Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 71 (2008) 1180–1190
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Natural Actor-Critic

Jan Peters[a,b,*], Stefan Schaal[b,c]

[a]Max-Planck-Institute for Biological Cybernetics, Tuebingen, Germany
[b]University of Southern California, Los Angeles, CA 90089, USA
[c]ATR Computational Neuroscience Laboratories, Kyoto 619-0288, Japan

Available online 1 February 2008

Abstract

In this paper, we suggest a novel reinforcement learning architecture, the Natural Actor-Critic. The actor updates are achieved using 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.

Keywords: Policy-gradient methods; Compatible function approximation; Natural gradients; Actor-Critic methods; Reinforcement learning; Robot learning

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 [24]. 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 [8]. 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 [6,8].

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 [25,15]. However, even when applied to simple examples with rather few states, policy-gradient methods often turn out to be quite inefficient [14], 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 [3], 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 [14], and subsequently shown in preliminary work to be the true natural policy gradient [21,4]. In this paper, we take this line of reasoning one step

*Corresponding author at: Max-Planck-Institute for Biological Cybernetics, Department of Empirical Inference, Spemannstr. 38, 72076 Tuebingen, Germany.
E-mail address: jan.peters@tuebingen.mpg.de (J. Peters).

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2007.11.026

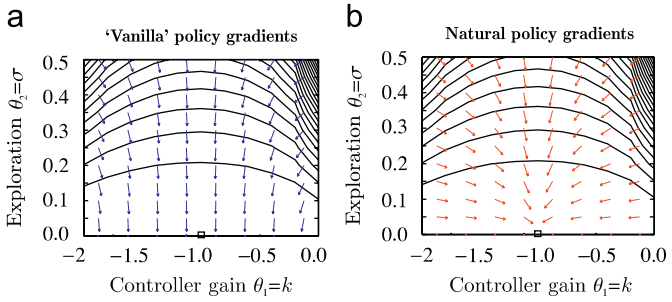

Fig. 1. When plotting the expected return landscape for simple problem as 1d linear-quadratic regulation, the differences between (a) 'vanilla' and (b) natural policy gradients becomes apparent [21].

further in Section 2.2 by introducing the 'Natural Actor-Critic (NAC)' 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 $\mathbb{U} = \mathbb{R}^m$ [8]. The assumption of an MDP comes with the limitation that very good state information and Markovian environment are assumed. However, similar as in [1], the results presented in this paper might extend to problems with partial state information.

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 $V^{\pi}(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{v}$. The general goal is to optimize the normalized expected return

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

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 improvement 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. [25,15]) which follow the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ of the expected return function $J(\boldsymbol{\theta})$ (where $\nabla_{\boldsymbol{\theta}} f = [\partial f/\partial\theta_1, \ldots, \partial f/\partial\theta_N]$) denotes the derivative of function f with respect to parameter vector $(\boldsymbol{\theta})$ often get stuck in plateaus as demonstrated in [14]. Natural gradients $\widetilde{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ avoid this pitfall as demonstrated for supervised learning problems [3], and suggested for reinforcement learning in [14]. These methods do not follow the steepest direction in parameter space but the steepest direction with respect to the Fisher metric given by

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

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 90°, i.e., convergence to the next local optimum can be assured. The 'vanilla' gradient is given by the policy-gradient theorem (see e.g. [25,15])

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

where $b^{\pi}(\boldsymbol{x})$ denotes a baseline. Refs. [25,15] 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

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

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

[25], 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) and (3), we derive an estimate of the policy gradient as

$$\nabla_\theta J(\theta) = \int_{\mathbb{X}} d^\pi(x) \int_{\mathbb{U}} \pi(u|x) \nabla_\theta \log \pi(u|x) \nabla_\theta \log \pi(u|x)^{\mathrm{T}} \, \mathrm{d}u \, \mathrm{d}x w = F_\theta w \tag{4}$$

as $\nabla_\theta \pi(u|x) = \pi(u|x) \nabla_\theta \log \pi(u|x)$. Since $\pi(u|x)$ is chosen by the user, even in sampled data, the integral

$$F(\theta, x) = \int_{\mathbb{U}} \pi(u|x) \nabla_\theta \log \pi(u|x) \nabla_\theta \log \pi(u|x)^{\mathrm{T}} \, \mathrm{d}u \tag{5}$$

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 $F_\theta = \int_{\mathbb{X}} d^\pi(x) F(\theta, x) \, \mathrm{d}x$ is still expensive since $d^\pi(x)$ is not known. However, Eq. (4) has more surprising implications for policy gradients, when examining the meaning of the matrix F_θ in Eq. (4). Kakade [14] argued that $F(\theta, x)$ is the point Fisher information matrix for state x, and that $F(\theta) = \int_{\mathbb{X}} d^\pi(x) \; F(\theta, x) \, \mathrm{d}x$, therefore, denotes a weighted 'average Fisher information matrix' [14]. However, going one step further, we demonstrate in Appendix A that F_θ 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

$$\widetilde{\nabla}_\theta J(\theta) = G^{-1}(\theta) F_\theta w = w, \tag{6}$$

since $F_\theta = G(\theta)$ (cf. Appendix A). Therefore we only need estimate w and *not* $G(\theta)$. The resulting policy improvement step is thus $\theta_{i+1} = \theta_i + \alpha w$ 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' [3].
- 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. Fig. 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 $\Delta\theta$ of the policy parameters. As we are interested in natural policy gradient updates $\Delta\theta = \alpha w$, we wish to employ the compatible function approximation $f_w^\pi(x, u)$ from Eq. (3) in this context. At this point, a most important observation is that the compatible function approximation $f_w^\pi(x, u)$ is mean-zero w.r.t. the action distribution, i.e.,

$$\int_{\mathbb{U}} \pi(u|x) f_w^\pi(x, u) \, \mathrm{d}u = w^{\mathrm{T}} \int_{\mathbb{U}} \nabla_\theta \pi(u|x) \, \mathrm{d}u = 0, \tag{7}$$

since from $\int_{\mathbb{U}} \pi(u|x) \, \mathrm{d}u = 1$, differention w.r.t. to θ results in $\int_{\mathbb{U}} \nabla_\theta \pi(u|x) \, \mathrm{d}u = \mathbf{0}$. Thus, $f_w^\pi(x, u)$ represents an *advantage function* $A^\pi(x, u) = Q^\pi(x, u) - V^\pi(x)$ in general. The essential differences between the advantage function and the state-action value function is demonstrated in Fig. 2. 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 $V^\pi(x)$ of the two adjacent states—but this value has been subtracted out in $A^\pi(x, u)$. Hence, a TD-like bootstrapping using exclusively the compatible function approximator is impossible.

