Policy Search Review

Emile Mathieu
Department of Computer Science
Ecole Nationale des Ponts et Chaussees
emile.mathieu@eleves.enpc.fr

Charles Reizine
Department of Computer Science
Ecole Nationale des Ponts et Chaussees
charles.reizine@eleves.enpc.fr

Abstract

Policy search is a subfield in reinforcement learning which deals with learning good parameters for a given policy parametrization. It can handle high-dimensional and continuous state and actions spaces. Model-free policy search is an approach to directly learn a policy based on sampled trajectories. Whereas model-based policy search adresses this issue by first learning a model of the dynamics, which then generates trajectories which are subsequently used for policy learning.

1 Introduction

1.1 Reinforcement Learning Problems

In a reinforcement learning (RL) problem, an actor lives in an high dimensional state space $\mathcal{X}$, and chooses actions $\boldsymbol{u}$ according to a control policy π. This control policy can either be stochastic, denoted by $\pi(\boldsymbol{u}|\boldsymbol{x})$, or deterministic and denoted as $\boldsymbol{u} = \pi(\boldsymbol{x})$. This action alters the state of the actor according to the dynamics of the environment, which is explicated by a probabilistic transition function $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)$. The sequence of states and actions jointly form what is called a trajectory $\boldsymbol{\tau} = (\boldsymbol{x}_1, \boldsymbol{u}_1, \boldsymbol{x}_2, \boldsymbol{u}_2, \dots)$.

We assume that the performance of the actor is evaluated by a numeric scoring system, which returns an accumulated reward $R(\boldsymbol{\tau})$ which assesses the quality of the actor's trajectory. This accumulated rewards is the sum of instantaneous rewards r_t accumulated during the trajectory: $R(\boldsymbol{\tau}) = r_T(\boldsymbol{x}_T) + \sum_{t=1}^{T-1} r_t(\boldsymbol{x}_t, \boldsymbol{u}_t)$.

Many tasks in reinforcement learning can be formulated as choosing an optimal control policy $\pi^\star$ that maximizes the expected accumulated reward $J_\pi = \mathbb{E}[R(\boldsymbol{\tau})|\pi] = \int R(\boldsymbol{\tau}) p_\theta(\boldsymbol{\tau}) d\boldsymbol{\tau}$, with $p_{\boldsymbol{\theta}}(\boldsymbol{\tau})$ the distribution over trajectories $\boldsymbol{\tau}$. For a stochastic policy: $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) = p(\boldsymbol{x}_1) \prod_{t=1}^{T} p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t) \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_t|\boldsymbol{x}_t, t)$ and for a deterministic policy $p_\pi(\boldsymbol{\tau}) = p(\boldsymbol{x}_1) \prod_{t=1}^{T} p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \pi_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t))$.

1.2 Learning paradigms

Traditional approaches in reinforcement learning try to estimate the expected long-term reward of a policy for each state $\boldsymbol{x}$ and time step t, which is called the *value function* $V_t^\pi(\boldsymbol{x})$. This value function assesses the quality of a specific action u in a state $\boldsymbol{x}$, and can then be used to directly compute the policy by action selection or to update the policy π.

However, this approach do not scale with high dimensional action and state spaces since it requires filling the complete state-action space with data. What is more, many tasks consider continuous action state space, and value functions do not easily extend to such settings.

In contrast to the value function approach, policy search methods employ a parametrized policy $\pi_{\boldsymbol{\theta}}$, and typically avoid learning a value function. Indeed, these methods directly operate in the parameter space $\boldsymbol{\theta}$ of parametrized policy. The usage of parametrized policies allows for scaling RL to high dimensional continuous action spaces by reducing the search space of possible policies.

In this review we distinguish policy search methods between model-free policy search methods, which learn policies directly based on sampled trajectories, and model-based approaches, which use the sampled trajectories to first build a model of the state dynamics, and, subsequently, use this model for policy improvement.

2 Model-free policy search

Model-free policy search approaches aim to update parameters $\boldsymbol{\theta}$ of a given parametrized policy $\pi_{\boldsymbol{\theta}}$, such that trajectories which have higher rewards become more likely to be observed by following the new policy. Equivalently, such methods learn parameters so as to increase the average return

$$J_{\boldsymbol{\theta}} = \mathbb{E}[R(\boldsymbol{\tau})|\boldsymbol{\theta}] = \int R(\boldsymbol{\tau}) p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) d\boldsymbol{\tau}$$

As suggested in [6], model-free policies search methods can be categorized by their exploration strategy, their policy evaluation strategy and their policy update strategy.

2.1 Exploration Strategies

The exploration strategy deals with generating new trajectory samples $\boldsymbol{\tau}^{[i]}$ which will then be used by the policy evaluation strategy and finally by the policy update strategy.

Different types of exploration can be distinguished, exploration in action space versus exploration in parameter space, and step-based versus episode-based exploration strategies.

Action space versus parameter space

The first exploration strategy considered is exploration in the action space by adding an (independant and zero-mean) Gaussian exploration noise ϵ_u to the executed actions: $\boldsymbol{u}_t = \mu(\boldsymbol{x}, t) + \epsilon_u$, with μ being a deterministic policy. Hence, we get the stochastic policy

$$\pi_{\boldsymbol{\theta}}(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{u}|\mu(\boldsymbol{x}, t), \Sigma_u)$$

Whereas exploration in parameter space is implemented by perturbing the parameter $\boldsymbol{\theta}$, at the beginning of an episode or at each time step: $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta} + \epsilon_t$.

Step-based versus episode-based

For step-based exploration strategies, a different exploration noise is used at each time step, thus exploration can be done either in action space or in parameter space.

