The Story So Far: MDPs and RL

**Known MDP: Offline Solution**

<table>
<thead>
<tr>
<th>Goal</th>
<th>Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute $V^<em>$, $Q^</em>$, $\pi^*$</td>
<td>Value / policy iteration</td>
</tr>
<tr>
<td>Evaluate a fixed policy $\pi$</td>
<td>Policy evaluation</td>
</tr>
</tbody>
</table>

**Unknown MDP: Model-Based**

<table>
<thead>
<tr>
<th>Goal</th>
<th>Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute $V^<em>$, $Q^</em>$, $\pi^*$</td>
<td>VI/PI on approx. MDP</td>
</tr>
<tr>
<td>Evaluate a fixed policy $\pi$</td>
<td>PE on approx. MDP</td>
</tr>
</tbody>
</table>

**Unknown MDP: Model-Free**

<table>
<thead>
<tr>
<th>Goal</th>
<th>Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute $V^<em>$, $Q^</em>$, $\pi^*$</td>
<td>Q-learning</td>
</tr>
<tr>
<td>Evaluate a fixed policy $\pi$</td>
<td>Value Learning</td>
</tr>
</tbody>
</table>

Example: Direct Evaluation

<table>
<thead>
<tr>
<th>Input Policy $\pi$</th>
<th>Observed Episodes (Training)</th>
<th>Output Values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Episode 1</td>
<td>$V_B = +8$</td>
</tr>
<tr>
<td></td>
<td>B, east, C, -1</td>
<td>$V_C = +4$</td>
</tr>
<tr>
<td></td>
<td>C, east, D, -1</td>
<td>$V_D = +10$</td>
</tr>
<tr>
<td></td>
<td>D, exit, x, +10</td>
<td>$V_E = -2$</td>
</tr>
<tr>
<td></td>
<td>Episode 2</td>
<td>$V_B = +8$</td>
</tr>
<tr>
<td></td>
<td>B, east, C, -1</td>
<td>$V_C = +4$</td>
</tr>
<tr>
<td></td>
<td>C, east, D, -1</td>
<td>$V_D = +10$</td>
</tr>
<tr>
<td></td>
<td>D, exit, x, +10</td>
<td>$V_E = -2$</td>
</tr>
<tr>
<td></td>
<td>Episode 3</td>
<td>$V_B = +8$</td>
</tr>
<tr>
<td></td>
<td>B, north, C, -1</td>
<td>$V_C = +4$</td>
</tr>
<tr>
<td></td>
<td>C, east, A, -1</td>
<td>$V_D = +10$</td>
</tr>
<tr>
<td></td>
<td>A, exit, x, -10</td>
<td>$V_E = -2$</td>
</tr>
<tr>
<td></td>
<td>Episode 4</td>
<td>$V_B = +8$</td>
</tr>
<tr>
<td></td>
<td>B, north, C, -1</td>
<td>$V_C = +4$</td>
</tr>
<tr>
<td></td>
<td>C, east, A, -1</td>
<td>$V_D = +10$</td>
</tr>
<tr>
<td></td>
<td>A, exit, x, -10</td>
<td>$V_E = -2$</td>
</tr>
</tbody>
</table>

Problems with Direct Evaluation

- What’s good about direct evaluation?
  - It’s easy to understand
  - It doesn’t require any knowledge of $T$, $R$
  - It eventually computes the correct average values, using just sample transitions

- What bad about it?
  - It wastes information about state connections
  - Each state must be learned separately
  - So, it takes a long time to learn

Why Not Use Policy Evaluation?

- Simplified Bellman updates calculate $V$ for a fixed policy:
  - Each round, replace $V$ with a one-step-look-ahead layer over $V$
  - $V_i(s) = 0$
  - $V_{i+1}(s) = \sum T(s, \pi(s), s') [R(s, \pi(s), s') + \gamma V_i(s')]$ |
  - This approach fully exploited the connections between the states
  - Unfortunately, we need $T$ and $R$ to do it!
  - Key question: how can we do this update to $V$ without knowing $T$ and $R$?
  - In other words, how to we take a weighted average without knowing the weights?

Sample-Based Policy Evaluation?

