Instructor: Anca Dragan

University of California, Berkeley

[Slides by Dan Klein, Pieter Abbeel, Anca Dragan. http://ai.berkeley.edu.]
Reinforcement Learning
Double Bandits
Double-Bandit MDP

- Actions: Blue, Red
- States: Win, Lose

No discount
10 time steps
Both states have the same value
Solving MDPs is offline planning
- You determine all quantities through computation
- You need to know the details of the MDP
- You do not actually play the game!
Let’s Play!
Online Planning

Rules changed! Red’s win chance is different.
Let’s Play!

$1 WINNER

$0 $0 $0 $2 $0

$2 $0 $0 $0 $0 $0

$2 $0 $0 $0 $0 $0
What Just Happened?

- That wasn’t planning, it was learning!
  - Specifically, reinforcement learning
  - There was an MDP, but you couldn’t solve it with just computation
  - You needed to actually act to figure it out

- Important ideas in reinforcement learning that came up
  - Exploration: you have to try unknown actions to get information
  - Exploitation: eventually, you have to use what you know
  - Regret: even if you learn intelligently, you make mistakes
  - Sampling: because of chance, you have to try things repeatedly
  - Difficulty: learning can be much harder than solving a known MDP
Reinforcement Learning

- Still assume a Markov decision process (MDP):
  - A set of states $s \in S$
  - A set of actions (per state) $A$
  - A model $T(s,a,s')$
  - A reward function $R(s,a,s')$

- Still looking for a policy $\pi(s)$

- New twist: don’t know $T$ or $R$
  - I.e. we don’t know which states are good or what the actions do
  - Must actually try actions and states out to learn
Reinforcement Learning

**Basic idea:**
- Receive feedback in the form of **rewards**
- Agent’s utility is defined by the reward function
- Must (learn to) act so as to **maximize expected rewards**
- All learning is based on observed samples of outcomes!
Example: Learning to Walk

Initial  A Learning Trial  After Learning [1K Trials]

[Kohl and Stone, ICRA 2004]
Example: Learning to Walk

[Video: AIBO WALK – initial]

[Kohl and Stone, ICRA 2004]
Example: Learning to Walk

Training

[Video: AIBO WALK - training]

[Kohl and Stone, ICRA 2004]
Example: Learning to Walk

[Video: AIBO WALK - finished]

[Kohl and Stone, ICRA 2004]
The Crawler!
Video of Demo Crawler Bot
Google DeepMind's Deep Q-learning

The algorithm will play Atari breakout.
Reinforcement Learning

- Still assume a Markov decision process (MDP):
  - A set of states \( s \in S \)
  - A set of actions (per state) \( A \)
  - A model \( T(s,a,s') \)
  - A reward function \( R(s,a,s') \)
- Still looking for a policy \( \pi(s) \)
- New twist: don’t know \( T \) or \( R \)
  - I.e. we don’t know which states are good or what the actions do
  - Must actually try actions and states out to learn
Offline (MDPs) vs. Online (RL)

Offline Solution

Online Learning
Model-Based Learning
Model-Based Learning

Model-Based Idea:
- Learn an approximate model based on experiences
- Solve for values as if the learned model were correct

Step 1: Learn empirical MDP model
- Count outcomes $s'$ for each $s$, $\hat{T}(s, a, s')$
- Normalize to $\hat{R}(s, a, s')$
- Discover each $\hat{R}(s, a, s')$ when we experience $(s, a, s')$

Step 2: Solve the learned MDP
- For example, use value iteration, as before
Example: Model-Based Learning

Assume: $\gamma = 1$

<table>
<thead>
<tr>
<th>Input Policy $\pi$</th>
<th>Observed Episodes (Training)</th>
<th>Learned Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Episode 1</td>
<td>$\hat{T}(s, a, s')$</td>
</tr>
<tr>
<td></td>
<td>B, east, C, -1</td>
<td>T(B, east, C) = 1.00</td>
</tr>
<tr>
<td></td>
<td>C, east, D, -1</td>
<td>T(C, east, D) = 0.75</td>
</tr>
<tr>
<td></td>
<td>D, exit, x, +10</td>
<td>T(C, east, A) = 0.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>Episode 2</td>
<td>$\hat{T}(s, a, s')$</td>
</tr>
<tr>
<td></td>
<td>B, east, C, -1</td>
<td>R(B, east, C) = -1</td>
</tr>
<tr>
<td></td>
<td>C, east, D, -1</td>
<td>R(C, east, D) = -1</td>
</tr>
<tr>
<td></td>
<td>D, exit, x, +10</td>
<td>R(D, exit, x) = +10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>Episode 3</td>
<td>$\hat{T}(s, a, s')$</td>
</tr>
<tr>
<td></td>
<td>E, north, C, -1</td>
<td>R(B, east, C) = -1</td>
</tr>
<tr>
<td></td>
<td>C, east, D, -1</td>
<td>R(C, east, D) = -1</td>
</tr>
<tr>
<td></td>
<td>D, exit, x, +10</td>
<td>R(D, exit, x) = +10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>Episode 4</td>
<td>$\hat{T}(s, a, s')$</td>
</tr>
<tr>
<td></td>
<td>E, north, C, -1</td>
<td>R(B, east, C) = -1</td>
</tr>
<tr>
<td></td>
<td>C, east, A, -1</td>
<td>R(C, east, D) = -1</td>
</tr>
<tr>
<td></td>
<td>D, exit, x, +10</td>
<td>R(D, exit, x) = +10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
Example: Expected Age

Goal: Compute expected age of cs188 students

<table>
<thead>
<tr>
<th>Known P(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ E[A] = \sum_a P(a) \cdot a = 0.35 \times 20 + \ldots ]</td>
</tr>
</tbody>
</table>

Without P(A), instead collect samples \([a_1, a_2, \ldots a_N]\)

**Unknown P(A): “Model Based”**

\[
\hat{P}(a) = \frac{\text{num}(a)}{N}
\]

\[ E[A] \approx \sum_a \hat{P}(a) \cdot a \]

Why does this work? Because eventually you learn the right model.

