A Reproducibility Study of Double Q-learning

authors
Ali FRADI, Joshua AMELIA and Ghofrane KHEDHIRI

Abstract

As Q-learning uses a maximizing function the estimates for the action value have a positive bias and over estimate the action value, Q(s, a). This can lead to poor performance with stochastic MDPs. The goal of Double Q-learning is to eliminate this overestimation.

1 Background and Motivation

Double Q-learning is focused on the update function of the Q-learning algorithm Hado van Hasselt:

$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t)[r_t + \gamma \underset{a}{max} Q_t(s_{t+1}, a) - Q_t(s_t, a_t)]$

To overcome the overestimation problem, Double Q-learning learns two sets of action values $Q_a(s, a)$ and $Q_b(s, a)$. Double q-learning estimates the action value of the next step using the opposite $Q(s, a)$. The paper proves that using a double estimator is unbiased and will not have the overestimation of Q-learning. Double Q-learning can occasionally underestimate the action value as the optimal action of $Q_a(s, a)$ is not necessarily the optimal action of $Q_b(s, a)$. On experiments, Q learning tends to be more optimistic than reality since maximizing on actions introduces some correlation with the product of cumulative density functions. Double Q-learning will converge to the optimum at the limit. The algorithm that we implemented for Double Q-learning is:

Hado van Hasselt

Algorithm 1 Double Q-learning algorithm

Initialize Q^A,Q^B,s
repeat
  Choose a, based on $Q^A(s, \cdot)$ and $Q^B(s, \cdot)$, observe r, s ´
  Choose (e.g. random) either UPDATE(A) or UPDATE(B)
  if UPDATE(A) then
    Define $a^* = argmax_a Q^A(s\prime, a)$
    $Q^A(s, a) \leftarrow Q^A(s, a) + \alpha(s, a)(r + \gamma Q^B(s\prime, a^*) - Q^A(s, a))$
  else if UPDATE(B) then
    Define $b^* = argmax_a Q^B(s\prime, a)$
    $Q^B(s, a) \leftarrow Q^B(s, a) + \alpha(s, a)(r + \gamma Q^A(s\prime, b^*) - Q^B(s, a))$
  end if
  s ← s ´
until end

2 Related Works

The idea behind Double Q-learning shine when we consider limiting the overestimation caused by maximum reward selection independently of what the next action should be. In fact, the method introduces a second approximator to permit a mutual interaction of both approximators where each of which cannot select the action and contribute to the update at the same time step. Instead, having learnt from two different disjoint samples of data, an approximator can select an action that it considers optimal and let the second

approximator evaluate this choice. This way can compensate the overestimation a single maximum estimator can induce. It also gives a certain flexibility to explore actions to get closer to effective observation value.

Though this intuition achieves a partial satisfaction, proved mathematically, it still seduces us to get performance better. Building on this logic, we suggested that a third can minimize a regret that a Double Q-learning can introduce whenever we take an action with minimum reward out of set of elite actions while it is possible that we pick an action with higher value. So the idea is as simple as introducing a third approximator in continuing interaction with both other approximators. At each time step, one of three is decided to pick an optimal action and give the permission to the others to evaluate the decision. Having this idea, we can go for different versions where we consider updating Q(s,a) using average, minimum of $\underset{a}{max}$(Q(s,a)) or whatever other criteria to define. The article Yi (2020) discusses the possibility of making use of maxMin (Q(s,a)) criteria to boost the Q-learning performance.

The logic running behind the scenes is as follows:

Algorithm 2 Maxmin Q-learning

Input: step-size α, exploration parameter > 0, number of action-value functions N
Initialize N action-value functions $Q^1, ..., Q^N$ randomly
Initialize empty replay buffer D
Observe initial state s
while Agent is interacting with the Environment do
  $Q^{min}(s,a) \leftarrow min_{k \in 1,...,N} Q^k(s,a), \forall a \in A$
  Choose action a by -greedy based on Q^{min}
  Take action a, observe r, $s^{'}$
  Store transition $(s, a, r, s^{'})$ in D
  Select a subset S from {1, . . . , N} (e.g., randomly select one i to update)
  for $i \in S$ do
    Sample random mini-batch of transitions $(s_D, a_D, r_D, s^{'}_D)$ from D
    Get update target: $Y^{MQ} \leftarrow r_D + \gamma max_{a' \in A} Q^{min}(s^{'}_D, a^{'})$
    Update action-value $Q^i : Q^i(s_D, a_D) \leftarrow Q^i(s_D, a_D) + \alpha[Y^{MQ} - Q_i(s_D, a_D)]$
  end
  s ← $s^{'}$
end

