Asynchronous Advantage Actor-Critic Algorithms Based on Residual Networks

Lili Tang
Suzhou Polytechnic Institute of Agriculture, Xiyuan Road 279, Suzhou, Jiangsu, 215008, China
Email: noel1031@hotmail.com

Abstract. Deep reinforcement learning is one of the fastest-growing technologies in machine learning. The Asynchronous Advantage Actor-Critic algorithm completely uses the actor-critic framework and utilizes the idea of asynchronous training, which greatly speeds up the training and improves performance. Although A3C algorithm puts actor-critic into multiple threads to train synchronously, effectively utilizes computer resources and improves training effectiveness, it is still difficult to train in deep neural network. Deep networks have proved to be capable of extending to thousands of layers and still have improved performance. However, every one percent increase in accuracy almost doubles the cost of layers, so it is not easy for A3C to train both actor and critic networks. In response to this problem, we innovatively utilize the residual network to apply to the asynchronous advantage actor-critic algorithm and has achieved improvement greatly in the inverted pendulum problem.

1. Introduction
In recent years, deep learning has been widely used in the fields of computer vision (1), speech recognition and synthesis (2), natural language processing (3), and robotics (4). Deep learning typically utilizes multiple layers of neural networks and nonlinear transformations to automatically learn the perception and expression of high-dimensional input data. As another research focuses in machine learning, reinforcement learning has been widely used in games (5), simulation (6), industrial control (7), robot control (8) and parameter optimization (9).

In reinforcement learning, the agent continuously interacts with the environment through trial and error forms to obtain feedback signals and learns the optimal strategy for solving problems by maximizing the cumulative reward. With the rapid development of human society, more and more complex decision-making problems need to use the powerful perception ability of deep learning to extract the abstract features of large-scale input data, self-inspired reinforcement learning based on this feature, and finally solve the optimal strategy for the problem. Therefore, deep learning with perceptual ability and reinforcement learning with decision making ability can be combined to form a deep reinforcement learning method.

The deep reinforcement learning combines the perceptual ability of deep learning with the decision-making ability of reinforcement learning. It belongs to an end-to-end intelligent system directly from input sensing signals to output control actions and is strong in solving large-scale decision problems. At each moment, the agent obtains a high-dimensional observation from the environment and uses the deep learning method to perceive the observation to obtain abstract, low-dimensional state features. The algorithm evaluates the value function of each action based on the reward signal obtained by the interaction between the agent and the environment and maps the current state to the corresponding action through a certain strategy. The environment reacts to the current...
action to obtain next observation. By continuously cycling through the above process, the final agent can get the maximum cumulative return value.

As a classic algorithm in deep reinforcement learning, the deep Q-network has been paid close attention by researchers and made some breakthroughs in theory and application. However, limited by the limitations of the algorithm itself, the deep Q network will still suffer from inaccurate Q estimation, low utilization of valuable samples and balance of exploration and utilization, etc. when completing some visual perception-based decision tasks. In response to the above problem, Hasselt et al. proposed the Deep Double Q-Network (DDQN) algorithm, which uses two different sets of network weights to separate the action selection from the policy evaluation to reduce the risk of overestimating the Q value. Hausknecht researched the effects of adding recurrence to a Deep Q-Network (DQN) by replacing the first post-convolutional fully connected layer with a recurrent LSTM. Hasselt et al. examined six extensions to the DQN algorithm and empirically studies their combination to perform well on the Atari 2600 benchmark.

By the limitations of the deep Q network method itself, the above related algorithms and models are only applicable to decision tasks in discrete action spaces. Therefore, many related works have studied how to extend the application scenario of the deep reinforcement learning method to the continuous action space scenario through the improvement of the algorithm framework. The actor-critic algorithm is more straightforward in solving reinforcement learning problems. It directly looks for a strategy and then uses a value function to evaluate the strategy. Because its output is a probability distribution, it can be used to solve continuous problems. One of the most common and widely used variants of the Actor-Critic algorithm is the asynchronous advantage actor-critic algorithm. It’s proposed by DeepMind in 2016, which improves the training speed of the model. Wu et al. proposed to apply trust region optimization to advantage actor-Critic algorithms using a Kronecker-factored approximation to the curvature. Shao et al. present the synchronous advantage actor-critic (A2C) with generalized advantage estimator (GAE) algorithm to tackle visual navigation tasks.

