University of Illinois Fall 2018

ECE 544NA: Pattern Recognition
Lecture 23: November 13

Lecturer: Alexander Schwing Scribe: Yao Xu

1 Overview

This lecture introduces reinforcement learning and Markov decision process (MDP).

1.1 Recap

Machine learning paradigms discussed so far: discriminative learning and generative learning

- Discriminative learning: regression, classification, SVM, etc.
  Discriminative models study the condition probability $\boldsymbol{P(y|x)}$ and predict label $\boldsymbol{y}$ given variable $\boldsymbol{x}$ [1].
- Generative learning: K-Means, GMM, HMM, etc.
  Generative models study the probability distribution of given variable $\boldsymbol{x}$, i.e. $\boldsymbol{P(x)}$ [2].

Goal of both discriminative learning and generative learning is to extract **parameters leading to maximum likelihood**.

1.2 Reinforcement Learning

Reinforcement learning studies how should an **agent** take **actions** in an **environment** to maximize the **cumulative reward** [3].
Goal of reinforcement learning is to find a **policy**.

1.3 Application Examples of Reinforcement Learning

- Fly stunt manoeuvres in a helicopter
- Play Atari games
- Defeat the world champion at Go
- Manage investment portfolio
- Control a power station
- Make a humanoid robot walk

2 Formula of Reinforcement Learning

For an agent at step $\boldsymbol{t}$

- Current state $\boldsymbol{s_t}$

- Action to perform $\boldsymbol{a_t}$
- Reward for this action $\boldsymbol{r_t}$
- Agent reaches state $\boldsymbol{s_{t+1}}$

Settings of reinforcement learning: deterministic and stochastic

- Deterministic: next state $\boldsymbol{s_{t+1}}$ is determined by current state $\boldsymbol{s_t}$ and action $\boldsymbol{a_t}$, i.e.

$$s_{t+1} = f(s_t, a_t) \tag{1}$$

- Stochastic: next state $\boldsymbol{s_{t+1}}$ is not determined by current state $\boldsymbol{s_t}$ and action $\boldsymbol{a_t}$, but can be effected by other random factors from environment, i.e., all possible future states with action $\boldsymbol{a_t}$ at current state $\boldsymbol{s_t}$ satisfies

