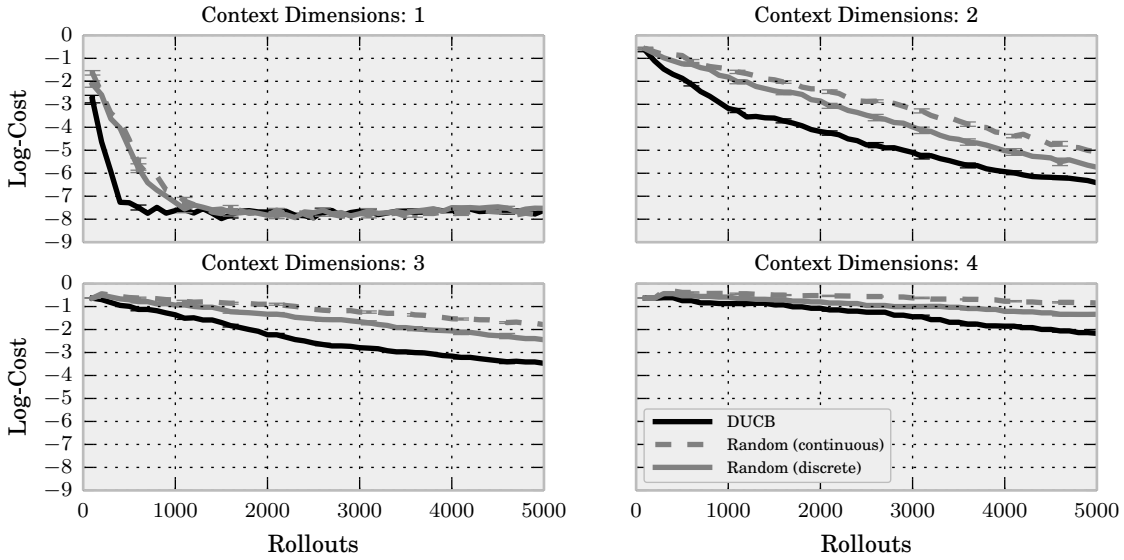


Figure 6: Learning Curves for Different Context Dimensionality. Shown are mean and standard error of the mean for 20 runs of 5000 rollouts.

suboptimal and unstable in the long run since only four of the tasks are ever sampled. Conversely, the "Monotonic Progress" intrinsic reward leads to a more balanced selection of tasks when converging and is thus favorable in the long run. In summary, D-UCB with the "Monotonic Progress" intrinsic reward selects tasks in a way which increased C-REPS' learning progress considerably and proved to be stable at the same time.

### 5.1.3 Dimensionality of the Context Space

In a second experiment, we compare the performance of D-UCB to uniform random task selection for different dimensionality of the context space. A discrete set $S_{\text{train}}$ of 25 contexts for training has been generated by selecting the $k$-the context $\boldsymbol{s}_k$ uniform random from $\mathbb{R}^{n_s}$ under the constraint $||\boldsymbol{s}_k||_2 = k/25$. The set of test contexts has been generated in the same way but with an other random seed. D-UCB has been combined with the "Monotonic Progress" heuristic. The D-UCB parameters have been set to $B = 1.0$, $\gamma = 0.99$, and $\xi = 10^{-8}$, and the C-REPS parameters to $\epsilon = 1.0$ and $N = 25n_s^2$, and an update was performed every $13n_s^2$ rollouts. As baseline, two different random context selection strategies have been tested: "Random (discrete)" chooses tasks uniform randomly from $S_{\text{train}}$ while "Random (continuous)"chooses tasks uniform randomly from $\mathbb{R}^{n_s}$ under the constraint $||\boldsymbol{s}_k||_2 \leq 1$.

Figure 6 shows the learning curves for $n_s \in \{1, 2, 3, 4\}$. For any value of $n_s$, D-UCB outperforms random task selection in terms of the initial learning progress (not in terms of the final performance). One can further see that selecting a finite, discrete set of training contexts from the continuous context space does not necessarily impair performance; conversely "Random (discrete)" typically performs slightly better than "Random (continuous)". While the general learning progress decreases considerably for higher dimensionality

$n_s$, this is not directly an issue of the task-selection strategy (since D-UCB still outperforms random task selection) but rather of the underlying contextual policy search method. Thus, the results show that active context selection with D-UCB works satisfyingly while higher dimensional context spaces remain a general challenge for contextual policy search.

