Reinforcement Learning for Parameterized Motor Primitives

Conference Paper · January 2006

DOI: 10.1109/IJCNN.2006.246662 · Source: DBLP

2 authors:

Jan Peters
Technische Universität Darmstadt
391 PUBLICATIONS **9,744** CITATIONS
SEE PROFILE

Stefan Schaal
University of Southern California
437 PUBLICATIONS **20,965** CITATIONS
SEE PROFILE

2006 International Joint Conference on Neural Networks
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Reinforcement Learning for Parameterized Motor Primitives

Jan Peters and Stefan Schaal

Abstract— **One of the major challenges in both action generation for robotics and in the understanding of human motor control is to learn the "building blocks of movement generation", called motor primitives. Motor primitives, as used in this paper, are parameterized control policies such as splines or nonlinear differential equations with desired attractor properties. While a lot of progress has been made in teaching parameterized motor primitives using supervised or imitation learning, the self-improvement by interaction of the system with the environment remains a challenging problem.**

In this paper, we evaluate different reinforcement learning approaches for improving the performance of parameterized motor primitives. For pursuing this goal, we highlight the difficulties with current reinforcement learning methods, and outline both established and novel algorithms for the gradient-based improvement of parameterized policies. We compare these algorithms in the context of motor primitive learning, and show that our most modern algorithm, the Episodic Natural Actor-Critic outperforms previous algorithms by at least an order of magnitude. We demonstrate the efficiency of this reinforcement learning method in the application of learning to hit a baseball with an anthropomorphic robot arm.

I. Introduction

In order to ever leave the well-structured environments of factory floors and research labs, future robots will require the ability to aquire novel behaviors and motor skills as well as to improve existing ones based on rewards and costs. Similarly, the understanding of human motor control would benefit significantly if we can synthesize simulated human behavior and its underlying cost functions based on insight from machine learning and biological inspirations. Reinforcement learning is probably the most general framework in which such learning problems of computational motor control can be phrased. However, in order to bring reinforcement learning into the domain of human movement learning, two deciding components need to be added to the standard framework of reinforcement learning: first, we need a domain-specific policy representation for motor skills, and, second, we need reinforcement learning algorithms which work efficiently with this representation while scaling into the domain of high-dimensional mechanical systems such as humanoid robots.

Traditional representations of motor behaviors in robotics are mostly based on desired trajectories generated from spline interpolations between points, i.e., spline nodes, which are part of a longer sequence of intermediate target points on the way to a final movement goal. While such a representation is easy to understand, the resulting control policies, generated from a tracking controller of the spline trajectories, have a variety of significant disadvantages, including that they are time-indexed and thus not robust towards unforeseen disturbances, that they do not easily generalize to new bahavioral sitations without complete recomputing of the spline, and that they cannot easily be coordinated with other events in the environment, e.g., synchronized with other sensory variables like visual perception during catching a ball. In the literature, a variety of other approaches for parameterizing motor primitives have been suggested to overcome these problems (see [1], [2] for more information). One of these [1], [2] proposed to use parameterized nonlinear dynamical systems as motor primitives, where the attractor properties of these dynamical systems defined the desired behavior. The resulting framework was particularly well suited for supervised imitation learning in robotics, exemplified by examples from humanoid robotics where a full-body humanoid learned tennis swings or complex polyrhythmic drumming pattern. One goal of this paper is to the application of reinforcement learning to both traditional spline-based representations as well as the more novel dynamic system based approach.

However, despite that reinforcement learning is the most general framework for discussing the learning of motor primitives for robotics, most of the methods proposed in the reinforcement learning community are not applicable to high-dimensional systems such as humanoid robots as these methods do not scale beyond systems with more than three or four degrees of freedom and/or cannot deal with parameterized policies. Policy gradient methods are a notable exception to this statement. Starting with the pioneering work of Gullapali, Franklin and Benbrahim [3], [4] in the early 1990s, these methods have been applied to a variety of robot learning problems ranging from simple control tasks (e.g., balancing a ball-on a beam [5] , and pole-balancing [6]) to complex learning tasks involving many degrees of freedom such as learning of complex motor skills [4], [7], [8] and locomotion [9]–[15].

The advantages of policy gradient methods for parameterized motor primitives are numerous. Among the most important ones are that the policy representation can be chosen such that it is meaningful for the task, i.e., we can use a suitable motor primitive representation, and that domain knowledge can be incorporated, which often leads to fewer parameters in the learning process in comparison to traditional value-function based approaches. Moreover, there exists 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 model-free and therefore also be applied

Jan Peters and Stefan Schaal are with the Department of Computer Science, University of Southern California, Los Angeles, CA 90089, USA (phone: 213-740-6717; fax: 213-740-1510; email: {jrpeters,sschaal}@usc.edu).

0-7803-9490-9/06/$20.00/©2006 IEEE

to problems without analytically known task and reward models.

Nevertheless, many recent publications on applications of policy gradient methods in robotics overlooked the newest developments in policy gradient theory and its original roots in the literature. Thus, a large number of heuristic applications of policy gradients can be found, where the success of the projects mainly relied on ingenious initializations and manual parameter tuning of algorithms. A closer inspection often reveals that the chosen methods might be highly biased, or even generate infeasible policies under less fortunate parameter settings, which could lead to unsafe operation of a robot. The main goal of this paper is to review which policy gradient methods are applicable to robotics and which issues matter, while also introducing some new policy gradient learning algorithms that seem to have superior performance over previously suggested methods. The remainder of this paper will proceed as follows: firstly, we will introduce the general assumptions of reinforcement learning, discuss motor primitives in this framework and pose the problem statement of this paper. Secondly, we will discuss the different approaches to policy gradient estimation and discuss their applicability to reinforcement learning of motor primitives. We focus on the most useful methods and discuss several algorithms in-depth. The presented algorithms in this paper are highly optimized versions of both novel and previous policy gradient algorithms. Thirdly, we show how these methods can be applied to motor skill learning in robotics and show learning results with a seven degrees of freedom, anthropomorphic SARCOS Master ARM.

