Machine Learning I
80-629A

Apprentissage Automatique I
80-629

Sequential Decision Making II
— Week #12
Introduction to Reinforcement Learning
Brief recap

- Markov Decision Processes (MDP)
- Offer a framework for sequential decision making
  \[ \langle A, S, P, R, \gamma \rangle \]
- Goal: find the optimal policy
- Dynamic programming and several algorithms (e.g., VI, PI)
From MDPs to RL

• In MDPs we assume that we know
From MDPs to RL

- In MDPs we assume that we know
  1. Transition probabilities: $P(s' | s, a)$
From MDPs to RL

• In MDPs we assume that we know
  1. Transition probabilities: $P(s' \mid s, a)$
  2. Reward function: $R(s)$
From MDPs to RL

• In MDPs we assume that we know

  1. Transition probabilities: $P(s' | s, a)$
  2. Reward function: $R(s)$

• RL is more general
From MDPs to RL

- In MDPs we assume that we know
  1. Transition probabilities: \( P(s' \mid s, a) \)
  2. Reward function: \( R(s) \)

- RL is more general
  - In RL both are typically unknown
From MDPs to RL

• In MDPs we assume that we know
  1. Transition probabilities: $P(s' \mid s, a)$
  2. Reward function: $R(s)$

• RL is more general
  • In RL both are typically unknown
  • RL agents navigate the world to gather this information
Experience

A. Supervised Learning:
   - Given fixed dataset
   - Goal: maximize objective on test set (population)

B. Reinforcement Learning
   - Collect data as agent interacts with the world
   - Goal: maximize sum of rewards
Example

Supervised Learning

Don't touch your tongue will get stuck.

Slide adapted from Pascal Poupart

Example

Supervised Learning

Reinforcement Learning

Don't touch your tongue will get stuck.

Slide adapted from Pascal Poupart
Example

Supervised Learning

Reinforcement Learning

Don't touch your tongue will get stuck.

Slide adapted from Pascal Poupart

RL applications
RL applications

• **Key:** decision making over time, uncertain environments
RL applications

• **Key**: decision making over time, uncertain environments

• Robot navigation: Self-driving cars, helicopter control
RL applications

- **Key**: decision making over time, uncertain environments
- Robot navigation: Self-driving cars, helicopter control
- Interactive systems: recommender systems, chatbots
RL applications

- **Key:** decision making over time, uncertain environments
- Robot navigation: Self-driving cars, helicopter control
- Interactive systems: recommender systems, chatbots
- Game playing: Backgammon, go
RL applications

- **Key**: decision making over time, uncertain environments
- Robot navigation: Self-driving cars, helicopter control
- Interactive systems: recommender systems, chatbots
- Game playing: Backgammon, go
- Healthcare: monitoring systems
Reinforcement learning and recommender systems

- Most users have multiple interactions with the system of time

- Making recommendations over time can be advantageous (e.g., you could better explore one's preferences)

- States: Some representation of user preferences (e.g., previous items they consumed)

- Actions: what to recommend (item 1, item 2, item 3, ...)

- Reward:
  - + user consumes the recommendation
  - - user does not consume the recommendation
Challenges of reinforcement learning
Challenges of reinforcement learning

- Credit assignment problem: which action(s) should be credited for obtaining a reward

- A series of actions (getting coffee from cafeteria)

- A small number of actions several time steps ago may be important (test taking: study before, getting grade long after)
Challenges of reinforcement learning

- Credit assignment problem: which action(s) should be credited for obtaining a reward

- A series of actions (getting coffee from cafeteria)

- A small number of actions several time steps ago may be important (test taking: study before, getting grade long after)

- Exploration/Exploitation tradeoff: As agent interacts should it exploit its current knowledge (exploitation) or seek out additional information (exploration)
Algorithms for Reinforcement Learning
Algorithm
Algorithm

- Input: an environment
- actions, states, discount factor
- starting state, method for obtaining next state
Algorithm

- Input: an environment
  - actions, states, discount factor
  - starting state, method for obtaining next state
- Output: an optimal policy
Algorithm

- Input: an environment
  - actions, states, discount factor
  - starting state, method for obtaining next state
- Output: an optimal policy
- In practice: need a simulator or a real environment for your agent to interact
Algorithms for RL

- Two main classes of approach
Algorithms for RL

• Two main classes of approach

1. Model-based

• Learns a model of the transition and uses it to optimize a policy given the model $P(s' | s, a)$
Algorithms for RL

- Two main classes of approach

1. Model-based
   - Learns a model of the transition and uses it to optimize a policy given the model $P(s' | s, a)$

2. Model-free
   - Learns an optimal policy without explicitly learning transitions $\pi$
Monte Carlo Methods
Monte Carlo Methods

