RL-5

Abhilash Jain

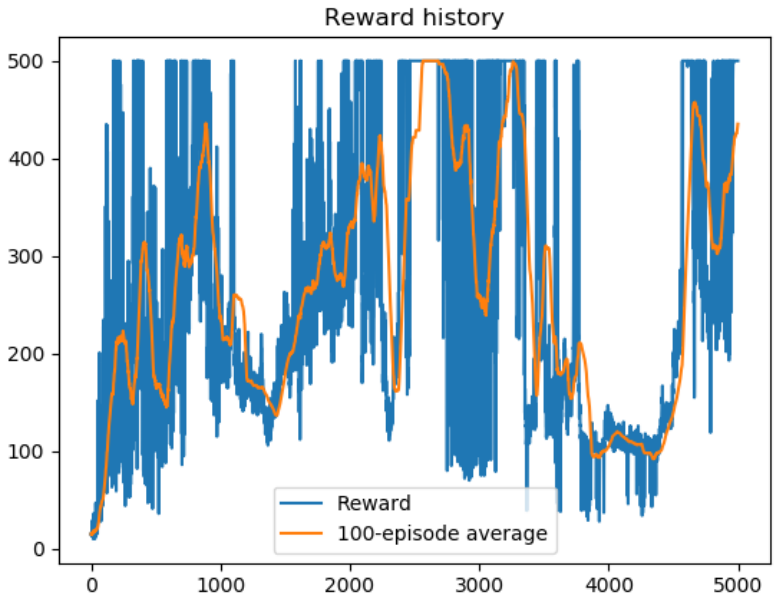
November 2019

1 Task 1

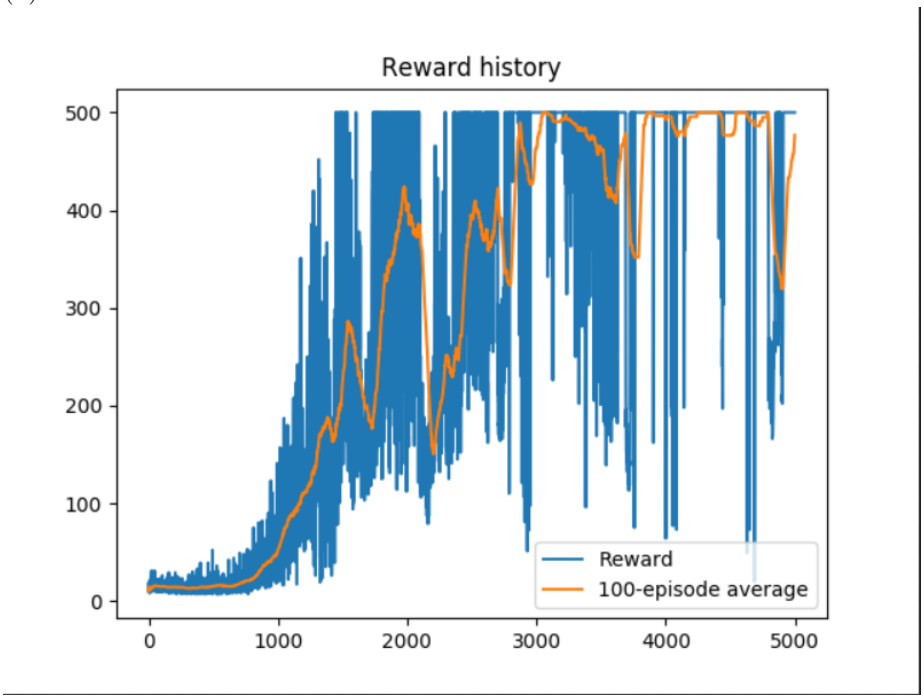
Implement policy gradient for the Cartpole environment with continuous action space. Use agent.py for implementing the reinforcement learning itself (for example, the agent and policy classes). Instantiate those classes in cartpole.py, similarly to how it was done in Exercise 1.

Use constant variance of 5 throughout the training.

