Chapter 4 - Dynamic programming (Aravind S - EE14B013 - RL project RA 1)

Policy evaluation (PE)

- Given the MDP and an arbitrary policy π, the corresponding value function can be found being greedy with respect to this policy.
- If it is a stochastic policy, it will be a weighted sum (with weights being the π(a|s)).
- Stochasticity of environment is embedded in the form of transition probabilities p(s', r | s, a).
- Can be implemented in an iterative fashion (derived from Bellman equation)

$$\begin{aligned} v_{k+1}(s) &\doteq \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_k(s')\Big] \end{aligned}$$

- $v_k = v_\pi$ is a fixed point and hence it converges to the same.

Policy improvement (PI)

- Having determined the value function according to a policy π, we still don't know that the given policy π is the optimal policy.
- But using action-value functions we can determine whether we need to make changes to the existing policy, thus making it closer to an optimal policy.
- Suppose there exists another policy π' such that

$$q_\pi(s, \pi'(s)) \geq v_\pi(s).$$ for all states s.

- In that case policy π' is better than π - because there is an alternate series of actions which can be taken and hence obtain a greater expected return from state 's'.

$$\begin{aligned} \pi'(s) &\doteq \arg\max_a q_\pi(s,a) \\ &= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \arg\max_a \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big], \end{aligned}$$

Policy iteration

- Now given a policy π we can evaluate the value function. And given a value function, we can generate a policy π' which is equal/better than π.
- These steps can be carried out successively to obtain better policies.
- To ensure convergence, argmax-es should return actions such that successive estimates of π's are similar.
- Since MDP is finite, there are finite set of policies and hence this converges to one of the optimal policies.
- Stochastic optimal policies can be obtained by allotting the probability among optimal actions (in any way)

Value iteration

- The PE & PI can be combined and a simple iterative algorithm can be derived as

$$\begin{aligned} v_{k+1}(s) &\doteq \max_a \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_k(s')\Big], \end{aligned}$$

- Having a closer look, this is a small modification of Bellman optimality equation.
- Converges to the optimal value function. Can be stopped when difference between successive estimates are sufficiently small.
- After convergence, policy can be estimated using a one-step look ahead using

$$\pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$$

Dynamic programming (DP)

- All the above approaches are instances of DP where we solve subproblems before solving actual ones.
- They do full backups (considers all possible successor states from the current state) and bootstrapping (update estimate of one state based on estimates of other - requires complete knowledge of MDP)
- For a large state space, DP approaches take a long time (making it unsolvable). Following variants are proposed:
  - Async DP: in-place algo; Faster convergence;
  - Realtime DP: real-time interaction with environment; uses updated policies to learn better;
- Better than exhaustive search ($O(k^n)$). DP takes polynomial time. [n states, k actions]

Generalised policy iteration (GPI)

- From a broader view, the policy iteration algorithm is very generic.
- Evaluation of value functions is done for a current estimate of policy π.
- Improvement of policy is done by being greedy w.r.t to the current estimate of the value function V.