Sparse Latent Space Policy Search

Kevin Sebastian Luck
Arizona State University
Interactive Robotics Lab
AZ 85281 Tempe, USA
mail@kevin-luck.net

Joni Pajarinen
Aalto University
Intelligent Robotics Group
02150 Espoo, Finland
Joni.Pajarinen@aalto.fi

Erik Berger
Technical University Bergakademie Freiberg
Institute of Computer Science
09599 Freiberg, Germany
erik.berger@informatik.tu-freiberg.de

Ville Kyrki
Aalto University
Intelligent Robotics Group
02150 Espoo, Finland
ville.kyrki@aalto.fi

Heni Ben Amor
Arizona State University
Interactive Robotics Lab
AZ 85281 Tempe, USA
hbenamor@asu.edu

Abstract

Computational agents often need to learn policies that involve many control variables, e.g., a robot needs to control several joints simultaneously. Learning a policy with a high number of parameters, however, usually requires a large number of training samples. We introduce a reinforcement learning method for sample-efficient policy search that exploits correlations between control variables. Such correlations are particularly frequent in motor skill learning tasks. The introduced method uses Variational Inference to estimate policy parameters, while at the same time uncovering a low-dimensional latent space of controls. Prior knowledge about the task and the structure of the learning agent can be provided by specifying groups of potentially correlated parameters. This information is then used to impose sparsity constraints on the mapping between the high-dimensional space of controls and a lower-dimensional latent space. In experiments with a simulated bi-manual manipulator, the new approach effectively identifies synergies between joints, performs efficient low-dimensional policy search, and outperforms state-of-the-art policy search methods.

Introduction

Reinforcement learning (RL) is a promising approach to automated motor skill acquisition (Peters et al. 2011). Instead of a human hand-coding specific controllers, an agent autonomously explores the task at hand through trial-and-error and learns necessary movements. Yet, reinforcement learning of motor skills is also considered to be a challenging problem, since it requires sample-efficient learning in high-dimensional state and action spaces. A possible strategy to address this challenge can be found in the human motor control literature (Bernstein 1967). Research on human motor control provides evidence for *motor synergies*; joint co-activations of a set of muscles from a smaller number of neural commands. The reduction in involved parameters results in a lower-dimensional latent space for control which, in turn, reduces cognitive effort and training time during skill acquisition. The existence of synergies has been reported in a variety of human motor tasks, e.g., grasping (Santello, Flanders, and Soechting 1998), walking (Wang, O'Dwyer, and Halaki 2013), or balancing (Torres-Oviedo and Ting 2010).

Recently, various synergy-inspired strategies have been put forward to improve the efficiency of RL for motor skill acquisition (Bitzer, Howard, and Vijayakumar 2010; Kolter and Ng 2007). Typically, these approaches use dimensionality reduction as a pre-processing step in order to extract a lower-dimensional latent space of control variables. However, extracting the latent space using standard dimensionality reduction techniques requires a significantly large training set of (approximate) solutions, prior simulations, or human demonstrations. Even if such data exists, it may drastically bias the search by limiting it to the subspace of initially provided solutions. In our previous work, we introduced an alternative approach called *latent space policy search* that tightly integrates RL and dimensionality reduction (Luck et al. 2014). Using an expectation-maximization (EM) framework (Dempster, Laird, and Rubin 1977) we presented a latent space policy search algorithm that iteratively refines both the estimates of the low-dimensional latent space, as well as the policy parameters. Only samples produced during the search process were used.

In this paper, we propose a different kind of latent space policy search approach, which similarly to our previous work combines RL and dimensionality reduction, but which also allows for prior structural knowledge to be included. Our method is based on the Variational Bayes (VB) (Neumann 2011; van de Meent et al. 2015) framework. Variational Bayes is a Bayesian generalization of the expectation-maximization algorithm, which returns a distribution over optimal parameters instead of a single point estimate. It is a powerful framework for approximating integrals that would otherwise be intractable. Our RL algorithm exploits these properties in order to (1) perform efficient policy search, (2) infer the low-dimensional latent space of the task, and

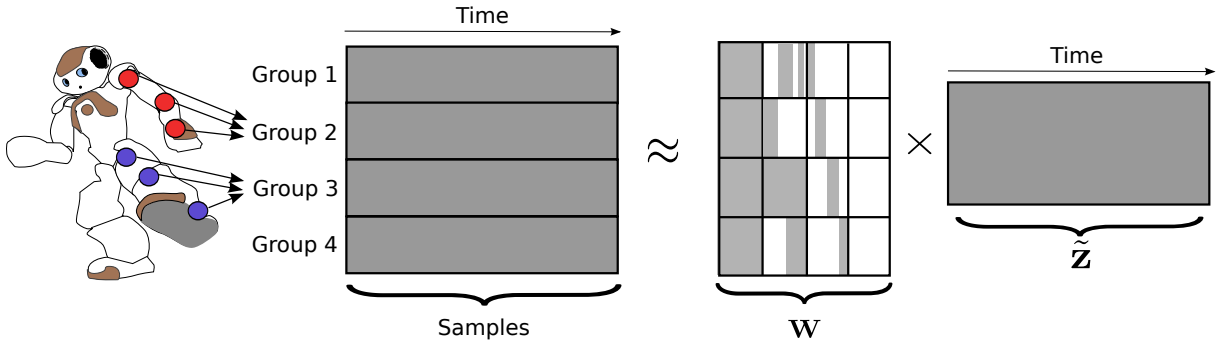

Figure 1: The main idea of Group Factor Policy Search: A number of variables, for example the joints of an arm or leg of a NAO robot, form one group. Given several of such groups for the action vector (left matrix) the transformation matrix $\mathbf{W}$ can be divided in several submatrices corresponding to those groups. Subsequently each factor, given by a column in $\mathbf{W}$, encodes information for all groups, e.g. four in the example given above. Factors may be non-zero for all groups, for a subset of groups, for exactly one group or zero for all groups. In the figure, grey areas correspond to non-zero values and white areas to zero values in the sparse transformation matrix. The transformation matrix is multiplied by the latent variables given by $\tilde{\mathbf{Z}} = (\tilde{\mathbf{z}}_1, \cdots, \tilde{\mathbf{z}}_t, \cdots, \tilde{\mathbf{z}}_T)$ distributed by $\tilde{\mathbf{z}}_t \sim \mathcal{N}(\mathbf{0}, \text{trace}(\boldsymbol{\phi}(\mathbf{s}_t, t)\boldsymbol{\phi}(\mathbf{s}_t, t)^{\mathrm{T}})\mathbf{I})$.

