Mach Learn (2011) 84:171–203
DOI 10.1007/s10994-010-5223-6

Policy search for motor primitives in robotics

Jens Kober · Jan Peters

Received: 2 December 2009 / Revised: 27 September 2010 / Accepted: 17 October 2010 /
Published online: 6 November 2010

Abstract Many motor skills in humanoid robotics can be learned using parametrized motor primitives. While successful applications to date have been achieved with imitation learning, most of the interesting motor learning problems are high-dimensional reinforcement learning problems. These problems are often beyond the reach of current reinforcement learning methods. In this paper, we study parametrized policy search methods and apply these to benchmark problems of motor primitive learning in robotics. We show that many well-known parametrized policy search methods can be derived from a general, common framework. This framework yields both policy gradient methods and expectation-maximization (EM) inspired algorithms. We introduce a novel EM-inspired algorithm for policy learning that is particularly well-suited for dynamical system motor primitives. We compare this algorithm, both in simulation and on a real robot, to several well-known parametrized policy search methods such as episodic REINFORCE, 'Vanilla' Policy Gradients with optimal baselines, episodic Natural Actor Critic, and episodic Reward-Weighted Regression. We show that the proposed method out-performs them on an empirical benchmark of learning dynamical system motor primitives both in simulation and on a real robot. We apply it in the context of motor learning and show that it can learn a complex Ball-in-a-Cup task on a real Barrett WAM™ robot arm.

Keywords Motor primitives · Episodic reinforcement learning · Motor control · Policy learning

Editors: S. Whiteson and M. Littman.

J. Kober (✉) · J. Peters
Dept. Empirical Inference, Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tübingen, Germany
e-mail: jens.kober@tuebingen.mpg.de

J. Peters
e-mail: jan.peters@tuebingen.mpg.de

1 Introduction

To date, most robots are still taught by a skilled human operator either via direct programming or a teach-in. Learning approaches for automatic task acquisition and refinement would be a key step for making robots progress towards autonomous behavior. Although imitation learning can make this task more straightforward, it will always be limited by the observed demonstrations. For many motor learning tasks, skill transfer by imitation learning is prohibitively hard given that the human teacher is not capable of conveying sufficient task knowledge in the demonstration. In such cases, reinforcement learning is often an alternative to a teacher's presentation, or a means of improving upon it. In the high-dimensional domain of anthropomorphic robotics with its continuous states and actions, reinforcement learning suffers particularly from the curse of dimensionality. However, by using a task-appropriate policy representation and encoding prior knowledge into the system by imitation learning, local reinforcement learning approaches are capable of dealing with the problems of this domain. Policy search (also known as policy learning) is particularly well-suited in this context, as it allows the usage of domain-appropriate pre-structured policies (Toussaint and Goerick 2007), the straightforward integration of a teacher's presentation (Guenter et al. 2007; Peters and Schaal 2006) as well as fast online learning (Bagnell et al. 2004; Ng and Jordan 2000; Hoffman et al. 2007). Recently, policy search has become an accepted alternative of value-function-based reinforcement learning (Bagnell et al. 2004; Strens and Moore 2001; Kwee et al. 2001; Peshkin 2001; El-Fakdi et al. 2006; Taylor et al. 2007) due to many of these advantages.

In this paper, we will introduce a policy search framework for episodic reinforcement learning and show how it relates to policy gradient methods (Williams 1992; Sutton et al. 2000; Lawrence et al. 2003; Tedrake et al. 2004; Peters and Schaal 2006) as well as expectation-maximization (EM) inspired algorithms (Dayan and Hinton 1997; Peters and Schaal 2007). This framework allows us to re-derive or to generalize well-known approaches such as episodic REINFORCE (Williams 1992), the policy gradient theorem (Sutton et al. 2000; Peters and Schaal 2006), the episodic Natural Actor Critic (Peters et al. 2003, 2005), and an episodic generalization of the Reward-Weighted Regression (Peters and Schaal 2007). We derive a new algorithm called Policy Learning by Weighting Exploration with the Returns (PoWER), which is particularly well-suited for the learning of trial-based tasks in motor control.

We evaluate the algorithms derived from this framework to determine how they can be used for refining parametrized policies in robot skill learning. To address this problem, we follow a methodology suitable for robotics where the policy is first initialized by imitation learning and, subsequently, the policy search algorithm is used for self-improvement. As a result, we need a suitable representation in order to apply this approach in anthropomorphic robot systems. In imitation learning, a particular kind of motor control policy has been very successful, which is known as dynamical system motor primitives (Ijspeert et al. 2002, 2003; Schaal et al. 2003, 2007). In this approach, dynamical systems are used to encode a control policy suitable for motor tasks. The representation is linear in the parameters; hence, it can be learned straightforwardly from demonstrations. Such dynamical system motor primitives can represent both point-to-point and rhythmic behaviors. We focus on the point-to-point variant which is suitable for representing single-stroke, episodic behaviors. As a result, they are particularly well-suited for episodic policy search.

We show that all presented algorithms work sufficiently well when employed in the context of learning dynamical system motor primitives in different benchmark and application settings. We compare these methods on the two benchmark problems from (Peters and Schaal 2006) for dynamical system motor primitives learning, the Underactuated Swing-Up (Atkeson 1994) robotic benchmark problem, and the Casting task. Using entirely different parametrizations, we evaluate policy search methods on the mountain-car benchmark (Sutton and Barto 1998) and the Tetherball Target Hitting task. On the mountain-car benchmark, we additionally compare to a value function based approach. The method with the best performance, PoWER, is evaluated on the complex task of Ball-in-a-Cup (Sumners 1997). Both the Underactuated Swing-Up as well as Ball-in-a-Cup are achieved on a real Barrett WAM™ robot arm. Please also refer to the videos at http://www.robot-learning.de/Research/ReinforcementLearning. For all real robot experiments, the presented movement is learned by imitation from a kinesthetic demonstration, and the Barrett WAM™ robot arm subsequently improves its behavior by reinforcement learning.

A preliminary version of some of the work in this paper was shown in (Kober et al. 2008; Kober and Peters 2009a, 2009b). It did not yet include the real robot comparisons of the algorithms on the Underactuated Swing-Up benchmark, nor the Tetherball Target Hitting and the Casting benchmarks. We also discuss our experience with transfer from simulation to real robot systems.

2 Policy search for parametrized motor primitives

Our goal is to find reinforcement learning techniques that can be applied in robotics in the context of learning high-dimensional motor control tasks. We first introduce the required notation for the derivation of the reinforcement learning framework in Sect. 2.1. We discuss the problem in the general setting of reinforcement learning using a generalization of the approach in (Dayan and Hinton 1997; Attias 2003; Peters and Schaal 2007). We extend the existing approach to episodic reinforcement learning for continuous states, in a manner suitable for robotics.

We derive a new expectation-maximization (EM) inspired algorithm (Dempster et al. 1977) called Policy Learning by Weighting Exploration with the Returns (PoWER) in Sect. 2.3 and show how the general framework is related to policy gradient methods and the Reward-Weighted Regression method in Sect. 2.2.

2.1 Problem statement & notation

In this paper, we treat motor primitive learning problems in the framework of reinforcement learning (Sutton and Barto 1998) with a strong focus on the episodic case. At time t, there is an actor in a state $\mathbf{s}_t$ that chooses an action $\mathbf{a}_t$ according to a stochastic policy $\pi(\mathbf{a}_t|\mathbf{s}_t, t)$. Such a policy is a probability distribution over actions given the current state and time. The stochastic formulation allows a natural incorporation of exploration, and the optimal time-invariant policy has been shown to be stochastic in the case of hidden state variables (Sutton et al. 2000; Jaakkola et al. 1994). Upon the completion of the action, the actor transfers to a state $\mathbf{s}_{t+1}$ and receives a reward r_t. As we are interested in learning complex motor tasks consisting of a single stroke (Schaal et al. 2007), we focus on finite horizons of length T with episodic restarts and learn the optimal parametrized, stochastic policy for such episodic reinforcement learning problems (Sutton and Barto 1998). We assume an explorative parametrized policy π with parameters $\boldsymbol{\theta} \in \mathbb{R}^n$. In Sect. 3.1, we discuss how the dynamical system motor primitives (Ijspeert et al. 2002, 2003; Schaal et al. 2003, 2007) can be employed in this setting. In this section, we will keep most derivations sufficiently general such that they are transferable to various other parametrized policies that are linear in the parameters.

The general goal in reinforcement learning is to optimize the *expected return* of the policy π with parameters $\boldsymbol{\theta}$ defined by

$$J(\boldsymbol{\theta}) = \int_{\mathbb{T}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) R(\boldsymbol{\tau}) d\boldsymbol{\tau},$$

where $\mathbb{T}$ is the set of all possible paths. A rollout $\boldsymbol{\tau} = [\mathbf{s}_{1:T+1}, \mathbf{a}_{1:T}]$, also called path, episode or trial, denotes a series of states $\mathbf{s}_{1:T+1} = [\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_{T+1}]$ and actions $\mathbf{a}_{1:T} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_T]$. The probability of rollout $\boldsymbol{\tau}$ is denoted by $p_{\boldsymbol{\theta}}(\boldsymbol{\tau})$, while $R(\boldsymbol{\tau})$ refers to its aggregated return. Using the standard Markov assumption and additive accumulated rewards, we can write

$$\begin{aligned} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) &= p(\mathbf{s}_1) \prod_{t=1}^{T} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \pi(\mathbf{a}_t|\mathbf{s}_t, t), \\ R(\boldsymbol{\tau}) &= T^{-1} \sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, t), \end{aligned} \tag{1}$$

where $p(\mathbf{s}_1)$ denotes the initial state distribution, $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ the next state distribution conditioned on the last state and action, and $r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, t)$ denotes the immediate reward.

While episodic Reinforcement Learning (RL) problems with finite horizons are common in both human Wulf (2007) and robot motor control problems, few methods exist in the RL literature. Examples are episodic REINFORCE (Williams 1992), the episodic Natural Actor Critic eNAC (Peters et al. 2003, 2005) and model-based methods using differential dynamic programming (Atkeson 1994).

2.2 Episodic policy learning

In this section, we discuss episodic reinforcement learning in policy space, which we will refer to as Episodic Policy Learning. We first discuss the lower bound on the expected return as suggested in (Dayan and Hinton 1997) for guaranteeing that policy update steps are improvements. In (Dayan and Hinton 1997; Peters and Schaal 2007) only the immediate reward case is discussed; we extend this framework to episodic reinforcement learning. Subsequently, we derive a general update rule, which yields the policy gradient theorem (Sutton et al. 2000), a generalization of the reward-weighted regression (Peters and Schaal 2007), as well as the novel Policy learning by Weighting Exploration with the Returns (PoWER) algorithm.

2.2.1 Bounds on policy improvements

Unlike in reinforcement learning, other branches of machine learning have focused on maximizing lower bounds on the cost functions, which often results in expectation-maximization (EM) algorithms (McLachan and Krishnan 1997). The reasons for this preference also apply in policy learning: if the lower bound also becomes an equality for the sampling policy, we can guarantee that the policy will be improved by maximizing the lower bound. Results from supervised learning can be transferred with ease. First, we generalize the scenario suggested by Dayan and Hinton (1997) to the episodic case. Here, we generate rollouts $\boldsymbol{\tau}$ using the current policy with parameters $\boldsymbol{\theta}$, which we then weight with the returns $R(\boldsymbol{\tau})$, and subsequently match it with a new policy parametrized by $\boldsymbol{\theta}'$. This matching of the success-weighted path distribution is equivalent to minimizing the Kullback-Leibler divergence $D(p_{\boldsymbol{\theta}}(\boldsymbol{\tau})R(\boldsymbol{\tau})\|p_{\boldsymbol{\theta}'}(\boldsymbol{\tau}))$ between the new path distribution $p_{\boldsymbol{\theta}'}(\boldsymbol{\tau})$ and the reward-weighted previous one $p_{\boldsymbol{\theta}}(\boldsymbol{\tau})R(\boldsymbol{\tau})$. The Kullback-Leibler divergence is considered a natural distance measure between probability distributions (Bagnell and Schneider 2003; Van Der Maaten et al. 2007). As shown in (Dayan and Hinton 1997; Peters and Schaal 2007), such a derivation results in a lower bound on the expected return using Jensen's inequality and the concavity of the logarithm. Thus, we obtain

$$\log J(\boldsymbol{\theta}') = \log \int_{\mathbb{T}} p_{\boldsymbol{\theta}'}(\boldsymbol{\tau})\, R(\boldsymbol{\tau})\, d\boldsymbol{\tau} = \log \int_{\mathbb{T}} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})}{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})} p_{\boldsymbol{\theta}'}(\boldsymbol{\tau})\, R(\boldsymbol{\tau})\, d\boldsymbol{\tau},$$
$$\geq \int_{\mathbb{T}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau})\, R(\boldsymbol{\tau}) \log \frac{p_{\boldsymbol{\theta}'}(\boldsymbol{\tau})}{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})} d\boldsymbol{\tau} + \text{const},$$

which is proportional to

$$-D\left(p_{\boldsymbol{\theta}}(\boldsymbol{\tau})\, R(\boldsymbol{\tau})\, \| p_{\boldsymbol{\theta}'}(\boldsymbol{\tau})\right) = L_{\boldsymbol{\theta}}(\boldsymbol{\theta}'),$$

where

$$D\left(p(\boldsymbol{\tau})\, \| q(\boldsymbol{\tau})\right) = \int p(\boldsymbol{\tau}) \log \frac{p(\boldsymbol{\tau})}{q(\boldsymbol{\tau})} d\boldsymbol{\tau}$$

denotes the Kullback-Leibler divergence, and the constant is needed for tightness of the bound. Note that $p_{\boldsymbol{\theta}}(\boldsymbol{\tau})R(\boldsymbol{\tau})$ is an improper probability distribution as pointed out by Dayan and Hinton (1997). The policy improvement step is equivalent to maximizing the lower bound on the expected return $L_{\boldsymbol{\theta}}(\boldsymbol{\theta}')$, and we will now show how it relates to previous policy learning methods.

2.2.2 *Resulting policy updates*

In this section, we will discuss three different policy updates, which are directly derived from the results of Sect. 2.2.1. First, we show that policy gradients (Williams 1992; Sutton et al. 2000; Lawrence et al. 2003; Tedrake et al. 2004; Peters and Schaal 2006) can be derived from the lower bound $L_{\boldsymbol{\theta}}(\boldsymbol{\theta}')$, which is straightforward from a supervised learning perspective (Binder et al. 1997). Subsequently, we show that natural policy gradients (Bagnell and Schneider 2003; Peters and Schaal 2006) can be seen as an additional constraint regularizing the change in the path distribution resulting from a policy update when improving the policy incrementally. Finally, we will show how expectation-maximization (EM) algorithms for policy learning can be generated.

