Scaling Reinforcement Learning Paradigms for Motor Control

Peters, J., Vijayakumar, S., & Schaal, S. (2003). Scaling Reinforcement Learning Paradigms for Motor Control. In Proc. 10th Joint Symposium on Neural Computation, UC Irvine.

Scaling Reinforcement Learning Paradigms for Motor Control

Jan Peters, Sethu Vijayakumar, Stefan Schaal
Computer Science & Neuroscience, Univ. of Southern California, Los Angeles, CA 90089
ATR Computational Neuroscience Laboratories, Kyoto 619-02, Japan
{jrpeters,sethu,sschaal}@usc.edu

June 9, 2003

Abstract

Reinforcement learning offers a general framework to explain reward related learning in artificial and biological motor control. However, cur-rent reinforcement learning methods rarely scale to high dimensional movement systems and mainly operate in discrete, low dimensional domains like game-playing, artificial toy problems, etc. This drawback makes them unsuitable for application to human or bio-mimetic motor control. In this poster, we look at promising approaches that can potentially scale and suggest a novel formulation of the actor-critic algorithm which takes steps towards alleviating the current shortcomings. We argue that methods based on greedy policies are not likely to scale into high-dimensional domains as they are problematic when used with function approximation a must when dealing with continuous domains. We adopt the path of direct policy gradient based policy improvements since they avoid the problems of unstabilizing dynamics encountered in traditional value iteration based updates. While regular policy gradient methods have demonstrated promising results in the domain of humanoid motor control, we demonstrate that these methods can be significantly improved using the natural policy gradient instead of the regular policy gradient. Based on this, it is proved that Kakades average natural policy gradient is indeed the true natural gradient. A general algorithm for estimating the natural gradient, the Natural Actor-Critic algorithm, is introduced. This algorithm converges with probability one to the nearest local minimum in Riemannian space of the cost function. The algorithm outperforms non-natural policy gradients by far in a cart-pole balancing evaluation, and offers a promising route for the development of reinforcement learning for truly high-dimensionally continuous state-action systems. **Keywords:** Reinforcement learning, neuro-dynamic programming, actor-critic methods, policy gradient methods, natural policy gradient

1 Introduction

Reinforcement learning algorithms based on value function approximation with discrete lookup table parameterization have been highly successful. However, when applied with continuous function approximation, many of these algorithms failed to generalize, and few convergence guarantees could be obtained [15]. The reason for this problem can largely be traced back to the greedy or ϵ-greedy policy update of most techniques, as it does not ensure a policy improvement when applied with an approximate value function [5]. During a greedy update small errors in the value function can cause large changes in the policy which in return cause large changes in the value function. This process, when applied repeatedly, can result in oscillations or divergence of the algorithms. Even in simple toy systems, such behavior can be shown for most of the well-known greedy reinforcement learning algorithms [2, 5].

As an alternative to greedy reinforcement learning, policy gradient methods have been suggested. Policy gradients have rather strong convergence guarantees, even when used in conjunction with function approximation for value functions, and recent results created a theoretically solid framework for policy gradient estimation from sampled data [16, 10]. However, even when applied to simple examples with rather few states, policy gradient methods often turn out to be quite inefficient [9], partially caused by the large plateaus in the expected return landscape where the gradients are small and often do not point towards the optimal solution. A simple example that demonstrates this behavior is given in Fig. 1.

Similar as in supervised learning, the steepest ascent in Riemannian space [1], called the 'natural' policy gradient, turns out to be significantly more efficient than normal gradients. Such an approach was first suggested for reinforcement learning as the 'average natural policy gradient' by Kakade [8], and subsequently shown in preliminary work to be the true natural policy gradient [13, 4]. In this paper, after a brief introduction to our framework of reinforcement learning in Section 2, we take this line of reasoning one step further in Section 3 by

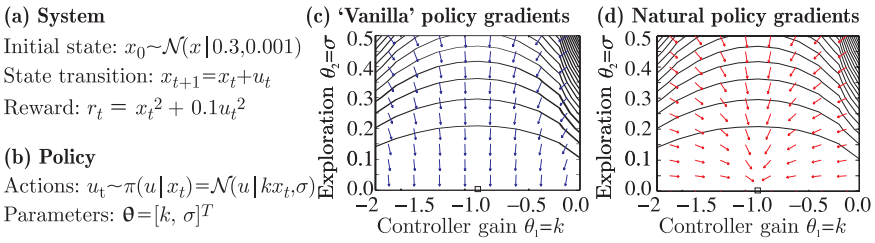

Figure 1: When plotting the expected return landscape, the differences between 'vanilla' and natural policy gradients become apparent for simple examples (a-b). 'Vanilla' policy gradients (c) point onto a plateau at $\theta_2 = 0$, while natural policy gradients direct to the optimal solution (d). The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

introducing the novel Natural Actor-Critic architecture which, under mild assumptions, converges to the next local minimum in Riemannian space with probability 1. Furthermore, in Section 4, we show that several successful previous reinforcement learning methods, including Sutton et al.'s original Actor-Critic, can be seen as special cases or approximations of this more general architecture. The paper concludes with empirical evaluations that demonstrate the effectiveness of the suggested methods in Section 5.

