University of Illinois Fall 2018

ECE 544NA: Pattern Recognition
Lecture 23: November 13

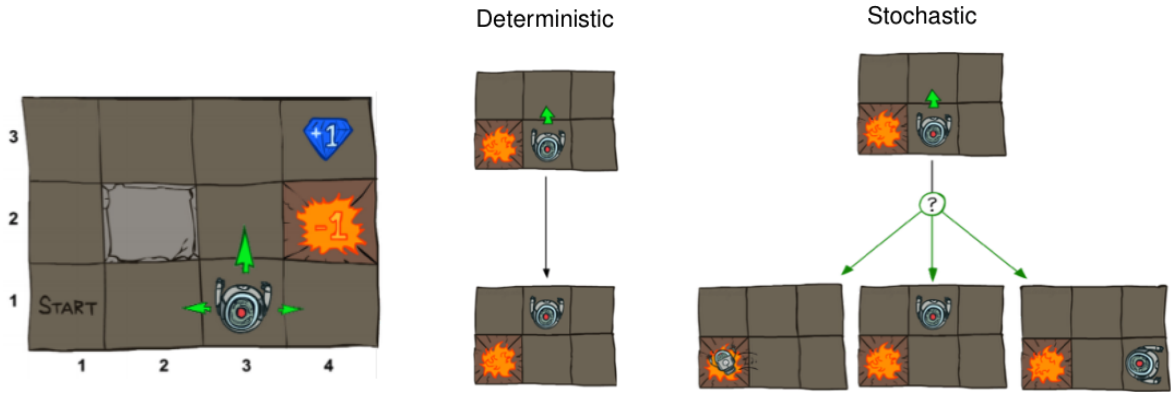
Lecturer: Alexander Schwing Scribe: Yulun Wu

1 Overview

In this lecture we study Markov Decision Process (MDP). MDP is one of the methods in studying optimization problems using reinforcement learning. So far we have studied two machine learning paradigms: discriminative learning, and generative learning. Logistic regression and Support Vector Machine (SVM) are typical approaches for discriminative learning, and Gaussian Mixture Model (GMM), Hidden Markov Model (HMM), Variational Auto-Encoders, and Generative Adversarial Nets are typical approaches for generative learning. Both of machine learning paradigms are supervised learning: they depend on the training process to find some suitable model parameters given labeled data. Reinforcement learning (RL), however, is differentiated from the two since RL is unsupervised, i.e., we don't train the system for optimum model parameters, but give a reward after the program finishes. The system doesn't know whether the reward is good. In such system, we have an "agent", and at each step t, the agent

- Thinks/Knows about being in state s_t
- Performs action a_t
- Receives scalar reward $r_t \in R$
- Finds in state s_{t+1}

and the agent repeats such processes until the program reaches the end (reaches the destination/finds the maximum reward). Figure 1(a) shows a specific example.



(a) (b)

Figure 1: (a) Setup of a scenario. (b) Deterministic and Stochastic setting of the program.

In the setup in Figure 1(a), we have $3 \times 4 = 12$ states, the goal of the agent is to collect +1 without falling into -1 trap. Figure 1(b) shows two different setups for solving this problem. The first

approach is to design a deterministic program that the agent only moves forward. While another is to design a stochastic program such that the agent is possible to move to forward, left, and right directions. In machine learning studies, we are more interested in the stochastic approach. Designing the stochastic program, we expect the agent to learn a route to maximum the reward. Markov Decision Process is a such stochastic process that allows the program to make the best decision (in this example, maximize the reward) without human training.

2 Markov Decision Process

Formally, a MDP contains five elements

- A set of states $s \in S$.
- A set of actions $a \in A_s$.
- A transition probability $P(s'|s,a)$.
- A reward function $R(s,a,s')$ (sometimes just $R(s')$ or $R(s)$).
- A start and maybe a terminal state.