A. General Assumptions and Notations

Most robotics domains require the state space and the action spaces to be continuous and high-dimensional such that learning methods based on discretizations are not applicable for higher dimensional systems. However, as the policy is usually implemented on a digital computer, 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 $\mathbf{u}_k \sim \pi_\theta\left(\mathbf{u}_k | \mathbf{x}_k\right)$ 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 [16]. The policy is assumed to be parameterized by some policy parameters $\theta \in \mathbb{R}^K$.The sequence of states and actions forms a trajectory (also called history or roll-out) denoted by $\tau = [\mathbf{x}_{0:H}, \mathbf{u}_{0:H}]$ where H denotes the horizon which can be infinite. At each instant of time, the learning system receives a reward denoted by $r\left(\mathbf{x}_k, \mathbf{u}_k\right) \in \mathbb{R}$.

B. Motor Primitive Policies

In this section, we first discuss how motor plans can be represented and then how we can bring these into the standard reinforcement learning framework. For this purpose, we consider two forms of motor plans, i.e., (1) *spline-based trajectory plans* and (2)*nonlinear dynamic motor primitives* introduced in [1]. Spline-based trajectory planning is well-known in the robotics literature, see e.g., [17], [18]. A desired trajectory is represented by piecewise connected polynomials, i.e., we have $y_i\left(t\right) = \theta_{0i} + \theta_{1i} t + \theta_{2i} t^2 + \theta_{3i} t^3$ in $t \in [t_i, t_{i+1}]$ under the constraints that both $y_i\left(t_{i+1}\right) = y_{i+1}\left(t_{i+1}\right)$ and $\dot{y}_i\left(t_{i+1}\right) = \dot{y}_{i+1}\left(t_{i+1}\right)$. A given tracking controller, e.g., a PD control law or an inverse dynamics controller, ensures that the trajectory is tracked well. For nonlinear dynamic motor primitives, we use the approach developed in [1] where movement plans $(\mathbf{q}_d, \dot{\mathbf{q}}_d)$ for each degree of freedom (DOF) of the robot are represented in terms of the time evolution of the nonlinear dynamical systems

$$\ddot{q}_{d,k} = h(q_{d,k}, \mathbf{z}_k, g_k, \tau, \theta_k) \tag{1}$$

where $(q_{d,k}, \dot{q}_{d,k})$ denote the desired position and velocity of a joint, z_k the internal state of the dynamic system, g_k the goal (or point attractor) state of each DOF, τ the movement duration shared by all DOFs, and θ_k the open parameters of the function h. The equations used in order to create Equation (1) are given in Appendix A. The original work in [1] demonstrated how the parameters θ_k can be learned to match a template trajectory by means of supervised learning – this scenario is, for instance, useful as the first step of an imitation learning system. Here we will add the ability of self-improvement of the movement primitives in Eq.(1) by means of reinforcement learning, which is the crucial second step in imitation learning. The system in Eq.(1) is a point-to-point movement, i.e., this task is rather well suited for the introduced episodic reinforcement learning methods.

In order to make the reinforcement framework feasible for learning motor primitives, we need to add exploration to the respective motor primitive framework, i.e., we need to add a small perturbation $\epsilon_{d,k} \sim \mathcal{N}(0, \sigma^2)$ with exploration rate σ^2 to each motor primitive output so that $\hat{\ddot{q}}_{d,k} = \ddot{q}_{d,k} + \epsilon_{d,k}$ where $\hat{\ddot{q}}_{d,k}$ denotes the target output. By doing so, we obtain a stochastic policy

$$\pi(\hat{\ddot{q}}_{d,k} | q_{d,k}, \mathbf{z}_k, g_k, \tau, \theta_k) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\hat{\ddot{q}}_{d,k} - \ddot{q}_{d,k})^2}{2\sigma^2}\right). \tag{2}$$

This policy will be used throughout the paper. It is particularly practical as the exploration can be easily controlled through only one variable $sigma$.

C. Problem Statement

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

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

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 robotics, we require that any change to the policy parameterization has to be smooth as drastic changes can be hazardous for the robot, and for its environnment 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

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

where $\alpha_h \in \mathbb{R}^+$ denotes a learning rate. If the gradient estimate is unbiased and learning rates fulfill $\sum_{h=0}^{\infty} \alpha_h > 0$ and $\sum_{h=0}^{\infty} \alpha_h^2 = 0$, the learning process is guaranteed to converge to at least a local minimum.

II. Policy Gradient Methods for Parameterized Motor Primitives

The main problem in policy gradient methods is to obtain a good estimator of the policy gradient $\left.\nabla_\theta J\right|_{\theta=\theta_h}$. Traditionally, people have used deterministic model-based methods for obtaining the gradient [19]–[21]. However, in order to become autonomous we cannot expect to be able to model every detail of the robot and environment. 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 which of these are useful in robotics.

A. General 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, more well-known as REINFORCE methods in reinforcement learning.

1) 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 by small increments $\mathbf{\Delta}\theta_i$ and for each policy parameter variation $\theta_h + \mathbf{\Delta}\theta_i$ roll-outs are performed which generate estimates $\Delta\hat{J}_j \approx J(\theta_h + \mathbf{\Delta}\theta_i) - J_{\text{ref}}$ 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 $J_{\text{ref}} = J(\theta_h - \mathbf{\Delta}\theta_i)$. The policy gradient estimate $\mathbf{g}_{\text{FD}} \approx \left.\nabla_\theta J\right|_{\theta=\theta_h}$ can be estimated by regression yielding

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

where $\mathbf{\Delta\Theta} = [\mathbf{\Delta}\theta_1, \ldots, \mathbf{\Delta}\theta_I]^T$ and $\mathbf{\Delta\hat{J}} = [\Delta\hat{J}_1, \ldots, \Delta\hat{J}_I]^T$ denote the I samples. This approach can been highly efficient in simulation optimization of deterministic systems [22] or when a common history of random numbers [23] is being used (the later is known as PEGASUS in reinforcement learning [24]), and can get close to a convergence rate of $O\left(I^{-1/2}\right)$ [23]. However, when used on a real system, the uncertainities degrade the performance resulting in convergence rates ranging between $O\left(I^{-1/4}\right)$ to $O\left(I^{-2/5}\right)$ depending on the chosen reference value [23]. An implementation of this algorithm is shown in Table I.

