Learning Boosts with Modified Differential Q-Learning

Damir Vrabac, Arec L. Jamgochian, Anmol M. Kagrecha

June 2021

Abstract

A common reinforcement learning objective is for an agent to learn how to behave optimally in an environment when able to query the environment with actions to observe state transitions and immediate rewards. Recent work achieves this objective without forcing discounting by maintaining an estimate of reward rate and performing temporal difference updates for reward received relative to that rate (Wan et al., 2020). In this project, we extend these updates to provide boosts towards convergence in environments where the joint state and action space can be partitioned usefully. In environments where rewards are tied within an aptly chosen partition, our method, Modified Differential Q-Learning, provides faster convergence. We introduce two environments which exhibit such partitions: `One-step Rewards` and `Slippery GridWorld`. We compare cumulative and average regret received by our method, the method of Wan et al., and vanilla Q-Learning, finding that our method outperforms the others.

1 Introduction

In this project, we consider the agent's environment to be a Markov decision process (MDP). In particular, we consider the average reward formulation of MDPs. In this environment, the experience is considered to be continuing and not broken up into episodes. An agent seeks to maximize the average reward per step or the reward rate, which we denote by λ^*. Average reward MDP formulation is considered an important step in developing agents that can operate in real world environments.

Popular reinforcement learning algorithms like Q-Learning may not converge in the average reward setting and would require a a discount factor to avoid diverging. Introducing a discount factor might lead to policies that are not optimal when trying to maximize the average reward per step. Even in environments where Q-learning learns an optimal policy, the learning is slow in practice and requires a lot of hyper-parameter tuning. This is largely because Q-values get updated very slowly and there is no effective information transfer among Q-values.

Differential Q-Learning (DQL) is an excellent algorithm proposed in Wan et al., 2020 which enjoys sound theoretical guarantees under very mild conditions and works quite well in practice. In addition to maintaining Q-values, DQL also maintains a statistic for the average reward. Maintaining this statistic helps in effectively transferring information across state-action pairs, which leads to much faster convergence.

One of the issues with DQL is that it tries to learn an asymptotic quantity, i.e., reward rate. It is reasonable to expect that if the environment exhibits phenomena that provide high rewards which are easier to learn than the reward rate, then learning appropriate statistics about the environment in addition to the reward rate would lead to better performance.

In this work we introduce `One-step Rewards` and `Slippery GridWorld`, two environments that captures how well an agent learns phenomena across different time-scales and its ability to leverage what it has learned to perform well in practice. Moreover, we extend the temporal difference in the Differential Q-Learning (DQL) algorithm to include statistics of partitions from the joint state and action space. Our proposed algorithm, Modified Differential Q-Learning (MDQL), outperforms DQL in both environments when the partitions of the state-action pair set are chosen appropriate to the environment.

2 Background

This section provides background on Q-Learning, relative value iteration (RVI) Q-Learning, and differential Q-Learning.

2.1 Q-Learning

In a reinforcement learning environment, an agent makes decisions based on a past history of actions and observations in order to maximize an internal notion of cumulative reward.

In a lot of reinforcement literature, three simplifying assumptions are made. The assumption of full observability states that the observation O_{t+1} an agent receives after from the environment after taking an action A_t is actually the true state of the environment—that is to say $O_t = S_t \forall t$. The Markov assumption states that future observations are conditionally independent of histories given the most recent action and observation. That is to say, $O_{t+1} \perp H_t \mid (O_t, A_t)$, or alternatively, $\mathbb{P}(O_{t+1} \mid H_t) = \mathbb{P}(O_{t+1} \mid O_t, A_t)$. Finally, an additional assumption is made that the agent receives the reward it must maximize from the environment, as opposed to defining the reward for itself.

With these three assumptions, the reinforcement learning objective often becomes to design a system in which the agent can learn to behave optimally when it does not know the internal dynamics of the environment (which under these assumptions degenerates to $\mathbb{P}(S_{t+1}, R_{t+1} \mid S_t, A_t)$). However, the agent is able to query the environment with actions to observe rewards and state transitions.

