Applying the Episodic Natural Actor-Critic Architecture to Motor Primitive Learning

Jan Peters[1], Stefan Schaal[1]

University of Southern California, Los Angeles CA 90089, USA

Abstract. In this paper, we investigate motor primitive learning with the Natural Actor-Critic approach. The Natural Actor-Critic consists out of actor updates which are achieved using natural stochastic policy gradients while the critic obtains the natural policy gradient by linear regression. We show that this architecture can be used to learn the "building blocks of movement generation", called motor primitives. Motor primitives are parameterized control policies such as splines or nonlinear differential equations with desired attractor properties. We show that our most modern algorithm, the Episodic Natural Actor-Critic outperforms previous algorithms by at least an order of magnitude. We demonstrate the efficiency of this reinforcement learning method in the application of learning to hit a baseball with an anthropomorphic robot arm.

1 Introduction

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

However, despite being the most general framework for discussing the learning of motor primitives for robotics, most of the methods proposed in the reinforcement learning community are either not scalable or cannot deal with parameterized policies. Policy gradient methods are a notable exception to this statement. They 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 [1]. However, even when applied to simple examples with rather few states, policy gradient methods often turn out to be quite inefficient [2], 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.

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 [2], and subsequently shown in preliminary work to be the true natural policy gradient [4, 5], work which resulted into the *Natural Actor-Critic* which will be used in this paper for optimizing motor primitives with application in robotics.

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 $\mathbb{U} = \mathbb{R}^m$. The system is at an initial state $\mathbf{x}_0 \in \mathbb{X}$ at time $t = 0$ drawn from the start-state distribution $p(\mathbf{x}_0)$. At any state $\mathbf{x}_t \in \mathbb{X}$ at time t, the actor will choose an action $\mathbf{u}_t \in \mathbb{U}$ by drawing it from a stochastic, parameterized policy $\pi(\mathbf{u}_t|\mathbf{x}_t) = p(\mathbf{u}_t|\mathbf{x}_t, \theta)$ with parameters $\theta \in \mathbb{R}^N$, and the system transfers to a new state $\mathbf{x}_{t+1}$ drawn from the state transfer distribution $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$. The system yields a scalar reward $r_t = r(\mathbf{x}_t, \mathbf{u}_t) \in \mathbb{R}$ after each action. We assume that the policy π_θ is continuously differentiable with respect to its parameters θ. For each considered policy π_θ, a state-value function $V^\pi(\mathbf{x})$, the state-action value function $Q^\pi(\mathbf{x}, \mathbf{u})$ exist and are given by

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

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

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

where $d^\pi(\mathbf{x}) = (1-\gamma)\sum_{t=0}^\infty \gamma^t p(\mathbf{x}_t = \mathbf{x})$ 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., [1]) which follow the gradient $\nabla_\theta J(\theta)$ of the expected return function $J(\theta)$ (where $\nabla_\theta f = [\partial f/\partial\theta_1, \ldots, \partial f/\partial\theta_N]$) denotes the derivative of function f with respect to parameter vector θ) often get stuck in plateaus as demonstrated in [2]. Natural gradients $\widetilde{\nabla}_\theta J(\theta)$ avoid this pitfall as demonstrated for supervised learning problems [3], and suggested for reinforcement learning in [2]. These methods do not follow the steepest direction in parameter space but the steepest direction with respect to the Fisher metric given by $\widetilde{\nabla}_\theta J(\theta) = \mathbf{G}^{-1}(\theta)\nabla_\theta J(\theta)$, where $\mathbf{G}(\theta)$ 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., [1]),

$$\nabla_\theta J(\theta) = \int_{\mathbb{X}} d^\pi(\mathbf{x}) \int_{\mathbb{U}} \nabla_\theta \pi(\mathbf{u}|\mathbf{x}) \left(Q^\pi(\mathbf{x},\mathbf{u}) - b^\pi(\mathbf{x})\right) d\mathbf{u} d\mathbf{x}, \tag{1}$$

