Real-time reinforcement learning by sequential Actor-Critics and experience replay

Real-Time Reinforcement Learning by Sequential Actor-Critics and Experience Replay

Paweł Wawrzyński

Warsaw University of Technology, Institute of Control and Computation Engineering, Poland

Abstract

Actor-Critics constitute an important class of reinforcement learning algorithms that can deal with continuous actions and states in an easy and natural way. This paper shows how these algorithms can be augmented by the technique of experience replay without degrading their convergence properties, by appropriately estimating the policy change direction. This is achieved by truncated importance sampling applied to the recorded past experiences. It is formally shown that the resulting estimation bias is bounded and asymptotically vanishes, which allows the experience replay-augmented algorithm to preserve the convergence properties of the original algorithm. The technique of experience replay makes it possible to utilize the available computational power to reduce the required number of interactions with the environment considerably, which is essential for real-world applications. Experimental results are presented that demonstrate that the combination of experience replay and Actor-Critics yields extremely fast learning algorithms that achieve successful policies for nontrivial control tasks in considerably short time. Namely, the policies for the cart-pole swing-up (Doya, 2000) are obtained after as little as 20 minutes of the cart-pole time and the policy for Half-Cheetah (a walking 6-degree-of-freedom robot) is obtained after four hours of Half-Cheetah time.

Key words: direct adaptive control, machine learning, reinforcement learining, experience replay

1. Introduction

Reinforcement learning (RL) addresses the problem of an agent that optimizes its reactive policy in a poorly structured and initially unknown environment (Sutton & Barto, 1998). Algorithms developed in this area can be viewed as computational processes that transform observations of states, actions and rewards into policy parameters. Several important RL algorithms, such as Q-Learning (Watkins & Dayan, 1992) and Actor-Critic methods (Barto et al., 1983; Kimura & Kobayashi, 1998; Konda & Tsitsiklis, 2003; Bhatnagar et al., 2008a), process the data sequentially. Each single observation is used for adjusting the algorithms' parameters and then becomes unavailable for further use. We shall call such methods *sequential*. They are based on a common assumption that RL applications to real-world learning control problems require large amounts of data which cannot be kept in a limited amount of memory assigned to the algorithm.

Sequential algorithms do not exploit all the information contained in the data and are known to require a large number of environment steps to obtain a satisfying policy. Usually, this number is large enough to make the learning process detrimental to any real device whose control policy we would hope to optimize by means of RL. However, there are other, *non-sequential* methods that require much fewer environment steps to obtain a policy of the same quality. They achieve this at the cost of collecting data and some extensive processing thereof. This distinction between sequential and non-sequential algorithms should not be confused with the distinction between online and offline algorithms. Here, we are interested *only* in online algorithms which improve the policy as the agent-environment interaction proceeds. Online sequential and non-sequential algorithms differ in the way of using the available computational power during the interaction.

The most obvious way to design the computations for a non-sequential algorithm is to estimate the model of the environment and use it in some variant of dynamic programming. This idea appears to have been applied for the first time in the DYNA architecture (Sutton, 1990). Further research in this area has brought the family of E^3 algorithms (Abbeel & Ng, 2005) applicable not only to finite-space problems but also to those with linear models. Recent work extends this methodology to problems without regular structure (Abbeel et al., 2006).

Another possible approach to non-sequential algorithms can be summarized as follows. Assume there exists a quality index $J : \Re^{n_\theta} \mapsto \Re$ that assigns to each policy parameter $\theta \in \Re^{n_\theta}$ a value $J(\theta)$ that evaluates the policy based on θ. Let us denote by $\widehat{J}_t(\theta)$ an estimator of $J(\theta)$ based on the course of the MDP before time t. The objective of data processing can be to optimize the policy by maximizing $\widehat{J}_t(\theta)$ with respect to θ. Shelton (2001) and Peshkin & Shelton (2002) present methods of policy optimization in Partially Observable Markov Decision Processes that are based on the history of agent-environment interactions and importance sampling estimators. Properties of estimators and bounds for the uniform convergence of estimates are discussed by Peshkin & Mukherjee (2001) and a way is suggested of using these bounds to select a policy class among candidate classes of varying complexity, similar to structural risk minimization.

In the third approach, a non-sequential algorithm is constructed by adding some non-sequential data processing to a sequential algorithm. It collects historical experiences (observations of states, actions and rewards) and applies to them the operations of the original sequential algorithm as if they have just taken place. This idea of repeating many similar operations to the same event, called *experience replay* (Mahadevan & Connell, 1992; Lin, 1992; Lin & Mitchell, 1992; Cichosz, 1999), was popular a few years ago but has received little attention recently. Unfortunately, experience replay is not automatically applicable to an arbitrary RL algorithm. In particular, it cannot be directly combined with on-policy methods, i.e., those based on the assumption that the actions producing data for policy improvements are drawn from the current policy. Consequently, the same experience cannot be applied many times to adjust a continuously changing policy.

The issue of policy optimization on the basis of past experience can be reduced to the problem of estimation of the expected value of a random vector with one distribution using samples from different distribution(s). That can be done with the use of the importance sampling (IS) estimators (Rubinstein, 1981). Various types of IS estimators have been applied in RL several times. Shelton (2001) and Peshkin & Shelton (2002) used them to define an estimator of a quality index of a given policy; Precup et al. (2001) and Frank et al. (2008) applied IS estimators to determine a value function approximation of a given policy on the basis of realization of a decision process driven by another policy; Uchibe & Doya (2004) applied a certain truncated IS estimator to optimize several policies of various complexities on the basis of a single decision process realization.

1.1. Contribution

In this paper we show how to apply experience replay to a family of RL algorithms called Actor-Critics (Barto et al., 1983; Kimura & Kobayashi, 1998; Konda & Tsitsiklis, 2003). Those are on-policy algorithms that have proved successful in a number of tasks, particularly with continuous actions which cannot be directly dealt with by the more popular Q-learning and related algorithms. The combination of Actor-Critics and experience replay has been attempted before with limited success (Lin, 1992), probably due to the above-mentioned difficulty resulting from the discrepancy between the current policy and the policy that generated a past experience. To overcome this difficulty, a novel variant of the experience replay technique is proposed in which the policy adjustments based on the data coming from a different policy are reconstructed to compensate for the discrepancy. The contribution of this paper may be summarized in the following points:

- It is shown here how to apply experience replay to any sequential Actor-Critic algorithm in such a way that (i) convergence properties and robustness of the algorithm are preserved and (ii) the speed of learning of the algorithm measured in terms of length of agent-environment interaction is increased, at the cost of additional background computations.

- The methodology presented here is based on improvement direction estimators computed with the use of data on past experience collected in a database. The randomized–truncated importance sampling estimators are introduced here for that purpose. Their properties are proved: bounded variance and asymptotic unbiasness.

- The experimental study is presented that demonstrates application of the introduced methodology to challenging control problems, namely the cart-pole swing-up task (Doya, 2000) and Half-Cheetah (Wawrzyński, 2007).

The idea presented in this paper is partially inspired by the significant evidence in neuropsychology suggesting that human brain does not learn "sequentially". Instead, it stores extracts of percepts in the short-time memory and uses them in a certain process that updates synaptic weights. Those results suggest that RL algorithms should be "non-sequential", or "batch", to be efficient. The design we propose supports this intuition, which appears to be somewhat neglected in the RL literature.

The paper is organized as follows. In Section 2 the problem of our interest is defined along with the class of algorithms that encompasses sequential Actor-Critics. It is shown that a given Actor-Critic method defines certain improvement directions of its parameter vectors. Section 3 shows how to estimate these directions using the data from the preceding state transition and to accelerate a sequential algorithm by combining these estimators with experience replay. Conditions for asymptotic unbiasedness of these estimators are established in Section 4 that enable the algorithm with experience replay to inherit the limit properties of the original sequential method. The experimental study on the introduced methodology is presented in Section 5 and the last Section concludes.

2. Problem Formulation

We will consider the standard RL setup (Sutton & Barto, 1998). A Markov Decision Process (MDP) defines a problem of an agent that observes its state s_t in discrete time $t = 1, 2, 3, \dots$, performs actions a_t, receives rewards r_t and moves to other states s_{t+1}. A particular MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, P_s, r \rangle$ where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, respectively; $\{P_s(\cdot|s,a) : s \in \mathcal{S}, a \in \mathcal{A}\}$ is a set of state transition distributions; we write $s_{t+1} \sim P_s(\cdot|s_t, a_t)$ and assume that each P_s is a density. Each state transition generates a *reward*, $r_t \in \Re$. Here we assume that each reward is depends deterministically on the current action and the next state, $r_t = r(a_t, s_{t+1})$.

Actions are generated according to a policy, π, which is a family of distributions parameterized by the state and a *policy vector* $\theta \in \Re^{n_\theta}$, namely $a_t \sim \pi(\cdot\,; s_t, \theta)$. The objective of reinforcement learning is, in general, to make the policy maximize future rewards by optimizing θ. The strict meaning of this goal may be specified in various ways. We may require the policy to maximize the average reward or to maximize the sum of future discounted rewards expected in each state.

We are interested in applications of the MDP framework to learning control tasks. We thus require the learning algorithm to be fast, i.e., to obtain a satisfying policy after as few agent steps as possible. The control efficiency should be maximized within short period of learning to keep the controlled machine from being damaged by too many wrong actions. We assume availability of a limited database that keeps a recent part of the agent-environment interaction history. This history is available between consecutive agent steps for a limited number of operations aiming at policy optimization.

Below we analyze a large class of the existing sequential RL methods suitable for policy determination in simulations and propose a way of their acceleration based on a more extensive data processing. Our intention is to design methods that obtain a satisfying policy after a much smaller amount of agent time but not necessarily after a smaller amount of computation.

2.1. Sequential Actor-Critic Algorithms

Actor-Critics constitute probably the most efficient and the most theoretically developed class of reinforcement learning algorithms. The actor-critic structure alone appears to have been presented for the first time by Barto et al. (1983). An important development was made by Gullapalli (1990), who introduced the actor adjustment scheme based on expectation gradient estimators computed with the use of likelihood ratios. These estimators were invented in the generic form earlier by Williams (1989). Efficient actor and critic training requires so called $TD(\lambda)$-estimators of future rewards. These were introduced into the critic training by Sutton (1988) and into the actor training by Kimura & Kobayashi (1998). The first step toward a proof of convergence of an approximation-based actor-critic algorithm was made by Sutton et al. (2000), who designed a critic that was in a special way compatible with the actor. The proof of convergence of an approximation-based algorithm with the actor-critic structure was given by Konda & Tsitsiklis (2003). Up to that time, actor-critic algorithms had been adjusting the policy vector along the policy gradient estimators. However, Peters et al. (2003) and Peters & Schaal (2008) introduced a very efficient actor training scheme based on adjusting the policy vector along estimators of the natural policy gradient (Kakade, 2002). The original algorithm presented by Peters et al. (2003) is not purely sequential as it trains the critic with the use of LSTD-Q(λ) (Nedić & Bertsekas, 2003) which is a method of batch mean-square approximation of the action-value function. However, Bhatnagar et al. (2008a) and Bhatnagar et al. (2008b) introduced algorithms based on natural policy gradient and fully sequential Actor- and Critic adjusting.