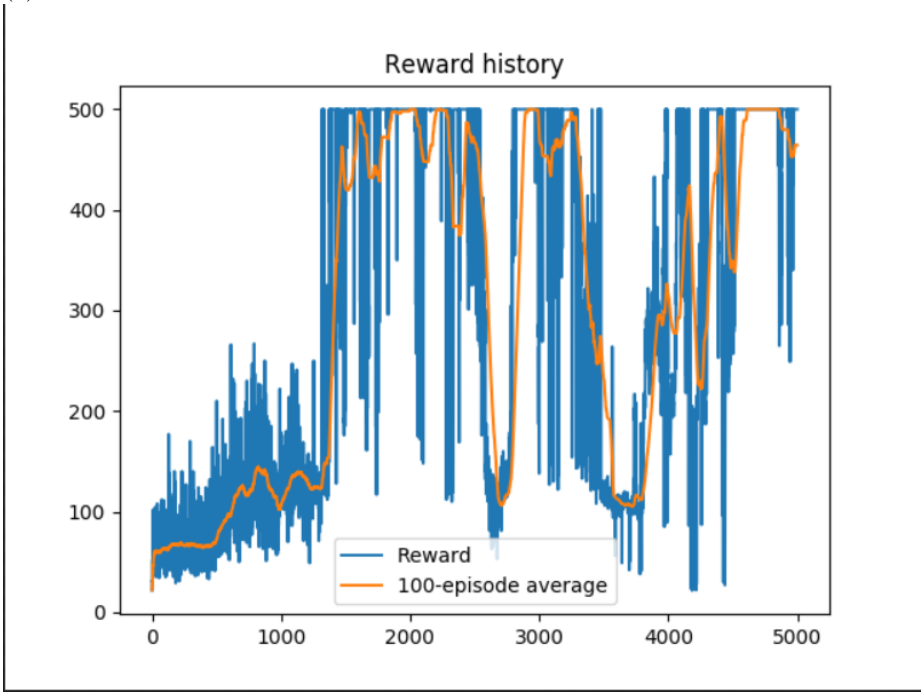
(a) basic REINFORCE without baseline



(b) REINFORCE with a constant baseline b 20



(c) REINFORCE with discounted rewards normalized

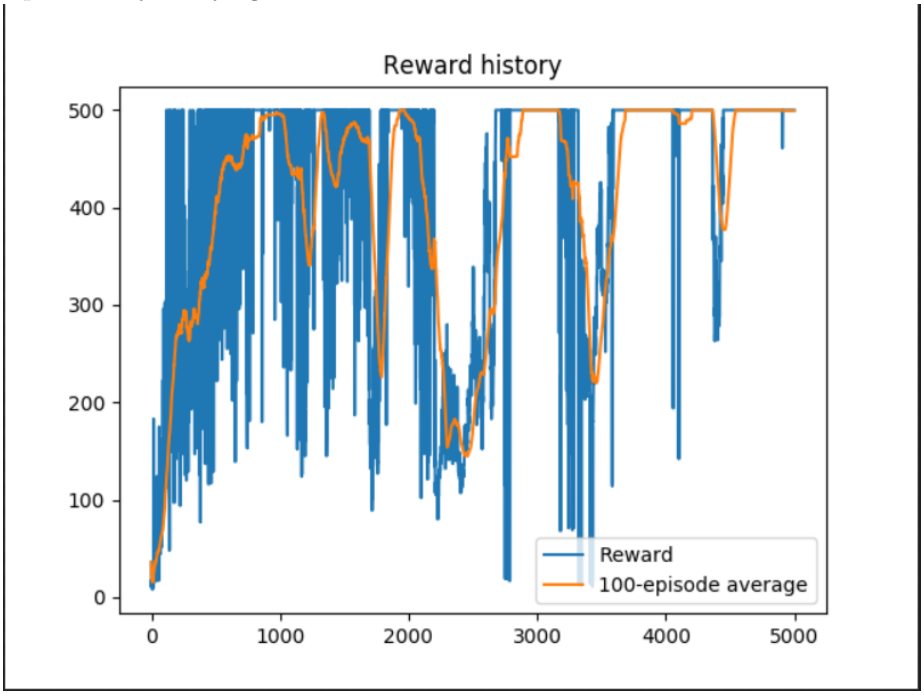


Question 1 How does the baseline affect the training? Why?

