# Weighted Double Q-Learning

Zhuoyu Wang (260454547)

zhuo.wang@mail.mcgill.ca

Department of Epidemiology and Biostatistics, McGill University

## 1. Introduction

Q-learning is a widely used reinforcement learning algorithm used to solve Markov Decision Processes (MDPs). It has been proven that Q-learning converges to the optimal value function with probability one in the limit under some mild conditions on the learning rates and exploration policy. It has been used to solve many problems and many variations, such as Delayed Q-learning, Phased Q-learning and Fitted Q-iteration, have been proposed.

However, many authors have shown that using maximum value as approximation for the maximum expected value in the Q-learning update formula can largely overestimate the action values and lead to a large performance penalty. To solve the issue, Hasselt (2000) proposed Double Q-learning to avoid the overestimation. However, Q-learning actually brings in underestimation issue. In this project, I proposed Weighted Double Q-Learning to reduce the bias and speed up the convergence to the optimal policy. The notations in this report follows these in the book of Sutton and Barto (2016).

I will discuss the issues of Q-Learning and Double Q-Learning and explain the idea about Weighted Double Q-Learning. An example of grid world will be used to show the performance of Weighted Double Q-Learning comparing with Q-Learning and Double Q-Learning. In the discussion section, I will list some potential further research that the new Weighted Double Q-Learning algorithm can be applied to.

Because of time limitation, some sections, such as prove of the convergence, are omitted. I will finish these sections later and try to submit this work to some conference or journal.

## 2. Estimating the Maximum Expected Value

Consider a set of *M* random variables X = {X$_1$ , . . . , X$_M$ }, with expected values $\mathrm{E}(\mathrm{X}) = \{\mu_1, \dots, \mu_M\}$. The interest is to obtain the maximum expected value of the variables in such a set:

$$\max_i \mu_i. \qquad (1)$$

When the exact value of $\mu_i$ is unknown, we can estimate it from samples of *X$_i$* using some unbiased estimator, such as $\widehat{\mu_i}(R_i) = \frac{1}{|R_i|}\sum_{r\in R_i} r$, where $R_i$ is a sample of *X$_i$*.

### 2.1 The Single Estimator

An common way to estimate (1) is to use the unbiased estimator $\widehat{\mu_i}(R_i)$ to replace $\mu_i$:

$$\max_i \mu_i \approx \max_i \widehat{\mu_i} \, .$$

This estimator was referred as single estimator by Hasselt (2000), and Q-learning uses this method to approximate the value of state $S_{t+1}$ by choosing the maximum estimated action values in state $S_t$.

The expected value of the single estimator is given by

$$\mathrm{E}\left\{\max_i \widehat{\mu_i}\right\} = \int_{-\infty}^{+\infty} x \frac{d}{dx} \prod_{i=1}^{M} F_i^{\widehat{\mu}}(x)\, dx = \sum_{j=1}^{M} \int_{-\infty}^{+\infty} x f_i^{\widehat{\mu}}(x) \prod_{i=1, i\neq j}^{M} F_i^{\widehat{\mu}}(x)\, dx,$$

where $f_i$ is the probability density function of X$_i$, and $F_i(x)$ is its cumulative distribution function. Smith and Winkler (2006) showed that

$$\mathrm{E}\left\{\max_i \widehat{\mu_i}\right\} \geq \max_i E(\widehat{\mu_i}) = \max_i \mu_i.$$

### 2.2 The Double Estimator

The Double Estimator proposed by Hasselt (2010) uses two sets of unbiased estimators: $\hat{\mu}^A = \{\widehat{\mu_1}^A, \dots, \widehat{\mu_M}^A\}$ and $\hat{\mu}^B = \{\widehat{\mu_1}^B, \dots, \widehat{\mu_M}^B\}$, where $\hat{\mu}^A$ and $\hat{\mu}^B$ are independent and estimated using different sets of random samples. Let $a^*$ be the index of an estimator such that $\widehat{\mu_{a^*}}^A = \max_i \widehat{\mu_i}^A$. Then $\widehat{\mu_{a^*}}^B$ is the Double Estimator and we get the approximation

$$\max_i \mu_i \approx \widehat{\mu_{a^*}}^B.$$

The expected value of the Double Estimator is given by

$$\sum_{j=1}^{M} P(j = a^*)\,\mathrm{E}\{\widehat{\mu_{a^*}}^B\} = \sum_{j=1}^{M} \mathrm{E}\{\widehat{\mu_j}^B\} \int_{-\infty}^{+\infty} f_i^A(x) \prod_{i=1, i\neq j}^{M} F_i^A(x)\, dx.$$

