Reinforcement Learning - Week 5
Policy Gradient and Actor Critic Approaches

Antonio Chiappetta

November 2019

1 Policy gradient

1.1 Task 1

The following plots show the reward history of the application of the REINFORCE algorithm in the three specified cases.

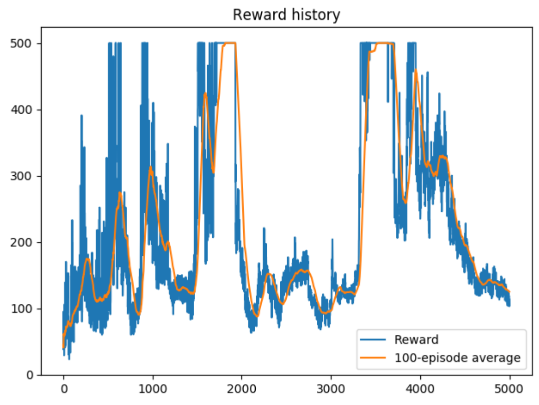

Figure 1: REINFORCE without baseline

1.2 Question 1

The introduction of the baseline has the effect of speeding up the learning process and getting close to the long-term total reward per episode in less time. This happens because the use of the baseline helps reducing the variance.

1.3 Task 2

The following plots show the reward history of the application of the REINFORCE algorithm with no baseline and normalized rewards in the two cases

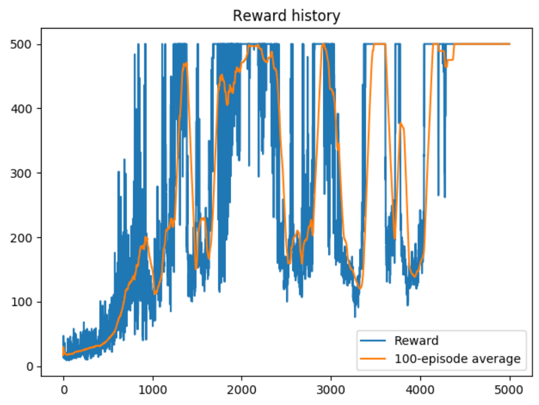

Figure 2: REINFORCE with constant baseline

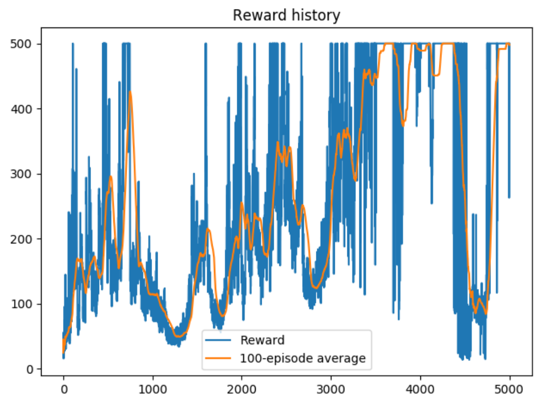

Figure 3: REINFORCE with normalized discounted rewards

of exponentially decaying variance and variance learned as a parameter of the network.

1.4 Question 2

1.4.1 Pros and cons of different approaches to variance

Compared to the approach with constant variance, the one where variance is decaying exponentially is converging faster to high values, but maintains the same instability across the training process. The approach with constant variance is highly dependent on the initialization value of the variance, while the other has a behavior similar to GLIE and explores more in the beginning to then exploit more in the following episodes.

