Lecture 6 - Markov Decision Process (MDP)

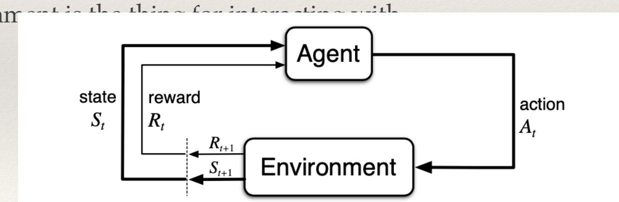
Instructor: GuangBing Yang, PhD

Introduction to MDPs

- MDPs stand for Markov decision processes — formally describe an environment for reinforcement learning.
- In MDPs, the environment is fully observable — the current state completely characterizes the process.
- The most of RL problems can be formalized as MDPs, e.g.,
  - Bandits are MDPs with only one state;
  - Partially observable problems can be converted into MDPs;
  - Optimal control deals with continue MDPs.

Introduction to MDPs

- MDPs frame the problem of learning from interaction to achieve a goal—namely the Agent-Environment Interface.
- Agent is the learner and decision maker
- Environment is the thing for interacting with



Introduction to MDPs

- The MDP and agent together give rise to a sequence of trajectory: $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3$.
- The random variables R and S have well defined discrete probability distributions dependent only on the preceding state and action.
- The probability of those values occurring at time t given particular values of the preceding state and action:
- $p(s', r \mid s, a) = Pr\{S_t = s, R_t = r \mid S_{t+1} = s, A_t = a\}$,

Markov Property

- In a Markov decision process, the probabilities given by p completely characterize the environment's dynamics.
- This is supported by Markov Property—which defines as:
  - A state S_t is Markov if and only if $P(S_{t+1} | S_t) = P(S_{t+1} | S_1, ..., S_t)$.
  - The state has all relevant information from the history
  - Once the state is known, the history can be discarded
  - The state is a sufficient statistic of the future
- In other words, "the future is independent of the past given the present"

State Transition Matrix

- For a Markov state s and its successor state s', the state transition probability is defined by: $p_{ss'} = P(S_{t+1} = s' | S_t = s)$
- Define the state transition matrix as the matrix of all transition probabilities from all states s to all successor states s', which is:
- $P = \begin{bmatrix} p_{11} \cdots p_{1n} \\ \cdots \\ p_{n1} \cdots p_{nn} \end{bmatrix}$, where $s \in n \times n$, and $s' \in n \times n$, and each row of the matrix sums to 1.

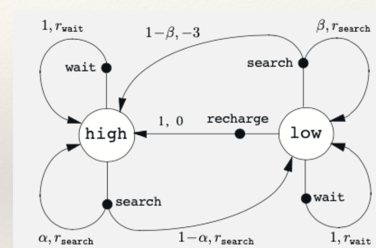
Markov Process

- A Markov process or Markov chain is defined as <S, P>
  - S is a set of states
  - P is a state transition probability matrix with $p_{ss'} = P(S_{t+1} = s' | S_t = s)$
- Example: Recycling Robot,

Example: Recycling Robot

- Don't consider the actions yet, it gives t state transition probability matrix as:
- p=

S | S'	High	Low
High	p(high | high)	p(high | low)
Low	p(low | high)	p(low | low)



Example: Recycling Robot

- Sample episodes for Recycling Robot Markov Chain starting from $S_1 = \text{high}, \;\; S_1, S_2, \ldots, S_T$.
- h, l, h, h, l…
- h, h, l, l, s…,

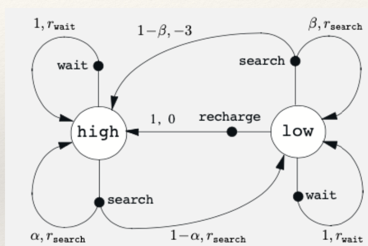
Markov Reward Process

- A Markov reward process is a Markov chain with reward values
- It is a tuple <S, P, R, γ>
  - S is a finite set of states
  - P is a state transition probability matrix,
  - R is a reward function, $R_s = \mathbb{E}[R_{t+1} \mid S_t = s]$
  - γ is a discount factor, $\gamma \in [0,1]$