### 5.2 Generalize Throwing Movements

In this section, we consider the problem of learning to throw a ball at a given target. A similar setting has been investigated by Wirkus et al. (2012), where the objective was to learn throwing an object at a specified target based on a forward model of the system. In our experiments, the target can be located at different positions in a predetermined area on the ground and we do not provide any model of the system, making it a contextual model-free policy search problem with the target position being the context. We consider two different reward functions for this experiment. One reward function is continuous (grid problem) while the other has discontinuities and flat regions without a "reward gradient" (dartboard problem), resulting in a setting with easy and more difficult tasks. We provide an empirical evaluation on the grid problem and apply the gained insight on the dartboard problem. In our experiments, we use a simulated Mitsubishi PA-10 robot arm with seven joints for throwing (see Figure 7).

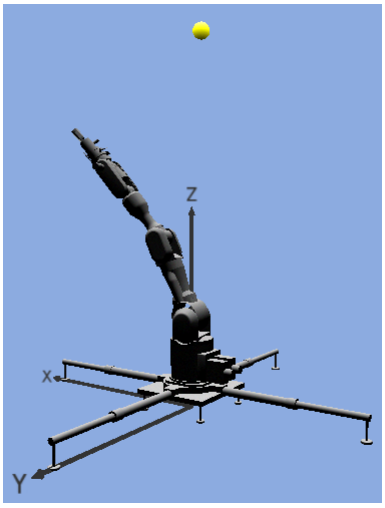


Figure 7: Visualization of the simulated Mitsubishi PA-10 throwing a ball.

#### 5.2.1 Policy Representation

A throwing behavior in this experiment consists of a sequence of two movement primitives (DMPs), where the first corresponds to the strike out and the second one to the actual throwing movement. Both primitives have a duration of $\tau_1 = \tau_2 = 0.5\,\mathrm{s}$. The movement primitives define joint trajectories directly for the seven joints of the Mitsubishi PA-10. The parameters of the two DMPs include the weights of the forcing terms and the respective meta-parameters (see Section 2.1). These parameters have been initialized such that the resulting throw hits the ground position $(-3.55, -3.55)$, where $(0, 0)$ is the ground position at which the PA-10 is mounted and the unit of the coordinate system is in meter. Some of the meta-parameters of the initial policy are to be adapted later on such that other ground positions are hit. These include the final state of the first movement primitive $\boldsymbol{g}_1$ and the velocities at the end of the two movement primitives $\dot{\boldsymbol{g}}_1, \dot{\boldsymbol{g}}_2$. Instead of modifying $\boldsymbol{\theta} = (\boldsymbol{g}_1, \dot{\boldsymbol{g}}_1, \dot{\boldsymbol{g}}_2)$ directly, we use the values from the initial policy $\boldsymbol{\theta}_0$ as the base and we modify the offset $\boldsymbol{\epsilon}_t$ so that $\boldsymbol{\theta}_t = \boldsymbol{\theta}_0 + \boldsymbol{\epsilon}_t$. The weights $\boldsymbol{w}$ of the forcing terms of the two primitives and the other movement primitives remain fixed during this adaptation. Thus,

the actual contextual learning task consists of finding a mapping for $3 \cdot 7 = 21$ parameters such that different target positions on the ground are hit.

#### 5.2.2 Methods

We use C-REPS for contextual policy search with the context $\boldsymbol{s}$ which contains the Cartesian coordinates of the target position for the throw. The mapping $\varphi$ projects the context to polar coordinates and generates all quadratic terms of the polar coordinates. This mapping results from the constraint of C-REPS to match feature averages instead of the actual context distribution, which would result in an infinite number of constraints (Deisenroth et al., 2013). A policy update is performed after every 50 trials and a memory of at most 300 trials is used for the update. This memory consists of the best $300/K$ trials for each of the $K$ tasks, which results in an update that is similar to importance sampling in PoWER (Kober and Peters, 2011). We restricted the maximum Kullback-Leibler divergence of the old and new policy distributions to $\epsilon = 0.5$, the initial weight matrix was set to $\boldsymbol{W} = \boldsymbol{0}$, and the initial covariance was set to $\boldsymbol{\Sigma} = \sigma_0^2 \boldsymbol{I}$ with $\sigma_0^2 = 0.02$. For D-UCB, we use $\gamma = 0.99$, $B = 10,000$ and $\xi = 10^{-9}$ in all cases.

#### 5.2.3 Grid Problem

We define two contextual learning problems in this setting whose main difference is the structure of the reward function. The first reward function provides the squared distance of the position hit by the throw to the target position as reward: $r(\boldsymbol{s}, \boldsymbol{\theta}) = -||\boldsymbol{s} - \boldsymbol{b}_{\boldsymbol{\theta}}||_2^2$, where $\boldsymbol{s}$ is the target and $\boldsymbol{b}_{\boldsymbol{\theta}}$ are the Cartesian coordinates of the ball when it hits the ground after executing policy $\pi_{\boldsymbol{\theta}}$. In this problem, we generate an equidistant grid of 25 targets for training over the area $[-3, -5] \times [-3, -5]$. Another set of 16 targets is used to test the generalization. These targets form an equidistant grid in the area $[-3.25, -4.75] \times [-3.25, -4.75]$. While different contexts share the same reward function, learning them might differ in complexity. Some of the relative positions of the target to the PA-10 arm are harder to hit because of the kinematic structure of the arm. In addition, the distance to the initial policy makes some contexts easier to solve by exploration than others.