Policy gradients When differentiating the function $L_{\boldsymbol{\theta}}(\boldsymbol{\theta}')$ that defines the lower bound on the expected return, we directly obtain

$$\partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = \int_{\mathbb{T}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) R(\boldsymbol{\tau}) \partial_{\boldsymbol{\theta}'} \log p_{\boldsymbol{\theta}'}(\boldsymbol{\tau}) d\boldsymbol{\tau} = E\left\{\left(\sum_{t=1}^{T} \partial_{\boldsymbol{\theta}'} \log \pi(\mathbf{a}_t|\mathbf{s}_t, t)\right) R(\boldsymbol{\tau})\right\}, \quad (2)$$

where

$$\partial_{\boldsymbol{\theta}'} \log p_{\boldsymbol{\theta}'}(\boldsymbol{\tau}) = \sum_{t=1}^{T} \partial_{\boldsymbol{\theta}'} \log \pi(\mathbf{a}_t|\mathbf{s}_t, t)$$

denotes the log-derivative of the path distribution. As this log-derivative depends only on the policy we can estimate a gradient from rollouts, without having a model, by simply replacing the expectation by a sum. When $\boldsymbol{\theta}'$ is close to $\boldsymbol{\theta}$, we have the policy gradient estimator, which is widely known as episodic REINFORCE (Williams 1992)

$$\lim_{\boldsymbol{\theta}'\to\boldsymbol{\theta}} \partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta}).$$

See Algorithm 1 for an example implementation of this algorithm and Appendix A.1 for the detailed steps of the derivation. A MATLAB™ implementation of this algorithm is available at http://www.robot-learning.de/Member/JensKober.

A reward, which precedes an action in a rollout, can neither be caused by the action nor cause an action in the same rollout. Thus, when inserting (1) into (2), all cross-products between r_t and $\partial_{\boldsymbol{\theta}'} \log \pi(\mathbf{a}_{t+\delta t}|\mathbf{s}_{t+\delta t}, t+\delta t)$ for $\delta t > 0$ become zero in expectation (Peters and Schaal 2006). Therefore, we can omit these terms and rewrite the estimator as

$$\partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}\left(\boldsymbol{\theta}'\right) = E\left\{\sum_{t=1}^{T} \partial_{\boldsymbol{\theta}'} \log \pi\left(\mathbf{a}_t|\mathbf{s}_t, t\right) Q^{\pi}(\mathbf{s}, \mathbf{a}, t)\right\}, \tag{3}$$

Algorithm 1 'Vanilla' Policy Gradients (VPG)

```
Input: initial policy parameters $\boldsymbol{\theta}_0$
repeat
    Sample: Perform $h = \{1, \ldots, H\}$ rollouts using $\mathbf{a} = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{s}, t) + \boldsymbol{\varepsilon}_t$ with $[\varepsilon_t^n] \sim \mathcal{N}(0, (\sigma^{h,n})^2)$ as stochastic policy and collect all $(t, \mathbf{s}_t^h, \mathbf{a}_t^h, \mathbf{s}_{t+1}^h, \boldsymbol{\varepsilon}_t^h, r_{t+1}^h)$ for $t = \{1, 2, \ldots, T+1\}$.

    Compute: Return $R^h = \sum_{t=1}^{T+1} r_t^h$, eligibility

        $\psi^{h,n} = \frac{\partial \log p(\boldsymbol{\tau}^h)}{\partial \theta^n} = \sum_{t=1}^{T} \frac{\partial \log \pi(a_t^h|s_t^h, t)}{\partial \theta^n} = \sum_{t=1}^{T} \frac{\varepsilon_t^{h,n}}{(\sigma_h^n)^2} \phi^n\left(s_t^{h,n}, t\right)$

    and baseline

        $b^n = \frac{\sum_{h=1}^{H} (\psi^{h,n})^2 R^h}{\sum_{h=1}^{H} (\psi^{h,n})^2}$

    for each parameter $n = \{1, \ldots, N\}$ from rollouts.

    Compute Gradient:

        $g_{\mathrm{VP}}^n = E\left\{\frac{\partial \log p(\boldsymbol{\tau}^h)}{\partial \theta^n}\left(R(\boldsymbol{\tau}^h) - b^n\right)\right\} = \frac{1}{H}\sum_{h=1}^{H} \psi^{h,n}(R^h - b^n).$

    Update policy using

        $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \mathbf{g}_{\mathrm{VP}}.$

until Convergence $\boldsymbol{\theta}_{k+1} \approx \boldsymbol{\theta}_k$.
```

where

$$Q^{\pi}\left(\mathbf{s},\mathbf{a},t\right) = E\left\{\sum_{\tilde{t}=t}^{T} r\left(\mathbf{s}_{\tilde{t}}, \mathbf{a}_{\tilde{t}}, \mathbf{s}_{\tilde{t}+1}, \tilde{t}\right) | \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a}\right\}$$

is called the state-action value function (Sutton and Barto 1998). Equation (3) is equivalent to the policy gradient theorem (Sutton et al. 2000) for $\boldsymbol{\theta}' \to \boldsymbol{\theta}$ in the infinite horizon case, where the dependence on time t can be dropped.

The derivation results in the episodic Natural Actor Critic as discussed in (Peters et al. 2003, 2005) when adding an additional cost in (2) to penalize large steps away from the observed path distribution. Such a regularization can be achieved by restricting the amount of change in the path distribution and subsequently, determining the steepest descent for a fixed step away from the observed trajectories. Change in probability distributions is naturally measured using the Kullback-Leibler divergence, thus after adding the additional constraint of

$$D\left(p_{\boldsymbol{\theta}}\left(\boldsymbol{\tau}\right) \| p_{\boldsymbol{\theta}'}\left(\boldsymbol{\tau}\right)\right) \approx 0.5 \left(\boldsymbol{\theta}' - \boldsymbol{\theta}\right)^{\mathrm{T}} \mathbf{F}\left(\boldsymbol{\theta}\right)\left(\boldsymbol{\theta}' - \boldsymbol{\theta}\right) = \delta$$

using a second-order expansion as an approximation where $\mathbf{F}(\boldsymbol{\theta})$ denotes the Fisher information matrix (Bagnell and Schneider 2003; Peters et al. 2003, 2005). See Algorithm 2 for an example implementation of the episodic Natural Actor Critic. A MATLAB™ implementation of this algorithm is available at http://www.robot-learning.de/Member/JensKober.

Policy search via expectation maximization One major drawback of gradient-based approaches is the learning rate, which is an open parameter that can be hard to tune in control problems but is essential for good performance. Expectation-Maximization algorithms are

Algorithm 2 episodic Natural Actor Critic (eNAC)

```
Input: initial policy parameters $\boldsymbol{\theta}_0$
repeat
    Sample: Perform $h = \{1, \dots, H\}$ rollouts using $\mathbf{a} = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{s}, t) + \boldsymbol{\varepsilon}_t$ with $[\varepsilon_t^n] \sim \mathcal{N}(0, (\sigma^{h,n})^2)$ as stochastic policy and collect all $(t, \mathbf{s}_t^h, \mathbf{a}_t^h, \mathbf{s}_{t+1}^h, \boldsymbol{\varepsilon}_t^h, r_{t+1}^h)$ for $t = \{1, 2, \dots, T+1\}$.

    Compute: Return $R^h = \sum_{t=1}^{T+1} r_t^h$ and eligibility $\psi^{h,n} = \sum_{t=1}^{T} (\sigma_h^n)^{-2} \varepsilon_t^{h,n} \phi^n(s_t^{h,n}, t)$ for each parameter $n = \{1, \dots, N\}$ from rollouts.

    Compute Gradient:
        $\left[\mathbf{g}_{\mathrm{eNAC}}^{\mathrm{T}}, R_{\mathrm{ref}}\right]^{\mathrm{T}} = \left(\boldsymbol{\Psi}^{\mathrm{T}}\boldsymbol{\Psi}\right)^{-1} \boldsymbol{\Psi}^{\mathrm{T}}\mathbf{R}$
    with $\mathbf{R} = [R^1, \dots, R^H]^{\mathrm{T}}$ and $\boldsymbol{\Psi} = \begin{bmatrix} \boldsymbol{\psi}^1, \dots, \boldsymbol{\psi}^H \\ 1, \dots, 1 \end{bmatrix}^{\mathrm{T}}$ where $\boldsymbol{\psi}^h = [\psi^{h,1}, \dots, \psi^{h,N}]^{\mathrm{T}}$.

    Update policy using
        $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \mathbf{g}_{\mathrm{eNAC}}.$
until Convergence $\boldsymbol{\theta}_{k+1} \approx \boldsymbol{\theta}_k$.
```

well-known to avoid this problem in supervised learning while even yielding faster convergence (McLachan and Krishnan 1997). Previously, similar ideas have been explored in immediate reinforcement learning (Dayan and Hinton 1997; Peters and Schaal 2007). In general, an EM-algorithm chooses the next policy parameters $\boldsymbol{\theta}_{n+1}$ such that

$$\boldsymbol{\theta}_{n+1} = \operatorname{argmax}_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}} \left(\boldsymbol{\theta}'\right).$$

In the case where $\pi(\mathbf{a}_t|\mathbf{s}_t, t)$ belongs to the exponential family, the next policy can be determined analytically by setting (2) or (3) to zero

$$E\left\{\sum_{t=1}^{T} \partial_{\boldsymbol{\theta}'} \log \pi \left(\mathbf{a}_t|\mathbf{s}_t, t\right) Q\left(\mathbf{s}, \mathbf{a}, t\right)\right\} = 0, \tag{4}$$

and solving for $\boldsymbol{\theta}'$. Depending on the choice of stochastic policy, we will obtain different solutions and different learning algorithms. It allows the extension of the reward-weighted regression to longer horizons as well as the introduction of the Policy learning by Weighting Exploration with the Returns (PoWER) algorithm.

2.3 Policy learning by weighting exploration with the returns (PoWER)

In most learning control problems, we attempt to have a deterministic mean policy $\bar{\mathbf{a}} = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{s}, t)$ with parameters $\boldsymbol{\theta}$ and basis functions $\boldsymbol{\phi}$. In Sect. 3.1, we will introduce a particular type of basis function well-suited for robotics. These basis functions derive from the motor primitive formulation. Given such a deterministic mean policy $\bar{\mathbf{a}} = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{s}, t)$, we generate a stochastic policy using additive exploration $\boldsymbol{\varepsilon}(\mathbf{s}, t)$ in order to make model-free reinforcement learning possible. We have a policy $\pi(\mathbf{a}_t|\mathbf{s}_t, t)$ that can be brought into the form

$$\mathbf{a} = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{s}, t\right) + \boldsymbol{\epsilon}\left(\boldsymbol{\phi}\left(\mathbf{s}, t\right)\right).$$

Previous work in this setting (Williams 1992; Guenter et al. 2007; Peters and Schaal 2006, 2007), with the notable exception of (Rückstieß et al. 2008), has focused on state-independent, white Gaussian exploration, namely $\boldsymbol{\epsilon}(\boldsymbol{\phi}(\mathbf{s}, t)) \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \boldsymbol{\Sigma})$. It is straightforward to obtain the Reward-Weighted Regression for episodic RL by solving (4) for $\boldsymbol{\theta}'$, which naturally yields a weighted regression method with the state-action values $Q^{\pi}(\mathbf{s}, \mathbf{a}, t)$ as weights. See Algorithm 3 for an exemplary implementation and Appendix A.2 for the derivation. An optimized MATLAB™ implementation of this algorithm is available at http://www.robot-learning.de/Member/JensKober. This form of exploration has resulted in various applications in robotics such as T-Ball batting (Peters and Schaal 2006), Peg-In-Hole (Gullapalli et al. 1994), constrained reaching movements (Guenter et al. 2007) and operational space control (Peters and Schaal 2007).

However, such unstructured exploration at every step has several disadvantages: (i) it causes a large variance in parameter updates that grows with the number of time-steps (Rückstieß et al. 2008; Peters and Schaal 2006), (ii) it perturbs actions too frequently as the system acts as a low pass filter, and the perturbations average out, thus their effect is washed out, and (iii) it can damage the system executing the trajectory. As the action is perturbed in every time-step the outcome of a trial can change drastically. This effect accumulates with the number of trials and the exploration is not equal over the progress of the trial. This behavior leads to a large variance in parameter updates. Random exploration in every time-step leads to jumps in the actions. A physical robot cannot execute instantaneous

Algorithm 3 episodic Reward Weighted Regression (eRWR)

```
Input: initial policy parameters $\boldsymbol{\theta}_0$
repeat
    Sample: Perform $h = \{1, \ldots, H\}$ rollouts using $\mathbf{a} = \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{s}, t) + \boldsymbol{\varepsilon}_t$ with $[\varepsilon_t^n] \sim \mathcal{N}(0, (\sigma^{h,n})^2)$ as stochastic policy and collect all $(t, \mathbf{s}_t^h, \mathbf{a}_t^h, \mathbf{s}_{t+1}^h, \boldsymbol{\varepsilon}_t^h, r_{t+1}^h)$ for $t = \{1, 2, \ldots, T+1\}$.

    Compute: State-action value function $Q_t^{\pi,h} = \sum_{\tilde{t}=t}^{T} r_{\tilde{t}}^h$ from rollouts.

    Update policy using
        $\boldsymbol{\theta}_{k+1}^n = \left( (\boldsymbol{\Phi}^n)^{\mathrm{T}} \mathbf{Q}^{\pi} \boldsymbol{\Phi}^n \right)^{-1} (\boldsymbol{\Phi}^n)^{\mathrm{T}} \mathbf{Q}^{\pi} \mathbf{A}^n$
    with basis functions
        $\boldsymbol{\Phi}^n = \left[ \boldsymbol{\phi}_1^{1,n}, \ldots, \boldsymbol{\phi}_T^{1,n}, \boldsymbol{\phi}_1^{2,n}, \ldots, \boldsymbol{\phi}_1^{H,n}, \ldots, \boldsymbol{\phi}_T^{H,n} \right]^{\mathrm{T}},$
    where $\boldsymbol{\phi}_t^{h,n}$ is the value of the basis function of rollout $h$ and parameter $n$ at time $t$, actions
        $\mathbf{A}^n = \left[ a_1^{1,n}, \ldots, a_T^{1,n}, a_1^{2,n}, \ldots, a_1^{H,n}, \ldots, a_T^{H,n} \right]^{\mathrm{T}},$
    and returns
        $\mathbf{Q}^{\pi} = \operatorname{diag}\left( Q_1^{\pi,1}, \ldots, Q_T^{\pi,1}, Q_1^{\pi,2}, \ldots, Q_1^{\pi,H}, \ldots, Q_T^{\pi,H} \right)$
until Convergence $\boldsymbol{\theta}_{k+1} \approx \boldsymbol{\theta}_k$.
```

changes in actions as either the controller needs time to react or the motor and the links of the robot have inertia that forces the robot to continue the motion induced by the previous actions. Globally speaking, the system acts as a low pass filter. If the robot tries to follow the desired high frequency action changes, a lot of strain is placed on the mechanics of the robot and can lead to oscillations. Furthermore, the accumulating effect of the exploration can lead the robot far from previously seen states, which is potentially dangerous.

