Reasoning Under Uncertainty

<table>
<thead>
<tr>
<th>Learn model of outcomes</th>
<th>Multi-armed bandits</th>
<th>Reinforcement Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Given model of stochastic outcomes</td>
<td>Decision theory</td>
<td>Markov Decision Processes</td>
</tr>
<tr>
<td>Actions Don’t Change State of the World</td>
<td></td>
<td>Actions Change State of the World</td>
</tr>
</tbody>
</table>

Carnegie Mellon University
Reinforcement Learning

Goal: Maximize expected sum of future rewards
Toddler Robot

https://www.youtube.com/watch?v=goqWX7bC-ZY&list=PL28ekMsVlKa1mKRif05Uxn-iuP2Ms6qDL
MDP Planning vs Reinforcement Learning

Don’t have a simulator! Have to actually learn what happens if take an action in a state.

Drawings by Ketrina Yim
Before figuring out how to act, let’s first just try to figure out how good a particular strategy is.
Policy Evaluation given MDP’s model parameters

Learning a Policy’s Value (Passive Reinforcement Learning)
Passive Reinforcement Learning

Two Approaches

1. Build a model

- Transition Model?
- $T(s_1,a_1,s_1) = 0.8,$
- $R(s_1,a_1,s_1) = 4, \ldots$

State → Action

Reward model? → Agent
Start at (1,1)

Adaption of drawing by Ketrina Yim
Start at (1,1)

\[ s=(1,1) \text{ action= tup} \]
Start at (1,1) 
s=(1,1) action= tup, s’=(1,2), r = -.01
Start at (1,1)
s=(1,1) action= tup, s’=(1,2), r = -.01
s=(1,2) action= tup
Start at (1,1)

s=(1,1) action= tup, s’=(1,2), r = -.01

s=(1,2) action= tup, s’=(1,2), r = -.01
Start at (1,1)

s=(1,1) action= tup, s’=(1,2), r = -.01

s=(1,2) action= tup, s’=(1,2), r = -.01

s=(1,2) action= tup, s’=(1,3), r = -.01

s=(1,3) action= tright, s’=(2,3), r = -.01

s=(2,3) action= tright, s’=(3,3), r = -.01

s=(3,3) action= tright, s’=(3,2), r = -.01

s=(3,2) action= tup, s’=(3,3), r = -.01

s=(3,3) action= tright, s’=(4,3), r = 1
Start at (1,1)

s=(1,1) action= tup, s'=(1,2), r = -.01
s=(1,2) action= tup, s'=(1,2), r = -.01
s=(1,2) action=tup, s'=(1,3), r = -.01
s=(1,3) action=tright, s'=(2,3), r = -.01
s=(2,3) action=tright, s'=(3,3), r = -.01
s=(3,3) action=tright, s'=(3,2), r = -.01
s=(3,3) action=tright, s'=(3,3), r = -.01
s=(3,2) action=tup, s'=(3,3), r = -.01
s=(3,3) action=tright, s'=(4,3), r = 1

Can use experience to estimate MDP T and R models
Start at (1,1)

\[ s = (1,1) \text{ action}= \text{tup}, \quad s' = (1,2), \quad r = -0.01 \]

\[ s = (1,2) \text{ action}= \text{tup}, \quad s' = (1,2), \quad r = -0.01 \]

\[ s = (1,2) \text{ action}=\text{tup}, \quad s' = (1,3), \quad r = -0.01 \]

\[ s = (1,3) \text{ action}=\text{tright}, \quad s' = (2,3), \quad r = -0.01 \]

\[ s = (2,3) \text{ action}=\text{tright}, \quad s' = (3,3), \quad r = -0.01 \]

\[ s = (3,3) \text{ action}=\text{tright}, \quad s' = (3,2), \quad r = -0.01 \]

\[ s = (3,2) \text{ action}=\text{tup}, \quad s' = (3,3), \quad r = -0.01 \]

\[ s = (3,3) \text{ action}=\text{tright}, \quad s' = (4,3), \quad r = 1 \]

Can use experience to estimate MDP T and R models

Estimate of \( T(\langle 1,2 \rangle, \text{tup}, \langle 1,3 \rangle) = 1 / 2 \)
• Given estimates of the transition model $T$ and reward model $R$, we can do MDP policy evaluation to compute the value of our policy
Model-Based Passive Reinforcement Learning

• Follow policy $\pi$

• Estimate MDP model parameters given observed transitions and rewards
  
  o If finite set of states and actions, can just count and average counts

• Use estimated MDP to do policy evaluation of $\pi$
Does This Give Us All the Parameters for a MDP?

