Exercise 5

Gengcong Yan - 1009903
ELEC-E8125 - Reinforcement Learning

November 1, 2021

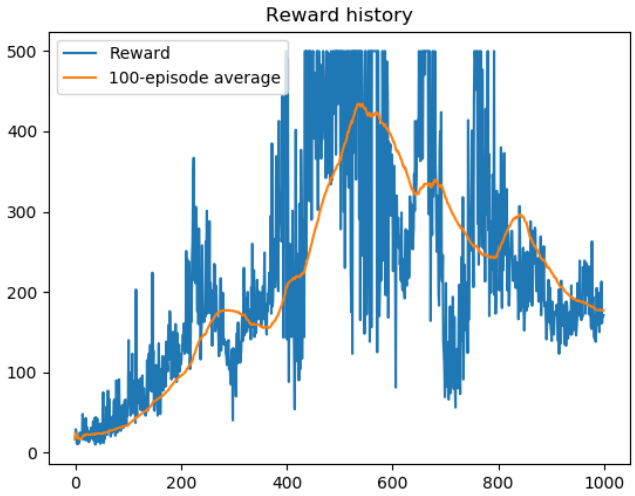

Figure 1: REINFORCE without baseline

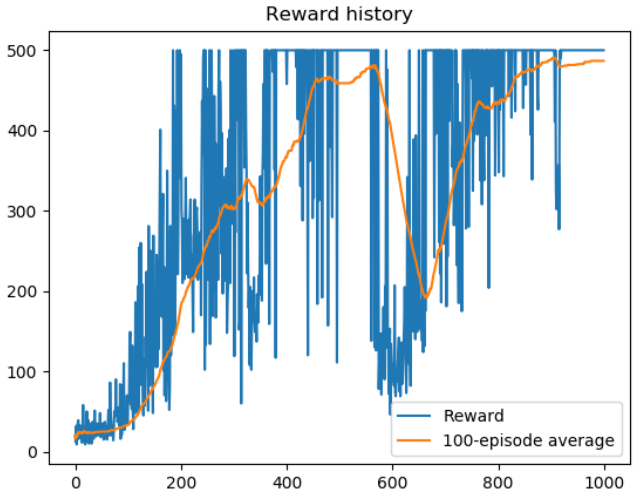

Figure 2: REINFORCE with baseline $b = 20$

1 Task 1

The plots in Fig.1, Fig.2, Fig.3 show reward history of the application of REINFORCE with 3 specific setting.

2 Question 1.1

The baseline can be any function as long as it does not depend on a_t. It can also be constant values. The baseline does not alter the expectation of the performance gradient. But in most

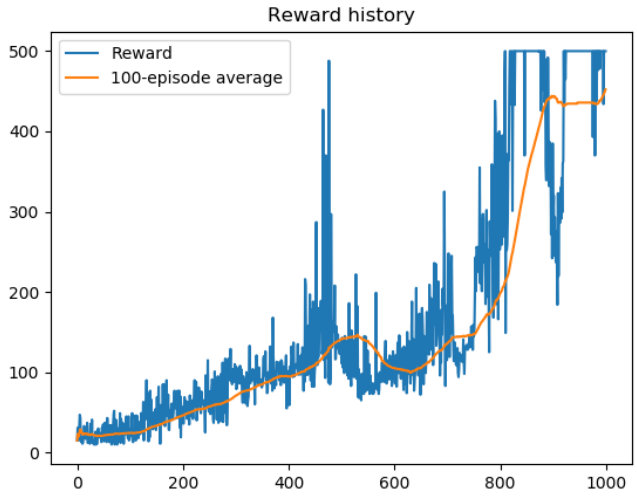

Figure 3: REINFORCE with discounted rewards

cases, we use the state value function $V(S_t)$ as the baseline. If the return is overestimated, the scaling factor is proportionally reduced by the value function, resulting in a lower variance. The value function is also parameterized and is jointly trained with the policy network.

3 Question 1.2

Baseline can be added to reduce variance of gradient estimate. It can speed up learning process of training, achieving final reward in less time.

