Original Article
International Journal of Fuzzy Logic and Intelligent Systems
Vol. 16, No. 4, December 2016, pp. 270-275
http://dx.doi.org/10.5391/IJFIS.2016.16.4.270

ISSN(Print) 1598-2645
ISSN(Online) 2093-744X

Actor-Critic Algorithm with Transition Cost Estimation

Denisov Sergey and Jee-Hyong Lee
Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon, Korea

Abstract

We present an approach for acceleration actor-critic algorithm for reinforcement learning with continuous action space. Actor-critic algorithm has already proved its robustness to the infinitely large action spaces in various high dimensional environments. Despite that success, the main problem of the actor-critic algorithm remains the same-speed of convergence to the optimal policy. In high dimensional state and action space, a searching for the correct action in each state takes enormously long time. Therefore, in this paper we suggest a search accelerating function that allows to leverage speed of algorithm convergence and reach optimal policy faster. In our method, we assume that actions may have their own distribution of preference, that independent on the state. Since in the beginning of learning agent act randomly in the environment, it would be more efficient if actions were taken according to the some heuristic function. We demonstrate that heuristically-accelerated actor-critic algorithm learns optimal policy faster, using Educational Process Mining dataset with records of students' course learning process and their grades.

Keywords: Actor-critic algorithm, Reinforcement learning, Continuous action space, Heuristic function

1. Introduction

Q-Learning is the task-solving method that explores task environment and receives the feedback in form of rewards. It is simple and in the same time powerful approach for problems where we need to establish sequence of actions that leads to the optimal goal. Structure of the basic reinforcement learning process you can see on Figure 1. After initial version of reinforcement learning equation appeared [1], there were proposed a lot of extensions that increase its performance. One of the latest presented methods was Deep Q Network (DQN) [2] (or Neural Q-Fitting algorithm [NFQ]) method that outperformed the human in arcade game environments [3]. The great success of this approach is driven by robustness of the algorithm to the high dimensional spaces. Instead of estimation q-value for each state as it was done before, it approximates q-values through neural network. However, it still has some limitations like a lack of stability during convergence of network or inability to be used in environments with high or continuous action space [4, 5]. Consequently, there was an improvement of the DQN that allows the methods to be efficient in high dimensional state and action spaces as well as in low dimensional spaces. The main idea is the dividing DQN into two parts; actor part, that chooses actions in environment, and critic part, that evaluates how appropriate was that action at the current state [6]. This significant progress helps to solve complex problems

Received: Nov. 28, 2016
Revised : Dec. 12, 2016
Accepted: Dec. 13, 2016

Correspondence to: Jee-Hyong Lee
(John@skku.edu)

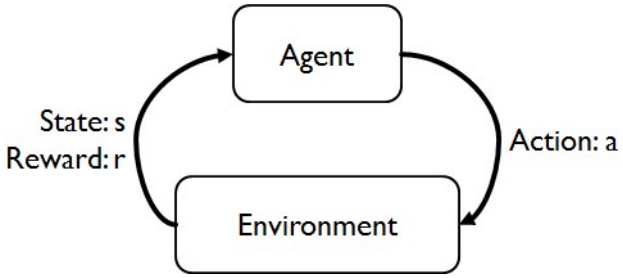


Figure 1. Reinforcement learning structure.

that require obtaining policy in continuous action space. However, to search the optimal policy it has to be trained over long time, since it maximizes action-value function iteratively at every step in infinite space of possible actions.

In this work we consider the heuristic function, which allows to speed up the training of actor-critic algorithm [6]. This function represents the initial action preference distribution, which is independent on state space. Using transition history, we are able to approximate the rewards that were obtained before as a result of some actions set. Knowing this approximation, we can state that some actions have higher probability to lead us to the optimal goal, so it's better to observe the environment in that direction. In case, when q-values just initialized, we can explore environment using action transition costs that give us hints which state is worth to observe during next step.