Concerning episode-based exploration strategies, exploration noise is used only at the beginning of the episode which naturally leads to exploration in parameter space.

2.2 Policy Evaluation Strategies

The policy evaluation strategy assesses the quality of a given policy. Such a strategy can try to estimate the quality of a unique state-action pair $\boldsymbol{x}_t, \boldsymbol{u}_t$, called a step-based evaluation. It can also try to determine the quality of an all episode, thus the quality of the episode's parameter $\boldsymbol{\theta}$, referred to as episode-based policy evaluation.

Step-based In step-based evaluation, the quality of an action is given by the expected accumulated future reward when executing $\boldsymbol{u}_t^{[i]}$ in state $\boldsymbol{x}_t^{[i]}$ at time step t and then following policy $\pi_{\boldsymbol{\theta}}(u|\boldsymbol{x})$:

$$Q_t^{[i]} = Q_t^{\pi}(\boldsymbol{x}_t^{[i]}, \boldsymbol{u}_t^{[i]}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})}\left[\sum_{h=t}^{T} r_h(\boldsymbol{x}_h, \boldsymbol{u}_h) \mid \boldsymbol{x}_t = \boldsymbol{x}_t^{[i]}, \boldsymbol{u}_t = \boldsymbol{u}_t^{[i]}\right]$$

Episode-based In episode-based evaluation, the quality of a parametrized policy is defined as the expected return which is given by the sum of the future immediate rewards:

$$R(\boldsymbol{\theta}^{[i]}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})}\left[\sum_{t=0}^{T} r_t \mid \boldsymbol{\theta} = \boldsymbol{\theta}^{[i]}\right]$$

2.3 Policy Update Strategies

The policy update strategy is responsible for updating the parameter $\boldsymbol{\theta}$ given trajectories $\boldsymbol{\tau}^{[i]}$ and a policy evaluation strategy. In [6], authors organize algorithms which have been explored in the literature in the following families: policy gradient methods, expectation-maximization based methods and information theoretic methods.

2.3.1 Policy Gradient

Policy gradient methods make use of gradient-ascent in order to maximize the expected return $J_{\boldsymbol{\theta}}$. The policy gradient update is therefore given by

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha_k \nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}$$

with α_k a learning rate, and the policy gradient is by definition given by

$$\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}} = \int_{\boldsymbol{\tau}} \nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) R(\boldsymbol{\tau}) d\boldsymbol{\tau}$$

However $\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}$ cannot analytically be computed and thus need to be estimated.

Finite Differences The easiest way to do so is with the finite difference policy gradient method, introduced in [9] and [16], which applies a small perturbation $\delta\boldsymbol{\theta}$ to the parameter $\boldsymbol{\theta}_k$ and then observes a change of the return $\delta R^{[i]} = R(\boldsymbol{\theta}_k + \delta\boldsymbol{\theta}^{[i]}) - R(\boldsymbol{\theta}_k)$. Using a first-order Taylor approximation and solving $\nabla_{\boldsymbol{\theta}}^{FD} J_{\boldsymbol{\theta}}$ in the least-square sense yields

$$\nabla_{\boldsymbol{\theta}}^{\mathrm{FD}} J_{\boldsymbol{\theta}} = \left(\delta\boldsymbol{\theta}^T \delta\boldsymbol{\theta}\right)^{-1} \delta\boldsymbol{\theta}^T \delta R$$

with $\delta\boldsymbol{\theta} = \left[\delta\boldsymbol{\theta}^{[1]}, \ldots, \delta\boldsymbol{\theta}^{[N]}\right]^T$ and $\delta R = \left[\delta R^{[1]}, \ldots, \delta R^{[N]}\right]^T$.

Likelihood-Ratio Policy Gradients An other way of computing $\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}$, named likelihood-ratio methods, is to make use of the *likelihood ratio* trick $\nabla p_{\boldsymbol{\theta}}(y) = p_{\boldsymbol{\theta}}(y) \nabla \log p_{\boldsymbol{\theta}}(y)$. Indeed we then get

$$\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}} = \int_{\boldsymbol{\tau}} p_{\boldsymbol{\theta}}(y) \nabla \log p_{\boldsymbol{\theta}}(y) R(\boldsymbol{\tau}) d\boldsymbol{\tau} = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})}[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) R(\boldsymbol{\tau})] \simeq \frac{1}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}^{[i]}) R(\boldsymbol{\tau}^{[i]})$$

where sampled trajectories $\boldsymbol{\tau}^{[i]} = (\boldsymbol{x}_1^{[i]}, \boldsymbol{u}_1^{[i]}, \ldots, \boldsymbol{x}_N^{[i]}, \boldsymbol{u}_N^{[i]})$ are generated by the policy $\pi_{\boldsymbol{\theta}}$.

In a step-based spirit, the trajectory distribution can be written as $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) = p(\boldsymbol{x}_1) \prod_{t=1}^{T} p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t) \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_t|\boldsymbol{x}_t, t)$, thus $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau})$ can be decomposed into single time steps as $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) = \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_t|\boldsymbol{x}_t, t)$ since transitions are independent of $\boldsymbol{\theta}$.