As an alternative, [25,15] suggested to approximate $f_w^\pi(x, u)$ from unbiased estimates $\hat{Q}^\pi(x, u)$ of the action value function, e.g., obtained from rollouts 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.

Furthermore, in continuous state-spaces a state (except for single start-states) will hardly occur twice; therefore, we can only obtain unbiased estimates $\hat{Q}^\pi(x, u)$ of $Q^\pi(x, u)$. This means the state-action value estimates $\hat{Q}^\pi(x, u)$ have to

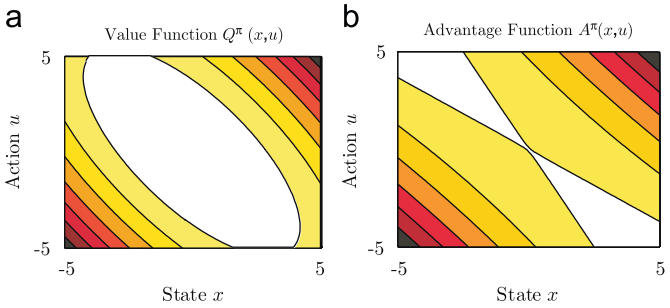

Fig. 2. The state-action value function in any stable linear-quadratic Gaussian regulation problems can be shown to be a bowl (a). The advantage function is always a saddle as shown in (b); it is straightforward to show that the compatible function approximation can exactly represent the advantage function—but projecting the value function onto the advantage function is non-trivial for continuous problems. This figure shows the value function and advantage function of the system described in the caption of Fig. 1.

be projected onto the advantage function $A^\pi(\boldsymbol{x},\boldsymbol{u})$. This projection would have to average out the state-value offset $V^\pi(\boldsymbol{x})$. For example, for linear-quadratic regulation, it is straightforward to show that the advantage function is saddle while the state-action value function is bowl—we therefore would be projecting a bowl onto a saddle; both are illustrated in Fig. 2. In this case, the distribution of the data has a drastic impact on the projection.

To remedy this situation, we observe that we can write the Bellman equations (e.g., see [5]) in terms of the advantage function and the state-value function

$$\begin{aligned} Q^\pi(\boldsymbol{x},\boldsymbol{u}) &= A^\pi(\boldsymbol{x},\boldsymbol{u}) + V^\pi(\boldsymbol{x}) \\ &= r(\boldsymbol{x},\boldsymbol{u}) + \gamma \int_{\mathbb{X}} p(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u}) V^\pi(\boldsymbol{x}')\,\mathrm{d}\boldsymbol{x}'. \end{aligned} \tag{8}$$

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 $V^\pi(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{v}$, we can rewrite the Bellman Equation, Eq. (8), as a set of linear equations

$$\begin{aligned} &\nabla_\theta \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^{\mathrm{T}}\boldsymbol{w} + \boldsymbol{\phi}(\boldsymbol{x}_t)^{\mathrm{T}}\boldsymbol{v} \\ &\quad = r(\boldsymbol{x}_t,\boldsymbol{u}_t) + \gamma\boldsymbol{\phi}(\boldsymbol{x}_{t+1})^{\mathrm{T}}\boldsymbol{v} + \varepsilon(\boldsymbol{x}_t,\boldsymbol{u}_t,\boldsymbol{x}_{t+1}), \end{aligned} \tag{9}$$

where $\varepsilon(\boldsymbol{x}_t,\boldsymbol{u}_t,\boldsymbol{x}_{t+1})$ denotes an error term which mean-zero as can be observed from Eq. (8). These equations enable us to formulate some novel algorithms in the next sections.

The linear appearance of $\boldsymbol{w}$ and $\boldsymbol{v}$ hints at a least squares to obtain. Thus, we now need to address algorithms that estimate the gradient efficiently using the sampled equations (such as Eq. (9)), and how to determine the additional basis functions $\boldsymbol{\phi}(\boldsymbol{x})$ for which convergence of these algorithms is guaranteed.

2.3.1. Critic evaluation with LSTD-Q(λ)

Using Eq. (9), a solution to Eq. (8) can be obtained by adapting the LSTD(λ) policy evaluation algorithm [9]. For this purpose, we define

$$\begin{aligned} \widehat{\boldsymbol{\phi}}_t &= [\boldsymbol{\phi}(\boldsymbol{x}_t)^{\mathrm{T}}, \nabla_\theta \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^{\mathrm{T}}]^{\mathrm{T}}, \\ \widetilde{\boldsymbol{\phi}}_t &= [\boldsymbol{\phi}(\boldsymbol{x}_{t+1})^{\mathrm{T}}, \boldsymbol{0}^{\mathrm{T}}]^{\mathrm{T}}, \end{aligned} \tag{10}$$

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 [9] 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 [10])—such input noise would violate the standard regression model that only takes noise in the regression targets into account. Alternatively, Bradtke et al. [10] 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. (10), 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.