One way of achieving this reinforcement learning objective is through Q-Learning. In Q-Learning, the agent stores statistics of the estimated return $Q(s, a)$ in state s if the agent were to take action a immediately and act optimally afterwards. The optimal policy can be extracted from a converged Q function by taking actions which maximize Q in each state (e.g. $\pi(s) \in \arg\max_a Q(s, a)$).

Since the estimated cumulative reward in a non-terminating environment can be infinite, it is often practical to introduce a discount factor $\gamma \in [0, 1)$ and track the expected cumulative discounted reward $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R_{t+1}]$ when starting in state s, taking action a immediately, and behaving optimally thereafter.

With the γ-discounted formulation of reward, one can formulate the vanilla Q-learning update for Q by computing a temporal difference δ_{t+1} between expected and received immediate reward, and by updating the Q statistic at the visited state-action pair:

$$\delta_t = R_{t+1} + \gamma \max_{a \in \mathcal{A}} Q_t(S_{t+1}, a) - Q_t(S_t, A_t) \tag{1}$$

$$Q_{t+1}(S_t, A_t) = Q_t(S_t, A_t) + \alpha_t \delta_t. \tag{2}$$

Here we note that at each point in time, only a single Q statistic is updated (at S_t, A_t), where other values stay the same[1].

[1]If introducing eligibility traces to handle delayed reward, it is common to update more than one statistic in a single time step to attribute rewards to former state-action pairs

It is shown that with certain assumptions on α_t (e.g. non-summable but square-summable) and with a sufficient exploration strategy, Q-Learning will converge to the optimal policy with respect to the discounted cumulative return. In this paper, we use an ϵ-greedy exploration policy, which chooses

$$A_t = \begin{cases} \text{unif}(\mathcal{A}) & \text{w.p. } \epsilon \\ \text{unif}(\arg\max_a Q_t(S_t, a)) & \text{w.p. } 1 - \epsilon. \end{cases} \tag{3}$$

2.2 RVI Q-Learning

One significant downside to vanilla Q-Learning is that it assumes (or artificially forces) a γ discount factor. That is, it builds policies based on expected cumulative discounted return, as opposed to expected (non-discounted) cumulative return. Unfortunately, making this simplifying discounting assumption can very easily lead to non-optimal policies with respect to the true expected cumulative return. An example of this is examined in Section 2.3.1.

In RVI Q-Learning (Abounadi et al., 2001), the assumption of discounting is relaxed. To prevent the Q values from diverging (which they would, since reward is continually added while $V(S_{t+1})$ is no longer discounted), a relative value function $f(Q)$ is subtracted uniformly from all temporal differences. The Q-Learning update functions become

$$\delta_{t+1} = R_{t+1} - f(Q) + \max_{a \in \mathcal{A}} Q_t(S_{t+1}, a) - Q_t(S_t, A_t) \tag{4}$$

$$Q_{t+1}(S_t, A_t) = Q_t(S_t, A_t) + \alpha_t \delta_{t+1}. \tag{5}$$

RVI Q-Learning is shown to converge with some mild assumptions on f, which may be, for example, the value at a fixed state.

2.3 Differential Q-Learning

Unfortunately, the effectiveness of RVI Q-Learning is highly dependent on the choice of f, a choice which is hard to make a priori. Wan et al. formulate a version of Q-Learning which operates on the expected cumulative reward *rate* when following a policy (Wan et al., 2020). The reward rate r^* is defined as $\sup_\pi r(\pi, s)$, where

$$r(\pi, s) = \lim_{n \to \infty} \frac{1}{n} \sum_{t=1}^{n} \mathbb{E}[R_t \mid S_0 = s, A_{0:t-1} \sim \pi]. \tag{6}$$