As a result, all methods relying on this state-independent exploration have proven too fragile for learning tasks such as the Ball-in-a-Cup (see Sect. 3.7) on a real robot system. Alternatively, as introduced in (Rückstieß et al. 2008), one could generate a form of structured, state-dependent exploration. We use

$$\boldsymbol{\epsilon}\left(\boldsymbol{\phi}\left(\mathbf{s}, t\right)\right) = \boldsymbol{\varepsilon}_t^{\mathrm{T}} \boldsymbol{\phi}\left(\mathbf{s}, t\right)$$

with $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0}, \hat{\boldsymbol{\Sigma}})$, where $\hat{\boldsymbol{\Sigma}}$ is a meta-parameter of the exploration that can be optimized in a similar manner (see Appendix A.3). This argument results in the policy

$$\mathbf{a} \sim \pi\left(\mathbf{a}_t | \mathbf{s}_t, t\right) = \mathcal{N}\left(\mathbf{a} | \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\phi}\left(\mathbf{s}, t\right), \boldsymbol{\phi}(\mathbf{s}, t)^{\mathrm{T}} \hat{\boldsymbol{\Sigma}} \boldsymbol{\phi}(\mathbf{s}, t) \right).$$

Inserting the resulting policy into (4), we obtain the optimality condition update and can derive the update rule

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + E\left\{\sum_{t=1}^{T} \mathbf{W}(\mathbf{s}, t)\, Q^{\pi}(\mathbf{s}, \mathbf{a}, t)\right\}^{-1} E\left\{\sum_{t=1}^{T} \mathbf{W}(\mathbf{s}, t)\, \boldsymbol{\varepsilon}_t\, Q^{\pi}(\mathbf{s}, \mathbf{a}, t)\right\}$$

with $\mathbf{W}(\mathbf{s}, t) = \boldsymbol{\phi}(\mathbf{s}, t)\boldsymbol{\phi}(\mathbf{s}, t)^{\mathrm{T}}(\boldsymbol{\phi}(\mathbf{s}, t)^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}\boldsymbol{\phi}(\mathbf{s}, t))^{-1}$.

In order to reduce the number of rollouts in this on-policy scenario, we reuse the rollouts through importance sampling as described, in the context of reinforcement learning, in (Andrieu et al. 2003; Sutton and Barto 1998). The expectations $E\{\cdot\}$ are replaced by the importance sampler denoted by $\langle\cdot\rangle_{w(\boldsymbol{\tau})}$. To avoid the fragility sometimes resulting from importance sampling in reinforcement learning, samples with very small importance weights are discarded. This step is necessary as a lot of rollouts with a low return accumulate mass and can bias the update. A simple heuristic that works well in practice is to discard all but the j best rollouts, where j is chosen in the same order of magnitude as the number of parameters N. The derivation is shown in Appendix A.3 and the resulting algorithm in Algorithm 4. Note that for our motor primitives, some simplifications of $\mathbf{W}$ are possible. These and other simplifications are shown in Appendix A.3. A MATLAB™ implementation of this algorithm in several variants is available at http://www.robot-learning.de/Member/JensKober. As we will see in Sect. 3, this PoWER method significantly outperforms all other described methods.

Algorithm 4 EM Policy learning by Weighting Exploration with the Returns (PoWER)

```
**Input:** initial policy parameters $\boldsymbol{\theta}_0$
**repeat**
    *Sample*: Perform rollout(s) using $\mathbf{a} = (\boldsymbol{\theta} + \boldsymbol{\varepsilon}_t)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{s}, t)$ with $\boldsymbol{\varepsilon}_t^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{s}, t) \sim \mathcal{N}(0, \boldsymbol{\phi}(\mathbf{s}, t)^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}\boldsymbol{\phi}(\mathbf{s}, t))$ as stochastic policy and collect all $(t, \mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, \boldsymbol{\varepsilon}_t, r_{t+1})$ for $t = \{1, 2, \ldots, T+1\}$.

    *Estimate*: Use unbiased estimate
        $\hat{Q}^{\pi}(\mathbf{s}, \mathbf{a}, t) = \sum_{\tilde{t}=t}^{T} r\left(\mathbf{s}_{\tilde{t}}, \mathbf{a}_{\tilde{t}}, \mathbf{s}_{\tilde{t}+1}, \tilde{t}\right).$

    *Reweight*: Compute importance weights and reweight rollouts, discard low-importance rollouts.

    *Update* policy using
        $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \left\langle\sum_{t=1}^{T} \mathbf{W}(\mathbf{s}, t)\, Q^{\pi}(\mathbf{s}, \mathbf{a}, t)\right\rangle_{w(\boldsymbol{\tau})}^{-1} \left\langle\sum_{t=1}^{T} \mathbf{W}(\mathbf{s}, t)\, \boldsymbol{\varepsilon}_t\, Q^{\pi}(\mathbf{s}, \mathbf{a}, t)\right\rangle_{w(\boldsymbol{\tau})}$
    with $\mathbf{W}(\mathbf{s}, t) = \boldsymbol{\phi}(\mathbf{s}, t)\boldsymbol{\phi}(\mathbf{s}, t)^{\mathrm{T}}(\boldsymbol{\phi}(\mathbf{s}, t)^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}\boldsymbol{\phi}(\mathbf{s}, t))^{-1}$.

**until** Convergence $\boldsymbol{\theta}_{k+1} \approx \boldsymbol{\theta}_k$.
```

PoWER is very robust with respect to reward functions. The key constraint is that it has to be an improper probability distribution which means that the rewards have to be positive. It can be beneficial for learning speed if the reward function sums up to one as a proper probability distribution.

Like most learning algorithms, PoWER achieves convergence faster for lower numbers of parameters. However, as it is an EM-inspired approach, it suffers significantly less from this problem than gradient based approaches. Including more prior knowledge, either in the parametrization or the initial policy, leads to faster convergence. As discussed above, changing exploration at every time-step has a number of disadvantages. Fixing the exploration for the whole episode (if each basis function is only active for a short time) or using a slowly varying exploration (for example based on random walks) can increase the performance. All algorithms introduced in this paper optimize locally and can get stuck in local optima. An initial policy should be chosen to avoid local optima on the progress towards the desired final solution.

3 Benchmark evaluation and application in robotics

In this section, we demonstrate the effectiveness of the algorithms presented in Sect. 2.3 in the context of motor primitive learning for robotics. We will first give a quick overview of how the motor primitives (Ijspeert et al. 2002, 2003; Schaal et al. 2003, 2007) work and how learning algorithms can be used to adapt them. Subsequently, we will discuss how we can turn the parametrized motor primitives (Ijspeert et al. 2002, 2003; Schaal et al. 2003, 2007) into explorative, stochastic policies (Rückstieß et al. 2008). We show that the novel PoWER algorithm outperforms many previous well-known methods, particularly 'Vanilla' Policy Gradients (Williams 1992; Sutton et al. 2000; Lawrence et al. 2003; Peters and Schaal 2006), Finite Difference Gradients (Sehnke et al. 2010; Peters and Schaal 2006), the episodic Natural Actor Critic (Peters et al. 2003, 2005), and the generalized Reward-Weighted Regression (Peters and Schaal 2007) on the two simulated benchmark problems suggested by Peters and Schaal (2006) and the Underactuated Swing-Up (Atkeson 1994). We compare policy search based algorithms to a value function based one on the mountain-car benchmark. Additionally, we evaluate policy search methods on the multidimensional robotic tasks Tetherball Target Hitting and Casting. As a significantly more complex motor learning task, we will show how the robot can learn a high-speed Ball-in-a-Cup movement (Sumners 1997) with motor primitives for all seven degrees of freedom of our Barrett WAM™ robot arm. An overview of the experiments is presented in Table 1.

3.1 Dynamical systems motor primitives

In the analytically tractable cases, episodic Reinforcement Learning (RL) problems have been studied deeply in the optimal control community. In this field it is well-known that for a finite horizon problem, the optimal solution is non-stationary (Kirk 1970) and, in general, cannot be represented by a time-independent policy. The motor primitives based on dynamical systems (Ijspeert et al. 2002, 2003; Schaal et al. 2003, 2007) represent a particular type of time-variant policy that has an internal phase, which corresponds to a clock with additional flexibility (for example, for incorporating coupling effects, perceptual influences, etc.). Thus, they can represent optimal solutions for finite horizons. We embed this internal clock or movement phase into our state and from an optimal control perspective have ensured that the optimal solution can be represented.

Table 1 Overview of the Experiments: 3.2 Basic Motor Learning, 3.3 Mountain-Car, 3.4 Tetherball Target Hitting, 3.5 Underactuated Swing-Up, 3.6 Casting, and 3.7 Ball-in-a-Cup

	Open parameters	DoF	Rollouts	Policy	Platform	Algorithms
3.2	10 (shape)	1	4400	motor primitive	simulation	FDG, VPG, eNAC, eRWR, PoWER
3.3	2 (switching)	1	80	bang-bang	simulation	FDG, PoWER, *k*NN-TD(λ)
3.4	6 (positions)	1	200	rhythmic	simulation	FDG, PoWER
3.5	10 (goal & shape)	1	200/100	motor primitive	simu/robot	FDG, VPG, eNAC, eRWR, PoWER
3.6	10 (shape)	2	200	motor primitive	simulation	eNAC, PoWER
3.7	217 (shape)	7	100	motor primitive	robot	PoWER

The original formulation of the dynamical system motor primitives in (Ijspeert et al. 2003) was a major breakthrough as the choice of dynamical systems allows the determination of the stability of the movement, choosing between a rhythmic and a discrete movement, and the behavior is invariant under rescaling in both time and movement amplitude. In this paper, we focus on single stroke movements as they frequently appear in human motor control (Wulf 2007; Schaal et al. 2007). Therefore, we will always choose the point attractor version of the dynamical system motor primitives. We use the most recent formulation of the discrete dynamical systems motor primitives (Schaal et al. 2007) where the phase z of the movement is represented by a single first order system

$$\dot{z} = -\tau \alpha_z z. \tag{5}$$

This canonical system has the time constant $\tau = 1/T$ where T is the duration of the motor primitive and a parameter α_z, which is chosen such that $z \approx 0$ at T. Subsequently, the internal state $\mathbf{x}$ of a second system is chosen such that positions $\mathbf{q}$ of all degrees of freedom are given by $\mathbf{q} = \mathbf{x}_1$, the velocities by $\dot{\mathbf{q}} = \tau \mathbf{x}_2 = \dot{\mathbf{x}}_1$, and the accelerations by $\ddot{\mathbf{q}} = \tau \dot{\mathbf{x}}_2$. The learned dynamics of Ijspeert motor primitives can be expressed in the following form

$$\begin{aligned} \dot{\mathbf{x}}_2 &= \tau \alpha_x \left(\beta_x \left(\mathbf{g} - \mathbf{x}_1 \right) - \mathbf{x}_2 \right) + \tau \mathbf{A} \mathbf{f}(z), \\ \dot{\mathbf{x}}_1 &= \tau \mathbf{x}_2. \end{aligned} \tag{6}$$

This set of differential equations has the same time constant τ as the canonical system, parameters α_x, β_x are set such that the system is critically damped, a goal parameter $\mathbf{g}$, a transformation function $\mathbf{f}$, and an amplitude matrix $\mathbf{A} = \operatorname{diag}(a_1, a_2, \ldots, a_I)$, with the amplitude modifier $\mathbf{a} = [a_1, a_2, \ldots, a_I]$. Schaal et al. (2007) use $\mathbf{a} = \mathbf{g} - \mathbf{x}_1^0$, with the initial position $\mathbf{x}_1^0$, which ensures linear scaling. Other choices are possibly better suited for specific tasks, as shown in (Park et al. 2008). The transformation function $\mathbf{f}(z)$ alters the output of the system in (5) so that the system in (6) can represent complex nonlinear patterns. It is given by

$$\mathbf{f}(z) = \sum_{n=1}^{N} \varphi_n(z) \boldsymbol{\theta}_n z. \tag{7}$$

Here $\boldsymbol{\theta}_n$ contains the nth adjustable parameter of all degrees of freedom (DoF), N is the number of parameters per degree of freedom, and $\varphi_n(z)$ are the corresponding weighting functions (Schaal et al. 2007). Normalized Gaussian kernels are used as weighting functions, given by

$$\varphi_n = \frac{\exp(-h_n(z-c_n)^2)}{\sum_{m=1}^{N}\exp(-h_m(z-c_m)^2)}. \tag{8}$$

These weighting functions localize the interaction in phase space using the centers c_n and widths h_n. As $z \approx 0$ at T, the influence of the transformation function $\mathbf{f}(z)$ in (7) vanishes and the system stays at the goal position $\mathbf{g}$. Note that the DoF are usually all modeled independently in the system in (6). The movement of all DoFs is synchronized as the dynamical systems for all DoFs start at the same time and have the same duration. The shape of the movement is generated using the transformation $\mathbf{f}(z)$ in (7), which is learned as a function of the shared canonical system in (5). Additional feedback terms can be added as shown in (Ijspeert et al. 2003; Schaal et al. 2007; Kober et al. 2008; Park et al. 2008).

One of the biggest advantages of this motor primitive framework (Ijspeert et al. 2002, 2003; Schaal et al. 2003, 2007) is that the second system, in (6), is linear in the policy parameters $\boldsymbol{\theta}$ and is therefore well-suited for both imitation learning as well as for the presented reinforcement learning algorithms. For example, if we would have to learn only a motor primitive for a single degree of freedom q_i, then we could use a motor primitive in the form $\bar{\ddot{q}}_i = \boldsymbol{\phi}(\mathbf{s})^{\mathrm{T}}\boldsymbol{\theta}$ where $\mathbf{s} = [q_i, \dot{q}_i, z]$ is the state and where time is implicitly embedded in z. We use the output of $\bar{\ddot{q}}_i = \boldsymbol{\phi}(\mathbf{s})^{\mathrm{T}}\boldsymbol{\theta} = \bar{a}$ as the policy mean. The perturbed accelerations $\ddot{q}_i = a = \bar{a} + \varepsilon$ are given to the system.