$$\sum_{s' \in S} P(s'|(s_t, a_t)) = 1 \tag{2}$$

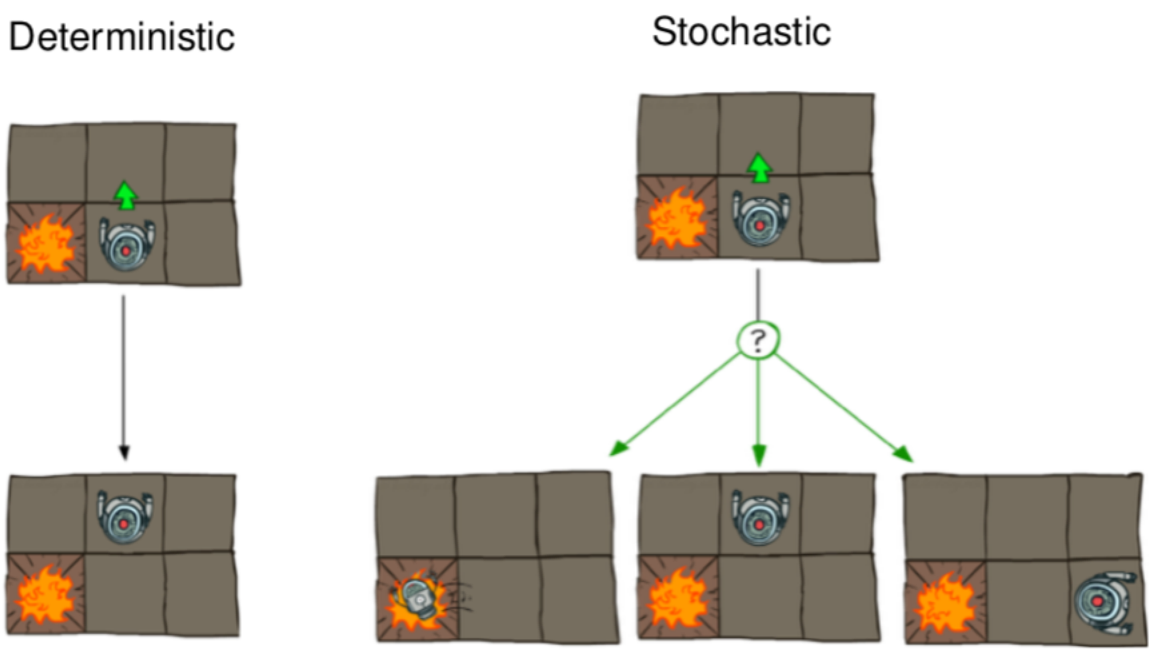

Figure 1: Deterministic and Stochastic setting

3 Markov Decision Process

3.1 Formula of Markov Decision Process

- A set of states $\boldsymbol{s \in S}$
- A set of actions $\boldsymbol{a \in A_s}$
- A transition probability $\boldsymbol{P(s'|(s, a))}$
  - The probability that an agent in state $\boldsymbol{s}$, and takes action $\boldsymbol{a}$, will transit itself to state $\boldsymbol{s'}$

- A reward function $\boldsymbol{R(s, a, s')}$
  - The reward the agent get by transiting from state $\boldsymbol{s}$ to state $\boldsymbol{s'}$ by action $\boldsymbol{a}$
  - May be simplified as $\boldsymbol{R(s)}$ or $\boldsymbol{R(s')}$
- A start state, and maybe a terminal state

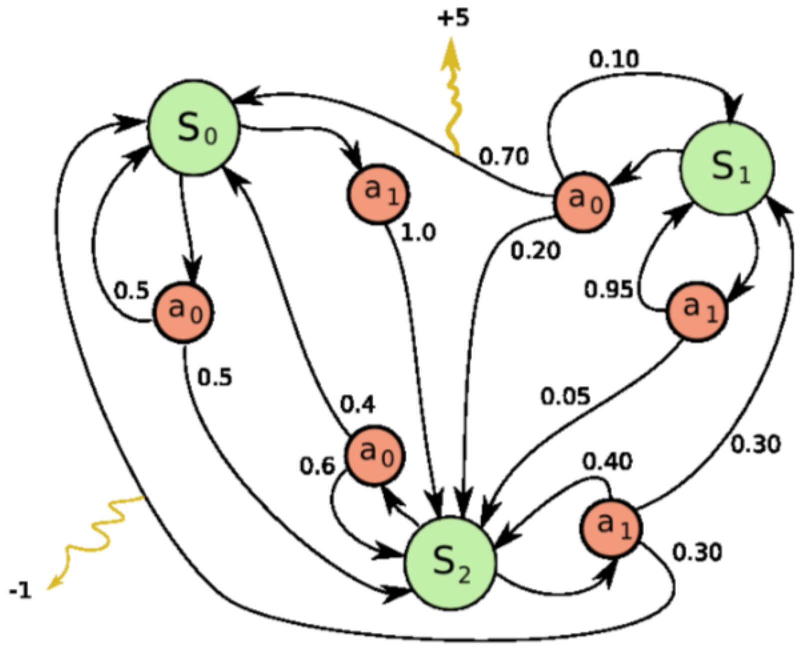

Figure 2: Example of an MDP

3.2 Markov Properties of an MDP

The future state $\boldsymbol{s_{t+1}}$ is only related to the current state $\boldsymbol{s_t}$ and the action $\boldsymbol{a_t}$. I.e., given the present state, the future and past are **independent**.