One property of MDP is that given the present state, the future and past are independent. In other words, the next state only depends on the current state and the action performs. Assume at step t, the agent is at state $S_t = s_t$ and it performs action $A_t = a_t$, and the next state $S_{t+1} = s'$, the transition probability at current step can be written as

$$\begin{aligned} &P(S_{t+1} = s'|S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, S_{t-2} = s_{t-2}, \cdots, , S_0 = s_0) \\ =&P(S_{t+1} = s'|S_t = s_t, A_t = a_t) \end{aligned} \tag{1}$$

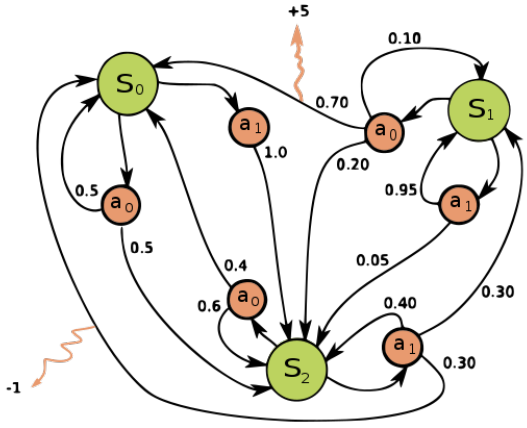


Figure 2: A simple MDP with three states (green circles) and two actions (red circles).

Figure 2 shows a pictorial example of MDP. In this MDP, there are three states $S = \{s_0, s_1, s_2\}$, and two actions $A_s = \{a_0, a_1\}$. Each edge is assigned with a transition probability $P(s'|s,a) \in [0,1]$, and a reward if the action is performed (in yellow arrow). For example, if we start from state S_0 and perform action a_0, we have 50% chance to go back to S_0, and 50% change to go to S_2. At state

S_2, if we choose to perform a_0 again, there's 40% change that we go to S_0 and 60% change to go back to S_2, and etc.

From the presentation above, we see reinforcement learning is very different from discriminative learning and generative learning. They have three major differences

1. Reinforcement learning has no supervisor but only reward signal.
2. Reinforcement learning has delayed feedback, the results cannot be shown immediately until the end of program.
3. In reinforcement learning, actions affect received data. Based on the different action the system performs, the output results can be different.

The core problem of MDP is to find a "policy" for the decision maker: we want to find a function $\pi(s) : S \to A_s$ to maximize the reward. After the best function $\pi^*(s)$ is determined for every state s, the transition possibility is further simplified to

$$P(s_{t+1} = s'|s_t = s_t, a_t = a_t) = P(s_{t+1} = s'|s_t = s_t) \tag{2}$$

which is recognized as a Markov chain. In rest of this lecture, we introduce three methods to find the best policy $\pi^*(s)$:

- Exhaustive search
- Policy iteration
- Value iteration

2.1 Exhaustive search

The simplest method to find the best policy $\pi^*(s)$ is to evaluate the expected future reward $V^\pi(s_0)$ for every possible policy $\pi(s)$, and choose the policy π^* with largest $V^\pi(s_0)$. However, given the total numbers of policies

$$\prod_{s \in S} |A_s| \tag{3}$$

that need to be computed, the exhaustive search method can be computationally expensive for large MDP. Next we use an example to show how to compute expected future reward $V^\pi(s)$ for a given policy using policy evaluation. Note that policy evaluation is different from policy iteration method we will introduce in next section.

Figure 3 shows a simple MDP with three states s_0, s_1, s_2, and two actions a_0, a_1. There are three route options, first let's choose $\pi(s_0) = a_1, \pi(s_1) = a_1$, the policy graph corresponding to this rule is shown in Figure 4. The policy graph in Figure 4 is straight forward, there is only one possible route, and the expected future rewards $V^\pi(s)$ for s_0 and s_1 can be calculated as

$$V^\pi(s_1) = 1 \tag{4}$$

$$V^\pi(s_0) = 10 + V^\pi(s_1) = 10 + 1 = 11 \tag{5}$$

Next we choose the policy to be $\pi(s_0) = a_2, \pi(s_2) = a_1$, the policy graph corresponding to this rule is shown in Figure 5. In this policy graph there are two route options with given possibilities, the