In Sects. 3.5 and 3.7, we use imitation learning from a single example to generate a sensible initial policy. This step can be performed efficiently in the context of dynamical systems motor primitives as the transformation function equation (7) is linear in its parameters. As a result, we can choose the weighted squared error (WSE)

$$\mathrm{WSE}_n = \sum_{t=1}^{T} \varphi_t^n \left(f_t^{\mathrm{ref}} - z_t\theta^n\right)^2 \tag{9}$$

as cost function and minimize it for all parameter vectors θ^n with $n \in \{1, 2, \ldots, N\}$. Here, the corresponding weighting functions are denoted by ψ_t^n and the basis function by z_t. The reference or target signal f_t^{ref} is the desired transformation function and t indicates the time-step of the sample. The error in (9) can be rewritten in matrix form as

$$\mathrm{WSE}_n = \left(\mathbf{f}^{\mathrm{ref}} - \mathbf{Z}\theta^n\right)^{\mathrm{T}} \boldsymbol{\Phi} \left(\mathbf{f}^{\mathrm{ref}} - \mathbf{Z}\theta^n\right)$$

with $\mathbf{f}^{\mathrm{ref}}$ containing the values of f_t^{ref} for all time-steps t, $\boldsymbol{\Phi} = \mathrm{diag}(\varphi_1^n, \ldots, \varphi_t^n, \ldots, \varphi_T^n)$, and $[\mathbf{Z}]_t = z_t$. As a result, we have a standard locally-weighted linear regression problem that is straightforward to solve and yields the unbiased parameter estimator

$$\theta^n = \left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\Psi}\mathbf{Z}\right)^{-1}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{f}^{\mathrm{ref}}.$$

This approach was originally suggested for imitation learning by Ijspeert et al. (2003). Estimating the parameters of the dynamical system is slightly more difficult; the duration of the movement is extracted using motion detection and the time-constant is set accordingly.

3.2 Benchmark comparison I: basic motor learning examples

As a benchmark comparison, we follow a previously studied scenario in order to evaluate, which method is best-suited for our problem class. We perform our evaluations on exactly the same benchmark problems as in (Peters and Schaal 2006) and use two tasks commonly studied in motor control literature for which the analytic solutions are known. The first task is a reaching task, wherein a goal has to be reached at a certain time, while the used motor commands have to be minimized. The second task is a reaching task of the same style with an additional via-point. The task is illustrated in Fig. 1. This comparison mainly shows the suitability of our algorithm (Algorithm 4) and that it outperforms previous methods such as Finite Difference Gradient (FDG) methods (Sehnke et al. 2010; Peters and Schaal 2006), see Algorithm 5, 'Vanilla' Policy Gradients (VPG) with optimal baselines (Williams 1992;

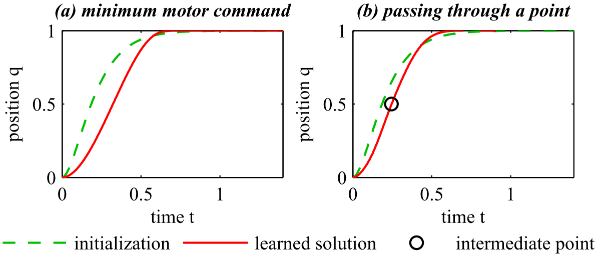


Fig. 1 (Color online) This figure shows the initial and the final trajectories for the two motor control tasks. Both start at 0 and have to go to 1 with minimal accelerations. From $T/2 = 0.75$ on the trajectory has to be as close to 1 as possible. For the passing through task the trajectory additionally has to pass through $p_M = 0.5$ at time $M = 7/40T$ indicated by the *circle*

Algorithm 5 Finite Difference Gradients (FDG)

```
Input: initial policy parameters $\boldsymbol{\theta}_0$
repeat
    Generate policy variations: $\Delta\boldsymbol{\theta}^h \sim \mathcal{U}_{[-\Delta\boldsymbol{\theta}_{\min}, \Delta\boldsymbol{\theta}_{\max}]}$ for $h = \{1, \ldots, H\}$ rollouts.

    Sample: Perform $h = \{1, \ldots, H\}$ rollouts using $\mathbf{a} = (\boldsymbol{\theta} + \Delta\boldsymbol{\theta}^h)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{s}, t)$ as policy and collect all $(t, \mathbf{s}_t^h, \mathbf{a}_t^h, \mathbf{s}_{t+1}^h, \boldsymbol{\varepsilon}_t^h, r_{t+1}^h)$ for $t = \{1, 2, \ldots, T+1\}$.

    Compute: Return $R^h(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}^h) = \sum_{t=1}^{T+1} r_t^h$ from rollouts.

    Compute Gradient:
        $$[\mathbf{g}_{\mathrm{FD}}^{\mathrm{T}}, R_{\mathrm{ref}}]^{\mathrm{T}} = \left(\Delta\boldsymbol{\Theta}^{\mathrm{T}}\Delta\boldsymbol{\Theta}\right)^{-1}\Delta\boldsymbol{\Theta}^{\mathrm{T}}\mathbf{R}$$
    with $\Delta\boldsymbol{\Theta} = \begin{bmatrix} \Delta\boldsymbol{\theta}^1, \ldots, \Delta\boldsymbol{\theta}^H \\ 1, \ldots, 1 \end{bmatrix}^{\mathrm{T}}$ and $\mathbf{R} = [R^1, \ldots, R^H]^{\mathrm{T}}$.

    Update policy using
        $$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha\mathbf{g}_{\mathrm{FD}}.$$
until Convergence $\boldsymbol{\theta}_{k+1} \approx \boldsymbol{\theta}_k$.
```

Sutton et al. 2000; Lawrence et al. 2003; Peters and Schaal 2006), see Algorithm 1, the episodic Natural Actor Critic (eNAC) (Peters et al. 2003, 2005), see Algorithm 2, and the new episodic version of the Reward-Weighted Regression (eRWR) algorithm (Peters and Schaal 2007), see Algorithm 3. MATLAB™ implementations of all algorithms are available at http://www.robot-learning.de/Member/JensKober. For all algorithms except PoWER, we used batches to update the policy. A sliding-window based approach is also possible. For VPG, eNAC, and eRWR a batch size of $H = 2N$ and for FDG a batch size of $H = N + 1$ are typical. For PoWER, we employed an importance sampling based approach, although a batch based update is also possible.

We consider two standard tasks taken from (Peters and Schaal 2006), but we use the newer form of the motor primitives from (Schaal et al. 2007). The first task is to achieve a goal with a minimum-squared movement acceleration and a given movement duration, that gives a return of

$$R(\boldsymbol{\tau}) = -\sum_{t=0}^{T/2} c_1 \ddot{q}_t^2 - \sum_{t=T/2+1}^{T} c_2 \left((q_t - g)^2 + \dot{q}_t^2\right)$$

for optimization, where $T = 1.5$, $c_1 = 1/100$ is the weight of the transient rewards for the movement duration $T/2$, while $c_2 = 1000$ is the importance of the final reward, extended over the time interval $[T/2 + 1, T]$ which insures that the goal state $g = 1.0$ is reached and maintained properly. The initial state of the motor primitive is always zero in this evaluation.

The second task involves passing through an intermediate point during the trajectory, while minimizing the squared accelerations, that is, we have a similar return with an additional punishment term for missing the intermediate point p_M at time M given by

$$R(\boldsymbol{\tau}) = -\sum_{t=0}^{T/2} \tilde{c}_1 \ddot{q}_t^2 - \sum_{t=T/2+1}^{T} \tilde{c}_2 \left((q_t - g)^2 + \dot{q}_t^2\right) - \tilde{c}_3 (q_M - p_M)^2$$

where $\tilde{c}_1 = 1/10000$, $\tilde{c}_2 = 200$, $\tilde{c}_3 = 20000$. The goal is given by $g = 1.0$, the intermediate point a value of $p_M = 0.5$ at time $M = 7/40T$, and the initial state was zero. This return yields a smooth movement, which passes through the intermediate point before reaching the goal, even though the optimal solution is not identical to the analytic solution with hard constraints.

All open parameters were manually optimized for each algorithm in order to maximize the performance while not destabilizing the convergence of the learning process. When applied in the episodic scenario, Policy learning by Weighting Exploration with the Returns (PoWER) clearly outperformed the episodic Natural Actor Critic (eNAC), 'Vanilla' Policy Gradient (VPG), Finite Difference Gradient (FDG), and the episodic Reward-Weighted Regression (eRWR) for both tasks. The episodic Reward-Weighted Regression (eRWR) is outperformed by all other algorithms suggesting that this algorithm does not generalize well from the immediate reward case. While FDG gets stuck on a plateau, both eNAC and VPG converge to the same good final solution. PoWER finds a slightly better solution while converging noticeably faster. The results are presented in Fig. 2.

3.3 Benchmark comparison II: mountain-car

As a typical reinforcement learning benchmark we chose the mountain-car task (Sutton and Barto 1998) as it can be treated with episodic reinforcement learning. In this problem we have a car placed in a valley, and it is supposed to go on the top of the mountain in front of

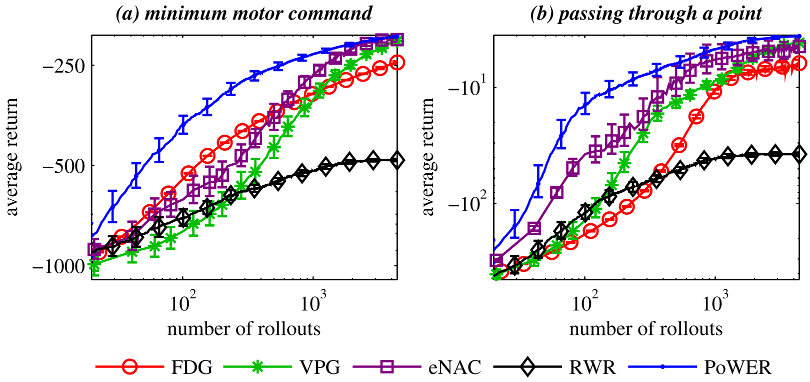
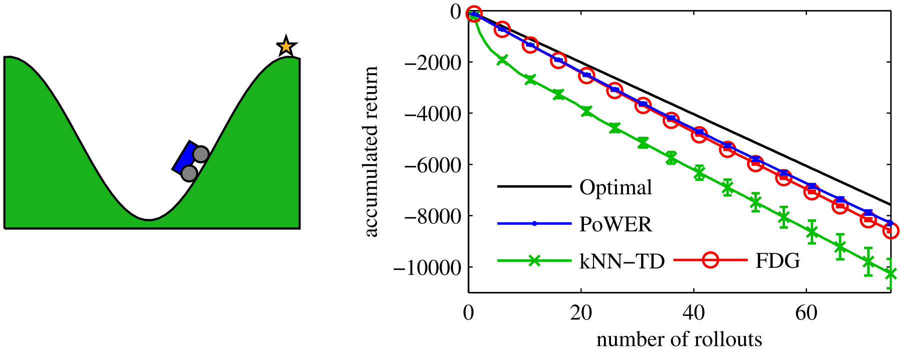


Fig. 2 (Color online) This figure shows the mean performance of all compared methods in two benchmark tasks averaged over twenty learning runs with the error bars indicating the standard deviation. Policy learning by Weighting Exploration with the Returns (PoWER) clearly outperforms Finite Difference Gradients (FDG), 'Vanilla' Policy Gradients (VPG), the episodic Natural Actor Critic (eNAC), and the adapted Reward-Weighted Regression (eRWR) for both tasks. Note that this plot has logarithmic scales on both axes, thus a unit difference corresponds to an order of magnitude. The omission of the first twenty rollouts was necessary to cope with the log-log presentation

it, but does not have the necessary capabilities of acceleration to do so directly. Thus, the car has to first drive up the mountain on the opposite side of the valley to gain sufficient energy. The dynamics are given in (Sutton and Barto 1998) as

$$\dot{x}_{t+1} = \dot{x}_t + 0.001 a_t - 0.0025 \cos(3x_t),$$
$$x_{t+1} = x_t + \dot{x}_{t+1},$$

with position $-1.2 \le x_{t+1} \le 0.5$ and velocity $-0.07 \le \dot{x}_{t+1} \le 0.07$. If the goal $x_{t+1} \ge 0.5$ is reached the episode is terminated. If the left bound is reached the velocity is reset to zero. The initial condition of the car is $x_0 = -0.5$ and $\dot{x}_0 = 0$. The reward is $r_t = -1$ for all time-steps until the car reaches the goal. We employed an undiscounted return. The set of actions a_t is slightly different to the setup proposed by Sutton and Barto (1998). We only have two actions, the full throttle forward ($a_t = +1$) and the full throttle reverse ($a_t = -1$). From a classical optimal control point of view, it is straightforward to see that a bang-bang controller can solve this problem. As an initial policy we chose a policy that accelerates forward until the car cannot climb the mountain further, accelerates reverse until the car cannot climb the opposite slope further, and finally accelerates forward until the car reaches the goal. This policy reaches the goal but is not optimal as the car can still accumulate enough energy if it reverses the direction slightly earlier. As a parametrization for the policy search approaches we chose to encode the switching points of the acceleration. The two parameters of the policy indicate at which timestep t the acceleration is reversed. For this kind of policy only algorithms that perturb the parameters are applicable and we compare a Finite Difference Gradient approach to PoWER. This parametrized policy is entirely different to motor primitives. Additionally we included a comparison to a value function based method. The Q-function was initialized with our initial policy. As the kNN-TD(λ) algorithm (Martín et al. 2009) won the Reinforcement Learning Competitions in 2008 and 2009, we selected it for this comparison. This comparison is contrived as our switching policy always starts in a similar initial state while the value function based policy can start in a wider range



(a) The tasks consists of driving the underpowered car to the target on the mountain indicated by the yellow star.

(b) This figure shows the mean accumulated returns of the methods compared on the mountain car benchmark. The results are averaged over fifty learning runs with error bars inidicating the standard deviation. Policy learning by Weighting Exploration with the Returns (PoWER) and Finite Difference Gradients (FDG) clearly outperform kNN-TD(λ). All methods converge to the optimal solution.