$$P(S_{t+1} = s' | (S_t = s_t, A_t = a_t), (S_{t-1} = s_{t-1}, A_{t-1} = a_{t-1}), ...(S_0 = s_0, A_0 = a_0)) = P(S_{t+1} = s' | (S_t = s_t, A_t = a_t)) \tag{3}$$

3.3 Goal of solving an MDP

Definition of policy: a policy is a mapping from states $\boldsymbol{S}$ to actions $\boldsymbol{A_s}$ that denotes action to be taken at state $\boldsymbol{s}$.

$$\pi(s) : \boldsymbol{S} \rightarrow \boldsymbol{A_s} \tag{4}$$

Goal of solving an MDP is to find **an optimal policy** $\boldsymbol{\pi^*(s)}$ that leads to maximum expected future reward $\boldsymbol{V^{\pi^*}(s_0)}$.

3.4 Policy Evaluation

The value of state $\boldsymbol{s}$ with policy $\boldsymbol{\pi}$, $\boldsymbol{V^{\pi}(s)}$, is defined as the possible future rewards.

For a terminal state $\boldsymbol{s_G}$, there is no future state, therefore,

$$V^{\pi}(s_G) = 0$$

For a deterministic environment, the future reward is recursively defined as the reward of transiting from current state $\boldsymbol{s}$ to next state $\boldsymbol{s'}$, plus the expected future reward of next state $\boldsymbol{s'}$. i.e.

$$V^{\pi}(s) = R(s,^{\pi}(s), s') + V^{\pi}(s')$$

For a stochastic environment, the next state $\boldsymbol{s'}$ is not determinate. Therefore, the future reward is defined as the expectation among all possibilities.

Generally, the value of a state $\boldsymbol{s}$ with policy $\boldsymbol{\pi}$ is defined as

$$V^{\pi}(s) = \begin{cases} 0, & \text{if } s \in \mathbf{G} \\ \sum_{s' \in S} P(s'|s, \pi(s))[R(s, \pi(s), s') + V^{\pi}(s')], & \text{otherwise} \end{cases} \tag{5}$$

Methods to compute expected future reward of initial state $\boldsymbol{V^{\pi}(s_0)}$ for policy $\boldsymbol{\pi}$:

- Direct Computation
- Backpropagation
- Linear System

Example: for an MDP shown as the graph below, green circles denotes states, red circles denotes actions, blue numbers denotes rewards, and red numbers denotes transition probabilities. This MDP has a terminal state $\boldsymbol{s_G}$

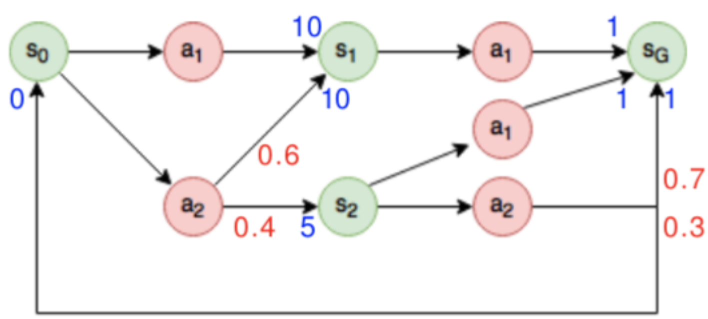

Figure 3: Example MDP

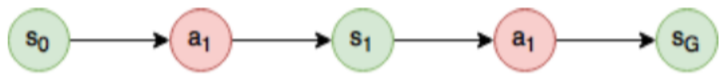

Figure 4: Policy π_1

3.4.1 Direct Computation

For policy $\boldsymbol{\pi_1(s) = \{\pi_1(s_0) = a_1, \pi_1(s_1) = a_1\}}$, $\boldsymbol{V^{\pi_1}(s_0)}$ can be computed directly:

$$
\begin{aligned}
V^{\pi_1}(s_G) &= 0 \\
V^{\pi_1}(s_1) &= R(s_1, \pi_1(s_1), s_G) + V^{\pi_1}(s_G) = 1 + 0 = 1 \\
V^{\pi_1}(s_0) &= R(s_0, \pi_1(s_0), s_1) + V^{\pi_1}(s_1) = 10 + 1 = 11
\end{aligned}
$$

