Natural Actor-Critic

Jan Peters[1], Sethu Vijayakumar[2], and Stefan Schaal[1]

[1] University of Southern California, Los Angeles CA 90089, USA
[2] University of Edinburgh, Edinburgh EH9 3JZ, United Kingdom

Abstract. This paper investigates a novel model-free reinforcement learning architecture, the Natural Actor-Critic. The actor updates are based on stochastic policy gradients employing Amari's natural gradient approach, while the critic obtains both the natural policy gradient and additional parameters of a value function simultaneously by linear regression. We show that actor improvements with natural policy gradients are particularly appealing as these are independent of coordinate frame of the chosen policy representation, and can be estimated more efficiently than regular policy gradients. The critic makes use of a special basis function parameterization motivated by the policy-gradient compatible function approximation. We show that several well-known reinforcement learning methods such as the original Actor-Critic and Bradtke's Linear Quadratic Q-Learning are in fact Natural Actor-Critic algorithms. Empirical evaluations illustrate the effectiveness of our techniques in comparison to previous methods, and also demonstrate their applicability for learning control on an anthropomorphic robot arm.

1 Introduction

Reinforcement learning algorithms based on value function approximation have been highly successful with discrete lookup table parameterization. However, when applied with continuous function approximation, many of these algorithms failed to generalize, and few convergence guarantees could be obtained [14]. The reason for this problem can largely be traced back to the greedy or ϵ-greedy policy updates of most techniques, as it does not ensure a policy improvement when applied with an approximate value function [6]. During a greedy update, small errors in the value function can cause large changes in the policy which in return can 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 unfortunate behavior can be found in many well-known greedy reinforcement learning algorithms [4,6].

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 approximate value functions, and recent results created a theoretically solid framework for policy gradient estimation from sampled data [15,11]. However, even when applied to simple examples with rather few states, policy gradient methods often turn out to be quite inefficient

J. Gama et al. (Eds.): ECML 2005, LNAI 3720, pp. 280–291, 2005.

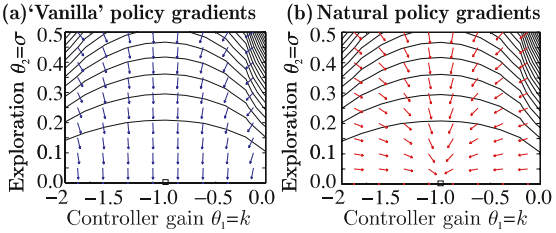

Fig. 1. When plotting the expected return landscape for simple problem as 1d linear quadratic regulation, the differences between 'vanilla' and natural policy gradients becomes apparent [13]

[10], partially caused by the large plateaus in the expected return landscape where the gradients are small and often do not point directly towards the optimal solution. A simple example that demonstrates this behavior is given in Fig. 1.

Similar as in supervised learning, the steepest ascent with respect to the Fisher information metric [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' in [10], and subsequently shown in preliminary work to be the true natural policy gradient [13,2]. In this paper, we take this line of reasoning one step further in Section 2.2 by introducing the "Natural Actor-Critic" which inherits the convergence guarantees from gradient methods. Furthermore, in Section 3, we show that several successful previous reinforcement learning methods can be seen as special cases of this more general architecture. The paper concludes with empirical evaluations that demonstrate the effectiveness of the suggested methods in Section 4.

2 Natural Actor-Critic

2.1 Markov Decision Process Notation and Assumptions

For this paper, 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 $\mathbb{U} = \mathbb{R}^m$ [6]. 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 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 are given by

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

where $\gamma \in [0, 1[$ denotes the discount factor, and τ a trajectory. 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 normalized expected return

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

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

2.2 Actor Improvements with Natural Policy Gradients

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. The policy improvement step is required to improve the policy on every step until convergence while being efficient.

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., [15,11]) which follow the gradient $\boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ of the expected return function $J(\boldsymbol{\theta})$ often get stuck in plateaus as demonstrated in [10]. Natural gradients $\widetilde{\boldsymbol{\nabla}}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ avoid this pitfall as demonstrated for supervised learning problems [1], and suggested for reinforcement learning in [10]. These methods do not follow the steepest direction in parameter space but the steepest direction with respect to the Fisher metric given by

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

where $\boldsymbol{G}(\boldsymbol{\theta})$ 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 assured. The 'vanilla' gradient is given by the policy gradient theorem (see e.g., [15,11]),

