Deep Reinforcement Learning of the Model Fusion with Double Q-learning

KANG WANG, WEI ZHANG, XU HE and SHENG GAO

ABSTRACT

Q-learning algorithm is a one-step algorithm, which can overestimate the function of action value in some cases. Aiming at this problem, this paper uses double Q-learning algorithm and the deep neural network to form deep Q network. Then combining with the different structure of deep neural network method to reduce sample correlation and increasing sample size in the experience bank. And it is tested in the Atari 2600 series video games, and it turns out that the method has the effect on some games.

KEYWORDS

Double q-learning, deep neural network, model fusion, neural network.

INTRODUCTION

Q-learning is one of the most popular reinforcement learning algorithms, and the goal of reinforcement learning [1, 2] is a good strategy for learning continuous decision-making problems by optimizing the accumulated future reward signals. But it is well known that sometimes you learn to fail to meet the actual high value of action value functions, because it includes a maximum step that tends to be highly valued. In previous work, overestimations have been attributed to insufficiently flexible function approximation and noise [3, 4]. Studies show that an overestimation of the action value is expected to occur, and it has a negative effect on performance in practice.

We investigate the recent performance of deep Q network (DQN) algorithm [5]. DQN is that q-learning combination of deep neural network and is tested in the Atari 2600 games in many games and it reaches human level. DQN provides the application scenarios for q-learning, providing flexible function approximation for potential approximation errors, and the deterministic environment which can prevent the harmful effects of noise.

Kang Wang, 785456510@qq.com, School of Shenyang Ligong University, Shenyang 110168, China
Wei Zhang, zhangwei@sia.cn, Xu He, hexu@sia.cn, Sheng Gao, gaosheng@sia.cn
Shenyang Institute of Automation Chinese Academy of Sciences, Shenyang 110016, China
Although with more advantageous setting, DQN sometimes overestimates the action value function. The literature propose double q-learning algorithm [6]. Double q-learning that can be generalized to arbitrary function approximation, including deep neural networks. We apply what we call a double DQN. Based on the double q-learning algorithm we adds different models of neural networks to form a fusion frame module, considering the different neural network structures increase the diversity of samples and experience on playback mechanism in the process of sampling. In this way reducing the correlation of the samples and improving the stability of the training process. The proposed method is simulated in Breakout, Pong, Seaquest, and Space invaders.

**RELATED WORK**

Section Headings.

To solve the problem of continuous decision making, the evaluation of each action can be learned, which is based on the future reward of each action under a certain strategy[7]. Given a policy $\pi$, action $a$ is under state $s$:

$$Q_\pi(s, a) \equiv E[R_1 + \gamma R_2 + \ldots | S_0 = s, A_0 = a, \pi]$$  \hspace{1cm} (1)$$

$Q_\pi(s, a)$ is action value function, $E$ is mathematical expectation, $\gamma \in [0, 1]$, it is a discount factor used to weight the importance of immediate returns $R$ and future long-term returns. The optimal strategy $\pi$ is to select the highest action value function in each state. Due to continuous problems different state may have many different movements, it is impossible for us to learn all action value function, but we can use with the value of the parameter function $Q(s, a; \theta)$, parameters of the update:

$$\theta_{t+1} = \theta + \alpha(Y_t^Q - Q(S_t, A_t; \theta))\nabla_\theta Q(S_t, A_t; \theta)$$  \hspace{1cm} (2)$$

$\alpha$ is the learning rate, $Y_t^Q$ is the objective function, its definition is:

$$Y_t^Q = R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta)$$  \hspace{1cm} (3)$$

$Q(S_t, A_t; \theta)$ is infinitely close to the target function $Y_t^Q$ by stochastic gradient descent.

In the DQN composed of deep neural network and q-learning, the most important one is the use of the experience playback and target network, and the deep Q network uses Q function with $\theta$ parameters $Q(s, a; \theta)$ to approximate the value function. The function of loss function is as follows:

$$L_t(\theta) = E_{(s, a, r, s')}[(Y_t^{Q_{\text{DQN}}} - Q(s, a; \theta))^2]$$  \hspace{1cm} (4)$$

Among them:
\[ y_{i}^{\text{DQN}} = r + \gamma \max_{a} Q(s, a'; \theta^{-}) \] (5)

\( \theta \) represents the network parameters in the learning process. After a period of study, new \( \theta \) update \( \theta^{-} \). Specific learning process

\[ \nabla_{\theta} L_{\gamma}(\theta) = E_{(s, a, r, s')}[(r + \gamma \max_{a} Q(s', a'; \theta^{-}) - Q(s, a; \theta))\nabla_{\theta} Q(s, a; \theta)] \] (6)