- We want to improve our estimate of $V$ by computing these averages:
  - $V_{i+1}(s) = \sum T(s, \pi(s), s') [R(s, \pi(s), s') + \gamma V_i(s')]$
  - Idea: Take samples of outcomes $s'$ (by doing the action!) and average
    - $sample_1 = R(s, \pi(s), s') + \gamma V_i(s')$
    - $sample_2 = R(s, \pi(s), s') + \gamma V_i(s')$
    - $\ldots$
    - $sample_n = R(s, \pi(s), s') + \gamma V_i(s')$
  - $V_{i+1}(s) = \frac{1}{n} \sum_{i=1}^{n} sample_i$

Temporal Difference Learning

- Big idea: learn from every experience!
  - Update $V(s)$ each time we experience a transition $s, a, s', r$.
  - Likely outcomes $s'$ will contribute updates more often
  - Temporal difference learning of values
    - Policy still fixed, still doing evaluation!
    - Move values toward value of whatever successor occurs: running average

Exponential Moving Average

- Exponential moving average
  - The running interpolation update: $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
  - The running average:
    - $\tilde{x}_n = (1 - \alpha) \cdot x_n + \alpha \cdot \tilde{x}_{n-1}$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$

- Exponential moving average
  - The running interpolation update: $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
  - The running average:
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$

- Exponential moving average
  - The running interpolation update: $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
  - The running average:
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$

- Exponential moving average
  - The running interpolation update: $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
  - The running average:
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$

- Exponential moving average
  - The running interpolation update: $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
  - The running average:
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$

- Exponential moving average
  - The running interpolation update: $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
  - The running average:
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$

- Exponential moving average
  - The running interpolation update: $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
  - The running average:
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$

- Exponential moving average
  - The running interpolation update: $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
  - The running average:
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
    - $\tilde{x}_n = (1 - \alpha) \cdot \tilde{x}_{n-1} + \alpha \cdot x_n$
Example: Temporal Difference Learning

### States

<table>
<thead>
<tr>
<th>States</th>
<th>Observed Transitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>east, C, -2</td>
</tr>
<tr>
<td>C</td>
<td>east, D, -2</td>
</tr>
</tbody>
</table>

### Example: Temporal Difference Learning

\[ V^n(s) \leftarrow (1 - \alpha) V^{n-1}(s) + \alpha [R(s, \pi(s), s') + \gamma V^{n}(s')] \]

Problems with TD Value Learning

- TD value leaning is a model-free way to do policy evaluation, mimicking Bellman updates with running sample averages.
- However, if we want to turn values into a (new) policy, we’re sunk:
  \[ \pi(s) = \arg \max Q(s, a) \]
  \[ Q(s, a) = \sum_r T(s, a, s') [R(s, a, s') + \gamma V^{n+1}(s')] \]
- Idea: learn Q-values, not values.
- Makes action selection model-free too!

Active Reinforcement Learning

### Active Reinforcement Learning

- Full reinforcement learning: optimal policies (like value iteration)
  - You don’t know the transitions \( T(s, a, s') \)
  - You don’t know the rewards \( R(s, a, s') \)
  - You choose the actions now
  - Goal: learn the optimal policy / values

- In this case:
  - Learner makes choices!
  - Fundamental tradeoff: exploration vs. exploitation
  - This is NOT offline planning: You actually take actions in the world and find out what happens...

### Detour: Q-Value Iteration

- Value iteration: find successive (depth-limited) values
  - Start with \( V_0(s) = 0 \), which we know is right
  - Given \( V_k \), calculate the depth \( k+1 \) values for all states:
    \[ V_{k+1}(s) \leftarrow \max_a \sum_r T(s, a, s') [R(s, a, s') + \gamma V_k(s')] \]

- But Q-values are more useful, so compute them instead
  - Start with \( Q_0(s, a) = 0 \), which we know is right
  - Given \( Q_k \), calculate the depth \( k+1 \) q-values for all q-states:
    \[ Q_{k+1}(s, a) \leftarrow \sum_r T(s, a, s') [R(s, a, s') + \gamma \max_{a'} Q_k(s', a')] \]

### Q-Learning

- Q-Learning: sample-based Q-value iteration
  - \[ Q_{k+1}(s, a) \leftarrow \sum_r T(s, a, s') [R(s, a, s') + \gamma \max_{a'} Q_k(s', a')] \]
- Learn Q(s,a) values as you go
  - Receive a sample \((s, a, r, s')\)
  - Consider your old estimate: \( Q(s,a) \)
  - Consider your new sample estimate:
    \[ \text{sample} = R(s, a, r) + \gamma \max_{a'} Q_k(s', a') \]
- Incorporate the new estimate into a running average:
  \[ Q(s, a) \leftarrow (1 - \alpha) Q(s, a) + \alpha \text{sample} \]

### Q-Learning Properties

- Amazing result: Q-learning converges to optimal policy -- even if you’re acting suboptimally!
- This is called off-policy learning
- Caveats:
  - You have to explore enough
  - You have to eventually make the learning rate small enough
  - But not decrease it too quickly
  - Basically, in the limit, it doesn’t matter how you select actions! (?)

Video of Demo Q-Learning -- Gridworld

Video of Demo Q-Learning -- Crawler

Video of Demo Q-Learning -- Auto – cliff grid
Video of Demo Q-Learning Auto Cliff Grid