A Note on Actor-Critic Algorithm

Yi-Feng Chen
Department of Computer Science
National Chiao Tung University
gavin0430.cs04@g2.nctu.edu.tw

1 Introduction

In this paper, they propose an algorithm called actor-critic algorithms[5] for simulation-based optimization of a Markov decision process. Actor-critic algorithm combines the following two majority of reinforcement Learning:

(a) Actor-only methods typically work with a parameterized family of policies over which optimization procedures can be used directly. The gradient of the performance is directly estimated by simulation, and the parameters are updated in a direction of improvement. The disadvantage of this method is that the gradient estimators may cause a large variance.

(b) Critic-only methods use temporal difference learning and have a lower variance in the estimates of expected returns. These type of methods use value function to approximate the solution to the Bellman equation. After constructing a good enough approximation, algorithm will select a greedy action which will get the highest expected return. Since it need to go through all action to get the optimal value, it will takes lots of resources especially if the action space is large. Moreover, Critic-only method which usually discretize the continuous action space to get a discrete action space is hard to find a true optimum in continuous action space.

Actor-critic methods combine the advantage of actor-only and critic-only methods and try to eliminate the disadvantage of both of them which are large variance and the problem of dealing with the continuous action space. The actor brings the advantage of computing continuous actions without the need for optimization procedures on a value function and the critic supplies the actor with low-variance knowledge of the performance.

A good analogy of the actor-critic is a young boy with his mother. The child (actor) constantly tries new things and exploring the environment around him. He eats its own toys, he touches the hot oven, he bangs his head in the wall (I mean why not). His mother (the critic) watches him and either criticize or compliment him. The child listen to what his mother told him and adjust his behavior. As the kid grows, he learns what actions are bad or good and he essentially learns to play the game called life. That's exactly the same way actor-critic works just like the figure 1[3].

The actor can be a function approximator like a neural network and its task is to produce the best action for a given state. Of course, it can be a fully connected neural network or a convolution neural network or anything else. The critic is another function approximator, which receives as input the environment and the action by the actor, concatenates them and output the action value (Q-value) for the given pair.

A distinction is made between algorithms that use a standard (sometimes also called vanilla) gradient and the natural gradient that became more popular in the course of the last decade. The remaining part of actor-critic algorithms consists mainly of algorithms that choose to update the actor by moving it towards the greedy policy underlying an approximate state-action value function.

Preprint. Under review.

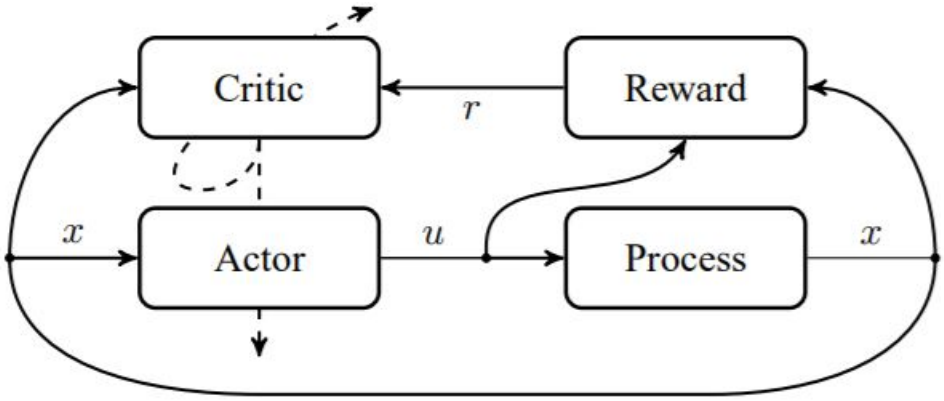

Figure 1: Schematic overview of an actor-critic algorithm. The dashed line indicates that the critic is responsible for updating the actor and itself.

Therefore, this study makes a major contribution to research on reinforcement learning by combining the most two popular methods to a new method, and simultaneously overcomes the two obstacles of them which are the high-variance and large time-consuming.

2 Problem Formulation

Consider a Markov decision process (MDP) with finite state space S, and finite action space A.