The differential Q-Learning algorithm implements the following updates:

$$\delta_t = R_{t+1} - \bar{R}_t + \max_{a \in \mathcal{A}} Q_t(S_{t+1}, a) - Q_t(S_t, A_t) \tag{7}$$

$$Q_{t+1}(S_t, A_t) = Q_t(S_t, A_t) + \alpha_t \delta_t \tag{8}$$

$$\bar{R}_{t+1} = \bar{R}_t + \eta \alpha_t \delta_t. \tag{9}$$

It is shown that under the assumption that for every pair of states, there exists a policy that can transition from one to the other in a finite number of steps with nonzero probability, then $\bar{R}_t$ will converge to r^* and

Q_t will converge to a solution of q in the Bellman equation

$$q(s,a) = \sum_{S',R} \mathbb{P}(S', R \mid S, A)(R - r^* + \max_{a'} q(S', a')) \tag{10}$$

The proof is made by tying the differential Q-Learning update to the RVI Q-Learning update.

2.3.1 Example: Delayed Reward Environment

To illustrate the necessity to consider the time-averaged reward of a non-discounted MDPs rather than the average artificially discounted reward, consider an environment where an agent had the choice between a left and right action from a root node. The left action explores a path in which an reward of $+\gamma$ is given to the agent after one turn, while the right action explores a path in which a reward of $+1$ is given to the agent after three turns. This environment is depicted in Figure 1.

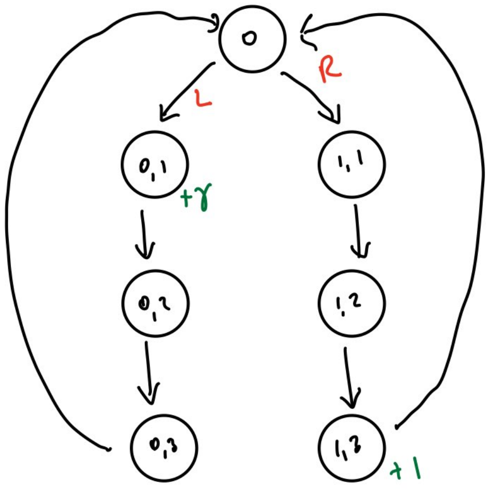
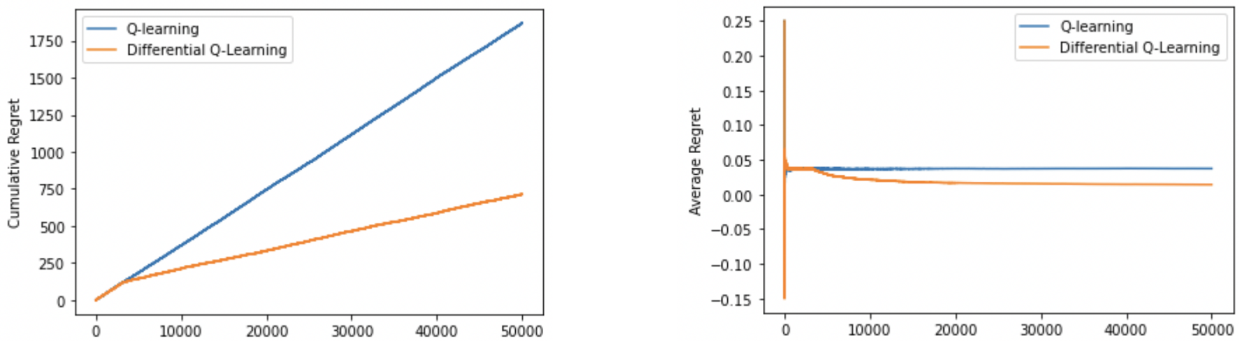


Figure 1: Simple `Delayed Reward` environment. Though the optimal policy traverses the right path, the optimal policy with artificial γ-discounting will traverse the left path, collecting suboptimal reward.

Though intuitively, the optimal policy will continually traverse the right path, collecting an time-averaged reward of 0.25, solving this environment with a discount factor of γ (as is the case in Q-Learning) will lead to a sub-optimal policy which continually traverses the left path. Differential Q-Learning on the other hand, would yield the correct policy since it operates on the true average return rather than the discounted return. We make this comparison in Figure 2, where we show differential Q-Learning leading to lower regret than Q-learning when using an $\epsilon-$greedy exploration policy.

