20th RoboCup International Symposium, Leipzig, Germany, July 2016.

Learning a Humanoid Kick With Controlled Distance

Abbas Abdolmaleki[1,2,3], David Simões[1], Nuno Lau[1], Luis Paulo Reis[2,3], 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
{abbas.a, david.simoes, nunolau}@ua.pt, lpreis@dsi.uminho.pt
neumann@ias.tu-darmstadt.de

Abstract. We investigate the learning of a flexible humanoid robot kick controller, i.e., the controller should be applicable for multiple contexts, such as different kick distances, initial robot position with respect to the ball or both. Current approaches typically tune or optimise the parameters of the biped kick controller for a single context, such as a kick with longest distance or a kick with a specific distance. Hence our research question is that, how can we obtain a flexible kick controller that controls the robot (near) optimally for a continuous range of kick distances? The goal is to find a parametric function that given a desired kick distance, outputs the (near) optimal controller parameters. We achieve the desired flexibility of the controller by applying a contextual policy search method. With such a contextual policy search algorithm, we can generalize the robot kick controller for different distances, where the desired distance is described by a real-valued vector. We will also show that the optimal parameters of the kick controller is a non-linear function of the desired distances and a linear function will fail to properly generalize the kick controller over desired kick distances.

Keywords: Contextual Policy Search - Motor Learning - Humanoid Robot - Non-linear Policies

1 Introduction

Designing optimal controllers for robotic systems is one of the major tasks in the robotics research field. Hence, it is desirable to have a controller that can control the robot for different tasks or contexts in real time, for example a soccer robot should be able to kick the ball for any desired kick distance which can be chosen from a continuous range of kick distances. We define a task as a context. Context is a vector of variables that do not change during a task's execution, but might change from task to task. In this paper for example, the context is the distance the ball travels after being kicked and can be chosen by the agent. The kick task is one of the most important skills in the context of robotic soccer[1]. Typically the kick controllers are only applicable for a discretized number of desired distances. For example three sets of parameters for the kick controller is obtained which are applicable for long, mid and short distance kicks.

Such a controller limits the robot to properly pass the ball to its teammates. Controlling the robot to kick the ball (near)optimally for different distances, allows the agents have a lot more control and options regarding their next decision, which could affects the game's outcome. Our goal is to find a parametric function that given a desired kick distance, outputs the (near) optimal controller parameters. In the other word we would like to obtain a policy $\pi(\theta|s)$ that sets the parameters θ of a robot kick controller given a context s which is the desired kick distance. In order to optimize the robot controller parameters given an objective function, there are many algorithms proposed by the scientific community [2–9]. However, many of these algorithms usually optimize a parameter set for a single context, such as optimizing a kick for the longest distance or the highest accuracy [10]. In other words, these algorithms fail to generalize the optimized movement for a context to different contexts. In order to generalize the kick motion to, for example, different kicking distances, typically the parameters are optimized for several target contexts independently. Afterwards, to generalize movements to new unseen contexts and to obtain a continuous policy $\pi(\theta|s)$, regression methods are commonly used [11, 12]. Although such approaches have been successfully used, they are time consuming and inefficient regarding the number of needed training samples. In such a method, data-points obtained from optimizing the kick controller for context s cannot be re-used to improve and accelerate the optimisation for context s'. This is due to the fact that optimizing the controller parameters and generalizing them are two independent processes and the correlation between different contexts is ignored during the optimisation. Therefore in this paper we propose to use contextual relative entropy policy search(CREPS) algorithm which searches for the optimal parameters of the policy $\pi(\theta|s)$ in one run optimisation process a. In the other word in CREPS, optimizing the controller parameters and generalizing them happens simultaneously and therfore the correlation between different contexts can be exploited in order to accelerate the optimisation. CREPS, however, has a major drawback related to its search distribution update. The distribution might collapse prematurely to a point-estimate, resulting in premature convergence. On the other hand, the CMA-ES algorithm [2] which is not a contextual algorithm has shown to be able to avoid premature convergence. Therefore we combine the update rules of CREPS and CMA-ES resulting to the contextual relative entropy policy search with covariance matrix adaptation(CREPS-CMA). We will show that CREPS-CMA avoids premature convergence. Hence we will use CREPS-CMA for optimising the kick controller. We will also show that a non-linear function of desired kick distance clearly outperforms a linear one. This effect has been also observed for the humanoid walking task [13]. Now our robot is able to kick the ball for a continuous range of desire kick distances. This is in contrast with our previous approach where we had 3 sets of parameters for short, mid and long distance kicks.

