Dimensionality Reduction and Prioritized Exploration for Policy Search

Marius Memmel TU Darmstadt
Puze Liu TU Darmstadt
Davide Tateo TU Darmstadt
Jan Peters TU Darmstadt

Abstract

Black-box policy optimization is a class of reinforcement learning algorithms that explores and updates the policies at the parameter level. This class of algorithms is widely applied in robotics with movement primitives or non-differentiable policies. Furthermore, these approaches are particularly relevant where exploration at the action level could cause actuator damage or other safety issues. However, Black-box optimization does not scale well with the increasing dimensionality of the policy, leading to high demand for samples, which are expensive to obtain in real-world systems. In many practical applications, policy parameters do not contribute equally to the return. Identifying the most relevant parameters allows to narrow down the exploration and speed up the learning. Furthermore, updating only the effective parameters requires fewer samples, improving the scalability of the method. We present a novel method to prioritize the exploration of effective parameters and cope with full covariance matrix updates. Our algorithm learns faster than recent approaches and requires fewer samples to achieve state-of-the-art results. To select the effective parameters, we consider both the Pearson correlation coefficient and the Mutual Information. We showcase the capabilities of our approach on the Relative Entropy Policy Search algorithm in several simulated environments, including robotics simulations. Code is available at git.ias.informatik.tu-darmstadt.de/ias_code/aistats2022/dr-creps.

Proceedings of the 25$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

1 INTRODUCTION

Black-Box Optimization (BBO) is a class of Policy Search methods that tackle Reinforcement Learning (RL) problems in the parameter space. These methods have been widely applied to robotics applications, such as Table Tennis (Peters et al., 2010), Beer Pong (Abdolmaleki et al., 2015), Pancake-Flipping (Kormushev et al., 2010), and Juggling (Ploeger et al., 2020). By using a search distribution, it conducts the exploration at the parameter level instead of exploring at the action level. The parameter space exploration perturbs the policy parameters at the beginning of the episode instead of adding noise to actions at each time step. This approach leads to more reliable policy updates and suits better real-world learning tasks, as the generated trajectories are smoother. Furthermore, in contrast to action level exploration, it avoids actuator damage and doesn't get low-pass filtered by the physical system (Deisenroth et al., 2013). Finally, BBO can be more suitable if the reward is not Markovian, e.g., the episodic reward only considers the maximum step reward during the trajectory.

One major challenge in BBO is the scalability in high-dimensional tasks. The search distribution of the policy parameters is often represented as a multivariate Gaussian distribution. While the search distribution parameterized by a diagonal covariance matrix is sufficient for simple tasks, more complex tasks require correlated exploration using a full covariance matrix. Furthermore, it has been shown that a full covariance matrix produces more informative samples w.r.t. the diagonal covariance and increases the learning speed (Deisenroth et al., 2013). However, the dimensionality of the full covariance matrix scales quadratically with the number of parameters. Therefore, the learning agent requires more samples to update the search distribution, because with an insufficient amount of samples the estimated covariance matrix may not be positive definite. Indeed, particularly when samples come directly from real robots, exploration is problematic in terms of time and resource costs.

To reach the optimal policy with fewer samples, we usually tackle the problem in two aspects: more informative samples or more effective policy updates. Noticeably, the policy parameters in a system generally do not have equal contribution to the return. For instance, many tasks using a 7 Degrees of Freedom (DoF) manipulator can be solved without utilizing all DoFs. We refer to the parameters that contribute most to the return as *effective parameters* and the remaining ones as *ineffective parameters*. Samples spreading over the *ineffective parameters* do not provide valuable information to the learning agent, and updating the distribution w.r.t such parameters does not have an impact on the learning performance. On the contrary, if the sampling and the policy update focus on the *effective parameters*, we could obtain better learning performance.

In this paper, we propose a new BBO algorithm, Dimensionality Reduced Constrained Relative Entropy Policy Search (DR-CREPS). Our method consists of 3 components: i. *Parameter Effectiveness Metric*. To estimate the effectiveness of the parameter, we estimate the correlation measurement (e.g., Pearson Correlation Coefficient (PCC), Mutual Information (MI)) between parameters and the episodic return. ii. *Prioritized Exploration*. During the exploration process, we concentrate the sampling on the *effective* parameters by decreasing the covariance entries w.r.t the *ineffective* parameters. iii. *Guided Dimensionality Reduction*. We propose a dimensionality reduction technique for policy search algorithms that scales to the full covariance case by partially updating the search distribution w.r.t. the effective parameters. For simplicity, we first present i. for the diagonal covariance matrix case. Then, we project the full covariance matrix into a rotated space where the covariance matrix is diagonal. Subsequently, we explain ii. and iii. in the rotated space. We update the policy in the rotated space following the Constrained Relative Entropy Policy Search (CREPS) algorithm (Abdolmaleki et al., 2017; Ploeger et al., 2020). Finally, we evaluate the proposed methodology in four continuous control and simulated robotics tasks. The empirical results show the benefit of considering effective parameters for BBO.

Assumptions and Limitations

In this work, we focus on learning the simple parametric policies, such as Movement Primitives and Linear Feedback Controller, under a multivariate Gaussian search distribution. While these policies can be high dimensional in the parameter space, each parameter may have a great impact on the policies' behavior. This class of policies includes many movement primitives used in robotics, e.g., Dynamic Movement Primitives (DMP) (Schaal, 2006) or Probabilistic Movement Primitives (ProMP) (Paraschos et al., 2013), linear policies, and ad-hoc, non-differentiable policies, but excludes the neural network scenario, where each parameter has no major impact on the policy. Thus, extending the dimensionality reduction techniques presented in this work to the neural network scenario is nontrivial and requires further investigation. One possible solution in the literature is to consider specific neural network structures (Choromanski et al., 2018).

In general, using a full covariance matrix will produce more informative samples but requires more samples for the policy updates. Our dimensionality reduction method allows us to generate informative samples and update policies with fewer samples simultaneously. For action space policies, e.g., neural networks, the correlation between the action dimensions is often less important than the temporal correlation, i.e., the correlation between different steps, and using the diagonal covariance of each time step does not significantly affect the performance.

Parameter exploration and BBO approaches might be well suited for a wide variety of tasks, e.g., when the reward function is not much informative. However, classical exploration could potentially be more sample efficient in environments where the local policy changes affect the total return, e.g., standard Mujoco environments. For this reason, we focus on specific robotics tasks where the task reward is sparse and the black box learning could be beneficial.

Related Work

Dimensionality reduction is an important approach to alleviate the problem of *"Curse of Dimensionality"*. It has been studied from various perspectives in Imitation Learning (IL) and RL, e.g., state space (Bitzer et al., 2010; Jaderberg et al., 2017), action space (Luck et al., 2014) and parameter space (Bitzer and Vijayakumar, 2009; Colomé et al., 2014). In this paper, we will focus on the dimensionality reduction problem in the parameter space.

Ben Amor et al. (2012) tackle this problem by directly projecting the movement primitives into a lower-dimensional space. In their method, they use the Principal Component Analysis (PCA) and show its application to a human grasping task. Colomé and Torras extensively studied the PCA approach in different types of movement primitives, such as DMP (Colomé and Torras, 2014; Colomé and Torras, 2018a) and ProMP (Colomé et al., 2014; Colomé and Torras, 2018b). The authors showcase the method for trajectory learning with movement primitives in various applications, including bimanual manipulation of

clothes. A different approach to dimensionality reduction for movement primitives is the reduction of the parameter space via a Mixture of Probabilistic Principal Component Analysis (MPPCA) by Tosatto et al. (2021) which makes the optimization feasible for the RL setting. Bitzer and Vijayakumar (2009) propose Gaussian Process Latent Variable Model (GPLVM) as a non-linear dimensionality reduction technique and apply it to high dimensional DMPs. Delgado-Guerrero et al. (2020) also utilize a GPLVM to project the data itself into a lower-dimensional space. They build a surrogate model of the reward function that they incentivize by weighing the data according to the MI with the reward.

Ewerton et al. (2019) exploit the connection of the PCC with the relevancy of parameters in trajectories to different objectives. They show that the computed PCC can improve the policy optimization of trajectory distributions by reducing the exploration space. To go beyond the PCC for dimensionality reduction, we also evaluate the MI as a non-linear correlation measure. The MI captures the information overlap between two random variables and allows us to measure how parameters influence the total return of an episode. Since its computation from samples is nontrivial, we turn to Carrara and Ernst (2020) and Kraskov et al. (2004) who provide suitable estimators. Another drawback of Ewerton et al. (2019)'s approach is the limitation to diagonal covariance matrices that we overcome using Singular Value Decomposition (SVD) and the importance estimation of the parameters.

Some recent works focus on scaling BBO evolutionary strategies to high dimensional parameter spaces, such as Neural Networks, by restricting the exploration space into a lower-dimensional space. Maheswaranathan et al. (2019) introduce a guided subspace in the form of a low-rank covariance matrix to reduce the search dimension, while Choromanski et al. (2019) turn to the PCA to approximate gradients in a lower-dimensional active subspace. While both approaches are limited to linear subspaces, Sener and Koltun (2020) propose to use neural networks to learn the low-dimensional manifold.

2 PROPOSED METHOD

Our method consists of three main components: First, we select the effective parameters using a correlation metric of choice. Second, we sample a new parameter vector by prioritizing exploration on the effective parameters. Third, we update the search distribution by prioritizing the important policy parameters. We provide a step-wise visualization of our methods applied to a full covariance matrix in Fig. 1.

2.1 Preliminaries

A Markov Decision Process (MDP) is defined as a 6-tuple $\mathcal{M} = \langle \mathcal{X}, \mathcal{U}, \mathcal{P}, r, \iota, \gamma \rangle$, where $\mathcal{X}$ is the state space, $\mathcal{U}$ is the action space, $\mathcal{P} : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is the transition kernel, $r : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is the reward function, $\iota : \mathcal{X} \to \mathbb{R}$ is the initial state distribution, and $\gamma \in (0, 1]$ is the discount factor.

A parametric policy π_θ, with $\boldsymbol{\theta} \in \mathbb{R}^n$ provides the action $\boldsymbol{u} \in \mathcal{U}$ to perform in each state $\boldsymbol{x} \in \mathcal{X}$. The performance of a policy, given a parameter vector $\boldsymbol{\theta}$, is evaluated through the return, i.e., the discounted cumulative reward:

$$J(\pi_\theta) = \mathop{\mathbb{E}}_{(\boldsymbol{x}_t, \boldsymbol{u}_t) \sim \pi_\theta, \mathcal{P}, \iota} \left[\sum_{t=0}^{T} \gamma^t r(\boldsymbol{x}_t, \boldsymbol{u}_t) \right],$$

where T is the length of the trajectory. Different to standard step-based RL, in the BBO setting we try to maximize the expected return $\mathcal{J}$ under a search distribution p:

$$\mathcal{J}(p) = \mathbb{E}_{\boldsymbol{\theta} \sim p} \left[J(\pi_\theta) \right].$$

In this paper, we focus on Gaussian search distributions. We define the distribution at epoch k as

$$p_k(\boldsymbol{\theta}) = \mathcal{N}\left(\cdot | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k \right),$$

with mean $\boldsymbol{\mu}_k \in \mathbb{R}^n$, and covariance $\boldsymbol{\Sigma}_k \in \mathbb{R}^{n \times n}$. To be a proper probability distribution, $\boldsymbol{\Sigma}_k$ must be positive definite.

A typical BBO approach explores the environment by sampling $\boldsymbol{\theta}$ from the search distribution p and evaluating the performance of the policy parameters by interacting with the environment. After the evaluation of N parameters, the algorithm updates the search distribution using the parameters matrix $\boldsymbol{\Theta} \in \mathbb{R}^{N \times n}$, which contains all the sampled parameter vectors, and the vector of returns $J(\boldsymbol{\Theta}) \in \mathbb{R}^N$, containing the corresponding performance of each parameter vector. In the rest of the paper, we denote the j-th component of i-th parameter sample $\boldsymbol{\theta}_i$ as θ_i^j.