(3) incorporate prior structural information. Prior knowledge about locality of synergies can be included by specifying distinct groups of correlated sub-components. Often such prior knowledge about *groups* of variables, e.g. co-activated joints and limbs, is readily available from the mechanical structure of a system. Structural prior knowledge is also common in other application domains. For example, in a wireless network the network topology defines receiver groups (Sagduyu and Ephremides 2004).

Our approach draws inspiration and incorporates ideas from Factor Analysis, in particular Group Factor Analysis (Klami et al. 2015), as can be seen in Fig. 1. Groups of variables, e.g., robot joints grouped into arms and legs, are provided as prior structural knowledge by a user. A factorized control policy is then learned through RL, which includes a transformation matrix $\mathbf{W}$. The transformation matrix holds factors that describe dependencies between either all of the groups or a subset of them. The individual factors can be regarded as synergies among the joints of the robot.

We will show that the resulting algorithm effectively ties together prior structural knowledge, latent space identification, and policy search in a coherent way.

Policy Search

Policy search methods try to find an optimal policy for an agent which acts in an uncertain world with an unknown world model. At each time step t the agent executes an action $\mathbf{a}_t$ in state $\mathbf{s}_t$ and moves to the next state $\mathbf{s}_{t+1}$ with probability $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$. After executing a certain number of actions, the agent receives a reward feedback given by an unknown reward function based on the performed execution trace (or trajectory/history) $\boldsymbol{\tau} = (\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T, \mathbf{s}_{T+1})$. The overall objective in policy search is to maximize the *expected reward* over trajectories and policy parameters $\boldsymbol{\theta}$. For bounded rewards, maximizing expected reward is equivalent to maximizing the probability of a binary reward r (Toussaint and Storkey 2006):

$$\mathbb{E}_{\boldsymbol{\tau}}[r = 1] = \iint p(\boldsymbol{\tau}, \boldsymbol{\theta}) p(r = 1|\boldsymbol{\tau}) \mathrm{d}\boldsymbol{\theta} \mathrm{d}\boldsymbol{\tau}, \qquad (1)$$

where the probability of the trajectory $p(\boldsymbol{\tau}, \boldsymbol{\theta})$ contains the (stochastic) policy, r is a binary variable indicating maximum reward, and $p(r = 1|\boldsymbol{\tau}) \propto \exp\{-c(\boldsymbol{\tau})\}$ (Toussaint 2009) is the conditional probability of receiving maximum expected reward given a cost function.

Assuming the Markov property and the independence of actions, the probability of a trajectory can be written as

$$p(\boldsymbol{\tau}, \boldsymbol{\theta}) = p(\boldsymbol{\theta}) p(\mathbf{s}_1) \prod_{t=1}^{T} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \pi(\mathbf{a}_t|\mathbf{s}_t, \boldsymbol{\theta}). \qquad (2)$$

The stochastic policy $\pi(\mathbf{a}_t|\mathbf{s}_t, \boldsymbol{\theta})$ depends on the parameters $\boldsymbol{\theta}$ for which we additionally introduce prior distributions $p(\boldsymbol{\theta})$. This formulation will subsequently be used for structuring the policy model. The prior distributions may also depend on hyperparameters – for reasons of clarity, however, we will omit any such parameters below. Furthermore, we assume that the initial state distribution $p(\mathbf{s}_1)$ and transition dynamics $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ are unknown but fixed. Thus, they will cancel out as constant values.

Group Factor Policy Search

We will now introduce a new policy search method, called Group Factor Policy Search (GrouPS), that uncovers the latent space on-the-fly based on prior structural information. In this section, we discuss how to incrementally improve the policy and the actual form of the new policy model. We parameterize the policy using Group Factor Analysis (Klami et al. 2015) in order to utilize prior information about the parameters and their correlations. Since our policy is a linear stochastic model with prior distributions, we first present a novel general Variational Inference framework for policy search that takes priors into account. Subsequently, we discuss how the policy is parameterized, and finally show

the policy model update equations for Group Factor Policy Search which we derive using the introduced Variational Inference method.

Variational Inference for Policy Search

In each iteration our new policy search method samples a distribution over trajectories $p_{\text{old}}(\boldsymbol{\tau})$ using the current policy, and based on $p_{\text{old}}(\boldsymbol{\tau})$ finds a new policy which maximizes a lower bound on the expected reward. This is repeated until convergence.

In order to find a new policy based on samples from the old one, we introduce the sampling distribution $p_{\text{old}}(\boldsymbol{\tau})$ and the approximated parameter distribution $q(\boldsymbol{\theta})$ (defined later) into Equation 1. By applying the log-function and using Jensen's inequality (Kober and Peters 2009; Bishop 2006, Eq. (1.115)) we derive the lower bound

$$\begin{aligned}
&\log \iint p_{\text{old}}(\boldsymbol{\tau}) q(\boldsymbol{\theta}) \frac{p(\boldsymbol{\tau}, \boldsymbol{\theta})}{p_{\text{old}}(\boldsymbol{\tau}) q(\boldsymbol{\theta})} p(r=1|\boldsymbol{\tau}) \mathrm{d}\boldsymbol{\theta} \mathrm{d}\boldsymbol{\tau} \\
&\geq \iint p_{\text{old}}(\boldsymbol{\tau}) q(\boldsymbol{\theta}) \log \left(\frac{p(\boldsymbol{\tau}, \boldsymbol{\theta})}{p_{\text{old}}(\boldsymbol{\tau}) q(\boldsymbol{\theta})} \right) p(r=1|\boldsymbol{\tau}) \mathrm{d}\boldsymbol{\theta} \mathrm{d}\boldsymbol{\tau}.
\end{aligned} \tag{3}$$

Since $p_{\text{old}}(\boldsymbol{\tau})$ was generated using the old policy it does not depend on $\boldsymbol{\theta}$ and we can simplify the lower bound to

$$\begin{aligned}
&\iint p_{\text{old}}(\boldsymbol{\tau}) q(\boldsymbol{\theta}) \log \left(\frac{p(\boldsymbol{\tau}, \boldsymbol{\theta})}{p_{\text{old}}(\boldsymbol{\tau}) q(\boldsymbol{\theta})} \right) p(r=1|\boldsymbol{\tau}) \mathrm{d}\boldsymbol{\theta} \mathrm{d}\boldsymbol{\tau} \\
&= const + \iint p_{\text{old}}(\boldsymbol{\tau}) q(\boldsymbol{\theta}) \\
&\quad \cdot \log \left(\frac{p\left(\boldsymbol{\theta}\right) \prod_{t=1}^{T} \pi(\mathbf{a}_t|\boldsymbol{\theta}, \mathbf{s}_t)}{q(\boldsymbol{\theta})} \right) p(r=1|\boldsymbol{\tau}) \mathrm{d}\boldsymbol{\theta} \mathrm{d}\boldsymbol{\tau},
\end{aligned} \tag{4}$$