3.4.2 Backpropagation

For policy $\boldsymbol{\pi_2(s) = \{\pi_2(s_0) = a_2, \pi_2(s_2) = a_1\}}$, $\boldsymbol{V^{\pi_2}(s_0)}$ can be computed by backpropogation:

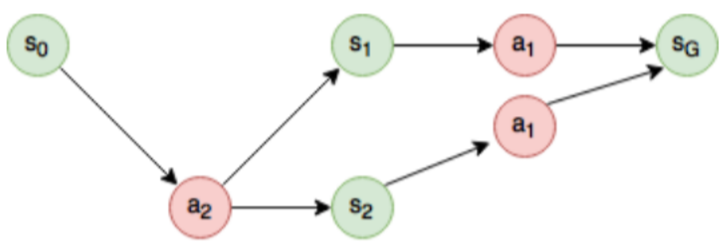

Figure 5: Policy π_2

$$
\begin{aligned}
V^{\pi_2}(s_G) &= 0 \\
V^{\pi_2}(s_1) &= R(s_1, \pi_2(s_1), s_G) + V^{\pi_2}(s_G) = 1 + 0 = 1 \\
V^{\pi_2}(s_2) &= R(s_2, \pi_2(s_2), s_G) + V^{\pi_2}(s_G) = 1 + 0 = 1
\end{aligned}
$$

$$
\begin{aligned}
V^{\pi_2}(s_0) &= P(s_1|s_0, \pi_2(s_0))[R(s_0, \pi_2(s_0), s_1) + V^{\pi_2}(s_1)] + P(s_2|s_0, \pi_2(s_0))[R(s_0, \pi_2(s_0), s_2) + V^{\pi_2}(s_2)] \\
&= 0.6(10 + 1) + 0.4(5 + 1) \\
&= 9
\end{aligned}
$$

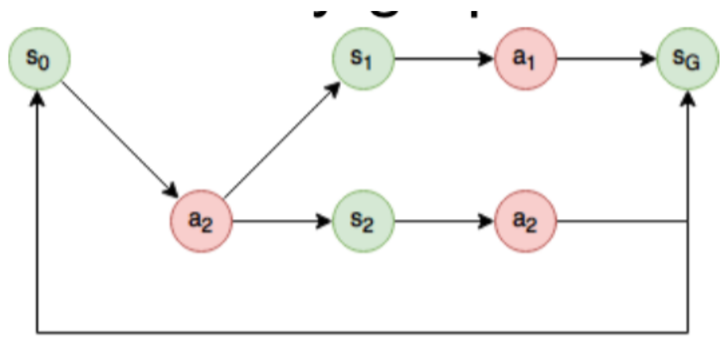

Figure 6: Policy π_3

3.4.3 Linear System

For policy $\boldsymbol{\pi_3(s) = \{\pi_3(s_0) = a_2, \pi_3(s_2) = a_2\}}$, $\boldsymbol{V^{\pi_3}(s_0)}$ can be computed by solving a linear system:

$$V^{\pi_3}(s_G) = 0$$

$$V^{\pi_3}(s_1) = R(s_1, \pi_2(s_1), s_G) + V^{\pi_3}(s_G) = 1 + 0 = 1$$

$$\begin{aligned} V^{\pi_3}(s_2) &= P(s_G|s_2, \pi_3(s_2))[R(s_2, \pi_3(s_2), s_G) + V^{\pi_3}(s_G)] + P(s_0|s_2, \pi_3(s_2))[R(s_2, \pi_3(s_2), s_0) + V^{\pi_3}(s_0)] \\ &= 0.7(1 + 0) + 0.3(0 + V^{\pi_3}(s_0)) \\ &= 0.7 + 0.3V^{\pi_3}(s_0) \end{aligned}$$

