CMPUT 366 Reading-Writing Exercise 6

Name: Minghan Li
Student ID: 1561234
CCID: minghan4

Chapter 7 introduce a class of algorithms that spans a spectrum between TD and Monte-Carlo method, which is *n-step TD method*. The agent using n-step TD won't have to wait until the end of the episode to do updates as in Monte-Carlo learning, as well as concur the tyranny of time steps which limits the horizon of the agent in one-step TD learning.

And once again the contents in this chapter can be divided into two familiar topics: prediction and control. The following concepts and algorithms will look similar to the ones in previous chapter but more general and can be easily cooperated into different settings.

For prediction tasks, we can use *n-step TD* learning algorithms, which will look at the next n-step rewards and bootstraps from the value of the state n step later as a proxy for the remaining rewards. Empirically, the intermediate value of n works best, but it also depends on the specific task.

For control tasks, the algorithms again fall into two categories: on-policy and off-policy methods. In on-policy control case, we use *n-step SARSA*, which execute the policy by n steps and then update the state value n steps before. In this case the effective reward signals can be propagated to states in earlier time-steps and accelerates the learning process.

Similarly, algorithms for n-step off-policy learning will be similar to off-policy one-step TD learning. One important technique we use in off-policy learning is importance sampling, which was mentioned in Chapter 5 Monte Carlo learning. If we view the n-step sum of discounted rewards as a truncated return, the learning will be exactly the same as in Monte-Carlo method with importance sampling. So high variance issue also occurs in off-policy n-step SARSA, which can be resolved by techniques like Per-reward importance sampling. By using importance sampling ratio ρ_t as a balance factor between n-step return and the current state value, we can greatly reduce variance without introduce any bias.

Moreover, another algorithm that doesn't require importance sampling is called *n-step tree-backup* algorithm. It can be viewed as a more general method of Expected SARSA, which will look at every action of every state in n steps as leaf node and using the probability of the action taken under the target-policy as the weight. In this way the agent is not only able to use the expectation of state-action value to do updates but also consider the long-term return more unbiasedly.

In the end, the authors introduce an unifying algorithms of n-step SARSA, n-step Expected SARSA and n-step tree backup algorithms, which is called *n-step Q(σ)*. The algorithm is basically a more general tree backup algorithm with a parameter σ that controls the degree of sampling of every state. By trading off expectation and sampling for every specific state may render us with more powerful learning algorithms.

To sum up, by picking n appropriately, n-step TD learning algorithm can perform better than on-step TD and Monte-Carlo learning in many tasks. Freeing the agent from the tyranny of the time steps can be a huge advantage that allows agent to consider long term return more unbiasedly as well as accelerates the learing process.