2 The Approach

We used a simulated Nao robot shown in Figure 1 for our experiments. Our movement pipeline is composed of two main parts: a kick controller, which receives parameters $\boldsymbol{\theta}$ and converts them into joint commands for the robot's servos; and a policy function, which maps a given context s for a specific kick distance into the corresponding param-

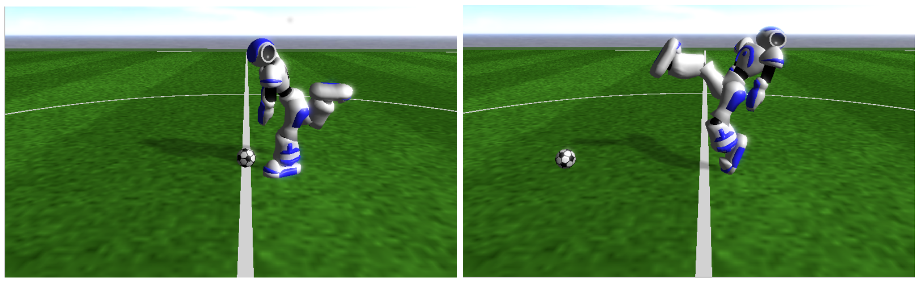


Fig. 1. The initial (left) and final (right) positions of an exemplary kick movement.

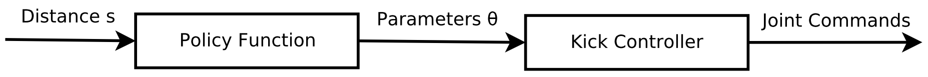


Fig. 2. The pipeline of our contextual kick movement.

eter vector $\boldsymbol{\theta}$. The pipeline for the kick task, whose context is the kick distance s with a straight kick direction with respect to the torso, is shown in Figure 2.

2.1 Kick Controller

We have a kick controller which is a simple keyframe-based [10] linear model and we also have stability module as in [1] that stabilize the robot during performing the kick movement. A keyframe, as defined in [10], is a complete description of joint angles, either absolute or relative to the previous keyframe. Our keyframe based controller is defined by the following parameters:

- The initial keyframe, represented as a vector α of joint angles with dimension l,
- The final keyframe, also represented as a vector β of joint angles with dimension l.
- The action time t that is the amount of time the robot takes to move from the initial to the final keyframe. The joint angles are linearly interpolated across t to create the corresponding movement.

During performing kick only the legs joints move and remaining joints (arms and head joints) are kept constant. As each leg has 6 joints, α and β are 12-dimensional vectors. Therefore considering the action time t, our kick controller has 25 parameters to set. The controller receives a 25-dimensional parameter vector $\boldsymbol{\theta}$, which is then interpolated and coded into motor commands. Figure 1 shows the initial and final positions of an exemplary kick. The stability module has its constant parameters which doesn't change from task to task, please see [1] for more details of our stability module. Now we need to find a policy function of kick distance s that sets our controller parameters with the proper parameters $\boldsymbol{\theta}$ for any given desired kick distance.

