Variational Inference for Policy Search in changing Situations

Gerhard Neumann GERHARD@IGI.TU-GRAZ.AC.AT
Institute for Theoretical Computer Science, Graz University of Technology, A-8010 Graz, Austria

Abstract

Many policy search algorithms minimize the Kullback-Leibler (KL) divergence to a certain target distribution in order to fit their policy. The commonly used KL-divergence forces the resulting policy to be 'reward-attracted'. The policy tries to reproduce all positively rewarded experience while negative experience is neglected. However, the KL-divergence is not symmetric and we can also minimize the the reversed KL-divergence, which is typically used in variational inference. The policy now becomes 'cost-averse'. It tries to avoid reproducing any negatively-rewarded experience while maximizing exploration.

Due to this 'cost-averseness' of the policy, Variational Inference for Policy Search (VIP) has several interesting properties. It requires no kernel-bandwith nor exploration rate, such settings are determined automatically by the inference. The algorithm meets the performance of state-of-the-art methods while being applicable to simultaneously learning in multiple situations.

We concentrate on using VIP for policy search in robotics. We apply our algorithm to learn dynamic counterbalancing of different kinds of pushes with human-like 2-link and 4-link robots.

1. Introduction

Variational inference is a widely used approximate inference method. While there exists first applications of variational inference for discrete reinforcement learning (Furmston & Barber, 2010), it has never been used for policy search in high dimensional parameter spaces. Variational inference introduces an approximate distribution q and iteratively minimizes the Kullback-Leibler divergence $\mathrm{KL}(q||p)$ between q and the target distribution p. This minimization is also known as I(nformation)-projection of distribution p.

In policy search, many algorithms also apply approximate inference. However, all these algorithm use the M(oment)-projection, which is given by the reversed KL-divergence $\mathrm{KL}(p||q)$ to estimate their policy. While at the first glance this might only be a minor difference, it turns out that the resulting policies may differ considerably. Policies calculated by the M-projection try to reproduce all experience with high reward, but neglect information coming from negative experience. We will therefore call these policies 'reward-attracted'. The I-projection forces the resulting policy to be 'cost-averse'. Here, the focus of the policy is to avoid reproducing negative experience, while exploration is maximized.

Which projection is better suited for policy search? We argue for the I-projection. When using a common Gaussian policy, the M-projection averages over all positively rewarded experience seen so far. However, in the case of a multi-modal or non-concave target distribution taking the average might be a bad choice. The I-projection always tries to exclude negative experience from the resulting distribution, and thus, concentrates at one mode of the target distribution. Non-concave target distributions typically occur if we want to apply policy search for multiple situations. The I-projection can be applied with ease in this context. The 'cost-averseness' also comes with additional advantages. The algorithm automatically determines the optimal kernel bandwidth for a new situation and adapts its exploration rate and used search directions.

In difference to the M-projection, the I-projection can't be minimized in closed form. We have to rely on non-linear optimization methods like gradient descent. Here, we present a new method where gradient descent is performed on meta-parameters of the approximate distribution q.

We will apply our new Variational Inference for Policy Search (VIP) algorithm to learn complex motor skills with robots. In robotics we often need to search for parametrized movement plans in related, but different scenarios. These movement plans, also called Dynamic Movement Primi-

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011.

tives (Ijspeert et al., 2002), Motion Templates (Neumann & Peters, 2009) or Muscle Synergies (E. Bizzi, 2008) are often only valid locally, and hence, need to be adjusted for a new situation.

For example, a tennis playing robot has to adapt its movement to the trajectory of the ball or a humanoid robot has to react differently to counter-balance different kinds of pushes. Hence, we need to find a policy $\pi(\mathbf{w}|\mathbf{s}^0)$ which is able to choose good parametric descriptions $\mathbf{w} \in \mathcal{W}$ of the movement plan given the initial conditions $\mathbf{s}^0$. Learning such a policy π is very challenging due to the high-dimensionality of parameter-space $\mathcal{W}$. Our algorithm is well suited for such tasks.

Many policy search algorithm like the CMA-ES (Heidrich-Meisner & Igel, 2009), Cross-Entropy search (Mannor et al., 2003) or the PoWER (Kober & Peters, 2009) algorithm are limited to the single-situation setting. Only few algorithms exist for learning in multiple initial conditions. Here, we can use Reward-Weighted Regression (RWR)(Kober et al., 2010) or Cost-Regularized-Kernel Regression (CRKR) (Kober et al., 2010), which is the kernelized version of RWR. Both algorithms use locally weighted linear regression methods to interpolate between different initial states $\mathbf{s}_i^0$. In addition to the local weighting, the data points are weighted by their corresponding rewards. The reward-weighted linear regression represents an M-projection of the reward distribution (see Section 3.1), therefore these algorithms suffer from the previously mentioned limitations of the M-projection.