- Model-free
Monte Carlo Methods

- Model-free
- Assume the environment is episodic
  - Think of playing a card game (like poker). An episode is a hand.
  - Updates the policy after each episode
Monte Carlo Methods

- Model-free
- Assume the environment is episodic
  - Think of playing a card game (like poker). An episode is a hand.
  - Updates the policy after each episode
- Intuition
  - Experience many episodes
    - Play many hands (of poker)
  - Average the rewards received at each state
    - What is the proportion of wins given your current cards
Prediction vs. control

1. Prediction: evaluate a given policy

2. Control: Learn a policy
   
   - Sometimes also called
     
     - passive (prediction)
     
     - active (control)
First-visit Monte Carlo

• Given a fixed policy (prediction)

• Calculate the value function $V(s)$ for each state

• Converges to $V_{\pi}(s)$ as the number of visits to each state goes to infinity

[First-visit MC prediction, for estimating $V \approx v_{\pi}$]

Initialize:
- $\pi$ ← policy to be evaluated
- $V$ ← an arbitrary state-value function
- $Returns(s)$ ← an empty list, for all $s \in S$

Repeat forever:
- Generate an episode using $\pi$
- For each state $s$ appearing in the episode:
  - $G$ ← the return that follows the first occurrence of $s$
  - Append $G$ to $Returns(s)$
  - $V(s) ← average(Returns(s))$

[Sutton & Barto, RL Book, Ch 5]
First-visit Monte Carlo

• Given a fixed policy (prediction)

• Calculate the value function $V(s)$ for each state

• Converges to $V_\pi(s)$ as the number of visits to each state goes to infinity

$$V(s) = \max_{a_t} \left\{ R(s_t) + \gamma \sum_{s_{t+1}} P(s_{t+1} | s_t, a_t) V(s_{t+1}) \right\}$$

[Note: The formula is a simplified version of the one from Sutton & Barto, RL Book, Ch 5]
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: $[1, \rightarrow]$
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

**Episode:** [1, ←]
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: $[1, \rightarrow]$
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: $[1, \rightarrow] \rightarrow [2, \rightarrow]$
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: $[1, \rightarrow] \rightarrow [2, \rightarrow]$
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: $\langle 1, \rightarrow \rangle \rightarrow \langle 2, \rightarrow \rangle \rightarrow \langle 3, \downarrow \rangle$
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: 

$\begin{align*}
\{1, \rightarrow\} & \rightarrow \{2, \rightarrow\} & \rightarrow \{3, \downarrow\}
\end{align*}$
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: $[1, \rightarrow] \rightarrow [2, \rightarrow] \rightarrow [3, \downarrow] \rightarrow [7, \downarrow]$
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: $[1, \rightarrow] \rightarrow [2, \rightarrow] \rightarrow [3, \downarrow] \rightarrow [7, \downarrow]$
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy \( \pi \) is given (gray arrows)

Episode: \( (1, \rightarrow) \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow) \rightarrow (7, \downarrow) \rightarrow (6, \rightarrow) \)
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: $(1, \rightarrow) \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow) \rightarrow (7, \downarrow) \rightarrow (6, \rightarrow)$
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: $[1, \rightarrow] \rightarrow [2, \rightarrow] \rightarrow [3, \downarrow] \rightarrow [7, \downarrow] \rightarrow [6, \rightarrow] \rightarrow [7, \downarrow]$
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: $(1, \rightarrow) \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow) \rightarrow (7, \downarrow) \rightarrow (6, \rightarrow) \rightarrow (7, \downarrow)$
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: $[1, \rightarrow] \rightarrow [2, \rightarrow] \rightarrow [3, \downarrow] \rightarrow [7, \downarrow] \rightarrow [6, \rightarrow] \rightarrow [7, \downarrow] \rightarrow [10, \downarrow]$
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: $(1, \rightarrow) \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow) \rightarrow (7, \downarrow) \rightarrow (6, \rightarrow) \rightarrow (7, \downarrow) \rightarrow (10, \downarrow)$

First-visit MC prediction, for estimating $V \approx v_\pi$

Initialize:
- $\pi$ ← policy to be evaluated
- $V$ ← an arbitrary state-value function
- $Returns(s)$ ← an empty list, for all $s \in S$