Fig. 3 (Color online) This figure shows an illustration of the mountain-car task and the mean accumulated returns of the compared methods

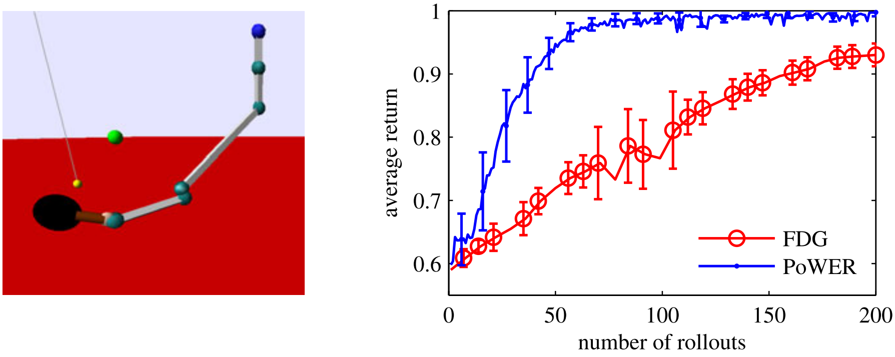
of states. Furthermore, the policy search approaches may be sped up by the initialization, while kNN-TD(λ) will learn the optimal policy without prior knowledge and does not benefit much from the initialization. However, the use of a global approach, such as kNN-TD(λ) requires a global search of the state space. Such a global search limits the scalability of these approaches. The more local approaches of policy search are less affected by these scalability problems. Figure 3(b) shows the performance of these algorithms. As kNN-TD(λ) initially explores the unseen parts of the Q-function, the policy search approaches converge faster. All methods find the optimal solution.

3.4 Benchmark comparison III: tetherball target hitting

In this task, a table tennis ball is hanging on an elastic string from the ceiling. The task consists of hitting the ball with a table tennis racket so that it hits a fixed target. The task is illustrated in Fig. 4(a). The return is based on the minimum distance between the ball and the target during one episode transformed by an exponential. The policy is parametrized as the position of the six lower degrees of freedom of the Barrett WAM™. Only the first degree of freedom (shoulder rotation) is moved during an episode. The movement is represented by a rhythmic policy with a fixed amplitude and period. Due to the parametrization of the task only PoWER and Finite Difference Gradients are applicable. We observed reliable performance if the initial policy did not miss the target by more than approximately 50 cm. In this experiment it took significantly less iterations to find a good initial policy than to tune the learning rate of Finite Difference Gradients, a problem from which PoWER did not suffer as it is an EM-like algorithm. Figure 4(b) illustrates the results. PoWER converges significantly faster.

3.5 Benchmark comparison IV: underactuated swing-up

As an additional simulated benchmark and for the real-robot evaluations, we employed the Underactuated Swing-Up (Atkeson 1994). Here, only a single degree of freedom is repre-



(a) The tasks consists of striking the yellow ball hanging on an elastic string such that it hits the green target.

(b) The returns are averaged over 20 learning runs with error bars indicating the standard deviation. Policy learning by Weighting Exploration with the Returns (PoWER) clearly outperforms Finite Difference Gradients (FDG).

Fig. 4 (Color online) This figure shows an illustration of the Tetherball Target Hitting task and the mean returns of the compared methods

sented by the motor primitive as described in Sect. 3.1. The goal is to move a hanging heavy pendulum to an upright position and to stabilize it there. The objective is threefold: the pendulum has to be swung up in a minimal amount of time, has to be stabilized in the upright position, and achieve these goals with minimal motor torques. By limiting the motor current for this degree of freedom, we can ensure that the torque limits described in (Atkeson 1994) are maintained and directly moving the joint to the right position is not possible. Under these torque limits, the robot needs to first move away from the target to reduce the maximal required torque during the swing-up, see Fig. 5. This problem is similar to the mountain-car problem (Sect. 3.3). The standard mountain-car problem is designed to get the car to the top of the mountain in minimum time. It does not matter if it stops at this point or drives at a high speed as usage of the accelerator and brake is not punished. Adding the requirement of stabilizing the car at the top of the mountain makes the problem significantly harder. These additional constraints exist in the Underactuated Swing-Up task where it is required that the pendulum (the equivalent of the car) stops at the top to fulfill the task.

The applied torque limits were the same as in (Atkeson 1994) and so was the reward function, except that the complete return of the trajectory was transformed by an $\exp(\cdot)$ to ensure positivity. The reward function is given by

$$r\left(t\right) = -c_1 q\left(t\right)^2 + c_2 \log\cos\left(c_3 \frac{u(t)}{u_{\max}}\right),$$

where the constants are $c_1 = 5/\pi^2 \approx 0.507$, $c_2 = (2/\pi)^2 \approx 0.405$, and $c_3 = \pi/2 \approx 1.571$. Please note that π refers to the mathematics constant here, and not to the policy. The first term of the sum is punishing the distance to the desired upright position $q = 0$, and the second term is punishing the usage of motor torques u. A different trade-off can be achieved by changing the parameters or the structure of the reward function, as long as it remains an improper probability function. Again all open parameters of all algorithms were manually optimized. The motor primitive with nine shape parameters and one goal parameter was initialized by imitation learning from a kinesthetic teach-in. Kinesthetic teach-in means

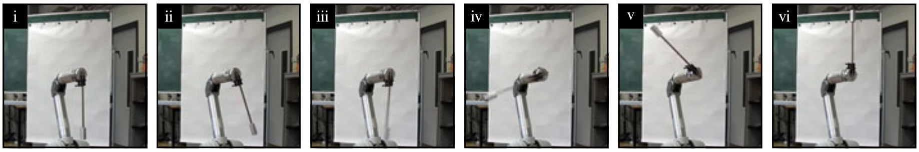


Fig. 5 This figure shows the time series of the Underactuated Swing-Up where only a single joint of the robot is moved with a torque limit ensured by limiting the maximal motor current of that joint. The resulting motion requires the robot to (ii) first move away from the target to limit the maximal required torque during the swing-up in (iii–v) and subsequent stabilization (vi)

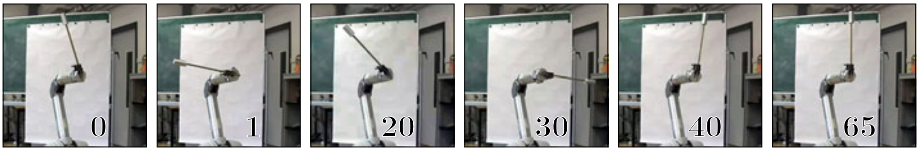


Fig. 6 This figure shows the improvement of the policy over rollouts. The snapshots from the video show the final positions. (0) Initial policy after imitation learning (without torque limit). (1) Initial policy after imitation learning (with active torque limit). (20) Policy after 20 rollouts, going further up. (30) Policy after 30 rollouts, going too far. (40) Policy after 40 rollouts, going only a bit too far. (65) Final policy after 65 rollouts

"taking the robot by the hand", performing the task by moving the robot while it is in gravity-compensation mode, and recording the joint angles, velocities, and accelerations. This initial demonstration needs to include all the relevant features of the movement, e.g., it should first move away from the target and then towards the upright position. The performance of the algorithms is very robust, as long as the initial policy with active torque limits moves the pendulum approximately above the horizontal orientation.

As the batch size, and, thus the learning speed, of the gradient based approaches depend on the number of parameters (see Sect. 3.2), we tried to minimize the number of parameters. Using more parameters would allow us to control more details of the policy which could result in a better final policy, but would have significantly slowed down convergence. At least nine shape parameters were needed to ensure that the imitation can capture the movement away from the target, which is essential to accomplish the task. We compared all algorithms considered in Sect. 3.2 and could show that PoWER would again outperform the others. The convergence is a lot faster than in the basic motor learning examples (see Sect. 3.2), as we do not start from scratch but rather from an initial solution that allows significant improvements in each step for each algorithm. The results are presented in Fig. 7. See Fig. 6 and Fig. 8 for the resulting real-robot performance.

3.6 Benchmark comparison V: casting

In this task a ball is attached to the endeffector of the Barrett WAM™ by a string. The task is to place the ball into a small cup in front of the robot. The task is illustrated in Fig. 9(a). The return is based on the sum of the minimum distance between the ball and the top, the center, and the bottom of the cup respectively during one episode. Using only a single distance, the return could be successfully optimized, but the final behavior often corresponded to a local maximum (for example hitting the cup from the side). The movement is in a plane and only one shoulder DoF and the elbow are moved. The policy is parametrized using motor

Fig. 7 (Color online) This figure shows the performance of all compared methods for the swing-up in simulation and the mean performance averaged over 20 learning runs with the error bars indicating the standard deviation. PoWER outperforms the other algorithms from 50 rollouts on and finds a significantly better policy

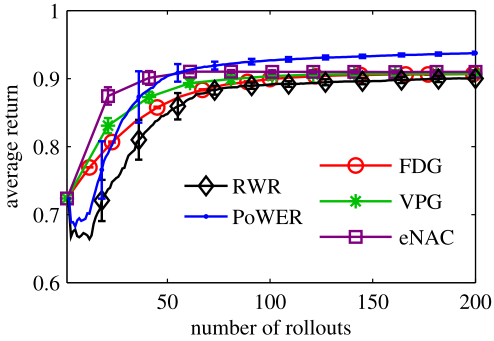
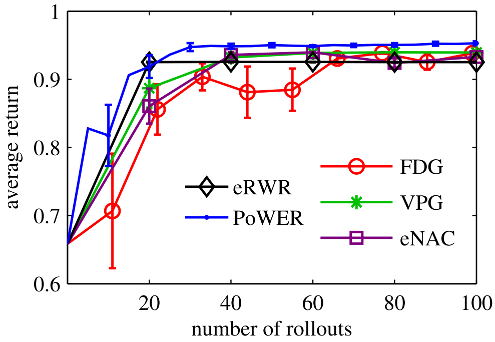
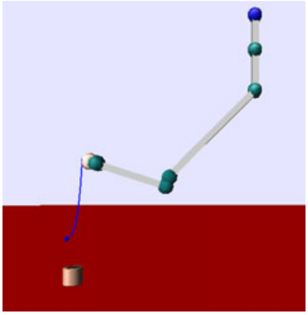
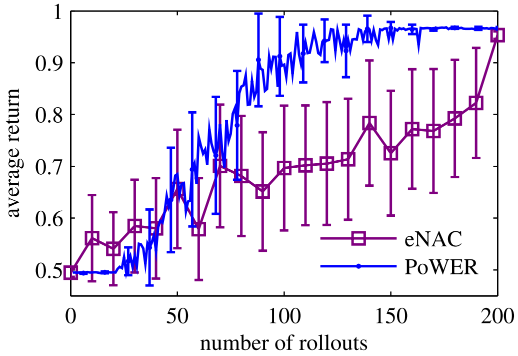


Fig. 8 (Color online) This figure shows the performance of all compared methods for the swing-up on the real robot and the mean performance averaged over 3 learning runs with the error bars indicating the standard deviation. PoWER outperforms the other algorithms and finds a significantly better policy





(a) The tasks consists of placing the blue ball in the brown cup.



(b) The returns are averaged over 20 learning runs with error bars indicating the standard deviation. Policy learning by Weighting Exploration with the Returns (PoWER) clearly outperforms episodic Natural Actor Critic (eNAC).

Fig. 9 (Color online) This figure illustrates the Casting task and shows the mean returns of the compared methods

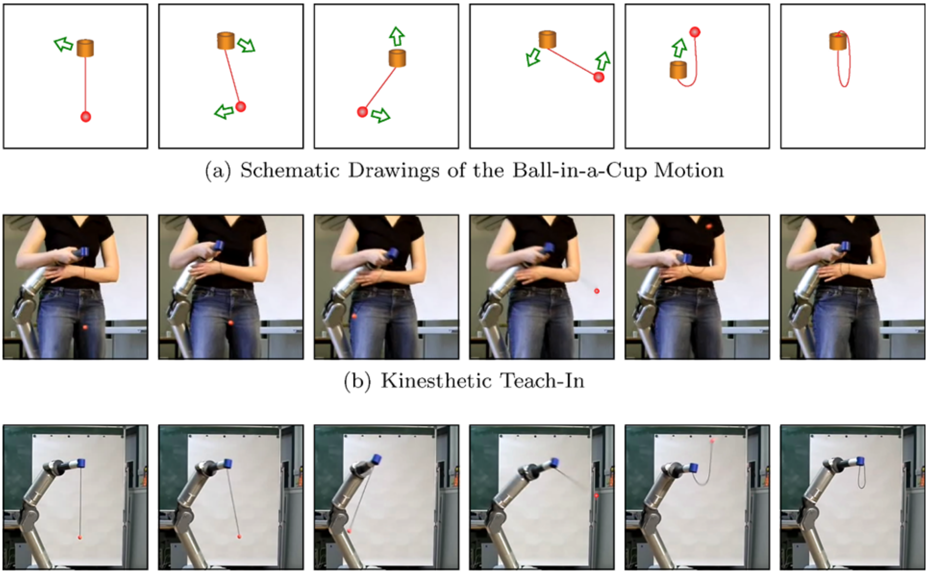
primitives with five shape parameters per active degree of freedom. The policy is initialized with a movement that results in hitting the cup from the side in the upper quarter of the cup. If the ball hits the cup below the middle, approximately 300 rollouts were required for

PoWER and we did not achieve reliable performance for the episodic Natural Actor Critic. We compare the two best performing algorithms from the basic motor learning examples (see Sect. 3.2) and the Underactuated Swing-Up (see Sect. 3.5), namely eNAC and PoWER. Figure 9(b) illustrates the results. PoWER again converges significantly faster.

3.7 Ball-in-a-Cup on a Barrett WAM™

The children's motor skill game Ball-in-a-Cup (Sumners 1997), also known as Balero, Bilboquet, and Kendama, is challenging even for adults. The toy has a small cup which is held in one hand (or, in our case, is attached to the end-effector of the robot) and the cup has a small ball hanging down on a string (the string has a length of 40 cm in our setup). Initially, the ball is hanging down vertically in a rest position. The player needs to move fast in order to induce a motion in the ball through the string, toss it up, and catch it with the cup. A possible movement is illustrated in Fig. 10 in the top row.