3 Discussion

3.1 Reproducing Results

Our first goal was to compare the estimations of the $Q(s,a)$ for the Q-learning and Double Q-learning. To reproduce the results we used the epsilon and alpha equations from the paper. $\epsilon(s) = \frac{1}{n(s)}$

$\alpha_t(s,a) = \frac{1}{n_t^{0.8}(s,a)}$

Where $n(s)$ is the number of times state s has been visited and $n_t(s,a)$ is the number of times action a was taking in state s. For Double Q-learning n_t is tracked separately when Q_a or Q_b is updated. The polynomial learning rate should have superior performanceEven-Dar & Mansour (2003).

3.1.1 Grid World

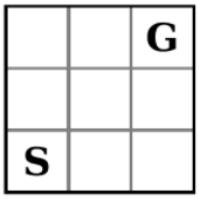


Figure 1: Grid world environment

The Grid World used in Hado van Hasselt was a 3x3 grid with the start in the bottom left corner and the goal in the top right corner. After each step a reward is given that is randomly chosen between -12 or +10. Any action while in the goal state ends the episode and returns a reward of +5. The average reward for a non terminal action is -1 and the optimal episode is 5 steps with an average reward of +0.2 per step.

Figure 3 shows the average reward per step and the maximum action value in the starting state for both the papers results and ours. The paper trains for 10,000 episodes and averages the results over 10,000 experiments. Using colab to recreate these tests would take far too long so our tests only train for 1,000 episodes. The Double Q-learning results were averaged over 1000 experiments and the Q-learning was averaged over only 200 experiments as the episodes took longer due to poorer performance on finding the goal.

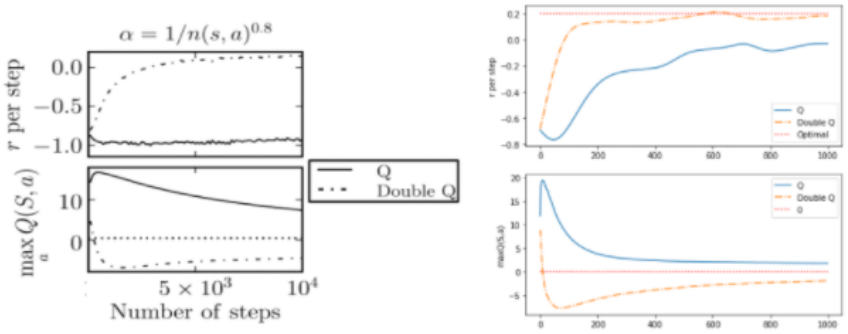
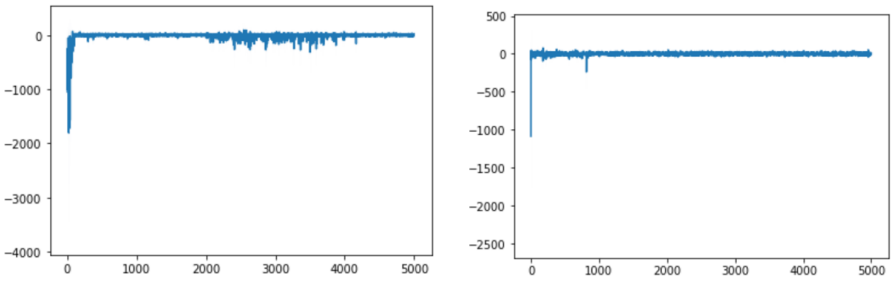


Figure 2: Comparing reproducibility results. Double Q-learning paper (left) and our results (right)

For Triple Q-learning versions we trained the models in Grid World environment with same settings mentioned in Hado van Hasselt.



(a) Average Triple Q-learning (b) Average Triple Q-learning

Figure 3: Results for Triple Q-learning algorithms

The results denoted in figures above are the output of implementing both MaxMin Triple Q-learning and Average Triple Q-learning on the Grid World used in research paper. The results show that MaxMin and triple Q-learning perform almost the same. MaxMin is faster to learn and presents a very low variance. MaxMin Triple Q-learning succeed to perform a robust optimization taking maximum of worst admissible action while for average triple Q-learning it clones in some way the Q-learning greedy behavior with 3 approximators.

