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

This slide deck courtesy of Dan Klein at UC Berkeley
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: Animal Learning

- RL studied experimentally for more than 60 years in psychology
  - Rewards: food, pain, hunger, drugs, etc.
  - Mechanisms and sophistication debated

- Example: foraging
  - Bees learn near-optimal foraging plan in field of artificial flowers with controlled nectar supplies
  - Bees have a direct neural connection from nectar intake measurement to motor planning area
Example: Backgammon

- Reward only for win / loss in terminal states, zero otherwise
- TD-Gammon learns a function approximation to $V(s)$ using a neural network
- Combined with depth 3 search, one of the top 3 players in the world

- You could imagine training Pacman this way…
- … but it’s tricky!  (It’s also P3)
Passive RL

- **Simplified task**
  - You are given a 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
  - … 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…
Example: Direct Evaluation

- **Episodes:**

  - (1,1) up -1  
  - (1,2) up -1  
  - (1,2) up -1  
  - (1,3) right -1  
  - (2,3) right -1  
  - (2,3) right -1  
  - (3,3) right -1  
  - (3,2) up -1  
  - (3,2) up -1  
  - (4,2) exit -100  
  - (3,3) right -1  
  - (3,3) right -1  
  - (done)  
  - (4,3) exit +100  
  - (done)  

  - $V(2,3) \sim \frac{96 + (-103)}{2} = -3.5$  
  - $V(3,3) \sim \frac{99 + 97 + (-102)}{3} = 31.3$
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 $R$

\[
V^\pi_0(s) = 0
\]

\[
V^\pi_{i+1}(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V^\pi_i(s')]
\]
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

\[
V_{i+1}^\pi(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_i^\pi(s')]
\]
Example: Model-Based Learning

- **Episodes:**

  (1,1) up -1  (1,1) up -1
  (1,2) up -1  (1,2) up -1
  (1,2) up -1  (1,3) right -1
  (1,3) right -1  (2,3) right -1
  (2,3) right -1  (3,3) right -1
  (3,3) right -1  (3,2) up -1
  (3,2) up -1  (4,2) exit -100
  (3,3) right -1  (done)
  (4,3) exit +100  (done)

\[ T(<3,3>, \text{right}, <4,3>) = \frac{1}{3} \]

\[ T(<2,3>, \text{right}, <3,3>) = \frac{2}{2} \]
### Example: Expected Age

**Goal:** Compute expected age of cs343 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 \([a_1, a_2, \ldots, a_N]\)

<table>
<thead>
<tr>
<th>Unknown P(A): “Model Based”</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \hat{P}(a) = \frac{\text{num}(a)}{N} ]</td>
</tr>
<tr>
<td>[ E[A] \approx \sum_a \hat{P}(a) \cdot a ]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Unknown P(A): “Model Free”</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ E[A] \approx \frac{1}{N} \sum_i a_i ]</td>
</tr>
</tbody>
</table>
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

$$x_i \sim P(x)$$

$$\hat{P}(x) = \text{num}(x) / N$$

$$E[f(x)] \approx \sum_x \hat{P}(x) f(x)$$

- Model-free: estimate expectation directly from samples

$$x_i \sim P(x)$$

$$E[f(x)] \approx \frac{1}{N} \sum_i f(x_i)$$

- Why does this work? Because samples appear with the right frequencies!
Sample-Based Policy Evaluation?

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

- Who needs T and R? Approximate the expectation with samples of $s'$ (drawn from T!)

$$\begin{align*}
\text{sample}_1 &= R(s, \pi(s), s'_1) + \gamma V_i^\pi(s'_1) \\
\text{sample}_2 &= R(s, \pi(s), s'_2) + \gamma V_i^\pi(s'_2) \\
&\vdots \\
\text{sample}_k &= R(s, \pi(s), s'_k) + \gamma V_i^\pi(s'_k) \\
\end{align*}$$

$$V_{i+1}^\pi(s) \leftarrow \frac{1}{k} \sum_i \text{sample}_i$$

Almost! But we can't rewind time to get sample after sample from state $s$. 
Tempor​al-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 $V(s)$: 

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

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
    \[ \bar{x}_n = \frac{x_n + (1 - \alpha) \cdot x_{n-1} + (1 - \alpha)^2 \cdot x_{n-2} + \ldots}{1 + (1 - \alpha) + (1 - \alpha)^2 + \ldots} \]
  - Forgets about the past (distant past values were wrong anyway)

