Appeared in *Proceedings of The International Joint Conference on Neural Networks, Budapest, July 2004,* pp. 1091-1096, © IEEE

Model-free off-policy reinforcement learning in continuous environment

Paweł Wawrzyński and Andrzej Pacut, *Senior Member, IEEE*
Institute of Control and Computation Engineering
Warsaw University of Technology
00–665 Warsaw, Poland
P.Wawrzynski@elka.pw.edu.pl, http://home.elka.pw.edu.pl/∼pwawrzyn
A.Pacut@ia.pw.edu.pl, http://www.ia.pw.edu.pl/∼pacut

***Abstract*— We introduce an algorithm of reinforcement learning in continuous state and action spaces. In order to construct a control policy, the algorithm utilizes the entire history of agent-environment interaction. The policy is a result of an estimation process based on all available information rather than result of stochastic convergence as in classical reinforcement learning approaches. The policy is derived from the history directly, not through any kind of a model of the environment.**
We test our algorithm in the Cart-Pole Swing-Up simulated environment. The algorithm learns to control this plant in about 100 trials, which corresponds to 15 minutes of plant's real time. This time is several times shorter than the one required by other algorithms.
Keywords: reinforcement learning, importance sampling, model-free control

I. Introduction

The most popular algorithms of Reinforcement Learning (RL) such as *Q-Learning* [13] and *actor-critic* methods [1], [12], [4] are implicitly based on two assumptions:

A1. The model of the agent's environment is initially unknown.
A2. The agent can only utilize consecutive events to incrementally change of some its parameters. The only "memory" of the past events available to the agent is very compactly represented, e.g. is in the form of *eligiblity trace*.

Commercial importance of such algorithms is rather little. On one hand, the first assumption prevents them to be utilized in simulations. One must know the model to build the simulator. What is then the sense of the assumption that the model is unknown? On the other hand, utilization of consecutive steps for incremental adjustments makes a very inefficient way of information processing, and the learning is slow. To give an illustration, suppose a neural network is employed in such algorithm and the algorithm is to learn to control a plant. How many examples must be the network fed to be sufficiently well trained? Obviously, this number is not expressed in thousands but rather in hundreds of thousand. The same network could, however, be taught appropriate function from maybe hundreds of training examples, yet processed repeatedly.

Usually an RL algorithm adjusts its parameters on the basis of the consecutive agent's steps. It is, however, not the only possible approach. DYNA, introduced by Sutton in [10] explores the dynamics of the environment to build its model. Parallelly, the model is utilized to build a control policy with the use of asynchronous dynamic programming (DP). In [6], this architecture is enchanced by the *prioritized sweeping* as an efficient way to implement asynchronous DP. The algorithms [10] and [6] are based on assumptions different than A1 and A2, namely

1. The model of the environment is initially unknown.
2. The entire history of agent-environment interactions can be stored and utilized in some computational process.

We consider these new assumptions more proper than A1 and A2 to build a reinforcement learning algorithm for use in adaptive control of physical devices. Our approach is based on 1 and 2.

Building a model of the environment and its usage with some form of dynamic programming may be satisfying in the case of finite state and action spaces. In such a setting, the model achieved this way can be precise. In the case of continuous environment, the precise model is usually impossible to obtain and the very idea of determining the policy directly from the experience is tempting. Such problems have already been approached, see e.g. in [5]. In this paper we follow the same direction, however a solution we provide is different.

To show our motivation, let us discuss the following problem. We are given samples $X_1, X_2, \ldots$ from some unknown scalar distribution. After each i-th sample, we are to provide an approximation m_i of the median of the distribution. One way is to employ the stochastic approximation, namely

$$m_i = m_{i-1} + \beta_i \operatorname{sign}(X_i - m_{i-1})$$

where $\{\beta_i\}$ is a decreasing sequence which satisfies some standard conditions.

Another way is to employ an estimation, namely to take $\lceil i/2 \rceil$-th highest value from among $X_1, \ldots, X_i$. Obviously, the second way provides better approximations. It, however, requires to remember the entire history of drawing and is more computationally expensive.

In this paper we continue a work begun in [15] and introduce an algorithm that utilizes the entire known history of interactions between an agent and its environment. We argue

in [15] that our algorithm is in a way similar to the ones discussed in [12], [4], however, it utilizes estimation, instead of stochastic approximation.

The paper is organized as follows. In Section II we discuss actor and critic components of the algorithm we introduce in this paper. We also formulate the problem that the algorithm solves. In Section III we discuss the reinforcement learning issue as estimation problem. In Section IV we recall the importance sampling and utilize it in Section V in off-line evaluation and optimization a randomized policy. In Section VI we introduce the Intensive Randomized Policy Optimizer (IRPO) algorithm. Sections VII and VIII are devoted to details of IRPO implementation. In Section IX our algorithm is applied to control of the Cart-Pole Swing-Up.

