Guided exploration in gradient based policy search with Gaussian processes

Hunor S. Jakab, jakabh@cs.ubbcluj.ro

Lehel Csató, csatol@cs.ubbcluj.ro

Babes-Bolyai University
Faculty of Mathematics and Computer Science
Mihail Kogalniceanu str. 1, RO-400084 Cluj-Napoca

1 Introduction

Applying reinforcement learning(RL) algorithms in robotic control proves to be challenging even in simple settings with a small number of states and actions. Value function based RL algorithms require the discretization of the state and action space, a limitation that is not acceptable in robotic control. The necessity to be able to deal with continuous state-action spaces led to the use of different types of function approximators in the RL literature. However when stochasticity is present int the environment and the state-action spaces are continuous and high dimensional, estimated value functions cannot represent exactly the true value function corresponding to a policy, which leads to convergence problems when the action-selection policy is built upon the estimated state-action values [Sutton and Barto, 1998]. Gradient based policy search algorithms are more suitable for these types of control problems. In policy gradient (PG) methods a parameterized policy is improved upon each step of the learning algorithm, the direction of improvement is given by the gradient of a performance function.

Convergence is guaranteed at least to a local minimum , and PG methods are computationally simple: the explicit representation of a value-function is not required. Moreover the incorporation of domain-specific knowledge is easily achieved through the parametric form of the policy and the underlying controller. The introduction of exploratory behavior however is difficult and often plays an important role in the performance of the algorithms.

In this article we investigate the benefits of a fully probabilistic estimation of the action-value function $\mathbf{Q}(\cdot,\cdot)$ through Gaussian process regression from the perspective of efficient exploration in policy gradient algorithms. We focus on the on-line learning of control policies in continuous space-action domains where the system dynamics is unknown and the environment presents a high degree of stochasticity. We use a state-action value function approximated with a Gaussian process (GP) and develop ways to alter search directions based on the accuracy and geometric structure of the approximated value function. Our methods allow the introduction of guided exploration based on current optimality beliefs while at the same time preserving the on-policy nature of PG learning.

The paper is structured as follows: Section 2 gives a brief introduction to RL and policy gradient algorithms. In section 3 we introduce Gaussian processes, the presentation is biased towards their application to value-function estimation based on [Jakab and Csató, 2010]. To be able to use the fully probabilistic GP model for exploration at each stage of the learning we need to avoid the re-estimation of the action-value function from scratch between gradient update steps. In section 4 we describe a method that enables us to maintain a stable action-value function approximation between gradient update steps. This method also enhances sample use efficiency. In section 5 we propose two different ways to influence search directions in PG algorithms and study the changes in gradient estimation induced by the modifications. The proposed exploration scheme bridges the gap between action-value based greedy action selection and stochastic exploration in PG algorithms. Section 6 gives an illustration of the proposed methods efficacy based on two simulated control tasks and in section 7 we provide performance analysis and draws conclusions.

2 Notation and background

A mathematical representation of the reinforcement learning problem is given using Markov decision processes (MDP) [Puterman, 1994]. Equivalent with a stochastic automata, an MDP is a quadruple $M(S, A, P, R)$ with the following elements: S the set of states; A the set of actions; $P(s'|s, a) : S \times S \times A \to [0, 1]$ the transition probabilities, and $R(s, a) : S \times A \to \mathbf{R}$ the reward function. Informally the MDP describes the environment where an *agent* can act; the interactions between the agent and the environment. To build a learning system, we have to define the *decisions* an agent takes at each step, this can be modeled with a *policy*. A deterministic policy – $\pi : S \to A$ – provides direct mapping from states to actions, in the stochastic case $\pi_{\boldsymbol{\theta}} : S \times A \to [0, 1]$ is interpreted as a *conditional* probability distribution: $\pi_\theta(a|s)$ of taking action a when in state s. The benefit of stochastic policies is that they allow nondeterministic action selection thereby the possibility of exploratory behavior. In control problems stochastic policies are constructed by adding *exploratory noise* to a controller function $f_{\boldsymbol{\theta}_f} : S \to A$ which provides direct mapping from states to actions. Frequently a Gaussian distribution is being used for the noise with variance $\boldsymbol{\sigma}_\epsilon$ [1]

$$\pi_{\boldsymbol{\theta}}(a|s) = f(s, \boldsymbol{\theta}') + \mathcal{N}(0, \boldsymbol{\sigma}_\epsilon I) \tag{1}$$

Where $\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_f^T & \boldsymbol{\sigma}_\epsilon^T \end{bmatrix}^T$ is the parameter vector of the policy composed of the controller parameters $\boldsymbol{\theta}_f$ and the parameters of the exploratory noise distribution $\boldsymbol{\sigma}_\epsilon$. To simplify the notation, we drop the explicit θ from the policy, but policy changes are made via the parameter set θ.