where the constant term can be ignored for the maximization of this term. By assuming the factorization $q(\boldsymbol{\theta}) = \prod q_i(\boldsymbol{\theta}_i)$ for the parameters and applying the Variational Bayes approach, we get the approximated distributions of the parameters:

$$\begin{aligned}
&\log q_j(\boldsymbol{\theta}_j) = const + \int_{\boldsymbol{\theta}_{-j}} \prod_{i \neq j} q_i(\boldsymbol{\theta}_i) \\
&\int p_{\text{old}}(\boldsymbol{\tau}) \log \prod_{t=1}^{T} \pi(\mathbf{a}_t, \boldsymbol{\theta}|\mathbf{s}_t) \frac{p(r=1|\boldsymbol{\tau})}{\widehat{R}} \mathrm{d}\boldsymbol{\tau} \mathrm{d}\boldsymbol{\theta}_{-j},
\end{aligned} \tag{5}$$

where the parameter vector $\boldsymbol{\theta}_{-j}$ contains all parameters except $\boldsymbol{\theta}_j$. The normalization constant $\widehat{R}$ is given by the integral

$$\widehat{R} = \int p_{\text{old}}(\boldsymbol{\tau}) p(r=1|\boldsymbol{\tau}) \mathrm{d}\boldsymbol{\tau}. \tag{6}$$

Formulation of Group Factor Policy Search

In order to identify sets of correlated variables during policy search, we use a linear stochastic policy of a form similar to the model used in Group Factor Analysis (GFA) (Klami et al. 2015). The main idea of GFA is to introduce prior distributions for the parameters, in particular a prior for a structured transformation matrix $\mathbf{W}$. The transformation matrix, responsible for mapping between a low-dimensional subspace and the original high-dimensional space, is forced to be sparse and constructed using prior knowledge about grouping of the dimensions, that is, $\mathbf{W}$ is a concatenation of transform matrices $\mathbf{W}^{(m)}$ for each group m. For example, if the dimensions of a vector represent the joints of a legged robot, we can group joints belonging to the same leg into the same group (see e.g. Fig. 1).

Applying the idea of Group Factor Analysis for directed sampling leads to a linear model, i.e. a stochastic policy

$$\mathbf{a}_t^{(m)} = \left(\mathbf{W}^{(m)} \mathbf{Z}_t + \mathbf{M}^{(m)} + \mathbf{E}_t^{(m)} \right) \boldsymbol{\phi}\left(\mathbf{s}_t, t\right) , \tag{7}$$

where, for group m, the action $\mathbf{a}_t^{(m)} \in \mathbb{R}^{D_m \times 1}$ is a linear projection of a feature vector $\boldsymbol{\phi}\left(\mathbf{s}_t, t\right) \in \mathbb{R}^{p \times 1}$. Each dimension of the feature vector is given by a basis function, which may depend on the current state and/or time. In the remainder of the paper, we will write $\boldsymbol{\phi}$ instead of $\boldsymbol{\phi}\left(\mathbf{s}, t\right)$ for simplicity, even though there is an implicit dependency

Input: Reward function $R\left(\cdot\right)$ and initializations of parameters. Choose number of latent dimension n. Set fixed hyper-parameters $a^{\tilde{\tau}}, b^{\tilde{\tau}}, a^{\alpha}, b^{\alpha}, \sigma^2$ and define groupings.

while *reward not converged* **do**
  for *h=1:H* **do** # Sample H rollouts
    for *t=1:T* **do**
      $\mathbf{a}_t = \mathbf{W}_i \mathbf{Z} \boldsymbol{\phi} + \mathbf{M} \boldsymbol{\phi} + \mathbf{E} \boldsymbol{\phi}$
      with $\mathbf{Z} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$ and $\mathbf{E} \sim \mathcal{N}\left(\mathbf{0}, \tilde{\boldsymbol{\tau}}\right)$, where $\tilde{\boldsymbol{\tau}}^{(m)} = \tilde{\tau}_m \mathbf{I}$
      Execute action $\mathbf{a}_t$
    Observe and store reward $R\left(\boldsymbol{\tau}\right)$
  Initialization of q-distribution
  while *not converged* **do**
    Update $q\left(\mathbf{M}\right)$ with Eq. (16)
    Update $q\left(\mathbf{W}\right)$ with Eq. (19)
    Update $q\left(\tilde{\mathbf{Z}}\right)$ with Eq. (22)
    Update $q\left(\boldsymbol{\alpha}\right)$ with Eq. (12)
    Update $q\left(\tilde{\boldsymbol{\tau}}\right)$ with Eq. (25)
  $\mathbf{M} = \mathbb{E}_{q(\mathbf{M})}\left[\mathbf{M}\right]$
  $\mathbf{W} = \mathbb{E}_{q(\mathbf{W})}\left[\mathbf{W}\right]$
  $\boldsymbol{\alpha} = \mathbb{E}_{q(\boldsymbol{\alpha})}\left[\boldsymbol{\alpha}\right]$
  $\tilde{\boldsymbol{\tau}} = \mathbb{E}_{q(\tilde{\boldsymbol{\tau}})}\left[\tilde{\boldsymbol{\tau}}\right]$

Result: Linear weights $\mathbf{M}$ for the feature vector $\boldsymbol{\phi}$, representing the final policy. The columns of $\mathbf{W}$ represents the factors of the latent space.

Algorithm 1: Outline of the Group Factor Policy Search (GrouPS) algorithm.

of $\boldsymbol{\phi}$ on the current state of a trajectory. $\mathbf{W}^{(m)} \in \mathbb{R}^{D_m \times l}$ is a transformation matrix mapping from the l-dimensional subspace to the original space. Each entry of the latent matrix $\mathbf{Z}_t \in \mathbb{R}^{l \times p}$ is distributed according to a standard normal distribution where $\mathcal{N}(0,1)$, $\mathbf{M}^{(m)} \in \mathbb{R}^{D_m \times p}$ is the mean matrix, and the entries of the noise matrix $\mathbf{E}_t^{(m)} \in \mathbb{R}^{D_m \times p}$ are distributed by $\mathcal{N}(0, \tilde{\tau}_m^{-1})$.

