CSC2542
Introduction to Markov Decision Processes

Sheila McIlraith
Department of Computer Science
Winter, 2019
Acknowledgements

With a few exceptions, these slides were developed by David Silver following the notation and development in:

Reinforcement Learning: An Introduction
Sutton & Barto, (Draft 2nd edition)

Thank you to David for sharing his slides.

NOTATION WARNING:
The notation in Sutton & Barto has changed from the 1\textsuperscript{st} to 2\textsuperscript{nd} edition of their book. As you read papers, you’ll also see that notation in the RL and MDP communities varies.
Different Classes Planning Problems

dynamics: deterministic, nondeterministic, probabilistic
observability: full, partial, none
horizon: finite, infinite
objective requirement: satisfying, optimizing

– classical planning
– conditional planning with full observability (FOND)
– conditional planning with partial observability (POND)
– conformant planning
– markov decision processes (MDP)
– partial observable MDP (POMDP)
– preference-based/over-subscription planning
Different Classes Planning Problems

dynamics: deterministic, nondeterministic, probabilistic

observability: full, partial, none

horizon: finite, infinite

objective requirement: satisfying, optimizing

... 

– classical planning
– conditional planning with full observability
– conditional planning with partial observability
– conformant planning

– markov decision processes (MDP)
– partial observable MDP (POMDP)
– preference-based/over-subscription planning
1. Markov Processes

2. Markov Reward Processes

3. Markov Decision Processes

4. Extensions to MDPs
Introduction to MDPs

- *Markov decision processes* formally describe an environment for reinforcement learning
- Where the environment is *fully observable*
- i.e. The current *state* completely characterises the process
- Almost all RL problems can be formalised as MDPs, e.g.
  - Optimal control primarily deals with continuous MDPs
  - Partially observable problems can be converted into MDPs
  - Bandits are MDPs with one state
“The future is independent of the past given the present”

Definition

A state $S_t$ is Markov if and only if

$$\mathbb{P} [S_{t+1} \mid S_t] = \mathbb{P} [S_{t+1} \mid S_1, \ldots, S_t]$$

- The state captures all relevant information from the history
- Once the state is known, the history may be thrown away
- i.e. The state is a sufficient statistic of the future
State Transition Matrix

For a Markov state $s$ and successor state $s'$, the state transition probability is defined by

$$P_{ss'} = \Pr \left[ S_{t+1} = s' \mid S_t = s \right]$$

State transition matrix $P$ defines transition probabilities from all states $s$ to all successor states $s'$,

$$P = \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix}$$

to

where each row of the matrix sums to 1.
A Markov process is a memoryless random process, i.e. a sequence of random states $S_1, S_2, ...$ with the Markov property.

**Definition**

A *Markov Process* (or *Markov Chain*) is a tuple $\langle S, P \rangle$

- $S$ is a (finite) set of states
- $P$ is a state transition probability matrix,
  
  $P_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s]$
Example: Student Markov Chain
Sample episodes for Student Markov Chain starting from $S_1 = C1$

$S_1, S_2, ..., S_T$

- C1 C2 C3 Pass Sleep
- C1 FB FB C1 C2 Sleep
- C1 C2 C3 Pub C2 C3 Pass Sleep
- C1 FB FB C1 C2 C3 Pub C1 FB FB FB C1 C2 C3 Pub C2 Sleep
Example: Student Markov Chain Transition Matrix

\[ P = \begin{pmatrix}
C1 & C2 & C3 & Pass & Pub & FB & Sleep \\
0.5 & 0.8 & 0.6 & 0.4 & 0.2 & 0.9 & 1.0 \\
0.2 & 0.4 & 0.4 & 0.1 & 0.4 & 0.9 & 1.0 \\
\end{pmatrix} \]
Markov Reward Process

A Markov reward process is a Markov chain with values.