The approach with learned variance follows instead a smoother curve, that

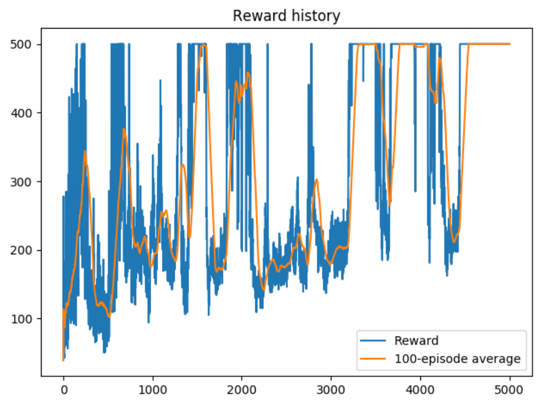

Figure 4: REINFORCE with decaying variance

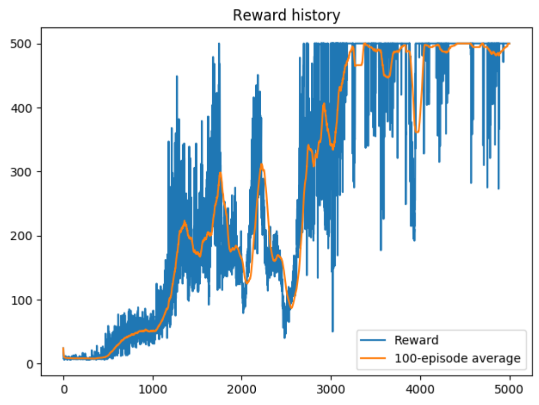

Figure 5: REINFORCE with variance learned as a parameter of the network

grows slower but then stabilized to high values and rarely diverges from those. This seems to be the best approach. This approach performs better because the variance can dynamically be adapted over the iterations, so the dependence on the initial value is reduced. Computation time is higher because we are using a neural network to learn a value that was set from the outside in the previous cases.

1.4.2 Impact of initialization on training performance

A too low value of the variance doesn't help our algorithm to learn. The higher the value, the faster the convergence to a high average return. A large variance in the beginning makes our agent start with more exploration and then adapt, which is good to learn a good policy in a few episodes and then exploit it properly.

1.4.3 Training time

The version with variance learned as a parameter of the network takes much more time to train. The additional complexity is due to the fact that the variance needs to be learnt through the network and is not set before.

1.5 Question 3

REINFORCE is an on-policy algorithm. Experience replay evaluates actions from a policy that is not the current one anymore, using stored transitions, and thus making it off-policy. The use of importance sampling is the trick that allows policy gradient methods to use off-policy samples: with importance sampling it's possible to estimate rewards of a policy through samples from another policy.

1.6 Question 4

If the action space is discrete and not too large, policy gradient methods can be used by using a parameterization of the policy in the form of numerical preferences for each state-action pair. The actions with the highest preferences in each state are given the highest probabilities of being selected, for example, according to an exponential soft-max distribution.

2 Actor critic

2.1 Task 3

The following plot shows the execution of the actor critic algorithm with updates at the end of each episode:

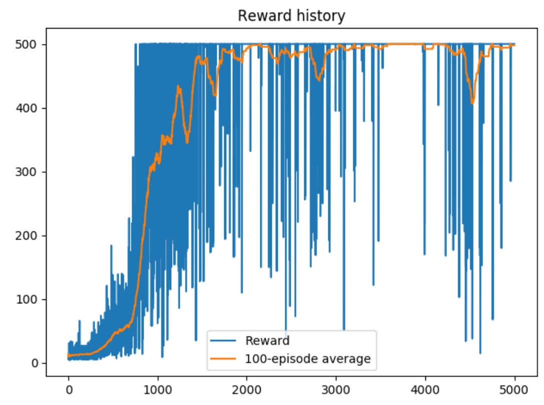

Figure 6: Actor critic with updates at the end of each episode

2.2 Task 4

I wasn't able to perform this task correctly, so no code nor plot is included in the submission.

2.3 Question 5

The advantages of using one approach or the other are a consequence of their different goals: while policy gradient methods aim at learning a mapping from state to action that can also be stochastic and work in continuous action spaces, action-value methods aim at learning a single deterministic action from a discrete set of actions by finding the maximum value.

As a result, policy gradient methods can solve problems that action-value methods cannot:

- **Large and continuous action space**. However, with value-based methods, this can still be approximated with discretization.
- **Stochastic policies**. An action-value method cannot solve an environment where the optimal policy is stochastic, because there is no way to control probabilities of actions and a deterministic agent is assumed to be optimal.

However, action-value methods like Q-learning have some advantages too:

- **Simplicity**. You can implement Q functions as simple discrete table, this is not possible with policy gradient methods.
- **Speed** TD learning methods that bootstrap are often much faster to learn a policy than methods which must purely sample from the environment in order to evaluate progress.

Sometimes other reasons could influence your choice, like the state representation of the problem lending itself more easily to either a value function or a policy function.

Depending on the variables listed below, the choice can be made of using one approach or the other, or to combine their strengths using actor-critic methods.

References