We compared our methods to three baselines: continuous random sampling of contexts from $[-3, -5] \times [-3, -5]$ ("Random (cont.)"), selection in a fixed order ("Round Robin") and the best policy that we have found in all experiments ("Best Policy"). On the left side of Figure 8 the learning curves of several intrinsic reward heuristics are shown and on the right side the best intrinsic reward heuristic is compared to the baselines.

The "Diversity" heuristic, which prefers selecting hard tasks in which a low reward is obtained, performs worse than round robin or random selection. Thus, selecting the more difficult tasks first is not beneficial to speed up learning in this setting. This effect might be even more pronounced in cases where the most difficult tasks are unsolvable. The "Best-Reward" heuristic performs worst here. We observed that it quickly converged to nearly always selecting the same task and hence failed to learn a generalizable upper-level policy. The reason for this behavior is that it encourages D-UCB to focus on the simplest task first. Because it improves quickly in this task, the reward will be considerably greater than the reward of the other tasks. Even though it periodically samples other tasks, the reward in these tasks will typically be smaller and thus, D-UCB sticks to the same task. Selecting

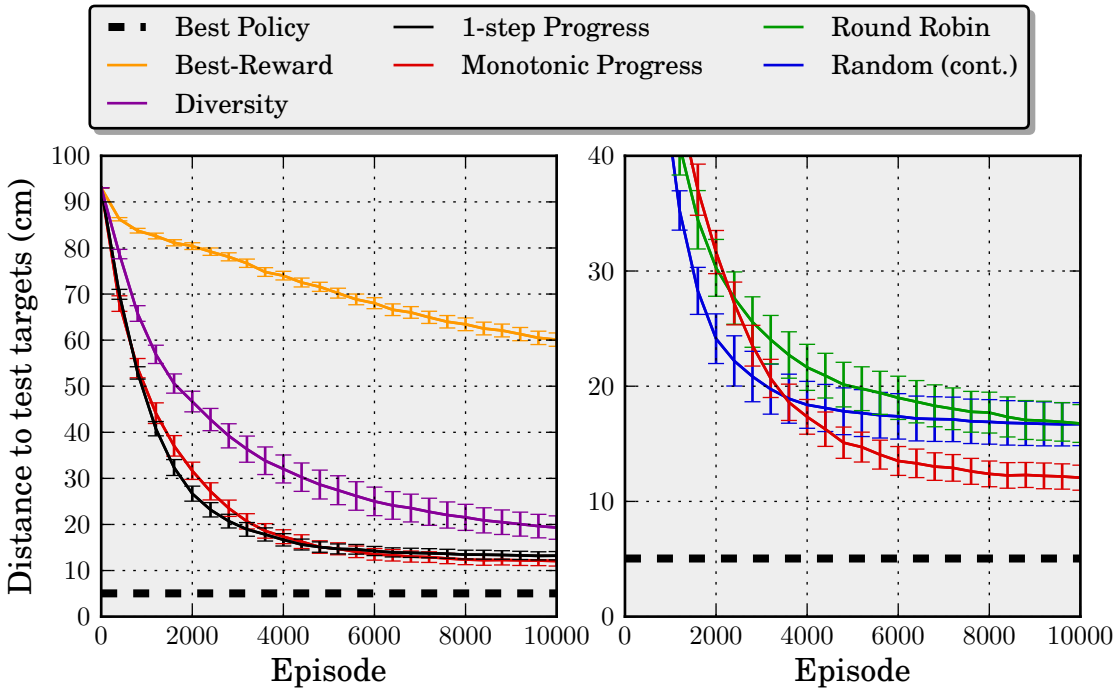


Figure 8: Left: Learning curves of several intrinsic reward heuristics for D-UCB in the grid problem. We measured the average distance to the test targets. The curves and the error bars show the mean and standard error of mean over 20 runs. "Best Policy" indicates the performance of the best policy that we have generated in all experiments. Right: Comparison of D-UCB with intrinsic reward based on the monotonic progress with the baselines.

the tasks in which one can make the greatest estimated learning progress gives a significant advantage in this problem. In contrast to the results in Section 5.1, the "1-step Progress" heuristic is on a par with the "Monotonic Progress" heuristic. A possible reason for this is that the inherent context complexities in this task do not vary as strongly as in the problem in Section 5.1.