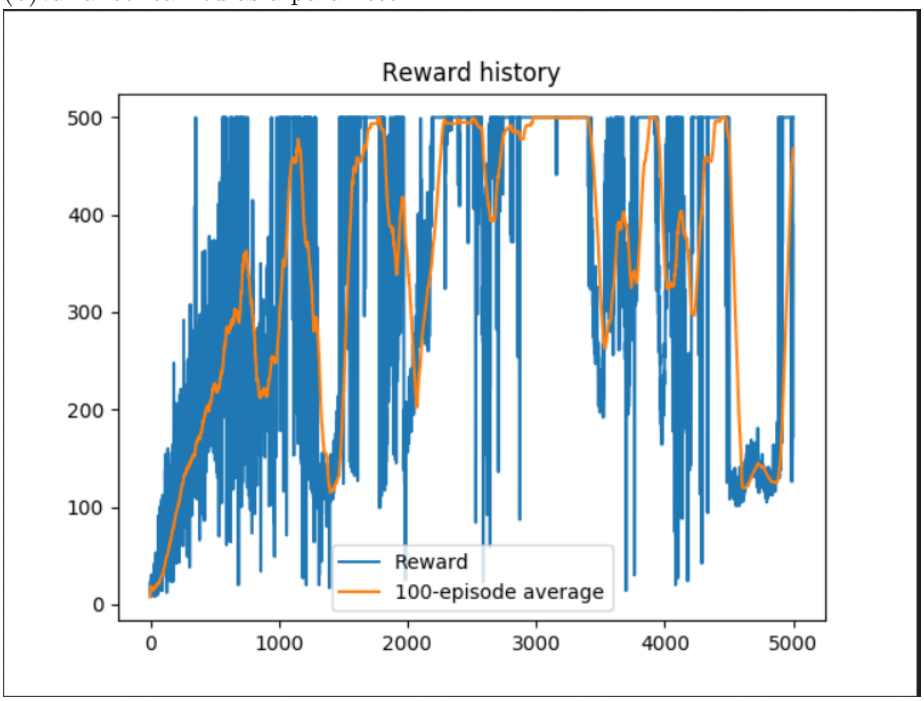
We select a baseline value and apply it in the algorithm in the step where we subtract this baseline value from the discounted rewards when we calculate the optimization term. We can see this baseline like a way of reducing variance and as an approximation of the state value function. There-fore in the training, 20 was a very good value for the training to converge as we can see in the graph. And a higher value like 30,in the training the model doesn't really converge as good as with the previous value. It was even worse for value 10. There-fore if we a good guess of the baseline, as above, we have a good approximation of the state value function which plays a role in the model towards convergence.

2 Task 2

(a)exponentially decaying variance



(b)variance learned as a parameter



2.1 Question

What are the strong and weak sides of each of those approaches?

The strong side of constant variance can be seen as : it runs faster, and if we have a good value the model may converge to constant 500 rewards. Weak side, would be convergence is not guaranteed and most of the times it doesn't converge. and the hyper parameter needs to be tuned for the model to perform better

The strong side of decay algorithms can be seen as : They are more reliable in terms of convergence and most of the times they do converge. The weak side would be we need to be sure of the function and the rate with which the sigma is decaying, which for more advanced applications is more of a tricky problem.

The strong side of learning the parameter, is ofcourse we don't really need to worry about the decaying of the variance or any change to be done to the variance while it trains and the network learns it on its own. The weak side is, atleast this exercise it doesn't converge on some runs and is poor sometimes, sensitive to initialization.

2.2 Question

In case of learned variance, what's the impact of initialization on the training performance?

The variance is a parameter which would signify the exploration space for the model, there-fore initializing a low variance would mean less exploration-higher rewards in the start and not that high gain or convergence in the end of the training and high variance would be alot more explorations and sometimes would not converge. There-fore a right balance needs to be achieved by choosing a good variance for initialization.

2.3 Question

Which takes longer to train?

The decaying variance and learning variance take longer to train because in most of the episodes they reach 500 timesteps as you can see in the graph. Learning Variance a bit longer as it has the variance parameter to learn as well.

But clearly the constant variance approaches take longer to CONVERGE when compared to Task 2

2.4 Question 3

Could policy gradient methods be used with experience replay? Why/why not? Explain your answer.

Policy gradient methods are on-policy, training samples are collected according to the target policy , but yes, policy gradient methods can be used with experience replay(past episodes.) But then the policy gradient would then become a off-policy gradient methodology. when applying policy gradient in the off-policy setting, we can simple adjust it with a weighted sum and the weight is the ratio of the target policy to the behavior policy The off-policy gradient methods have few advantages like:

1)The off-policy approach does not require full trajectories and can reuse any past episodes ("experience replay") for much better sample efficiency.

2)The sample collection follows a behavior policy different from the target policy, bringing better exploration.