Thus we get one of the first policy gradient algorithm introduced in [23]: REINFORCE, which uses the following policy gradient:

$$\nabla_{\boldsymbol{\theta}}^{\mathrm{RF}} J_{\boldsymbol{\theta}} = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})}[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_t|\boldsymbol{x}_t, t) R(\boldsymbol{\tau})] \simeq \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_t^{[i]}|\boldsymbol{x}_t^{[i]}, t) R(\boldsymbol{\tau}^{[i]})$$

Natural Policy Gradient This method has been proposed in [18] so as to achieve a more stable behaviour of the learning process, but still rely on previous policy gradient methods to compute

$\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}$. The idea is to maintain a limited step-width in the trajectory distribution space, between two subsequent distributions. This is enforced by the following constraint:

$$KL(p_{\boldsymbol{\theta}}(\boldsymbol{\tau})||p_{\boldsymbol{\theta}+\delta\boldsymbol{\theta}}(\boldsymbol{\tau})) \simeq \delta\boldsymbol{\theta}^T F_{\boldsymbol{\theta}} \delta\boldsymbol{\theta} \leq \epsilon$$

with $F_{\boldsymbol{\theta}} = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})}[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau})^T]$ being the Fisher transition matrix.

The natural gradient update $\delta\boldsymbol{\theta}^{NG}$ is then defined as the most similar 'vanilla' gradient update $\delta\boldsymbol{\theta}^{VG}$ that respect the preceding bounded distance. This can be formulated by the following optimization program

$$\delta\boldsymbol{\theta}^{NG} = \arg\max_{\delta\boldsymbol{\theta}} \delta\boldsymbol{\theta}^T \delta\boldsymbol{\theta}^{VG} \quad \text{s.t.} \quad \delta\boldsymbol{\theta}^T F_{\boldsymbol{\theta}} \delta\boldsymbol{\theta} \leq \epsilon$$

which solution is $\delta\boldsymbol{\theta}^{NG} \propto F_{\boldsymbol{\theta}}^{-1} \delta\boldsymbol{\theta}^{VG}$. The natural policy gradient $\nabla_{\boldsymbol{\theta}}^{NG} J_{\boldsymbol{\theta}}$ is therefore given by

$$\nabla_{\boldsymbol{\theta}}^{NG} J_{\boldsymbol{\theta}} = F_{\boldsymbol{\theta}}^{-1} \nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}$$

Guided Policy Search One major issue with policy gradient methods, such as likelihood-ratio methods, is that new trajectories $\boldsymbol{\tau}^{[i]}$ are required at each gradient step. Indeed, the gradient is estimated with $\mathbb{E}[\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}] \simeq \frac{1}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}^{[i]}) R(\boldsymbol{\tau}^{[i]})$. The importance sampling method allows using off-policy samples since $\mathbb{E}[J_{\boldsymbol{\theta}}] \simeq \frac{1}{Z(\boldsymbol{\theta})} \sum_{i=1}^{N} \frac{\pi_{\boldsymbol{\theta}}(\boldsymbol{\tau}^{[i]})}{q(\boldsymbol{\tau}^{[i]})} R(\boldsymbol{\tau}^{[i]})$, with $\boldsymbol{\tau}^{[i]} \sim q$. The distribution q can be a previous policy, or a guiding distribution constructed with differential dynamic programming (DDP) such as proposed in [10]. The idea of guided policy search is to supplement the sample set with off-policy guiding samples that guide the policy search to regions of high reward.

Given a reward function and a dynamic model, with the iterative LQR algorithm [22] (a variant of DDP), one can optimize a trajectory and yielding an optimal policy. If the dynamics and the reward functions are unknown, they can be estimated thanks to finite differences methods. The LQR algorithm uses a linear reward and quadratic dynamics approximations so as to estimates the Q-function, the value function and terms defining the optimal deterministic policy. Then a new trajectory is generated with this policy and the algorithm repeats those steps until the reward of sampled trajectories converges. Finally, a stochastic policy $q(\boldsymbol{\tau})$ can be constructed as a Gaussian distribution with its mean defined by the optimal deterministic policy.

2.3.2 Expectation Maximization

The main issue with policy gradient algorithms is the setting of the learning rate α which must be specified by the user and may lead to slow convergence or instabilities [8]. This issue can be avoided with the Expectation Maximization (EM) algorithm [11] which is well known for computing the maximum likelihood solution of a probabilistic model with latent variables.

Overview of Expectation Maximization In the general setting, we observe *iid* data $Y = [y^{[1]}, \ldots, y^{[N]}]^T$ and want to maximize the log-marginal-likelihood w.r.t. parameters $\boldsymbol{\theta}$

$$\log p_{\boldsymbol{\theta}}(Y) = \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(y^{[i]}) = \sum_{i=1}^{N} \log \int p_{\boldsymbol{\theta}}(y^{[i]}, z)$$

The idea is to introduce an auxiliary distribution $q(Z)$ to avoid the costly latent variable marginalization. We get

$$\log p_{\boldsymbol{\theta}}(Y) = \int q(Z) \log p_{\boldsymbol{\theta}}(Y) dZ = \mathcal{L}_{\boldsymbol{\theta}}(q) + KL(q(Z)||p_{\boldsymbol{\theta}}(Z|Y))$$

with $\mathcal{L}_{\boldsymbol{\theta}}(q) = \int q(Z) \log \frac{p_{\boldsymbol{\theta}}(Y,Z)}{q(Z)} dZ = \mathbb{E}_{Z\sim q}[\log \frac{p_{\boldsymbol{\theta}}(Y,Z)}{q(Z)}]$ being a lower bound of $\log p_{\boldsymbol{\theta}}(Y)$ since the KL divergence is nonnegative.

The EM algorithm alternates between two steps: the first one being the maximization of the lower bound (of $\log p_{\boldsymbol{\theta}}(Y)$) $\mathcal{L}$ (M-Step). During this step, we compute

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}} \mathcal{Q}_{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{Z\sim p(Z|Y)}[\log p_{\boldsymbol{\theta}}(Y, Z)]$$

where $\mathcal{Q}_{\boldsymbol{\theta}}$ is called the *complete likelihood*. The other step consists in minimizing the KL-divergence term (E-Step) which yields $q(Z) = p(Z|Y)$.