In comparison to the baselines, D-UCB under the monotonic progress intrinsic reward is on a par with round robin selection and slightly worse than continuous random selection. Continuous random sampling learns more quickly in the beginning because the comparison of different methods is done on a set of test contexts that are maximally dissimilar from the discrete training context set. Methods that use only the discrete set of training contexts during the training phase have thus a disadvantage compared to continuous random sampling which typically samples closer to the test contexts.

In the long-term, however, D-UCB performs better than both baselines and its average performance gets close to the best policy's performance. This shows that active task selection can accelerate learning considerably and improve the reliability of the result of contextual policy search. A potential reason why continuous random sampling performs worse in the long term is that it cannot use the strategy otherwise employed in C-REPS,

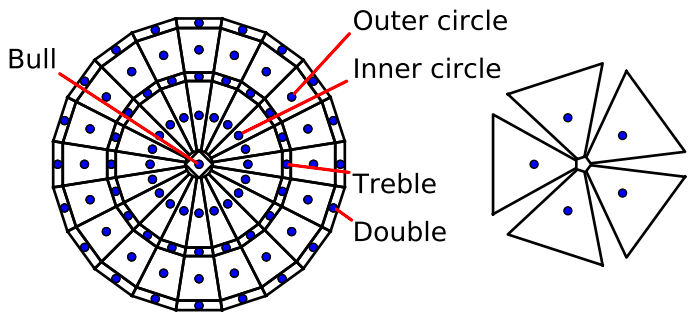


Figure 9: Left: The dartboard is divided into 81 quadrangular fields. Bull and bull's eye are regarded as one field, the rest belongs to one of the four circles (inner circle, outer circle, doubles, and trebles). The initial policy given by $\boldsymbol{\theta}_0$ would throw at the center of the dartboard, that is the bull. Right: The target regions of the bull, inner circle, and outer circle are greater than the corresponding fields on the dartboard and overlap neighboring fields. Shown here are 5 of the 20 target regions of the inner circle. These target regions are only used to compute the reward. Larger target regions can be assumed to make the corresponding learning tasks easier.

namely to keep a separate history of samples per context of identical size ($300/K$ rollouts per context), because it does not encounter any context twice.

#### 5.2.4 Dartboard Problem

As second scenario with the PA-10, we consider a problem which poses tasks of more significantly varying complexity onto the agent. The targets are designed to correspond to the fields on a dartboard. This virtual dartboard is placed on the ground in front of the PA-10. Placing the dartboard on the ground instead of a wall was mainly done to simplify implementation. The diameter is set to 1.4 m (real dartboards have a diameter of 0.451 m). Each field is approximated by quadrangles (see Figure 9) and the center of this field is the context of the task corresponding to the field. For each target, we assign a corresponding *target area* which is usually the quadrangular field. For some tasks, we enlarge each side of the quadrangular field by the factor 3.5 to build the corresponding target region (see Figure 9 for details). This will make these tasks considerably easier than others. The reward function gives a constant negative reward outside of this target area and only provides a reward gradient inside of the target area. The reward function is defined as

$$r(\boldsymbol{s}, \boldsymbol{\theta}) = \begin{cases} -\frac{10{,}000}{d_{\boldsymbol{s}}} ||\boldsymbol{s} - \boldsymbol{b}_{\boldsymbol{\theta}}||_2 & \text{if } \boldsymbol{b}_{\boldsymbol{\theta}} \text{ is within the target area of } \boldsymbol{s} \\ -10,000 & \text{otherwise} \end{cases},$$

where $d_{\boldsymbol{s}}$ is the respective maximum distance to the center within the target area of context $\boldsymbol{s}$. Providing a constant reward outside of the target area complicates the problem compared