When applying reinforcement learning in the context of robotics, we distinguish two major types of problems: (1)motor control and (2)motor planning. We refer to the learning problem as a "motor control problem"when our parameterized deterministic controller $f_{\boldsymbol{\theta}_f}$ directly computes a mapping between state and action vectors. The parameters $\boldsymbol{\theta}_f$ influence the generated actions therefore the size of the effective search space increases exponentially with the complexity of the controllable system. The magnitude of parameter change that we can make is also limited if we want to avoid chaotic behavior or even damage in case of experimenting with a real robot. On the other hand in case of "motor planning problems "the controller parameters change the shape/duration of a motion trajectory for example learning cyclic movements or improving gait sequences in case of legged robots. In these cases through the policy parameters $\boldsymbol{\theta}_f$

[1] Our treatment equally applies for multi-dimensional actions. In this case $\boldsymbol{\Sigma} = \sigma \mathbf{I}$ would be a vector containing the parameters of the covariance matrix.

we influence the desired joint configurations of the robot during a movement sequence(which can be time or phase dependent). Such polycies are called dynamic motor primitives [Ijspeert et al., 2002]. This formulation is preferred when an invers dynamics/kinematics model is available or can be accurately approximated. In this work we are going to focus on problems from the first category, however the developed methodology can easily be applied also in case of dynamic motor primitives.

The goal of RL problems is to solve the MDP, and the solution is defined as an optimal policy π^* maximizing the *expected cumulative* reward:

$$\begin{aligned} \pi^* &= \arg\max_{\pi \in \Omega_\theta} J^\pi \\ J^\pi &= E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \right] \end{aligned} \tag{2}$$

Here Ω_θ denotes the set of all possible policies determined by the parametrization, $E_\pi[\cdot]$ is the expectation with respect to a policy π, $r_{t+1} = r(s_{t+1}, a_{t+1})$ is the immediate reward, and γ is a discount factor.

We focus on gradient-based policy search algorithms that optimize eq. (2). One of the earliest PG algorithm was Williams' REINFORCE [Williams, 1992], other algorithms have been built on the same principles like TD policy gradients [Ghavamzadeh and Engel, 2007a], vanilla policy gradients [Peters and Schaal, 2008], natural policy gradients [Kakade, 2002] and a wide variety of their extensions that provide performance enhancements of some form. The gradient of J – according to the policy gradient theorem [Sutton et al., 1999] – with respect to θ is:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int ds\, p(s) \int da\, \pi(a|s)\, \nabla_{\boldsymbol{\theta}} \log \pi(a|s) Q(s,a) \tag{3}$$

where $\int ds\, p(s)$ is weighted average operator with $p(s)$ the probability distribution. The importance of the above reformulation is that in eq. (3) there is no state transition – $p(s'|s,a)$ –, making possible the approximation of the integrals with sample averages without knowing the dynamics of the system. The difficulty is that the action value function $Q(s_t, a_t)$ is still not known, however it can be replaced by Monte Carlo estimates of the true value function. These simplifications are the core of Williams' REINFORCE algorithm [Williams, 1992] where the integral representation from eq. (3) is replaced with:

$$\nabla_{\boldsymbol{\theta}} J = E \left[\sum_{t=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log \pi(a_t|s_t) \sum_{i=0}^{H-t} \gamma^i R(s_{t+i}, a_{t+i}) \right]_\tau \tag{4}$$

Here $E_\tau[\cdot]$ is sample average for *roll-outs* – i.e. different experiments with the same policy – and summations stand for empirical averages. Although episodic REINFORCE is one of the most basic policy gradient algorithms it is a good candidate to evaluate the efficiency of exploration schemes.

The convergence of the algorithm can be guaranteed at least a to local maximum, but the high variance of the estimated gradient in eq. (4) leads to very slow convergence of the algorithm. An improvement possibility is to *approximate* the action-value function $Q(s,a)$ and use it directly in eq. (3) instead of the high-variance Monte Carlo samples from eq. (4). For value function approximation we use GP-s, presented in the next section.

3 Gaussian process value function approximation

To approximate action value functions we use as training data the state-action pairs $x_t \doteq (s_t, a_t)$ encountered during trajectories and the corresponding – possibly – discounted cumulative rewards $\hat{Q}(s_t, a_t) = \sum_{i=0}^{H-t} \gamma^i r_{t+i}$ as noisy targets.[2] The regression is performed directly in a function space, with the resulting $f(\cdot, \cdot)$ the approximation of the action-value function. The elements of the kernel matrix $\mathbf{K}_q$ are given by $\mathbf{K}_q(i,j) = k_q(x_i, x_j)$ and we used the notation k_q to emphasize that the kernel function operates on state-action pairs. Assume that n points have already been processed, therefore we have a GP built on the data set $\mathcal{D} = (x_i, \hat{Q}_i)_{i=1,n}$. Since GP's provide a random function, to estimate the action-value of a new state-action pair, $x_{n+1} \doteq (s_{n+1}, a_{n+1})$, we compute the predictive mean (5) and variance (6) functions conditioned on the data, given by the posterior GP [Rasmussen and Williams, 2006]:

