Policy gradient methods

Jan Peters (2010), Scholarpedia, 5(11):3698. doi:10.4249/scholarpedia.3698 revision #137199 [link to/cite this article]

- **Prof. Jan Peters**, Max-Planck Institute, Germany & University of Southern California, USC

Policy gradient methods are a type of reinforcement learning techniques that rely upon optimizing parametrized policies with respect to the expected return (long-term cumulative reward) by gradient descent. They do not suffer from many of the problems that have been marring traditional reinforcement learning approaches such as the lack of guarantees of a value function, the intractability problem resulting from uncertain state information and the complexity arising from continuous states & actions.

Contents

Introduction

Reinforcement learning is probably the most general framework in which reward-related learning problems of animals, humans or machine can be phrased. However, most of the methods proposed in the reinforcement learning community are not yet applicable to many problems such as robotics, motor control, etc. This inapplicability may result from problems with uncertain state information. Thus, those systems need to be modeled as partially observable Markov decision problems which often results in excessive computational demands. Most traditional reinforcement learning methods have no convergence guarantees and there exist even divergence examples. Continuous states and actions in high dimensional spaces cannot be treated by most off-the-shelf reinforcement learning approaches.

Policy gradient methods differ significantly as they do not suffer from these problems in the same way. For example, uncertainty in the state might degrade the performance of the policy (if no additional state estimator is being used) but the optimization techniques for the policy do not need to be changed. Continuous states and actions can be dealt with in exactly the same way as discrete ones while, in addition, the learning performance is often increased. Convergence at least to a local optimum is guaranteed.

The advantages of policy gradient methods for real world applications are numerous. Among the most important ones are that the policy representations can be chosen so that it is meaningful for the task and can incorporate domain knowledge, that often fewer parameters are needed in the learning process than in value-function based approaches and that there is a variety of different algorithms for policy gradient estimation in the literature which have a rather strong theoretical underpinning. Additionally, policy gradient methods can be used either model-free or model-based as they are a generic formulation.

Of course, policy gradients are not the salvation to all problems but also have significant problems. They are by definition on-policy (note that tricks like importance sampling can slightly alleviate this problem) and need to forget data very fast in order to avoid the introduction of a bias to the gradient estimator. Hence, the use of

sampled data is not very efficient. In tabular representations, value function methods are guaranteed to converge to a global maximum while policy gradients only converge to a local maximum and there may be many maxima in discrete problems. Policy gradient methods are often quite demanding to apply, mainly because one has to have considerable knowledge about the system one wants to control to make reasonable policy definitions. Finally, policy gradient methods always have an open parameter, the learning rate, which may decide over the order of magnitude of the speed of convergence, these have led to new approaches inspired by expectation-maximization (see, e.g., Vlassis et al., 2009; Kober & Peters, 2008).

Nevertheless, due to their advantages stated above, policy gradient methods have become particularly interesting for robotics applications as these have both continuous actions and states. For example, there has been a series of successful applications in robot locomotion, where good policy parametrizations such as CPGs are known. Benbrahim & Franklin (1997) already explored 2D dynamic biped walking, Tedrake et al. (2004) extended these results to 3D passive dynamics-based walkers and Endo (2005) showed that a full-body gait with sensory feedback can be learned with policy gradients. Kohl & Stone (2004) were able to apply policy gradients to optimize quadruped gaits. There have also been various applications in skill learning starting with the peg-in-a-hole tasks learned by Gullapalli et al. (1994) and ranging to Peters & Schaals' optimizations of discrete movements primitives such as T-Ball swings.

Note that in most applications, there exist many local maxima; for example, if we were told build a high jumping robot, there is a multitude of styles. Current policy gradient methods would be helpful for improving a jumping style of a teacher, let's say the classical straddle jump. However, discovering a Fosbery flop when starting with a basic straddle jump policy is probably not possible with policy gradient methods.