2.2 Policy Function

Our goal is to find a function in form of

$$\mu(\boldsymbol{s}) = \boldsymbol{A}^T\varphi(\boldsymbol{s}),$$

that given a context vector $\boldsymbol{s}$ with dimension d_s, outputs a optimal parameter vector $\boldsymbol{\theta}$ with dimension $d_{\boldsymbol{\theta}}$ such that it maximise our objective function $R(\boldsymbol{\theta}, \boldsymbol{s}) : \{\mathbb{R}^{d_s}, \mathbb{R}^{d_{\boldsymbol{\theta}}}\} \rightarrow \mathbb{R}$. Where $\varphi(\boldsymbol{s})$ is an arbitrary feature function of context $\boldsymbol{s}$ that outputs a feature vector with dimension d_φ and the gain matrix A_π is a $d_{\boldsymbol{\theta}} \times d_\varphi$ matrix. Typically $\varphi(\boldsymbol{s}^{[i]}) = [1 \quad \boldsymbol{s}^{[i]}]$, which results in linear generalization over contexts. In order to achieve non-linear generalization over contexts, we can use normalized radial basis features (RBF) as a feature function:

$$\varphi(\boldsymbol{s}^{[i]}) = \frac{\psi_j(\boldsymbol{s}^{[i]})}{\sum_{j=1}^{K}\psi_j(\boldsymbol{s}^{[i]})}, \quad \psi_j(\boldsymbol{s}^{[i]}) = \exp(-\frac{(\boldsymbol{s}^{[i]} - c_j)^2}{2\sigma^2}),$$

where K is the number of RBFs and centres $\{c_j\}_{j=1...K}$ are equally spaced in the range of $\boldsymbol{s}$, based on the desired number of RBFs K, and σ^2 is the bandwidth of the RBF. The bandwidth represents how related contexts are. A large bandwidth means that contexts are very similar and therefore the relationship is (near)linear. A bandwidth of 0 is an extreme case where movements are not generalizable at all, and each context has its independent optimal parameters. Both K and σ^2 are hand-tuned parameters. 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 walking [13], so we expected a performance increase. Now the task is to learn the optimal gain matrix A. As we don't have the labelled data to fit A, we need to use a reinforcement learning method.

2.3 Learning Policy Function

In order to learn the policy function $\mu(\boldsymbol{s})$ we use a contextual policy search algorithm called CREPS-CMA. CREPS-CMA is an extension of contextual REPS [14, 8] which is capable of multi-task learning. The goal of CREPS-CMA is to find a function $\mu(\boldsymbol{s})$ that given a context $\boldsymbol{s}$, it outputs a parameter vector $\boldsymbol{\theta}$ such that $\{\boldsymbol{s}, \boldsymbol{\theta}\}$ maximises the objective function $R(\boldsymbol{s}, \theta)$. The only accessible information on the objective function $R(\boldsymbol{s}, \theta)$ are evaluations $\{R_k\}_{k=1...k}$ of samples $\{\boldsymbol{s}_k, \boldsymbol{\theta}_k\}_{k=1...k}$, where k is the index of the sample, ranging from 1 to the number of samples N. CREPS-CMA maintains a stochastic search distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ over the parameter space $\boldsymbol{\theta}$ of the objective function which is used to generate samples $\boldsymbol{\theta}$ given $\boldsymbol{s}$. The search distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ is modelled as a linear Gaussian policy, i.e.,