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^\pi(\boldsymbol{x}) \int_{\mathbb{U}} \boldsymbol{\nabla}_{\boldsymbol{\theta}} \pi(\boldsymbol{u}|\boldsymbol{x}) \left(Q^\pi(\boldsymbol{x}, \boldsymbol{u}) - b^\pi(\boldsymbol{x}) \right) d\boldsymbol{u} d\boldsymbol{x}, \tag{2}$$

where $b^\pi(\boldsymbol{x})$ denotes a baseline. [15] and [11] demonstrated that in Eq. (2), the term $Q^\pi(\boldsymbol{x}, \boldsymbol{u}) - b^\pi(\boldsymbol{x})$ can be replaced by a compatible function approximation

$$f_w^\pi(\boldsymbol{x}, \boldsymbol{u}) = \left(\boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) \right)^T \boldsymbol{w} \equiv Q^\pi(\boldsymbol{x}, \boldsymbol{u}) - b^\pi(\boldsymbol{x}), \tag{3}$$

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 [15], the baseline may still be useful in order to reduce the variance of the gradient estimate when Eq.(2) is approximated from samples. Based on Eqs.(2, 3), we derive an estimate of the policy gradient as

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^\pi(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x})^T d\boldsymbol{u} d\boldsymbol{x} \, \boldsymbol{w} = \boldsymbol{F}_{\boldsymbol{\theta}} \boldsymbol{w}. \tag{4}$$

as $\nabla_{\boldsymbol{\theta}}\pi(\boldsymbol{u}|\boldsymbol{x}) = \pi(\boldsymbol{u}|\boldsymbol{x})\nabla_{\boldsymbol{\theta}}\log\pi(\boldsymbol{u}|\boldsymbol{x})$. 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 executing all actions. It is also noteworthy that the baseline does not appear in Eq. (4) as it integrates out, thus eliminating the need to find an optimal selection of this open parameter. Nevertheless, the estimation of $F_{\boldsymbol{\theta}} = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x})F(\boldsymbol{\theta},\boldsymbol{x})d\boldsymbol{x}$ is still expensive since $d^{\pi}(\boldsymbol{x})$ ist not known. However, Equation (4) has more surprising implications for policy gradients, when examining the meaning of the matrix $F_{\boldsymbol{\theta}}$ in Eq.(4). Kakade [10] 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 a weighted 'average Fisher information matrix'[10]. However, going one step further, we demonstrate in Appendix A 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.(4) and (1) combined imply that the natural gradient can be computed as

$$\widetilde{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = G^{-1}(\boldsymbol{\theta})F_{\boldsymbol{\theta}}\boldsymbol{w} = \boldsymbol{w}, \tag{5}$$

since $F_{\boldsymbol{\theta}} = G(\boldsymbol{\theta})$ (c.f. Appendix A). Therefore we only need estimate $\boldsymbol{w}$ and *not* $G(\boldsymbol{\theta})$. The resulting policy improvement step is thus $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha\boldsymbol{w}$ where α denotes a learning rate. Several properties of the natural policy gradient are worthwhile highlighting:

- Convergence to a local minimum guaranteed as for 'vanilla gradients'. [1]
- By choosing a more direct path to the optimal solution in parameter space, the natural gradient has, from empirical observations, faster convergence and avoids premature convergence of 'vanilla gradients' (cf. Figure 1).
- The natural policy gradient can be shown to be **covariant**, i.e., independent of the coordinate frame chosen for expressing the policy parameters (cf. Section 3.1).
- As the natural gradient analytically averages out the influence of the stochastic policy (including the baseline of the function approximator), it requires fewer data point for a good gradient estimate than 'vanilla gradients'.

2.3 Critic Estimation with Compatible Policy Evaluation

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.(3) in this context. At this point, a most important observation is that the compatible function approximation $f_{\boldsymbol{w}}^{\pi}(\boldsymbol{x},\boldsymbol{u})$ is mean-zero w.r.t. the action distribution, i.e.,

$$\int_{\mathbb{U}}\pi(\boldsymbol{u}|\boldsymbol{x})f_{\boldsymbol{w}}^{\pi}(\boldsymbol{x},\boldsymbol{u})d\boldsymbol{u} = \boldsymbol{w}^T\int_{\mathbb{U}}\nabla_{\boldsymbol{\theta}}\pi(\boldsymbol{u}|\boldsymbol{x})d\boldsymbol{u} = 0, \tag{6}$$

