Peters J, Vijayakumar S, Schaal S (2003) Reinforcement learning for humanoid robotics. In: Humanoids2003, Third IEEE-RAS International Conference on Humanoid Robots, Karlsruhe, Germany, Sept.29-30

Reinforcement Learning for Humanoid Robotics

Jan Peters, Sethu Vijayakumar, Stefan Schaal

Computational Learning and Motor Control Laboratory
Computer Science & Neuroscience, University of Southern California
3461 Watt Way – HNB 103, Los Angeles, CA 90089-2520, USA
&
ATR Computational Neuroscience Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan
http://www-clmc.usc.edu
{jrpeters,sethu,sschaal}@usc.edu

Abstract. Reinforcement learning offers one of the most general framework to take traditional robotics towards true autonomy and versatility. However, applying reinforcement learning to high dimensional movement systems like humanoid robots remains an unsolved problem. In this paper, we discuss different approaches of reinforcement learning in terms of their applicability in humanoid robotics. Methods can be coarsely classified into three different categories, i.e., greedy methods, 'vanilla' policy gradient methods, and natural gradient methods. We discuss that greedy methods are not likely to scale into the domain humanoid robotics as they are problematic when used with function approximation. 'Vanilla' policy gradient methods on the other hand have been successfully applied on real-world robots including at least one humanoid robot [3]. We demonstrate that these methods can be significantly improved using the natural policy gradient instead of the regular policy gradient. A derivation of the natural policy gradient is provided, proving that the average policy gradient of Kakade [10] is indeed the true natural gradient. A general algorithm for estimating the natural gradient, the Natural Actor-Critic algorithm, is introduced. This algorithm converges to the nearest local minimum of the cost function with respect to the Fisher information metric under suitable conditions. The algorithm outperforms non-natural policy gradients by far in a cart-pole balancing evaluation, and for learning nonlinear dynamic motor primitives for humanoid robot control. It offers a promising route for the development of reinforcement learning for truly high-dimensionally continuous state-action systems.

1 Introduction

In spite of tremendous leaps in computing power as well as major advances in the development of materials, motors, power supplies and sensors, we still lack the ability to create a humanoid robotic system that even comes close to a similar level of robustness, versatility and adaptability as biological systems. Classical robotics and also the more recent wave of humanoid and toy robots still rely heavily on teleoperation or fixed "pre-canned" behavior based control with very little autonomous ability to react to the environment. Among the key missing elements is the ability to create control systems that can deal with a large movement repertoire, variable speeds, constraints and most importantly, uncertainty in the real-world environment in a fast, reactive manner.

One approach of departing from teleoperation and manual 'hard coding' of behaviors is by learning from experience and creating appropriate adaptive control systems. A rather general approach to learning control is the framework of 'reinforcement learning'. Reinforcement learning typically requires an unambiguous representation of states and actions and the existence of a scalar reward function. For a given state, the most traditional of these implementations would take an action, observe a reward, update the value function, and select as the new control output the action with the highest expected value in each state (for a greedy policy evaluation). Updating of value function and controls is repeated until convergence of the value function and/or the policy. This procedure is usually summarized under "value update – policy improvement" iterations.

The reinforcement learning paradigm described above has been successfully implemented for many well-defined, low dimensional and discrete problems [14] and has also yielded a variety of impressive applications in rather complex domains in the last decade. These applications range from backgammon playing on grandmaster level to robotic toy applications such as cart-pole or acrobot swing-ups. However, various pitfalls have been encountered when trying to scale up these methods to high dimensional, continuous control problems, as typically faced in the domain of humanoid robotics. The goal of this paper is to discuss the state-of-the-art in reinforcement learning and investigate how it may be possible to make reinforcement learning useful for humanoid robotics. Initially, in Section 2.2, we will focus on traditional value function based approaches and policy gradient methods and discuss their shortcomings in the context of humanoid control. In Section 2.2, we will motivate and formulate a novel policy gradient based reinforcement learning method, the natural policy gradient, and in Section 3 derive an efficient Natural Actor-Critic algorithm that can address various of the current shortcomings of reinforcement learning for humanoid robotics. This algorithm seems to be a promising candidate for reinforcement learning to become applicable in for complex movement systems like humanoids.

2 Reinforcement Learning Approaches for Humanoid Robotics

Humanoid Robots and, in general, high dimensional movement systems have additional demands and requirements as compared to the conventional control problems in which reinforcement learning approaches have been tried. From a purely algorithmic perspective, the learning method has to make efficient use of data, scale to high dimensional state and action spaces and be computationally cheap in order to work online. In addition, methods have to work in continuous state and action space and also be easily parametrized through function approximation techniques [20, 21]. The aim of our research is to make progress towards fulfilling all of the above mentioned requirements and evaluate reinforcement learning methods according to which are the most promising for robotics.

2.1 Policy Evaluation

As mentioned in the introduction, most reinforcement learning methods can be described in terms of two steps: the policy evaluation and the policy improvement step – we will later discuss a few exceptions, such as direct policy learning, which can be seen as special cases of this iterative loop. In *policy evaluation*, the prospect of a motor command $\boldsymbol{u} \in \mathbb{U}$ for a given state $\boldsymbol{x} \in \mathbb{X}$ is evaluated. This step is usually performed by computing the action-value function:

$$Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) = E\left\{ \sum_{t=0}^{\infty} \gamma^t r_t \middle| \boldsymbol{x}_0 = \boldsymbol{x}, \boldsymbol{u}_0 = \boldsymbol{u} \right\} \tag{1}$$

where the superscript π denotes the current (fixed) policy from which actions are determined in each state, most generally formulated as a conditional probability $p(\boldsymbol{u}|\boldsymbol{x}) = \pi(\boldsymbol{u}|\boldsymbol{x})$, and γ is a discount factor (with γ in $[0, 1]$) that models the reduced trust in the reward r_t with increasing value of the discrete time t, i.e., the further a reward lies in the future. Alternatively, by averaging over all actions in each state, the value function can be formulated solely as a function of state as:

$$V^{\pi}(\boldsymbol{x}) = E\left\{ \sum_{t=0}^{\infty} \gamma^t r_t \middle| \boldsymbol{x}_0 = \boldsymbol{x} \right\}. \tag{2}$$

As both motor commands and states are very high-dimensional in complex movement systems, finding the value function for a given policy is a non-trivial problem. As discussed in [14], *Monte Carlo methods*, i.e., methods which evaluate the expectation operator above directly by averaging over multiple trajectory rollouts, are usually too inefficient, even for smaller problems, while *dynamic programming* solution methods, based on solving the Bellman equations

$$Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) = r(\boldsymbol{x}, \boldsymbol{u}) + \gamma \int_{\mathbb{X}} p(\boldsymbol{x}'|\boldsymbol{x}, \boldsymbol{u}) V^{\pi}(\boldsymbol{x}') d\boldsymbol{x}', \tag{3}$$

$$V^{\pi}(\boldsymbol{x}) = \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) Q^{\pi}(\boldsymbol{x}, \boldsymbol{u})\, d\boldsymbol{u}, \tag{4}$$

can only be computed numerically or analytically in few restricted cases (e.g., discrete state-action spaces of low dimensionality or linear quadratic regulator problems). In between Monte Carlo methods and dynamic programming lies a third approach to policy evaluation, *temporal difference learning* (TD). In this approach, the rewards and the Bellman equation are used to obtain the temporal error in the value function which can be expressed by

$$\delta^{\pi}(\boldsymbol{x}_t, \boldsymbol{u}_t) = E_{\boldsymbol{u}_{t+1}}\{r(\boldsymbol{x}_t, \boldsymbol{u}_t) + \gamma Q^{\pi}(\boldsymbol{x}_{t+1}, \boldsymbol{u}_{t+1}) - Q^{\pi}(\boldsymbol{x}_t, \boldsymbol{u}_t)\}, \tag{5}$$

$$= r(\boldsymbol{x}_t, \boldsymbol{u}_t) + \gamma V^{\pi}(\boldsymbol{x}_{t+1}) - Q^{\pi}(\boldsymbol{x}_t, \boldsymbol{u}_t), \tag{6}$$

which can serve as an update for $Q^{\pi}(\boldsymbol{x}_t, \boldsymbol{u}_t)$, and $V^{\pi}(\boldsymbol{x}_t)$. Learning algorithms using this temporal difference error have been shown to be rather efficient [14]. The convergence with probability one can be proven for several algorithms such as TD(λ) in the discrete state-action case [14], and LSTD(λ), a method that approximates the value function with a function approximator that is linear in its parameters [12]. Furthermore, Benbrahim and Franklin showed the potential of these methods to scale into the domain of humanoid robotics [3].