Let $g\colon S \times A \to \mathbb{R}$ be a given cost function. A randomized stationary policy (RSP) is a mapping μ that assigns to each state x a probability distribution over the action space A. We consider a set of randomized stationary policies $\mathbb{P} = \{\mu_\theta; \theta \in \mathbb{R}^n\}$, parameterized in terms of a vector θ. For each pair $(x, u) \in S \times A$, $\mu_\theta(x, u)$ denotes the probability of taking action u when the state x is encountered, under the policy corresponding to θ. Let $p_{xy}(u)$ denote the probability that the next state is y, given that the current state is x and the current action is u. Note that under any RSP, the sequence of states $\{X_n\}$ and of state-action pairs $\{X_n, U_n\}$ of the Markov decision process form Markov chains with state spaces S and $S \times A$, respectively. They make the following assumptions.

(a) They assume that the map $\theta \mapsto \mu_\theta(x, u)$ is twice differentiable for all $x \in S$ and $u \in A$. Furthermore, there exists a $\mathbb{R}^n$ valued function $\psi_\theta(x, u)$ such that $\bigtriangledown\mu_\theta(x, u) = \mu_\theta(x, u)\psi_\theta(x, u)$ where the mapping $\theta \mapsto \mu_\theta(x, u)$ is bounded and has first bounded derivatives for any fixed x and u.

(b) They assume that the Markov chains $\{X_n\}$ and $\{X_n, U_n\}$ are irreducible and aperiodic, with stationary probabilities $\pi_\theta(x)$ and $\eta_\theta(x, u) = \pi_\theta(x)\mu_\theta(x, u)$,respectively, for all $\theta \in \mathbb{R}^n$.

According to assumption (a), whenever $\mu_\theta(x, u)$ is nonzero we have

$$\psi_\theta(x, u) = \frac{\bigtriangledown\mu_\theta(x,u)}{\mu_\theta(x,u)} = \bigtriangledown \ln \mu_\theta(x, u)$$

Also, they consider the average cost function $\lambda : \mathbb{R}^n \mapsto \mathbb{R}$,given by

$$\lambda(\theta) = \sum_{x\in S, u\in A} g(x, u)\eta_\theta(x, u). \tag{1}$$

,which means the cost times the probability of each action and state. We are interested in finding a θ which get the minimal $\lambda(\theta)$. For each $\theta \in \mathbb{R}^n$, let $V_\theta : S \mapsto \mathbb{R}$ be the "differential" cost function, defined as solution of Poisson equation:

$$\lambda(\theta) + V_\theta(x) = \sum_{u\in A} \mu_\theta(x, u)[g(x, u) + \sum_y p_{xy}(u)V_\theta(y)] \tag{2}$$

Intuitively, $V_\theta(x)$ can be viewed as the "disadvantage" of state x: it is the expected excess cost - besides the average cost - incurred if we start at state x. It plays a role similar to that played by the

more familiar value function that arises in total or discounted cost Markov decision problems. Finally, for every $\theta \in \mathbb{R}^n$, we define the q-function $q_\theta : S \times A \mapsto \mathbb{R}$, by

$$q_\theta(x,u) = g(x,u) - \lambda(\theta) + \sum_y p_{xy}(u) V_\theta(y) \psi_\theta(x,u) \tag{3}$$

Theorem 1.

$$\frac{\partial}{\partial \theta_i}\lambda(\theta) = \sum_{x,u} \eta_\theta(x,u) q_\theta(x,u) \psi_\theta^i(x,u) \tag{4}$$

$$where\ \ \psi_\theta^i(x,u)\ \ stands\ for\ the\ i-th\ component\ of\ \psi_\theta.$$

$q_\theta(x,u)$ in the above formula is considered to be expected excess cost in the Markov chain $\{X_n, U_n\}$, and is then estimated by means of simulation, leading to actor-only algorithms. Here, they provide an alter the formula in Theorem 1 to an inner product version, and thus derive a different set of algorithms, which is the general case of an infinite space as well.