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 $\nabla_\theta \log \pi(\boldsymbol{u}|\boldsymbol{x})$ 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: $\widetilde{\boldsymbol{\phi}}_t = [\boldsymbol{\phi}(\boldsymbol{x}_{t+1})^{\mathrm{T}}, \boldsymbol{0}^{\mathrm{T}}]^{\mathrm{T}}$, $\widehat{\boldsymbol{\phi}}_t = [\boldsymbol{\phi}(\boldsymbol{x}_t)^{\mathrm{T}}, \nabla_\theta \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^{\mathrm{T}}]^{\mathrm{T}}$,
4.2: statistics: $\boldsymbol{z}_{t+1} = \lambda\boldsymbol{z}_t + \widehat{\boldsymbol{\phi}}_t; \boldsymbol{A}_{t+1} = \boldsymbol{A}_t + \boldsymbol{z}_{t+1}(\boldsymbol{\phi}_t - \gamma\widetilde{\boldsymbol{\phi}}_t)^{\mathrm{T}}$; $\boldsymbol{b}_{t+1} = \boldsymbol{b}_t + \boldsymbol{z}_{t+1} r_t$,
4.3: critic parameters: $[\boldsymbol{v}^{\mathrm{T}}_{t+1}, \boldsymbol{w}^{\mathrm{T}}_{t+1}]^{\mathrm{T}} = \boldsymbol{A}^{-1}_{t+1}\boldsymbol{b}_{t+1}$.
5: **Actor:** If gradient estimate is accurate, $\measuredangle(\boldsymbol{w}_t, \boldsymbol{w}_{t-1}) \leqslant \varepsilon$, 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.**

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 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 [7] 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 [7], we at least could guarantee that we are not affected for $\lambda = 1$.

2.3.2. Episodic NAC

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 Eq. (9) along a sample path, we obtain

$$\begin{aligned} &\sum_{t=0}^{N-1} \gamma^t A^\pi(\boldsymbol{x}_t,\boldsymbol{u}_t) \\ &\quad = V^\pi(\boldsymbol{x}_0) + \sum_{t=0}^{N-1} \gamma^t r(\boldsymbol{x}_t,\boldsymbol{u}_t) - \gamma^N V^\pi(\boldsymbol{x}_N). \end{aligned} \tag{11}$$

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 $\boldsymbol{\theta} = \boldsymbol{\theta}_0$, and derivative $\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x})$.

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 $f^{\pi}_{\boldsymbol{w}}(\boldsymbol{x}_t, \boldsymbol{u}_t)$.
  Update: Determine basis functions: $\boldsymbol{\phi}_t = [\sum_{t=0}^{N} \gamma^t \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^{\mathrm{T}}, 1]^{\mathrm{T}}$; reward statistics: $R_t = \sum_{t=0}^{N} \gamma^t r$;
  Actor-Update: When the natural gradient is converged, $\measuredangle(\boldsymbol{w}_{t+1}, \boldsymbol{w}_{t-\tau}) \leqslant \varepsilon$, update the policy parameters: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \boldsymbol{w}_{t+1}$.
6: **end**.

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 rollout 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:

$$\sum_{t=0}^{N-1} \gamma^t \nabla \log \pi(\boldsymbol{u}_t, \boldsymbol{x}_t)^{\mathrm{T}} \boldsymbol{w} + J = \sum_{t=0}^{N-1} \gamma^t r(\boldsymbol{x}_t, \boldsymbol{u}_t) \tag{12}$$

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.

3. Properties of NAC

In this section, we will emphasize certain properties of the NAC. In particular, we want to give a simple proof of covariance of the natural policy gradient, and discuss [14] observation that in his experimental settings the natural policy gradient was non-covariant. Furthermore, we will discuss another surprising aspect about the NAC which is its relation to previous algorithms. We briefly demonstrate that established algorithms like the classic Actor-Critic [24], and Bradtke's Q-Learning [10] can be seen as special cases of NAC.

3.1. On the covariance of natural policy gradients

When [14] 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 [14] 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 $\nabla_{\boldsymbol{h}}\boldsymbol{\theta}$ and its inverse $\nabla_{\boldsymbol{h}}\boldsymbol{\theta}^{-1}$.*

For the proof see Appendix B. Practical experiments show that the problems occurred for Gaussian policies in [14] are in fact due to the selection the stepsize α which determines the length of $\Delta\boldsymbol{\theta}$. As the linearization $\Delta\boldsymbol{\theta} = \nabla_{\boldsymbol{h}}\boldsymbol{\theta}^{\mathrm{T}}\Delta\boldsymbol{h}$ does not hold for large $\Delta\boldsymbol{\theta}$, this can cause divergence between the algorithms even for analytically determined natural policy gradients which can partially explain the difficulties occurred by Kakade [14].

3.2. NAC's relation to previous algorithms

Original Actor-Critic. Surprisingly, the original Actor-Critic algorithm [24] is a form of the NAC. 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 $V^{\pi}(x) = \boldsymbol{\phi}(x)^{\mathrm{T}}\boldsymbol{v}$ 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

$$\theta^{t+1}_{xu} = \theta^{t}_{xu} + \alpha_1(r(x,u) + \gamma v_{x'} - v_x),$$
$$v^{t+1}_{x} = v^{t}_{x} + \alpha_2(r(x,u) + \gamma v_{x'} - v_x),$$

where α_1, α_2 denote learning rates. The update of the critic parameters v^t_x equals the one of the NAC in expectation as TD(0) critics converges to the same values as LSTD(0) and LSTD-Q(0) for discrete problems [9]. 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

$$\begin{aligned} A(x,u) &= A(x,u) - \sum_b \pi(b|x)A(x,b) \\ &= \sum_b \frac{\partial \log \pi(b|x)}{\partial \theta_{xu}} A(x,b). \end{aligned}$$

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., $\boldsymbol{w} = [A(x,u)]$. 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.