2.2 Policy Improvement

Knowing the approximate value of each motor command $\boldsymbol{u}$ in each state $\boldsymbol{x}$ for a given policy from the rather well-developed recipes for policy evaluation in the previous section leads to the question of how a policy can be optimized – a step known as *policy improvement*. In the following, we will discuss which policy improvement method exist and how they scale, particularly in the context of humanoid robotics. For this purpose, we classify methods into *greedy policy improvements* [14], the *'vanilla' policy gradient* approach [1, 23, 13, 24, 21, 22], and the *natural policy gradient* approach [10].

Greedy Policy Improvement. One of the early and most famous solutions to policy improvement originated in the 1960s due to Richard Bellman, who introduced the policy iteration framework [2]. Here, first a value function $Q^{\pi_i}(\boldsymbol{x}, \boldsymbol{u})$ of policy π_i is computed, and subsequently, an improved policy is derived by always taking the best motor command $\boldsymbol{u}^*$ for each state using the value function of the last policy, i.e., $Q^{\pi_i}(\boldsymbol{x}, \boldsymbol{u})$. For a deterministic policy, this greedy update can be formalized as:

$$\pi_{i+1}(\boldsymbol{x}, \boldsymbol{u}) = \begin{cases} 1 \text{ if } \boldsymbol{u} = \boldsymbol{u}^* = \operatorname{argmax}_{\tilde{\boldsymbol{u}}} Q^{\pi_i}(\boldsymbol{x}, \tilde{\boldsymbol{u}}), \\ 0 \text{ if otherwise.} \end{cases} \tag{7}$$

In many cases, an ϵ-greedy policy is used which introduces a certain amount of random exploration, thus making the policy stochastic. The greedy action $\boldsymbol{u}^*$ is now taken with probability $\epsilon \in [0, 1]$, while all other actions can be drawn from the remaining probability of $1 - \epsilon$ uniformly distributed over all other actions. Algorithms which use data generated by either the current policy (i.e., on-policy methods) and/or different policies (i.e., off-policy) have been presented in the literature. When applied to discrete state-action domains (e.g., with a look-up table representation), and assuming that the value function has been approximated with perfect accuracy, a policy improvement can be guaranteed and the policy will converge to the optimal policy within a finite amount of policy evaluation – policy improvement steps.

While this approach has been rather successful in discrete problem domains, it comes with a major drawback in continuous domains or in usage with parameterized (or function approximation based) policy representations: in these cases the greedy "max" operator in (7) can destabilize the policy evaluation and improvement iterations. A small change or error in the value function approximation can translate into a large, discontinuous change in the policy without guarantee of a policy improvement[1]. This large change will in turn result in a large change in the value function estimate [19], often resulting in destabilizing dynamics. For these reasons, the traditional greedy policy improvement based approaches seem to be less likely to scale for continuous and high dimensional domains [1].

'Vanilla' Policy Gradients Improvements. In light of the problems that can be encountered by greedy policy improvement, *policy gradient* methods with small incremental steps in the change of the policy are an attractive alternative. Policy gradient methods differ from greedy methods in several respects. Policies are usually parameterized with a small number of meaningful parameters θ – such policies can be conceived of as motor primitives or nonlinear controllers in humanoid robotics. The optimized objective function is usually more compact, i.e., instead of having a function of state and action, only a single average expected return $J(\boldsymbol{\theta})$ is employed as an objective function:

$$J(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) r(\boldsymbol{x}, \boldsymbol{u}) d\boldsymbol{u} d\boldsymbol{x}, \tag{8}$$

where $r(\boldsymbol{x}, \boldsymbol{u})$ denotes the reward, and $d^{\pi}(\boldsymbol{x}) = (1-\gamma) \sum_{i=0}^{\infty} \gamma^i \Pr\{\boldsymbol{x}_i = \boldsymbol{x}\}$ denotes the discounted state distribution which becomes a stationary distribution for $\gamma \to 1$ – the discounted state distribution depends on the start state distribution, while the stationary distribution does not. The most straightforward approach of policy improvement is to follow the gradient in policy parameter space using steepest gradient ascent, i.e.,

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha \boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_i). \tag{9}$$

The central problem is to estimate the policy gradient $\boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. Early methods like (nonepisodic) REINFORCE [26] focused on optimizing the *immediate* reward by approximating the gradient as

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \boldsymbol{\nabla}_{\boldsymbol{\theta}} \pi(\boldsymbol{u}|\boldsymbol{x}) r(\boldsymbol{x}, \boldsymbol{u}) d\boldsymbol{u} d\boldsymbol{x}, \tag{10}$$

thus neglecting temporal credit assignment problems for future rewards by assuming $\boldsymbol{\nabla}_{\boldsymbol{\theta}} d^{\pi}(\boldsymbol{x}) \approx 0$. An alternative version, episodic REINFORCE, addressed the temporal credit assignment problem by performing roll-outs and taking into account only the final reward [26]. These methods were extended by Gullapalli

[1] In fact, the greedy improvement can make the policy worse. The only existing guarantee is that if the value function approximation has a maximal error ε, then the true value function of the new policy will fulfill $V^{\pi_{i+1}}(\boldsymbol{x}) \geq V^{\pi_i}(\boldsymbol{x}) - 2\gamma\varepsilon/(1-\gamma)$ after a greedy update for all statesm, indicating that the greedy step is not necessarily an improvement unless the maximal error ε is zero.

[13] who used the temporal difference error from an additionally estimated value function as reward signal for computing the policy gradient:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \nabla_{\boldsymbol{\theta}} \pi(\boldsymbol{u}|\boldsymbol{x}) \delta^{\pi}(\boldsymbol{x}, \boldsymbol{u}) d\boldsymbol{u} d\boldsymbol{x}. \tag{11}$$

In view of the recent results on the policy gradient theory, the above approach can indeed be shown to be the true gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ (cf. eq. (12) below).

The early work in policy gradients in the late 1980s and beginning 1990s is somehow surprising from the view of robotics. While lacking a complete mathematical development, researchers achieved impressive robotic applications using policy gradients. Gullapalli demonstrated that policy gradient methods can be used to learn fine manipulation control as it was required for performing the classical 'peg in a hole' task [13]; Benbrahim and Franklin showed that policy gradient methods can be used for learning biped walking with integrated trajectory generation and execution [3], and Ilg showed similar results for the related quadruped walking [15]. These early methods have yielded not only some of the most significant applications of reinforcement learning in robotics, but also probably the first using a real humanoid robot [3].

Since one of the major problems of greedy policy improvements arose from value function estimation and its use in policy updates, several researchers started estimating policy gradients using Monte Carlo roll-outs [23]. While these methods avoid problems of learning a value function, they have an increased variance in their gradient estimates as detailed in the work by Baxter et al. [23] [2]. Other researchers extended the work of Gullapalli [1, 23, 22, 24, 20, 21] leading to a hallmark in the policy gradient theory, the policy gradient theorem. This theorem proves that the true policy gradient can generally be estimated by

$$\nabla J(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \nabla_{\boldsymbol{\theta}} \pi(\boldsymbol{u}|\boldsymbol{x}) \left(Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) - b^{\pi}(\boldsymbol{x}) \right) d\boldsymbol{u} d\boldsymbol{x}, \tag{12}$$

where $b^{\pi}(\boldsymbol{x})$ denotes an arbitrary function of $\boldsymbol{x}$, often called a baseline. While in theory the baseline $b^{\pi}(\boldsymbol{x})$ is irrelevant as it does not introduce bias into the gradient estimate, it plays an important role in the derivation of algorithms as it can be used to minimize the estimate's variance.

A significant advance of policy gradient theory was introduced in [20], and independently in [21]. These authors demonstrated that in eq.(12), the action value function $Q^{\pi}(\boldsymbol{x}, \boldsymbol{u})$ can be replaced by an approximation $f^{\pi}_{w}(\boldsymbol{x}, \boldsymbol{u})$, parameterized by the vector $\boldsymbol{w}$, *without* affecting the unbiasedness of the gradient estimate. Such unbiasedness, however, requires a special, linear parameterization of $f^{\pi}_{w}(\boldsymbol{x}, \boldsymbol{u})$ in terms of

$$f^{\pi}_{w}(\boldsymbol{x}, \boldsymbol{u}) = \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x})^{T} \boldsymbol{w} \tag{13}$$

[2] In [23], a discrete Markov decision problem with three states and two actions with a discount factor of 0.95 required approximately 1000 samples in order to obtain a policy gradient estimate which had an angle with less than 45 degrees off the true gradient. For comparison: Gullapalli's estimator would have achieved this result in less than 20 samples.