For any $\theta \in \mathbb{R}^n$, we define the inner product $\langle\ ,\rangle_\theta$ of two real valued functions $q1$,$q2$ on $S \times A$, viewed as vectors in $\mathbb{R}^{|S||A|}$, by

$$\langle q_1\ , q_2 \rangle_\theta = \sum_{x,u} \eta_\theta(x,u) q_1(x,u) q_2(x,u) \tag{5}$$

With this notation, we can rewrite the theorem 1 as

$$\tfrac{\partial}{\partial \theta_i}\lambda(\theta) = \langle q_\theta\ , \psi_\theta^i \rangle_\theta \quad , i = 1,\ \ldots\ , n.$$

Let $\|\cdot\|_\theta$ denote the norm induced by this inner product on $\mathbb{R}^{|S||A|}$. For each $\theta \in \mathbb{R}^n$ let Ψ denote the span of the vectors $\{\psi_\theta^i; 1 \le i \le n\}$ in $\mathbb{R}^{|S||A|}$. Note that although the gradient of λ depends on the q-function, which is a vector in a possibly very high dimensional space $\mathbb{R}^{|S||A|}$, the dependence is only through its inner products with vectors in Ψ_θ . Thus, instead of "learning" the function q_θ, it would suffice to learn the projection of q_θ on the subspace Ψ_θ.

Indeed, let $\Pi_\theta\ :\ \mathbb{R}^{|S||A|} \mapsto \Psi_\theta$ be the projection operator defined by

$$\Pi_\theta q\ =\ \arg\min_{\hat{q} \in \Psi_\theta}\ \|q - \hat{q}\|_\theta$$

Since

$$\langle q_\theta\ , \psi_\theta \rangle_\theta = \langle \Pi_\theta q_\theta\ , \psi_\theta \rangle_\theta, \tag{6}$$

it is enough to compute the projection of q_θ onto ψ_θ in the next section.

3 Theoretical Analysis

Please present the theoretical analysis in this section. Moreover, please formally state the major theoretical results using theorem/proposition/corollary/lemma environments. Also, please clearly highlight your new proofs or extensions (if any).

In actor-critic algorithms, we apply stochastic gradient algorithms on the actor and the job of the critic is to compute an approximation of the projection $\Pi_\theta q_\theta$ of q_θ onto ψ_θ. The actor uses the approximate gradient from critic to update its policy.The analysis[8][9] shows that this is precisely what TD algorithms try to do, i.e., to compute the projection($\Pi_\theta q_\theta$) of the value function(q_θ) onto a subspace spanned(Ψ_θ) by feature vectors. This allows us to implement the critic by using a TD algorithm. In their algorithms, the control policy as well as the features need to change as the actor updates its parameters. Also some research[2][6] has shown that this need not pose any problems, as long as the actor parameters are updated on a slower time scale.

They describe two actor-critic algorithms, which are only different in one factor which is that the critic updates or not. In both variants, the critic is a TD algorithm with a linearly parameterized approximation architecture for the q-function, of the form

$$Q_r^\theta(x,u) = \sum_{j=1}^m r^j \phi_\theta^j(x,u), \tag{7}$$

where $r = (r^1, \ldots, r^m) \in \mathbb{R}^n$ denotes the parameter vector of the critic. The features $\phi_\theta^j, j = 1, \ldots, m$, used by the critic are dependent on the actor parameter vector θ and are chosen such that their span in $\mathbb{R}^{|S||A|}$, denoted by Φ_θ, contains Ψ_θ.Note that the formula (6) still holds if Π_θ is redefined as projection onto Φ_θ as long as Φ_θ contains Ψ_θ. The most straightforward choice would be to let $m = n$ and $\phi_\theta^i = \psi_\theta^i$ for each i. Nevertheless, we allow the possibility that $m > n$ and ϕ_θ properly contains ψ_θ, so that the critic uses more features than that are actually necessary. This added flexibility may turn out to be useful in the following ways:

- Using richer set of features ψ_θ^i may avoid that the operator Π_θ becomes ill-conditioned and the algorithms can become unstable which are caused by the features ψ_θ are either close to zero or are almost linearly dependent for some certain θ.
- For the second algorithm that (TD(α) $\alpha < 1$) critic can only compute approximate - rather than exact - projection. The use of additional features can result in deceasing the approximation error.