2 Markov Decision Processes

We assume that the underlying control problem is a *Markov Decision Process* (MDP) in discrete time with continuous state set $\mathbb{X} = \mathbb{R}^n$, and a continuous action set $\mathbb{U} = \mathbb{R}^m$ [5]. The system is at an initial state $\boldsymbol{x}_0 \in \mathbb{X}$ at time $t = 0$ drawn from the start-state distribution $p(\boldsymbol{x}_0)$. At any state $\boldsymbol{x}_t \in \mathbb{X}$ at time t, the actor will choose an action $\boldsymbol{u}_t \in \mathbb{U}$ by drawing it from a stochastic, parameterized policy $\pi(\boldsymbol{u}_t|\boldsymbol{x}_t) = p(\boldsymbol{u}_t|\boldsymbol{x}_t, \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta} \in \mathbb{R}^N$, and the system transfers to a new state $\boldsymbol{x}_{t+1}$ drawn from the state transfer distribution $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)$. The system yields a scalar reward $r_t = r(\boldsymbol{x}_t, \boldsymbol{u}_t) \in \mathbb{R}$ after each action. We assume that the policy $\pi_{\boldsymbol{\theta}}$ is continuously differentiable with respect to its parameters $\boldsymbol{\theta}$, and that for each considered policy $\pi_{\boldsymbol{\theta}}$, a state-value function $V^\pi(\boldsymbol{x})$, and the state-action value function $Q^\pi(\boldsymbol{x}, \boldsymbol{u})$ exist and can be defined by

$$V^\pi(\boldsymbol{x}_i) = E\left\{\sum_{t=0}^{\infty}\gamma^t r_{t+i}\middle|\, \boldsymbol{x}_i\right\}, Q^\pi(\boldsymbol{x}_i, \boldsymbol{u}_i) = E\left\{\sum_{t=0}^{\infty}\gamma^t r_{t+i}\middle|\, \boldsymbol{x}_i, \boldsymbol{u}_i\right\}, \tag{1}$$

where $\gamma \in [0, 1)$ denotes the discount factor. It is assumed that some basis functions $\boldsymbol{\phi}(\boldsymbol{x})$ are given so that the state-value function can be approximated with linear function approximation $V^\pi(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{v}$. The general goal is to optimize the expected return

$$J(\boldsymbol{\theta}) = E\left\{(1-\gamma)\sum_{t=0}^{\infty}\gamma^t r_t\middle|\, \boldsymbol{\theta}\right\} = \int_{\mathbb{X}} d^\pi(\boldsymbol{x})\int_{\mathbb{U}}\pi(\boldsymbol{u}|\boldsymbol{x})r(\boldsymbol{x}, \boldsymbol{u})d\boldsymbol{u}d\boldsymbol{x}, \tag{2}$$

where $d^\pi(\boldsymbol{x}) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t p(\boldsymbol{x}_t = \boldsymbol{x})$ denotes the discounted state distribution.

3 Natural Actor-Critic

Actor-Critic and many other policy iteration architectures consist of two steps, a policy evaluation step and a policy improvement step. The main requirements for the policy evaluation step are that it makes efficient usage of experienced data and converges with probability 1 to a good value function approximation of the current policy within the representational power of the value function parameterization. The policy improvement step is required to improve the policy on every step until convergence to a (local) optimum of the expected return while, on this way, taking a short trajectory in parameter space of the policy.

3.1 Actor Improvements with Natural Policy Gradient Updates

The requirements on the policy improvement step rule out greedy methods as, at the current state of knowledge, a policy improvement for approximated value functions cannot be guaranteed, even on average. 'Vanilla' policy gradient improvements (see e.g., [16, 10]) which follow the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ of the expected return function $J(\boldsymbol{\theta})$ (where $\nabla_{\boldsymbol{\theta}}$ denotes the derivative with respect to parameters $\boldsymbol{\theta}$) often get stuck in plateaus as demonstrated in [9]. Natural gradients $\widetilde{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ avoid this pitfall as demonstrated for supervised learning problems by Amari [1], and

suggested for reinforcement learning by Kakade [8]. These methods do not follow the steepest direction in the Euclidean space of the parameters but the steepest direction in Riemannian space given by

$$\widetilde{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = G^{-1}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}), \tag{3}$$

where $G(\boldsymbol{\theta})$ denotes the Fisher information matrix. It is guaranteed that the angle between natural and ordinary gradient is never larger than ninety degrees, i.e., convergence to the next local optimum can be guaranteed. The gradient in Euclidean space is given by the policy gradient theorem (see e.g., [16, 10]),

