The AAAI-17 Workshop on
Human-Machine Collaborative Learning
WS-17-11

Stochastic Search In Changing Situations

Abbas Abdolmaleki,[1,2,3,5] **David Simões,**[1] **Nuno Lau,**[1]
Luis Paulo Reis,[2,3] **Bob Price,**[5] **Gerhard Neumann**[4]

1: IEETA, DETI, University of Aveiro, Aveiro, Portugal
2: DSI, University of Minho, Braga, Portugal
3: LIACC, University of Porto, Porto, Portugal
4: CLAS, TU Darmstadt, Darmstadt, Germany
5: PARC, Xerox, Palo Alto, USA

Abstract

Stochastic search algorithms are black-box optimizer of an objective function. They have recently gained a lot of attention in operations research, machine learning and policy search of robot motor skills due to their ease of use and their generality. However, when the task or objective function slightly changes, many stochastic search algorithms require complete re-learning in order to adapt thesolution to the new objective function or the new context. As such, we consider the contextual stochastic search paradigm. Here, we want to find good parameter vectors for multiple related tasks, where each task is described by a continuous context vector. Hence, the objective function might change slightly for each parameter vector evaluation. In this paper, we investigate a contextual stochastic search algorithm known as Contextual Relative Entropy Policy Search (CREPS), an information-theoretic algorithm that can learn from multiple tasks simultaneously. We show the application of CREPS for simulated robotic tasks.

Introduction

Stochastic search algorithms are gradient-free black-box optimizers of some performance function dependent on a high-dimensional parameter vector. They directly evaluate the execution of a parameter vector by using the return of an episode. Stochastic search algorithms (Hansen, Muller, and Koumoutsakos 2003; Sun et al. 2009; Stulp and Sigaud 2012; Rückstieß, Felder, and Schmidhuber 2008) typically maintain a search distribution over the parameters that we want to optimise, which is used to create samples of the parameter vector. Subsequently, the performance of the sampled parameters is evaluated. Using the samples and their evaluations, a new search distribution is computed by computing gradient based updates (Sun et al. 2009; Rückstieß, Felder, and Schmidhuber 2008), evolutionary strategies (Hansen, Muller, and Koumoutsakos 2003), the cross-entropy method (Mannor, Rubinstein, and Gat 2003), path integrals (Stulp and Sigaud 2012; Theodorou, Buchli, and Schaal 2010), or information-theoretic policy updates (Kupcsik et al. 2013; Abdolmaleki et al. 2015a). However, many of the previously mentioned algorithms cannot be applied to multi-task learning. In other words, if the task setup or objective function changes slightly, re-learning is needed to adapt the solution to the new situation or the new context. For example, consider optimising the parameters of a humanoid robot controller to kick a ball. Once the characteristics of the ball, such as weight or material, or objective function, such as desired kick distance, change, re-learning is needed. One could independently optimize for several target contexts in order to generalize a task, for example optimizing to kick the ball for different distances(context). Subsequently, when a new unseen context is presented, the optimized contexts can be generalized through regression methods (Niehaus, Röfer, and Laue 2007; Wang, Fleet, and Hertzmann 2009). However now optimizing for different contexts and then generalizing between the optimized parameters for different unseen contexts are two independent processes. Therefore, even though such approaches have been used successfully, they are time consuming as well as inefficient in terms of the number of needed training samples. In other words, we cannot reuse data-points obtained from optimizing a task with context s to improve and accelerate the optimization of a task with context s'. As such, it is desirable to learn the selection of the parameters for multiple tasks at once without restarting the learning process once we see a new task. This problem setup is also known as contextual policy search (Kupcsik et al. 2013; Kober, Oztop, and Peters 2010). Recently, such multi-task learning capability was established for information-theoretic policy search algorithms (Peters, Mülling, and Altun 2010), such as the episodic Contextual Relative Entropy Policy Search (CREPS) algorithm (Daniel, Neumann, and Peters 2012; Kupcsik et al. 2013). In (Abdolmaleki et al. 2015c), CREPS was successfully used to optimize a walking controller for different speeds. Despite its advantages, CREPS has a major set-back that does not allow it to perform favourably. Like many other stochastic search algorithms, CREPS maintains a Gaussian search distribution and it updates the mean and covariance matrix of its search distribution iteratively. However due to the covariance matrix update rule of CREPS, we will show that, search distribution might collapse prematurely to a point-estimate before finding a good solution, resulting in a premature convergence which is highly undesirable. Although, this multi-task learning capability is not found in other stochastic search algorithms (Hansen, Muller, and Koumoutsakos 2003; Sun et al. 2009), such as CMA-ES and NES, or commonly used policy search methods

(Stulp and Sigaud 2012; Kober and Peters 2010), they typically don't suffer from premature convergence. Therefore, to solve premature convergence problem of CREPS we use the rank-μ covariance matrix update rule of CMA-ES (Hansen, Muller, and Koumoutsakos 2003) along with CREPS. Now we combine the old coavarinace matrix with the estimated coavariance matrix from samples. We call the resulting algorithm Contextual Relative Entropy Policy Search with Covariance Matrix Adaptation (CREPS-CMA). We evaluate CREPS-CMA on standard functions and simulated robotic tasks including a complex humanoid robot kicking task. We will show that CREPS-CMA works favourably.