**Unknown P(A): “Model Free”**

\[ E[A] \approx \frac{1}{N} \sum_i a_i \]

Why does this work? Because samples appear with the right frequencies.
Model-Free Learning
Passive Reinforcement Learning
Simplified task: policy evaluation

- Input: a fixed policy $\pi(s)$
- You don’t know the transitions $T(s,a,s')$
- You don’t know the rewards $R(s,a,s')$
- Goal: learn the state values

In this case:

- Learner is “along for the ride”
- No choice about what actions to take
- Just execute the policy and learn from experience
- This is NOT offline planning! You actually take actions in the world.
Direct Evaluation

- Goal: Compute values for each state under $\pi$

- Idea: Average together observed sample values
  - Act according to $\pi$
  - Every time you visit a state, write down what the sum of discounted rewards turned out to be
  - Average those samples

- This is called direct evaluation
Example: Direct Evaluation

Input Policy $\pi$

Assume: $\gamma = 1$

Observed Episodes (Training)

<table>
<thead>
<tr>
<th>Episode 1</th>
<th>Episode 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>B, east, C, -1</td>
<td>B, east, C, -1</td>
</tr>
<tr>
<td>C, east, D, -1</td>
<td>C, east, D, -1</td>
</tr>
<tr>
<td>D, exit, x, +10</td>
<td>D, exit, x, +10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Episode 3</th>
<th>Episode 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>E, north, C, -1</td>
<td>E, north, C, -1</td>
</tr>
<tr>
<td>C, east, D, -1</td>
<td>C, east, A, -1</td>
</tr>
<tr>
<td>D, exit, x, +10</td>
<td>A, exit, x, -10</td>
</tr>
</tbody>
</table>

Output Values

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Episode 1</td>
<td>-10</td>
<td>+8</td>
<td>+4</td>
<td>+10</td>
</tr>
<tr>
<td>Episode 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Episode 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Episode 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

E, north, C, -1
C, east, D, -1
D, exit, x, +10
A, exit, x, -10
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

Output Values

If B and E both go to C under this policy, how can their values be different?
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 $V$

$$V_0^\pi(s) = 0$$

$$V_{k+1}^\pi(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_k^\pi(s')]$$

- 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 can we take a weighted average without knowing the weights?
We want to improve our estimate of $V$ by computing these averages:

$$V_{k+1}^\pi(s) \leftarrow \sum_{s'} T(s, \pi(s), s') [R(s, \pi(s), s') + \gamma V_k^\pi(s')]$$

Idea: Take samples of outcomes $s'$ (by doing the action!) and average

$$sample_1 = R(s, \pi(s), s'_1) + \gamma V_k^\pi(s'_1)$$
$$sample_2 = R(s, \pi(s), s'_2) + \gamma V_k^\pi(s'_2)$$
$$\ldots$$
$$sample_n = R(s, \pi(s), s'_n) + \gamma V_k^\pi(s'_n)$$

$$V_{k+1}^\pi(s) \leftarrow \frac{1}{n} \sum_{i} sample_i$$
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

Sample of $V(s)$:

$$sample = R(s, \pi(s), s') + \gamma V^\pi(s')$$

Update to $V(s)$:

$$V^\pi(s) \leftarrow (1 - \alpha)V^\pi(s) + (\alpha)sample$$

Same update:

$$V^\pi(s) \leftarrow V^\pi(s) + \alpha(sample - V^\pi(s))$$
Exponential Moving Average

- Exponential moving average
  - The running interpolation update: \( \bar{x}_n = (1 - \alpha) \cdot \bar{x}_{n-1} + \alpha \cdot x_n \)
  - Makes recent samples more important
  - Forgets about the past (distant past values were wrong anyway)

- Decreasing learning rate (alpha) can give converging averages
Example: Temporal Difference Learning

Assume: \( \gamma = 1, \alpha = 1/2 \)

\[
V^\pi(s) \leftarrow (1 - \alpha)V^\pi(s) + \alpha \left[ R(s, \pi(s), s') + \gamma V^\pi(s') \right]
\]
Problems with TD Value Learning

- TD value learning 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:

\[ \pi(s) = \arg \max_a Q(s, a) \]

\[ Q(s, a) = \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma V(s') \right] \]

- Idea: learn Q-values, not values.

- Makes action selection model-free too!
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:
    \[ V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma V_k(s') \right] \]

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

Q-Learning: sample-based Q-value iteration

\[ Q_{k+1}(s, a) \leftarrow \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right] \]

Learn Q(s,a) values as you go

0 Receive a sample \((s, a, s', r)\)
0 Consider your old estimate: \(Q(s, a)\)
0 Consider your new sample estimate:

\[ \text{sample} = R(s, a, s') + \gamma \max_{a'} Q(s', a') \]

0 Incorporate the new estimate into a running average:

\[ Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + (\alpha) \text{[sample]} \]
Video of Demo Q-Learning -- Gridworld
Video of Demo Q-Learning -- Crawler
Active Reinforcement Learning
Q-Learning:
act according to current optimal (and also explore…)

- 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…
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 (!)
Discussion: Model-Based vs Model-Free RL