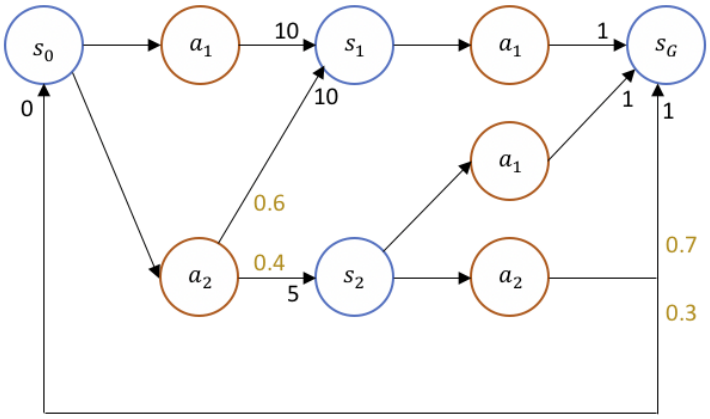


Figure 3: MDP with three states (blue circles) and two actions (red circles).

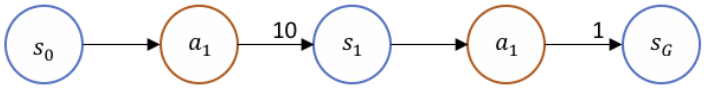


Figure 4: Policy graph that corresponds to $\pi(s_0) = a_1, \pi(s_1) = a_1$.

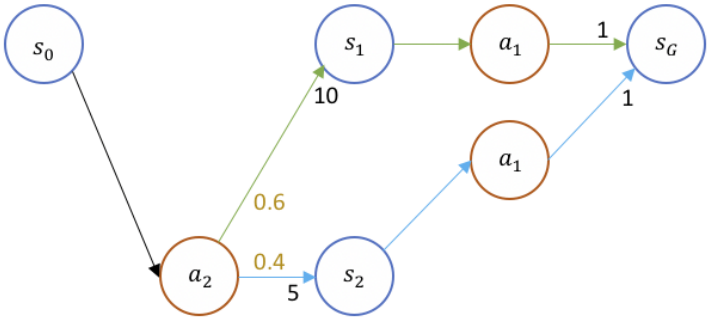


Figure 5: Policy graph that corresponds to $\pi(s_0) = a_2, \pi(s_2) = a_1$.

expected future rewards $V^\pi(s)$ for s_1, and s_2 can be calculated as

$$V^\pi(s_1) = 1 \tag{6}$$
$$V^\pi(s_2) = 1 \tag{7}$$

Using the back-propagation, we find $V^\pi(s_0)$ is the sum of the future expected reward at each route multiply by its transition probability given current state s and action a

$$\begin{aligned} V^\pi(s_0) &= P(s_1|s_0, a_2) \cdot (10 + V^\pi(s_1)) + P(s_2|s_0, a_2) \cdot (5 + V^\pi(s_2)) \\ &= 0.6 \cdot (10 + 1) + 0.4 \cdot (5 + 1) = 9 \end{aligned} \tag{8}$$

At last we choose the policy to be $\pi(s_0) = a_2, \pi(s_2) = a_2$, the policy graph corresponding to this rule is shown in Figure 6.

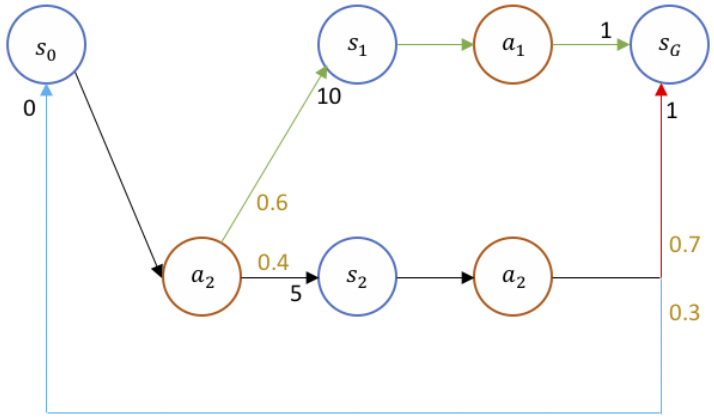


Figure 6: Policy graph that corresponds to $\pi(s_0) = a_2, \pi(s_1) = a_2$.

