Empirical Study of Differential Q-learning on Continuing Control Tasks

Bahar Boroomand Ghahnavieh
University of Alberta
bborooma@ualberta.ca

Mohammadreza Hami
University of Alberta
hami@ualberta.ca

Shayan Karimi
University of Alberta
skarimi3@ualberta.ca

Parham Mohammad Panahi
University of Alberta
parham1@ualberta.ca

Abstract

Differential Q-learning is a recently developed average reward algorithm for off policy control. We conduct an empirical study of differential Q-learning on two continuing tasks, Catch and Pendulum, in the linear function approximation setting. To the best of our knowledge, this is the first comparison between discounted and differential Q-learning in the literature. We provide performance results and sensitivity analyses. Our results show that differential Q-learning is no worse than discounted Q-learning on Pendulum but is not a suitable algorithm for Catch.

1 Introduction

One of the most important problem settings for the future of AI is where an agent is continually interacting with its environment, without restarts or cutoffs. This is called the continuing control problem in reinforcement learning. Many real-world situations such as autonomous driving can be modeled as continuing control tasks. Even in many cases where experience can be broken into episodes, they are mostly in such a long time frame that it is better to look at it as a continuing problem rather than episodic. Arguably, many biological intelligent organisms are also interacting with continuing problems. Therefore, it is crucial to examine the methods for solving these problems.

It is common practice to use discounting for continuing tasks which gives more value to immediate rewards and less to future rewards. Using this discounting method is problematic in large-scale continuing problems. One problem with discounting is that the agent places more importance on immediate reward than on delayed reward. This is undesirable when we face delayed rewards, as we are in many continuing problems [Sutton and Barto, 2018]. Another issue is that discounting for continuing problems is not a valid optimization problem and therefore incompatible with function approximation [Naik et al., 2019].

Another introduced approach to tackle this type of problem is using average reward reinforcement learning [Sutton and Barto, 2018]. In the average reward setting, we do not discount future rewards, we give equal importance to each reward in the future. In this setting, the quality of a policy is defined by the average rate of reward.

Surveys about the average-reward learning method are provided by Das et al. [1999] and Dewanto et al. [2020]. In Das et al. [1999] authors present the Semi-Markov Average Reward Technique on a combinatorially large problem of determining the optimal preventive maintenance schedule of a production inventory system. In Dewanto et al. [2020] authors extended the previous work and also covered policy iteration and function approximation methods for average reward methods. One of the recent algorithms developed under the average reward paradigm is Differential Q-learning [Wan et al., 2021]. This algorithm is the primary object of study in this work.

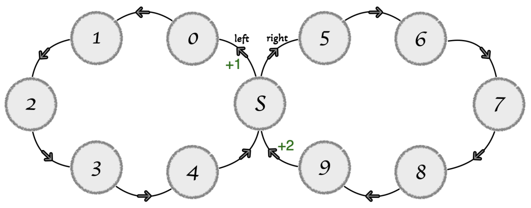


Figure 1: The two-choice MDP

In this paper, we conduct experiments on differential Q-learning with linear function approximation to provide empirical evidence for the extension of this algorithm to the function approximation setting. We compare discounted Q-learning and differential Q-learning on two continuing control task, Catch and Pendulum. These tasks differ in the way they generate delayed reward. We investigate performance, sensitivity to hyperparameters, and experiment with different features.

2 Background

The agent and environment interaction is mostly formulated in the Markov Decision Process in reinforcement learning. In this setting, we have 4 components which are the State set, the Action set, Rewards, and Environment dynamics or Transition dynamics.

This dynamic is defined as $p : S \times R \times S \times A \to [0, 1]$, where S is the state set, R is the set of rewards and A is the action set.

$$p\left(s^{\prime}, r \mid s, a\right) \doteq Pr\left\{S_t = s^{\prime}, R_t = r \mid S_{t-1} = s, A_{t-1} = a\right\} \tag{1}$$

At each time step, the agent is in a state s and takes an action a in that state. In consequence, in the next time step, it receives the specific amount of reward r and moves to the new state s' according to MDP dynamics. Also, there is a function called *policy* which can map each state to a specific action or a distribution over all actions in the action set.