The Lemma 1 in Hasselt (2010) showed that

$$\mathrm{E}\{\widehat{\mu_{a^*}}^B\} \leq \max_i \mu_i.$$

### 2.3 The Weighted Double Estimator

Since the Single Estimator overestimates the maximum expected value and the Double Estimator underestimates it, it is desirable to construct a weighted estimator such that the value of the new estimator is between the Single Estimator and the Double Estimator with the hope that the new estimator is less biased. Such Weighted Double Estimator is given by

$$\widehat{\mu_w} = \mathrm{w}\widehat{\mu_{a^*}}^A + (1 - \mathrm{w})\widehat{\mu_{a^*}}^B,$$

where w is the weight with value between 0 to 1.

The expected value of the Weighted Double Estimator is given by

$$\sum_{j=1}^{M} P(j = a^*)\,\mathrm{E}\{\widehat{\mu_w}\} = \sum_{j=1}^{M} \int_{-\infty}^{+\infty} (wx + (1-w)\mathrm{E}\{\widehat{\mu_j}^{B}\}) f_i^A(x) \prod_{i=1, i\neq j}^{M} F_i^A(x)\, dx.$$

The bias in this new estimator depends on the value of w. And there is no guarantee that the Weighted Double Estimator has less bias than both of the previous two estimators. However, it is guaranteed that it has less bias than at least one of the two estimators. In addition, since the unbiased estimators $\widehat{\mu_i}$ are usually normally distributed with bell shapes, even a weight of w=0.5 should provide a better estimator for most of the time.

## 3. Weighted Double Q-Learning

The Weighted Double Q-Learning algorithm uses the weighted double estimator to estimate the value of the next state. The algorithm is given in Algorithm 1.

**Algorithm 1** Weighted Double Q-Learning

```
1:  Initialize $Q^A$, $Q^B$, s
2:  repeat
3:     Choose $a$, based on $Q^A(s, .)$ and $Q^B(s, .)$, observe r, s'
4:     Choose (e.g. random) either UPDATE(A) or UPDATE(B)
5:     if UPDATE(A) then
6:        Define $a^*=\text{argmax}_a Q^A(s', a)$
7:        $Q^A(s,a) \leftarrow Q^A(s,a) + \alpha(s,a)(r + \gamma(wQ^A(s',a^*) + (1-w)Q^B(s',a^*)) - Q^A(s,a)$
8:     Else if UPDATE(B) then
9:        Define $b^*=\text{argmax}_a Q^B(s', a)$
10:       $Q^B(s,a) \leftarrow Q^B(s,a) + \alpha(s,a)(r + \gamma(wQ^B(s',a^*) + (1-w)Q^A(s',a^*)) - Q^B(s,a)$
11:    end if
12:    $s \leftarrow s'$
13  until end
```

Similar to the Double Q-Learning, the Weighted Double Q-Learning also stores two Q functions: $Q^A$ and $Q^B$. The only difference is that when updating a Q function, its updated value is a weighted sum of the two Q functions for the next state.

The convergence of the Weighted Double Q-Learning in the limit is easy to see. Since both Q-Learning and Double Q-learning converge in limit, the Weighted Double Q-Learning, which is between these two, must converge in limit. The complete proof of the convergence in the limit has not been finished yet and will be added to the manuscript.

## Experiment

We used the same grid world in Hasselt (2010) to show the performance of the Weighted Double Q-Learning compared with Q-Learning and Double Q-Learning.

