Reinforcment Learning Exercise 5

Franceschini Riccardo 760697

November 2019

1 Task 1

The REINFORCE algorithm was implemented, the forward function with the normal distribution was done, also, the training and the get action functions were implemented, to do those function we had to sample and compute the action log probabilities from the distribution, for the get action function and compute the reward loss and update the network in the episode finished function.

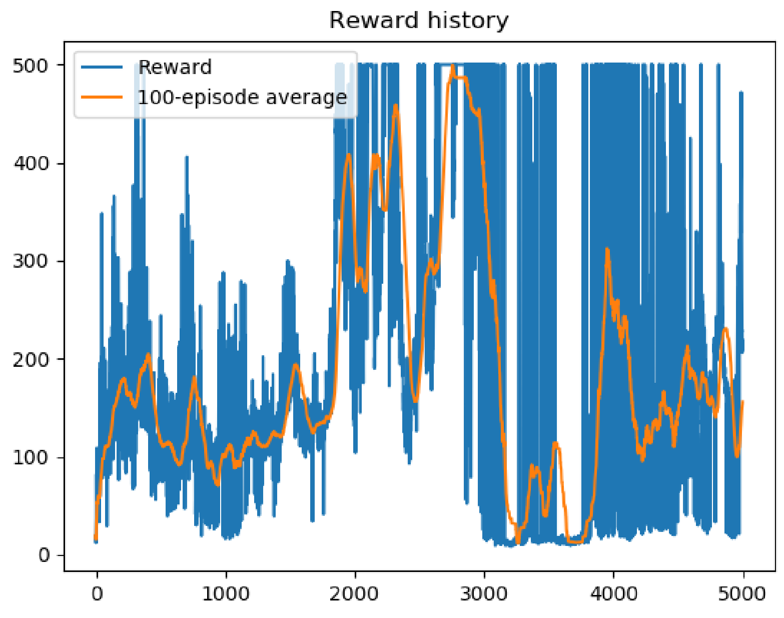


Figure 1: Basic REINFORCE

b)Also, the task b was implemented adding the baseline, as it possible to see from the fig.2 the baseline is able to reduce the variance

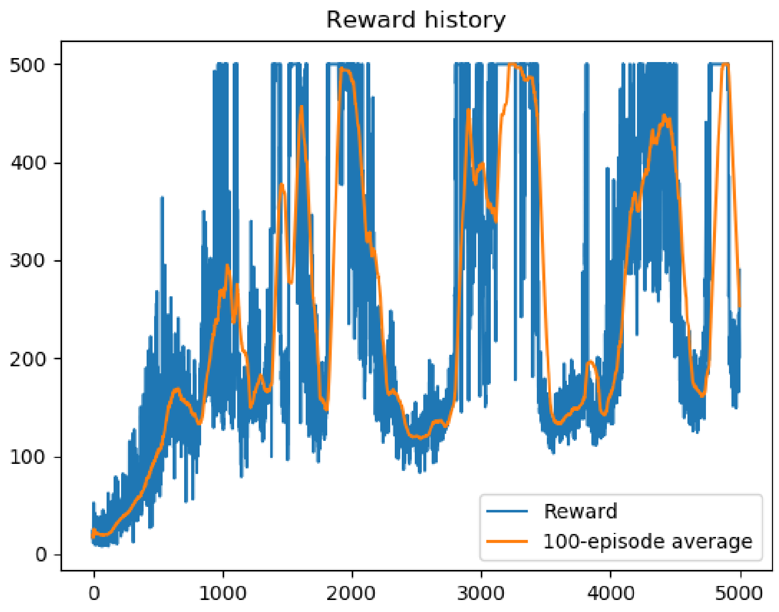


Figure 2: With baseline = 20

c) Normalizing the discounted reward to zero mean and unit variance we obtain.