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

where the mean of the distribution is our policy function $\mu(\boldsymbol{s})$ we are searching for and covariance matrix $\varSigma_\pi$ controls the exploration of the algorithm. CREPS-CMA is an iterative algorithm. First it initializes the search distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ by defining matrix and covariance matrix Σ_π with arbitrary values[1]. Afterwards in each iteration, given context samples[2] $\{\boldsymbol{s}_k\}_{k=1\dots k}$, the current search distribution $q(\boldsymbol{\theta}|\boldsymbol{s})$ is used to create samples $\{\boldsymbol{\theta}_k\}_{k=1\dots k}$ of the parameter vector $\boldsymbol{\theta}$. Subsequently, the evaluation $\{R_k\}_{k=1\dots k}$ of samples $\{\boldsymbol{s}_k, \boldsymbol{\theta}_k\}_{k=1\dots k}$ is obtained by querying the objective function $R(\boldsymbol{s}, \boldsymbol{\theta})$. And dataset $\{\boldsymbol{s}_k, \boldsymbol{\theta}_k, R_k\}_{k=1\dots k}$ is used to compute a weight $\{d_k\}_{k=1\dots k}$ for all samples. Each weight is a pseudo-probability for the corresponding sample. Subsequently, using $\{\boldsymbol{s}_k, \boldsymbol{\theta}_k, d_k\}_{k=1\dots k}$, a new Gaussian search distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ is estimated by estimating a new A matrix and covariance matrix Σ_π. The new search distribution will give more probabilities to the samples $\{\boldsymbol{s}_k, \boldsymbol{\theta}_k\}_{k=1\dots k}$ with better returns $\{R_k\}_{k=1\dots k}$. This process runs iteratively until the algorithm converges to a solution. After all we are interested in the matrix A to construct our policy function $\mu(\boldsymbol{s})$. Algorithm 1 shows a compact representation of contextual stochastic search methods. Now we briefly explain how CREPS-CMA computes weights and what are the update rules of the search policy.

Algorithm 1 Contextual stochastic search algorithm

```
Initialize $\pi(\theta|\boldsymbol{s})$
Repeat
    Set $q(\theta|\boldsymbol{s})$ to $\pi(\theta|\boldsymbol{s})$
    Use a uniform distribution to generate context samples $\{s_k\}_{k=1\dots N}$
    Sample parameters $\{\theta_k\}_{k=1\dots N}$ from current search distribution $q(\theta|\boldsymbol{s})$ given context samples $\{s_k\}_{k=1\dots N}$
    Evaluate the reward $R_k$ of each sample in the sample set $\{s_k, \theta_k\}_{k=1\dots N}$
    Use the data set $\{\boldsymbol{\theta}_k, s_k, R_k\}_{k=1\dots N}$ to compute a weight $d_k$ for each sample
    Use the data set $\{s_k, \theta_k, d_k\}_{k=1\dots N}$ to update the new search distribution $\pi(\theta|\boldsymbol{s})$
Until search distribution $\pi(\theta|\boldsymbol{s})$ converges.
```

2.4 CREPS-CMA

The key idea behind contextual REPS [8] 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 $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ while maximising the expected return. This results in a weight

$$d_k = \exp\left((\mathcal{R}_{\boldsymbol{s\theta}} - V(\boldsymbol{s}))/\eta\right)$$

for each sample $[\boldsymbol{s}_k, \boldsymbol{\theta}_k]$, which we can use to estimate a new search distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})$. $\mathcal{R}_{\boldsymbol{s\theta}}$ denotes the expected performance when evaluating parameter vector $\boldsymbol{\theta}$ in context

[1] With initializing we can define the region of the space that we would like the algorithm starts searching

[2] 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 $\boldsymbol{s}_k$ which is desired kick distance. The intuition behind it is that all the kick distances have same importance for us.

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

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

Where $\hat{\boldsymbol{\varphi}} = \sum_{K=1}^{N}\boldsymbol{\varphi}(\boldsymbol{s}^{[k]})$ is the expected feature vector for the given context samples. We optimize this convex dual function by gradient decent. Now given dataset $\{\boldsymbol{s}_k, \boldsymbol{\theta}_k, d_k\}_{k=1\ldots N}$ and the old Gaussian search distribution

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

we want to find the new search distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ by finding A_π and Σ_π. Therefore we need two update rules, one for updating the context-dependent policy function $\boldsymbol{\mu}_\pi(\boldsymbol{s})$ of the search distribution and another one for updating the covariance matrix Σ_π of the distribution.

