# Part VI Markov Decision Process

- MDP is theoretic foundation of reinforcement learning
- RL can be viewed as an approximate solution to MDP.
- MDP is a generalization of contextual bandit in which the next step depends on current action & state.

## Definitions

- Agent : player, learner, ...
- Environment : Thing that interacts with agent.
- State $S_t$: Representation of situation at time $t$
- State space $\mathcal{S}$ : collection of possible states
- Action $A_t$ : Action taken by agent at time $t$
- Action space $\mathcal{A}(s)$ : collection of possible actions at state $s \in \mathcal{S}$.
- Reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ reward received at the beginning of $t+1$
- Transition probabilities : $\forall s, s' \in \mathcal{S}$, $a \in \mathcal{A}(s)$, $r \in \mathcal{R}$

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

satisfying $\sum_{s'} \sum_{r} p(s', r \mid s, a) = 1, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$

- Policy $\pi$: How agent acts on the basis of $S_t$. Specifically, $\pi(a|s)$ defines the prob. that agent picks action $a \in A(s)$ given $S_t = s$.

MDP formulates a sequential decision making problem.

MDP and agent's policy together give rise a trajectory

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \cdots$$

Question: What is Markov? Why Markov?

Markov property means that given the present, the past and future are independent.

> How I reach present state doesn't affect my future, the only thing matters is that I'm at this state.

Mathematically, it says

history doesn't matter

$$Pr(S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a, S_1^{t-1}, A_1^{t-1})$$

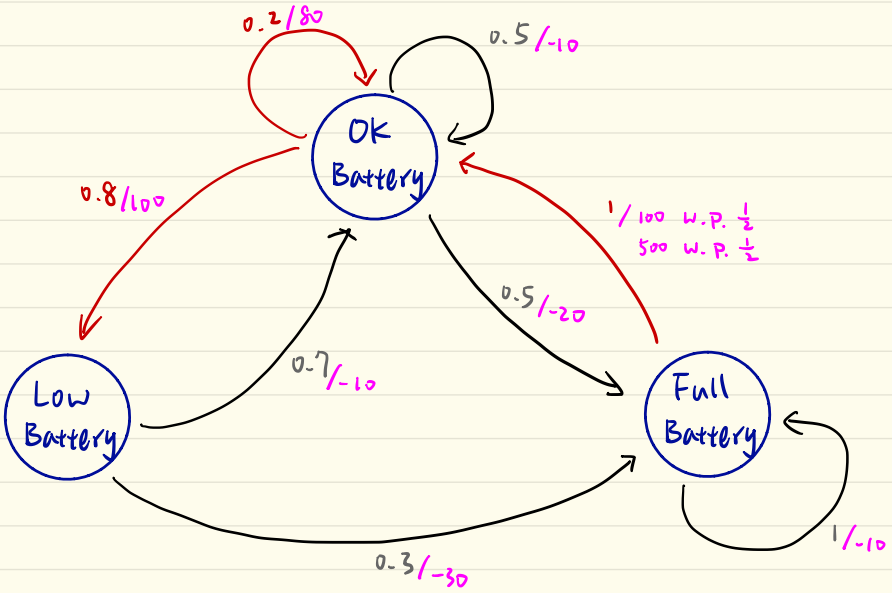
$$= Pr(S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a)$$

Is this practical? Yes if we carefully define states.

Ex: UberEats with GoGoRo

Red: Take order  Grey: Transition probability

Black: Charge  purple: Reward




Three states: $S = \{L, O, F\}$

Three action sets: $A(L) = \{C\}$, $A(O) = \{C, T\}$, $A(F) = \{C, T\}$

Transition probabilities:

$P(O, -10 \mid L, C) = 0.7$
$P(F, -30 \mid L, C) = 0.3$

$P(O, -10 \mid O, C) = 0.5$
$P(F, -20 \mid O, C) = 0.5$
$P(O, 80 \mid O, T) = 0.2$
$P(L, 100 \mid O, T) = 0.8$

$P(F, -10 \mid F, C) = 1$
$P(O, 100 \mid F, T) = \frac{1}{2}$
$P(O, 500 \mid F, T) = \frac{1}{2}$

**Agent's Goal**: The agent's goal in MDP is to maximize not immediate reward but cumulative reward in the long run.

Suppose the trajectory is $S_0, A_0, R_1, S_1, A_1, R_2, \cdots$

the cumulative discounted reward from $t$ onwards is

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

where $\gamma \in [0, 1]$ is the discount factor.

Note the recursive form

$$G_t = R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \cdots)$$

$$= R_{t+1} + \gamma G_{t+1}$$

Why discount? 1) Mathematically, it makes sure the convergence

2) Practically, tradeoff between immediate & future rewards

$\gamma \to 0$ myopic $\gamma \to 1$ farsighted.