- Decreasing learning rate can give converging averages
Example: TD Policy Evaluation

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

(1,1) up -1
(1,2) up -1
(1,2) up -1
(1,3) right -1
(1,3) right -1
(2,3) right -1
(2,3) right -1
(3,3) right -1
(3,3) right -1
(3,2) up -1
(3,2) up -1
(4,2) exit -100
(4,3) exit +100
(done)

Take \( \gamma = 1 \), \( \alpha = 0.5 \)
Problems with TD Value Learning

- TD value leaning is a model-free way to do policy evaluation.
- 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 directly.
- Makes action selection model-free too!
Active RL

- Full reinforcement learning
  - You don’t know the transitions $T(s,a,s')$
  - You don’t know the rewards $R(s,a,s')$
  - You can choose any actions you like
  - Goal: learn the optimal policy / values
  - … what value iteration did!

- 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 approx optimal values
  - Start with $V_0^*(s) = 0$, which we know is right (why?)
  - Given $V_i^*$, calculate the values for all states for depth $i+1$:
    \[
    V_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma V_i(s') \right]
    \]

- But Q-values are more useful!
  - Start with $Q_0^*(s,a) = 0$, which we know is right (why?)
  - Given $Q_i^*$, calculate the q-values for all q-states for depth $i+1$:
    \[
    Q_{i+1}(s, a) \leftarrow \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma \max_{a'} Q_i(s', a') \right]
    \]
Q-Learning

- Q-Learning: sample-based Q-value iteration
- Learn $Q^*(s,a)$ values
  - Receive a sample $(s,a,s',r)$
  - Consider your old estimate: $Q(s,a)$
  - Consider your new sample estimate:
    $$Q^*(s,a) = \sum_{s'} T(s,a,s') \left[ R(s,a,s') + \gamma \max_{a'} Q^*(s',a') \right]$$
    $$sample = R(s,a,s') + \gamma \max_{a'} Q(s',a')$$
  - Incorporate the new estimate into a running average:
    $$Q(s,a) \leftarrow (1 - \alpha)Q(s,a) + (\alpha) \left[ sample \right]$$
Q-Learning Properties

- Amazing result: Q-learning converges to optimal policy
  - If you explore enough
  - If you make the learning rate small enough
  - … but not decrease it too quickly!
  - Basically doesn’t matter how you select actions (!)

- Neat property: off-policy learning
  - learn optimal policy without following it (some caveats)
Exploration / Exploitation

- Several schemes for forcing exploration
  - Simplest: random actions ($\varepsilon$ greedy)
    - Every time step, flip a coin
    - With probability $\varepsilon$, act randomly
    - With probability $1-\varepsilon$, act according to current policy

- Problems with random actions?
  - You do explore the space, but keep thrashing around once learning is done
  - One solution: lower $\varepsilon$ over time
  - Another solution: exploration functions
Q-Learning

- Q-learning produces tables of q-values:

![Q-Values after 1000 episodes](image)
Exploration Functions

- When to explore
  - Random actions: explore a fixed amount
  - Better idea: explore areas whose badness is not (yet) established

- Exploration function
  - Takes a value estimate and a count, and returns an optimistic utility, e.g. $f(u, n) = u + k/n$  
    (exact form not important)

$$Q_{i+1}(s, a) \leftarrow \alpha R(s, a, s') + \gamma \max_{a'} Q_i(s', a')$$

$$Q_{i+1}(s, a) \leftarrow \alpha R(s, a, s') + \gamma \max_{a'} f(Q_i(s', a'), N(s', a'))$$
The Story So Far: MDPs and RL

Things we know how to do:

- If we know the MDP
  - Compute $V^*$, $Q^*$, $\pi^*$ exactly
  - Evaluate a fixed policy $\pi$

- If we don’t know the MDP
  - We can estimate the MDP then solve
    - We can estimate $V$ for a fixed policy $\pi$
    - We can estimate $Q^*(s,a)$ for the optimal policy while executing an exploration policy

Techniques:

- Model-based DPs
  - Value Iteration
  - Policy evaluation

- Model-based RL
- Model-free RL
  - Value learning
  - Q-learning