3 Methodology

In this section, we provide motivation for and describe our Modified Differential Q-Learning algorithm, as well as describe some environments in which we see it to be useful.



(a) Cumulative Regret

(b) Average Regret

Figure 2: Comparison of cumulative and average regret between Q-Learning and differential Q-learning in the `Delayed Reward` environment.

3.1 Modified Differential Q-Learning

In our modification of differential Q-Learning, we take advantage of the knowledge that certain environments have state-action partitions in which rewards for all state-action pairs in a particular partition may be tied to one another. For example, you might consider an environment in which taking action a is always better than taking action b, regardless of the starting state or true reward received. Our motivation is that this knowledge might help to learn to act optimally quicker than differential Q-learning.

We make this modification by introduction a new statistic $\bar{B}(\chi)$ which encodes the boost over average reward $\bar{R}$ an agent might expect when finding themselves in χ, a subset of $\mathcal{S} \times \mathcal{A}$. With this formulation, we hope for $\bar{R}$ to converge to the average reward received among all states and for $\bar{R} + \bar{B}(\chi)$ to converge to the average reward received among all state-action pairs in χ when following an optimal policy.

Our modified differential Q-Learning (MODQ) update is as follows:

$$\delta_t = R_{t+1} - \bar{R}_t + \max_{a \in \mathcal{A}} Q_t(S_{t+1}, a) - Q_t(S_t, A_t) \tag{11}$$

$$\sigma_t = R_{t+1} - \bar{R}_t - \bar{B}_t(\chi_t) + \max_{a \in \mathcal{A}} Q_t(S_{t+1}, a) - Q_t(S_t, A_t) \tag{12}$$

$$Q_{t+1}(S_t, A_t) = Q_t(S_t, A_t) + \alpha_t \sigma_t \tag{13}$$

$$\bar{R}_{t+1} = \bar{R}_t + \eta \alpha_t \delta_t \tag{14}$$

$$\bar{B}_{t+1}(\chi_t) = \bar{B}_t(\chi_t) + \beta \alpha_t \sigma_t. \tag{15}$$

Here, $\bar{R}$ makes the same update as in differential Q-Learning, covering the gap between received reward and expected average reward *without* state partitioning. However, Q and $\bar{B}$ make updates which cover the gap between received reward and expected reward *with* state partitioning. This ensures that $\bar{R}$ remains a mean statistic while $\bar{B}$ encodes just the boost from being in the particular partition.

3.2 Environment: `One-step Rewards`

In the `One-step Rewards` environment, we have N states and $N \times M$ actions. With each action, the agent chooses the state to transition to next, and the reward to receive in the current state. $\mathcal{A} = \{(n', m) : n' \in \{0, .., N-1\}, m \in \{0, ..., M-1\}\}$, $\mathbb{P}(s' = n' \mid s, a = (n', m)) = 1 \ \forall s$, and $R(s = n, a = (n', m)) = \frac{n+m}{\Delta}$ with probability 1, where $\Delta = N + M - 2$ so that $R \in [0, 1]$. This environment is depicted in Figure 3.

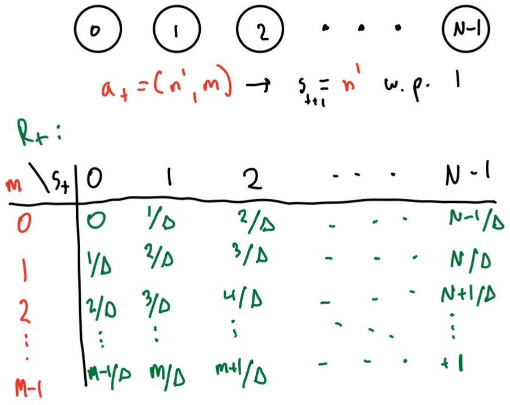

0
1
2
N-1
a_t = (n', m) → s_{t+1} = n' w.p. 1
R_t :
m \ S_t
0 1 2 . . . N-1
0 | 0 1/Δ 2/Δ . . . N-1/Δ
1 | 1/Δ 2/Δ 3/Δ . . . N/Δ
2 | 2/Δ 3/Δ 4/Δ . . . N+1/Δ
M-1 | M-1/Δ M/Δ M+1/Δ . . . +1

Figure 3: The `One-step Rewards` environment.