- Follow policy $\pi$
- Estimate MDP model parameters given observed transitions and rewards
  - If finite set of states and actions, can just count and average counts
- Use estimated MDP to do policy evaluation of $\pi$
Start at (1,1)

s=(1,1) action= tup, s’=(1,2), r = -.01
s=(1,2) action= tup, s’=(1,2), r = -.01
s=(1,2) action=tup, s’=(1,3), r = -.01
s=(1,3) action=tright, s’=(2,3), r = -.01
s=(2,3) action=tright, s’=(3,3), r = -.01
s=(3,3) action=tright, s’=(3,2), r = -.01
s=(3,2) action=tup, s’=(3,3), r = -.01
s=(3,3) action=tright, s’=(4,3), r = 1

Estimate of T(<1,2>,tright,<1,3>)?
No idea! Never tried this action…

Adaption of drawing by Ketrina Yim
Does This Give Us All the Parameters for a MDP? No.  
Have all Parameters We Need! (After We Visit All States at Least Once)  

- Follow policy $\pi$  
- Estimate MDP model parameters given observed transitions and rewards  
  - If finite set of states and actions, can just count and average counts  
- Use estimated MDP to do policy evaluation of $\pi$
2 episodes of experience in MDP. Use to estimate MDP parameters & evaluate $\pi$
Start at (1,1)
s=(1,1) action= tup, s'=(1,2), r = -.01
s=(1,2) action= tup, s'=(1,2), r = -.01
s=(1,2) action=tup, s'=(1,3), r = -.01
s=(1,3) action=tright, s'=(2,3), r = -.01
s=(2,3) action=tright, s'=(3,3), r = -.01
s=(3,3) action=tright, s'=(3,2), r = -.01
s=(3,2) action=tup, s'=(3,3), r = -.01
s=(3,3) action=tright, s'=(4,3), r = 1

s=(1,1) action= tup, s'=(2,1), r = -.01
s=(2,1) action=tright, s'=(3,1), r = -.01
s=(3,1) action= tup, s'=(4,1), r = -.01
s=(4,1) action= tleft, s'=(3,1), r = -.01
s=(3,1) action= tup, s'=(3,2), r = -.01
s=(3,2) action= tup, s'=(4,2), r = -1

2 episodes of experience in MDP. Use to estimate MDP parameters & evaluate $\pi$

Is the computed policy value likely to be correct?

(1) Yes (2) No (3) Not sure
Start at (1,1)

s=(1,1) action= tup, s'=(1,2), r = -0.01

s=(1,2) action= tup, s'=(1,2), r = -0.01

s=(1,2) action= tup, s'=(1,3), r = -0.01

s=(1,3) action= tright, s'=(2,3), r = -0.01

s=(2,3) action= tright, s'=(3,3), r = -0.01

s=(3,3) action= tright, s'=(3,2), r = -0.01

s=(3,2) action= tup, s'=(3,3), r = -0.01

s=(3,3) action= tright, s'=(4,3), r = 1

s=(1,1) action= tup, s'=(2,1), r = -0.01

s=(2,1) action= tright, s'=(3,1), r = -0.01

s=(3,1) action= tup, s'=(4,1), r = -0.01

s=(4,1) action= tleft, s'=(3,1), r = -0.01

s=(3,1) action= tup, s'=(3,2), r = -0.01

s=(3,2) action= tup, s'=(4,2), r = -1

2 episodes of experience in MDP. Use to estimate MDP parameters & evaluate $\pi$

Is the computed policy value likely to be correct?

No. Approximation.. With limited Data, Poor Estimates

Adaption of drawing by Ketrina Yim
Passive Reinforcement Learning

Two Approaches

1. Build a model
2. Model-free: directly estimate $V^\pi$

Transition Model?

State

V^\pi(s1)=1.8, V^\pi(s2)=2.5,…

Reward model?

Agent

Carnegie Mellon University
Temporal Difference Learning

- No explicit model of T or R!
- But still estimate V and expectation **through samples**
- Update 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 sample of V: running average

Sample of V(s): \[ \text{sample} = R(s, \pi(s), s') + \gamma V^\pi(s') \]
Update to V(s): \[ V^\pi(s) \leftarrow (1 - \alpha)V^\pi(s) + (\alpha)\text{sample} \]

Slide adapted from Klein and Abbeel
Exponential Moving Avg

- 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:
    \[
    \bar{x}_n = \frac{x_n + (1 - \alpha) \cdot x_{n-1} + (1 - \alpha)^2 \cdot x_{n-2} + \ldots}{1 + (1 - \alpha) + (1 - \alpha)^2 + \ldots}
    \]
  - Forgets about the past
- Decreasing learning rate (\(\alpha\)) can give converging avgs