Context-Dependent Mean-Function Update Rule The matrix A can be obtained by the weighted maximum likelihood, i.e.,

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

where $\boldsymbol{\Phi}^T = [\varphi^{[1]}, ..., \varphi^{[N]}]$ contains the feature vector for all context samples $\{\boldsymbol{s}_k\}_{k=1\ldots N}$, $\boldsymbol{U} = [\theta^{[1]}, ..., \theta^{[N]}]$ contains all the sample parameters, $\boldsymbol{D}$ is the diagonal weighting matrix containing the weightings $\{\dot{k}\}_{k=1\ldots N}$ and $\lambda\boldsymbol{I}$ is a regularization term. λ is a very small number such as $1e-8$.

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

$$\boldsymbol{S} = \frac{\sum_{k=1}^{N} d_k\left(\boldsymbol{\theta}_k - \boldsymbol{A}^T\varphi(\boldsymbol{s}_k)\right)\left(\boldsymbol{\theta}_k - \boldsymbol{A}_\pi^T\varphi(\boldsymbol{s}_k)\right)^T}{Z}, \tag{2}$$
$$Z = \frac{(\sum_{k=1}^{N} d_k)^2 - \sum_{k=1}^{N}(d_k)^2}{\sum_{k=1}^{N}(d_k)}.$$

It has been shown that the sample covariance matrix from Equation 2 is not a good estimate of the true covariance matrix [15], since it biases the search distribution towards a specific region of the search space. In other words, the search distribution loses its exploration entropy along many dimensions of the parameter space, which causes premature convergence. This is a highly unwanted effect in policy search. To alleviate this problem, inspired by rank-μ update rule of CMA-ES [2], which is not a contextual algorithm, we combine the old covariance matrix and the sample covariance matrix from Equation 2, 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 [15], 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. We will extended REPS by using the rank-μ covariance matrix adaptation method of CMA-ES algorithm [2] which has been shown to be effective for avoiding premature convergence, i.e.,

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

where ϕ_{eff} is the number of effective samples and $d_{\boldsymbol{\theta}}$ is the dimension of the parameter space $\boldsymbol{\theta}$.

3 Experiments

[3] In this section, first we evaluate CREPS-CMA algorithm. Hence we use standard optimization test functions [16], such as the Sphere, the Rosenbrock and the Rastrigin (multi-modal) functions. We extend these functions to be applicable for the contextual setting. The task is to find the optimum 15 dimensional parameter vector $\boldsymbol{\theta}$ for a given 1 dimensional context $\boldsymbol{s}$. We will show that CREPS-CMA performs favourably. Afterwards, We use CREPS-CMA to optimize our kick controller for different desired kick distances for a simulated Nao robot[4] and will show our accuracy results, with both linear and non-linear policies. According to the results non-linear policy outperforms the linear one.

3.1 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

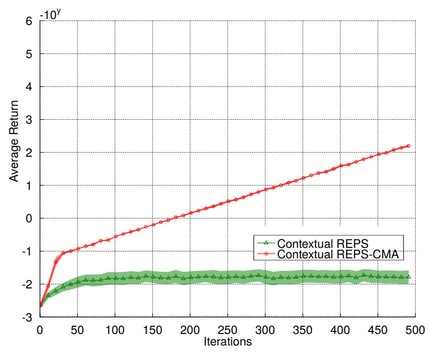
$$f(\boldsymbol{s}, \theta) = \sum_{i=1}^{p-1} [100(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i^2)^2 + (1 - \boldsymbol{x}_i)^2],$$

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

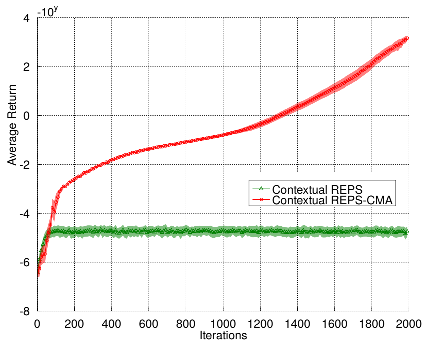
$$f(\boldsymbol{s}, \theta) = 10p + \sum_{i=1}^{p} [\boldsymbol{x}_i^2 - 10\cos(2\pi\boldsymbol{x}_i)],$$