In reinforcement learning, tasks are either continuing or episodic. In continuing tasks, the agent-environment interaction does not break into subsequences [Sutton and Barto, 2018] and there is no terminal state. In order to solve the continuing tasks, two main approaches exist, which are discounted methods and average reward methods. In what follows, an example of a continuing task is given which helps to explain the average reward methods better.

In figure 1, there is an MDP with two cycles in which the agent starts at state s and can either take action *right* or action *left*. The reward at each step is zero except for transitioning from state S to state 0 in the left wing in which the agent gets $+1$ reward and moving from state 9 to state S in which the agent receives $+2$ reward. The goal is to find the optimal policy in this problem setting.

In the mentioned MDP, in case of a small discount factor, the agent is going to prefer the closer reward and will not act farsighted. Hence, it takes the *left* action in state S, which is the wrong policy to follow, because, in case of choosing to go *right*, it could get a reward of $+2$. So choosing the small discount factor is misleading. As a result, the gamma must be quite large. This MDP can be generalized for many more steps on each side and the agent will need a larger gamma to be more farsighted. So γ will get very close to 1 but it can not be equal to that due to the possibility of getting an infinite return. Average reward formulation on the other hand can overcome this issue and it can be used in order to decide which policy is better than the other one. Here, if the agent utilizes the average reward setting, it can correctly choose to follow the best policy in which the right wing is selected.

The average reward (reward rate) is calculated according to equation 2.

$$r(\pi) = \sum_s \mu_\pi(s) \sum_a \pi(a \mid s) \sum_{s^{\prime}, r} p\left(s^{\prime}, r \mid s, a\right) r \tag{2}$$

In the following equation π is the policy and $\mu(s)$ is the steady-state distribution,

$$\mu_\pi(s) \doteq \lim_{t \to \infty} Pr\left\{S_t = s \mid A_{0:t-1} \sim \pi\right\} \tag{3}$$

which is assumed to exist for any policy π and to be independent of the initial state. This is the ergodicity assumption in MDPs [Sutton and Barto, 2018].

In the discounted setting, in order to decide which action in one state is better to take, we need to concentrate on the discounted expected reward. But the return in the average reward setting is defined differently. The return shows how much more reward the agent will receive from the current state and action compared to the average reward of the selected policy. The return in this setting is defined as the following form which is the differential return.

$$G_t \doteq R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + R_{t+3} - r(\pi) + \cdots . \tag{4}$$

The methods can be divided based on their policy settings which are on-policy and off-policy. Some important model-free algorithms like discounted Q-learning are considered off-policy, in which agents try to learn the target policy by following another policy which is the behavior policy.

One-step off-policy control methods are the ones that are mostly related to this research work. The differential Q-learning algorithm is an off-policy learning control algorithm without a reference function for the average reward setting. Also, it converges for general MDPs based on theorem 1 of Wan et al. [2021]'s paper.

Algorithm 1: Differential Q-learning (one-step off-policy control)

```
  Input: The policy b to be used (e.g., $\epsilon$-greedy)
  Algorithm parameters: step-size parameters $\alpha$, $\eta$
1 Initialize $Q(s,a)$ $\forall s, a$ and $\bar{R}$ arbitrary (e.g.,to zero)
2 Obtain initial $S$
3 while still time to train do
4     $A \leftarrow$ action given by $b$ for $S$
5     Take action $A$, observe $R$, $S'$
6     $\delta = R$ - $\bar{R}$ + $\max_a Q$(S',a) - Q(S,A)
7     $Q(S,A) = Q(S,A) + \alpha\delta$
      $\bar{R}$ = $\bar{R}$ + $\alpha\eta\delta$
8     $S = S'$
9 Return $Q$
```

The action-values update in differential Q-learning is given in equation 5:

$$Q_{t+1}(S_t, A_t) \doteq Q_t(S_t, A_t) + \alpha_t \delta_t \tag{5}$$

In this equation, α is the method's step size and δ_t is temporal difference (TD) error which is shown in equation 6.

$$\delta_t \doteq R_{t+1} - \bar{R}_t + \max_a Q_t(S_{t+1}, a) - Q_t(S_t, A_t) \tag{6}$$

The formula for reward rate update is given in equation 7. Here η is a positive constant which is also one important hyper parameter alongside the step size.

$$\bar{R}_{t+1} \doteq \bar{R}_t + \eta\alpha_t\delta_t \tag{7}$$

Based on theorem 1 in Wan et al. [2021], R_t converges to best reward rate r^* and Q_t converges to solution of q in bellman equation as equation 8 demonstrates.