$$\begin{aligned} f_{n+1}|\mathcal{D} &\sim \mathcal{N}(\mu_{n+1}, \mathrm{cov}(f_{n+1})) \\ \mu_{n+1} &= \mathbf{k}_{n+1}\boldsymbol{\alpha}_n \qquad (5) \\ \mathrm{cov}(f_{n+1}) &= k_q(x_{n+1}, x_{n+1}) - \mathbf{k}_{n+1}\mathbf{C}_n\mathbf{k}_{n+1}^T, \qquad (6) \end{aligned}$$

where $\boldsymbol{\alpha}_n$ and $\mathbf{C}_n$ are the parameters of the posterior GP:

$$\boldsymbol{\alpha}_{n+1} = [\mathbf{K}_q^n + \boldsymbol{\Sigma}_n]^{-1}\hat{\mathbf{Q}}, \quad \mathbf{C}_{n+1} = [\mathbf{K}_q^n + \boldsymbol{\Sigma}_n]^{-1}. \qquad (7)$$

with $\boldsymbol{\Sigma}_n$ covariance of the observation noise and $\mathbf{k}_{n+1}$ denotes a vector containing the covariances between the new point and the training points:

$$\mathbf{k}_{n+1} = [k_q(x_1, x_{n+1}), \ldots, k_q(x_n, x_{n+1})]. \qquad (8)$$

The parameters $\boldsymbol{\alpha}$ and $\mathbf{C}$ of the Gaussian process can be updated iteratively each time a new point $\{x_{n+1}, \hat{Q}_{n+1}\}$ is processed, this is achieved by combining the likelihood of the new datapoint and the Gaussian process from the previous step and making use of the parameterization of the posterior moments from [Csató and Opper, 2002]. Replacing $\sum_{i=0}^{H-t} \gamma^i R(s_{t+i}, a_{t+i})$ from eq. (4) with the posterior mean from eq. (5),[3] we get the GP version of the policy gradient algorithm:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = E_\tau \left[\sum_{t=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log \pi(a_t|s_t) Q_{GP}(s_t, a_t) \right] \qquad (9)$$

Using the predictive mean of the estimated action-value function instead of Monte Carlo samples significantly reduces the variance of the gradient estimates and improves the convergence rate. The probabilistic nature of the Gaussian process provides new possibilities to improve performance by influencing exploratory action selection, presented in section 5.

3.1 Related work

The use of Gaussian processes for value function approximation purposes has been investigated a number of references. [Rasmussen and Williams, 2006] modeled the value function and the system dynamics using GPs, and proposed a policy iteration algorithm. Only a batch version

[2] We assume that the targets have Gaussian noise with equal variance; one can easily use different *known* noise variances within the same framework

[3] For simplicity we denote the GP predictive mean for a state action pair (s, a) with $Q_{gp}(s, a) = \mu_{n+1}$ where $x_{n+1} = (s, a)$ and prediction is based on the previously visited data-points.

algorithm has been developed and applied for the mountain car problem. The gain of their work is that they showed that GPs are feasible in the RL framework. [Ghavamzadeh and Engel, 2007b] applied GPs in a policy gradient framework. They used a GP to approximate the gradient of the expected return function with the help of Bayesian quadratures. This extension allowed a full Bayesian view of the gradient estimation. The algorithm has been applied for the bandit problem. [Deisenroth et al., 2009] modeled both the value function and the action-value function with GPs, and proposed a dynamic programming solution using these approximations. The algorithm is called GPDP. Although these methods brought major advances in extending RL to continuous domains, in order to make them implementable on actual control problems some major simplifications have been made.Most notable is the fact that none of them deals with continuous and multi-dimensional action spaces. Some other simplifications are the assumption of Gaussian noise at the stochastic state transitions and received rewards or the availability of generative models for the system dynamics.

4 Sample reuse and continuity

In this section we address the problem of restarting the action-value function approximation after a policy change occurs. After a gradient update step we would like to build upon our previously estimated value function model while simultaneously incorporating new measurements which provide useful information. To achieve this we make use a modified version of the Kullback Leibler distance-based sparsification scheme from [Csató and Opper, 2002]. The sparsification scheme in our case serves two purposes: (1) it decreases computational costs by discarding unimportant inputs (2) it provides a way to exchange obsolete measurements with newly acquired ones.