Assumptions and Notation

We assume that we can model the control system in a discrete-time manner and we will denote the current time step by k . In order to take possible stochasticity of the plant into account, we denote it using a probability distribution $\mathbf{x}_{k+1} \sim p\left(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k\right)$ as model where $\mathbf{u}_k \in \mathbb{R}^M$ denotes the current action, and $\mathbf{x}_k$, $\mathbf{x}_{k+1} \in \mathbb{R}^N$ denote the current and next state, respectively. We furthermore assume that actions are generated by a policy $\mathbf{u}_k \sim \pi_\theta(\mathbf{u}_k|\mathbf{x}_k)$ which is modeled as a probability distribution in order to incorporate exploratory actions; for some special problems, the optimal solution to a control problem is actually a stochastic controller (Sutton, McAllester, Singh, and Mansour, 2000). The policy is assumed to be parameterized by K policy parameters $\theta \in \mathbb{R}^K$ .The sequence of states and actions forms a trajectory denoted by $\tau = [\mathbf{x}_{0:H}, \mathbf{u}_{0:H}]$ where H denotes the horizon which can be infinite. In this article, we will use the words trajectory, history, trial, or roll-out interchangeably. At each instant of time, the learning system receives a reward denoted by $r_k = r(\mathbf{x}_k, \mathbf{u}_k) \in \mathbb{R}$.

The general goal of policy optimization in reinforcement learning is to optimize the policy parameters $\theta \in \mathbb{R}^K$ so that the expected return

$$J\left(\theta\right) = E\left\{ \sum_{k=0}^{H} a_k r_k \right\}$$

is optimized where a_k denote time-step dependent weighting factors, often set to $a_k = \gamma^k$ for discounted reinforcement learning (where γ is in $[0, 1]$) or $a_k = 1/H$ for the average reward case.

For real-world applications, we require that any change to the policy parameterization has to be smooth as drastic changes can be hazardous for the actor as well as useful initializations of the policy based on domain knowledge would otherwise vanish after a single update step. For these reasons, policy gradient methods which follow the steepest descent on the expected return are the method of choice. These methods update the policy parameterization according to the gradient update rule

$$\theta_{h+1} = \theta_h + \alpha_h \nabla_\theta J|_{\theta=\theta_h},$$

where $\alpha_h \in \mathbb{R}^+$ denotes a learning rate and $h \in \{0, 1, 2, \ldots\}$ the current update number.

The time step k and update number h are two different variables. In actor-critic-based policy gradient methods, the frequency of updates of h can be nearly as high as of k . However, in most episodic methods, the policy update h will be significantly less frequent. Here, cut-off allows updates before the end of the episode (for $a_k = \gamma^k$, it is obvious that there comes a point where any future reward becomes irrelevant; a generically good cut-off point). If the gradient estimate is unbiased and learning rates fulfill $\sum_{h=0}^{\infty} \alpha_h > 0$ and $\sum_{h=0}^{\infty} \alpha_h^2 = \text{const}$, the learning process is guaranteed to converge at least to a local minimum.

The main problem in policy gradient methods is to obtain a good estimator of the policy gradient $\nabla_\theta J|_{\theta=\theta_h}$. In robotics and control, people have traditionally used deterministic model-based methods for obtaining the gradient (Jacobson & Mayne, 1970; Dyer & McReynolds, 1970; Hasdorff, 1976). However, in order to become autonomous we cannot expect to be able to model every detail of the system. Therefore, we need to estimate the policy gradient simply from data generated during the execution of a task, i.e., without the need for a model. In this section, we will study different approaches and discuss their properties.

Approaches to Policy Gradient Estimation

The literature on policy gradient methods has yielded a variety of estimation methods over the last years. The most prominent approaches, which have been applied to robotics are finite-difference and likelihood ratio methods, better known as REINFORCE in reinforcement learning.

Finite-difference Methods

Finite-difference methods are among the oldest policy gradient approaches; they originated from the stochastic simulation community and are quite straightforward to understand. The policy parameterization is varied I times by small increments $\Delta\theta_i,\ i = 1 \dots I$ and for each policy parameter variation $\theta_h + \Delta\theta_i$ roll-outs (or trajectories) are performed which generate estimates $\Delta\hat{J}_i \approx J(\theta_h + \Delta\theta_i) - J_{\text{ref}}$ of the expected return. There are different ways of choosing the reference value J_{ref} , e.g. forward-difference estimators with $J_{\text{ref}} = J(\theta_h)$ and central-difference estimators with $J_{\text{ref}} = J(\theta_h - \Delta\theta_i)$. The policy gradient estimate $\mathbf{g}_{\text{FD}} \approx \nabla_\theta J|_{\theta=\theta_h}$ can be estimated by regression yielding