Due to the simplicity of this approach, such methods have been successfully applied to robot motor skill learning in numerous applications [8], [11], [13]. However, the straightforward application is not without peril as the generation of the $\mathbf{\Delta}\theta_j$ requires proper knowledge on the system, as badly chosen $\mathbf{\Delta}\theta_j$ can destabilize the policy so that the system becomes instable and the gradient estimation process is prone to fail. Practical problems often require that each element of the vector $\mathbf{\Delta}\theta_j$ has a different order of magnitude, making the generation particularly difficult. Therefore, this approach can only applied under strict supervision of human.

2) Likelihood Ratio Methods / REINFORCE: Likelihood ratio methods are driven by an important different insight. Assume that trajectories τ are generated from a system by roll-outs, i.e., $\tau \sim p_\theta(\tau) = p(\tau|\theta)$ with rewards $r(\tau) = \sum_{k=0}^{H} a_k r_k$. In this case, the policy gradient can be estimated using the likelihood ratio (see e.g. [23], [25]) or REINFORCE [26] trick, i.e., by

$$\nabla_\theta J(\theta) = \int_{\mathbb{T}} \nabla_\theta p_\theta(\tau)\, r(\tau) d\tau = E\left\{\nabla_\theta \log p_\theta(\tau)\, r(\tau)\right\},$$

as $\int_{\mathbb{T}} \nabla_\theta p_\theta(\tau)\, r(\tau) d\tau = \int_{\mathbb{T}} p_\theta(\tau)\, \nabla_\theta \log p_\theta(\tau)\, r(\tau) d\tau$. Importantly, the derivative $\nabla_\theta \log p_\theta(\tau)$ can be computed without knowleged of the generating distribution $p_\theta(\tau)$ as $p_\theta(\tau) = p(\mathbf{x}_0)\prod_{k=0}^{H} p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k)\,\pi_\theta(\mathbf{u}_k|\mathbf{x}_k)$ implies that

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

i.e., the derivatives through the control system do not have to be computed[1]. As $\int_{\mathbb{T}} \nabla_\theta p_\theta(\tau)\, d\tau = 0$, a constant baseline

[1]This result makes an important difference: in stochastic system optimization, finite difference estimators are often prefered as the derivative through system is required but not known. In policy search, we always know the derivative of the policy with respect to its parameters and therefore we can make use of the theoretical advantages of likelihood ratio gradient estimators.

TABLE I

Finite difference gradient estimator.

input: policy parameterization θ_h.	
1	**repeat**
2	generate policy variation $\mathbf{\Delta}\theta_1$.
3	estimate $\hat{J}_j \approx J(\theta_h + \mathbf{\Delta}\theta_i) = \left\langle \sum_{k=0}^{H} a_k r_k \right\rangle$ from roll-out.
4	estimate $\hat{J}_{\text{ref}}$, e.g., $\hat{J}_{\text{ref}} = J(\theta_h - \mathbf{\Delta}\theta_i)$ from roll-out.
5	compute $\Delta\hat{J}_j \approx J(\theta_h + \mathbf{\Delta}\theta_i) - J_{\text{ref}}$.
6	compute gradient $\mathbf{g}_{\text{FD}} = \left(\mathbf{\Delta\Theta}^T\mathbf{\Delta\Theta}\right)^{-1}\mathbf{\Delta\Theta}^T\mathbf{\Delta\hat{J}}$.
7	**until** gradient estimate $\mathbf{g}_{\text{FD}}$ converged.
return: gradient estimate $\mathbf{g}_{\text{FD}}$.	

can be inserted resulting into the gradient estimator

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

where $b \in \mathbb{R}$ can be chosen arbitrarily [26] but usually with the goal to minimize the variance of the gradient estimator. Therefore, the general path likelihood ratio estimator or episodic REINFORCE gradient estimator is given by

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

where $\langle\cdot\rangle$ denotes the average over trajectories [26]. This type of method is guaranteed to converge to the true gradient at the fastest theoretically possible pace of $O\left(I^{-1/2}\right)$ where I denotes the number of roll-outs [23] even if the data is generated from a highly stochastic system. An implementation of this algorithm will be shown in Table II together with the estimator for the optimal baseline.

Besides the theoretically faster convergence rate, likelihood ratio gradient methods have a variety of advantages in comparison to finite difference methods. 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 [22], [27] viable for a good policy update step, thus reducing the amount of roll-outs needed. Finally, this approach has yielded the most real-world robot motor learning results [3], [4], [7], [9], [12], [14], [15]. In the subsequent two sections, we will strive to explain and improve this type of gradient estimator.

B. 'Vanilla' Policy Gradient Approaches

Despite the fast asymptotic convergence speed of the gradient estimate, the variance of the likelihood-ratio gradient estimator can be problematic in practice. For this reason, we will discuss several advances in likelihood ratio policy gradient optimization, i.e., the policy gradient theorem/GPOMDP and optimal baselines[2].

[2]Note that the theory of the compatible function approximation [16] is omitted at this point as it does not contribute to practical algorithms in this context. For a thorough discussion of this topic see [7], [28].

