Dealing with Unknown MDP

- So let's suppose you don't know the MDP.
- But the state space is not too large — can still implement a tabular methods.
- So what can you do? Well, you can interact with the MDP.
- Example to keep in mind: computer game.
  Can be modeled as an MDP, but it's a pain to really do this.
  Imagine you somehow have so much computational power that a tabular method is OK.
- What would you do?
- Let's explore several basic ideas. We'll largely follow the Sutton + Barto textbook here: http://incompleteideas.net/book/RLbook2020.pdf

Dealing with Unknown MDP

- OK, so our goal is to implement some version of

  $J \leftarrow TJ$

  or

  $$J_{t+1}(s) = \max_a r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) J_t(s')$$

  ...but we don't know $P(s' \mid s,a)$ and we don't know $r(s,a)$.
- At an intuitive level, we need to learn to do something close to this update from experience.
- One possibility: learn all the $P(s' \mid s,a)$, $r(s,a)$.
- ...too much work. Is there something better?

- Let's introduce another definition.
- Just like

$$J_\pi(s) = E_{s,\pi}\left[\sum_{t=0}^{+\infty} \gamma^t r_t\right]$$

is the expected discounted stream of rewards after following policy π from state s, we can define

$$Q_\pi(s,a) = E_{s,a,\pi}\left[\sum_{t=0} \gamma^t r_t\right]$$

be the expected discounted stream of rewards after following policy π after choosing action a in state s.

- To clarify, suppose $\pi(1) = 2$. Can still talk about $Q_\pi(s = 1, a = 1)$. This is the expected discounted stream of rewards after you take action 1 in state 1, and follow policy π thereafter.
- In both cases, if we remove the π and put a star, we are talking about value and q-value under the optimal policy.

- Consider an MDP with a single state and two actions. Any action brings you back to the current state. Action 1 has a reward of 1, action 2 has a reward of 2, discount factor is 1/2.
- Consider the policy π of always choosing action 1.
- $Q_\pi(s = 1, a = 1) = 1 + 1 \cdot (1/2) + 1(1/2)^2 + \cdots = 2.$
- $Q_\pi(s = 1, a = 2) = 2 + 1 \cdot (1/2) + 1 \cdot (1/2)^2 + \cdots = 3$

- Consider an MDP with a single state and two actions. Any action brings you back to the current state. Action 1 has a reward of 1, action 2 has a reward of 2, discount factor is 1/2.
- Consider the policy π of always choosing action 2.
- $Q_\pi(s = 1, a = 1) = 1 + 2 \cdot (1/2) + 2(1/2)^2 + \cdots = 3.$
- $Q_\pi(s = 1, a = 2) = 2 + 2 \cdot (1/2) + 2 \cdot (1/2)^2 + \cdots = 4$

- You have one state. You have two actions. Action 1 gives you a reward of one. Action 2 gives you a reward of two.
- Your policy π is to choose the action with a 50-50 probability.
- At any time step except the initial one, we have that $E[r_t] = 3/2$.
- If you take action 1 at the initial time step, your expected discounted reward will be

$$Q_\pi(s = 1, a = 1) = 1 + \gamma(3/2) + \gamma^2(3/2) + \cdots = 1 + \gamma\frac{3/2}{1-\gamma}.$$

- If you choose action 2 at the initial time step and then follow the policy, your expected discount reward will be $Q_\pi(s = 1, a = 2) = 2 + \gamma\frac{3/2}{1-\gamma}$
- $J_\pi(s = 1) = 1.5 + \gamma\frac{3/2}{1-\gamma}$

- Suppose you knew all the Q-values under the optimal policy in a state.
- That is, you know $Q_*(s, a)$ for all actions a in state s under the best policy π_*.
- What action should you take?
- Optimal action: $\pi_*(s) = \arg\max_a Q_*(s, a)$ when a unique action achieves the maximum.
- If several actions achieves the maximum, $\pi_*(s)$ could be any of them, or any distribution over them.