2. Related Work

2.1. Residual Network

The residual network (ResNet) is a deep convolutional network proposed in 2015. The residual network is easier to optimize and can increase the accuracy by increasing the depth. The core of residual network is to solve the degradation problem caused by the increase in depth, which can improve network performance by simply increasing the network depth.

The residual network draws on the idea of a cross-layer link in the High-Speed Network. It is improved by replacing original weight with an identity map. Suppose that the input of a certain neural network is x, and the expected output is $F(x)$ which is expected complex potential mapping. If it is to learn such a deep model, the training difficulty will be relatively large.

However, if ResNet have learned the more saturated accuracy or the underlying error is found to be large, the learning goal is transformed into the learning of the identity map, that is, the input x is approximated to the output $F(x)$, which will not result in a decrease in accuracy.

![Figure 1. The structure of residual network](image-url)
As is shown in Figure 1, the input \( x \) is directly passed to the output as the initial result by the “shortcut connections”, and the output result is

\[
F(x) = G(x) + x
\]  

(1)

When \( G(x)=0 \), \( F(x)=x \), which is the identity map mentioned above. Therefore, ResNet is equivalent to changing the learning target. Instead of learning a complete output, ResNet learns the difference between the target value \( G(x) \) and \( x \), which is called the residual \( G(x):=F(x)-x \). The latter training goal is to approximate the residual result to 0, so that the accuracy does not decrease as the network deepens.

This kind of residual structure breaks the convention that the output of the traditional neural network \( n-1 \) layer can only give the \( n \) layer as an input, so that the output of ascertain layer can directly cross several layers as the input of the latter layer. It provides a new direction for the problem of superimposing the multi-layer network and making the error rate of the whole learning model not fall.

The equivalent mapping function may not be so well optimized, but for residual learning, the solver will be more likely to find the perturbation based on the equivalent mapping of the input. In short, it is much easier than directly learning an equivalent mapping function. It can be found that the learned residual function usually has a small response value, and the equivalent map provides a reasonable precondition.

2.2. Asynchronous Advantage Actor-Critic Algorithm

Reinforcement learning \cite{18} is a method that deals with sequential decision-making tasks, which solves the problem of decision-making optimization by obtaining the maximum cumulative reward. The agent performs autonomous learning according to the observed environmental state. So, the process satisfies the learning conditions of the Markov decision process (MDP) \cite{19}. Reinforcement learning includes action value fitting method and action probability-based learning method. Reinforcement learning based on action value fitting includes Q-learning algorithm, Sarsa algorithm and so on. Reinforcement learning based on action probability includes strategic gradient method. The actor-critic algorithm combines the value function learning method and the strategy gradient learning method, with the strategy gradient method as the actor algorithm for action selection; the value function method as the critic Algorithm for commenting on the quality of the action.

Unlike symmetric encryption algorithm, asymmetric encryption algorithm requires two keys. One is a public key and the other is a private key. The public key and the private key are a pair. If the data is encrypted with the public key, only the corresponding private key can be decrypted; if the data is encrypted with the private key, only the corresponding public key can be decrypted. Because encryption and decryption use two different keys, this algorithm is known as asymmetric encryption algorithm.

The training strategy and value network for the policy network expresses the possibility of each optional action in state \( s \), so the loss function is not easy to construct. One of the simplest ideas is to adjust the probability of an action occurring based on the size of the reward after the action is performed. The greater the reward, the greater the probability of the action appearing. An evaluation network can be constructed to judge the quality of the current decision to guide the training of the strategy network, which is the basic idea of the actor-critic model. The structure is shown in Figure 2.
The actor-critic model is mainly composed of actor part and critic part. The input of critic part is the game state \( s_t \) and the output is an estimate of the state \( s_t \). This estimate represents the average expected value for a given state, and its mathematical representation is as Equation 2. Therefore, the critic part can be iteratively updated, and the general loss function also uses the squared error. Refer to the parameter update equation of the DQN network above to get the Critic update Equation 3. \( \theta_v \) represents the parameters of the critic network:

\[
V(s) = E[G_t \mid s_t = s] \\
= E[R_t + \gamma V(S_{t+1}) \mid s_t = s]
\]

\[
\theta_v \leftarrow \theta_v - \alpha \nabla_{\theta_v} L(\theta_v)
\]