To improve the robustness of the learning behavior and avoid premature convergence, many RL algorithms, such as Peters et al. (2010) and Schulman et al. (2015), apply the Kullback-Leibler Divergence (KL) constraint to the policy update. The resulting BBO optimization problem is

$$\max_{p_k} \mathcal{J}(p_k), \quad \text{s.t. } \mathrm{KL}\left(p_k \| p_{k-1}\right) \leq \epsilon. \tag{1}$$

Using the method of Lagrangian multipliers, Relative Entropy Policy Search (REPS) calculates the solution for this problem as $p_k(\boldsymbol{\theta}) \propto p_{k-1}(\boldsymbol{\theta}) \exp(J(\pi_\theta)/\eta)$, with the temperature parameter η automatically chosen to fulfill the KL constraint. As the proposed

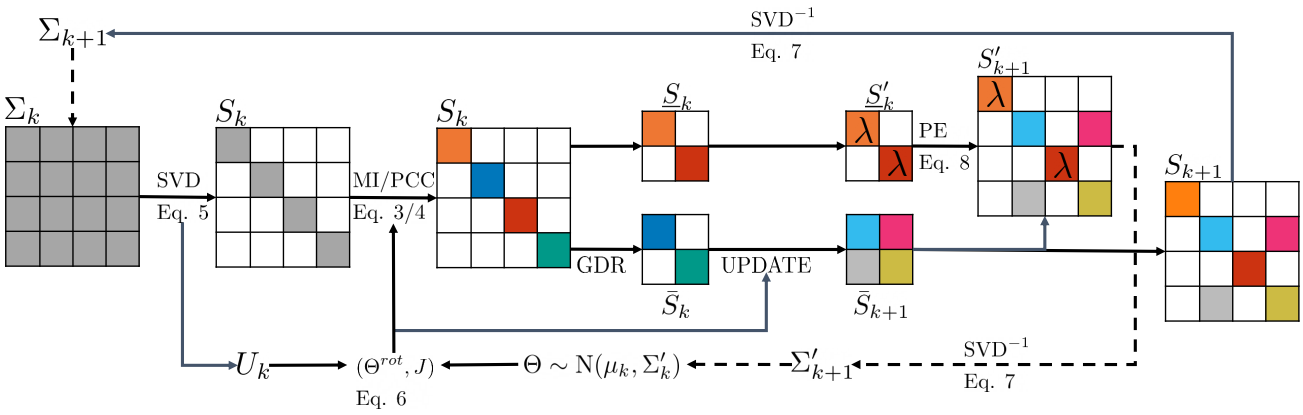


Figure 1: Guided Dimensionality Reduction (GDR) and Prioritized Exploration (PE) on a full covariance update for a generic policy search algorithm (UPDATE).

update is intractable, the authors approximate the new policy p_k using a Gaussian $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$, through sample-based Weighted Maximum Likelihood Estimate (WMLE). However, this results in a distribution that is no longer guaranteed to fulfill the original KL constraint. CREPS, introduced by Ploeger et al. (2020), addresses this problem by adding the KL constraint to the WMLE fit.

The Constrained WMLE (CWMLE) update is formulated as:

$$\max_{p_{k+1}} \sum_{i=1}^{N} d_i \log p_{k+1}(\boldsymbol{\theta}_i)$$
$$\text{s.t.} \quad \mathrm{KL}(p_k || p_{k+1}) \leq \epsilon, \quad \mathrm{H}(p_k) - \kappa \leq \mathrm{H}(p_{k+1}), \quad (2)$$

with the search distribution p_k at epoch k, the weight d_i computed by the REPS algorithm corresponding to the parameter sample $\boldsymbol{\theta}_i$, the KL bound ϵ, and the maximally allowed entropy decrease κ.

2.2 Estimating Effective Parameters

To determine the *effective parameters* and conversely the *ineffective parameters*, we need a parameter effectiveness metric to measure the influence of the policy parameters on the episodic return. Several metrics are proposed in the pioneering work, such as Pearson Correlation Coefficient for policy updates (Ewerton et al., 2019) and Mutual Information (referred to as PIC (Furuta et al., 2021)) for the task complexity evaluation.

Pearson Correlation Coefficient measures the linear relationship between two random variables that can be estimated directly with samples. The coefficient is bound by $[+1, -1]$ for a fully linear positive and negative correlation respectively. We measure the PCC of the j-th component $\boldsymbol{\Theta}^j$ with the cumulative return $\boldsymbol{J}$ based on the samples and use the absolute value as the correlation measure

$$C_{PCC}[\boldsymbol{\Theta}^j; \boldsymbol{J}] = \frac{\left|\sum_i \left(\theta_i^j - \hat{\theta}\right)\left(J_i - \hat{J}\right)\right|}{\sqrt{\sum_i \left(\theta_i^j - \hat{\theta}^j\right)^2 \sum_i \left(J_i - \hat{J}\right)^2}}, \quad (3)$$

where $\hat{\theta}^j$ and $\hat{J}$ are the respective means.

Mutual Information possesses an attractive information-theoretic interpretation and can be used to measure non-linear correlations. In theory, MI can capture a more complex relationship between two random variables than PCC. On the flipside, estimating it directly from samples is difficult, since it is defined in terms of probability densities over the random variables which require a density estimation from samples. The MI between the j-th policy parameter and the return is

$$C_{MI}[\boldsymbol{\Theta}^j; \boldsymbol{J}] = \int p\left(\boldsymbol{\theta}^j, \boldsymbol{J}\right) \log \frac{p\left(\boldsymbol{\theta}^j, \boldsymbol{J}\right)}{p\left(\boldsymbol{\theta}^j\right) p\left(\boldsymbol{J}\right)} d\boldsymbol{\theta}^j d\boldsymbol{J}. \quad (4)$$

Since a sample-based estimation of the MI is harder and has higher variance than the PCC, we provide a detailed comparison of different estimators in Sec. E of the Supplementary Materials.

After measuring the correlation between the parameters and the return, we select a fixed number m of parameters with the highest correlation and assign them to the effective class $\bar{\boldsymbol{\varepsilon}}$. The remaining parameters are considered as the ineffective class $\underline{\boldsymbol{\varepsilon}}$.

2.3 The Full Covariance Case

As presented by Ewerton et al. (2019), finding the corresponding effectiveness or relevance of the parameters is trivial for the diagonal covariance case. However, this approach neglects the inter-dimensional dependencies and leads to limitations in the policy complexity and the learning efficiency. We, therefore, consider the full covariance case and propose an algorithm to overcome these limitations.

Identifying the effectiveness of the parameters in the full covariance matrix becomes nontrivial due to the relationships between them. To achieve independence among the parameters, we transform the covariance matrix into a space where every dimension is uncorrelated. This can be achieved using the SVD, which decomposes the covariance matrix into two rotation matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ as well as a diagonal matrix $\boldsymbol{S}$ with the singular values on the diagonal. More formally, decomposing covariance matrix $\boldsymbol{\Sigma}_k$ gives us

$$\boldsymbol{\Sigma}_k = \boldsymbol{U}_k \boldsymbol{S}_k \boldsymbol{V}_k^T, \tag{5}$$

with $\boldsymbol{U}_k, \boldsymbol{S}_k, \boldsymbol{V}_k \in \mathbb{R}^{n \times n}$. $\boldsymbol{\Sigma}_k$ is a positive definite symmetric matrix, therefore we have $\boldsymbol{U}_k = \boldsymbol{V}_k$.

We then define a policy in the rotated space as

$$p_k^{rot} = \mathcal{N}\left(\cdot | \boldsymbol{\mu}_k^{\text{rot}}, \boldsymbol{S}_k\right),$$

with $\boldsymbol{\mu}_k^{\text{rot}} = \mathbf{0} \in \mathbb{R}^n$, $\boldsymbol{S}_k = \operatorname{diag}\left(\sigma_0^2, ..., \sigma_n^2\right) \in \mathbb{R}^{n \times n}$. Since the parameters are no longer correlated in the rotated space, it again becomes trivial to identify the covariance entries corresponding to the effective parameters.

To estimate the contribution of the parameters to the total return, we must also rotate each parameter sample $\boldsymbol{\theta}$ by

$$\boldsymbol{\theta}^{rot} = \boldsymbol{U}_k^T \left(\boldsymbol{\theta} - \boldsymbol{\mu}_k\right), \tag{6}$$

which projects them into the rotated space of $\boldsymbol{S}_k$. We use the projected samples to compute the correlation between the diagonal of $\boldsymbol{S}_k$ and $\boldsymbol{J}(\boldsymbol{\Theta})$.

The introduced transformations allow us to work on a rotated policy parameterized by a diagonal covariance matrix. Note that we have to recompute the SVD at every epoch since updating $\boldsymbol{S}_k$ (in Sec. 2.5) can lead to a non-diagonal structure. Through the re-computation, we guarantee a diagonal structure of the covariance matrix before every modification or update of it. Assuming an update of $(\boldsymbol{\mu}_k^{rot}, \boldsymbol{S}_k)$ returning $(\boldsymbol{\mu}_{k+1}^{rot}, \boldsymbol{S}_{k+1})$, we must project the rotated policy p_{k+1}^{rot} back to the original space for the policy evaluations. We utilize the previously computed $\boldsymbol{U}_k$ to project back the updated mean and the updated covariance matrix

$$\boldsymbol{\mu}_{k+1} = \boldsymbol{\mu}_k + \boldsymbol{U}_k \boldsymbol{\mu}_{k+1}^{\text{rot}}, \quad \boldsymbol{\Sigma}_{k+1} = \boldsymbol{U}_k \boldsymbol{S}_{k+1} \boldsymbol{U}_k^T, \tag{7}$$

yielding our new policy p_{k+1}.

2.4 Prioritized Exploration

In the following paragraph, we describe our Prioritized Exploration (PE) approach. Having identified the contribution to the total return of each parameter in the rotated space, we split them up into sets of effective $\bar{\boldsymbol{\varepsilon}}$ and ineffective parameters $\underline{\boldsymbol{\varepsilon}}$ with fixed size each. Both sets $\bar{\boldsymbol{\varepsilon}}, \underline{\boldsymbol{\varepsilon}}$ capture the effectiveness of the parameters in the rotated space, i.e., similarity between $\boldsymbol{\Theta}^{\text{rot}}$ and $\boldsymbol{J}(\boldsymbol{\Theta})$ corresponding to the elements on the diagonal of $\boldsymbol{S}$. Without loss of generality, we decompose the total diagonal covariance matrix $\boldsymbol{S}$ into covariance matrices w.r.t. effective parameters and ineffective parameters denoted as $\bar{\boldsymbol{S}} \in \mathbb{R}^{m \times m}, \underline{\boldsymbol{S}} \in \mathbb{R}^{(n-m) \times (n-m)}$, respectively. We use this separation to prioritize the exploration on the effective parameters by reducing exploration to the ineffective ones. Therefore, we scale the ineffective covariance $\underline{\boldsymbol{S}}$ by hyperparameter $\lambda \in (0, 1)$ during the sampling process described as

$$\underline{\boldsymbol{S}}' = \lambda \underline{\boldsymbol{S}}. \tag{8}$$

Then, we substitute the covariance matrix corresponding to the ineffective parameter $\underline{\boldsymbol{S}}$ by the modified one $\underline{\boldsymbol{S}}'$, i.e. $\boldsymbol{S}' = \textsc{subst}(\boldsymbol{S}, \underline{\boldsymbol{S}}', \underline{\boldsymbol{\varepsilon}})$. The SUBST function is simply placing the modified entries of $\underline{\boldsymbol{S}}'$ at their original position in $\boldsymbol{S}$ as visualized in Fig. 1.

After the substitution, sampling $\boldsymbol{\Theta}$ involves the modified search distribution $\mathcal{N}(\cdot | \boldsymbol{\mu}, \boldsymbol{S}')$ parameterized by the modified covariance matrix $\boldsymbol{S}'$ and the mean $\boldsymbol{\mu}$ of the true search distribution $\mathcal{N}(\cdot | \boldsymbol{\mu}, \boldsymbol{S})$. After obtaining the dataset, we update the policy with an arbitrary policy search algorithm. Note that we estimate the parameter effectiveness before each sampling step. Therefore, we only use the modified $\boldsymbol{S}'$ for the sampling process and always update the unmodified $\boldsymbol{S}$. Otherwise, the covariance would shrink over time since $\lambda \in (0, 1)$.

2.5 Guided Dimensionality Reduction

To tackle the scalability issue of policy search algorithms, we further exploit our estimation of effective parameters $\bar{\boldsymbol{\varepsilon}}$ to introduce Guided Dimensionality Reduction (GDR). Congruent with our intuition about the varying contribution of parameters to the total return, we now only update the effective ones that promise to increase the total return the most. We again focus on the rotated space and the transformation described in Sec. 2.3.