TABLE II
GENERAL LIKELIHOOD RATIO POLICY GRADIENT ESTIMATOR "EPISODIC REINFORCE" WITH AN OPTIMAL BASELINE.

input: policy parameterization θ_h.

repeat
  perform a trial and obtain $\mathbf{x}_{0:H}, \mathbf{u}_{0:H}, r_{0:H}$
  for each gradient element g_h
    estimate optimal baseline
    $$b^h = \frac{\left\langle \left(\sum_{k=0}^{H} \nabla_{\theta_h} \log \pi_\theta(\mathbf{u}_k|\mathbf{x}_k)\right)^2 \sum_{l=0}^{H} a_l r_l \right\rangle}{\left\langle \left(\sum_{k=0}^{H} \nabla_{\theta_h} \log \pi_\theta(\mathbf{u}_k|\mathbf{x}_k)\right)^2 \right\rangle}$$
    estimate the gradient element
    $$g_h = \left\langle \left(\sum_{k=0}^{H} \nabla_{\theta_h} \log \pi_\theta(\mathbf{u}_k|\mathbf{x}_k)\right)\left(\sum_{l=0}^{H} a_l r_l - b^h\right)\right\rangle.$$
  end for.
until gradient estimate $\mathbf{g}_{\text{FD}} = [g_1, \ldots, g_h]$ converged.

return: gradient estimate $\mathbf{g}_{\text{FD}} = [g_1, \ldots, g_h]$.

1) Policy gradient theorem/GPOMDP: The trivial observation that future actions do not depend on past rewards (unless the policy has been changed) can result in a significant reduction of the variance of the policy gradient estimate. This insight can be formalized as $E\left\{\nabla_\theta \log \pi_\theta(\mathbf{u}_l|\mathbf{x}_l) r_k\right\} = 0$ for $l > k$ which is straightforward to verify. This allows two variations of the previous algorithm which are known as the policy gradient theorem [16]

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

or G(PO)MD [27]

$$\mathbf{g}_{\text{GMDP}} = \left\langle \sum_{l=0}^{H}\left(\sum_{k=0}^{l} \nabla_\theta \log \pi_\theta(\mathbf{u}_k|\mathbf{x}_k)\right)(a_l r_l - b_l)\right\rangle.$$

While these algorithms *look* different, they are *exactly equivalent* in their gradient estimate[3], i.e., $\mathbf{g}_{\text{PGT}} = \mathbf{g}_{\text{GMPD}}$, and have been derived previously in the simulation optimization community [29]. An implementation of this algorithm is shown together with the optimal baseline in Table III.

However, in order to clarify the relationship to [16], [27], we note that the term $\sum_{l=k}^{H} a_l r_l$ in the policy gradient theorem is equivalent to a monte-carlo estimate of the value function $Q_{k:H}^\theta(\mathbf{x}_k, \mathbf{u}_k) = E\left\{\sum_{l=k}^{H} a_l r_l \middle| \mathbf{x}_k, \mathbf{u}_k\right\}$ and the term $\sum_{k=0}^{l} \nabla_\theta \log \pi_\theta(\mathbf{u}_k|\mathbf{x}_k)$ becomes the log-derivative of the distribution of states $\mu_\theta^k(\mathbf{x}_k)$ at step k in expectation, i.e., $\nabla_\theta \log \mu_\theta(\mathbf{x}_k) = E\left\{\sum_{k=0}^{k} \nabla_\theta \log \pi_\theta(\mathbf{u}_k|\mathbf{x}_k) \middle| \mathbf{x}_k\right\}$. When either of these two constructs can be easily obtained by derivation or estimation, the variance of the gradient can be reduced significantly.

Without a formal derivation of it, the policy gradient theorem has been applied using estimated value functions $Q_{k:H}^\theta(\mathbf{x}_k, \mathbf{u}_k)$ instead of the term $\sum_{l=k}^{H} a_l r_l$ and a baseline $b_k = V_{k:H}^\theta(\mathbf{x}_k) = E\left\{\sum_{l=k}^{H} a_l r_l \middle| \mathbf{x}_k\right\}$ [9], [30].

2) Optimal Baselines: Above, we have already introduced the concept of a baseline which can decrease the variance of a policy gradient estimate by orders of magnitude. Thus, an optimal selection of such a baseline is essential. An optimal baseline minimizes the variance $\sigma_h^2 = \text{Var}\{g_h\}$ of each element g_h of the gradient $\mathbf{g}$ *without* biasing the gradient estimate, i.e., violating $E\{\mathbf{g}\} = \nabla_\theta J$. This can be phrased as having a seperate baseline b^h for every element of the gradient[4]. Due to the requirement of unbiasedness of the gradient estimate, we have $\sigma_h^2 = E\left\{g_h^2\right\} - \left(\nabla_{\theta_h} J\right)^2$ and due to $\min_{b_h} \sigma_h^2 \geq E\left\{\min_{b_h} g_h^2\right\} - \left(\nabla_{\theta_h} J\right)^2$, the optimal baseline for each gradient element g_h can always be given

[3]Note that [27] additionally add an eligibility trick for reweighting trajectory pieces. This trick can be highly dangerous in robotics as can be demonstrated that already in linear-quadratic regulation, this trick can result into divergence as the optimial policy for small planning horizons (i.e., small eligibility rates) is often instable.

[4]A single baseline for all parameters can also be obtained and is more common in the reinforcement learning literature [26], [31]–[35]. However, such a baseline is of course suboptimal.

by

$$b^h = \frac{\left\langle \left(\sum_{k=0}^{H} \nabla_{\theta_h} \log \pi_\theta \left(\mathbf{u}_k \left| \mathbf{x}_k \right. \right) \right)^2 \sum_{l=0}^{H} a_l r_l \right\rangle}{\left\langle \left(\sum_{k=0}^{H} \nabla_{\theta_h} \log \pi_\theta \left(\mathbf{u}_k \left| \mathbf{x}_k \right. \right) \right)^2 \right\rangle}$$

for the general likelihood ratio gradient estimator, i.e., Episodic REINFORCE. The algorithmic form of the optimal baseline is shown in Table II in line 4. If the sums in the baselines are modified appropriately, we can obtain the optimal baseline for the policy gradient theorem or G(PO)MPD. We only show G(PO)MDP in this paper in Table III as the policy gradient theorem is numerically equivalent.