$$\begin{aligned} V^{\pi_3}(s_0) &= P(s_1|s_0, \pi_3(s_0))[R(s_0, \pi_3(s_0), s_1) + V^{\pi_3}(s_1)] + P(s_2|s_0, \pi_3(s_0))[R(s_0, \pi_3(s_0), s_2) + V^{\pi_3}(s_2)] \\ &= 0.6(10 + 1) + 0.4(5 + V^{\pi_3}(s_2)) \\ &= 8.6 + 0.4V^{\pi_3}(s_2) \end{aligned}$$

3.4.4 Iterative Refinement

Solving system of linear equations is expensive. Substantially, iterative refinement is an approach for an acceptable solution.

- Initialize $\boldsymbol{V_0^{\pi}(s)}$ for all $\boldsymbol{s \in S}$
- iteratively refine $\boldsymbol{V^{\pi}(s)}$ by

$$V_{i+1}^{\pi}(s) \leftarrow \sum_{s' \in S} P(s'|s, \pi(s))[R(s, \pi(s), s') + V_i^{\pi}(s')] \tag{6}$$

3.5 Methods to find best policy π^*

3.5.1 Exhaustive Search

- Enumerate all policies $\boldsymbol{\pi(s)}$
- Compute expected future reward $\boldsymbol{V^{\pi}(s_0)}$
- Choose policy $\boldsymbol{\pi^*}$ with largest expected future reward $\boldsymbol{V^{\pi^*}(s_0)}$

Drawback: exploring all possible policies is expensive. Number of policies may reach to

$$\prod_{s \in S} |A_s| \tag{7}$$

3.5.2 Policy Iteration

- Initialize policy $\boldsymbol{\pi(s)}$
- Repeat the iteration below until $\boldsymbol{\pi(s)}$ does not change
  - Solve policy evaluation equations (directly or iteratively)

$$V^{\pi}(s) = \sum_{s' \in S} P(s'|s, \pi(s))[R(s, \pi(s), s') + V^{\pi}(s')] \tag{8}$$

  - For each state $\boldsymbol{s}$, update $\boldsymbol{\pi(s)}$ with action that leads to **maximum expected future rewards**

$$\pi(s) = arg\max_{a \in A_S} \underbrace{\sum_{s' \in S} P(s'|s, a)[R(s, a, s') + V^{\pi}(s')]}_{Q(s,a)} \tag{9}$$

For the example MDP mentioned above, suppose $\boldsymbol{\pi(s)}$ is initialized with $\boldsymbol{\{\pi(s0) = a_2, \pi(s2) = a_1\}}$.

1. Policy evaluation:

$$V^{\pi}(s_G) = 0$$
$$V^{\pi}(s_1) = 1$$
$$V^{\pi}(s_2) = 1$$
$$V^{\pi}(s_0) = 9$$

2. Update state $\boldsymbol{s_2}$:

$$Q(s_2, a_1) = R(s_2, a_1, s_G) + V^{\pi}(s_G) = 1 + 0 = 1$$

$$\begin{aligned} Q(s_2, a_2) &= P(s_2, a_2, s_G)[R(s_2, a_2, s_G) + V^{\pi}(s_G] + P(s_2, a_2, s_0)[R(s_2, a_2, s_0) + V^{\pi}(s_0)] \\ &= 0.7(1 + V^{\pi}(s_G)) + 0.3(0 + V\pi(s_0)) \\ &= 3.4 \end{aligned}$$