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \nabla_{\boldsymbol{\theta}} \pi(\boldsymbol{u}|\boldsymbol{x}) \left(Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) - b^{\pi}(\boldsymbol{x})\right) d\boldsymbol{u} d\boldsymbol{x}, \tag{4}$$

where $b^{\pi}(\boldsymbol{x})$ denotes a baseline. [16] and [10] demonstrated that in Eq. (4), the term $Q^{\pi}(\boldsymbol{x}, \boldsymbol{u})$ can be replaced by a compatible function approximation

$$f_w^{\pi}(\boldsymbol{x}, \boldsymbol{u}) = \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x})^T \boldsymbol{w}, \tag{5}$$

parameterized by the vector $\boldsymbol{w}$, *without* affecting the unbiasedness of the gradient estimate and irrespective of the choice of the baseline $b^{\pi}(\boldsymbol{x})$. However, as mentioned in [16], the baseline may still be useful in order to reduce the variance of the gradient estimate when Eq. 4 is approximated from samples. Based on Eqs.(4, 5), we derive an estimate of the policy gradient as

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x})^T d\boldsymbol{u} d\boldsymbol{x} \, \boldsymbol{w} = F(\boldsymbol{\theta}) \boldsymbol{w}. \tag{6}$$

Since $\pi(\boldsymbol{u}|\boldsymbol{x})$ is chosen by the user, even in sampled data, the integral $F(\boldsymbol{\theta}, \boldsymbol{x}) = \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x})^T d\boldsymbol{u}$ can be evaluated analytically or empirically without actually taking all actions. It is also noteworthy that the baseline does not appear in Eq. 6 as it integrates out, thus eliminating the need to find an optimal selection of this open parameter. Nevertheless, the estimation of $F(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) F(\boldsymbol{\theta}, \boldsymbol{x}) d\boldsymbol{x}$ still requires expensive roll-outs since $d^{\pi}(\boldsymbol{x})$ is not known; these roll-outs can become a severe bottleneck. However, Equation (6) has more surprising implications for policy gradients, when examining the meaning of the matrix $F(\boldsymbol{\theta})$ in Eq.(6). Kakade [8] argued that $F(\boldsymbol{\theta}, \boldsymbol{x})$ is the point Fisher information matrix for state $\boldsymbol{x}$, and that $F(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \, F(\boldsymbol{\theta}, \boldsymbol{x}) d\boldsymbol{x}$, therefore, denotes the 'average Fisher information matrix'. However, going one step further, we demonstrate in Endnote[1] 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. Eqs.(6) and (3) combined imply that the natural gradient can be computed as

$$\widetilde{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = G^{-1}(\boldsymbol{\theta}) F(\boldsymbol{\theta}) \boldsymbol{w} = \boldsymbol{w}, \tag{7}$$

since $F(\boldsymbol{\theta}) = G(\boldsymbol{\theta})$ (c.f. Endnote [1]). The resulting policy improvement step is thus $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha \boldsymbol{w}$ where α denotes a learning rate. Under an appropriate choice of α, this update converges to the next *local minimum in Riemannian space with probability 1*. The estimation of the natural gradient $\boldsymbol{w}$ is treated in Section 3.2.

When comparing these different policy gradients already for a simple continuous dynamic systems as in Figure 1, the 'vanilla' policy gradient will reach plateaus with low exploration (i.e., small σ) before the optimal controller gain k is computed, leading to premature convergence. On the other hand, the natural gradient points towards the optimal solution. For visibility, the gradients are normalized.

3.2 Critic Estimation with Compatible Policy Evaluation: LSTD-Q(λ)

The critic evaluates the current policy π in order to provide the basis for an actor improvement, i.e., the change $\Delta\boldsymbol{\theta}$ of the policy parameters. As we are interested in natural policy gradient updates $\Delta\boldsymbol{\theta} = \alpha \boldsymbol{w}$, we wish to employ the compatible function approximation $f_w^{\pi}(\boldsymbol{x}, \boldsymbol{u})$ from Eq.(5) in this context. Unfortunately, an obstacle in this regard is that the compatible function approximation $f_{\boldsymbol{w}}^{\pi}(\boldsymbol{x}, \boldsymbol{u})$ is mean-zero, i.e.,

