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
  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 (I)

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
Experience (II)

• Any supervised learning problem can be mapped to a reinforcement learning problem
  • Loss function can be used as a reward function
  • Dependent variables (Y) become the actions
  • States encode information from the independent variables (x)
• Such mapping isn’t very useful in practice
  • SL is easier than RL
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 over 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.

- 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 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
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

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

- Model-free
Monte Carlo Methods

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

- Model-free
- Assume the environment is episodic
  - Think of playing poker. An episode is a poker 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 of each hand
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

  $\begin{align*}
  \text{Initialize:} \\
  \pi & \leftarrow \text{policy to be evaluated} \\
  V & \leftarrow \text{an arbitrary state-value function} \\
  Returns(s) & \leftarrow \text{an empty list, for all } s \in S \\
  \text{Repeat forever:} \\
  \text{Generate an episode using } \pi \\
  \text{For each state } s \text{ appearing in the episode:} \\
  G & \leftarrow \text{the return that follows the first occurrence of } s \\
  \text{Append } G \text{ to } Returns(s) \\
  V(s) & \leftarrow \text{average(}Returns(s)\text{)}
  \end{align*}$

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

[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: $(1, \rightarrow)$

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

Initialize:

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

Repeat forever:

Generate an episode using $\pi$
For each state $s$ appearing in the episode:

$G \leftarrow$ the return that follows the first occurrence of $s$
Append $G$ to $Returns(s)$
$V(s) \leftarrow$ 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: $(1, \rightarrow)$
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: \((1, \leftarrow)\)
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: \( (1, \rightarrow) \rightarrow (2, \rightarrow) \)

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

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

Repeat forever:
- Generate an episode using \( \pi \)
- For each state \( s \) appearing in the episode:
  - \( G \rightarrow \) the return that follows the first occurrence of \( s \)
  - Append \( G \) to \( \text{Returns}(s) \)
- \( V(s) \leftarrow \) average(\( \text{Returns}(s) \))
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: $(1, \rightarrow) \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow)$
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: $(1, \rightarrow) \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow)$
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, →) → (2, →) → (3, ↓) → (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: $(1, \rightarrow) \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow) \rightarrow (7, \downarrow) \rightarrow (6, \rightarrow)$
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, →) → (2, →) → (3, ↓) → (7, ↓) → (6, →)
```
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: $(1, \rightarrow) \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow) \rightarrow (7, \downarrow) \rightarrow (6, \rightarrow) \rightarrow (7, \downarrow) \rightarrow (10, \downarrow)$

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

Initialise:
- $\pi \leftarrow$ policy to be evaluated
- $V \leftarrow$ an arbitrary state-value function
- $\text{Returns}(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:

1. Generate an episode using $\pi$
2. For each state $s$ appearing in the episode:
   - $G \leftarrow$ the return that follows the first occurrence of $s$
   - Append $G$ to $\text{Returns}(s)$
3. $V(s) \leftarrow$ average($\text{Returns}(s)$)
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: $(1, \rightarrow) \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow) \rightarrow (7, \downarrow) \rightarrow (6, \rightarrow) \rightarrow (7, \downarrow) \rightarrow (10, \downarrow)$
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: $(1, \rightarrow) \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow) \rightarrow (7, \downarrow) \rightarrow (6, \rightarrow) \rightarrow (7, \downarrow) \rightarrow (10, \downarrow) \rightarrow (13, \downarrow)$

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

- **Initialize:**
  - $\pi \leftarrow$ policy to be evaluated
  - $V \leftarrow$ an arbitrary state-value function
  - $\text{Returns}(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

- **Repeat forever:**
  - Generate an episode using $\pi$
  - For each state $s$ appearing in the episode:
    - $G \leftarrow$ the return that follows the first occurrence of $s$
    - Append $G$ to $\text{Returns}(s)$
    - $V(s) \leftarrow \text{average}(\text{Returns}(s))$
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: $(1, \rightarrow) \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow) \rightarrow (7, \downarrow) \rightarrow (6, \rightarrow) \rightarrow (7, \downarrow) \rightarrow (10, \downarrow) \rightarrow (13, \downarrow) \rightarrow (17, \rightarrow)$

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 $s \in \mathcal{S}$

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 $Returns(s)$
- $V(s) ← \text{average}(Returns(s))$
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: $(1, \rightarrow) \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow) \rightarrow (7, \downarrow) \rightarrow (6, \rightarrow) \rightarrow (7, \downarrow) \rightarrow (10, \downarrow) \rightarrow (13, \downarrow) \rightarrow (17, \rightarrow)$

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

Initialise:

$\pi \leftarrow$ policy to be evaluated
$V \leftarrow$ arbitrary state-value function
$Returns(s) \leftarrow$ empty list, for all $s \in \mathcal{S}$

Repeat forever:

1. $G \leftarrow$ the return that follows the first occurrence of $s$
2. Append $G$ to $Returns(s)$
3. $V(s) \leftarrow \text{average}(Returns(s))$
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: $(1, \rightarrow) \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow) \rightarrow (7, \downarrow) \rightarrow (6, \rightarrow) \rightarrow (7, \downarrow) \rightarrow (10, \downarrow) \rightarrow (13, \downarrow) \rightarrow (17, \rightarrow)$
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: $(1, \rightarrow) \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow) \rightarrow (7, \downarrow) \rightarrow (6, \rightarrow) \rightarrow (7, \downarrow) \rightarrow (10, \downarrow) \rightarrow (13, \downarrow) \rightarrow (17, \rightarrow)$

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

Initilaize:
- $\pi$ ← 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 $\text{Returns}(s)$
  - $V(s) \leftarrow \text{average}(\text{Returns}(s))$
Example: grid world

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

For state 7:

```
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:**

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

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:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<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>

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:

\[ (1, \rightarrow) \rightarrow (2, \rightarrow) \rightarrow (3, \downarrow) \rightarrow (7, \downarrow) \rightarrow (6, \rightarrow) \rightarrow (7, \downarrow) \rightarrow (10, \uparrow) \rightarrow (13, \downarrow) \rightarrow (17, \rightarrow) \]

For state 7:

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

\[
= \gamma^6 10
\]

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

[Figure 5.1, Sutton & Barto]
Q-value for control

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

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

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

• When state transitions are unknown what can we do?

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

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

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

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

Repeat forever:
- Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ 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 $\text{Returns}(s, a)$
- $Q(s, a) \leftarrow \text{average}(\text{Returns}(s, a))$
- For each $s$ in the episode:
  - $\pi(s) \leftarrow \text{argmax}_a Q(s, a)$

First-visit MC prediction, for estimating $V \approx V^*$

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

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

[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
Learning without “exploring starts”

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

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

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

- Repeat forever:
  1. Generate an episode using \(\pi\)
  2. For each pair \((s,a)\) appearing in the episode:
     - \(G \leftarrow \text{the return that follows the first occurrence of } s,a\)
     - Append \(G\) to \(\text{Returns}(s,a)\)
     - \(Q(s,a) \leftarrow \text{average}(\text{Returns}(s,a))\)
  3. For each \(s\) in the episode:
     - \(A^* \leftarrow \text{arg max}_a Q(s,a)\)
     - 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 \(\varepsilon\)-soft policies**

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

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

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

- Policy value cannot decrease

\[ V_{\pi'}(S) \geq V_\pi(S), \quad \forall S \in S \]

\( \pi' \): policy at next step
\( \pi \): policy at current step

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

- Allow a policy to be learned through interactions
- 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 stats)
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:

Step size
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:

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

First-visit MC prediction, for estimating value

Initialize:
- \( \pi \) → policy to be evaluated
- \( V \) → an arbitrary state-value function
- \( \text{Returns}(s) \) → 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 \( \text{Returns}(s) \)
  - \( V(s) \) ← average(\( \text{Returns}(s) \))
Temporal Difference (TD) Learning

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

- Monte Carlo methods

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

  \[ G_t = \sum_t \gamma^t R(s_t) \]

- TD(0)

  - updates “instantly”

  \[ V(s_t) = V(s_t) + \alpha [R_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \]
TD(0) for prediction

**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):
    - \( A \leftarrow \text{action given by } \pi \text{ for } S \)
    - **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 $Q(\text{terminal}, \cdot) = 0$
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 $\text{TD}(0)$ 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
• Demo: http://www.cs.toronto.edu/~lcharlin/courses/80-629/reinforcejs/gridworld_td.html

(from Andrej Karpathy)
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 $\pi \approx \pi^*$

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

[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 $Q(\text{terminal-state}, \cdot) = 0$
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 $\pi \approx \pi_*$

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 $\pi \approx \pi_*$

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

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 $R, S'$
  - $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$

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

- 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]
Approximation techniques

- Methods we studied are “tabular”

- State value functions (and Q) can be approximated
  - Linear approximation: \( V(s) = w^\top x(s) \)
  - 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: \( V(s) = w^\top x(s) \)
- Objective: \( \sum_{s \in S} [v_{\pi}(s) - w^\top x(s)]^2 \)
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:** \( \sum_{s \in S} [v_{\pi}(s) - w^T x(s)]^2 \)

- **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 \( \hat{v} = v_\pi \)**

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

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 \( V = v_\pi \)**

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

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

[Sutton & Barto, RL Book, Ch.9]
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 \( \hat{v} : \mathcal{S} \times \mathbb{R}^d \rightarrow \mathbb{R} \)

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 \( t = 0, 1, \ldots, T-1 \):
  \[ w \leftarrow w + \alpha [G_t - \hat{v}(S_t, w)] \nabla \hat{v}(S_t, w) \]

\( G_t \) 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