Appeared in *Proceedings of the 7th Int. Conf. on Artificial Intelligence and Soft Computing, Zakopane, Poland, June 2004,* pp. 934-941, © Springer-Verlag

Intensive versus non-intensive actor-critic reinforcement learning algorithms

Paweł Wawrzyński[1] and Andrzej Pacut *Senior Member, IEEE*[2]

Institute of Control and Computation Engineering
Warsaw University of Technology
00–665 Warsaw, Poland

[1] `P.Wawrzynski@elka.pw.edu.pl`, `http://home.elka.pw.edu.pl/~pwawrzyn`
[2] `A.Pacut@ia.pw.edu.pl`, `http://www.ia.pw.edu.pl/~pacut`

Abstract. Algorithms of reinforcement learning usually employ consecutive agent's actions to construct gradients estimators to adjust agent's policy. The policy is a result of some kind of stochastic approximation. Because of the slowness of stochastic approximation, such algorithms are usually much too slow to be employed, e.g. in real-time adaptive control. In this paper we analyze the replacing of the stochastic approximation with the estimation based on the entire available history of an agent-environment interaction. We design an algorithm of reinforcement learning in continuous space/action domain that is of orders of magnitude faster then the classical methods.
Keywords: reinforcement learning, model-free control, importance sampling

1 Introduction

The most popular algorithms of reinforcement learning such as *Q-Learning* [11] and actor-critic methods [1, 10, 3] are constructed as a certain loop. A single step of the loop comprises (i) observing the agent's state, (ii) suggesting an action to perform, (iii) observing the consequence of the action, (iv) performing some finite (usually very small) set of operations on algorithm's parameters. For example, if the algorithm uses a neural network, weights of the network are modified along a certain gradient. Convergence of such algorithms is then as slow as slow is the network's training. The slowness is not a problem when we think of reinforcement learning as a tool for determining the policy in simulation tasks. If, however, we think of reinforcement learning as a tool for adaptive control, the slowness becomes disturbing.

Obviously, the rationing of computational expense discussed above is not the only possible approach. In [8], Sutton introduces an architecture that is much more computationally intensive. The architecture explores the dynamics of the environment to build its model. In the meantime, the model is utilized to design a policy with the use of asynchronous dynamic programming. In [5], the *prioritized sweeping* is employed to optimize such an approach.

Building the model of environment and its use with some form of dynamic programming may be satisfying in the case of finite state and action spaces. In such a setting, the resulting model can be precise. In the case of continuous environment, the precise model is usually impossible to obtain and the idea of determining the policy directly from the experience is tempting. Such the approach has already been utilized, see e.g. [4]. 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 $X_1, X_2, \dots$ 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 β_i is decreasing sequence which satisfies some standard conditions.

Another way is to employ a "batch" estimate, i.e. to take $\lceil i/2 \rceil$-th highest value among $X_1, \dots, X_i$. Obviously the second way provides better approximations. It, however, requires remembering the entire history of sampling and is more computationally expensive.

In this article, we analyze a simple actor-critic algorithm [10, 3] and discuss an issue of achieving a policy as a result of direct estimation, rather then as a result of stochastic approximation. The paper is devoted to a discussion of certain concepts, while the details of derivations are left to [13].

2 An actor-critic algorithm

We discuss a discounted Reinforcement Learning problem [9] with continuous (possibly multi-dimensional) states $s \in \mathcal{S}$, continuous (possibly multi-dimensional) actions $a \in \mathcal{A}$, rewards $r \in \mathbb{R}$, and discrete time $t \in \{1, 2, \dots\}$.

2.1 Actor

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

For example, $\varphi(a;\theta)$ may be the normal density with the mean equal to 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 which depends only on $\mathbf{w}_\theta$. We denote this policy by $\pi(\mathbf{w}_\theta)$. For each fixed $\mathbf{w}_\theta$, the sequence of states $\{s_t\}$ forms a Markov chain. Suppose $\{s_t\}$ has stationary distribution $\eta(s, \mathbf{w}_\theta)$.

To determine our objective, let us define in a standard manner the value function for a generic policy π, namely $V^\pi(s) = \mathcal{E}\left(\sum_{i\geq 0}\gamma^i r_{t+1+i}\middle| s_t = s; \pi\right)$. 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

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

2.2 Critic

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

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

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

The action-value function $Q^\pi : \mathcal{S}\times\mathcal{A} \mapsto \mathbb{R}$ is typically defined as the expected value of future discounted rewards the agent may expect starting from the state s, performing the action a, and following the policy π afterwards [11], namely

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

We are interested in the parameter that governs the action selection rather then the action itself. 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, performing an action drawn from the distribution characterized by the parameter θ, and following the policy π afterwards [12]:

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

where $\mathcal{E}_\theta$ denotes the expected value calculated for the 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, the considered problem is to find $\mathbf{w}_\theta$ that maximizes

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

This in turn requires to solve the auxiliary problem of minimization of

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

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

3 Two alternative approaches to the problem

Both approaches we discuss are based on *policy iteration*: they alternate two steps: (i) modification of $\mathbf{w}_\theta$ to optimize the first step of each trajectory, i.e. to maximize $U^{\pi(\mathbf{w}_\theta)}(s, \widetilde{\theta}(s; \mathbf{w}_\theta))$ with $\pi(\mathbf{w}_\theta)$ fixed and (ii) improvements of the estimates of $U^{\pi(\mathbf{w}_\theta)}$.

