Note for Reinforcement Learning 2nd Edition

August 15, 2018

1 Finite Markov Decision Process

MDP framework consists of an environment and agent. For $t = 0, 1, 2...t$, agent receives observed state $S_t \in \mathcal{S}$ based on which the agent perform an action $A_t \in \mathcal{A}$. The dynamic of the environemtn then returns a reward $R_{t+1} \in \mathcal{R}$. This forms a trajectory $S_0, A_0, R_1, S_1, A_1, R_2, \ldots$. The dynamic for MDP is defined to be

$$p(s', r \mid s, a) = Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

Several commonly use quantities:
state-transition probability

$$p(s' \mid s, a) = \sum_r p(s', r \mid s, a)$$

expected reward for state-action pair

$$\begin{aligned} r(s, a) &= E[R_t \mid S_{t-1} = s, A_{t-1} = a] \\ &= \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a) \end{aligned}$$

expected reward for state-action-next-state triples

$$\begin{aligned} r(s, a, s') &= E[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'] \\ &= \sum_{r \in \mathcal{R}} r p(r \mid s, a, s') \\ &= \sum_{r \in \mathcal{R}} r \frac{p(s', r \mid s, a)}{p(s' \mid s, a, s')} \end{aligned}$$

For episodic tasks, we have nonterminal states $\mathcal{S}$ and terminal states $\mathcal{S}^+$. Time of termination T is a random variable between episodes.

Returns at time step t is

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

All reinforcement learning algorithm involve estimating 3 quantities: value function, state-action function.

A policy is a probability distribution on $\mathcal{A}$ given s denoted as $\pi(a \mid s)$.

A value function of a state s under a policy π is the expected return when starting in s and following π thereafter.

$$\begin{aligned} v_\pi(s) &= E_\pi[G_t \mid S_t = s] \\ &= E_\pi\left[\sum_{i=0}^{\infty} \gamma^i R_{t+i+1} \middle| S_t = s\right], \quad \text{(for all } s \in \mathcal{S}) \\ &= E_\pi\left[R_{t+1} + \gamma \sum_{i=0}^{\infty} \gamma^i R_{(t+1)+i+1} \middle| S_t = s\right] \\ &= E_\pi[R_{t+1} \mid S_t = s] + \gamma E_\pi[G_{t+1} \mid S_t = s] \\ &= \sum_{a,r,s'} r p(s', r \mid s, a)\pi(a \mid s), \quad \text{(Law of Total Expectation)} \\ &\quad + \gamma \sum_{a,s'} E_\pi[G_{t+1} \mid S_{t+1} = s']p(s' \mid s, a)\pi(a \mid s) \\ &= \sum_{a,r,s'} r p(r \mid s, a)\pi(a \mid s) + \gamma \sum_{a,r,s'} v_\pi(s')p(s', r \mid s, a)\pi(a \mid s) \\ &= \sum_a \pi(a \mid s) \sum_{r,s'} p(s', r \mid s, a)\left[r + \gamma v_\pi(s')\right] \end{aligned}$$

The state-action function is the expected return of taking an action a at state s before following the policy thereafter.

$$\begin{aligned} q_\pi(s, a) &= E_\pi[G_t \mid S_t = s, A_t = a] \\ &= E_\pi\left[\sum_{i=0}^{\infty} \gamma^i R_{t+i+1} \middle| S_t = s, A_t = a\right] \\ &= E_\pi[R_{t+1} \mid S_t = s, A_t = a] + \gamma E_\pi[G_{t+1} \mid S_t = s, A_t = a] \\ &= \sum_{s',r} r p(s', r \mid s, a) + \gamma \sum_{s',a'} E_\pi[G_{t+1} \mid s', a']p(s', a' \mid s, a) \\ &= \sum_{s',r} r p(s', r \mid s, a) + \gamma \sum_{s',a'} q(s', a')p(a' \mid s', s, a)p(s' \mid s, a) \\ &= \sum_{s',r} r p(s', r \mid s, a) + \gamma \sum_{s',a',r} q(s', a')\pi(a' \mid s')p(s', r \mid s, a) \\ &= \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma \sum_{a'} q(s', a')\pi(a' \mid s')\right] \quad (*) \end{aligned}$$