Running a full covariance policy search update on $\bar{\boldsymbol{S}}_k$ with $\boldsymbol{\Theta}^{rot}$, and $J(\boldsymbol{\Theta})$ gives us the updated covariance matrix $\bar{\boldsymbol{S}}_{k+1} \in \mathbb{R}^{m \times m}$. Note that $\bar{\boldsymbol{S}}_{k+1}$ is no longer guaranteed to be diagonal due to the full covariance update. Then, we compose the new covariance matrix $\boldsymbol{S}_{k+1}$ in the rotated space by substituting the entries of $\bar{\boldsymbol{S}}_k$ with $\bar{\boldsymbol{S}}_{k+1}$, i.e., $\boldsymbol{S}_{k+1} = \textsc{subst}(\boldsymbol{S}_k, \bar{\boldsymbol{S}}_{k+1}, \bar{\boldsymbol{\varepsilon}})$. We can then project $\boldsymbol{S}_{k+1}$ back into the original space via an inversion of the SVD resulting in $\boldsymbol{\Sigma}_{k+1}$. Analogously, we update $\bar{\boldsymbol{\mu}}_k^{\text{rot}} = \mathbf{0}$ corresponding to the effective parameters to obtain $\bar{\boldsymbol{\mu}}_{k+1}^{\text{rot}}$. Substituting the updated values back into $\boldsymbol{\mu}_k^{rot}$ we get

Algorithm 1 DR-CREPS

procedure UPDATE($\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \boldsymbol{\Theta}, \boldsymbol{J}$)
  $\boldsymbol{U_k}, \boldsymbol{S_k} \leftarrow \text{SVD}(\boldsymbol{\Sigma}_k)$
  $\boldsymbol{\Theta}^{\text{rot}} \leftarrow$ SAMPLEPROJ($\boldsymbol{U}_k, \boldsymbol{\mu}_k, \boldsymbol{\Theta}$), Eq. (6)
  $\boldsymbol{C} \leftarrow$ CORRELATION($\boldsymbol{\Theta}^{\text{rot}}, \boldsymbol{J}$), Eq. (3) or (4)
  $\bar{\boldsymbol{\varepsilon}}, \underline{\boldsymbol{\varepsilon}} \leftarrow$ IDENTIFYEFFECTIVEPARAMETERS($\boldsymbol{C}$)
  $\eta^* \leftarrow$ MINIMIZEDUALFUNCTION($\boldsymbol{\Theta}^{\text{rot}}, \boldsymbol{J}$)
  $\boldsymbol{d}_\theta \leftarrow \exp\left(\frac{\boldsymbol{J} - \max \boldsymbol{J}}{\eta^*}\right)$
  $\boldsymbol{\mu}_{k+1}, \boldsymbol{\Sigma}_{k+1} \leftarrow$ DRCWMLE($\boldsymbol{U}_k, \boldsymbol{S}_k, d_\theta, \bar{\boldsymbol{\varepsilon}}$)
  return $\boldsymbol{\mu}_{k+1}, \boldsymbol{\Sigma}_{k+1}$
end procedure

procedure DRCWMLE($\boldsymbol{U}_k, \boldsymbol{S}_k, d_\theta, \bar{\boldsymbol{\varepsilon}}$)
  $\bar{\boldsymbol{\mu}}_k^{rot}, \bar{\boldsymbol{S}}_k \leftarrow$ GETEFFECTIVEDIST($\boldsymbol{S}_k, \bar{\boldsymbol{\varepsilon}}$)
  $\bar{\boldsymbol{\mu}}_{k+1}^{rot}, \bar{\boldsymbol{S}}_{k+1} \leftarrow$ CWMLE($\bar{\boldsymbol{\mu}}_k^{rot} \bar{\boldsymbol{S}}_k, d_\theta$)
  $\boldsymbol{\mu}_{k+1}^{rot} \leftarrow$ SUBST($\boldsymbol{\mu}_k^{rot}, \bar{\boldsymbol{\mu}}_{k+1}^{rot}, \bar{\boldsymbol{\varepsilon}}$)
  $\boldsymbol{S}_{k+1} \leftarrow$ SUBST($\boldsymbol{S}_k, \bar{\boldsymbol{S}}_{k+1}, \bar{\boldsymbol{\varepsilon}}$)
  $\boldsymbol{\mu}_{k+1}, \boldsymbol{\Sigma}_{k+1} \leftarrow$ BACKPROJ($\boldsymbol{\mu}_{k+1}^{rot}, \boldsymbol{S}_{k+1}$), ▷ Eq. (7)
  return $\boldsymbol{\mu}_{k+1}, \boldsymbol{\Sigma}_{k+1}$
end procedure

procedure SAMPLEPE($\boldsymbol{U}_k$, $\boldsymbol{S}_{k+1}$, $\underline{\boldsymbol{\varepsilon}}$)
  $\underline{\boldsymbol{S}}'_{k+1} \leftarrow \lambda \underline{\boldsymbol{S}}_{k+1}$, Eq. (8)
  $\boldsymbol{S}'_{k+1} \leftarrow$ SUBST($\boldsymbol{S}_{k+1}, \underline{\boldsymbol{S}}'_{k+1}, \underline{\boldsymbol{\varepsilon}}$)
  $\boldsymbol{\Sigma}'_{k+1} \leftarrow \boldsymbol{U}_k \boldsymbol{S}'_{k+1} \boldsymbol{U}_k^T$
  $\boldsymbol{\Theta} \sim \mathcal{N}(\cdot|\boldsymbol{\mu}_{k+1}, \boldsymbol{\Sigma}'_{k+1})$
  $\boldsymbol{J} \leftarrow \text{Rollout}(\boldsymbol{\Theta})$
  return $\boldsymbol{\Theta}$, $\boldsymbol{J}$
end procedure

$\boldsymbol{\mu}_{k+1}^{rot} = \text{SUBST}(\boldsymbol{\mu}_k^{rot}, \bar{\boldsymbol{\mu}}_{k+1}^{rot}, \bar{\boldsymbol{\varepsilon}})$. Projecting it back to the original space we yield $\boldsymbol{\mu}_{k+1}$.

Since we choose $m \ll n$, updating the covariance matrix in the rotated space requires fewer samples than in the original space. Updating only the effective parameters lets us make the most efficient use of these samples without sacrificing performance. Reducing the number of samples makes it viable to use a full covariance matrix, even for high dimensional tasks, assuming an appropriate number of effective parameters m is chosen. Furthermore, this approach complements PE perfectly since it maintains a diagonal structure of the ineffective parameters. Assuming similar estimations of the parameter effectiveness at each epoch, the samples would not contain much useful information to update the ineffective parameters as a result of the reduced exploration in exactly those.

2.6 Practical Implementation

The proposed techniques can be applied to a wide variety of BBO algorithms. In Alg. 1 we show how the dimensionality reduction for the full covariance case can be adapted to the CREPS algorithm. In Fig. 1 we provide a visual intuition of the algorithm update and sampling technique. For simplicity, we omit the hyperparameters ϵ and κ, i.e. the KL bound and entropy decrease bound parameters, needed to solve the optimization problems described in Eq. (1) and Eq. (2).

Each update step starts by projecting the full covariance matrix into a space where all dimensions are uncorrelated using SVD. We also rotate the parameter samples $\boldsymbol{\Theta}$ into that space by applying the obtained square rotation matrix $\boldsymbol{U}_k$. The procedure CORRELATION uses the rotated samples $\boldsymbol{\Theta^{rot}}$ to compute the $\boldsymbol{C}_{MI}/\boldsymbol{C}_{PCC}$ of each parameter w.r.t. $\boldsymbol{J}$. The procedure IDENTIFYEFFECTIVEPARAMETERS determines the set of effective parameters $\bar{\boldsymbol{\varepsilon}}$ and ineffective parameters $\underline{\boldsymbol{\varepsilon}}$. To obtain the optimal value of the Lagrangian multiplier η^*, we minimize the dual function of REPS and use η^* as temperature parameter to compute the weight vector $\boldsymbol{d}_\theta$ for the CWMLE.

To perform CWMLE on the reduced dimensionality, we first call the procedure GETEFFECTIVEDISTRIBUTION to create a new diagonal distribution $\mathcal{N}(\cdot|\bar{\boldsymbol{\mu}}_k^{rot}, \bar{\boldsymbol{S}}_{\boldsymbol{k}})$ based on the effective parameters $\bar{\boldsymbol{\varepsilon}}$. We then apply the policy search update using the previously computed weights $\boldsymbol{d}_\theta$. Next, we substitute the updated distribution parameters $(\bar{\boldsymbol{\mu}}_{k+1}^{rot}, \bar{\boldsymbol{S}}_{k+1})$ back to obtain the $\mathcal{N}(\cdot|\boldsymbol{\mu}_{k+1}^{rot}, \boldsymbol{S}_{k+1})$. Finally, we rotate the resulting distribution back to the original space yielding a new search distribution $p_{k+1}(\cdot) = \mathcal{N}(\cdot|\boldsymbol{\mu}_{k+1}, \boldsymbol{\Sigma}_{k+1})$.

We apply PE by multiplying $\underline{\boldsymbol{S}}_{k+1}$ with λ before the substitution into $\boldsymbol{S}_{k+1}$. Since $\bar{\boldsymbol{S}}$ and $\underline{\boldsymbol{S}}$ are uncorrelated in the rotated space, scaling $\underline{\boldsymbol{S}}$ by λ will not affect $\bar{\boldsymbol{S}}$. We rotate the sampling covariance $\boldsymbol{S}'_{k+1}$ back to the original space via $\boldsymbol{\Sigma}'_{k+1} = \boldsymbol{U}_k \boldsymbol{S}'_{k+1} \boldsymbol{U}_k^T$ and obtain the prioritized samples from $\mathcal{N}(\cdot|\boldsymbol{\mu}_{k+1}, \boldsymbol{\Sigma}'_{k+1})$. As described in Sec. 2.4, we only use $\boldsymbol{\Sigma}'_{k+1}$ for the sampling process, while we still consider $\boldsymbol{\Sigma}_{k+1}$ as target distribution for the KL and entropy bounds.

3 EXPERIMENTAL EVALUATION

In this section, we evaluate the proposed method in four different environments. All our implementations are based on the MushroomRL library (D'Eramo et al., 2021). In all experiments, we use 25 random seeds for evaluation. For each learning curve, we plot the mean and 95% confidence interval. We set a fixed number of episodes for each epoch. The performance is evaluated at the end of each epoch by sampling parameters from the current search distribution.

First, we test our algorithm in a modified LQR problem where only a part of the controller parameters is

effective. We construct a 10-dimensional LQR corresponding to a 10×10 weight matrix of the linear regressor, i.e., 100 parameters. The effective parameters are the non-zero entries of the optimal gain matrix which we can compute iteratively. Then, we demonstrate several robotic simulations including *ShipSteering* (steering a ship through a gate), *AirHockey* (shooting a goal in a game of air hockey) described in (Liu et al., 2021), and *BallStopping* (stopping a ball from rolling off a table) visualized in Fig. 2a, Fig. 2b, and Fig. 2c, respectively.

We use a multivariate Gaussian distribution as search distribution and a deterministic linear policy. This policy can be a linear controller (*LQR*), a CMAC approximator (Albus, 1975) with rectangular tiles (*ShipSteering*), or a ProMP (*AirHockey*, *BallStopping*).

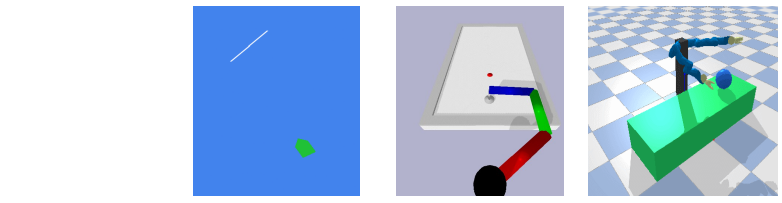
We compare our approach mainly to BBO approaches. As the main baseline, we select CREPS and Model-Based Relative Entropy Stochastic Search (MORE) (Abdolmaleki et al., 2015) representing policy search algorithms shown to outperform Reward-Weighted Regression (RWR) and REPS. We include in our comparison also two gradient-based deep actor-critic methods, Trust Region Policy Optimization (TRPO) (Schulman et al., 2015) and Proximal Policy Optimization (PPO) (Schulman et al., 2017), using neural policies. Lastly, we also compare our method with Natural Evolution Strategies (NES) (Wierstra et al., 2014), an evolutionary search strategy. Additional baselines can be found in the appendix.

