CMPUT 366 Reading-Writing Exercise 7

Name: Minghan Li
Student ID: 1561234
CCID: minghan4

In the former chapters we have discussed two main kinds of Reinforcement Learning algorithms. Model based RL such as Dynamic Programming and Model Free RL such as Monte Carlo learning and TD learnig. It seems that both types of algorithms have their merits and Chapter 8 tends to introduce a new method that unifies both search and learning method: The agent is *learning* using raw experience and in the meanwhile it is also *planning* using a simulated model. But unlike planning in DP, which uses a *distribution model*, here we utilize a *sample model* to do planning which is more computational efficient. Moreover, model-free learning is sometimes called *direct RL* because it only uses raw experiences and planning is called *indirect RL* which requires a model of the environment. There's no reason that we cannot combine these two ideas since the objective of both learning and planning methods is the up-dating estimation of value functions by using backing-ups, so integraing these two methods is appropriate and can lead to more powerful algorithms, such as *Dyna*.

Dyna is an algorithm that combines both learning and planning in a way such that the agent can update its value function using raw experience in the outer loop, and utilize a sample models to generate simulate experience to do planning in the inner loop. It is much more efficient than direct RL methods such as TD learning especially when the rewards in the environment are sparse and doesn't require a exact model of the environment as DP.

However, this kind of algorithm can have poor performance when the model is wrong, in other words if the environment is changing, it might take a long time for the agent to adjust its model, but it also gains us insights that sometimes suboptimal policies can lead to the quick discovery of the changes in the environment and correction of the modeling error. Algorithms such as *Dyna* Q^{+} introduces extra exploration bonus for the states it hasn't visited for a long time. In nature, this problem occurred in Dyna is still the good old exploitation and exploration problem. Luckily we've already learned some solutions to deal with this issue.

The chapter also introduces a *search control* method called *Proritized Sweeping* to speed up the planning when the state space is large. By utilizing a priority queue, we can update the state value that have dramatic changes in the learning process and temporarily ignore the states which have small changes in their estimate. This method is much faster and guranteed to converge to the optimal policy if we eventually sweep through the whole state space.

At the end of the chapter the authors discuss the relative advantages of expected and sample updates and several other methods that combine learning and planning together. The expected updates will generally lead to better results than that of sample updates but less efficient in computational time. So in the problem of large state–action space, we may prefer sample updates at many state–action pairs than with expected updates at a few pairs. Several other methods are also discussed such as *Trajectory Sampling*, which samples the experience according to on-policy distribution, avoiding exhaustive sweep through the whole state space; *Real Time Dynamic Programming* is an on-policy trajectory-sampling version of DP's value- iteration algorithm, which is guaranteed to find a optimal policy on the relevant states without visiting the entire state space. Heuristic search and MCTS are also discussed at the end, suggesting that there are various kinds of methods that we can incorporate into direct reinforcement learning and obtain much more powerful algorithms.