II. An actor-critic algorithm

We discuss a discounted Reinforcement Learning problem [11] with continuous (possibly multi-dimensional) states $s \in \mathcal{S}$, continuous (possibly multi-dimensional) actions $u \in \mathcal{U}$, rewards $r \in \mathbb{R}$, and discrete time $t \in \{1, 2, \dots\}$.

A. Actor

At each state s the action u is drawn from the density $\varphi(u, \theta)$. This density is parameterized by the vector $\theta \in \mathbf{\Theta} \subset \mathbb{R}^m$ whose value is determined by parametric approximator $\widetilde{\theta}(s; \mathbf{w}_\theta)$. The approximator is parameterized with the weight vector $\mathbf{w}_\theta$. We assume that φ satisfy the following conditions:
a) $\varphi(u, \theta) > 0$ for $u \in \mathcal{U}, \theta \in \mathbf{\Theta}$,
b) for every $u \in \mathcal{U}$, the mapping $\theta \mapsto \ln \varphi(u, \theta)$ is continuous and differentiable.

For example, $\varphi(u, \theta)$ can be the normal density with the mean θ and a constant variance whereas $\widetilde{\theta}(s, \mathbf{w}_\theta)$ can be a neural network. In this example, the output of the network determines a center of the distribution the action is drawn from.

For the given φ and $\widetilde{\theta}$, the discussed action selection mechanism forms a policy that depends only on $\mathbf{w}_\theta$. We denote this policy by $\pi(\mathbf{w}_\theta)$. For the fixed $\mathbf{w}_\theta$, the sequence of states $\{s_t\}$ forms a Markov chain. Suppose $\{s_t\}$ has the stationary distribution $\eta(s, \mathbf{w}_\theta)$.

To determine the objective, let us define the value function of a generic policy π:

$$V^\pi(s) = \mathcal{E}\left(\sum_{i \geq 0} \gamma^i r_{t+1+i} \Big| s_t = s; \pi\right)$$

The ideal but not necessary realistic objective is to find $\mathbf{w}_\theta$ that maximizes $V^{\pi(\mathbf{w}_\theta)}(s)$ for each s. The realistic objective is to maximize the averaged value function, namely

$$\Phi(\mathbf{w}_\theta) = \int\limits_{s \in \mathcal{S}} V^{\pi(\mathbf{w}_\theta)}(s) \, \mathrm{d}\eta(s, \mathbf{w}_\theta)$$

B. Critic

In general, actor-critic algorithms employ some estimators of $\nabla\Phi(\mathbf{w}_\theta)$ to maximize $\Phi(\mathbf{w}_\theta)$. In order to construct such the estimators, we employ an approximator $\widetilde{V}(s; \mathbf{w}_V)$ of the value function $V^{\pi(\mathbf{w}_\theta)}(s)$ of the current policy. The approximator (e.g., a neural network) is parameterized by the weight vector $\mathbf{w}_V$. For approximator $\widetilde{V}$ to be useful in policy improvement, it should minimize the mean-square error

$$\Psi(\mathbf{w}_V, \mathbf{w}_\theta) = \int\limits_{s \in \mathcal{S}} \left(V^{\pi(\mathbf{w}_\theta)}(s) - \widetilde{V}(s; \mathbf{w}_V)\right)^2 \mathrm{d}\eta(s, \mathbf{w}_\theta)$$

with respect to $\mathbf{w}_V$.

The action-value function $Q^\pi : \mathcal{S} \times \mathcal{U} \mapsto \mathbb{R}$ is typically defined as the expected value of future discounted rewards the agent may expect starting from the state s, performing the action u, and following the policy π afterwards [13]:

$$Q^\pi(s, u) = \mathcal{E}\left(r_{t+1} + \gamma V^\pi(s_{t+1}) \Big| s_t = s, u_t = u\right) \tag{1}$$

We are interested in parameters that govern action selection, rather than particular actions. Let us define the pre-action-value function $U^\pi : \mathcal{S} \times \mathbf{\Theta} \mapsto \mathbb{R}$, as the expected value of future discounted rewards the agent may expect starting from the state s and performing an action drawn from the distribution characterized by the parameter θ, and following the policy π afterwards [14]:

$$\begin{aligned} U^\pi(s, \theta) &= \mathcal{E}\left(r_{t+1} + \gamma V^\pi(s_{t+1}) \Big| s_t = s; u_t \sim \varphi(., \theta)\right) \\ &= \mathcal{E}_\theta Q^\pi(s, \mathbf{Y}) \end{aligned} \tag{2}$$

where by $a \sim \varphi$ we mean that a has distribution φ, and $\mathcal{E}_\theta$ denotes the expected value calculated for random vector $\mathbf{Y}$ drawn from $\varphi(., \theta)$. Note that by definition

$$V^{\pi(\mathbf{w}_\theta)}(s) = U^{\pi(\mathbf{w}_\theta)}\left(s, \widetilde{\theta}(s; \mathbf{w}_\theta)\right).$$