The Supplementary Materials provides detailed information on the environmental setup in Sec. A, hyperparameter sweep in Sec. B, and an ablation study on the number of selected parameters m and λ in Sec. D.1.

3.1 Effect of Prioritized Exploration

To better analyze the behavior of our approach, we introduce a 10-dimensional Linear Quadratic Regulator (LQR) environment characterized by three effective and seven ineffective dimensions.

We begin our evaluation on the modified LQR and use the diagonal covariance matrix to showcase the performance of PE without GDR. In Fig. 3, PE shows significantly faster learning when applied to both REPS and CREPS. Especially REPS with PE and $\lambda = 0.1$ profits from the focus on the effective parameters. Reducing the ineffective parameters' covariance to a mere 10% causes the learning curve to jump during the initial epochs. This is a clear sign that the effective parameters were leveraged. However, for REPS this behavior is too greedy, causing premature convergence as also experienced by Pearson-Correlation-Based Relevance Weighted Policy Optimization (PRO) (Ewerton et al., 2019) and RWR (Deisenroth et al., 2013). Tuning hyperparameter λ to a higher value allows for some exploration in the other parameters compensating potential mistakes in the parameter identification which would otherwise cause a too greedy behavior. Choosing $\lambda = 0.9$ makes REPS with PE converge to the optimal policy while still outperforming the vanilla version. Note that CREPS with PE does not suffer from this issue due to the additional entropy constraint designed to counteract an excessive greedy behavior, avoiding premature convergence.



(a) ShipSteering (b) AirHockey (c) BallStopping

Figure 2: Test environments

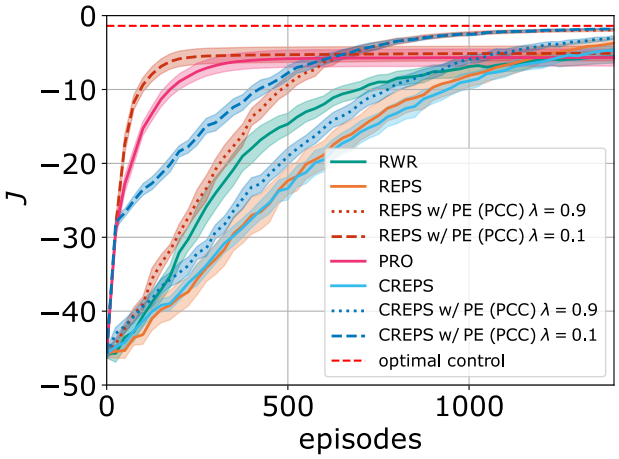


Figure 3: LQR - DiagCov

3.2 Effect of Dimensionality Reduction with Full Covariance Matrix

We demonstrate the learning performance in the LQR environment with full covariance matrix. When using this covariance parametrization, more episodes are required to maintain the positive definiteness during the update. We, therefore, also include the number of episodes per policy update in the parameter sweep for each algorithm.

In Fig. 4, we show the best learning performance of each algorithm. MORE requires 250 episodes per policy update, CREPS requires 150 episodes, TRPO and PPO requires 100 episodes each, NES requires 200

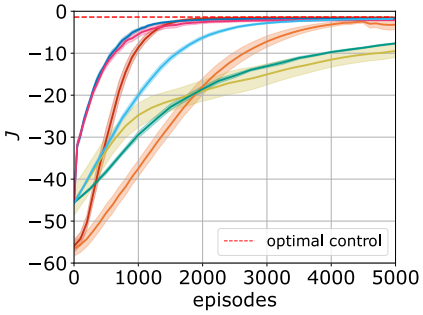


Figure 4: LQR - FullCov

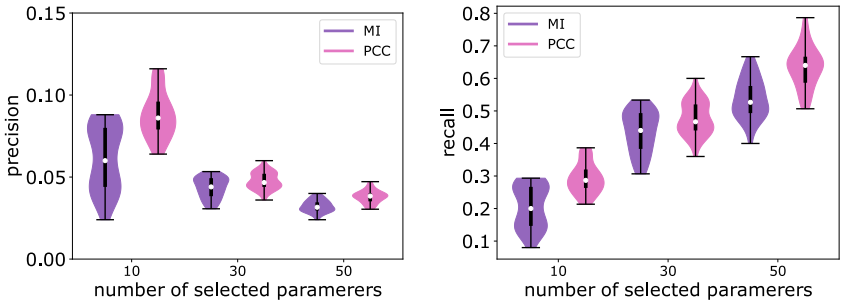
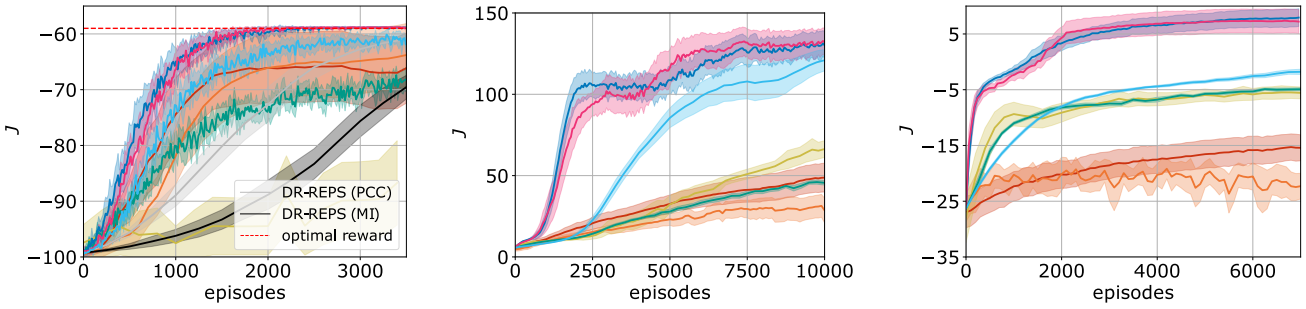


Figure 5: Precision and recall of effective parameter estimation



(a) ShipSteering (b) AirHockey (c) BallStopping

Figure 6: Experimental results on the simulated environments. DR-CREPS learns faster and converges to better optima than vanilla CREPS and MORE.

episodes, and DR-CREPS requires only 50 episodes. Consequently, DR-CREPS is able to perform more policy updates than MORE and CREPS, resulting in better learning performance. The GDR combined with PE can further exploit the additional updates by focusing the exploration on the effective parameters. The combination of both techniques leads to further improvements in the learning performance. For a more detailed ablation on the relationship between those methods please refer to Fig. 10 of the Supplementary Materials.

3.3 Comparison between MI and PCC

We empirically evaluate the effect of MI and PCC as the correlation measures for selecting effective parameters in DR-CREPS on all environments. Both experience similar learning and achieve similar performance, as visualized in Fig. 4 and Fig. 6. To further investigate the performance, we again focus on the modified LQR environment where it is easy to impose a set of effective parameters. In this modified LQR environment, 3 parameters out of 100 are effective parameters. We estimate PCC from Eq. (3) and the MI with scikit-learn's mutual information regressor (Pedregosa et al., 2011). We compute the precision and recall w.r.t. the identified effective parameters and show our results for different numbers of selected parameters $\{10, 30, 50\}$ in Fig. 5. PCC thereby outperforms MI by a short margin on all tested configurations which we accredit to the easier estimation of PCC which does not require density estimation of the distributions over the parameters and the return. Computing the PCC with fewer samples is therefore expected to achieve a better estimation than MI. Besides reducing the number of samples required for each update, our objective is also to reduce the dimensionality by assuming a small number of effective parameters. This lets us tend towards selecting less parameters, i.e., $20-50\%$ of the total parameters. These propensities explain why PCC and MI have similar learning performances in all experiments with PCC being slightly superior. We provide a more elaborate ablation study of the correlation measure as a comparison to a random selection of the parameters in Sec. D.2 of the Supplementary Materials. In most environments, the number of ground truth effective parameters is generally not known. It is possible that the number of selected effective parameters is much less than the real number of parameters. Misidentified parameters, however, can prevent overconfident sampling and overly restrictive policy updates, which may lead to premature convergence.

3.4 Learning in Simulated Robotics Environments

The simulated robotic environments are not only more difficult to solve than the LQR but at the same time enable to validate the existence of effective parameters in real-world applications. In all environments, GDR allows updating the policy with fewer episodes than the original CREPS. Together with PE this results in DR-CREPS learning a good policy after only a fraction of the learning epochs.

The *ShipSteering* environment is characterized by a high dimensional policy, i.e., 450 parameters. Learning a full covariance matrix while keeping the positive definiteness using REPS would require an unmanageable number of episodes, and it is therefore not feasible. However, if we combine REPS with GDR, we can run the algorithm with only 250 episodes per update. We call the combination of these two approaches Dimensionality Reduced Relative Entropy Policy Search (DR-REPS). The results in Fig. 6a show that DR-REPS learns a policy on par with CREPS and MORE, while the DR-CREPS algorithm outperforms all other methods.

In *AirHockey*, DR-CREPS learns to score a goal after approx. 2500 episodes indicated by the jump in the total return. Subsequently, it improves its reliability to shoot a goal in the subsequent episodes. MORE, on the other hand, does not show the characteristic jump which is an indicator of the policy not scoring a goal. Here, small differences in the collision between two rounded-shaped objects, i.e., the puck and the end effector, result in drastically different trajectories. As a result, the cumulative return has a high variance shown in the learning curve.

DR-CREPS can intercept the ball after only approx. 500 episodes in the *BallStopping* environment, and the final policy learns to reliably stop it. Observing the learned policies, the advantage of our approach is clear: CREPS moves the arm towards the ball but fails to reliably stops it from rolling off the table while MORE completely fails to move the robotic arm towards the ball or moving the arm too rapidly, propelling the ball back of the table. These behaviors are a result of exploration in the ineffective parameters which cause the arm to act almost randomly and either move it away from the table or let it experience jerky movements. Instead, DR-CREPS reduces the exploration in the ineffective parameters thus, it almost completely prevents those faulty movements leading to a smoother behavior that reliably stops the ball. The GDR then further speeds up the learning by only updating the effective parameters, i.e., the ones required to move the end effector in front of the ball.

To show the relevance of our approach, we also include Deep Reinforcement Learning (DRL) baselines in our evaluation, which use Deep Neural Networks as policies. These approaches show good learning performances on the *LQR* and the *ShipSteering* tasks, while they fail on more difficult ones like *AirHockey* and *BallStopping*. Indeed, in robotic tasks, a structured policy rather than a neural network can significantly reduce the task complexity, and consequently, reduce the number of episodes required to solve the task. A similar observation holds for the NES algorithm, which requires even more episodes to maintain a population large enough to perform meaningful updates. Our approach outperforms gradient-based and random search methods by combining more efficient sampling techniques and policy update methods with structured policies. Please refer to Sec. C in the Supplementary Materials for additional comparison with other baselines.

4 CONCLUSIONS

In this paper, we propose Guided Dimensionality Reduction, an approach to dimensionality reduction for policy search algorithms in BBO that enables more efficient learning of policies parameterized by full covariance matrices. We utilize the Pearson Correlation Coefficient and the Mutual Information to estimate effective parameters that contribute most to the total return. By updating only those, we reduce the number of samples required for each update allowing our algorithms to boost the learning performance. Furthermore, we present Prioritized Exploration which focuses the exploration on the effective parameters and thereby reduces the exploration space. We empirically demonstrate both methods on DR-CREPS, an adaptation of CREPS, in four different environments including simulated robotics. DR-CREPS outperforms recent approaches to dimensionality reduction in BBO. However, our experimental evaluation focuses mainly on robotic tasks with structured policies. Further investigation is needed to assess the scalability of our approach to Deep Network policies and to see how our method performs in comparison with more recent evolutionary strategy approaches. Anyway, structured policies are currently the state-of-the-art control method for many robotics applications, and our approach is a clean, simple, and intuitive way to improve learning performance in these scenarios. Future work will focus on bringing the proposed methods to real-world robots and evaluating different parameter effectiveness metrics.

Acknowledgements

The support provided by China Scholarship Council (No. 201908080039) is acknowledged.

References

Abdolmaleki, A., Lioutikov, R., Peters, J. R., Lau, N., Pualo Reis, L., and Neumann, G. (2015). Model-based relative entropy stochastic search. In *NeurIPS*.

Abdolmaleki, A., Price, B., Lau, N., Reis, L. P., and Neumann, G. (2017). Deriving and improving cmaes with information geometric trust regions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 657–664.

