Journal of KIIT. Vol. 18, No. 7, pp. 15-20, Jul. 31, 2020. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2020.18.7.15

Scheme of a Classic Control-Based Program Model with Non-Symmetric Deep Auto-Encoder of Actor-Critic

Chayoung Kim*

Abstract

Reinforcement learning (RL), particularly Actor-Critic (A2C), one of policy gradient (PG) algorithms becomes a mainstream. A design of a classic control-based program model requires domain-indicated experience knowledge based on a neural network for incorporating into the model strategy, which becomes A2C-powered. There are some research studies on the program model using general Artificial Intelligence (AI) techniques, in other words, A2C and neural networks. The previous well-known algorithms can proceed from a simple Convolutional Neural Network (CNN), to attempting an experiment with more complicated additions, such as LSTM (Long Short-Term Memory models). However, there are concerns about the requirements of the current experimental environment when faced with the demands of significant computational powers. Therefore, we employ a definition of Actor-Critic with non-symmetric deep auto-encoder to find an optimal behavior strategy for the agent to obtain optimal rewards with the bottommost environmental resources. Algorithm comparisons show that our proposed model demonstrates improvements of optimal rewards with the minimum limit over the developed deep neural network combined Actor-Critic.

요 약

강화학습의 액터-크리틱은 정책 그라디언트 알고리즘으로써, 매우 활용도가 높다. 전형적인 제어-기반 프로그램 모델의 설계에 있어서 뉴럴 네트웍에 기반한 도메인 지식의 접목은 해당 액터-크리틱의 장점을 이용할 수 있다. 그리하여, 액터-크리틱과 뉴럴 네트웍의 AI(인공지능) 기술을 접목한 프로그램 모델 기술에 대한 연구가 활발하다. 기존 연구들에서는 CNN을 주로 사용하고, 더 복잡하고 지능적인 것을 원하는 경우, LSTM을 사용하기도 한다. 하지만, 컴퓨팅 파워를 많이 요구하는 실세계와 연관된 분야에서는 조금 더 작은 자원을 사용하는 것을 선호한다. 그리하여, 본 연구는 비대칭 딥 오토앤코더를 가진 액터-크리틱을 에이전트에 적용하여, 환경에 대한 리소스를 가장 기본적으로 활용하면서 최적의 보상을 얻을 수 있도록 한다. 단순한 딥 뉴럴 네트웍을 가진 기존의 알고리즘과 본 연구가 제안하는 알고리즘의 비교에서 적은 자원으로 본 모델이 조금 더 나은 리워드(Reward)를 받는 것을 보여주고 있다.

Keywords
actor-critic, program model, OpenAI gym, non-symmetric deep auto-encoder, reinforcement learning

Ⅰ. Introduction

Scheme of a classic control-based program model has becoming apparent as a favorable and special playground for encouraging the application of AI, specifically reinforcement learning (RL) [1][2]. One of the core challenging issues with these applications acquires enough data-sets that a neural network learns

* Assistant Professor, Div. of General Studies, Kyonggi University
- ORCID: https://orcid.org/0000-0002-4186-5882

· Received: May 27, 2020, Revised: Jul. 16, 2020, Accepted: Jul. 19, 2020
· Corresponding Author: Chayoung Kim
Div. of General Studies, 154-42, Gwanggyosan-ro, Yeongtong-gu, Suwon-si, Gyeonggi-do, Korea, 16227.
Tel.: +82-31-249-9509, Email: kimcha0@kgu.ac.kr

appropriately, and OpenAI Gym [3] for RL allows utilizing a powerful interface with a variety of different environments. Similar to [4], our proposed model explores and exploits RL on a variety of OpenAI Gym. With respect to these applications, the input to the well-known convolutional neural networks (CNN) [5][6] will be a sequence of arrays established by a specific OpenAI Gym. In [4], they utilize deep Q-Network (DQN) [5][6] with a couple of CNN to select an action, a, from environments for maximizing future reward from a given state, s. However, in terms of CNN, the research [4] can exploit the huge amount of computational resources. Especially, in a well-known control-based environment of OpenAI Gym, such as CartPole-V1 [7], an image from a hugh frame can adopt a number of moves for formatting the image to utilize an input to the developed neural network model.