$$\int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) f_{\boldsymbol{w}}^{\pi}(\boldsymbol{x}, \boldsymbol{u}) d\boldsymbol{u} = \int_{\mathbb{U}} \nabla_{\boldsymbol{\theta}} \pi(\boldsymbol{u}|\boldsymbol{x})^T d\boldsymbol{u} \boldsymbol{w} = 0, \tag{8}$$

since differentiating $\int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) d\boldsymbol{u} = 1$ implies $\int_{\mathbb{U}} \nabla_{\boldsymbol{\theta}} \pi(\boldsymbol{u}|\boldsymbol{x})^T d\boldsymbol{u} = 0$. Thus, $f_w^{\pi}(\boldsymbol{x}, \boldsymbol{u})$ represents in general an *advantage function* $A^{\pi}(\boldsymbol{x}, \boldsymbol{u}) = Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) - V^{\pi}(\boldsymbol{x})$. The advantage function *cannot* be learned with TD-like bootstrapping without knowledge of the value function as it is the essence of TD is to compare the value $V^{\pi}(\boldsymbol{x})$ of the two adjacent states – but this value has been subtracted out in $A^{\pi}(\boldsymbol{x}, \boldsymbol{u})$. Hence, a TD-like bootstrapping using exclusively the compatible function approximator is impossible.

As a way out, we observe that we can write the Bellman equations (e.g., see [3]) in terms of the advantage function and the state-value function

$$Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) = A^{\pi}(\boldsymbol{x}, \boldsymbol{u}) + V^{\pi}(\boldsymbol{x}) = r(\boldsymbol{x}, \boldsymbol{u}) + \gamma \int_{\mathbb{X}} p(\boldsymbol{x}'|\boldsymbol{x}, \boldsymbol{u}) V^{\pi}(\boldsymbol{x}') d\boldsymbol{x}'. \tag{9}$$

Table 1: Natural Actor-Critic Algorithm with LSTD-Q(λ)

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 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 $f_{\boldsymbol{w}}^{\pi}(\boldsymbol{x}_t, \boldsymbol{u}_t) = \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^T \boldsymbol{w}_t$. Update:
4.1: basis functions: $\widetilde{\boldsymbol{\phi}}_t = [\boldsymbol{\phi}(\boldsymbol{x}_{t+1})^T, \boldsymbol{0}^T]^T$; $\widehat{\boldsymbol{\phi}}_t = [\boldsymbol{\phi}(\boldsymbol{x}_t)^T, \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^T]^T$,
4.2: 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: critic parameters: $[\boldsymbol{w}_{t+1}^T, \boldsymbol{v}_{t+1}^T]^T = \boldsymbol{A}_{t+1}^{-1} \boldsymbol{b}_{t+1}$.
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.**

Inserting $A^{\pi}(\boldsymbol{x}, \boldsymbol{u}) = f_w^{\pi}(\boldsymbol{x}, \boldsymbol{u})$ 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., (9), as a set of linear equations

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^T \boldsymbol{w} + \boldsymbol{\phi}(\boldsymbol{x}_t)^T \boldsymbol{v} = \left\langle r(\boldsymbol{x}_t, \boldsymbol{u}_t) + \gamma \boldsymbol{\phi}(\boldsymbol{x}_{t+1})^T \boldsymbol{v} \right\rangle. \tag{10}$$

Using these simultaneous sets of linear equations, a solution to Equation (9) can be obtained by adapting the LSTD(λ) policy evaluation algorithm [6, 7, 12]. For this purpose, we define

$$\widehat{\boldsymbol{\phi}}_t = [\boldsymbol{\phi}(\boldsymbol{x}_t)^T, \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^T]^T, \ \widetilde{\boldsymbol{\phi}}_t = [\boldsymbol{\phi}(\boldsymbol{x}_{t+1})^T, \boldsymbol{0}^T]^T, \tag{11}$$

as new basis functions, where $\boldsymbol{0}$ is the zero vector. This definition of basis function reduces bias and variance of the learning process in comparison to SARSA and previous LSTD(λ) algorithms for state-action value functions [11] as the basis functions $\widetilde{\boldsymbol{\phi}}_t$ do not depend on stochastic 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 violates the standard regression model that only takes noise in the regression targets into account. Alternatively, Bradtke et al. [7] assume $V^{\pi}(\boldsymbol{x}) = Q^{\pi}(\boldsymbol{x}, \overline{\boldsymbol{u}})$ where $\overline{\boldsymbol{u}}$ is the average future action, and choose their basis functions accordingly; however, this is only given for deterministic policies, i.e., policies without exploration and not applicable in our framework. LSTD(λ) with the basis functions in Eq.(11), 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.

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.