Albus, J. S. (1975). A new approach to manipulator control: The cerebellar model articulation controller (cmac). *Journal of Dynamic Systems, Measurement and Control*.

Ben Amor, H., Kroemer, O., Hillenbrand, U., Neumann, G., and Peters, J. (2012). Generalization of human grasping for multi-fingered robot hands. In *IROS*.

Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*. Springer.

Bitzer, S., Howard, M., and Vijayakumar, S. (2010). Using dimensionality reduction to exploit constraints in reinforcement learning. In *IROS*.

Bitzer, S. and Vijayakumar, S. (2009). Latent spaces for dynamic movement primitives. In *Humanoids*.

Carrara, N. and Ernst, J. (2020). On the estimation of mutual information. In *Multidisciplinary Digital Publishing Institute Proceedings*.

Choromanski, K., Rowland, M., Sindhwani, V., Turner, R., and Weller, A. (2018). Structured evolution with compact architectures for scalable policy optimization. In *ICML*.

Choromanski, K. M., Pacchiano, A., Parker-Holder, J., Tang, Y., and Sindhwani, V. (2019). From complexity to simplicity: Adaptive es-active subspaces for blackbox optimization. In *NeurIPS*.

Colomé, A., Neumann, G., Peters, J., and Torras, C. (2014). Dimensionality reduction for probabilistic movement primitives. In *Humanoids*.

Colomé, A. and Torras, C. (2014). Dimensionality reduction and motion coordination in learning trajectories with dynamic movement primitives. In *IROS*.

Colomé, A. and Torras, C. (2018a). Dimensionality reduction for dynamic movement primitives and application to bimanual manipulation of clothes. *IEEE Transactions on Robotics*.

Colomé, A. and Torras, C. (2018b). Dimensionality reduction in learning gaussian mixture models of movement primitives for contextualized action selection and adaptation. *IEEE Robotics and Automation Letters*.

Deisenroth, M. P., Neumann, G., Peters, J., et al. (2013). A survey on policy search for robotics. *Foundations and trends in Robotics*.

Delgado-Guerrero, J. A., Colomé, A., and Torras, C. (2020). Sample-efficient robot motion learning using gaussian process latent variable models. In *ICRA*.

D'Eramo, C., Tateo, D., Bonarini, A., Restelli, M., and Peters, J. (2021). Mushroomrl: Simplifying reinforcement learning research. *JMLR*.

Ewerton, M., Arenz, O., Maeda, G., Koert, D., Kolev, Z., Takahashi, M., and Peters, J. (2019). Learning trajectory distributions for assisted teleoperation and path planning. *Frontiers in Robotics and AI*.

Fletcher, R. (1987). *Practical methods of optimization*. John Wiley & Sons.

Furuta, H., Matsushima, T., Kozuno, T., Matsuo, Y., Levine, S., Nachum, O., and Gu, S. S. (2021). Policy Information Capacity: Information-Theoretic Measure for Task Complexity in Deep Reinforcement Learning. In *ICML*.

Ghavamzadeh, M. and Mahadevan, S. (2003). Hierarchical policy gradient algorithms. *ICML*.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. (2017). Reinforcement learning with unsupervised auxiliary tasks. In *ICLR*.

Kormushev, P., Calinon, S., and Caldwell, D. G. (2010). Robot motor skill coordination with em-based reinforcement learning. In *IROS*.

Kozachenko, L. and Leonenko, N. (1987). Sample estimate of the entropy of a random vector. *Probl. Peredachi Inf.*

Kraskov, A., Stögbauer, H., and Grassberger, P. (2004). Estimating mutual information. *Phys. Rev. E*.

Liu, P., Tateo, D., Ammar, H. B., and Peters, J. (2021). Robot reinforcement learning on the constraint manifold. In *CoRL*.

Luck, K. S., Neumann, G., Berger, E., Peters, J., and Amor, H. B. (2014). Latent space policy search for robotics. In *IROS*.

Maheswaranathan, N., Metz, L., Tucker, G., Choi, D., and Sohl-Dickstein, J. (2019). Guided evolutionary strategies: augmenting random search with surrogate gradients. In *ICML*.

Paraschos, A., Daniel, C., Peters, J. R., and Neumann, G. (2013). Probabilistic movement primitives. In *NeurIPS*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *JMLR*, 12:2825–2830.

Peters, J., Mülling, K., and Altün, Y. (2010). Relative entropy policy search. In *AAAI*.

Ploeger, K., Lutter, M., and Peters, J. (2020). High acceleration reinforcement learning for real-world juggling with binary rewards. In *CoRL*.

Ross, B. C. (2014). Mutual information between discrete and continuous data sets. *PLOS ONE*.

Rubinstein, R. (1999). The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*.

Schaal, S. (2006). Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*. Springer Tokyo.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *ICML*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sener, O. and Koltun, V. (2020). Learning to guide random search. In *ICLR*.

Tosatto, S., Chalvatzaki, G., and Peters, J. (2021). Contextual latent-movements off-policy optimization for robotic manipulation skills. In *ICRA*.

Wierstra, D., Schaul, T., Glasmachers, T., Sun, Y., Peters, J., and Schmidhuber, J. (2014). Natural evolution strategies. *JMLR*.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*.

Supplementary Material: Dimensionality Reduction and Prioritized Exploration for Policy Search

A ENVIRONMENT DESCRIPTIONS

In the following, we describe the environments used for testing the algorithms. If not further specified, we use the default parameters of MushroomRL (D'Eramo et al., 2021). For implementation details please visit git.ias.informatik.tu-darmstadt.de/ias_code/aistats2022/dr-creps.

A.1 Linear Quadratic Regulator

The LQR is characterized by a linear system with quadratic rewards. The state transition and reward are defined as follows:

$$x_{t+1} = Ax_t + Bu_t \qquad\qquad r_t = -(x_t^T Q x_t + u_t^T R u_t)$$

where x_t is the state, u_t is the control signal, and r_t is the reward at time step t. Matrix A denotes the state dynamics, B the action dynamics and, Q and R the reward weights for state and action respectively. All four matrices are diagonal and we limit the actions and states to a maximum value of 1. to further simplify the task. To simulate a real-world example where some parameters do not contribute to the objective, we randomly set elements on the diagonal of Q and corresponding B close to zero (1e-20). These dimensions then represent the ineffective parameters. We choose a 10-dimensional LQR with 7 ineffective and 3 effective dimensions, i.e., we replace 7 values at the same positions on each diagonal. Since the matrices remain non-singular, we can solve for the optimal control policy iteratively and assess the convergence of the algorithms to the optimal solution.

A.2 ShipSteering

In the ship steering task, the agent has to maneuver a ship through a gate placed at a fixed position (Ghavamzadeh and Mahadevan, 2003). Initial position, orientation, and turning rate are chosen randomly at the start of each epoch and the ship has to maintain a constant speed while moving. The state is represented by the position (x and y) and the orientation. We transform those features using three rectangular tilings over the state dimension with the number of tiles being 5, 5, and 6, respectively. It is also straightforward to compute the optimal return by taking the shortest route from start to gate. We include this return in our plots to assess algorithm performance.

A.3 AirHockey

The air hockey environment features a planar robot arm with 3 DoF whose objective is to hit the puck coming towards it and score a goal on the opponents' side of the field. During an episode, the end effector has to stay on the table and we terminate an episode if it leaves the table boundaries. We simulate with a horizon of 120 steps.

A.4 BallStopping

In this environment, a ball is placed on a table and starts rolling off of it. The goal then is to stop it from doing so using a 7 DoF robotic arm. A penalty is given when the ball drops off the table with a discount factor decreasing this penalty over time. We run an episode for a horizon of 750, which marks the time when the ball would've dropped off the table without any intervention by the robot. The state consists of the joint position and velocity as well as the orientation and position of the ball, which results in a 21-dimensional observation space.

B EXPERIMENT DETAILS

In this section, we describe the sweep over the hyperparameters as well as the parameters selected for the plots in the paper. First, we describe the parametric policies as well as the search distributions for each environment in Tab. 1. For the sweep we define [inclusive start|exclusive end|step size] to denote the range of parameters explored. Besides Tab. 2 and Tab. 5 which represent diagonal covariance cases, all rest tables in this section represent the full covariance case.

For the diagonal covariance case, we use the same number of episodes for all algorithms, since we do not apply GDR yet. For full covariance matrices, we sweep multiples of 5 (starting from 15) and select the lowest possible number of episodes per fit that leads to a satisfaction of the constraints and does not violate the positive definiteness of the covariance matrix. In general, we search algorithm-specific parameters $(\beta, \epsilon, \kappa)$ on the vanilla versions of REPS and CREPS, and use those as fixed parameters for the sweep on the remaining hyperparameters of DR-REPS and DR-CREPS. For assessing PE, we keep the hyperparameter values from the corresponding DR-CREPS and just remove the exploration modification.

The standard algorithmic implementation of CREPS does not always fulfill the positive definiteness of the covariance matrix, particularly for high values of the KL bound ϵ. This happens due to an insufficient number of samples w.r.t. to a high dimensional covariance matrix, leading to computational errors. If this problem appears, we check the corresponding parameter values for DR-CREPS. In the case of *ShipSteering*, DR-CREPS allows selecting a higher ϵ, without violating the positive definiteness, than the one used by CREPS.

Table 1: Search distribution and corresponding parametric policy for each environment.

	LQR	ShipSteering	AirHockey	BallStopping
parametric policy	linear regressor	linear regressor	ProMP	ProMP
bases	-	-	radial basis function	radial basis function
bases/dimension	-	-	30	20
basis width	-	-	0.001	0.001
parameters	100	450	90	140
search distribution	$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
$\boldsymbol{\mu}_{\text{init}}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
$\boldsymbol{\Sigma}_{\text{init}} = diag(\boldsymbol{\sigma}_0, ..., \boldsymbol{\sigma}_n)$	$\boldsymbol{\sigma} = 0.3$	$\boldsymbol{\sigma} = 0.07$	$\boldsymbol{\sigma} = 1.0$	$\boldsymbol{\sigma} = 1.0$

Table 2: Hyperparameters for Fig. 3, LQR - Diagonal Covariance Matrix

	RWR	PRO	REPS	REPS w/ PE	CREPS	CREPS w/PE
Sweep						
β	[0.1|1.5|0.1]	-	-	-	-	-
ϵ	-	-	[0.1|1.5|0.1]	-	[1.5|3.5|0.2]	-
κ	-	-	-	-	[1.|10.|1.]	-
m	-	-	-	[5|100|5]	-	[5|100|5]
γ	-	-	-	[0.1|1.0|0.1]	-	[0.1|1.0|0.1]
Experiment						
β	0.2	0.2	-	-	-	-
ϵ	-	-	0.4	0.4	2.5	2.5
κ	-	-	-	-	6.0	6.0
DR	no	no	no	no	no	no
PE	no	no	no	yes	no	yes
m	-	-	-	30	-	30
γ	-	-	-	0.1/0.9	-	0.1
correlation measure	-	PCC	-	MI	-	MI
number of epochs	80	80	80	80	80	80
episodes per fit	25	25	25	25	25	25

Table 3: Hyperparameters for Fig. 4 and Fig. 7, LQR - Full Covariance Matrix

	MORE	CREPS	DR-CREPS (PCC)	DR-CREPS w/o PE (PCC)	DR-CREPS (MI)	DR-CREPS w/o PE (MI)
Sweep						
ϵ	[1.9|6.0|0.2]	[1.9|6.0|0.2]	-	-	-	-
κ	[1.|20.|1.]	[1.|20.|1.]	-	-	-	-
m	-	-	[5|100|5]	-	[5|100|5]	-
γ	-	-	[0.1|1.0|0.1]	-	[0.1|1.0|0.1]	-
Experiment						
ϵ	4.7	4.7	4.7	4.7	4.7	4.7
κ	17.0	17.0	17.0	17.0	17.0	17.0
DR	no	no	yes	yes	yes	yes
PE	no	no	yes	no	yes	no
m	-	-	50	50	50	50
γ	-	-	0.1	0.1	0.1	0.1
correlation measure	-	-	PCC	PCC	MI	MI
number of epochs	20	33	100	100	100	100
episodes per fit	250	150	50	50	50	50