Algorithm 1. The Basic Actor-Critic. $\gamma \in (0,1)$ is a discount factor, $\lambda \in [0,1]$, $\bar{V}$ is the value function approximator (the critic) parameterized by vector v. β_t^θ and β_t^v are the step-sizes.

```
 0: Set $y_v = 0, y_\theta = 0, t := 1$. Initialize $\theta$ and $v$.
 1: Draw the action, $a_t \sim \pi(\cdot\,; s_t, \theta)$.
 2: Execute $a_t$, evaluate the next state $s_{t+1}$ and
    the reward $r_t$.
 3: Calculate the temporal difference of the
    form
 4:    $d_t(v) = r_t + \gamma\bar{V}(s_{t+1}; v) - \bar{V}(s_t; v)$.
 5: Adjust the actor:
 6:    $y_\theta := (\gamma\lambda)y_\theta + \beta_t^\theta \nabla_\theta \ln \pi(a_t; s_t, \theta)$
 7:    $\theta := \theta + y_\theta d_t(v)$
 8: Adjust the critic:
 9:    $y_v := (\gamma\lambda)y_v + \beta_t^v \nabla_v \bar{V}(s_t; v)$
10:    $v := v + y_v d_t(v)$.
11: Set $t := t + 1$ and repeat from Point 1.
```

2.1.1. The Basic Actor-Critic

Let us analyze the first example of methods that will be of interest for us: the algorithm presented by Kimura & Kobayashi (1998), quoted in the table beside, and called here the Basic Actor-Critic. The actor is represented here, as usually, by the parameterized policy π. The critic is represented by the approximator $\bar{V}(s; \upsilon)$ parameterized by the critic vector $\upsilon \in \Re^{n_\upsilon}$. The critic approximates the value function V^π, that is for a given policy, π, equal to the sum of future discounted rewards expected in a given state, namely

$$V^\pi(s) = E\left(\sum_{i\geq 0} \gamma^i r_{t+i} \big| s_t = s, \pi\right)$$

where $\gamma \in (0, 1)$ is a discount factor. The values β_t^θ and β_t^υ are the step-sizes: they are positive reals decreasing with growing t. Also, they should satisfy the standard stochastic approximation conditions (Kushner & Yin, 1997), namely

$$\sum_{t\geq 1} \beta_t = \infty, \quad \sum_{t\geq 1} \beta_t^2 < \infty.$$

Below, we show how this algorithm works: After each instant t, it performs a sequence of adjustments of θ and υ that modify these vectors along estimators of certain quality indexes associated with instant t. Before we explain the mechanics of Actor and Critic training, we have to present the fundamental concepts of $TD(\lambda)$ (Sutton, 1988) and incremental expectation maximization based on likelihood ratio (Williams, 1989).

$TD(\lambda)$-estimators of future rewards. Let us consider the following sum

$$R_t^k = r_t + \gamma r_{t+1} + \cdots + \gamma^{k-1} r_{t+k} + \gamma^k \bar{V}(s_{t+k+1}).$$

We can see that this is an estimator of the expected value of the sum of discounted rewards when starting from state s_t with action a_t and then following the current policy. The larger k, the smaller is the bias of the estimator (the less it relies on the non-precise approximator $\bar{V}$) and the larger is its variance. Then, we define

$$R_t^{(\lambda)} = \begin{cases} \sum_{k\geq 0} (1-\lambda)\lambda^k R_t^k & \text{for } \lambda \in [0, 1) \\ \lim_{\lambda\to 1} \sum_{k\geq 0} (1-\lambda)\lambda^k R_t^k = R_t^\infty & \text{for } \lambda = 1. \end{cases}$$

Hence, $R_t^{(\lambda)}$ is a weighed average of R_t^k. For small λ it weighs more those R_t^k with small k (more relying on the approximation $\bar{V}$) and for large λ, it relies more on those R_t^k with large k (more relying on rewards actually received). One can verify the equalities (Sutton, 1988)

$$R_t^{(\lambda)} = r_t + \bar{V}(s_{t+1}; \upsilon) + \sum_{k\geq 1} (\gamma\lambda)^k \Big(r_{t+i} + \gamma \bar{V}(s_{t+i+1}; \upsilon) - \bar{V}(s_{t+i}; \upsilon) \Big) \tag{1}$$

$$R_t^{(\lambda)} - \bar{V}(s_t; \upsilon) = \sum_{k\geq 0} (\gamma\lambda)^k \Big(r_{t+i} + \gamma \bar{V}(s_{t+i+1}; \upsilon) - \bar{V}(s_{t+i}; \upsilon) \Big) \tag{2}$$

which express $R_t^{(\lambda)}$ as a sum of diminishing elements and play a crucial role in the Actor-Critic design.

Incremental expectation maximization with the use of likelihood ratio. Suppose $R_t^{(\lambda)}$ is influenced by θ only through a_t. Then, we could exploit the following equation (Williams, 1989) to adjust θ in order to maximize the expected $R_t^{(\lambda)}$:

$$\nabla_\theta E_\theta\left(R_t^{(\lambda)}\right) = E_\theta\left(\nabla_\theta \ln \pi(a_t; s, \theta)(R_t^{(\lambda)} - c)\right) \tag{3}$$

where $c \in \Re$ is a nonrandom baseline and $\nabla_\theta \ln \pi$ is the *likelihood ratio.* Equation (3) states that the term $\nabla_\theta \ln \pi(a_t; s, \theta)(R_t^{(\lambda)} - c)$ is an unbiased estimator of the gradient of the quality index $E_\theta R_t^{(\lambda)}$. In particular, for $c = \bar{V}(s_t; \upsilon)$, we obtain a random vector

$$\nabla_\theta \ln \pi(a_t; s, \theta)\left(R_t^{(\lambda)} - \bar{V}(s_t; \upsilon)\right) \tag{4}$$

is an unbiased estimator of the gradient of the expected $R_t^{(\lambda)}$.

The Actor training. Let us consider the total adjustment of the policy vector θ during the work of the algorithm. To this end, we analyze the value of y_θ by the end of the algorithm's loop. Since y_θ results from summation in Point 6., at instant t it becomes equal to

$$y_\theta = \sum_{k=0}^{t-1} (\gamma\lambda)^k \beta_{t-k}^\theta \nabla_\theta \ln \pi(a_{t-k}; s_{t-k}, \theta).$$

Therefore, the total adjustment is equal to

$$\Delta\theta = \sum_{t>0} d_t(\upsilon) \sum_{k=0}^{t-1} (\gamma\lambda)^k \beta_{t-k}^\theta \nabla_\theta \ln \pi(a_{t-k}; s_{t-k}, \theta)$$

By changing the summation order we obtain

$$\Delta\theta = \sum_{t>0} \beta_t^\theta \nabla_\theta \ln \pi(a_t; s_t, \theta) \sum_{k\geq 0} (\gamma\lambda)^k d_{t+k}(\upsilon). \tag{5}$$

We can see that an element of the above sum, namely

$$\beta_t^\theta \nabla_\theta \ln \pi(a_t; s_t, \theta) \sum_{k\geq 0} (\gamma\lambda)^k d_{t+k}(\upsilon) \tag{6}$$

attributes to the state s_t the total adjustment of the policy vector that the visit in this state induces. By combining Eqs. (4), (2), and (6) we can see that the total adjustment induced at time t is equal to

$$\beta_t^\theta \nabla_\theta \ln \pi(a_t; s_t, \theta)\left(R_t^{(\lambda)} - \bar{V}(s_t; \upsilon)\right),$$

that is, a product of the step-size and an estimator of the gradient of the expected value of $R_t^{(\lambda)}$. Therefore, what the algorithm is trying to do is maximization of the sum of future discounted rewards expected in each state.

The critic training. A similar analysis reveals the compact form of the total adjustment of the critic vector. Namely, at the end of the algorithm's loop, the vector y_v is equal to

$$y_v = \sum_{k=0}^{t-1} (\gamma\lambda)^k \nabla_v \bar{V}(s_{t-k}; v).$$

Therefore, the total adjustment of the critic vector is equal to

$$\Delta v = \sum_{t>0} \beta_t^v d_t(v) \sum_{k=0}^{t-1} (\gamma\lambda)^k \nabla_v \bar{V}(s_{t-k}; v)$$

By changing the summation order we obtain

$$\Delta v = \sum_{t>0} \beta_t^v \nabla_v \bar{V}(s_t; v) \sum_{k\geq 0} (\gamma\lambda)^k d_{t+k}(v).$$

From Eq. (2) we also know that

$$\Delta v = \sum_{t>0} \beta_t^v \nabla_v \bar{V}(s_t; v) \left(R_t^{(\lambda)} - \bar{V}(s_t; v) \right).$$

We can see that an element of the above sum attributes to the state s_t the total adjustment of the critic vector that the visit in this state induces. In order to understand the character of this adjustment, suppose for a while that $R_t^{(\lambda)}$ does not depend on v. Then, it is easy to verify that

$$\nabla_v E_v \frac{1}{2} \left(R_t^{(\lambda)} - \bar{V}(s_t; v) \right)^2 = E_v \nabla_v \bar{V}(s_t; v) \left(\bar{V}(s_t; v) - R_t^{(\lambda)} \right).$$

Hence, the critic training consists in minimization of the quality index

$$E_v \frac{1}{2} \left(R_t^{(\lambda)} - \bar{V}(s_t; v) \right)^2$$

that represents the mean quadratic discrepancy between $\bar{V}(s_t; v)$ and $R_t^{(\lambda)}$. The minimization is performed by adjusting the critic vector along estimators of the gradient of this quality index.

2.1.2. The Natural Actor-Critic

Another example of methods that will be of interest for us is quoted below. The algorithm is called, for the purposes of this paper, the Natural Actor-Critic. It is based on actor adjustments along the natural policy gradient estimators Peters & Schaal (2008), but is fully sequential. It is similar to Algorithm 3 of (Bhatnagar et al., 2008a) with some minor modifications: (i) it maximizes discounted rewards rather than the average reward, (ii) it is based on TD(λ)-estimators of future rewards rather than TD(0)-estimators, (iii) the algorithm applies „forgetting" of the critic vector (v'', Step 10.) in a slightly simplified way, it avoids computing large matrices, (iv) the critic here need not to be linear.

The actor is represented, as usually, by the parameterized policy π. The critic is represented by the approximator $\bar{Q}(a,s;v,\theta)$ parameterized by $v = \begin{bmatrix} v' \\ v'' \end{bmatrix} \in \Re^{n_v}$, and the policy vector θ. The critic approximates the action-value function Q^π, that is for a given policy, π, equal to the sum of future discounted rewards expected in a given state, provided the given initial action, namely

Algorithm 2. The Natural Actor-Critic. $\gamma \in (0,1)$ is a discount factor, $\lambda \in [0,1]$, $\bar{Q}$ is the action-value function approximator (the critic) parameterized by vectors v and θ. β_t^θ and β_t^v are the step-sizes.

```
 0: Set $y_v = 0, t := 1$. Initialize $\theta$ and $v$. Draw $a_1 \sim \pi(\cdot\,; s_1, \theta)$.
 1: Execute $a_t$, evaluate the next state $s_{t+1}$ and the reward $r_t$.
 2: Draw the next action, $a_{t+1} \sim \pi(\cdot\,; s_{t+1}, \theta)$.
 3: Calculate the *temporal difference* of the form
 4:    $e_t(\theta, v) = r_t + \gamma\bar{V}(a_{t+1}, s_{t+1}; v') - \bar{Q}(a_t, s_t; v, \theta)$
 5: Adjust the critic:
 6:    $y_v := (\gamma\lambda)y_v + \beta_t^v \nabla_v \bar{Q}(a_t, s_t; v, \theta)$
 7:    $v := v + y_v e_t(\theta, v)$
 8: Adjust the actor:
 9:    $\theta := \theta + \beta_t^\theta v''$
10:    $v'' := (1 - \beta_t^\theta)v''$
11: Set $t := t + 1$ and repeat from Step 1.
```

$$\bar{Q}(a,s;v,\theta) \approx Q^\pi(a,s) = E\left(\sum_{i\geq 0} \gamma^i r_{t+i} \middle| s_t = s, a_t = a, \pi\right).$$

The approximator $\bar{Q}$ is of the form

$$\bar{Q}(a,s;v,\theta) = \bar{V}(s;v') + \nabla_\theta^T \ln \pi(a;s,\theta)v'' \tag{7}$$

where $\bar{V}$ is the value-function approximator linearly parameterized in v'. Because the critic is here partially defined by the actor, π, they are called *compatible* actor and critic.

The values β_t^θ and β_t^v are the step-sizes. They satisfy the standard stochastic approximation conditions and, furthermore, there exists $d > 0$ such that

$$\sum_{t>0} \left(\frac{\beta_t^\theta}{\beta_t^v}\right)^d < \infty.$$

This means that β_t^θ decreases "faster" than β_t^v; see (Konda & Tsitsiklis, 2003).

The scheme of the critic training in the Natural Actor-Critic remains the same as in the basic algorithm. The only difference lies in the subject of training: it is the action-value function approximator rather than the value function approximator. A similar argument to the one applied to the Basic Actor-Critic shows that the present algorithm minimizes the mean-quadratic discrepancy between $\bar{Q}(a_t, s_t; v, \theta)$ and $\bar{R}_t^{(\lambda)}$, where $\bar{R}_t^{(\lambda)}$ is defined as

$$\bar{R}_t^{(\lambda)} = r_t + \gamma\bar{V}(s_{t+1};v) + \sum_{i\geq 1}(\gamma\lambda)^i \left(r_{t+i} + \gamma\bar{V}(s_{t+i+1};v) - \bar{Q}(a_{t+i}, s_{t+i}; v, \theta)\right).$$

See Eq. (1) to compare $R_t^{(\lambda)}$ with $\bar{R}_t^{(\lambda)}$.

