Announcements

- P0 / P1 / W1 / W2 in glookup
  - If you have no entry, etc, email staff list!
  - If you have questions, see one of us or email list.
- W1, W2: can be picked up from 188 return box in 283 Soda
- W3: Utilities --- Due Thursday.
- Recall: readings for current material
- Online book: Sutton and Barto

Announcements II

- Section:
  - 101: Tue 3-4pm, 285 Cory
  - 104: Tue 4-5pm, 285 Cory
  - 102: Wed 11-noon, 285 Cory
  - 103: Wed noon-1pm, 285 Cory

MDPs recap

- Markov decision processes:
  - States S
  - Actions A
  - Transitions P(s'|s,a) (or T(s,a,s'))
  - Rewards R(s,a,s') (and discount \( \gamma \))
- Solution methods:
  - Value iteration (VI)
  - Policy iteration (PI)
  - Asynchronous value iteration
- Current limitations:
  - Relatively small state spaces
  - Assumes T and R are known

MDP Example: Grid World

- The agent lives in a grid
- Walls block the agent's path
- The agent's actions do not always go as planned:
  - 80% of the time, the action North takes the agent North (if there is no wall there)
  - 10% of the time, North takes the agent West; 10% East
  - If there is a wall in the direction the agent would have been taken, the agent stays put
- Rewards come at the end
- Goal: maximize sum of rewards

MDP Example: Grid World

\[
\begin{align*}
\text{MDP} = (S, A, T, R, s_0, \gamma) \\
\text{Set of states } S &= \{ (0,0), (0,1), (0,2), (0,3), (1,0), (1,1), (1,2), (1,3) \} \\
\text{Set of actions } A &= \{ \text{North, East, South, West} \} \\
\text{Transition model } T &= T(s,a,s') \\
T((0,0), \text{North}, (0,1)) &= 0.8 \\
T((0,0), \text{North}, (1,0)) &= 0.2 \\
T((0,1), \text{North}, (1,0)) &= 0.1 \\
T((0,1), \text{North}, (1,1)) &= 0.1 \\
\text{Initial state } s_0 &= (0,0) \\
\text{Discount factor } \gamma &= 0.9
\end{align*}
\]
Value Iteration

- Idea:
  - $V_i(s)$: the expected discounted sum of rewards accumulated when starting from state $s$ and acting optimally for a horizon of $i$ time steps.
  - Start with $V_0(s) = 0$, which we know is right (why?)
  - Given $V_i(s)$, calculate the values for all states for horizon $i+1$:
    $$ V_{i+1}(s) = \max_a \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma V_i(s') \right] $$

- This is called a value update or Bellman update
- Repeat until convergence
- Theorem: will converge to unique optimal values

Complete procedure

1. Run value iteration (off-line)
   Returns $V$, which (assuming sufficiently many iterations is a good approximation of $V^*$)

2. Agent acts.
   At time $t$ the agent is in state $s_t$ and takes the action $a_t$:

Reinforcement Learning

- Reinforcement learning:
  - Still assume an 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. don't know which states are good or what the actions do
  - Must actually try actions and states out to learn

Example: learning to walk

Before learning (hand-tuned)  One of many learning runs  After learning [After 1000 field traversals]

[Institute and Stone, ICRA 2004]

Passive Learning

- Simplified task
  - You don’t know the transitions $T(s, a, s')$
  - You don’t know the rewards $R(s, a, s')$
  - You are given a policy $\pi(s)$
  - Goal: learn the state values
  - ... what policy evaluation did

- In this case:
  - Learner “along for the ride”
  - No choice about what actions to take
  - Just execute the policy and learn from experience
  - We’ll get to the active case soon

This is NOT offline planning! You actually take actions in the world and see what happens...
Recap: Model-Based Policy Evaluation

- Simplified Bellman updates to calculate $V$ for a fixed policy:
  - New $V$ is expected one-step-look-ahead using current $V$
  - Unfortunately, need $T$ and $R$

\[
V^*_T(s) = 0 \\
V^*_{T+1}(s) = \sum_{s'} \binom{(s, \pi(s), s')} R(s, \pi(s), s') + \gamma V^*_T(s')
\]

Why does this work? Because samples appear with the right frequencies.