Both algorithms require that the user specifies the shape or bandwidth of the receptive fields or kernels. This shape is not only kept constant during the learning phase, it is also constant in the whole state space. Therefore the user always has to make a tradeoff between fast learning speed and good quality of the final performance. Because the I-projection always wants to exclude samples with low reward, the 'kernel shape' automatically adapts to the data density as well as to the shape of the target distribution.

Note that CRKR and RWR have only been used to learn meta-parameters of the motion (Kober et al., 2010) (like the duration or the end-point of the motion). The remaining (typically higher-dimensional) parametrization for the shape of the trajectory was kept fixed. Therefore, the application is limited to similar shapes of the movement. The VIP approach allows learning with the full-parametric representation of a movement for multiple scenarios, and therefore, can find completely different movements for different subregions of the state space.

We will apply our method to a 2-link and a 4-link dynamic robot balancing task where the robot has to counterbalance different kinds of pushes.

2. Kullback Leibler (KL) Divergences

We quickly review concept of KL-divergences because it is of great importance for this paper. The KL divergence between two probability distributions q and p is defined as

$$\mathrm{KL}(q||p) = -\int_{\mathbf{X}} q(\mathbf{X}) \log \frac{p(\mathbf{X})}{q(\mathbf{X})} d\mathbf{X}$$

It is zero if and only if the two distributions are equal. Since the KL-divergence is *not* symmetric, there are 2 kinds of KL-divergences which we can minimize in order to approximate a target distribution p with an approximate distribution q.

- The **M-projection** $q = \mathrm{argmin}_q \mathrm{KL}(p||q)$: The M-projection forces the approximate distribution q to have high probability everywhere where p has high probability. Therefore, if distribution q is a Gaussian, the M-projection tries to average over all modes of p.
- The **I-projection** $q = \mathrm{argmin}_q \mathrm{KL}(q||p)$: It forces the approximate distribution q to be zero everywhere where p is zero. Can not be calculated in closed form for the most distributions. When using a Gaussian distribution q, the I-projection typically concentrates on a single mode of the target distribution.

These differences between the projections are well known (Bishop, 2006), however, the effect of these difference for policy search have never been evaluated.

3. Inference for policy search

Many policy search algorithms (Kober & Peters, 2009; Vlassis et al., 2009; Heidrich-Meisner & Igel, 2009) use inference or inference related methods to iteratively optimize the policy.

In order to use inference for policy search we define a binary reward event $R = 1$ as observed variable. To simplify notation we will always write R when we mean $R = 1$. The probability of this reward event is given by $p(R|\tau) \propto \exp(-C(\tau))$, where τ is a trajectory and $C(\tau)$ are the associated costs. This is a common method to transform an optimization problem into an inference problem (Toussaint, 2009). We want to find parameter vectors θ with high evidence

$$p(R;\theta) = \int_{\tau} p(R|\tau) p(\tau;\theta) d\tau,$$

where τ is a trajectory and $p(\tau;\theta)$ is the parametric model of the trajectory distribution. The policy π is contained in this model.

We can now introduce a variational distribution $q(\tau)$ which is used to decompose the log-evidence

$$\log p(R;\theta) = \mathcal{L}(q,\theta) + \mathrm{KL}(q||p_R), \tag{1}$$

where

$$\mathcal{L}(q,\theta) = \int_\tau q(\tau) \log \frac{p(R|\tau)p(\tau;\theta)}{q(\tau)} d\tau$$

is the lower bound of the log evidence and

$$\mathrm{KL}(q||p_R) = -\int_\tau q(\tau) \log \frac{p(\tau|R;\theta)}{q(\tau)} d\tau \tag{2}$$

is the KL-divergence between the q and the *reward-weighted* trajectory distribution

$$p_R(\tau) = p(\tau|R;\theta) = \frac{p(R|\tau)p(\tau;\theta)}{p(R;\theta)} \tag{3}$$

The correctness of Equation (1) can be easily verified by substituting Equation (3) into Equation (2). Note that this decomposition is the same as used in expectation-maximization (EM) and variational inference algorithms. It has also already been used in (Furmston & Barber, 2010) for using variational inference for learning the model of discrete MDPs.

The lower bound $\mathcal{L}(q,\theta)$ is now iteratively improved by an expectation (E-) and a maximization (M-) step. In the E-step, we minimize $\mathrm{KL}(q||p_R)$ with respect to q. Since $\log p(R;\theta)$ is fixed, the lower bound has to increase. In the M-step we maximize the lower bound $\mathcal{L}(q,\theta)$ with respect to θ.