$$\mathbf{g}_{\text{FD}} = \left(\Delta\boldsymbol{\Theta}^T\Delta\boldsymbol{\Theta}\right)^{-1}\Delta\boldsymbol{\Theta}^T\Delta\hat{\mathbf{J}},$$

where $\Delta\boldsymbol{\Theta} = [\Delta\theta_1, \dots, \Delta\theta_I]^T$ and $\Delta\hat{\mathbf{J}} = [\Delta\hat{J}_1, \dots, \Delta\hat{J}_I]^T$ denote the I samples. This approach can be highly efficient in simulation optimization of deterministic systems (Spall, 2003) or when a common history of random numbers (Glynn, 1987) is being used (the latter is known as PEGASUS in reinforcement learning (Ng & Jordan, 2000)), and the error of the gradient estimate can get close to $O\left(I^{-1/2}\right)$ (Glynn, 1987). However, the uncertainty of real systems will result in stochasticity and an artificial common history of random numbers can no longer be applied. Hence, when used on a real system, the performance degrades in a gradient estimate error ranging between $O\left(I^{-1/4}\right)$ to $O\left(I^{-2/5}\right)$ depending on the chosen reference value (Glynn, 1987). An implementation of this algorithm is shown below.

input: policy parameterization θ_h
for $i = 1$ **to** I **do**
    generate policy variation $\Delta\theta_i$
    estimate $\hat{J}_i \approx J(\theta_h + \Delta\theta_i) = \left\langle \sum_{k=0}^{H} a_k r_k \right\rangle$ from roll-out
    estimate $\hat{J}_{\text{ref}}$, e.g., $\hat{J}_{\text{ref}} = J(\theta_h - \Delta\theta_i)$ from roll-out
    compute $\Delta\hat{J}_i \approx J(\theta_h + \Delta\theta_i) - J_{\text{ref}}$
end for
return gradient estimate $\mathbf{g}_{\text{FD}} = \left(\Delta\boldsymbol{\Theta}^T\Delta\boldsymbol{\Theta}\right)^{-1}\Delta\boldsymbol{\Theta}^T\Delta\hat{\mathbf{J}}$

Note that an alternative implementation would keep increasing I until the gradient estimate converges. The choice of I can be essential; empirically it can be observed that taking I as twice the number of parameters yields very accurate gradient estimates.

Due to the simplicity of this approach, such methods have been successfully applied to numerous applications. However, the straightforward application is not without peril as the generation of the $\mathbf{\Delta}\theta_i$ requires proper knowledge of the system, as badly chosen $\mathbf{\Delta}\theta_i$ can destabilize the policy so that the system becomes instable and the gradient estimation process is prone to fail. If the parameters differ highly in scale, significant difficulties could be the consequences.

Advantages of this approach: Finite-difference methods require very little skill and can usually be implemented out of the box. They work both with stochastic and deterministic policies without any change. It is highly efficient in simulation with a set of common histories of random numbers and on totally deterministic systems.

Disadvantages of this approach: The perturbation of the parameters is a very difficult problem often with disastrous impact on the learning process when the system goes unstable. In the presence of noise on a real system, the gradient estimate error decreases much slower than for the following methods. Performance depends highly on the chosen policy parametrization.

Sehnke et al. (2010) show several interesting newer methods developed in this domain.

Likelihood Ratio Methods and REINFORCE

Likelihood ratio methods are driven by a different important insight. Assume that trajectories τ are generated from a system by roll-outs, i.e., $\tau \sim p_\theta(\tau) = p\big(\tau|\theta\big)$ with return $r(\tau) = \sum_{k=0}^{H} a_k r_k$ which leads to $J(\theta) = E\{r(\tau)\} = \int_{\mathbb{T}} p_\theta(\tau) r(\tau) d\tau$. In this case, the policy gradient can be estimated using the likelihood ratio (see e.g. Glynn, 1987; Aleksandrov, Sysoyev, and Shemeneva, 1968) better known as REINFORCE (Williams, 1992) trick, i.e., by using

$$\nabla_\theta p_\theta(\tau) = p_\theta(\tau) \nabla_\theta \log p_\theta(\tau)$$

from standard differential calculus ($\nabla_\theta \log p_\theta(\tau) = \nabla_\theta p_\theta(\tau) / p_\theta(\tau)$), we obtain

$$\nabla_\theta J\big(\theta\big) = \int_{\mathbb{T}} \nabla_\theta p_\theta(\tau) r(\tau) d\tau = \int_{\mathbb{T}} p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) r(\tau) d\tau = E\Big\{\nabla_\theta \log p_\theta(\tau) r(\tau)\Big\}.$$