Expectation Maximization for policy search We will formulate policy search as an inference problem. First, we define a binary reward event R as our observed variable. The probability of this reward event is given by $p(R = 1|\boldsymbol{\tau})$, simplified as $p(R|\boldsymbol{\tau})$. Such a probability distribution is defined from a transformation of $R(\boldsymbol{\tau})$. The trajectory $\boldsymbol{\tau}$ plays the role of the latent variable. These relations are represented in a graphical model in Figure 1. We want to find the maximum solution $\boldsymbol{\theta}^\star$ for the log marginal-likelihood:

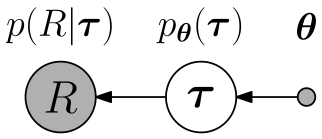

Figure 1: Graphical model for inference-based policy search.

$$\log p_{\boldsymbol{\theta}}(R) = \int_{\boldsymbol{\tau}} p(R|\boldsymbol{\tau}) p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) d\boldsymbol{\tau}$$

By introducing an auxiliary distribution $q(\boldsymbol{\tau})$, one can again decompose the log-marginal likelihood as

$$\log p_{\boldsymbol{\theta}}(R) = \mathcal{L}_{\boldsymbol{\theta}}(q) + KL(q(\boldsymbol{\tau})||p_{\boldsymbol{\theta}}(\boldsymbol{\tau}|R))$$

For most common policies, the M-Step yields closed form solution for parameters. However, the E-Step cannot be computed exactly, and approximations must be used such as Monte-Carlo approaches [8, 17] or variational methods. [13].

Monte-Carlo EM-based Policy Search Monte-Carlo Expectation-Maximization (MC-EM) methods use a sample based approximation for the auxiliary distribution q. They use trajectories $\boldsymbol{\tau}^{[i]}$ sampled from the old distribution $p_{\boldsymbol{\theta}'}(\boldsymbol{\tau})$, $\boldsymbol{\theta}'$ being the old parameters, so as to represent the auxiliary distribution $q(\boldsymbol{\tau}) \propto p(R|\boldsymbol{\tau}) p_{\boldsymbol{\theta}'}(\boldsymbol{\tau})$. Since $\boldsymbol{\tau}^{[i]}$ is already sampled from $p_{\boldsymbol{\theta}'}(\boldsymbol{\tau})$, we get $q(\boldsymbol{\tau}^{[i]}) \propto p(R|\boldsymbol{\tau}^{[i]})$.

Then, the expectation of the complete data log-likelihood can be estimated with the same samples:

$$\mathcal{Q}_{\boldsymbol{\theta}}(\boldsymbol{\theta}') \simeq \sum_{\boldsymbol{\tau}^{[i]} \sim p_{\boldsymbol{\theta}'}(\boldsymbol{\tau})} p(R|\boldsymbol{\tau}^{[i]}) \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}^{[i]})$$

Variational Inference-based Policy Search MC-EM methods use a weighted maximum likelihood estimate, which can efficiently be computed. Yet, these methods may average over several modes of the reward function and thus yield low rewards.

On the other hand, variational inference-based methods use a parametrized distribution q_β, instead of a sample-based approximation to approximate the auxiliary distribution q. The idea is to choose parameters of q_β so as to minimize the distance between q and q_β. Hence choose β such as

$$\beta \in \arg\min_\beta KL(q_\beta(\boldsymbol{\tau})||p(R|\boldsymbol{\tau}) p_{\boldsymbol{\theta}}(\boldsymbol{\tau})) \simeq \arg\min_\beta \sum_{\boldsymbol{\tau}^{[i]}} q_\beta(\boldsymbol{\tau}^{[i]}) \log \frac{q_\beta(\boldsymbol{\tau}^{[i]})}{p(R|\boldsymbol{\tau}^{[i]}) p_{\boldsymbol{\theta}}(\boldsymbol{\tau}^{[i]})}$$

2.3.3 Information Theoretical Approaches

The general idea of information theoretical approaches is that the trajectory distribution after the policy update should not be far from the trajectory distribution before the policy update. This is enforced by bounding the distance between the old trajectory distribution $q(\boldsymbol{\tau})$ and the newly estimated trajectory distribution $p(\boldsymbol{\tau})$ (in the Kullback-Leibler divergence sense). Such a regularization avoids that the new distribution $p(\boldsymbol{\tau})$ to prematurely concentrates on local optima of the reward landscape. It can be seen as limiting the information loss of the updates.

Natural policy gradient algorithms [18] were the first algorithms to implement these ideas. Unfortunately as policy gradient algorithms, they require a user defined learning rate. The Relative Entropy Policy Search (REPS) algorithm [15] combines both advantages of EM and natural gradient policy.