3.3 Sensitivity to Imperfect Basis Functions

For a perfect state value function parameterization $V^{\pi}(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{v}$, LSTD-Q can be guaranteed to converge to the correct natural policy gradient, which follows directly from the results in [12]. The case of imperfect basis function, however, requires a more thorough inspection. We define $\boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x})$ to make the equations in this section more readable. For the case of $\lambda = 1$, i.e., Monte-Carlo rollouts, LSTD-Q needs to solve

$$\begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{v} \end{bmatrix} = \begin{bmatrix} \langle \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{u}) \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{u})^T \rangle & \boldsymbol{0} \\ \boldsymbol{0} & \langle \boldsymbol{\phi}(\boldsymbol{x}) \boldsymbol{\phi}(\boldsymbol{x})^T \rangle \end{bmatrix}^{-1} \begin{bmatrix} \langle \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{u}) \hat{Q}^{\pi}(\boldsymbol{x}, \boldsymbol{u}) \rangle \\ \langle \boldsymbol{\phi}(\boldsymbol{x}) \hat{Q}^{\pi}(\boldsymbol{x}, \boldsymbol{u}) \rangle \end{bmatrix},$$

where $\hat{Q}^\pi(\boldsymbol{x}, \boldsymbol{u})$ is an estimate for the true $Q^\pi(\boldsymbol{x}, \boldsymbol{u})$ obtained from summing up all rewards along a sample trajectory. As the cross terms $\langle\, \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{u})\boldsymbol{\phi}(\boldsymbol{x})^T \,\rangle = 0$ between both basis functions vanish, for LSTD-Q(1) the natural gradient estimation is independent of the choice of basis functions ϕ. This result corresponds to Konda and Tsitsiklis' [10] observation that the compatible function approximation can be seen as a projection of the state-action value function $Q^\pi(\boldsymbol{x}, \boldsymbol{u})$, i.e.,

$$\boldsymbol{w} = \text{argmin}_{\boldsymbol{\nu}} E_{\boldsymbol{x}\sim d^\pi(\boldsymbol{x}), \boldsymbol{u}\sim\pi(\boldsymbol{u}|\boldsymbol{x})}\{(Q^\pi(\boldsymbol{x}, \boldsymbol{u}) - f^\pi_{\boldsymbol{\nu}}(\boldsymbol{x}, \boldsymbol{u}))^2\}. \tag{12}$$

Interestingly, we note that $\langle\, \boldsymbol{\phi}(\boldsymbol{x})\hat{Q}^\pi(\boldsymbol{x}, \boldsymbol{u}) \,\rangle = \boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$, and $\langle\, \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{u})\boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{u})^T \,\rangle = F(\boldsymbol{\theta})$ when $\gamma \to 1$, which means that this solution for $\boldsymbol{w}$ corresponds to the definition of the natural gradient in Eq.(3) for the average reward case and for the discounted case when the process restarts from the stationary distribution [6]. However, learning purely from rollouts is rather inefficient. Unfortunately, for all other choices of λ, γ, and $\boldsymbol{\phi}(\boldsymbol{x})$, it is not easily possible to make an analytical statement of how the estimate of $\boldsymbol{w}$ is affected by the choice of basis functions. Thus, we conclude at the moment that basis function should be chosen such that they have a good representational power for the expected state-action value function, e.g., a quadratic polynomial expansion for LQR problems or a mixture of quadratic polynomial expansions for a mixture of local linear controllers.

4 The Natural Actor-Critic's Relation to Previous Algorithms

A surprising aspect about the Natural Actor-Critic (NAC) is its relation to previous algorithms. In this section, we briefly demonstrate that established algorithms like the classic Actor-Critic architecture[15], ϵ-soft SARSA [15], and Bradtke's Q-Learning [7] can be seen as special cases or approximations of NAC.

4.1 Original Actor-Critic and ϵ-soft SARSA approximates NAC

Sutton et al. [15] proposed the original actor-critic choosing a Gibbs policy $\pi(u_t|x_t) = \exp(\theta_{xu})/\sum_b \exp(\theta_{xb})$, with all parameters θ_{xu} lumped in the vector $\boldsymbol{\theta}$, denoted as $\boldsymbol{\theta} = [\theta_{xu}]$ from now on, in a discrete setup with tabular representations of transition probabilities and rewards. A linear function approximation $V^\pi(x) = \boldsymbol{\phi}(x)^T\boldsymbol{v}$ with $\boldsymbol{v} = [v_x]$ and basis functions $\boldsymbol{\phi}(x) = [0, ..., 0, 1, 0, ..., 0]^T$ was employed, where the 1 is in the element corresponding to state x. Sutton et al. online update rule is $\theta^{t+1}_{xu} = \theta^t_{xu} + \alpha_1\delta^\pi(x, u, x')$, $v^{t+1}_x = v^t_x + \alpha_2\delta^\pi(x, u, x')$, where $\delta^\pi(x, u, x') = (r(x, u) + \gamma V^\pi(x') - V^\pi(x))$ denotes the TD(0) error, and α_1, α_2 are learning rates. The update of the critic parameters v^t_x equals the one of the Natural Actor-Critic in expectation as TD(0) critics converges to the same values as LSTD(0) and LSTD-Q(0) for discrete problems [6]. For this Gibbs policy we have (i) $\partial \log \pi(b|a)/\partial\theta_{xu} = 1 - \pi(b|a)$ if $a = x$ and $b = u$, (ii) $\partial \log \pi(b|a)/\partial\theta_{xu} = -\pi(b|a)$ if $a = x$ and $b \neq u$, and (iii) $\partial \log \pi(b|a)/\partial\theta_{xu} = 0$ otherwise. Since, furthermore, $\sum_b \pi(b|x)A(x, b) = 0$, we can evaluate the advantage function and derive