**Definition**

A *Markov Reward Process* is a tuple $\langle S, P, R, \gamma \rangle$

- $S$ is a finite set of states
- $P$ is a state transition probability matrix, $P_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s]$
- $R$ is a reward function, $R_s = \mathbb{E}[R_{t+1} | S_t = s]$
- $\gamma$ is a discount factor, $\gamma \in [0, 1]$
Example: Student MRP

- **Facebook**: 0.9 → Class 1 (R = -1) 0.1 → Sleep (R = 0)
- **Sleep**: 0.5 → Class 2 (R = -2) 0.5 → Facebook (R = -1)
- **Class 1**: 0.5 → Class 2 (R = -2) 0.5 → Class 3 (R = -2)
- **Class 2**: 0.2 → Class 3 (R = -2) 0.8 → Class 1 (R = -2)
- **Class 3**: 0.6 → Pass (R = +10) 0.4 → Class 2 (R = -2)
- **Pass**: 1.0 → Class 3 (R = -2)
- **Pub**: 0.2 → Class 1 (R = -2) 0.4 → Class 3 (R = -2) 0.4 → Class 2 (R = -2)
- **R = +10**: 0.5 → Class 1 (R = -2) 0.5 → Class 3 (R = -2) 0.2 → Class 2 (R = -2)
The *return* $G_t$ is the total discounted reward from time-step $t$.

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

- 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 $\gamma^k R$.
- This values immediate reward above delayed reward.
  - $\gamma$ close to 0 leads to "myopic" evaluation
  - $\gamma$ close to 1 leads to "far-sighted" evaluation
Why discount?

Most Markov reward and decision processes are discounted. Why?

- Mathematically convenient to discount rewards
- Avoids infinite returns in cyclic Markov processes
- Uncertainty about the future may not be fully represented
- If the reward is financial, immediate rewards may earn more interest than delayed rewards
- Animal/human behaviour shows preference for immediate reward
- It is sometimes possible to use \textit{undiscounted} Markov reward processes (i.e. $\gamma = 1$), e.g. if all sequences terminate.
The value function $v(s)$ gives the long-term value of state $s$

**Definition**

The *state value function* $v(s)$ of an MRP is the expected return starting from state $s$

$$v(s) = \mathbb{E} [G_t \mid S_t = s]$$
Example: Student MRP Returns

Sample **returns** for Student MRP:
Starting from $S_1 = C1$ with $\gamma = \frac{1}{2}$

$$G_1 = R_2 + \gamma R_3 + \ldots + \gamma^{T-2} R_T$$

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Value Calculation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1 C2 C3 Pass Sleep</td>
<td>$v_1 = -2 - 2 \times \frac{1}{2} - 2 \times \frac{1}{4} + 10 \times \frac{1}{8}$</td>
<td>$-2.25$</td>
</tr>
<tr>
<td>C1 FB FB C1 C2 Sleep</td>
<td>$v_1 = -2 - 1 \times \frac{1}{2} - 1 \times \frac{1}{4} - 2 \times \frac{1}{8} - 2 \times \frac{1}{16}$</td>
<td>$-3.125$</td>
</tr>
<tr>
<td>C1 C2 C3 Pub C2 C3 Pass Sleep</td>
<td>$v_1 = -2 - 2 \times \frac{1}{2} - 2 \times \frac{1}{4} + 1 \times \frac{1}{8} - 2 \times \frac{1}{16} \ldots$</td>
<td>$-3.41$</td>
</tr>
<tr>
<td>C1 FB FB C1 C2 C3 Pub C1</td>
<td>$v_1 = -2 - 1 \times \frac{1}{2} - 1 \times \frac{1}{4} - 2 \times \frac{1}{8} - 2 \times \frac{1}{16} \ldots$</td>
<td>$-3.20$</td>
</tr>
<tr>
<td>FB FB FB C1 C2 C3 Pub C2 Sleep</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: State-Value Function for Student MRP (1)
Example: State-Value Function for Student MRP (2)
Example: State-Value Function for Student MRP (3)
Bellman Equation for MRPs

The value function can be decomposed into two parts:

- immediate reward $R_{t+1}$
- discounted value of successor state $\gamma v(S_{t+1})$

$$v(s) = \mathbb{E}[G_t \mid S_t = s]$$
$$= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s]$$
$$= \mathbb{E}[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \ldots) \mid S_t = s]$$
$$= \mathbb{E}[R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$
$$= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]$$
Bellman Equation for MRPs (2)

\[ v(s) = \mathbb{E} [R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s] \]

\[ v(s) = R_s + \gamma \sum_{s' \in S} P_{ss'} v(s') \]
Example: Bellman Equation for Student MRP

\[ 4.3 = -2 + 0.6 \times 10 + 0.4 \times 0.8 \]
The Bellman equation can be expressed concisely using matrices,

\[ \mathbf{v} = \mathbf{R} + \gamma \mathbf{P} \mathbf{v} \]

where \( \mathbf{v} \) is a column vector with one entry per state

\[
\begin{bmatrix}
  \mathbf{R}_1 \\
  \vdots \\
  \mathbf{R}_n
\end{bmatrix}
\begin{bmatrix}
  \mathbf{P}_{11} & \ldots & \mathbf{P}_{1n} \\
  \vdots & \ddots & \vdots \\
  \mathbf{P}_{11} & \ldots & \mathbf{P}_{nn}
\end{bmatrix}
\begin{bmatrix}
  \mathbf{v}(1) \\
  \vdots \\
  \mathbf{v}(n)
\end{bmatrix}
\]
Solving the Bellman Equation

- The Bellman equation is a linear equation
- It can be solved directly:

\[ v = R + \gamma P v \]

\[ (I - \gamma P) v = R \]

\[ v = (I - \gamma P)^{-1} R \]

- Computational complexity is \( O(n^3) \) for \( n \) states
- Direct solution only possible for small MRPs
- There are many iterative methods for large MRPs, e.g.
  - Dynamic programming
  - Monte-Carlo evaluation
  - Temporal-Difference learning
A Markov decision process (MDP) is a Markov reward process with decisions. It is an environment in which all states are Markov.

**Definition**

A *Markov Decision Process* is a tuple \( \langle S, A, P, R, \gamma \rangle \)

- \( S \) is a finite set of states
- \( A \) is a finite set of actions
- \( P \) is a state transition probability matrix, 
  \[
  P_{ss'}^a = \mathbb{P} [S_{t+1} = s' \mid S_t = s, A_t = a]
  \]
- \( R \) is a reward function, 
  \[
  R_s^a = \mathbb{E} [R_{t+1} \mid S_t = s, A_t = a]
  \]
- \( \gamma \) is a discount factor \( \gamma \in [0, 1] \).
Example: Student MDP
Policies (1)

Definition

A policy $\pi$ is a distribution over actions given states,

$$\pi(a|s) = P[A_t = a | S_t = s]$$

- A policy fully defines the behaviour of an agent
- MDP policies depend on the current state (not the history)
- i.e. Policies are stationary (time-independent), $A_t \sim \pi(\cdot|S_t)$, $\forall t > 0$
Policies (2)

- Given an MDP \( \mathcal{M} = \langle S, A, \mathcal{P}, \mathcal{R}, \gamma \rangle \) and a policy \( \pi \)
- The state sequence \( S_1, S_2, \ldots \) is a Markov process \( \langle S, \mathcal{P}^\pi \rangle \)
- The state and reward sequence \( S_1, R_2, S_2, \ldots \) is a Markov reward process \( \langle S, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle \)
- where

\[
\mathcal{P}^\pi_{s,s'} = \sum_{a \in A} \pi(a|s) \mathcal{P}^a_{ss'} \\
\mathcal{R}^\pi_s = \sum_{a \in A} \pi(a|s) \mathcal{R}^a_s
\]
Value Function

**Definition**

The *state-value function* $v_\pi(s)$ of an MDP is the expected return starting from state $s$, and then following policy $\pi$

$$v_\pi(s) = \mathbb{E}_\pi [G_t \mid S_t = s]$$

**Definition**

The *action-value function* $q_\pi(s, a)$ is the expected return starting from state $s$, taking action $a$, and then following policy $\pi$

$$q_\pi(s, a) = \mathbb{E}_\pi [G_t \mid S_t = s, A_t = a]$$
Example: State-Value Function for Student MDP

\( v_{\pi}(s) \) for \( \pi(a|s) = 0.5, \gamma = 1 \)
The state-value function can again be decomposed into immediate reward plus discounted value of successor state,

$$v_\pi(s) = \mathbb{E}_\pi [R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