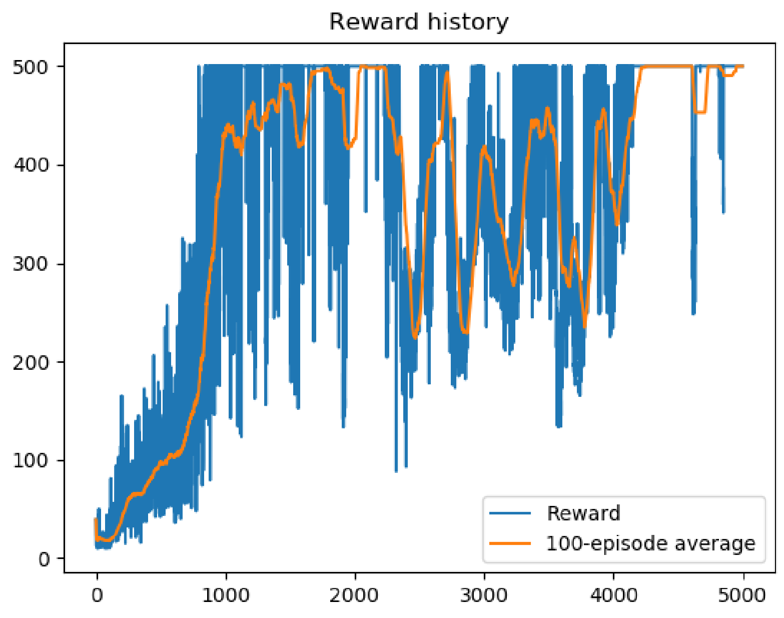


Figure 3: Normalized and unit variance discounted rewards

2 Q-1

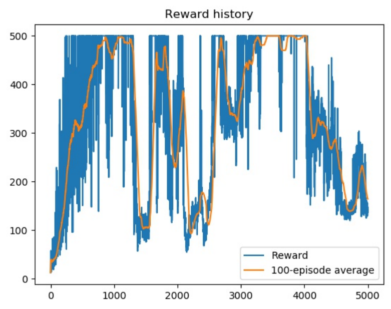
The baseline is able to affect the training because it reduce the variance and so the performances are more stable. The reason why the baseline work is due to the fact that in a complex task we need to collect a lot of different information in order to obtain stable behaviour because the policy may receive different rewards for similar behaviour. For this reason, using the baseline we are able to compute how much the actual action is better than the one that we usually do and this allows to reduce the variance.

3 Task 2

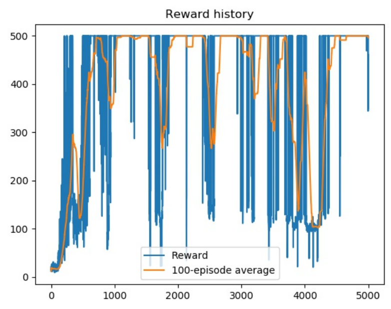
Modifying the Policy network we are able to implement the two different versions of σ. In the first one at the end of each episode we update the value using:

$$\sigma = \sigma_0 \cdot e^{-c \cdot k} \tag{1}$$

where k is the number of episodes $\sigma_0 = 10$ and $c = 5 \cdot 10^{-4}$. In the second one σ is encoded inside the network and is learned while is training.



(a) decay σ



(b) learned σ

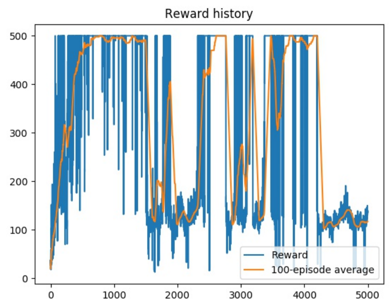
4 Q-2

4.1 Q-2.1

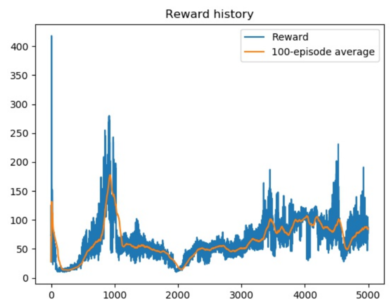
Using the learned σ we are able to obtain better performance because the parameter is inside the network and the agent is able to learn it during the training. On the other hand, the decay σ takes less to train compared to the learned one because the agent doesn't have to learn another parameter.

4.2 Q-2.2

Testing the algorithm with differents σ initialization is possible to see that the performances are affected by the σ initialization.



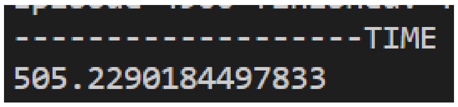
(a) σ initiliazed to 50



(b) σ initiliazed to 1

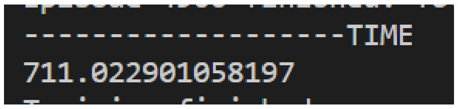
4.3 Q-2.3

The learned σ parameter takes longer to train because we are adding a parameter to the network and the network has to learn it



```
--------------------TIME
505.2290184497833
```

(a) decay σ training time



```
------------------TIME
711.022901058197
```

(b) learned σ training time

5 Q-3

Policy gradient algorithms are on policy methods which means that they have to they are using directly the policy to takes action on the environment and learn directly form their action and reward. In order to do perform experience replay we should use an off policy method which means that the agent is able to learn the policy just knowing states action ad reward without directly taking action in the environment. For this reason is not possible to use experience replay.