Repeat forever:
- Generate an episode using $\pi$
- For each state $s$ appearing in the episode:
  - $G$ ← the return that follows the first occurrence of $s$
  - Append $G$ to $Returns(s)$
- $V(s) ← \text{average}(Returns(s))$
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: $[1, \rightarrow] \rightarrow [2, \rightarrow] \rightarrow [3, \downarrow] \rightarrow [7, \downarrow] \rightarrow [6, \rightarrow] \rightarrow [7, \downarrow] \rightarrow [10, \downarrow] \rightarrow [13, \downarrow]$
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: $(1, \rightarrow) \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow) \rightarrow (7, \downarrow) \rightarrow (6, \rightarrow) \rightarrow (7, \downarrow) \rightarrow (10, \downarrow) \rightarrow (13, \downarrow)$
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: $\langle 1, \rightarrow \rangle \rightarrow \langle 2, \rightarrow \rangle \rightarrow \langle 3, \downarrow \rangle \rightarrow \langle 7, \downarrow \rangle \rightarrow \langle 6, \rightarrow \rangle \rightarrow \langle 7, \downarrow \rangle \rightarrow \langle 10, \downarrow \rangle \rightarrow \langle 13, \downarrow \rangle \rightarrow \langle 17, \rightarrow \rangle$
Example: grid world

- Start state is top-left (start of episode)

- Bottom right is absorbing (end of episode)

- Policy $\pi$ is given (gray arrows)

Episode: $1 \rightarrow 2 \rightarrow 3 \downarrow \rightarrow 7 \downarrow \rightarrow 6 \rightarrow 7 \rightarrow 10 \rightarrow 13 \rightarrow 17$
### Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
</tbody>
</table>

Episode: $(1, \rightarrow) \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow) \rightarrow (7, \downarrow) \rightarrow (6, \rightarrow) \rightarrow (7, \downarrow) \rightarrow (10, \downarrow) \rightarrow (13, \downarrow) \rightarrow (17, \rightarrow)$

---

First-visit MC prediction, for estimating $V \approx v_{\pi}$

**Initialize:**

1. $\pi \leftarrow$ policy to be evaluated
2. $V \leftarrow$ an arbitrary state-value function
3. $Returns(s) \leftarrow$ an empty list, for all $s \in S$

**Repeat forever:**

For each state $s$ appearing in the episode:

1. $G \leftarrow$ the return that follows the first occurrence of $s$
2. Append $G$ to $Returns(s)$
3. $V(s) \leftarrow$ average($Returns(s)$)
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: 1 $\rightarrow$ 2 $\rightarrow$ 3 $\rightarrow$ 7 $\downarrow$ $\rightarrow$ 6 $\rightarrow$ 7 $\downarrow$ $\rightarrow$ 10 $\downarrow$ $\rightarrow$ 13 $\downarrow$ $\rightarrow$ 17 $\rightarrow$ 18
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: 
$\{1, \rightarrow\} \rightarrow \{2, \rightarrow\} \rightarrow \{3, \downarrow\} \rightarrow \{7, \downarrow\} \rightarrow \{6, \rightarrow\} \rightarrow \{7, \downarrow\} \rightarrow \{10, \downarrow\} \rightarrow \{13, \downarrow\} \rightarrow \{17, \rightarrow\}$

For state 7:
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: $(1, \leftarrow) \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow) \rightarrow (7, \downarrow) \rightarrow (6, \rightarrow) \rightarrow (7, \downarrow) \rightarrow (10, \uparrow) \rightarrow (13, \downarrow) \rightarrow (17, \rightarrow)$

For state 7: 
\[
\text{return}(7) = \gamma R(6) + \gamma^2 R(7) + \gamma^3 R(10) + \gamma^4 R(13) + \gamma^5 R(17) + \gamma^6 R(18) \\
= \gamma^6 10
\]
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode:

$\begin{align*}
(1, \leftarrow) & \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow) \rightarrow (7, \downarrow) \rightarrow (6, \rightarrow) \rightarrow (7, \downarrow) \rightarrow (10, \uparrow) \rightarrow (13, \downarrow) \rightarrow (17, \rightarrow)
\end{align*}$

For state 7: \[
\text{return}(7) = \gamma R(6) + \gamma^2 R(7) + \gamma^3 R(10) + \gamma^4 R(13) + \gamma^5 R(17) + \gamma^6 R(18)
\]
\[
= \gamma^6 10
\]
Example: grid world

- Start state is top-left (start of episode)
- Bottom right is absorbing (end of episode)
- Policy $\pi$ is given (gray arrows)

Episode: \( (1, \rightarrow) \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow) \rightarrow (7, \downarrow) \rightarrow (6, \rightarrow) \rightarrow (7, \downarrow) \rightarrow (10, \downarrow) \rightarrow (13, \downarrow) \rightarrow (17, \rightarrow) \)

For state 7: \[ \text{return}(7) = \gamma R(6) + \gamma^2 R(7) + \gamma^3 R(10) + \gamma^4 R(13) + \gamma^5 R(17) + \gamma^6 R(18) \]
\[ = \gamma^6 10 \]

\[ V(7) = \gamma^6 \times 10 \]
Summary

- Introduced terminology:
  - model based, model-free
- First algorithm for policy evaluation (First-visit MC)
- Compared to MDPs
  - We the agent now has to explore the world to evaluate its value function
Algorithms for RL Control
Q-value function for control