3.1 Non-intensive approach based on stochastic approximation

An actor-critic algorithm based on the standard (*non-intensive*, as it might be called) approach comprises the following operations at consecutive t:

1. Regard s_t as drawn from $\eta(., \mathbf{w}_\theta)$. Draw the control action $a_t \sim \varphi(.; \widetilde{\theta}(s_t; \mathbf{w}_\theta))$. Observe r_{t+1} and s_{t+1}.
2. *Policy improvement.* Modify $\mathbf{w}_\theta$ along the direction of some estimator of
$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{w}_\theta} U^\pi(s_t, \widetilde{\theta}(s_t; \mathbf{w}_\theta))$$
for fixed $\pi = \pi(\mathbf{w}_\theta)$.
3. *Policy evaluation.* Modify $\mathbf{w}_V$ along the direction of
$$\Big(v_t - \widetilde{V}(s_t; \mathbf{w}_V)\Big) \frac{\mathrm{d}\widetilde{V}(s_t, \mathbf{w}_V)}{\mathrm{d}\mathbf{w}_V}$$
where v_t is some „better" estimation of $V^{\pi(\mathbf{w}_\theta)}(s_t)$. In the simplest case, $v_t = r_{t+1} + \gamma \widetilde{V}(s_{t+1}; \mathbf{w}_V)$.

The estimator utilized in step 2. has the form
$$\frac{\mathrm{d}\widetilde{\theta}(s; \mathbf{w}_\theta)}{\mathrm{d}\mathbf{w}_\theta} g_t$$
where g_t is some estimator of
$$\frac{\mathrm{d}U(s_t, \widetilde{\theta}(s_t; \mathbf{w}_\theta))}{\mathrm{d}\widetilde{\theta}(s_t; \mathbf{w}_\theta)}$$
and U is in turn an approximation of $U^{\pi(\mathbf{w}_\theta)}$ for a fixed $\pi(\mathbf{w}_\theta)$. A way to construct the estimator g_t is known since the Williams' REINFORCE algorithm [14]. It is based on the following generic property:
$$\nabla_\theta \mathcal{E}_\theta f(\mathbf{Y}) = \mathcal{E}_\theta\big((f(\mathbf{Y}) - c)\nabla_\theta \ln \varphi(\mathbf{Y}; \theta)\big) \tag{5}$$
where $\mathbf{Y}$ is drawn from the distribution $\varphi(.; \theta)$. The equation holds for any function f and constant c if certain liberal regularity conditions are satisfied. Why is this property so important? We draw $\mathbf{Y}$ according to $\varphi(.; \theta)$ (i.e., perform the action a_t) and obtain the return $f(\mathbf{Y})$ (or an estimation of $Q^{\pi(\mathbf{w}_\theta)}(s_t, a_t)$). The

larger is the return, the higher is the need to change the parameter θ to make generating of this $\mathbf{Y}$ more plausible. Equation (5) states that in order to maximize $\mathcal{E}_\theta f(\mathbf{Y})$, θ should be adjusted along the direction of $(f(\mathbf{Y})-c)\nabla_\theta \ln\varphi(\mathbf{Y};\theta)$. An implementation in reinforcement learning is straightforward: the larger is the estimate of $Q^{\pi(\mathbf{w}_\theta)}(s_t,a_t)$, the larger should be the change of $\widetilde{\theta}(s_t;\mathbf{w}_\theta)$ to make the action a_t more plausible. The g_t estimator should be then something like

$$(Q(s_t,a_t)-c(s_t))\nabla\ln\varphi(a_t;\widetilde{\theta}(s_t;\mathbf{w}_\theta))$$

where $Q(s_t,a_t)$ is an estimator of $Q^{\pi(\mathbf{w}_\theta)}(s_t,a_t)$ and c is any function. In [12], we propose to employ:

$$g_t=(r_{t+1}+\gamma\widetilde{V}(s_{t+1};\mathbf{w}_V)-\widetilde{V}(s_t;\mathbf{w}_V))\nabla\ln\varphi(a_t;\widetilde{\theta}(s_t;\mathbf{w}_\theta))$$

which gives a satisfying behavior.

3.2 Intensive approach based on direct estimation

An algorithm based on the alternative approach comprises 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:
   (a) *Policy evaluation.* 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.
   (b) *Policy improvement.* 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.

