Problem 1

Anshuman Senapati

November 12, 2019

1 Results: Solving CartPole Environment with DQN

After tuning the hyperparameters, the experiment is carried out several times and the model is evaluated on an average basis over these experiments.
The environment was solved in (119 ± 9) episodes on an average over 50 independent runs of the experiment, using an experience replay as well as a target network.

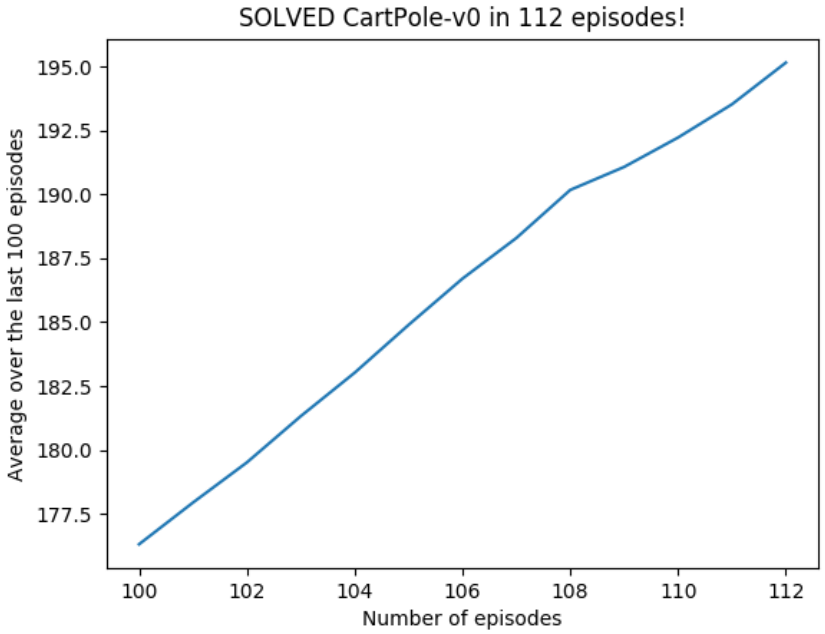

Figure 1: A plot from one such experiment.

2 Hyper-parameters

2.1 Hidden Layers

The neural network used for approximating the Q-function consisted of two hidden layers with equal number of neurons each. And an identical network was used as the target. This number was varied as 3, 12, 24, 32, 48, and so on. An average over several experiments showed that the value of **24** works the best.

As a rule of thumb, sizes of the hidden layers should be between the input and output layer sizes, but those shapes being very small, it forms a very simple model and hence results in ineffective learning.

Increasing the number of neurons increases performance till a certain point after which the model becomes too complex again resulting in poor learning.

2.2 Minibatch size

Randomly drawn samples from the replay memory were fed to the network in batches, typically in powers of 2 such as 16, 32, 64, 128,... The ideal minibatch size turns out to be **64**.

Other sizes typically result in the problem remaining unsolved at the end of all the learning trials.

A lower batch size might fail as relatively lesser number of samples make gradient computing very inaccurate, the gradient bounces around in random directions towards a minimum which probably takes a lot of time to converge.

On the other hand, using a larger batch size is known to provide unsatisfactory convergence results as they are known to converge into sharp minima where it might get stuck. So, an intermediate batch size, such as above, makes sense.

2.3 Epsilon

Epsilon is initialized to 1 in the beginning and is allowed to decay at a certain rate until it reaches a set minimum epsilon. So that exploration is encouraged at initial stages and exploitation subsequently.

2.3.1 Decay rate

The decay rate of epsilon are tried at intervals of $\sim$0.005 such as 0.999, 0.995, 0.990, and so on. The optimal decay rate appears to be **0.995**.

The lower decay rates (those closer to 0) have poor convergence results. Either the problem remains unsolved at the end of the experiment, or it takes relatively higher number of episodes to get solved and the plot obtained is highly zigzagged. A probable cause could be, a decay rate closer to 0 leads to a higher decay and consequently lesser exploration before reaching the minimum epsilon. Hence, the agent is unable to discover better policies.

While decay rates closer to 1 give smoother plots and converge within fewer

episodes than the above case, on an average, they take a relatively more number of episodes than when the decay rate is 0.995.

2.3.2 Minimum Epsilon

The predetermined minimum epsilon value is tried to be 0.001, 0.005, 0.01, and so on. The optimal value comes out to be **0.01**.

Although as far as the initial learning process is concerned (and as long as the minimum epsilon value is small, that is, around 0.01), it doesn't bring about much difference in the results (such as the number of episodes for solving the environment or the nature of the plot obtained). After the action value function is learned, a smaller epsilon is needed to act according to the learned policy, but large enough to still continue some exploration. On an average a value of 0.01 works best.

2.4 Some Other Hyperparameters

2.4.1 Memory

The size of the replay memory is also a significant hyperparameter. Experiments were done taking it to be a size of 1000, 10000, 100000, and so on. A size of **1000** works best.