More identities:

$$\begin{aligned}
q_\pi(s,a) &= E_\pi[G_t \mid S_t = s, A_t = a] \\
&= E_\pi[R_{t+1} + \gamma G_{t+1} \mid s, a] \\
&= E_\pi[R_{t+1} \mid s, a] + \gamma \sum_{s'} E_\pi[G_{t+1} \mid s'] p(s' \mid s, a) \\
&= \sum_{s',r} r p(s', r \mid s, a) + \gamma \sum_{s'} v_\pi(s') \sum_r p(s', r \mid s, a) \\
&= \sum_{s',r} p(s', r \mid s, a)(r + \gamma v_\pi(s')) \\
&= E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid s, a]
\end{aligned}$$

Optimal policy and value fucntions

A policty $\pi' \geq \pi$ iff $v'_\pi(s) \geq v_\pi(s), \forall s \in \mathcal{S}$. Optimal state value function is defined as

$$v_*(s) = \max_\pi v_\pi(s), \forall s \in \mathcal{S}$$

Similarly, optimal state action value function is defined as

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

Bellman optimality equation for v_*

$$\begin{aligned}
v_*(s) &= \max_a q_{\pi_*}(s,a) = \max_a E_\pi[G_t \mid S_t = s, A_t = a] \\
&= \max_a E_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\
&= \max_a [E_{\pi_*}[R_{t+1} \mid s, a] \\
&+ \gamma \sum_{s',r} E_{\pi_*}[G_{t+1} \mid S_{t+1} = s'] p(s', r \mid s, a)] \\
&= \max_a \left[\sum_{s',r} r p(s', r \mid s, a) + \gamma \sum_{s',r} v_*(s') p(s', r \mid s, a) \right] \\
&= \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_*(s')]
\end{aligned}$$

Bellman optimality equation for q_* $q_*(s,a)$ is the value of taking action a at state s, then follow optimal policy after. Hence we can also write.

$$\begin{aligned}
q_*(s,a) &= \max_\pi q_\pi(s,a) \\
&= \max_\pi E_\pi[R_{t+1} + \gamma G_{t+1} \mid s, a] \\
&= \max_\pi (E_\pi[R_{t+1} \mid s, a] + \gamma E_\pi[G_{t+1} \mid s, a]) \\
&= \max_\pi \left(E_\pi[R_{t+1} \mid s, a] + \gamma \sum_{s'} E_\pi[G_{t+1} \mid s'] p(s' \mid s, a) \right) \\
&= \left(E_\pi[R_{t+1} \mid s, a] + \gamma \max_\pi \sum_{s'} v_\pi(s') p(s' \mid s, a) \right) \\
&= \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma \max_\pi v_\pi(s') p(s', r \mid s, a) \right] \\
&= \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_*(s') p(s', r \mid s, a) \right] \\
&= \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma \max_{a'} q_*(s', a') p(s', r \mid s, a) \right]
\end{aligned}$$

Once we have $v_*(s)$, optimal policy is just a one step search, it selects the action that maximize the sum of immediate reward and the value of the next state i.e. $\max\{R_{t+1} + \gamma v_*(S_{t+1})\}$. If we have $q_*(s,a)$ instead, then for any s, the optimal policy will be just the action that maximize $q_*(s,a)$.

Solving bellman equation is not pratical since we all satisfy the following 3 conditions:.

1. Know dynamics of the environemtn (Poker provides partially observed state only)
2. Have enough computation resource (Go has huge state space exceeed the number of atoms in the universe)
3. Problem needs to have markov property (Weather predition cannot be based on yesterday only)

Therefore we typically find approximations to the value functions.

Solving Bellman Equation with DP

Policy Evaluation

Recall that for any policy π,

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{r,s'} p(s', r \mid s, a) [r + \gamma v_\pi(s')]$$

If the environment is known, we can solve it as a set of linear equations. If not, we can approximate v_π given π by initializing v_0 and following the iteration: (Note that unique existence and convergence is guaranteed as long as $\gamma < 1$ (e.g. $||v_{k+1} - v_k||_\infty \to 0$).)