Note that $G_t$ is a random variable evaluated at a outcome trajectory, which is not something we have control over. But we can control the expectation of this RV.

# More Definitions

- State-transition prob

$$p(s' \mid s, a) = P_r\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in R} p(s', r \mid s, a)$$

- Expected reward for $(s, a)$

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

- Expected reward for $(s, a, s')$

$$r(s, a, s') = \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s']$$

$$= \sum_{r \in R} r \cdot p(r \mid s, a, s') = \sum_{r \in R} r \cdot \frac{p(s', r \mid s, a)}{p(s' \mid s, a)}$$

To evaluate a policy $\pi$, we define

- Value of state $s$ under $\pi$

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k \cdot R_{t+k+1} \mid S_t = s\right]$$

state-value function for $\pi$

- Value of action $a$ in state $s$ under $\pi$

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k \cdot R_{t+k+1} \mid S_t = s, A_t = a\right]$$

action-value function for $\pi$, or Q value

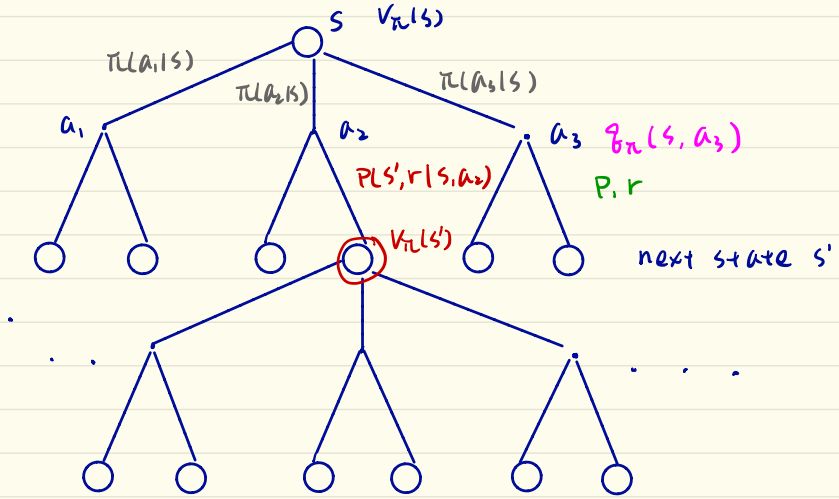
Focus on $V_\pi(s)$ for now:

$$V_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

$$= \sum_a \pi(a|s) \cdot \sum_{s'} \sum_r p(s', r \mid s, a) \cdot \left[ r + \gamma \mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s'] \right]$$

$$= \sum_a \pi(a|s) \cdot \sum_{s'} \sum_r p(s', r \mid s, a) \cdot \left[ r + \gamma V_\pi(s') \right] \quad \cdots (*)$$

This is a so called Bellman equation for $V_\pi$




This helps us evaluate a policy. But how do find best policy?

# Bellman Optimality Equation

- Optimal state-value function
- Optimal action-value function

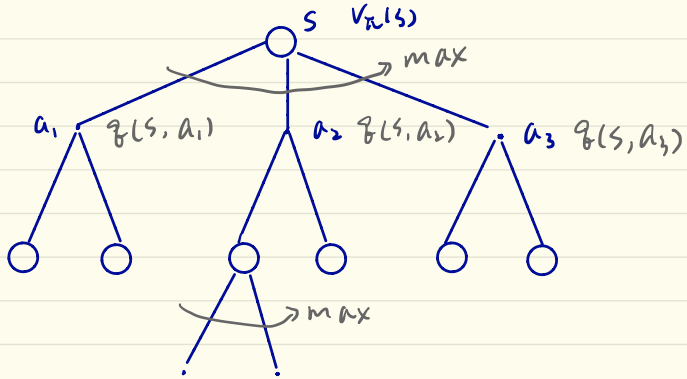
$$v^*(s) = \max_\pi v_\pi(s) \qquad q^*(s,a) = \max_\pi q_\pi(s,a)$$

$$v^*(s) = \max_{a \in A(s)} q^*(s,a) = \max_{a \in A(s)} \mathbb{E}_{\pi^*}[G_t \mid S_t = s, A_t = a]$$

$$= \max_{a \in A(s)} \mathbb{E}_{\pi^*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a]$$

$$= \max_{a \in A(s)} \mathbb{E}[R_{t+1} + \gamma \cdot v^*(S_{t+1}) \mid S_t = s, A_t = a]$$

$$= \max_{a \in A(s)} \sum_{s', r} p(s', r \mid s, a)[r + \gamma \cdot v^*(s')]$$




$v^*(s)$ must be equal to the expected return for the best action from that state $s$.

Remarks: 1) $\pi^*$ is not unique but $V^*$ is unique

2) It is very difficult (if not impossible) to solve Bellman equation in closed form. There are several ways to approximate it in dynamic programming.