Problem Statement

Given a context vector $\boldsymbol{s}$ with m dimensions that defines a task, we want to find a function $\boldsymbol{m}(\boldsymbol{s}) : \mathbb{R}^m \rightarrow \mathbb{R}^n$ that outputs a parameter vector $\boldsymbol{\theta}$ with n dimensions such that it maximizes an objective function $R(\boldsymbol{\theta}, \boldsymbol{s}) : \{\mathbb{R}^n, \mathbb{R}^m\} \rightarrow \mathbb{R}$. The only accessible information on $R(\boldsymbol{\theta}, \boldsymbol{s})$ are evaluations $\{R^{[k]}\}_{k=1...N}$ of samples $\{\boldsymbol{s}^{[k]}, \boldsymbol{\theta}^{[k]}\}_{k=1...N}$, where k is the index of the sample and N is number of samples. We maintain a search distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ over the parameter space $\boldsymbol{\theta}$ of the objective function $R(\boldsymbol{\theta}, \boldsymbol{s})$. The search distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ is modeled as linear Gaussian policy, i.e.,

$$\pi(\theta|s) = \mathcal{N}\left(\boldsymbol{\theta}|\boldsymbol{m}_\pi(\boldsymbol{s}) = \boldsymbol{A}_\pi^T \varphi(\boldsymbol{s}), \Sigma_\pi\right),$$

where $\varphi(\boldsymbol{s})$ is an arbitrary feature function of context $\boldsymbol{s}$.

In each iteration, a new coefficient matrix $\boldsymbol{A}_\pi$ and a new covariance matrix Σ_π is obtained. Typically, $\boldsymbol{\varphi}(\boldsymbol{s}) = [1 \;\; \boldsymbol{s}]$, which results in a linear generalization over contexts. However, one could use non-linear feature functions such as radial basis functions (RBF) (Broomhead and Lowe 1988) for non-linear generalization over contexts. RBF features have been shown to enable algorithms to learn non-linear policies which greatly outperform their linear counterparts on non-linear tasks, such as humanoid walking or humanoid kicking (Abdolmaleki et al. 2015c).

In each iteration, given context samples[1] $\boldsymbol{s}^{[k]}$, the current search distribution $q(\boldsymbol{\theta}|\boldsymbol{s})$ is used to create samples $\boldsymbol{\theta}^{[k]}$ of the parameter vector $\boldsymbol{\theta}$. Subsequently, the evaluation $R^{[k]}$ of $\{\boldsymbol{s}^{[k]}, \boldsymbol{\theta}^{[k]}\}$ is obtained by querying the objective function $R(\boldsymbol{\theta}, \boldsymbol{s})$. In many algorithms, the samples $\{\boldsymbol{s}^{[k]}, \boldsymbol{\theta}^{[k]}, R^{[k]}\}_{k=1...N}$ are used to compute a weight $d^{[k]}$ for each sample k. Subsequently, using $\{\boldsymbol{s}^{[k]}, \boldsymbol{\theta}^{[k]}, d^{[k]}\}_{k=1...N}$, a new Gaussian search distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ is estimated. This process will run iteratively until the algorithm converges to a solution. Algorithm 1 shows a compact representation of contextual stochastic search methods.

Contextual Stochastic Search

The Relative Entropy Policy Search (REPS) algorithm was originally proposed as a step-based policy search algorithm (Peters, Mülling, and Altun 2010). Recently, an information-theoretic stochastic search algorithm was presented as a modification of REPS (Kupcsik et al. 2013). However, as we will show, the algorithm suffers from premature convergence. In order to solve this problem, we will extend the algorithm by using the rank-μ covariance matrix adaptation method of CMA-ES algorithm resulting to CREPS-CMA algorithm. The CMA-ES algorithm doesn't work for contextual setting.

Algorithm 1 Contextual stochastic search algorithm

Repeat

Set $q(\theta|\boldsymbol{s})$ **to** $\pi(\theta|\boldsymbol{s})$

Generate context samples $\{s^{[k]}\}_{k=1...N}$

Sample parameters $\{\theta^{[k]}\}_{k=1...N}$ **from current search distribution** $q(\theta|\boldsymbol{s})$ **given context samples** $\{s^{[k]}\}_{k=1...N}$

Evaluate the reward $R^{[k]}$ **of each sample in the sample set** $\{s^{[k]}, \theta^{[k]}\}_{k=1...N}$

Use the data set $\{s^{[k]}, R^{[k]}\}_{k=1...N}$ **to compute a weight** $d^{[k]}$ **for each sample**

Use the data set $\{s^{[k]}, \theta^{[k]}, d^{[k]}\}_{k=1...N}$ **to update the new search distribution** $\pi(\theta|\boldsymbol{s})$

Until search distribution $\pi(\theta|\boldsymbol{s})$ **converges.**

Given a dataset $\{\boldsymbol{s}^{[k]}, \boldsymbol{\theta}^{[k]}, R^{[k]}\}_{k=1...N}$ and the current distribution $q(\boldsymbol{\theta}|\boldsymbol{s})$ that has been used to generate the unweighed samples $\{\boldsymbol{\theta}^{[k]}\}_{k=1...N}$, CREPS-CMA first computes a weight $d^{[k]}$ for each sample and subsequently, using these weights, a new search distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ is computed. Therefore, we start by explaining how the weights are computed and, after that, we explain the search distribution update rules.