(Episode-based) Relative Entropy Policy Search In its episode-based formulation, REPS learns an upper-level policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta})$ which selects the parameters of the (lower-level) policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{u}|\boldsymbol{x})$. It

aims at maximizing the average return $J_{\boldsymbol{\omega}} = \int_{\boldsymbol{\theta}} \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) \int_{\boldsymbol{\tau}} p(\boldsymbol{\tau}|\boldsymbol{\theta}) R(\boldsymbol{\tau}) d\boldsymbol{\tau} d\boldsymbol{\theta} = \int_{\boldsymbol{\theta}} \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) R(\boldsymbol{\theta}) d\boldsymbol{\theta}$. The policy update is thus formulated as the following optimization problem:

$$\begin{aligned} \underset{\pi}{\text{maximize}} \quad & \int \pi(\boldsymbol{\theta}) R(\boldsymbol{\theta}) d\boldsymbol{\theta} \\ \text{subject to} \quad & \pi(\boldsymbol{\theta}) \log \frac{\pi(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} \leq \epsilon \\ & \int \pi(\boldsymbol{\theta}) d\boldsymbol{\theta} = 1 \end{aligned}$$

An closed-form solution for the new policy can be obtained via the Lagrangian and is given by $\pi(\boldsymbol{\theta}) \propto q(\boldsymbol{\theta}) \exp(\frac{R(\boldsymbol{\theta})}{\eta})$, with η being the Lagrangian multiplier associated with the KL bound. The parameter η is thus obtained by maximizing the dual function

$$g(\eta) = \eta\epsilon + \eta \log \int q(\boldsymbol{\theta}) \exp\left(\frac{R(\boldsymbol{\theta})}{\eta}\right) d\boldsymbol{\theta} \simeq \eta\epsilon + \eta \log \sum_{i=1}^{N} \frac{1}{N} \exp\left(\frac{R(\boldsymbol{\theta}^{[i]})}{\eta}\right)$$

The new policy $\pi(\boldsymbol{\theta})$ is only known for samples $\boldsymbol{\theta}^{[i]}$ where $R(\boldsymbol{\theta}^{[i]})$ has been evaluated. Thus a parametric distribution $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta})$ needs to be fitted to these samples.

3 Model-based Policy search

In computer simulation where the dynamic of the system is known, it is relatively straightforward to sample trajectories. However, when working with dynamical systems, the real dynamic is unknown. The policy learning process requires a large amount of manually generated trajectories. Depending on the task studied it can therefore be, either easier to learn the policy directly or to learn the model and use computer simulations to learn a policy.

Model based methods aim at solving the problem of sample inefficiency. The general idea is to use the observed data to learn the forward model of a system dynamic and use this model to learn a policy. In most cases, we assume that the state x evolves according to the Markovian dynamics :

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{w}$$

where f is a non linear function, $\boldsymbol{u}$ is an action and $\boldsymbol{w}$ is additive noise. We will also consider finite horizons problems. This means that the objective of the policy search is to find:

where r is an immediate reward, γ a discount factor, and the policy is parametrized by θ. For some problems, model-based methods require fewer real measurements than the model free model because of an efficient use of the model learnt. This consideration justifies the interest if this approach.

The idea of RL based methods is to learn the model and the policy simultaneously as followed :

- The learned model will be use for internal simulations. This will provide a set of predictions of how the system would behave with the environment if it followed the current policy.
- Considering a given model and the simulations use evaluation and improvement of policy processes until optimal policy is learned for the model.
- Using the new policy, new data are recorded to complete the dataset and improve the model of the dynamics.

In this process, real measurements are only performed after the new policy has been computed. On the other hand, internal simulations and policy learning only use simulations. The quality of the learned policy therefore strongly relies on the quality of the learned forward model, the core of this subject will be focusing on the model building. When the model exactly corresponds to the true dynamics on the system, sampling from it is equivalent to sampling from the model.

In practice the learned model is not exact and this difference between the real dynamics and the model can lead to errors or even non real behaviours (negative mass for a dynamic system for example) in region with sparse training. The uncertainty of RL approaches due to inaccuracy of the knowledge of the dynamic of the system have been studied in robotics namely in [1, 3, 21]. This is why instead of considering only a model f, it's relevant to consider also the uncertainty we have in our estimation. As exposed in [6] the challenges of model prediction are often split into three categories :

- What model to learn?
- How to use the model for long-term predictions?
- How to update the policy based on the long-term predictions?

3.1 Probabilistic forward Models

3.1.1 Locally Weighted Bayesian Regression

The idea of the *locally weighted linear regression* (LWR) introduced in [5] and used in [3] is to take advantage of the good properties of the Linear Regression but to allow a more general class of functions : the locally linear approximation functions. Locally, the dynamic of the system will be ruled as follows :

$$\boldsymbol{x}_{t+1} = [\boldsymbol{x}_t, \boldsymbol{u}_t]^T \psi + \boldsymbol{w}$$

where ψ is the parameter of the Bayesian regression model and w is an iid Gaussian system noise.

In this approach, as in Bayesian linear regression, a posterior distribution over the parameters, ψ is computed for each query point ($\boldsymbol{x}_t$, $\boldsymbol{u}_t$) using Bayes' theorem.

$$p(\boldsymbol{\psi}|X, U, y) \quad \propto \quad p(\boldsymbol{\psi})p(y|X, U, \boldsymbol{\psi})$$

Let us assume a known noise covariance matrix Σ_w and a zero-mean prior Gaussian distribution $\mathcal{N}(\psi|0, S)$ on the parameters $\boldsymbol{\psi}$. The posterior mean and covariance of ψ can then be computed considering the similarity between the training inputs and the modelled ones. Let $(b_1, ...b_n)$ be the similarities between the n query points and their modelled values. The advantage of this Bayesian approach compared to a fully deterministic one is that it provides information on the uncertainty of the model and the confidence at each query point ($\boldsymbol{x}_t$, $\boldsymbol{u}_t$). The posterior mean and covariance are then given by :

$$\mathbb{E}[\boldsymbol{\psi}|\widetilde{X}, \boldsymbol{y}] = S\widetilde{X}B\boldsymbol{\Omega}^{-1}y$$
$$\text{cov}(\boldsymbol{\psi}|\widetilde{X}, \boldsymbol{y}) = S - S^T\widetilde{X}B\boldsymbol{\Omega}^{-1}B\widetilde{X}^TS$$