$$q(s,a) = \sum_{s',r} p(s', r \mid s, a)\Big(r - \bar{r} + \max_{a'} q(s', a')\Big) \tag{8}$$

Differential Q-Learning can also be extended to linear function approximation setting [Wan et al., 2021]. In this setting at each timestep t, the agent observes the current feature vector x_t, takes an action A_t, observes new a feature vector x'_t, and receives reward R_{t+1}. Here, we try to approximate the action-value function using the linear function of the feature vector as equation 9. Here, $\hat{q}_t$ is an approximated action-value function and w_{A_t} is a weight matrix.

$$\hat{q}_t \doteq \mathbf{w}_{a_t}^{\mathrm{T}} x_t \tag{9}$$

Here Algorithm 2 , is the straight forward extension of Algorithm 1.

Algorithm 2: Differential Q-learning with linear function approximation

```
Algorithm parameters: step-size parameters $\alpha$, $\eta$
1  Initialize $w_a \in \mathbb{R}^d \; \forall a$ and $\bar{R}$ arbitrary (e.g.,to zero)
2  Obtain initial observation vector $x$
3  for each timestep do
4      $A \leftarrow$ action given by $b$ for $S$
5      Take action $A$ (using,say, an $\epsilon$-greedy policy w.r.t. $\hat{q}$),obtain$R$,$x'$.
6      $\delta = R$ - $\bar{R}$ + $\max_a \mathbf{w}_a^{\mathrm{T}} x' - \mathbf{w}_A^{\mathrm{T}} x$
7      $w_A \leftarrow w_A + \alpha\delta x$
8      $\bar{R} \leftarrow \bar{R}$ + $\alpha\eta\delta$
9      $x = x'$
10 end
```

3 Experimental Analysis

We present a variety of empirical results for differential Q-learning in the linear function approximation setting. Our goal is to study how average reward and discounting interact with different kinds of continual tasks. To this end, we chose two classic control problems. Catch and Pendulum. The main difference between these tasks is the effect of current action on future rewards. In Catch, the reward at each step can be attributed at most to the past ten actions. But actions in Pendulum leave a lasting effect and can change the entire trajectory.

We first consider Catch and study long-term performance. We make comparisons with the discounted setting and study sensitivity to hyperparameters. We then investigate different kinds of linear features. We then look at Pendulum and study differential Q-learning with state aggregation and tile coding while making comparisons with discounted Q-learning. Also, here we investigate the reward rate changes over time, and we do sensitivity analysis for hyperparameters.

3.1 Empirical Results for Catch

In this section, we present empirical results for the Catch control problem. This task involves a 10 by 5 grid world with a moving paddle at the bottom trying to catch balls that fall across the screen. At each time step, the agent can move this paddle one unit to the left or right or leave its position unchanged. Any balls caught lead to a +1 reward and any missed ball has a reward of -1, all other states receive a reward of 0. Observations from the environment are a 10 by 5 binary matrix where each one in the first nine rows represents a ball and the one in the last row represents the paddle. We use flattened observation as linear features.

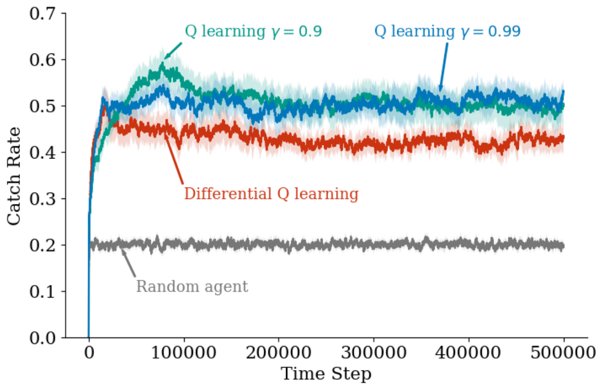


Figure 2: Long-term performance on Catch

Our first experiment investigates the long-term performance of differential Q-learning in comparison with the discounted setting. We report the catch rate, which is an exponentially weighted moving average of the ratio of balls successfully caught. Figure 2 shows the catch rate of differential Q-learning, two discounted Q-learning with 0.9 and 0.99 discount factors, and a baseline random agent. We run each agent 30 times, the solid lines show mean performance and shaded regions are %95 student's t confidence intervals. We see a similar leading performance in the two discounted agents and an inferior performance in differential Q-learning. It is notable that this difference is only apparent after fifty thousand steps. Note that the best-achieved performance is less than %60 catch rate, which is not high, considering the difficulty of the task.

