**Künstliche Intelligenz manuscript No.**
(will be inserted by the editor)

# Accounting for Task-Difficulty in Active Multi-Task Robot Control Learning

**Alexander Fabisch*** · **Jan Hendrik Metzen*** · **Mario Michael Krell** · **Frank Kirchner**

Received: date / Accepted: date

**Abstract** Contextual policy search is a reinforcement learning approach for multi-task learning in the context of robot control learning. It can be used to learn versatilely applicable skills that generalize over a range of tasks specified by a context vector. In this work, we combine contextual policy search with ideas from active learning for selecting the task in which the next trial will be performed. Moreover, we use active training set selection for reducing detrimental effects of exploration in the sampling policy. A core challenge in this approach is that the distribution of the obtained rewards may not be directly comparable between different tasks. We propose the novel approach PUBSVE for estimating a reward baseline and investigate empirically on benchmark problems and simulated robotic tasks to which extent this method can remedy the issue of non-comparable reward.

**Keywords** Contextual Policy Search · Multi-task Learning · Active Learning

## 1 Introduction

An artificial system situated in a complex real world environment which is supposed to act autonomously on a long-term mission will necessarily be confronted with situations that have not been foreseen at its design time. Thus, a means for adapting to such unforeseen situations is a vital component for any long-term autonomous systems. One of the most promising technical means for this is machine learning, which allows learning and adapting models of the system and its environment, acquiring and improving system behaviors and control policies, and detecting anomalies and system failure. However, learning in an autonomous system cannot be divided into a training and a deployment phase since this would limit the system's capability of adapting to unforeseen situations. Thus, machine learning in long-term autonomous systems needs to be a "lifelong learning" approach [17, 22].

In contrast to classical offline, batch machine learning, where the goal of learning is to derive inductively prediction and decision rules from a fixed, often manually curated training set for a clearly defined problem, lifelong learning imposes additional challenges: (1) due to possibility of unforeseen situations, the problem domain is not clearly defined and the system needs to be able to facilitate transfer of learned knowledge from expected to unexpected situations. (2) An autonomous system must not rely on an externally curated training set since the necessity to learn might arise in situations where no external help or teacher is available. (3) The system must be able to store, reuse, and combine learned knowledge such that it can efficiently address novel situations. We consider transfer and multi-task learning approaches as suitable approaches to challenge (1) and active learning as promising approach to challenge (2). Challenge (3) might be addressed by hierarchical learning approaches.

In this work, we focus on robot control learning, specifically its active and multi-task learning aspects.

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).
*Authors AF and JHM contributed equally.

Alexander Fabisch†, Jan Hendrik Metzen†,⋆, Mario Michael Krell†, Frank Kirchner†,⋆
† Robotics Group, Universität Bremen, Bremen, Germany
⋆ Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI), Bremen, Germany
E-mail: {afabisch,jhm}@informatik.uni-bremen.de,krell@uni-bremen.de,frank.kirchner@dfki.de

We present a review of works in Section 2. In Section 3, we propose the novel approach PUBSVE for addressing a challenge arising in multi-task control learning: the incommensurability of performance in different tasks. In Section 4, we present empirical evidence that this approach can improve the stability of multi-task control learning methods by allowing to select the training set of the policy update. Section 5 presents results indicating that PUBSVE can also improve the performance of active multi-task control learning.

## 2 Review of Active Multi-Task Control Learning

Thrun and Schwartz [21] suggested the notion of *lifelong learning* in the context of supervised learning for object recognition. In lifelong learning, a learner experiences a sequence of different but related tasks. Due to this relatedness, learned knowledge can be transferred across multiple learning tasks, which can allow generalizing more accurately from less training data. The concept of lifelong learning was extended to RL by, e.g., Sutton, Koop, and Silver [19] and by Ring [13] under the term "continual learning". Two important aspects of lifelong learning, on which we focus in this work, are multi-task/transfer learning and active learning.

*Multi-task and transfer learning* in robot control learning is based on the idea that a control policy can be represented hierarchically, for instance in a combination of a lower-level policy and an upper-level policy. The lower-level policy, e.g., a dynamical movement primitive (DMP) [4], typically encodes a behavior for a specific task in a parameter vector while the upper-level policy encodes a mapping from task to a parametrization of the lower-level policy. Two different kinds of approaches for learning such a two-layer, hierarchical policy have been proposed; namely transfer-learning approaches, where the two layers are learned consecutively, and multi-task learning approaches, where both layers are learned jointly.

One transfer learning approach was proposed by da Silva [16]. In this approach, the PoWER [5] method is used to learn near-optimal low-level policies for each of a set of tasks separately. Based on this, a training set is constructed, which consists of parameters for the near-optimal low-level policies in different tasks. Thereupon, a regression algorithm is used to infer a deterministic upper-level policy, the so-called *parameterized skill*, which generalizes over the entire task space. The authors considered the problem of simulated 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.