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

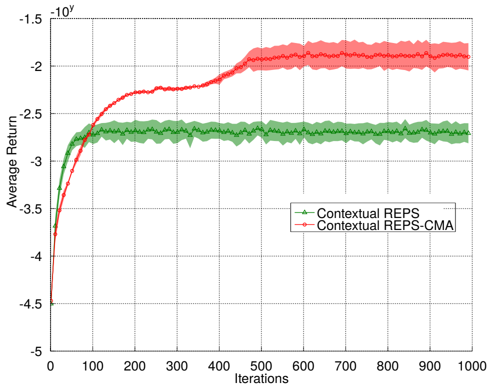
[4] `https://www.ald.softbankrobotics.com/en`



(a) Sphere



(b) Rosenbrock



(c) Rastrigin

Fig. 3. The performance comparison of CREPS and CREPS-CMA for optimising contextual versions of standard functions (a) Sphere, (b) Rosenbrock and (c) Rastrigin. The results show that CREPS-CMA clearly outperforms CREPS in all three benchmarks while CREPS suffers from premature convergence.

where p is the number of dimensions of $\boldsymbol{\theta}$ and $\boldsymbol{x} = \boldsymbol{\theta} + \boldsymbol{A}\boldsymbol{s}$. 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 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 3 show that CREPS-CMA could successfully learn the contextual tasks while standard Contextual REPS suffers from premature convergence.

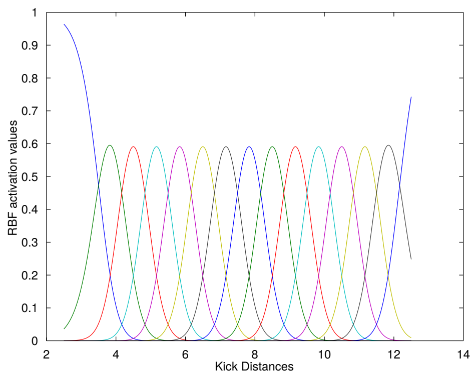


Fig. 4. The 15 RBFs setup used for generating features.

3.2 Kick Task Results

We use a Nao humanoid robot simulated in RoboCup 3D simulation environment which is based on SimSpark [5]: a generic physical multiagent system simulator. The robot has 22 degrees of freedom, six in each leg, four in each arm, and two in the neck. We use CREPS-CMA to train a simulated NAO robot by optimising the kick controller explained in section 2 using both linear policies, i.e., $\varphi(\boldsymbol{s}^{[i]}) = [1 \;\; \boldsymbol{s}^{[i]}]$ and a RBF based non-linear policy. The desired kick distance s varies from 2.5m to 12.5m. For the non-linear policy, we choose $K = 15$ normalized RBFs and σ^2 is set to 0.5. Both K and the σ^2 parameters were chosen by trial and error to maximize the results accuracy. Figure 4 shows the setup of the used RBFs over the context range.

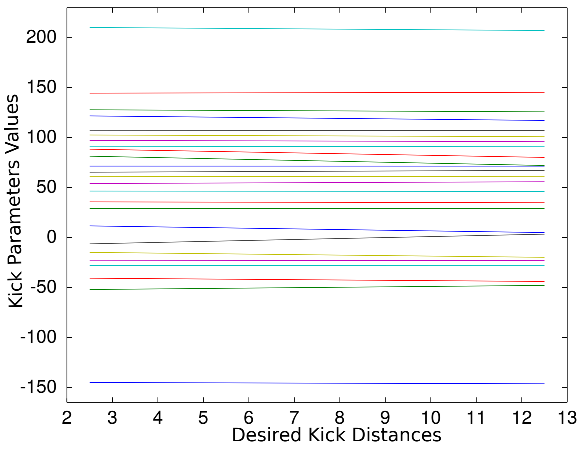
We maximize a context dependent objective function