To decide upon the addition of a new input to the support set of the GP we test for approximate linear independence in feature space. The projection error in feature space of input $n+1$ onto the space of existing feature vectors can be expressed as:

$$\gamma_{n+1} = k_q(x_{n+1}, x_{n+1}) - \mathbf{k}_{n+1}\hat{\boldsymbol{e}}_{n+1} \tag{10}$$

$$\hat{\boldsymbol{e}}_{n+1} = \mathbf{K}_n \mathbf{k}_{n+1}^T \tag{11}$$

where $\hat{\boldsymbol{e}}_{n+1}$ is the vector of projection coordinates minimizing the projection error, $\mathbf{K}_n$ is the kernel gram matrix and γ_{n+1} is the residual. By setting a threshold value for the residual we can decide which inputs are going to be added to the support set. Additionally we assign a time variable to every included data point in $\mathcal{D}$ which signifies at which stage of the learning process the data point has been added to the basis vector set. We also limit the basis vector set size.

$$D = \{(x_i, \hat{Q}_i)\} \rightarrow \{(x_i, \hat{Q}_i, t_i)\} \quad i = \overline{1,n} \tag{12}$$

Whenever a new data point is being processed which needs to be included but the maximum number of basis vectors has been reached we compute a modified score function ε for each data-point:

$$\varepsilon(i) = \frac{\alpha^2(i)}{q(i) + c(i)} + \lambda g(t(i)) \tag{13}$$

The first term in eq. (13) is the Kullback Leibler distance between two GPs $KL(GP'\|GP)$ where GP' contains the new data point and GP is obtained by replacing the data-point with its projection to the space spanned by existing basis vectors[4]. The second term $g(\cdot)$ penalizes basis

[4]For details of derivation of the KL distance see [Csató, 2002]

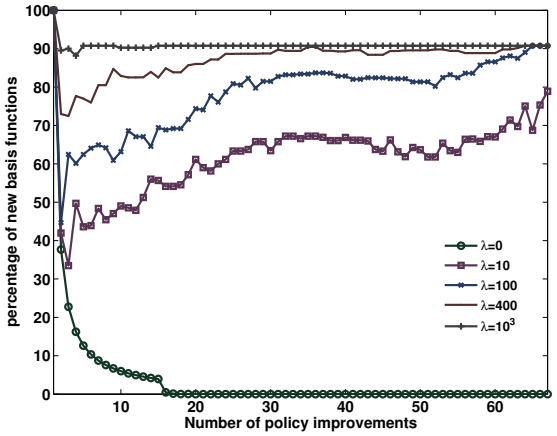


Figure 1: Composition of basis vector set as a function of λ

vectors that have been introduced in early stages of the learning process, it is a function of the time variable assigned to each basis vector. Since we want to favor the removal of out-of date basis vectors, this function needs to be monotonically increasing. In our experiments we used an exponential of the form:

$$g(t_i) = e^{c(t_i - \min_i(t_i))} \quad i = 1 \dots |D| \tag{14}$$

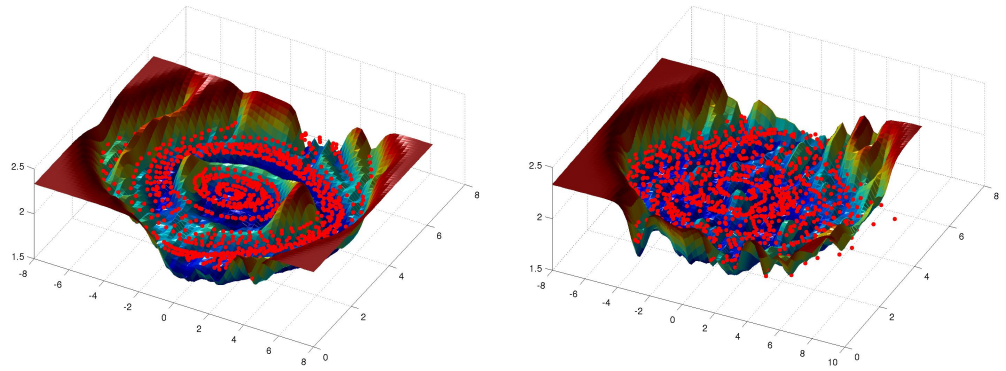
We replace the lowest scoring data point from the BV set with the new measurement. The λ term from eq. (13) serves as a trade off factor between loss of information and accuracy of representation, c is a constant. In figure 1 we see how much the choice of λ influences the composition of the basis vector set during on-line learning.

If we set λ to be a large number the time-dependent factor from the scores will outweigh the KL distance based factor in eq.(13) leading to the inclusion of all newly acquired measurements into the BV set. Setting it too small will allow too many out-of date basis vectors to be present in our representation which leads to inaccurate gradient estimates and a poor policy.

5 Guided exploration

In direct policy search algorithms the use of parameterized functions for policy representation induces a large search space, which becomes impossible to fully explore as the number of parameters increases. As a consequence, the agent has to restrict its exploration to a *subset of the search space* that is the most "promising". Several variants of policy gradient algorithms were applied to robotic control for policy optimization [Kim and Uther, 2003; Hornby et al., 2000; Peters and Schaal, 2008; Kohl and Stone, 2004; Bagnell and Schneider, 2001], where a starting policy was obtained via imitation learning or manual setup and the search procedure was restricted to the immediate neighborhood of the initial policy. The drawback of these methods is that without the existence of a starting policy, random exploration is inefficient and extremely costly. The availability of a fully probabilistic model for the value function provides an interesting opportunity to introduce directed exploratory behavior in our learning algorithms.