A disadvantage of this approach is that transfer is only possible once a set of near-optimal low-level policies have been learned. In contrast, multi-task learning approaches try to achieve positive transfer between tasks already for suboptimal low-level policies and be more sample-efficient accordingly. One such approach is cost-regularized kernel regression (CrKR) [6], in which a non-parametric upper-level policy is learned based on cost-regularized regression. CrKR has been used to learn throwing movements as well as table tennis. Another algorithm that can be used to learn the upper-level policy is contextual relative entropy policy search (C-REPS) [8]. REPS [12] 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 this new policy and the policy from which the training data has been sampled. Bounding this divergence enforces that the new policy is not too different from the data generating 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 multi-task learning scenario. A further alternative is Variational Inference for Policy Search (VIP) [11], which is also based on the Kullback-Leibler divergence but is cost-averse rather than reward-attracted and can thus deal better with multi-modal or non-concave target distributions.

An interesting extension of transfer and multi-task learning is a combination with *active learning* [14]. In this context, active learning refers to a setting where the agent can select autonomously in which task it would like to perform the next trial during learning rather than being confronted with an externally selected task. This has the advantage that the agent might focus on tasks in which its current performance is suboptimal but it is confident that it may improve its performance by further experience.

Da Silva et al. [15] proposed an extension of their parameterized skill approach with a novel criterion for skill selection. In this criterion, the performance in a task is modeled using Gaussian process regression with a spatiotemporal kernel which addresses the inherent non-stationarity of tracking the skill performance. The next task is chosen based on this estimate of the task performance such that the maximum expected improvement in this tasks performance would be obtained if the outcome of learning this task is assumed to be an optimistic upper bound.

Recently, Fabisch and Metzen proposed an active approach for multi-task learning [2], which is based on heuristically optimizing the expected learning progress. For this, different proxies of the learning progress are

defined and non-stationary multi-armed bandit learning is used for task selection based on an intrinsic reward that is related to this learning progress proxies.

An important yet largely unexplored issue in active multi-task robot control learning is the incommensurability of performance in different tasks, i.e., how a learning system can account for the relative (unknown) *difficulty* of a task: for instance, if a relatively small reward is obtained when executing a specific low-level policy in a task, is it because the low-level policy is not well adapted to the task or because the task is inherently more difficult than other tasks? Giving a principled answer to this question promises performance gains in active multi-task learning. We present one approach for estimating the task-difficulty in the next section.

## 3 Estimating Task Difficulty

In this section, we introduce contextual policy search more formally and discuss the issue of task incommensurability. We then outline an approach to make the rewards obtained in different tasks comparable by estimating an upper bound and a typical value of the possible rewards in different contexts.

### 3.1 Contextual Policy Search

A probabilistic formulation of contextual policy search has been presented by Deisenroth et al.[1]; however, in this work, we restrict ourselves to deterministic, unimodal, continuous upper-level policies $\pi : S \mapsto A$, where $S \subseteq \mathbb{R}^M$ is the space of contexts, where we use the terms context and task interchangeably, and $A \subseteq \mathbb{R}^N$ is the feasible set of, e.g., lower-level policy parameters or actions. In this setting, contextual policy search can be considered as a contextual function optimization problem:

$$\pi(\boldsymbol{s}) = \arg\max_{\boldsymbol{a} \in A} \mathbb{E}\{r(\boldsymbol{s}, \boldsymbol{a})\},$$

where $\mathbb{E}\{r(\boldsymbol{s}, \boldsymbol{a})\}$ is the expected reward for $\boldsymbol{a}$ in context $\boldsymbol{s}$. Thus, we are looking for $\pi$ that generates an optimal $\boldsymbol{a} \in A$ for a specific context $\boldsymbol{s}$. In practice, a close-to-optimal $\boldsymbol{a}$ is sufficient. Contextual function optimization can be regarded as a family of standard optimization problems parametrized by the context parameter.

Samples $(\boldsymbol{s}, \boldsymbol{a}, r(\boldsymbol{s}, \boldsymbol{a}))$ are required for contextual policy search to update the estimate of $\pi$, i.e. we perform trials by testing parameters $\boldsymbol{a}$ in context $\boldsymbol{s}$ and observing the corresponding reward $r(\boldsymbol{s}, \boldsymbol{a})$. Popular approaches are based on cost-regularized regression [6] or weighted maximum likelihood [8]. These algorithms internally weight samples based on obtained rewards: higher rewards result in higher weights. An issue with this approach is that rewards are often incommensurable between contexts: the maximum reward value in one context could be far from optimal in an other context. Algorithms like C-REPS [8] learn a baseline, which maps contexts to a "typical" reward level to make rewards comparable. What is considered a typical reward level depends on an additional temperature parameter and the amount of exploration of the agent. A related issue arises in *active* contextual policy search, in which the learning agent can actively control in which contexts additional samples for the policy update are acquired. As proposed by Fabisch and Metzen [2], performing trials in contexts in which the learning progress is comparatively large can speed up contextual policy search. As the learning progress is related to an increase in reward, the issue of incommensurable rewards needs to be addressed in such an active learning approach as well.

In the following, we propose a novel approach for addressing task incommensurability. This approach allows to estimate an upper boundary of the achievable reward and a typical reward range and thus, allows to make rewards comparable by normalizing them. We can then use these normalized rewards for weighting experience samples in the policy update (see Section 4) and to identify contexts in which we can make greater learning progress (see Section 5).