We can derive a stochastic policy from the model defined in Equation 7. Since

$$\mathbf{Z}_t \boldsymbol{\phi} \sim \mathcal{N}(\mathbf{0}, \mathrm{trace}(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}})\mathbf{I}) \tag{8}$$

holds (see e.g. (Luck et al. 2014)), we can substitute $\mathbf{Z}_t \boldsymbol{\phi}$ by the random variable $\tilde{\mathbf{z}}_t \in \mathbb{R}^{l \times 1}$ resulting in the policy

$$\pi(\mathbf{a}_t | \boldsymbol{\theta}, \mathbf{s}_t) = \prod_{m=1}^{M} \mathcal{N}\left(\mathbf{a}_t^{(m)} \middle| \mathbf{W}^{(m)} \tilde{\mathbf{z}}_t + \mathbf{M}^{(m)} \boldsymbol{\phi}, \frac{\mathrm{Tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}{\tilde{\tau}_m} \mathbf{I} \right). \tag{9}$$

If we take a closer look at the latent space given by $\mathbf{W}\tilde{\mathbf{z}}_t$ we first find that the length of each factor is determined by $\|\boldsymbol{\phi}(\mathbf{s}_t, t)\|_2^2$. Secondly, a factor may be non-zero only for one or a subset of groups as can be seen in Fig. 1. This leads to a sparse transformation matrix and specialized factors and dimensions.

As mentioned before, the form of our linear model in Equation 7 above is based on Group Factor Analysis. While GFA typically maps a vector from the latent space to the high-dimensional space, we map here a matrix from the latent space to the original space and then use this matrix as a linear policy on the feature vectors. GFA does not apply factor analysis (see e.g. (Harman 1976)) on each group of variables separately, but instead introduces a sparsity prior on the transformation matrix $\mathbf{W}$ thereby forcing correlations between groups:

$$p(\mathbf{W}|\boldsymbol{\alpha}) = \prod_{m=1}^{M} \prod_{k=1}^{K} \prod_{d=1}^{D_m} \mathcal{N}\left(\mathbf{w}_{d,k}^{(m)} \middle| \mathbf{0}, \alpha_{m,k}^{-1} \right), \tag{10}$$

with M being number of groups, D_m the number of dimensions of the m-th group and K the number of factors, i.e. number of columns of $\mathbf{W}$. The precision $\boldsymbol{\alpha}$ is given by a log-linear model with

$$\log \boldsymbol{\alpha} = \mathbf{U}\mathbf{V}^{\mathrm{T}} + \boldsymbol{\mu}_u \mathbf{1}^{\mathrm{T}} + \mathbf{1}\boldsymbol{\mu}_v^{\mathrm{T}}, \tag{11}$$

where $\mathbf{U} \in \mathbb{R}^{M \times \mathcal{R}}$, $\mathbf{V} \in \mathbb{R}^{K \times \mathcal{R}}$ and $\boldsymbol{\mu}_u \in \mathbb{R}^M$ as well as $\boldsymbol{\mu}_v \in \mathbb{R}^K$ model the mean profile. $\mathcal{R}$ defines the rank of the linear model and is chosen $\mathcal{R} \ll \min(M, K)$. However, for the special case of $\mathcal{R} = \min(M, K)$ the precision is given by a simple gamma distribution (Klami et al. 2015)

$$q(\boldsymbol{\alpha}_{m,k}) = \mathcal{G}\left(a_m^{\alpha}, b_{m,k}^{\alpha}\right) \tag{12}$$

with parameters

$$a_m^{\alpha} = a^{\alpha} + \frac{D_m}{2}, \tag{13}$$

$$b_{m,k}^{\alpha} = b^{\alpha} + \frac{1}{2} \mathbb{E}_{q(\mathbf{W})} \left[\mathbf{w}_k^{(m)\mathrm{T}} \mathbf{w}_k^{(m)} \right]. \tag{14}$$

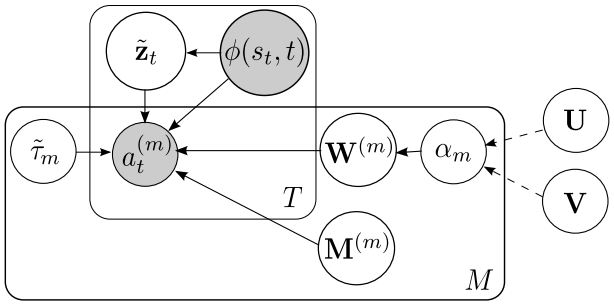

Figure 2: Graphical model in Plate notation for Group Factor Policy Search. The basis functions $\boldsymbol{\phi}(\mathbf{s}_t, t)$ as well as the action vector $\mathbf{a}_t^{(m)}$ are observed. Equation 9 shows the dependencies for $\mathbf{a}_t^{(m)}$. The latent variables $\tilde{\mathbf{z}}_t$ depend on the feature vector as stated in Equation (8) . The parameter α_m might either be given by a Gamma distribution as stated in Equation (12) or by a log-linear model with dependencies on parameters $\mathbf{U}$ and $\mathbf{V}$.

The hyper-parameters a^{α} and b^{α} are fixed and set to a small positive value. The prior distributions given above will lead to three kind of factors: (1) factors which are nonzero for only one group, (2) factors which are nonzero for several groups or (3) factors which are zero. In addition to the standard GFA prior distributions above, we introduce further prior distributions for $\mathbf{M}$ and $\tilde{\mathbf{z}}$ such that all prior distributions are given with

$$\mathbf{M} \sim \mathcal{N}\left(\mathbf{M}_{\text{old}}, \sigma^2 \mathbf{I}\right), \qquad \tilde{\mathbf{z}} \sim \mathcal{N}\left(\mathbf{0}, \mathrm{Tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\mathbf{I}\right),$$
$$\alpha_{m,k} \sim \mathcal{G}\left(a^{\alpha}, b^{\alpha}\right), \qquad \tilde{\tau}_m \sim \mathcal{G}\left(a^{\tilde{\tau}}, b^{\tilde{\tau}}\right).$$

Fig. 2 shows a graphical model of Group Factor Policy Search, given by the distributions stated above. Instead of $\mathbf{Z}$ the latent variable $\tilde{\mathbf{z}}_t$ is used, which depends on $\boldsymbol{\phi}(\mathbf{s}_t, t)$ given a state and a point in time.

Derivation of Update Equations

We assume fixed hyper-parameters a^{α}, b^{α}, $a^{\tilde{\tau}}$ and $b^{\tilde{\tau}}$ for the distributions which we determine using the Variational Inference method presented earlier, assuming a factorization of the q-distributions

$$q(\boldsymbol{\theta}) = q(\tilde{\mathbf{Z}}) q(\mathbf{W})\, q(\tilde{\boldsymbol{\tau}})\, q(\mathbf{M})\, q(\boldsymbol{\alpha}) \tag{15}$$

and additionally the assumption $q(\tilde{\mathbf{Z}}) = \prod^{T} q(\tilde{\mathbf{z}}_t)$ with $\tilde{\mathbf{Z}}_{:,t} = \tilde{\mathbf{z}}_t$.