As the expectation $E\{\cdot\}$ can be replaced by sample averages, denoted by $\langle \cdot \rangle$, only the derivative $\nabla_\theta \log p_\theta(\tau)$ is needed for the estimator. Importantly, this derivative can be computed without knowledge of the generating distribution $p_\theta(\tau)$ as

$$p_\theta(\tau) = p(\mathbf{x}_0) \prod_{k=0}^{H} p\big(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k\big) \pi_\theta(\mathbf{u}_k|\mathbf{x}_k)$$

implies that

$$\nabla_\theta \log p_\theta(\tau) = \sum_{k=0}^{H} \nabla_\theta \log \pi_\theta(\mathbf{u}_k|\mathbf{x}_k)$$

as only the policy depends on θ .

Thus, the derivatives of $p\big(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k\big)$ do not have to be computed and no model needs to be maintained. However, if we had a deterministic policy $\mathbf{u} = \pi(\mathbf{x})$ instead of a stochastic policy $\mathbf{u} \sim \pi(\mathbf{u}|\mathbf{x})$, computing such a derivative would require the derivative $\nabla_\theta \log p\big(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k\big) = \nabla_{\mathbf{u}_k} \log p\big(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k\big) \nabla_\theta \pi_\theta(\mathbf{x}_k)$ to compute $\nabla_\theta \log p_\theta(\tau)$ and, hence, it would require a system model.

In order to reduce the variance of the gradient estimator, a constant baseline can be subtracted from the gradient, i.e.,

$$\nabla_\theta J\big(\theta\big) = E\Big\{\nabla_\theta \log p_\theta(\tau) \big(r(\tau) - b\big)\Big\},$$

where the baseline $b \in \mathbb{R}$ can be chosen arbitrarily (Williams, 1992). It is straightforward to show that this baseline does not introduce bias in the gradient as differentiating $\int_{\mathbb{T}} p_\theta(\tau) d\tau = 1$ implies that

$$\int_{\mathbb{T}} \nabla_\theta p_\theta(\tau) d\tau = 0\,,$$

and, hence, the constant baseline will vanish for infinite data while reducing the variance of the gradient estimator for finite data. See Peters & Schaal, 2008 for an overview of how to choose the baseline optimally. Therefore, the general path likelihood ratio estimator or episodic REINFORCE gradient estimator is given by

$$\mathbf{g}_{\mathrm{RF}} = \left\langle \left(\sum_{k=0}^{H} \nabla_\theta \log \pi_\theta(\mathbf{u}_k|\mathbf{x}_k) \right) \left(\sum_{l=0}^{H} a_l r_l - b \right) \right\rangle,$$

where $\langle \cdot \rangle$ denotes the average over trajectories. This type of method is guaranteed to converge to the true gradient at the fastest theoretically possible error decrease of $O\left(I^{-1/2}\right)$ where I denotes the number of roll-outs (Glynn, 1987) even if the data is generated from a highly stochastic system. An implementation of this algorithm will be shown below together with the estimator for the optimal baseline.

```
input: policy parameterization θ_h
repeat
    perform a trial and obtain x_{0:H}, u_{0:H}, r_{0:H}
    for each  gradient element g_k
        estimate optimal baseline
            b = <(Σ_{h=0}^{H} ∇_{θ_k} log π_θ(u_h|x_h))^2 Σ_{l=0}^{H} a_l r_l> / <(Σ_{h=0}^{H} ∇_{θ_k} log π_θ(u_h|x_h))^2>
        estimate the gradient element
            g_k = <(Σ_{h=0}^{H} ∇_{θ_k} log π_θ(u_h|x_h)) (Σ_{l=0}^{H} a_l r_l - b)>
    end for
until  gradient estimate g_RF = [g_1, ..., g_K] converged
return gradient estimate g_RF = [g_1, ..., g_K]
```