The actor training is designed to make the policy parameter θ follow the direction of the natural policy gradient (Kakade, 2002). Below we provide some intuition why the actor training in this algorithm may work. For more detailed discussion we refer an interested reader to (Peters & Schaal, 2008). Suppose $\bar{Q}$ is a good approximation of the true action-value function. What happens with the value $\bar{Q}(a, s; \upsilon, \theta)$ expected in a given state s when we adjust the actor vector by $\delta\upsilon''$ for a small $\delta > 0$?

$$\begin{aligned}
E_{s,\theta+\delta\upsilon''}\bar{Q}(a, s; \upsilon, \theta) &= \int \left(\bar{V}(s; \upsilon') + \nabla_\theta^T \ln \pi(a; s, \theta)\upsilon''\right) \pi(a; s, \theta + \delta\upsilon'')\mathrm{d}a \\
&\approx \int \left(\bar{V}(s; \upsilon') + \nabla_\theta^T \ln \pi(a; s, \theta)\upsilon''\right) \left(\pi(a; s, \theta) + \nabla_\theta^T \pi(a; s, \theta)\delta\upsilon''\right) \mathrm{d}a \\
&= \int \left(\bar{V}(s; \upsilon') + \nabla_\theta^T \ln \pi(a; s, \theta)\upsilon''\right) \left(1 + \nabla_\theta^T \ln \pi(a; s, \theta)\delta\upsilon''\right) \pi(a; s, \theta)\mathrm{d}a \\
&= \int \left(\bar{V}(s; \upsilon') + \nabla_\theta^T \ln \pi(a; s, \theta)\upsilon''\right) \pi(a; s, \theta)\mathrm{d}a \\
&\quad + \bar{V}(s; \upsilon')E_{s,\theta}\Big(\ln \pi(a; s, \theta)\Big)\delta\upsilon'' \\
&\quad + \int \nabla_\theta^T \ln \pi(a; s, \theta)^T \upsilon'' \nabla_\theta^T \ln \pi(a; s, \theta)\delta\upsilon''\pi(a; s, \theta)\mathrm{d}a \\
&= E_{s,\theta}\bar{Q}(a, s; \upsilon, \theta) + 0 + \delta\|\upsilon''\|^2 \int \|\nabla_\theta \ln \pi(a; s, \theta)\|^2 \pi(a; s, \theta)\mathrm{d}a
\end{aligned}$$

We can see that the expected value of $\bar{Q}$ increases. The vector υ'' is applied to adjust the policy in Step 9 of the algorithm and is shortened afterwards. The use of υ'' to policy improvement makes the value of this vector obsolete. Its influence on further policy improvement is hence reduced; similarly as it is done in the original algorithm presented by Peters et al. (2003).

2.1.3. Generalization

In general, we will consider sequential actor-critic-type algorithms characterized by the following features:

1. Actions are generated by a stationary policy (actor) i.e., a distribution π parameterized by state s_t and the policy vector $\theta \in \Re^{n_\theta}$, namely
$$a_t \sim \pi(\cdot\,; s_t, \theta).$$
2. A visit in state s_t causes a modification of the policy vector θ by a product
$$\beta_t^\theta \widehat{\phi}_t.$$
$\widehat{\phi}_t$ on average indicates the direction in which θ assures larger future rewards expected in state s_t whereas $(\beta_t^\theta, t = 1, 2, \dots)$ is a vanishing sequence of step-sizes.

3. The algorithm may compute $\widehat{\phi}_t$ with the use of an auxiliary parameters $\upsilon \in \Re^{n_\upsilon}$. A visit in state s_t results in a modification of υ by a vector
$$\beta_t^\upsilon \widehat{\psi}_t.$$
$\widehat{\psi}_t$ on average points into the direction where υ assures better quality of $\widehat{\phi}_t$ whereas $(\beta_t^\upsilon, t = 1, 2, \dots)$ is a vanishing sequence of step-sizes.
4. The vectors $\widehat{\phi}_t$ and $\widehat{\psi}_t$, while different from one another, are of the same form
$$G_t(\theta, \upsilon) \sum_{k \geq 0} (\alpha\rho)^k z_{t,k}(\theta, \upsilon) \tag{8}$$
where G_t is a vector defined by s_t and a_t, $\alpha \in [0, 1)$, $\rho \in [0, 1)$, and $z_{t,k} \in \Re$ is defined by $s_{t+k}, a_{t+k}, r_{t+k}, s_{t+k+1}$, and possibly a_{t+k+1}.

A number of RL algorithms fit into the above schema. In the Basic Actor-Critic algorithm mentioned above we have
$$\begin{aligned}\widehat{\phi}_t &= \nabla_\theta \ln \pi(a_t; s_t, \theta) \sum_{k \geq 0} (\gamma\lambda)^k d_{t+k}(\upsilon), \\ \widehat{\psi}_t &= \nabla_\upsilon \bar{V}(s_t; \upsilon) \sum_{k \geq 0} (\gamma\lambda)^k d_{t+k}(\upsilon),\end{aligned}$$
which means that both $\widehat{\phi}_t$ and $\widehat{\psi}_t$ are of the form (8) with $\gamma\lambda = \alpha\rho$, and
$$G_t(\theta, \upsilon) = \nabla_\theta \ln \pi(a_t; s_t, \theta), \quad z_{t,k}(\theta, \upsilon) = d_{t+k}(\upsilon)$$
for the actor, and
$$G_t(\theta, \upsilon) = \nabla_\upsilon \bar{V}(s_t; \upsilon), \quad z_{t,k}(\theta, \upsilon) = d_{t+k}(\upsilon)$$
for the critic.

In the Natural Actor-Critic algorithm we have
$$\begin{aligned}\widehat{\phi}_t &= \upsilon'', \\ \widehat{\psi}_t &= \nabla_\upsilon \bar{Q}(a_t, s_t; \upsilon, \theta) \sum_{k \geq 0} (\gamma\lambda)^k e_{t+k}(\upsilon, \theta),\end{aligned}$$
which means that both $\widehat{\phi}_t$ and $\widehat{\psi}_t$ are of the form (8) with $\gamma\lambda = \alpha\rho$, and
$$G_t(\theta, \upsilon) = \upsilon'', \quad z_{t,k}(\theta, \upsilon) = \begin{cases} 1 \text{ for } k = 0, \\ 0 \text{ for } k > 0 \end{cases}$$
for the actor, and
$$G_t(\theta, \upsilon) = \nabla_\upsilon \bar{Q}(a_t, s_t; \upsilon, \theta), \quad z_{t,k}(\theta, \upsilon) = e_{t+k}(\upsilon, \theta)$$
for the critic.

Important algorithms that also fit into the discussed schema are the actor-critic presented by Konda & Tsitsiklis (2003) and OLPOMDP (Bartlett & Baxter, 2000), although OLPOMDP does not employ any auxiliary vector υ.

Analysis of RL methods usually assumes that for each policy parameter θ, the sequence of states $\{s_t, t = 1, 2, \dots\}$ forms an aperiodic and irreducible Markov chain with a stationary distribution. In our presentation of the mechanics of the sequential RL algorithms, we will also assume that the states visited when the policy parameter θ is applied are drawn independently from a stationary distribution. A reader interested in a deeper analysis and proofs of convergence is referred to (Bartlett & Baxter, 2000; Konda & Tsitsiklis, 2003).

Let us analyze the average direction of $\widehat{\phi}_t$ and $\widehat{\psi}_t$. Namely, let ϕ be a function defined as

$$\phi(s, \theta, \upsilon) = E_{\theta,\upsilon,\beta}\left(\widehat{\phi}_t \middle| s_t = s\right). \tag{9}$$

The definition of ϕ is based on the assumption that θ, υ, and the step-sizes remain constant when $\widehat{\phi}_t$ is calculated. In fact, they slightly vary and each $\widehat{\phi}_t$ is in fact a biased estimator of $\phi(s_t, \theta, \upsilon)$ for θ and υ used at time t. However, this bias is small and since the dynamics of the parameters decreases in time, the bias asymptotically vanishes. The average $\phi(s, \theta, \upsilon)$ weighed by the steady-state distribution defines the direction of the drift of the policy vector.

The drift of υ may be analyzed in a similar way. Namely, let ψ be a function defined as

$$\psi(s, \theta, \upsilon) = E_{\theta,\upsilon,\beta}\left(\widehat{\psi}_t \middle| s_t = s\right). \tag{10}$$

As above, the definition of ψ requires that θ, υ, and the step-sizes remain constant during the time when $\widehat{\psi}_t$ is computed. The drift of υ is defined by the average $\psi(s, \theta, \upsilon)$ weighed by the steady-state distribution. The usual role of the drift of the auxiliary parameter is to move it toward the point $\upsilon^*(\theta)$ such that the average $\phi(s, \theta, \upsilon^*(\theta))$ approximates either a policy gradient or a natural policy gradient. Hence, adjustments of θ ultimately lead to policy improvement.

3. Experience Replay

The main idea analyzed in this paper is to apply to the agent's experience the same processing as a sequential actor-critic-type algorithm would, yet more intensively. A generic algorithm augmented by experience replay is presented in the table below.

After each instant t, the original sequential algorithm estimates $\phi(s_t, \theta, \upsilon)$ i.e., the direction of policy improvement, and adjusts the policy vector θ along the estimate. Within each instant t, the modified algorithm repeatedly draws one of the recently visited states, s_i, estimates $\phi(s_i, \theta, \upsilon)$, and modifies the policy vector along the estimate. Essentially both algorithms achieve the same goal but (i) the modified one does it more intensively and (ii) it employs experience gathered after visiting state s_i to adjust various policies characterized by different policy vectors. The auxiliary vector υ undergoes similar operations in both algorithms.

Because the policy vector is constantly changing, each time its adjustment is performed, it is computed on the basis of the new values of θ and v. The intensity of replaying, $\nu(t)$, must be bounded for the sake of correctness of the algorithm. It is also limited by the computation power available during the agent–environment interaction. $\nu(t)$ should be additionally limited for small t to prevent many recalculations of few tuples in the database and to avoid overtraining.

Designing the estimators of ϕ and ψ for Steps 7 and 9 of Algorithm 3 we have to guarantee that their variance is bounded and their bias asymptotically vanishes. This is the only way for the algorithm to preserve the limit properties of the original sequential method. In the following section we prove that both the requirements are met if the estimators are of the same generic form defined below.

Algorithm 3. Actor-Critic with Experience Replay. Estimators mentioned in Steps 6 and 7 are based on the data in a database.

```
0:  $t := 1$. Initialize $\theta$ and $v$.
1:  Draw and execute an action, $a_t \sim \pi(\cdot\,; s_t, \theta)$.
2:  Register the tuple $\langle s_t, a_t, \theta, r_t, s_{t+1}\rangle$ in the database.
3:  Make sure only $N$ most recent tuples remain in the database.
4:  Repeat $\nu(t)$ times:
5:     Draw $i \in \{t-N+1, t-N+2, \dots, t\}$.
6:     Adjust $\theta$ along an estimator of $\phi(s_i, \theta, v)$:
7:        $\theta := \theta + \beta^\theta_t \widehat{\phi}^r_i(\theta, v)$.
8:     Adjust $v$ along an estimator of $\psi(s_i, \theta, v)$:
9:        $v := v + \beta^v_t \widehat{\psi}^r_i(\theta, v)$.
10: Assign $t := t + 1$ and repeat from Step 1.
```

Let $b > 1$, θ_{i+j} be the policy vector applied to generate a_{i+j}, and K be drawn independently from $Geom(\rho)$, the geometric distribution[1] with parameter $\rho \in [0, 1)$. Also, let χ be equal to 0 if $z_{t,k}$ is not explicitly defined by a_{t+k+1} (as in the basic AC), and to 1 otherwise (as in AC of Konda & Tsitsiklis (2003)). We introduce the *randomized-truncated estimators* $\widehat{\phi}^r_i(\theta, v)$ and $\widehat{\psi}^r_i(\theta, v)$ of the same generic form

$$\sum_{k=0}^{K} G_i(\theta, v)\alpha^k z_{i,k}(\theta, v) \min\left\{\prod_{j=0}^{k+\chi} \frac{\pi(a_{i+j}; s_{i+j}, \theta)}{\pi(a_{i+j}; s_{i+j}, \theta_{i+j})}, b\right\}. \qquad (11)$$

where $\widehat{\phi}^r_i$ is defined by those G and z defining $\widehat{\phi}_t$, and $\widehat{\psi}^r_i$ is defined by those G and z that define $\widehat{\psi}_t$. We can see that (11) closely resembles the original form (8) of $\widehat{\phi}_t$ and $\widehat{\psi}_t$. Though, there are two important differences. First, the infinite sum is replaced by the finite one with the appropriately designed random limit. Second, truncated density ratios are introduced in order to compensate for the fact that the current policy is different than the one that generated the actions $a_t, a_{t-1}, \dots$ that are present in the database.