3.1. M-Projection: Monte-Carlo EM-based Policy Search Algorithms

Monte-Carlo (MC) EM-based algorithms (Kober & Peters, 2009; Vlassis et al., 2009) use a sample based approximation for q, i.e. in the E-step they minimize the KL-divergence $\mathrm{KL}(q||p_R)$ by setting $q(i) \propto p(R|\tau_i)p(\tau_i;\theta)$ for a discrete set of samples τ_i. Subsequently, the $q(i)$ are used to replace the integral in the lower bound $\mathcal{L}(q,\theta)$ by a sum. The lower bound therefore reads

$$\begin{aligned}\mathcal{L}(q,\theta_{\text{new}}) &= \sum_{\tau_i} p(R|\tau_i)p(\tau_i;\theta_{\text{old}}) \log \frac{p(\tau_i;\theta_{\text{new}})}{p(\tau_i;\theta_{\text{old}})} \\ &= -\mathrm{KL}(p_R(\tau)||p(\tau;\theta_{\text{new}})) + \text{const}\end{aligned}$$

As we can see maximizing the lower bound with respect to the new parameter vector θ_{new} is equivalent to calculating the *M-projection* of $p_R(\tau)$. Note that this is exactly the same lower bound as given in (Kober & Peters, 2009) for the PoWER and RWR algorithm. Thus, these algorithms are special cases of the decomposition shown in Equation 1.

3.2. I-projection: Variational Inference for Policy Search

In the variational approach, a parametric representation of q is used instead of a sample-based approximation. We choose $q(\tau;\omega)$ to be from the same family of distributions as $p(\tau;\theta)$. Now, we will use a sample-based approximation to replace the integral in the KL-divergence $\mathrm{KL}(q||p_R)$ needed for the E-step. Thus we need to minimize

$$\mathrm{KL}(q||p_R) = -\sum_{\tau_i} q(\tau_i;\omega)/Z_q \log \frac{p_R(\tau_i)/Z_p}{q(\tau_i;\omega)/Z_q}, \tag{4}$$

with respect to ω, which is equivalent to the *I-projection* of $p_R(\tau)$. The terms Z_q and Z_p are used to normalize the sample-based approximations. The M-step now trivially reduces to setting the new parameter vector θ_{new} to ω.

Both algorithms only differ in the used projections of $p_R(\tau)$. As the projections are in general different, they converge to a different (local) maximum of the lower bound $\mathcal{L}(q,\theta)$. When using a Gaussian model distribution, the I-projection concentrates on a single mode. This is not a problem if all modes are almost equally good, however, the I-projection might also choose a sub-optimal mode (which has lower reward probability). In our evaluations we could not observe this problem. The M-projection always averages over all modes and therefore might also include large areas of low reward in the distribution. Hence, we consider the use of the I-projection to be less harmful. If the target distribution is concave, both projections yield almost the same solutions, however, using the I-projection is computationally more demanding.

The discussed projections are applicable for any kind of policy search problems, however, in this paper we will focus on single-step decision problems with high dimensional action spaces because these problems are of high importance for motor skill learning with motion primitives.

4. Policy Search in multiple situations

In this paper we concentrate on policy search in multiple situations. Thus, we want to learn a policy $\pi(\mathbf{w}|\mathbf{s}^0;\theta)$ for choosing the parametric description $\mathbf{w}$ of our movement plan when being in situation $\mathbf{s}^0$.

We will treat the policy search problem as 1-step reinforcement learning problem and neglect any sequential nature of the decision problem. The agent chooses its desired trajectory description $\mathbf{w}$ in the initial state $\mathbf{s}^0$ and then observes the whole trajectory τ and the associated costs $C(\tau)$ as one big step. The trajectory τ_i itself is therefore determined by the state-action pair $\langle \mathbf{s}_i^0, \mathbf{w}_i \rangle$ and its associated costs $C(i)$. All derivations from Section 3 are still valid, we just replace the trajectories τ_i with the state-action pairs $\langle \mathbf{s}_i^0, \mathbf{w}_i \rangle$.