### 3.2 Positive Upper Boundary Support Vector Estimation

We are given a set of observations $\mathcal{D} = \{(\boldsymbol{s}_i, r_i)\}_{i=1}^n$. We assume that $r_i$ depends on $\boldsymbol{s}_i$ via $r_i = V(\boldsymbol{s}_i) - e_i$, where $e_i$ is some noise term. In contrast to standard regression problems, we assume $e_i \geq 0$, i.e., we always observe values $r_i$ which are less than or equal to the true function value $V(\boldsymbol{s}_i)$. This model is appropriate for instance when $V(\boldsymbol{s}_i)$ is a value function, that is when it returns the maximum reward possible in a context $\boldsymbol{s}_i$, i.e., $V(\boldsymbol{s}_i) = \max_{\boldsymbol{a} \in A} r(\boldsymbol{s}_i, \boldsymbol{a})$, and $r_i = r(\boldsymbol{s}_i, \boldsymbol{a}_i)$ are the actual, noise-free rewards obtained by a learning agent which often selects suboptimal $\boldsymbol{a}_i$.

We are interested in inferring the function $V$ from observations $\mathcal{D}$, i.e., learn an estimate $\hat{V}$ of $V$. One natural constraint on the estimate is that $\hat{V}(\boldsymbol{s}_i) \geq r_i$, i.e., $\hat{V}$ shall be an *upper boundary* on $\mathcal{D}$. Since this constraint does not uniquely determine the estimate, we add two additional objectives: (a) $\hat{V}$ should be smooth, i.e., similar inputs $\boldsymbol{s}_1 \approx \boldsymbol{s}_2$ should have similar values $V(\boldsymbol{s}_1) \approx V(\boldsymbol{s}_2)$. (b) $\hat{V}$ should be less or equal to $V$, i.e.,

it should be a pessimistic estimate with a bias towards too small values. Since $V$ corresponds to the value of the optimal policy, the current policy will always obtain rewards smaller or equal to $V$ and a pessimistic estimate $\hat{V}$ will necessarily be closer to the level of performance of the current policy than a too optimistic estimate with the same overall error $|V - \hat{V}|$.

Since a linear model of $\hat{V}$ is often too restrictive, we use a non-parametric, kernelized model for $\hat{V}$, e.g., $\hat{V}(\boldsymbol{s}) = b + \sum_{i=1}^{n} \alpha_i k(\boldsymbol{s}_i, \boldsymbol{s})$ with offset $b$ and RBF kernel $k(\boldsymbol{s}_i, \boldsymbol{s}_j) = \exp(-\gamma ||\boldsymbol{s}_i - \boldsymbol{s}_j||^2)$ for bandwidth $\gamma$. This model allows both a very smooth (constant) upper boundary by setting $\alpha_i = 0$ and $b \geq \max_{i'} y_{i'}$ and a very pessimistic upper boundary for large $\gamma$ and small $b$. We propose now a new method, the *positive upper boundary support vector estimation* (PUBSVE), for learning such a model of the upper boundary. Without loss of generality we assume that all $r_i$ are positive (because we subtract $\min_{i'} r_{i'}$ for normalization). This implies that $b \geq 0$. The PUBSVE has the following objective

$$\min_{\boldsymbol{\alpha}, b} \frac{1}{2} \sum_{i,j} \boldsymbol{\alpha_i} \boldsymbol{\alpha_j} k(\boldsymbol{s}_i, \boldsymbol{s}_j) + \frac{H}{2} b^2$$

$$\text{subject to: } b + \sum_{j} \boldsymbol{\alpha_j} k(\boldsymbol{s}_i, \boldsymbol{s}_j) \geq r_i \, \forall i \, .$$

$H$ is a hyperparameter which allows to handle the trade-off between a very pessimistic upper boundary ($b \to 0$ when $H \to \infty$) and a constant upper boundary ($b \to \max_{i'} r_{i'}$ and $\boldsymbol{\alpha} \to 0$ when $H \to 0$). Typically, $H = 100 \gg 0$ gives good results empirically and is used in our experiments. The RBF's bandwidth $\gamma$ controls the generalization between similar data: the greater $\gamma$ is the less similarity between similar inputs is assumed and the more local will be the generalization. The model has similarities to support vector machines and allows to use related implementation techniques [9,18]. Further details are provided by Krell [7].

### 3.3 Incremental Reward Normalization

In this section, we describe how PUBSVE allows to normalize the obtained reward: for a given context $\boldsymbol{s}$, we map the PUBSVE prediction $\hat{V}(\boldsymbol{s})$ onto 1 and the context's typical reward level $\tilde{r}(\boldsymbol{s})$ onto 0. This can be achieved via $\overline{r_i} = \frac{r_i - \tilde{r}(\boldsymbol{s}_i)}{\hat{V}(\boldsymbol{s}_i) - \tilde{r}(\boldsymbol{s}_i)}$. Estimating a lower boundary on the rewards cannot be performed analogously to estimating an upper boundary since contextual policy search tries to avoid sampling regions of small reward and in some problems, no lower boundary exists. For that reason, we normalize rewards based on their estimated typical reward level $\tilde{r}(\boldsymbol{s})$ obtained from a standard support vector regression (SVR) model trained on $\mathcal{D}$. The reason for using SVR is that it is relatively robust to outliers.