[1] That is, random variable K of values in $\{0, 1, 2, \dots\}$ has distribution $Geom(\rho)$, iff $P(K = m) = (1-\rho)\rho^m$ for nonnegative integer m.

4. Properties of the Randomized-Truncated Estimator

The purpose of this section is to establish conditions under which estimator (11) is asymptotically unbiased and of bounded variance. To this end, we will first discuss the estimation technique called *importance sampling* in Sec. 4.1. This technique, in its pure form, suffers from the problem of excessive variance of estimators. We cope this problem by truncating the density ratios. This simple trick may be successful if it is applied to a certain class of families of probabilistic distributions analyzed in Sec. 4.2. Sufficient conditions for bounded variance and asymptotic unbiasness of (11) are finally established in 4.3.

4.1. Importance Sampling

Let $g(\cdot\,;\theta)$ be a density of random elements a in $\mathcal{A}$ parametrized by a vector $\theta \in \Re^{n_\theta}$. Suppose for a given parameter θ_0 an action a_0 is drawn from $g(\cdot\,;\theta_0)$. The action $a_0 \in \mathcal{A}$ yields a certain vector $d(a_0)$ where $d : \mathcal{A} \mapsto \Re^n$. We are interested in estimation of the expected $d(a)$ for actions drawn with the use of an arbitrary parameter θ, namely

$$D(\theta) = E_\theta d(a) = \int_{\mathcal{A}} d(\alpha) g(\alpha;\theta)\,\mathrm{d}\alpha. \tag{12}$$

In order to construct an estimator of $D(\theta)$, we may apply the estimation technique called *importance sampling* (Rubinstein, 1981) frequently applied in RL (Peshkin & Shelton, 2002; Uchibe & Doya, 2004; Frank et al., 2008). Namely, given θ_0 and θ, the statistic

$$\widehat{D}(\theta) = d(a_0)\, g(a_0;\theta)/g(a_0;\theta_0) \tag{13}$$

is an unbiased estimator of $D(\theta)$ since

$$E_{\theta_0}\widehat{D}(\theta) = \int d(\alpha)\frac{g(\alpha;\theta)}{g(\alpha;\theta_0)}\, g(\alpha;\theta_0)\,\mathrm{d}\alpha = D(\theta). \tag{14}$$

A well-known drawback of estimator (13) is its excessive variance when distributions $g(\cdot\,;\theta_0)$ and $g(\cdot\,;\theta)$ are far from one another (Wawrzyński & Pacut, 2006). A simple yet practically satisfying way of fighting the variance problem is to truncate the density ratio in (13). Namely, the modified estimator has the form

$$\widehat{D}^b(\theta) = d(a_0) \min\left\{\frac{g(a_0;\theta)}{g(a_0;\theta_0)}, b\right\} \tag{15}$$

for constant $b > 1$. The above *truncated IS estimator* has also been applied in (Uchibe & Doya, 2004). Let us analyze its properties. Obviously if $\|d\|$ is bounded, then the variance of (15) is also bounded. Importantly, its bias happens to be bounded as well if only g satisfies certain regularity conditions and θ is close enough to θ_0. We specify those conditions below.

4.2. Regular Distributions

It seems reasonable to expect that the bias of the truncated estimator (15) is small for θ close to θ_0. It is proved below that in principle it is true if only g is regular in a certain sense. Then, for small $\|\theta - \theta_0\|$, the densities ratio is rarely truncated as it rarely exceeds b. For the purpose of this discussion, the family of densities $g(\cdot\,;\theta)$ parameterized by θ will be called (M_1, M_2)-*regular* if it has the following features:

(i) for every $a \in \mathcal{A}$ the mapping $\theta \mapsto g(a;\theta)$ is continuous and differentiable,

(ii) the trace of the Fisher information i.e., $I(\theta) = E_\theta \frac{\partial \ln g(a;\theta)}{\partial \theta}\frac{\partial \ln g(a;\theta)}{\partial \theta^T}$, is bounded by M_1,

(iii) absolute eigenvalues of the Hessian $\frac{\partial^2 \ln g(a;\theta)}{\partial \theta^T \partial \theta}$ are bounded by M_2.

The conditions above are typically satisfied by distributions that we could think of. For instance, the normal distribution $N(\theta, C)$ with nonsingular C is (M_1, M_2)-*regular* for $M_1 = \operatorname{tr} C^{-1}$ and $M_2 = \max \operatorname{eig} C^{-1}$. In the experimental study presented in Sec. 5 a policy is applied in the form of the normal distributions with constant variance and mean value being a certain function of θ. Let us denote this function by f. We thus have

$$g(a;\theta) = \frac{1}{\sqrt{2\Pi}^n |C|} \exp\left(-0.5(a - f(\theta))^T C^{-1}(a - f(\theta))\right).$$

It is easy to verify that this distribution is also (M_1, M_2)-regular if only C is nonsingular and the f function and its derivatives are uniformly continuous and differentiable.

The propositions below apply the definition of (M_1, M_2)-regular distribution to establish bounds of the bias of the truncated importance sampling estimator (15). The first one relates this bias to the probability that the density ratio is truncated.

Proposition 1. *For all θ_0, θ, it is true that $\widehat{D}^b(\theta)$ defined by* (15) *satisfies*

$$\left\| E_{\theta_0} \widehat{D}^b(\theta) - D(\theta) \right\| \le M P_\theta \left(\frac{g(a;\theta)}{g(a;\theta_0)} > b \right),$$

where $M = \sup_{a \in \mathcal{A}} \|d(a)\|$.

Proof: We split $E_{\theta_0} \widehat{D}^b(\theta)$ and $D(\theta)$ into pairs of integrals. For the first element of each pair the density ratio does not reach its upper bound and for the second one it does. The first elements cancel each other out and we derive a bound for the second ones. Namely, let us split $\mathcal{A}$ into two parts

$$A^b_{\theta,\theta_0} = \left\{ a : \frac{g(a;\theta)}{g(a;\theta_0)} \le b \right\}, \qquad \overline{A}^b_{\theta,\theta_0} = \left\{ a : \frac{g(a;\theta)}{g(a;\theta_0)} > b \right\} = \mathcal{A} - A^b_{\theta,\theta_0}.$$

Then, we have

$$\begin{aligned}
&\left\| E_{\theta_0}\widehat{D}^b(\theta) - D(\theta) \right\| \\
&= \left\| \int\limits_{A^b_{\theta,\theta_0} \cup \overline{A}^b_{\theta,\theta_0}} d(\alpha) \min\left\{ \frac{g(\alpha;\theta)}{g(\alpha;\theta_0)},\, b \right\} g(\alpha;\theta_0)\mathrm{d}\alpha - \int\limits_{A^b_{\theta,\theta_0} \cup \overline{A}^b_{\theta,\theta_0}} d(\alpha) g(\alpha;\theta)\mathrm{d}\alpha \right\| \\
&= \left\| \int\limits_{A^b_{\theta,\theta_0}} \Big[d(\alpha)\frac{g(\alpha;\theta)}{g(\alpha;\theta_0)} g(\alpha;\theta_0) - d(\alpha)g(\alpha;\theta) \Big] \mathrm{d}\alpha + \int\limits_{\overline{A}^b_{\theta,\theta_0}} \Big[d(\alpha) b g(\alpha;\theta_0) - d(\alpha)g(\alpha;\theta) \Big] \mathrm{d}\alpha \right\| \\
&= \left\| 0 + \int\limits_{\overline{A}^b_{\theta,\theta_0}} d(\alpha) \Big[b\frac{g(\alpha;\theta_0)}{g(\alpha;\theta)} - 1 \Big] g(\alpha;\theta)\mathrm{d}\alpha \right\|
\end{aligned}$$

Because for $\alpha \in \overline{A}^b_{\theta,\theta_0}$ we have

$$b\frac{g(\alpha;\theta_0)}{g(\alpha;\theta)} \leq 1,$$

and we finally obtain

$$\left\| E_{\theta_0}\widehat{D}^b(\theta) - D(\theta) \right\| \leq \int\limits_{\overline{A}^b_{\theta,\theta_0}} \|d(\alpha)\| g(\alpha;\theta)\mathrm{d}\alpha \leq M P_\theta(a \in \overline{A}^b_{\theta,\theta_0}) = M P_\theta \left(\frac{g(a;\theta)}{g(a;\theta_0)} > b \right)$$

■
Note that P_θ is the probability defined by the condition that the random value a is drawn from the density parameterized by θ. Another proposition defines a bound of P_θ.

Proposition 2. *For each $M_1, M_2 > 0$, and $b > 1$ there exists $M > 0$ such that if the density g is (M_1, M_2)-regular, then for all θ_0, θ*

$$P_\theta\big(g(a;\theta)/g(a;\theta_0) > b\big) \leq M\|\theta - \theta_0\|^2.$$

In the proof of the above proposition, the following notation is applied that "converts" predicates into numeric values:

$$[predicate] = \begin{cases} 1 \text{ if the } predicate \text{ is true} \\ 0 \text{ otherwise} \end{cases} \tag{16}$$

E.g. $|x| = x[x > 0] - x[x < 0]$. This notation will also be applied in the experimental study (Sec. 5).

Proof of Proposition 2: We have

$$g(a;\theta)/g(a;\theta_0) > b \Leftrightarrow g(a;\theta) > bg(a;\theta_0) \Leftrightarrow \frac{g(a;\theta)}{g(a;\theta_0)} > b$$
$$\Leftrightarrow \ln\frac{g(a;\theta)}{g(a;\theta_0)} > \ln b \Leftrightarrow (\ln b)^{-1}\big(\ln g(a;\theta) - \ln(a;\theta_0)\big) > 1.$$

Consequently

$$P_\theta(g(a;\theta)/g(a;\theta_0) > b) = P_\theta\left((\ln b)^{-1}\big(\ln g(a;\theta) - \ln(a;\theta_0)\big) > 1\right).$$

Then, we obtain

$$\begin{aligned}
&P_\theta(g(a;\theta)/g(a;\theta_0) > b)\\
&\quad= \int \left[(\ln b)^{-1}\big(\ln g(\alpha;\theta) - \ln g(\alpha;\theta_0)\big) > 1\right] g(\alpha;\theta)\mathrm{d}\alpha\\
&\quad= \int \left[(\ln b)^{-2}\big(\ln g(\alpha;\theta) - \ln g(\alpha;\theta_0)\big)^2 > 1\right] g(\alpha;\theta)\mathrm{d}\alpha\\
&\quad\leq \int (\ln b)^{-2}\big(\ln g(\alpha;\theta) - \ln g(\alpha;\theta_0)\big)^2 g(\alpha;\theta)\mathrm{d}\alpha
\end{aligned}$$

The last inequality results from the fact that $[x > 1] \leq x$ if only $x \geq 0$. Because g is (M_1, M_2)-regular, there exists M_2 such that $(\forall\alpha,\theta)\|\nabla^2_\theta \ln(\alpha;\theta)\| < M_2$ and we have

$$\begin{aligned}
&P_\theta(g(a;\theta)/g(a;\theta_0) > b)\\
&\quad\leq (\ln b)^{-2}\int \big(\|\nabla_\theta \ln g(\alpha;\theta)\|\|\theta_0-\theta\| + M_2\|\theta_0-\theta\|^2\big)^2 g(\alpha;\theta)\mathrm{d}\alpha && (17)\\
&\quad\leq \|\theta_0-\theta\|^2(\ln b)^{-2}\int \|\nabla_\theta \ln g(\alpha;\theta)\|^2 g(\alpha;\theta)\mathrm{d}\alpha && (18)\\
&\qquad + \|\theta_0-\theta\|^3 2M_2(\ln b)^{-2}\int \|\nabla_\theta \ln g(\alpha;\theta)\| g(\alpha;\theta)\mathrm{d}\alpha && (19)\\
&\qquad + \|\theta_0-\theta\|^4 M_2^2(\ln b)^{-2}\int g(\alpha;\theta)\mathrm{d}\alpha && (20)\\
&\quad\leq \|\theta_0-\theta\|^2(\ln b)^{-2}\left(M_1 + 2\|\theta_0-\theta\|M_2\sqrt{M_1} + \|\theta_0-\theta\|^2 M_2^2\right) && (21)
\end{aligned}$$