Therefore, the research model in [4] should de-noise the image by using an adaptive threshold function and apply the adaptive Gaussian threshold for identifying that its value is noise or not. The research model [4] should run on Google Cloud Compute Engine with 8 cores and an Nvidia Tesla K80 GPU [4]. However, in some cases, only a laptop equipped with a budget GPU for a certain program model is exploited. With respect to the scheme of the program model on the budget, we attempt to resolve the modeling issues with a specified strategy combining small human demonstrations and preferences similar to research [8].

Therefore, we suggest to make an RL scheme with the bottommost computational resources. For complex real-world problems, some program model requires a comparatively high level of human expert interactions and the expert knowledge for identifying useful data and patterns. However, a learning from human feedback such as expert demonstrations and trajectory preferences can occur an expensive process of computer [8]. When a program model provides some human expert information, we can exploit manually-specified reward functions or expert demonstrations [5]. Therefore we take advantages of a basic RL [1][2] in the framework of Markov decision process (MDP), which is a Markov reward process with decisions. MDP model is a decision-making process using probability [1][2].

As reinforcement learning (RL), particularly policy gradient (PG) strategy [1][2] goals optimizing policy and modeling directly. PG is modeled with parameterized-function respect to θ, $\pi_\theta(a|s)$ [1][2] where θ is parameters for the best optimized policy, π. In PG strategies, the model can learn directly the policy function mapping state, s to action, a not based on a value function. One of big issues with value-based strategies known as Q-learning [5][6] is they can be oscillated during training, given to a big variance. Because of the selection of action, strategies can change extremely for an arbitrarily small change in the estimated action values. However, also PG strategies have a big disadvantage, such as following on the gradient for the best parameters.

Therefore, PG strategies can experience a smoother update at each step and because of that, they can guarantee to converge on a local maximum or global maximum. It cannot guarantee all-global maximum. It can be certainly to converge on a local maximum rather than on the global optimum. So, the deep learning approaches can help to reach the global maximum, which can converge slower and take longer to train. Similar to deep Q-learning research [9], we can utilize non-symmetric deep auto-encoder. The deep learning is the advanced sub-sets of machine learning, that can overcome some of the limitations of shallow machine learning [9][10]. It has potential for facilitating a deeper analysis of data-sets and faster identification of any feature patterns. Our proposed model, similar to [9], has capable of facilitating a deeper learning model of environment, particularly, from OpenAI Gym interfaces.

In this paper, we employ a definition of one of PG strategies, Actor-Critic [1][2] with non-symmetric deep auto-encoder to find an optimal behavior strategy for the agent to obtain optimal rewards with bottommost environmental resources. As similar in [11], the auto-encoder can work as a generator of model.

Therefore, the framework of our proposed model attempts to achieve the powerful learning capabilities with the minimum requirement in challenging environments because input data-sets from OpenAI Gym will have a wider diversity [12]. The proposed model emulates the environment with OpenAI Gym [3][7]. Similar to recent research in [8], we evaluate practically our model using TensorFlow [13] and Keras [14]. Algorithm comparisons show that the proposed model demonstrates improvements with the minimum limit over the developed deep neural network.

The rest of the paper is organized as follows. The background methods are in Section 2. Section 3 describes the proposed algorithm and experimental results and comparison. Finally, we conclude our work in Section 4.