SARSA. SARSA with a tabular, discrete state-action value function $Q^\pi(x,u)$ and an ε-soft policy improvement

$$\pi(\boldsymbol{u}_t|\boldsymbol{x}_t) = \exp(Q^\pi(x,u)/\varepsilon)/\sum_{\hat{u}} \exp(Q^\pi(x,u)/\varepsilon)$$

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 $w_{xu} = \varepsilon A(x,u) \approx \varepsilon E_{x'}\{r(x,u) + \gamma Q(x',u') - Q(x,u)|x,u\}$. However, the SARSA-TD error equals the advantage function only for policies where a single action u^* has much better action values $Q(x,u^*)$ than all other actions; *for such special cases*, ε-soft SARSA can be seen as an approximation of NAC. This also corresponds to Kakade's [14] observation that greedy update step (such as the ε-soft greedy update), approximates the natural policy gradient.

Bradtke's Q-Learning. Bradtke [10] proposed an algorithm with policy $\pi(u_t|\boldsymbol{x}_t) = \mathcal{N}(u_t|\boldsymbol{k}_i^{\mathrm{T}}\boldsymbol{x}_t, \sigma_i^2)$ and parameters $\boldsymbol{\theta}_i = [\boldsymbol{k}_i^{\mathrm{T}}, \sigma_i]^{\mathrm{T}}$ (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 $r(\boldsymbol{x}_t,\boldsymbol{u}_t) = \boldsymbol{x}_t^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x}_t + Ru_t^2$. 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:

$$\begin{aligned}\boldsymbol{k}_{i+1}^{\text{Bradtke}} &= \arg\max_{\boldsymbol{k}_{i+1}} Q^\pi(\boldsymbol{x}_t,\boldsymbol{u}_t = \boldsymbol{k}_{i+1}^{\mathrm{T}}\boldsymbol{x}_t)\\ &= -(R + \gamma\boldsymbol{b}^{\mathrm{T}}\boldsymbol{P}_i\boldsymbol{b})^{-1}\gamma\boldsymbol{b}\boldsymbol{P}_i\boldsymbol{A},\end{aligned}$$

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 $\boldsymbol{w} = [\boldsymbol{w}_{\boldsymbol{k}}, w_\sigma]^{\mathrm{T}}$, as yielded by LSTD-Q($\lambda$) analytically using the compatible function approximation and the same quadratic basis functions. As discussed in detail in [21], this gives us

$$\boldsymbol{w}_{\boldsymbol{k}} = (\gamma\boldsymbol{A}^{\mathrm{T}}\boldsymbol{P}_i\boldsymbol{b} + (R + \gamma\boldsymbol{b}^{\mathrm{T}}\boldsymbol{P}_i\boldsymbol{b})\boldsymbol{k})^{\mathrm{T}}\sigma_i^2,$$

$$w_\sigma = 0.5(\boldsymbol{R} + \gamma\boldsymbol{b}^{\mathrm{T}}\boldsymbol{P}_i\boldsymbol{b})\sigma_i^3.$$

Similarly, it can be derived that the expected return is $J(\boldsymbol{\theta}_i) = -(R + \gamma\boldsymbol{b}^{\mathrm{T}}\boldsymbol{P}_i\boldsymbol{b})\sigma_i^2$ for this type of problems, see [21]. For a learning rate $\alpha_i = 1/\|J(\boldsymbol{\theta}_i)\|$, we see

$$\begin{aligned}\boldsymbol{k}_{i+1} &= \boldsymbol{k}_i + \alpha_t\boldsymbol{w}_{\boldsymbol{k}} = \boldsymbol{k}_i - (\boldsymbol{k}_i + (R + \gamma\boldsymbol{b}^{\mathrm{T}}\boldsymbol{P}_i\boldsymbol{b})^{-1}\gamma\boldsymbol{A}^{\mathrm{T}}\boldsymbol{P}_i\boldsymbol{b})\\ &= \boldsymbol{k}_{i+1}^{\text{Bradtke}},\end{aligned}$$

which demonstrates that *Bradtke's Actor Update is a special case of the NAC*. 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 NAC 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

Cart-Pole Balancing is a well-known benchmark for reinforcement learning. We assume the cart as shown in Fig. 3(a) can be described by

$$ml\ddot{x}\cos\theta + ml^2\ddot{\theta} - mgl\sin\theta = 0,$$

$$(m + m_{\mathrm{c}})\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F,$$

with $l = 0.75\,\mathrm{m}$, $m = 0.15\,\mathrm{kg}$, $g = 9.81\,\mathrm{m/s^2}$ and $m_{\mathrm{c}} = 1.0\,\mathrm{kg}$. The resulting state is given by $\boldsymbol{x} = [x, \dot{x}, \theta, \dot{\theta}]^{\mathrm{T}}$, and the action $\boldsymbol{u} = F$. The system is treated as if it was sampled at a rate of $h = 60\,\mathrm{Hz}$, and the reward is given by $r(\boldsymbol{x},\boldsymbol{u}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{u}$ with $\boldsymbol{Q} = \mathrm{diag}(1.25, 1, 12, 0.25)$, $\boldsymbol{R} = 0.01$.

The policy is specified as $\pi(\boldsymbol{u}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{K}\boldsymbol{x}, \sigma^2)$. 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 $\boldsymbol{\theta} = [\boldsymbol{K}^{\mathrm{T}}, \eta]^{\mathrm{T}}$ and has the analytically computable optimal solution $\boldsymbol{K} \approx [5.71, 11.3, -82.1, -21.6]^{\mathrm{T}}$, and $\sigma = 0.1$, corresponding to $\eta \to \infty$. As $\eta \to \infty$ is hard to visualize, we show σ in Fig. 3(b) despite the fact that the update takes place over the parameter η.

For each initial policy, samples $(\boldsymbol{x}_t, \boldsymbol{u}_t, r_{t+1}, \boldsymbol{x}_{t+1})$ are being generated using the start-state distributions, transition probabilities, the rewards and the policy. The samples arrive at a sampling rate of 60 Hz, and are immediately sent to the NAC module. The policy is updated when $\measuredangle(\boldsymbol{w}_{t+1}, \boldsymbol{w}_t) \leqslant \varepsilon = \pi/180$. At the time of update, the true 'vanilla' policy gradient, which can be computed analytically,[1] is used to update a separate policy. The true 'vanilla' policy gradients these serve as a baseline for the comparison. If the pole leaves the acceptable region of $-\pi/6 \leqslant \phi \leqslant \pi/6$, and $-1.5\,\mathrm{m} \leqslant x \leqslant +1.5\,\mathrm{m}$, it is reset to a new starting position drawn from the start-state distribution.