By using the factorization given above and the Variational Inference rule for deriving the parameter distribution in Equation (5), we can derive the approximated parameter distributions, which maximize the expected reward.

The approximated distribution for the mean matrix is given by a multiplicative normal distribution

$$q_{\mathbf{M}}(\mathbf{M}) = \prod_{m=1}^{M} \prod_{j=1}^{D_m} \mathcal{N}\left(\mathbf{m}_{j,:}^{(m)\mathrm{T}} \middle| \boldsymbol{\mu}_{mj}^{\mathbf{M}}, \boldsymbol{\Sigma}_j^{\mathbf{M}} \right) \tag{16}$$

where the mean and covariance parameters in dependency of the group and dimension are given by

$$\boldsymbol{\Sigma}_j^{\mathbf{M}} = \left(\sigma^{-2}\mathbf{I} + \mathbb{E}_{p(\boldsymbol{\tau})}\left[\frac{p(r=1|\boldsymbol{\tau})}{\widehat{R}}\left(\sum_{t=1}^{T}\frac{\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}}{\mathrm{Tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\mathbb{E}_{\tilde{\tau}}\left[\tilde{\tau}_m\right]\right)\right]\right)^{-1} \tag{17}$$

and

$$\boldsymbol{\mu}_{mj}^{\mathbf{M}} = \boldsymbol{\Sigma}_j^{\mathbf{M}} \cdot \frac{\mathbf{m}_{\mathrm{old}_{j,:}}{}^{\mathrm{T}}}{\sigma^2} + \boldsymbol{\Sigma}_j^{\mathbf{M}} \cdot \mathbb{E}_{p(\boldsymbol{\tau})}\left[\frac{p(r=1|\boldsymbol{\tau})}{\widehat{R}}\sum_{t=1}^{T}\frac{\boldsymbol{\phi}\left(a_{t,j}^{(m)} - \mathbb{E}_{\mathbf{w}}\left[\mathbf{w}_{j,:}^{(m)}\right]\mathbb{E}_{\tilde{\mathbf{z}}}\left[\tilde{\mathbf{z}}_t\right]\right)}{\mathrm{Tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\mathbb{E}_{\tilde{\tau}}\left[\tilde{\tau}_m\right]^{-1}}\right] \tag{18}$$

with $\mathbf{m}_{j,:}$ given by the j-th row of $\mathbf{M}$.

The q-distribution for the transformation matrix is similarly given by

$$q_{\mathbf{W}}\left(\mathbf{W}\right) = \prod_{m=1}^{M}\prod_{j=1}^{D_m}\mathcal{N}\left(\mathbf{w}_{j,:}^{(m)\mathrm{T}}|\boldsymbol{\mu}_{mj}^{\mathbf{W}}, \boldsymbol{\Sigma}_m^{\mathbf{W}}\right) \tag{19}$$

with the mean and covariance parameters

$$\boldsymbol{\Sigma}_m^{\mathbf{W}} = \left(\mathbb{E}_{p(\boldsymbol{\tau})}\left[\frac{p(r=1|\boldsymbol{\tau})}{\widehat{R}}\left(\sum_{t=1}^{T}\frac{\mathbb{E}_{\tilde{\mathbf{z}}}\left[\tilde{\mathbf{z}}_t\tilde{\mathbf{z}}_t^{\mathrm{T}}\right]}{\mathrm{Tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\mathbb{E}_{\tilde{\tau}}\left[\tilde{\tau}_m\right]^{-1}} + \bar{\bar{\boldsymbol{\alpha}}}_{m,K}\right)\right]\right)^{-1}, \tag{20}$$

and

$$\boldsymbol{\mu}_{mj}^{\mathbf{W}} = \boldsymbol{\Sigma}_m^{\mathbf{W}} \cdot \mathbb{E}_{p(\boldsymbol{\tau})}\left[\frac{p(r=1|\boldsymbol{\tau})}{\widehat{R}}\sum_{t=1}^{T}\frac{\left(a_{t,j}^{(m)} - \mathbb{E}_{\mathbf{M}}\left[\mathbf{m}_{j,:}^{(m)}\right]\boldsymbol{\phi}\right)\mathbb{E}_{\tilde{\mathbf{z}}}\left[\tilde{\mathbf{z}}_t\right]^{\mathrm{T}}}{\mathrm{Tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\mathbb{E}_{\tilde{\tau}}\left[\tilde{\tau}_m\right]^{-1}}\right]. \tag{21}$$

The diagonal matrix $\bar{\bar{\boldsymbol{\alpha}}}_{m,K}$ is given by $\mathrm{diag}\left(\bar{\bar{\boldsymbol{\alpha}}}_{m,K}\right) = \left(\alpha_{m,1}, \alpha_{m,2}, \cdots, \alpha_{m,K}\right)$. The distribution for the latent variables $\tilde{\mathbf{Z}}$ depends on the trajectory and time. Hence the reward can be seen as a probabilistic weight $\tilde{R}$ of a multiplicative normal distribution. However, since we assume independent latent variables $\tilde{\mathbf{z}}_t^h$ we can ignore the reward and get

$$q_{\tilde{\mathbf{Z}}}\left(\tilde{\mathbf{Z}}\right) = \prod^{H}\tilde{R}\prod_{t=1}^{T}\mathcal{N}\left(\tilde{\mathbf{z}}_t^h|\boldsymbol{\mu}_t^{\tilde{\mathbf{Z}}}, \boldsymbol{\Sigma}_t^{\tilde{\mathbf{Z}}}\right), \tag{22}$$

with time-dependent parameters

$$\boldsymbol{\Sigma}_t^{\tilde{\mathbf{Z}}} = \left(\mathrm{Tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)^{-1}\mathbf{I} + \sum_{m=1}^{M}\frac{\mathbb{E}_{\mathbf{W}}\left[\mathbf{W}^{(m)\mathrm{T}}\mathbf{W}^{(m)}\right]}{\mathrm{Tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\mathbb{E}_{\tilde{\tau}_m}\left[\tilde{\tau}_m^{-1}\right]}\right)^{-1}, \tag{23}$$

and

$$\boldsymbol{\mu}_t^{\tilde{\mathbf{Z}}} = \boldsymbol{\Sigma}_t^{\tilde{\mathbf{Z}}} \cdot \left(\sum_{m=1}^{M}\frac{\mathbb{E}_{\mathbf{W}}\left[\mathbf{W}^{(m)}\right]^{\mathrm{T}}\left(\mathbf{a}_t^{(m)} - \mathbf{M}^{(m)}\boldsymbol{\phi}\right)}{\mathrm{Tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\mathbb{E}_{\tilde{\tau}}\left[\tilde{\tau}_m\right]^{-1}}\right). \tag{24}$$