- We know about state-value functions $V(s)$
Q-value function for control

• We know about state-value functions $V(s)$

• If state transitions are known then they can be used to derive an optimal policy [recall value iteration]:

$$
\pi^*(s) = \arg \max_a \left\{ R(s) + \gamma \sum_{s'} P(s' | s, a) V^*(s') \right\} \quad \forall s
$$
Q-value function for control

• We know about state-value functions $V(s)$

• If state transitions are known then they can be used to derive an optimal policy [recall value iteration]:

$$
\pi^*(s) = \arg \max_a \left\{ R(s) + \gamma \sum_{s'} P(s' | s, a) V^*(s') \right\} \quad \forall s
$$

• When state transitions are unknown what can we do?
Q-value function for control

• We know about state-value functions $V(s)$

  • If state transitions are known then they can be used to derive an optimal policy [recall value iteration]:

    $$\pi^*(s) = \arg \max_a \left\{ R(s) + \gamma \sum_{s'} P(s' \mid s, a) V^*(s') \right\} \ \forall s$$

  • When state transitions are unknown what can we do?

    • $Q(s,a)$ the value function of a (state,action) pair

    $$\pi^*(s) = \arg \max_a \{ Q^*(s, a) \} \ \forall s$$
Monte Carlo ES (control)

[Sutton & Barto, RL Book, Ch.5]
Monte Carlo ES (control)

- Strong reasons to believe that it converges to the optimal policy

- “Exploring starts” requirement may be unrealistic

[Sutton & Barto, RL Book, Ch.5]
Learning without “exploring starts”

• “Exploring starts” insures that all states can be visited regardless of the policy
  
  • (Specific policy may not visit all states)
  
  • Unrealistic in real-world settings
Learning without “exploring starts”

- “Exploring starts” insures that all states can be visited regardless of the policy
  - (Specific policy may not visit all states)
  - Unrealistic in real-world settings
- Solution: inject some uncertainty in the policy
Monte Carlo without exploring starts (on policy)

On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi^*$

- Initialize, for all $s \in S$, $a \in A(s)$:
  - $Q(s, a) \leftarrow$ arbitrary
  - $\text{Returns}(s, a) \leftarrow$ empty list
  - $\pi(a|s) \leftarrow$ an arbitrary $\varepsilon$-soft policy

- Repeat forever:
  1. Generate an episode using $\pi$
  2. For each pair $s, a$ appearing in the episode:
     - $G \leftarrow$ the return that follows the first occurrence of $s, a$
     - Append $G$ to $\text{Returns}(s, a)$
     - $Q(s, a) \leftarrow \text{average}(\text{Returns}(s, a))$
  3. For each $s$ in the episode:
     - $A^* \leftarrow \arg\max_a Q(s, a)$ (with ties broken arbitrarily)
     - For all $a \in A(s)$:
       - $\pi(a|s) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|A(s)| & \text{if } a = A^* \\ \varepsilon/|A(s)| & \text{if } a \neq A^* \end{cases}$

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi^*$

- Initialize, for all $s \in S$, $a \in A(s)$:
  - $Q(s, a) \leftarrow$ arbitrary
  - $\pi(s) \leftarrow$ arbitrary
  - $\text{Returns}(s, a) \leftarrow$ empty list

- Repeat forever:
  1. Choose $S_0 \in S$ and $A_0 \in A(S_0)$ s.t. all pairs have probability $> 0$
  2. Generate an episode starting from $S_0, A_0$, following $\pi$
  3. For each pair $s, a$ appearing in the episode:
     - $G \leftarrow$ the return that follows the first occurrence of $s, a$
     - Append $G$ to $\text{Returns}(s, a)$
     - $Q(s, a) \leftarrow \text{average}(\text{Returns}(s, a))$
  4. For each $s$ in the episode:
     - $\pi(s) \leftarrow \arg\max_a Q(s, a)$

[Sutton & Barto, RL Book, Ch.5]
Monte Carlo without exploring starts (on policy)

On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi^*$

Initialize, for all $s \in S$, $a \in A(s)$:
- $Q(s, a) \leftarrow$ arbitrary
- $\text{Returns}(s, a) \leftarrow$ empty list
- $\pi(a|s) \leftarrow$ an arbitrary $\varepsilon$-soft policy

Repeat forever:
- (a) Generate an episode using $\pi$
- (b) For each pair $s, a$ appearing in the episode:
  - $G \leftarrow$ the return that follows the first occurrence of $s, a$
  - Append $G$ to $\text{Returns}(s, a)$
  - $Q(s, a) \leftarrow \text{average} \left( \text{Returns}(s, a) \right)$
- (c) For each $s$ in the episode:
  - $A^* \leftarrow \arg \max_a Q(s, a)$ (with ties broken arbitrarily)
  - For all $a \in A(s)$:
    - $\pi(a|s) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon / |A(s)| & \text{if } a = A^* \\ \varepsilon / |A(s)| & \text{if } a \neq A^* \end{cases}$