Considering the nature of Catch, longer reward horizons (larger γ), do not lead to better performance because current action does not affect future rewards beyond 10 steps. To tune the aforementioned agents, we searched for the best average performance over a specified range of hyperparameters for the first 50 thousand time steps. Next up, we discuss the sensitivity of differential and discounted Q-learning to step size and discount factor.

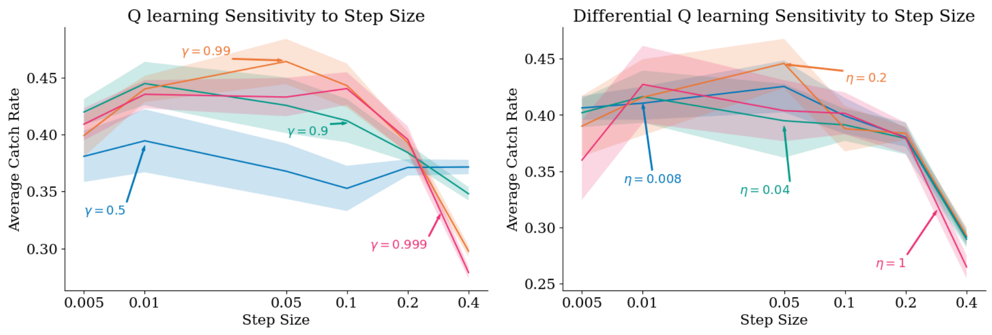


Figure 3: Sensitivity to step size. Left plot: Q-learning. Right plot: Differential Q-learning.

The left plot in Figure 3 shows the average catch rate of Q-learning over 50 thousand time steps for different discount factors and step sizes. Each parameter combination is repeated for 30 runs and error bars are student's t %95 confidence intervals. We can see similar sensitivity for values of γ larger than 0.9. It is also apparent that $\gamma = 0.5$ is too small a reward horizon to achieve comparable performance. The agents presented in Figure 2 as representatives for discounted Q-learning, use the best achieved average performance according to this plot for $\gamma = 0.9$ and $\gamma = 0.99$.

The right plot in Figure 3 illustrates the sensitivity of differential Q-learning. We see little sensitivity in smaller step sizes and more sensitivity in larger step sizes. Also, all η values behave similarly which shows insensitivity to η. Peak performance is achieved for $\eta = 0.2$ and $\alpha = 0.05$ which we use in performance comparison (Fig 2). One potential reason for low sensitivity to η can be how the reward is designed in Catch. In most time steps there is no ball to catch so the agent gets 0 rewards over and over. This can cause the TD between consecutive steps to be small.

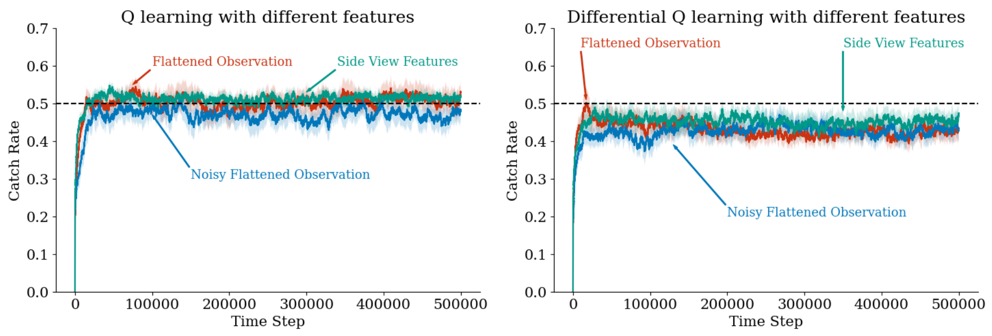


Figure 4: Effect Noise and minimal features. Left plot: Q-learning ($\gamma = 0.99$, $\alpha = 0.05$). Right plot: Differential Q-learning ($\eta = 0.2$, $\alpha = 0.05$).