II. Background

In RL, agent acts stochastically by choosing actions in an environment to make a maximum cumulative reward. The problem of RL follows MDP comprising a state s, an action a, an initial state distribution $p(s_0)$, a state transition distribution $p(s_{t+1}|s_t,a_t)$ over the trajectory $s_1,a_1,s_2,a_2,...,s_T,a_T$ and reward function $r:S\times A\rightarrow R$ [15]. A policy is stochastic for selecting actions in the MDP and denoted by $\pi_\theta : S\rightarrow p(A)$, where $P(A)$ is the set of probability on A and $\theta\in R^n$ is a vector of n parameters, and $\pi_\theta(a_t,s_t)$ is the conditional probability at a_t. The agent uses its policy for interacting the MDP over the trajectory [15]. The return r_t^γ is the total discounted reward from time-step t, $r_t^\gamma=\sum_{k=t}^{\infty}\gamma^{k-t}r(s_k,a_k)$, where $0<\gamma<1$ [15]. The expected total reward is $V^\pi(s)=E[r_1^\gamma|S_1=s;\pi]$ and $Q_\pi(s,a)=E[r_1^\gamma|S_1=s,$ $A_1=a;\pi]$ [15]. The goal of the agent is to obtain a policy maximizing the reward by $J_{(\pi)}=E[r_1^\gamma|\pi]$. With the density distribution$p(s\rightarrow s',t,\pi)$ and the state distribution $p^\pi(s')=\int\sum_{St=1}^{\infty}r^{t-1}p1(s)p(s\rightarrow s',t,\pi)ds$ [15], the performance objective is

$$J(\pi_\theta)=\int_S p^\pi\int_A\pi_\theta(s,a)r(s,a)dads\,\mathrm{J}(\pi\Theta) \quad (1)$$
$$=E_{s\sim p^\pi,a\sim\pi_\theta}[r(s,a)]$$

where $E_{s\sim p}[\bullet]$ is the expected value over the discounted state distribution $p(s)$.

The basic idea behind the PG is to adjust the parameters θ of the policy in the direction of the performance gradient $\Delta_\theta J(\pi_\theta)$ [15]. The fundamental result is

$$\Delta_\theta J(\pi_\theta)=\int_S p^\pi(s)\int_A\Delta_\theta\pi_\theta(a|s)Q^\pi(s,a)dads \quad (2)$$
$$=E_{s\sim p^\pi,a\sim\pi_\theta}\left[\Delta_\theta\log\pi_\theta(a|s)Q^\pi(s,a)\right]$$

One issue of the PG is that it must address how to estimate the action-value $Q^\pi(s,a)$. One simplest method is to exploit a sample return r_t^γ to estimate the value of $Q^\pi(s_t,a_t)$. An off-policy actor-critic is useful to estimate the PG over the trajectories sampled from a distinct policy $B(a|s)\neq\pi_\theta(a|s)$. In the off-policy actor-critic, the performance is modified to become the value function of the target policy [15].

$$J_\beta(\pi\theta)=\int_S p^\beta(s)V^\pi(s)ds \quad (3)$$
$$J(\pi_\theta)=\int_S\int_A p^\beta(s)\pi_\theta(a|s)Q^\pi(s,a)dads$$

Differentiating the performance and applying an approximation makes the off-policy policy-gradient [15]

$$\Delta_\theta J(\pi_\theta)\approx\int_S\int_A p^\beta(s)\Delta_\theta\pi_\theta(a|s)Q^\pi(s,a)dads \quad (4)$$

$$=E_{s\sim p^\beta,a\sim\pi_\beta}\left[\frac{\pi_\theta(a|s)}{\beta_\theta(a|s)}\Delta_\theta\log\pi_\theta(a|s)Q^\pi(s,a)\right] \quad (5)$$

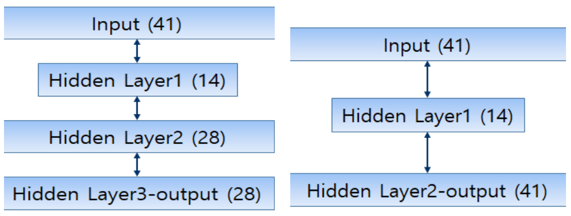
This approximation are according to the action-value gradient $\Delta_\theta Q^\pi(s,a)$ [15]. The off-policy actor-critic uses a policy $\beta(a|s)$ to generate trajectories.