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi^*$

Initialize, for all $s \in S$, $a \in A(s)$:
- $Q(s, a) \leftarrow$ arbitrary
- $\pi(s) \leftarrow$ arbitrary
- $\text{Returns}(s, a) \leftarrow$ empty list

Repeat forever:
- Choose $S_0 \in S$ and $A_0 \in A(S_0)$ s.t. all pairs have probability $> 0$
- Generate an episode starting from $S_0, A_0$, following $\pi$
- For each pair $s, a$ appearing in the episode:
  - $G \leftarrow$ the return that follows the first occurrence of $s, a$
  - Append $G$ to $\text{Returns}(s, a)$
  - $Q(s, a) \leftarrow \text{average} \left( \text{Returns}(s, a) \right)$
- For each $s$ in the episode:
  - $\pi(s) \leftarrow \arg \max_a Q(s, a)$

[Sutton & Barto, RL Book, Ch.5]
Monte Carlo without exploring starts (on policy)

[Sutton & Barto, RL Book, Ch.5]
Monte Carlo without exploring starts (on policy)

- Policy value cannot decrease
  \[ v_{\pi'}(s) \geq v_{\pi}(s), \forall s \in S \]

\[\begin{align*}
\text{On-policy first-visit MC control (for } \varepsilon\text{-soft policies), estimates } \pi \approx \pi^*.
\text{Initialize, for all } s \in S, a \in A(s):
\quad Q(s, a) \leftarrow \text{arbitrary}
\quad Returns(s, a) \leftarrow \text{empty list}
\quad \pi(a|s) \leftarrow \text{an arbitrary } \varepsilon\text{-soft policy}

\text{Repeat forever:}
\quad (a) \text{ Generate an episode using } \pi
\quad (b) \text{ For each pair } s, a \text{ appearing in the episode:}
\quad \quad G \leftarrow \text{return that follows the first occurrence of } s, a
\quad \quad \text{Append } G \text{ to } Returns(s, a)
\quad \quad Q(s, a) \leftarrow \text{average}(Returns(s, a))
\quad \quad (c) \text{ For each } s \text{ in the episode:}
\quad \quad \quad A^* \leftarrow \text{arg max}_a Q(s, a)
\quad \quad \quad \text{For all } a \in A(s):
\quad \quad \quad \quad \pi(a|s) \leftarrow \begin{cases} 
1 - \varepsilon + \varepsilon / |A(s)| & \text{if } a = A^* \\
\varepsilon / |A(s)| & \text{if } a \neq A^*
\end{cases}

\text{Monte Carlo ES (Exploring Starts), for estimating } \pi \approx \pi^*.
\text{Initialize, for all } s \in S, a \in A(s):
\quad Q(s, a) \leftarrow \text{arbitrary}
\quad \pi(s) \leftarrow \text{arbitrary}
\quad Returns(s, a) \leftarrow \text{empty list}

\text{Repeat forever:}
\quad \text{Choose } S_0 \in S \text{ and } A_0 \in A(S_0) \text{ s.t. all pairs have probability } > 0
\quad \text{Generate an episode starting from } S_0, A_0, \text{ following } \pi
\quad \text{For each pair } s, a \text{ appearing in the episode:}
\quad \quad G \leftarrow \text{return that follows the first occurrence of } s, a
\quad \quad \text{Append } G \text{ to } Returns(s, a)
\quad \quad Q(s, a) \leftarrow \text{average}(Returns(s, a))
\quad \quad \pi(s) \leftarrow \text{arg max}_a Q(s, a)
\end{align*}\]

\[\pi \triangleright \pi' \text{ : policy at current step}
\pi' \triangleright \text{ : policy at next step}\]

[Sutton & Barto, RL Book, Ch.5]
Monte-Carlo methods
summary

• Allow a policy to be learned through interactions

  • (Does not learn transitions)

• States are effectively treated as being independent

  • Focus on a subset of states (e.g., states for which playing optimally is of particular importance)

• Episodic (with or without exploring starts)
Temporal Difference (TD) Learning
Temporal Difference (TD) Learning

- One of the “central ideas of RL” [Sutton & Barto, RL book]
Temporal Difference (TD) Learning

- One of the “central ideas of RL” [Sutton & Barto, RL book]

- Monte Carlo methods

\[ V'(s_t) = V(s_t) + \alpha [G_t - V(s_t)] \]

\[ G_t = \sum_{t} \gamma^t R(s_t) \]

Observed returned: \( G_t \)

Step size: \( \alpha \)
Temporal Difference (TD) Learning

- One of the “central ideas of RL” [Sutton & Barto, RL book]