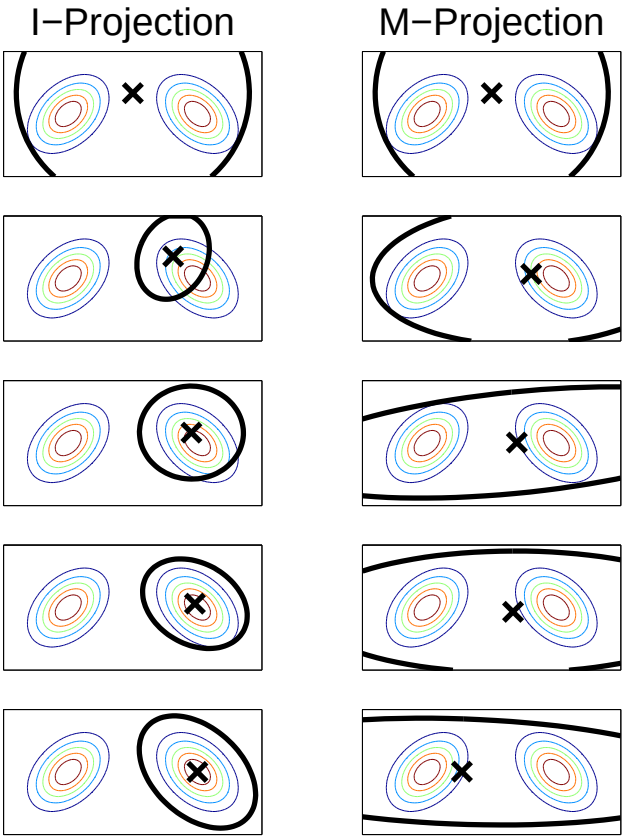


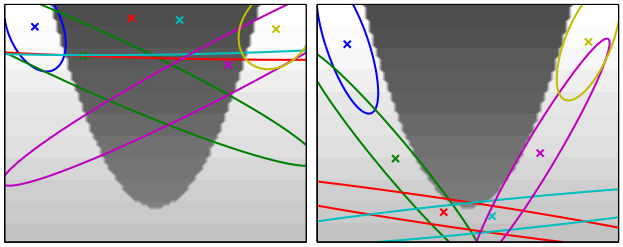
Figure 1. Comparison of the Variational Policy Search algorithm using the I-projection against the M-projection on a bi-modal reward function. A simple Gaussian distribution was used as model distribution. The M-projection tries to average over both modes, while the I-projection concentrates on a single mode.

As samples we will always use the whole history of the agent, i.e. we will use samples from all situations $\mathbf{s}_i^0$ experienced so far. For the sake of simplicity, we neglected any importance weights in Equation 4 which should be used to compensate for the fact that the history of the agent is usually not sampled uniformly from the state-action space. In the subsequent discussion we assume that each dimension of the parameter vector has been scaled to the interval $[0;1]$.

If the reward weighted probability $p_R(\tau_i)$ is very close to 0 we can't use the log function. Instead, we use a penalty term of $-P_z$ for $\log p_R(\tau_i)$. It turned out that reasonable settings of this value have to scale exponentially with the number of dimensions of the parameter space to account for the increasing volume of the search space.

4.1. Approximate Distribution

For representing $p(\mathbf{s}_i^0, \mathbf{w}_i; \theta)$ we use Gaussian distributions $\mathcal{N}([\mathbf{s}^0; \mathbf{w}]|\mu, \Sigma)$. Since a Gaussian is a rather simple representation we re-estimate the Gaussian for a new, currently



(a) M-projection (b) I-projection

Figure 2. Comparison of I-projection and M-projection on a non-concave reward function. Dark background indicates negative reward. The model distribution is a Gaussian of which the mean (indicated by 'x') of the state variable (x-axis) has been clamped at different locations. The M-projection again tries to average over the non-concave function while the I-projection nicely approximates the desired policy.

active situation $\mathbf{s}_t^0$. For every re-estimation, the state components of μ are clamped at $\mathbf{s}_t^0$ by putting a sharply peaked prior on these components (see next section).

In Figure 1 and 2, we illustrated the difference of policy search with the M- and the I-projection for bimodal and non-concave target distributions. For the bi-modal distribution, the M-projection concentrates on both modes while the I-projection only tries to cover one mode. For the non-concave target distribution we assumed that the first variable represents a state variable which is observed. Therefore, we clamped this dimension of the mean of the Gaussian to be the observed value. Again, the M-projection tries to average over the non-concave function, and hence also includes regions of low reward, while the I-projection nicely approximates the desired distribution.

4.2. Minimization of the I-projection

The I-projection $\mathrm{KL}(q||p_R)$ is difficult to use because it can't be calculated in closed form. We have to rely on non-linear optimization methods, i.e. gradient descent. However, optimizing directly the parameters of a Gaussian is difficult because of the quadratic number of parameters needed to represent the covariance matrix.

Hence, we propose a sample oriented approach which is computationally more tractable. For each sample we introduce a weighting v_i. These weightings are used to calculate the weighted maximum likelihood (ML) estimate from the data-points. We will denote the weighted sample mean as $\mathbf{m}$ and the weighted sample covariance matrix as $\mathbf{S}$. The weights v_i are normalized such that $\max_i v_i = 1$.