Unlike the other distributions, the precision is given by a multiplicative gamma distribution

$$q_{\tilde{\tau}}\left(\tilde{\tau}\right) = \prod_{m=1}^{M}\mathcal{G}\left(\tilde{\tau}_m|a^{\tilde{\tau}} + \frac{1}{2}D_m T, b^{\tilde{\tau}} + \frac{1}{2}b_m^{\tilde{\tau}\,\prime}\right) \tag{25}$$

with one fixed parameter and one variable parameter. Estimation of the parameter $b_m^{\tilde{\tau}\,\prime}$ is the most complex and computationally expensive operation given by

$$\begin{aligned} b_m^{\tilde{\tau}\,\prime} = \mathbb{E}_{p(\boldsymbol{\tau})}\Bigg[\frac{p(r=1|\boldsymbol{\tau})}{\widehat{R}}\sum_{t=1}^{T}\mathrm{Tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)^{-1}\Big(&\mathbf{a}_t^{(m)\mathrm{T}}\mathbf{a}_t^{(m)} \\ &- 2\mathbf{a}_t^{(m)\mathrm{T}}\mathbb{E}_{\mathbf{M}}\left[\mathbf{M}^{(m)}\right]\boldsymbol{\phi} \\ &+ 2\mathbb{E}_{\tilde{\mathbf{z}}}\left[\tilde{\mathbf{z}}_t\right]^{\mathrm{T}}\mathbb{E}_{\mathbf{W}}\left[\mathbf{W}^{(m)}\right]^{\mathrm{T}}\mathbb{E}_{\mathbf{M}}\left[\mathbf{M}^{(m)}\right]\boldsymbol{\phi} \\ &- 2\mathbf{a}_t^{(m)\mathrm{T}}\mathbb{E}_{\mathbf{W}}\left[\mathbf{W}^{(m)}\right]\mathbb{E}_{\tilde{\mathbf{z}}}\left[\tilde{\mathbf{z}}_t\right] \\ &+ \boldsymbol{\phi}^{\mathrm{T}}\mathbb{E}_{\mathbf{M}}\left[\mathbf{M}^{(m)\mathrm{T}}\mathbf{M}^{(m)}\right]\boldsymbol{\phi} \\ &+ \mathrm{Tr}\left(\mathbb{E}_{\mathbf{W}}\left[\mathbf{W}^{(m)\mathrm{T}}\mathbf{W}^{(m)}\right]\mathrm{Cov}_{\tilde{\mathbf{z}}}\left[\tilde{\mathbf{z}}_t\right]\right) \\ &+ \mathbb{E}_{\tilde{\mathbf{z}}}\left[\tilde{\mathbf{z}}_t\right]^{\mathrm{T}}\mathbb{E}_{\mathbf{W}}\left[\mathbf{W}^{(m)\mathrm{T}}\mathbf{W}^{(m)}\right]\mathbb{E}_{\tilde{\mathbf{z}}}\left[\tilde{\mathbf{z}}_t\right]\Big)\Bigg]. \end{aligned} \tag{26}$$

Algorithm

Algorithm 1 summarizes all update steps for performing Group Factor Policy Search. The reward function $R(\cdot)$, number n of latent dimensions, and a set of hyperparameters need to be provided by the user.

Evaluation

For evaluations and experiments the expectation $\mathbb{E}_{p(\boldsymbol{\tau})}[\cdot]$ used above in Eq.(16-20,25) was approximated by a sample mean,

$$\mathbb{E}_{p(\boldsymbol{\tau})}[f(\boldsymbol{\tau})] \approx \frac{1}{H}\sum_{i=1}^{H}f(\boldsymbol{\tau}_i) \tag{27}$$

as proposed in (Kober and Peters 2009), where $\boldsymbol{\tau}_i$ is the i-th of the H realized trajectories and $f(\boldsymbol{\tau})$ a function value, vector or matrix for $\boldsymbol{\tau}_i$ and will be replaced by the parameter approximations given above.

Setup of the Evaluation

For the comparison between the above presented GrouPS algorithm and previous policy search algorithms, a simulated task of a bi-manual robot operating in a planar task space was used. Each of the two arms (see Fig. 3) has six degrees-of-freedom and the same base for the first joint. The initial

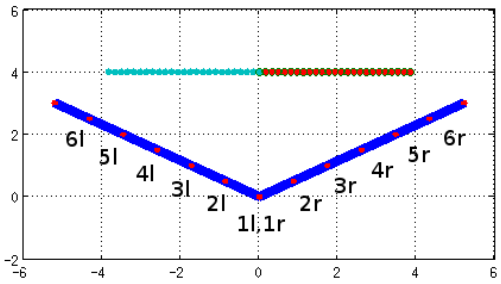

Figure 3: Two simulated arms with six degrees-of-freedom and the same base in their initial position. Each end effector has a desired position for each time step, s shown by the green and red dots. The final position at time step 25 is given by the coordinate $(0, 4)$. The numbers represent the joints with l for left and r for right.

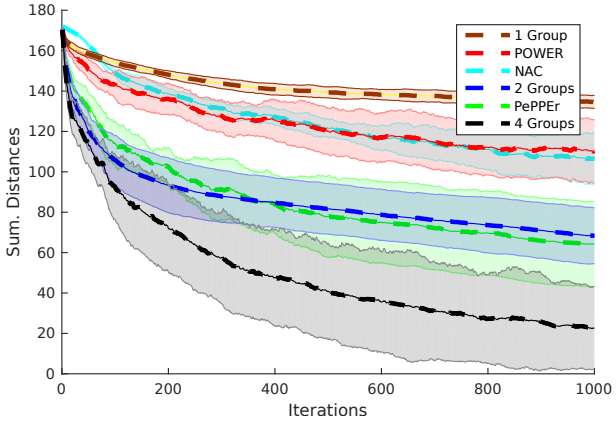

Figure 4: Comparison between PePPEr, PoWER, Natural Actor-Critic and three instances of the GrouPS algorithm on the presented simulated task. Values correspond to the summarized distances between each end effector and its desired position given the current policy for the iteration. The mean value as well as the standard deviations are shown.

configuration of the arms is presented in Fig. 3 as well as the desired positions for each end effector (tip of an arm). At each of the 25 time steps we give a different goal position for each arm's end effector, starting from the left for the left arm and starting from the right for the right arm, with the same final position at $(0, 4)$ for both arms. In this task, the 12 dimensions of the action vector $\mathbf{a}$ represent the joint angles for each arm. For the basis functions eleven isotropic Gaussian distributions were used with $\phi_i(t) = \mathcal{N}(t|\mu_i^{\Phi}, 3)$ for $t \in \{1, 2, \ldots, 24, 25\}$. In total, 132 parameters have to be estimated given $\mathbf{M} \in \mathbb{R}^{12\times 11}$.