Results are illustrated in Fig. 3. In Fig. 3(b), a sample run is shown: the NAC algorithms estimates the optimal solution within less than 10 min of simulated robot trial time. The analytically obtained policy gradient for comparison takes over 2 h 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

[1] The true natural policy gradient can also be computed analytically. However, it is not shown as the difference in performance to the Natural Actor-Critic gradient estimate is negligible.

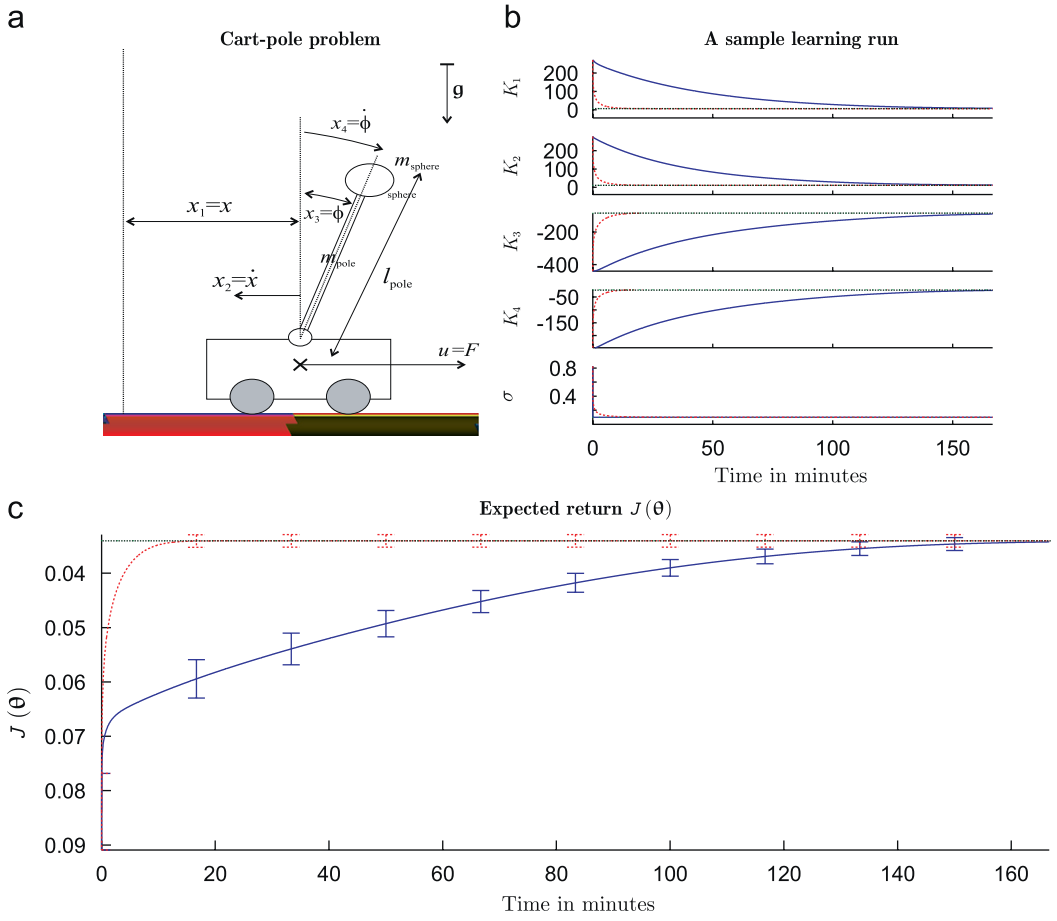

Fig. 3. This figure shows the performance of Natural Actor-Critic in the Cart-Pole Balancing framework. In (a), you can see the general setup of the pole mounted on the cart. In (b), a sample learning run of the both Natural Actor-Critic and the true policy gradient is given. The dashed line denotes the Natural Actor-Critic performance while the solid line shows the policy gradients performance. In (c), the expected return of the policy is shown. This is an average over 100 randomly picked policies as described in Section 4.1.

by the NAC algorithm. The performance difference between the true natural gradient and the NAC algorithm is negligible and, therefore, not shown separately. By the time of the conference, we hope to have this example implemented on a real anthropomorphic robot. In Fig. 3(c), the expected return over updates is shown averaged over all hundred initial policies.

In this experiment, we demonstrated that the NAC is comparable with the ideal natural gradient, and outperforms the 'vanilla' policy gradient 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. The only suitable greedy improvement method, to our knowledge, is Bradtke's Adaptive Policy Iteration [10]. However, this method is problematic in real-world application due to the fact that 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. Therefore, it can only work in simulated environments with an absence of noise in the state estimates and rewards.

4.2. Motor primitive learning for baseball

This section will turn towards optimizing nonlinear dynamic motor primitives for robotics. In [13], 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

$$\dot{q}_{d,k} = h(q_{d,k}, \boldsymbol{z}_k, g_k, \tau, \theta_k), \tag{13}$$

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 [13] 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. (13) by means of reinforcement learning, which is the crucial second step in imitation learning. The system in Eq. (13) is a point-to-point movement, i.e., this task is rather well suited for episodic NAC.

In Fig. 4, 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. In Fig. 4(a), we show how the expected cost decreases for both GPOMDP and the episodic NAC. The positions of the motor primitives are shown in Fig. 4(b) and in Fig. 4(c) the accelerations are given. In 4(b,c), the dashed line shows the initial configurations, which is accomplished by zero parameters for the motor primitives. The solid line shows the analytically optimal solution, which is unachievable for the motor primitives, but nicely approximated by their best solution, presented by the dark dot-dashed line. This best solution is reached by both learning methods. However, for GPOMDP, this requires approximately 10^6 learning steps while the NAC takes less than 10^3 to converge to the optimal solution.