In order to clamp the state-space part of the mean μ at the current initial state $\mathbf{s}_t^0$, we combine the ML-estimate $\mathbf{m}$ with a Gaussian prior distribution $P(\mu|\mathbf{s}_t^0) = \mathcal{N}(\mu|\mu_0, S_0)$ with

$\mu_0 = \left[\mathbf{s}_t^0, \mathbf{0.5}\right]^T$ and S_0 is a diagonal matrix which is set such that the prior is sharply peaked for the state variables $\mathbf{s}^0$ and almost flat in the action space. The mean μ of our Gaussian distribution is then given by

$$\mu = (S_0^{-1} + \mathbf{S}^{-1})^{-1}(S_0^{-1}\mu_0 + \mathbf{S}^{-1}\mathbf{m})$$

For the covariance matrix Σ of our model, we also use a combination of a prior covariance matrix C_0 and the weighted sample covariance S.

$$\Sigma = \frac{\sum_i v_i \mathbf{S} + \alpha C_0}{\sum_i v_i + \alpha}, \quad C_0 = k \cdot \mathrm{diag}([\sigma_i^2]) + \sum_{j=1}^{l-1} c_j \Sigma_j,$$

where Σ_j are the covariance matrices of the previous iterations of VIP. The Σ_j are used to incorporate previous search directions into the current search. The parameters k, $\sigma_{1:d}^2$ and $c_{1:l-1}$ are also optimized by gradient descent.

After calculating μ and Σ we can evaluate the KL-divergence $\mathrm{KL}(q||p_R)$ on our sample points by the use of Equation 4. The gradient with respect to v_i, α, k, σ_i^2 and c_j is calculated numerically by finite differences. Subsequently we apply standard gradient descent augmented by a line search algorithm to estimate the optimal learning rate. The algorithm always runs for 10 iterations.

We also use a slight modification of the original variational algorithm. Instead of using the model distribution $p(\mathbf{s}_i^0, \mathbf{w}_i; \theta)$ for calculating the reward weighted trajectory distribution $p_R(i)$ we use the sample weights v_i found by the previous KL-divergence minimization, i.e $p_R(i) = v_i p(R|\mathbf{s}_i^0, \mathbf{w}_i)$. This turned out to be numerically more stable in high dimensional parameter spaces.

4.3. Reward Transformation

Instead of using the standard reward transformation $p(R|\mathbf{s}_i^0, \mathbf{w}_i) = \exp(-C(\mathbf{s}_i^0, \mathbf{w}_i))$, we will use a baseline $V(\mathbf{s}_i^0)$ and also introduce a scaling factor ρ to the costs, i.e. $p(R|\mathbf{s}_i^0, \mathbf{w}_i) = \exp\left(-(C(\mathbf{s}_i^0, \mathbf{w}_i) - V(\mathbf{s}_i^0))/\rho\right)$. Both mechanisms help to improve accuracy of the algorithm as well as to reduce the number of required iterations.

As baseline we use an estimate of the value $V(\mathbf{s}_i^0) = \int_{\mathbf{w}} C(\mathbf{s}_i^0, \mathbf{w}) p(\mathbf{w}|\mathbf{s}_i^0; \theta) d\tau$ at state $\mathbf{s}_i^0$. In order to do so, we use the tuples $\langle \mathbf{s}_i^0, C_i \rangle$ as data points to estimate a Gaussian cost model. Each data point gets again weighted by the weights v_i found by the previous KL-minimization. Subsequently, we condition this Gaussian cost model on the scenario states $\mathbf{s}_i^0$ of our samples. This results in a linear Gaussian model from which we use the (state-dependent) mean as baseline $V(\mathbf{s}_i^0)$.

The scaling factor ρ regulates the greediness of our distribution $p(R|\mathbf{s}^0, \mathbf{w})$. We use the standard deviation of the conditioned Gaussian cost model to determine ρ.

Algorithm 1 Variational Policy Search

Require: History of the agent $H = \langle \mathbf{s}_i^0, \mathbf{w}_i, C_i \rangle$, current scenario $\mathbf{s}^0$, initial covariance Σ_0.

$\mu_0 = [\mathbf{s}_0; \mathbf{1}/2]$
$v_i = \mathcal{N}([\mathbf{s}_i^0; \mathbf{w}_i] | \mu_0, \Sigma_0)$ for all i
for $l = 1$ to L **do**
  Estimate $V(\mathbf{s}_i^0)$ and ρ by calculating a Gaussian cost model using v_i.
  Calculate cost weighted trajectory distribution
    $p_R(i) = v_i \exp\left(-\left(C_i - V(\mathbf{s}_i^0)\right)/\rho\right)$
  Check effective number of examples, eventually reduce sharpness of p_R
  while $\sum_i p_R(i) / \max_j p_R(j) < n_{\mathrm{act}}$ **do**
    $p_R(i) = p_R(i)^{0.9}$ for all i
  end while
  Acquire new v_i, μ and Σ (minimize $\mathrm{KL}(q||p_R)$)
    $[v_i, \mu_l, \Sigma_l] = \text{I-project}(p_R, H, \{\Sigma_0, \cdots \Sigma_{l-1}\})$
  Set new model distribution...
    $p(\mathbf{s}^0, \mathbf{w}; \theta) = \mathcal{N}([\mathbf{s}^0; \mathbf{w}] | \mu_l, \Sigma_l)$
end for
Calculate policy (conditional Gaussian)
$\pi(\mathbf{w}|\mathbf{s}^0; \theta) = p(\mathbf{s}^0, \mathbf{w}; \theta) / p(\mathbf{s}^0; \theta)$

If the effective number of activations of our target distribution $p_R(i)$ gets too small (i.e. $\sum_i p_R(i) / \max_j p_R(j) < n_{\mathrm{act}}$) we do not have enough data-points to reliably estimate the Gaussian models. Hence, we iteratively reduce the sharpness of $p_R(i)$ by setting all $p_R(i)$ to $p_R(i)^{0.9}$ until the effective number of samples is larger than n_{act}. The parameter n_{act} has to be specified by the user and depends on the dimensionality of the state-space (in our experiments we varied the value between 5 and 15).