In order to evaluate our method we applied modified actor-critic algorithm to the educational process mining dataset. This dataset consist of 115 students' records of course taken during one semester. It contains the set of sequences, where the name of course activity done is stated and time required for its completeness is recorded. We use activities that took more than 5 minutes only, to get rid of unnecessary noise. Result of experiments shows that modified actor-critic algorithm converged faster compare to the initial version.

In the next section we will discuss related work and earlier researches. After that, we will describe our heuristic acceleration to the existing methods and in Section 4, we will observe the method evaluation and results of experiment. In the last section, we will discuss results and make conclusion over all work that was done.

2. Related Work

2.1 Original Q-Learning Algorithm

The original q-learning paper described the algorithm using the q-learning function (1).

$$Q_n(s,\ a) = (1-\alpha)\, Q_{n-1}(s,\ a) + \alpha\left[r_n + \gamma \max Q_{n-1}(s,a)\right]. \quad (1)$$

Here we have $Q_n(s,\ a)$ as a q-value of n-th state s, after action a. α is the learning rate and γ is the discount factor for the previous state q-value. r_n is the reward we can obtain at current that state. This equation describes an agent movement in some environment with evaluation of every step. First, agent observes its current state, after that it selects and performs an action and then observes the subsequent state. After doing an action, agent received immediate feedback in terms of reward and adjusts its q-value using a learning factor [1]. This equation has a lot of extensions like a SARSA algorithm [7] that is being used for DQN often. However, it is not possible to observe high dimension environment with those algorithms, because computation of q-value for each state would take huge amount of time.

2.2 DQN Algorithm

Therefore, there were researches how to apply neural networks to approximate the q-value function to be able to learn policies from high dimensional state spaces. The DQN algorithm has a structure of neural network that takes a state and action pair as an input and shows the approximated q-value as follows:

$$\begin{cases} r + \gamma \max Q(s', a'), & s' \neq \text{final}, \\ r, & s' = \text{final}. \end{cases}$$

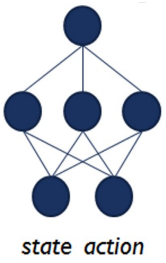


Figure 2. Structure of the DQN.

On Figure 2 we can see the neural network structure that approximates state, action pair to the reward r added to the discounted maximum next q-value in case if state is not final [8].

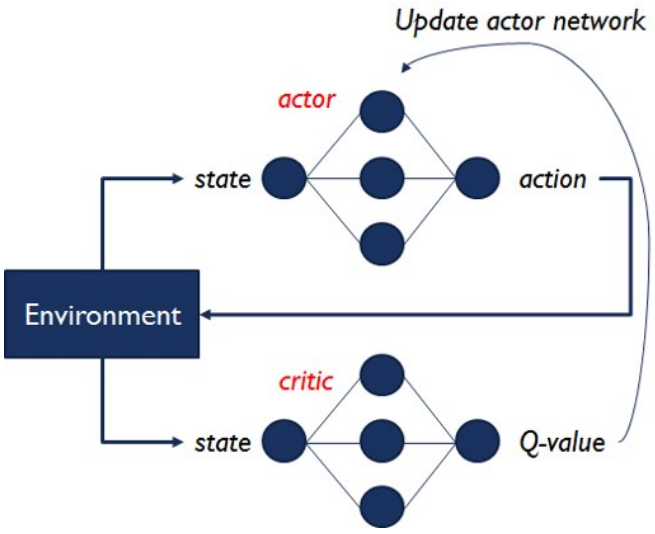


Figure 3. Structure of actor-critic algorithm.

And if state is final, we just approximate it to the reward we can get on that state. DQN algorithms moved reinforcement learning research toward to the new level, where computers started to be competitive to the human in the fields, nobody could ever think computer would success [9].

2.3 Actor-Critic Algorithm

Recently, during this year, there were a lot of researches related to q-learning in continuous action space that suggested a solution to the problem q-learning had before [10]. Actor-critic algorithm, that uses policy gradient for updating actor parameters, become well-known during short time, due to its robustness to the variated sort of tasks. The structure of the actor-critic algorithm is showed on Figure 3.