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

Since we are averaging the next states, this is called expected update. There are two ways to update algorithm, "in-place update" where we do a sweep through a single array ($v(s')$ can be the updated value for some s') and "double buffers update" where we keep two arrays, update the back buffer array using the front buffer array and swap the reference. Both will converge. We usually prefer the former.

Policy Improvement

The policy improvement theorem says that if π, π' are two deterministic policies such that

$$q_\pi(s, \pi'(s)) \geq v_\pi(s)$$

for all $s \in \mathcal{S}$. Then π' is better or as good as π, e.g

$$v_{\pi'}(s) \geq v_\pi(s), \quad \forall s \in \mathcal{S}$$

(TODO: Derive this).

Given a policy π, we can compute v_π. And from v_π, we can find a better policy π' with a greedy one step lookahead search.

$$\begin{aligned}\pi'(s) &= \operatorname*{argmax}_a q_\pi(s,a) \\ &= \operatorname*{argmax}_a E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid s,a] \\ &= \operatorname*{argmax}_a p(s', r \mid s,a)[r + \gamma v_\pi(s')]\end{aligned}$$

Policy Iteration

We start with a policy π_0 and alternate between policy evaluation (E) and policy improvement (I) until it converges:

$$\pi_0 \xrightarrow{E} v_0 \xrightarrow{I} \pi_1 \xrightarrow{E} \cdots \xrightarrow{E} v_* \xrightarrow{I} \pi_*$$

Value Iteration

Policy evaluation and improvement can be combined such that one update includes one sweep of policy evaluation and one sweep of policy improvement. We call this value iteration which can be obtained from Bellman's optimality equation directly:

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

We can terminate when the distance between v_k and v_{k+1} is small enough.

Async DP

Notice that in order to computer $v_{k+1}(s)$, we have to computer $v_k(s)$ for all $s \in \mathcal{S}$. If $\mathcal{S}$ is large, this will be a long sweep. Many of the states are not relevant to the optimal policy and they shouldn't be updated as often as the relevant ones. Async DP update states in place using previous value (could be values from much older updates) without sweeping all the states, as long as each states are updated infinitely many times, it will converge.

General Policy Iteration

GPI is the term to describe two interacting processes: Policy evaluation and Policy Improvement. They are competing in the sense that policy improvement invalidates the value function by changing the policy (making it greedy so value function no long correspond to the new policy). They are also cooperating in the sense that policy evaluation updates the value function making it consistent with the current policy.

2 Monte Carlo Methods

Value function estimation

Monte Carlo methods learns from experiences, a set of episodes

$$\{(s_0, a_0, r_1, s_1, a_1, r_2 \ldots, s_N) \mid N \text{ is random variable}\}$$

Unlike the DP method, no knowledge about the transition distribution is required. It still needs a model but the model only needs to generate episodes.

Note that for in any episode, each occurrence of s is called a visit to s. **First-visit MC methods** estimates $v_\pi(s)$ averages the returns of the first visit to s whereas **Every-visit MC methods** averages the return of all visits to s in an episode.

In First-visit MC method, for each episode, we compute $G_0, G_1, \ldots G_N$ and only store G_i in array A_{s_i} if s_i is seen the first time. After looping through all episodes, $v(s) = average(A_s)$. Since each episode is independent, all G values in A_s are independent, by law of large number, their average to converge to the real value.

Average visit MC method's convergence is more complicated since multiple G in A_s can come from the same episode and they are not independent. (TODO: justify its convergence).

Action-value function estimation

If we don't have the dynamics of a model, we cannot do one step search. In this case, we will estimate the action value function. Similarly, a state-action pair (s,a) is visited if state s is visited and action a is taken in it. Both MC methods above apply. The problem here is if the policy is deterministic, a lot of state-action pairs will not be visited at all which we will need to choose from. One way is to use exploring starts which assign non zero probability to all state as starting state. The other way is to only use stochastic policy.

Monte Carlo Control ES(Exploring Starts)

We alternate the process of policy evaluation and value iteration between episodes. For each episode, we randomly choose a start state, and generate the episode sequence. First visit is used to update the action-value function. Every time the running average $q(s,a) = average(A_{s,a})$ is updated after inserting a new G to $A_{s,a}$, we also update the policy by $\pi(S) = \operatorname{argmax}_a q(s,a)$