Reading – Writing Assignment: Chapter 4

Brian (Boyuan) Lu

Dynamic Programming (DP) refers to blocks of codes that used to compute the optimal model under a MDP environment. One connection with chapter 3 is that after we know that policies for optimal value functions (v* and q*), we can use DP to perform the recursive calculation on our policies. For example, performing DP on Bellman optimality equation will break the Bellman equation into three assignments that can update rules in current iteration.

Firstly, DP for state-value function is called policy evaluation. The purpose is to iteratively go through v0, v1, v2 for each mapping S (S contains real numbers). And then estimate each mapping, find the optimal S. The details of this operation are performing expected update. It produces estimate of next possible state based on previous successor state and values, and then estimate the immediate reward which is a one-step transaction under the evaluation policy. The codes on this operation has two approaches. One uses two arrays which one array store the old values, another one stores the old values. Another approach consists with a single in-place array, once a new value comes out, updated the old value in the array. In general, the in-place algorithm has a faster convergence and it was usually used in DP algorithms.

Policy improvement basically computing the value function of each policy and help to find better policies. The evaluation on the policy is based on the action-value functions. Using policy improvement theorem. Compare π, and π', if the action-value function of π' is greater than the state-value function of π, then π' must be a as good as or even better policy than π. So usually, this can be approached using a greedy algorithm that seeking for the π' of all possible actions and selecting at each state the action that appears best according to action-value functions.

Policy iteration is performed once we decided π' is better policy and then compute this policy further to yield any possible better policy. The way we use iteration to find the optimal policy is called policy iteration. Policy iteration usually converges very fast, probability in few iterations. Another iteration is called Value Iteration, it can be represented as a simple update operation that combines policy improvement and truncated policy evaluation. Value iteration terminates when value function changes by only a small amount of sweep. It evaluations the effect of sweep on policy improvement and policy evaluation.

DPs is a very efficient strategy to approach the MDPs. To find the optimal policy in polynomial number of states and actions. Even the number of polices are very large, say k^n, DP can guarantee to find optimal in polynomial run time. However, DP is sometimes considered to be limited because the curse of dimensionality, which means the number of policies increases exponentially according to the number of state variables. But this only exists in theoretical level since today's DP can handle millions of states. In conclusion, DP is a very important algorithm method for solving finite MDPs. By performing policy evaluation and policy improvement, we can do value iteration and policy iteration to find the optimal policy. And DP is the best approach in this logic.