As reference algorithms PoWER (Kober and Peters 2009), Natural Actor-Critic (NAC) (Peters and Schaal 2008) and PePPEr (Luck et al. 2014) were chosen: NAC is a policy gradient method while PoWER is an efficient policy search method based on expectation maximization (EM). PoWER has been experimentally validated in both simulated and physical robotic experiments (Kober and Peters 2011). PePPEr is also based on EM and incorporates policy search and dimensionality reduction, but without priors and thus without a structured transformation matrix. For comparison with PePPEr and PoWER the GrouPS algorithm was evaluated in three different configurations: (1) One group which contains all joints of both arms, (2) two groups, where each group contains the joints of one arm and (3) four groups with two groups per arm and joints 1-4 in one and joints 5-6 in the second group. The hyper-parameters of GrouPS were set to $a^{\tilde{\tau}} = b^{\tilde{\tau}} = 1000$, $a^{\alpha} = b^{\alpha} = 1$ and $\sigma^2 = 100$. No optimizations of the hyper-parameters were performed. Furthermore, to prevent early convergence and collapsing of the distributions due to small sample sizes the parameter $\mathbf{W}$ and $\tilde{\tau}$ are resized after each iteration by a factor of 1.5. The same is done after each iteration for PePPEr. However, the factor was set to $\sqrt{1.09}$ since higher numbers lead to divergence in the parameters of the algorithm with unstable and divergent results. PePPEr was implemented as presented in (Luck et al. 2014) and in each iteration 20 inner iterations for the optimizations of the parameters were used. The same setup was used for GrouPS and for both algorithms the number of latent dimensions were set to six. The static variance parameter for PoWER as presented in (Kober and Peters 2009) and the initial variance of the other algorithms were all set to $10^{1.5}$, also for NAC with learning parameter set to 0.5. In each iteration, we sampled 30 trajectories and evaluated the trajectories based on the reward function

$$R(\boldsymbol{\tau}) = \sum_{t=1}^{25} \exp\left(- \parallel \text{eff}_l(\mathbf{a}_t) - \text{pos}_l(t) \parallel_2\right) \cdot \exp\left(- \parallel \text{eff}_r(\mathbf{a}_t) - \text{pos}_r(t) \parallel_2\right), \tag{28}$$

where the function $\text{eff}_l(\mathbf{a}_t)$ returns the position of the left end effector given the action vector and $\text{pos}_l(t)$ the corresponding desired goal position for time point t. $\text{eff}_r(\mathbf{a}_t)$ and $\text{pos}_r(t)$ return the actual and desired positions, respectively, for the right end effector. Then the 15 best trajectories are chosen for the computation of the parameters for each algorithm as described in (Kober and Peters 2009).

Results

Fig. 4 depicts the results of the explained experiment. For each algorithm ten different runs were executed and both mean and standard deviation computed. As can be seen in the figure, PePPEr outperforms both PoWER and NAC, as well as our method in case only one group spanning all variables is used. However, using two groups (one for each arm) already leads to comparable performance. Finally, the GrouPS algorithm with 4 different groups significantly outperforms the comparison methods.

Importance of the Choice of Groups

In order to investigate the effect of choosing joint groups we conducted an additional experiment. Our working hypothesis throughout the paper is that structural information about inherent groups of correlated variables will improve the search. Conversely, if we provide wrong in-

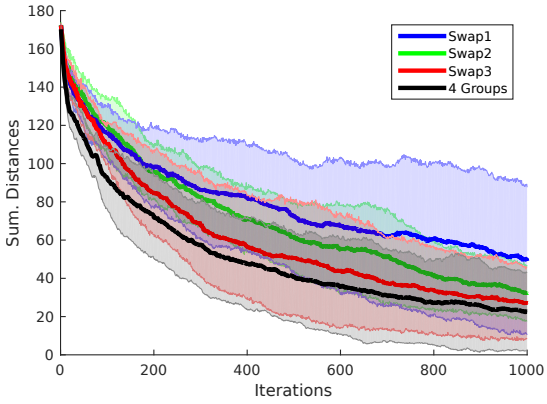

Figure 5: Comparison between the original chosen four groups and three permutations of the Groups. Values correspond to the summarized distance between each end effector and its desired position for each time step given the current policy for the iteration.

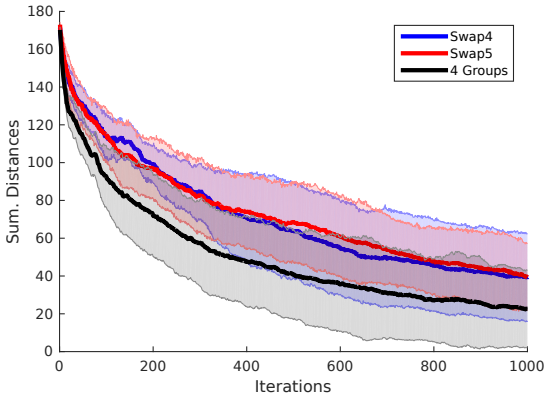

Figure 6: Comparison between the original grouping and two other variants with a different splitting point. Again, the values represent the summarized distances and shaded ares corresponds to the standard deviation given ten executions.

formation about groupings the performance of the algorithm should deteriorate. To evaluate this hypothesis, we took the original partitioning of the joints into four groups and swapped two, later three pairs of joints randomly. As described above, the original group partitioning is $\{(1l, 2l, 3l, 4l), (5l, 6l), (1r, 2r, 3r, 4r), (5r, 6r)\}$. Performing two random swaps between the left and right side results in $\{(1l, 2l, 2r, 4l), (5l, 5r), (1r, 3l, 3r, 4r), (6l, 6r)\}$ (Fig. 6, Swap4). For three swaps the resulting partition is $\{(1l, 6r, 2r, 4l), (3r, 6l), (1r, 3l, 5l, 4r), (5r, 2l)\}$ (Fig. 6, Swap5). Furthermore, three other groupings with different splitting points were evaluated: $\{(1l, 2l), (3l, 4l, 5l, 6l), (1r, 2r), (3r, 4r, 5r, 6r)\}$ (Fig. 5, Swap1), $\{(1l, 2l), (3l, 4l), (5l, 6l), (1r, 2r), (3r, 4r), (5r, 6r)\}$ (Fig. 5, Swap2) and $\{(1l, 2l, 3l), (4l, 5l, 6l), (1r, 2r, 3r), (4r, 5r, 6r)\}$ (Fig. 5, Swap3). The result of executing GrouPS with these groupings can be seen in Fig. 5 and Fig. 6. All new groupings (resulting from above swaps) are clearly outperformed by the original partition. This result corroborates our assumption that a proper selection of groups can ameliorate the performance of the policy search algorithm.