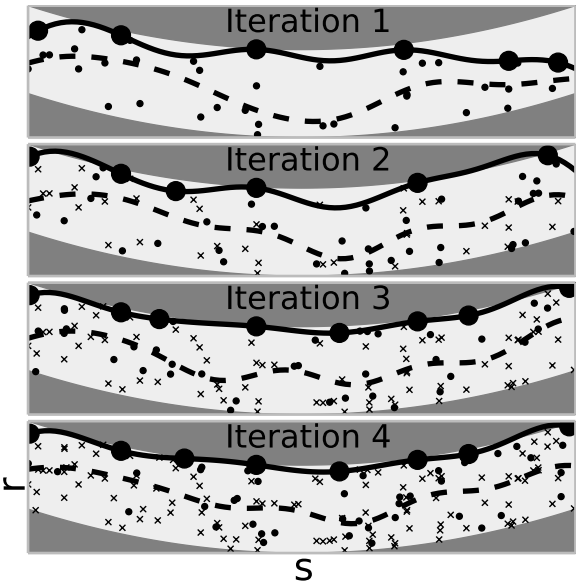



**Fig. 1** Four iterations of the incremental learning of upper boundary $\hat{V}$ (solid line) and medium reward level $\tilde{r}$ (dashed line): for each update of the PUBSVE, the new samples (black dots) and the support vectors of PUBSVE from the previous iteration (highlighted by large circles) are used as training set. All previous samples that are not used for the incremental training are displayed as small crosses. $\tilde{r}$ is learned using an SVR on the whole data. New samples are drawn uniform randomly from the white background area.

Since the PUBSVE needs to be updated once new samples arrive, an incremental training procedure is desirable. For this, instead of training the PUBSVE on the whole set $\mathcal{D} = \{(\boldsymbol{s}_i, r_i)\}_{i=1}^{n}$, we can update it incrementally [20], where we forget every old example $(\boldsymbol{s}_i, r_i)$ except the support vectors $\boldsymbol{s}_i$ with $\alpha_i > 0$, collect new samples, and use the new samples and the retained support vectors to update $\hat{V}$. Note that this heuristic does not guarantee that the same results are obtained as in the non-iterative PUBSVE; however, it provides good results in practice. For the SVR model, we don't perform an incremental update but keep the computational cost limited by subsampling the set $\mathcal{D}$ to a size of 5000. An illustration of this approach is given in Figure 1.

## 4 Training Data Selection

Contextual policy search methods like C-REPS perform a search through the space of policies where updates of the policy are done such that one moves in the direction of increasing expected return while, at the same time, bounding the loss of information (measured using relative entropy) between the observed data distribu-

tion and the data distribution generated by the new policy [12]. Bounding the information loss results effectively in "small steps" in the policy space and avoids that a large step is taken into an unknown area of the policy space, which might contain policies whose execution is dangerous for a robot.

It remains to define the observed data distribution. Possible choices are the samples drawn from the last policy or from the last $K$ policies [1]. The former choice has the disadvantage that all prior experience not generated by the last policy is effectively "forgotten" and the latter choice typically slows down learning initially as the new policy is enforced for $K$ iterations to stay close to the data generated by the initial policies, which are typically behaving badly and strongly exploratory. In principle, it would be appealing if the old data distribution would consist of the "best" samples from the entire history as this enforces the new policy to stay close to what has worked well in the past. Note that one potential risk of this strategy is premature convergence to local optima; it is thus important to maintain a sufficient level of exploration.

As discussed above, the main challenge for determining the "best" samples is the incommensurability of rewards in different contexts. Fabisch and Metzen [2] proposed an approach to a simplified setup, in which only a discrete set of $K$ contexts exists, by storing the best $N/K$ obtained rewards for each context and use these for the update of the upper-level policy, where $N$ is the number of examples for each update. As the method proposed in Section 3.2 allows to make the rewards from different contexts comparable, we can now extend this training data selection to continuous context spaces by selecting the training examples with the highest normalized reward for training. However, to avoid premature convergence and to be more robust to errors in the estimate $\hat{V}$, we suggest to use a softmax for training set selection instead of the maximum, which means that each of the samples $(\boldsymbol{s}_i, \boldsymbol{a}_i, r_i)$ that we observed will be selected with probability

$$p(\boldsymbol{s}_i, \boldsymbol{a}_i, r_i) = \frac{\exp(\tau \overline{r_i})}{\sum_j \exp(\tau \overline{r_j})},$$

where $\tau = 10$ in our experiments and $\overline{r_i}$ is the normalized reward (see Section 3.3). We call the combination of C-REPS with this *active* training set selection approach *aC-REPS*. We compare standard C-REPS and aC-REPS in the following sections.

### 4.1 Benchmark Function

We evaluate both standard C-REPS and aC-REPS on a benchmark function. The solution $\pi$ of the benchmark is a quadratic function and the upper reward boundary $V$ is quadratic as well.