Similar to the calculations above, the expected future rewards $V^\pi(s)$ for s_0, s_1, and s_2 can be calculated as

$$V^\pi(s_1) = 1 \tag{9}$$

$$\begin{aligned} V^\pi(s_2) &= P(s_G|s_2, a_2) \cdot (1 + V^\pi(s_G)) + P(s_0|s_2, a_2) \cdot (0 + V^\pi(s_0)) \\ &= 0.7 \cdot 1 + 0.3 \cdot V^\pi(s_0) \end{aligned} \tag{10}$$

$$\begin{aligned} V^\pi(s_0) &= P(s_1|s_0, a_2) \cdot (10 + V^\pi(s_1)) + P(s_2|s_0, a_2) \cdot (5 + V^\pi(s_2)) \\ &= 0.6 \cdot (10 + V^\pi(s_1)) + 0.4 \cdot (5 + V^\pi(s_2)) \end{aligned} \tag{11}$$

Equations (9)-(11) form a linear system

$$\begin{bmatrix} 1 & -0.6 & -0.4 \\ 0 & 1 & 0 \\ -0.3 & 0 & 1 \end{bmatrix} \begin{bmatrix} V^\pi(s_0) \\ V^\pi(s_1) \\ V^\pi(s_2) \end{bmatrix} = \begin{bmatrix} 8 \\ 1 \\ 0.7 \end{bmatrix} \tag{12}$$

Solving this linear system, we find $V^\pi(s_0) = 10.09$, $V^\pi(s_1) = 1$, and $V^\pi(s_2) = 3.73$.

Comparing the results from three policies above, we find the first policy $\pi(s_0) = a_1$ in Figure 4 is the best policy for state s_0, the maximum expected future reward $V^*(s_0) = 11$. The best policy for state s_1 is $\pi(s_1) = a_1$ since there's only one action available for state s_1. And the third policy $\pi(s_2) = a_2$ in Figure 6 is the best policy for state s_2, the maximum expected future reward $V^*(s_s) = 3.73$.

In summary, the policy evaluation above requires to solve linear systems of equations

$$V^\pi(s) = 0 \text{ if } s \in G \tag{13}$$

$$V^\pi(s) = \sum_{s' \in S} P(s'|s, \pi(s))[R(s, \pi(s), s') + V^\pi(s')] \tag{14}$$

However, such approach can be expensive to compute since the time used for solving a linear system is $O(n^2)$ with linear solver, or $O(n^3)$ with direct inversion of matrix. Instead of solving system of

linear equations, we can use iterative refinement

$$V_{i+1}^{\pi}(s) \leftarrow \sum_{s' \in S} P(s'|s, \pi(s))[R(s, \pi(s), s') + V_i^{\pi}(s)] \tag{15}$$

to find the expected reward at time $i+1$ based on the reward at time i, but this iterative refinement still requires to search over all policies, which is also computationally expensive.

2.2 Policy iteration

Policy iteration is another method to solve for the best policy π^* [2]. The idea of policy iteration is very similar to Gradient Descent (GD): in GD we initialize the program with a random stating point, then iterative the program towards the descent direction (usually negative gradient direction) until the local minimum is found; similarly in policy iteration method, the policy is initialized randomly, and it is improved corresponds to the maximum expected future rewards iteratively. In general, the algorithm for policy iteration can be summarized as follows

- Initial policy π
- Repeat until policy π does not change
  - Solve system of equations (e.g., iteratively)

$$V^{\pi}(s) = \sum_{s' \in S} P(s'|s, \pi(s))[R(s, \pi(s), s') + V^{\pi}(s')] \tag{16}$$

  - Extract new policy using

$$\pi(s) = \arg\max_{a \in A_s} V^{\pi}(s, a) \tag{17}$$

$$\text{where } V^{\pi}(s, a) = P(s'|s, a)[R(s, a, s') + V^{\pi}(s')] \tag{18}$$

In this algorithm, the first step is solve system of equations for state $s \in S$ to find $V^{\pi}(s)$, then we compare the actions at state s, and choose the action that returns maximum expected rewards. To better illustrate the idea, we repeat the example in Section 2.1 using policy iteration, and compare the results with the results in Section 2.1.