- Monte Carlo methods

\[ V'(s_t) = V(s_t) + \alpha \left( G_t - V(s_t) \right) \]

Observed returned:

\[ G_t = \sum_{t} \gamma^t R(s_t) \]

\[ \alpha \]

Step size

First-visit MC prediction, for estimating \( V(s_t) \):

Initialize:
- \( \pi \) policy to be evaluated
- \( V \) an arbitrary state-value function
- \( Returns(s) \) an empty list, for all \( s \in S \)

Repeat forever:
- Generate an episode using \( \pi \)
- For each state \( s \) appearing in the episode:
  - \( G \) the return that follows the first
  - Append \( G \) to \( Returns(s) \)
- \( V(s) \) average (\( Returns(s) \)
Temporal Difference (TD) Learning

- One of the “central ideas of RL” [Sutton & Barto, RL book]

- Monte Carlo methods

\[ V'(s_t) = V(s_t) + \alpha [G_t - V(s_t)] \]

- TD(0)

  - updates “instantly”

\[ G_t = \sum_{t}^{T} \gamma^t R(s_t) \]
Temporal Difference (TD) Learning

- One of the “central ideas of RL” [Sutton & Barto, RL book]

- Monte Carlo methods

  \[ V'(s_t) = V(s_t) + \alpha \left[ G_t - V(s_t) \right] \]

- TD(0)

  - updates “instantly”

  \[ V'(s_t) = V(s_t) + \alpha [R(s_t) + \gamma V(s_{t+1}) - V(s_t)] \approx G_t \]
TD(0) for prediction

Table: Tabular TD(0) for estimating $v_\pi$

- **Input**: the policy $\pi$ to be evaluated
- **Initialize** $V(s)$ arbitrarily (e.g., $V(s) = 0$, for all $s \in S^+$)
- **Repeat** (for each episode):
  - **Initialize** $S$
  - **Repeat** (for each step of episode):
    - $A \leftarrow$ action given by $\pi$ for $S$
    - Take action $A$, observe $R$, $S'$
    - $V(S) \leftarrow V(S) + \alpha [R + \gamma V(S') - V(S)]$
    - $S \leftarrow S'$
  - until $S$ is terminal

[Sutton & Barto, RL Book, Ch.6]
TD for control

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in S^+, a \in A(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
- Initialize $S$
- Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
- Loop for each step of episode:
  - Take action $A$, observe $R, S'$
  - Choose $A'$ from $S'$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
  - $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)]$
  - $S \leftarrow S'$; $A \leftarrow A'$
- until $S$ is terminal

Tabular TD(0) for estimating $v_\pi$

Input: the policy $\pi$ to be evaluated
Initialize $V(s)$ arbitrarily (e.g., $V(s) = 0$, for all $s \in S^+$)
Repeat (for each episode):
- Initialize $S$
- Repeat (for each step of episode):
  - $A \leftarrow$ action given by $\pi$ for $S$
  - Take action $A$, observe $R, S'$
  - $V(S) \leftarrow V(S) + \alpha [R + \gamma V(S') - V(S)]$
  - $S \leftarrow S'$
- until $S$ is terminal
Comparing TD and MC

• MC requires going through full episodes before updating the value function. Episodic.
  • Converges to the optimal solution

• TD updates each V(s) after each transition. Online.
  • Converges to the optimal solution (some conditions on $\alpha$)
  • Empirically TD methods tend to converge faster
Q-learning for control

Q-learning (off-policy TD control) for estimating \( \pi \approx \pi_* \)

1. Initialize \( Q(s,a) \), for all \( s \in S, a \in A(s) \), arbitrarily, and \( Q(terminal-state, \cdot) = 0 \)
2. Repeat (for each episode):
   1. Initialize \( S \)
   2. Repeat (for each step of episode):
      1. Choose \( A \) from \( S \) using policy derived from \( Q \) (e.g., \( \epsilon \)-greedy)
      2. Take action \( A \), observe \( R, S' \)
      3. \( Q(S,A) \leftarrow Q(S,A) + \alpha [R + \gamma \max_a Q(S',a) - Q(S,A)] \)
      4. \( S \leftarrow S' \)
   3. until \( S \) is terminal

[Sutton & Barto, RL Book, Ch.6]
Q-learning for control

Q-learning (off-policy TD control) for estimating \( \pi \approx \pi_* \)

1. Initialize \( Q(s, a) \), for all \( s \in S, a \in A(s) \), arbitrarily, and \( Q(terminal-state, \cdot) = 0 \)
2. Repeat (for each episode):
   - Initialize \( S \)
   - Repeat (for each step of episode):
     - Choose \( A \) from \( S \) using policy derived from \( Q \) (e.g., \( \epsilon \)-greedy)
     - Take action \( A \), observe \( R, S' \)
     - \( Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)] \)
     - \( S \leftarrow S' \)
   - until \( S \) is terminal