Advantages of this approach: Besides the theoretically faster convergence rate, likelihood ratio gradient methods have a variety of advantages in comparison to finite difference methods when applied to robotics. As the generation of policy parameter variations is no longer needed, the complicated control of these variables can no longer endanger the gradient estimation process. Furthermore, in practice, already a single roll-out can suffice for an unbiased gradient estimate viable for a good policy update step, thus reducing the amount of roll-outs needed. Finally, this approach has yielded the most real-world robotics results (Peters & Schaal, 2008) and the likelihood ratio gradient is guaranteed to achieve the fastest convergence of the error for a stochastic system.

Disadvantages of this approach: When used with a deterministic policy, likelihood ratio gradients have to maintain a system model. Such a model can be very hard to obtain for continuous states and actions, hence, the simpler finite difference gradients are often superior in this scenario. Similarly, finite difference gradients can still be more useful than likelihood ratio gradients if the system is deterministic and very repetitive. Also, the practical implementation of a likelihood ratio gradient method is much more demanding than the one of a finite difference method.

Natural Policy Gradients

One of the main reasons for using policy gradient methods is that we intend to do just a small change $\Delta\theta$ to the policy π_θ while improving the policy. However, the meaning of small is ambiguous. When using the Euclidian metric of $\sqrt{\Delta\theta^T \Delta\theta}$, then the gradient is different for every parameterization θ of the policy π_θ even if these parameterization are related to each other by a linear transformation (Kakade, 2002). This problem poses the question of how we can measure the closeness between the current policy and the updated policy based upon the distribution of the paths generated by each of these. In statistics, a variety of distance measures for the closeness of two distributions (e.g., $p_\theta(\tau)$ and $p_{\theta+\Delta\theta}(\tau)$) have been suggested, e.g., the Kullback-Leibler divergence $d_{\mathrm{KL}}\left(p_\theta, p_{\theta+\Delta\theta}\right)$, the Hellinger distance d_{HD} and others (Su & Gibbs, 2002). Many of these distances (e.g., the previously mentioned ones) can be approximated by its second order Taylor expansion, i.e., by

$$d_{\mathrm{KL}}\left(p_\theta, p_{\theta+\Delta\theta}\right) \approx \Delta\theta^T \mathbf{F}_\theta \, \Delta\theta,$$

where

$$\mathbf{F}_\theta = \int_{\mathbb{T}} p_\theta(\tau) \nabla \log p_\theta(\tau) \nabla \log p_\theta(\tau)^T d\tau = \left\langle \nabla \log p_\theta(\tau) \nabla \log p_\theta(\tau)^T \right\rangle$$

is known as the Fisher-information matrix. Let us now assume that we restrict the change of our policy to $d_{\mathrm{KL}}\left(p_\theta, p_{\theta+\Delta\theta}\right) \approx \Delta\theta^T \mathbf{F}_\theta \, \Delta\theta = \varepsilon$, where ε needs to be a very small number (i.e., close to zero).

In that case, the natural gradient is defined by Amari (1998) as the update $\Delta\tilde{\theta}$ that is most similar to the true gradient $\nabla_\theta J$ while the change in our path distribution is limited to ε . Hence, it is given by the program

$$\mathrm{argmax}_{\Delta\theta} \, \Delta\theta^T \nabla_\theta J \quad \textbf{s. t.} \quad \Delta\theta^T \mathbf{F}_\theta \, \Delta\theta = \varepsilon.$$

The solution to this program is given by

$$\Delta\theta \propto \mathbf{F}_\theta^{-1} \nabla_\theta J,$$

where $\nabla_\theta J$ denotes the regular likelihood ratio policy gradient from the previous section. The update step is unique up to a scaling factor, which is often subsumed into the learning rate. It can be interpreted as follows: determine the maximal improvement $\Delta\theta$ of the policy for a constant fixed change of the policy $\Delta\theta^T \mathbf{F}_\theta \Delta\theta$.

As illustrated in Figure 1, the natural gradient update in Figure 1 (b) corresponds to a slightly rotated regular policy gradient update in Figure 1 (a). It can be guaranteed that it is always turned by less than 90 degrees (Amari, 1998), hence all convergence properties of the regular policy gradient transfer.