In our experiments, we explore a partition of the state-action space based on the second index of the action, noting that higher second-actions will always yield higher rewards. That is, $\chi_t(S_t, A_t = (n'_t, m_t)) = m_t$.

3.3 Environment: `Slippery GridWorld`

The `Slippery GridWorld` environment is similar to `GridWorld`, except it is easier to move horizontally than vertically. We have L^2 states and 5 actions—our state space is an L-by-L grid, and our action space $\mathcal{A} = \{0, L, R, U, D\}$. The 0 action does nothing, the L/R actions move left or right with success probability 0.9, and the U/D actions move up or down with success probability 0.5. The agent receives reward $R(s, a, s' = (s'_h, s'_v)) = \frac{s'_h \times s'_v}{\Delta}$ with probability 1, where $\Delta = L^2$ so that $R \in [0, 1]$

This environment is depicted in Figure 4.

In our experiments, we explore a partition of that state-action space based on the horizontal state, noting that higher horizontal-states will yield higher rewards, and it is relatively easy to transition horizontally. That is, $\chi_t(S_t = (S_{h,t}, S_{v,t}), A_t) = S_{h,t}$.

4 Experiments

In our experiments, we measure the performance of our modified differential Q-Learning method in comparison the vanilla Q-learning and differential Q-learning on the environments proposed in Section 3.

4.1 Experiment setup

We compare performance in our two environments with the appropriate boosting partitions as described in Sections 3.2 and 3.3. We consider a `One-step Rewards` environment with $N = 4$ states and $M = 4$ rewards per state. In the `Slippery GridWorld Environment`, we use a side length of $L = 8$.

To compare the methods, we generate plots of both cumulative regret and average regret when applying the different methods with an ϵ−greedy exploration policy, where the ϵ's follow a universal, appropriate

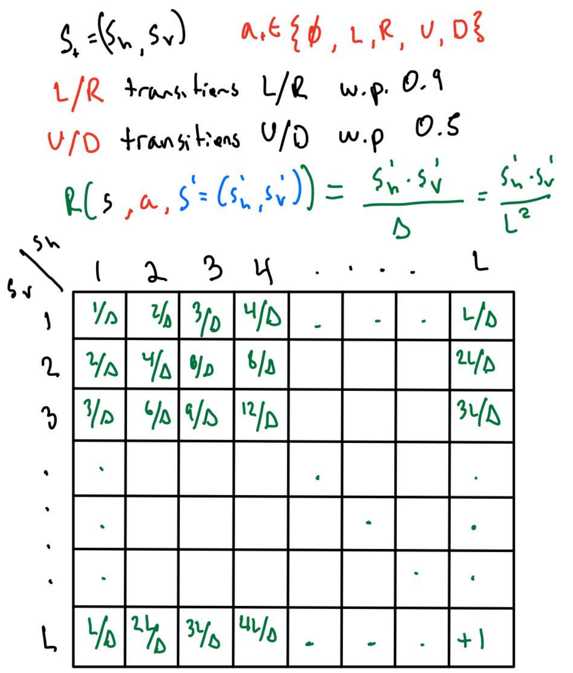

$S_t = (S_h, S_v)$ $a_t \in \{\emptyset, L, R, U, D\}$
L/R transitions L/R w.p. 0.9
U/D transitions U/D w.p 0.5
$R(s, a, s' = (s_h', s_v')) = \frac{s_h' \cdot s_v'}{\Delta} = \frac{s_h' \cdot s_v'}{L^2}$
s_h
s_v
1 2 3 4 . . . L
1/Δ 2/Δ 3/Δ 4/Δ L/Δ
2/Δ 4/Δ 6/Δ 8/Δ 2L/Δ
3/Δ 6/Δ 9/Δ 12/Δ 3L/Δ
L/Δ 2L/Δ 3L/Δ 4L/Δ +1