Weight Computation

The weight computation for CREPS-CMA is essentially the same as for CREPS (Kupcsik et al. 2013). The key idea behind CREPS is to ensure a smooth and stable learning process by bounding the relative entropy between the old search distribution $q(\boldsymbol{\theta}|\boldsymbol{s})$ and the newly estimated policy $\pi(\boldsymbol{\theta}|\boldsymbol{s})$. To do so, we want to optimize the following performance criteria

$$\begin{aligned} &\max_\pi \iint \mu(\boldsymbol{s})\pi(\boldsymbol{\theta}|\boldsymbol{s})\mathcal{R}_{\boldsymbol{s}\boldsymbol{\theta}} d\boldsymbol{\theta} d\boldsymbol{s}, \\ &\text{s.t.} \int \mu(\boldsymbol{s}) \mathrm{KL}\big(\pi(\boldsymbol{\theta}|\boldsymbol{s})||q(\boldsymbol{\theta}|\boldsymbol{s})\big) d\boldsymbol{s} \leq \epsilon, \\ &\quad \forall \boldsymbol{s}: \quad 1 = \int \pi(\boldsymbol{\theta}|\boldsymbol{s}) d\boldsymbol{\theta}, \end{aligned} \tag{1}$$

where $\mu(\boldsymbol{s})$ denotes the distribution over the context which is specified by the task and $\mathcal{R}_{\boldsymbol{s}\boldsymbol{\theta}}$ denotes the expected performance when evaluating parameter vector $\boldsymbol{\theta}$ in context $\boldsymbol{s}$. This optimization problem can be solved efficiently by the method of Lagrangian multipliers (Boyd and Vandenberghe

[1] Please note that the way we sample contexts $\boldsymbol{s}^{[k]}$ depends on the task. Throughout this paper we use a uniform distribution to sample contexts $\boldsymbol{s}^{[k]}$.

2004). The closed form solution for policy $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ is given by

$$\pi(\boldsymbol{\theta}|\boldsymbol{s}) \propto q(\boldsymbol{\theta}|\boldsymbol{s}) \exp\left(\mathcal{R}_{\boldsymbol{s}\boldsymbol{\theta}}/\eta\right),$$

where η is a Lagrangian multiplier that sets the temperature of the soft-max distribution given in the previous equation. The temperature parameter η can be found efficiently by optimizing the dual function

$$g(\eta) = \eta\epsilon + \eta \int \mu(\boldsymbol{s}) \log \left(\int q(\boldsymbol{\theta}|\boldsymbol{s}) \exp\left(\frac{\mathcal{R}_{\boldsymbol{s}\boldsymbol{\theta}}}{\eta}\right) d\boldsymbol{\theta} \right) d\boldsymbol{s}. \tag{2}$$

The optimal value for η can be obtained by minimizing the dual function $g(\eta)$ such that $\eta > 0$, see (Boyd and Vandenberghe 2004). However, approximating the log integral in the dual function (Equation 2) is not feasible as we would need a lot of samples $\boldsymbol{\theta}^{i,k}$ for each context $\boldsymbol{s}^k$. As the context can often not be directly controlled, we have only access to a single action $\boldsymbol{\theta}^k$ per context $\boldsymbol{s}^k$.

In order to achieve this requirement, the performance criteria is reformulated. CREP optimizes for the joint probabilities $p(\boldsymbol{s}, \boldsymbol{\theta})$ instead of for the policy $\pi(\boldsymbol{\theta}|\boldsymbol{s})$. Additionally, CREPS enforces that $p(\boldsymbol{s}) = \int_{\boldsymbol{\theta}} p(\boldsymbol{s}, \boldsymbol{\theta}) d\boldsymbol{\theta}$ still reproduces the correct context distribution $\mu(\boldsymbol{s})$ by using the constraints $\forall \boldsymbol{s} : p(\boldsymbol{s}) = \mu(\boldsymbol{s})$. However, there are now an infinite number of constraints. In order to avoid this, these constraints are approximately fulfilled by matching feature expectations, instead of matching single probabilities, i.e.,

$$\int_{\boldsymbol{s}} p(\boldsymbol{s}) \boldsymbol{\phi}(\boldsymbol{s}) d\boldsymbol{s} = \hat{\boldsymbol{\phi}},$$

where $\hat{\boldsymbol{\phi}} = \int_{\boldsymbol{s}} \mu(\boldsymbol{s}) \boldsymbol{\phi}(\boldsymbol{s}) d\boldsymbol{s}$ is the expected feature vector for the given context distribution $\mu(\boldsymbol{s})$ and a given feature space $\boldsymbol{\phi}$. For example, if the feature vector $\boldsymbol{\phi}(\boldsymbol{s})$ contains all linear and squared terms $[\boldsymbol{s}, \boldsymbol{s}^2]$ of the context vector $\boldsymbol{s}$, these constraints correspond to matching the first and second moment of $p(\boldsymbol{s})$ with the moments of $\mu(\boldsymbol{s})$.[2] Note that ϕ is typically different from φ. The resulting optimization program yields