The time difference error \( TD = r_t + \gamma V(S_{t+1}) - V(S_t) \) in critic part is used as the average of the action \( a_t \) which the actor selected in state \( s_t \) is better than the given state. The evaluation of the cumulative reward, according to the evaluation result adjusts the probability of the action \( a_t \), constructs the loss function of the Actor network as the Equation 4, and uses \( \theta_{\pi} \) to represent the parameters of the Actor network:

\[
L(\theta_{\pi}) = - \sum \log(\pi(a_t \mid s_t))^*TD
\]

When the ratio is better than the average value, the probability of occurrence of the action should be increased. Currently, the TD is positive, the log function part is negative, and the minimum loss function \( L(\theta_{\pi}) \) satisfies the target. When the action is inferior to the average value The parameter update of the network of actor part is obtained by the Equation 4 as the Equation 5:

\[
\theta_{\pi} \leftarrow \theta_{\pi} + \alpha TD \nabla_{\theta_{\pi}} \log \pi(a_t \mid s_t)
\]

Since the network of actor part takes the probability output, this makes the model more applicable and more suitable for solving the problem of continuous action. In addition, the use of probabilistic output allows the model to have more and more reasonable action choices under the same situation.

3. Asynchronous Advantage Actor Critic Algorithm Based on Residual Network

3.1. Experimental Platform

This paper proposes an asynchronous superior actor critic algorithm based on residual network. At present, few researchers have applied the residual network to the asynchronous dominant actor critic algorithm. The residual network is a widely used technology in the modern era. Based on the application of asynchronous superior actor critic, this paper adds a residual network, combines the residual network with the asynchronous superior actor critic, and applies it to the simulation
environment, which improves the convergence speed and saves a lot of time. This section describes the specific training process of the RN-A3C algorithm.

The game environment used in this article is based on the inverted pendulum environment in the gym toolkit developed by the artificial intelligence company OpenAI. The gym package is a toolkit for developing and comparing reinforcement learning algorithms. It provides a variety of game environment interfaces and provides researchers with a variety of experimental platforms. The computer processor used in this experiment is Intel i7-7820X, and the memory is 16GB. In addition, since the convolution operation and matrix operation are mostly used in the model, the GTX 1080Ti GPU is used to assist the model in accelerating operations.

3.2. Model Structure and Analysis
The asynchronous dominant actor critic algorithm based on the residual network uses the residual network model architecture. Since the low-dimensional state feature representation (position, velocity, etc.) is used in the experiment, only a shallow neural network model is needed to represent the value network and the policy network, respectively. Both the value network and the policy network are set up as a fully connected network with one residual network and two layers of hidden units, each with 500, 400 and 300 neurons, and each hidden layer uses the RELU function. Perform a nonlinear transformation. In order to ensure that the output values of the initial time value network and the policy network are close to zero, the output layer weights of the two networks are initialized to a uniform distribution within the interval of $[-3 \times 10^{-3}, 3 \times 10^{3}]$. The weights of the input layer and the hidden layer are initialized to $[-(\sqrt{f_{in}})^{-1}, (\sqrt{f_{in}})^{-1}]$, where $f_{in}$ represents the input value of the current layer.

The innovation and strength of the residual network-based asynchronous dominant actor commentator algorithm lies in that RNA3C algorithm adopts the idea of asynchronous training to improve the training speed, using multiple threads. Each thread is equivalent to an agent exploring randomly, multiple agents exploring together, parallel computing strategy gradient, updating parameters. In other words, start multiple training environments at the same time, sample at the same time, and train directly with the collected samples. Here, data are obtained asynchronously. Compared with the original algorithm, A3C algorithm does not need to use experience pool to store historical samples and randomly extract training to disturb data correlation, save storage space, and use asynchronous training and residual network, greatly doubling. The speed of data sampling is improved, and the training speed is also improved. At the same time, samples are collected in different training environments, and the distribution of samples is more uniform, which is more conducive to the training of neural networks. We propose asynchronous advantage actor-critic algorithms based on residual networks, as algorithm 1.