Figure 4: The `Slippery GridWorld` environment.

decay schedule. Cumulative regret is defined as

$$\text{Regret}_T = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \lambda_* - R_{t+1} \right], \tag{16}$$

where the optimal average reward $\lambda_* = \sup_\pi \lambda_\pi$, with

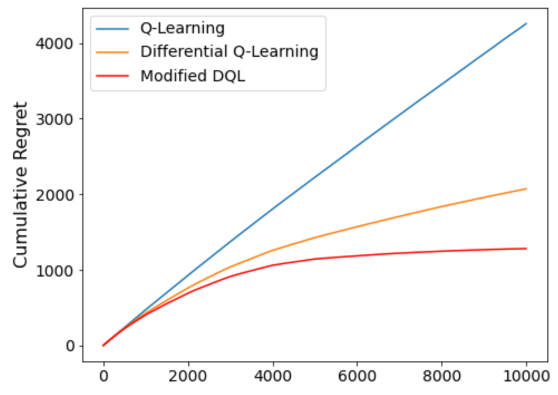
$$\lambda_\pi = \liminf_{T \to \infty} \mathbb{E}_\pi \left[\frac{1}{T} \sum_{t=0}^{T-1} R_{t+1} \middle| \mathcal{E} \right]. \tag{17}$$

The average regret received up to time T is defined as $\frac{1}{T}\text{Regret}_T$.

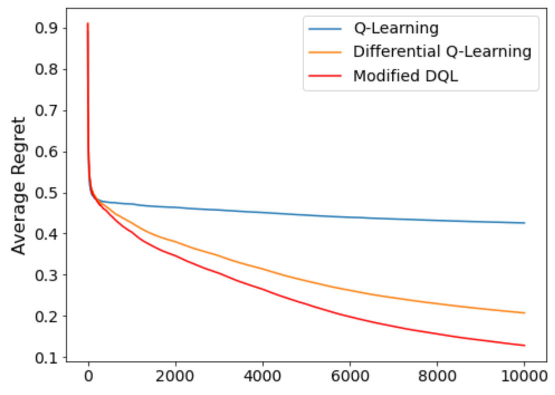
4.2 Results and discussion

In Figure 5, we compare the three methods on the `One-Step Rewards` environment, using a boosting partitioning along the second dimension of the action space. In Figure 6, we compare the three methods on the `Slippery GridWorld` environment, using a boosting partitioning along the first dimension of the state space. In all figures, lines indicate the regret averaged among 50 experiments. In both experiments, we find that our boosting modification converges more quickly to an optimal policy compared to differential Q-learning and Q-learning with an added discount factor γ. We noticed that our improvement, in terms of regret, was even starker when using ϵ-greedy with a fixed ϵ as opposed to a decaying schedule (not shown).

To observe the effect of the choice of state-action transition, we also attempt to partition the `Slippery GridWorld` environment by its actions rather than states. In Figure 7, we compare the three methods with action partitioning, which is not at all a natural partitioning for that environment. We find that our modification of differential Q-Learning significantly underperforms compared to differential Q-learning.