The next experiment investigates different linear features. We use three features. First, the flattened features that have been used until now, as a baseline and with which the agents are tuned. Second, a noisy feature is created by adding 5 random bits to each flattened feature vector. Finally, we created a feature with limited knowledge of the observation. We call this feature, the Side View feature. This is created by collapsing the first 9 rows of the observation into a single vector of size 5 and combining this vector with the last row of the observation which contains information about the location of the paddle. The collapsed vector is populated by 1s if the corresponding column in the observation contains any balls and 0s otherwise.

Figure 4 shows the superior performance of discounted Q-learning over differential Q-learning. Adding noise to the features seems to degrade the performance of discounted Q-learning. We can see the same degradation in differential Q-learning, but the agent recovers in the long term and reaches the same level as the noise-free performance. This result is evidence in support of the robustness of differential Q-learning to noise. Regarding the Side View features, the results are surprising. We are hiding all the temporal information of observations from our agent but the long-term performance is no worse than using full observation data. Since this pattern is present in both discounted and average reward settings, and because neither setting reaches a very high level of performance, the problem may be due to poor representation learning. This can be hinting toward the need for better function approximators for this task such as neural networks.

3.2 Empirical Results for Pendulum

In this classic control task, the goal is to hold the end of the pendulum upright and apply different amounts of torque τ at that position so it can maintain its balance position for all time. One important feature of this continuing task is having continuous state and action spaces. Therefore, using function approximation techniques like using tile coding or neural networks can be beneficial for this problem. In this paper, we intend to solve this problem by using the state aggregation and tile coding techniques since they are two primary methods among function approximation techniques. First, we utilize the idea of state aggregation and action space discretization in order to solve the pendulum problem. Then, we investigate the idea of tile coding and linear function approximation for solving this problem.

Since the environment is constructed based on OpenAI Gym brockman2016openai, the problem has some specific features. We have a continuous 3-dimensional state space including the pendulum coordinates system which are:

1. Cartesian coordinates of the pendulum's endpoint in meters in the range of -1 and $+1$
2. Angular Velocity in the range of -8 and $+8$

The action is defined as the torque applied to the end of the pendulum which is in the range of -2 and $+2$. The reward function is also defined as the equation below:

$$r = -(\theta^2 + 0.1\,\dot{\theta}^2 + 0.001\,\tau^2) \tag{10}$$

θ is the angle calculated in radians for the endpoint of the pendulum. As a result, the agent can get the maximum reward of 0 when it is located in an upright position and a minimum reward of -16.27.

As mentioned, we want to make a comparison between the discounted Q-learning algorithm and the differential version of it to observe the difference between their performance. The performance of these algorithms is measured based on the total rewards that agents get during the specified number of time steps. In this problem when using state aggregation idea, since we have an infinite number of states we need to do state aggregation for creating a finite set of states. There are a large number of states $(21 * 21 * 65)$ and 17 actions according to the state aggregation and action set discretization setting. Since we want to make sure that our agent explores sufficiently and effectively, we set a maximum time step number to 10 million and also use the epsilon decay method to make sure that the agent explores massively in the beginning phase of learning and then continues with exploitation afterward. Each of our methods is executed 20 times during 10 million steps, and the results related to total return and reward rate, are based on the average performance of these 20 runs. Shaded regions are 99 percent confidence intervals according to t-distribution. Moreover, the sensitivity of the reward rate's step size to TD update step size in differential Q-learning is investigated for 5 runs. In addition, we also analyze the effect of different discretization settings in state space for discounted q-learning based on different step sizes.

When using tile coding, we set the number of tiles to 4 and tilings to 16, since we want to have large tiles and large number of tilings.In this combination of tile and tilings, we can get both acceptable

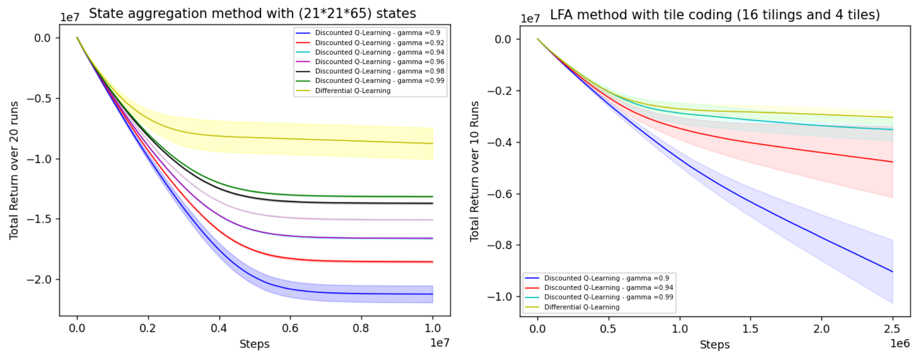


Figure 5: Differential Q-learning and Discounted Q-learning comparisons based on different discount factors values - The plot on the right is for state aggregation method and the one on the left is for the linear function approximation using tile coding