In what follows we explore two modalities to influence the exploration process: either to modify the exploratory noise or to modify the direction of the exploration process.



(a) (b)

Figure 2: *Predictive variances* of the GP-estimated state-value functions.The horizontal axes correspond to the state variables, and the vertical axis displays the predictive variance.The dots are the visited states of (a) fixed exploratory noise, (b) adaptive noise variance.

5.1 Influencing the exploratory noise

Our first guided exploration method is based on changing the variance of the exploratory noise $\boldsymbol{\sigma}_{ex}$ in eq. (1). We employ the properties of the *estimated* state-action value function $Q_{GP}(s,a)$. Since it is a random variable, we have access to the posterior variance of the function, providing information about the uncertainty present in different regions of the parameter space. Our modification is that in regions with high uncertainty the exploratory noise should be higher; this is achieved by replacing the *fixed* noise with one obtained from the GP model of the state-action value function in (7). The modified policy is:

$$\begin{aligned} \pi_{\boldsymbol{\theta}} &= f(s,\boldsymbol{\theta}) + \mathcal{N}(0,\boldsymbol{\sigma}^2_{GP} I) \\ \boldsymbol{\sigma}^2_{GP} &= \lambda \left(k_q(x^*,x^*) - \mathbf{k}^* \mathbf{C}_n \mathbf{k}^{*T} \right) \end{aligned} \quad \text{with } x^* = \{s, f(s,\boldsymbol{\theta})\} \tag{15}$$

Here $\mathbf{k}^* = [k_q(x^*,x_1)\dots k_q(x^*,x_n)]$ is the vector containing the covariances of the new data-point $x*$ and the data-points from $\mathcal{D}$. When starting, the GP-based approximation is inaccurate, therefore the predictive variance is large everywhere. The large predictive variance facilitates higher exploration rates in the early phase of the learning. As learning progresses and we add more data, the predictive variance decreases in the neighborhood of these points, decreasing also the added exploratory noise.

The effect of this exploration scheme is displayed on Fig. 2, where – for illustration – we plotted two surfaces corresponding to the predictive variances corresponding to the different guided exploration strategies. [5] The control task – presented in detail in Section 6 – was the inverted pendulum control where the state-space contains the angle and the angular velocity respectively. We see that in case of fixed exploratory noise the visited states tend to lie on tighter regions of the state-space, whereby for guided exploratory noise a much better coverage of the important regions is provided. The guided exploratory noise changes the derivative of the

[5]Similar graphs were obtained for the *mean* action-value functions, here we used state-value functions for better visibility.

policy since the noise term also depends on the policy parameters through x^* from eq. (15).

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \log \pi(a|s) &= \frac{(a - f_\theta(s))}{\boldsymbol{\sigma}^2_{GP}} \partial_{\boldsymbol{\theta}} f_\theta(s) \\ &+ \frac{(a - f_\theta(s))^2 - \boldsymbol{\sigma}^2_{GP}}{\boldsymbol{\sigma}^4_{GP}} \partial_{\boldsymbol{\theta}} \boldsymbol{\sigma}^2_{GP} \end{aligned} \tag{16}$$

The first part of eq. (16) involves the derivative of the deterministic controller, easily calculated for a variety of controller implementations. The second term involves differentiation through the covariance function of the GP approximator:

$$\begin{aligned}\nabla_\theta \boldsymbol{\sigma}^2_{GP} &= \left(\frac{\delta \boldsymbol{\sigma}^2_{GP}}{\delta \theta_1}, \ldots, \frac{\delta \boldsymbol{\sigma}^2_{GP}}{\delta \theta_m} \right) \quad m = |\theta| \\ \frac{\delta \boldsymbol{\sigma}^2_{GP}}{\delta \theta_i} &= \frac{\delta}{\delta \theta_i} k_q(x^*, x^*) \\ &- \frac{\delta}{\delta \theta_i} \left(\sum_{i,j=1}^{N} C_{i,j} k_q(x_i, x^*) \cdot k_q(x_j, x^*) \right) \end{aligned} \tag{17}$$

We consider the covariance matrix C constant at the time of the differentiation, it does not need to be differentiated. We get the following expression:

$$\frac{\delta \boldsymbol{\sigma}^2_{GP}}{\delta \theta_i} = \frac{\delta}{\delta \theta_i} k_q(x^*, x^*) - 2C\mathbf{k}^* \frac{\delta \mathbf{k}^*}{\delta \theta_i} \tag{18}$$

The derivation of $k_q(\cdot, \cdot)$ with respect to the parameters θ_i, $i = \overline{1, m}$ can be calculated for several covariance functions.