Summing up our introduction to the problem, we intend to find $\mathbf{w}_\theta$ that maximizes

$$\Phi(\mathbf{w}_\theta) = \int\limits_{s \in \mathcal{S}} U^{\pi(\mathbf{w}_\theta)}\left(s, \widetilde{\theta}(s; \mathbf{w}_\theta)\right) \mathrm{d}\eta(s, \mathbf{w}_\theta) \tag{3}$$

which requires a solution of the auxiliary problem of minimization of

$$\Psi(\mathbf{w}_V, \mathbf{w}_\theta) = \int\limits_{s \in \mathcal{S}} \left(U^{\pi(\mathbf{w}_\theta)}\left(s, \widetilde{\theta}(s; \mathbf{w}_\theta)\right) - \widetilde{V}(s; \mathbf{w}_V)\right)^2 \mathrm{d}\eta(s, \mathbf{w}_\theta) \tag{4}$$

with respect to $\mathbf{w}_V$.

III. The reinforcement learning problem as a problem of estimation

The algorithm we introduce in this paper, consists of two activities performed simultaneously:

1) Exploration of the environment by performing consecutive actions based on the current policy $\pi(\mathbf{w}_\theta)$.
2) Approximation of the policy iteration, which realizes both elements of actor-critic schemes, namely
   Policy evaluation (critic training). Adjustment of $\mathbf{w}_V$ to minimize an estimate $\widehat{\Psi}_t(\mathbf{w}_V, \mathbf{w}_\theta)$ of $\Psi(\mathbf{w}_V, \mathbf{w}_\theta)$ based on all events up to the current step t.
   Policy improvement (actor training). Adjustment of $\mathbf{w}_\theta$

to maximize an estimate $\widehat{\Phi}_t(\mathbf{w}_\theta, \mathbf{w}_V)$ of $\Phi(\mathbf{w}_\theta)$ based on all events up to the current step t.

The policy employed in step (1) is the one repeatedly modified by the process (2).

The policy is formed by the proces in a way similar to the maximum likehood estimation in which one looks for the parameter that maximizes the probability that the given data wolud be generated. Here, we look for the parameters most plausible to generate the data we would like to draw.

In order to construct $\widehat{\Phi}_t(\mathbf{w}_\theta, \mathbf{w}_V)$, we treat all previous states s_i as drawn from $\eta(., \mathbf{w}_\theta)$ and approximate the integral with the average value

$$\widehat{\Phi}_t(\mathbf{w}_\theta, \mathbf{w}_V) = \frac{1}{t}\sum_{i=1}^{t} \widehat{U}(s_i, \widetilde{\theta}(s_i; \mathbf{w}_\theta)) \tag{5}$$

Here $\widehat{U}$ is an estimator of $U^{\pi(\mathbf{w}_\theta)}$. It somehow utilizes $\widetilde{V}$ (otherwise the critic would be useless) and hence $\widehat{\Phi}_t$ depends also on $\mathbf{w}_V$.

In the same way we construct $\widehat{\Psi}_t$, namely

$$\widehat{\Psi}_t(\mathbf{w}_V, \mathbf{w}_\theta) = \frac{1}{t}\sum_{i=1}^{t} \widehat{e_i^2} \tag{6}$$

where $\widehat{e_i^2}$ is an estimator of

$$\left(U^{\pi(\mathbf{w}_\theta)}\big(s_i, \widetilde{\theta}(s_i; \mathbf{w}_\theta)\big) - \widetilde{V}(s_i; \mathbf{w}_V)\right)^2.$$

Particular forms of $\widehat{U}$ and $\widehat{e_i^2}$ may be based on importance sampling [9] and the ideas developed by Precup *et al.* in e.g. [7], [8]. We recall some basic results of importance sampling.

IV. Importance sampling

Suppose we are given the triple $\langle \theta_0, \mathbf{Y}_0, f(\mathbf{Y}_0)\rangle$ where $\mathbf{Y}_0$ has been drawn from the density $\varphi(., \theta_0)$. Two problems are of interest to us.

The first problem is to estimate $\mathcal{E}_\theta f(\mathbf{Y})$ for a given θ. The quantity

$$f(\mathbf{Y}_0)\frac{\varphi(\mathbf{Y}_0, \theta)}{\varphi(\mathbf{Y}_0, \theta_0)} \tag{7}$$

is an unbiased estimator of $\mathcal{E}_\theta f(\mathbf{Y})$, since

$$\begin{aligned}\mathcal{E}_{\theta_0}\left(f(\mathbf{Y})\frac{\varphi(\mathbf{Y}, \theta)}{\varphi(\mathbf{Y}, \theta_0)}\right) &= \int f(\mathbf{z})\frac{\varphi(\mathbf{z}, \theta)}{\varphi(\mathbf{z}, \theta_0)}\varphi(\mathbf{z}, \theta_0)\,\mathrm{d}\mathbf{z} \\ &= \int f(\mathbf{z})\varphi(\mathbf{z}, \theta)\,\mathrm{d}\mathbf{z} \\ &= \mathcal{E}_\theta f(\mathbf{Y})\end{aligned} \tag{8}$$