since from $\int_{\mathbb{U}}\pi(\boldsymbol{u}|\boldsymbol{x})d\boldsymbol{u} = 1$, differention w.r.t. to θ results in $\int_{\mathbb{U}}\nabla_{\boldsymbol{\theta}}\pi(\boldsymbol{u}|\boldsymbol{x})d\boldsymbol{u} = \mathbf{0}$. Thus, $f_w^{\pi}(\boldsymbol{x},\boldsymbol{u})$ represents an *advantage function* $A^{\pi}(\boldsymbol{x},\boldsymbol{u}) = Q^{\pi}(\boldsymbol{x},\boldsymbol{u}) - V^{\pi}(\boldsymbol{x})$ in general. The advantage function *cannot* be learned with TD-like bootstrapping

without knowledge of the value function as 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 an alternative, [15,11] suggested to approximate $f_w^\pi(\boldsymbol{x}, \boldsymbol{u})$ from unbiased estimates $\hat{Q}^\pi(\boldsymbol{x}, \boldsymbol{u})$ of the action value function, e.g., obtained from roll-outs and using least-squares minimization between f_w and $\hat{Q}^\pi$. While possible in theory, one needs to realize that this approach implies a function approximation problem where the parameterization of the function approximator only spans a much smaller subspace of the training data – e.g., imagine approximating a quadratic function with a line. In practice, the results of such an approximation depends crucially on the training data distribution and has thus unacceptably high variance – e.g., fit a line to only data from the right branch of a parabula, the left branch, or data from both branches.

To remedy this situation, we observe that we can write the Bellman equations (e.g., see [3]) in terms of the advantage function and the state-value function

$$Q^\pi(\boldsymbol{x}, \boldsymbol{u}) = A^\pi(\boldsymbol{x}, \boldsymbol{u}) + V^\pi(\boldsymbol{x}) = r(\boldsymbol{x}, \boldsymbol{u}) + \gamma \int_{\mathbb{X}} p(\boldsymbol{x}'|\boldsymbol{x}, \boldsymbol{u}) V^\pi(\boldsymbol{x}') d\boldsymbol{x}'. \quad (7)$$

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., (7), as a set of linear equations

$$\boldsymbol{\nabla_\theta} \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^T \boldsymbol{w} + \boldsymbol{\phi}(\boldsymbol{x}_t)^T \boldsymbol{v} = r(\boldsymbol{x}_t, \boldsymbol{u}_t) + \gamma \boldsymbol{\phi}(\boldsymbol{x}_{t+1})^T \boldsymbol{v} + \epsilon(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{x}_{t+1}) \quad (8)$$

where $\epsilon(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{x}_{t+1})$ denotes an error term which mean-zero as can be observed from Eq.(7). These equations enable us to formulate some novel algorithms in the next sections.

Critic Evaluation with LSTD-Q($\boldsymbol{\lambda}$). Using Eq.(8), a solution to Equation (7) can be obtained by adapting the LSTD(λ) policy evaluation algorithm [7].

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 $\boldsymbol{\nabla_\theta} \log \pi(\boldsymbol{u}|\boldsymbol{x})$ and basis functions $\boldsymbol{\phi}(\boldsymbol{x})$ for the value function $V^\pi(\boldsymbol{x})$.
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(λ)):** Update
4.1: basis functions: $\widetilde{\boldsymbol{\phi}}_t = [\boldsymbol{\phi}(\boldsymbol{x}_{t+1})^T, \boldsymbol{0}^T]^T$, $\widehat{\boldsymbol{\phi}}_t = [\boldsymbol{\phi}(\boldsymbol{x}_t)^T, \boldsymbol{\nabla_\theta} \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^T]^T$,
4.2: 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:** When the natural gradient is converged, $\measuredangle(\boldsymbol{w}_{t+1}, \boldsymbol{w}_{t-\tau}) \leq \epsilon$, update
5.1: policy parameters: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \boldsymbol{w}_{t+1}$,
5.2: forget statistics: $\boldsymbol{z}_{t+1} \leftarrow \beta \boldsymbol{z}_{t+1}, \boldsymbol{A}_{t+1} \leftarrow \beta \boldsymbol{A}_{t+1}, \boldsymbol{b}_{t+1} \leftarrow \beta \boldsymbol{b}_{t+1}$.
6: **end.**

For this purpose, we define

$$\widehat{\boldsymbol{\phi}}_t = [\boldsymbol{\phi}(\boldsymbol{x}_t)^T, \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^T]^T, \;\; \widetilde{\boldsymbol{\phi}}_t = [\boldsymbol{\phi}(\boldsymbol{x}_{t+1})^T, \mathbf{0}^T]^T, \tag{9}$$

as new basis functions, where $\mathbf{0}$ 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 [7] 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 $\boldsymbol{u}_{t+1}$ (e.g., as in [8]) – 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.(9), 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. 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 $A^\pi(\boldsymbol{x}_t, \boldsymbol{u}_t) + V^\pi(\boldsymbol{x}_t)$, 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 (NAC) algorithm is shown in Table 1.