5.2 Influencing search directions

A second proposed modification to improve exploration is to influence not only the variance of the noise but also controller output. The underlying idea is that the agent should better explore regions of the state-action space which potentially have higher Q-values, leading to an increase in the expected cumulative reward and possibly a better policy. Consider the case when the agent is in a state s_t at time t. The next step of the algorithm is to choose an action according to the action selection policy $\pi(a|s)$.

We are interested in constructing a policy favoring actions with higher estimated Q-values, still taking into account the output of our deterministic controller. We propose a policy $\pi(a|s)$ as being a Gibbs distribution over actions from the neighborhood of $f_\theta(s)$ [Neal, 2010]:

$$\pi(a|s) = \frac{e^{\beta E(s,a)}}{Z(\beta)}, \quad \text{where } Z(\beta) = \int da\, e^{\beta E(s,a)} \tag{19}$$

The term $Z(\boldsymbol{\theta})$ is a normalizing constant and β is the inverse temperature. To include the deterministic controller f_θ in the action selection, we construct the energy function $E(s, a)$ such that only actions neighboring $f_\theta(s)$ have significant selection probability. At the same time we want to assign higher probability to actions that – in the current state s_t – have higher estimated Q-values and the energy function has the following form:

$$E(s, a) = Q_{GP}(s, a) \cdot \exp\left[-\frac{\| a - f_\theta(s) \|^2}{2\sigma_e^2} \right]$$

It is composed of the GP-estimated Q-value $Q_{GP}(s, a)$ for the state-action pair (s, a) and a Gaussian on the action space to limit the selection to the neighborhood of the controller output $f_\theta(s)$. The variance parameter σ_e is fixed, but making it dependent on the GP predictive variance can also be considered. To implement this exploration scheme, we have to compute the log-derivative of the policy from eq. (19) that is:

$$\begin{aligned} \frac{\delta}{\delta \boldsymbol{\theta}} \log \pi(a|s) &= \frac{\delta}{\delta \boldsymbol{\theta}} \left(\beta E(s, a) - \log Z(\beta)\right) \\ &= \beta \frac{\delta}{\delta \boldsymbol{\theta}} E(s, a) - \frac{\beta}{Z(\beta)} \int da\, e^{\beta E(s,a)} \frac{\delta}{\delta \boldsymbol{\theta}} E(s, a) \\ &= \beta \left(\frac{\delta}{\delta \boldsymbol{\theta}} E(s, a) - \int da\, \pi(s, a)\, \frac{\delta}{\delta \boldsymbol{\theta}} E(s, a) \right) \end{aligned} \tag{20}$$

Differentiating the energy function is not difficult, since only the Gaussian term depends on the parameters:

$$\frac{\delta}{\delta \boldsymbol{\theta}} E(s, a) = E(s, a) \frac{(a - f_\theta(s))}{\sigma_e^2} \frac{\delta}{\delta \boldsymbol{\theta}} f_\theta(s) \tag{21}$$

Combining eqs. (21), (20), and inserting into eq. (9), we have the expression for the gradients. In practice, the integral from eq. (20) cannot be evaluated, we instead sample actions from the neighborhood of $a = f(s, \theta)$ from a Gaussian distribution with variance corresponding to the model confidence at (s, a). We calculate the predictive Q-values for these points with the help of our GP action-value function approximator and use a discrete Gibbs distribution in the selection process. Fig. 3 illustrates the results of the proposed methodology on the inverted pendulum problem. We plotted the surface corresponding to the estimated state-value function.[6] We see on the first figure that, with random exploration in sub-figure (a), there are regions on the perimeter of the state space with high estimated value that are not explored properly. If we use the exploratory mechanism defined above, the high values of these regions facilitate exploration, thereby improve the algorithm performance. Moreover, we see that, for guided exploration, regions of the state space with high values have a higher concentration of visited points. This is important since small differences in value on high importance region of the state space can influence the performance of the learned policy.

6 Performance evaluation

We test the above presented methods on a simulated pendulum control problem where both the state and the action spaces are continuous. A state variable consists of the angle and angular velocity of the pendulum $s = \begin{bmatrix} \phi & \omega \end{bmatrix}^T$ and we normalized the angle to the $[0 2\pi]$ interval. Actions are the torques that we can apply to the system, and are limited to a $[-5, 5]$ interval. The Hamiltonian for the pendulum is:

$$\mathcal{H} = \frac{1}{2ml} p_\phi^2 - mgl\cos(\phi) \quad p_\phi = ml^2\omega \quad q_\phi = \phi \tag{22}$$