3.1.2 Frozen Lake

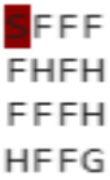


Figure 4: Frozen Lake environment

We used a 4x4 frozen lake with a slip rate of 0.1 to compare Q-learning and Double Q-learning. S is the starting state. G is the goal state. H are holes and F are basic states. Every step has a reward of -1. Any action in a hole has a reward of -100 and ends the episode. Any action in the goal state has a reward of 0. If 100 steps are taken the episode is terminated and a reward of -50 is given. To compare the two algorithms we trained the models for 5000 episodes and averaged over 100 experiments. We compared the undiscounted returns as well as the maximum action value of the starting state. The same method to choose and were used as with the Grid World. No hyper-parameters were changed other than the number of tests and episodes were changed between the two tests.

Figure 5 shows the Q(s,a) for double Q-learning is lower than for Q-learning as expected. For this environment Double Q-learning performed poorer than Q-learning for the set of hyper-parameters chosen. Double Q-learnings performance would likely be improved with a different method of calculating ϵ and α.

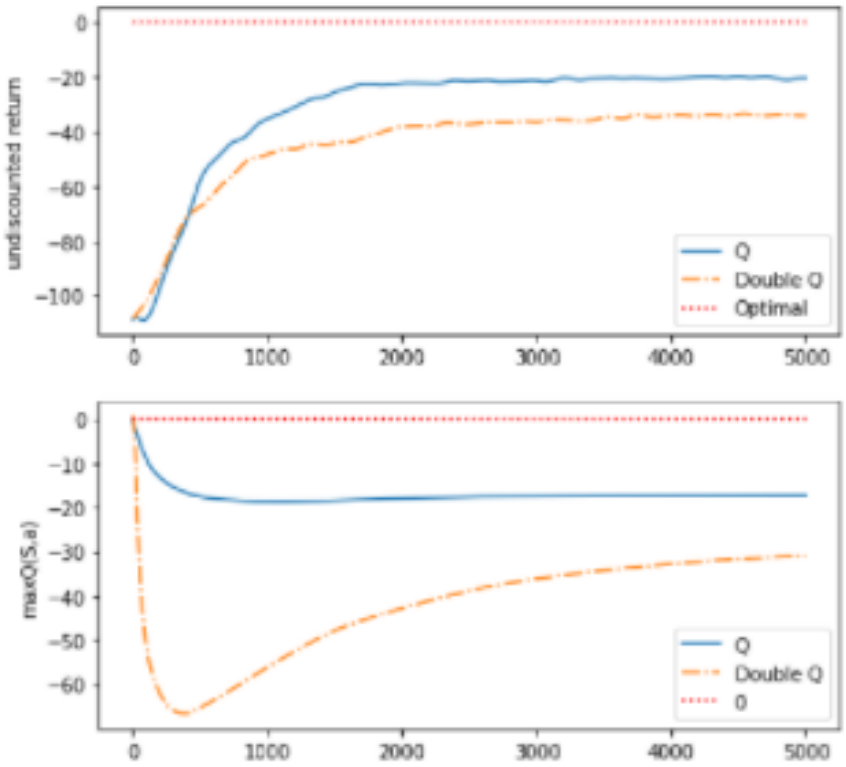
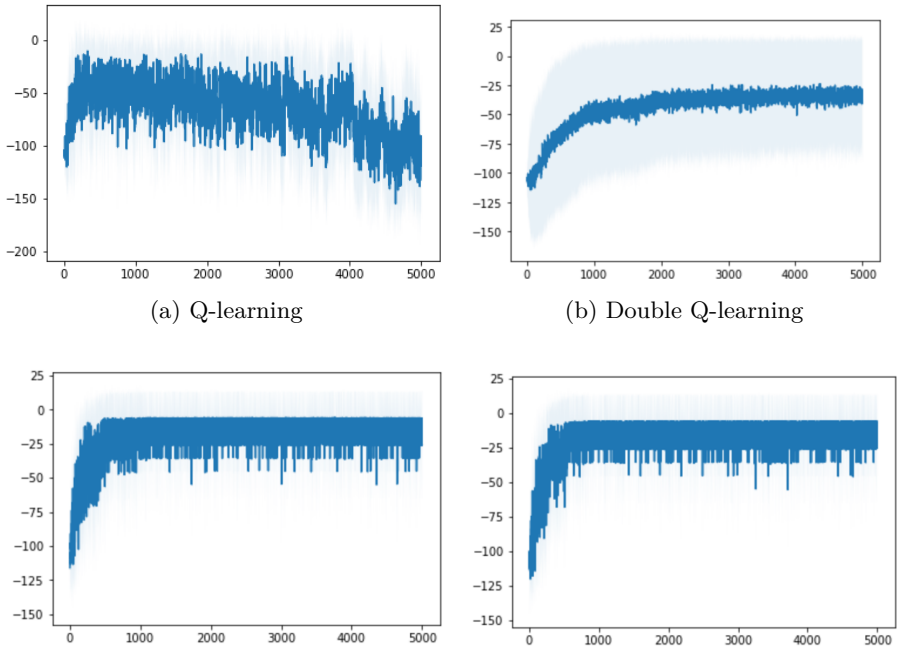