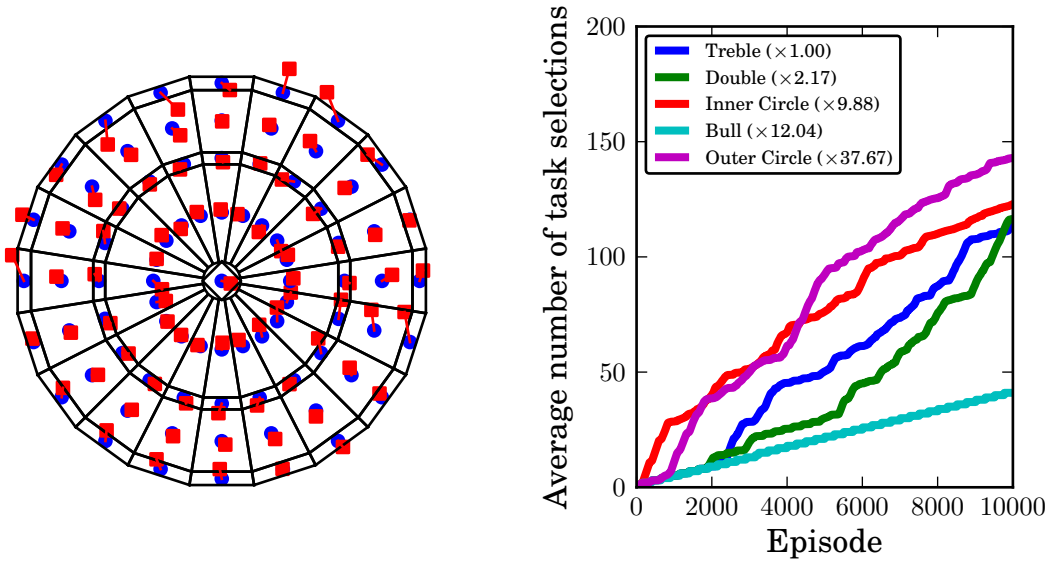


Figure 10: Left: Final upper-level policy. The blue circles mark the center of each field, the red squares that are connected to the corresponding centers with a red line show the position at which the robot arm has actually thrown the ball. Right: Accumulated average number of selections for different task categories. Larger target areas and target areas that are closer to the initial policy are selected more often at the beginning. The relative size in comparison to the smallest target region is given in brackets in the legend.

to the grid setting as no reward gradient helps the agent in improving its policy if it does not hit the target area, within which a reward gradient guides the agent to the center of the field. Thus, the tasks corresponding to larger fields can be considered to be easier since it is more likely that the agent finds a reward gradient during the initial exploration. The agent should thus focus first on these tasks and select the more difficult tasks not until it has improved its contextual policy so far that it is able to hit the corresponding target areas.

We trained with all 81 targets for $50,000$ trials and use the monotonic progress intrinsic reward. The result is displayed in Figure 10: the learned policy hits the "bull", 18 of 20 fields in the inner circle, 15 of 20 trebles, 20 of 20 fields in the outer circle, and 9 of 20 doubles. The overall mean error is 4.62 cm, the median error is 3.77 cm, and the maximum error is 15.96 cm.

Doubles are the most difficult tasks to learn for the agent because they are far away from the initial policy, the target region is small and we cannot transfer much knowledge from neighboring tasks, because they are on the edge of the dartboard. For this reason, some doubles have not been learned very well. In contrast, the trebles can be learned easily, because the solution can be obtained approximately by transferring the solutions of neighboring fields from the inner circle and the outer circle.

In Figure 10 we display which kind of tasks have been selected in the initial learning phase. We can see that the inner circle and the outer circle, which have the greatest target

regions are selected most often in the beginning. The targets from the inner circle are selected even more often than the targets from the outer circle during the first 1,000 episodes even though they are considerably smaller. This is because they are more likely to be reached when exploring from the initial policy which throws at the center of the dartboard. For the same reason trebles are selected more often than doubles at the beginning. After this initial phase, the fields of the outer circle are selected more often because they are now much easier to learn. The bull is so rarely selected because the initial policy will already generate a very good result for this context, hence, improvement is hardly possible.

## 6. Conclusion and Outlook

We have considered the problem of active task selection in contextual policy search. The hypothesis investigated in this paper is that learning all tasks with the same priority in a round robin or random fashion is not always optimal, in particular if the tasks have different characteristics which make some of them more difficult than others. We have proposed an active task-selection heuristic based on the non-stationary bandit algorithm D-UCB which learns to select tasks that have a large intrinsic reward. These intrinsic rewards can be considered as proxies for the actual learning progress which encourage the agent to engage in those tasks in which its performance is increasing the most. The underlying model of the learning process assumes that each task has a different intrinsic expected learning progress which might change abruptly in the context space and that knowledge can be transferred between neighboring contexts.

Our empirical results have shown that active task selection can make a considerable difference for the learning speed of contextual policy search. In general, we found that a task-selection method should explore in the beginning, then focus on several easy tasks to acquire some initial procedural knowledge, thereupon transfer knowledge to similar but more difficult tasks, and concentrate in the end on those tasks that have not been learned yet. Some of the proposed intrinsic reward heuristics provide a considerable advantage over round robin or uniform random selection in our empirical evaluations on a contextual function maximization problem and a simulated robotic ball throwing experiment.

We restricted ourselves to a discrete set of context vectors for training. In the future it would be interesting to explore not only a discrete set of tasks but a continuous range of task parameters such as arbitrary rather than grid-based target positions for ball throwing as in Section 5.2.3. This could be modeled as a non-stationary continuum-armed bandit problem. However, the non-stationary version of the continuum-armed bandit problem (Agrawal, 1995a) has not been explored thoroughly yet.