However, it becomes fairly obvious that the basis functions can have an influence on our gradient estimate. When using the counterexample in [5] with a typical Gibbs policy, we will realize that the gradient is affected for $\lambda < 1$; for $\lambda = 0$ the gradient is flipped and would always worsen the policy. However, unlike in [5], we at least could guarantee that we are not affected for $\lambda = 1$.

Episodic Natural Actor-Critic. Given the problem that the additional basis functions $\phi(\boldsymbol{x})$ determine the quality of the gradient, we need methods which guarantee the unbiasedness of the natural gradient estimate. Such method can be determined by summing up Equation (8) along a sample path, we obtain

$$\textstyle\sum_{t=0}^{N-1} \gamma^t A^\pi(\boldsymbol{x}_t, \boldsymbol{u}_t) = V^\pi(\boldsymbol{x}_0) + \sum_{t=0}^{N-1} \gamma^t r(\boldsymbol{x}_t, \boldsymbol{u}_t) - \gamma^N V^\pi(\boldsymbol{x}_N) \tag{10}$$

It is fairly obvious that the last term disappears for $N \to \infty$ or episodic tasks (where $r(\boldsymbol{x}_{N-1}, \boldsymbol{u}_{N-1})$ is the final reward); therefore each roll-out would yield one equation. If we furthermore assume a single start-state, an additional scalar value function of $\phi(x) = 1$ suffices. We therefore get a straightforward regression problem:

$$\textstyle\sum_{t=0}^{N-1} \gamma^t \boldsymbol{\nabla} \log \pi(\boldsymbol{u}_t, \boldsymbol{x}_t)^T \boldsymbol{w} + J = \sum_{t=0}^{N-1} \gamma^t r(\boldsymbol{x}_t, \boldsymbol{u}_t) \tag{11}$$

with exactly $\dim \theta + 1$ unknowns. This means that for non-stochastic tasks we can obtain a gradient after $\dim \theta + 1$ rollouts. The complete algorithm is shown in Table 2.

Table 2. Episodic Natural Actor-Critic Algorithm (eNAC)

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 $\boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}\|\boldsymbol{x})$.

For $u = 1, 2, 3, \ldots$ **do**
  For $e = 1, 2, 3, \ldots$ **do**
  Execute Rollout: Draw initial state $\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0)$.
  For $t = 1, 2, 3, \ldots, N$ **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.
  end.
  Critic Evaluation (Episodic): Determine value function $J = V^{\pi}(\boldsymbol{x}_0)$, compatible function approximation $f_{\boldsymbol{w}}^{\pi}(\boldsymbol{x}_t, \boldsymbol{u}_t)$.
  Update: Determine basis functions: $\boldsymbol{\phi}_t = \left[\sum_{t=0}^{N} \gamma^t \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}_t\|\boldsymbol{x}_t)^T, 1\right]^T$;
  reward statistics: $R_t = \sum_{t=0}^{N} \gamma^t r$;
  Actor-Update: When the natural gradient is converged, $\measuredangle(\boldsymbol{w}_{t+1}, \boldsymbol{w}_{t-\tau}) \leq \epsilon$, update the policy parameters: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \boldsymbol{w}_{t+1}$.
6: **end.**

3 Properties of Natural Actor -Critic

In this section, we will emphasize certain properties of the natural actor-critic. In particular, we want to give a simple proof of covariance of the natural policy gradient, and discuss [10] observation that in his experimental settings the natural policy gradient was non-covariant. Furthermore, we will discuss another surprising aspect about the Natural Actor-Critic (NAC) which is its relation to previous algorithms. We briefly demonstrate that established algorithms like the classic Actor-Critic [14], and Bradtke's Q-Learning [8] can be seen as special cases of NAC.

3.1 On the Covariance of Natural Policy Gradients

When [10] originally suggested natural policy gradients, he came to the disappointing conclusion that they were not covariant. As counterexample, he suggested that for two different linear Gaussian policies, (one in the normal form, and the other in the information form) the probability distributions represented by the natural policy gradient would be affected differently, i.e., the natural policy gradient would be non-covariant. We intend to give a proof at this point showing that the natural policy gradient is in fact covariant under certain conditions, and clarify why [10] experienced these difficulties.