4.4. Estimating the policy

So far we have estimated a model which describes the probability of whole trajectories, i.e. in our case a probability distribution over the state *and* action space. In order to determine the policy $\pi(\mathbf{w}|\mathbf{s}_t^0; \theta)$ we just have to condition on the current state $\mathbf{s}_t^0$. This is again a linear Gaussian model which can be easily calculated.

The whole algorithm is summarized in Algorithm 1. The number of iterations L was always set to 10. For performance reasons we only use the last N examples (between 100 and 10000) from the history. The initial covariance Σ_0 as given in Algorithm 1 is typically almost flat in the action space and state space. The method is almost invariant to this setting.

In difference to MC-EM based algorithms like RWR or CRKR we use several iterations to estimate the model distribution. Additionally, the introduced scaling factor ρ of the reward function helps to set the greediness of the result-

ing distribution correctly. If we would use the M-projection and only apply one iteration ($L = 1$) without the scaling factor ρ and the baseline $V(\mathbf{s}_i^0)$, VIP reduces to RWR.

5. Experiments

In our evaluations of the algorithms we always use the median over 20 trials. The median is used to get rid of outliers, 1 or 2 trials out of 20 usually did not find good solutions.

We first evaluate our algorithm on a Cannon Toy Task. Here, the task is to hit a target located at distance d with a cannon ball. The controls are the launching angle α and the launching velocity v of the cannon ball. The angle was restricted to $[0;\pi/2]$ and the velocity to $[0;10]m/s$. The cannon-ball was modelled as 1-kg point mass, gravity and a horizontal wind force f act on the ball. The wind force f can be in the range of $[0;1]$ and the target locations were also restricted to $[0;10]$. This results in a 2-dimensional state space $\mathbf{s}^0 = [d,w]$ and a two dimensional parameterspace $\mathbf{w} = [\alpha, v]$. As reward function we used 20 times the negative squared distance of the impact position to the target. Note there are several solutions to hit a target at a certain distance, rendering the reward function multimodal. We compared our algorithm using the I-projection and M-projection against the CRKR algorithm. Every 50 episodes we evaluated the policy at 20 randomly chosen states (which were fixed for every evaluation). The parameter n_{act} was set to 5 and P_z to 10.

The result can be seen in Figure 3. The I-projection clearly outperformed the M-projection in learning speed as well as in the quality of the learned policy. The final average distance to the target was $0.08m$ with the I-projection while the final policy of the M-projection missed the target at a average distance of $0.26m$. The learning speed of CRKR matched the speed of the M-projection, but could not find as good solutions. We also compared both approaches with the finite difference policy gradient algorithm using a fixed set of basis functions (Kober et al., 2010) in the state space. The algorithm did converge after approximately 10^5 episodes, which is not shown in Figure 3 due to the bad performance.

5.1. 2 and 4 Link Humanoid Balancing

Here we use a 2-link and a 4-link model to learn dynamic humanoid balancing strategies. The masses and lengths of the links as well as the maximum torques were chosen to crudely match a human.

The joints of the 2-link model resemble the ankle and the hip joints. For a more exact description of the model please refer to (Atkeson & Stephens, 2007). The robot is pushed with a certain force $0 \leq F \leq 25$Ns which results in an immediate jump in the joint velocities. The robot has to learn

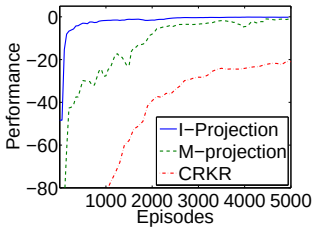


Figure 3. Evaluation of VIP on Cannon-Toy task. We compared our algorithm using the I and the M-projection. The I-projection converges much faster and also produces a final policy with higher quality. The competing algorithm CRKR could not find as good solutions.

to keep balance. This requires completely different strategies for different forces (Atkeson & Stephens, 2007). If the joints leave the intervals $\phi_1 \in [-0.4;0.8]$ or $\phi_2 \in [-0.1;1.6]$ the robot has fallen and the episode is terminated. An episode is considered as successful if the robot has managed to keep balance for $5s$. The state space is defined by the applied (one dimensional) force F. We used the following reward function

$$C(\tau) = -2000(T-5)^2 - 0.01 \sum_{t=1}^{T} \mathbf{u}_t^T \mathbf{u}_t,$$

where T is the point in time the robot falls over (or 5s if the robot keeps balance).