Example: Recycling Robot MRP

- Don’t consider the actions yet, it gives t state transition probability matrix as:
- p=

S	S′	P
High	p(h | h)	α
H	p(l | h)	$1-\alpha$
Low	p(h | l)	$1-\beta$
L	p(l | l)	β



Return of Reward

- The return G_t is the total discounted reward from time-step t.
- $G_t = R_{t+1} + rR_{t+2} + \ldots, + R_T = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
  - T is a final step —when the agent-environment interaction breaks naturally into subsequences, called **episodes.**
  - Terminal state — the end of episode
  - The discount $\gamma \in [0,1]$ is the present value of future rewards
  - The value of receiving reward R after k+1 time-steps is $\gamma^k R$.
    - If $\gamma = 0$, the agent is “myopic” in being concerned only with maximizing immediate rewards
    - If $\gamma \to 1$, the return objective takes future rewards into account more strongly; the agent becomes more farsighted

What is the purpose of the discount

- Mathematically convenient to discount rewards
- Avoid infinite returns in cyclic Markov processes
- Uncertainty about the future many not be fully represented at the time-step t
- A larger immediate rewards may reduce the exploration which responses delayed rewards
- Animal/human behaviour shows preference for immediate reward
- If all sequences terminate, use undercounted Markov reward processes ($\gamma = 1$)

Value Function

- It gives the long-term value of state s
- The state value function $v(s) = \mathbb{E}[G_t | S_t = s], \forall s \in S$, is the expected return starting from state s
- Sample returns for Recycling Robot MRP: given $G_t = R_{t+1} + rR_{t+2} + \ldots, + R_T = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
  - starting from S_1 = high with $\gamma = 0.5$ (assume)
  - first row: high -> high, $v_1 = R_{\text{search}} + \gamma R_{\text{search}} = \alpha + \gamma\alpha = 1.5\alpha$
  - second row: high -> low, $v_1 = R_{\text{search}} + \gamma R_{\text{search}} = \alpha + \gamma(1-\alpha) = 0.5 + 0.5\alpha$
  - so on so forth, low -> low, recharge, $v_1 = R_{\text{search}} + \gamma R_{\text{search}} = 0 + \gamma 0 = 0$

Value Function

- starting from S_1 = high, if $\gamma = 1$, the highly farsighted.
- first row: high -> high, $v_1 = R_{\text{search}} + \gamma R_{\text{search}} = \alpha + \gamma\alpha = 2\alpha$
- second row: high -> low, $v_1 = R_{\text{search}} + \gamma R_{\text{search}} = \alpha + \gamma(1-\alpha) = \alpha + 1 - \alpha = 1$
- so on so forth, low -> low, recharge, $v_1 = R_{\text{search}} + \gamma R_{\text{search}} = 0 + \gamma 0 = 0$

Value Function

- starting from S_1 = high, if $\gamma = 0$, no discount, “myopic”.
- first row: high -> high, $v_1 = R_{\text{search}} + \gamma R_{\text{search}} = \alpha + \gamma\alpha = \alpha$
- second row: high -> low, $v_1 = R_{\text{search}} + \gamma R_{\text{search}} = \alpha + \gamma(1-\alpha) = \alpha$
- so on so forth, low -> low, recharge, $v_1 = R_{\text{search}} + \gamma R_{\text{search}} = 0 + \gamma 0 = 0$
- The return values are various due to the difference of the discount value γ.