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 Fig. 5(b); however, it fails to reproduce the behavior as shown in Fig. 5(c); subsequently, we improve the performance using the episodic NAC which yields the performance shown in Fig. 5(a) and the behavior in Fig. 5(d).

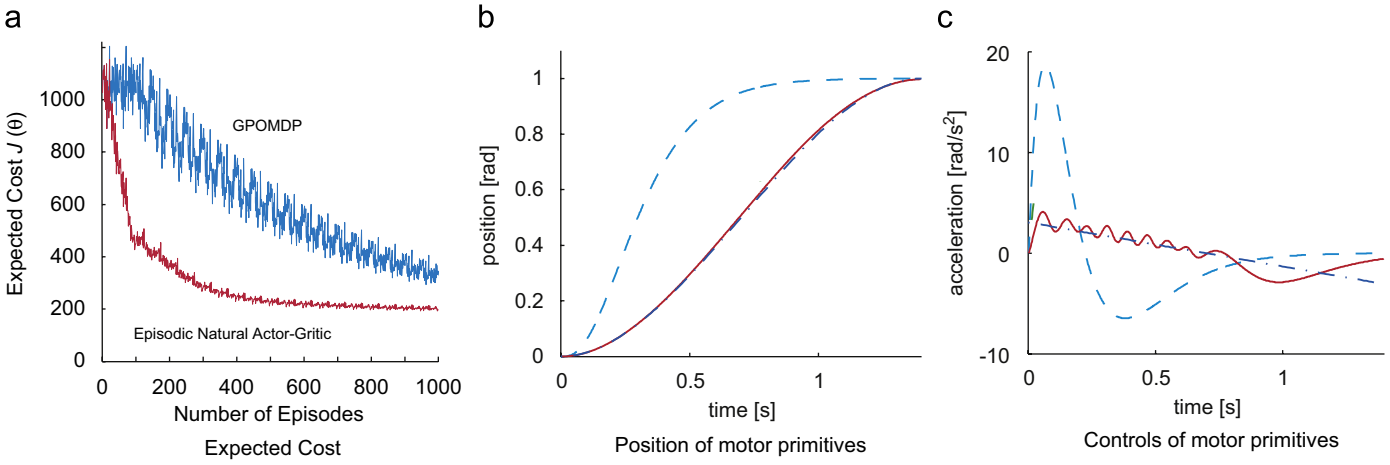

Fig. 4. This figure illustrates the task accomplished in the toy example. In (a), we show how the expected cost decreases for both GPOMDP and the episodic Natural Actor-Critic. The positions of the motor primitives are shown in (b) and in (c) the accelerations are given. In (b,c), the dashed line shows the initial configurations, which is accomplished by zero parameters for the motor primitives. The solid line shows the analytically optimal solution, which is unachievable for the motor primitives, but nicely approximated by their best solution, presented by the dark dot-dashed line. This best solution is reached by both learning methods. However, for GPOMDP, this requires approximately 10^6 learning steps while the NAC takes less than 10^3 to converge to the optimal solution.

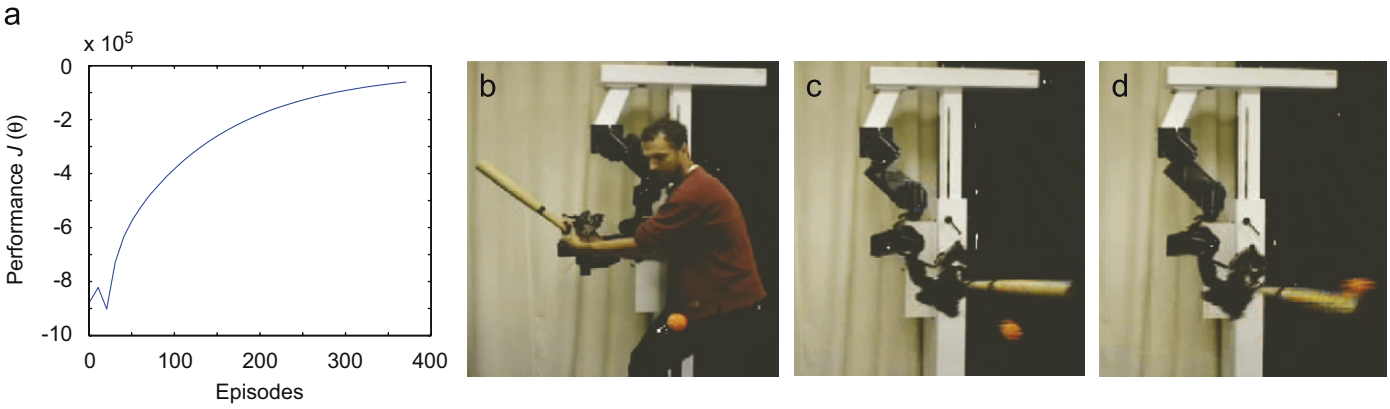

Fig. 5. The figure shows (a) the performance of a baseball swing task when using the motor primitives for learning. In (b), the learning system is initialized by imitation learning, in (c) it is initially failing at reproducing the motor behavior, and (d) after several hundred episodes exhibiting a nicely 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 NAC 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.

Recently, our NAC architecture [19,21] has gained a lot of traction in the reinforcement learning community. According to Aberdeen, the NAC is the 'Current method of choice' [2]. Additional to our work presented at ESANN 2007 in [19] and its earlier, preliminary versions (see e.g. [22,21,18,20]), the algorithm has found a variety of applications in largely unmodified form in the last year. The current range of additional applications includes optimization of constrained reaching movements of humanoid robots [12], traffic-light system optimization [23], multi-agent system optimization [11,28], conditional random fields [27] and gait optimization in robot locomotion [26,17]. All these new developments indicate that the NAC is about to become a standard architecture in the area of reinforcement learning as it is among the few approaches which have scaled towards interesting applications.