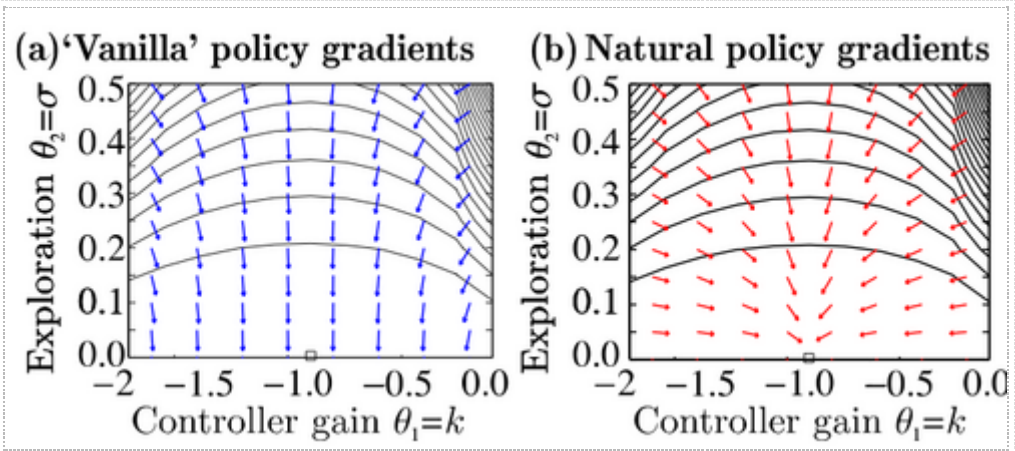

Figure 1: When plotting the expected return landscape for a simple problem such as an 1d linear quadratic regulation, the differences between regular ('vanilla') and natural policy gradients becomes apparent.

This type of approach has its origin in supervised learning (Amari, 1998). It was first suggested in the context of reinforcement learning by Kakade (2002) and has been explored in greater depth in (Bagnell & Schneider, 2003; Peters, Vijayakumar & Schaal, 2003, 2005; Peters & Schaal, 2008). The strongest theoretical advantage of this approach is that its performance no longer depends on the parameterization of the policy and is therefore safe to be used for arbitrary policies. Hence, the regular policy gradient is sometimes referred to as a flavored or vanilla gradient, as it keeps the `vanilla flavor' of the policy. However, a well-chosen policy parametrization can sometimes results in better convergence of the policy gradient than the natural policy gradient. Nevertheless, in practice, a learning process based on natural policy gradients often converges significantly faster for most practical cases. Figure 1 gives an impression of the differences in the learning process: while the regular policy gradient often points to plateaus with little exploration, the natural gradient points to the optimal solution.

One of the fastest general algorithms for estimating natural policy gradients which does not need complex parameterized baselines is the episodic natural actor critic. This algorithm, originally derived in (Peters, Vijayakumar & Schaal, 2003), can be considered the `natural' version of REINFORCE with a baseline optimal for this gradient estimator. However, for steepest descent with respect to a metric, the baseline also needs to minimize the variance with respect to the same metric. In this case, we can minimize the whole covariance matrix of the natural gradient estimate $\Delta\hat{\theta}$ given by

$$\mathbf{\Sigma} = \mathrm{Cov}\left\{\Delta\hat{\theta}\right\}_{\mathbf{F}_\theta} = E\left\{\left(\Delta\hat{\theta} - \mathbf{F}_\theta^{-1} \mathbf{g}_{\mathrm{RF}}(b)\right)^T \mathbf{F}_\theta \left(\Delta\hat{\theta} - \mathbf{F}_\theta^{-1} \mathbf{g}_{\mathrm{RF}}(b)\right)\right\},$$

with

$$\mathbf{g}_{\mathrm{RF}}(b) = \left\langle \nabla \log p_\theta(\tau)\big(r(\tau) - b\big) \right\rangle$$

being the REINFORCE gradient with baseline b . The re-weighting with the Fisher information ensures that the best identifiable gradient components get the highest weight. As outlined in (Peters & Schaal, 2008), it can be shown that the minimum-variance unbiased natural gradient estimator can be determined as shown below.