   Therefore, update $\boldsymbol{\pi(s_2) = a_2}$

3. Update state $\boldsymbol{s_0}$:

$$Q(s_0, a_1) = R(s_0, a_1, s_1) + V^{\pi}(s_1) = 10 + 1 = 11$$

$$\begin{aligned} Q(s_0, a_2) &= P(s_0, a_2, s_1)[R(s_0, a_2, s_1) + V^{\pi}(s_1)] + P(s_0, a_2, s_2)[R(s_0, a_2, s_2) + V^{\pi}(s_2)] \\ &= 0.6(10 + V^{\pi}(s_1)) + 0.4(5 + V^{\pi}(s_2)) \\ &= 9 \end{aligned}$$

   Therefore, keep $\boldsymbol{\pi(s_0) = a_2}$

4. Repeat step 1-4 again, and no update found this time
5. Output: $\boldsymbol{\{\pi^*(s_0) = a_2, \pi^*(s_2) = a_2\}}$ with $\boldsymbol{V^{\pi^*}(s_0) = 11}$

3.5.3 Value Iteration (Bellman optimality principle)

Instead of refining a policy, value iteration method refines the expected future reward $\boldsymbol{V(s)}$ to find $\boldsymbol{V^*(s)}$, and then extract $\boldsymbol{\pi^*(s)}$.
For each state $\boldsymbol{s}$, the maximum expected future reward is

$$V^*(s) = \max_{a \in A_s} \underbrace{\sum_{s' \in S} P(s'|s,a)[R(s,a,s') + V^*(s')]}_{Q^*(s,a)} \tag{10}$$

I.e., if $\boldsymbol{Q(s,a)}$ denote expected future reward at state $\boldsymbol{s}$, and the next action to be taken is $\boldsymbol{a}$,

$$Q^*(s,a) = \sum_{s' \in S} P(s'|s,a)[R(s,a,s') + \max_{a' \in A'_s} Q^*(s',a')] \tag{11}$$

For very small MDPs, $\boldsymbol{V^*(s)}$ can be solved via linear program. Otherwise, iteratively refinement is needed.

- For every state, initialize $\boldsymbol{V_0(s)}$
- In every iteration, compute

$$V_{i+1}(s) \leftarrow \max_{a \in A_s} \sum_{s' \in S} P(s'|s,a)[R(s,a,s') + V_i(s')] \tag{12}$$

To extract optimal policy:

$$\pi^*(s) = arg \max_{a \in A_s} \sum_{s' \in S} P(s'|s,a)[R(s,a,s') + V*(s')] \tag{13}$$

I.e.

$$\pi^*(s) = arg \max_{a \in A_s} Q^*(s,a) \tag{14}$$

4 Answers to Quiz Questions

4.1 What differentiates Reinforcement Learning from supervised learning?

- No supervisor, only reward signal
- Delayed feedback
- Actions effect received data

4.2 What is an MDP?

A Markov Decision Process is a 4-tuple $\boldsymbol{(S, A_s, P(s|a,s'), R_a(s,a,s'))}$, where

- $\boldsymbol{S}$ is a set of states
- $\boldsymbol{A_s}$ is a set of actions can be taken at state $\boldsymbol{s}$
- $\boldsymbol{P(s|a,s')}$ is the probability that action $\boldsymbol{a}$ can transit state $\boldsymbol{s}$ to state $\boldsymbol{s'}$
- $\boldsymbol{R(s,a,s')}$ is the reward of the transition mentioned above

4.3 What differentiates policy iteration from policy evaluation?

Policy iteration is a method to find the optimal policy. It evaluates the current policy and updates it with some strategies.

Policy evaluation is not a method to find a policy, but a method to evaluate a given policy.

References

[1] Wikipedia. Discriminative model. https://en.wikipedia.org/wiki/Discriminative_model. Accessed November 20, 2018.

[2] Wikipedia. Generative model. https://en.wikipedia.org/wiki/Generative_model. Accessed November 20, 2018.

[3] Wikipedia. Reinforcement learning. https://en.wikipedia.org/wiki/Reinforcement_learning. Accessed November 20, 2018.