This result is among the first that demonstrated how function approximation can be used safely in reinforcement learning, i.e., without the danger of divergence during learning, constituting an important step forward in order to scale up reinforcement learning to domains like humanoid robotics. Interestingly, this appealing progress comes at a price: since it is rather easy to verify $\int_{\mathbb{U}} \boldsymbol{\nabla}_{\boldsymbol{\theta}} \pi(\boldsymbol{u}|\boldsymbol{x}) d\boldsymbol{u} = 0$, it becomes obvious that this so called "compatible function approximator" in eq.(13) can only approximate an *advantage function*, i.e., the advantage of every action over the average performance in a state $\boldsymbol{x}$:

$$A(\boldsymbol{x}, \boldsymbol{u}) = Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) - V^{\pi}(\boldsymbol{x}). \tag{14}$$

Importantly, the advantage function *cannot* be learned without knowledge of the value function and hence, a TD-like bootstrapping using exclusively the compatible function approximator is impossible – the essence of TD (and the Bellman equation in general) is to compare the value $V^{\pi}(\boldsymbol{x})$ of two adjacent states, but his value has been subtracted in eq.(14). When re-evaluating Eq. (5) using the compatible function approximation as value function, the temporal difference error would become

$$\delta^{\pi}\left(\boldsymbol{x}_t, \boldsymbol{u}_t\right) = E_{\boldsymbol{u}_{t+1}}\{r\left(\boldsymbol{x}_t, \boldsymbol{u}_t\right) + \gamma f_w^{\pi}(\boldsymbol{x}_{t+1}, \boldsymbol{u}_{t+1}) - f_w^{\pi}(\boldsymbol{x}_t, \boldsymbol{u}_t)\}, \tag{15}$$

$$= r\left(\boldsymbol{x}_t, \boldsymbol{u}_t\right) - f_w^{\pi}(\boldsymbol{x}_t, \boldsymbol{u}_t). \tag{16}$$

This equation implies that the compatible function approximation would only learn the immidiate reward, i.e., $f_w^{\pi}(\boldsymbol{x}_t, \boldsymbol{u}_t) = r\left(\boldsymbol{x}_t, \boldsymbol{u}_t\right)$. TD($\lambda$) methods for learning $f_w^{\pi}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ as used in regular TD learning (as suggested by Konda & Tsitsiklis [21]), can therefore be shown to bebiased for $\lambda < 1$ as they do not address the temporal credit assignment problem appropriately.

Based on eqs.(12) and (13), a low variance estimate of the policy gradient can be derived by estimating $\boldsymbol{w}$ and an additional matrix $F(\boldsymbol{\theta})$:

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = F(\boldsymbol{\theta})\boldsymbol{w} \tag{17}$$

where

$$F(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x})^T d\boldsymbol{u} d\boldsymbol{x}. \tag{18}$$

As this method integrates over all actions – thus also called an 'all-action' algorithm – it does not require baselines [19] anymore. Furthermore, the all-action matrix $F(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) F(\boldsymbol{\theta}, \boldsymbol{x}) d\boldsymbol{x}$ is easier to approximate since

$$F(\boldsymbol{\theta}, \boldsymbol{x}) = \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x})^T d\boldsymbol{u} \tag{19}$$

can often even be evaluated analytically or, at least, without performing all actions since $\pi(\boldsymbol{u}|\boldsymbol{x})$ is a known function. This algorithm has first been suggested in [19], and appears to perform well in practice. However, it should be noted that when dealing with problems in high dimensional state spaces $\mathbb{X}$, we still require expensive roll-outs for estimating $F(\boldsymbol{\theta})$, which can become a severe bottleneck for applications on actual physical systems.

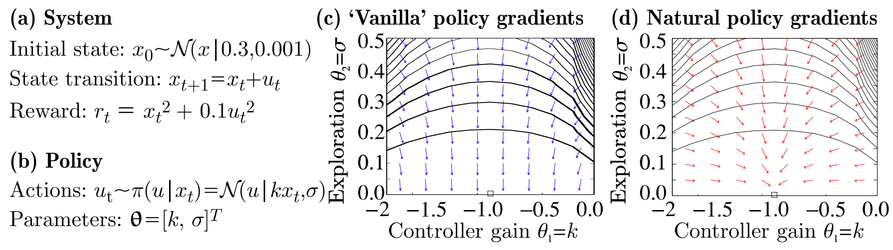

Fig. 1. When plotting the expected return landscape, the differences between 'vanilla' and natural policy gradients become apparent for simple examples (a-b). 'Vanilla' policy gradients (c) point onto a plateau at $\theta_2 = 0$, while natural policy gradients direct to the optimal solution (d). The gradients are normalized for improved visability.

Natural Policy Gradient Improvement. In supervised learning, a variety of significantly faster second order gradient optimization methods have been suggested. Among others, these algorithms include well-established algorithms such as Conjugate Gradient, Levenberg-Marquard, and Natural Gradients. It turns out that the latter has an interesting application to reinforcement learning, similar to natural gradient methods for supervised learning originally suggested by Amari [27]. If an objective function $L(\boldsymbol{\theta}) = \int_{\mathbb{X}} p(\boldsymbol{x}) l(\boldsymbol{x}, \boldsymbol{\theta}) d\boldsymbol{x}$ and its gradient $\boldsymbol{\nabla}_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$ are given, the steepest ascent in a Riemannian space with respect to the Fisher information metric $G(\boldsymbol{\theta})$ does not point along $\boldsymbol{\nabla}_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$ but along

$$\widetilde{\boldsymbol{\nabla}}_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = G^{-1}(\boldsymbol{\theta}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} L(\boldsymbol{\theta}). \tag{20}$$

The metric $G(\boldsymbol{\theta})$ is defined as

$$G(\boldsymbol{\theta}) = \int_{\mathbb{X}} p(\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log p(\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log p(\boldsymbol{x})^T d\boldsymbol{x}. \tag{21}$$

It is guaranteed that the angle between natural and ordinary gradient is never larger than ninety degrees, i.e., convergence to a local optimum is guaranteed. This result has surprising implications for policy gradients, when examining the meaning of the matrix $F(\boldsymbol{\theta})$ in eq.(18). Kakade [10] argued that $F(\boldsymbol{\theta}, \boldsymbol{x})$ is the point Fisher information matrix for state $\boldsymbol{x}$, and that $F(\boldsymbol{\theta})$, therefore, denotes the 'average Fisher information matrix'. However, going one step further, we can show here that $F(\boldsymbol{\theta})$ is indeed the true Fisher information matrix and does not have to be interpreted as the 'average' of the point Fisher information matrices. The proof of this key result is given in Section 3.1. This result does in fact imply that for reinforcement learning, the natural gradient (c.f. eq.(17)) can be computed as

$$\widetilde{\boldsymbol{\nabla}}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = G^{-1}(\boldsymbol{\theta}) F(\boldsymbol{\theta}) \boldsymbol{w} = \boldsymbol{w}, \tag{22}$$

since $F(\boldsymbol{\theta}) = G(\boldsymbol{\theta})$ based on the proof outlined in Section 3.1. An illustrative example comparing regular and natural policy gradients is given in Figure 1.

The significance of this result can be appreciated by considering a control problem in the domain of humanoid robotics. For a robot with 30 degrees of freedom (DOF), at least 60 dimensional state vectors and 30 dimensional action vectors need to be considered (assuming rigid body dynamics). An update of the symmetric 'All-Action Matrix' $F(\boldsymbol{\theta})$ would involve estimation of at least 4140 open parameters for gradient updates, which can be quite costly in the context of Monte-Carlo roll-outs. Since normally, significantly more data are required in order to obtain a good estimate of $F(\boldsymbol{\theta})$ than for a good estimate of $\boldsymbol{w}$, it makes a compelling case for avoiding estimation of the former. The theory of natural policy gradients and the result of eq.(22) allow us to get a better(natural) gradient estimate at lower cost. In the next section, we present an algorithm which is capable of exploiting this result and of estimating the natural gradient online.

3 Natural Actor-Critic

In this section, we present a novel reinforcement learning algorithm, the Natural Actor-Critic algorithm, which exploits the natural gradient formulation from the previous section. As the equivalence of the All-Action matrix and the Fisher information matrix has not been presented in the literature so far, we first provide an outline this proof. Then we derive the Natural Actor-Critic Algorithm for the general case, and as an empirical evaluation, we apply it to the classical 'cart-pole' problem, illustrating that this approach has the potential to scale well to continuous, high dimensional domains. For an application to humanoid robotics, we show that a special version of Natural Actor-Critic for episodic tasks can be applied for efficiently optimizing nonlinear motor primitives.