Inequality (17) results from the second order Taylor expansion of $\ln g(\alpha;\theta+\Delta\theta)$ and the bound of Hessian eigenvalues. The bounds of (18) and (19) result from Property (ii) of (M_1, M_2)-regular distributions. The bound of (19) results also from the fact that $E|f(a)| \leq (E|f(a)^p|)^{1/p}$ for $p > 1$. We now exploit the fact that the required probability is bounded by 1. We search for M in the form

$$M = (\ln b)^{-2}M_1 + M_3 + M_4$$

such that

$$\min\{\|\theta_0-\theta\|^3(\ln b)^{-2}2M_2\sqrt{M_1}, 1\} \leq M_3\|\theta_0-\theta\|^2,$$
$$\min\{\|\theta_0-\theta\|^4(\ln b)^{-2}M_2^2, 1\} \leq M_4\|\theta_0-\theta\|^2.$$

It is easy to verify that the smallest such M_3 and M_4 have the values

$$\begin{aligned} M_3 &= (\ln b)^{-4/3} 2^{2/3} M_2^{2/3} M_1^{1/3}, \\ M_4 &= (\ln b)^{-1} M_2. \end{aligned}$$

We thus obtain

$$\begin{aligned} &P_\theta(g(a;\theta)/g(a;\theta_0) > b) \\ &\quad \le \min\left\{ \|\theta_0-\theta\|^2 (\ln b)^{-2} \left(M_1 + 2\|\theta_0-\theta\| M_2 \sqrt{M_1} + \|\theta_0-\theta\|^2 M_2^2 \right), 1 \right\} \\ &\quad \le \min\{\|\theta_0-\theta\|^2 (\ln b)^{-2} M_1, 1\} + \min\left\{ 2\|\theta_0-\theta\|^3 (\ln b)^{-2} M_2 \sqrt{M_1}, 1 \right\} \\ &\qquad + \min\{\|\theta_0-\theta\|^4 (\ln b)^{-2} M_2^2, 1\} \\ &\quad \le M\|\theta_0-\theta\|^2 \\ &\quad \le \left((\ln b)^{-2} M_1 + (\ln b)^{-4/3} 2^{2/3} M_2^{2/3} M_1^{1/3} + (\ln b)^{-1} M_2 \right) \|\theta_0-\theta\|^2. \end{aligned}$$

■

In the present work a special case of the analyzed problem is discussed in which d is a function of a finite sequence of independent random elements. In fact, both the above propositions apply to this case because a sequence of random vectors is also a random vector. However, a joint density of a random sequence need not be (M_1, M_2)-regular even if densities of all its elements are. This causes difficulties in application of Proposition 2. The proposition below treats this issue.

Proposition 3. *For each $M_1, M_2 > 0$, and $b > 1$ there exists $M > 0$ such that if all densities in the sequence $g_1, \ldots, g_k$ are (M_1, M_2)-regular, then for each set of vectors $(\theta_1, \ldots, \theta_k, \theta)$,*

$$P_\theta\left(\prod_{i=1}^k \frac{g_i(a_i;\theta)}{g_i(a_i;\theta_i)} > b \right) \le \frac{k^2 M}{(\ln b)^2} \sum_{i=1}^k \|\theta-\theta_i\|^2.$$

provided random elements $a_i, i = 1, \ldots, k$ are drawn independently from $g_i(\cdot\,;\theta_i)$.

Proof: Let us denote

$$A = \left\{ \prod_{i=1}^k \frac{g_i(a_i;\theta)}{g_i(a_i;\theta_i)} > b \right\}, \quad A_i = \left\{ \frac{g_i(a_i;\theta)}{g_i(a_i;\theta_i)} > \sqrt[k]{b} \right\},$$

for $i = 1, \ldots, k$. We have

$$\bigcap_{i=1}^k (\Omega - A_i) \subseteq \Omega - A$$

and

$$\prod_{i=1}^k (1 - P_\theta(A_i)) \le 1 - P_\theta(A).$$

A simple induction shows that

$$1 - \sum_{i=1}^{k} P_\theta(A_i) \le \prod_{i=1}^{k} (1 - P_\theta(A_i)).$$

A combination of the above two inequalities yields

$$P_\theta(A) \le \sum_{i=1}^{k} P_\theta(A_i).$$

Application of Proposition 2 completes the proof. ■

4.3. Bounds for Variance and Bias of the Randomized-Truncated Estimator

We are now ready to derive properties of estimator (11). Because ϕ and ψ have the same form, further in this section we will focus on ϕ, understanding that exactly the same can be said about ψ. Because of truncation, it is easy to verify that variance of $\widehat{\phi}_i^r$ is bounded if only G and z are bounded. However, also because of truncation, the estimator is biased. Fortunately, the bias vanishes when the parameters θ_{i+j} of distributions that generated actions get closer to the analyzed value θ. The following proposition clarifies this issue. In order to prove that the variance of $\widehat{\phi}_i^r(\theta, \upsilon)$ is bounded, it is enough to establish a bound of the trace of the variance. This is what the following proposition does.

Proposition 4. *Suppose G_i and $z_{i,k}$ are uniformly bounded such that $\|G_i z_{j,k}\| \le M$. Then*

$$\operatorname{tr} \mathcal{V}\widehat{\phi}_i^r(\theta, \upsilon) \le \frac{b^2 M^2}{(1-\alpha)^2}.$$

Proof: We have

$$\operatorname{tr} \mathcal{V}\widehat{\phi}_i^r(\theta, \upsilon) \le E\left(\widehat{\phi}_i^r(\theta, \upsilon)^T \widehat{\phi}_i^r(\theta, \upsilon)\right) \le \left(\sum_{j\ge 0} \alpha^j M b\right)^2 \le \frac{b^2 M^2}{(1-\alpha)^2}$$

■

Another proposition establishes bounds of bias of (11).

Proposition 5. *Suppose G_i and $z_{i,k}$ are uniformly bounded and distributions $\pi(\cdot\,; s_{i+j}, \cdot\,)$ are (M_1, M_2)-regular for certain common M_1, M_2 and all $j \ge 0$. Then, there exists c such that*

$$\|E\widehat{\phi}_i^r(\theta, \upsilon) - \phi(s_i, \theta, \upsilon)\| \le c\delta^2$$

for all i, θ, υ and

$$\delta = \sup_{j\ge 0} \|\theta - \theta_{i+j}\|.$$

Proof: ϕ_t (8) is a weighed, by $(\alpha\rho)^k$, sum of components of the form $G_t(\theta, v) z_{t,k}(\theta, v)$. Each of them is a function of the random elements $a_t, s_{t+1}, \dots, a_{t+k}, s_{t+k+1}$, and possibly a_{t+k+1}. On the other hand, each element of $\widehat{\phi}_i^r$ (11) is a truncated importance-sampling estimator of $E_{\theta,v}\big(G_i(\theta, v) z_{i,k}(\theta, v)|s_i\big)$, scaled by α^k. Propositions 1 and 3 define a bound of the bias of these estimators, which is of the form

$$\frac{M(k+2)^2}{(\ln b)^2} \sum_{j=0}^{k+1} \|\theta - \theta_{i+j}\|^2$$

Because the sum in the above formula is no greater than $(k+2)\delta^2$, and because $P(K \geq m) = \rho^m$ (which is a simple property of $Geom(\rho)$), we obtain

$$\begin{aligned} \|E\widehat{\phi}_i^r(\theta, v) - \phi(s_i, \theta, v)\| &\leq E\left(\sum_{k=0}^{K} \alpha^k \frac{M(k+2)^3\delta^2}{(\ln b)^2}\right) \\ &= \sum_{k\geq 0} (\rho\alpha)^k \frac{M(k+2)^3\delta^2}{(\ln b)^2} = \delta^2 \frac{M}{(\ln b)^2} \sum_{k\geq 0} (\rho\alpha)^k (k+2)^3. \end{aligned}$$

Because $\rho\alpha \in (0, 1)$, the value of the last sum is bounded, which completes the proof. ■

Let us now see how the estimator (11) is asymptotically unbiased. Because only a certain constant number, N, of recent events are kept in the database, at the moment t the indexes i are drawn randomly from the set $\{t-N+1, t-N+2, \dots, t\}$. Let us analyze δ_t as the upper bound of $\|\theta - \theta_{i+j}\|$ for i drawn at moment t. Because the step-sizes β_t^θ vanish in time, the changes of the policy vector, θ, also vanish. Therefore, the value δ_t must vanish as well, the conditions of Proposition 5 are met, and the estimators $\widehat{\phi}_i^r$ and $\widehat{\psi}_i^r$ become asymptotically unbiased.

Another consideration is associated with the changes of steady-state distribution that follow the adjustments of the policy vector. Our objective is to change θ along the average $\phi(s, \theta, v)$ and v along the average $\psi(s, \theta, v)$, both weighed by the steady-state distribution. Even if $\widehat{\phi}_i^r(\theta, v)$ is a good estimator of $\phi(s_i, \theta, v)$, this is not enough for it to be a good estimator of the average ϕ. However, this *would* be enough if s_i could be regarded as drawn from the steady-state distribution. Once again, we exploit the facts that only N events are kept in the database and that, for growing t, the changes of θ get smaller. Then, whenever $\widehat{\phi}_i^r(\theta, v)$ is computed at moment t, state s_i has been visited when the policy in use has been similar to the one with parameter θ.

5. Experimental Study

This section reports experiments on application of the presented methodology to challenging learning-control tasks. First, the Basic Actor-Critic augmented by experience replay is applied to the cart-pole swing-up problem (Doya, 2000). Second, the Natural Actor-Critic with experience replay is confronted with Half-Cheetah (Wawrzyński, 2007). In all the experiments the controlled system is emulated i.e., simulated in real time. A quantum of system's real time is equal to a quantum of the corresponding computer time, which means that the computer has a lot of spare time that can be devoted to parallel computations. The setup of our experiments is designed to closely resemble a situation when control of a physical machine is to be optimized in real time by means of learning. The purpose of this study is to verify the presented methodology in those circumstances.

5.1. The Basic Actor-Critic for Cart-Pole Swing-Up

In this section we report how experience replay can be applied to one of the simplest RL algorithms, namely the Basic Actor-Critic. To continue the discussion in Section 2, we base our experiments on the version of the algorithm presented by Kimura & Kobayashi (1998).

The cart-pole swing-up task (Doya, 2000) is used for the experiments. It is an issue of control of a cart moving along a track with a freely hanging pole (see Fig. 1). The system is controlled by a force applied to the cart. The objective is to avoid hitting the track bounds, swing the pole, turn it up, and stabilize upwards.

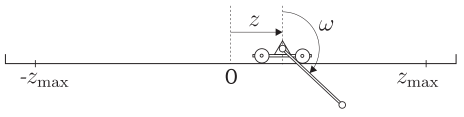

Figure 1. The cart-pole swing-up problem.

In the experiments the actor, π, is comprised of two parts: a neural network and a normal distribution. The input of the network is the state. The network has a hidden layer with 20 sigmoidal (arctg) neurons and a linear output neuron. The output becomes a mean value of the normal distribution with variance $\sigma^2 = 4$. The distribution generates actions. The critic is a neural network of the same structure as above with 50 neurons in its hidden layer. All neurons in the networks have a constant input (bias). The initial weights in the hidden layers are drawn randomly from

Algorithm 4. The Basic Actor-Critic with Experience Replay (Replaying BAC).

```
 0:  $t := 1$. Initialize $\theta$ and $\upsilon$.
 1:  Draw and execute an action, $a_t \sim \pi(\cdot\,; s_t, \theta)$.
 2:  Register the tuple $\langle s_t, a_t, \theta, r_t, s_{t+1}\rangle$ in the database.
 3:  Make sure only $N$ most recent tuples remain in the
     database.
 4:  Repeat $\nu(t)$ times:
 5:    Draw $i \in \{t-N+1, t-N+2, \dots, t\}$.
       Draw $K \sim Geom(\rho)$.
       Calculate $SUM$ equal to
       $\sum_{k=0}^{K} \alpha^k d_{i+k} \min\left\{\prod_{j=0}^{k} \frac{\pi(a_{i+j}; s_{i+j}, \theta)}{\pi(a_{i+j}; s_{i+j}, \theta_{i+j})}, b\right\}$
       for $d_{i+k} = r_{i+k} + \gamma\bar{V}(s_{i+k+1}; \upsilon) - \bar{V}(s_{i+k}; \upsilon)$.
 6:    Adjust $\theta$ along an estimator of $\phi(s_i, \theta, \upsilon)$:
 7:      $\theta := \theta + \beta_t^\theta \nabla_\theta \ln \pi(a_i; s_i, \theta) SUM$.
 8:    Adjust $\upsilon$ along an estimator of $\psi(s_i, \theta, \upsilon)$:
 9:      $\upsilon := \upsilon + \beta_t^\upsilon \nabla_\upsilon \bar{V}(s_i; \upsilon) SUM$.
10:  Assign $t := t + 1$ and repeat from Step 1.
```

the normal distribution $N(0,1)$ and weights of the output layers are initially set to zero.