\( \epsilon \)-greedy policy

\[
\begin{align*}
\epsilon \text{-greedy policy} \quad a &= \begin{cases} 
\arg \max_a Q(a, s) & \text{with probability } 1 - \epsilon, \\
\text{random } a & \text{with probability } \epsilon.
\end{cases}
\end{align*}
\]

[Sutton & Barto, RL Book, Ch.6]
Q-learning for control

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Initialize $Q(s, a)$, for all $s \in S, a \in A(s)$, arbitrarily, and $Q(terminal-state, \cdot) = 0$
Repeat (for each episode):
  Initialize $S$
  Repeat (for each step of episode):
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
    Take action $A$, observe $R, S'$
    $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$
    $S \leftarrow S'$
  until $S$ is terminal

$\epsilon$-greedy policy

$$a = \begin{cases} \arg \max_a Q(a, s) & \text{with probability } 1 - \epsilon, \\ \text{random } a & \text{with probability } \epsilon. \end{cases}$$

[Sutton & Barto, RL Book, Ch.6]
Q-learning for control

Q-learning (off-policy TD control) for estimating $\pi \approx \pi^*$

Initialize $Q(s, a)$, for all $s \in S, a \in A(s)$, arbitrarily, and $Q(terminal-state, \cdot) = 0$
Repeat (for each episode):
    Initialize $S$
    Repeat (for each step of episode):
        Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
        Take action $A$, observe $R, S'$
        $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$
        $S \leftarrow S'$
    until $S$ is terminal

$\epsilon$-greedy policy

$a = \begin{cases} 
\arg \max_a Q(a, s) & \text{with probability } 1 - \epsilon, \\
\text{random } a & \text{with probability } \epsilon.
\end{cases}$
Q-learning for control

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Initialize $Q(s,a)$, for all $s \in S, a \in A(s)$, arbitrarily, and $Q(\text{terminal-state,} \cdot) = 0$

Repeat (for each episode):
  Initialize $S$
  Repeat (for each step of episode):
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
    Take action $A$, observe $R, S'$
    $Q(S,A) \leftarrow Q(S,A) + \alpha [R + \gamma \max_a Q(S',a) - Q(S,A)]$
    $S \leftarrow S'$
  until $S$ is terminal

$\epsilon$-greedy policy

$$a = \begin{cases} 
\arg \max_a Q(a,s) & \text{with probability } 1 - \epsilon, \\
\text{random } a & \text{with probability } \epsilon. 
\end{cases}$$

[Sutton & Barto, RL Book, Ch.6]
Q-learning for control

Q-learning (off-policy TD control) for estimating $\pi \approx \pi^*$

- Converges to $Q^*$ as long as all (s,a) pairs continue to be updated and with minor constraints on learning rate

[Sutton & Barto, RL Book, Ch.6]
Summary

- Introduced algorithms for learning the optimal policy
  - Monte Carlo and TD methods
  - On-policy and off-policy methods
Practical difficulties

- Compared to supervised learning setting up an RL problem is often harder
  - Need an environment (or at least a simulator)
- Rewards
  - In some domains it’s clear (e.g., in games)
  - In others it’s much more subtle (e.g., you want to please a human)
Extra material
(Some will be used for this week’s exercises)
Example: Black Jack

- Episode: one hand
- States: Sum of player’s cards, dealer’s card, usable ace
- Actions: {Stay, Hit}
- Rewards: {Win +1, Tie 0, Loose -1}
- A few other assumptions: infinite deck
• Evaluates the policy that hits except when the sum of the cards is 20 or 21

[Figure 5.1, Sutton & Barto]
• Evaluates the policy that hits except when the sum of the cards is 20 or 21

[Figure 5.1, Sutton & Barto]
• Evaluates the policy that hits except when the sum of the cards is 20 or 21

[Figure 5.1, Sutton & Barto]
• Evaluates the policy that hits except when the sum of the cards is 20 or 21
Approximation techniques

- Methods we studied are “tabular”

- State value functions (and Q) can be approximated
  - Linear approximation: $V(s) = w^T x(s)$
  - Coupling between states through $x(s)$
  - Adapt the algorithms for this case.