Bellman Equation for MPRs

- The value function actually consists of two parts:
  - immediate reward R_{t+1}
  - discounted value of successor state $\gamma v(S_{t+1})$
- Thus, the Bellman equation is given as:

$$
\begin{aligned}
v(s) &= E[G_t \mid S_t = s] \\
&= E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s] \\
&= E[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \ldots) \mid S_t = s] \\
&= E[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= E[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s], \forall s \in S
\end{aligned}
$$

Bellman Equation for MPRs

- It is really a sum over all values of the three variables, a, s, and r. For each triple, we compute its probability, $\pi(a \mid s)p(s', r \mid s, a)$, weight the quantity in brackets by that probability, then sum over all possibilities to get an expected value.
- Thus, the Bellman equation is given as:
- $v(s) = R(s) + \gamma \sum_{s' \in S} p_{ss'} v(s')$

Bellman Equation for MPRs

- The Bellman equation can be given as in matrices form: $v = R + \gamma P v$
- where v is a column vector with one entry per state
- $$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} R_1 \\ \vdots \\ R_n \end{bmatrix} + \gamma \begin{bmatrix} p_{11} \ldots p_{1n} \\ \vdots \\ p_{n1} \ldots p_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

Bellman Equation for MPRs

- The Bellman equation $v = R + \gamma P v$ is a linear equation, so we have:
- $v = (I - \gamma P)^{-1} R$,
- The computational complexity is $O(n^3)$ for n states, and direct solution only possible for small MRPs
- Normally, there are many methods for large MRPs, for example,
  - Dynamic programming
  - Monte-Carlo evaluation
  - Temporal-Difference learning (TD)

Markov Decision Process

- A Markov decision process (MDP) actually is a Markov reward process with decision. Its all states satisfy Markov property.
- Definition: A Markov Decision Process is a tuple <S, A, P, R, γ>
  - S is a finite set of states
  - A is a finite set of actions
  - P is a state transition probability matrix, $P(s, s'|a) = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$
  - R is a reward function, $R(s, a) = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$
  - γ is a discount factor $\gamma \in [0,1]$

yguangbing@gmail.com, Guang.B@chula.ac.th Feb 23 2021

Policy

- A policy π is a probability distribution over actions given states, it is a kind of mapping of probabilities of selecting each possible action.
- Definition: $\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$
  - a policy fully defines the behaviour of an agent
  - MDP policies depend on the current state not the history — why
  - Policies are stationary (time-independent) $A_t \sim \pi(\,.\,|S_t), \ \forall t > 0$
  - Given an MDP <S, A, P, R, γ> and a policy π
  - The state sequence $S_1, S_2, ...$ is a Markov process <S, $P(\pi)$>
  - The state and reward sequence $S_1, R_1, S_2, R_2, ...$ is a Markov reward process <S, $P(\pi)$, $R(\pi)$, γ>

yguangbing@gmail.com, Guang.B@chula.ac.th Feb 23 2021

Policy

- Definition: $\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$
  - $$P_\pi(s, s') = \sum_{a \in A} \pi(a|s) P(s'|s, a)$$
  - $$R_\pi(s) = \sum_{a \in A} \pi(a|s) R(s, a)$$

yguangbing@gmail.com, Guang.B@chula.ac.th Feb 23 2021

Value Function of MDP

- The Bellman expression of the state-value function is given as immediate reward plus discounted value of successor state:
  - $v_\pi(s) = \mathbb{E}[G_t | S_t = s]$
- Definition: the action-value function $q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy π
  - $q_\pi(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a]$

yguangbing@gmail.com, Guang.B@chula.ac.th Feb 23 2021

Bellman Expectation Equation

- Definition: the state-value function $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π
  - $v_\pi(s) = \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$
- The action-value function $q_\pi(s,a)$ can be expressed as:
  - $q_\pi(s,a) = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a]$

yguangbing@gmail.com, Guang.B@chula.ac.th Feb 23 2021

Bellman Expectation Equation for v_π, Q_π

- $v_\pi(s) = \sum_{a \in A} \pi(a \mid s) q_\pi(s,a)$
- $q_\pi(s,a) = R(s,a) + \gamma \sum_{s' \in S} p_{ss'}(a) v_\pi(s')$
- substitute $q_\pi(s,a)$ with above equation, we have:
  - $v_\pi(s) = \sum_{a \in A} \left(R(s,a) + \gamma \sum_{s' \in S} p_{ss'}(a) v_\pi(s') \right)$, and the same as
  - $q_\pi(s,a) = R(s,a) + \gamma \sum_{s' \in S} p_{ss'}(a) \sum_{a' \in A} \pi(a' \mid s') q_\pi(s', a')$