Implementation of the Basic Actor-Critic augmented by experience replay (Replaying BAC, in short) follows the discussion in Section 3. We use ν in the form

$$\nu(t) = \min\{c_1 t, c_0\}. \tag{22}$$

that defines the replying intensity. It is characterized by c_0 and c_1. The computer speed together with the period length between two actions for the emulated system bound the maximal value of the ultimate intensity c_0 of the experience replay.

There are four state variables: position of the cart z, the pole angle ω, and their time derivatives $\dot{z}$, $\dot{\omega}$. The force F applied to the cart once 0.1 sec. is a single variable. A typical reward used in this problem is equal to the elevation of the pole tip. Initially, the waving pole hovers and the rewards are close to -1. When the goal is reached and the pole is stabilized upwards, the rewards are close to 1. Physical restrictions make the state of the cart-pole *acceptable* if the cart position is within the track limits $[-2.4, 2.4]$. The action a_t is equal to the force and is limited to an interval $[-10, 10]$, i.e., $F_t = \min\{\max\{-10, a_t\}, 10\}$. The reward is calculated as

$$r_t = r_t^0 - 0.2|F_t - a_t| \quad \text{with} \quad r_t^0 = \begin{cases} \cos\omega & \text{if } |z| \le 1.2 \wedge |\dot{\omega}| < 2\pi, \\ -1 & \text{if } |z| \in (1.2, 2.4] \vee |\dot{\omega}| \ge 2\pi, \\ -30 & \text{if } |z| > 2.4, \end{cases}$$

for z, ω, and $\dot{\omega}$ measured at $t+1$. The state vector s_t feeding the approximators is normalized, namely

$$s_t = \left[\frac{z}{2}, \frac{\dot{z}}{3}, \frac{\sin\omega}{0.8}, \frac{\cos\omega}{0.8}, \frac{\dot{\omega}}{4}\right]^T.$$

The training consists of a sequence of *trials*. A trial ends when the pendulum's state becomes unacceptable. Otherwise, the trial lasts for a random real time drawn from the exponential distribution with the expected value equal to 20 sec. (each moment has equal probability of forcing the trial to end). At the beginning of each trial, the state is reset by drawing z and ω from the uniform distributions $U(-2.4, 2.4)$ and $U(0, 2\pi)$, respectively, and setting $\dot{z}$, $\dot{\omega}$ to zero.

The parameters of the Basic Actor-Critic taken in our simulations are: $\gamma = 0.95$, $\lambda = 0.5$, $\beta_t^\theta = \beta_t^v \equiv 0.003$, and $\lambda = 0.5$. The algorithm usually obtains the satisfying behavior after 2000 trials (about 6 hours of the agent's real time), see the left part of Fig. 2.

Replaying AC (see the middle part of Fig. 2) exhibits a satisfying behavior after just 90 trials (about 20 minutes of the cart-pole's time). The parameters used in the replaying AC are: $\beta_t^\theta = \beta_t^v \equiv 0.001$, $b = 2$, $c_0 = 1000$, $c_1 = 0.3$, $N = 10^5$, $\alpha = \gamma = 0.95$ and $\rho = \lambda = 0.5$. Note that the difference in efficiency of the Basic AC and the replaying AC is so large that the results must be presented in different figures.

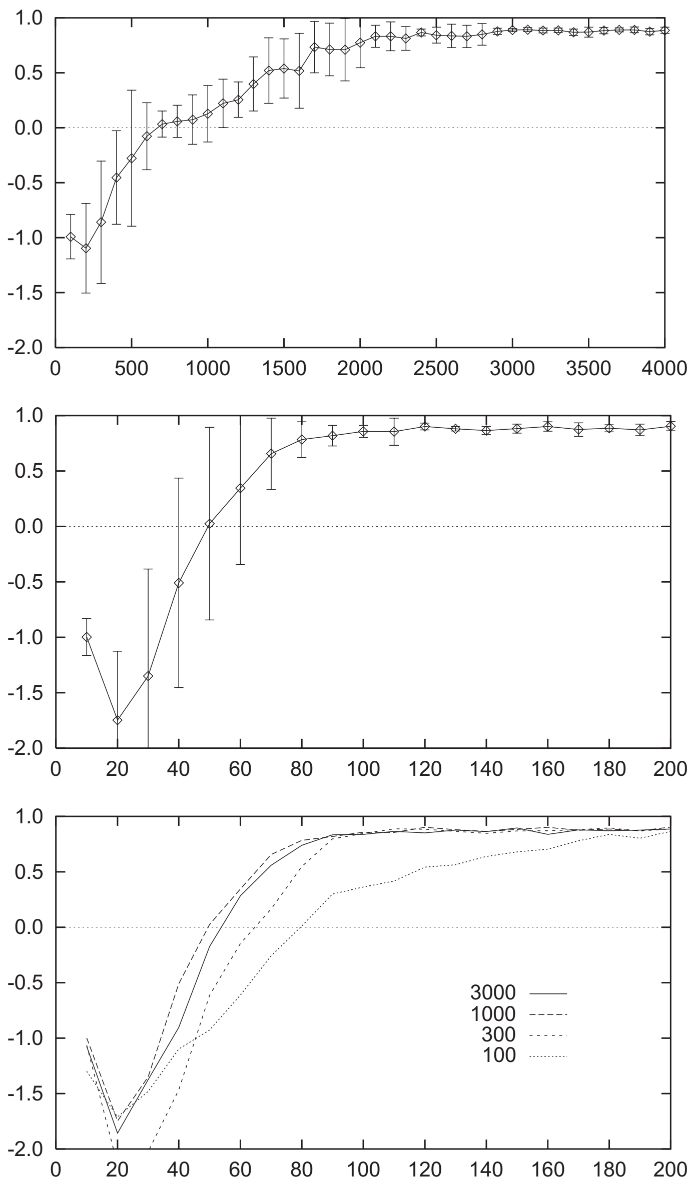

Figure 2: Actor-Critics for cart-pole swing-up: The average reward vs. trial number. The average duration of a trial is 20 sec. (200 time instants). Each curve averages 10 runs. *Top:* The Basic AC. Each point averages 100 consecutive trials. The one-sigma limits are calculated to assess run-to-run variability of trial averages. *Middle:* The Basic AC with Experience Replay (ER). Note that the number of trials in this figure is about 20 times smaller than that of the top figure for the basic method. Each point averages only 10 consecutive trials. *Bottom:* The Basic AC with ER for various replaying intensity, c_0.

The right part of Fig. 2 presents learning curves for the replaying AC with various intensity of experience replay i.e., the parameter c_0. We can see, that the larger intensity, the fastest convergence. However, intensity above 1000 does not yield further improvement. It appears that the value $c_0 = 1000$ is sufficient to utilize on-line all the information from the learning process for policy determination.

5.2. The Natural Actor-Critic for Half-Cheetah

We are interested in applications of the MDP framework to learning reactive policies of machines. In this section we analyze a challenging problem of this type, namely learning to run a planar model of a large cat. The cat robot, called Half-Cheetah, is presented in Fig. 3. It is a planar kinematic chain of 9 links, 8 joints, and 2 "paws". Because 5-th joint is fixed at 180^o, and its adjacent links have the same length, joints 4-th and 6-th are always at the same position; therefore, the object does not look like a chain. The angles of 4-th and 5-th joint are fixed, all the the others are controllable. Consequently, Half-Cheetah is a model of a 6-degree-of-freedom walking robot.

The torque τ_i applied at i-th joint is calculated as

$$\tau_i = T_i \min\left\{\max\{-1, \tau_i^0 + a_i^0\}, 1\right\}$$

where τ_i^0 is a "spontaneous" torque at i-th joint, a_i^0 is the output of the learning controller, and T_i expresses "strength" of i-th joint. The spontaneous torque τ_i^0 is implemented as a PD-controller with saturation. It roughly stabilizes the i-th joint at its initial angle. We follow a typical setting of control system design: While it is usually relatively easy to provide a controller that stabilizes a system around a certain state (e.g. by using PD-controllers), it is much more difficult to design a controller that makes the system perform a certain nontrivial activity. In our paper, the controller is to *learn* to make Half-Cheetah run.

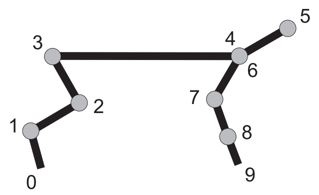

Figure 3. The initial position of Half-Cheetah. It consists of 9 links, 8 joints among which 2 are fixed (4, 5), and 2 paws (0, 9).

The fact that Half-Cheetah is designed as a kinematic chain allows to efficiently emulate its dynamics, e.g. using the Newton-Euler method (Walker & Orin, 1982). The mass of the object is concentrated in its joints which weigh

$$1,\ 1,\ 1,\ 4,\ 1,\ 2,\ 1,\ 1,\ 1,\ \text{and}\ 1$$

kilos, respectively. The consecutive links of Half-Cheetah are of lengths

$$\frac{0.2}{\cos\frac{\pi}{12}},\ \frac{0.15}{\cos\frac{\pi}{3}},\ \frac{0.25}{\cos\frac{\pi}{6}},\ 1,\ 0.3,\ 0.3,\ \frac{0.25}{\cos\frac{\pi}{9}},\ \frac{0.2}{\cos\frac{\pi}{9}},\ \frac{0.15}{\cos\frac{\pi}{9}}$$

meters. The initial angle between the first link and the perpendicular is $-\pi/12$. The initial angles of the joints, ω_i^0 for $i = 1, \dots, 8$, are

$$-\frac{5}{12}\pi,\ \frac{1}{2}\pi,\ -\frac{2}{3}\pi,\ \frac{1}{6}\pi,\ -\pi,\ \frac{1}{6}\pi,\ \frac{5}{18}\pi,\ \text{and } 0.$$

The movement of joints are limited to intervals:

$$\left[-\frac{5}{6}\pi, -\frac{1}{6}\pi\right],\ \left[\frac{1}{4}\pi, \frac{3}{4}\pi\right],\ \left[-\frac{5}{6}\pi, -\frac{1}{3}\pi\right],\ \left[\frac{1}{6}\pi, \frac{1}{6}\pi\right],\ [-\pi, -\pi],$$
$$\left[\frac{1}{18}\pi, \frac{1}{2}\pi\right],\ \left[0, \frac{8}{9}\pi\right],\ \left[-\frac{2}{3}\pi, \frac{1}{9}\pi\right].$$

The object is controlled by applying torques at the joints. The torque τ_i at i-th joint is computed in two phases: the variable

$$\tau_i^0 = 2\pi^{-1}\text{arctg}\left(-2(\omega_i - \omega_i^0) - 0.05\dot{\omega}_i\right)$$

that expresses a "spontaneous" torque at i-th joint implied by its angle, ω_i, and angular velocity, $\dot{\omega}_i$, is added to the signal a_i^0 coming from the learning controller, projected on the interval $[-1, 1]$, and multiplied by the "strength", T_i, of i-th joint, namely

$$\tau_i = T_i \min\left\{\max\{-1, \tau_i^0 + a_i^0\}, 1\right\}.$$

The "strength" in consecutive joints are as follows

$$60,\ 90,\ 120,\ -,\ -,\ 90,\ 60,\ 30,$$

where "–" corresponds to undefined strengths of fixed joints.

The objective of control is to make Half-Cheetah run forward as fast as possible. Notice that without the external control (i.e. for $a_i^0 \equiv 0$), the torque $T_i\tau_i^0$ implements a PD control (a differentiator with proportional gain) with saturation. It stabilizes the joint angle close to its initial value which makes Half-Cheetah stand still.