Table 4: Hyperparameters for Fig. 4 and Fig. 7, LQR - Full Covariance Matrix

	TRPO	PPO	REINFORCE	NES	CEM
Sweep					
actor lr	-	[3e-2, 3e-3, 3e-4]	-	-	-
citic lr	[3e-2, 3e-3, 3e-4]	[3e-2, 3e-3, 3e-4]	-	-	-
max KL	[1e-0, 1e-1, 1e-2]	-	-	-	-
optimizer step size ϵ	-	-	[1e-1, 1e-2, 1e-3]	-	-
optimizer lr	-	-	-	[3e-1, 3e-2, 3e-3]	-
population size	-	-	-	[50, 100, 200, 500]	-
Experiment	-	-	-	-	-
actor lr	-	3e-3	-	-	-
citic lr	3e-3	3e-3	-	-	-
max KL	1e-0	-	-	-	-
optimizer step size ϵ	-	-	1e-2	-	-
optimizer lr	-	-	-	3e-2	-
rollouts	-	-	-	2	-
population size	-	-	-	100	-
elites	-	-	-	-	25
number of epochs	50	50	50	200	100
episodes per fit	100	100	100	25	50

Table 5: Hyperparameters for Fig. 5, Precision and Recall for Effective Parameter Estimation

	DR-CREPS
Experiment	
ϵ	4.5
κ	15.
DR	no
PE	yes
m	10/30/50
γ	0.1
correlation measure	PCC/MI
number of epochs	40
episodes per fit	50

Table 6: Hyperparameters for Fig. 6a and Fig. 7, ShipSteering - Full Covariance Matrix

	DR-REPS (PCC)	DR-REPS (MI)	MORE	CREPS	DR-CREPS (PCC)	DR-CREPS (MI)
Sweep						
ϵ	-	-	[2.4|6.4|1.0]	[1.9|3.9|0.5]	[2.4|4.4|1.0]	[2.4|4.4|1.0]
κ	-	-	-	[14|30|3]	-	
m	[100|500|50]	[100|500|50]	-	-	[100|500|50]	[100|500|50]
γ	[0.1|0.9|0.2]	[0.1|0.9|0.2]	-	-	[0.1|0.9|0.2]	[0.1|0.9|0.2]
Experiment						
ϵ	0.5	0.5	4.4	2.4	3.4	3.4
κ	-	-	20.	20.	20.	20.
DR	yes	yes	no	no	yes	yes
PE	yes	yes	no	no	yes	yes
m	100	100	-	-	200	200
γ	0.1	0.1	-	-	0.1	0.1
correlation measure	PCC	MI	-	-	PCC	MI
number of epochs	14	14	233	233	233	233
episodes per fit	250	250	15	15	15	15

Table 7: Hyperparameters fo Fig. 6a and Fig. 7, ShipSteering - Full Covariance Matrix

	TRPO	PPO	REINFORCE	NES	CEM
Sweep					
actor lr	-	[3e-2, 3e-3, 3e-4]	-	-	-
citic lr	[3e-2, 3e-3, 3e-4]	[3e-2, 3e-3, 3e-4]	-	-	-
max KL	[1e-0, 1e-1, 1e-2]	-	-	-	-
optimizer step size ϵ	-	-	[1e-1, 1e-2, 1e-3]	-	-
optimizer lr	-	-	-	[3e-1, 3e-2, 3e-3]	-
population size	-	-	-	[50, 100, 200, 500]	-
Experiment	-	-	-	-	-
actor lr	-	3e-4	-	-	-
citic lr	3e-2	3e-4	-	-	-
max KL	1e-2	-	-	-	-
optimizer step size ϵ	-	-	1e-3	-	-
optimizer lr	-	-	-	3e-2	-
rollouts	-	-	-	2	-
population size	-	-	-	100	-
elites	-	-	-	-	25
number of epochs	35	35	35	200	70
episodes per fit	100	100	100	17	50

Table 8: Hyperparameters for Fig. 6b and Fig. 7, AirHockey - Full Covariance Matrix

	MORE	CREPS	DR-CREPS (PCC)	DR-CREPS (MI)
Sweep				
ϵ	[1.4|8.4|1.0]	[0.4|3.4|0.5]	-	-
κ	[6.|20.|2.]	[6.|20.|2.]	-	-
m	-	-	[10|90|10]	[10|90|10]
γ	-	-	[0.1|0.9|0.2]	[0.1|0.9|0.2]
Experiment				
ϵ	2.4	2.0	2.0	2.0
κ	12.	12.	12.	12.
DR	no	no	yes	yes
PE	no	no	yes	yes
m	-	-	30	30
γ	-	-	0.5	0.5
correlation measure	-	-	PCC	MI
number of epochs	40	40	200	200
episodes per fit	250	250	50	50

Table 9: Hyperparameters for Fig. 6b and Fig. 7, AirHockey - Full Covariance Matrix

	TRPO	PPO	REINFORCE	NES	CEM
Sweep					
actor lr	-	[3e-2, 3e-3, 3e-4]	-	-	-
citic lr	[3e-2, 3e-3, 3e-4]	[3e-2, 3e-3, 3e-4]	-	-	-
max KL	[1e-0, 1e-1, 1e-2]	-	-	-	-
optimizer step size ϵ	-	-	[1e-1, 1e-2, 1e-3]	-	-
optimizer lr	-	-	-	[3e-1, 3e-2, 3e-3]	-
population size	-	-	-	[50, 100, 200, 500]	-
Experiment	-	-	-	-	-
actor lr	-	3e-3	-	-	-
citic lr	3e-3	3e-3	-	-	-
max KL	1e-1	-	-	-	-
optimizer step size ϵ	-	-	1e-1	-	-
optimizer lr	-	-	-	3e-1	-
rollouts	-	-	-	2	-
population size	-	-	-	100	-
elites	-	-	-	-	25
number of epochs	100	100	100	200	200
episodes per fit	100	100	100	50	50

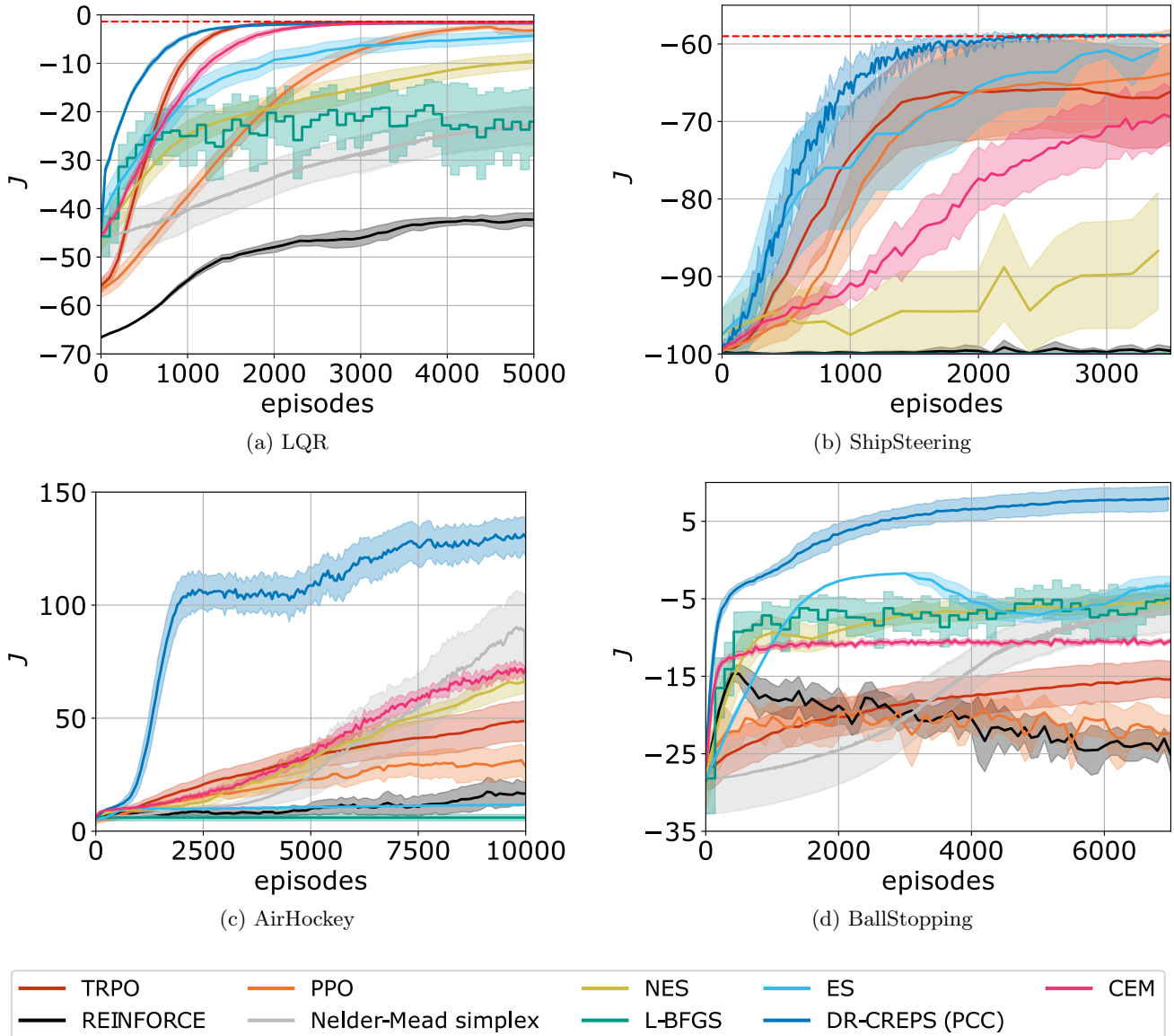
Table 10: Hyperparameters for Fig. 6c and Fig. 7, BallStopping - Full Covariance Matrix

	MORE	CREPS	DR-CREPS (PCC)	DR-CREPS (MI)
Sweep				
ϵ	-	[1.5|6.0|0.5]	-	-
κ	-	[5|30|5]	-	-
m	-	-	[15|140|15]	[15|140|15]
γ	-	-	[0.1|1.0|0.5]	[0.1|1.0|0.5]
Experiment				
ϵ	4.5	4.5	4.5	4.5
κ	20.	20.	20.	20.
DR	no	no	yes	yes
PE	no	no	yes	yes
m	-	-	30	30
γ	-	-	0.5	0.5
correlation measure	-	-	PCC	MI
number of epochs	28	28	116	116
episodes per fit	250	250	60	60

Table 11: Hyperparameters for Fig. 6c and Fig. 7, BallStopping - Full Covariance Matrix

	TRPO	PPO	REINFORCE	NES	CEM
Sweep					
actor lr	-	[3e-2, 3e-3, 3e-4]	-	-	-
citic lr	[3e-2, 3e-3, 3e-4]	[3e-2, 3e-3, 3e-4]	-	-	-
max KL	[1e-0, 1e-1, 1e-2]	-	-	-	-
optimizer step size ϵ	-	-	[1e-1, 1e-2, 1e-3]	-	-
optimizer lr	-	-	-	[3e-1, 3e-2, 3e-3]	-
population size	-	-	-	[50, 100, 200, 500]	-
Experiment	-	-	-	-	-
actor lr	-	3e-3	-	-	-
citic lr	3e-2	3e-3	-	-	-
max KL	1e-0	-	-	-	-
optimizer step size ϵ	-	-	1e-3	-	-
optimizer lr	-	-	-	3e-1	-
rollouts	-	-	-	2	-
population size	-	-	-	100	-
elites	-	-	-	-	25
number of epochs	100	100	100	200	140
episodes per fit	70	70	70	35	50

C ADDITIONAL BASELINES



(a) LQR

(b) ShipSteering

(c) AirHockey

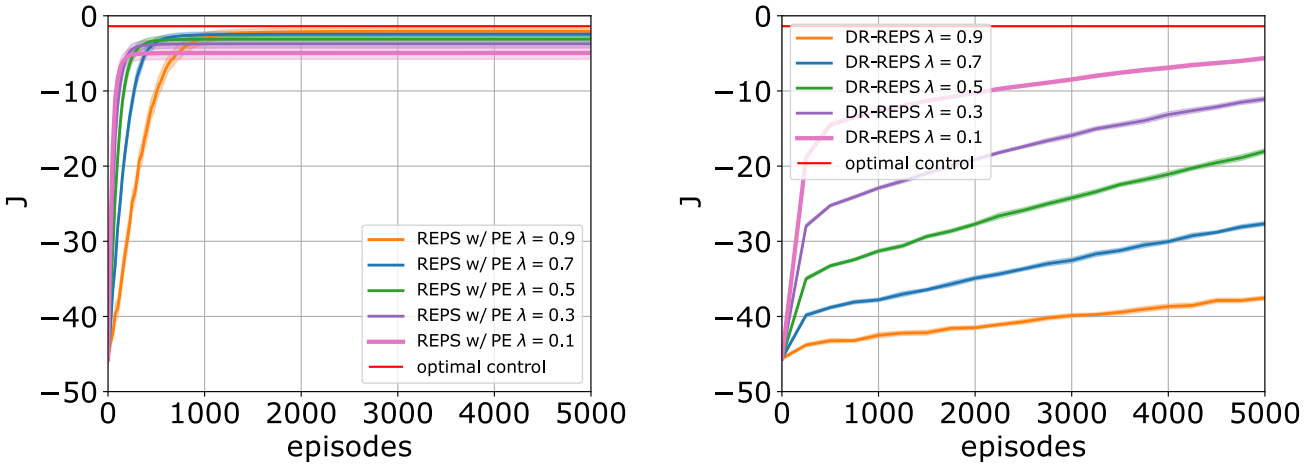
(d) BallStopping

Figure 7: Additional baselines for all environments.