Theorem 1. *Natural policy gradients updates are covariant for two policies* $\pi_{\boldsymbol{\theta}}$ *parameterized by* $\boldsymbol{\theta}$ *and* $\pi_{\boldsymbol{h}}$ *parameterized by* $\boldsymbol{h}$ *if (i) for all parameters* θ_i *there exists a function* $\theta_i = f_i(h_1, \ldots, h_k)$*, (ii) the derivative* $\boldsymbol{\nabla}_{\boldsymbol{h}} \boldsymbol{\theta}$ *and its inverse* $\boldsymbol{\nabla}_{\boldsymbol{h}} \boldsymbol{\theta}^{-1}$.

For the proof see Appendix B. Practical experiments show that the problems occurred for Gaussian policies in [10] are in fact due to the selection the stepsize α which determines the length of $\boldsymbol{\Delta\theta}$. As the linearization $\boldsymbol{\Delta\theta} = \boldsymbol{\nabla_h\theta}^T \boldsymbol{\Delta h}$ does not hold for large $\boldsymbol{\Delta\theta}$, this can cause divergence between the algorithms even for analytically determined natural policy gradients which can partially explain the difficulties occurred by Kakade [10].

3.2 NAC's Relation to Previous Algorithms

Original Actor-Critic. Surprisingly, the original Actor-Critic algorithm [14] is a form of the Natural Actor-Critic. By 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 $\boldsymbol{\theta} = [\theta_{xu}]$) 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 unit basis functions $\boldsymbol{\phi}(x) = \boldsymbol{u}_x$ was employed. Sutton et al. online update rule is given by

$$\theta_{xu}^{t+1} = \theta_{xu}^t + \alpha_1 \left(r(x,u) + \gamma v_{x'} - v_x\right), v_x^{t+1} = v_x^t + \alpha_2 \left(r(x,u) + \gamma v_{x'} - v_x\right),$$

where α_1, α_2 denote learning rates. The update of the critic parameters v_x^t 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 [7]. Since for the Gibbs policy we have $\partial \log \pi(b|a)/\partial\theta_{xu} = 1 - \pi(b|a)$ if $a = x$ and $b = u$, $\partial \log \pi(b|a)/\partial\theta_{xu} = -\pi(b|a)$ if $a = x$ and $b \neq u$, and $\partial \log \pi(b|a)/\partial\theta_{xu} = 0$ otherwise, and as $\sum_b \pi(b|x)A(x,b) = 0$, we can evaluate the advantage function and derive

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

Since the compatible function approximation represents the advantage function, i.e., $f_{\boldsymbol{w}}^\pi(\boldsymbol{x},\boldsymbol{u}) = A(x,u)$, we realize that the advantages equal the natural gradient, i.e., $\boldsymbol{w} = [A(x,u)]$. Furthermore, the TD(0) error of a state-action pair (x,u) equals the advantage function in expectation, and therefore the natural gradient update$w_{xu} = A(x,u) = E_{x'}\{r(x,u) + \gamma V(x') - V(x)|x,u\}$, corresponds to the average online updates of Actor-Critic. As both update rules of the Actor-Critic correspond to the ones of NAC, we can see both algorithms as equivalent.

Bradtke's Q-Learning. Bradtke [8] proposed 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 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:

$$\boldsymbol{k}_{i+1}^{\text{Bradtke}} = \text{argmax}_{\boldsymbol{k}_{i+1}} Q^\pi(\boldsymbol{x}_t, \boldsymbol{u}_t = \boldsymbol{k}_{i+1}^T \boldsymbol{x}_t) = -(R + \gamma \boldsymbol{b}^T \boldsymbol{P}_i \boldsymbol{b})^{-1} \gamma \boldsymbol{b} \boldsymbol{P}_i \boldsymbol{A}, \quad (12)$$

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. Similarly, we can obtain the natural

policy gradient $\boldsymbol{w} = [\boldsymbol{w_k}, w_\sigma]^T$, as yielded by LSTD-Q($\lambda$) analytically using the compatible function approximation and the same quadratic basis functions. As discussed in detail in [13], this gives us

$$\boldsymbol{w_k} = (\gamma \boldsymbol{A}^T \boldsymbol{P}_i \boldsymbol{b} + (R + \gamma \boldsymbol{b}^T \boldsymbol{P}_i \boldsymbol{b})\boldsymbol{k})^T \sigma_i^2, w_\sigma = 0.5(\boldsymbol{R} + \gamma \boldsymbol{b}^T \boldsymbol{P}_i \boldsymbol{b})\sigma_i^3. \quad (13)$$

Similarly, it can be derived 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, see [13]. For a learning rate $\alpha_i = 1/\left\|J(\boldsymbol{\theta}_i)\right\|$, we see

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

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 – which was not possible in Bradtke's greedy framework.