III. The Proposed Algorithm

In this section, we propose a definition of one of PG strategies, Actor-Critic with non-symmetric deep auto-encoder to find an optimal behavior strategy for the agent to obtain optimal rewards with the bottommost environmental resources. In deep learning researches, the specific strategy of a model has been dictating its success. However, research studies are unable to explain what makes a successful deep learning architecture. Our purpose has been shown that despite of being succeeded in terms of deep learning, there is still room for improvement, in particular, such as the minimum requirement of resources. Although, the theoretical approaches are seldom in terms of success of deep learning, so many empirical results, such as the research [9] show that it is worth to keep exploiting the deep learning. Because RL is popular unexpectedly, combining the deep learning area and RL is still in an on-going development stage. Most recent research groups are experimenting on combining various algorithms (e.g. training, optimization, activation and classification) and layering approaches to produce the most accurate and efficient solution for specific data-sets [4]-[6].

Therefore, the proposed non-symmetric deep auto-encoder learning model with Actor-Critic can make a valid contribution to both deep learning [9][10] and RL [1][2]. Our model is based on the PG of RL for achieving the best learning performance continuously in a control-based program model using small human demonstrations and preferences similar to [8] in a budget resource. The proposed PG model can take actions for higher learning performance in those resources.

Fig. 1 shows the comparison between the non-symmetric deep auto-encoder and the typical deep auto-encoder. Fig. 2 shows the comparison between (a) the normal deep neural network [18] and the non-symmetric deep auto-encoders (NDAE) [9] for our deep learning model combined with Actor-Critic [1][2]. Fig. 3 shows the proposed Actor-Critic with NDAE. The NDAE has three hidden layers, with each hidden layer using the different number of neurons as that of features.



(a) (b)

Fig. 1. (a) Non-symmetric deep auto-encoder in [9], (b) Typical deep auto-encoder

```
#(a)Deep Neural Network in [18]
    def build_actor(self):
        # Deep neural network
        actor = Sequential()
        actor.add(Dense(24, input_dim=self.state_size, activation='relu',
                        kernel_initializer='he_uniform'))
        actor.add(Dense(self.action_size, activation='softmax',
                        kernel_initializer='he_uniform'))
        return actor
```

(a)

```
#(b)Non-Symmetric Deep Auto-Encoder in [9] for our proposed model
    def build_actor(self):
        model = Sequential()
        model.add(Dense(41, input_dim=self.state_size, activation='relu'
                        kernel_initializer='he_uniform'))
        model.add(Dense(14, activation='relu'))
        model.add(Dense(28, activation='relu'))
        model.add(Dense(28, activation='relu'))
        model.add(Dense(self.action_size, activation='softmax',
                        kernel_initializer='he_uniform'))
        return model
```

(b)

Fig. 2. (a) Deep neural network in [18], (b) Non-symmetric deep auto-encoder in [9] for our proposed model

```
Algorithm: An Actor-Critic for a classic control-based game
Require: Training step T, discount factor $\gamma$
Ensure: the non-symmetric deep auto-encoder, the action,
the state, and the rewards
Initialize linear value function approximation $Qw(s, a) = \phi(s, a)^T w$
with random weights
  for episode =1, $M$ do
      Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = (s_1)$
      Sample a ~ $\pi_\theta$
      for t=1, $T$ do
          Sample reward $r_t = R(s, a)$ and the next state $s' \sim P(s'| s, a)$
          Sample the next action a' ~ $\pi_\theta(a'|s')$
          Update the policy parameters by non-symmetric auto-encoder
              in Fig. 3(b)
              $\theta \leftarrow \theta + \alpha_\theta \phi(s, a)^T w \ \nabla_\theta ln \pi_\theta(a|s)$
          Compute the correction (TD error) for action-value at a time t,
              $\phi(s, a)^T w$, by non-symmetric auto-encoder in Fig. 3(b)
              $\delta t \leftarrow r_t + (s', a')^T w - \phi(s, a)^T w$
          Use $\delta t$ for updating the parameters of action-value function
              $w \leftarrow w + \alpha_w \delta t \nabla_w \phi(s, a)^T w$
          Update $a \leftarrow a'$ and $s \leftarrow s'$
      end for
end for
```