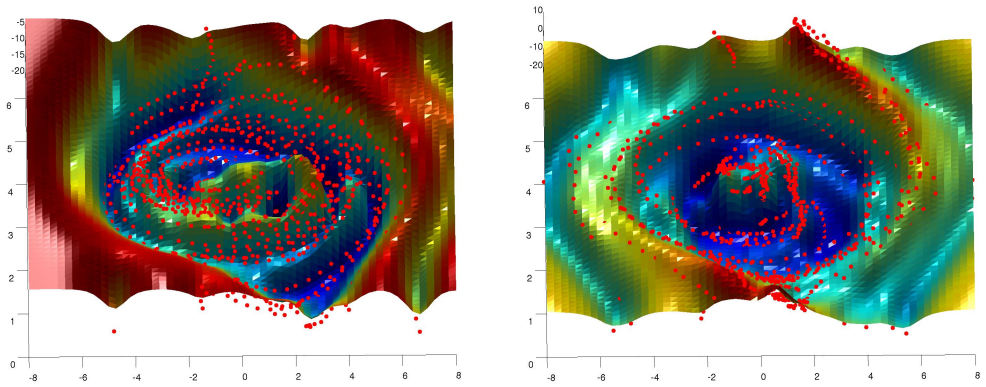
The experiments were performed with a quadratic reward function with added gaussian noise:

$$R(s, a) = (s_1 - \pi)^2 - \left(\frac{s_2}{4}\right)^2 + \epsilon \quad \epsilon \sim N(0, \sigma_r) \tag{23}$$

[6] The state action-value function cannot be graphically represented, therefore we used the state-value function for illustration.

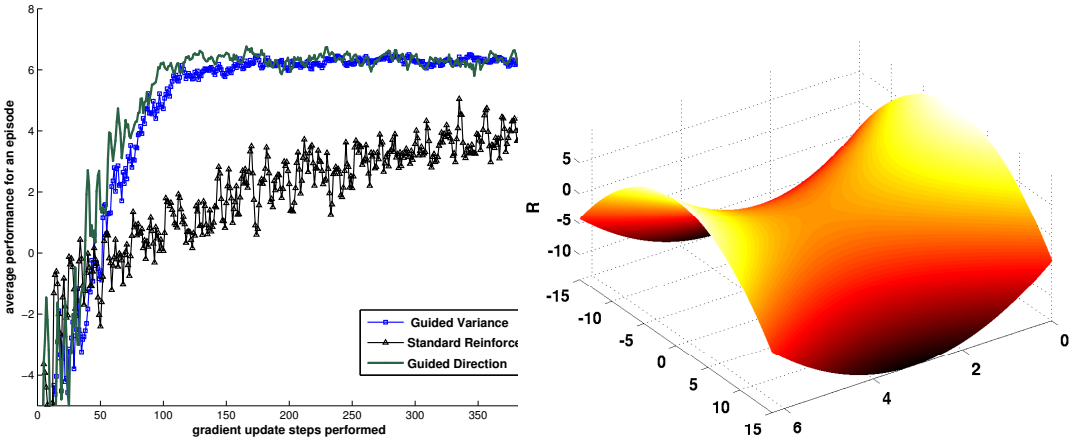
Algorithm 1 REINFORCE with GP guided exploration

Initialize policy parameterization $\pi(s, a|\boldsymbol{\theta})$
Initialize GP parameters $\boldsymbol{\alpha} = 0, \mathbf{C} = 0$
$\mathcal{D} = \emptyset$,M=const,λ=const, maxBV=const, n=0
repeat
  for $t = \overline{1, H}$ **do**
    $a_t \sim \pi(a_t|s_t)$ eq. (15), (19)
    $\gamma_{n+1} = k_q(x_{n+1}, x_{n+1}) - \mathbf{k}_{n+1}\hat{\mathbf{e}}_{n+1}$ eq. (13)
    if $\gamma_{n+1} >$ threshold **then**
      if $n + 1 >$ maxBV **then**
        for $i = \overline{1, n}$ **do**
          $\varepsilon(i) = \frac{\alpha^2(i)}{q(i)+c(i)} + \lambda g(t(i))$
        end for
        exchange D_i where $i = \arg\max \varepsilon(i)$
      else
        update $\boldsymbol{\alpha}, \mathbf{C}$ eq. (7)
        n=n+1, $\mathcal{D} = \mathcal{D} \cup (s_t, a_t)$
      end if
    else
      discard (s_t, a_t)
    end if
  end for
  Estimate $\nabla J_{\boldsymbol{\theta}}$ eq. (16), (9) or eq. (20), (9)
  if $\nabla J_{\boldsymbol{\theta}}$ converged **then**
    Update policy parameters
  end if
until policy has converged



(a) (b)

Figure 3: *GP-estimated state-value functions* of the inverted pendulum control task. The two horizontal axis correspond to the state variables namely the angle and the angular velocity, and the vertical axes to the state-action value function. The dots are the visited states and corresponding noisy value measurements in case of (a) fixed exploratory noise, (b) guided search directions.