The whole movement representation consisted of 19 parameters. Since the exact representation of the movement is of minor importance for this paper we refer to the supplementary material for further information. For performance reasons, we always create 30 samples from the currently estimated policy. The parameter n_{act} was again set to 5 and the punishment for including samples with zero probabilities to $P_z = 300$ In our first experiment we compared our algorithm to CMA-ES, which is a highly competitive stochastic optimizer, in a single situation setup with $F = 25$Ns. We evaluated VIP one time with learning all diagonal entries σ_i^2 of the covariance matrix and VIP when keeping these factors fixed. As we can see in Figure 4(a), VIP with the full representation performed best. VIP with the fixed diagonal entries showed similar performance as the CMA-ES algorithm. Because of the huge computational requirements of the full representation (one trial runs for $10h$) we will only use the fixed diagonal representation (one trial runs for 90min) for the remaining experiments. In the next experiment (Figure 4(b)) we used a small noise for our force F which was uniformly sampled from interval $[-2.5;0]$Ns. This noise was known to the VIP, however, as CMA-ES is inherently unaware of the state $\mathbf{s}^0$, it could not learn a useful policy. The VIP algorithm was only slightly affected by the noise. The final performance was similar as learning without noise.

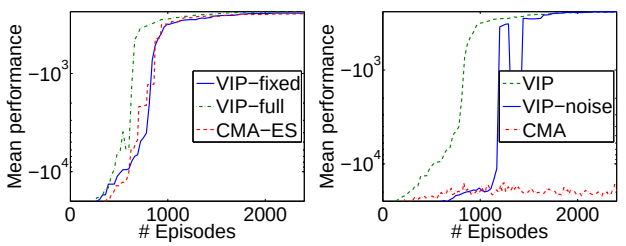


Figure 4. (a)Comparison of the VIP with full representation of the covariance (VIP-full) and the fixed representation (VIP-diag) with the CMA-ES algorithm. In order to compare our algorithm to CMA-ES, we only used a single force $F = 25$Ns. We use the maximum value seen so far for the plot of CMA-ES. (b) In this experiment we added a uniformly distributed noise $\varepsilon \in [-2.5; 0]$ to the force F. As the CMA-ES is unaware of this noise it could not cope with this setting. VIP was only sightly disturbed and could find solutions of the same quality as in (a).

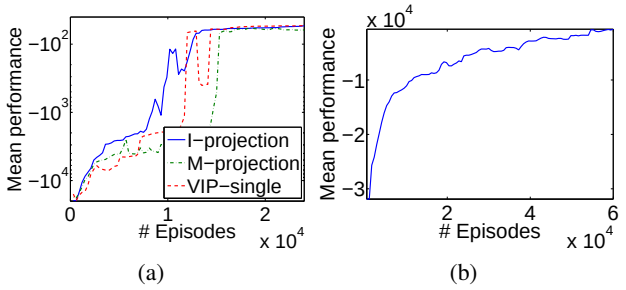


Figure 5. (a)Comparison of the I-projection and M-projection on the multi-force setup. VIP-single denotes learning for each force separately. The I-projection could outperform the M-projection and also slightly the VIP-single setup. (b) Learning curve of the 4-link balancing experiment with random forces.

Next, we evaluated the VIP algorithm once with the M-projection and the I-projection on the multi-force setup. The force was chosen uniformly from the interval $[0, 25]$Ns. We also compared our algorithm to the noisy single situation setting. Here, we used 10 different forces from 2.5 to 25Ns and performed individual learning trials for each force (we again added a noise of $[-1.25, 1.25]$ to the force). The result can be seen in Figure 5, again, the I-projection outperformed the M-projection, however, the difference was not that extreme as in the Cannon task. Still, the final performance of the I-projection (-51.2) was better than the M-projection (-62.1) by 20%. We can also see that learning with all forces at once could slightly improve the learning speed in comparison to the average of the noisy single-force setup.

The 4-link model consisted of an ankle, a knee, a hip and a shoulder joint. In this experiment the force $\mathbf{F}$ was a

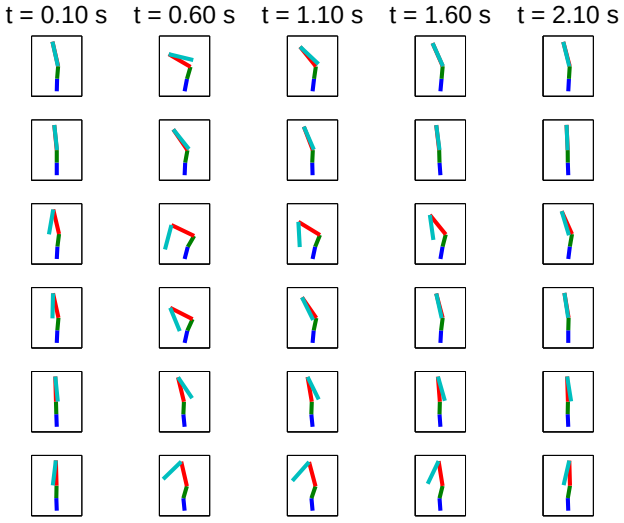


Figure 6. Learned balancing strategies for different random forces (with $|\mathbf{F}| = 25$Ns). The robot has learned to apply completely different strategies in different situations.