where $b^\pi(\mathbf{x})$ denotes a baseline. [1] showed that the term $Q^\pi(\mathbf{x},\mathbf{u}) - b^\pi(\mathbf{x})$ in Eq. (1) can be replaced by a compatible function approximation $f_w^\pi(\mathbf{x},\mathbf{u}) = (\nabla_\theta \log \pi(\mathbf{u}|\mathbf{x}))^T \mathbf{w} \equiv Q^\pi(\mathbf{x},\mathbf{u}) - b^\pi(\mathbf{x})$, parameterized by the vector $\mathbf{w}$, *without* affecting the unbiasedness of the gradient estimate and irrespective of the choice of the baseline $b^\pi(\mathbf{x})$. Thus, we have an estimate of the policy gradient as

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

as $\nabla_\theta \pi(\mathbf{u}|\mathbf{x}) = \pi(\mathbf{u}|\mathbf{x}) \nabla_\theta \log \pi(\mathbf{u}|\mathbf{x})$. Extending Kakade [2], we could show that F_θ is indeed the true Fisher information matrix [4,5]. Thus, the natural gradient can be computed as

$$\widetilde{\nabla}_\theta J(\theta) = G^{-1}(\theta) F_\theta \mathbf{w} = \mathbf{w}. \tag{2}$$

Therefore we only need estimate $\mathbf{w}$ and *not* $G(\theta)$. The resulting policy improvement step is thus $\theta_{i+1} = \theta_i + \alpha \mathbf{w}$ where α denotes a learning rate.

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 $\Delta\theta = \alpha \mathbf{w}$, we wish to employ the compatible function approximation $f_w^\pi(\mathbf{x},\mathbf{u})$ in this context. At this point, a most important observation is that the compatible function approximation $f_{\mathbf{w}}^\pi(\mathbf{x},\mathbf{u})$ is mean-zero w.r.t. the action distribution, i.e., $\int_{\mathbb{U}} \pi(\mathbf{u}|\mathbf{x}) f_{\mathbf{w}}^\pi(\mathbf{x},\mathbf{u}) d\mathbf{u} = \mathbf{w}^T \int_{\mathbb{U}} \nabla_\theta \pi(\mathbf{u}|\mathbf{x}) d\mathbf{u} = 0$, since from $\int_{\mathbb{U}} \pi(\mathbf{u}|\mathbf{x}) d\mathbf{u} = 1$, differention w.r.t. to θ results in $\int_{\mathbb{U}} \nabla_\theta \pi(\mathbf{u}|\mathbf{x}) d\mathbf{u} = \mathbf{0}$. Thus, $f_w^\pi(\mathbf{x},\mathbf{u})$ represents an *advantage function* $A^\pi(\mathbf{x},\mathbf{u}) = Q^\pi(\mathbf{x},\mathbf{u}) - V^\pi(\mathbf{x})$ in general. The advantage function is essentially different from the state-action value function and *cannot* be learned with TD-like bootstrapping without knowledge of the value function [4].

The critic's parameters can be determined by summing up Bellmans' equations such as $Q^\pi(\mathbf{x},\mathbf{u}) = A^\pi(\mathbf{x},\mathbf{u}) + V^\pi(\mathbf{x}) = r(\mathbf{x},\mathbf{u}) + \gamma \int_{\mathbb{X}} p(\mathbf{x}'|\mathbf{x},\mathbf{u}) V^\pi(\mathbf{x}') d\mathbf{x}'$ along a sample path and inserting $A^\pi(\mathbf{x},\mathbf{u}) = f_w^\pi(\mathbf{x},\mathbf{u})$, we obtain

$$\sum_{t=0}^{N-1} \gamma^t A^\pi(\mathbf{x}_t,\mathbf{u}_t) = \sum_{t=0}^{N-1} \gamma^t r(\mathbf{x}_t,\mathbf{u}_t) + \gamma^N V^\pi(\mathbf{x}_N) - V^\pi(\mathbf{x}_0) \tag{3}$$

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

$$\sum_{t=0}^{N-1} \gamma^t \nabla \log \pi(\mathbf{u}_t,\mathbf{x}_t)^T \mathbf{w} + J = \sum_{t=0}^{N-1} \gamma^t r(\mathbf{x}_t,\mathbf{u}_t) \tag{4}$$

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 1.