$$\begin{aligned} &\max_{p} \iint p(\boldsymbol{s}, \boldsymbol{\theta}) \mathcal{R}_{\boldsymbol{s}\boldsymbol{\theta}} d\boldsymbol{s} d\boldsymbol{\theta} \\ &\text{s.t. } \epsilon \geq \text{KL}(p(\boldsymbol{s}, \boldsymbol{\theta}) || \mu(\boldsymbol{s}) q(\boldsymbol{\theta}|\boldsymbol{s})), \\ &\hat{\boldsymbol{\phi}} = \iint p(\boldsymbol{s}, \boldsymbol{\theta}) \boldsymbol{\phi}(\boldsymbol{s}) d\boldsymbol{s} d\boldsymbol{\theta}, \\ &1 = \iint p(\boldsymbol{s}, \boldsymbol{\theta}) d\boldsymbol{s} d\boldsymbol{\theta}. \end{aligned} \tag{3}$$

The solution for $p(\boldsymbol{s}, \boldsymbol{\theta})$ is now given by

$$p(\boldsymbol{s}, \boldsymbol{\theta}) \propto q(\boldsymbol{\theta}|\boldsymbol{s}) \mu(\boldsymbol{s}) \exp\left((\mathcal{R}_{\boldsymbol{s}\boldsymbol{\theta}} - V(\boldsymbol{s}))/\eta\right),$$

where $V(\boldsymbol{s}) = \boldsymbol{\phi}(\boldsymbol{s})^T \boldsymbol{w}$ is a context dependent baseline which is subtracted from the return $\mathcal{R}_{\boldsymbol{s}\boldsymbol{\theta}}$. The parameters $\boldsymbol{w}$ and η are again Lagrangian multipliers that can be obtained by optimizing the dual function, given as

$$\begin{aligned} g(\eta, \boldsymbol{w}) =& \eta\epsilon + \hat{\boldsymbol{\phi}}^T \boldsymbol{w} \\ &+ \eta \log \left(\iint \mu(\boldsymbol{s}) q(\boldsymbol{\theta}|\boldsymbol{s}) \exp\left(\frac{\mathcal{R}_{\boldsymbol{s}\boldsymbol{\theta}} - \boldsymbol{\phi}(\boldsymbol{s})^T \boldsymbol{w}}{\eta} \right) d\boldsymbol{\theta} d\boldsymbol{s} \right). \end{aligned} \tag{4}$$

This policy update results in a weight

$$d^{[k]} = \exp\left((\mathcal{R}_{\boldsymbol{s}\boldsymbol{\theta}} - V(\boldsymbol{s}))/\eta\right)$$

for each sample $[\boldsymbol{s}^{[k]}, \boldsymbol{\theta}^{[k]}]$, which we can use to estimate a new search distribution $\pi_\pi(\boldsymbol{\theta}|\boldsymbol{s})$.

Search Distribution Update Rule

Given dataset $\{\boldsymbol{s}^{[k]}, \boldsymbol{\theta}^{[k]}, d^{[k]}\}_{k=1\ldots N}$ and the old Gaussian search distribution

$$q(\boldsymbol{\theta}|\boldsymbol{s}) = \mathcal{N}\left(\boldsymbol{\theta}|\boldsymbol{m}_q(\boldsymbol{s}) = \boldsymbol{A}_q^T \varphi(\boldsymbol{s}), \Sigma_q\right),$$

we want to find the new search distribution

$$\pi(\boldsymbol{\theta}|\boldsymbol{s}) = \mathcal{N}\left(\boldsymbol{\theta}|\boldsymbol{m}_\pi(\boldsymbol{s}) = \boldsymbol{A}_\pi^T \varphi(\boldsymbol{s}), \Sigma_\pi\right),$$

by finding A_π and Σ_π. Therefore we need two update rules, one for updating the context-dependent mean function $\boldsymbol{m}_\pi$ of the search distribution and another one for updating the covariance matrix of the distribution Σ_π.

Context-Dependent Mean-Function Update Rule In order to find $\boldsymbol{m}_\pi$, the parameters $\boldsymbol{A}$ can be obtained by the weighted maximum likelihood

$$\boldsymbol{A} = \left(\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{U}, \tag{5}$$

where $\boldsymbol{\Phi}^T = [\varphi^{[1]}, ..., \varphi^{[N]}]$ contains the feature vector for all samples, $\boldsymbol{U} = [\theta^{[1]}, ..., \theta^{[N]}]$ contains all the sample parameters and $\boldsymbol{D}$ is the diagonal weighting matrix containing the weightings $d^{[k]}$.