Slide adapted from Klein and Abbeel
TD Learning Example

Initialize all $V^\pi(s)$ values: $V^\pi(s) = 0$

<table>
<thead>
<tr>
<th>State</th>
<th>$V^\pi(s)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,1)</td>
<td>0</td>
</tr>
<tr>
<td>(1,2)</td>
<td>0</td>
</tr>
<tr>
<td>(1,3)</td>
<td>0</td>
</tr>
<tr>
<td>(4,1)</td>
<td>0</td>
</tr>
<tr>
<td>(2,1)</td>
<td>0</td>
</tr>
<tr>
<td>(2,3)</td>
<td>0</td>
</tr>
<tr>
<td>(3,1)</td>
<td>0</td>
</tr>
<tr>
<td>(3,2)</td>
<td>0</td>
</tr>
<tr>
<td>(3,3)</td>
<td>0</td>
</tr>
</tbody>
</table>
TD Learning Example \( \alpha=0.1, \gamma=1 \)

s=(1,1) action= tup, s’=(1,2), r = -.01

Update \( V^\pi((1,1)) \)

\[
\text{sample} = R(s, \pi(s), s') + \gamma V^\pi(s')
\]

\[
V^\pi(s) \leftarrow (1 - \alpha)V^\pi(s) + (\alpha)\text{sample}
\]

sample = -0.01 + \( V^\pi((1,2)) = -0.01 \)

\[
V^\pi((1,1)) = (1-\alpha) V^\pi((1,1)) + \alpha\cdot\text{sample}
\]

\[
= .9\cdot0 + 0.1\cdot-0.01 = -0.001
\]

<table>
<thead>
<tr>
<th>State</th>
<th>( V^\pi(s) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,1)</td>
<td>0</td>
</tr>
<tr>
<td>(1,2)</td>
<td>0</td>
</tr>
<tr>
<td>(1,3)</td>
<td>0</td>
</tr>
<tr>
<td>(4,1)</td>
<td>0</td>
</tr>
<tr>
<td>(2,1)</td>
<td>0</td>
</tr>
<tr>
<td>(2,3)</td>
<td>0</td>
</tr>
<tr>
<td>(3,1)</td>
<td>0</td>
</tr>
<tr>
<td>(3,2)</td>
<td>0</td>
</tr>
<tr>
<td>(3,3)</td>
<td>0</td>
</tr>
</tbody>
</table>
TD Learning Example \( \alpha=0.1, \gamma=1 \)

\( s=(1,1) \) action= tup, \( s'=(1,2) \), \( r = -0.01 \)

Update \( V^\pi((1,1)) \)

\[
\text{sample} = R(s, \pi(s), s') + \gamma V^\pi(s')
\]

\[
V^\pi(s) \leftarrow (1 - \alpha)V^\pi(s) + (\alpha) \text{sample}
\]

\[
\text{sample} = -0.01 + V^\pi((1,2)) = -0.01
\]

\[
V^\pi((1,1)) = (1-\alpha) V^\pi((1,1)) + \alpha \text{sample}
\]

\[
= .9 \times 0 + 0.1 \times -0.01 = -0.001
\]

<table>
<thead>
<tr>
<th>State</th>
<th>( V^\pi(s) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,1)</td>
<td>-0.001</td>
</tr>
<tr>
<td>(1,2)</td>
<td>0</td>
</tr>
<tr>
<td>(1,3)</td>
<td>0</td>
</tr>
<tr>
<td>(4,1)</td>
<td>0</td>
</tr>
<tr>
<td>(2,1)</td>
<td>0</td>
</tr>
<tr>
<td>(2,3)</td>
<td>0</td>
</tr>
<tr>
<td>(3,1)</td>
<td>0</td>
</tr>
<tr>
<td>(3,2)</td>
<td>0</td>
</tr>
<tr>
<td>(3,3)</td>
<td>0</td>
</tr>
</tbody>
</table>
TD Learning Example $\alpha=0.1$, $\gamma=1$

$s=(1,1)$ action= tup, $s'=(1,2)$, $r = -0.01$
$s=(1,2)$ action= tup, $s'=(1,2)$, $r = -0.01$