First we can initial the policy to $\pi(s_0) = \pi(s_2) = a_2$ and $\pi(s_1) = a_1$ since s_1 state only has action a_1, then write down the linear system given this policy

$$\begin{aligned} V^{\pi}(s_0) &= P(s_1|s_0, a_2) \cdot (10 + V^{\pi}(s_1)) + P(s_2|s_0, a_2) \cdot (5 + V^{\pi}(s_2)) \\ &= 0.6 \cdot (10 + V^{\pi}(s_1)) + 0.4 \cdot (5 + V^{\pi}(s_2)) \end{aligned} \tag{19}$$

$$V^{\pi}(s_1) = 1 \tag{20}$$

$$\begin{aligned} V^{\pi}(s_2) &= P(s_G|s_2, a_2) \cdot (1 + V^{\pi}(s_G)) + P(s_0|s_2, a_2) \cdot (0 + V^{\pi}(s_0)) \\ &= 0.7 \cdot 1 + 0.3 \cdot V^{\pi}(s_0) \end{aligned} \tag{21}$$

We recognize Equations (18)-(20) are identical to Equations (9)-(11) in Section 2.1, and the results for this linear system are $V^{\pi}(s_0) = 10.09$, $V^{\pi}(s_1) = 1$, and $V^{\pi}(s_2) = 3.73$. Next we update the

policy

$$\pi(s_0) = a_1\text{: } V^\pi(s_0, a_1) = P(s_1|s_0, a_1) \cdot (10 + V^\pi(s_1)) \\ = 1 \cdot (10 + 1) = 11 \tag{22}$$

$$\pi(s_0) = a_2\text{: } V^\pi(s_0, a_2) = P(s_1|s_0, a_2) \cdot (10 + V^\pi(s_1)) + P(s_2|s_0, a_2) \cdot (5 + V^\pi(s_2)) \\ = 0.6 \cdot (10 + 1) + 0.4 \cdot (5 + 3.73) = 10.09 \tag{23}$$

$$\pi(s_2) = a_1\text{: } V^\pi(s_2, a_1) = P(s_G|s_2, a_1) \cdot (1 + V^\pi(s_G)) \\ = 1 \cdot (1 + 0) = 1 \tag{24}$$

$$\pi(s_2) = a_2\text{: } V^\pi(s_2, a_2) = P(s_G|s_2, a_2) \cdot (1 + V^\pi(s_G)) + P(s_0|s_2, a_2) \cdot (0 + V^\pi(s_0)) \\ = 0.7 \cdot (1 + 0) + 0.3 \cdot (0 + 10.09) = 3.73 \tag{25}$$

such that we choose new policy to be $\pi(s_0) = a_1$, $\pi(s_2) = a_2$ and $\pi(s_1) = a_1$. Using the updated policy we repeat the process, the new linear system is

$$V^\pi(s_0) = P(s_1|s_0, a_1) \cdot (10 + V^\pi(s_1)) \\ = 1 \cdot (10 + V^\pi(s_1)) \tag{26}$$

$$V^\pi(s_1) = 1 \tag{27}$$

$$V^\pi(s_2) = P(s_G|s_2, a_1) \cdot (1 + V^\pi(s_G)) \\ = 1 \cdot (1 + V^\pi(s_G)) \tag{28}$$

solving the linear system we find $V^\pi(s_0) = 11$, $V^\pi(s_1) = 1$, and $V^\pi(s_2) = 1$. Next we update policy again

$$\pi(s_0) = a_1\text{: } V^\pi(s_0, a_1) = P(s_1|s_0, a_1) \cdot (10 + V^\pi(s_1)) \\ = 1 \cdots (10 + 1) = 11 \tag{29}$$

$$\pi(s_0) = a_2\text{: } V^\pi(s_0, a_2) = P(s_1|s_0, a_2) \cdot (10 + V^\pi(s_1)) + P(s_2|s_0, a_2) \cdot (5 + V^\pi(s_2)) \\ = 0.6 \cdot (10 + 1) + 0.4 \cdot (5 + 1) = 10.09 \tag{30}$$

$$\pi(s_2) = a_1\text{: } V^\pi(s_2, a_1) = P(s_G|s_2, a_1) \cdot (1 + V^\pi(s_G)) \\ = 1 \cdot (1 + 0) = 1 \tag{31}$$

$$\pi(s_2) = a_2\text{: } V^\pi(s_2, a_2) = P(s_G|s_2, a_2) \cdot (1 + V^\pi(s_G)) + P(s_0|s_2, a_2) \cdot (0 + V^\pi(s_0)) \\ = 0.7 \cdot (1 + 0) + 0.3 \cdot (0 + 11) = 4 \tag{32}$$