3.1 Proof of the Equivalence of Fisher Information Matric and All-Action Matrix

In Section 22, we explained that the all-action matrix $F(\boldsymbol{\theta})$ equals in general the Fisher information matrix $G(\boldsymbol{\theta})$. As this result has not been presented yet, we outline the proof in following paragraphs. In [25], we can find the well-known lemma that by differentiating $\int_{\mathbb{R}^n} p(\boldsymbol{x})d\boldsymbol{x} = 1$ twice with respect to the parameters $\boldsymbol{\theta}$, we can obtain

$$\int_{\mathbb{R}^n} p(\boldsymbol{x})\boldsymbol{\nabla}_{\boldsymbol{\theta}}^2 \log p(\boldsymbol{x})d\boldsymbol{x} = -\int_{\mathbb{R}^n} p(\boldsymbol{x})\boldsymbol{\nabla}_{\boldsymbol{\theta}} \log p(\boldsymbol{x})\boldsymbol{\nabla}_{\boldsymbol{\theta}} \log p(\boldsymbol{x})^T d\boldsymbol{x} \tag{23}$$

for any probability density function $p(\boldsymbol{x})$. Furthermore, we can rewrite the probability $p(\boldsymbol{\tau}_{0:n})$ of a rollout or trajectory $\boldsymbol{\tau}_{0:n} = [\boldsymbol{x}_0, \boldsymbol{u}_0, r_0, \boldsymbol{x}_1, \boldsymbol{u}_1, r_1, \ldots, \boldsymbol{x}_n, \boldsymbol{u}_n, r_n, \boldsymbol{x}_{n+1}]^T$ as

$$p\left(\boldsymbol{\tau}_{0:n}\right) = p\left(\boldsymbol{x}_0\right)\prod_{t=0}^{n} p\left(\boldsymbol{x}_{t+1}\left|\boldsymbol{x}_t, \boldsymbol{u}_t\right.\right)\pi\left(\boldsymbol{u}_t\left|\boldsymbol{x}_t\right.\right), \tag{24}$$

$$\Longrightarrow \boldsymbol{\nabla}^2_{\boldsymbol{\theta}}\log p\left(\boldsymbol{\tau}_{0:n}\right) = \sum_{t=0}^{n}\boldsymbol{\nabla}^2_{\boldsymbol{\theta}}\log\pi\left(\boldsymbol{u}_t\left|\boldsymbol{x}_t\right.\right).$$

Using Equations (23, 24), and the definition of the Fisher information matrix [27], we can determine the Fisher information matrix for the average reward case in sample notation, i.e,

$$\begin{aligned} G(\boldsymbol{\theta}) &= \lim_{n\to\infty}\frac{1}{n}E_{\boldsymbol{\tau}_{0:n}}\left\{\boldsymbol{\nabla}_{\boldsymbol{\theta}}\log p(\boldsymbol{\tau}_{0:n})\boldsymbol{\nabla}_{\boldsymbol{\theta}}\log p(\boldsymbol{\tau}_{0:n})^T\right\}, \\ &= -\lim_{n\to\infty}\frac{1}{n}E_{\boldsymbol{\tau}_{0:n}}\left\{\boldsymbol{\nabla}^2_{\boldsymbol{\theta}}\log p(\boldsymbol{\tau}_{0:n})\right\}, \\ &= -\lim_{n\to\infty}\frac{1}{n}E_{\boldsymbol{\tau}_{0:n}}\left\{\sum_{t=0}^{n}\boldsymbol{\nabla}^2_{\boldsymbol{\theta}}\log\pi\left(\boldsymbol{u}_t\left|\boldsymbol{x}_t\right.\right)\right\}, \\ &= -\int_{\mathbb{X}} d^{\pi}(\boldsymbol{x})\int_{\mathbb{U}}\pi(\boldsymbol{u}|\boldsymbol{x})\boldsymbol{\nabla}^2_{\boldsymbol{\theta}}\log\pi(\boldsymbol{u}|\boldsymbol{x})d\boldsymbol{u}d\boldsymbol{x}, \\ &= \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x})\int_{\mathbb{U}}\pi(\boldsymbol{u}|\boldsymbol{x})\boldsymbol{\nabla}_{\boldsymbol{\theta}}\log\pi(\boldsymbol{u}|\boldsymbol{x})\boldsymbol{\nabla}_{\boldsymbol{\theta}}\log\pi(\boldsymbol{u}|\boldsymbol{x})^T d\boldsymbol{u}d\boldsymbol{x}, \\ &= F(\boldsymbol{\theta}) \end{aligned} \tag{25}$$

This development proves that the all-action matrix is indeed the Fisher information matrix for the average reward case. For the discounted case, with a discount factor γ, we realize that we can rewrite the problem where the probability of a rollout is given by $p_\gamma(\boldsymbol{\tau}_{0:n}) = p(\boldsymbol{\tau}_{0:n})(\sum_{i=0}^{n}\gamma^i\mathbb{I}_{x_i,u_i})$. It is straightforward to show that $\boldsymbol{\nabla}^2_{\boldsymbol{\theta}}\log p\left(\boldsymbol{\tau}_{0:n}\right) = \boldsymbol{\nabla}^2_{\boldsymbol{\theta}}\log p_\gamma(\boldsymbol{\tau}_{0:n})$. This rewritten probability allows us to repeat the transformations in Equation 25, and show that again the all-action matrix equals the Fisher information matrix by

$$\begin{aligned} G(\boldsymbol{\theta}) &= \lim_{n\to\infty}(1-\gamma)E_{\boldsymbol{\tau}_{0:n}}\left\{\boldsymbol{\nabla}_{\boldsymbol{\theta}}\log p_\gamma(\boldsymbol{\tau}_{0:n})\boldsymbol{\nabla}_{\boldsymbol{\theta}}\log p_\gamma(\boldsymbol{\tau}_{0:n})^T\right\}, \\ &= -\lim_{n\to\infty}(1-\gamma)E_{\boldsymbol{\tau}_{0:n}}\left\{\boldsymbol{\nabla}^2_{\boldsymbol{\theta}}\log p(\boldsymbol{\tau}_{0:n})\right\}, \\ &= -\lim_{n\to\infty}(1-\gamma)E_{\boldsymbol{\tau}_{0:n}}\left\{\sum_{t=0}^{n}\boldsymbol{\nabla}^2_{\boldsymbol{\theta}}\log\pi\left(\boldsymbol{u}_t\left|\boldsymbol{x}_t\right.\right)\right\}, \\ &= -\int_{\mathbb{X}} d^{\pi}_{\gamma}(\boldsymbol{x})\int_{\mathbb{U}}\pi(\boldsymbol{u}|\boldsymbol{x})\boldsymbol{\nabla}^2_{\boldsymbol{\theta}}\log\pi(\boldsymbol{u}|\boldsymbol{x})d\boldsymbol{u}d\boldsymbol{x}, \\ &= \int_{\mathbb{X}} d^{\pi}_{\gamma}(\boldsymbol{x})\int_{\mathbb{U}}\pi(\boldsymbol{u}|\boldsymbol{x})\boldsymbol{\nabla}_{\boldsymbol{\theta}}\log\pi(\boldsymbol{u}|\boldsymbol{x})\boldsymbol{\nabla}_{\boldsymbol{\theta}}\log\pi(\boldsymbol{u}|\boldsymbol{x})^T d\boldsymbol{u}d\boldsymbol{x}, \\ &= F(\boldsymbol{\theta}), \end{aligned} \tag{26}$$

with $d^{\pi}_{\gamma}(\boldsymbol{x}) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t\Pr\{\boldsymbol{x}_t = \boldsymbol{x}\}$. Therefore, we can conclude that for both the average reward and the discounted case, the Fisher information and all-action matrix are the same, i.e., $G(\boldsymbol{\theta}) = F(\boldsymbol{\theta})$.

3.2 Derivation of the Natural Actor-Critic Algorithm

The algorithm suggested in this section relies on several observation described in this paper. First, we discussed that policy gradient methods are a rather promising Reinforcement Learning technique in terms of scaling to high dimensional continuous control systems. Second, we derived that natural policy gradients are easier to estimate than regular gradients, which should lead to faster converge to the nearest local minima in the with respect to the Fisher information metric, such that natural gradients are in general more efficient. Third, Nedic and Bertsekas[12] provided another important ingredient to our algorithm, showing that a least-squares policy evaluation method, LSTD(λ), converges with probability one for function approximation (although it can differ from a supervised Monte-Carlo solution $\lambda < 1$ due to a remaining function approximation error [30]). Based on these observations, we will develop an algorithm which exploits the merits of all these techniques.