Note that learning of Ball-in-a-Cup and Kendama has previously been studied in robotics. We are going to contrast a few of the approaches here. While we learn directly in the joint space of the robot, Takenaka (1984) recorded planar human cup movements and determined the required joint movements for a planar, three degree of freedom (DoF) robot, so that it could follow the trajectories while visual feedback was used for error compensation. Both Sato et al. (1993) and Shone et al. (2000) used motion planning approaches which relied on very accurate models of the ball and the string while employing only one DoF in (Shone


(a) Schematic Drawings of the Ball-in-a-Cup Motion

(b) Kinesthetic Teach-In

(c) Final learned Robot Motion

Fig. 10 This figure shows schematic drawings of the Ball-in-a-Cup motion (**a**), the final learned robot motion (**c**), as well as a kinesthetic teach-in (**b**). The *arrows* show the directions of the current movements in that frame. The human cup motion was taught to the robot by imitation learning with 31 parameters per joint for an approximately 3 seconds long trajectory. The robot manages to reproduce the imitated motion quite accurately, but the ball misses the cup by several centimeters. After approximately 75 iterations of our Policy learning by Weighting Exploration with the Returns (PoWER) algorithm the robot has improved its motion so that the ball goes in the cup. Also see Fig. 12

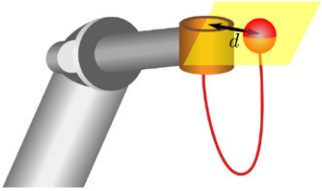


Fig. 11 This figure illustrates how the reward is calculated. The plane represents the level of the upper rim of the cup. For a successful rollout the ball has to be moved above the cup first and is then flying in a downward direction into the opening of the cup. The reward is calculated as the distance d of the center of the cup and the center of the ball on the plane at the moment the ball is passing the plane in a downward direction. If the ball is flying directly into the center of the cup, the distance is 0 and through the transformation $\exp(-d^2)$ yields the highest possible reward of 1. The further the ball passes the plane from the cup, the larger the distance and thus the smaller the resulting reward

et al. 2000) or two DoF in (Sato et al. 1993) so that the complete state-space could be searched exhaustively. Interestingly, exploratory robot moves were used in (Sato et al. 1993) to estimate the parameters of the employed model. Probably the most advanced preceding work on learning Kendama was carried out by Miyamoto et al. (1996) who used a seven DoF anthropomorphic arm and recorded human motions to train a neural network to reconstruct via-points. Employing full kinematic knowledge, the authors optimize a desired trajectory.

The state of the system can be described by joint angles and joint velocities of the robot as well as the Cartesian coordinates and velocities of the ball. The actions are the joint space accelerations where each of the seven joints is driven by a separate motor primitive, but with one common canonical system. The movement uses all seven degrees of freedom and is not in a plane. All motor primitives are perturbed separately but employ the same joint final reward given by

$$r(t) = \begin{cases} \exp(-\alpha(x_c - x_b)^2 - \alpha(y_c - y_b)^2) & \text{if } t = t_c, \\ 0 & \text{otherwise,} \end{cases}$$

where we denote the moment when the ball passes the rim of the cup with a downward direction by t_c, the cup position by $[x_c, y_c, z_c] \in \mathbb{R}^3$, the ball position by $[x_b, y_b, z_b] \in \mathbb{R}^3$, and a scaling parameter by $\alpha = 100$ (see also Fig. 11). The algorithm is robust to changes of this parameter as long as the reward clearly discriminates good and suboptimal trials. The directional information is necessary as the algorithm could otherwise learn to hit the bottom of the cup with the ball. This solution would correspond to a local maximum whose reward is very close to the optimal one, but the policy very far from the optimal one. The reward needs to include a term avoiding this local maximum. PoWER is based on the idea of considering the reward as an improper probability distribution. Transforming the reward using the exponential enforces this constraint. The reward is not only affected by the movements of the cup but foremost by the movements of the ball, which are sensitive to small changes in the cup's movement. A small perturbation of the initial condition or during the trajectory can change the movement of the ball and hence the outcome of the complete movement. The position of the ball is estimated using a stereo vision system and is needed to determine the reward.

Due to the complexity of the task, Ball-in-a-Cup is a hard motor learning task for children, who usually only succeed at it by observing another person playing combined with a lot of improvement by trial-and-error. Mimicking how children learn to play Ball-in-a-Cup, we first initialize the motor primitives by imitation learning and, subsequently, improve

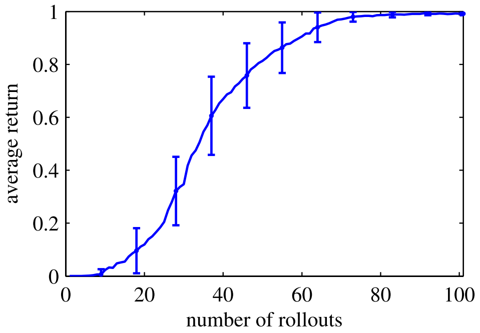


Fig. 12 This figure shows the expected return of the learned policy in the Ball-in-a-Cup evaluation averaged over 20 runs

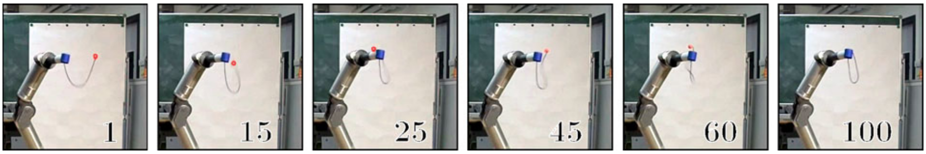


Fig. 13 This figure shows the improvement of the policy over rollouts. The snapshots from the video show the position of the ball closest to the cup during a rollout. (1) Initial policy after imitation learning. (15) Policy after 15 rollouts, already closer. (25) Policy after 25 rollouts, going too far. (45) Policy after 45 rollouts, hitting the near rim. (60) Policy after 60 rollouts, hitting the far rim. (100) Final policy after 100 rollouts

them by reinforcement learning. We recorded the motions of a human player by kinesthetic teach-in to obtain an example for imitation as shown in Fig. 10(b). A single demonstration was used for imitation learning. Learning from multiple demonstrations did not improve the performance as the task is sensitive to small differences. As expected, the robot fails to reproduce the presented behavior even if we use all the recorded details for the imitation. Thus, reinforcement learning is needed for self-improvement. The more parameters used for the learning, the slower the convergence is. Due to the imitation, the ball must go above the rim of the cup such that the algorithm gets at least a small positive reward for all rollouts. This way exhaustive exploration is avoided as the algorithm can compare the performance of the different rollouts. We determined that 31 shape-parameters per motor primitive are needed. With less parameters the ball does not go above the rim of the cup in the initial trial and the algorithm does not receive any meaningful information about the trial using the aforementioned reward function. More shape-parameters will lead to a more accurate reproduction of the demonstrated movement and, thus, to a better initial policy. However, there is a trade-off between this better initial policy and a potentially lower learning speed. Using three times as many parameters the algorithm converged at roughly the same time. The meta-parameters σ_{ij} are initially set in the same order of magnitude as the median of the parameters for each motor primitive and are then optimized alongside the shape-parameters by PoWER. The performance of the algorithm is fairly robust for values chosen in this range. Figure 12 shows the expected return over the number of rollouts where convergence to a maximum is clearly recognizable. The robot regularly succeeds at bringing the ball into the cup after approximately 75 iterations. Figure 13 shows the improvement of the policy over the rollouts. From our experience, nine year old children get the ball in the cup for the first time after about 35 trials while the robot gets the ball in for the first time after 42–45 rollouts. However, after 100 trials, the robot exhibits perfect runs in every single trial while children do not have a comparable success rate. Of course, such a comparison with children is contrived as a robot can precisely reproduce movements unlike any human being, and children can most likely adapt faster to changes in the setup.

4 Discussion and conclusion

In Sect. 4.1, we will discuss robotics as a benchmark for reinforcement learning, in Sect. 4.2 we discuss different simulation to robot transfer scenarios, and we will draw our conclusions in Sect. 4.3.

4.1 Discussion: robotics as benchmark for reinforcement learning?

Most reinforcement learning algorithms are evaluated on synthetic benchmarks, often involving discrete states and actions. Agents in many simple grid worlds take millions of actions and episodes before convergence. As a result, many methods have focused too strongly on such problem domains. In contrast, many real world problems such as robotics are best represented with high-dimensional, continuous states and actions. Every single trial run is costly and as a result such applications force us to focus on problems that will often be overlooked accidentally in synthetic examples. Simulations are a helpful testbed for debugging algorithms. Continuous states and actions as well as noise can be simulated, however simulations pose the temptation of using unrealistically many iterations and also allow us to exploit the perfect models.

Our typical approach consists of testing the algorithm in a simulation of the robot and the environment. Once the performance is satisfactory we replace either the robot or the environment with its real counterpart depending on the potential hazards. Replacing the robot is always possible as we can still simulate the environment taking into account the state of the real robot. Learning with a simulated robot in the real environment is not always a possibility especially if the robot influences the observed environment, such as in the Ball-in-a-Cup task. The final evaluations are done on the real robot in the real environment.

Our experience in robotics show that the plant and the environment can often not be represented accurately enough by a physical model and that the learned policies are thus not entirely transferable. If sufficiently accurate models were available, we could resort to the large body of work on optimal control (Kirk 1970), which offers alternatives to data driven reinforcement learning. However, when a model with large errors is used, the solution suffers severely from an optimization bias as already experienced by Atkeson (1994). Here, the reinforcement learning algorithm exploits the imperfections of the simulator rather than yielding an optimal policy.

None of our learned policies could be transferred from a simulator to the real system without changes despite that the immediate errors of the simulator have been smaller than the measurement noise. Methods which jointly learn the models and the policy as well as perform some of the evaluations and updates in simulation (such as Dyna-like architectures as in Sutton 1990) may alleviate this problem. In theory, a simulation could also be used to eliminate obviously bad solutions quickly. However, the differences between the simulation and the real robot do accumulate over time and this approach is only feasible if the horizon is short or the simulation very accurate. Priming the learning process by imitation learning and optimizing in its vicinity achieves the desired effect better and has thus been employed in this paper.

Parametrized policies greatly reduce the need of samples and evaluations. Choosing an appropriate representation like motor primitives renders the learning process even more efficient.

One major problem with robotics benchmarks is the repeatability of the experiments. The results are tied to the specific hardware and setup used. Even a comparison with simulated robots is often not possible as many groups rely on proprietary simulators that are not freely available. Most of the benchmarks presented in this paper rely on initial demonstrations to speed up the learning process. These demonstrations play an important part in the learning process. However, these initial demonstrations are also tied to the hardware or the simulator and are, thus, not straightforward to share. Comparing new algorithms to results from different authors usually requires the reimplementation of their algorithms to have a fair comparison.

Reproducibility is a key requirement for benchmarking but also a major challenge for robot reinforcement learning. To overcome this problem there are two orthogonal approaches: (i) a central organizer provides an identical setup for all participants and (ii) all participants share the same setup in a benchmarking lab. The first approach has been majorly pushed by funding agencies in the USA and Europe. In the USA, there have been special programs on robot learning such as DARPA Learning Locomotion (L2), Learning Applied to Ground Robotics (LAGR) and the DARPA Autonomous Robot Manipulation (ARM) (DARPA 2010a, 2010b, 2010c). However, the hurdles involved in getting these robots to work have limited the participation to strong robotics groups instead of opening the robot reinforcement learning domain to more machine learning researchers. Alternative ways of providing identical setups are low cost standardized hardware or a system composed purely of commercially available components. The first suffers from reproducibility and reliability issues while the latter results in significant system integration problems. Hence, it may be more suitable for a robot reinforcement learning challenge to be hosted by a robot learning group with significant experience in both domains. The host lab specifies tasks that they have been able to accomplish on a real robot system. The hosts also need to devise a virtual robot laboratory for allowing the challenge participants to program, test and debug their algorithms. To limit the workload and the risk of the organizers, a first benchmarking round would be conducted using this simulated setup to determine the promising approaches. Successful participants will be invited by the host lab in order to test these algorithms in learning on the real system where the host lab needs to provide significant guidance. To our knowledge, no benchmark based on this approach has been proposed yet. The authors of this paper are currently evaluating possibilities to organize a challenge using such a shared setup in the context of the PASCAL2 Challenge Program (PASCAL2 2010).

To successfully apply reinforcement learning to robotics, a fair level of knowledge on the limitations and maintenance of the robot hardware is necessary. These limitations include feasible actions, feasible run-time, as well as measurement delay and noise. Cooperation with a strong robotics group is still extremely important in order to apply novel reinforcement learning methods in robotics.

4.2 Discussion on simulation to robot transfer scenarios

In this paper, we have discussed reinforcement learning for real robots with highly dynamic tasks. The opposite extreme in robotics would be, for example, a maze navigation problem where a mobile robot that has macro-actions such as “go left” and the lower level control moves the robot exactly by a well-defined distance to the left. In this scenario, it would probably be easier to transfer simulation results to real systems. For highly dynamic tasks or environments, accurate simulations are generically difficult. Besides fast moving objects and many interacting objects, as well as deformable objects (often called soft bodies), like cloth, string, fluids, hydraulic tubes and other elastic materials are hard to simulate reliably and, thus, have an enormous impact on transferability. Additionally, the level and quality of measurement noise has a direct implication on the difficulty and the transferability of the learning task.

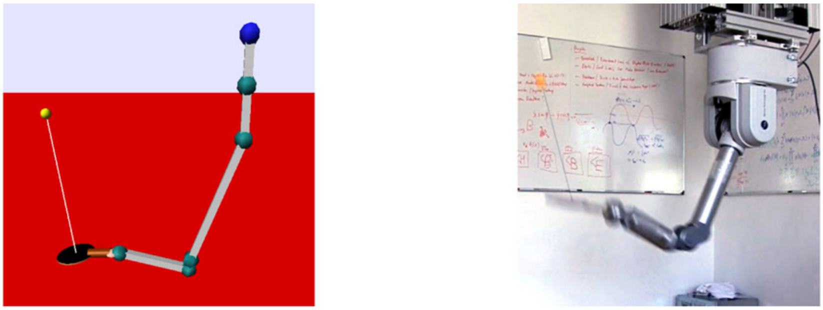
Better simulations often alleviate some of these problems. However, there is always a trade-off as more detailed simulations also require more precise model identification, higher temporal resolution, and, frequently even finite elements based simulations. Such detailed simulations may even be much slower than real-time, thus defeating one of the major advantages of simulations.

Aside from these clear difficulties in creating simulations that allow the transfer to real systems, we have observed three critically different scenarios for reinforcement learning in robotics. These scenarios are characterized by the energy flow between the policy and the system. In the energy-absorbing scenario, the task has passive dynamics and, hence, it is safer and easier to learn on a real robot. We are going to discuss the examples of Ball-Paddling, foothold selection in legged locomotion, and grasping (see Sect. 4.2.1). The second scenario has a border-line behavior: the system conserves most of the energy but the policy also only needs to inject energy into the system for a limited time. We will discuss Ball-in-a-Cup, Tetherball Target Hitting, and Mountain-Car as examples for this scenario (see Sect. 4.2.2). In the energy-emitting scenario energy is inserted due to the system dynamics even if the policy does not transfer energy into the system. The classical examples are Cart-Pole and inverted helicopters, and we also have the Underactuated Swing-Up which has to stabilize at the top (see Sect. 4.2.3). These different scenarios have implications on the relative utility of simulations and real robots.

As we are discussing our experience in performing such experiments, it may at times appear anecdotal. We hope the reader benefits from our insights nevertheless. However, the resulting classification bears similarities with insights on control law derivation (Fantoni and Lozano 2001).

4.2.1 Energy-absorbing scenario

In this scenario, the system absorbs energy from the actions. As shown in Fig. 14, we learned a Ball-Paddling task where a ping-pong ball is attached to a paddle by an elastic string and the ball has to be kept in the air by repeatedly hitting it from below. In this setup, the elastic



(a) Reinforcement learning required unrealistically many trials in simulation.

