

Report Prioritized Experience Replay

Siebbe Dekker, Fredrik Sjögren, Dimitar Dimitrov, Leon Becker

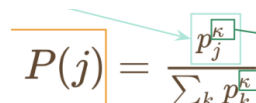
Explanation of method:

Prioritized Experience Replay is a technique based on the sampling of experiences. We take an experience buffer with a maximum capacity and fill it with a random subset of all already existing experiences. Every subset which goes to the experience buffer will be used to refresh the weights of the used network.

One of the biggest challenges is to determine which experience should go to the experience buffer. Therefore we assign probabilities to every experience, in this case we are using the td-error to assign the probability. So the probability of an experience is the td-error.

$$p_j = |\text{TD_error}_j|$$

To now determine the probability that this experience get chosen for the experience buffer we have to divide this p_j by the sum of all other probabilities.


$$P(j) = \frac{p_j^k}{\sum_k p_k^k}$$

After now deciding which experience makes it into our experience buffer we take those selected and use them to update our network.

Implementation in Code:

The implementation needs changes in two different passages of the DQN Code.

First we have to change the “Replay Buffer” to an “Prioritized Replay Buffer”. We configured the old buffer with every skill needed, most important with the distribution of probability. The most important method we added was the sample-method. This method determines the chosen experiences to go to in to a sample for updating the weights. This is based on the td-error of each experience.

An other important method we implemented in the buffer is the method to priorities, because they are continuously changing.

The second big part we changed is the train method. Here we collect a batch from the buffer and get all needed information from it. After transforming the states, rewards, actions, next states and “done” into numpy arrays we can get each tensor by using pytorch. With the help of that we are able to calculate the needed q values and the loss. At the end we are calculating the new td-error and refresh it by using the buffer.

Results

The results of the DQN with prioritized experience replay can be seen in table 1 below, where they are compared to previously obtained results for the base-, Double-Q-, n-step-, and Dueling DQNs. For each of the methods, the agent was trained 10 times. For each of the 10 training rounds, the trained agent was tested by running 10 episodes. The results of the tests were evaluated based on three metrics:

1. No. Times converged: Out of the 10 training rounds, this is the amount of rounds where the model converged. The model was considered to have converged once the mean return of the 100 latest rewards was > 195 .
2. Convergence time: If, during a training round, the model converged, this is the number of episodes that passed before the model converged. This is counted from the start of the 100 episodes whose mean was > 195 . For example, if episodes 43-143 are the first 100 consecutive episodes with a mean of > 195 , the convergence time is 43 episodes.
3. Return per episode: For each training round, the return (cumulative reward) of each episode was averaged. The cumulative reward of an episode is equal to the length (number of steps) of that episode. The episode ends if the pole falls over, the cart moves too far to one side, or if the maximum episode length of 200 steps is reached. This means that the maximum return of an episode is 200, and anything lower means that the agent failed the task of balancing the pole on top of the cart.

Table 1. The table shows the results from training the agent 10 times. The table includes the total amount of times the model converged, as well as the means and standard deviations of the convergence time and returns per episode, computed across the 10 training rounds.

Method	No. Times converged	Convergence time – Mean (No. episodes)	Convergence time – Standard deviation (No. Episodes)	Return per episode - Mean	Return per episode – Standard deviation

Base DQN	1	43.0	N/A	193.1	13.0
Double-Q DQN	2	37.5	10.6	190.3	20.2
n-step DQN	2	20.0	5.7	199.6	1.4
Dueling DQN	0	N/A	N/A	162.7	66.3
Base DQN with prioritized experience replay	2	25.0	7.1	185.6	17.2

Conclusion

Prioritized experience replay shows to be a effective method for fitting problems. in comparison to the other methods its quick to converge and leads to fast and good results. The only problem is the implementation which can be tricky and has a lot of challenges and threads.