Moreover, it would also be desirable to stop sampling of tasks for which the policy has reached an acceptable level of performance, that is, tasks that can be regarded as solved. Since the expected learning progress becomes very small in such tasks, an active context selection mechanism should typically learn this automatically. However, since D-UCB assumes non-stationary rewards, it will continue to sample such tasks as it assumes that the expected reward could increase again. Active context selection algorithms which take the property of the learning progress to converge to zero in the long run into account could thus be candidates for improving over D-UCB.

The goal of the proposed method is to reduce the wear and tear of (possibly robotic) agents as well as the human intervention during learning. Selecting the context actively typically requires some mechanism that sets the environment in the requested context. To limit the amount of human intervention, it is very desirable that the agent can set the desired context on its own. This is typically trivial in simulation. In real-world problems, it can often be achieved by means of previously learned or hard-coded skills. In case that a human supervisor needs to produce the requested context, the feasibility of active context selection depends on the amount of work imposed on the human supervisor and thus, on the specific task.

Among the kind of tasks for which we consider active contextual policy search to be promising are: various kind of goal-directed reaching, hitting and throwing problems like for example ball throwing, darts and hockey. Moreover, scenarios in which an agent can select from a small set of predefined contexts, for example grasping one object from a set of objects with varying size and shape, are promising. Another scenario could be that a robot has to learn how to walk on a restricted set of different terrain surfaces, where the context consists of the properties of the surfaces.

In the future, within the research we will conduct in the context of the project "Behaviors for Mobile Manipulation" (BesMan),[2] we plan to evaluate the proposed approach on different robotic target platforms, among others the humanoid robot AILA. One of the challenges for this is to bridge the simulation-reality gap: multiple executions of the same policy often have significantly different outcomes on actual robots like AILA due to different initial states and other unobserved properties. Thus, the learning approach needs to be able to deal with noise in the executions.

## Acknowledgments

This work was supported through two grants of the German Federal Ministry of Economics and Technology (BMWi, FKZ 50 RA 1216 and FKZ 50 RA 1217). In particular, we would like to thank Yohannes Kassahun for helpful comments.

## Appendix A. Implementation of Contextual REPS

Contextual relative entropy policy search has been explained in detail by Deisenroth et al. (2013) and Kupcsik et al. (2013). We summarize C-REPS in Algorithm 2. A stochastic policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$ is used to generate low-level controllers $\pi_{\boldsymbol{\theta}}$ for a context $\boldsymbol{s}$. In this paper, we use a Gaussian policy with linear mean (Deisenroth et al., 2013) so that $\boldsymbol{\omega} = (\boldsymbol{W}, \boldsymbol{\Sigma})$. After collecting $N$ experience tuples $(\boldsymbol{s}_i, \boldsymbol{\theta}_i, r(\boldsymbol{s}_i, \boldsymbol{\theta}_i))$ (lines 2-6), C-REPS determines a weight $d_i$ for each experience based on the context-dependent baseline $V(\boldsymbol{s}) = \boldsymbol{v}^T\varphi(\boldsymbol{s})$ (lines 7-10). With the obtained weights, C-REPS updates the policy through weighted maximum likelihood (lines 11-17).

Implementing C-REPS is not straightforward. In our experiments and for our reward functions it was sometimes numerically unstable. To improve the reproducibility of the

2. See `http://robotik.dfki-bremen.de/en/research/projects/besman-1.html` for more information.

**Algorithm 2** Contextual relative entropy policy search (C-REPS)

**Require:** $\epsilon$: maximum Kullback-Leibler divergence between two successive policy distributions; $\varphi(\boldsymbol{s})$: extracts features from the context $\boldsymbol{s}$; $N$: number of samples per update; $\boldsymbol{W}$: initial weights of upper level policy; $\boldsymbol{\Sigma}$: initial covariance of upper level policy distribution

1: **while** not converged **do**
2:  **for** $i \in \{1, \ldots, N\}$ **do**
3:  Select context $\boldsymbol{s}_i$ # For example, based on *some* specified task-selection method
4:  $\boldsymbol{\theta}_i \sim \mathcal{N}(\boldsymbol{W}^T \varphi(\boldsymbol{s}_i), \boldsymbol{\Sigma})$ # Draw policy parameters
5:  Obtain $r(\boldsymbol{s}_i, \boldsymbol{\theta}_i)$ # Return of policy $\pi_{\boldsymbol{\theta}_i}$ in environment with context $\boldsymbol{s}_i$
6:  **end for**
7:  Solve constrained optimization problem $[\eta, \boldsymbol{v}] = \arg\min_{\eta', \boldsymbol{v}'} g(\eta', \boldsymbol{v}')$, s.t. $\eta > 0$

$$g(\eta, \boldsymbol{v}) = \eta\epsilon + \boldsymbol{v}^T \hat{\varphi} + \eta \log \left( \sum_{i=1}^{N} \frac{1}{N} \exp \left( \frac{r(\boldsymbol{s}_i, \boldsymbol{\theta}_i) - \boldsymbol{v}^T \varphi(\boldsymbol{s}_i)}{\eta} \right) \right)$$