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

Worth noting is a similarity between the above policy determination and the maximum likehood estimation. In the former, we look for the parameter that maximizes the probability of generating the available data. 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 replace the integral with the average value

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

where $\widehat{U}$ is an estimator of $U^{\pi(\mathbf{w}_\theta)}$. The estimator 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

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

where $\widehat{e_i^2}$ is an estimator of $\Big(U^{\pi(\mathbf{w}_\theta)}\big(s_i, \widetilde{\theta}(s_i;\mathbf{w}_\theta)\big) - \widetilde{V}(s_i;\mathbf{w}_V)\Big)^2$.

To construct particular forms of $\widehat{U}$ and $\widehat{e_i^2}$, one must take into consideration that each action a_i has been drawn from the distribution $\varphi(.;\theta_i)$, where, in general, $\theta_i \neq \widetilde{\theta}(s_i;\mathbf{w}_\theta)$ for the current value of $\mathbf{w}_\theta$. In other words, the action has been drawn from a "wrong" distribution. The construction of the appropriate estimators may be based on importance sampling and the ideas developed by Precup *et al.* in e.g. [6], [7]. We discuss the details in [13].

4 Illustration: Cart-Pole Swing-Up

The discussion above leads to two algorithms of reinforcement learning. First, the *non-intensive* algorithm can be treated as a special version of the generic algorithm discussed in [10]. We call it Randomized Policy Optimizer (RPO). The second, the *intensive* one, is based on direct estimation and we call it the Intensive Randomized Policy Optimizer (IRPO). We discuss its implementation in more details in [13].

In this section, we briefly illustrate a behavior of RPO and IRPO algorithms used to control the Cart-Pole Swing-Up [2], which is a modification of the *inverted pendulum* frequently used as a benchmark for reinforcement learning algorithms. The control objective is to, by moving cart, swing up the pendulum attached to the cart and stabilize it upwards.

The algorithms employed by Doya [2] utilize the exact model of the plants dynamics. They control the plant sufficiently well after 700 trials.

The learning curves for our algorithms are presented in Fig. 1. The RPO algorithm, a simple model-free actor-critic algorithm learns a good behavior after about 3000 trials (see [12] for details of its implementation). The IRPO achieves a satisfying behavior after about 100 trials (see [13]), which makes about 15 minutes of the real time of the plant. Our algorithms do not use the Cart-Pole Swing-Up model and yet RPO behavior is comparable to the one obtained in [2] with the model, while IRPO behaves better then model-based technique. Furthermore, IRPO performs all the computation in the real time of the plant; the entire process of learning lasts for about 15 minutes.

5 Conclusions and further work

We replaced the stochastic approximation in actor-critic algorithms with estimation based on the entire available history of actor-environment interactions. The resulting algorithm of reinforcement learning in continuous space of states and actions seems to be powerful enough to be used in adaptive control tasks in real time. The algorithm does not use nor build a model of the environment.

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

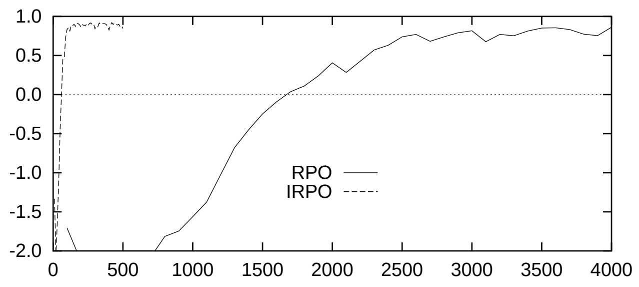

Fig. 1. RPO and IRPO applied to the Cart-Pole Swing-Up: The average reinforcement as a function of the trial number. Each point averages the reinforcements in 10 consecutive trials and each curve averages 10 runs (IRPO) or 100 runs (RPO).

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. K. Doya, "Reinforcemente learning in continuous time and space," *Neural Computation,* 12:243-269, 2000.
3. V. R. Konda and J. N. Tsitsiklis, "Actor-Critic Algorithms," *SIAM Journal on Control and Optimization,* Vol. 42, No. 4, pp. 1143-1166, 2003.
4. M. G. Lagoudakis and R. Paar, "Model-free least-squares policy iteration," *Advances in Neural Information Processing Systems,* volume 14, 2002.
5. A. W. Moore and C. G. Atkeson, "Prioritized Sweeping: Reinforcement Learning with Less Data and Less Real Time," *Machine Learning,* Vol. 13, October, 1993.
6. 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.
7. 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.
8. 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.
9. R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 1998.
10. 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.
11. C. Watkins and P. Dayan, "Q-Learning," *Machine Learning*, vol. 8, pp. 279-292, 1992.
12. 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.

13. P. Wawrzynski, A. Pacut, “Model-free off-policy reinforcement learning in continuous environment,” submitted for publication, available at http://home.elka.pw.edu.pl/∼pwawrzyn, 2004.
14. R. Williams, “Simple statistical gradient following algorithms for connectionist reinforcement learning,” *Machine Learning,* 8:299-256, 1992.