6 Q-4

Policy gradient algorithms are meant for continuous action space but it is possible to discretize the output of the network using values that are meaningful for our task for example only two actions for the cart pole instead of infinite values.

7 Task 3

The Actor critic model was implemented adding another layer to the policy network in order to predict the value function. After that, the update function was modified in order to compute properly the advantage and the different loss functions. The result of the training is possible to see below:

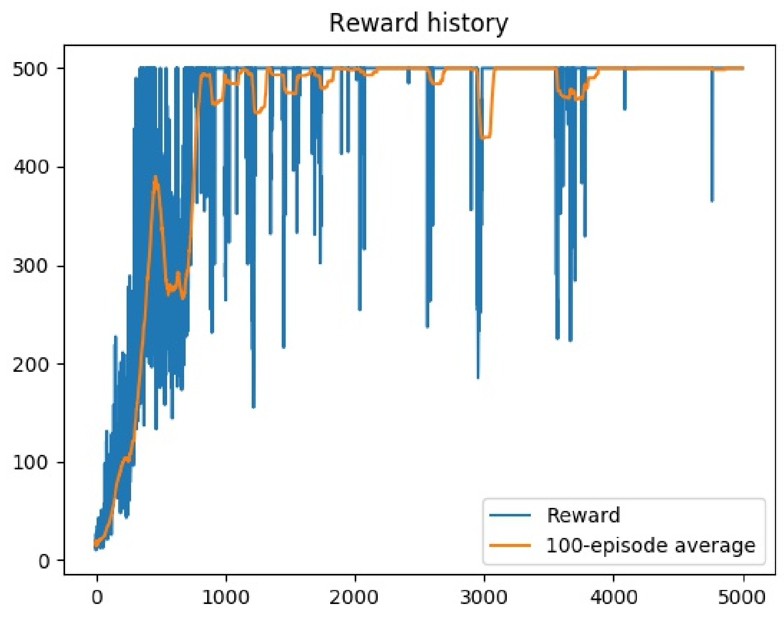


Figure 7: Actor Critic reward

As it possible to see the actor critic implementation outperform all the previous implementation.

8 Task 4

In order to implement the TD(0) the following advantage has to be computed:

$$A = r + \gamma V_{t+1} - V_t \tag{2}$$

so for doing that I stored also the value prediction inside the agent setting the value of the next state to 0 if the episode was finished.

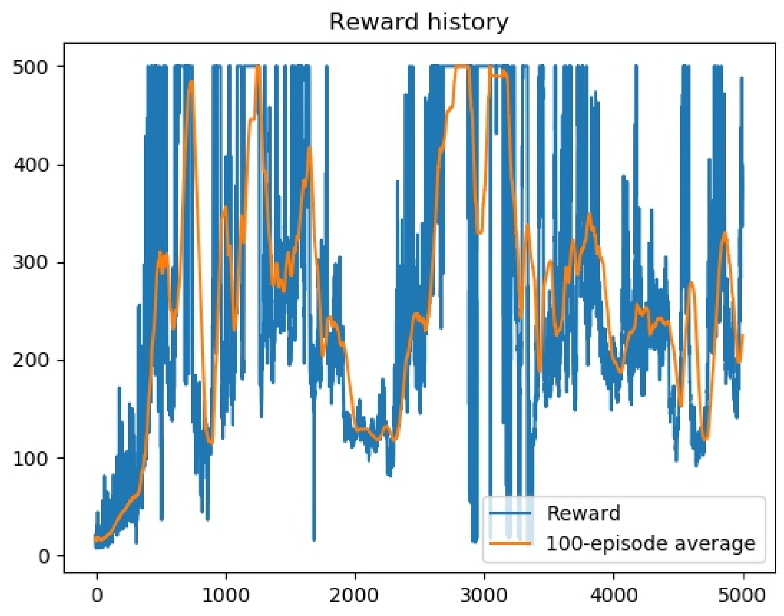


Figure 8: Actor Critic reward using TD(0) update every 10 timesteps

As it is possible to see, the agent is still able to learn how to balance the cart pole. However, updating the network so frequently introduces a lot of variance in the performance and for this reason, the reward goes up and down.

9 Q-5

Policy methods are generally faster to compute because they have to encode inside only the policy which is generally more compact that the value function and also policy gradient method are meant for continuous action space. On the other hand, action value methods are guaranteed to converge to the optimal policy while the policy gradient method cannot converge because of a local optimal. For this reason, Q-Learning can be used when the complexity of the system is not so huge and we are working in discrete action space. Policy gradient methods are useful for large and continuous action space. So, depending on the problem and what you are interested it is possible to choose one approach or the other.