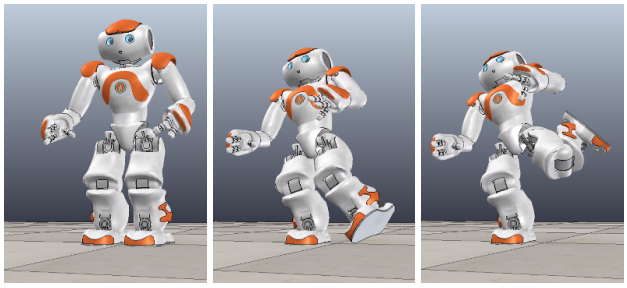

Figure 7: Final policy found by the GrouPS algorithm after 100 iterations. A high reward is given if the head as well as the left foot of the robot are high above the ground.

Experiment: Lifting a Leg

To test the GrouPS algorithm in experiments following the real world closely, we reproduced the experiment stated in (Luck et al. 2014): We simulate a NAO robot (Gouaillier et al. 2008) using the V-REP framework (Rohmer, Singh, and Freese 2013) in the task of lifting its left leg without falling. The same reward function was used as presented in (Luck et al. 2014, Eq. (22)) with parameters $\alpha = 5$, $\beta = 10$, $\gamma = 10$ and $\lambda_{max} = 6$. The V-REP framework (Rohmer, Singh, and Freese 2013) allows for simulations with high physical accuracy by utilizing the bullet physics library. In this experiment, the actions represent the 26 joint velocities for each of the 15 points in time. Again, for feature functions Gaussian distributions were used and the same parameters for GrouPS were chosen like given in the evaluation above.

We ran GrouPS for 100 iterations. In each iteration, we used a set of 20 samples, of which ten were randomly selected from the set of 20 in the previous iteration and ten generated by the current policy. We used ten best samples out of this set of 20 for computing the new policy parameters. The groups were created in such a manner that the joints of each arm or leg form a single group as well as the joints of the head. The results are given in Fig. 7, where we find that the GrouPS algorithm is able to find a satisfactory solution even with a relatively small number of samples: the head and left leg of the NAO robot are at high positions corresponding to a high reward.

Conclusion and Future Work

In this paper, we introduced a novel algorithm for reinforcement learning in low-dimensional latent spaces. To this end, we derived a Variational Inference framework for policy search that takes prior structural information into account. The resulting policy search algorithm can efficiently learn new policy parameters, while also uncovering the underlying latent space of solutions, and incorporating prior knowledge about groups of correlated parameters. In experiments using motor skill learning tasks, we showed that the introduced GrouPS algorithm efficiently learns new motor skills. It significantly outperformed state-of-the-art policy

search methods, whenever prior information about structural groups was provided.

So far, the dimensionality of the latent space needs to be provided as a parameter to the reinforcement learning algorithm. We plan to investigate automatic adjustments of the dimensionality using current rewards. In this paper, we focused on intra-group correlations. In future work, we plan to investigate correlations among extracted group factors, e.g., correlations between arms and legs.

Acknowledgments

J.Pajarinen and V.Kyrki were supported by the Academy of Finland, decision 271394.

References

Bernstein, N. A. 1967. *The co-ordination and regulation of movements*. Pergamon Press.

Bishop, C. 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 *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3219–3225. IEEE.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* 39(1):1–38.

Gouaillier, D.; Hugel, V.; Blazevic, P.; Kilner, C.; Monceaux, J.; Lafourcade, P.; Marnier, B.; Serre, J.; and Maisonnier, B. 2008. The nao humanoid: a combination of performance and affordability. *arXiv preprint arXiv:0807.3223*.

Harman, H. H. 1976. *Modern factor analysis*. University of Chicago Press.

Klami, A.; Virtanen, S.; Leppaaho, E.; and Kaski, S. 2015. Group factor analysis. *IEEE Transactions on Neural Networks and Learning Systems* 26(9):2136–2147.

Kober, J., and Peters, J. 2009. Policy search for motor primitives in robotics. In *Advances in Neural Information Processing Systems (NIPS)*, 849–856.

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

Kolter, J. Z., and Ng, A. Y. 2007. Learning omnidirectional path following using dimensionality reduction. In *Proceedings of the Robotics: Science and Systems (R:SS) conference*. The MIT Press.

Luck, K. S.; Neumann, G.; Berger, E.; Peters, J.; and Ben Amor, H. 2014. Latent space policy search for robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1434–1440. IEEE.

Neumann, G. 2011. Variational inference for policy search in changing situations. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 817–824.

Peters, J., and Schaal, S. 2008. Natural actor-critic. *Neurocomputing* 71(7):1180–1190.

Peters, J.; Mülling, K.; Kober, J.; Nguyen-Tuong, D.; and Krömer, O. 2011. Towards motor skill learning for robotics. In *Robotics Research*. Springer. 469–482.

Rohmer, E.; Singh, S. P.; and Freese, M. 2013. V-REP: A versatile and scalable robot simulation framework. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1321–1326. IEEE.

Sagduyu, Y. E., and Ephremides, A. 2004. The problem of medium access control in wireless sensor networks. *IEEE Wireless Communications* 11(6):44–53.

Santello, M.; Flanders, M.; and Soechting, J. 1998. Postural hand synergies for tool use. *The Journal of Neuroscience* 18(23).

Torres-Oviedo, G., and Ting, L. H. 2010. Subject-specific muscle synergies in human balance control are consistent across different biomechanical contexts. *Journal of Neurophysiology* 103(6):3084–3098.

Toussaint, M., and Storkey, A. 2006. Probabilistic inference for solving discrete and continuous state Markov Decision Processes. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 945–952.

Toussaint, M. 2009. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th annual International Conference on Machine Learning (ICML)*, 1049–1056. ACM.

van de Meent, J.-W.; Tolpin, D.; Paige, B.; and Wood, F. 2015. Black-box policy search with probabilistic programs. *arXiv preprint arXiv:1507.04635*.

Wang, X.; O'Dwyer, N.; and Halaki, M. 2013. A review on the coordinative structure of human walking and the application of principal component analysis. *Neural Regeneration Research* 8(7):662–670.