4 Evaluations and Applications

In this section, we present several evaluations comparing the episodic Natural Actor-Critic architectures with previous algorithms. We compare them in optimization tasks such as cart-pole balancing and simple motor primitive evaluations and compare them only with episodic NAC. Furthermore, we apply the combination of episodic NAC and the motor primitive framework to a robotic task on a real robot, i.e., 'hitting a T-ball with a baseball bat'.

4.1 Cart-Pole Balancing

Cartpole balancing is a well-known benchmark for reinforcement learning. We assume the cart as shown in Figure 2 (1.a) can be described by

$$ml\ddot{x}\cos\theta + ml^2\ddot{\theta} - mgl\sin\theta = 0, (m + m_c)\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F,$$

with $l = 0.75$m, $m = 0.15$kg, $g = 9.81$m/s^2 and $m_c = 1.0$kg. The resulting state is given by $\boldsymbol{x} = [x, \dot{x}, \theta, \dot{\theta}]^T$, and the action $\boldsymbol{u} = F$. The system is treated as

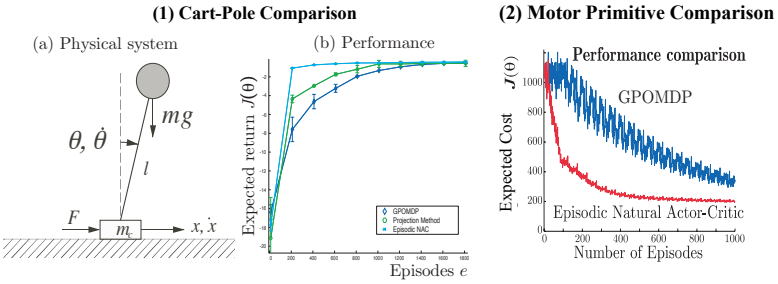

Fig. 2. This figure shows two comparisons, (1) cart-pole and (2) motor primitive learning. The physical set-up of a cart-pole balancing is shown in (1.a), and in (1.b) the performance of GPOMDP, the projection natural gradient, and the Episodic Natural Actor-Critic in comparison. The latter clearly outperforms the first two.

if it was sampled at a rate of $h = 60$Hz, and the reward is given by $r(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{u}^T \boldsymbol{R} \boldsymbol{u}$ with $\boldsymbol{Q} = \text{diag}(1.25, 1, 12, 0.25)$, $\boldsymbol{R} = 0.01$. We chose a linear Gaussian policy given by $\pi(\boldsymbol{u}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{k}^T \boldsymbol{x}, 1/(1+\exp(-\xi)))$, with parameters $\boldsymbol{\theta}^T = [\boldsymbol{k}^T, \xi]$. While this can also be treated with LSTD-Q(λ), see [13], we will focus on comparing it with GPOMDP and the projection suggested in [11], and in [10]. The results can be seen in Figure 2 (1.b) which makes clear that episodic natural actor-critic clearly outperforms both other methods.

4.2 Motor Primitive Learning for Baseball

This section will turn towards optimizing nonlinear dynamic motor primitives for robotics. In [9], a novel form of representing movement plans $(\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d)$ for the degrees of freedom (DOF) robot systems was suggested in terms of the time evolution of the nonlinear dynamical systems

$$\dot{q}_{d,k} = h(q_{d,k}, \boldsymbol{z}_k, g_k, \tau, \theta_k) \tag{14}$$

where $(q_{d,k}, \dot{q}_{d,k})$ denote the desired position and velocity of a joint, z_k the internal state of the dynamic system, g_k the goal (or point attractor) state of each DOF, τ the movement duration shared by all DOFs, and θ_k the open parameters of the function h. The original work in[9] 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.(14) by means of reinforcement learning, which is the crucial second step in imitation learning. The system in Eq.(14) is a point-to-point movement, i.e., this task is rather well suited for episodic Natural Actor-Critic. In Figure 2 (2), we show a comparison with GPOMDP for simple, single DOF task with a reward of $r_k(x_{0:N}, u_{0:N}) = \sum_{i=0}^{N} c_1 \dot{q}_{d,k,i}^2 + c_2 (q_{d;k;N} - g_k)^2$; where $c_1 = 1$, $c_2 = 1000$, and g_k is chose appropriately. We also evaluated the same setup in a challenging robot task, i.e., the planning of these motor primitives for a seven DOF robot task. The task of the robot is to hit the ball properly so that it flies as far as possible. Initially, it is taught in by supervised learning as can be seen in Figure 3 (b); however, it fails to reproduce the behavior as shown in (c); subsequently, we improve the performance using the episodic Natural Actor-Critic which yields the performance shown in (a) and the behavior in (d).