(b) Imitation learning only was sufficient on the real robot.

Fig. 14 This figure illustrates the Ball-Paddling task in simulation and on the real robot. The difference between simulation and robotics can be particularly emphasized in this problem where unrealistically many trials were needed on the simulation for reinforcement learning while the real world behavior could be learned by imitation learning. It illustrates the energy-consuming scenario and the difficulties of realistic learning in the presence of contact forces

string pulls the ball back towards the center of the paddle and the contact forces between the ball and the paddle are very complex. We modeled the system in as much detail as possible, including friction models, restitution models, dampening models, models for the elastic string, and air drag. However, in simulation the paddling behavior was still highly unpredictable and we needed a few thousand iterations to learn an optimal frequency, amplitude, and movement shape. The number of simulated trials exceeded the feasible amount on a real system. In contrast, when learning on the real system, we obtained a stable paddling behavior by imitation learning using the initial demonstration only and no further reinforcement learning was needed.

In general, scenarios with complex contact forces often work better in a real-world experiment. This problem was particularly drastic in locomotion experiments on rough terrain where the real world was an easier learning environment due to favorable friction properties during foot contact (Peters 2007). In this experiment, learning was significantly harder in simulation and the learned policy could not be transferred. The same effect occurs in grasping when objects often cannot be grasped in simulation due to slip but the real world friction still allows them to be picked up. Hence, in this scenario, policies from simulations are frequently not helpful and learning in simulation is harder than on the real system. The results only transfer in a few cases. A simulation is therefore only recommended as a feasibility study and for software debugging. As most contacts differ significantly due to the current properties (which vary with time and temperature) of the two interacting objects, only a learned simulator is likely to grasp all relevant details.

4.2.2 Border-line scenario

In this scenario, adding too much energy to a system does not necessarily lead to a negative outcome. For example, in the Mountain-Car problem (see Sect. 3.3), inserting more energy and driving through the goal at a higher velocity does not affect task achievement. In contrast not inserting enough energy will result in a failure as the car cannot reach the top of the mountain. The Tetherball Target Hitting application presented in Sect. 3.4 exhibits a very similar behavior. The Ball-in-a-Cup experiment (see Sect. 3.7) highlights the resulting similarities between learning in good simulations and the real world for this scenario. Success is possible if more energy is inserted and the ball flies higher. However, when using too little energy the ball will stay below the opening of the cup. In this favorable scenario the “classical” strategy can be applied: learn how to learn in simulation. The policy learned in simulation does not necessarily transfer to the real world and the real-world scenario can be highly different but the learning speed and behavior are similar. Hence, meta parameters such as learning and exploration rates can be tuned in simulation. The learning algorithm may take longer due to increased errors, modeling problems and uncertainty. Still, good practice is to create a sufficiently accurate simulator and to adapt the learning strategy subsequently to the real system.

4.2.3 Energy-emitting scenario

Energy emission causes very different problems. Uncertainty in states will cause overreactions, hence, drastic failures are likely to occur when the system becomes unstable in a control theory sense. This system excitability often makes the task significantly harder to learn on a real robot in comparison to a simulated setup. Here, pre-studies in simulations are a necessary but not sufficient condition. Due to unmodeled nonlinearities, the exploration will affect various algorithms differently. Classical examples are helicopters in inverted flight (Ng et al. 2004) and the pendulum in a Cart-Pole task in an upright position

(Sutton and Barto 1998) as these have to be constantly stabilized. Additionally the pendulum in the Swing-Up has to be stabilized in the final position or it will fall over and cause a failure. In this paper, we take the example of the Swing-Up to illustrate how some methods unexpectedly do better in the real world as exhibited by Figs. 7 and 8. The learning progress of all algorithms is noisier and the eRWR performs better on the real robot. The form of exploration employed by PoWER seems to give it an additional advantage in the first 20 rollouts as direct exploration on the actions is partially obscured by measurement noise. In order to cope with differences to the real-world, simulations need to be more stochastic than the real system (as suggested by Ng et al. 2004) and should be learned to make transferring the results easier (as for example in Schaal et al. 2002).

4.3 Conclusion

In this paper, we have presented a framework for deriving several policy learning methods that are applicable in robotics and an application to a highly complex motor learning task on a real Barrett WAM™ robot arm. We have shown that policy gradient methods are a special case of this framework. During initial experiments, we realized that the form of exploration highly influences the speed of the policy learning method. This empirical insight resulted in a novel policy learning algorithm, Policy learning by Weighting Exploration with the Returns (PoWER), an EM-inspired algorithm that outperforms several other policy search methods both on standard benchmarks as well as on a simulated Underactuated Swing-Up.

We have successfully applied this novel PoWER algorithm in the context of learning two tasks on a physical robot, namely the Underacted Swing-Up and Ball-in-a-Cup. Due to the curse of dimensionality, we cannot start with an arbitrary solution. Instead, we mimic the way children learn Ball-in-a-Cup and first present an example movement for imitation learning, which is recorded using kinesthetic teach-in. Subsequently, our reinforcement learning algorithm learns how to move the ball into the cup reliably. After only realistically few episodes, the task can be regularly fulfilled and the robot shows very good average performance. After 100 rollouts, the meta parameters, such as the exploration rate, have converged to negligible size and do not influence the outcome of the behavior any further. The experiments in this paper use the original variant of the motor primitives which cannot deal with large perturbations. However, the extended variable-feedback variant presented in (Kober et al. 2008) can deal with a variety of changes directly (for example, in the length of the string or the size or weight of the ball) while the approach presented in this paper will recover quickly by learning an adjusted policy in a few roll-outs. In (Kober et al. 2008), we have also shown that learning a strategy of pulling the ball up and moving the cup under the ball (as in Kendama) is possible in approximately the same number of trials. We have discovered a variety of different human strategies for Ball-in-a-Cup in movement recordings, see (Chiappa et al. 2009).

A preliminary version of some of the work in this paper was shown in (Kober et al. 2008; Kober and Peters 2009a, 2009b). It did not yet include the real robot comparisons of the algorithms on the Underactuated Swing-Up benchmark as well as the Tetherball Target Hitting and the Casting benchmarks. We also discuss our experience with transfer from simulation to real robot systems. Our approach has already given rise to follow-up work in other contexts, for example, (Vlassis et al. 2009; Kormushev et al. 2010). Theodorou et al. (2010) have shown that an algorithm very similar to PoWER can also be derived from a completely different perspective, that is, the path integral approach.

Acknowledgements We thank the anonymous reviewers for their valuable suggestions that helped us to significantly extend and improve the discussions in Sect. 4.

Appendix A: Derivations

In this appendix, we provide the derivations of various algorithms in more details than in the main text. We first present, how the episodic REINFORCE (Williams 1992) can be obtained (Sect. A.1). Subsequently, we show how the episodic Reward Weighted Regression (eRWR) (Peters and Schaal 2007) can be generalized for the episodic case (Sect. A.2), and finally we derive EM Policy learning by Weighting Exploration with the Returns (PoWER) and show a number of simplifications (Sect. A.3).

A.1 Reinforce

If we choose a stochastic policy in the form $a = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{s}, t) + \varepsilon_t$ with $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$, we have

$$\pi\left(a_t|\mathbf{s}_t, t\right) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}\left(a - \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi}\right)^2\right),$$

and, thus,

$$\partial_{\boldsymbol{\theta}} \log \pi = \sigma^{-2}\left(a - \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi}\right)\boldsymbol{\phi}^{\mathrm{T}}.$$

Therefore, the gradient, in (2), becomes

$$\partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = E\left\{\sum_{t=1}^{T} \sigma^{-2}\left(a - \boldsymbol{\theta}'^{\mathrm{T}}\boldsymbol{\phi}\right)\boldsymbol{\phi}^{\mathrm{T}} R\right\} = E\left\{\sum_{t=1}^{T} \sigma^{-2}\varepsilon_t \boldsymbol{\phi}^{\mathrm{T}} R\right\}, \tag{10}$$

which corresponds to the episodic REINFORCE algorithm (Williams 1992).

A.2 Episodic reward weighted regression (eRWR)

Setting (10) to zero

$$\partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = E\left\{\sum_{t=1}^{T} \sigma^{-2}\left(a - \boldsymbol{\theta}'^{\mathrm{T}}\boldsymbol{\phi}\right)\boldsymbol{\phi}^{\mathrm{T}} R\right\} = 0,$$

we obtain

$$E\left\{\sum_{t=1}^{T} \sigma^{-2} a R \boldsymbol{\phi}^{\mathrm{T}}\right\} = E\left\{\sum_{t=1}^{T} \sigma^{-2}\left(\boldsymbol{\theta}'^{\mathrm{T}}\boldsymbol{\phi}\right) R \boldsymbol{\phi}^{\mathrm{T}}\right\}.$$

Since σ is constant, we have $E\{\sum_{t=1}^{T} a R\boldsymbol{\phi}^{\mathrm{T}}\} = \boldsymbol{\theta}'^{\mathrm{T}} E\{\sum_{t=1}^{T} \boldsymbol{\phi} R \boldsymbol{\phi}^{\mathrm{T}}\}$. The $\boldsymbol{\theta}'$ minimizing the least squares error can be found by locally weighted linear regression (R as weights and $\boldsymbol{\phi}$ as basis functions) considering each time-step and rollout separately

$$\boldsymbol{\theta}' = \left(\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{R}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{RA},$$

with $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1^1, \dots, \boldsymbol{\phi}_T^1, \boldsymbol{\phi}_1^2, \dots, \boldsymbol{\phi}_1^H, \dots, \boldsymbol{\phi}_T^H]^{\mathrm{T}}$, $\mathbf{R} = \mathrm{diag}(R^1, \dots, R^1, R^2, \dots, R^H, \dots, R^H)$, and $\mathbf{A} = [a_1^1, \dots, a_T^1, a_1^2, \dots, a_1^H, \dots, a_T^H]^{\mathrm{T}}$ for H rollouts.

The same derivation holds if we use (3) instead of (2). Then $\mathbf{R}$ in the regression is replaced by $\mathbf{Q}^\pi = \mathrm{diag}(Q_1^{\pi,1}, \dots, Q_T^{\pi,1}, Q_1^{\pi,2}, \dots, Q_1^{\pi,H}, \dots, Q_T^{\pi,H})$. Using the state-action value function Q^π yields slightly faster convergence than using the return R.

A.3 EM policy learning by weighting exploration with the returns (PoWER)

If we choose a stochastic policy in the form $a = (\boldsymbol{\theta} + \boldsymbol{\varepsilon}_t)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{s}, t)$ with $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0}, \hat{\boldsymbol{\Sigma}})$, we have

$$\pi\,(a_t|\mathbf{s}_t, t) = \mathcal{N}\left(\mathbf{a}|\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi}\,(\mathbf{s}, t)\,, \boldsymbol{\phi}(\mathbf{s}, t)^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}\boldsymbol{\phi}(\mathbf{s}, t)\right) = \left(2\pi\boldsymbol{\phi}^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}\boldsymbol{\phi}\right)^{-1/2}\exp\left(\frac{-(a-\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi})^2}{2\boldsymbol{\phi}^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}\boldsymbol{\phi}}\right),$$

and, thus, $\partial_{\boldsymbol{\theta}} \log \pi = (a - \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi})\boldsymbol{\phi}^{\mathrm{T}}/(\boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{\phi})^2$. Therefore (3) becomes

$$\partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = E\left\{\sum_{t=1}^{T} \frac{(a - \boldsymbol{\theta}'^{\mathrm{T}}\boldsymbol{\phi})\boldsymbol{\phi}^{\mathrm{T}}}{\boldsymbol{\phi}^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}\boldsymbol{\phi}} Q^{\pi}\right\}.$$

Setting this equation to zero is equivalent to

$$E\left\{\sum_{t=1}^{T} \frac{a\boldsymbol{\phi}^{\mathrm{T}}}{\boldsymbol{\phi}^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}\boldsymbol{\phi}} Q^{\pi}\right\} \equiv E\left\{\sum_{t=1}^{T} \frac{((\boldsymbol{\theta} + \boldsymbol{\varepsilon}_t)^{\mathrm{T}}\boldsymbol{\phi})\boldsymbol{\phi}^{\mathrm{T}}}{\boldsymbol{\phi}^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}\boldsymbol{\phi}} Q^{\pi}\right\} = E\left\{\sum_{t=1}^{T} \frac{(\boldsymbol{\theta}'^{\mathrm{T}}\boldsymbol{\phi})\boldsymbol{\phi}^{\mathrm{T}}}{\boldsymbol{\phi}^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}\boldsymbol{\phi}} Q^{\pi}\right\}.$$

This equation yields

$$\begin{aligned}\boldsymbol{\theta}'^{\mathrm{T}} &= E\left\{\sum_{t=1}^{T} \frac{((\boldsymbol{\theta} + \boldsymbol{\varepsilon}_t)^{\mathrm{T}}\boldsymbol{\phi})\boldsymbol{\phi}^{\mathrm{T}}}{\boldsymbol{\phi}^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}\boldsymbol{\phi}} Q^{\pi}\right\} E\left\{\sum_{t=1}^{T} \frac{\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}}{\boldsymbol{\phi}^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}\boldsymbol{\phi}} Q^{\pi}\right\}^{-1} \\ &= \boldsymbol{\theta}^{\mathrm{T}} + E\left\{\sum_{t=1}^{T} \frac{\boldsymbol{\varepsilon}_t^{\mathrm{T}}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}}{\boldsymbol{\phi}^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}\boldsymbol{\phi}} Q^{\pi}\right\} E\left\{\sum_{t=1}^{T} \frac{\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}}{\boldsymbol{\phi}^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}\boldsymbol{\phi}} Q^{\pi}\right\}^{-1}\end{aligned}$$

and finally with $\mathbf{W} = \boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{\phi}^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}\boldsymbol{\phi})^{-1}$ we get

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + E\left\{\sum_{t=1}^{T} \mathbf{W}Q^{\pi}\right\}^{-1} E\left\{\sum_{t=1}^{T} \mathbf{W}\boldsymbol{\varepsilon}_t Q^{\pi}\right\}. \tag{11}$$

If $\hat{\boldsymbol{\Sigma}}$ is diagonal, that is, the exploration of the parameters is pairwise independent, all parameters employ the same exploration, and the exploration is constant over rollouts, $\mathbf{W}$ simplifies to $\mathbf{W} = \boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{\phi}^{\mathrm{T}}\boldsymbol{\phi})^{-1}$. Normalized basis functions $\boldsymbol{\phi}$ further simplify $\mathbf{W}$ to $\mathbf{W}(\mathbf{s}, t) = \boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}$.