The estimator (7) is unbiased, yet its variance can be large when the distributions $\varphi(., \theta)$ and $\varphi(., \theta_0)$ differ too much. Bounding the variance is an issue of importance. A derivation similar to (8) suggests that

$$a + (f(\mathbf{Y}_0) - a)\frac{\varphi(\mathbf{Y}_0, \theta)}{\varphi(\mathbf{Y}_0, \theta_0)} \tag{9}$$

is also an unbiased estimator of $\mathcal{E}_\theta f(\mathbf{Y})$ for each constant (non-random) reference point a. The question arise, how to choose the reference point. It seems reasonable to take a equal to some preliminary assessment of $\mathcal{E}_\theta f(\mathbf{Y})$. If this assessment is correct, it bounds the absolute value of $f(\mathbf{Y}_0) - a$ for large values of the fraction $\varphi(\mathbf{Y}_0, \theta)/\varphi(\mathbf{Y}_0, \theta_0)$ and consequently confines the variance of (9). Another method of suppressing the variance is to replace the division of densities with

$$\rho_b(\mathbf{Y}_0, \theta, \theta_0) = \min\left\{\frac{\varphi(\mathbf{Y}_0, \theta)}{\varphi(\mathbf{Y}_0, \theta_0)}, b\right\} \tag{10}$$

where b is a number greater then 1, equal to, say, 5. We obtain the estimator

$$a + (f(\mathbf{Y}_0) - a)\rho_b(\mathbf{Y}_0, \theta, \theta_0) \tag{11}$$

which is, however, biased.

The second problem is to estimate $\mathcal{E}_\theta(f(\mathbf{Y}) - c)^2$ for given θ and c. By applying the same argument as previously, we obtain that

$$(f(\mathbf{Y}_0) - c)^2\frac{\varphi(\mathbf{Y}_0, \theta)}{\varphi(\mathbf{Y}_0, \theta_0)} \tag{12}$$

is an estimator we look for, and it is unbiased. Another estimator

$$(f(\mathbf{Y}_0) - c)^2\rho_b(\mathbf{Y}_0, \theta, \theta_0) \tag{13}$$

has smaller variance, yet it is biased.

V. Off-line evaluation and optimization of randomized policy

We now implement the importance sampling to construct $\widehat{\Psi}_t$ and $\widehat{\Phi}_t$, which are the tools that allow us for off-line evaluation and optimization of the policy $\pi(\mathbf{w}_\theta)$.

We are given the history of the agent-environment interaction up to the moment t: the states visited $\{s_i, i = 1, \ldots, t\}$, the rewards received $\{r_{i+1}, i = 1, \ldots, t\}$, and the control actions that have been performed, namely $\{u_i, i = 1, \ldots, t\}$. The actions have been drawn according to $\varphi(., \theta_i)$ where $\{\theta_i, i = 1, \ldots, t\}$ are given.

We will utilize the derivations from the previous section. Let us denote

$$q_i = r_{i+1} + \gamma\widetilde{V}(s_{i+1}; \mathbf{w}_V) \tag{14}$$

Let us now establish relations between generic terms from the previous section and the formulation of our problem.

1) $\mathbf{Y}_0 \leftrightarrow u_i$, the drawing,
2) $\theta_0 \leftrightarrow \theta_i$, the parameter used for drawing,
3) $f(\mathbf{Y}_0) \leftrightarrow Q^{\pi(\mathbf{w}_\theta)}(s_i, u_i)$, the return; $Q^{\pi(\mathbf{w}_\theta)}(s_i, u_i)$ is estimated by q_i,
4) $\mathcal{E}_\theta f(\mathbf{Y}) \leftrightarrow U^{\pi(\mathbf{w}_\theta)}(s_i, \widetilde{\theta}(s_i; \mathbf{w}_\theta))$, the value we want to estimate and maximize.

A. Evaluation

In order to construct $\widehat{\Psi}_t$ (6), we need the statistics $\widehat{e_i^2}$. Implementing (13), we introduce

$$\widehat{e_i^2} = \big(q_i - \widetilde{V}(s_i; \mathbf{w}_V)\big)^2\rho_b(a_i, \widetilde{\theta}(s_i; \mathbf{w}_\theta), \theta_i).$$

Hence, in order to achieve the mean-squared approximation of $V^{\pi(\mathbf{w}_\theta)}$, we minimize

$$\widehat{\Psi}_t(\mathbf{w}_V, \mathbf{w}_\theta) = \frac{1}{t}\sum_{i=1}^{t}\left(q_i - \widetilde{V}(s_i;\mathbf{w}_V)\right)^2 \rho_b(a_i, \widetilde{\theta}(s_i;\mathbf{w}_\theta), \theta_i) \tag{15}$$

with respect to $\mathbf{w}_V$. Note that $\widetilde{V}$ is employed to calculate q_i and, as a consequence, q_i should be recalculated after each step of the minimization.

