Dynamic Programming

Optimal Substructure

The Optimal Solution to **all subproblems** produces the optimal solution to the original problem. In other words, if you find the **optimal policy for all states** in a problem, you find the optimal policy for **the whole problem**.

Thereforce, DP turns Bellman equations into Update Rules:

$$v_*(s) = \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_*(s')] \tag{1}$$

becomes

$$v(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v(s')]. \tag{2}$$

You need to know in advance how the state changes and what rewards we get from performing each action in each state: $p(s', r \mid s, a)$.

Value Iteration

Update v iteratively to get better approximation of optimal value of **every action** at every iteration:

$$v(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v(s')] \tag{3}$$

where $v(s')$ an old value of this current value table for the next state.

Throughout the value iterations, the policy is always constant (usually random uniform), and for every iteration, you evaluate the value for **every** action. Only at the end of all the iterations, you devise a policy, π, by using the state values $V(s)$ and the models that gives transition probabilites $p(s', r' \mid s, a)$:

$$\pi(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v(s')]. \tag{4}$$

Policy Iteration

While value iteration iterates through all actions evenly, policy iteration iterates values and then update the policy (Policy Improvement) to carry out another "value iteration" using the newly set policy (Policy Evaluation):

$$v(s) \leftarrow \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)[r + \gamma v(s')] \tag{5}$$

where in every evaluation, you evaluate for only **one** action that is chosen by your current policy π, and thus the $\sum_a \pi(a \mid s)$ expression instead of $\max_a$.

Or you can even get rid of state value table v and just track for the state-action valute table q since you cannot use v in model-free environments to derive the policy:

$$q(s,a) \leftarrow \sum_{s',r} p(r,s' \mid s,a)[r + \gamma \sum_{a'} \pi(a' \mid s') q_\pi(s',a')] \tag{6}$$

where $p(s,a)$ is implicitly captured by the q table throughout the iterations.

Thus, when deriving the policy at the end of all the iterations, you don't need the model p that gives transition probabilites, because you already have the policy π that was updated throughout the learning process:

$$\pi(s) = \arg\max_a Q(s,a). \tag{7}$$

Policy Improvement Theorem

If the new value of the state after choosing action with new policy, $q_\pi(s, \pi'(s))$, is greater than the original value, $v_\pi(s)$, then the new state value, $v_{\pi'}(s)$ is greater than original state value, $v_\pi(s)$:

$$q_\pi(s, \pi'(s)) \geq v_\pi(s) \Rightarrow v_{\pi'}(s) \geq v_\pi(s) \tag{8}$$

where $q_\pi(s,a) = \sum_{s',r} p(s', r \mid s, a)[r + \gamma v(s')]$.