The contextual optimization objective is defined in terms of a function $f$ from the BBOB testbed [3]:

$$r(\boldsymbol{a}, \boldsymbol{s}) = -f(\boldsymbol{a}) + \frac{1}{10M} \boldsymbol{s}^T \boldsymbol{V} \boldsymbol{s}, \quad -5 \leq \boldsymbol{a}_i \leq 5.$$

The parameters $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_N \in \mathbb{R}^{M \times M}, \boldsymbol{V} \in \mathbb{R}^{M \times M}$, and $b_1 \ldots, b_N \in \mathbb{R}$ are generated randomly and scaled so that all context-dependent optima are within $[-\mathbf{5}, \mathbf{5}]$. The optimum of the benchmark function depends on the context via $\boldsymbol{a}_{opt,i}(\boldsymbol{s}) = \frac{1}{100M} \boldsymbol{s}^T \boldsymbol{W}_i \boldsymbol{s} + b_i$. In the following evaluation, we use the Rastrigin function $f_3$ from the BBOB testbed as $f$. The contextual objective function as well as the optimal $\pi$ and a non-optimal solution are displayed in Figure 2 (a).

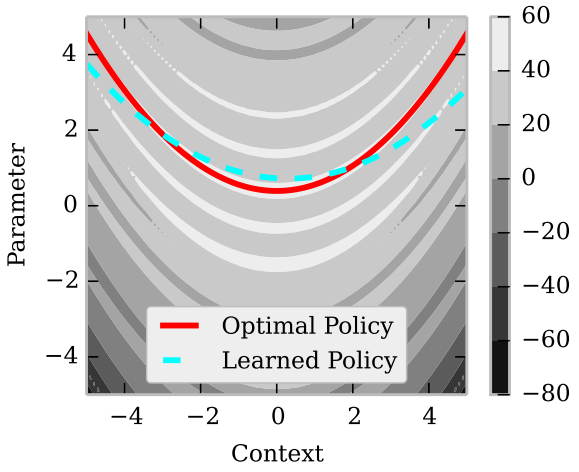
We use 200 samples for each policy update and set the minimum allowed temperature $\eta$ to $10^{-8}$; moreover, we employ quadratic features of the context for the linear upper-level policy, bound the output of the upper-level policy to the interval $[-5, 5]$, and initialize the upper-level policy such that it generates zeros for all inputs. The parameter $\gamma$ for the kernels of the boundary estimations is set to $10\overline{d}$, where $\overline{d}$ is the average distance of the training contexts. The specific choice is often not critical as long as it does not make the model too smooth. We are looking for a configuration that is stable on the one hand and fast on the other hand. To guarantee both, we select critical parameters by a grid search so that the performance after 5,000 episodes is optimized. There are two critical parameters of (a)C-REPS that have to be selected: the number of episodes between policy updates (options: 50, 100, 200) and the upper limit $\epsilon$ on the allowed Kullback-Leibler divergence between successive search distribution (options: 0.1, 0.2, 0.5, 1, 2).

We perform 50 runs with 10,000 episodes. The learning curves are displayed in Figure 2 (b). We can see that both methods reach a similar optimum but the active training set selection ("aC-REPS") is much faster in the beginning. There are configurations of C-REPS that reach the optimum similarly fast but these are unstable so that the algorithm diverges after finding a good solution.

### 4.2 Catapult Domain

The catapult benchmark problem has been introduced by da Silva et al. [15]. It is shown in Figure 3 (a). The goal is to learn an upper-level policy that generates appropriate actions $\boldsymbol{a}_i$, consisting of the angle $\theta_i \in [0, \pi/2]$ and velocity $v_i \in [5, 10]$ of the catapult's shot, such that a specific target-position, the context $\boldsymbol{s}_i \in [2, 10]$, is hit.

(a) Rastrigin




(b) Learning curves

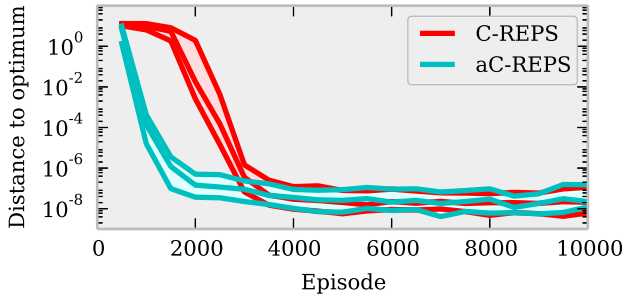
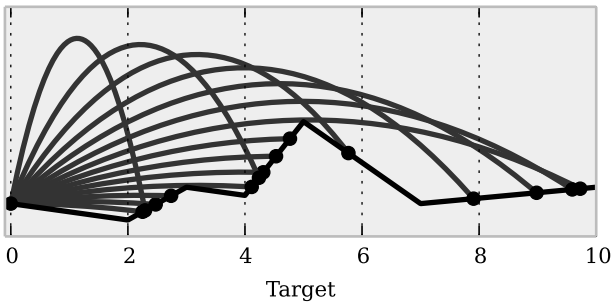



**Fig. 2** (a) Contextual Rastrigin function. The function value is indicated by the background color. (b) Learning curves for standard C-REPS (update after 200 episodes, $\epsilon = 1$) and aC-REPS (update after 100 trials, $\epsilon = 2$). The 3 lines correspond to the quartiles.