where $\widetilde{X} = [X, U]$, $\boldsymbol{\Omega} = B\widetilde{X}^TS\widetilde{X}B + \boldsymbol{\Sigma}_w$ and $B = diag(b_i)$. The predictive distribution can then be computed :

$$\boldsymbol{\mu}_{\boldsymbol{x}}^{t+1} = [\boldsymbol{x}_t, \boldsymbol{u}_t]^T\mathbb{E}[\boldsymbol{\psi}|\widetilde{X}, \boldsymbol{y}]$$
$$\boldsymbol{\Sigma}_{\boldsymbol{x}}^{t+1} = [\boldsymbol{x}_t, \boldsymbol{u}_t]^T\text{cov}[\psi|\widetilde{X}, \boldsymbol{y}][\boldsymbol{x}_t, \boldsymbol{u}_t]$$

3.1.2 Gaussian Process Regression

A Gaussian process is a distribution ρ(f) over functions f. Formally, it is a collection of variables such as any finite numbers of them are Gaussian distributed. It is a non parametric model and can specify high level assumptions such as differentiability or periodicity.

A GP is fully described by a mean function m(.) and a positive semi-definite covariance function k. A standard assumption [19] that can be made is to consider a zero-mean prior function and a covariance function :

$$k(\widetilde{\boldsymbol{x}}_p, \widetilde{\boldsymbol{x}}_q) = \sigma_f^2\text{exp}(-\frac{1}{2}(\widetilde{\boldsymbol{x}}_p - \widetilde{\boldsymbol{x}}_q)^T\Lambda^{-1}(\widetilde{\boldsymbol{x}}_p - \widetilde{\boldsymbol{x}}_q)) + \delta_{pq}\sigma_w^2$$

where $\widetilde{\boldsymbol{x}} = \left[\boldsymbol{x}^T, \boldsymbol{u}^T\right]^T$, $\Lambda = diag(l_1^2, ..., l_D^2)$ which depends on the characteristic length-scales, and σ_f^2 which is the prior variance of the latent function *f*. The posterior GP hyper-parameters (l_i, σ_f and σ_w) are learned using evidence maximisation[19] on a training input set $[\widetilde{\mathbf{x}}_1, ..., \widetilde{\mathbf{x}}_d]$ and the corresponding training targets : $[y_1, ...y_n]$.

The GP is a one step prediction model and the predicted successor state $\boldsymbol{x}_{t+1}$ is Gaussian distributed :

$$p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{x}_{t+1}|\boldsymbol{\mu}^x_{t+1}, \boldsymbol{\Sigma}^x_{t+1})$$

where

$$\boldsymbol{\mu}^x_{t+1} = \mathbb{E}_f[f(\boldsymbol{x}_t, \boldsymbol{u}_t)] = \boldsymbol{k}_*^T \boldsymbol{K}^{-1} \boldsymbol{y}$$

$$\boldsymbol{\Sigma}^x_{t+1} = var_f[f(\boldsymbol{x}_t, \boldsymbol{u}_t)] = k_{**} - \boldsymbol{k}_*^T \boldsymbol{K}^{-1} \boldsymbol{k}_*$$

with $\boldsymbol{k_*} := k(\widetilde{\boldsymbol{X}}, \widetilde{\boldsymbol{x}}_t), k_{**} = k(\widetilde{\boldsymbol{x}}_t, \widetilde{\boldsymbol{x}}_t)$ and $\boldsymbol{K}$ is the kernel matrix with entries $\boldsymbol{K}_{ij} = k(\widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{x}}_j)$.

4 Long term Predictions with a given Model

Considering that we have a model, we have to build long term predictions in order to learn the optimal policy. At this point, we are trying to estimate the quality of a given policy knowing a model of the dynamics of the system :

To do so, two different approaches can be considered : one based on Monte-Carlo sampling and a deterministic approximate inference.

4.1 Approach based on Monte-Carlo sampling

Using a Monte-Carlo approximation method, this will require many sample trajectories. PEGASUS (Policy Evaluation-of-Goodness And Search Using Scenarios) [14] is a conceptual framework for sampling in stochastic MDPs. The key idea is to turn stochastic MDP in an augmented deterministic one. To do so, a sequence of random values $w_0, w_1...$ is taken at the beginning of the algorithm. The state x is augmented by those values so as to add the given noise to a parameter of the MDP. As the noise now belongs to the state, the problem virtually becomes deterministic. Several deterministic MDP can now be performed.

Informally PEGASUS can be seen as generating M Monte-Carlo trajectories and taking their average reward. The difference stems from the fact that the randomisation is determined in advance. It is a way of reducing drastically the random variance of the evaluation.