- Recall: given any two random variables X, Y, we have that

$$\sum_y E[X \mid Y = y]P(Y = y) = E[X]$$

- Sometimes written as $E[E[X \mid Y]] = E[X]$
- Example: Your grade on the midterm is either 100,75, or 50, all with equal probability?
- Your grade in the class is equal to your grade on the midterm with probability 50%, and equal to the two other options with probability 25%.
- For example, if you get 100% on the midterm, then with probability 50% you get 100 in the class, and with probability 25% you get 75, and with probability 25% you get 50.
- What is your expected grade in the class?
- Let X be your grade in the class, Y be your grade on the midterm.

$$E[X] = \sum_y E[X \mid Y = y]P(Y = y)$$

- $$= \left(\left(100 \cdot \frac{1}{2} + 75 \cdot \frac{1}{4} + 50 \cdot \frac{1}{4} \right) \cdot \frac{1}{3} + \left(75 \cdot \frac{1}{2} + 100 \cdot \frac{1}{4} + 50 \cdot \frac{1}{4} \right) \cdot \frac{1}{3} + \left(50 \cdot \frac{1}{2} + 100 \cdot \frac{1}{4} + 75 \cdot \frac{1}{4} \right) \cdot \frac{1}{3} \right)$$

- Let's revisit the equation $V^*(s) = \max_a r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^*(s')$
- Your TOTAL discounted reward after choosing action a at state s at time 0 and following the best policy, is a random variable. Let's call it X. Then $E[X] = Q^*(s,a)$.
- Let Y be your state at time 1.
- $E[X] = \sum_y E[X \mid Y = y] P(Y = y)$
- $Q^*(s,a) = \sum_{s'} E\left(\text{ reward from choosing } a \text{ in state } s + \gamma \text{ all future rewards starting from } s'\right) P(s' \mid s,a)$
- $Q^*(s,a) = \sum_{s'} \left(r(s,a) + \gamma V^*(s')) P(s' \mid s,a)\right)$
- $Q^*(s,a) = r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^*(s')$
- $V^*(s) = \max_a Q^*(s,a)$
- $V^*(s) = \max_a r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^*(s')$

- Suppose you knew all the Q-values under some policy π.

  That is, you know $Q_\pi(s, a)$.

  What is $J_\pi(s)$?
- Well, with probability $\pi(a \mid s)$ you choose action a in state s.
- If you choose action a, your expected-discounted-sum-of-rewards (reward-to-go) is $Q_\pi(s, a)$.
- So $J_\pi(s) = \sum_a \pi(a \mid s) Q_\pi(s, a)$
- Let's get a feel for this. What is $\sum_a \pi(a \mid s)$?
- So $J_\pi(s)$ is a sum of all the $Q_\pi(s, a)$ multiplied by nonnegative numbers that add up to one. This is called a **convex combination.**

- OK, we now have the concept of a Q-value. Now let's address our main question: how do we learn the optimal policy without knowing the MDP?
- In other words, we want to learn J^* without knowing the transition probabilities $P(s, a)$ or even the rewards $r(s, a)$.
- Actually, let's first consider an easier problem: learning J_π for a policy π.
- Need to understand a few bits about recursions first.
- Consider the recursion:

  $$y_1 = x_1$$
  $$y_t = y_{t-1} + \frac{1}{t}(x_t - y_{t-1}).$$

  What is y_t in terms of x_t?

- $y_1 = x_1$
  $y_2 = y_1 + \frac{1}{2}(x_2 - y_1)$
  $y_3 = y_2 + \frac{1}{3}(x_3 - y_2)$

- Consider the recursion:

  $$y_1 = x_1$$
  $$y_t = y_{t-1} + \frac{1}{t}(x_t - y_{t-1}).$$

  What is y_t in terms of x_t?

- Indeed, y_t is the running average: $y_t = \frac{x_1 + \cdots x_t}{t}$.

- Proof by induction: holds at $t = 1$. Then
  $$y_t = \frac{1}{t}x_t + \frac{t-1}{t}y_{t-1} = \frac{1}{t}x_t + \frac{t-1}{t}\frac{x_1 + \cdots + x_{t-1}}{t-1} = \frac{x_1 + \cdots + x_t}{t}.$$

- This is a good way to keep track of a running average when you don't know how many iterations you'll do ahead of time.