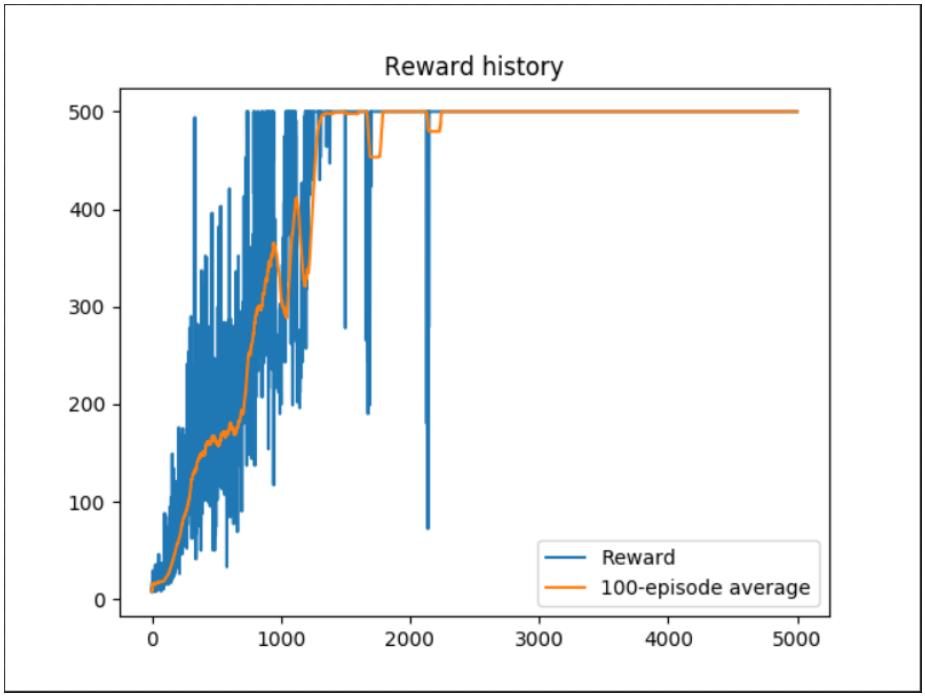
REFERENCE: https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html the off-policy gradient methodology section.

2.5 Question 4

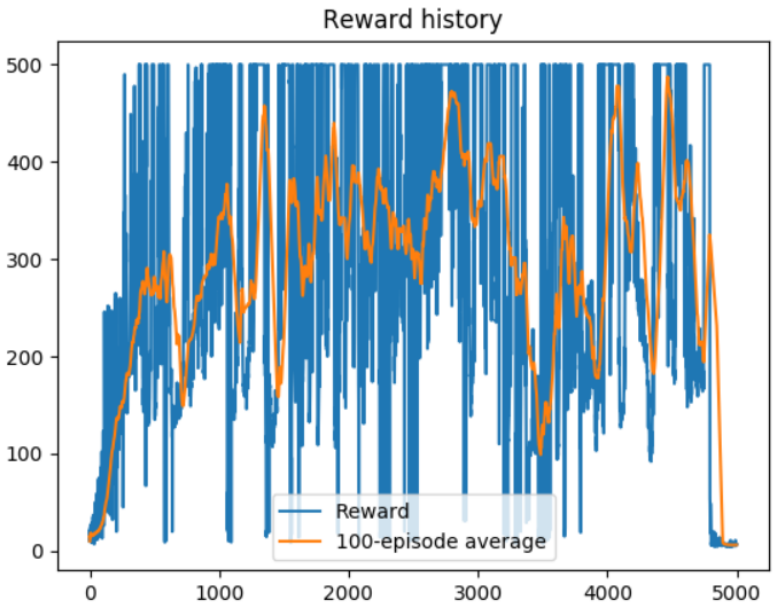
Can policy gradient methods be used with discrete action spaces? Why/why not? Which steps of the algorithm would be problematic to perform, if any?

Even though policy gradient methods are not meant for discrete action spaces , they can be used for discrete action spaces, how well they perform when compared to methods like q-learning would be an experiment. And in implementation, for example of a discrete action, π could be bernoulli with p parameterized by the output of the network.

3 Task


Reward history
500
400
300
200
100
0
0
1000
2000
3000
4000
5000
Reward
100-episode average

4 Task



5 Question

What are the advantages of policy gradient methods compared to action-value methods such as Q-learning? When would you use Q-learning? When would you use a policy gradient method?

Policy gradient have the following advantages over value-based methods such as they can solve large and continuous action spaces without the need of discretisation (as in Q-learning). Moreover, value-based method cannot solve an environment where the optimal policy is stochastic.

However Q-learning is beneficial in tabular cases and when you want deterministic policy. Mainly it also depends on the state representation of the problem as in some cases we would want to learn the value function and in others the policy.

Policy gradient methods can be used when the action space is continuous Also we can employ an approach which uses the strengths of both methods together, as in Actor-Critic method.