$$A(x, u) = A(x, u) - \sum\nolimits_b \pi(b|x)A(x, b) = \sum\nolimits_b \frac{\partial \log \pi(b|x)}{\partial\theta_{xu}} A(x, b) = \frac{\partial \log \pi(u|x)}{\partial\boldsymbol{\theta}}^T \boldsymbol{a}.$$

with $\boldsymbol{a} = [A(x, u)]$. Since the compatible function approximation represents the advantage function, i.e., $f^\pi_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{u}) = A(x, u)$, we realize that the advantages equal the natural gradient, i.e., $\boldsymbol{a} = \boldsymbol{w}$. Furthermore, the TD(0) error of a state-action pair (x, u) equals the advantage function in expectation, and therefore the natural gradient update

$$w_{xu} = A(x, u) = E_{p(x'|x,u)}\{r(x, u) + \gamma V(x') - V(x)\} = E_{p(x'|x,u)}\{\delta^\pi(x, u, x')\},$$

corresponds to the average online updates of Actor-Critic. As both update rules of the Actor-Critic equal the one of NAC in expectation, it follows that *the Actor-Critic approximates the Natural Actor-Critic*.

SARSA with a tabular, discrete state-action value function $Q^\pi(x, u)$ and an ϵ-soft policy improvement $\pi(\boldsymbol{u}_t|\boldsymbol{x}_t) = \exp(Q^\pi(x, u)/\epsilon)/\sum_{\hat{u}} \exp(Q^\pi(x, u)/\epsilon)$ can also be seen as an approximation of NAC. When treating the table entries as parameters of a policy $\boldsymbol{\theta}_{xu} = Q^\pi(x, u)$, we realize that the TD update of these parameters corresponds approximately to the natural gradient update since $w_{xu} = \epsilon A(x, u) \approx \epsilon E_{p(x'|x,u)}\{r(x, u) + \gamma Q(x', u') - Q(x, u)\}$, i.e., *$\epsilon$-soft SARSA with lookup-tables can be seen as an approximation of NAC*.

4.2 Bradtke's Q-Learning for LQR is a Special Case of NAC

Bradtke et al. [7] propose an algorithm with policy $\pi(u_t|\boldsymbol{x}_t) = \mathcal{N}(u_t|\boldsymbol{k}_i^T\boldsymbol{x}_t, \sigma_i^2)$ and parameters $\boldsymbol{\theta}_i = [\boldsymbol{k}_i^T, \sigma_i]^T$ (where σ_i denotes the exploration, and i the policy update time step) in a linear control task with linear state

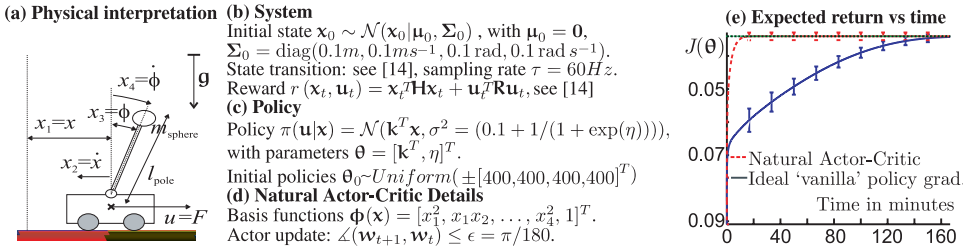

Figure 2: A comparison of Natural Actor Critic to policy gradients methods on a standard cart-pole balancing [15, 14] task. For each actor update of NAC, an update along the true 'vanilla' policy gradient is performed with the maximal convergent learning rate – and NAC still outperforms the policy gradient.