- Can consider the iteration

  $$y_1 = x_1$$

  $$y_t = y_{t-1} + \alpha_t(x_t - y_{t-1}).$$

- Intuition: you get "nudged" in the direction of the latest x_t with step-size α_t.
- When $\alpha_t = 1/t$, your nudges decay to zero in strength...which makes sense, since any one sample has less and less of an effect on sample average over time.
- But we can also consider $\alpha_t = 1/\sqrt{t}$. This tends to over-weight more recent samples compared to the previous ones.
- An extreme is $\alpha_t = \alpha$ where $\alpha \in (0,1)$. In this case,

  $$y_t = (1 - \alpha)y_{t-1} + \alpha x_t$$

- There's a particular case worth considering which is good for building your intuition here.
- Suppose we have a random variable X with unknown mean m. We want to estimate m. We can sample from X.
- One approach: we generate an infinite sequence $X_1, X_2, X_3, \ldots$ of i.i.d samples from X.
- Initialize $y_0 = X_1$. Update as $y_t = y_{t-1} + \alpha_t \left(X_t - y_t\right)$.
- We know that $y_t = \dfrac{X_1 + \cdots X_t}{t}$ when $\alpha_t = 1/t$. What happens in the limit as $t \to +\infty$?
- Interpretation: at every time, we "nudge" ourselves towards a random sample that has expectation m. The strength of the nudges decays to zero over time. This results in convergence to the unknown mean m.

- OK, so we have a policy π.

  We want to evaluate $J_\pi(s)$ for various states s. The method we describe is called **temporal difference learning**.
- Suppose we run an iterative process. By step t of this process, we have somehow obtained a vector J_t which we think of as an estimate of J_π.
- Now at step t, we choose action a_t in state s_t and transition to state s_t'.

  We get a reward of r_t from this.
- Key idea: view $r_t + \gamma J_t(s_t')$ as new information about the value of state s_t.
- Update as $J_{t+1}(s_t) = J_t(s_t) + \alpha_t(r_t + \gamma J_t(s_t') - J_t(s_t))$.

  For all states $s \neq s_t$, just keep the previous estimate: $J_{t+1}(s) = J_t(s)$.

- Let's discuss this on a simple example — an MDP with one state and one action.
  Let's say the reward is one, the discount factor is 1/2.
- We initialize $J_1 = 0$.
  We take the step-size $\alpha_t = 1/t$.
  The true value is two.
- Update: $J_{t+1}(s_t) = J_t(s_t) + \alpha_t(r_t + \gamma J_t(s_t') - J_t(s_t))$
- After one step, we will have
  $J_2 = J_1 + \frac{1}{1}(r_1 + \gamma J_1 - J_1)$
  $J_2 = 0 + \frac{1}{1}(1 + \gamma \cdot 0 - 0) = 1.$
- Then $J_3 = 1 + \frac{1}{2}(1 + \gamma \cdot 1 - 1) = \frac{5}{4}$
- In general, $J_{t+1} = J_t + \frac{1}{t}(1 + \frac{1}{2}J_t - J_t) = J_t + \frac{1}{t}\left(1 - \frac{J_t}{2}\right)$
- Observe that as long $J_t < 2$, this is increasing.
  Not hard to show that $\lim_t J_t = 2$.

- In general, things are not always deterministic.
- Consider the same MDP, but reward is 1 with probability 1/2 and 2 with probability 1/2.
- So with probability 1/2 we will have

  $$J_{t+1} = J_t + \alpha_t(1 + \gamma J_t - J_t)$$

  and with probability 1/2 we will have

  $$J_{t+1} = J_t + \alpha_t(2 + \gamma J_t - J_t).$$
- Why should this work?
- Intuition: roughly half the time the reward will be one, roughly half the time it will be two, so it should not be all that different from having the reward always be equal to 3/2.

- We were a little bit vague about how to choose the state s_t at each time step.
- Natural choice: generate a path from the policy π and then go through all the states on that path.
- That is, generate a path $s_0, a_0, r_1, s_1, a_1, s_2, \ldots$
  and then go through the path applying the the temporal difference update to every pair of successive states.
- Alternatively: generate some collection of paths and do this.
- By the way, this is called **TD(0) or temporal difference learning.**
- If the problem is terminal, then whenever the game terminates, just restart.
- **Theorem**: this works. More formally, if with probability one you visit every state infinitely often, then $J_t(s) \to J_\pi(s)$ for all states s.
- This is not obvious and kind of deep if you think about it.
- If time permits, we will go back and prove this result towards the end of the class.