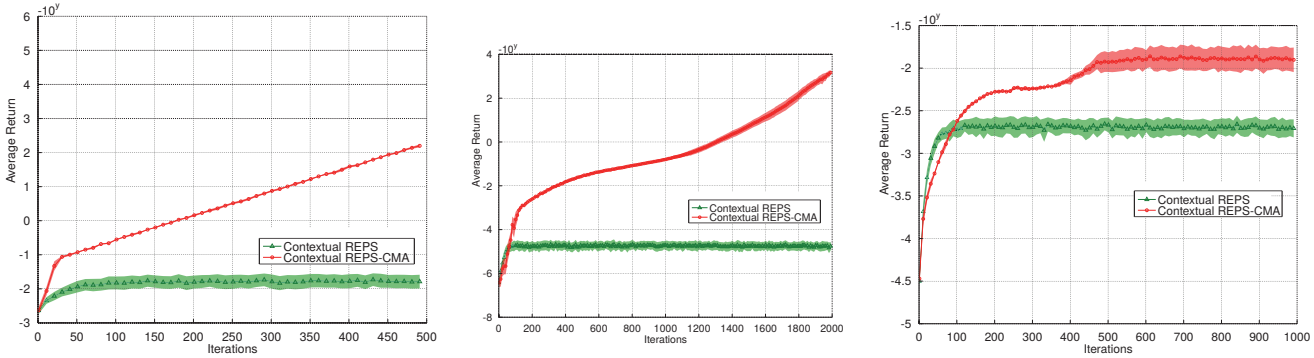
Covariance Matrix Update Rule Standard CREPS directly uses the weighted sample covariance matrix S as $\boldsymbol{\Sigma}_\pi$ which is obtained by

$$\boldsymbol{S} = \frac{\sum_{k=1}^N d^{[k]} \left(\boldsymbol{\theta}^{[k]} - \boldsymbol{A}^T \varphi(\boldsymbol{s}^{[k]})\right) \left(\boldsymbol{\theta}^{[k]} - \boldsymbol{A}_\pi^T \varphi(\boldsymbol{s}^{[k]})\right)^T}{Z}, \tag{6}$$

$$Z = \frac{(\sum_{k=1}^N d^{[k]})^2 - \sum_{k=1}^N (d^{[k]})^2}{\sum_{k=1}^N (d^{[k]})}.$$

Typically, the number of samples used for the estimated sample covariance matrix $\boldsymbol{S}$ is much smaller than number of free parameters of the covariance matrix. In this case, it has been shown that the sample covariance matrix from Equation 6 is not a good estimate of the true covariance matrix (Abdolmaleki et al. 2015b) and biases the search distribution towards a specific region of the search space. Due to this effect, the search distribution uncontrollably loses its exploration entropy along many dimensions of the parameter space which causes premature convergence. Such loss of exploration is a highly unwanted effect in stochastic

[2] In this paper, in all experiments $\boldsymbol{\phi}(\boldsymbol{s}) = [\boldsymbol{s}, \boldsymbol{s}^2]$, therefore we will always match the first and second moment of $p(\boldsymbol{s})$ with the moments of $\mu(\boldsymbol{s})$.



(a) Sphere (b) Rosenbrock (c) Rastrigin

Figure 1: The performance comparison of stochastic search methods for optimising contextual version of standard functions (a) Sphere, (b) Rosenbrock and (c) Rastrigin. The results show that while CREPS-CMA performs well, Contextual REPS suffers from premature convergence.

search. In order to alleviate this problem, inspired by CMA-ES (Hansen, Muller, and Koumoutsakos 2003), which is not a contextual algorithm, we combine the old covariance matrix and the sample covariance matrix from Equation 6, i.e.,

$$\boldsymbol{\Sigma}_\pi = (1-\lambda)\boldsymbol{\Sigma}_q + (\lambda)\boldsymbol{S}.$$

There are different ways to determine the interpolation factor $\lambda \in [0,1]$ between the sample covariance matrix $\boldsymbol{S}$ and the old covariance matrix $\boldsymbol{\Sigma}_q$. For example, in (Abdolmaleki et al. 2015b), the factor $\lambda \in [0,1]$ is chosen in such a way that the entropy of the new search distribution is reduced by a certain amount, while also being scaled with the number of effective samples. For CREPS-CMA, we use the rank-μ update in CMA-ES algorithm (Hansen, Muller, and Koumoutsakos 2003), i.e.,

$$\phi_{\text{eff}} = \frac{1}{\sum_{k=1}^{N}(d^{[k]})^2}, \quad \lambda = min\left(1, \frac{\phi_{\text{eff}}}{p^2},\right)$$

where ϕ_{eff} is the number of effective samples and p is the dimension of the parameter space $\boldsymbol{\theta}$. See Algorithm 2 for a compact representation of the CREPS-CMA algorithm.

Experiments

In this section, We evaluate CREPS-CMA algorithm.[3] We chose three series of optimization tasks for comparisons. In the first series, we use standard optimization test functions (Molga and Smutnicki 2005), such as the Sphere, the Rosenbrock and the Rastrigin (multi-modal) functions. We extend these functions to be applicable on the contextual setting. The task is to find the optimum 15 dimensional parameter vector $\boldsymbol{\theta}$ for a given 1 dimensional context $\boldsymbol{s}$.

For the second series of optimization tasks, we use a 5-link planar robot that has to reach a given point, which is varied by the context. We used dynamic movement primitives (DMPs) (Ijspeert and Schaal 2003) as the underlying policy representation with 25 parameters (5 basis functions per dimension). The context is the two dimensional position of the via-point to reach with the end-effector. Figure 2 shows the setup of the robot.