Fig. 3. Proposed actor-critic with NDAE

This proposed model exploits the NDAE, which features non-symmetrical multiple hidden layers introduced by [9]. Given by the learning architecture, it is possible to reduce both computational resources with minimum computational resources in a budget resource. There are input vectors to NDAE, the states, the output, and the Q-values for each action. The reward for every action, $r_t = E(\theta) = \ (x^{(i)} - y^{(i)})^2$ following the equation in [9].

The difference between the Q-values predicted by a deep neural network and the target Q-values given by the PG equation is considerable to be the error (or loss function) of prediction by the deep neural network. However, the proposed NDAE would have adjusted its weights to predict correct $Q(s,a)$ values, so that the agent would be able to act with optimal policies in any given state.

The proposed NDAE to learn $Q(s,a)$ values are implemented using TensorFlow [13] and Keras [14]. The proposed model emulates the environment with OpenAI Gym and takes actions such as “left” or “right” and receives rewards from the simulated environment such as positive rewards for “success” and negative rewards for “fail” for better training results. Because our proposed model is relying on the OpenAI Gym, the states of the environment are from data-sets in CartPole-V1 and rewards are from the same environment by calculating all sums of the difference expectations of NDAE as an approximation function.

Fig. 4 shows comparison between the normal deep neural network (DNN) with Actor-Critic and the non-symmetric deep auto-encoder (NDAE) with Actor-Critic in “CartPole-v0”. The maximum number of episodes is 500, which is for prevention of a unlimited run. In terms of the worst cases, both DNN and NDAE are similar. DNN and NDAE could not reach the topmost reward, 500 in a row until the end of episodes. However, in terms of best cases, our proposed model NDAE with Actor-Critic is shorter than DNN with Actor-Critic. However, in most cases, which means the normal cases, there is a tiny difference between DNN and NDAE.

Because of randomness of runs in the RL, our major concerns focus on the qualitative differences, not the quantitative ones. Our next step is that we attempt to find out the perturbed variables in terms of hyper-parameters for the optimized policy, π. Table. 1 shows the quantitative comparison of DNN and NDAE. Most cases are similar. However, in therms of “in score 150”, NDAE is better than DNN.

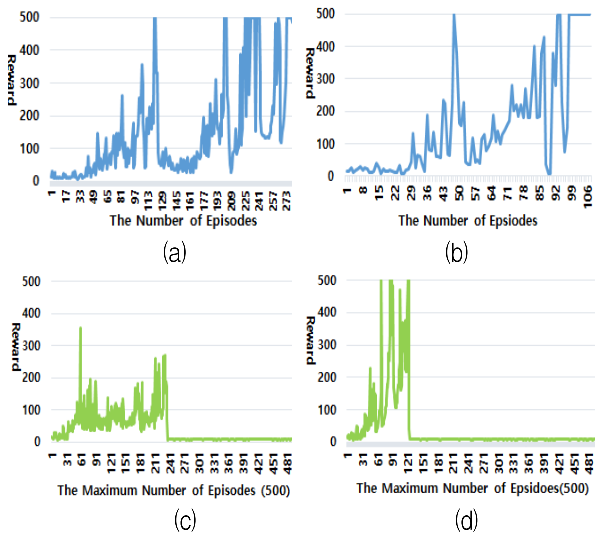


Fig. 4. Comparison between the worst cases (a) and (b) and the best cases (c) and (d)
(a) Best case of deep neural network, (b) Best case of non-symmetric deep auto-encoder, (c) Worst case of deep neural network, (d) Worst case of non-symmetric deep auto-encoder