Table 1: Episodic Natural Actor-Critic Algorithm (eNAC)

Input: Parameterized policy $\pi(\mathbf{u}|\mathbf{x}) = p(\mathbf{u}|\mathbf{x}, \theta)$ with initial parameters $\theta = \theta_0$, its derivative $\nabla_\theta \log\pi(\mathbf{u}|\mathbf{x})$.

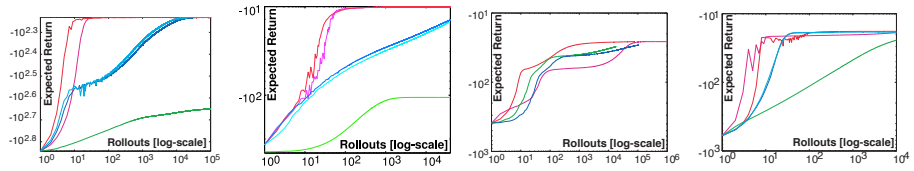
For update$k = 1, 2, 3, \ldots$ **do**
  For episode$e = 1, 2, 3, \ldots$ **do**
    Execute Rollout: Draw initial state $\mathbf{x}_0 \sim p(\mathbf{x}_0)$.
    For $t = 1, 2, 3, \ldots, N$ **do**
      Draw action $\mathbf{u}_t \sim \pi(\mathbf{u}_t|\mathbf{x}_t)$,
      observe next state $\mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$,
      and reward $r_t = r(\mathbf{x}_t, \mathbf{u}_t)$.
    end.
  end.
  Critic Evaluation (Episodic): Determine value function $J = V^\pi(\mathbf{x}_0)$, compatible function approximation $f^\pi_{\mathbf{w}}(\mathbf{x}_t, \mathbf{u}_t)$.
  Update: Determine
    basis functions: $\phi_k = \left[\sum_{t=0}^N \gamma^t \nabla_\theta \log \pi(\mathbf{u}_t|\mathbf{x}_t)^T, 1\right]^T$;
    reward statistics: $R_k = \sum_{t=0}^N \gamma^t r$;
  Actor-Update: When the natural gradient is converged, update the policy parameters: $\theta_{k+1} = \theta_k + \alpha \mathbf{w}_{k+1}$.
6: **end.**

3 Evaluations and Application to Motor Primitives

In this section, we compare the episodic Natural Actor-Critic (eNAC) on motor primitives with previous algorithms and evaluate it in motor learning.

3.1 Comparisons for Motor Primitive Policies

In this section, we first discuss how motor plans can be represented and then how we can bring these into the standard reinforcement learning framework. For this purpose, we consider two forms of motor plans, i.e., (1) *spline-based trajectory plans* and (2)*nonlinear dynamic motor primitives* introduced in [6]. Spline-based trajectory planning is well-known in the robotics literature [6]. A desired trajectory is represented by piecewise connected polynomials, i.e., we have $y_i(t) = \theta_{0i} + \theta_{1i}t + \theta_{2i}t^2 + \theta_{3i}t^3$ in $t \in [t_i, t_{i+1}]$ under the constraints that both $y_i(t_{i+1}) = y_{i+1}(t_{i+1})$ and $\dot{y}_i(t_{i+1}) = \dot{y}_{i+1}(t_{i+1})$. A given tracking controller, e.g., a PD control law or an inverse dynamics controller, ensures that the trajectory is tracked well. For nonlinear dynamic motor primitives, we use the approach developed in [6] where movement plans $(\mathbf{q}_d, \dot{\mathbf{q}}_d)$ for each degree of freedom (DOF) of the robot are represented in terms of the time evolution of the nonlinear dynamical systems $\ddot{q}_{d,k} = h(q_{d,k}, \mathbf{z}_k, g_k, \tau, \theta_k)$ 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



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

Fig. 1: This figure shows different experiments with motor task learning. In (a,b), we see how the learning system creates minimum motor command goal-achieving plans using both (a) splines and (b) motor primitives. For this problem, the natural actor-critic methods beat all other methods by several orders of magnitude. In (c,d), the plan has to achieve an intermediary goal.

duration shared by all DOFs, and θ_k the open parameters of the function h. The equations used in order to create motor primitives are given in [6].