The idea of actor-critic algorithm is simple. We have an agent network that acts in environment and we have critic network that gives a grade for every action [11]. We update critic network like every other network, using mean square error as a loss function. However, in case of actor network, we use the policy gradient, that leads us to the higher q-value with each update. Actor-critic algorithm deals well with the problems that DQN could not deal with. However the main problem of actor-critic algorithm is its speed of convergence.

In the next section, we will describe how we modified that algorithm to accelerate the searching speed.

3. Heuristic Acceleration Function

3.1 Motivation

For increasing the speed of algorithm convergence, we implemented heuristic acceleration function. Using neural network, it approximates the award of every action independently on states. New exploration strategy helps to find out the optimal policy from transition history faster than if we would do it randomly. There were studies [12] related to the introducing heuristic function for multiagent reinforcement learning [13], however, it was able to perform only in deterministic action space. Combining heuristic function and actor-critic algorithm should lead to increasing the speed of algorithm convergence, in case when the optimal policy should be established from the set of previous interactions.

3.2 Implementation

The best solution for function approximation is the using the neural network. For our purposes, we defined neural network with action for input and average reward for this action over one episode as an output. The mean squared error function or just loss function for our method looks like as it is shown on Eq. (2).

$$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - avg(r(a \mid \theta)))^2, \tag{2}$$

where N is the length of the dataset, y_i is the target value and $r(a \mid \theta)$ is the average reward for action a of some student approximated by network parameters θ.

However, increasing q-value with every action may lead to losing the update stability of the critic network, because target value would grow with every iteration and network won't be able to converge. Therefore, we implemented the transition cost as a heuristic acceleration, to make the target of q-value decrease with each step.

Moreover, the last thing we want to do is to make algorithm to do not converge at all. Following this, transition costs should be small to do not impact on the training later, when the optimal policy will be already near to be obtained. To deal with that problem, we normalized our reward, and final version looks like it stated on Eq. (3).

$$r = \frac{R(a \mid \theta)}{\max_r(a_1 \cdots a_N)} - 1. \tag{3}$$

Here we normalize our received reward $R(a \mid \theta)$ using the maximum reward of the whole set of actions. Then we subtracting

one, to receive a transition cost. In case if our network will predict the reward for some action bigger than the maximum reward from set of experienced actions, that we have in our transaction history, it will mean that it found better direction. Then we will have a positive feedback for that action. Oppositely, if we will have the smallest predicted value, our cost for this action will be the largest one.

We used the version of algorithm, stated in the work of Lillicrap et al. [6]. He proposed to use target networks to actor and critic networks to increase the stability of updating. He also used experience replay technique (replay buffer), random process (Ornstein-Uhlenbeck process, Eq. (4)) for action space exploration and minibatch training mode.

$$dx_i = \theta(\mu - x_i)dt + \sigma dW_i. \tag{4}$$

3.3 Complete Algorithm

With our heuristic accelerated function the complete actor-critic algorithm will look like on Figure 4.

This algorithm was adapted from 'Continuous Control Deep Reinforcement Learning' article written by Lillicrap et al. [6]. We calculate the transition cost in the moment before updating critic network to receive modified q-value.

In the next section, we will discuss details of the experiment and observed obtained results.

4. Experiment

4.1 Dataset

To test our approach, we decided to extract optimal policy from Educational Process Mining dataset. This dataset contains a one semester information about students. In records you can find the activity and its duration that student was doing during the course. We did not include activities that were shorter than 5 minutes, because we did not want noisy data to imply our experiment. As a reward we used the grades of the students in the end of semester. State was represented by the overall time of each activity student has done until current moment. Action was defined by the option how long and which activity to do the next.