Along with the parameter vector r, the critic stores some auxiliary parameters: these are a scalar to estimate λ. The actor and critic updates during the simulation of a single sample path of the controlled Markov chain. Let r_t, Z_t, λ_t be the parameters of the critic, and let θ_t be the parameter vector of the actor, at time t. Let (X_t, U_t) be the state-action pair at that time. Let X_{t+l} be the new state, obtained after action U_t is applied. A new action U_{t+1} is generated according to the RSP corresponding to the actor parameter vector θ_k.The critic carries out an update similar to the average cost temporal-difference method of [9]:

$$\lambda_{t+1} = \lambda_t + \gamma_t(g(X_t, U_t) - \lambda_t),$$
$$r_{t+1} = r_t + \gamma_t(g(X_t, U_t) - \lambda_t + Q_{r_t}^{\theta_t}(X_{t+1}, U_{t+1}) - Q_{r_t}^{\theta_t}(X_t, U_t))z_t.$$

(Here, γ_t is a positive stepsize parameter.) The two variants of the critic use different ways of updating z_t:

$TD(1)\ Critic$: Let x^* be a state in S.

$$z_{t+1} = z_t + \phi_{\theta_t}(X_{t+1}, U_{t+1}), \quad if X_{t+1} \neq x^*$$
$$= \phi_{\theta_t}(X_{t+1}, U_{t+1}), \quad otherwise.$$

$TD(\alpha)\ Critic, 0 \leq \alpha < 1$:

$$z_{t+1} = \alpha z_t + \phi_{\theta_t}(X_{t+1}, U_{t+1})$$

$Actor$: Actor updates its parameter vector by

$$\theta_{t+1} = \theta_t - \beta_t \Gamma(r_t) Q_{r_t}^{\theta_t}(X_{t+1}, U_{t+1}) \psi_{\theta_t}(X_{t+1}, U_{t+1}).$$

Here, β_t is a positive stepsize and $\Gamma(r_t) > 0$ is a normalization factor which is Lipschitz continuous. Also, there exists $C > 0 s.t.$

$$\Gamma(r) \leq \frac{C}{1+\|r\|}$$

The above presented algorithms are only two out of many variations.

Since the actor-critic algorithms are gradient-based, we can't expect to get the globally optimal policy. The best case is the convergence of $\bigtriangledown\lambda(\theta)$ to zero; in practical terms,this will usually be considered to converge to a local minimum of $\lambda(\theta)$.

4 Conclusion

The key observation in this paper is that the actor parameterization and the critic parameterization should not be chosen independently, which means their can share some parameter in the first few layers.Rather, an appropriate approximation architecture for the critic is directly prescribed by the parameterization used in actor.

Capitalizing on the above observation, they have presented a class of actor-critic algorithms, combining the advantages of actor-only and critic-only methods. In contrast to existing actor-critic methods,

their methods apply to high-dimensional action space problems, and are mathematically sound in the sense that they possess certain convergence properties.

Also there are researches that have been focused on natural Actor-Critic Algorithms which use natural policy gradient to replace vanilla policy gradient, which have been shown some advantages[1]. Besides, there are a better version of actor-critic algorithm called Asynchronous Advantage Actor Critic (A3C)[7] which implements parallel training where multiple workers in parallel environments independently update a global value function.

But, actor-critic algorithm also suffer form some challenges which are high sample complexity and brittleness to hyperparameters.Both of these challenges severely limit the applicability of such methods to complex, real-world domains. There are later actor-critic algorithm called soft actor-critic algorithm[4] where the actor aims to simultaneously maximize expected return and entropy for reinforcement learning framework.

References

[1] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

[2] Vivek S. Borkar. Stochastic approximation with two time scales. *Syst. Control Lett.*, 29(5):291–294, February 1997.

[3] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, 2012.

[4] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018.

[5] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

[6] Vijaymohan R. Konda and Vivek S. Borkar. Actor-critic like learning algorithms for markov decision processes. *SIAM J. Control Optim.*, 38(1):94–123, November 1999.

[7] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016.

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

[9] J. N. Tsitsiklis and B. Van Roy. Average cost temporal-difference learning. In *Proceedings of the 36th IEEE Conference on Decision and Control*, volume 1, pages 498–502 vol.1, 1997.