**Algorithm 1:** Synchronous Advantage Actor-Critic Algorithms Based on Residual Networks(RNA3C)
Assume global shared parameter vectors $\theta_{\pi}$ and $\theta_{v}$, global shared counter $T=0$
Assume thread-specific parameter vectors $\theta'_{\pi}$ and $\theta'_{v}$

**Input:** the initial parameter vectors $\theta_{\pi}$ and $\theta_{v}$

**Output:** the updated parameter vectors $\theta_{\pi}$ and $\theta_{v}$

1. **Repeat:**
2. Reset gradients: $d\theta_{\pi} \leftarrow 0, d\theta_{v} \leftarrow 0$
3. Synchronize thread-specific parameters $\theta'_{\pi} \leftarrow \theta_{\pi}$ and $\theta'_{v} \leftarrow \theta_{v}$
4. $t_{\text{start}} \leftarrow t$
5. Get state $s_{i}$
6. **Repeat:**
7. Perform $a_{i}$ according to policy
8. Receive reward $r_{i}$ and new state $s_{i+1}$
9. \( s_t \leftarrow s_{t+1} \)
10. \( t \leftarrow t + 1 \)
11. \( T \leftarrow T + 1 \)
12. \textbf{Until} terminal \( s_t \) or \( t - t_{\text{start}} = t_{\text{max}} \)
13. \( R \leftarrow \begin{cases} 0, & \text{for terminal } s_t \\ V(s_t; \theta'), & \text{for non-terminal } s_t \end{cases} \)
14. \textbf{For} \( i = t - 1 \) to \( t_{\text{start}} \) \textbf{Do:}
15. \( R \leftarrow r_i + \gamma R \)
16. Accumulate gradients wrt \( \theta'_{\pi} \) using residual network
17. Accumulate gradients wrt \( \theta'_{v} \) using residual network
18. \textbf{End For}
19. Perform asynchronous update of \( \theta_{\pi} \) using \( d\theta_{\pi} \) and \( \theta_{v} \) using \( d\theta_{v} \)
20. \textbf{Until} \( T > T_{\text{max}} \)
21. Return \( \theta_{\pi} \) and \( \theta_{v} \)

### 3.3. Experimental Results and Analysis

In the experiment, RNA3C algorithm is applied to the task of two-stage inverted pendulum to verify the performance of the algorithm in the continuous action space problem. In addition, in order to fully illustrate the advantages of RNA3C algorithm compared with traditional strategy optimization methods, the experiment chooses the optimization algorithm such as A3C algorithm to solve the continuous action space problem as the comparative experiment. In order to ensure the consistency of parameters, the different algorithms use 1200 stages as the training cycle. Each stage contains 500 time steps for parameter updates.

The performance of RNA3C algorithm in two-stage inverted pendulum problem is analysed. Because RNA3C is based on the improvement of A3C algorithm, the performance difference between the two algorithms is mainly analysed. As can be seen from Figure 3, the convergence speed of RNA3C algorithm is much faster than that of A3C algorithm. Specifically embodied in: RNA3C algorithm in the balance bar task needs about 400 training stages to achieve convergence, A3C algorithm needs about 800 training stages to barely converge. The main reason for accelerating convergence speed is that RNA3C algorithm uses residual network to increase the utilization rate of valuable transfer samples in the early stage of training, thus promoting the learning of agents. In addition, we can see from Figure 3 that the performance of RNA3C algorithm is very stable. This shows that the performance curve of RNA3C algorithm can keep rising smoothly before convergence and stable after convergence, while the performance curve of A3C algorithm has been fluctuating during training. The reason for the stability improvement of RNA3C algorithm lies in: on the one hand, the residual network is used to make every transfer sample in the sample space accessed with a non-zero probability, which reduces the risk of the algorithm falling into the local optimal solution to a certain extent; on the other hand, the algorithm combines the method of A3C to reduce the error of estimating the target Q value.
RNA3C algorithm can still achieve faster convergence speed and more stable performance in tasks with larger state feature space. In traditional strategy optimization methods, the number of trainable weights contained in the linear approximator is insufficient, and the problem of under-fitting and local optimal solution is prone to occur in the face of large-scale state space decision-making tasks. In the RNA3C algorithm, the strategy network and value network with more weights are used to generalize the strategy and value function in large-scale state space. Therefore, RNA3C algorithm has more obvious advantages in convergence speed and stability in the face of decision-making tasks. In summary, RNA3C algorithm has the advantages of fast convergence, good performance and high stability in continuous action space tasks.