- Summary: last time we discussed the temporal difference update

  $J_{t+1}(s_t) = J_t(s_t) + \alpha_t \left(r_t + \gamma J_t(s_t') - J_t \right)$

  where s_t, a_t, r_t, s_t' are, respectively,

  the state at time t;

  the action taken at time t, which must be taken from policy π;

  the reward at time t;

  the state at the next time step;

  and

  $J_{t+1}(s) = J_t(s)$ for all $s \neq s_t$.

- Can generate one super long path from π and go through it implementing the above update.

  Or: can generate many short paths.

- Theorem: if every state s is visited infinitely often, then $J_t(s) \to J_\pi(s)$ for all states s.
- Punchline: can do policy evaluation just by generating long chains of state-action-reward-next-state; don't need to know the transition probabilities.

- Let's think a little bit more about how we came up with

  $$J_{t+1}(s_t) = J_t(s_t) + \alpha_t \left(r_t + \gamma J_t(s_t') - J_t \right)$$

- Key idea: if $X_1, X_2, \ldots$ is a sequence of random variables of unknown mean m, then the recursion

  $$y_1 = X_1$$

  $$y_t = y_{t-1} + \frac{1}{t}(X_t - y_{t-1})$$

  converges to m.

- Intuition: think of

  $$X_i = m + (X_i - m)$$

  ...each X_i is "noisy information" about m.

- Intuition: pretend that $r_t + \gamma J_t(s_t')$ is "noisy information" about $J_\pi(s_t)$.
- ....OK, this intuition kind of works. Let's think a little bit deeper about what is going on here.

- Intuition: we are looking for J_π which is the fixed point

  $$J_\pi = T_\pi J_\pi$$

- We find that fixed point with the update

  $$J_{t+1} = T_\pi J_t$$

  This is policy evaluation, which we have already covered.

- We can write this out as

  $$J_{t+1}(s) = \sum_a \pi(a \mid s) r(s,a) + \gamma \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s, a) J_t(s')$$

- We can’t do this!

- Instead of

  $$J_{t+1}(s) = \sum_a \pi(a \mid s) r(s,a) + \gamma \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s, a) J_t(s')$$we

  update

  $$J_{t+1}(s_t) = J_t(s_t) + \alpha_t \left(r_t + \gamma J_t(s_t') - J_t(s_t) \right)$$

- At each step, we nudge $J_t(s_t)$ towards the random quantity $r_t + \gamma J_t(s_t')$.
- In expectation, this quantity is what $J_{t+1}(s_t)$ really should be:

  $$E[r_t + \gamma J_t(s_t')] = \sum_a \pi(a \mid s) r(s_t, a) + \gamma \sum_a \pi(a \mid s_t) \sum_{s'} P(s' \mid s_t, a) \gamma J_t(s')$$

- So the update has the right expectation!

- Next: do the same for Q-values.
- Plan:

  (1) Write Q-values as a fixed point of something.

  (2) Create a method to compute this fixed point by applying that something.

  Will need to know the MDP to apply this.

  (3) Write down a recursion that keeps nudging towards a random quantity that has the same expectation as the recursion in part (2).

- We can also apply a similar idea to estimate Q-values!
- The starting point for the logic on the previous page was the equation

  $J_\pi = T_\pi J_\pi$

  which can be written as

  $$J_\pi(s) = \sum_a \pi(a \mid s) r(s,a) + \gamma \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s, a) J_\pi(s')$$

- Can interpret it by conditioning!
- Total discounted reward overall time following policy π = immediate reward + $\gamma \cdot$ (reward starting from the next time step)
- Apply this logic to the Q-value:

  $$Q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} P(s' \mid s, a) J_\pi(s')$$

  $$= r(s,a) + \gamma \sum_{s'} P(s' \mid s, a) \sum_{a'} \pi(a' \mid s') Q_\pi(s', a')$$

- Let's use our logic to describe we can learn Q-values from data.