$s_{t,1} = (y_0 - 0.1)/0.1$	$s_{t,6} = \dot{y}_3/0.7$	$s_{t,11+3j} = \cos\omega_j^l/0.7$
$s_{t,2} = \dot{y}_0$	$s_{t,7} = \dot{y}_4/0.7$	$s_{t,12+3j} = \sin\omega_j^l/0.7$
$s_{t,3} = ([y_0 = 0] - 0.5)/0.5$	$s_{t,8} = (y_9 - 0.1)/0.1$	$s_{t,13+3j} = \dot{\omega}_j^l/6$
$s_{t,4} = ((y_3 + y_4)/2 - 0.6)/0.2$	$s_{t,9} = \dot{y}_9$	$s_{t,23+3k} = \cos\omega_{6+k}^l/0.7$
$s_{t,5} = (\dot{x}_3 + \dot{x}_4)/4 - 1/2$	$s_{t,10} = ([y_9 = 0] - 0.5)/0.5$	$s_{t,24+3k} = \sin\omega_{6+k}^l/0.7$
		$s_{t,25+3k} = \dot{\omega}_{6+k}^l/6$

Table 1. Variables of Half-Cheetah state. $j = 0, 1, 2, 3$ (behind the neck) and $k = 0, 1, 2$ (in front of the neck).

State and reward. In order to apply the reinforcement learning to Half-Cheetah, one must define the states and the rewards the learning algorithm has access to. In the specification below, the joints are indexed from 0 to 9, whereas the links have indexes from 0 to 8. The 2-dimensional position of i-th joint is denoted by (x_i, y_i) and its speed by $(\dot{x}_i, \dot{y}_i)$. The angle between i-th link and the horizontal line is denoted by ω_i^l and the corresponding angular speed is $\dot{\omega}_i^l$. Each state variable is normalized to roughly cover the interval $[-1, 1]$. We apply the notation (16). For instance, the term $[y_0 = 0]$ is equal to 1 if the hind paw stands on the ground and 0 otherwise. All 31 variables describing the state of Half-Cheetah are depicted in Table 1.

In our definition of reward we apply a *soft-step function* of the form

$$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ 2x^2 & \text{for } x \in (0, 0.5] \\ 1-2(x-1)^2 & \text{for } x \in (0.5, 1] \\ 1 & \text{for } x > 1. \end{cases}$$

It is mainly applied to penalize Half-Cheetah for touching the ground with heels, knees, etc. Note that f may by approximated to the ordinary step function by multiplying its argument by a large positive number. The reward is calculated as a sum of basic rewards that play various roles in different stages of learning. Its definition is provided below for $I_j = \{1, 2, 3, 6, 7, 8\}$,

$$\begin{aligned} r = {} & 0.5(\dot{x}_3 + \dot{x}_4) && (23) \\ & - 0.05 \sum_{i \in I_j} \max\left\{ \left|\tau_i^0 + a_i^0\right| - 1, 0 \right\} && (24) \\ & - 0.1 \sum_{i \in I_j} \min\{|\tau_i^w|, 50\} && (25) \\ & + \big(\max\big\{ \min\{\dot{y}_3, 1\}, f(0.5[y_0 > 0] + 5y_0), \\ & \qquad \min\{\dot{y}_4, 1\}, f(0.5[y_9 > 0] + 5y_9) \big\} - 1 \big) \times && (26) \\ & \quad \times (1 - f(\dot{x}_3/2 + \dot{x}_4/2)) \\ & + \big(f(9y_1) - 1\big) + \big(f(5y_2) - 1\big) + \big(f(2y_5) - 1\big). && (27) \end{aligned}$$

The meaning of particular elements of the reward is as follows:

(23) the reward for moving ahead with the speed as large as possible,

(24) a penalty for an attempt to apply torque from outside of the permissible interval,

(25) a penalty for the internal force that keeps the joint angle within its bounds; τ_i^w is a torque applied by the "tendon" (if i-th joint angle not equal to either of its bounds, then $\tau_i^w = 0$),

(26) a penalty for not moving the trunk up and keeping the paws on the ground when the animal is not moving forward,

(27) penalties for touching the ground with the heel, the knee, and the head, respectively.

Although reward is quite complex, its main part is the Half-Cheetah speed [m/s].

The learning algorithm. In order to make Half-Cheetah run, we combine the Natural Actor-Critic and the idea of experience replay. The algorithm we apply (Replaying NAC, in short) is specified below.

The policy applied to Half-Cheetah is similar to the one applied to the cart-pole swing-up problem. Namely, it is comprised of two parts: a neural network and a normal distribution. The input of the network is the state. The output becomes a mean value of the normal distribution with covariance matrix $C = 5^2 I$. The distribution generates actions. The elements of the 6-dimensional action a are transformed into the control stimuli a^0 as

Algorithm 5. The Natural Actor-Critic with Experience Replay (Replaying NAC).

```
0:  $t := 1$. Initialize $\theta$ and $\upsilon$.
1:  Draw and execute an action, $a_t \sim \pi(\cdot\,; s_t, \theta)$.
2:  Register the tuple $s_t, a_t, \theta, r_t, s_{t+1}$ in the database.
3:  Make sure only $N$ most recent tuples remain in the
    database.
4:  Repeat $\nu(t)$ times:
5:    Draw $i \in \{t-N+1, t-N+2, \dots, t\}$.
      Draw $K \sim Geom(\rho)$.
      Calculate $SUM$ equal to
      $\sum_{k=0}^{K} \alpha^k e_{i+k} \min\left\{\prod_{j=0}^{k} \frac{\pi(a_{i+j}; s_{i+j}, \theta)}{\pi(a_{i+j}; s_{i+j}, \theta_{i+j})}, b\right\}$
      for $e_{i+k}$ equal to
      $r_{i+k} + \gamma \bar{V}(s_{i+k+1}; \upsilon) - \bar{Q}(a_{i+k}, s_{i+k}; \upsilon, \theta)$.
6:    Adjust $\theta$ along an estimator of $\phi(s_i, \theta, \upsilon)$:
7:      $\theta := \theta + \beta_t^\theta \upsilon''$.
8:    Adjust $\upsilon$ along an estimator of $\psi(s_i, \theta, \upsilon)$:
9:      $\upsilon := \upsilon + \beta_t^\upsilon \nabla_\upsilon \bar{Q}(a_i, s_i; \upsilon, \theta) SUM$.
        $\upsilon'' := (1 - \beta_t^\theta)\upsilon''$.
10: Assign $t := t + 1$ and repeat from Step 1.
```

$$a_i^0 = a_j/30$$

where the indexes $i = 1, 2, 3, 6, 7, 8$ correspond to $j = 1, 2, 3, 4, 5, 6$, respectively. The constant $1/30$ results from the fact that the control signals a_i^0 cover the interval $[-1, 1]$ while we would prefer the outputs of the network to be of larger scale, in this case roughly cover the interval $[-30, 30]$.

The second approximator used by the learning algorithm is the critic, i.e. the approximation of the action-value function (7). The critic output is a sum of two elements. The first one results from the actor and the second one, $\bar{V}$, is implemented by a feedforward neural network. Both the critic network and the actor network have the form of two layer perceptron with linear output layer. Their hidden layers consist of M^A (the actor) and M^C (the critic) sigmoidal (arctan) elements. Each neuron has a constant input (bias). The initial weights of the hidden layers are drawn randomly from the normal distribution $N(0, 1)$ and the initial weights of the output layers are set to zero.

The parameters of the Natural Actor-Critic are as follows: (the actor) $M^A = 80$, $C = 5^2 I$, (the critic) $M^C = 160$, (step-sizes) $\beta_t^\theta \equiv 10^{-3}$, $\beta_t^\upsilon \equiv 3.10^{-5}$, (estimation) $\gamma = 0.99$, $\lambda = 0.9$. The resulting learning curves are depicted on the left-hand part of Fig. 5.

The parameters of the replaying NAC are as follows: (the actor) $M^A = 80$, $C = 5^2 I$, (the critic) $M^C = 160$, (step-sizes) $\beta_t^\theta \equiv 10^{-3}$, $\beta_t^\upsilon \equiv 2.10^{-5}$, (database) $N = 3.10^4$, (estimation) $\gamma = \alpha = 0.99$, $\lambda = \rho = 0.9$, (computational effort) $c_0 = 30$, $c_1 = 0.3$. With a computer equipped with Intel Quad$^{\text{TM}}$Q9300, the simulations were carried on in real time of Half-Cheetah.

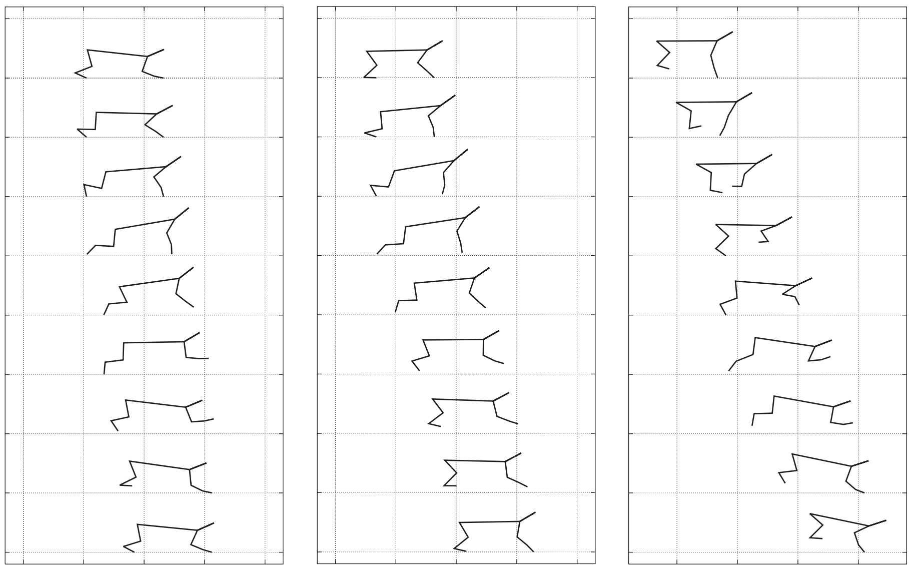

Figure 4: Typical sequences of Half-Cheetah states at various stages of learning by the Replaying NAC. *Left:* Awkward walk after 1 hours of training. *Middle:* Trot after 2.5 hours of training. *Right:* Nimble run after 7 hours of training.

Experiments. Learning curves for the setting discussed above is shown in Fig. 5. A single trial lasts, on the average, for 5 sec. The left-hand part of Fig. 5 reports experiments with the Natural Actor-Critic applied to Half-Cheetah. It is seen, that the algorithm learns to control Half-Cheetah in about 3000 trials, which is about 42 hours of Half-Cheetah. The middle, and the right-hand part of Fig. 5 presents the averaged learning curve for the Replaying NAC applied to the same problem. The curve reports about 7 hours of learning. The algorithm learns to control Half-Cheetah in about 4500 trials, which is about 6 hours of Half-Cheetah time. It is interesting to observe the Half-Cheetah learned policy at various stages of learning (Fig. 4). In about 1000 trials (1 hour of computer, or real, time) Half-Cheetah walks without significant difficulties, albeit awkwardly. After about 2000 trials $(2,5$ hours) it trots. The movement obtained in 4500 trials (6 hours) is a fairly nimble run, very similar to what one might expect from an animal constructed that way.

It is interesting to watch Half-Cheetah learning. At first it only stands shaking because of the noise in the control system. Thank to the main part of the reward (23), it quickly learns to lean ahead. But then its start to fall on its head which it quickly starts to avoid thank to component (27) of the reward. All the time Half-Cheetah learns to jump, thank to component (26) of the reward. After some time, it can move ahead by means of small jumps. Then, these jumps are smoothed to become a nimble run.