input: policy parameterization θ_h
repeat
  perform M trials and obtain $\mathbf{x}_{0:H}, \mathbf{u}_{0:H}, r_{0:H}$ for each trial
  Obtain the sufficient statistics
    Policy derivatives $\psi_k = \nabla_\theta \log \pi_\theta(\mathbf{u}_k|\mathbf{x}_k)$
    Fisher matrix $\mathbf{F}_\theta = \left\langle \left(\sum_{k=0}^{H} \psi_k\right)\left(\sum_{l=0}^{H} \psi_l\right)^T \right\rangle$
    Vanilla gradient $\mathbf{g} = \left\langle \left(\sum_{k=0}^{H} \psi_k\right)\left(\sum_{l=0}^{H} a_l r_l\right) \right\rangle$
    Eligibility $\phi = \left\langle \left(\sum_{k=0}^{H} \psi_k\right) \right\rangle$
    Average reward $\bar{r} = \left\langle \sum_{l=0}^{H} a_l r_l \right\rangle$
  Obtain natural gradient by computing
    Baseline $b = \mathbf{Q}\left(\bar{r} - \phi^T \mathbf{F}_\theta^{-1} \mathbf{g}\right)$
      with $\mathbf{Q} = M^{-1}\left(1 + \phi^T \left(M\mathbf{F}_\theta - \phi\phi^T\right)^{-1}\phi\right)$
    Natural gradient $\mathbf{g}_{\mathrm{NG}} = \mathbf{F}_\theta^{-1}\left(\mathbf{g} - \phi b\right)$
until gradient estimate $\mathbf{g}_{\mathrm{NG}} = [g_1, \ldots, g_h]$ converged
return gradient estimate $\mathbf{g}_{\mathrm{NG}} = [g_1, \ldots, g_h]$

For the derivation, see (Peters & Schaal, 2008).

Advantages of this approach: Natural policy gradients differ in a deciding aspect from both finite difference gradients and regular likelihood ratio gradients, i.e., they are independent from the choice of policy parametrization if the choices have the same representational power. As a result, they can be an order of magnitude faster than the regular gradient. They also profit from most other advantages of the regular policy gradients.

Disadvantages of this approach: In comparison to the regular policy gradient, there are three disadvantages: first, the matrix inversion in the gradient estimators may be numerically brittle and may scale worse (note that there are tricks to alleviate this problem). Second, if we can find a special policy parametrization that trivializes a problem, the natural policy gradient may not make use of it. Third, the natural policy gradient estimators are often much harder to implement.

Conclusion

We have presented an quick overview on policy gradient methods. While many details needed to be omitted and may be found in (Peters & Schaal, 2008), this entry roughly represents the state of the art in policy gradient methods. All three major ways of estimating first order gradients, i.e., finite-difference gradients, vanilla policy gradients and natural policy gradients are discussed in this article and practical algorithms are given.

References

- V. Aleksandrov, V. Sysoyev, and V. Shemeneva, *Stochastic optimization,* Engineering Cybernetics, vol. 5, pp. 11-16, 1968.
- S. Amari, *Natural gradient works efficiently in learning,* Neural Computation, vol. 10, 1998.