By rewriting the Bellman equation (c.f. eq.(3)) using the compatible function approximation (c.f. eq.(14))[3] we arrive at

$$Q^{\pi}(\boldsymbol{x},\boldsymbol{u}) = A^{\pi}(\boldsymbol{x},\boldsymbol{u}) + V^{\pi}(\boldsymbol{x}) = r\left(\boldsymbol{x},\boldsymbol{u}\right) + \gamma \int_{\mathbb{X}} p(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u}) V^{\pi}(\boldsymbol{x}') d\boldsymbol{x}'. \tag{27}$$

Inserting $A^{\pi}(\boldsymbol{x},\boldsymbol{u}) = f^{\pi}_{w}(\boldsymbol{x},\boldsymbol{u})$ and an appropriate basis functions representation of the value function as $V^{\pi}(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{v}$, we can rewrite the Bellman Equation, Eq., (27), as a set of linear equations

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^T \boldsymbol{w} + \boldsymbol{\phi}(\boldsymbol{x}_t)^T \boldsymbol{v} = \left\langle r(\boldsymbol{x}_t,\boldsymbol{u}_t) + \gamma \boldsymbol{\phi}(\boldsymbol{x}_{t+1})^T \boldsymbol{v} \right\rangle. \tag{28}$$

Using this set of simultaneous linear equations, a solution to Equation (27) can be obtained by adapting the LSTD(λ) policy evaluation algorithm [5, 7, 12]. For this purpose, we define

$$\widehat{\boldsymbol{\phi}}_t = [\boldsymbol{\phi}(\boldsymbol{x}_t)^T, \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^T]^T, \;\; \widetilde{\boldsymbol{\phi}}_t = [\boldsymbol{\phi}(\boldsymbol{x}_{t+1})^T, \boldsymbol{0}^T]^T, \tag{29}$$

as new basis functions, where $\boldsymbol{0}$ is the zero vector. This definition of basis function is beneficial for a low variance of the value function estimate as the basis functions $\widetilde{\boldsymbol{\phi}}_t$ do not depend on future actions $\boldsymbol{u}_{t+1}$ – i.e., the input variables to the LSTD regression are not noisy due to $\boldsymbol{u}_{t+1}$; such input noise would violate the standard regression model that only takes noise in the regression targets into account. LSTD(λ) with the basis functions in Eq.(29), called LSTD-Q(λ) from now on, is thus currently the theoretically cleanest way of applying LSTD to state-value function estimation. It is exact for deterministic or weekly noisy state transitions and arbitrary stochastic policies, and, as all previous LSTD suggestions, it loses accuracy with increasing noise in the state transitions since $\widetilde{\boldsymbol{\phi}}_t$ becomes a random variable. The complete LSTD-Q(λ) algorithm is given in the *Critic Evaluation* (lines 4.1-4.3) of Table 1.

[3] Baird[28] introduced a similar Bellman equation for a greedy algorithm 'Advantage Updating'.

Table 1. Natural Actor-Critic Algorithm with LSTD-Q(λ) for infinite horizon tasks

```
Input: Parameterized policy $\pi(\boldsymbol{u}|\boldsymbol{x}) = p(\boldsymbol{u}|\boldsymbol{x},\boldsymbol{\theta})$ with initial parameters $\boldsymbol{\theta} = \boldsymbol{\theta}_0$,
    its derivative $\nabla_{\boldsymbol{\theta}} \log\pi(\boldsymbol{u}|\boldsymbol{x})$and basis functions $\boldsymbol{\phi}(\boldsymbol{x})$for state value
    function parameterization $V^{\pi}(\boldsymbol{x})$.

1: Draw initial state $\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0)$, and select parameters $\boldsymbol{A}_{t+1} = \boldsymbol{0}$, $\boldsymbol{b}_{t+1} = \boldsymbol{z}_{t+1} = \boldsymbol{0}$.
2: For $t = 0, 1, 2, \ldots$ do
3:   Execute: Draw action $\boldsymbol{u}_t \sim \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)$, observe next state $\boldsymbol{x}_{t+1} \sim p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)$,
     and reward $r_t = r(\boldsymbol{x}_t, \boldsymbol{u}_t)$.

4:   Critic Evaluation (LSTD-Q): Determine state-value function
     $V^{\pi}(\boldsymbol{x}_t) = \boldsymbol{\phi}(\boldsymbol{x}_t)^T \boldsymbol{v}_t$ and the compatible advantage function approximation
     $f^{\pi}_{\boldsymbol{w}}(\boldsymbol{x}_t, \boldsymbol{u}_t) = \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^T \boldsymbol{w}_t$.
4.1:   Update basis functions:
       $\widetilde{\boldsymbol{\phi}}_t = [\boldsymbol{\phi}(\boldsymbol{x}_{t+1})^T, \boldsymbol{0}^T]^T$;  $\widehat{\boldsymbol{\phi}}_t = [\boldsymbol{\phi}(\boldsymbol{x}_t)^T, \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^T]^T$,
4.2:   Update sufficient statistics:
       $\boldsymbol{z}_{t+1} = \lambda \boldsymbol{z}_t + \widehat{\boldsymbol{\phi}}_t$; $\boldsymbol{A}_{t+1} = \boldsymbol{A}_t + \boldsymbol{z}_{t+1}(\widehat{\boldsymbol{\phi}}_t - \gamma \widetilde{\boldsymbol{\phi}}_t)^T$; $\boldsymbol{b}_{t+1} = \boldsymbol{b}_t + \boldsymbol{z}_{t+1} r_t$,
4.3:   Update critic parameters:
       $[\boldsymbol{v}^T_{t+1}, \boldsymbol{w}^T_{t+1}]^T = \boldsymbol{A}^{-1}_{t+1} \boldsymbol{b}_{t+1}$.

5:   Actor-Update: When the natural gradient is converged over a window $h$,
     i.e., $\forall \tau \in [0, ..., h] : \measuredangle(\boldsymbol{w}_{t+1}, \boldsymbol{w}_{t-\tau}) \leq \epsilon$, update the
     parameterized policy $\pi(\boldsymbol{u}_t|\boldsymbol{x}_t) = p(\boldsymbol{u}_t|\boldsymbol{x}_t, \boldsymbol{\theta}_{t+1})$:
5.1:   policy parameters:
       $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \boldsymbol{w}_{t+1}$,
5.2:   forget some of sufficient statistics with $\beta \in [0, 1]$:
       $\boldsymbol{z}_{t+1} \leftarrow \beta \boldsymbol{z}_{t+1}, \boldsymbol{A}_{t+1} \leftarrow \beta \boldsymbol{A}_{t+1}, \boldsymbol{b}_{t+1} \leftarrow \beta \boldsymbol{b}_{t+1}$.
6: end.
```

Once LSTD-Q(λ) converges to an approximation of $A^{\pi}(\boldsymbol{x}_t, \boldsymbol{u}_t) + V^{\pi}(\boldsymbol{x}_t)$ (which it does with probability 1 as shown in [12]), we obtain two results: the value function parameters $\boldsymbol{v}$, and the natural gradient $\boldsymbol{w}$. The natural gradient $\boldsymbol{w}$ serves in updating the policy parameters $\Delta\boldsymbol{\theta}_t = \alpha \boldsymbol{w}_t$. After this update, the critic has to forget at least parts of its accumulated sufficient statistics using a forgetting factor $\beta \in [0, 1]$ (cf. Table 1). For $\beta = 0$, i.e., complete resetting, and appropriate basis functions $\phi(\boldsymbol{x})$, convergence to the true natural gradient can be guaranteed. The complete Natural Actor Critic algorithm is shown in Table 1.

This algorithm has rather strong guarantees. The usage of LSTD(λ) for obtaining the natural gradient simultaneously with policy evaluation ensures that the algorithm converges to a natural policy gradient estimate before a policy update step (Step 3) is performed. If appropriate basis functions are chosen, LSTD(λ) can be shown to converge with probability one [12]; however, the gradient can differ from the supervised solution for $\lambda < 1$ as pointed out in [30], implying that the manifold on which the gradient descent is being performed differs depending λ). Following the natural gradient guarantees to reach the closest local minimum w.r.t. to the criterion in Eq.(8) [27].

Nevertheless, there are some critical design issues which influence the performance of this algorithm. First, a sufficiently good set of basis functions $\boldsymbol{\phi}(\boldsymbol{x})$ has to be chosen to represent the value function. Second, two open parameters, i.e., the eligibility rate λ, and the learning rate α, have to be determined. We have considered a stochastic policy in all our examples. As the exploration parameter σ approaches 0, i.e. the policy becomes more deterministic, it becomes harder to ensure that $\boldsymbol{A}_{t+1}$ remains invertible. Hence, strongly deteministic policies are complicated to handle. Another issue concerns that the gradient estimate as well as the value function estimate are reset after every gradient update. In future work, we hope to use the estimate of the previous gradients to initialize our subsequent rollouts using the results given in [12] – basically, by using LSTD-like algorithms which can incorporate an initial guess for the parameters and still be unbiased.

