Weekly report (May 19, 2019)

Rong Ding
dingrong@sjtu.edu.cn

Abstract

These weeks I learned about policy gradient in reinforcement learning. Besides, I had some try to train the Flappybird game.

1 Policy gradient

Policy-based RL has no value function and learn policy directly. It has several advantages compared with Value-based RL, whereas also has some disadvantages.

Advantages

- Better convergence properties: we are guaranteed to converge on a local maximum (worst case) or global maximum (best case)
- Policy gradients are more effective in high-dimensional action space
- Policy gradients can learn stochastic policies, while value function can't .

Disadvantages

- typically converges to a local optimum
- evaluating a policy is inefficient and high variance

Policy gradient suffers from large variance. We can reduce it using causality and baseline.

2 Flappybird

I found a Pytorch version codes using reinforcement learning to run the Flappybird game. I read the codes carefully and made detailed comments to make sure I understand the principle. Here I want to show what I have done and problems I have met.

2.1 Tune the model

I tried to tune the model with randomly initiallized parameters at start. The bird kept going down and ended its adventure because of knocking on the ground. Generally, it learned to fly high to stay away from the floor. However, then it acted not so properly. The bird learned to fly on the top of sky and inevitably knocked to the pipe over and over again. Like the figures below:

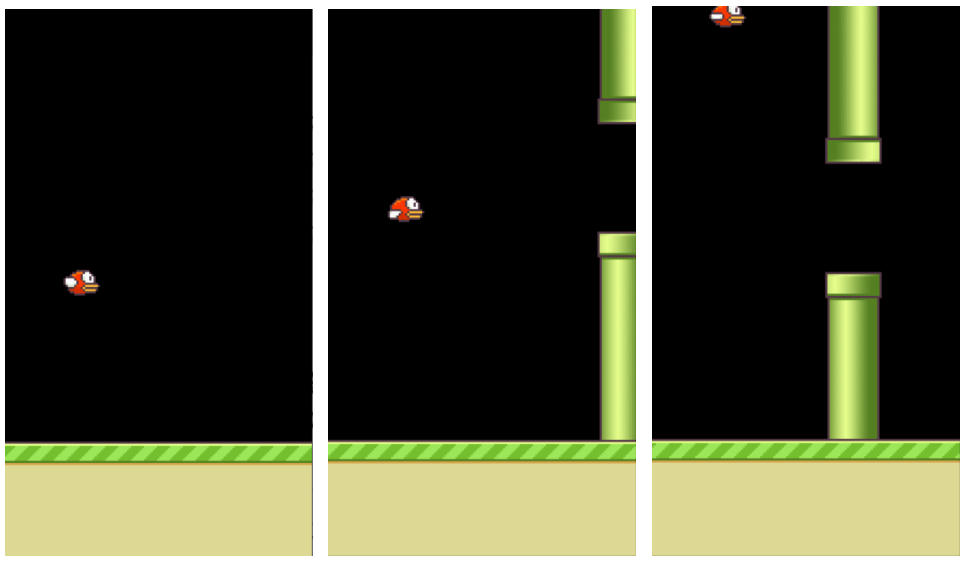

Figure 1: The improper actions of bird

Obviously, I have met with the problem of local optimal. Since the bird learned to fly high to not knocking the ground, it ignored the possibility of going through pipes and gain more rewards. I tried several methods to solve this problem and I haven't really succeeded up to now. However, I do some really useful works and I want to show the process to get some suggestions.

2.2 Problems and Methods

2.2.1 Problem 1: Strange performance of the bird

I tried to retrain the model from beginning. However, I met with a strange situation that although without any pretrained data loaded, my bird keeps flying upwards! It really shocked me and I did some experiments. I found that for each step, if the bird chooses to wings at the probability of 0.5, then it will get a huge accelerated velocity upwards and in general it will have the trend to fly to the top. What a strange! Since the deep network is initialled randomly and yields the scores of the two choices, wings or do nothing, randomly in beginning. It's no wonder that the bird keeps flying upwards.

The problem of this scenario is, when the bird knocking on the floor, it will 'die' and receive low rewards. So it can learn to leave the floor to obtain more rewards. However, if it reaches the top of sky, it will stay alive until knocked on the pipe, receiving a higher reward than falling to the ground. Then the bird may learn to be carefully, unwilling to go downwards, which will prevent it to find the gaps between the pipes so prevent learning! It's a really serious problem. (By the way,in my first train, the bird is prone to go down , it is really lucky since the initial random parameters of deep network is fine.)

My first method to solve this problem is to reduce the frequency of actions. By doing so, the bird chooses whether to wings, or do nothing, slower than before. So it will be decelerated by gravity for a longer time. This method does prevent the bird flying too high, however, it also has its drawbacks.

2.3 Problem 2: Latency

By applying the method above, the bird doesn't fly that high and the first problem seems to be solved. However, I don't think it's a perfect solution because the bird becomes 'dull'. It makes decisions slower and can't handle the situation timely. So I conceived a new method.

The core problem is that the bird will fly to the top without proper initial parameters in deep network, which will do harm to learning badly. So I tried to give fine-initialized parameters for the output of deep network from beginning. It is notable that, if the bird is prone to go downwards rather than upwards, it is OK because it can provide useful information for learning. How can I make the bird act like this ? Firstly, I add a softmax layer in the deep network to make its output values between 0 and 1(The output of the network is an array of length 2. The first element indicates the weight of doing nothing and the second indicates that of wings). Then, I add 0.3 to the first value of this array, to make the weight of 'do nothing' bigger such that it has a larger probability to be chosen. By doing so, I don't need to reduce the frequency of actions of bird and also receive good results. I catch a snapshot of my bird when writing this report, it successfully go through the first pair of pipes.

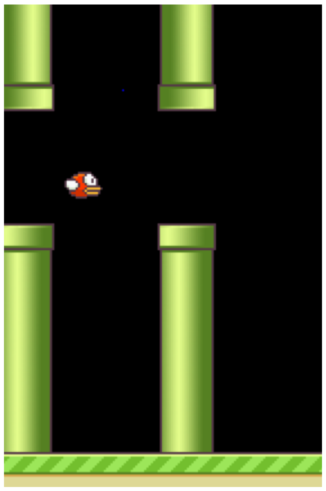

Figure 2: A success of the bird

3 Questions

- When my model is stuck in local optimal solution, should I retrain the model from beginning or is there any methods that is more efficient?
- Among so many kinds of RL algorithms, how should I choose my best algorithm, or if it works then I can think it is fine?
- How can I visualize the convergency of RL, since I don't know whether the model is really stuck because it has converged or it's because my training loops are not enough.

4 Plans

- This model is running on the CPU now. I will write a GPU version and run it on GPU next week.
- The algorithm of the game is an on-policy model, I'll modify it to an off-policy model for better performance if I have time.