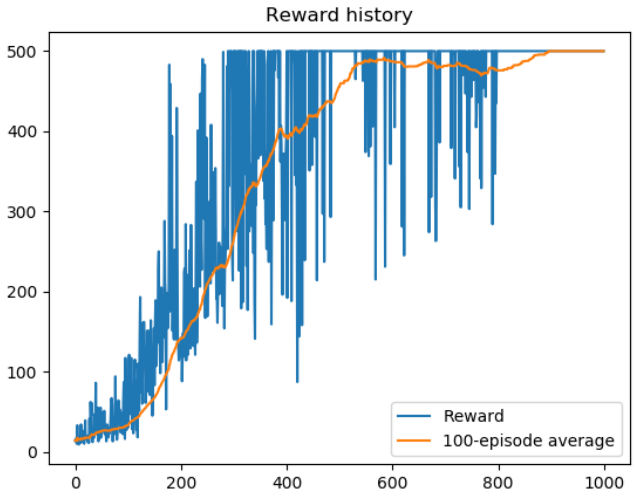

Figure 4: REINFORCE with decaying variance

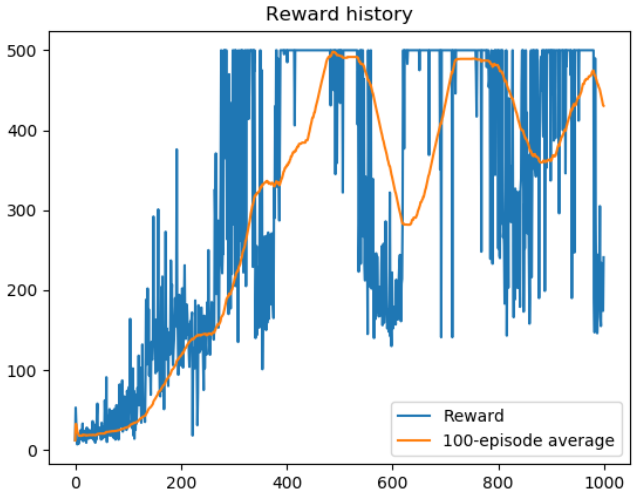

Figure 5: REINFORCE with learned variance

4 Task 2

The plots in Fig.4, Fig.5 show reward history of the application of REINFORCE with decaying variance and learned variance

5 Question 3

5.1 Question 3.1

- **Constant value**: It's the easiest way to implement with a certain value.But it's highly dependent on the initialization value of variance. How to choose a appropriate values is a question.
- **Decaying variance**: It converges faster to the bigger rewards The training curve looks more smooth.Because of the change in variance, the approach will explores more in the beginning and exploit more in the later stage.
- **Learning variance during training**:the variance will adapted over iterations according to the computation in training, thus the dependence on initial values is reduced. But the model still remains some instability during training process. Variance becomes a parameter in neural networks, causing more time in training compared to preset value from outside in previous cases

5.2 Question 3.2

if the initial value is too small, the model won't learn any useful behaviors. A proper variance helps us to learn useful information quickly, converging reward faster. A larger value at the very beginning is more conducive to explore and learning more new information, after which the value gradually becomes smaller during the training process and the whole model is exploiting more, gradually choosing the current optimal actions as the solution.

6 Question 4

6.1 Question 4.1

REINFORCE is an on-policy algorithm, but the experience replay should be utilized in an off-policy environment, which evaluates actions from a policy that is not the current one, using stored transitions. But with the help of importance sampling, we can perform off-policy policy gradient through samples from another policy. if the new action is relatively close to the old one, importance sampling allows us to calculate the new rewards based on the old calculation, estimating the value functions for a policy with samples collected previously from an older policy.

6.2 Question 4.2

if we update our policy based on stored transitions, like in experience replay, we are actually evaluating actions from a policy that is no longer the current one, since it evolved in time, thus making it no longer on-policy. Instead of sampling from data distribution, we calculate the result from sampling pools. But how to make the transitions in sampling have the best representation to restore the original data distribution as much as possible is the problem we need to solve.

7 Question 5

7.1 Question 5.1

Having an unbounded action space does indeed seem to imply that the action space has an infinite size. Perhaps the reward calculation will cause the model to not learn the action correctly, because just surviving is the reward that allows the actions going to many unknown meaningless action spaces that do not make practical use of the model. For example, in an airplane model, having the plane only fly forward can keep it alive without any real meaning. Perhaps the model needs to be guided using a reward function that is more realistic for the scenario.

7.2 Question 5.2

We need to modify our reward function to make it more adaptable to the various requirements in the specific model. Having only one survival function as a reward does not work in complex environments. Although there are no boundary restrictions in the action space, they can be considered in the reward function. When the action is out of bounds a huge negative reward is obtained. After training, the algorithm will not tend to leave the bound set in the reward function, because it will get a negative reward which is not be encouraged.

8 Question 6

If the action space is not too large, we can perform policy gradient in discrete action space. we calculate the probabilities of the actions with better performance in each state, then according to certain distribution, assign highest probabilities to the actions being selected. This could be a neural network with a softmax output layer. This is similar to how we simulate the process of computing probabilities in a continuous space. But such large action spaces are difficult to explore efficiently, and thus successfully training networks in this context is likely intractable. Additionally, naive discretization of action spaces needlessly throws away information about the structure of the action domain, which may be essential for solving many problems. So, The fundamental issue is inability to efficiently explore such a large discrete action space .

9 Feedback

1. 10h