4 Empirical Evalutations

4.1 Example I: Controlling an Unstable Plant

One of the most well-known benchmarks in reinforcement learning is 'pole balancing' [14, 29] – the setup is shown in Figure 2 (a). In this control task, a pole with a point-mass on its upper end is mounted on a cart or robot arm, and it has to be stabilized by the motions of the cart/robot arm. This problem is sufficiently difficult as an empirical evaluations, while remaining analytically solvable given some minor assumptions.

System and Rewards. For such regulator tasks, it is common to approximate the control system by a linearization about a given setpoint, chosen to be the zero vector in our example. The density function of the start-state is given by a Gaussian distribution $p\left(\boldsymbol{x}_0\right) = \mathcal{N}(\boldsymbol{x}_0|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ with $\boldsymbol{\mu}_0 = \mathbf{0}$, and $\boldsymbol{\Sigma}_0 = \text{diag}(0.1, 0.1, 0.1, 0.1)$ (using SI units). Near the mean of the start-state distribution of the pole, the pole dynamics can be described approximately by $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t) = \mathcal{N}\left(\boldsymbol{x}_{t+1}\left|\boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t, \boldsymbol{\Sigma}_T\right.\right)$, with

$$\boldsymbol{x}_t = \begin{bmatrix} x_t \\ \dot{x}_t \\ \alpha \\ \dot{\alpha} \end{bmatrix}, \boldsymbol{A} = \begin{bmatrix} 1 & \tau & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \tau \\ 0 & 0 & \nu\tau & 1 \end{bmatrix}, b = \begin{bmatrix} 0 \\ \tau \\ 0 \\ \nu\tau/g \end{bmatrix}$$

and $\boldsymbol{u}_t = F$, $\tau = 1/60s$, $\nu = 13.2s^{-2}$, $\boldsymbol{\Sigma}_T = 0.01\boldsymbol{\Sigma}_0$ as in [31]. The rewards are given by $r\left(\boldsymbol{x}_t, \boldsymbol{u}_t\right) = \boldsymbol{x}_t^T\boldsymbol{Q}\boldsymbol{x}_t + \boldsymbol{u}_t^T\boldsymbol{R}\boldsymbol{u}_t$ with $\boldsymbol{Q} = \text{diag}(1.25,\ 1,\ 12,\ 0.25)$, and $\boldsymbol{R} = 0.01$ as in [29].

Parameterized Policy. The policy is specified as $\pi(\boldsymbol{u}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{Kx}, \sigma^2)$. In order to ensure that the learning algorithm cannot exceed an acceptable parameter range, the variance of the policy is defined as $\sigma = 0.1 + 1/(1 + \exp(\eta))$. Thus, the policy parameter vector becomes $\boldsymbol{\theta} = [\boldsymbol{K}^T, \eta]^T$ and has the analytically computable optimal solution $\boldsymbol{K} \approx [5.71, 11.3, -82.1, -21.6]^T$, and $\sigma = 0.1$, corresponding to $\eta \to \infty$. As $\eta \to \infty$ is hard to visualize, we show σ in Figure 2 (b) despite the fact that the update takes place over the parameter η.

These experiments required that the function $J(\boldsymbol{\theta}) = E\left\{\sum_{t=0}^{\infty} \gamma^t r_t | \boldsymbol{\theta}\right\}$ exists for the initial policy with $\boldsymbol{\theta}_0$; this is equivalent to requirement of the largest eigenvalue of a deterministic closed loop system obeys $\gamma^{-2} > \text{eig}(\boldsymbol{A} + \boldsymbol{BK})$. Interestingly, this prerequisite does not require a stabilizing initial policy stability for $\gamma = 0.95$ – however, for $\gamma = 1$ a stabilizing policy would be required. One hundred initial policies are generated by uniformly selecting $\boldsymbol{K}$ from $[10 \pm 400, 15 \pm 400, -90 \pm 400, -25 \pm 400]^T$ and $\sigma = 0.5 \pm 0.4$, while rejecting the ones where $\gamma^{-2} \leq \text{eig}(\boldsymbol{A} + \boldsymbol{BK})$.

Experiments. For each initial policy, samples $(\boldsymbol{x}_t, \boldsymbol{u}_t, r_{t+1}, \boldsymbol{x}_{t+1})$ were generated using the start-state distribution, transition probabilities, the rewards, and the policy. The samples arrive at a rate of 60 Hz and are immediately incorporated by the Natural Actor-Critic module. The policy is updated when $\measuredangle(\boldsymbol{w}_{t+1}, \boldsymbol{w}_t) \leq \epsilon = \pi/180$. At the time of update, the true 'vanilla' policy gradient, which can be computed analytically[4], is used to update a separate policy, thus serving as a baseline for the comparison. If the pole leaves the acceptable region of $-\pi/6 \leq \phi \leq \pi/6$, and $-1.5m \leq x \leq +1.5m$, it is reset to a new starting position drawn from the start-state distribution.

The additional basis functions for the Natural Actor-Critic are chosen as $\boldsymbol{\phi}(\boldsymbol{x}) = [x_1^2,\ x_1x_2,\ x_1x_3,\ x_1x_4,\ x_2^2,\ x_2x_3,\ x_2x_4,\ x_3^2,\ x_3x_4,\ x_4^2,\ 1]^T$. These basis functions are sufficiently rich to represent the quadratic value function for linear problems.

Results and Discussion. Results are illustrated in Figure 2. In 2 (b), a sample run is shown: the natural-actor critic algorithms estimates the optimal solution within less than ten minutes of simulated robot trial time. The analytically obtained policy gradient for comparison takes over two hours of robot experience to get to the true solution. In a real world application, a significant amount of time would be added for the vanilla policy gradient as it is more unstable and leaves the admissible area more often. The policy gradient is clearly outperformed by the natural actor-critic algorithm. The performance difference between the true natural gradient and the natural actor-critic algorithm is negligible and, therefore, not shown separately. In Figure 2 (c), the expected return over updates is shown averaged over all hundred initial policies.

In this experiment, we demonstrated that the natural actor critic is comparable with the ideal natural gradient, and outperforms the 'vanilla' policy gradient

[4] The true *natural* policy gradient can also be computed analytically. In the following evaluations, however, it is not shown as the difference in performance to the Natural Actor Critic gradient estimate is negligible.

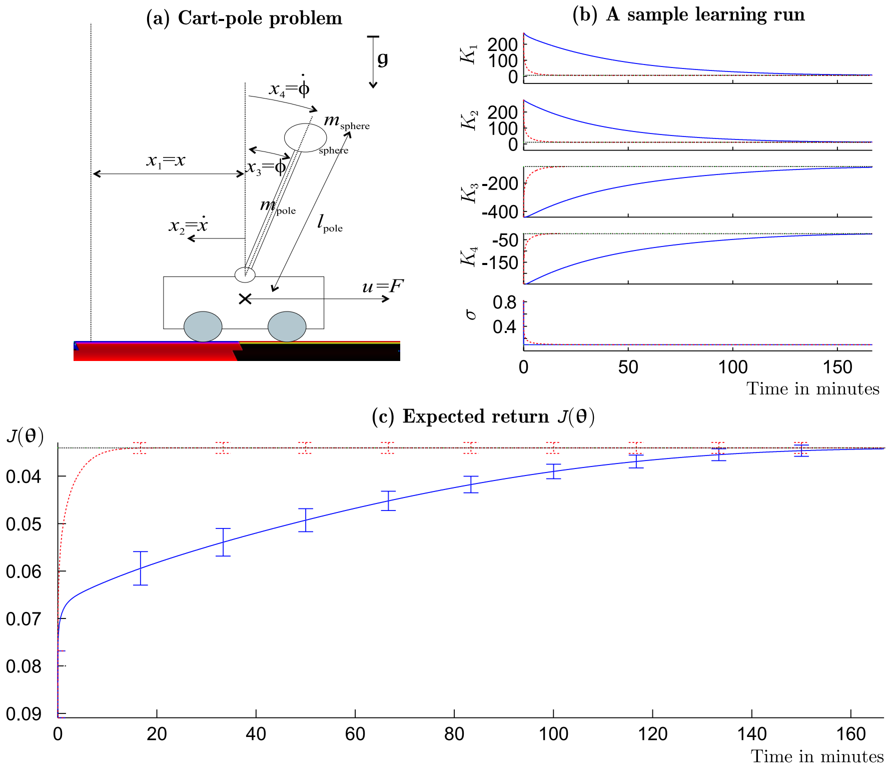

Fig. 2. Performance of Natural Actor-Critic in the Cart-Pole Balancing framework. (a) General setup of the cart-pole system. (b) A typical sample learning run of both the natural actor-critic (dashed line) and the true policy gradient (solid line). (c) The expected return of the policy over time. Curves are an average over 100 randomly picked policies as described in Section 4.1.

significantly. Greedy policy improvement methods do not compare easily. Discretized greedy methods cannot compete due to the fact that the amount of data required would be significantly increased. Model-based dynamic programming based methods as described in the linear quadratic regulation literature work well, but require the estimation of a model [9] and off-line iterations to determine the policy. The only suitable greedy improvement method, to our knowledge, is Bradtke's Adaptive Policy Iteration [7]. However, this method is problematic in real-world application as the policy in Bradtke's method is deterministic: the estimation of the action-value function is an ill-conditioned regression problem with redundant parameters and no explorative noise. For these reasons, we focused on the comparison of 'vanilla' and natural policy gradients.