The work is of great significance to deep reinforcement learning. By (4) and (6), because it contains a maximized step in the prediction of the action value, it leads to excessive predictive value, which makes it possible to learn the unrealistic high action value. So they use the same value to choose and evaluate an action[8], which means that the overestimate value is more likely to be selected. To solve this problem, the double q-learning will evaluate and choose separation, the experience of the two value function through random update one of them, so there are two sets of weights \( \theta \) and \( \theta^{-} \), for each update, a weight is used to determine the greedy strategy, another is used to determine its value, so the evaluation and selection in q-learning can be rewritten as:

\[ Y_{t}^{Q} = R_{t+1} + \gamma Q(S_{t+1}, \arg \max_{a} Q(S_{t+1}, a; \theta_{t}); \theta_{t}) \] (7)

The two-step q-learning can be written as:

\[ Y_{t}^{\text{DoubleQ}} = R_{t+1} + \gamma Q(S_{t+1}, \arg \max_{a} Q(S_{t+1}, a; \theta_{t}); \theta_{t}) \] (8)

Can get from (8), the action selection still rely on online weight \( \theta_{t} \), this means that, like q-learning, we can still according to the current value, using the greedy strategy is used to estimate its value. Then we use another set of weights \( \theta^{-} \) to evaluate the value of the policy, the set of the second weight, which can be updated by swapping the roles of the two weights.

**DOUBLE DQN BASED ON MODEL FUSION**

**Network structure design**

In order to reduce the calculation, the original game frame (210 * 160 pixel 128) color processing, firstly, the image RGB trichromatic image is converted to grayscale graph and the sampling, then the input image is truncated to 84 * 84 pixel, and the image area roughly cover the game area. Intercept 84 * 84 pixels because the 2D convolution requires square input. In subsequent experiments, the function is responsible for preprocessing the recent 4-frame images in the historical frame and using them as input to the network. Using four consecutive frames as network input, here choose ReLU (Rectified Linear Units) as a network activation function, ReLU
has quick computing speed, high accuracy. The functional form \( f(x) = \max(0, x) \). The input of the network is a pre-processed image of 84 * 84 * 4. The first hidden layer is composed of 32 * 8 * 8 convolution kernels, step length is 4, and use the ReLu nonlinear transformation. After the transformation of the convolution kernel, the layer has 20 * 20 * 32 nodes, and the second hidden layer contains 48 * 4 * 4 convolved nuclei, and the step length is 2. Similarly, ReLu is used to do nonlinear transformation. After the transformation of the convolution kernel, the layer has 9 * 9 * 48 nodes. The last hidden layer contains 512 fully connected ReLU units with the previous layer. The output layer unit is a linear function that is fully connected to the previous layer, and each output corresponds to a possible action.

**DRL network structure of model fusion**

Using the historical empirical data store in different structures by the experience playback mechanism to update the power values of the network, we call it the neural network of the model fusion. When the neural network is trained, the hypothesis is independent and distributed. There is a correlation between the data collected through intensive learning, and the use of these data for sequential training, and the neural network is unstable. Experience playback can break the correlation between data. The aim of this method is to improve the stability of the training process and accelerate the convergence rate. Specifically, agents still use \( \epsilon \text{-greedy} \) strategies to play games on the test set, which can lead to this uncertainty. Specifically, agents still use strategies to play games on the test set, which can lead to this uncertainty. We hope that the agent collects as much training data as possible for training by sampling from a large number of historical samples and using the empirical data of multiple different DNN (deep neural network) architectures. When the data in the sampling is replayed, different sampling methods are used and the sample is sampled as far as possible. For example, the two samples are more than 4 frames apart, and the frame that does not sample is in the terminal state, and the frame that is in the terminal state has no subsequent frame. Different network structures also indirectly increase historical samples. Different DRL (deep reinforcement learning) uses different convolutional neural network architecture:

1) The convolution step is different, with (4, 2) changed to (2, 2).
2) Change the sampling size of mini-batch from 48 to 16.
3) Change the preprocessing method of input image.
4) Different layer number or nodes in the full connection layer, change the full connection layer node from 512 to 256.