B. Optimization

To construct $\widehat{\Phi}_t$ (5), we need to introduce $\widehat{U}(s_i, \widetilde{\theta}(s_i;\mathbf{w}_\theta))$, as an estimator of $U^{\pi(\mathbf{w}_\theta)}(s_i, \widetilde{\theta}(s_i;\mathbf{w}_\theta))$. We employ the estimator based on (11), taking $\widetilde{V}(s_i;\mathbf{w}_V)$ as the reference point, namely

$$\widehat{U}(s_i,\widetilde{\theta}(s_i;\mathbf{w}_\theta)) = \widetilde{V}(s_i;\mathbf{w}_V) + (q_i - \widetilde{V}(s_i;\mathbf{w}_V))\rho_b(u_i, \widetilde{\theta}(s_i;\mathbf{w}_\theta), \theta_i).$$

We thus obtain:

$$\widehat{\Phi}_t(\mathbf{w}_\theta, \mathbf{w}_V) = \frac{1}{t}\sum_{i=1}^{t}\widetilde{V}(s_i;\mathbf{w}_V) + (q_i - \widetilde{V}(s_i;\mathbf{w}_V))\times \times \rho_b(u_i, \widetilde{\theta}(s_i;\mathbf{w}_\theta), \theta_i) \tag{16}$$

During the maximization, $\widetilde{\theta}(s_i;\mathbf{w}_\theta)$ should be kept within some bounded area for $i = 1, \ldots, t$. Such the constraint is necessary, otherwise the solution might not have a physical sense, since the best policy could appear to be the one not explored so far, e.g. the one with infinite parameters.

Let us recall the *temporal difference*:

$$d_i = r_{i+1} + \gamma\widetilde{V}(s_{i+1};\mathbf{w}_V) - \widetilde{V}(s_i;\mathbf{w}_V)$$

and rewrite (16) as

$$\widehat{\Phi}_t(\mathbf{w}_\theta, \mathbf{w}_V) = \frac{1}{t}\sum_{i=1}^{t} d_i\rho_b(u_i, \widetilde{\theta}(s_i;\mathbf{w}_\theta), \theta_i) + \overline{V} \tag{17}$$

where $\overline{V}$ does not depend on $\mathbf{w}_\theta$. Maximization of (17) with respect to $\mathbf{w}_\theta$ optimizes the first step along each trajectory. It changes the policy $\pi(\mathbf{w}_\theta)$ whose value function $V^{\pi(\mathbf{w}_\theta)}$ should be approximated again. Generally speaking, minimizing of (15) and maximizing of (17) are mutually dependent optimization tasks. The optimization procedure is thus associated with problems similar to the ones associated with the classical policy iteration.

VI. Intensive Randomized Policy Optimizer

The discussion above leads to the algorithm of reinforcement learning based on estimation which we call the Intensive Randomized Policy Optimizer (IRPO). The IRPO algorithm uses two parametric approximators: $\widetilde{\theta}$ as a base for randomized control policy $\pi(\mathbf{w}_\theta)$ and $\widetilde{V}$ to approximate the value function $V^{\pi(\mathbf{w}_\theta)}$. The algorithm is comprised of two loops: (i) an *exploration loop* in which the agent interacts with its environment and collects history of this interaction, (ii) an *internal loop* in which the agent optimizes its policy on the basis of the events experienced so far.

The exploration loop of IRPO consists of the following steps:

1) Draw the control action u_t:
$$u_t \sim \varphi(.,\widetilde{\theta}(s_t;\mathbf{w}_\theta))$$
where $\mathbf{w}_\theta$ is a vector calculated in the internal loop.
2) Perform control u_t, and observe the next state s_{t+1} and the reinforcement r_{t+1}.
3) Add the quintet $\langle s_t, u_t, r_{t+1}, s_{t+1}, \widetilde{\theta}(s_t;\mathbf{w}_\theta)\rangle$ to the dataset; $\mathbf{w}_\theta$ is the same value as served to action selection.
4) Set $t := t + 1$ and repeat from Step 1.

The following optimization tasks are performed by the algorithm in its internal loop until convergence:

1) Policy evaluation. Adjust $\mathbf{w}_V$ for $\widetilde{V}(s_i;\mathbf{w}_V)$ to approximate $V^{\pi(\mathbf{w}_\theta)}(s_i)$ for all $i \in \{1, \ldots, t\}$, i.e. to minimize $\widehat{\Psi}_t(\mathbf{w}_V, \mathbf{w}_\theta)$ (15).
2) Policy improvement. Adjust $\mathbf{w}_\theta$ to maximize an estimator of $U^{\pi}(s_i, \widetilde{\theta}(s_i;\mathbf{w}_\theta))$ for all $i \in \{1, \ldots, t\}$ and fixed $\pi = \pi(\mathbf{w}_\theta)$, i.e. to maximize $\widehat{\Phi}_t(\mathbf{w}_\theta, \mathbf{w}_V)$ (17).