transitions $\boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{b}u_t$, and quadratic rewards $r(\boldsymbol{x}_t, \boldsymbol{u}_t) = \boldsymbol{x}_t^T\boldsymbol{H}\boldsymbol{x}_t + Ru_t^2$. They evaluated $Q^\pi(\boldsymbol{x}_t, \boldsymbol{u}_t)$ with LSTD(0) using a quadratic polynomial expansion as basis functions, and applied greedy updates:

$$\boldsymbol{k}_{i+1} = \mathrm{argmax}_{\boldsymbol{k}_{i+1}} Q^\pi(\boldsymbol{x}_t, \boldsymbol{u}_t = \boldsymbol{k}_{i+1}^T\boldsymbol{x}_t) = -(R + \gamma\boldsymbol{b}^T\boldsymbol{P}_i\boldsymbol{b})^{-1}\gamma\boldsymbol{b}\boldsymbol{P}_i\boldsymbol{A}, \tag{13}$$

where $\boldsymbol{P}_i$ denotes policy-specific value function parameters related to the gain $\boldsymbol{k}_i$; no update the exploration σ_i was included. After inserting the compatible function approximation in Eqn. (9) and solving analytically for the natural gradient $\boldsymbol{w}$, we obtain

$$\boldsymbol{w} = [\boldsymbol{w}_{\boldsymbol{k}}, w_\sigma]^T = [-(\gamma\boldsymbol{A}^T\boldsymbol{P}_i\boldsymbol{b} + (R + \gamma\boldsymbol{b}^T\boldsymbol{P}_i\boldsymbol{b})\boldsymbol{k})^T\sigma_i^2, -0.5(\boldsymbol{R} + \gamma\boldsymbol{b}^T\boldsymbol{P}_i\boldsymbol{b})\sigma_i^3]. \tag{14}$$

One can derive that the expected return is $J(\boldsymbol{\theta}_i) = -(R + \gamma\boldsymbol{b}^T\boldsymbol{P}_i\boldsymbol{b})\sigma_i^2$ for this type of problems. For a learning rate $\alpha_i = 1/\left\|J(\boldsymbol{\theta}_i)\right\|$, we see that

$$\boldsymbol{k}_{i+1} = \boldsymbol{k}_i + \alpha_t\boldsymbol{w}_{\boldsymbol{k}} = \boldsymbol{k}_i - (\boldsymbol{k}_i + (R + \gamma\boldsymbol{b}^T\boldsymbol{P}_i\boldsymbol{b})^{-1}\gamma\boldsymbol{A}^T\boldsymbol{P}_i\boldsymbol{b}) = (R + \gamma\boldsymbol{b}^T\boldsymbol{P}_i\boldsymbol{b})^{-1}\gamma\boldsymbol{A}^T\boldsymbol{P}_i\boldsymbol{b},$$

which demonstrates that *Bradtke's Actor Update is a special case of the Natural Actor-Critic*. NAC extends Bradtke's result as it gives an update rule for the exploration, correctly deals with stochastic policies in LSTD-Q, and allows smaller learning rates as needed for nonlinear extensions.

5 Empirical Evaluation & Conclusion

We evaluated the Natural Actor-Critic algorithm on the standard benchmark of cart-pole balancing [15, 14] as described in Figure 2. We compare the results of the Natural Actor Critic algorithm to the results of the ideal 'vanilla' policy gradient which can be computed analytically; both algorithms started with the same initial policy parameters, and the true 'vanilla' policy gradient updates were done simultaneaously with each NAC update. The true policy gradient used the maximal learning rate where it would still converge. Still, the NAC converged significantly faster as can be seen in Figure 2. 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.

In summary, we focussed on the theoretical development of a novel 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, e.g., with sufficiently rich basis functions, this method guarantees convergence with probability one to the next local minimum in Riemannian space. We apply this algorithm successfully in a simple robotic task, i.e., pole balancing. By the time of the NIPS 2003 conference, we hope to augment this paper by an implementation of the pole-balancing task on an actual anthropomorphic robot, and by presenting extensions of the suggested framework to nonlinear control systems by means of embedding the Natural Actor-Critic in a mixture model approach.

References

[1] Amari, S. (1998) Natural Gradient Works Efficiently in Learning. Neural Computation 10

[2] Baird, L. C., & Moore, A. W. (1999). Gradient descent for general reinforcement learning. Advances in Neural Information Processing Systems 11.

[3] Baird, L. C. (1993) Advantage Updating. Technical Report WL-TR-93-1146, Wright-Patterson Air Force Base Ohio: Wright Laboratory.

[4] Bagnell, J., Schneider, J. (2003) Covariant Policy Search. Submitted to IJCAI.

[5] Bertsekas, D. P. & Tsitsiklis, J. N.(1996) Neuro-Dynamic Programming. Athena Scientific, Belmont, MA.

[6] Boyan, J.,(1999) Least-squares temporal difference learning. ICML.