Let us now analyze whether the concepts introduced in the paper indeed

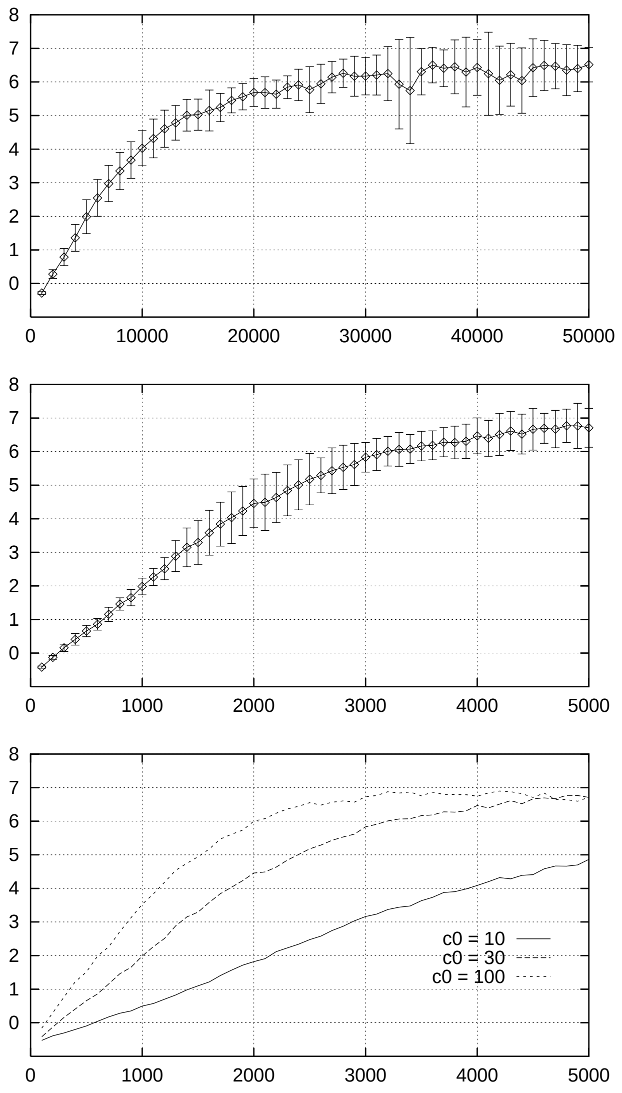

Figure 5. Actor-Critics for Half-Cheetah: The average reward vs. trial number. *Top:* The Natural AC. Each point averages 1000 consecutive trials. The curve averages 10 runs. The one-sigma limits are calculated to assess run-to-run variability of trial averages. *Middle:* The Natural AC with Experience Replay (ER). Note that the number of trials in this figure is about 10 times smaller than that of the top figure for the basic method. The curve averages 10 runs and each point averages only 100 consecutive trials. *Bottom:* The Natural AC with ER for various replaying intensity, c_0. Each curve averages 10 runs. It is seen that the larger intensity, the faster convergence.

improve the quality and the speed of learning. The right part of Fig. 5 demonstrates how the intensity of the computation process translates into the speed of learning. It is seen that the larger intensity, the faster convergence. Plausibly for a certain large c_0, further increase of this parameter does not yield learning speed improvement. However, it is quite time-consuming to investigate high values of c_0. In fact, for $n > 30$ the computations are too slow to take place in real time of Half-Cheetah. The computer time of a single run is then proportional to c_0 and for $c_0 = 100$ it is around 31 hours. Obviously, it is only a matter of computer power. With a fast enough computer, the processing for $c_0 = 100$ could be performed in real time of Half-Cheetah. A satisfying policy could be then obtained after 3200 trails, which is about 4 hours of Half-Cheetah time.

5.3. Discussion

Step-sizes applied in all the presented experiments were set invariable in time in order to make equal terms for comparison of various methods. While convergence of all analyzed algorithms requires the step-sizes to be decreasing in time, in this section we were rather interested in the speed of learning of various methods than in their limit properties analyzed earlier. In the experiments with the Basic Actor-Critic and the Natural Actor-Critic, the step-sizes were of the largest values still giving good ultimate policies but not causing instability of the learning process. The following heuristics was applied, rather successfully, to determine step-sizes for the algorithms with experience replay: When introducing experience replay to a given Actor-Critic algorithm working with a given problem, one should decrease the step-sizes about 3 times. The only exception is the actor step-size in the Natural Actor-Critic; it should be kept unchanged. This is because the actor vector in this method is not adjusted by a function of actions; therefore, it can be adjusted in the algorithm with experience replay in the same way as in the original one, rather than by means of a randomized-truncated estimator.

The experiments suggest that the algorithms augmented by experience replay learn policies of the same quality in much shorter time in comparison the the original methods. Growing intensity of the experience replay generally increases the speed of learning. Obviously there are some limits of this growth. These limits were possible to determine in the case of the Replaying BAC and cart-pole swing-up. Namely, more than 1000 replays per time-step does not yield any further increase in the speed of learning. Experiments with the Replaying NAC and Half-Cheetah did not detect these limits: the more computational power was engaged, the larger speed of learning was obtained. In both cases, application of experience replay caused the increase in the speed of learning of sequential Actor-Critic algorithms up to 20 times.

All implementations of Actor-Critic algorithms presented in this study applied nonlinear critics taught incrementally by means of the TD(λ) estimators of future rewards. In fact this method is proved to be convergent only for linear approximators (Tsitsiklis & Van Roy, 1997) and it surely does not converge for certain nonlinear structures. However, our experiments are based on the assumption that this method is also convergent for a large class of nonlinear approximators that encompasses feedforward neural networks. Available empirical results confirm that assumption.

Introduction of experience replay into an actor-critic algorithm requires providing a few additional coefficients. One of these is the threshold for the density ratios truncating, denoted above by b. Our experience suggests that its value around 2 or 3 gives satisfying behavior. Its smaller values gives poor ultimate performance (large bias of the improvement direction estimators) and larger ones result in instability of the learning process (large variance of the estimators). However, it is unknown whether this experience is representative. Another unobvious parameter is the size of the database of registered events, denoted above by N. The following rule of defining its value has been applied quite successfully in the experiments. The size of the database should be (i) at least an order of magnitude larger than the sum of dimensions of the actor vector and the critic vector, and (ii) large enough for the database to contain a representative history of an agent-environment interaction.

6. Conclusions

In this paper we combined the technique of experience replay for reinforcement learning speedup with sequential Actor-Critic-like algorithms. These algorithms deserve serious attention since they represent the most successful approach to applying reinforcement learning to realistic control tasks with continuous state and action spaces. Unlike the Q-learning algorithm for which experience replay was used in prior work, they are on-policy algorithms that require the actions producing observations for policy improvements be drawn from the current, actually followed policy. Therefore, to make this combination correct, it was necessary to ensure that the discrepancy between current policy and the policy used at the time when the replayed experience was generated is compensated appropriately. This is achieved by a novel variant of experience replay, using importance sampling for the estimation of the direction of policy modification. The analysis of the corresponding estimators established bounds of their variance and bias, and the latter was shown to asymptotically vanish when the step-size is reduced. This form of experience replay can be therefore used to augment Actor-Critic reinforcement learning algorithms without degrading their convergence properties.

As it has been verified experimentally, experience replay gives a radical learning speedup by allocating the available computational power to processing past observations. The required number of interactions with the environment, which is critical for the applicability of reinforcement learning to real-world tasks, can be considerably reduced. For a moderately difficult cart-pole swing-up task we observed a speedup factor of 20, allowing a satisfactory policy to be reached after as little as 20 minutes of the cart-pole time, compared to about 6 hours required by the basic Actor-Critic and 2 hours obtained by the method of Doya (2000). For the fairly difficult Half-Cheetah task we observed a speedup factor of

10, allowing a satisfactory policy to be reached after as little as 4 hours of Half-Cheetah time (assuming availability of very large computation power), compared to about 42 hours required by the natural Actor-Critic. This is definitely an important step toward enhancing the application potential of reinforcement learning to realistic, challenging control tasks.

References

Abbeel, P., & Ng, A. Y. (2005). Exploration and apprenticeship learning in reinforcement learning. In *Proceedings of the 22nd International Conference on Machine learning*, (pp. 1–8). New York, NY, USA: ACM.

Abbeel, P., Quigley, M., & Ng, A. Y. (2006). Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd International Conference on Machine learning*, (pp. 1–8). New York, NY, USA: ACM.

Bartlett, P. L., & Baxter, J. (2000). Stochastic optimization of controlled partially observable markov decision processes. In *Proceedings of the 39th IEEE Conference on Decision and Control (CDC00)*, vol. 1, (pp. 124–129).

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike adaptive elements that can learn difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, *13*, 834–846.

Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., & Lee, M. (2008a). Incremental natural actor-critic algorithms. In *Advances in Neural Information Processing Systems*, vol. 21.

Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., & Lee, M. (2008b). Incremental natural actor-critic algorithms. *Automatica, to appear*.

Cichosz, P. (1999). An analysis of experience replay in temporal difference learning. *Cybernetics and Systems*, *30*, 341–363.

Doya, K. (2000). Reinforcement learning in continuous time and space. *Neural Computation*, *12*(1), 219–245.

Frank, J., Mannor, S., & Precup, D. (2008). Reinforcement learning in the presence of rare events. In *Proceedings of the 25th International Conference on Machine Learning*, (pp. 580–587).

Gullapalli, V. (1990). A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, *3*(6), 671–692.

Kakade, S. (2002). A natural policy gradient. In *Advances in Neural Information Processing Systems*, vol. 14, (pp. 1531–1538). MIT Press.

Kimura, H., & Kobayashi, S. (1998). An analysis of actor/critic algorithm using eligibility traces: Reinforcement learning with imperfect value functions. In *Proceedings of the 15th International Conference on Machine Learning*, (pp. 278–286).

Konda, V., & Tsitsiklis, J. (2003). Actor-critic algorithms. *SIAM Journal on Control and Optimization*, *42*(4), 1143–1166.

Kushner, H. J., & Yin, G. (1997). *Stochastic Approximation Algorithms and Applications*. New York: Springer-Verlag.

Lin, L.-J. (1992). *Reinforcement learning for robots using neural networks*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, USA.

Lin, L.-J., & Mitchell, T. M. (1992). Memory approaches to reinforcement learning in non-markovian domains. Tech. Rep. CMU-CS-92-138, Carnegie Mellon University, Pittsburgh, PA 15213.

Mahadevan, S., & Connell, J. (1992). Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, *55*(2-3), 311–365.

Nedić, A., & Bertsekas, D. P. (2003). Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems*, *13*(1-2), 79–110.

Peshkin, L., & Mukherjee, S. (2001). Bounds on sample size for policy evaluation in Markov environments. In *Proceedings of the 14th Annual Conference on Computational Learning Theory*, vol. 2111, (pp. 616–629). Springer, Berlin.

Peshkin, L., & Shelton, C. (2002). Learning from scarce experience. In *Proceedings of the 19th International Conference on Machine Learning*, (pp. 498–505).

Peters, J., & Schaal, S. (2008). Natural actor-critic. *Neurocomputing*, *71*(7-9), 1180–1190.

Peters, J., Vijayakumar, S., & Schaal, S. (2003). Reinforcement learning for humanoid robotics. In *Humanoids2003, Third IEEE-RAS International Conference on Humanoid Robots*. Karlsruhe, Germany.

Precup, D., Sutton, R. S., & Dasgupta, S. (2001). Off-policy temporal-difference learning with function approximation. In *Proceedings of the 18th International Conference on Machine Learning*, (pp. 417–424). Morgan Kaufmann, San Francisco, CA.

Rubinstein, R. Y. (1981). *Simulation and the Monte Carlo Method*. New York, USA: John Wiley & Sons, Inc.

Shelton, C. R. (2001). Policy improvement for POMDPs using normalized importance sampling. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, (pp. 496–503).

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*, 9–44.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the 7th International Conference on Machine Learning*, (pp. 216–224). Morgan Kaufmann.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

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*, vol. 12, (pp. 1057–1063). MIT Press.

Tsitsiklis, J. N., & Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, *42*(5), 674–690.

Uchibe, E., & Doya, K. (2004). Competitive-cooperative-concurrent reinforcement learning with importance sampling. In *Proceedings of International Conference on Simulation of Adaptive Behavior: From Animals and Animats*, (pp. 287–296).

Walker, M. W., & Orin, D. E. (1982). Efficient dynamic computer simulation of robotic mechanisms. *Journal of Dynamic Systems, Measurement and Control*, *104*, 205–211.

Watkins, C., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*, 279–292.

Wawrzyński, P. (2007). Learning to control a 6-degree-of-freedom walking robot. In *Proceedings of EUROCON 2007*, (pp. 698–705). IEEE.

Wawrzyński, P., & Pacut, A. (2006). Balanced importance sampling estimation. In *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems*, (pp. 66–73).

Williams, R. J. (1989). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*(23).

Figure 1: The cart-pole swing-up.

Figure 2: Actor-Critics for cart-pole swing-up

1. Introduction

2. Problem Formulation

3. Experience Replay

4. Properties

4.1. Importance sampling

4.2. Regular distributions

4.3. Bounds for variance and bias of (11)

5. Experimental Study

5.1. The basic Actor-Critic vs. Cart-Pole Swing-Up

5.2. The natural Actor-Critic vs. Half-Cheetah

6. Conclusions

Figure 3: The initial position of Half-Cheetah.

Table 1: Variables of Half-Cheetah state.

Figure 4: Typical sequences of Half-Cheetah states at various stages of learning.

Figure 5: Actor-Critics for Half-Cheetah: The average reward vs. trial number.