The optimal baseline which does not bias the gradient in Episodic REINFORCE can only be a single number for all trajectories and in G(PO)MPD it can also depend on the time-step [36]. However, in the policy gradient theorem it can depend on the current state and, therefore, if a good parameterization for the baseline is known, e.g., in a generalized linear form $b\left(\mathbf{x}_k\right) = \phi\left(\mathbf{x}_k\right)^T \omega$, this can significantly improve the gradient estimation process. However, the selection of the basis functions $\phi\left(\mathbf{x}_k\right)$ can be difficult and often impractical in robotics. See [26], [31]–[35] for more information on this topic.

C. Natural Actor-Critic Approaches

One of the main reasons for using policy gradient methods is that we intend to do just a small change $\boldsymbol{\Delta}\theta$ to the policy π_θ while improving the policy. However, the meaning of small is ambiguous. When using the Euclidian metric of $\sqrt{\boldsymbol{\Delta}\theta^T \boldsymbol{\Delta}\theta}$, 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 [37]. This 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., $p_\theta\left(\tau\right)$ and $p_{\theta+\boldsymbol{\Delta}\theta}\left(\tau\right)$) have been suggested, e.g., the Kullback-Leibler divergence[5] $d_{\mathrm{KL}}\left(p_\theta, p_{\theta+\boldsymbol{\Delta}\theta}\right)$, the Hellinger distance d_{HD} and others [39]. Many of these distances (e.g., the previously mentioned ones) can be approximated by the same second order Taylor expansion, i.e., by

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

where $\mathbf{F}_\theta = \int_{\mathbb{T}} p_\theta\left(\tau\right) \nabla \log p_\theta\left(\tau\right) \nabla \log p_\theta\left(\tau\right)^T d\tau = \left\langle \nabla \log p_\theta\left(\tau\right) \nabla \log p_\theta\left(\tau\right)^T \right\rangle$ is known as the Fisher-information matrix. Let us now assume that we restrict the change of our policy to the length of our step-size α_n, i.e., we have a restricted step-size gradient descent approach well-known in the optimization literature [40], and given by

$$\boldsymbol{\Delta}\theta = \mathrm{argmax}_{\boldsymbol{\Delta}\tilde{\theta}} \frac{\alpha_n \boldsymbol{\Delta}\tilde{\theta}^T \nabla_\theta J}{\boldsymbol{\Delta}\tilde{\theta}^T \mathbf{F}_\theta \ \boldsymbol{\Delta}\tilde{\theta}} = \alpha_n \mathbf{F}_\theta^{-1} \nabla_\theta J,$$

where $\nabla_\theta J$ denotes the 'vanilla' policy gradient from Section II-B. This update step can be interpreted as follows: determine the maximal improvement $\boldsymbol{\Delta}\tilde{\theta}$ of the policy for a constant fixed change of the policy $\boldsymbol{\Delta}\tilde{\theta}^T \mathbf{F}_\theta \boldsymbol{\Delta}\tilde{\theta}$.

This type of approach is known as Natural Policy Gradients and has its separate origin in supervised learning [41]. It was first suggested in the context of reinforcement learning by Kakade [37] and has been explored in greater depth in [7], [28], [36], [42]. The strongest theoretical advantage of this approach is that its performance no longer depends on the parameterization of the policy and it is therefore safe to use for arbitrary policies[6]. In practice, the learning process converges significantly faster in most practical cases.

1) Episodic Natural Actor-Critic: One of the fastest general algorithms for estimating natural policy gradients which does not need complex parameterized baselines is

[5]While being 'the natural way to think about closeness in probability distributions' [38], this measure is technically not a metric as it is not commutative.

[6]There is a variety of interesting properties to the natural policy gradient methods which are explored in [7].

TABLE III

SPECIALIZED LIKELIHOOD RATIO POLICY GRADIENT ESTIMATOR "G(PO)MDP"/POLICY GRADIENT WITH AN OPTIMAL BASELINE.

input: policy parameterization θ_h.

1 **repeat**
2 perform trials and obtain $\mathbf{x}_{0:H}, \mathbf{u}_{0:H}, r_{0:H}$
3 **for each** gradient element g_h
4 **for each** time step k
estimate baseline for time step k by
$$b_k^h = \frac{\left\langle \left(\sum_{\kappa=0}^{k} \nabla_{\theta_h} \log \pi_\theta(\mathbf{u}_\kappa|\mathbf{x}_\kappa) \right)^2 a_k r_k \right\rangle}{\left\langle \left(\sum_{\kappa=0}^{k} \nabla_{\theta_h} \log \pi_\theta(\mathbf{u}_\kappa|\mathbf{x}_\kappa) \right)^2 \right\rangle}$$
5 **end for**.
6 estimate the gradient element
$$g_h = \left\langle \sum_{l=0}^{H} \left(\sum_{k=0}^{l} \nabla_{\theta_h} \log \pi_\theta \left(\mathbf{u}_k \left| \mathbf{x}_k \right. \right) \right) \left(a_l r_l - b_l^h \right) \right\rangle.$$
7 **end for**.
8 **until** gradient estimate $\mathbf{g}_{\mathrm{FD}} = [g_1, \ldots, g_h]$ converged.

return: gradient estimate $\mathbf{g}_{\mathrm{FD}} = [g_1, \ldots, g_h]$.

TABLE IV

EPISODIC NATURAL ACTOR CRITIC

input: policy parameterization θ_h.