Table 1. Quantitative comparison of (a) Deep neural network and (b) Non-symmetric auto-encoder

The best cases (in between 100&200)	The Ave. cases (in between 300&500)	The worst cases (Never ending until max score 500)
(a) 12%, (b) 12%	(a) 86%, (b) 86%	(a) 2%, (b) 2%
In score 150, (a) 40%, (b) 60%		

Ⅳ. Conclusion

We have suggested an Actor-Critic, one of PG strategies with a non-symmetric deep auto-encoder (NDAE) for a control-based program model by using

OpenAI Gym environment. The Actor-Critic based on RL with non-symmetric deep auto-encoder is trained with the minimum computational resources by using only a laptop with a GPU. There have been researches for the better results, however, there is still room for improvement, in particular, such as combining small human demonstrations and preferences into the simulation environment for better exploration. Our proposed model demonstrates improvements with the minimum limit over the previously developed deep neural network combined Actor-Critic.

References

[1] R. S. Sutton and A. G. Barto, "Reinforcement learning", An introduction, Vol. 1. MIT press Cambridge, 1998.

[2] R. S. Sutton, "Temporal credit assignment in reinforcement learning", Doctoral Dissertation, 1984.

[3] Open AI Gym Toolkit, https://github.com/openai/gym [accessed: Jan. 05. 2019]

[4] R. Chuchro, and D. Gupta, Game Playing with Deep Q-Learning using OpenAI Gym, cs231n. stanford. edu/reports/2017/pdfs/616.pdf. 2017.

[5] D. Silver and A. Huang, et al., "Mastering the game of go with deep neural networks and tree search", Nature, Vol. 529, No. 7587, pp. 484-489, Jan. 2016.

[6] V. Mnih and K. Kavukcuoglu, et al.,, "Playing atari with deep reinforcement learning", eprint arXiv:1312.5602, NIPS Deep Learning Workshop 2013, Dec. 2013.

[7] Cart-Pole, https://github.com/openai/gym/wiki/Cart-Pole-v0 [accessed: Jan. 05. 2019]

[8] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei, "Reward learning from human preferences and demonstrations in Atari", NIPS 2018, https://arxiv.org/abs/1811.06521. Nov. 2018.

[9] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection", IEEE Transactions on Emerging Topics in Computational Intelligence, Vol. 2, No. 1, pp. 41-50, Feb. 2018.

[10] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude", Coursera: Neural Networks for Machine Learning, Vol. 4, No. 2, pp. 26-31, Oct. 2012.

[11] D. Kimura, "DAQN: Deep Auto-Encoder and Q-Network", IBM Research AI, arXiv:1806.00630v1, 2018.

[12] X. Zhao, J. Wu, Y. Zhang, Y. Shi, and L. Wang, "Fault Diagnosis of Motor in Frequency Domain Signal by Stacked De-noising Auto-encoder", CMC: Computers, Materials & Continua, Vol. 57, No. 2, pp. 223-242, Jan. 2018, doi:10.32604/cmc.2018.02490

[13] Tensorflow, https://github.com/tensorflow/tensorflow [accessed: Oct. 31. 2018]

[14] Keras, https://keras.io/ [accessed: Oct. 31. 2018]

[15] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic Policy Gradient Algorithms", Proceeding of International Conference on Machine Learning (ICML), Beijing, China, Vol. 32, pp. I-387-I-395, Jun. 2014.

Author

Chayoung Kim

Feb. 2006 : Ph.D., Computer Science, Korea University

Nov. 2005. ~ Dec. 2008. : Senior Project Researcher, KISTI

Mar. 2009. ~ Feb. 2018. : Adjunct Professor, Dept. of Computer Science, Kyonggi University

Mar. 2018. ~ present : Assistant Professor, Div. of General Studies, Kyonggi University

Reserach Interest : Big-Data, Machine Learning, Deep Learning, Reinforcement Learning