yguangbing@gmail.com, Guang.B@chula.ac.th Feb 23 2021

Bellman Expectation Equation for v_π, Q_π

- The Bellman equation can be given as in matrices form: $v_\pi = R(\pi) + \gamma P(\pi) v_\pi$
- where v is a column vector with one entry per state
- $$\begin{bmatrix} v_\pi(1) \\ \vdots \\ v_\pi(n) \end{bmatrix} = \begin{bmatrix} R_1(\pi) \\ \vdots \\ R_n(\pi) \end{bmatrix} + \gamma \begin{bmatrix} p_{11}(\pi) \ldots p_{1n}(\pi) \\ \vdots \\ p_{n1}(\pi) \ldots p_{nn}(\pi) \end{bmatrix} \begin{bmatrix} v_\pi(1) \\ \vdots \\ v_\pi(n) \end{bmatrix}$$
- $v_\pi = (I - \gamma P(\pi))^{-1} R(\pi)$

yguangbing@gmail.com, Guang.B@chula.ac.th Feb 23 2021

Optimal Value Function

- The optimal state-value function is the maximum value function over all policies:
  - $v_*(s) = \max_\pi v_\pi(s)$
- The optimal action-value function is the maximum action-value function over all policies:
  - $q_*(s,a) = \max_\pi q_\pi(s,a)$
- An MDP is solved when knowing the optimal value function.

yguangbing@gmail.com, Guang.B@chula.ac.th Feb 23 2021

Optimal Policy

- Define a partial ordering over policies: $\pi \geq \pi'$ if $v_\pi(s) \geq v_{\pi'}(s), \forall s$
- Behind theorem:
  - There exists an optimal policy π_* that is better than or equal to all other policies, $\pi_* \geq \pi, \forall \pi$
  - All optimal policies achieve the optimal value function: $v_{\pi_*}(s) = v_*(s)$
  - All optimal policies achieve the optimal action-value function: $q_{\pi_*}(s, a) = q_*(s, a)$

Search for an Optimal Policy

- An optimal policy can be found by maximizing over $q_*(s, a)$
  - $\pi_*(a \mid s) = 1$ if $a = \text{argmax} q_*(s, a)$, or
  - $\pi_*(a \mid s) = 0$ otherwise
- Thus, there is always a deterministic optimal policy for any MDP, and if knowing $q_*(s, a)$, we immediately have the optimal policy.

Bellman Optimality Equation for v_* and q_*

- $v_*(s) = \max_a R(s, a) + \gamma \sum_{s' \in S} P_{ss'}(a) v_*(s')$
- $q_*(s, a) = R(s, a) + \gamma \sum_{s' \in S} p_{ss'}(a)\ v_*(s')$
- $q_*(s, a) = R(s, a) + \gamma \sum_{s' \in S} p_{ss'}(a)\ \max_{a' \in A} q_*(s', a')$

Solving Bellman Optimality Equation

- Bellman optimality equation is non-linear
- No closed form
- use Value Iteration,
- policy iteration
- Q-learning
- SARSA

Recap

- Reinforcement learning is about learning from interaction how to behave in order to achieve a goal.
- Agent and its environment interact over a sequence of discrete time steps.
- A policy is a stochastic rule by which the agent selects actions as a function of states.
- MDPs stand for Markov decision processes — formally describe an environment for reinforcement learning.
- Defined transition probabilities constitute a Markov decision process (MDP)
- A finite MDP is an MDP with finite state, action, and (as we formulate it here) reward sets.
- The return is the function of future rewards that the agent seeks to maximize.

Recap

- The undiscounted formulation is appropriate for episodic tasks, in which the agent–environment interaction breaks naturally into episodes;
- the discounted formulation is appropriate for continuing tasks.
- A policy's value functions assign to each state, or state–action pair.
- Optimal policy has the maximum expected returns from the optimal value functions.
- Any policy that is greedy with respect to the optimal value functions must be an optimal policy.
- The Bellman optimality equations are special consistency conditions that the optimal value functions must satisfy.

Questions and Lab