(a) Catapult




(b) Comparison of (a)C-REPS after 7,500 episodes

| $\epsilon$ | 0.1 | 0.2 | 0.5 | 1 | 2 |
|---|---|---|---|---|---|
| Update after 50 samples | + | + | + | + | o |
| Update after 100 samples | + | + | + | + | o |
| Update after 200 samples | + | + | o | o | o |

(c) Learning curves

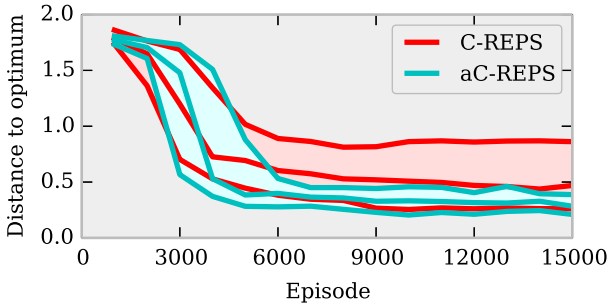



**Fig. 3** (a) Illustration of the catapult problem: the objective is to shoot an object from a catapult situated at position 0 such that it hits a pre-specified target position $\boldsymbol{s}$. Several example trajectories are shown. (b) Comparison of aC-REPS and C-REPS for several configurations with Wilcoxon signed-rank test. "+" means aC-REPS is significantly better and "o" means the difference is not significant. (c) Learning curves for standard C-REPS (update after 200 episodes, $\epsilon = 0.5$) and aC-REPS (update after 200 trials, $\epsilon = 0.2$). The 3 lines correspond to the quartiles.

The target surface is unknown to the agent, which requires him to learn by trial-and-error. The reward of a trial is computed as $r_i = -|\boldsymbol{s}_i - \boldsymbol{s}_h| - 0.5v_i$, where $\boldsymbol{s}_h$ is the position that was actually hit. Because certain target positions such as those behind the top of a hill are more challenging than others, the rewards obtained in different contexts are not directly comparable.

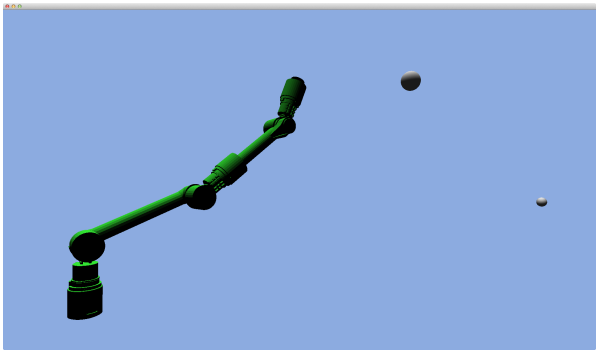
We use 200 samples for each policy update and set the minimum allowed temperature $\eta$ to $10^{-8}$; moreover, we employ a Nyström approximation [23] of an RBF kernel with $\gamma = 10^{-5}$ and 10 components as context features for the linear upper-level policy and initialize the upper-level policy such that it generates velocity 7.5 and angle $\pi/4$ for each context. The optimum parameter configuration is determined after 7,500 episodes similar to Section 4.1.

We perform 50 runs with 15,000 episodes. We found that aC-REPS is more robust with respect to parameter configuration which we demonstrate with a Wilcoxon signed-rank test in Table 3 (b). This property is desirable for learning e.g. with robots because we would otherwise require a lot of episodes to find the optimum configuration. The learning curves in Figure 3 (c) show that both variants of C-REPS improve initially; however, the best configuration of standard C-REPS is not as reliable as aC-REPS because there are some runs that do not improve beyond a moderate level of performance. aC-REPS does not exhibit this problem.

### 4.3 Ball Throwing

We use the simulated arm of the robot Artemis [10] (see Figure 4 (a)) with 6 degrees of freedom to generalize a throwing movement over a target area on the ground. The targets are drawn from $\boldsymbol{s} \in [-3, 3] \times [3, 6]$. We use a dynamical movement primitive (DMP) as de-

(a) Artemis robot arm



(b) Learning curves

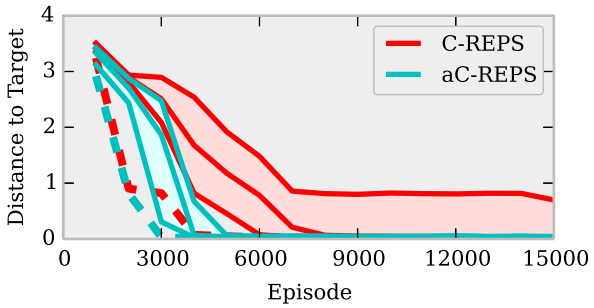



**Fig. 4** (a) Artemis robot arm. (b) Learning curves for standard C-REPS (update after 200 trials, $\epsilon = 2$) and aC-REPS (update after 50 trials, $\epsilon = 2$). The 3 continuous lines correspond to the quartiles. The dashed lines correspond to the best values over all runs.

fined by Ijspeert et al. [4] with an execution time of 200 ms, a control frequency of 250 Hz, and 10 basis functions per dimension to generate the throwing motion in joint space. The ball is attached to the end-effector and is released after 140 ms. An initial throwing motion, which throws into the middle of the target area, has been learned with standard REPS [12]. Only the 60 weights of the DMP are to be adapted.

Each policy update is based on 200 samples and the minimum value allowed for $\eta$ is $10^{-8}$. We employ quadratic features of the context. The optimum configurations are determined with grid search so that we optimize the performance after 7,500 episodes.

In the experiment, we perform 30 independent runs and 15,000 episodes per run. Figure 4 (b) shows the results of the experiment: while aC-REPS learns good policies fast and reliably, standard C-REPS is slower and less reliable which is consistent to the results of Sections 4.1 and 4.2.