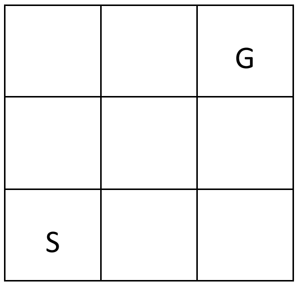

Figure 1. The grid world.

The grid world MDP is shown in Figure 1. The starting state is S and the goal state G. Each state has four actions, corresponding to go one step towards to north, east, south or west. Each time the agent walks off the grid, it goes back to the same state. Each time the agent reaches a non-G state, it receives a random reward of −12 or +10 with equal probability. When the agent reaches the G state, it receives +5 and the episode ends.

We considered Q-learning, Double Q-learning, and Weighted Double Q-learning with weights of 0.25, 0.5 and 0.75. For all these algorithms, the exploration was ε-greedy with ε (s) = $1/\sqrt{n(s)}$, where n(s) is the number of times visiting state s. Two types of learning rate were considered: linear $\alpha_t(s,a) = 1/n_t(s,a)$, or polynomial $\alpha_t(s,a) = 1/n_t(s,a)^{0.8}$. For Double Q-learning and Weighted Double Q-Learning, $n_t(s,a) = n_t^A(s,a)$ if $Q^A$ is updated and $n_t(s,a) = n_t^B(s,a)$ if $Q^B$ is updated, where

$n_t^A(s,a)$ and $n_t^A(s,a)$ store the number of updates for $Q^A$ (s, a) and $Q^B$ (s, a). The discount rate is $\gamma = 0.95$. In this setting, the optimal policy ends an episode in 4 steps with expected total rewards of +2. So the optimal average reward per step is +0.5. The optimal value of the maximally valued action in the starting state is $5\gamma^4 - \sum_{k=0}^{3}\gamma^k$ ≈0.36.

We ran 500 experiments, 20,000 steps each. The average reward per step and the value of the maximally valued action in the starting state are plotted in Figures 2 and 3.When linear learning rate is applied, none of the reward per step or the value of the maximally valued action in the starting state converges after 20,000 steps, though Double Q-learning and Weighted Double Q-learning are closer to the optimal reward per step, they still do not converge even after 20,000 steps. However, when polynomial learning rate is used, the convergence in limit is speeded up for all five algorithms. In addition, the Double Q learning ,Weighted Double Q-learning convergences much faster than other parameters.

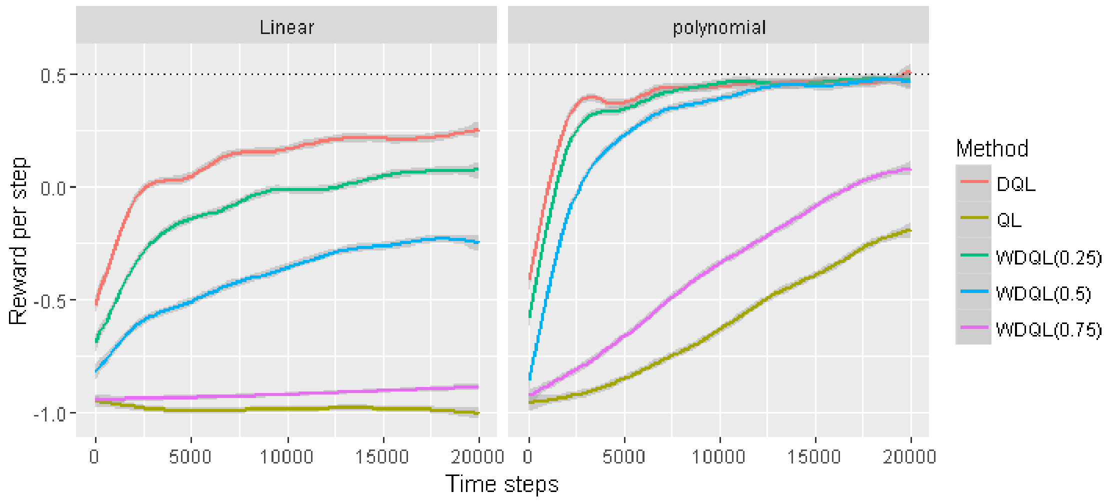

Figure 2. The average reward per step over 20,000 steps. The dash line is the optimal average reward per step.