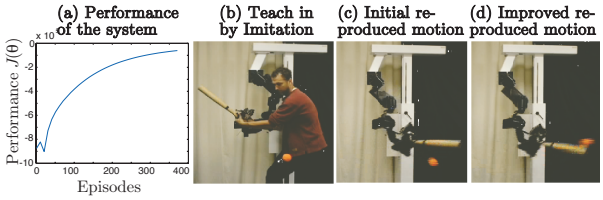

Fig. 3. This figure shows(a) the performance of a baseball swing task when using the motor primitives for learning. In (b), the learning system is initialized by imitation learning, in (c) it is initially failing at reproducing the motor behavior, and (d) after several hundred episodes exhibiting a nicely learned batting.

5 Conclusion

In this paper, we have summarized novel developments in policy-gradient reinforcement learning, and based on these, we have designed a novel reinforcement learning architecture, the Natural Actor-Critic algorithm. This algorithm comes in (at least) two forms, i.e., the LSTD-Q(λ) form which depends on sufficiently rich basis functions, and the Episodic form which only requires a constant as additional basis function. We compare both algorithms and apply the latter on several evaluative benchmarks as well as on a baseball swing robot example.

References

1. S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10:251–276, 1998.
2. J. Bagnell and J. Schneider. Covariant policy search. In *International Joint Conference on Artificial Intelligence*, 2003.
3. L.C. Baird. *Advantage Updating*. Wright Lab. Tech. Rep. WL-TR-93-1146, 1993.
4. L.C. Baird and A.W. Moore. Gradient descent for general reinforcement learning. In *Advances in Neural Information Processing Systems 11*, 1999.
5. P. Bartlett. An introduction to reinforcement learning theory: Value function methods. In *Machine Learning Summer School*, pages 184–202, 2002.
6. D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.
7. J. Boyan. Least-squares temporal difference learning. In *Machine Learning: Proceedings of the Sixteenth International Conference*, pages 49–56, 1999.
8. S. Bradtke, E. Ydstie, and A.G. Barto. *Adaptive Linear Quadratic Control Using Policy Iteration*. University of Massachusetts, Amherst, MA, 1994.
9. A. Ijspeert, J. Nakanishi, and S. Schaal. Learning rhythmic movements by demonstration using nonlinear oscillators. In *IEEE International Conference on Intelligent Robots and Systems (IROS 2002)*, pages 958–963, 2002.
10. S. A. Kakade. Natural policy gradient. In *Advances in Neural Information Processing Systems 14*, 2002.
11. V. Konda and J. Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems 12*, 2000.
12. T. Moon and W. Stirling. *Mathematical Methods and Algorithms for Signal Processing*. Prentice Hall, 2000.
13. J. Peters, S. Vijaykumar, and S. Schaal. Reinforcement learning for humanoid robotics. In *IEEE International Conference on Humandoid Robots*, 2003.
14. R.S. Sutton and A.G. Barto. *Reinforcement Learning*. MIT Press, 1998.
15. R.S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, 2000.

A Fisher Information Property

In Section 5, we explained that the all-action matrix $F_{\boldsymbol{\theta}}$ equals in general the Fisher information matrix $G(\boldsymbol{\theta})$. In [12], 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

$$\int_{\mathbb{R}^n} p(\boldsymbol{x})\nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{x})d\boldsymbol{x} = -\int_{\mathbb{R}^n} p(\boldsymbol{x})\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x})\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x})^T d\boldsymbol{x} \tag{15}$$

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 $p(\boldsymbol{\tau}_{0:n}) = p(\boldsymbol{x}_0) \prod_{t=0}^{n} p(\boldsymbol{x}_{t+1} | \boldsymbol{x}_t, \boldsymbol{u}_t) \pi(\boldsymbol{u}_t | \boldsymbol{x}_t)$ which implies that

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

Using Equations (15, A), and the definition of the Fisher information matrix [1], we can determine Fisher information matrix for the average reward case by

$$\begin{aligned} G(\boldsymbol{\theta}) &= \lim_{n\to\infty} n^{-1} E_{\boldsymbol{\tau}}\{\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}_{0:n})^T\} = -\lim_{n\to\infty} n^{-1} E_{\boldsymbol{\tau}}\left\{\nabla_{\boldsymbol{\theta}}^2 \log p(\boldsymbol{\tau})\right\}, \\ &= -\lim_{n\to\infty} n^{-1} E_{\boldsymbol{\tau}}\left\{\sum_{t=0}^{n} \nabla_{\boldsymbol{\theta}}^2 \log \pi(\boldsymbol{u}_t | \boldsymbol{x}_t)\right\} = -\int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \nabla_{\boldsymbol{\theta}}^2 \log \pi(\boldsymbol{u}|\boldsymbol{x}) \\ &d\boldsymbol{u}d\boldsymbol{x} = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x})^T d\boldsymbol{u}d\boldsymbol{x} = F_{\boldsymbol{\theta}} \end{aligned} \tag{16}$$