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

return: gradient estimate $\mathbf{g}_{\mathrm{NG}} = [g_1, \ldots, g_h]$.

the episodic natural actor critic. This algorithm, originally derived in [7], [28], [36], 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 $\boldsymbol{\Delta}\hat{\theta}$ given by

$$\begin{aligned}\boldsymbol{\Sigma} &= \mathrm{Cov}\left\{\boldsymbol{\Delta}\hat{\theta}\right\}_{\mathbf{F}_\theta} \\ &= E\left\{\left(\boldsymbol{\Delta}\hat{\theta} - \mathbf{F}_\theta^{-1}\mathbf{g}_{\mathrm{LR}}\left(b\right)\right)^T \mathbf{F}_\theta \left(\boldsymbol{\Delta}\hat{\theta} - \mathbf{F}_\theta^{-1}\mathbf{g}_{\mathrm{LR}}\left(b\right)\right)\right\},\end{aligned}$$

with $\mathbf{g}_{\mathrm{LR}}\left(b\right) = \left\langle \nabla \log p_\theta\left(\tau\right)\left(r\left(\tau\right) - b\right)\right\rangle$ being the REINFORCE gradient with baseline b. As outlined in [7], [28], [36], it can be shown that the minimum-variance unbiased natural gradient estimator can be determined as shown in Table IV.

2) Episodic Natural Actor Critic with a Time-Variant Baseline: The episodic natural actor critic described in the previous section suffers from drawback: it does not make use of intermediate data just like REINFORCE. For policy gradients, the way out was G(PO)MDP which left out terms which would average out in expectation. In the same manner, we can make the argument for a time-dependent baseline which then allows us to reformulate the Episodic Natural Actor Critic. This results in the algorithm shown in Table V. The advantage of this type of algorithms is two-fold: the variance of the gradient estimate is often lower and it can take time-variant rewards significantly better into account.

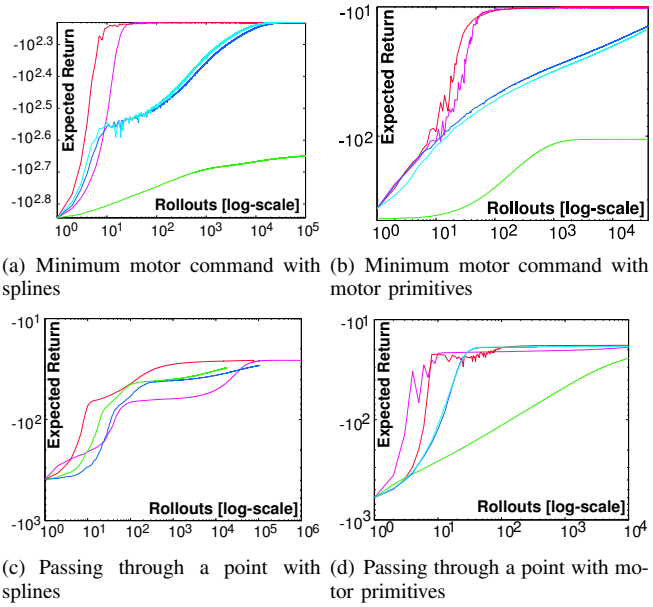
III. Experiments & Results

In the previous section, we outlined the five first-order, model-free policy gradient algorithms which are most relevant for robotics (further ones exist but are do not scale into high-dimensional robot domains). In this section, we will demonstrate how these different algorithms compare in practice in different areas relevant to robotics. For this pupose, we will show experiments on both simulated plants as well as on real robots and we will compare the algorithms for the optimization of control laws and for learning of motor skills.

TABLE V

Episodic Natural Actor Critic with a Time-Variant Baseline

input: policy parameterization θ_h.	
1	**repeat**
2	perform M trials and obtain $\mathbf{x}_{0:H}, \mathbf{u}_{0:H}, r_{0:H}$ for each trial. Obtain the sufficient statistics
3	Policy derivatives $\psi_k = \nabla_\theta \log \pi_\theta\left(\mathbf{u}_k \left|\mathbf{x}_k\right.\right)$.
4	Fisher matrix $\mathbf{F}_\theta = \left\langle \sum_{k=0}^{H}\left(\sum_{l=0}^{k}\psi_l\right)\psi_k^T\right\rangle$. Vanilla gradient $\mathbf{g} = \left\langle \sum_{k=0}^{H}\left(\sum_{l=0}^{k}\psi_l\right)a_k r_k\right\rangle$,
5	Eligibility matrix $\boldsymbol{\Phi} = [\phi_1, \phi_2, \ldots, \phi_K]$ with $\phi_h = \left\langle\left(\sum_{k=0}^{h}\psi_k\right)\right\rangle$.
6	Average reward vector $\bar{\mathbf{r}} = [\bar{r}_1, \bar{r}_2, \ldots, \bar{r}_K]$ with $\bar{r}_h = \langle a_h r_h\rangle$. Obtain natural gradient by computing
7	Baseline $\mathbf{b} = \mathbf{Q}\left(\bar{\mathbf{r}} - \boldsymbol{\Phi}^T\mathbf{F}_\theta^{-1}\mathbf{g}\right)$ with $\mathbf{Q} = M^{-1}\left(\mathbf{I}_K + \boldsymbol{\Phi}^T\left(M\mathbf{F}_\theta - \boldsymbol{\Phi}\boldsymbol{\Phi}^T\right)^{-1}\boldsymbol{\Phi}\right)$.
8	Natural gradient $\mathbf{g}_{\mathrm{NG}} = \mathbf{F}_\theta^{-1}\left(\mathbf{g} - \boldsymbol{\Phi}\mathbf{b}\right)$.
9	**until** gradient estimate $\mathbf{g}_{\mathrm{NG}} = [g_1, \ldots, g_h]$ converged.
return: gradient estimate $\mathbf{g}_{\mathrm{NG}} = [g_1, \ldots, g_h]$.	



(a) Minimum motor command with splines (b) Minimum motor command with motor primitives

(c) Passing through a point with splines (d) Passing through a point with motor primitives

Fig. 1. This figure shows different experiments with motor task learning. In (a,b), we see how the learning system creates minimum motor command goal-achieving plans using both (a) splines and (b) motor primitives. For this problem, the natural actor-critic methods beat all other methods by several orders of magnitude. In (c,d), the plan has to achieve an intermediary goal. While the natural actor-critic methods still outperform previous methods, the gap is lower as the learning problem is easier. Note that these are double logarithmic plots.