(a) (b)

Figure 4: Evolution of the average return in case of the episodic reinforce, reinforce with GP guided exploratory noise and reinforce with GP guided exploration directions

The reward function penalizes distance from the target region as well as velocity, figure **??** shows a graphical representation.

We used as a baseline for our benchmark Williams' episodic REINFORCE algorithm [Williams, 1992]. The two versions of guided exploration discussed in section 4 were also implemented. The performance data is averaged over 10 separate experiments for each algorithm. The initial values of the learning parameters start states start-state variances were the same for all algorithms. During one experiment we performed 400 gradient update steps starting with a predefined policy parameter set. The gradient estimates are obtained from performing 3 episodes each consisting of 50 steps. In total during each experiment we executed 6000 steps. Figure **??** shows the performance evaluation of our algorithms. The vertical axis denotes the average reward received during a 50 step episode while the horizontal axis denotes the number of gradient update steps performed.

We see that the GP guided versions of the reinforce algorithm clearly outperform Williams' basic version in both convergence speed and in achieved performance. The learning curve in case of both our algorithms becomes much steeper in the early phase of the learning which can be explained with the added flexibility of exploring more important regions of the state-action space. The difference in final performance can be associated with the fact that standard reinforce tends to rapidly decrease the exploratory variance σ_{ex} from eq. (1) which leads to higher immediate rewards but poorer learned policy. The GP guided exploratory noise however does not depend directly on the policy parameters, hence it cannot be rapidly decreased by policy parameter updates. In case of GP influenced search directions as long as the controller has not converged to at least a local optimum point the Gibbs distribution based policy from eq. (19) will always maintain some degree of exploration.

7 Conclusions

In this article we presented two new modalities for adjusting different characteristics of the exploration in policy gradient algorithms with the help of Gaussian process action-value function approximation. An algorithmic form of our methods is provided in Algorithm 1. We have

shown that by using our methods the search for an optimal policy can be restricted to certain regions of the state-action space. This is especially important in case of continuous state-action spaces where full exploration is impossible. Our experimental results show that by using our guided exploration method better convergence performance can be achieved in policy gradient algorithms. The presented methods can also be viewed as a transition between off-policy and on-policy learning, which opens up further interesting research directions. In a future work we will perform more in-depth theoretical analysis and detailed testing on high dimensional simulated control problems with continuous state and action spaces.

Acknowledgements

The authors acknowledge the financial support from grant PN-II-RU-TE-2011-3-0278.

References

J. A. D. Bagnell and J. Schneider. Autonomous helicopter control using reinforcement learning policy search methods. In *Proceedings of the International Conference on Robotics and Automation 2001*. IEEE, May 2001.

L. Csató. *Gaussian Processes – Iterative Sparse Approximation*. PhD thesis, Neural Computing Research Group, March 2002.

L. Csató and M. Opper. Sparse on-line Gaussian Processes. *Neural Computation*, 14(3):641–669, 2002.

M. P. Deisenroth, C. E. Rasmussen, and J. Peters. Gaussian process dynamic programming. *Neurocomputing*, 72(7-9):1508–1524, 2009. ISSN 0925-2312. doi: http://dx.doi.org/10.1016/j.neucom.2008.12.019.

M. Ghavamzadeh and Y. Engel. Bayesian actor-critic algorithms. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 297–304, New York, NY, USA, 2007a. ACM. ISBN 978-1-59593-793-3.

M. Ghavamzadeh and Y. Engel. Bayesian policy gradient algorithms. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *NIPS '07: Advances in Neural Information Processing Systems 19*, pages 457–464, Cambridge, MA, 2007b. MIT Press.

G. Hornby, S. Takamura, J. Yokono, O. Hanagata, T. Yamamoto, and M. Fujita. Evolving robust gaits with aibo, 2000.

J. A. Ijspeert, J. Nakanishi, and S. Schaal. movement imitation with nonlinear dynamical systems in humanoid robots. In *international conference on robotics and automation (icra2002)*, 2002. URL `http://www-clmc.usc.edu/publications/I/ijspeert-ICRA2002.pdf`.

H. Jakab and L. Csató. Using Gaussian processes for variance reduction in policy gradient algorithms. In A. E.-N. E. K. G. K. G. K. T. Tómács, editor, *ICAI2010: Proceedings of the 8th International Conference on Applied Informatics*, volume 1, pages 55–63, Eger, Hungary, 2010. BVB.

S. Kakade. A natural policy gradient. *Advances in Neural Information Processing Systems 14*, 2: 1531–1538, 2002.

M. S. Kim and W. Uther. Automatic gait optimisation for quadruped robots. In *In Australasian Conference on Robotics and Automation*, 2003.

N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *in Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2619–2624, 2004.

R. M. Neal. MCMC using Hamiltonian dynamics. In S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, editors, *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC Press, 2010.

J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008.

M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, 1994.

C. E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *NIPS*, pages 1057–1063, 1999.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.