8:  **for** $i \in \{1, \ldots, N\}$ **do**
9:  $d_i \leftarrow \exp\left( \frac{r(\boldsymbol{s}_i, \boldsymbol{\theta}_i) - \boldsymbol{v}^T \varphi(\boldsymbol{s}_i)}{\eta} \right)$ # Determine weight for each experience tuple
10:  **end for**
11:  **for** $i \in \{1, \ldots, N\}$ **do** # Prepare policy update
12:  $\boldsymbol{\Phi}_i = \varphi(\boldsymbol{s}_i)^T$
13:  $\boldsymbol{\Theta}_i = \boldsymbol{\theta}_i^T$
14:  **end for**
15:  $\boldsymbol{D} = \begin{pmatrix} d_1 & & 0 \\ & \ddots & \\ 0 & & d_N \end{pmatrix}$
16:  $\boldsymbol{W}_{\text{new}} \leftarrow (\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Theta}$ # Update policy mean
17:  $\boldsymbol{\Sigma}_{\text{new}} \leftarrow \frac{\sum_i d_i}{\left(\sum_i d_i\right)^2 - \sum_i d_i^2} (\boldsymbol{\Theta} - \boldsymbol{\Phi}\boldsymbol{W})^T \boldsymbol{D} (\boldsymbol{\Theta} - \boldsymbol{\Phi}\boldsymbol{W})$ # Update exploration covariance
18: **end while**

results, we explain some of the modifications that we had to make in addition to the log-sum-exp trick (in line 7) that is already mentioned by Deisenroth et al. (2013).

The weighted least squares problem is sometimes ill-conditioned. Hence, we added the regularization term $\lambda \boldsymbol{I}$ so that

$$\boldsymbol{W}_{\text{new}} \leftarrow (\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1} \boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Theta}.$$

In our experiments, we use $\lambda = 10^{-4}$.

At the end of the learning process, $\eta$ sometimes becomes very small which causes numerical problems. It is helpful to use an other lower bound $\eta_{\min}$, so that $\eta > \eta_{\min}$, for example $\eta_{\min} \in (10^{-6}, 10^{-4})$.

For very large negative rewards the weights $d_i$ are sometimes so small that they cannot be represented properly with 64-bit floating-point numbers. But we can replace $d_i$ by the

softmax-like term

$$\overline{d}_i = \frac{d_i}{\sum_j d_j} = \frac{\exp\left(\frac{r(\boldsymbol{s}_i,\boldsymbol{\theta}_i)-\boldsymbol{v}^T\varphi(\boldsymbol{s}_i)}{\eta}\right)}{\sum_j \exp\left(\frac{r(\boldsymbol{s}_j,\boldsymbol{\theta}_j)-\boldsymbol{v}^T\varphi(s_j)}{\eta}\right)},$$

which does neither change the weights of the linear policy $\boldsymbol{W}_{\text{new}}$, nor the covariance $\boldsymbol{\Sigma}_{\text{new}}$. Softmax can be implemented numerically stable:

$$\overline{d}_i = \frac{\exp\left(\frac{r(\boldsymbol{s}_i,\boldsymbol{\theta}_i)-\boldsymbol{v}^T\varphi(\boldsymbol{s}_i)}{\eta}\right)}{\sum_j \exp\left(\frac{r(\boldsymbol{s}_j,\boldsymbol{\theta}_j)-\boldsymbol{v}^T\varphi(s_j)}{\eta}\right)} = \frac{\exp\left(\frac{r(\boldsymbol{s}_i,\boldsymbol{\theta}_i)-\boldsymbol{v}^T\varphi(\boldsymbol{s}_i)}{\eta} - m\right)}{\sum_j \exp\left(\frac{r(\boldsymbol{s}_j,\boldsymbol{\theta}_j)-\boldsymbol{v}^T\varphi(s_j)}{\eta} - m\right)},$$

where $m = \max_i \frac{r(\boldsymbol{s}_i,\boldsymbol{\theta}_i)-\boldsymbol{v}^T\varphi(\boldsymbol{s}_i)}{\eta}$.

In addition, for the experiments in Section 5.2 we use a kind of importance sampling to reuse experience tuples from previous rollouts: for every task we maintain a memory of the best $N/K$ rollouts, where $N$ is the maximum number of samples for a policy update and $K$ is the number of tasks. Note that this might have a negative effect if there are multiple local optima that are separated by regions with low return in some context $\boldsymbol{s}$.