## 5 Active Task Selection

In a setting in which the agent is able to actively select the contexts that it explores, selecting each context randomly or equally often is not necessarily the optimal strategy because some contexts might be easier to learn at the beginning and the knowledge that the agent can extract from these contexts can be transferred to similar but more difficult contexts later on. Fabisch and Metzen [2] proposed a solution for the context-selection problem by modeling it as a non-stationary multi-armed bandit problem (MABP) with custom intrinsic reward heuristics that approximate the learning progress.

Some of the evaluated intrinsic reward heuristics use a baseline to estimate the learning progress so that it is comparable between different contexts. This baseline can be an estimate of the upper boundary of the reward in the corresponding context; for instance, the monotonic progress heuristic computes the learning progress depending on the maximum of the previous rewards $b$ as $\max(0, r(\boldsymbol{s}, \boldsymbol{a}) - b)$. We can now replace $b$ by $\hat{V}(\boldsymbol{s})$. Note that we have to select a set of training contexts in this approach.

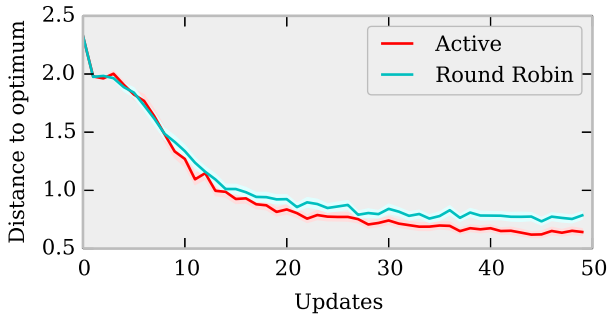
We compare the active context selection method that has been proposed by Fabisch and Metzen [2], using the monotonic progress heuristic and the estimate of the upper reward boundary $\hat{V}(\boldsymbol{s})$ as baseline, with context selection in a fixed order ("Round Robin") in the catapult domain. We use the same parameters for C-REPS as in Section 4.2, $\epsilon = 2$, and the parameters $\gamma = 0.99$, $B = 10$, and $\xi = 10^{-4}$ for the context selection method. The result is displayed in Figure 5. 100 runs with 50 updates are performed and 200 samples are used for each update. The learning curve shows the mean and the standard error over all 100 runs.

We can see that initially, both approaches perform about equal, but in the long run, active context selection learns slightly better policies. Closer inspection shows that these policies are better primarily in the more difficult context behind the hill at $\boldsymbol{s} = 5$. This can be explained by the observation that active context selection focuses more on improving in this area since it is still possible to make progress here.

## 6 Conclusion and Outlook

We have proposed the novel approach PUBSVE for addressing the incommensurability of rewards in multi-task learning with tasks of different difficulty. This approach can be employed in contextual policy search and achieves more robust and faster learning by allowing to select high quality data for the policy update and to select the task of the next trial actively. Potential future work might employ manifold learning approaches to address situations in which policy and value function are non-smooth. Moreover, active contextual policy search

(a) Learning curve




(b) Context-dependent performance

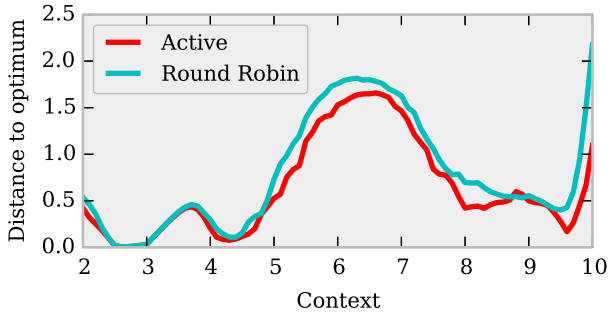



**Fig. 5** Comparison of round-robin and active context selection in catapult benchmark (see Figure 3 (a)). (a) Learning curve. It can be seen that active context selection achieves lower error in the long run. (b) Error of learned policy in different contexts. Active context selection focuses more strongly on hard contexts ($s \in [5, 7]$ correspond to targets behind a hill and $s \approx 10$ are difficult because of the RBF-based policy.)

algorithms that do not require a discretization of the context space are desirable. A combination of PUBSVE with other contextual policy search algorithms such as CrKR or VIP would also be worth investigation.