For the third series of optimization tasks, we used a simulated humanoid robot that has to kick a ball with a certain distance, which is varied by the context. We used a simple linear model as a controller, which takes as input the starting and end joint positions of the robot's legs and the time the movement is intended to take. The model linearly interpolates the starting and end positions through time. The context is the distance that the ball should reach. Figure 3 shows an example of the robot movement.

In Figures 1 and 2, we show the average, as well as two times the standard deviation of the results, over 10 trials for each experiment. Note that the y-axis of all plots is in a logarithmic scale. Figure 3(c) shows the average and two times the standard deviation of the results over 10 trials for several contexts of the humanoid kick.

Standard Optimization Test Functions

We chose three standard optimization functions, which are the Sphere function

$$f(\boldsymbol{s},\theta) = \sum_{i=1}^{p} \boldsymbol{x}_i^2,$$

the Rosenbrock function

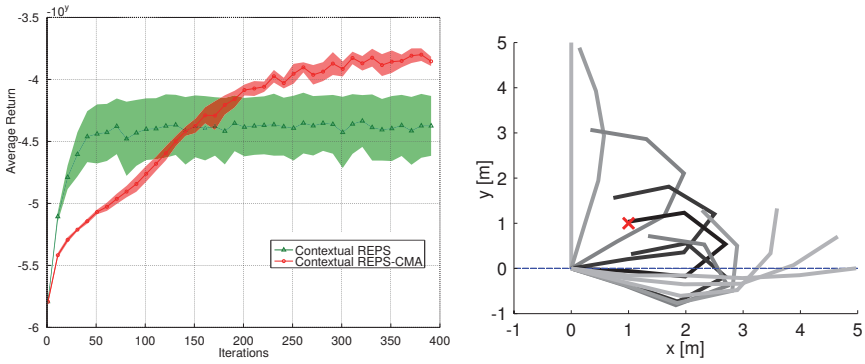
$$f(\boldsymbol{s},\theta) = \sum_{i=1}^{p-1}[100(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i^2)^2 + (1-\boldsymbol{x}_i)^2],$$

and also a multi-modal function, known as the Rastgirin function

$$f(\boldsymbol{s},\theta) = 10p + \sum_{i=1}^{p}[\boldsymbol{x}_i^2 - 10\cos(2\pi\boldsymbol{x}_i)],$$

where $\boldsymbol{x} = \boldsymbol{\theta} + \boldsymbol{A}\boldsymbol{s}$. The matrix A is a constant matrix that was chosen randomly. In our case, because the context $\boldsymbol{s}$ is 1 dimensional, A is a $p \times 1$ dimensional vector. Our definition

[3]Matlab source-code is available on-line at https://dl.dropboxusercontent.com/u/16387578/ContextualREPSCMA.zip



(a) Reaching task algorithm comparison (b) Planar robot

Figure 2: (a) Algorithmic comparison for a planar reaching task (5 joints, 25 parameters). In this task, CREPS-CMA has converged faster and learned the task well. Contextual REPS suffers from premature convergence and cannot learn the task. (b) The planar reaching task used for our comparisons. A 5-link planar robot has to reach a via-point $\boldsymbol{v}_{50} = [1, 1]$ in task space. The via-point position is the 2 dimensional context vector and is given. The via-point is indicated by the red cross. The postures of the resulting motion are shown as overlay, where darker postures indicate a posture which is close in time to the via-point.

for $\boldsymbol{x}$ means the optimum $\boldsymbol{\theta}$ for these functions is linearly dependent on the given context $\boldsymbol{s}$. The initial search area of $\boldsymbol{\theta}$ for all experiments is restricted to the hypercube $-5 \leq \boldsymbol{\theta}_i \leq 5, i = 1, \ldots, p$ and contexts are uniformly sampled from the interval $0 \leq \boldsymbol{s}_i \leq 3, i = 1, \ldots, z$ where z is the dimension of the context space $\boldsymbol{s}$. In our experiments, the mean of the initial distributions has been chosen randomly in the defined search area.

We compared CREPS-CMA with the standard Contextual REPS. In each iteration, we generated 50 new samples. The results in Figure 1 show that CREPS-CMA could successfully learn the contextual tasks while standard Contextual REPS suffers from premature convergence.

Planar Reaching

In this task, we used a 5-link planar robot with DMPs (Ijspeert and Schaal 2003) as the underlying control policy. Each link had a length of $1m$. The robot is modelled as a decoupled linear dynamical system. The robot has to reach a via-point $\boldsymbol{v}_{50} = [1, 1]$ at time step 50 with its end effector and, at the final time step $T = 100$, the point $\boldsymbol{v}_{100} = [5, 0]$. We varied the first via-point from 0 to 2 in both dimensions.

The reward was given by a context-dependent quadratic cost term for the two via-points as well as quadratic costs for high accelerations. Note that this performance function is highly non-linear in the parameters as the via-points are defined in end effector space. We used 5 basis functions per degree of freedom for the DMPs while the goal attractor for reaching the final state was assumed to be known. Hence, our parameter vector had 25 dimensions.