4.2 Settings

For noise function (Ornstein-Uhlenbeck process) we chose $\mu = 0$, $\theta = 0.15$, and $\sigma = 0.2$ [6]. Our proposed neural network was trained on 1,645 examples of actions with 15 dimensions having continuous values from 0 to 1. We also created simulation network that was imitating the environment trained on the original history transitions set. Actor-Critic network was trained over 130,000 iterations and it predicted 10,000 sequences in total.

For evaluation of both algorithms we trained simulation neural network that approximates the student states and his grade. Therefore, when we will receive some new state we can predict its grade or, in another words, evaluate it using our reward simulation neural network.

Randomly initialize critic network $Q(s, a \mid \theta^Q)$ and actor $\mu(s \mid \theta^\mu)$ with weights θ^Q and Q^μ
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode= 1, M do
  Initialize a random process N for action exploration
  Receive initial observation state s_1
  for $t = 1, T$ do
    Select action $a_t = \mu(s_t \mid \theta^\mu) + N_t$ according to the current policy and exploration noise
    Execute transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Calculate $r_i = \dfrac{R(a \mid \theta)}{\max_r(a_1 \cdots a_N)} - 1$
    Set target of critic network:

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1} \mid \theta^\mu) \mid \theta^{Q'})$$

    Update critic by minimizing the loss:

$$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - Q(s_i, a_i \mid \theta^Q))^2$$

    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J = \frac{1}{N}\sum_{i=1}^{N}\left(\nabla_a Q(s, a \mid \theta^Q) \mid_{s=s_i, a=\mu(s_i)} \times \nabla_{\theta^\mu}\mu(s \mid \theta^\mu) \mid_{s_i}\right)$$

    Update the target networks:
      $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
      $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
  end for
end for

Figure 4. Complete actor-critic algorithm with heuristic acceleration.

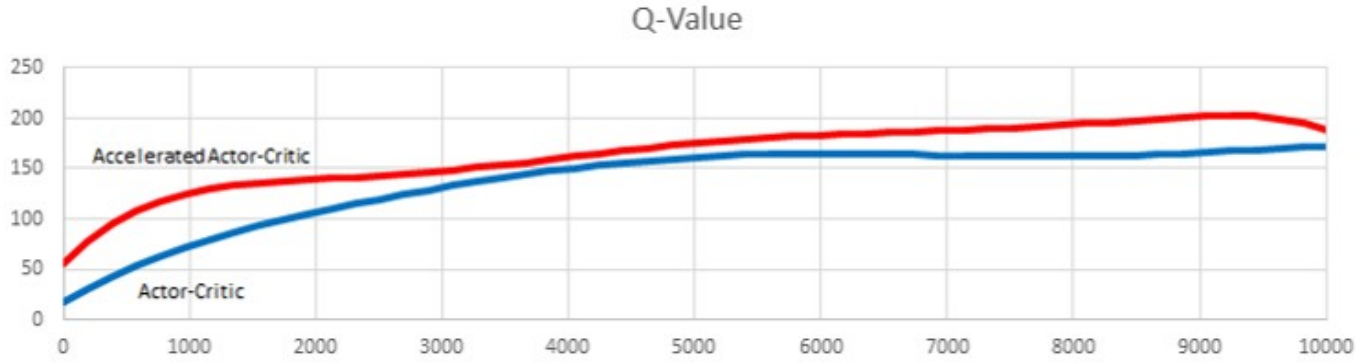


Figure 5. Results of experiment.

4.3 Results

On Figure 5 we can see the graph of comparison q-values between two simulated policies extracted from dataset. Red line that characterize our modified algorithm shows the increase of policy right after start of training. However, blue line is not following red line, what means we achieved our purpose of accelerating the basic algorithm. After 3,000th episode curves became closer to each other, however after 2,000 iterations difference is increasing again. After 7,000th episode both lines started to slow down increasing and be more stable. It means that algorithm has obtained the optimal policy.