3) $\max_{a \in A(s)} \sum_{s',r} p(s', r \mid s, a)\,[r + \gamma \cdot \underbrace{v^*(s')}_{(**)}]$ looks very greedy. Why is it optimal? The future expected reward is encoded in (**).

4) Historically, MDP was heavily studied for Stochastic Optimal Control.

In what follows, we introduce two ways to solve Bellman equation by techniques from dynamic programming

# Solving Bellman

Two approaches:

- Policy iteration = Policy evaluation + Policy improvement

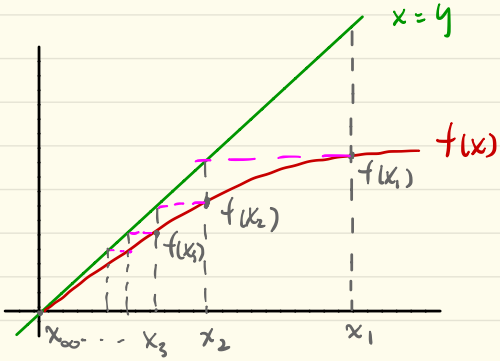
To evaluate a policy $\pi$, we need to solve

$$V_\pi(s) = \sum_a \pi(a|s) \cdot \sum_{s'} \sum_r p(s', r | s, a) \cdot \left[ r + \gamma V_\pi(s') \right]$$

same $V_\pi(\cdot)$

$$= f(V_\pi)$$

Consider solving $x = f(x)$, i.e., a fixed point of $f(\cdot)$




Iterative method

Eventually, we will converge to a fixed point if it exists.

Two questions: Existence & Uniqueness

Let $V_k(\cdot)$ be the value that we obtain in the $k$-th iteration under $\pi$. Iterative policy evaluation update rule

$$V_{k+1}(s) = \sum_a \pi(a|s) \cdot \sum_{s'} \sum_r p(s', r | s, a) \cdot \left[ r + \gamma V_k(s') \right]$$

Let $\pi_k$ be the policy adopted at $k$-th iteration

Find an improved policy based on current estimated $V_{\pi_k}$

Policy improvement:

$$\pi_{k+1} = \arg\max_a \sum_{s',r} p(s', r \mid s, a) \left[ r + \gamma V_{\pi_k}(s') \right]$$

Policy iteration: $\pi_0 \xrightarrow{E} V_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V_{\pi_1} \longrightarrow \cdots \xrightarrow{I} \pi^* \xrightarrow{E} V_{\pi^*}$

For finite MDP, since there are only finite number of policies this sequence always converges to optimal policy & value.

Drawback: Complexity! Need to run policy iteration many times.

Value iteration:

Directly turning Bellman optimality equation into update rule.

$$V_{k+1}(s) = \max_a \sum_{s'} \sum_{r} p(s', r \mid s, a) \cdot \left[ r + \gamma V_k(s') \right]$$

We stop once value's improvement become small.

Remark: Solving MDP requires the full knowledge of dynamics. Even if we have, it is still super complex. This is where RL can help.

# Reinforcement Learning

Idea: MDP is awesome, but requires knowledge of underlying distributions. Besides, it is very difficult to solve.

RL = Monte Carlo Method + MDP

MC method: Use experience! In MC, we use repeated simulations to estimate.

- Model-based RL: Estimate underlying transition probabilities

  $p(s', r \mid s, a) \quad \forall s \in S, s' \in S, a \in A(s), r \in R.$

- Model-free RL: Why bother to learn distributions if all we care is value? Directly learn value! (much easier!)

Now, since we don't know whether our current knowledge is accurate, we again face exploration vs exploitation!

The problem is a generalization of contexture bandit but now your current action affects next state.

There are many RL algorithm, we will talk about only one.

## Q-learning (Watkins 1989)

At state $S_t$, choose an action $A_t \in \mathcal{A}(S_t)$ according to Q function

Observe reward $R_t$ and use this reward to update Q function

- Action selection:

  At state $S_t$, select $A_t = \arg\max_a Q(S_t, a)$ w.p. $1-\varepsilon$

  Randomly pick $A_t \in \mathcal{A}(S_t)$ w.p. $\varepsilon$ i.e., $\varepsilon$-greedy

- Update:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \cdot \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

$$\text{New} \leftarrow \text{old} + \text{step size}\,[\text{target} - \text{old}]$$

where $\alpha$ is the step size introduced in Bandit problem and

$\gamma$ is the discounting factor introduced in MDP.

Remark: 1) $\alpha \in [0,1]$ : To deal with non-stationary prolem, we give more recent observations higher weight.

$\gamma \in [0,1]$ : To trade off between immediate and future rewards.

2) Deep reinforcement learning : Q-learning + NN-based nonlinear function approximation. A.K.A DQN.