Appendix A. Fisher information property

In Section 6, we explained that the all-action matrix F_θ equals in general the Fisher information matrix $G(\boldsymbol{\theta})$. In [16], we can find the well-known lemma that by differentiating $\int_{\mathbb{R}^n} p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = 1$ twice with respect to the parameters $\boldsymbol{\theta}$, we can obtain

$$\int_{\mathbb{R}^n} p(\boldsymbol{x})\nabla_\theta^2 \log p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = -\int_{\mathbb{R}^n} p(\boldsymbol{x})\nabla_\theta \log p(\boldsymbol{x})\nabla_\theta \log p(\boldsymbol{x})^{\mathrm{T}}\,\mathrm{d}\boldsymbol{x} \tag{A.1}$$

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

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

Using Eq. (A.1), and the definition of the Fisher information matrix [3], we can determine Fisher information matrix for the average reward case by

$$\begin{aligned} G(\boldsymbol{\theta}) &= \lim_{n\to\infty} n^{-1} E_\tau\{\nabla_\theta \log p(\boldsymbol{\tau})\nabla_\theta \log p(\boldsymbol{\tau}_{0:n})^{\mathrm{T}}\} \\ &= -\lim_{n\to\infty} n^{-1} E_\tau\{\nabla_\theta^2 \log p(\boldsymbol{\tau})\}, \qquad \text{(A.2)} \\ &= -\lim_{n\to\infty} n^{-1} E_\tau\left\{\sum_{t=0}^{n} \nabla_\theta^2 \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)\right\} \\ &= -\int_{\mathbb{X}} d^\pi(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x})\nabla_\theta^2 \log \pi(\boldsymbol{u}|\boldsymbol{x})\,\mathrm{d}\boldsymbol{u}\,\mathrm{d}\boldsymbol{x} \qquad \text{(A.3)} \\ &= \int_{\mathbb{X}} d^\pi(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x})\nabla_\theta \log \pi(\boldsymbol{u}|\boldsymbol{x}) \\ &\quad \nabla_\theta \log \pi(\boldsymbol{u}|\boldsymbol{x})^{\mathrm{T}}\,\mathrm{d}\boldsymbol{u}\,\mathrm{d}\boldsymbol{x} = F_\theta. \qquad \text{(A.4)} \end{aligned}$$

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

$$p_\gamma(\boldsymbol{\tau}_{0:n}) = p(\boldsymbol{\tau}_{0:n})\left(\sum_{i=0}^{n} \gamma^i I_{x_i,u_i}\right)$$

and derive that the all-action matrix equals the Fisher information matrix by the same kind of reasoning as in Eq. (A.4). Therefore, we can conclude that in general, i.e., $G(\boldsymbol{\theta}) = F_\theta$.

Appendix B. Proof of the covariance theorem

For small parameter changes $\Delta\boldsymbol{h}$ and $\Delta\boldsymbol{\theta}$, we have $\Delta\boldsymbol{\theta} = \nabla_{\boldsymbol{h}}\boldsymbol{\theta}^{\mathrm{T}}\Delta\boldsymbol{h}$. 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 $\nabla_\theta J(\boldsymbol{\theta})$ for the same scalar step-size α. By differentiation, we can obtain $\nabla_{\boldsymbol{h}}J(\boldsymbol{h}) = \nabla_{\boldsymbol{h}}\boldsymbol{\theta}\nabla_\theta J(\boldsymbol{\theta})$. It is straightforward to show that the Fisher information matrix includes the Jacobian $\nabla_{\boldsymbol{h}}\boldsymbol{\theta}$ twice as factor

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

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

$$\begin{aligned} \widetilde{\nabla}_{\boldsymbol{h}}J(\boldsymbol{h}) &= \boldsymbol{F}^{-1}(\boldsymbol{h})\nabla_{\boldsymbol{h}}J(\boldsymbol{h}) \\ &= (\nabla_{\boldsymbol{h}}\boldsymbol{\theta}\boldsymbol{F}(\boldsymbol{\theta})\nabla_{\boldsymbol{h}}\boldsymbol{\theta}^{\mathrm{T}})^{-1}\nabla_{\boldsymbol{h}}\boldsymbol{\theta}\nabla_\theta J(\boldsymbol{\theta}). \end{aligned}$$

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

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

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

The assumptions underlying this proof require that the learning rate is very small in order to ensure a covariant gradient descent process. However, single update steps will always be covariant and, thus, this requirement is only formally necessary but barely matters in practice. Similar as in other gradient descent problems, learning rates can be chosen to optimize the performance without changing the fact that the covariance of a single update step direction will not be affected.

References

[1] D. Aberdeen, Policy-gradient algorithms for partially observable Markov decision processes, Ph.D. Thesis, Australian National Unversity, 2003.

[2] D. Aberdeen, POMDPs and policy gradients, in: Proceedings of the Machine Learning Summer School (MLSS), Canberra, Australia, 2006.

[3] S. Amari, Natural gradient works efficiently in learning, Neural Comput. 10 (1998) 251–276.

[4] J. Bagnell, J. Schneider, Covariant policy search, in: International Joint Conference on Artificial Intelligence, 2003.

[5] L.C. Baird, Advantage updating, Technical Report WL-TR-93-1146, Wright Lab., 1993.

[6] L.C. Baird, A.W. Moore, Gradient descent for general reinforcement learning, in: Advances in Neural Information Processing Systems, vol. 11, 1999.

[7] P. Bartlett, An introduction to reinforcement learning theory: value function methods, in: Machine Learning Summer School, 2002, pp. 184–202.

[8] D.P. Bertsekas, J.N. Tsitsiklis, Neuro-Dynamic Programming, Athena Scientific, Belmont, MA, 1996.

[9] J. Boyan, Least-squares temporal difference learning, in: Machine Learning: Proceedings of the Sixteenth International Conference, 1999, pp. 49–56.