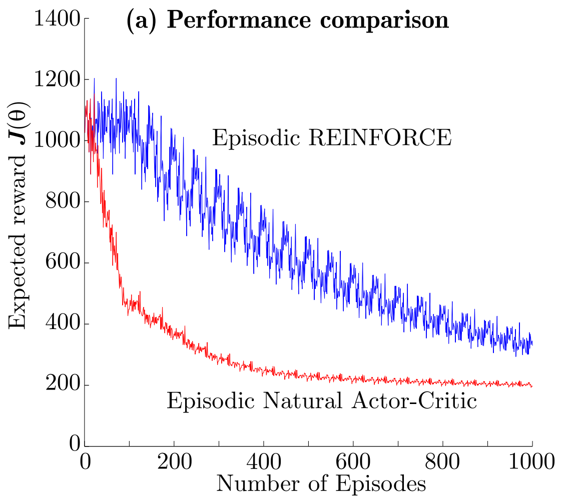

Fig. 3. Episodic natural actor-critic for learning dynamic movement primitives. (a) Learning curves comparing the episodic Natural Actor Critic to episodic REINFORCE. (b) Humanoid robot DB which was used for this task. Note that the variance of the episodic Natural Actor Critic learning is significantly lower than the one of episodic REINFORCE, with about 10 times faster convergence.

4.2 Example II: Optimizing Nonlinear Motor Primitives for Humanoid Motion Planning

While the previous example demonstrated the feasibility and performance of the Natural Actor Critic in a classical example of motor control, this section will turn towards an application of optimizing nonlinear dynamic motor primitives for a humanoid robot. In [17, 16], a novel form of representing movement plans $(\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d)$ for the degrees of freedom (DOF) of a robotic system was suggested in terms of the time evolution of the nonlinear dynamical systems

$$\dot{q}_{d,k} = h(q_{d,k}, \boldsymbol{z}_k, g_k, \tau, \theta_k) \tag{30}$$

where $(q_{d,k}, \dot{q}_{d,k})$ denote the desired position and velocity of a joint, $\boldsymbol{z}_k$ 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 original work in [17, 16] 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.(30) by means of reinforcement learning, which is the crucial second step in imitation learning.

The system in Eq.(30) is a point-to-point movement, i.e., an episodic task from the view of reinforcement learning – continuous (e.g., periodic) movement can be considered, too, with minor modifications of our framework. In order to apply the Natural Actor Critic, two issues need to be addressed: what are appropriate basis functions $\boldsymbol{\phi}(\boldsymbol{x})$ for the value function, and what modifications are required for this *finite* horizon task. It turns out that a straightforward simplification of the Natural Actor-Critic can deal with both issues elegantly. For the deterministic dynamics of the movement primitives, we can formulate the discounted sum of advantages along one roll-out as

$$\begin{aligned}\sum_{i=0}^{N}\gamma^i A^\pi(\boldsymbol{x}_i,\boldsymbol{u}_i) &= \sum_{i=0}^{N}\gamma^i(r(\boldsymbol{x}_i,\boldsymbol{u}_i)+\gamma V^\pi(\boldsymbol{x}_{i+1})-V^\pi(\boldsymbol{x}_i)), \\ &= \sum_{i=0}^{N}\gamma^i r(\boldsymbol{x}_i,\boldsymbol{u}_i)+\gamma^{N+1}V^\pi(\boldsymbol{x}_{N+1})-V^\pi(\boldsymbol{x}_0),\end{aligned} \tag{31}$$

i.e., for episodic tasks where $V^\pi(\boldsymbol{x}_{N+1})=0$, we can set the additional basis functions $\boldsymbol{\phi}(\boldsymbol{x})$ to $\boldsymbol{\phi}(\boldsymbol{x})=1$ – indeed, only one basis function is needed, as only $V^\pi(\boldsymbol{x}_0)$ is unknown. Thus, after inserting the compatible function approximator for $A^\pi(\boldsymbol{x}_i,\boldsymbol{u}_i)$, we arrive at a well-defined regression problem for $\boldsymbol{w}$ after multiple roll-outs, which corresponds to LSTD-Q(1):

$$\sum_{i=0}^{N}\gamma^i\boldsymbol{\nabla}_{\boldsymbol{\theta}}\log\pi(\boldsymbol{u}_i|\boldsymbol{x}_i)^T\boldsymbol{w}+v=\sum_{i=0}^{N}\gamma^i r(\boldsymbol{x}_i,\boldsymbol{u}_i). \tag{32}$$

In order to yield an unbiased estimated of $\boldsymbol{w}$, this equation requires a distribution of start states $p(\boldsymbol{x}_0)$ of roll-outs that has its mean at $\boldsymbol{x}_0$, which is different from the general form in Eq.(32) that does not have such a condition. A special algorithm for this episodic version of the Natural Actor Critic is given in Table 2.

We evaluated this algorithm on a typical problem of motor learning, the acquistion of smooth reaching trajectories for arm movements, using the movement primitives in Eq.(30). The reward of a trajectory for every DOF is given by

$$r_k(\boldsymbol{x}_{0:N},\boldsymbol{u}_{0:N})=\sum_{i=0}^{N}c_1\dot{q}_{d,k,i}^2+c_2(q_{d,k,N}-g_k),$$

where $c_1=1$, $c_2=1000$, and g_k is chose appropriately to not violate the range of motion of each DOF. The reward along a trajectory corresponds to $\sum_{i=0}^{N}\gamma^i r(\boldsymbol{x}_i,\boldsymbol{u}_i)$ in Eq.(32) with a discount factor of $\gamma=1$. Starting from a zero parameter vector in each DOF, we optimized the parameters (for each DOF independently) to accomplish a minimal value under the reward criterion. Ten parameters per DOF were employed. For a comparision, we also tested the episodic REINFORCE algorithm. Trajectories were executed on the humanoid robot DB. As can be seen in Figure 3, the episodic Natural Actor-Critic exhibited rather promising results: within less than 100 trials, a fairly low value of the cost criterion was accomplished, and final convergence was reached between 300 to 400 trials. To the best of our knowledge, this algorithm is the fastest policy gradient method which can be applied on this task – episodic REINFORCE