We generated 100 new samples in each iteration. The results in Figure 2(a) shows that CREPS-CMA successfully learned the task while contextual REPS cannot complete it.

Humanoid Kick

In this task, roobot should learn to kick the ball to different distances. The context here is the distance that ball travels which varies from $3m$ to $12m$. The robot kick controller is a simple linear model with three input groups: the action time t, the initial position of the robot, represented as a vector of joint angles with dimension l, and the final robot position, also represented as a vector of joint angles with dimension l. The action time is the amount of time the robot takes to move from the initial to the final position. The joint angles are linearly interpolated across t to create the corresponding movement.

Our humanoid robot has $l = 6$ joints in each leg, and other remaining joints (arms and head) are ignored. In other words, the joint angles are 12-dimensional vectors and the controller receives a 25-dimensional parameter vector, which is then interpolated and coded into motor commands. Figures 3(a) and 3(b) show the initial and final positions of an exemplary kick.

The reward was based on the reward function

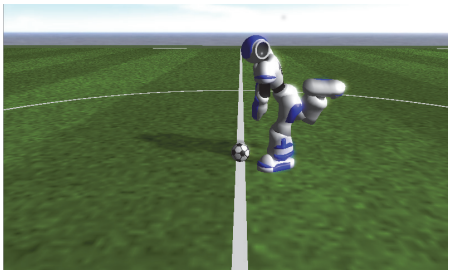
$$R(s, \theta) = -(x - s)^2 - y^2,$$

where x and y are the distances between the target point and the ball along the x and y axes. We use CREPS-CMA to optimise this reward function and similar to (Abdolmaleki et al. 2015c) we use radial basis feature function for non-linear generalization over contexts. After 1000 iterations with 20 new samples per iteration, the robot could kick the ball with a good accuracy shown on Figure 3(c).[4] The average error distance of the ball to the target was $0.34 \pm 0.11m$.

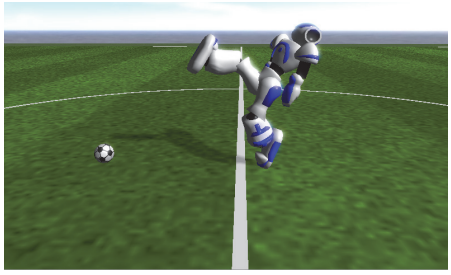
Conclusion

Many optimization algorithms have been proposed by the scientific community. However, these algorithms usually optimize for a single context of a task, like the ideal gait for the highest speed, the lowest energy consumption, or both.

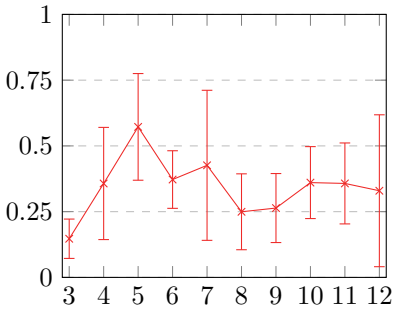
[4]Demonstration video is available on-line at https://dl.dropboxusercontent.com/u/16387578/ICARSC16kick.mp4. The robot is placed at several points and kicks the ball to the centre of the field using the learned policy.



(a) Initial position of kick movement.



(b) Final position of kick movement.



(c) Accuracy of the kick.

Figure 3: (a) The initial position of an exemplary humanoid kick. (b) The final position of an exemplary humanoid kick. (c) The results of the task. The y-axis represents the distance at which the ball was from the intended target, in meters, while the x-axis represents the distance at which the ball was being kicked from, also in meters.

Algorithm 2 CREPS-CMA

Input : Data Set $\mathcal{D}\{\boldsymbol{s}^{[k]}, \boldsymbol{\theta}^{[k]}, R^{[k]}\}_{k=1...N}$ and the old covariance matrix $\boldsymbol{\Sigma}_q$

Compute the weights $d^{[k]}$ for each sample:

1- Optimize the dual function g and find optimum η and $\boldsymbol{w}$

$$g(\eta, \boldsymbol{w}) = \eta\epsilon + \hat{\boldsymbol{\phi}}^T \boldsymbol{w} + \eta \log \left(\sum_{K=1}^{N} \frac{1}{N} \exp \left(\frac{R^{[k]} - \boldsymbol{\phi}(\boldsymbol{s}^{[k]})^T \boldsymbol{w}}{\eta} \right) \right).$$

2- Compute weights $d^{[k]} = \exp \left(\frac{R^{[k]} - \boldsymbol{\phi}(\boldsymbol{s}^{[k]})^T \boldsymbol{w}}{\eta} \right)$.

3- Normalize $d^{[k]}$ such that $\sum_{k=1}^{N} d^{[k]} = 1$.

Compute the new mean function $m_\pi(\boldsymbol{s})$:

Use weighted maximum likelihood to estimate parameters $\boldsymbol{A}_\pi$ of the new mean function

$$\boldsymbol{A}_\pi = (\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1} \boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{U},$$

where $\boldsymbol{\Phi}^T = [\varphi^{[1]}, ..., \varphi^{[N]}]$ contains the feature vector for all samples, $\boldsymbol{U} = [\theta^{[1]}, ..., \theta^{[N]}]$ contains all the sample parameters and $\boldsymbol{D}$ is the diagonal weighting matrix containing the weights $d^{[k]}$.