4. Conclusion

Aiming at the difficulty of traditional depth deterministic strategy gradient method in estimating target Q value, we propose a residual network based A3C. This paper applies RNA3C algorithm to the classical continuous action space task problem. From the experimental results, it can be concluded that RNA3C algorithm has the advantages of fast convergence and stable performance in solving control tasks in continuous action space.

5. References

[1] Szegedy C, Vanhoucke V, Ioffe S, et al. [IEEE 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) - Las Vegas, NV, USA (2016.6.27-2016.6.30)] 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) - Rethinking the Inception Architecture for Computer Vision[J]. 2016:2818-2826.
[2] Noda K, Yamaguchi Y, Nakadai K, et al. Audio-visual speech recognition using deep learning[J]. Applied Intelligence, 2015, 42(4):722-737.
[3] Goldberg Y. A primer on neural network models for natural language processing[J]. Journal of Artificial Intelligence Research, 2016, 57(1): 345-420.
[4] Lenz I, Lee H, Saxena A, et al. Deep learning for detecting robotic grasps[J]. The International Journal of Robotics Research, 2015, 34(4): 705-724
[5] Lanctot M, Zambaldi V F, Gruslys A, et al. A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning[J]. neural information processing systems, 2017: 4190-4203.
[6] Marco A, Berkenkamp F, Hennig P, et al. Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization[J]. international conference on robotics and automation, 2017: 1557-1563.
[7] Liu L, Wang Z, Zhang H, et al. Adaptive Fault-Tolerant Tracking Control for MIMO Discrete-Time Systems via Reinforcement Learning Algorithm With Less Learning Parameters[J]. IEEE Transactions on Automation Science and Engineering, 2017, 14(1): 299-313.
[8] Gu S, Holly E, Lillicrap T P, et al. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates[J]. international conference on robotics and automation, 2017: 3389-3396.

[9] Zoph B, Vasudevan V, Shlens J, et al. Learning Transferable Architectures for Scalable Image Recognition[J]. computer vision and pattern recognition, 2018: 8697-8710.

[10] Hester T, Vecerik M, Pietquin O, et al. Deep Q-learning from Demonstrations[J]. national conference on artificial intelligence, 2018: 3223-3230.

[11] Van Hasselt H, Guez A, Silver D, et al. Deep reinforcement learning with double Q-Learning[J]. national conference on artificial intelligence, 2016: 2094-2100.

[12] Hausknecht M J, Stone P. Deep Recurrent Q-Learning for Partially Observable MDPs.[J]. national conference on artificial intelligence, 2015: 29-37.

[13] Hessel M, Modayil J, Van Hasselt H, et al. Rainbow: Combining Improvements in Deep Reinforcement Learning[J]. national conference on artificial intelligence, 2018: 3215-3222.

[14] Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning[J]. international conference on machine learning, 2016: 1928-1937.

[15] Wu Y, Mansimov E, Grosse R B, et al. Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation[J]. neural information processing systems, 2017: 5279-5288.

[16] Shao K, Zhao D, Zhu Y, et al. Visual Navigation with Actor-Critic Deep Reinforcement Learning[C]. international joint conference on neural network, 2018: 1-6.

[17] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition[J]. computer vision and pattern recognition, 2016: 770-778.

[18] Sutton R S, Barto A G. Reinforcement Learning: An Introduction[C]. neural information processing systems, 1999.

[19] Turchetta M, Berkenkamp F, Krause A, et al. Safe exploration in finite markov decision processes with Gaussian processes[J]. neural information processing systems, 2016: 4312-4320.