4-dimensional vector, denoting the force value applied to each body part. Thus, our state space is 4 dimensional. The movement representation for this task had 39 parameters. We always normalized the force vector $\mathbf{F}$, such that $|\mathbf{F}| = 25$Ns. In this experiment we used 16 randomly chosen force vectors, which were additionally perturbed by a uniformly sampled noise in the interval ± 2.5Ns. The parameter n_{act} was set to 10 and P_z to 10^5. The learning curve for this experiment can be seen in Figure 5(b). After 60000 episodes the agent was able to balance almost all experienced forces. The resulting balancing strategies for different forces can be seen in Figure 6. As we can see, the robot has learned to apply completely different strategies in different situations.

6. Conclusion and future work

Existing policy search algorithms typically approximate the policy by using the M-projection to the reward-weighted trajectory distribution. In this paper we proposed to use the I-projection of the reward-weighted trajectory distribution as interesting alternative. The I-projection alleviates many problems connected to the M-projection. While the I-projection is computationally a much more difficult operation, the 'cost-averse' policy resulting from the I-projection comes along with several advantages. Because the I-projection always wants to exclude negative examples, the algorithm does not suffer from problems which occur by averaging over non-concave or multi-modal target distributions. Consequently, it shows an increased learning speed, improved performance of the final policy and it can also be applied with ease to the learning in multiple situa-

tions simultaneously.

The main restriction of VIP is the computation time. In future, we plan to use mixture of Gaussian models to alleviate this problem. This should give us considerable speed up because we do not have to re-estimate our distributions over and over again. Furthermore, a more efficient method for calculating the I-projection is needed.

VIP is not limited to the single step reinforcement learning setup. In the future we plan to use the algorithm also for sequential decision tasks. In this case, message passing algorithms like the one presented in (Toussaint, 2009) could extend our framework.

7. Acknowledgments

Written under partial support by the European Union project # FP7-216593 (SECO), project # FP7-506778 (PASCAL2), project # 248311 (AMARSi) and by the Austrian Science Fund FWF, project # P17229-N04.

References

Atkeson, C. and Stephens, B. Multiple balance strategies from one optimization criterion. In *Proceedings of the 7th IEEE-RAS International Conference on Humanoid Robots*, 2007.

Bishop, Christopher M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, 2006.

E. Bizzi, V.C.K Cheung, A. d'Avella P. Saltiel M. Tresch. Combining modules for movement. *Brain Research Reviews*, 57:125–133, 2008.

Furmston, Thomas and Barber, David. Variational methods for reinforcement learning. In *Proceedings of the Thirteenth Conference on Artificial Intelligence and Statistics*, 2010.

Heidrich-Meisner, V. and Igel, C. Neuroevolution strategies for episodic reinforcement learning. *Journal of Algorithms*, 64(4):152–168, 2009.

Ijspeert, A.J., Nakanishi, J., and Schaal, S. Learning attractor landscapes for learning motor primitives. In *Proceedings of the Advances in Neural Information Processing Systems 15 (NIPS2002)*, pp. 1547–1554, 2002.

Kober, J. and Peters, J. Policy search for motor primitives in robotics. In *Proceedings of the Advances in Neural Information Processing Systems 22 (NIPS 2008)*. MA: MIT Press, 2009.

Kober, Jens, Oztop, Erhan, and Peters, Jan. Reinforcement learning to adjust robot movements to new situations. In *Proceedings of the 2010 Robotics: Science and Systems Conference (RSS 2010)*, 2010.

Mannor, Shie, Rubinstein, Reuven, and Gat, Yohai. The cross entropy method for fast policy search. In *Proceedings of the International Conference for Machine Learning (ICML) 2003*, pp. 512–519, 2003.

Neumann, G. and Peters, J. Learning complex motions by sequencing simpler motion templates. In *Proceedings of the International Conference on Machine Learning (ICML 2009)*, 2009.

Toussaint, Marc. Robot trajectory optimization using approximate inference. In *Proceedings of the International Conference on Machine Learning (ICML 2009)*, pp. 132, 2009.

Vlassis, Nikos, Toussaint, Marc, Kontes, Georgios, and Piperidis, Savas. Learning model-free robot control by a Monte Carlo EM algorithm. *Autonomous Robots*, 27(2): 123–130, 2009.