It is not necessary to perform a full optimization in neither of the two steps.

Note that the two steps of the internal loop perform the approximate policy iteration: the first step calculates the value function of the current policy and the second one optimizes the first step along each trajectory.

The algorithm does not use a model of the environment. It utlize agent's experience directly to build the policy. It is thus a *direct adaptive control* method.

VII. Implementation of the internal loop

On the first sight, the implementation of IRPO may seem to be quite complicated. It encompasses two mutually dependent optimization processes whose complexity increase in time. Our experiments prove that the algorithm is feasible, provided certain implementational issues are taken care of.

The approach we may recommend the most employs optimizations of $\widehat{\Psi}_t$ and $\widehat{\Phi}_t$ with the use of stochastic approximation. Note that both functions are constructed as certain sums. The optimization of such sums may proceed in the loop as follows:

I1. Pick a random $i \in \{1, \ldots, t\}$.
I2. Adjust $\mathbf{w}_V$ along the gradient of i-th component of $\widehat{\Psi}_t$ on $\mathbf{w}_V$.
I3. Adjust $\mathbf{w}_\theta$ along the gradient of i-th component of $\widehat{\Phi}_t$ on $\mathbf{w}_\theta$.

It is suggested that the consecutive i-s form random combinations of all numbers in $\{1, \ldots, t\}$.

If $\widetilde{V}$ and $\widetilde{\theta}$ are implemented as neural networks, the optimization problems discussed above are special version of the issue how to train the neural network when the training set increases due to generation of new data. We may choose from among classical optimization methods or stochastic approximation. Taking into account that the final training

set can be very large (it contains entire history of agent-environment interaction), the recommended choice seems to be the stochastic approximation. Additionally, when neural networks are employed, it is wise to start the internal loop when some data is already collected. Otherwise the networks could get stuck in some local minima.

We employ the above remarks in the algorithm presented below. In the internal loop, steps I1, I2, I3 are repeated n times after every step of the exploration loop.

VIII. Examples of valid distributions

We will now discuss the selection of φ. Generally speaking, the choice of the family of densities φ that governs the action selection should depend on what kind of randomization is acceptable and what kind of exploration seems fruitful in the particular reinforcement learning problem. We present two examples illustrating the possible choices.

A. Normal distribution

The control selection mechanism may be designed as follows. $\mathcal{U} = \mathbf{\Theta} = \mathbb{R}^m$ and the action in each state is a sum of the actor's output and a zero-mean normal noise. The noise is necessary for exploration and optimization of the actor. In this case φ is a family of normal distributions $N(\theta, \mathbf{C})$ of constant covariance matrix equal to $\mathbf{C}$, parameterized by the expected value θ:

$$\varphi(u,\theta) = \frac{1}{\sqrt{2\pi|\mathbf{C}|}} \exp\left(-\frac{1}{2}(u-\theta)^T \mathbf{C}^{-1}(u-\theta)\right) \quad (18)$$

The "larger" $\mathbf{C}$ we take, the faster will be the convergence and the poorer the final performance. By manipulating $\mathbf{C}$, we control a balance between the exploration and the exploitation of the environment.

B. Log-normal distribution

Suppose that the actions are positive real numbers. Suppose also, that varying the actions within some *relative* distance from their optimal values does not affect the performance crucially. Let σ (equal to, say, 0.01) be the required relative accuracy.

In this case we may use $\mathbf{\Theta} = \mathbb{R}$ and a family of log-normal distributions

$$\varphi(u,\theta) = \frac{1}{\sigma a\sqrt{2\pi}} \exp\left(-\frac{(\ln u - \theta)^2}{2\sigma^2}\right)$$

To sample according to this density, one may simply take $a = \exp Y$ where Y is drawn from the normal distribution $N(\theta, \sigma^2)$ of density (18).

Extending the discussion obove to the case of multidimensional actions and θ is straightforward.

IX. Illustration: Cart-Pole Swing-Up

In this section we present an experimental study of *Intensive Randomized Policy Optimizer* applied to a problem of reinforcement learning with continuous state and action spaces.

As a platform for illustration, we chose the *Cart-Pole Swing-Up* [3], which is a modification of the *inverted pendulum* frequently used as a benchmark for reinforcement learning algorithms. There are four state variables: position of the cart x, arc of the pole ω, and time derivatives $\dot{x}$, $\dot{\omega}$. There is a single control variable, which is force applied to the cart F. Dynamics of the Cart-Pole Swing-Up, is the same as that of the inverted pendulum described e. g. in [1].