We provide an additional comparison of our method to gradient-based methods (TRPO (Schulman et al., 2015), PPO (Schulman et al., 2017), REINFORCE (Williams, 1992)), evolutionary strategies (Evolution Strategies (ES), NES (Wierstra et al., 2014)), the Cross-entropy Method (CEM) (Rubinstein, 1999), and classic optimization algorithms (Nelder-Mead Simplex (NM), Limited-memory BFGS (L-BFGS) (Fletcher, 1987)) in Fig. 7. Details on the hyperparameter sweep are given in Sec. B. Nonetheless, our proposed approach outperforms all baselines by a significant margin with respect to the speed of learning and the final learned policy.

D ABLATION STUDIES

To investigate the contributions of each of our components, we conduct an ablation study. We analyze the influence of hyperparameter λ on the PE and examine its interplay with the number of selected parameters. Finally, we compare the MI and the PCC to a random selector as means of selecting the effective parameters. If not further specified, we use the hyperparameters described in Sec. B.



(a) LQR - DiagCov
30 selected parameters

(b) LQR - FullCov
10 selected parameters

(c) LQR - FullCov
30 selected parameters

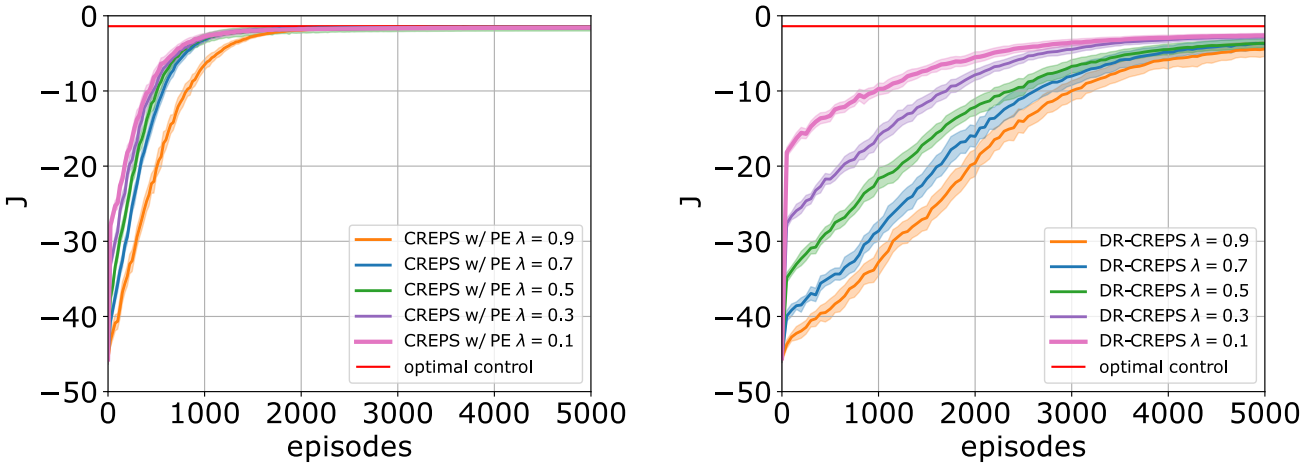
(d) LQR - FullCov
50 selected parameters

Figure 8: Ablation study on the LQR with the DR-REPS algorithm.

D.1 Influence of the number of selected parameters m and hyperparameter λ

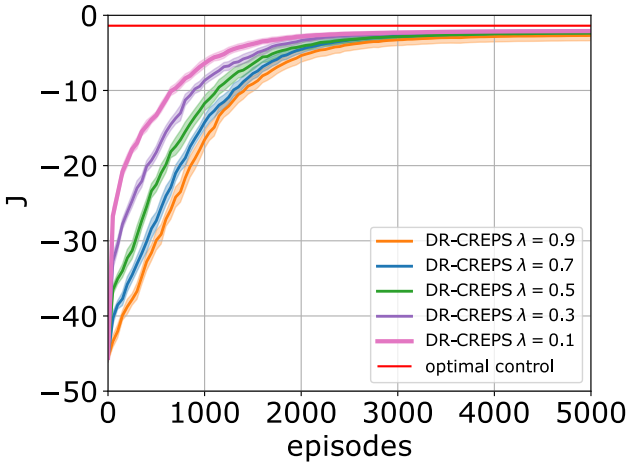
In Fig. 8 and Fig. 9 we show the results of our ablation study on the LQR environment for the DR-REPS and DR-CREPS algorithm, respectively. We plot the results for the number of selected parameters separately and vary the value for λ inside each plot.

On the diagonal case in Fig. 8a, hyperparameter λ seems to correlate directly with the initial learning speed with significant jumps during the first epochs. This jump comes from focusing the exploration on only the effective parameters and, in the case of $\lambda = 0.1$, nearly completely neglecting the ineffective ones by reducing their covariance to a mere 10% of the original value. In this experiment, $\lambda = 0.9$ is too close to the behavior without PE i.e, $\lambda = 1.0$, to show a significantly improved learning performance. These findings get amplified when moving to the full covariance case in Fig. 8b, Fig. 8c, and Fig. 8d when we extend the algorithms with GDR. The less effective parameters we select, the larger the initial gain in learning. This gain is because we apply the full step size ϵ to a subset of the parameters. Especially when selecting only 10 parameters, we observe large jumps at the beginning, which shows that updating only a subset of the parameter, i.e., the effective ones, and focusing the exploration leads to a great boost for the learning process. Selecting fewer parameters, however, comes with the downside of a slightly worse optimum reached. A reason for this might be that too large update steps on too few parameters prevent the algorithm from exploring some effective parameters that would lead to convergence to the optimum.

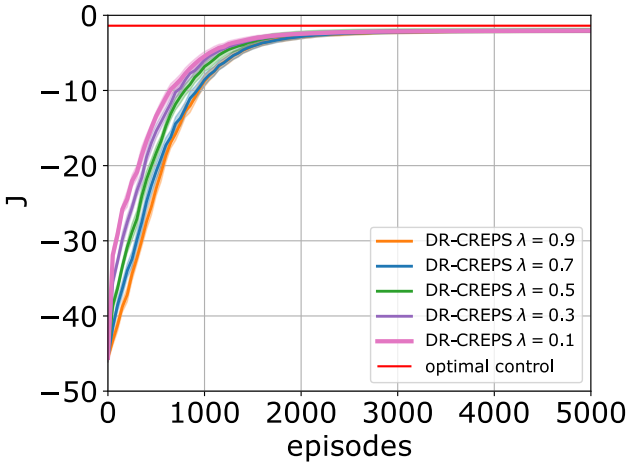


(a) LQR - DiagCov
30 selected parameters

(b) LQR - FullCov
10 selected parameters



(c) LQR - FullCov
30 selected parameters

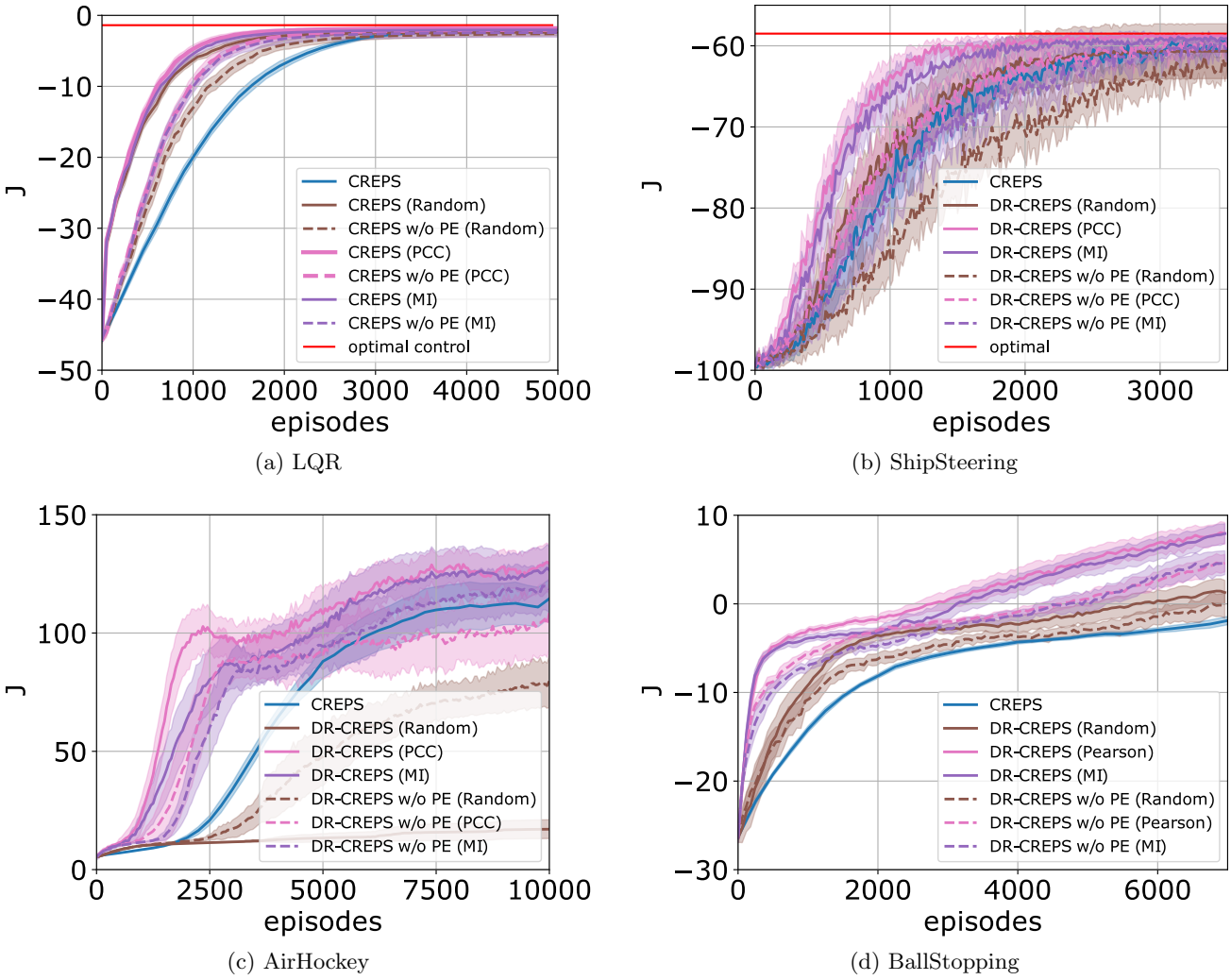


(d) LQR - FullCov
50 selected parameters

Figure 9: Ablation study on the LQR with the DR-CREPS algorithm.

To prevent premature convergence CREPS introduces the entropy constraint. Since REPS does not include such, the algorithm experiences premature convergence more easily and we have to be cautious when selecting a value for λ that is too close to 0. This is clearly shown in Fig. 9a which presents the diagonal covariance case on the LQR. The closer λ is to 0, the larger the initial performance gain but also the loss of final return. Updating only the effective parameters seems again to magnify this tradeoff as visualized in Fig. 9b, Fig. 9c, and Fig. 9d, respectively. In the given number of episodes, $\lambda = 0.1$ reaches a better optimum. However, the last episodes of Fig. 9d hint that higher values might reach a better optimum but require significantly more episodes to do so. This effect is also only really impactful when 50 parameters, i.e., 50% of the total number of parameters, are selected, which is not feasible for larger dimensional problems as the number of episodes required per update becomes unfeasible.