Figure 5: Results of Q-learning and Double Q-learning on the frozen lake environment

For Triple Q-learning versions, models are trained on 5000 episodes and averaged over 100 independent runs with exploration rate $\epsilon = 0.02$, discount $\gamma = 0.99$, and learning rate $\alpha = 0.1$. We compared the undiscounted rewards for average triple Q-learning and MaxMin triple Q-learning. and then compare them with previous results from Q-learning and double Q-learning implementations.



(a) Q-learning

(b) Double Q-learning

(c) Average Triple Q-learning

(d) MaxMin Triple Q-learning

Figure 6: Undiscounted returns of Q-learning algorithms on the frozen lake environment

As depicted in the above figures, we can obviously see that Q-learning has the most variance due to its greedy policy and overestimation of the target value$\hat{Q}$(s',a'). Double Q-learning corrects this high variance, but in this environment it performs worse than Q-learning since it converges to a smaller return. Triple Q-learning and and MaxMin Q-learning perform both better than the two previous methods and in a slightly different way from each other. Both the two methods resulted in a good trade off between exporation and exploitation, reason why both reduce the variance and converge fast to a better return .Though, MaxMin performed slightly better due to more accurate estimate of the target value $\hat{Q}$(s',a').

3.1.3 Cartpole

We used Cartpole environment from openAI. Cartpole environment suggest putting pole in equilibrium while the cart is moving. The states are described through a tuple of 4 variables [x, v, ϑ, ω] denoting the horizontal position, the horizontal velocity of the cart, the angle between the pole and the vertical position and the angular velocity of the pole.

x $\in [-4.8, 4.8]$, v$\in [-4, 4]$, $\theta \in [-0.418, 0.418]$ and $\omega \in [-4, 4]$.

Two actions are possible: move the cart in the direction of x (left/right). An episode ends when a pole deviates in a direction more than 15 degrees. Cumulative reward increases by 1 until the episode ends. We set penalty for ending an episode. We set number of episodes to 2000, learning $\alpha = 0.1$, discount factor $\gamma = 0.99$ and exploration rate $\epsilon = 0.02$. Certainly we needed to transform continuous states to discrete format: we assumed that observed states can be assigned to 20 intervals that lay between lower and upper bound of each variable.

The figure below depicts the performance of Q-learning, Double Q-learning, Triple Average Q-learning, Triple maxMin Q-learning on a Cartpole environment with the discretization:

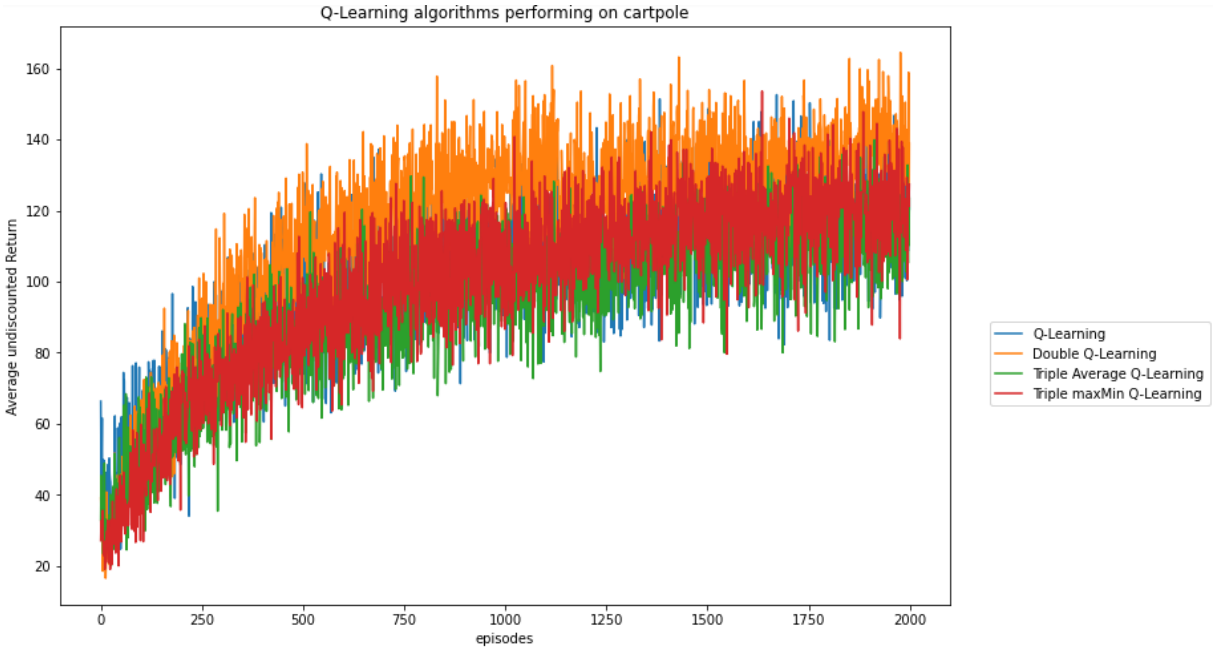


Figure 7: Comparison between Q-learning algorithms performances on Cartpole environment

Figure 6 proves a better performance, in this specific environment, of Double Q-learning. It has the least variance and faster learning is credited to Double Q-learning as well. The algorithm that yields second best performance is the Triple maxMin Q-learning version. The rest of two algorithms are almost performing the same. Algorithms other than simple Q-learning seem to present a promoting improvement by the end of 2000 episodes. Hence, we can say nothing about whether a further tweaking of parameters may result in different behavior of the presented algorithms. If we focus on what the plots are giving here, simple Q-learning shows the highest variance as it is the greediest of all versions. Double Q-learning seems to find an optimal exploration-exploitation trade-off to maximize yield guaranteed by a mutual interaction of two approximators on two disjoint independent data sets. maxMin Q-learning algorithm is likely to be stuck in sub-optimal policy while minimizing an unquantified regret (if an action's estimation is lower than the maximum yield which is truly the case). The average Triple Q-learning may show better performance for a longer run (not guaranteed but hopefully) but in our case it seem that updating Q(s,a) through the average of other two approximators do nothing more than cloning Q-learning behavior.

4 Conclusion

Double and Triple Q-learning were suggested to handle the downfalls of simple Q-learning as outcomes of all time exploitation and overestimating state-action combinations that are up to downgrade the overall return. However those algorithms showed a remarkable improvement on Frozen Lake environment and seemed to be equally or slightly better performing on Cartpole environment (compared to simple Q-learning). This may have a relation with the amount of data we lost due to the discretization process and the overall hyper-parameters.

Considering values of Q(s,a) on Frozen Lake and for 0 and 10% chance to slip, we can see a significant improvement of policy. In fact, states that are close to holes in states 6, 8, 12 and 13, logically should have lower value estimate such that the agent avoids them to build an optimal policy: an agent if he follows a risky way, he might fall by chance/exploration out of borders.

Double Q-learning, as an example, shows a better ability to find a safe way to the goal while maximizing the yield. It succeeds to get expectations as close as possible to reality and is aware of huge possible penalties on the way downgrading a solution even if it seems optimal in terms of distance and average income. We say that adding approximators to work all together to update Q(s,a) values shifts the Q-learning algorithm from being greedy and optimistic to a more robust and a conservative approach to avoid worst scenarios.

5 Code

- Ali : https://colab.research.google.com/drive/1pj2F6HzbpDWDEKhgWrhDClykiml9GqYk?usp=sharing
- Josh : https://colab.research.google.com/drive/1K6Qnu_SGqnAYAO3Dw2Rh3eyrDj6jiU5g?usp=sharing
- Ghofrane: https://colab.research.google.com/drive/1_F0cPhlV2rN5nJMldyQkU1W19MM2Iwyq?usp=sharing

6 Contributions

Josh

- Implementation of Double Q-learning
- Reproducing comparison of Q-learning and Double Q-learning on Grid World
- Comparing results of Q-learning and Double Q-learning Frozen Lake

Ali

- Implementation of simple, double , triple average/maxMin algorithms on Cartpole environment.
- Comparison of results on Cartpole
- Contributed to overall analysis and report reshaping.

Ghofrane

- Documented the Triple Q-learning methods with scientific articles.
- Implemented versions of Triple Q-learning on Frozen Lake and Grid World.
- Contributed to overall analysis.

References

E. Even-Dar and Y. Mansour. Learning rates for Q-learning. Journal of Machine Learning Research, 2003. 3.1

Hado van Hasselt. Double Q-learning. Centrum Wiskunde Informatica. 1, 3.1.1, 3.1.1

LiMing Yi. Lane change of vehicles based on dqn. COMPUTER TECHNOLOGY AND TRANSPORTATION, pp. 593–597, 2020. 2