The action-value function can similarly be decomposed,

$$q_\pi(s, a) = \mathbb{E}_\pi [R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$
Bellman Expectation Equation for $V^\pi$

$$v_\pi(s) \leftarrow s$$

$$q_\pi(s, a) \leftarrow a$$

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) q_\pi(s, a)$$
Bellman Expectation Equation for $Q^\pi$

$$q_\pi(s, a) \leftarrow s, a$$

$$v_\pi(s') \leftarrow s'$$

$$q_\pi(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s')$$
Bellman Expectation Equation for $v_\pi$ (2)

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \left( R^a_s + \gamma \sum_{s' \in S} P^a_{ss'} v_\pi(s') \right)$$
Bellman Expectation Equation for $q_\pi$ (2)

$$q_\pi(s, a) = R_s + \gamma \sum_{s' \in S} \sum_{a' \in A} P^a_{ss'} \pi(a' | s') q_\pi(s', a')$$
Example: Bellman Expectation Equation in Student MDP

7.4 = 0.5 \times (1 + 0.2 \times -1.3 + 0.4 \times 2.7 + 0.4 \times 7.4) + 0.5 \times 10
The Bellman expectation equation can be expressed concisely using the induced MRP,

\[ v_\pi = R^{\pi} + \gamma P^{\pi} v_\pi \]

with direct solution

\[ v_\pi = \left( I - \gamma P^{\pi} \right)^{-1} R^{\pi} \]
The optimal state-value function $v^*(s)$ is the maximum value function over all policies $\pi$:

$$v^*(s) = \max_{\pi} v_{\pi}(s)$$

The optimal action-value function $q^*(s, a)$ is the maximum action-value function over all policies $\pi$:

$$q^*(s, a) = \max_{\pi} q_{\pi}(s, a)$$

- The optimal value function specifies the best possible performance in the MDP.
- An MDP is “solved” when we know the optimal value function.
Example: Optimal Value Function for Student MDP

\[ v^*(s) \text{ for } \gamma = 1 \]
Example: Optimal Action-Value Function for Student MDP

Facebook
\( R = -1 \)
\( q^* = 5 \)

Quit
\( R = 0 \)
\( q^* = 6 \)

Study
\( R = -2 \)
\( q^* = 6 \)

Sleep
\( R = 0 \)
\( q^* = 0 \)

Facebook
\( R = -1 \)
\( q^* = 5 \)

Study
\( R = -2 \)
\( q^* = 8 \)

Study
\( R = +10 \)
\( q^* = 10 \)

Pub
\( R = +1 \)
\( q^* = 8.4 \)
Define a partial ordering over policies

\[ \pi \geq \pi' \text{ if } v_\pi(s) \geq v_{\pi'}(s), \forall s \]

**Theorem**

*For any Markov Decision Process*

- There exists an optimal policy \( \pi_* \) 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, 
  \[ q_{\pi_*}(s, a) = q_*(s, a) \]
Finding an Optimal Policy

An optimal policy can be found by maximising over $q_\ast(s, a)$,

$$\pi_\ast(a|s) = \begin{cases} 
1 & \text{if } a = \arg\max_{a \in A} q_\ast(s, a) \\
0 & \text{otherwise}
\end{cases}$$

- There is always a deterministic optimal policy for any MDP
- If we know $q_\ast(s, a)$, we immediately have the optimal policy
Example: Optimal Policy for Student MDP

\[ \pi^*(a|s) \text{ for } \gamma = 1 \]

\[ q^* = 5 \]
\[ q^* = 6 \]
\[ q^* = 6 \]
\[ q^* = 5 \]
\[ q^* = 8 \]
\[ q^* = 0 \]
\[ q^* = 10 \]
\[ q^* = 8.4 \]
The optimal value functions are recursively related by the Bellman optimality equations:

$$v^*_*(s) = \max_a q^*_*(s, a)$$
Bellman Optimality Equation for $Q^*$

$$q^*(s, a) = R_s + \gamma \sum_{s' \in S} P_{sa} v^*(s')$$
Bellman Optimality Equation for $V^*$ (2)