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

where s is the desired kick distance, and x and y are the ball distances travelled along the x- and y-axes using the kick controller with the given parameter set θ. We initialize the search distribution π with a hand tuned kick policy, which was able to kick the ball over $15m$. We optimized the kick with 1000 iterations. Each iteration generates 20 new samples where the contexts were sampled uniformly. Each sample was evaluated 5 times, and was averaged to smooth out the noisy returns. In order to simulate competition conditions, for evaluating each sample, we placed the robot in 5 different positions around the ball and it had to perceive the ball, move towards it, position itself in place and then kick it towards the target goal using the kick controller. We compared the performance of the linear policy with non-linear one. Figure 6 shows that the non-linear policy clearly outperforms the linear one and the accuracy of the non-linear policy is considerable. [6] The average error of the linear policy was $0.82 \pm 0.10m$ while

[5] `http://simspark.sourceforge.net/`

[6] Demonstration video of the non-linear kick controller using the magma challenge tool[17] is available on-line at `https://www.dropbox.com/sh/0iimyykf6xejj6g/AADg9iCNJZAbu3Voe2UKsmQza?dl=0`.



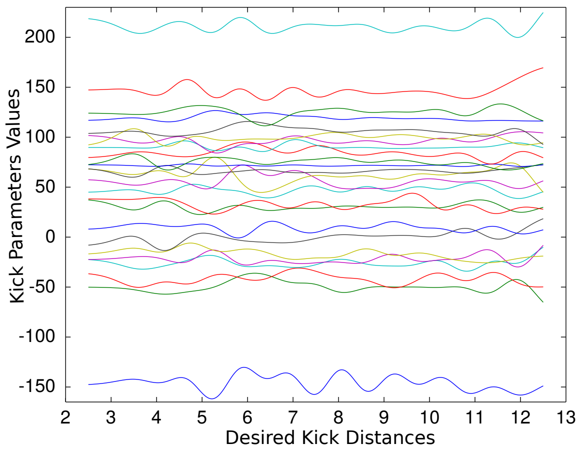


Fig. 5. The learned linear (left) and non-linear (right) policies for kick distances of 2.5 to 12.5 meters. The y-axis represents the controller parameter values for a given desired kick distance, and the x-axis represents the desired kick distance.

we achieved an average error of $0.34 \pm 0.11m$ using the non-linear policy. As expected, using a non-linear policy improves the accuracy of the results with order of magnitude. In fact, the average error is more than halved. This also demonstrates the non-linearity nature of robotic tasks such as kick task and the usefulness of using RBF functions to capture this non-linearity. Figure 5 shows the learned linear and non-linear policies for generalizing the 25 parameter kick controller for different kick distances.We can see that the learned linear policy is a linear approximation of its corresponding non-linear policy.

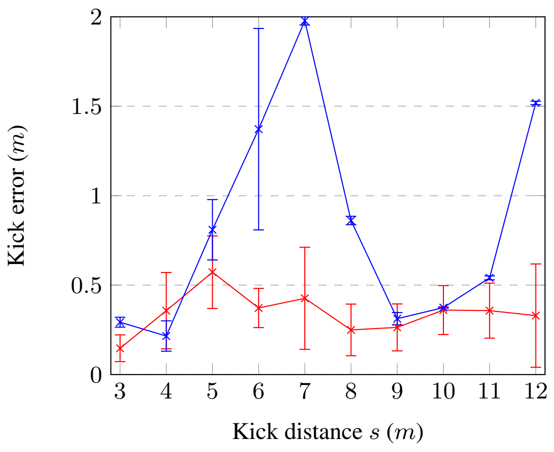


Fig. 6. The performance of the learned linear (blue) and non-linear (red) policies. The x-axis represents the desired kick distance, in meters, while the y-axis represents the error with respect to desired kick distance, also in meters.

4 Conclusion

We used a recently proposed contextual policy search algorithm to generalize a robot kick controller for different desired kick distances, where a context is described by a real-valued vector of distances. We have modified the algorithm, naming it CREPS-CMA. Using CREPS-CMA, we have successfully learned linear and non-linear policies over the context of kick distances. The non-linear policy outperforms its linear counterpart, and allows a humanoid robot to kick a ball with flexible distances and with satisfactory accuracy results, which could leads to a better control and coordination in a robotic soccer match. In this research, we also demonstrated the non-linearity of a kick task. In future we will use more complex kick controllers such as dynamic motor primitives.