Similarly, when polynomial learning rate is used, the convergences are much faster than when the linear learning rate is used. And the Weighted Double Q-Learning converges within 5000 times and is much better than any other algorithms.

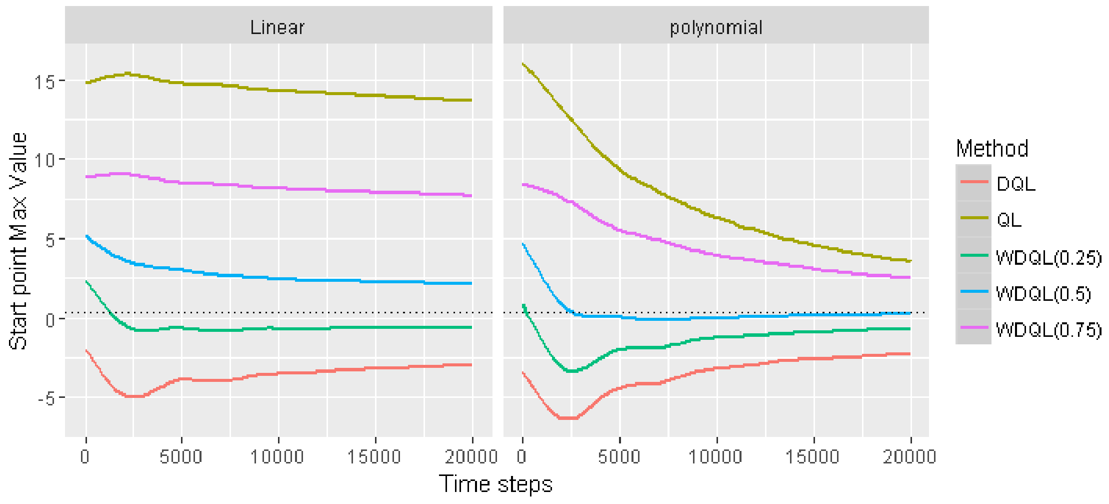

Figure 3. The value of the maximally valued action in the starting state. The dash line is the optimal value of the maximally valued action in the starting state.

## Discussion

In this project, we proposed the new Weighted Double Q-Learning algorithm. The illustration in this project showed that this algorithm converges much faster than Q-learning and Double Q-learning, especially when the learning rate is polynomial. Though more investments on the performance of the Weighted Double Q-Learning algorithm are required, these results show considerable potential of this new algorithm.

One question remain in this project is how to choose the weight. More experiments are needed to show if a default weight of 0.5 works well in most of the applications. Or should we adaptively update the weight? One potential way to deal with it is to update the weight according to the standard deviations of the two estimators, which are determined by $n_t(s, a)$.

Since there are many variations of Q-learning algorithms, including Delayed Q-learning, Phased Q-learning and Fitted Q-iteration, it would interesting to see if the convergence performances would be improved when Weighted Double Q-learning is applied to them.

Another potential research question is to extend DQN into Weighted Double DQN and evaluate the performance improvement.

## Reference

C. J. C. H. Watkins. Learning from Delayed Rewards. PhD thesis, King's College, Cambridge,England, 1989.

T. Jaakkola, M. I. Jordan, and S. P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. Neural Computation, 6:1185–1201, 1994.

E. Even-Dar and Y. Mansour. Learning rates for Q-learning. Journal of Machine Learning Research, 5:1–25, 2003.

R. S. Sutton and A. G. Barto. Reinforcement Learning: An Introduction. (Draft) 2016.

J. E. Smith and R. L. Winkler. The optimizer's curse: Skepticism and postdecision surprise in decision analysis. Management Science, 52(3):311–322, 2006.

Van Hasselt, Hado, Guez, Arthur, and Silver, David. Deep reinforcement learning with double q-learning. arXiv preprint arXiv:1509.06461, 2015.

Hasselt, Hado V. Double q-learning. In Advances in Neural Information Processing Systems, pp. 2613–2621, 2010.