![Figure 1. Model integrates DRL flowchart.](image-url)
When we use different neural network models, in order to fully rational utilization of storage of sample data, reflect different model structures in the importance of updating parameters, we give the different models to join a Weight ($w$):

$$weight_{ij} = \frac{G_{ij}}{\sum_j G_{ij}}$$

$G_{ij}$ represents the score of the $i$th model in the $j$th game, which represents the weight of the first model. Final decision $s$ action $a$:

$$a = \arg \max_{a} \left\{ \sum_{i=1}^{\text{model set}} Q(s,a;\theta_i) \ast weight_i \right\}$$

SIMULATION ANALYSIS

The Atari 2600 simulator in openai generates 60 frames per second, and we set every 4 frames to send 1 frame, because the neural network does not process the data so fast, too much input or result in cartoon. Each Atari's game scoring evaluation standard is different, and it is designated as a unified standard, which means that the agent will make a favorable action score +1 each time, making the negative action -1, the unchanged is 0. The evaluation method of the algorithm is as follows: the game is naturally divided into multiple episode processes, and each episode starts from the frame after the resetting of the command, ends with the test to the end of the game, or more than 5 minutes of actual game time. A reinforcement learning algorithm is studied from 1000 training sessions, and then evaluates in the 200 non-learning stage, the performance of agent is measured with the mean score of the evaluation stage. As Space Invaders

For example, the game is shown as:

Figure 2. (a) Double DQN game score.

Figure 2. (b) DQN game score.
By comparing the changes in average score of two different models, it can be found that after certain training the agent scoring ability is increasing, and the overall score is getting higher and higher. Fig.2 (a) Double DQN has a higher performance score after training, and the curve changes faster shows that this method converges faster. In addition, the stability of learning is better than Fig.2 (b) DQN, indicating that these unstable factors are caused by excessively optimistic estimation of q learning. However, it can be seen that the curve shows that the noise amplitude and frequency of DRL with the model fusion is much smaller, which obviously performs better in adjusting the weight of the neural network.

To analyze game scores, we score points by the following formula:

\[
\text{score} = \frac{\text{score}_{\text{agent}} - \text{score}_{\text{random}}}{\text{score}_{\text{human}} - \text{score}_{\text{random}}}
\]

Among them, random and human scores were based on Mnih et al. (2015). The average return value of agent on each game is still full of noise. This is because we have luck factor in playing games, and in the game scoring change is bigger. We remove the game score from the random strategy in the formula. The agent still adopt \(\varepsilon\)-greedy strategy game on the test set, and using the random strategy could lead to the uncertainty. Weight change may cause strategy swept the state of the very different. The state is the graphics so different action choice can also lead to the change of the next frame, the result caused by the cumulative change is enormous.

From table 1 analysis of data, it is known that double DQN is effective in the game of high estimation phenomenon, such as in pong, seaquest, and space invaders, which are better than original DQN. In addition, the double DQN based on model fusion performs well in most games than the original double DQN and DQN. This is the advantage of breaking data correlation and increasing the diversity of samples based on fusion models. However, you can see that in the seaquest three modes are lower than those of human players, and in the following study, we can explore what type of game DQN is worse than the human player. Moudel fusion double DQN is MFD DQN, double DQN is DDQN.

<table>
<thead>
<tr>
<th>model</th>
<th>breakout</th>
<th>pong</th>
<th>seaquest</th>
<th>spaceinvaders</th>
</tr>
</thead>
<tbody>
<tr>
<td>DQN</td>
<td>1333%</td>
<td>132%</td>
<td>26%</td>
<td>131%</td>
</tr>
<tr>
<td>DDQN</td>
<td>1250%</td>
<td>139%</td>
<td>42%</td>
<td>199%</td>
</tr>
<tr>
<td>MFDDQN</td>
<td>1573%</td>
<td>163%</td>
<td>79%</td>
<td>273%</td>
</tr>
</tbody>
</table>

**CONCLUSION**

This paper introduces the overestimation phenomenon of deep reinforcement learning in the application, and this phenomenon has a negative effect on the application. Therefore, it is applied to solve this problem by using double DQN. Based on this, we design the DQN of the model fusion with different structures, and use the empirical data stored by the experience playback mechanism in different network structures to increase of sample diversity. The test results show that the algorithm not only produce more accurate estimation, but also score much higher in tested games. This suggests that the original DQN overestimate really learn isn't the best strategy. Reducing the overestimation is beneficial. The experiment proves that the double
DQN based on model fusion has successfully learned more effective control strategy in Atari 2600 game, and it is better in stability and learning effect.

ACKNOWLEDGEMENTS

Corresponding author: Wei Zhang male Researcher of shenyang institute of automation, Chinese academy of sciences; Research direction: artificial intelligence e-mail: zhangwei@sia.cn Tel: 134 7886 0406

REFERENCES