The reward in this problem is typically equal to the elevation of the pole top. Initially, the waving pole hovers, and the rewards are close to -1. The goal of control is to avoid hitting the track bounds, swing the pole, turn it up and stabilize upwards. Rewards are then close to 1. Controller's interventions take place every 0.1s.

Note that τ here is the continuous time of the plant which remains in a simple relation with the discrete time t of the controller.

The force $F(\tau)$ is calculated from the action u_t as

$$F(\tau) = \begin{cases} -10 & \text{if } u_t < -10 \\ u_t & \text{if } u_t \in [-10, 10] \\ 10 & \text{otherwise} \end{cases}$$

The action u_t is determined every 0.1 second.

State of the plant is *acceptable* if and only if $x(\tau) \in [-2.4, 2.4]$. If the next state is not acceptable then the reinforcement is set to $-30 - 0.2\big|u_t - F(\tau)\big|$. Otherwise, the reinforcement is determined as

$$r(\tau) = -0.2\,\big|u_t - F(\tau)\big| + \begin{cases} \cos\omega(\tau) & \text{if } |\dot{\omega}(\tau)| < 2\pi \\ -1 & \text{otherwise} \end{cases}$$

The learning process consists of a sequence of *trials*. The trial may end in one of two possible situations. First, the pendulum's state may become unacceptable. It is then associated with an appropriately low reinforcement. Otherwise, the trial lasts for a random number of steps drawn from the geometric distribution with expected value equal to 300 (corresponding to 30 sec. of real time). The trial begins with a state reset, which consists of drawing $x(\tau)$ and $\omega(\tau)$ from the uniform distributions $U(-2.4, 2.4)$ and $U(0, 2\pi)$, respectively, and setting $\dot{x}(\tau)$, $\dot{\omega}(\tau)$ to zero.

Each approximator employed by the algorithm was implemented as a two layer perceptron with sigmoidal (arctan) activation function in the hidden layer and the linear output layer. Each neuron had a constant input (bias). The initial weights of the hidden layer were drawn randomly from the normal distribution $N(0, 0.2)$ and the initial weights of the output layer were set to zero.

The discount factor γ was set to 0.95. The state vector s_t feeding the approximators was normalized, namely

$$s_t = \left[\frac{x(\tau)}{2}, \frac{\dot{x}(\tau)}{3}, \frac{\sin\omega(\tau)}{0.8}, \frac{\cos\omega(\tau)}{0.8}, \frac{\dot{\omega}(\tau)}{4}\right]^T$$

The exact formulation of the algorithm in use is as follows. The formulation constrains both the exploration loop and the internal loop. We employed the normal family of distributions (18).

1) Draw the control action u_t:

$$u_t \sim \varphi(.\,, \widetilde{\theta}(s_t; \mathbf{w}_\theta))$$

TABLE I

IRPO IN COMPARISON TO KNOWN ALGORITHMS.

Algorithm	no of trials	real time
Intensive Randomized Policy Optimizer	100	15 min.
A model based real-time actor-critic method [3]	700	2 hours
Standard model-free actor-critic methods [3], [14]	2500	6 hours
Action-Dependent Heuristic Dynamic Programming [14]	30000	3 days

Data in columns 2 and 3 express the number of trials and the real time of the plant the algorithms require to adapt the control to the unknown plant.

2) Perform control u_t, and observe the next state s_{t+1} and the reinforcement r_{t+1}
3) Add the five $\langle s_t, u_t, r_{t+1}, s_{t+1}, \widetilde{\theta}(s_t; \mathbf{w}_\theta)\rangle$ to the dataset.
4) (The internal loop) [1] If $t > 422$, repeat n times:
   a) Calculate $d_i = r_{t_i+1} - \widetilde{V}(s_{t_i}; \mathbf{w}_V)$ if s_{t_i+1} was not acceptable and $d_i = r_{t_i+1} + \gamma \widetilde{V}(s_{t_i+1}; \mathbf{w}_V) - \widetilde{V}(s_{t_i}; \mathbf{w}_V)$ otherwise.
   b) Adjust the approximation of value function:

$$\mathbf{w}_V := \mathbf{w}_V + \beta_i^V d_i \rho_b(u_{t_i}, \widetilde{\theta}(s_{t_i}; \mathbf{w}_\theta), \theta_{t_i}) \times \times \frac{\mathrm{d}\widetilde{V}(s_{t_i}; \mathbf{w}_V)}{\mathrm{d}\mathbf{w}_V}$$

   c) Adjust the policy:

$$g_i = G\left(\frac{\mathrm{d}\rho_b(u_{t_i}, \widetilde{\theta}(s_{t_i}; \mathbf{w}_\theta), \theta_{t_i})}{\mathrm{d}\widetilde{\theta}(s_{t_i}; \mathbf{w}_\theta)}, \widetilde{\theta}(s_{t_i}; \mathbf{w}_\theta)\right)$$

$$\mathbf{w}_\theta := \mathbf{w}_\theta + \beta_i^\theta d_i \frac{\mathrm{d}\widetilde{\theta}(s_{t_i}; \mathbf{w}_\theta)}{\mathrm{d}\mathbf{w}_\theta} g_i$$

   d) Set $i := i + 1$.
5) Set $t := t + 1$ and repeat from Step 1.