Compute the sample covariance $\boldsymbol{S}$:

$$\boldsymbol{S} = \sum_{k=1}^{N} d^{[k]} \left(\boldsymbol{\theta}^{[k]} - \boldsymbol{A}_\pi^T \boldsymbol{\varphi}(\boldsymbol{s}^{[k]})\right) \left(\boldsymbol{\theta}^{[k]} - \boldsymbol{A}_\pi^T \boldsymbol{\varphi}(\boldsymbol{s}^{[k]})\right)^T / Z,$$

Compute the number of effective samples ϕ_{eff} and λ:

$$\phi_{\text{eff}} = \frac{1}{\sum_{k=1}^{N} (d^{[k]})^2}, \quad \lambda = min\left(1, \frac{\phi_{\text{eff}}}{p^2}\right)$$

Compute the new covariance matrix $\boldsymbol{\Sigma}$:

$$\boldsymbol{\Sigma}_\pi = \lambda \boldsymbol{\Sigma}_q + (1 - \lambda) \boldsymbol{S}.$$

Therefore, in this paper, we investigated contextual stochastic search methods for multi-task learning. We alleviated the premature convergence problem of contextual REPS, which resulted in the CREPS-CMA algorithm. The results show that the algorithm performs favourably and solves the premature convergence issue. We also show its applicability in practical situations, such as a humanoid robot kick task. Regarding future work, it is worthwhile to incorporate the step-size control feature of CMA-ES into CREPS-CMA. This step-size control aims to make consecutive movements of the distribution mean orthogonal in expectation. It effectively prevents premature convergence while allowing fast convergence to an optimum.

References

Abdolmaleki, A.; Lioutikov, R.; Peters, J.; Lua, N.; Reis, L.; and Neumann, G. 2015a. Regularized covariance estimation for weighted maximum likelihood policy search methods. In *Advances in Neural Information Processing Systems (NIPS), MIT Press*.

Abdolmaleki, A.; Lua, N.; Reis, L.; and Neumann, G. 2015b. Regularized covariance estimation for weighted maximum likelihood policy search methods. In *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*.

Abdolmaleki, A.; Lua, N.; Reis, L.; Peters, J.; and Neumann, G. 2015c. Contextual Policy Search for Generalizing a Parameterized Biped Walking Controller. In *IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*.

Boyd, S., and Vandenberghe, L. 2004. *Convex optimization*. Cambridge university press.

Broomhead, D. S., and Lowe, D. 1988. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, DTIC Document.

Daniel, C.; Neumann, G.; and Peters, J. 2012. Hierarchical Relative Entropy Policy Search. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Hansen, N.; Muller, S.; and Koumoutsakos, P. 2003. Re-

ducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*.

Ijspeert, A., and Schaal, S. 2003. Learning Attractor Landscapes for Learning Motor Primitives. In *Advances in Neural Information Processing Systems 15(NIPS)*.

Kober, J., and Peters, J. 2010. Policy Search for Motor Primitives in Robotics. *Machine Learning* 1–33.

Kober, J.; Oztop, E.; and Peters, J. 2010. Reinforcement Learning to adjust Robot Movements to New Situations. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*.

Kupcsik, A.; Deisenroth, M. P.; Peters, J.; and Neumann, G. 2013. Data-Efficient Contextual Policy Search for Robot Movement Skills. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Mannor, S.; Rubinstein, R.; and Gat, Y. 2003. The Cross Entropy method for Fast Policy Search. In *Proceedings of the 20th International Conference on Machine Learning(ICML)*.

Molga, M., and Smutnicki, C. 2005. Test Functions for Optimization Needs. In *http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf*.

Niehaus, C.; Röfer, T.; and Laue, T. 2007. Gait optimization on a humanoid robot using particle swarm optimization. In *Proceedings of the Second Workshop on Humanoid Soccer Robots in conjunction with the*, 1–7.

Peters, J.; Mülling, K.; and Altun, Y. 2010. Relative Entropy Policy Search. In *Proceedings of the 24th National Conference on Artificial Intelligence (AAAI)*. AAAI Press.

Rückstieß, T.; Felder, M.; and Schmidhuber, J. 2008. State-dependent Exploration for Policy Gradient Methods. In *Proceedings of the European Conference on Machine Learning (ECML)*.

Stulp, F., and Sigaud, O. 2012. Path Integral Policy Improvement with Covariance Matrix Adaptation. In *International Conference on Machine Learning (ICML)*.

Sun, Y.; Wierstra, D.; Schaul, T.; and Schmidhuber, J. 2009. Efficient Natural Evolution Strategies. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation(GECCO)*.

Theodorou, E.; Buchli, J.; and Schaal, S. 2010. A Generalized Path Integral Control Approach to Reinforcement Learning. *The Journal of Machine Learning Research*.

Wang, J. M.; Fleet, D. J.; and Hertzmann, A. 2009. Optimizing walking controllers. *ACM Transactions on Graphics (TOG)* 28(5):168.