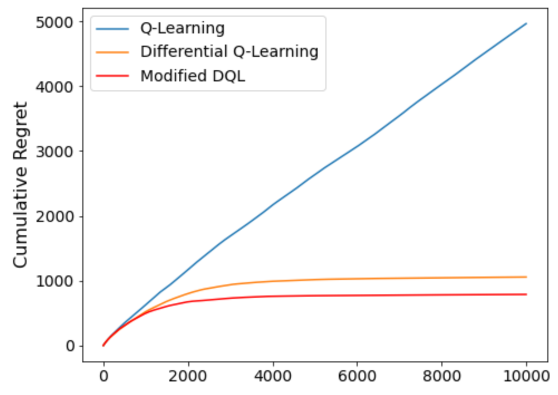


(a) Cumulative Regret

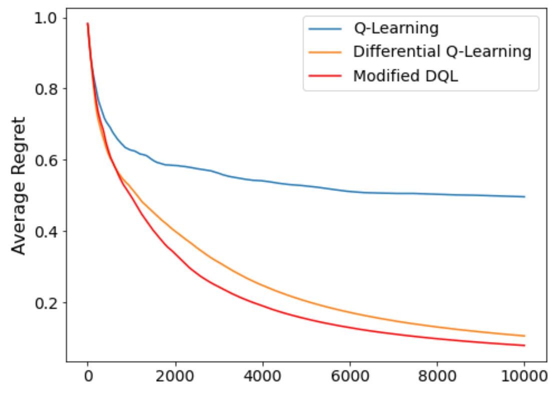


(b) Average Regret

Figure 5: Comparison of cumulative and average regret between Q-Learning, differential Q-learning, and modified differential Q-learning when partitioning based on actions in the `One-Step Rewards` environment.

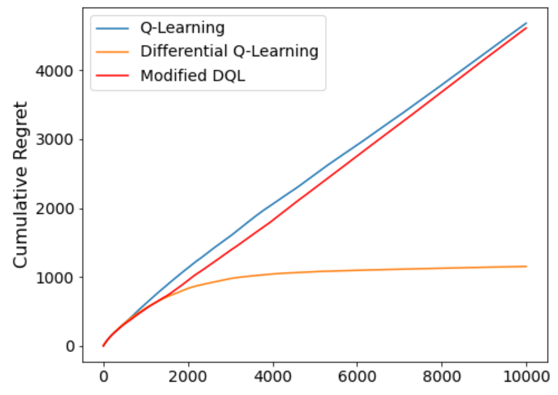


(a) Cumulative Regret

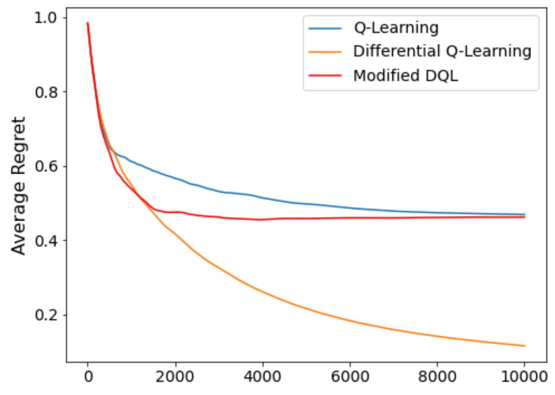


(b) Average Regret

Figure 6: Comparison of cumulative and average regret between Q-Learning, differential Q-learning, and modified differential Q-learning when partitioning based on horizontal state in the `Slippery GridWorld` environment.



(a) Cumulative Regret



(b) Average Regret

Figure 7: Comparison of cumulative and average regret between Q-Learning, differential Q-learning, and modified differential Q-learning when partitioning based on actions in the `Slippery GridWorld` environment.

5 Conclusion

For many complex environments Q-Learning may not be an appropriate algorithm for learning the optimal policy since it does not necessarily converge to the optimal average reward. However, Differential Q-Learning has been proven to converge to the optimal policy given some assumptions on the environment, in particular that the environment can be formulated as a communicating MDP. We introduced two environments, `One-step Rewards` and `Slippery GridWorld` to analyze this phenomena. Moreover, we proposed a new algorithm, Modified Differential Q-Learning, which adds new statistics of partitions of the joint state-action set to the temporal difference. We showed that it is possible to converge even faster than Differential Q-Learning, in particular that the Modified Differential Q-Learning does that in both environments when the partitions are chosen appropriately.

References

Abounadi, J., D. Bertsekas, and V. S. Borkar (2001). "Learning algorithms for Markov decision processes with average cost". In: *SIAM Journal on Control and Optimization* 40.3, pp. 681–698.

Wan, Y., A. Naik, and R. S. Sutton (2020). "Learning and Planning in Average-Reward Markov Decision Processes". In: *arXiv preprint arXiv:2006.16318*.