discrimination and generalization. We have 17 actions exactly like state aggregation setting. In this setting, we use linear function approximation method for getting approximated action values. Here, we use same exploration method just like previous setting and the maximum time step is set to 2.5 million. Also, each of our methods is executed 10 times during this period of time. Here, shaded regions represent 80 percent confidence intervals according to t-distribution. Similiar to state aggregation setting, here the average reward's step size to TD update step size in differential Q-learning is investigated for 5 runs. In the rest of this section, we analyze the results of the pendulum problem for state aggregation and tile coding method. We use total reward as our main performance comparison metric.

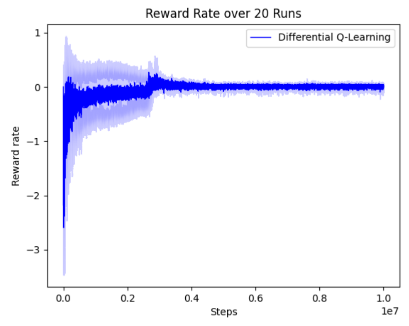
When comparing differential Q-learning and discounted version in pendulum problem, since in pendulum control task agent cares about receiving small negative rewards in the future, agent's horizon should be infinite. Therefore, we can expect that in discounted Q-learning, large discount factors should give us a better performance. Also, we expect that differential Q-learning performs better in this setting, since the problem is continuing and average reward methods are suitable for continuing task.

Figure 5 demonstrates the comparison between the total return of discounted Q-learning agents with different gamma values and one differential Q-learning agent for both state aggregation and tile coding method. All of the chart are plotted according to their best hyperparameters.As the chart on the left which is related to state aggregation implies, all the agents learn the true q-values in this period. After some specific amount of steps, the slope of the lines will gradually decrease, which means the rewards of each step are less negative and the pendulum endpoint stays in a stable upright position. The chart is plotted for a 99 percent confidence interval based on the student's t-distribution. Also in the chart on a right which is for tile coding method, we can see the same patterns. The chart for this setting is plotted for a 80 percent confidence interval for student's t-distribution.

What we understand from the charts in Figure 5 is that the differential Q-learning agent has a higher total reward than other discounted agents. One important observation is that when the gamma values increase, the discounted Q-learning agents' performances are getting closer to differential Q-learning.

It is also interesting to see how the reward rate is changed during the learning process, since if our agent learns optimal policy after specific number of time steps, it can apply actions in a way that pendulum's end point holds its upright position for the rest of the time and can get zero reward. Therefore, we expect that our agent's average reward converges to zero after specific number of time steps.

Figure 6 shows that in differential Q-learning, there is a fluctuation in reward rate at the beginning of the learning phase and after some specified time steps, the agent learns to apply torques in a way to hold the pendulum upright. In this phase of training most of the time it gets zero rewards. Therefore, our reward rate converges to zero with a 95 percent confidence interval along with the student's t-distribution.



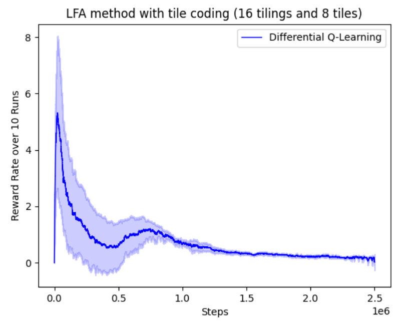
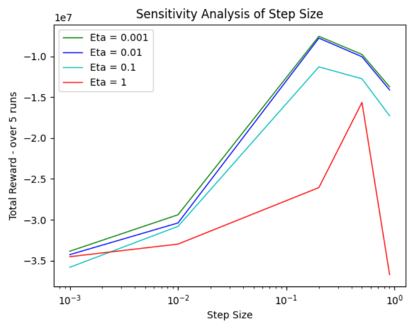