Model-Based Learning

- Idea:
  - Learn the model empirically through experience
  - Solve for values as if the learned model were correct

- Simple empirical model learning
  - Count outcomes for each $s, a$
  - Normalize to give estimate of $T(s, a, s')$
  - Discover $R(s, a, s')$ when we experience $(s, a, s')$

- Solving the MDP with the learned model
  - Iterative policy evaluation, for example

\[
V^*_{T+1}(s) = \sum_{s'} \binom{(s, \pi(s), s')} R(s, \pi(s), s') + \gamma V^*_T(s')
\]

Example: Model-Based Learning

- Episodes:

<table>
<thead>
<tr>
<th>(1,1) up-1</th>
<th>(1,1) up-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2) up-1</td>
<td>(1,2) up-1</td>
</tr>
<tr>
<td>(1,3) right-1</td>
<td>(1,3) right-1</td>
</tr>
<tr>
<td>(2,2) right-1</td>
<td>(2,2) right-1</td>
</tr>
<tr>
<td>(2,3) right-1</td>
<td>(2,3) right-1</td>
</tr>
<tr>
<td>(3,2) up-1</td>
<td>(3,2) up-1</td>
</tr>
<tr>
<td>(3,3) right-1</td>
<td>(3,3) right-1</td>
</tr>
<tr>
<td>(4,3) exit +100</td>
<td>(4,3) exit +100</td>
</tr>
</tbody>
</table>

- Make $(s, a, s')$ union

\[
V(s, \pi(s), s') = \frac{1}{k} \sum_{i=1}^{k} (R(s, \pi(s), s') + \gamma V^*_T(s'))
\]

Model-Free Learning

- Want to compute an expectation weighted by $P(x)$:

\[
E[f(x)] = \sum_x P(x) f(x)
\]

- Model-based: estimate $P(x)$ from samples, compute expectation

\[
x_i \sim P(x) \quad E[f(x)] \approx \frac{1}{k} \sum_i P(x) f(x)
\]

- Model-free: estimate expectation directly from samples

\[
x_i \sim P(x) \quad E[f(x)] \approx \frac{1}{k} \sum_i f(x_i)
\]

Why does this work? Because samples appear with the right frequencies!

Example: Direct Estimation

- Episodes:

<table>
<thead>
<tr>
<th>(1,1) up-1</th>
<th>(1,1) up-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2) up-1</td>
<td>(1,2) up-1</td>
</tr>
<tr>
<td>(1,3) right-1</td>
<td>(1,3) right-1</td>
</tr>
<tr>
<td>(2,2) right-1</td>
<td>(2,2) right-1</td>
</tr>
<tr>
<td>(2,3) right-1</td>
<td>(2,3) right-1</td>
</tr>
<tr>
<td>(3,2) up-1</td>
<td>(3,2) up-1</td>
</tr>
<tr>
<td>(3,3) right-1</td>
<td>(3,3) right-1</td>
</tr>
<tr>
<td>(4,3) exit +100</td>
<td>(4,3) exit +100</td>
</tr>
</tbody>
</table>

- Who needs $T$ and $R$? Approximate the expectation with samples (drawn from $T$)

\[
V(s, \pi(s), s') = \frac{1}{k} \sum_i (R(s, \pi(s), s') + \gamma V^*_T(s'))
\]

Sample-Based Policy Evaluation?

- Almost! But we only actually make progress when we move to $V^*_T$
Temporal-Difference Learning

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

Sample of $V(s)$:

$\text{same update:}$

$$V^T(s) \leftarrow (1-\alpha)V^T(s) + \alpha \cdot \text{sample}$$

Exponential Moving Average

- Exponential moving average
  - Makes recent samples more important
  - Forgets about the past (distant past values were wrong anyway)
  - Easy to compute from the running average
  - Decreasing learning rate can give converging averages

Policy evaluation when $T$ (and $R$) unknown --- recap

- Model-based:
  - Learn the model empirically through experience
  - Solve for values as if the learned model were correct
- Model-free:
  - Direct estimation:
    - $V(s) = \text{sample estimate of sum of rewards accumulated from state } s \text{ onwards}$
  - Temporal difference (TD) value learning:
    - Move values toward value of whatever successor occurs: running average!

$$\text{same update:}$$

$$V^T(s) \leftarrow (1-\alpha)V^T(s) + \alpha \cdot \text{sample}$$