A. Comparing Policy Gradient Methods

Initially, we compare the different policy gradient methods in motor primitive planning tasks using both spline-based and dynamical system based desired trajectories. In Figure 1 (a) and (b), we show a comparison of the presented algorithms for a simple, single DOF task with a reward of $r_k(x_{0:N}, u_{0:N}) = \sum_{i=0}^{N} c_1 \dot{q}_{d,k,i}^2 + c_2(q_{d;k;N} - g_k)^2$; where $c_1 = 1$, $c_2 = 1000$ for both splines and dynamic motor primitives. In Figure 1 (c) and (d) we show the same with an additional punishment term for going through a intermediate point p_F at time F, i.e., $r_k(x_{0:N}, u_{0:N}) = \sum_{i=0}^{N} \tilde{c}_1 \dot{q}_{d,k,i}^2 + \tilde{c}_2(q_{d;k;N} - g_k)^2 + \tilde{c}_2(q_{d;F;N} - p_F)^2$. It is quite clear from the results that the natural actor-critic methods outperform both the vanilla policy gradient methods as well as the likelihood ratio methods. Finite difference gradient methods behave differently from the likelihood ratio methods as there is no stochasticity in the system, resulting in a cleaner gradient but also in local minima not present for likelihood ratio methods where the exploratory actions are stochastic. From this comparison, we can conclude that natural actor-critic methods are the best suited for motor primitive learning.

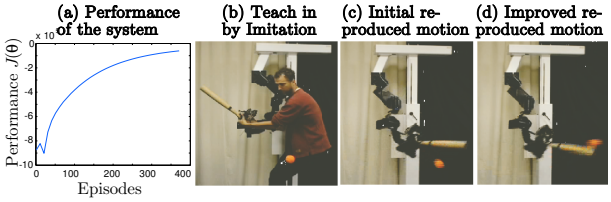


Fig. 2. This figure shows (a) the performance of a baseball swing task when using the motor primitives for learning. In (b), the learning system is initialized by imitation learning, in (c) it is initially failing at reproducing the motor behavior, and (d) after several hundred episodes exhibiting a nicely learned batting.

B. Robot Application: Motor Primitive Learning for Baseball

We also evaluated the same setup in a challenging robot task, i.e., the planning of these motor primitives for a seven DOF robot task using our SARCOS Master Arm. The task of the robot is to hit the ball properly so that it flies as far as possible; this game is also known as T-Ball. The state of the robot is given by its joint angles and velocities while the action are the joint accelerations. The reward is extracted using color segment tracking with a NewtonLabs vision system. Initially, we teach a rudimentary stroke by supervised learning as can be seen in Figure 2 (b); however, it fails to reproduce the behavior as shown in (c); subsequently, we improve the performance using the episodic Natural Actor-Critic which yields the performance shown in (a) and the behavior in (d). After approximately 200-300 trials, the ball can be hit properly by the robot.

IV. Conclusion

We have presented an extensive survey of policy gradient methods. While some developments needed to be omitted as they are only applicable for very low-dimensional state-spaces, this paper represents the state of the art in policy gradient methods and can deliver a solid base for future applications of policy gradient methods in robotics. 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 paper and practical algorithms are given. The experiments presented here show that the time-variant episodic natural actor critic is the preferred method when applicable; however, if a policy cannot be differentiated with respect to its parameters, the finite difference methods may be the only method applicable. The example of motor primitive learning for baseball underlines the efficiency of natural gradient methods.

Appendix

A. Motor Primitive Equations

The motor primitives from [1], [2] in their most recent reformualtion are given by a canonical system

$$\tau^{-1}\dot{v} = \alpha_v\left(\beta_v\left(g - x\right) - v\right), \tag{3}$$

$$\tau^{-1}\dot{x} = v, \tag{4}$$

which represents the phase of the motor process. It has a goal g, a time constant τ and some parameters α_v,β_v which are chosen so that the system is stable. Additionally, we have a transformed system

$$\tau^{-1}\dot{z} = \alpha_z\left(\beta_z\left(s - x\right) - v\right) + f\left(x, v, g\right), \tag{5}$$

$$\tau^{-1}\dot{y} = z, \tag{6}$$

$$\tau^{-1}\dot{s} = \alpha_s\left(g - s\right), \tag{7}$$

which has the same time-constant τ as the canonical system, appropriately set parameters α_z,β_z,α_s, and a transformation function $f\left(x, v, g\right)$. The transformation function transforms the output of the canonical system so that the transformed system can represent complex nonlinear patterns and is given by

$$f\left(x, v, g\right) = \frac{\sum_{i=1}^{N}\psi_i\left(x\right)\theta_i v}{\sum_{i=1}^{N}\psi_i\left(x\right)}, \tag{8}$$

where θ_i are adjustable parameters and it has localization weights defined by

$$\psi_i\left(x\right) = \exp\left(-h_i\left(\frac{x - x_0}{g - x_0} - c_i\right)^2\right) \tag{9}$$

with offset x_0, centers c_i and width h_i.

References

[1] A. Ijspeert, J. Nakanishi, and S. Schaal, "Learning rhythmic movements by demonstration using nonlinear oscillators," pp. 958–963, 2002.

[2] ——, "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Information Processing Systems 15*. Cambridge, MA: MIT Press, 2003.

[3] H. Benbrahim and J. Franklin, "Biped dynamic walking using reinforcement learning," *Robotics and Autonomous Systems*, vol. 22, pp. 283–302, 1997.

[4] V. Gullapalli, J. Franklin, and H. Benbrahim, "Aquiring robot skills via reinforcement learning," *IEEE Control Systems*, vol. -, no. 39, 1994.

[5] H. Benbrahim, J. Doleac, J. Franklin, and O. Selfridge, "Real-time learning: A ball on a beam," in *Proceedings of the 1992 International Joint Conference on Neural Networks*, Baltimore, MD, 1992.