Q-value itself is also higher from the policy of modified actor-critic algorithm. It is starting to be higher from the beginning, because it is already initially accelerated by heuristic function, when for original algorithm transition reward was all the time equal to zero. If we will compare the points on the given graph with the same q-values for both networks, we will see that under q-value 150, proposed method converged to the same q-value almost twice faster than original algorithm. It is proves the advantage of using heuristic algorithm.

Overall results shows, that performance of modified actor-critic algorithm increased and it obtained the optimal policy faster than its previous version, what means we achieved our purpose.

5. Conclusion

In this paper we proposed the heuristic acceleration function for actor-critic algorithm. This function is approximated thorough neural network and used during the calculation of the target value of the critic network. Introduced method has an assumption that every action may have its own distribution of preference that does not depend on state and can be found be looking through the transition history rewards. After obtaining reward for each action, we train neural network to be able to establish rewards for new actions suggested by actor. Because action rewards will give the initial direction to the optimal goal, actor-critic is able to converge faster than before implementation of our method.

In further research, we are planning to test our proposed method in the more environments. We will test it in fields with higher action and state spaces and observe the behavior of the policy during training.

Conflict of Interest

No potential conflict of interest relevant to this article was reported.

Acknowledgements

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2014M3C4A7030503). Also, this research is supported by Ministry of Culture, Sports and Tourism (MCST) and Korea Creative Content Agency (KOCCA) in the Culture Technology (CT) Research & Development Program 2016.

References

[1] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279-292, 1992. http://dx.doi.org/10.1023/A:1022676722315

[2] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *Journal of Machine Learning Research*, vol. 6, pp. 503-556, 2005.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," Available https://arxiv.org/abs/1312.5602

[4] Y. Tkachenko, "Autonomous CRM control via CLV approximation with deep reinforcement learning in discrete and continuous action space," Available https://arxiv.org/abs/1504.01840

[5] M. Riedmiller, "Neural fitted Q iteration: first experiences with a data efficient neural reinforcement learning method," in *Machine learning: ECML 2005*, J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, and L. Torgo, Eds. Berlin: Springer Berlin Heidelberg, 2005, pp. 317-328. http://dx.doi.org/10.1007/11564096_32

[6] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," Available https://arxiv.org/abs/1509.02971

[7] R. Sutton, "Generalization in reinforcement learning: successful examples using sparse coarse coding," *Advances in Neural Information Processing Systems*, vol. 8, pp. 1038-1044, 1996.

[8] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," Available https://arxiv.org/abs/1509.06461

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, et al, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529-533, 2015. http://dx.doi.org/10.1038/nature14236

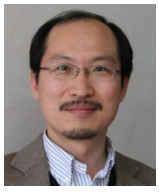
[10] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, Beijing, China, 2014, pp. 387-395.

[11] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems 12*, vol. 99, pp. 1057-1063, 2000.

[12] L. A. Celiberto, C. H. C. Ribeiro, A. H. R. Costa, and R. A. C. Bianchi, "Heuristic reinforcement learning applied to robocup simulation agents," in *RoboCup 2007: Robot Soccer World Cup XI*, U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, Eds. Berlin: Springer Berlin Heidelberg, 2008, pp 220-227. http://dx.doi.org/10.1007/978-3-540-68847-1_19

[13] R. A. C. Bianchi, M. F. Martins, C. H. C. Ribeiro, and A. H. R. Costa, "Heuristically-accelerated multiagent reinforcement learning," *IEEE Transactions on Cybernetics*, vol. 44, no. 2, pp. 252-265, 2014. http://dx.doi.org/10.1109/TCYB.2013.2253094

Denisov Sergey s an M.S. candidate at Department of Computer Science, Sungkyunkwan University, Korea. He majored in Computer Engineering in his bachelor study. He has been working on research related to machine learning and reinforcement learning.
E-mail: denisovser@naver.com



Jee-Hyong Lee is a professor at Sungkyunkwan University. He has been working on research related to fuzzy theory and application intelligent system. He published more than 100 publications in peer-reviewed journals or conferences, and books on intelligent systems, data mining, and machine learning.
E-mail: john@skku.edu