[10] S. Bradtke, E. Ydstie, A.G. Barto, Adaptive Linear Quadratic Control Using Policy Iteration, University of Massachusetts, Amherst, MA, 1994.

[11] O. Buffet, A. Dutech, F. Charpillet, Shaping multi-agent systems with gradient reinforcement learning, Autonomous Agents Multi-Agent Syst. 15 (2) (October 2007) 1387–2532.

[12] F. Guenter, M. Hersch, S. Calinon, A. Billard, Reinforcement learning for imitating constrained reaching movements, RSJ Adv. Robotics 21 (13) (2007) 1521–1544.

[13] A. Ijspeert, J. Nakanishi, S. Schaal, Learning rhythmic movements by demonstration using nonlinear oscillators, in: IEEE International Conference on Intelligent Robots and Systems (IROS 2002), 2002, pp. 958–963.

[14] S.A. Kakade, Natural policy gradient, in: Advances in Neural Information Processing Systems, vol. 14, 2002.

[15] V. Konda, J. Tsitsiklis, Actor-critic algorithms, in: Advances in Neural Information Processing Systems, vol. 12, 2000.

[16] T. Moon, W. Stirling, Mathematical Methods and Algorithms for Signal Processing, Prentice-Hall, Englewood Cliffs, NJ, 2000.

[17] J. Park, J. Kim, D. Kang, An RLS-based Natural Actor-Critic algorithm for locomotion of a two-linked robot arm, in: Proceedings of Computational Intelligence and Security: International Conference (CIS 2005), Xi'an, China, December 2005, pp. 15–19.

[18] J. Peters, S. Schaal, Policy gradient methods for robotics, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Beijing, China, 2006.

[19] J. Peters, S. Schaal, Applying the episodic natural actor-critic architecture to motor primitive learning, in: Proceedings of the 2007 European Symposium on Artificial Neural Networks (ESANN), 2007.

[20] J. Peters, S. Vijayakumar, S. Schaal, Scaling reinforcement learning paradigms for motor learning, in: Proceedings of the 10th Joint Symposium on Neural Computation (JSNC), Irvine, CA, May 2003.

[21] J. Peters, S. Vijaykumar, S. Schaal, Reinforcement learning for humanoid robotics, in: IEEE International Conference on Humanoid Robots, 2003.

[22] J. Peters, S. Vijayakumar, S. Schaal, Natural Actor-Critic, in: Proceedings of the European Machine Learning Conference (ECML), Porto, Portugal, 2005.

[23] S. Richter, D. Aberdeen, J. Yu, Natural Actor-Critic for road traffic optimisation, in: Advances in Neural Information Processing Systems, 2007.

[24] R.S. Sutton, A.G. Barto, Reinforcement Learning, MIT Press, Cambridge, MA, 1998.

[25] R.S. Sutton, D. McAllester, S. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in: Advances in Neural Information Processing Systems, vol. 12, 2000.

[26] T. Ueno, Y. Nakamura, T. Shibata, K. Hosoda, S. Ishii, Fast and Stable learning of quasi-passive dynamic walking by an unstable biped robot based on off-policy Natural Actor-Critic, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2006.

[27] S.V. N Vishwanathan, X. Zhang, D. Aberdeen, Conditional random fields for reinforcement learning, in: Y. Bengio, Y. LeCun (Eds.), Proceedings of the 2007 Snowbird Learning Workshop, San Juan, Puerto Rico, March 2007.

[28] X. Zhang, D. Aberdeen, S.V.N. Vishwanathan, Conditional random fields for multi-agent reinforcement learning, in: Proceedings of the 24th International Conference on Machine Learning (ICML 2007), ACM International Conference Proceeding Series, Corvalis, Oregon, 2007, pp. 1143–1150.

Jan Peters heads the Robot Learning Lab (RoLL) at the Max-Planck Institute for Biological Cybernetics (MPI) while being an invited researcher at the Computational Learning and Motor Control Lab at the University of Southern California (USC). Before joining MPI, he graduated from University of Southern California with a Ph.D. in Computer Science in March 2007. Jan Peters studied Electrical Engineering, Computer Science and Mechanical Engineering. He holds two German M.Sc. degrees in Informatics and in Electrical Engineering (Dipl-Informatiker from Hagen University and Diplom-Ingenieur from Munich University of Technology/TUM) and two M.Sc. degrees in Computer Science and Mechanical Engineering from University of Southern California (USC). During his graduate studies, Jan Peters has been a visiting researcher at the Department of Robotics at the German Aerospace Research Center (DLR) in Oberpfaffenhofen, Germany, at Siemens Advanced Engineering (SAE) in Singapore, at the National University of Singapore (NUS), and at the Department of Humanoid Robotics and Computational Neuroscience at the Advanced Telecommunication Research (ATR) Center in Kyoto, Japan. His research interests include robotics, nonlinear control, machine learning, and motor skill learning.

Stefan Schaal is an Associate Professor at the Department of Computer Science and the Neuroscience Program at the University of Southern California, and an Invited Researcher at the ATR Human Information Sciences Laboratory in Japan, where he held an appointment as Head of the Computational Learning Group during an international ERATO project, the Kawato Dynamic Brain Project (ERATO/JST). Before joining USC, Dr. Schaal was a postdoctoral fellow at the Department of Brain and Cognitive Sciences and the Artificial Intelligence Laboratory at MIT, an Invited Researcher at the ATR Human Information Processing Research Laboratories in Japan, and an Adjunct Assistant Professor at the Georgia Institute of Technology and at the Department of Kinesiology of the Pennsylvania State University. Dr. Schaal's research interests include topics of statistical and machine learning, neural networks, computational neuroscience, functional brain imaging, nonlinear dynamics, nonlinear control theory, and biomimetic robotics. He applies his research to problems of artificial and biological motor control and motor learning, focusing on both theoretical investigations and experiments with human subjects and anthropomorphic robot equipment.