[6] H. Kimura and S. Kobayashi, "Reinforcement learning for continuous action using stochastic gradient ascent," in *The 5th International Conference on Intelligent Autonomous Systems*, 1998.

[7] J. Peters, S. Vijayakumar, and S. Schaal, "Natural actor-critic," in *Proceedings of the European Machine Learning Conference (ECML)*, 2005.

[8] N. Mitsunaga, C. Smith, T. Kanda, H. Ishiguro, and N. Hagita, "Robot behavior adaptation for human-robot interaction based on policy gradient reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, 2005, pp. 1594–1601.

[9] H. Kimura and S. Kobayashi, "Reinforcement learning for locomotion of a two-linked robot arm," *Proceedings of the 6th Europian Workshop on Learning Robots EWLR-6*, pp. 144–153, 1997.

[10] M. Sato, Y. Nakamura, and S. Ishii, "Reinforcement learning for biped locomotion," in *International Conference on Artificial Neural Networks (ICANN)*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2002, pp. 777–782.

[11] 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.

[12] 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.

[13] R. Tedrake, T. W. Zhang, and H. S. Seung, "Learning to walk in 20 minutes," in *Proceedings of the Fourteenth Yale Workshop on Adaptive and Learning Systems*, 2005.

[14] T. Mori, Y. Nakamura, M. aki Sato, and S. Ishii, "Reinforcement learning for cpg-driven biped robot," in *AAAI 2004*, 2004, pp. 623–630.
[15] S. I. Yutaka Nakamura, Takeshi Mori, "Natural policy gradient reinforcement learning for a cpg control of a biped robot," in *PPSN 2004*, 2004, pp. 972–981.
[16] R. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*, vol. 12, no. 22, 2000.
[17] L. Sciavicco and B. Siciliano, *Modelling and Control of Robot Manipulators*. Springer Verlag, 2000.
[18] H. Miyamoto, S. Schaal, F. Gandolfo, Y. Koike, R. Osu, E. Nakano, Y. Wada, and M. Kawato, "A kendama learning robot based on bi-directional theory," *Neural Networks*, vol. 9, no. 8, pp. 1281–1302, 1996.
[19] D. H. Jacobson and D. Q. Mayne, *Differential Dynamic Programming*. New York: American Elsevier Publishing Company, Inc., 1970.
[20] P. Dyer and S. R. McReynolds, *The Computation and Theory of Optimal Control.* New York: Academic Press, 1970.
[21] L. Hasdorff, *Gradient optimization and nonlinear control*. John Wiley & Sons, 1976.
[22] J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Hoboken, NJ: Wiley, 2003.
[23] P. Glynn, "Likelihood ratio gradient estimation: an overview," in *Proceedings of the 1987 Winter Simulation Conference*, Atlanta, GA, 1987, pp. 366–375.
[24] A. Y. Ng and M. Jordan, "Pegasus: A policy search method for large mdps and pomdps," in *Uncertainty in Artificial Intelligence, Proceedings of the Sixteenth Conference*, 2000.
[25] V. Aleksandrov, V. Sysoyev, and V. Shemeneva, "Stochastic optimization," *Engineering Cybernetics*, vol. 5, pp. 11–16, 1968.
[26] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 23, 1992.
[27] J. Baxter and P. Bartlett, "Direct gradient-based reinforcement learning," *Journal of Artificial Intelligence Research*, 1999. [Online]. Available: citeseer.nj.nec.com/baxter99direct.html
[28] J. Peters, S. Vijayakumar, and S. Schaal, "Reinforcement learning for humanoid robotics," in *IEEE/RSJ International Conference on Humanoid Robotics*, 2003.
[29] P. Glynn, "Likelihood ratio gradient estimation for stochastic systems," *Communications of the ACM*, vol. 33, no. 10, pp. 75–84, October 1990.
[30] V. Gullapalli, "Associative reinforcement learning of real-value functions," *SMC*, vol. -, no. -, 1991.
[31] L. Weaver and N. Tao, "The optimal reward baseline for gradient-based reinforcement learning," *Uncertainty in Artificial Intelligence: Proceedings of the Seventeenth Conference*, vol. 17, no. 29, 2001.
[32] E. Greensmith, P. L. Bartlett, and J. Baxter, "Variance reduction techniques for gradient estimates in reinforcement learning," *Journal of Machine Learning Research*, vol. 5, pp. 1471–1530, Nov. 2004.
[33] L. Weaver and N. Tao, "The variance minimizing constant reward baseline for gradient-based reinforcement learning," *Technical Report ANU*, vol. -, no. 30, 2001.
[34] G. Lawrence, N. Cowan, and S. Russell, "Efficient gradient estimation for motor control learning," in *Proc. UAI-03*, Acapulco, Mexico, 2003.
[35] E. Greensmith, P. Bartlett, and J. Baxter, "Variance reduction techniques for gradient estimates in reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 14, no. 34, 2001.
[36] J. Peters, "Machine learning of motor skills for robotics," Ph.D. Thesis Proposal / USC Technical Report, Tech. Rep., 2005.
[37] S. Kakade, "A natural policy gradient," in *Advances in Neural Information Processing Systems*, vol. 14, no. 26, 2001.
[38] V. Balasubramanian, "Statistical inference, occam's razor, and statistical mechanics on the space of probability distributions," *Neural Computation*, vol. 9, no. 2, pp. 349–368, 1997.
[39] F. Su and A. Gibbs, "On choosing and bounding probability metrics," *International Statistical Review*, vol. 70, no. 3, pp. 419–435, 2002.
[40] R. Fletcher and R. Fletcher, *Practical Methods of Optimization*. John Wiley & Sons, 2000.
[41] S. Amari, "Natural gradient works efficiently in learning," *Neural Computation*, vol. 10, 1998.
[42] J. Bagnell and J. Schneider, "Covariant policy search," *International Joint article on Artificial Intelligence*, 2003.