- J. Bagnell and J. Schneider, *Covariant policy search,* International Joint article on Artificial Intelligence, 2003.
- J. Baxter and P. Bartlett, *Direct gradient-based reinforcement learning,* Journal of Artificial Intelligence Research, 1999.
- H. Benbrahim and J. Franklin, *Biped dynamic walking using reinforcement learning,* Robotics and Autonomous Systems, vol. 22, pp. 283–302, 1997.
- A. Berny, *Statistical machine learning and combinatorial optimization,* In L. Kallel, B. Naudts, and A. Rogers, editors, Theoretical Aspects of Evolutionary Computing, Lecture Notes in Natural Computing, 2000.
- P. Dyer and S. R. McReynolds, *The Computation and Theory of Optimal Control*. New York: Academic Press, 1970.
- G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng, *Learning CPG sensory feedback with policy gradient for biped locomotion for a full-body humanoid,* in AAAI 2005, 2005.
- R. Fletcher and R. Fletcher, *Practical Methods of Optimization*. John Wiley & Sons, 2000.
- P. Glynn, *Likelihood ratio gradient estimation for stochastic systems*, Communications of the ACM, vol. 33, no. 10, pp. 75-84, October 1990.
- P. Glynn, *Likelihood ratio gradient estimation: an overview,* in Proceedings of the 1987 Winter Simulation Conference, Atlanta, GA, 1987, pp. 366--375.
- E. Greensmith, P. Bartlett, and J. Baxter, *Variance reduction techniques for gradient estimates in reinforcement learning,* Advances in Neural Information Processing Systems, 2001.
- V. Gullapalli, *Associative reinforcement learning of real-value functions*, SMC, 1991.
- V. Gullapalli, J. Franklin, and H. Benbrahim, *Acquiring robot skills via reinforcement learning,* IEEE Control Systems, vol. 14, no. 39, 1994.
- L. Hasdorff, *Gradient optimization and nonlinear control*. John Wiley & Sons, 1976.
- D. H. Jacobson and D. Q. Mayne, *Differential Dynamic Programming*. New York: American Elsevier Publishing Company, Inc., 1970.
- J. Kober and J. Peters, *Policy Search for Motor Primitives in Robotics*, Advances in Neural Information Processing Systems 22 (NIPS), Cambridge, MA: MIT Press, 2008.
- N. Kohl and P. Stone, *Policy gradient reinforcement learning for fast quadrupedal locomotion*, in Proceedings of the IEEE International Conference on Robotics and Automation, New Orleans, LA, May 2004.
- G. Lawrence, N. Cowan, and S. Russell, *Efficient gradient estimation for motor control learning,* in Proc. UAI-03, Acapulco, Mexico, 2003.
- A. Y. Ng and M. Jordan, *PEGASUS: A policy search method for large MPDs and POMDPs,* in *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 2000.
- J. Peters and S. Schaal, (2008). "Natural actor critic," Neurocomputing, 71, 7-9, pp.1180-1190.
- J. Peters and S. Schaal, (2008). "Reinforcement learning of motor skills with policy gradients," Neural Networks, 21, 4, pp.682-97.
- J. Peters, S. Vijayakumar, and S. Schaal, *Reinforcement learning for humanoid robotics,* in IEEE/RSJ International Conference on Humanoid Robotics, 2003.
- J. Peters, S. Vijayakumar, and S. Schaal, *Natural actor-critic,* in Proceedings of the European Machine Learning Conference (ECML)}, 2005.
- F. Sehnke, C. Osendorfer, T. Rückstiess, A. Graves, J. Peters, J. Schmidhuber. *Parameter-exploring policy gradients*. Neural Networks 23(2), 2010.
- J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Hoboken, NJ: Wiley, 2003.

- F. Su and A. Gibbs, *On choosing and bounding probability metrics*, International Statistical Review, vol. 70, no. 3, pp. 419-435, 2002.
- R. Sutton, D. McAllester, S. Singh, and Y. Mansour, *Policy gradient methods for reinforcement learning with function approximation,* Advances in Neural Information Processing Systems, 2000.
- R. Tedrake, T. W. Zhang, and H. S. Seung. *Stochastic policy gradient reinforcement learning on a simple 3D biped*. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), vol. 3, pages 2849-2854, Sendai, Japan, September 2004.
- R. J. Williams, *Simple statistical gradient-following algorithms for connectionist reinforcement learning,* Machine Learning, vol. 8, no. 23, 1992.
- N. Vlassis, M. Toussaint, G. Kontes, and S. Piperidis. *Learning Model-free Robot Control by a Monte Carlo EM Algorithm*. Autonomous Robots, 27(2):123-130, 2009.

See also

Reinforcement learning

Sponsored by: Eugene M. Izhikevich, Editor-in-Chief of Scholarpedia, the peer-reviewed open-access encyclopedia

Reviewed by (http://www.scholarpedia.org/w/index.php?title=Policy_gradient_methods&oldid=82031) : Anonymous

Reviewed by (http://www.scholarpedia.org/w/index.php?title=Policy_gradient_methods&oldid=82031) : Dr. Christoph Kolodziejski, Max Planck Institute for Dynamics and Self-Organization, Göttingen, Germany

Accepted on: 2010-10-12 14:11:57 GMT (http://www.scholarpedia.org/w/index.php?title=Policy_gradient_methods&oldid=82031)

Categories: Computational Intelligence | Machine learning | Pattern Recognition | Reinforcement learning