If only one parameter is active at each time step, $\mathbf{W}(\mathbf{s}, t)$ is diagonal and (11) simplifies to

$$\theta_i' = \theta_i + \frac{E\{\sum_{t=1}^{T} \phi_i^2/(\boldsymbol{\phi}^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}\boldsymbol{\phi})\varepsilon_{i,t} Q^{\pi}\}}{E\{\sum_{t=1}^{T} \phi_i^2/(\boldsymbol{\phi}^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}\boldsymbol{\phi}) Q^{\pi}\}} \tag{12}$$

$$= \theta_i + \frac{E\{\sum_{t=1}^{T} \sigma_i^{-1}\varepsilon_{i,t} Q^{\pi}\}}{E\{\sum_{t=1}^{T} \sigma_i^{-1} Q^{\pi}\}}, \tag{13}$$

where θ_i' is one individual parameter, ϕ_i and $\varepsilon_{i,t}$ are the corresponding elements of $\boldsymbol{\phi}$ and $\boldsymbol{\varepsilon}_t$, and σ_i is the respective entry of the diagonal of $\hat{\boldsymbol{\Sigma}}$. If the σ_i are constant over the rollouts we get $\theta_i' = \theta_i + E\{\sum_{t=1}^{T} \varepsilon_{i,t} Q^{\pi}\}/E\{\sum_{t=1}^{T} Q^{\pi}\}$. The independence simplification in (12), (13) works well in practice, even if there is some overlap between the activations, such as in the case of dynamical system motor primitives (Ijspeert et al. 2002, 2003; Schaal et al. 2003, 2007). Weighting the exploration with the basis functions, as in (12), yields slightly better results than completely ignoring the interactions, as in (13).

The policy can be equivalently expressed as

$$\pi\left(a_t|\mathbf{s}_t,t\right)=p\left(a_t|\mathbf{s}_t,t,\boldsymbol{\varepsilon}_t\right)p\left(\boldsymbol{\varepsilon}_t|\mathbf{s}_t,t\right)=p\left(a_t|\mathbf{s}_t,t,\boldsymbol{\varepsilon}_t\right)\mathcal{N}\left(\boldsymbol{\varepsilon}_t|\mathbf{0},\hat{\boldsymbol{\Sigma}}\right).$$

Applying (3) to the variance $\hat{\boldsymbol{\Sigma}}$ we get

$$\partial_{\hat{\boldsymbol{\Sigma}}'}L_{\hat{\boldsymbol{\Sigma}}}\left(\hat{\boldsymbol{\Sigma}}'\right)=E\left\{\sum_{t=1}^{T}\partial_{\hat{\boldsymbol{\Sigma}}'}\log\mathcal{N}\left(\boldsymbol{\varepsilon}_t|\mathbf{0},\hat{\boldsymbol{\Sigma}}'\right)Q^{\pi}\right\},$$

as $p(a_t|\mathbf{s}_t,t,\boldsymbol{\varepsilon}_t)$ is independent from the variance. Setting this equation to zero and solving for $\hat{\boldsymbol{\Sigma}}'$ yields

$$\hat{\boldsymbol{\Sigma}}'=\frac{E\{\sum_{t=1}^{T}\boldsymbol{\varepsilon}_t\boldsymbol{\varepsilon}_t^{\mathrm{T}}Q^{\pi}\}}{E\{\sum_{t=1}^{T}Q^{\pi}\}},$$

which is the same solution as we get in standard maximum likelihood problems.

The same derivation holds if we use (2) instead of (3). Then, the state-action value function Q^{π} is replaced everywhere by the return R.

References

Andrieu, C., de Freitas, N., Doucet, A., & Jordan, M. I. (2003). An introduction to MCMC for machine learning. *Machine Learning*, *50*(1), 5–43.

Atkeson, C. G. (1994). Using local trajectory optimizers to speed up global optimization in dynamic programming. In *Advances in neural information processing systems* (Vol. 6, pp. 503–521), Denver, CO, USA.

Attias, H. (2003). Planning by probabilistic inference. In *Proceedings of the ninth international workshop on artificial intelligence and statistics (AISTATS)*, Key West, FL, USA.

Bagnell, J., & Schneider, J. (2003). Covariant policy search. In *Proceedings of the international joint conference on artificial intelligence (IJCAI)* (pp. 1019–1024), Acapulco, Mexico.

Bagnell, J., Kadade, S., Ng, A., & Schneider, J. (2004). Policy search by dynamic programming. In *Advances in neural information processing systems* (Vol. 16), Vancouver, BC, CA.

Binder, J., Koller, D., Russell, S., & Kanazawa, K. (1997). Adaptive probabilistic networks with hidden variables. *Machine Learning*, *29*(2–3), 213–244.

Chiappa, S., Kober, J., & Peters, J. (2009). Using Bayesian dynamical systems for motion template libraries. In D. Koller, D. Schuurmans, Y. Bengio, & L. Bottou (Eds.), *Advances in neural information processing systems* (Vol. 21, pp. 297–304).

DARPA (2010a). Learning locomotion (L2). http://www.darpa.mil/ipto/programs/ll/ll.asp.

DARPA (2010b). Learning applied to ground robotics (LAGR). http://www.darpa.mil/ipto/programs/lagr/lagr.asp.

DARPA (2010c). Autonomous robot manipulation (ARM). http://www.darpa.mil/ipto/programs/arm/arm.asp.

Dayan, P., & Hinton, G. E. (1997). Using expectation-maximization for reinforcement learning. *Neural Computation*, *9*(2), 271–278.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, *39*, 1–38.

El-Fakdi, A., Carreras, M., & Ridao, P. (2006). Towards direct policy search reinforcement learning for robot control. In *Proceedings of the IEEE/RSJ 2006 international conference on intelligent robots and systems (IROS)*, Beijing, China.

Fantoni, I., & Lozano, R. (2001). *Non-linear control for underactuated mechanical systems*. New York: Springer.

Guenter, F., Hersch, M., Calinon, S., & Billard, A. (2007). Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics, Special Issue on Imitative Robots*, *21*(13), 1521–1544.

Gullapalli, V., Franklin, J., & Benbrahim, H. (1994). Acquiring robot skills via reinforcement learning. *IEEE Control Systems Journal, Special Issue on Robotics: Capturing Natural Motion*, *4*(1), 13–24.

Hoffman, M., Doucet, A., de Freitas, N., & Jasra, A. (2007). Bayesian policy learning with trans-dimensional MCMC. In *Advances in neural information processing systems* (Vol. 20), Vancouver, BC, CA.

Ijspeert, A. J., Nakanishi, J., & Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings of IEEE international conference on robotics and automation (ICRA)* (pp. 1398–1403), Washington, DC.

Ijspeert, A. J., Nakanishi, J., & Schaal, S. (2003). Learning attractor landscapes for learning motor primitives. In *Advances in neural information processing systems* (Vol. 15, pp. 1547–1554), Vancouver, BC, CA.

Jaakkola, T., Jordan, M. I., & Singh, S. P. (1994). Convergence of stochastic iterative dynamic programming algorithms. In J. D. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in neural information processing systems* (Vol. 6, pp. 703–710). San Mateo: Morgan Kaufmann.

Kirk, D. E. (1970). *Optimal control theory*. Englewood Cliffs: Prentice-Hall.

Kober, J., & Peters, J. (2009a). Learning motor primitives for robotics. In *Proceedings of IEEE international conference on robotics and automation (ICRA)* (pp. 2112–2118).

Kober, J., & Peters, J. (2009b). Policy search for motor primitives in robotics. In D. Koller, D. Schuurmans, Y. Bengio, & L. Bottou (Eds.), *Advances in neural information processing systems* (Vol. 21, pp. 849–856).

Kober, J., Mohler, B., & Peters, J. (2008). Learning perceptual coupling for motor primitives. In *Proceedings of the IEEE/RSJ 2008 international conference on intelligent robots and systems (IROS)* (pp. 834–839), Nice, France.

Kormushev, P., Calinon, S., & Caldwell, D. G. (2010). Robot motor skill coordination with em-based reinforcement learning. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Kwee, I., Hutter, M., & Schmidhuber, J. (2001). Gradient-based reinforcement planning in policy-search methods. In M. A. Wiering (Ed.), *Cognitieve Kunstmatige Intelligentie: Vol. 27. Proceedings of the 5th European workshop on reinforcement learning (EWRL)* (pp. 27–29), Lugano. Manno: Onderwijsinsituut CKI, Utrecht University.

Lawrence, G., Cowan, N., & Russell, S. (2003). Efficient gradient estimation for motor control learning. In *Proceedings of the international conference on uncertainty in artificial intelligence (UAI)* (pp. 354–361), Acapulco, Mexico.

Martín, H. J. A., de Lope, J., & Maravall, D. (2009). The knn-td reinforcement learning algorithm. In *Proceedings of the 3rd international work-conference on the interplay between natural and artificial computation (IWINAC)* (pp. 305–314). Berlin: Springer.

McLachan, G. J., & Krishnan, T. (1997). *Wiley series in probability and statistics*. *The EM algorithm and extensions*. New York: Wiley.

Miyamoto, H., Schaal, S., Gandolfo, F., Gomi, H., Koike, Y., Osu, R., Nakano, E., Wada, Y., & Kawato, M. (1996). A Kendama learning robot based on bi-directional theory. *Neural Networks*, *9*(8), 1281–1302.

Ng, A. Y., & Jordan, M. (2000). Pegasus: A policy search method for large mdps and pomdps. In *Proceedings of the international conference on uncertainty in artificial intelligence (UAI)* (pp. 406–415), Palo Alto, CA.

Ng, A. Y., Kim, H. J., Jordan, M. I., & Sastry, S. (2004). Inverted autonomous helicopter flight via reinforcement learning. In *Proceedings of the international symposium on experimental robotics (ISER)*. Cambridge: MIT Press.

Park, D. H., Hoffmann, H., Pastor, P., & Schaal, S. (2008). Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In *IEEE international conference on humanoid robots (HUMANOIDS)* (pp. 91–98).

PASCAL2 (2010). Challenges. http://pascallin2.ecs.soton.ac.uk/Challenges/.

Peshkin, L. (2001). *Reinforcement learning by policy search*. PhD thesis, Brown University, Providence, RI.

Peters, J. (2007). *Machine learning of motor skills for robotics*. PhD thesis, University of Southern California, Los Angeles, CA, 90089, USA.

Peters, J., & Schaal, S. (2006). Policy gradient methods for robotics. In *Proceedings of the IEEE/RSJ 2006 international conference on intelligent robots and systems (IROS)* (pp. 2219–2225), Beijing, China.

Peters, J., & Schaal, S. (2007). Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the international conference on machine learning (ICML)*, Corvallis, OR, USA.

Peters, J., Vijayakumar, S., & Schaal, S. (2003). Reinforcement learning for humanoid robotics. In *Proceedings of the IEEE-RAS international conference on humanoid robots (HUMANOIDS)* (pp. 103–123), Karlsruhe, Germany.

Peters, J., Vijayakumar, S., & Schaal, S. (2005). Natural actor-critic. In *Proceedings of the European conference on machine learning (ECML)* (pp. 280–291), Porto, Portugal.

Rückstieß, T., Felder, M., & Schmidhuber, J. (2008). State-dependent exploration for policy gradient methods. In *Proceedings of the European conference on machine learning (ECML)* (pp. 234–249), Antwerp, Belgium.

Sato, S., Sakaguchi, T., Masutani, Y., & Miyazaki, F. (1993). Mastering of a task with interaction between a robot and its environment: "kendama" task. *Transactions of the Japan Society of Mechanical Engineers C*, *59*(558), 487–493.

Schaal, S., Atkeson, C. G., & Vijayakumar, S. (2002). Scalable techniques from nonparameteric statistics for real-time robot learning. *Applied Intelligence*, *17*(1), 49–60.

Schaal, S., Peters, J., Nakanishi, J., & Ijspeert, A. J. (2003). Control, planning, learning, and imitation with dynamic movement primitives. In *Proceedings of the workshop on bilateral paradigms on humans and humanoids, IEEE international conference on intelligent robots and systems (IROS)*, Las Vegas, NV, October 27–31, 2003.

Schaal, S., Mohajerian, P., & Ijspeert, A. J. (2007). Dynamics systems vs. optimal control—a unifying view. *Progress in Brain Research*, *165*(1), 425–445.

Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., & Schmidhuber, J. (2010). Parameter-exploring policy gradients. *Neural Networks*, *21*(4), 551–559.

Shone, T., Krudysz, G., & Brown, K. (2000). *Dynamic manipulation of Kendama* (Tech. rep.). Rensselaer Polytechnic Institute.

Strens, M., & Moore, A. (2001). Direct policy search using paired statistical tests. In *Proceedings of the 18th international conference on machine learning (ICML)*.

Sumners, C. (1997). *Toys in space: exploring science with the astronauts*. New York: McGraw-Hill.

Sutton, R., & Barto, A. (1998). *Reinforcement learning*. Cambridge: MIT Press.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the international machine learning conference* (pp. 9–44).

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems (NIPS)* (Vol. 13, pp. 1057–1063), Denver, CO, USA.

Takenaka, K. (1984). Dynamical control of manipulator with vision: "cup and ball" game demonstrated by robot. *Transactions of the Japan Society of Mechanical Engineers C*, *50*(458), 2046–2053.

Taylor, M. E., Whiteson, S., & Stone, P. (2007). Transfer via inter-task mappings in policy search reinforcement learning. In *Proceedings of the sixth international joint conference on autonomous agents and multiagent systems (AAMAS)*.

Tedrake, R., Zhang, T. W., & Seung, H. S. (2004). Stochastic policy gradient reinforcement learning on a simple 3d biped. In *Proceedings of the IEEE 2004 international conference on intelligent robots and systems (IROS)* (pp. 2849–2854).

Theodorou, E. A., Buchli, J., & Schaal, S. (2010). Reinforcement learning of motor skills in high dimensions: a path integral approach. In *Proceedings of IEEE international conference on robotics and automation (ICRA)* (pp. 2397–2403).

Toussaint, M., & Goerick, C. (2007). Probabilistic inference for structured planning in robotics. In *Proceedings of the IEEE/RSJ 2007 international conference on intelligent robots and systems (IROS)*, San Diego, CA, USA.

Van Der Maaten, L., Postma, E., & Van Den Herik, H. (2007). *Dimensionality reduction: a comparative review*. Preprint.

Vlassis, N., Toussaint, M., Kontes, G., & Piperidis, S. (2009). Learning model-free robot control by a Monte Carlo EM algorithm. *Autonomous Robots*, *27*(2), 123–130.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*, 229–256.

Wulf, G. (2007). *Attention and motor skill learning*. Champaign: Human Kinetics.