We see the policy stays the same. Therefore the best policy for this example is $\pi(s_0) = \pi(s_1) = a_1$ and $\pi(s_2) = a_2$. This result is consistent with the best policy we found in Section 2.1. In Section 2.1 we calculated all three possible routes, and the first policy $\pi(s_0) = \pi(s_0) = a_1$ returns the highest expected reward for state s_0, and the third policy $\pi(s_2) = a_2$ returns the highest expected reward for state s_2.

2.3 Value iteration

Value iteration is another commonly used method to find the best policy. Instead of finding the best policy directly, value iteration search over the value space to find the optimal value function V^*. After the optimal V^* is found, the best policy is decoded from V^*. In general, the algorithm for value iteration can be summarized as follows

- Search over values (Bellman optimality principle [1])

$$V^*(s) = \max_{a \in A_s} \sum_{s' \in S} P(s'|s,a)[R(s,a,s') + V^*(s')] \tag{33}$$

- Compute the resulting policy at the end (decoding policy)

$$\pi(s) = \arg\max_{a \in A_s} \sum_{s' \in S} P(s'|s,a)[R(s,a,s') + V^*(s')] \tag{34}$$

In order to solve for $V^*(s)$, we can choose between two methods

- Solve via linear program (for very small MDPs)
- Iterative refine

$$V_{i+1}(s) \leftarrow \max_{a \in A_s} P(s'|s,a)[R(s,a,s') + V_i(s')] \tag{35}$$

Indtead of solving for V^* directly, we can also introduce an intermediate variable

$$Q^*(s,a) = \sum_{s' \in S} P(s'|s,a)[R(s,a,s') + V^*(s')] \tag{36}$$

then Equation (33) and (34) can be written as

$$V^*(s) = \max_{a \in A_s} Q^*(s,a) \tag{37}$$

$$\pi(s) = \arg\max_{a \in A_s} Q^*(s,a) \tag{38}$$

We see $Q^*(s,a)$ is the expected future reward at state s and perform action a, $V^*(s)$ is the maximum expected future reward by choosing the maximum of $Q^*(s,a)$ among actions $a \in A_s$, then naturally the best policy is the one that results the maximized $Q^*(s,a)$.

3 Summary

In this lecture we studied Markov Decision Process. With a known MDP, we learned to compute V^*, Q^*, π^* using exhaustive search or policy/value iteration. Exhaustive search method requires us to find the expected future value $V(s)$ for every possible combination of policies, which can be very expensive to compute if the MDP is large. In order to reduce the computation complexity, we introduce policy iteration and value iteration methods. Both methods are very similar except policy iteration searches over policies to find the best policy directly, while value iteration searches over values to find the optimal value function V^*, then decode the best policy from V^*. All three methods above depends on the known MDP model. In next lecture, we study the reinforcement learning when the MDP is not known.

Quiz

1. What differentiates RL from supervised learning?
   Reinforcement learning is unsupervised, the program doesn't require the training process with some labeled dataset. For instead we assign reward to each action that the program performs, and let the program learn the route to gain the maximum reward.

2. What is a MDP?
   MDP is a stochastic process that contains a set of states S, a set of actions A_s, a transition probability $P(s'|s,a)$, and a reward function $R(s,a,s')$. It can be used in the situation when the program involves some random decision by decision maker.

3. What differentiates policy iteration from policy evaluation?
   Policy iteration is one optimization method to find best policy. The principle of policy iteration is similar to gradient descent, that the program find the local optimal at given policies, and update the policies until converge. Policy evaluation is only the method to find the expected future reward given a fixed policy, it is not a optimization method.

References

[1] R. Bellman. A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957.

[2] R. A. Howard. *Dynamic Programming and Markov Processes*. The M.I.T. Press., 1960.