## References


Rajeev Agrawal. The continuum-armed bandit problem. *SIAM Journal on Control and Optimization*, 33(6):1926–1951, 1995a.

Rajeev Agrawal. Sample mean based index policies with O(log n) regret for the multi-armed bandit problem. *Advances in Applied Probability*, 27(4):1054–1078, 1995b.

Adrien Baranes and Pierre-Yves Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1): 49–73, 2013.

Andrew G. Barto, Satinder Singh, and Nuttapong Chentanez. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the 3rd International Conference of Developmental Learning*, pages 112–119, LaJolla, CA, USA, 2004.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th International Conference on Machine Learning*, pages 41–48, 2009.

Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

William S. Cleveland and Susan J. Devlin. Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83 (403):596–610, 1988.

Bruno Castro da Silva, George Konidaris, and Andrew G. Barto. Learning parameterized skills. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Bruno Castro da Silva, George Konidaris, and Andrew Barto. Active learning of parameterized skills. In *Proceedings of the 31st International Conference on Machine Learning*, Beijing, China, 2014.

Christian Daniel, Gerhard Neumann, and Jan Peters. Hierarchical relative entropy policy search. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics, AISTATS 2012*, pages 273–281, 2012.

Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1–2):328–373, 2013.

Aurelélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In *Proceedings of the 22nd International Conference on Algorithmic Learning Theory*, pages 174–188, 2011.

Jacqueline Gottlieb, Pierre-Yves Oudeyer, Manuel Lopes, and Adrien Baranes. Information-seeking, curiosity, and attention: computational and neural mechanisms. *Trends in Cognitive Sciences*, 17(11):585–93, 2013.

Caglar Gulcehre and Yoshua Bengio. Knowledge matters: Importance of prior information for optimization. In *International Conference on Learning Representations*, 2013.

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, 2013.

S. Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

Jens Kober and Jan Peters. Policy search for motor primitives in robotics. *Machine Learning*, 84(1–2):171–203, 2011.

Jens Kober, Andreas Wilhelm, Erhan Oztop, and Jan Peters. Reinforcement learning to adjust parametrized motor primitives to new situations. *Autonomous Robots*, 33(4):361–379, 2012.

Levente Kocsis and Csaba Szepesvári. Discounted UCB. In *2nd PASCAL Challenges Workshop*, 2006.

Klas Kronander, Mohammad S. M. Khansari-Zadeh, and Aude Billard. Learning to control planar hitting motions in a minigolf-like task. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 710–717, 2011.

Andras G. Kupcsik, Marc Peter Deisenroth, Jan Peters, and Gerhard Neumann. Data-efficient generalization of robot skills with contextual policy search. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2013.

Jan Hendrik Metzen, Alexander Fabisch, Lisa Senger, José de Gea Fernández, and Elsa Andrea Kirchner. Towards learning of generic skills for robotic manipulation. *Künstliche Intelligenz*, 2013. doi: 10.1007/s13218-013-0280-1.

Katharina Mülling, Jens Kober, and Jan Peters. A biomimetic approach to robot table tennis. *Adaptive Behavior*, 19(5):359–376, 2011.

Katharina Mülling, Jens Kober, Oliver Krömer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. *International Journal of Robotics Research*, 32(3), 2013.

Pierre-Yves Oudeyer and Frederic Kaplan. Intelligent adaptive curiosity: a source of self-development. In Luc Berthouze, Hideki Kozima, Christopher G. Prince, Giulio Sandini, Georgi Stojanov, G. Metta, and C. Balkenius, editors, *Proceedings of the 4th International Workshop on Epigenetic Robotics*, pages 127–130. Lund University Cognitive Studies, 2004.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

Jan Peters, Katharina Mülling, Jens Kober, Duy Nguyen-Tuong, and Oliver Krömer. Robot skill learning. In *Proceedings of the 20th European Conference on Artificial Intelligence*, pages 40–45, 2012.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

Paul Ruvolo and Eric Eaton. Scalable lifelong learning with active task selection. In *Proceedings of the AAAI 2013 Spring Symposium on Lifelong Machine Learning*, 2013.

Burr Settles. Active Learning Literature Survey. Technical Report 1648, University of Wisconsin–Madison, 2010.

Joshua B. Tenenbaum, Vin De Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

Aleš Ude, Andrej Gams, Tamim Asfour, and Jun Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics*, 26 (5):800–815, 2010.

Robert W. White. Motivation reconsidered: the concept of competence. *Psychological review*, 66:297–333, 1959.

Malte Wirkus, José de Gea Fernández, and Yohannes Kassahun. Realizing target-directed throwing with a real robot using machine learning techniques. In *Proceedings of the 5th International Workshop on Evolutionary and Reinforcement Learning for Autonomous Robot Systems*, pages 37–43, 2012.