5 Acknowledgment

The first author was supported by FCT under grant SFRH/BD/81155/2011. The work was also partially funded by the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 and by FCT Portuguese Foundation for Science and Technology under projects PEst-OE/EEI/UI0027/2013 and UID/CEC/00127/2013 (IEETA and LIACC). The work was also funded by project EuRoC, reference 608849 from call FP7-2013-NMP-ICT-FOF.

References

1. Rui Ferreira, Luís Paulo Reis, António Paulo Moreira, and Nuno Lau. Development of an omnidirectional kick for a nao humanoid robot. In *Advances in Artificial Intelligence–*

IBERAMIA 2012, pages 571–580. Springer, 2012.
2. N. Hansen, S.D. Muller, and P. Koumoutsakos. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*, 2003.
3. Y. Sun, D. Wierstra, T. Schaul, and J. Schmidhuber. Efficient Natural Evolution Strategies. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation(GECCO)*, 2009.
4. F. Stulp and O. Sigaud. Path Integral Policy Improvement with Covariance Matrix Adaptation. In *International Conference on Machine Learning (ICML)*, 2012.
5. T. Rückstieß, M. Felder, and J. Schmidhuber. State-dependent Exploration for Policy Gradient Methods. In *Proceedings of the European Conference on Machine Learning (ECML)*, 2008.
6. S. Mannor, R. Rubinstein, and Y. Gat. The Cross Entropy method for Fast Policy Search. In *Proceedings of the 20th International Conference on Machine Learning(ICML)*, 2003.
7. E. Theodorou, J. Buchli, and S. Schaal. A Generalized Path Integral Control Approach to Reinforcement Learning. *The Journal of Machine Learning Research*, 2010.
8. A. Kupcsik, M. P. Deisenroth, J. Peters, and G. Neumann. Data-Efficient Contextual Policy Search for Robot Movement Skills. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2013.
9. A. Abdolmaleki, R. Lioutikov, J. Peters, N. Lua, L.P. Reis, and G. Neumann. Regularized covariance estimation for weighted maximum likelihood policy search methods. In *Advances in Neural Information Processing Systems (NIPS), MIT Press*, 2015.
10. Mike Depinet, Patrick MacAlpine, and Peter Stone. Keyframe sampling, optimization, and behavior integration: Towards long-distance kicking in the robocup 3d simulation league. In *RoboCup 2014: Robot World Cup XVIII*, pages 571–582. Springer, 2014.
11. Jack M Wang, David J Fleet, and Aaron Hertzmann. Optimizing walking controllers. *ACM Transactions on Graphics (TOG)*, 28(5):168, 2009.
12. Cord Niehaus, Thomas Röfer, and Tim Laue. Gait optimization on a humanoid robot using particle swarm optimization. In *Proceedings of the Second Workshop on Humanoid Soccer Robots in conjunction with the*, pages 1–7, 2007.
13. A. Abdolmaleki, N. Lua, L.P. Reis, J. Peters, and G. Neumann. Contextual Policy Search for Generalizing a Parameterized Biped Walking Controller. In *IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2015.
14. C. Daniel, G. Neumann, and J. Peters. Hierarchical Relative Entropy Policy Search. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.
15. A. Abdolmaleki, N. Lua, L.P. Reis, and G. Neumann. Regularized covariance estimation for weighted maximum likelihood policy search methods. In *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*, 2015.
16. M. Molga and C. Smutnicki. Test Functions for Optimization Needs. In *http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf*, 2005.
17. The MagmaOffenburg RoboCup 3D Simulation Team. Magma challenge tool[computer software]. *Retrieved from http://robocup.hs-offenburg.de/en/nc/downloads*.