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

Initially, we compare the different policy gradient methods in motor primitive planning tasks using both spline-based and dynamical system based desired trajectories. In Figure 1 (a) and (b), we show a comparison of the presented algorithms for a simple, single DOF task with a reward of $r_k(x_{0:N}, u_{0:N}) = \sum_{i=0}^{N} c_1 \dot{q}_{d,k,i}^2 + c_2 (q_{d;k;N} - g_k)^2$; where $c_1 = 1$, $c_2 = 1000$ for both splines and dynamic motor primitives. In Figure 1 (c) and (d) we show the same with an additional punishment term for going through a intermediate point p_F at time F, i.e., $r_k(x_{0:N}, u_{0:N}) = \sum_{i=0}^{N} \tilde{c}_1 \dot{q}_{d,k,i}^2 + \tilde{c}_2 (q_{d;k;N} - g_k)^2 + \tilde{c}_2 (q_{d;F;N} - p_F)^2$. It is quite clear from the results that the natural actor-critic methods outperform both the vanilla policy gradient methods as well as finite difference gradient methods. From this comparison, we can conclude that natural actor-critic methods are the best suited for motor primitive learning.

3.2 Robot Application: Motor Primitive Learning for Baseball

We also evaluated the same setup in a challenging robot task, i.e., the planning of these motor primitives for a seven DOF robot task using our SARCOS Master Arm. The task of the robot is to hit the ball properly so that it flies as far as possible; this game is also known as T-Ball. The state of the robot is given by its joint angles and velocities while the action are the joint accelerations. The reward is extracted using color segment tracking with a NewtonLabs vision system. Initially, we teach a rudimentary stroke by supervised learning as can

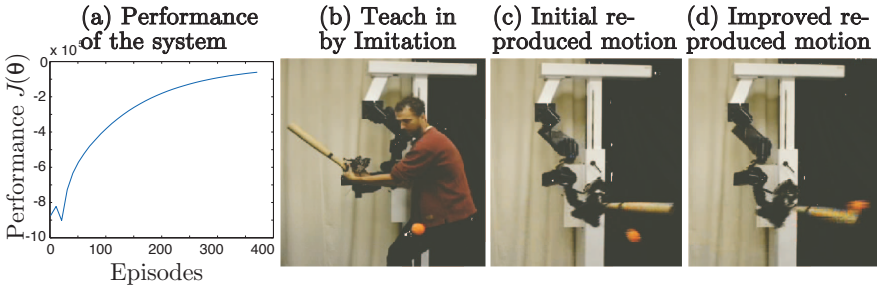


Fig. 2: This figure shows (a) the performance of a baseball swing task when using the motor primitives for learning. In (b), the learning system is initialized by imitation learning, in (c) it is initially failing at reproducing the motor behavior, and (d) after several hundred episodes exhibiting a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

be seen in Figure 2 (b); however, it fails to reproduce the behavior as shown in (c); subsequently, we improve the performance using the episodic Natural Actor-Critic which yields the performance shown in (a) and the behavior in (d). After approximately 200-300 trials, the ball can be hit properly by the robot.

4 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. We compare both algorithms and apply the latter on several evaluative benchmarks as well as on a baseball swing robot example.

References

[1] R.S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proc. NIPS 12*, 2000.

[2] S. A. Kakade. Natural policy gradient. In *Advances in Neural Information Processing Systems 14*, 2002.

[3] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10:251–276, 1998.

[4] J. Peters, S. Vijaykumar, and S. Schaal. Reinforcement learning for humanoid robotics. In *HUMANODS*, 2003.

[5] J. Bagnell and J. Schneider. Covariant policy search. In *Proc. IJCAI*, 2003.

[6] 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.