Last time

- Markov decision processes
- Computing the optimal policy
  - value iteration
  - policy iteration
Still have an MDP:
- A set of states
- A set of actions (per state) $A$
- A model $T(s,a,s')$
- A reward function $R(s,a,s')$

Still looking for an optimal policy $\pi^*(s)$

New twist: don’t know $T$ and/or $R$, but can observe $R$
- Learning by doing
- can have multiple episodes (trials)
Example: animal learning

- Studied experimentally for more than 60 years in psychology
  - Rewards: food, pain, hunger, drugs, etc.
What can you do with RL

- Stanford autonomous helicopter
  
  http://heli.stanford.edu/
Choosing a treasure box

- First time playing this game
  - Can only open one chest

- Let’s cheat by googling a walkthrough
  - A: 1, 10, 10, 10, 1, 10, 10, 10, 10,...
  - B: 1, 1, 1, 100, 1, 100, 1, 1, 1, 1, 1, 1, 1, 1,...
Reinforcement learning methods

- **Model-based learning**
  - learn the model of MDP (transition probability and reward)
  - compute the optimal policy as if the learned model is correct

- **Model-free learning**
  - learn the optimal policy without explicitly learning the transition probability
  - Q-learning: learn the Q-state $Q(s,a)$ directly
Choosing a treasure box

Let’s cheat by googling a walkthrough

- A: 1, 10, 10, 10, 1, 10, 10, 10, 10, 10, ...
- B: 1, 1, 1, 100, 1, 100, 1, 1, 1, 1, ...

Model-based

- $p(A=1)=2/10$, $p(A=10)=8/10$
- $p(B=1)=8/10$, $p(B=100)=2/10$

Model-free

- $E(A)=8.2$, $E(B)=20.8$
Model-Based Learning

➢ Idea:
  • Learn the model empirically in trials (episodes)
  • 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

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

Trials:

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

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

\[ \gamma = 1 \]

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

\[ T(<2,3>, \text{right}, <3,3>) = \frac{2}{2} \]
Sample-Based Policy Evaluation

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

- Approximate the expectation with samples (drawn from an unknown \( T! \))

\[
\begin{align*}
\text{sample}_1 &= R(s, \pi(s), s'_1) + \gamma V_i^{\pi}(s'_1) \\
\text{sample}_2 &= R(s, \pi(s), s'_2) + \gamma V_i^{\pi}(s'_2) \\
&\vdots \\
\text{sample}_k &= R(s, \pi(s), s'_k) + \gamma V_i^{\pi}(s'_k) \\
V_{i+1}^{\pi}(s) &\leftarrow \frac{1}{k} \sum_{i} \text{sample}_i
\end{align*}
\]

Almost! But we cannot rewind time to get samples from \( s \) in the same trial.
**Temporal-Difference Learning**

- **Big idea:** learn from every experience!
  - Update $V_\pi(s)$ each time we experience $(s,a,s',R)$
  - Likely $s'$ will contribute updates more often

- **Temporal difference learning**
  - Policy still fixed

- Sample of $V(s)$: \[ \text{sample} = R(s, \pi(s), s') + \gamma V_\pi(s') \]
- Update: \[ V_\pi(s) \leftarrow (1 - \alpha) V_\pi(s) + \alpha \text{ sample} \]
- Same as: \[ V_\pi(s) \leftarrow V_\pi(s) + \alpha (\text{sample} - V_\pi(s)) \]
Exponential Moving Average

- Exponential moving average
  - Makes recent samples more important
    \[ x_n = \frac{x_n + (1-\alpha)x_{n-1} + (1-\alpha)^2x_{n-2} + \cdots}{1 + (1-\alpha) + (1-\alpha)^2 + \cdots} \]
  - Forgets about the past (distant past values were wrong anyway)
  - Easy to compute from the running average
    \[ \bar{x}_n = (1-\alpha)x_{n-1} + \alpha x_n \]

- Decreasing learning rate can give converging averages
Problems with TD Value Learning

- TD value learning does not do policy evaluation.
- Hard to turn values into a (new) policy.

Idea: learn Q-value directly.

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

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

- Makes action selection model-free too!
Active Learning

- 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

- In this case:
  - Learner makes choices!
  - Fundamental tradeoff: exploration vs. exploitation
    - exploration: try new actions
    - exploitation: focus on optimal actions based on the current estimation
  - This is NOT offline planning! You actually take actions in the world and see what happens…
Detour: Q-Value Iteration

Value iteration: find successive approx optimal values
- Start with \( V_0(s) = 0 \)
- 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 \( Q_0(s,a) = 0 \)
- 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 $(s,a,s')$ with reward $R(s,a,s')$
- Consider your old estimate: $Q(s,a)$
- Consider your new sample estimate:
  \[
  \text{sample} = R(s,a,s') + \gamma \max_a Q(s',a')
  \]
- Incorporate the new estimate into a running average
  \[
  Q(s,a) \leftarrow (1-\alpha) Q(s,a) + \alpha \text{ sample}
  \]

17
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
Q-Learning

Q-learning produces tables of Q-values:
Several schemes for forcing exploration

- Simplest: random actions ($\epsilon$ greedy)
  - Every time step, flip a coin
  - With probability $\epsilon$, act randomly
  - With probability $1-\epsilon$, 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 $\epsilon$ over time
  - Another solution: exploration functions
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)

$$\text{sample} = R(s, a, s') + \gamma \max_{a'} f\left(Q_i(s', a'), N(s', a')\right)$$
Recap

- Model-based learning
  - using sampling to learn T and R
  - policy iteration

- Model-free learning
  - directly learn Q-states
  - Q learning
Overview of Project 3

➤ MDPs
  • Q1: value iteration
  • Q2: find parameters that lead to certain optimal policy
  • Q3: similar to Q2

➤ Q-learning
  • Q4: implement the Q-learning algorithm
  • Q5: implement $\epsilon$ greedy action selection
  • Q6: try the algorithm

➤ Approximate Q-learning and state abstraction
  • Q7: Pacman

➤ Tips
  • make your implementation general