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 $p_{\gamma}(\boldsymbol{\tau}_{0:n}) = p(\boldsymbol{\tau}_{0:n})(\sum_{i=0}^{n} \gamma^i \mathbb{I}_{x_i,u_i})$, and derive that the all-action matrix equals the Fisher information matrix by the same kind of reasoning as in Eq.(16). Therefore, we can conclude that in general, i.e., $G(\boldsymbol{\theta}) = F_{\boldsymbol{\theta}}$.

B Proof of the Covariance Theorem

For small parameter changes $\Delta\boldsymbol{h}$ and $\Delta\boldsymbol{\theta}$, we have $\Delta\boldsymbol{\theta} = \nabla_{\boldsymbol{h}}\boldsymbol{\theta}^T \Delta\boldsymbol{h}$. If the natural policy gradient is a covariant update rule, a change $\Delta\boldsymbol{h}$ along the gradient $\nabla_{\boldsymbol{h}} J(\boldsymbol{h})$ would result in the same change $\Delta\boldsymbol{\theta}$ along the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ for the same scalar step-size α. By differentiation, we can obtain $\nabla_{\boldsymbol{h}} J(\boldsymbol{h}) = \nabla_{\boldsymbol{h}}\boldsymbol{\theta} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$.It is straightforward to show that the Fisher information matrix includes the Jacobian $\nabla_{\boldsymbol{h}}\boldsymbol{\theta}$ twice as factor,

$$\begin{aligned} \boldsymbol{F}(\boldsymbol{h}) &= \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \nabla_{\boldsymbol{h}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) \nabla_{\boldsymbol{h}} \log \pi(\boldsymbol{u}|\boldsymbol{x})^T d\boldsymbol{u}d\boldsymbol{x}, \\ &= \nabla_{\boldsymbol{h}}\boldsymbol{\theta} \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x})^T d\boldsymbol{u}d\boldsymbol{x} \nabla_{\boldsymbol{h}}\boldsymbol{\theta}^T, \\ &= \nabla_{\boldsymbol{h}}\boldsymbol{\theta} \boldsymbol{F}(\boldsymbol{\theta}) \nabla_{\boldsymbol{h}}\boldsymbol{\theta}^T. \end{aligned}$$

This shows that natural gradient in the $\boldsymbol{h}$ parameterization is given by

$$\widetilde{\nabla}_{\boldsymbol{h}} J(\boldsymbol{h}) = \boldsymbol{F}^{-1}(\boldsymbol{h}) \nabla_{\boldsymbol{h}} J(\boldsymbol{h}) = \left(\nabla_{\boldsymbol{h}}\boldsymbol{\theta} \boldsymbol{F}(\boldsymbol{\theta}) \nabla_{\boldsymbol{h}}\boldsymbol{\theta}^T\right)^{-1} \nabla_{\boldsymbol{h}}\boldsymbol{\theta} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}).$$

This has a surprising implication as it makes it straightforward to see that the natural policy is covariant since

$$\begin{aligned} \Delta\boldsymbol{\theta} &= \alpha \nabla_{\boldsymbol{h}}\boldsymbol{\theta}^T \Delta\boldsymbol{h} = \alpha \nabla_{\boldsymbol{h}}\boldsymbol{\theta}^T \widetilde{\nabla}_{\boldsymbol{h}} J(\boldsymbol{h}), = \alpha \nabla_{\boldsymbol{h}}\boldsymbol{\theta}^T \left(\nabla_{\boldsymbol{h}}\boldsymbol{\theta} \boldsymbol{F}(\boldsymbol{\theta}) \nabla_{\boldsymbol{h}}\boldsymbol{\theta}^T\right)^{-1} \nabla_{\boldsymbol{h}}\boldsymbol{\theta} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}), \\ &= \alpha \boldsymbol{F}^{-1}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \alpha \widetilde{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}), \end{aligned}$$

assuming that $\nabla_{\boldsymbol{h}}\boldsymbol{\theta}$ is invertible. This concludes that the natural policy gradient is in fact a **covariant** gradient update rule.