CMPUT 366 Reading-Writing Exercise 5

Name: Minghan Li
Student ID: 1561234
CCID: minghan4

Chapter 6 introduces a central and significant learning method in Reinforcement Learning– *Temporal Difference* Learning. This method combines the advantages of Monte-Carlo learning and Dynamic Programming, and achieves great performance on both prediction and control tasks. The idea of learning one estimate from the next estimate may sound unsafe, but it actually helps us to approach the true state value faster and closer. Moreover, temporal difference learning also occurs in many applications such as board game, robotics and etc.

For prediction tasks, temporal difference learning has significant advantages over Monte-Carlo learning and dynamic programming in terms of estimating the true state value:

(1) Compared to DP method, TD learning doesn't require the model of environment's dynamic and only do updates based on the raw experience.

(2) Compared to MC method, TD learing doesn't need to wait until the end of the episode to update the value function, but perform learning just after one time step. This not only accelerates the learning process but also allows us to derive an on-line learinig algorithm.

For control tasks the algorithms are divided into two categories: on-policy learning and off-policy learning. For on-policy learning, *SARSA* is able to use the actual stream of experience to do temporal difference learning, while for off-policy learning, *Q-learning* use the biggest q-value of the next state to update the value function. Both algorithms are able to find the optimal policy, but SARSA finds the one for the behavior policy (which is also the only policy), while *Q-learning* finds the optimal policy for the target policy which is the greedy policy.

Although Q-learning achieves great performance on control tasks, there are still some issues with it, resulting in *maximization bias*, which means Q-learning often over-estimates the state-action value. This can be solved by *Double Q-learning*, which has two Q-value function and by alternatively updating them, the agent can update its estimate with less bias.

There is also another algorithm which naturally avoids the maximization bias problem, that is the *Expected Sarsa*. Unlike either Sarsa or Q-learning, it use the expected q-value of the next state over all actions to update the q-value function rather than a single one, which looks very similar to dynamic programming, and it tends to give better performance and more robust to the step size parameter.

For now we can see that TD learning plays a key role in both prediction and control tasks. But updating one estimate from the next estimate sounds unsafe to us, which gets us to concern whether it might not work in some tasks. Fortunately, it has already been proven that TD converges to the *certainty-equivalence* estimate of the underlying MDP, and it has better and faster empirical convergence results over algorithms like Monte-Carlo learning.

At the end, the authors also introduce another relatively unconventional concept of states called *afterstate*. This concept is applied in many applications such as board games, where we have complete information of the environment. Also, learning afterstate value function can greatly speed up the learning process if the state space is too large.