[7] Bradtke, S., Ydstie, E. and Barto, A.(1994) Adaptive Linear Quadratic Control Using Policy Iteration. Technical report UM-CS-1994-049, University of Massachusetts.

[8] Kakade, S.(2002) A Natural Policy Gradient. Advances in Neural Information Processing Systems 14.

[9] Kakade, S.(2002) Approximately Optimal Reinforcement Learning. ICML.

[10] Konda, V., and Tsitsiklis, J.N.(2000) Actor-Critic Algorithms. In Advances in Neural Information Processing Systems 12. MIT Press.

[11] Lagoudakis, M., and Parr, R., (2001) Model-Free Least-Squares policy iteration, Technical Report CS-2000-05, Duke University.

[12] Nedic, A. and Bertsekas, D. P.,(2002) Least-Squares Policy Evaluation Algorithms with Linear Function Approximation, to appear in J. of Discrete Event Systems.

[13] Peters, J., Vijaykumar, S., Schaal, S.(2003) Reinforcement Learning for Humanoid Robotics. IEEE Humandoids Proceedings.

[14] Schaal, S.(1997) Learning from demonstration, Advances in Neural Information Processing Systems 9, MIT Press.

[15] Sutton, R.S., and Barto, A.G.(1998) Reinforcement Learning, The MIT Press.

[16] Sutton, R.S., McAllester, D., Singh, S., and Mansour, Y.(2000) Policy gradient methods for reinforcement learning with function approximation. In Advances in Neural Information Processing Systems 12. MIT Press.

[1] In here we add the proof that the all-action matrix $F(\boldsymbol{\theta})$ in general equals the Fisher information matrix $G(\boldsymbol{\theta})$. By differentiating $\int_{\mathbb{R}^n} p(\boldsymbol{x})d\boldsymbol{x} = 1$ twice with respect to the parameters $\boldsymbol{\theta}$, we obtain $\int_{\mathbb{R}^n} p(\boldsymbol{x})\nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{x})d\boldsymbol{x} = -\int_{\mathbb{R}^n} p(\boldsymbol{x})\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x})\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x})^T d\boldsymbol{x}$ for any probability density function $p(\boldsymbol{x})$. The probability $p(\boldsymbol{\tau}_{0:n})$ of a trajectory $\boldsymbol{\tau}_{0:n} = [\boldsymbol{x}_0, \boldsymbol{u}_0, r_0, \ldots, \boldsymbol{x}_n, \boldsymbol{u}_n, r_n, \boldsymbol{x}_{n+1}]^T$ is given as $p(\boldsymbol{\tau}_{0:n}) = p(\boldsymbol{x}_0)\prod_{t=0}^{n} p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)\pi(\boldsymbol{u}_t|\boldsymbol{x}_t) \Rightarrow \nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}_{0:n}) = \sum_{t=0}^{n} \nabla^2_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)$. Combining these facts, and using the definition of the Fisher information matrix [1], we can determine Fisher information matrix for the average reward case, i.e,

$$\begin{aligned}
G(\boldsymbol{\theta}) &= \lim_{n\to\infty} n^{-1} E_{\boldsymbol{\tau}_{0:n}}\{\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}_{0:n}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}_{0:n})^T\} \\
&= -\lim_{n\to\infty} n^{-1} E_{\boldsymbol{\tau}_{0:n}}\{\nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}_{0:n})\}, \\
&= -\lim_{n\to\infty} n^{-1} E_{\boldsymbol{\tau}_{0:n}}\{\textstyle\sum_{t=0}^{n} \nabla^2_{\boldsymbol{\theta}} \log \pi\left(\boldsymbol{u}_t\,|\boldsymbol{x}_t\right)\} \\
&= -\textstyle\int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \nabla^2_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) d\boldsymbol{u} d\boldsymbol{x}, \\
&= \textstyle\int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x})^T d\boldsymbol{u} d\boldsymbol{x} = F(\boldsymbol{\theta}),
\end{aligned} \tag{15}$$

This 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 rollout is given by $p_\gamma(\boldsymbol{\tau}_{0:n}) = p(\boldsymbol{\tau}_{0:n})(\sum_{i=0}^{n} \gamma^i 1_{x_i,u_i})$. It is straightforward to show that $\nabla^2_{\boldsymbol{\theta}} \log p\left(\boldsymbol{\tau}_{0:n}\right) = \nabla^2_{\boldsymbol{\theta}} \log p_\gamma(\boldsymbol{\tau}_{0:n})$. This rewritten probability allows us to repeat the transformations from before, and show that again the all-action matrix equals the Fisher information matrix. Therefore, we can conclude that for both the average reward and the discounted case, these Fisher information and all-action matrix are the same, i.e., $G(\boldsymbol{\theta}) = F(\boldsymbol{\theta})$. A similar theorem has been derived in [13, 4].