Although a major difference is not seen among different sizes, higher sizes of memory retain examples from a earlier stage in the learning process which might not be as valuable as the recent transitions, slowing down the learning process a little bit.

2.4.2 Reset Steps

After every fixed number of steps, the target network is reset to the learned action-value network. Number of steps is tried as 50, 100, 150, 200, and so on, among which, a step of **100** works best.

A smaller number of steps leads to resetting the networks too frequently as would be without a target network, it makes the learning very unstable.

A higher number of steps leads to resetting the networks too less frequently, holding back the learning as it will take a lot of steps to update to a better target action-value function.

3 Effect of Removing the Replay Memory and/or the Target Network

3.1 Without the Target Network

First, the target network is removed while still using the replay memory. Now the sampled batches from the memory are fed into a single network used for both learning the action-value function and also as a target.

The environment doesn't get solved within the set number of trials in this situation. Moreover, the plots obtained show that the learning process is highly unstable probably because the targets are non-stationary and keep moving after each update.

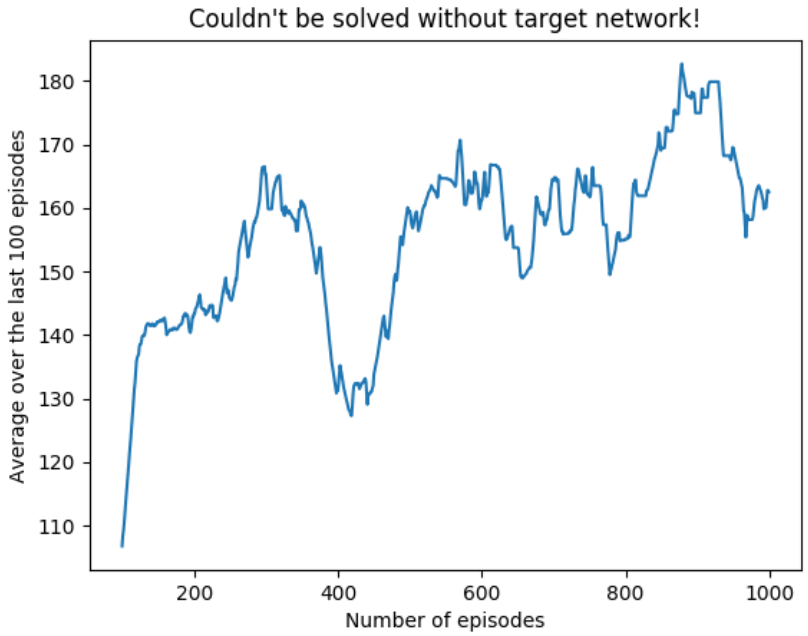

Figure 2: A plot obtained without using a target network

3.2 Without both the Experience Replay as well as the Target Network

The plots obtained from the experiments show that the performance significantly worsens on removal of the experience replay. Apart from the environment remaining unsolved, the average rewards do not improve at all even till the very end of the learning trials.

This problem might be arising as the samples are no longer randomly chosen, hence not identical and independently distributed anymore as required in a supervised learning process, rather are sequential and are bound to have some correlation among them. This leads to a much slower convergence of the network weights.

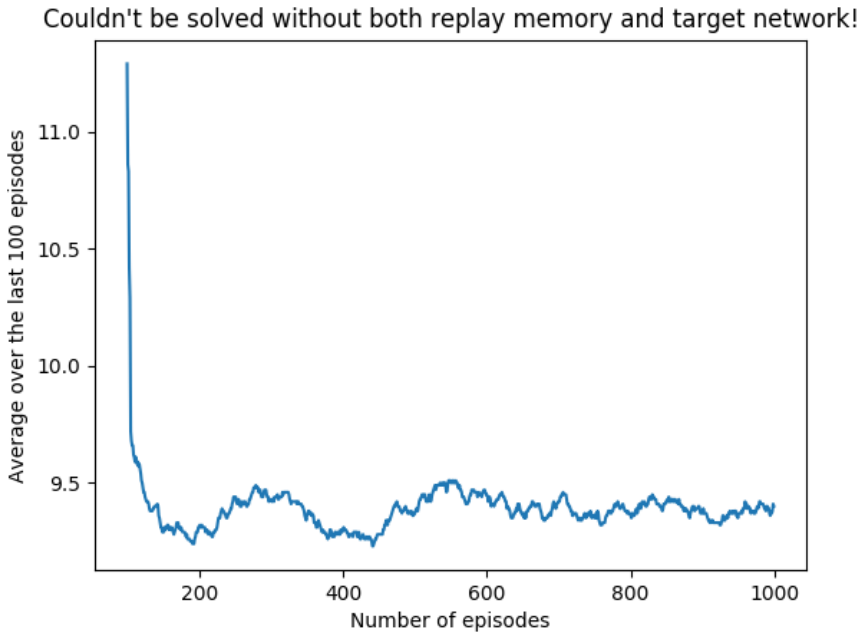

Figure 3: A plot obtained without the replay memory as well as the target network