4.2 Approach based on deterministic long-term predictions

Instead of performing a stochastic sampling over trajectories in order to use a Monte-Carlo approach, the deterministic long-term predictions will compute a distribution $\rho(\boldsymbol{\tau})$ over trajectories. This distribution could then be used to estimate the quality of a given policy. Assuming a Gaussian joint distribution : p($\boldsymbol{x}_t$,$\boldsymbol{u}_t$))= $\mathcal{N}([\boldsymbol{x}_t, \boldsymbol{u}_t]\,|\boldsymbol{\mu}^{xu}_t, \boldsymbol{\Sigma}^{xu}_t)$, the problem corresponding to finding the successor state distribution corresponds to solving the integral

$$p(\boldsymbol{x}_{t+1}) = \iiint p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t) p(\boldsymbol{x}_t, \boldsymbol{u}_t) d\boldsymbol{x}_t d\boldsymbol{u}_t d\boldsymbol{w}_t$$

where $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{w}$ with f a non parametric functions. The problem is that is f is non linear, $p(\boldsymbol{x}_{t+1})$ is non Gaussian and therefore needs to be approximated. A convenient approximation of $p(\boldsymbol{x}_{t+1})$ is the Gaussian $\mathcal{N}(\boldsymbol{x}_{t+1}|\boldsymbol{\mu}^x_{t+1}, \boldsymbol{\Sigma}^x_{t+1})$. There are several ways of estimating $\boldsymbol{\mu}^x_{t+1}$ and $\boldsymbol{\Sigma}^x_{t+1}$. It can for instance be performed thanks to *linearization*[2], *sigma-point* methods[7] or *moment matching*[7].

The idea of **linearization** is to locally approximate the transition function f around $(\boldsymbol{\mu}^x_t, \boldsymbol{\mu}^u_t)$. Although, it's a straightforward and efficient method, it requires f to be differentiable. It can also severely underestimate the predictive variance and cause issues in the final results.

The **sigma points** approach considers that the distribution of $p(\boldsymbol{x}_t, \boldsymbol{u}_t)$ can be represented by a set of deterministically chosen sigma points chosen samples from the joint distribution of (X_t, U_t). For these points, the value of $f(\boldsymbol{x}_t, \boldsymbol{u}_t)$ can be computed. The mean and the covariance of the distribution of $p(x_{t+1})$ can be then be estimated from the weighted sigma points previously chosen.

The idea of **moment matching** is to compute the predictive mean and covariance of $p(\boldsymbol{x}_{t+1})$ exactly and to approximate $p(\boldsymbol{x}_{t+1})$ by a Gaussian that possesses the exact mean and covariance. This method does not need any approximation of the joint distribution $p(\boldsymbol{x}_t, \boldsymbol{u}_t)$ nor the transition function f. It is the best unimodal approximation of the distribution in the sense that it minimizes the Kullback-Leibler between the true predictive distribution and the unimodal approximation[4]. The problem of this method stems from the fact that the computation of the mean and the covariance might be intractable and in general cases this method is computationally more complex than the two previous ones.

5 Policy Updates

The final part of model based policy learning consists in using model previously built and the policy to update it. To proceed to this update, we could use approaches only using the policy evaluation but also approaches using it gradient.

5.1 Model-based Policy Updates without Gradient Information

For this approach, there is no need to estimate the policy gradient. Therefore, we only need to estimate the value of a given policy. Simple and efficient methods such as Nelder-Mead (a variation of the simplex method)[12] or hill-climbing methods (similar to the simulated annealing)[20] can be used. The simplicity and computational efficiency has made this methods commonly used in the context of model-based policy search. Unfortunately the convergence rate of this methods is very slow and this justifies the interest we have for policy updates with Gradient Information.

5.2 Model-based Policy Updates with Gradient Information

Those approach are expected to yield a faster convergence rate than the previous one. However, as the policy gradient is unknown, it has to be estimated. There are two ways to do so : a sample-based estimation of the policy gradients and an analytic computation of the policy gradient $dJ_\theta(\theta)/d_\theta$

5.2.1 Sampling based method

The idea of this method is to take advantage of the fact that when we sample trajectories to estimate the long term reward J_θ, we can also approximate the gradient $dJ_\theta(\theta)/d_\theta$. This estimation can be performed via *finite difference* method. However, it requires $O(F)$ evaluations to be efficient (where F is a number of policy parameters).

5.2.2 Analytic policy gradient

Under the hypothesis that the policy, the reward function and the learned transition model are differentiable, an other approach consists in an analytic gradient computation. It has the advantage of not suffering from the sampling variance. Moreover, on complex policies that can have thousands of parameters, a sampling approach would be inefficient as mentioned already.

Deterministic model:
Let us consider the example where the model is deterministic and relies on a non parametric transition function f such that : $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t) = f(\boldsymbol{x}_t, \pi_\theta(\boldsymbol{x}_t, \boldsymbol{\theta}))$. We are trying to estimate the gradient of $J_\theta = \sum_t \gamma^t r(\boldsymbol{x}_t)$. Let $\boldsymbol{\theta}$ be the parameters of the given policy.

$$\frac{\mathrm{d}J_\theta}{\mathrm{d}\boldsymbol{\theta}} = \sum_t \gamma^t \mathrm{d}r(\boldsymbol{x}_t) = \sum_t \gamma^t \frac{\partial r(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t} \frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}\boldsymbol{\theta}}$$

Using the chain-rule we find that:

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}\boldsymbol{\theta}} = \frac{\mathrm{d}\boldsymbol{x}_{t-1}}{\mathrm{d}\theta} \frac{\partial \boldsymbol{x}_t}{\boldsymbol{x}_{t-1}} + \frac{\mathrm{d}\boldsymbol{u}_{t-1}}{\mathrm{d}\theta} \frac{\partial \boldsymbol{x}_t}{\boldsymbol{u}_{t-1}}$$

Observing that the total derivative $\frac{\mathrm{d}\boldsymbol{x}t}{\mathrm{d}\theta}$ depends on the total derivative $\frac{\mathrm{d}\boldsymbol{x}_{t-1}}{\mathrm{d}\theta}$, we conclude that the gradient can be computed iteratively.