## References


1. Deisenroth, M.P., Neumann, G., Peters, J.: A survey on policy search for robotics. Foundations and Trends in Robotics **2**(1–2), 328–373 (2013)
2. Fabisch, A., Metzen, J.H.: Active contextual policy search. Journal of Machine Learning Research **15**, 3371–3399 (2014)
3. Hansen, N., Auger, A., Ros, R., Finck, S., Posik, P.: Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In: Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation (2010)
4. Ijspeert, A.J., Nakanishi, J., Hoffmann, H., Pastor, P., Schaal, S.: Dynamical movement primitives: learning attractor models for motor behaviors. Neural Computation **25**(2), 328–373 (2013)
5. Kober, J., Peters, J.: Policy search for motor primitives in robotics. Machine Learning **84**(1–2), 171–203 (2011)
6. Kober, J., Wilhelm, A., Oztop, E., Peters, J.: Reinforcement learning to adjust parametrized motor primitives to new situations. Autonomous Robots **33**(4), 361–379 (2012)
7. Krell, M.M.: Generalizing, Decoding, and Optimizing Support Vector Machine Classification. PhD thesis, University of Bremen, Bremen (2015).
8. Kupcsik, A.G., Deisenroth, M.P., Peters, J., Neumann, G.: Data-efficient generalization of robot skills with contextual policy search. In: Proceedings of the National Conference on Artificial Intelligence (AAAI) (2013)
9. Mangasarian, O.L., Musicant, D.R.: Successive Overrelaxation for Support Vector Machines. IEEE Transactions on Neural Networks **10**, 1032 – 1037 (1998)
10. Manz, M., Sonsalla, R., Hilljegerdes, J., Oekermann, C., Schwendner, J., Bartsch, S., Ptacek, S.: Design of a rover for mobile manipulation in uneven terrain in the context of the spacebot cup. In: Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2014) (2014). http://spacebot.dfki-bremen.de/
11. Neumann, G.: Variational Inference for Policy Search in changing situations. In: Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp. 817–824 (2011)
12. Peters, J., Mülling, K., Altun, Y.: Relative entropy policy search. In: D.P. Fox M. (ed.) Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, pp. 1607–1612. AAAI Press, Atlanta, Georgia, USA (2010)
13. Ring, M.B.: CHILD: a first step towards continual learning. Machine Learning **28**(1), 77–104 (1997)
14. Ruvolo, P., Eaton, E.: Active task selection for lifelong machine learning. In: Twenty-Seventh AAAI Conference on Artificial Intelligence (2013)
15. da Silva, B.C., Konidaris, G., Barto, A.: Active learning of parameterized skills. In: Proceedings of the 31st International Conference on Machine Learning (ICML 2014). Beijing, China (2014)
16. da Silva, B.C., Konidaris, G., Barto, A.G.: Learning parameterized skills. In: Proceedings of the 29th International Conference on Machine Learning (ICML 2012). Edinburgh, Scotland (2012)
17. Silver, D.L., Yang, Q., Li, L.: Lifelong machine learning systems: Beyond learning algorithms. In: 2013 AAAI Spring Symposium Series (2013)
18. Steinwart, I., Hush, D., Scovel, C.: Training SVMs without offset. Journal of Machine Learning Research **12**, 141–202 (2009)
19. Sutton, R.S., Koop, A., Silver, D.: On the role of tracking in stationary environments. In: Proceedings of the 24th International Conference on Machine Learning, pp. 871–878. ACM (2007)
20. Syed, N.A., Liu, H., Sung, K.K.: Handling concept drifts in incremental learning with support vector machines. In: Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining - KDD '99, pp. 317–321. ACM Press, New York, New York, USA (1999)
21. Thrun, S.: Is learning the n-th thing any easier than learning the first? In: Advances in Neural Information Processing Systems, pp. 640–646. The MIT Press (1996)
22. Thrun, S., Mitchell, T.M.: Lifelong robot learning. In: L. Steels (ed.) The Biology and Technology of Intelligent Autonomous Agents, 144, pp. 165–196. Springer Berlin Heidelberg (1995)
23. Williams, C., Seeger, M.: Using the Nyström method to speed up kernel machines. In: Advances in Neural Information Processing Systems 13, pp. 682–688. MIT Press (2001)

**Alexander Fabisch** received his diploma degree in computer science from the University of Bremen in 2012. Since 2012, he works as a researcher at the robotics research group of the University of Bremen. His scientific interests are in the fields of machine learning and black-box optimization with robotic applications.

**Jan Hendrik Metzen** is a researcher at the robotics research group of the University of Bremen and the Robotics Innovation Center (RIC) of the German Research Center for Artificial Intelligence (DFKI) in Bremen. Since 2013 he is leader of the team "Sustained Learning" of the DFKI RIC. He received his diploma and doctoral degree in computer science in 2006 and 2014, respectively. His scientific interests are in the fields of reinforcement learning, transfer learning, and biomedical pattern recognition.

**Mario Michael Krell** received his diploma degree in mathematics in 2009 from the Humboldt University of Berlin. Since 2009, he is with the Robotics Innovation Center (RIC) of the DFKI in Bremen and the robotics research group at the University of Bremen where he received his Dr. rer. nat. in 2015. Currently, he is main developer of the signal processing and classification environment pySPACE. His research areas are support vector machines, theoretical optimization, appliance of optimization theory, signal processing, and machine learning.

**Frank Kirchner** studied computer science and neurobiology at the University Bonn from 1989 until 1994, where he was awarded a Dr. rer. nat. in computer science in 1999. Since 1994 he was senior scientist at the Gesellschaft für Mathematik und Datenverarbeitung (GMD) in Sankt Augustin, which in 1999 became part of the Fraunhofer Institute. Since 1998 he was also a senior scientist at the Department for Electrical Engineering at Northeastern University in Boston where he was first appointed adjunct and then tenure track assistant professor. In 2002, he was appointed a full professor at the University of Bremen and since 2005, he is also director of the Robotics Innovation Center of the DFKI. Since 2013 he is also scientific director of the Brazilian Institute of Robotics (BIR). His scientific interests are in the fields of robotics and AI.