<table>
<thead>
<tr>
<th>State</th>
<th>$V^\pi(s)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,1)</td>
<td>-0.001</td>
</tr>
<tr>
<td>(1,2)</td>
<td>0</td>
</tr>
<tr>
<td>(1,3)</td>
<td>0</td>
</tr>
<tr>
<td>(4,1)</td>
<td>0</td>
</tr>
<tr>
<td>(2,1)</td>
<td>0</td>
</tr>
<tr>
<td>(2,3)</td>
<td>0</td>
</tr>
<tr>
<td>(3,1)</td>
<td>0</td>
</tr>
<tr>
<td>(3,2)</td>
<td>0</td>
</tr>
<tr>
<td>(3,3)</td>
<td>0</td>
</tr>
</tbody>
</table>
Problems with Passive Learning

• Agent wants to ultimately learn to act to gather high reward in the environment.
• If have a deterministic policy, gives no experience for other actions
Recall Model-Based Passive Reinforcement Learning

• Follow policy $\pi$
• Estimate MDP model parameters given observed transitions and rewards
  ○ If finite set of states and actions, can just count and average counts
• Use estimated MDP to do policy evaluation of $\pi$
Can We Learn Optimal Values & Policy?

• Consider acting randomly in the world
• Can such experience allow the agent to learn the optimal values and policy?
Model-Based RL w/ Random Actions

- Choose actions randomly
- Estimate MDP model parameters given observed transitions and rewards
  - If finite set of states and actions, can just count and average counts
- Use estimated MDP to compute estimate of optimal values and policy

Will the computed values and policy converge to the true optimal values and policy in the limit of infinite data?
Reachability

• When acting randomly forever, still need to be able to visit each state and take each action many times
• Want all states to be reachable from any other state
• Quite mild assumption but doesn’t always hold

Model-Free Learning with Random Actions?

- Previously introduced temporal-difference learning for policy evaluation:
  - As act in the world visit \((s, a, r, s', a', r', \ldots)\)
  - Update \(V^\pi\) estimates at each step

Sample of \(V^\pi(s)\):

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

Update to \(V^\pi(s)\):

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

- Over time updates mimic Bellman updates

Slide adapted from Klein and Abbeel
Q-Learning

- Running estimate of state-action Q values (instead of V in TD learning)
- Update Q(s,a) every time experience (s,a,s’,r(s,a,s’))
  - Consider old estimate Q(s,a)
  - Create new sample estimate
    \[
    \text{sampleQ} \ (s,a) = R(s,a,s') + \gamma \max_{a'} Q(s',a')
    \]
  - Update estimate of Q(s,a)
    \[
    Q(s,a) = (1 - \alpha)Q(s,a) + \alpha \times \text{sampleQ}(s,a)
    \]
Q-Learning

• Update $Q(s,a)$ every time experience $(s,a,s',r(s,a,s'))$

$$sampleQ\ (s,a) = R(s,a,s') + \gamma \max_a Q(s',a')$$

$$Q(s,a) = (1 - \alpha)Q(s,a) + \alpha * sampleQ(s,a)$$

• Intuition: using samples to approximate
  
  o Future rewards
  
  o Expectation over next states due to transition model uncertainty
Q-Learning Properties

• If acting randomly, Q-learning converges to optimal state—action values, and also therefore finds optimal policy

• Off-policy learning
  o Can act in one way
  o But learning values of another policy (the optimal one!)
Towards Gathering High Reward

- Fortunately, acting randomly is sufficient, but not necessary, to learn the optimal values and policy
Leveraging Learned Values

• Initialize $s$ to a starting state
• Initialize $Q(s,a)$ values
• For $t=1,2,...$
  o Choose $a = \text{argmax } Q(s,a)$
  o Observe $s',r(s,a,s')$
  o Update/Compute $Q$ values (using model-based or $Q$-learning approach)
To Explore or Exploit?
**E-greedy**

- With probability $1-e$
  - Choose $\text{argmax}_a Q(s,a)$
- With probability $e$
  - Select random action

- Guaranteed to compute optimal policy
- But even after millions of steps still won’t always be following policy compute (the argmax $Q(s,a)$)
Greedy in Limit of Infinite Exploration (GLIE)

- Greedy in the Limit of Infinite Exploration
- E-Greedy approach
- But decay epsilon over time
- Eventually will be following optimal policy almost all the time
• AI in the real world: Flappy Bird

• http://sarvagyavaish.github.io/FlappyBirdRL/
Generalization in RL

• Q Learning uses Tables for each state
• Works well in small space (e.g. small maze), but in large spaces we may not be able to reasonably visit every state to get estimate
• Treat Q Learning as Learning a Function Approximator
  o Linear Functions
  o Non-Linear.. Neural Nets
• Compression achieved by function approximation allows the learning agent to generalize from states it has visited to states it has NOT visited (More important than space saving)
  o Alpha Go, Atari Games, ..