Stochastic model:

If we consider that the state $\boldsymbol{x_t}$ is represented by a probability distribution $p(\boldsymbol{x_t})$, we have to compute the expected reward $\mathbb{E}[r(\boldsymbol{x}_t)]$. We also need to compute the derivative of the distribution p. Let's consider that $\boldsymbol{x_t}$ is Gaussian distributed as stated $p(\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{\mu}_t^x, \boldsymbol{\Sigma}_t^x)$. As well as before, we can prove using the chain rule that $\boldsymbol{\mu}_t^x$ and $\boldsymbol{\Sigma}_t^x$ can be iteratively computed as previously using $\boldsymbol{\mu}_{t-1}^x$ and $\boldsymbol{\Sigma}_{t-1}^x$. As this method relies on the knowledge of $p(\boldsymbol{x_t})$, they have to be used with approximate inference methods (seen previously). This analytic methods has this advantage of of using the exact gradient of the approached policy evaluation. The approximation errors therefore only stems from the error made in approximating J_θ.

6 Conclusion

We have introduced two ways of learning an optimal policy where the model is unknown : model-free and model-based approaches. Both of this methods take a different point of view and come with advantages and disadvantages. The choice of a given approach therefore strongly relies on the problem at hand.

In the literature, model free approaches remain more common as they avoid the difficulty of modelling the system. However, the policies that can be computed are often found using a local research and therefore risk being local optimal policies. Moreover, this method requires that the number of parameters of the policy is not too important (less than 100 parameters) and demands a lot of experimental data (human intervention).

On the other hand, model-based approaches have the advantage of estimating a model of the system's dynamics. This model can then be used to run computer simulations. This approach, theoretically, needs less human intervention than the previous one. However, it considers that the model is relatively easy to learn compared with the optimal policy. This method strongly relies on the quality of the modelling in the sense that errors made in the model can directly lead to massive errors in the optimal policy found.

References

[1] P. ABBEEL, M. QUIGLEY, AND A. Y. NG, *Using inaccurate models in reinforcement learning*, in Proceedings of the 23rd international conference on Machine learning, ACM, 2006, pp. 1–8.

[2] B. ANDERSON, *Jb moore, optimal filtering*, Eaglewood Cliffs, NJ: Prentice-Hall, (1979).

[3] J. A. BAGNELL AND J. G. SCHNEIDER, *Autonomous helicopter control using reinforcement learning policy search methods*, in Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on, vol. 2, IEEE, 2001, pp. 1615–1620.

[4] C. BISHOP, *Bishop pattern recognition and machine learning*, 2001.

[5] W. S. CLEVELAND AND S. J. DEVLIN, *Locally weighted regression: an approach to regression analysis by local fitting*, Journal of the American Statistical Association, 83 (1988), pp. 596–610.

[6] M. P. DEISENROTH, G. NEUMANN, AND J. PETERS, *A survey on policy search for robotics*, Foundations and Trends in Robotics, 2 (2013), pp. 1–142.

[7] S. J. JULIER AND J. K. UHLMANN, *Unscented filtering and nonlinear estimation*, Proceedings of the IEEE, 92 (2004), pp. 401–422.

[8] J. KOBER AND J. PETERS, *Policy search for motor primitives in robotics*, Mach. Learn., 84 (2011), pp. 171–203.

[9] N. KOHL AND P. STONE, *Policy gradient reinforcement learning for fast quadrupedal locomotion*, in Proceedings of the IEEE International Conference on Robotics and Automation, May 2004.

[10] S. LEVINE AND V. KOLTUN, *Guided policy search*, in ICML '13: Proceedings of the 30th International Conference on Machine Learning, 2013.

[11] R. NEAL AND G. HINTON, *A view of the EM algorithm that justifies incremental, sparse, and other variants*, Learning in graphical models, 89 (1998), pp. 355–368.

[12] J. A. NELDER AND R. MEAD, *A simplex method for function minimization*, The computer journal, 7 (1965), pp. 308–313.

[13] G. NEUMANN, *Variational inference for policy search in changing situations*, in Proceedings of the International Conference on Machine Learning (ICML 2011), 2011.

[14] A. Y. NG AND M. JORDAN, *Pegasus: A policy search method for large mdps and pomdps*, in Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc., 2000, pp. 406–415.

[15] J. PETERS, K. MÜLLING, AND Y. ALTÜN, *Relative entropy policy search*, in Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010), M. Fox and D. Poole, eds., AAAI Press, 2010, pp. 1607–1612.

[16] J. PETERS AND S. SCHAAL, *Policy gradient methods for robotics*, in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Beijing, China, 2006.

[17] ——, *Applying the episodic natural actor-critic architecture to motor primitive learning*, in Proceedings of the 2007 European Symposium on Artificial Neural Networks (ESANN), 2007.

[18] ——, *Reinforcement learning of motor skills with policy gradients*, Neural Networks, 21 (2008), pp. 682–697.

[19] C. E. RASMUSSEN, *Gaussian processes for machine learning*, Citeseer, 2006.

[20] S. J. RUSSELL, P. NORVIG, J. F. CANNY, J. M. MALIK, AND D. D. EDWARDS, *Artificial intelligence: a modern approach*, vol. 2, Prentice hall Upper Saddle River, 2003.

[21] J. G. SCHNEIDER, *Exploiting model uncertainty estimates for safe dynamic control learning*, Advances in neural information processing systems, (1997), pp. 1047–1053.

[22] Y. TASSA, T. EREZ, AND E. TODOROV, *Synthesis and stabilization of complex behaviors through online trajectory optimization.*, in IROS, IEEE, 2012, pp. 4906–4913.

[23] R. J. WILLIAMS, *Simple statistical gradient-following algorithms for connectionist reinforcement learning*, Machine Learning, 8 (1992), pp. 229–256.