Figure 6: Differential Q-learning reward rate value. The plot on the left is for state aggregation method and the one on the right is for the linear function approximation using tile coding



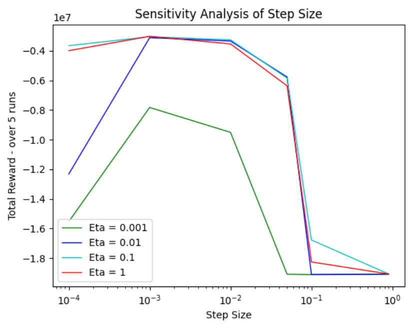


Figure 7: Sensitivity to step size based on total reward. The plot on the left is for state aggregation method and the one on the right is for the linear function approximation using tile coding

In Figure 7, we can see the sensitivity analysis plots for both state aggregation and tile coding method. Figure 7 demonstrates the sensitivity of reward rate step size (η) to differential Q-learning step size, based on total return . The best combination of our agent's hyperparameters is step size=0.2 and $\eta = 0.01$ for 5 runs and 5 million time steps for state aggregation method. Also, the best hyperparameters for linear function approximation with tile coding is step size=0.001 and $\eta = 0.1$ for 5 runs and 2.5 million time steps.

4 Discussion

In this paper, we studied differential Q-learning with linear function approximation on two contrasting continuing tasks, Catch and Pendulum. We provided empirical results for differential Q-learning and made comparisons with the discounted setting. The results for Pendulum show a clear lead for differential Q-learning, but the story is different in Catch. All of the experiments on Catch indicated better performance for discounted agents.

On the Catch task, discounted Q-learning outperformed the average reward algorithm. Increasing the discount factor beyond 0.9 did not affect the long-term performance. This outcome might be due to the limited window of future rewards affected by action in Catch. Sensitivity analysis shows that differential Q-learning is less sensitive to its hyperparameters η and α which can be due to sparse rewards in Catch. This claim requires further investigation to confirm. Experimenting with noisy features indicates that differential Q-learning might be more robust to noise. Finally, both algorithms can achieve comparable performance using a minimal feature vector. The previous observation might be due to insufficient representation learning and show the need for neural networks to achieve better performance for both algorithms.

In pendulum task, we investigated two primary methods of function approximations which are state aggregation and tile coding with linear function approximation. After finding the best setting for hyperparameters, the results indicate that differential Q-learning method outperforms the discounted Q-learning method based on total return metric. Also, we can see that in discounted Q-learning method, agents with higher discount factor value achieve better performance in the long run. The results can confirm the effectiveness of this novel average reward method in this continuing task. We can point out the nature of this continuing task as a primary reason for the effectiveness of differential Q-learning in this problem.

We believe that since the nature of these continuing tasks are different from each other, which is related to how much agents should care about receiving future rewards(horizon), can be the main reason for getting different performances for differential Q-Learning in both of these tasks.

5 Next Steps

Experiments on Catch reveal the need for further empirical investigation into this task. A better understanding of the inner workings of TD-based algorithms on this task can help us understand the shortcomings of differential Q-learning and the average reward formulation in general. One area of investigation is the fluctuations of TD error and how that explains the low sensitivity of differential Q-learning to η.

By expanding the scope of this study to other control tasks, we can gain a more complete picture of differential Q-learning. The goals of this study will be to (1) use empirical evidence to effectively recommend algorithms for new problems and (2) gain a better understanding of average reward and its limitations on problems with sparse rewards and limited reward horizons (like Catch).

References

Tapas K. Das, Abhijit Gosavi, Sridhar Mahadevan, and Nicholas Marchalleck. Solving semi-markov decision problems using average reward reinforcement learning. *Management Science*, 1999.

Vektor Dewanto, George Dunn, Ali Eshragh, Marcus R. Gallagher, and Fred Roost. Average-reward model-free reinforcement learning: a systematic review and literature mapping. *ArXiv*, abs/2010.08920, 2020.

Abhishek Naik, Roshan Shariff, Niko Yasui, Hengshuai Yao, and Richard S. Sutton. Discounted reinforcement learning is not an optimization problem, 2019.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, second edition, 2018.

Yi Wan, Abhishek Naik, and Richard S Sutton. Learning and planning in average-reward markov decision processes. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10653–10662. PMLR, 18–24 Jul 2021.