Table 2. Natural Actor-Critic Algorithm adapted for episodic task.

```
Input: Parameterized policy $\pi(\boldsymbol{u}|\boldsymbol{x}) = p(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta})$ with initial parameters $\boldsymbol{\theta} = \boldsymbol{\theta}_0$,
    its derivative $\boldsymbol{\nabla}_{\boldsymbol{\theta}}\log\pi(\boldsymbol{u}|\boldsymbol{x})$and one basis function $\boldsymbol{\phi}(\boldsymbol{x_0}) = 1$ as needed for the start state value
    function parameterization $V^{\pi}(\boldsymbol{x_0})$.

1: Select parameters $\boldsymbol{A}_0 = \boldsymbol{0}$, $\boldsymbol{b}_0 = \boldsymbol{0}$.
2: For $e = 0, 1, 2, \ldots$ do
3:   Draw initial state $\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0)$.
     For $t = 0, 1, 2, \ldots, N$ do
       Draw action $\boldsymbol{u}_t \sim \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)$, observe next state $\boldsymbol{x}_{t+1} \sim p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)$,
       and reward $r_t = r(\boldsymbol{x}_t, \boldsymbol{u}_t)$.
     end;

4:   Critic Evaluation (LSTD-Q(1)): Determine state-value function
     $V^{\pi}(\boldsymbol{x}_0) = \boldsymbol{\phi}(\boldsymbol{x}_o)v_e = v_e$ of the initial state $\boldsymbol{x}_o$ and the compatible advantage
     function approximation $f^{\pi}_{\boldsymbol{w}}(\boldsymbol{x}_t, \boldsymbol{u}_t) = \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^T \boldsymbol{w}_e$.
4.1:   Determine
       $\widehat{\boldsymbol{\phi}}_e = \sum_{t=0}^{N} \gamma^t [1, \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^T]^T$,
       $r_e = \sum_{t=0}^{N} \gamma^t r_t$,
4.2:   Update sufficient statistics:
       $\boldsymbol{A}_{e+1} = \boldsymbol{A}_e + \widehat{\boldsymbol{\phi}}_e \widehat{\boldsymbol{\phi}}_e^T$;  $\boldsymbol{b}_{e+1} = \boldsymbol{b}_e + \widehat{\boldsymbol{\phi}}_e r_e$,
4.3:   Update critic parameters:
       $[v_{e+1}, \boldsymbol{w}_{e+1}^T]^T = \boldsymbol{A}_{e+1}^{-1} \boldsymbol{b}_{e+1}$.

5:   Actor-Update: When the natural gradient is converged over a window $h$,
     i.e., $\forall\tau \in [0, ..., h] : \measuredangle(\boldsymbol{w}_{e+1}, \boldsymbol{w}_{e-\tau}) \leq \epsilon$, update the
     parameterized policy $\pi(\boldsymbol{u}_t|\boldsymbol{x}_t) = p(\boldsymbol{u}_t|\boldsymbol{x}_t, \boldsymbol{\theta}_{e+1})$:
5.1:   policy parameters:
       $\boldsymbol{\theta}_{e+1} = \boldsymbol{\theta}_e + \alpha \boldsymbol{w}_{e+1}$,
5.2:   forget some of sufficient statistics with $\beta \in [0, 1]$:
       $\boldsymbol{A}_{e+1} \leftarrow \beta \boldsymbol{A}_{e+1}, \boldsymbol{b}_{e+1} \leftarrow \beta \boldsymbol{b}_{e+1}$.
6: end.
```

required about 10 times more trials until a similar level of performance. In complex learning task, humans show similar learning performance. The smoothness of the reduction of the cost criterion in Figure 3 should be noted, too, indicating a smooth performance improvement that is desirable for the safety of a physical system and its environment.

5 Conclusion

In this paper, we reviewed the concepts and problems of traditional and novel reinforcement learning algorithms with a focus on applicability to humanoid motor control. We highlighted that greedy policy-improvement algorithms are likely to fail to scale to high dimensional movement systems as their large changes in the policy during learning makes stable algorithms so far infeasible. Policy gradients, on the other hand, have been successfully applied in humanoid robotics for both walking and fine manipulation; this success indicates that these methods could be potentially useful for humanoid robotics. Research of this paper demonstrated that policy gradient methods can even be taken to a new level of theory and applicability by using the natural instead of 'vanilla' policy gradient.

We derived an algorithm, the Natural Actor-Critic, which uses two essential components. First, it approximates the natural gradient directly in the policy evaluation loop using the compatible function approximator. This part is based on LSTD(λ) and inherits several strong theoretical properties from previous work. Second, the natural gradient is used in order to improve the policy; under suitable conditions, this method guarantees convergence with probability one to the next local minimum of the average reward function. We applied the infinite horizon version of the Natural Actor-Critic algorithm successfully in a simple robotic task, i.e., pole balancing, while applying the episodic version of the Natural Actor-Critic algorithm to the optimization of humanoid motor primitives. Future work will address the evaluation of the suggested learning framework in more complex tasks, in particular in conjunction with priming a control policy from imitation learning and subsequent self-improvement.

References

1. Baird, L. C., & Moore, A. W. (1999). Gradient descent for general reinforcement learning. Advances in Neural Information Processing Systems 11
2. Bellman, R., Dynamic Programming, Princeton, NJ: Princeton University Press, 1957
3. Benbrahim, H. & Franklin, J., Biped dynamic walking using reinforcement learning, Robotics and Autonomous Systems Journal, 1997
4. Bertsekas, D. P. & Tsitsiklis, J. N.. Neuro-Dynamic Programming. Athena Scientific, Belmont, MA, 1996
5. Boyan, J., Least-squares temporal difference learning. In I. Bratko and S. Dzeroski, editors, Machine Learning: Proceedings of the Sixteenth International Conference, pages 49– 56. Morgan Kaufmann, San Francisco, CA, 1999
6. Bradtke, S. and Barto, A.. Linear least-squares algorithms for temporal difference learning, Machine Learning, 22, 1996
7. Bradtke, S., Ydstie, E. and Barto, A. Adaptive Linear Quadratic Control Using Policy Iteration. Technical report UM-CS-1994-049, University of Massachusetts, 1994.
8. Dayan, P.. The convergence of TD(lambda) for general lambda. Machine Learning, 8, 341-362, 1992
9. Dorato, P., Abdallah, C., and Cerone, V. Linear-Quadratic Control: An Introduction. Prentice Hall, 1998.
10. Kakade, S. A Natural Policy Gradient. Advances in Neural Information Processing Systems 14, 2002
11. Lagoudakis, M., and Parr, R., Model-Free Least-Squares policy iteration, Technical Report CS-2000-05, Duke University, 2001

12. Nedic, A. and Bertsekas, D. P., Least-Squares Policy Evaluation Algorithms with Linear Function Approximation, LIDS Report LIDS-P-2537, Dec. 2001; to appear in J. of Discrete Event Systems, 2002.
13. Gullapalli, V., Learning Control Under Extreme Uncertainty, In Advances in Neural Information Processing Systems 5 (NIPS 5), SJ Hanson, JD Cowan, CL Giles, Eds. pp. 327-334, Morgan Kaufmann 1993.
14. Sutton, R.S., and Barto, A.G. Reinforcement Learning, The MIT Press, 1998
15. Ilg, J. Albiez, K. Berns, R. Dillmann. Adaptive Biologically Inspired Control for the Four-Legged Walking Machine BISAM. In International Conference on Climbing and Walking Robots (CLAWAR), S. 809-818, Portsmouth, Oktober 1999.
16. Ijspeert, A., Nakanishi, J., Schaal, S. Learning attractor landscapes for learning motor primitives, in: Becker, S.;Thrun, S.;Obermayer, K. (eds.), Advances in Neural Information Processing Systems 15, MIT Press, 2003.
17. Ijspeert, J.A.;Nakanishi, J.;Schaal, S. Learning rhythmic movements by demonstration using nonlinear oscillators, IEEE International Conference on Intelligent Robots and Systems (IROS 2002), pp.958-963, Piscataway, NJ: IEEE, 2002.
18. Peters, J., Vijaykumar, S., Schaal, S. Comparing Policy Gradient Methods. In preparation for ICML 2004.
19. Sutton, R.S., Comparing Policy Gradient Methods, unfinished paper draft.
20. Sutton, R.S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In Advances in Neural Information Processing Systems 12. MIT Press, 2000.
21. Konda, V., and Tsitsiklis, J.N. Actor-Critic Algorithms. In Advances in Neural Information Processing Systems 12. MIT Press, 2000.
22. Marbach, P. and Tsitsiklis, J.N. Simulation-based optimization of Markov reward processes: implementation issues, in Proceedings of the 38th IEEE Conference on Decision and Control, 1999
23. Baxter, J., Bartlett, P., and Weaver, L. Experiments with Infinite-Horizon, Policy-Gradient Estimation, Journal of Artificial Intelligence Research 15, 2001.
24. Kimura, H. and Kobayashi, S.: Reinforcement Learning for Continuous Action using Stochastic Gradient Ascent, The 5th International Conference on Intelligent Autonomous Systems, 1998.
25. Moon, T., and Stirling, W. Mathematical Methods and Algorithms for Signal Processing, Prentice Hall, 2000.
26. Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 8, 1992.
27. Amari, S. Natural Gradient Works Efficiently in Learning. Neural Computation 10, 1998
28. Baird, L. C. Advantage Updating. Technical Report WL-TR-93-1146, Wright-Patterson Air Force Base Ohio: Wright Laboratory, 1993.
29. Schaal, S. Learning from demonstration, in: Mozer, M.C.;Jordan, M.;Petsche, T. (eds.), Advances in Neural Information Processing Systems 9, pp.1040-1046, MIT Press, 1997.
30. Schoknecht, R., Optimality of Reinforcement Learning Algorithms with Linear Function Approximation, in: Becker, S.;Thrun, S.;Obermayer, K. (eds.), Advances in Neural Information Processing Systems 15, MIT Press, 2003.
31. Mehta, B., Schaal, S. Forward models in visuomotor control, J Neurophysiol, 88, 2, pp.942-53, 2002.