- Updates to the value function now imply updating the weights $w$ using a gradient
Approximation techniques (prediction)

- Linear approximation: $V(s) = w^T x(s)$
Approximation techniques (prediction)

- Linear approximation: $V(s) = w^T x(s)$
- Objective: $\sum_{s \in S} [v_{\pi}(s) - w^T x(s)]^2$
Approximation techniques (prediction)

- Linear approximation: \( V(s) = w^T x(s) \)

- Objective: \( \sum_{s \in S} [v_\pi(s) - w^T x(s)]^2 \)

- Gradient update: \( w_{t+1} = w_t - 2\alpha \sum_{s \in S} [v_\pi(s) - w^T x(s)] x(s) \)
Approximation techniques (prediction)

• Linear approximation:  \( V(s) = w^T x(s) \)

• Objective:  \( \sum_{s \in \mathcal{S}} [v_{\pi}(s) - w^T x(s)]^2 \)

• Gradient update:  \( w_{t+1} = w_t - 2\alpha \sum_{s \in \mathcal{S}} [v_{\pi}(s) - w^T x(s)] x(s) \)

---

**Gradient Monte Carlo Algorithm for Estimating \( \hat{v} \approx v_\pi \)**

Input: the policy \( \pi \) to be evaluated
Input: a differentiable function \( \hat{v} : \mathcal{S} \times \mathbb{R}^d \to \mathbb{R} \)
Initialize value-function weights \( w \) as appropriate (e.g., \( w = 0 \))
Repeat forever:
  Generate an episode \( S_0, A_0, R_1, S_1, A_1, \ldots, R_T, S_T \) using \( \pi \)
  For \( t = 0, 1, \ldots, T-1 \):
    \( w \leftarrow w + \alpha [G_t - \hat{v}(S_t, w)] \nabla \hat{v}(S_t, w) \)

---

**First-visit MC prediction, for estimating \( V \approx v_\pi \)**

Initialize:
  \( \pi \) ← policy to be evaluated
  \( V \) ← an arbitrary state-value function
  \( \text{Returns}(s) \) ← an empty list, for all \( s \in \mathcal{S} \)
Repeat forever:
  Generate an episode using \( \pi \)
  For each state \( s \) appearing in the episode:
    \( G \) ← the return that follows the first occurrence of \( s \)
    Append \( G \) to \( \text{Returns}(s) \)
  \( V(s) \) ← average(\( \text{Returns}(s) \))
Approximation techniques (prediction)

- **Linear approximation:** \( V(s) = w^T x(s) \)

- **Objective:** \( \sum_{s \in S} [v_{\pi}(s) - w^T x(s)]^2 \)

- **Gradient update:** \( w_{t+1} = w_t - 2\alpha \sum_{s \in S} [v_{\pi}(s) - w^T x(s)] x(s) \)

---

**Gradient Monte Carlo Algorithm for Estimating \( \hat{v} \approx v_{\pi} \)**

Input: the policy \( \pi \) to be evaluated
Input: a differentiable function \( \hat{v} : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R} \)

Initialize value-function weights \( w \) as appropriate (e.g., \( w = 0 \))

Repeat forever:
- Generate an episode \( S_0, A_0, R_1, S_1, A_1, \ldots, R_T, S_T \) using \( \pi \)
- For \( t = 0, 1, \ldots, T-1 \):
  \[
  w \leftarrow w + \alpha [G_t - \hat{v}(S_t, w)] \nabla \hat{v}(S_t, w)
  \]

---

**First-visit MC prediction, for estimating \( V \approx v_{\pi} \)**

Initialize:
- \( \pi \) → policy to be evaluated
- \( V \) → an arbitrary state-value function
- \( Returns(s) \) → an empty list, for all \( s \in \delta \)

Repeat forever:
- Generate an episode using \( \pi \)
- For each state \( s \) appearing in the episode:
  - Generate an episode using \( \pi \)
  - For each state \( s \) appearing in the episode:
    - \( G \) → the return that follows the first occurrence of \( s \)
    - Append \( G \) to \( Returns(s) \)
  - \( V(s) \) → average(\( Returns(s) \))

---

\( G_t \) is an unbiased estimator of \( v_{\pi}(s_t) \)

[Sutton & Barto, RL Book, Ch.9]
Approximation techniques

- Works both for prediction and control
Approximation techniques

- Works both for prediction and control
- Any model can be used to approximate
Approximation techniques

- Works both for prediction and control
- Any model can be used to approximate
- Recent work using deep neural networks yield impressive performance on computer (Atari) games
Summary

• Today we have defined RL studied several algorithms for solving RL problems (mostly for tabular case)

• Main challenges
  • Credit assignment
  • Exploration/Exploitation tradeoff

• Algorithms
  • Prediction
    • Monte Carlo and TD(0)
  • Control
    • Q-learning

• Approximation algorithms can help scale reinforcement learning
Practical difficulties

• Compared to supervised learning setting up an RL problem is often harder
  • Need an environment (or at least a simulator)
  • Rewards
    • In some domains it’s clear (e.g., in games)
    • In others it’s much more subtle (e.g., you want to please a human)
Acknowledgements

- The algorithms are from “Reinforcement Learning: An Introduction” by Richard Sutton and Andrew Barto
- The definitive RL reference
- Some of these slides were adapted from Pascal Poupart’s slides (CS686 U.Waterloo)
- The TD demo is from Andrej Karpathy