$\{t_i\}$ is a random sequence of time indexes currently available in the dataset (it is a "better" form of simple drawing a random sample from the data). Function G is used to keep $\widetilde{\theta}(s_t; \mathbf{w}_\theta)$ within a bounded area:

$$G(g, \theta) = \begin{cases} g & \text{if } \theta \in [-10, 10] \\ \max\{g, 0.2\} & \text{if } \theta < -10 \\ \min\{g, -0.2\} & \text{otherwise} \end{cases}$$

The remaining parameters are as follows:
$M^\theta = 20$ — number of hidden neurons of $\widetilde{\theta}$.
$M^V = 40$ — number of hidden neurons of $\widetilde{V}$.
$\beta_i^\theta = \beta_i^V \equiv 0.001$ — learning rates of $\widetilde{\theta}$ and $\widetilde{V}$.
$\sigma^2 = 4$ — the variance of φ.
$b = 5$ — the upper bound of ρ.
$n = 1000$ — the numer of internal loop steps per a single step of the exploration loop. $n = 1000$ was feasible in our simulations (PC with Athlon$^{\text{TM}}$1400 MHz) to be caried in real time of the plant.

The IRPO algorithm achieved a satisfactory behavior after about 100 trials which was equivalent to about 15 minutes of the real time of the plant (see Table I).

[1] There are 422 weights in both networks.

X. CONCLUSIONS AND FURTHER WORK

The algorithm of reinforcement learning in continuous space of states and actions introduced in this paper seems to be powerful enough to tackle adaptive control tasks in real time. Its high speed of convergence is a consequence of calculations of the policy directly from the history of agent-environment interactions. No explicit model of the plant/environment is built, instead, the algorithm developes the control policy directly from available observations, thus realizing a direct control scheme.

Most algorithms of reinforcement learning suppress the randomization as the learning continues. This of course is also possible in IRPO. Extension of the presented methodology to incorporate a decreasing exploration is a topic of our current research.

REFERENCES

[1] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike Adaptive Elements That Can Learn Difficult Learning Control Problems, " *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, pp. 834-846, Sept.-Oct. 1983.

[2] R. Coulom, *Reinforcement Learning Using Neural Networks, with Applications to Motor Control*, PhD thesis, Institut National Polytechnique de Grenoble, 2002.

[3] K. Doya, "Reinforcement learning in continuous time and space," *Neural Computation*, 12:243-269, 2000.

[4] V. R. Konda and J. N. Tsitsiklis, "Actor-Critic Algorithms," *SIAM Journal on Control and Optimization*, Vol. 42, No. 4, pp. 1143-1166, 2003.

[5] M. G. Lagoudakis and R. Paar, "Model-free least-squares policy iteration," *Advances in Neural Information Processing Systems*, volume 14, 2002.

[6] A. W. Moore and C. G. Atkeson, "Prioritized Sweeping: Reinforcement Learning with Less Data and Less Real Time," *Machine Learning*, Vol. 13, October, 1993.

[7] D. Precup, R. S. Sutton, S. Singh, "Eligibility Traces for Off-Policy Policy Evaluation," *Proceedings of the 17th International Conference on Machine Learning*, Morgan Kaufmann, 2000.

[8] D. Precup, R. S. Sutton, S. Dasgupta, "Off-policy temporal-difference learning with function approximation," *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.

[9] R. Rubinstein, *Simulation and the monte carlo method*. New York, Wiley, 1981.

[10] R. S. Sutton, "Integrated Architectures For Learning, Planning, and Reacting Based on Approximating Dynamic Programming," *Proceedings of the Seventh Int. Conf. on Machine Learning*, pp. 216-224, Morgan Kaufmann, 1990.

[11] R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 1998.

[12] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy Gradient Methods for Reinforcement Learning with Function Approximation," *Advances in Information Processing Systems 12*, pp. 1057-1063, MIT Press, 2000.

[13] C. Watkins and P. Dayan, "Q-Learning," *Machine Learning*, vol. 8, pp. 279-292, 1992.

[14] P. Wawrzynski, A. Pacut, "A simple actor-critic algorithm for continuous environments," submitted for publication, available at http://home.elka.pw.edu.pl/∼pwawrzyn, 2003.

[15] P. Wawrzynski, A. Pacut, "Intensive versus non-intensive actor-critic reinforcement learning algorithms," submitted for publication, available at http://home.elka.pw.edu.pl/∼pwawrzyn, 2004.