Finally, our hyperparameter sweeps in Tab. 8 and Tab. 10 show that ProMP are also quite sensitive to λ and the best performance/learning tradeoff is achieved at $\lambda = 0.5$. Therefore, we can not always select λ as low as possible, but we should search for the best hyperparameter. However, our findings suggest that for a linear regressor $\lambda = 0.1$, and ProMPs $\lambda = 0.5$ seem like good and mostly sufficient starting points.



(a) LQR

(b) ShipSteering

(c) AirHockey

(d) BallStopping

Figure 10: Comparison of MI, PCC, and a random selector (Random) to select the effective parameters.

D.2 Comparison between Mutual Information, Pearson Correlation Coefficient, and Random

To verify that using MI and PCC benefits the parameter selection process, we compare it to a random selector that represents the most primitive baseline. We conduct this part of the ablation study in all environments.

The results on the LQR shown in Fig. 10a suggest that our method already accounts for some of the performance gains and can outperform the vanilla CREPS in terms of learning speed even when using a random selector. We attribute this to the fact that reducing the covariance of some subset of parameters limits the current exploration space and makes it easier to learn. If some effective parameters are selected using a correlation metric or even randomly, these benefits increase. The former is to be preferred as the correlation metric picks parameters associated with higher rewards. The experiments on the simulated environments support these assumptions.

In *ShipSteering*, the main contribution to the learning speed comes from PE. GDR does not have a significant impact besides reducing the number of episodes per fit. Updating arbitrary parameters makes the updates noisy and leads to worse behavior than CREPS. The reason for this behavior is the observation space which is transformed via rectangular tilings. Since those are binary, updating a subspace does not provide additional gains as only the parameters corresponding to the activated tiles are updated. PE, however, helps to focus the exploration on only the effective parameters which correspond to the tiles required to find a good or even optimal path leading to faster learning. The random selection is not sufficient and provides no guarantee of convergence indicated by the large confidence interval.

The *AirHockey* task shows a high instability of both the random selector as well as GDR without exploration steering noticeable on the high confidence interval. We further document failure rates due to a violation of the positive definiteness of the covariance matrix of 16% DR-CREPS w/o PE (MI), 48% DR-CREPS w/o PE (PCC), 44% DR-CREPS w/o PE (Random), and 24% DR-CREPS w/o PE (Random). This result is similar to the one in *ShipSteering* where PE allows us to run larger update steps using fewer episodes. To reduce the failure rate it's possible to either decrease ϵ or increase the number of episodes per update. Since both solutions would perturb the comparison, we stick to reporting the failure rates instead. Nonetheless, the random selector shows this faulty behavior even when PE is applied.

The results on the *BallStopping* task in Fig. 10d support the findings on the LQR showing that the correlation measure methods outperform the random selector, even if the latter improves the learning process.

In conclusion, the above-mentioned experiments show that using the correlation measures as inductive bias significantly improves the performance over a random selector.

E MUTUAL INFORMATION ESTIMATION

We investigate three ways to estimate the mutual information. Therefore, we construct a toy example with two random variables $\boldsymbol{X}, \boldsymbol{Y}$ and noise $\boldsymbol{E}$ all Gaussian distributed which allow for an analytical estimation of the MI. According to Bishop and Nasrabadi (2006) exploiting the properties of multivariate Gaussian distributions, we get the following marginal, joint, and conditional distributions which are also Gaussian distributed:

$$
\begin{aligned}
p(\boldsymbol{E}) &= N(\boldsymbol{e}|\boldsymbol{\mu_e}, \boldsymbol{\Sigma_e}) \\
p(\boldsymbol{X}) &= N(\boldsymbol{x}|\boldsymbol{\mu_{xx}}, \boldsymbol{\Sigma_{xx}}) \\
p(\boldsymbol{Y}|\boldsymbol{X}) &= N(\boldsymbol{y}|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{\mu_e}, \boldsymbol{A}\boldsymbol{\Sigma_{xx}}\boldsymbol{A}^T + \boldsymbol{\Sigma_e}) \\
p(\boldsymbol{Y}) &= N(\boldsymbol{y}|\boldsymbol{A}\boldsymbol{\mu_{xx}}, \boldsymbol{\Sigma_{yx}} + \boldsymbol{A}\boldsymbol{\Sigma_{xx}}\boldsymbol{A^T}) \\
p(\boldsymbol{X}|\boldsymbol{Y}) &= N(\boldsymbol{x}|\boldsymbol{\Sigma_{xy}}\{\boldsymbol{A^T}\boldsymbol{\Sigma_{yx}^{-1}}(\boldsymbol{y} - \boldsymbol{\mu_e}) + \boldsymbol{\Sigma_{xx}^{-1}}\boldsymbol{\mu_{xx}}\}, \boldsymbol{\Sigma_{xy}}) \\
&\textit{where} \quad \boldsymbol{\Sigma_{xy}} = (\boldsymbol{\Sigma_{xx}^{-1}} + \boldsymbol{A^T}\boldsymbol{\Sigma_{yx}^{-1}}\boldsymbol{A})^{-1}
\end{aligned}
$$

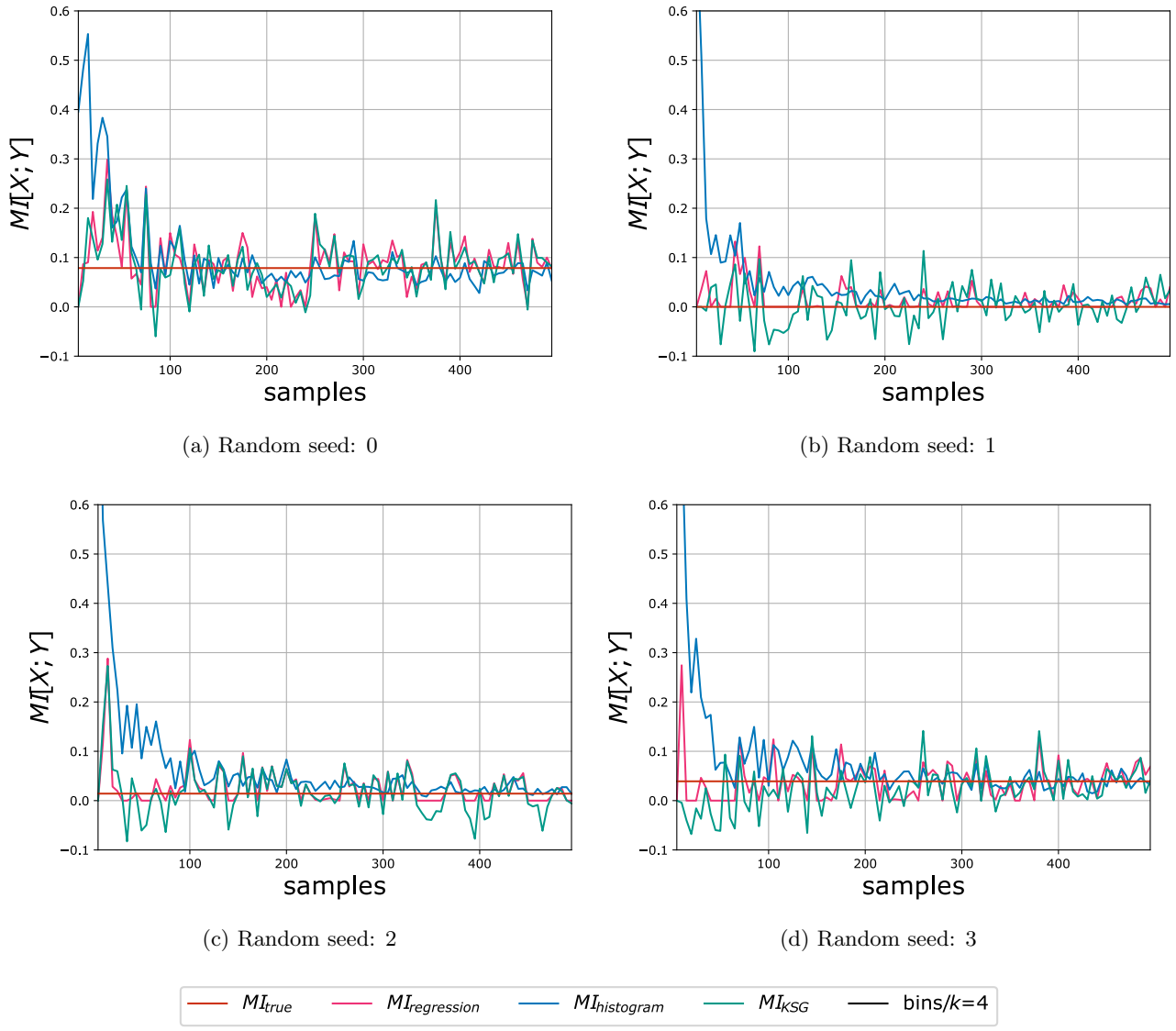
If $\boldsymbol{X}$ has dimensionality M and $\boldsymbol{Y}$ has dimensionality N, and $\boldsymbol{E}$ has dimensionality N, $\boldsymbol{A}$ denotes a transformation matrix of full rank with dimensionality $N \times M$. In this case $\boldsymbol{A}, \boldsymbol{\mu_e}, \boldsymbol{\mu_{xx}}$ are the parameters governing the means. We compute the MI analytically as well as through three sample-based estimators.

First, we take the trivial approach of estimating the MI through its probabilistic formulation referenced in Eq. 4 which we refer to as $\boldsymbol{MI_{histogram}}$. We use histograms to estimate the densities of the underlying probability distributions and subsequently compute their entropy $H(\cdot)$. The relation of the entropy to the MI is as follows:

$$
\begin{aligned}
MI[\boldsymbol{X}; \boldsymbol{Y}] &= H(\boldsymbol{X}) - H(\boldsymbol{X}|\boldsymbol{Y}) \\
&= H(\boldsymbol{Y}) - H(\boldsymbol{Y}|\boldsymbol{X}) \\
&= H(\boldsymbol{X}) + H(\boldsymbol{X}) + H(\boldsymbol{X}, \boldsymbol{Y}) \\
&= H(\boldsymbol{X}, \boldsymbol{Y}) - H(\boldsymbol{X}|\boldsymbol{Y}) - H(\boldsymbol{Y}|\boldsymbol{X})
\end{aligned}
$$

Second, we turn to the estimator proposed by *Kraskov, Stögbauer, Grassberger* (KSG) (Kraskov et al., 2004) referred to as $\boldsymbol{MI_{KSG}}$. Lastly, we evaluate scikit-learn's mutual information regressor (Pedregosa et al., 2011) referred to as $\boldsymbol{MI_{regression}}$. The implementation utilizes a combination of (Ross, 2014; Kraskov et al., 2004) which are both based on (Kozachenko and Leonenko, 1987).

We choose hyperparameter *bins* per histogram ($MI_{histogram}$) and k nearest neighbors ($MI_{regression}$, MI_{ksg}) to reflect our goal of using fewer samples per update and estimation step. Both parameters must be less than the number of samples used for the estimation. We found that $MI_{histogram}$ is more sensitive to *bins* while $MI_{regression}$ and MI_{ksg} produce similar results with varying k. We choose values of $bins = 4$ and $k = 4$ as a tradeoff of bias and performance (Pedregosa et al., 2011).



(a) Random seed: 0

(b) Random seed: 1

(c) Random seed: 2

(d) Random seed: 3

Figure 11: Comparison of different MI estimators on our toy problem over multiple random seeds.

For visualization purposes, we evaluate the estimators in the simplified case of $M = 1, N = 1$ and compute results for 4 rollouts. For each rollout, we randomize the Gaussian noise $p(\boldsymbol{E})$, sampling process $\boldsymbol{X} \sim p(\boldsymbol{x}), \boldsymbol{E} \sim p(\boldsymbol{E})$, and transformation matrix $\boldsymbol{A}$ while using the same samples from $p(\boldsymbol{X})$.

Fig. 11 shows the results of this experiment. Since each dimension experiences a different transformation and noise the analytical solutions and estimates for the MI vary between them. The $MI_{histogram}$ based approach requires far more samples to achieve a stable estimate of the MI compared to the other methods. As expected, $MI_{regression}$ and MI_{ksg} show similar behavior because they use the same underlying estimator. However, $MI_{regression}$ is limited to estimates ≥ 0 and standardizes the samples. These changes lead to a more stable estimation especially in the low samples regime of $0-200$ which is similar to the range we use for our DR-CREPS experiments. These findings support our decision of using the $MI_{regression}$ estimator with hyperparameter $k = 4$ for all experiments in the main paper.