$$v_*(s) = \max_a R^a_s + \gamma \sum_{s' \in S} P^a_{ss'} v_*(s')$$
Bellman Optimality Equation for $Q^*$ (2)

$$q^*(s, a) \leftarrow s, a$$

$$q^*(s', a') \leftarrow a'$$

$$q^*(s, a) = R_s + \gamma \sum_{s' \in S} P^{a}_{ss'} \max_{a'} q^*(s', a')$$
Example: Bellman Optimality Equation in Student MDP

\[ 6 = \max \{-2 + 8, -1 + 6\} \]
Solving the Bellman Optimality Equation

- Bellman Optimality Equation is non-linear
- No closed form solution (in general)
- Many iterative solution methods
  - Value Iteration
  - Policy Iteration
  - Q-learning
  - Sarsa
Extensions to MDPs

- Infinite and continuous MDPs
- Partially observable MDPs
- Undiscounted, average reward MDPs
The following extensions are all possible:

- **Countably infinite state and/or action spaces**
  - Straightforward

- **Continuous state and/or action spaces**
  - Closed form for linear quadratic model (LQR)

- **Continuous time**
  - Requires partial differential equations
  - Hamilton-Jacobi-Bellman (HJB) equation
  - Limiting case of Bellman equation as time-step → 0
A Partially Observable Markov Decision Process is an MDP with hidden states. It is a hidden Markov model with actions.

Definition

A **POMDP** is a tuple $\langle S, A, O, P, R, Z, \gamma \rangle$

- $S$ is a finite set of states
- $A$ is a finite set of actions
- $O$ is a finite set of observations
- $P$ is a state transition probability matrix,
  $$P_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$
- $R$ is a reward function,
  $$R_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$
- $Z$ is an observation function,
  $$Z_{s'o}^a = \mathbb{P}[O_{t+1} = o \mid S_{t+1} = s', A_t = a]$$
- $\gamma$ is a discount factor $\gamma \in [0, 1]$. 
Belief States

**Definition**

A *history* $H_t$ is a sequence of actions, observations and rewards,

$$H_t = A_0, O_1, R_1, ..., A_{t-1}, O_t, R_t$$

**Definition**

A *belief state* $b(h)$ is a probability distribution over states, conditioned on the history $h$

$$b(h) = (\mathbb{P} [S_t = s^1 \mid H_t = h], ..., \mathbb{P} [S_t = s^n \mid H_t = h])$$
The history $H_t$ satisfies the Markov property

The belief state $b(H_t)$ satisfies the Markov property

A POMDP can be reduced to an (infinite) history tree

A POMDP can be reduced to an (infinite) belief state tree
An ergodic Markov process is
- **Recurrent**: each state is visited an infinite number of times
- **Aperiodic**: each state is visited without any systematic period

**Theorem**

An ergodic Markov process has a limiting stationary distribution \( d^\pi(s) \) with the property

\[
 d^\pi(s) = \sum_{s' \in S} d^\pi(s') P_{s's}
\]
An MDP is ergodic if the Markov chain induced by any policy is ergodic.

For any policy $\pi$, an ergodic MDP has an average reward per time-step $\rho^\pi$ that is independent of start state.

$$\rho^\pi = \lim_{T \to \infty} \frac{1}{T} \mathbb{E} \left[ \sum_{t=1}^{T} R_t \right]$$
The value function of an undiscounted, ergodic MDP can be expressed in terms of average reward.

\[ \tilde{v}_\pi(s) = \mathbb{E}_\pi \left[ \sum_{k=1}^{\infty} (R_{t+k} - \rho^\pi) \mid S_t = s \right] \]

There is a corresponding average reward Bellman equation,

\[ \tilde{v}_\pi(s) = \mathbb{E}_\pi \left[ (R_{t+1} - \rho^\pi) + \sum_{k=1}^{\infty} (R_{t+k+1} - \rho^\pi) \mid S_t = s \right] \]

\[ = \mathbb{E}_\pi \left[ (R_{t+1} - \rho^\pi) + \tilde{v}_\pi(S_{t+1}) \mid S_t = s \right] \]
Questions?

*The only stupid question is the one you were afraid to ask but never did.*

-Rich Sutton
KEEP CALM AND MAKE A PLAN