- For simplicity, suppose π is deterministic.
- Then

  $$Q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) \sum_{a'} \pi(a' \mid s') Q_\pi(s',a')$$

  simplifies

  $$Q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) Q_\pi(s', \pi(s'))$$

- So to compute $Q_\pi(s,a)$ we really want to update as

  $$Q_{t+1}(s,a) = r(s,a) + \sum_{s'} P(s' \mid s,a) Q_t(s', \pi(s')) \quad (*)$$

- ....but for this, we need to know the MDP.
- But if we are at state s_t, take action a_t, and move to state s'_t, and then take action $\pi(s'_t)$, then the quantity

  $$r_t + \gamma Q_t(s'_t, a'_t)$$

  has the right expectation:

  $$E[r_t + \gamma Q_t(s'_t, a'_t)] = r(s_t, a_t) + \gamma \sum_{s'} P(s' \mid s_t, a_t)) Q_t(s'_t, \pi(s'_t))$$

- So what we will do: at every step, nudge yourself in the direction of

  $r(s_t, a_t) + \gamma Q_t(s_{t+1}, \pi(s_{t+1}))$
- First, generate a sample path

  $s_0, a_0, r_1, s_1, a_1, r_2, \ldots$

  (or multiple sample paths if the MDP is terminal)

  by taking actions from policy π.
- Update as

  $$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t \left[r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \right]$$
- Theorem: this will converge to the correct Q-values, provided every state-action pair is chosen infinitely often.
- We will not give a formal proof.

- Next: do the same for Q-values **under the best policy**.
- Plan:

  (1) Write Q^* as a fixed point of something.

  (2) Create a method to compute this fixed point by applying that "something."

  Will need to know the MDP to apply this.

  (3) Write down a recursion that keeps nudging towards a random quantity that has the same expectation as the recursion in part (2).

- Start for deriving a fixed point equation for the Q-values under the best policy.
- $Q^*(s, a)$ is the expected sum of all future rewards, discounted.
- That divides into (immediate reward) + γ (sum of discounted rewards starting at the next time step)
- The immediate reward is $r(s, a)$.
- With probability $P(s' \mid s, a)$ you you transition to state s'.
- Once there, what action do you take under the best policy?
- You take the action $\arg\max_{a'} Q^*(s', a')$.
- So:

$$Q^*(s, a) = r(s, a) + \gamma \sum_{s'} P(s' \mid s, a) \max_{a'} Q^*(s', a')$$

- From the equation,

  $$Q^*(s,a) = r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) \max_{a'} Q^*(s',a')$$

  what we really want to do is update

  $$Q_{t+1}(s,a) = r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) \max_{a'} Q_t(s',a')$$

  ... but we need to know the MDP to do that.

- But if we transition from state s_t after taking action a_t to state s'_t obtaining a reward of r_t in the process, then the quantity $r_t + \gamma \max_{a'} Q_t(s'_t, a')$

  has the right expectation:

  $$E[r_t + \gamma \max_{a'} Q_t(s'_t, a')] = r(s_t, a_t) + \gamma \sum_{s'} P(s' \mid s_t, a_t) \max_{a'} Q_t(s', a')$$

- **Pure Q-learning**:

  generate a sample path

  $s_0, a_0, r_1, s_1, a_1, r_2, \ldots$

  (or multiple sample paths if the MDP is terminal)
- Maintain the quantities $Q_t(s, a)$. At every state, choose the action that maximizes the Q-value from your vector Q_t
- For example,

  $a_0 = \max_a Q_0(s_0, a)$

  $a_1 = \max_a Q_1(s_1, a)$

  ..and so forth
- Update as

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t \left[r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \right]$$

- **Theorem:** let π^* be the optimal policy. Then as long as every state-actin pair is visited infinitely often, we have that $Q_t(s,a) \to Q_{\pi^*}(s,a)$ for all state-action pairs (s,a).
- Once you know the quantities $Q_{\pi^*}(s,a)$, it is not hard to recover π^* (how would you do it?).
- How do you make sure every state, action pair is visited repeatedly?
- One solution: some fraction ϵ of the time, take a uniformly random action. The remainder of the time, choose the action according to the Q-learning algorithm.

  Decrease ϵ to zero over time.

- Let's go back to one of our basic examples: a single MDP with two actions.
  Action 1 gives a reward of two.
  Action 2 gives a reward of four.
  Discount is 1/2. Step-size is $\alpha_t = 1/t$.
- Let's say we initialize at $Q_1(1,1) = 8, Q_1(1,2) = 4$.
- Suppose we generate a sample path by choosing everything at random:
  $s_0 = 1, a_0 = 1, r_1 = 2, s_1 = 1, a_1 = 2, r_2 = 4, \ldots$
- First update:
  $$Q_2(1,1) = Q_1(1,1) + \alpha_1 \left(2 + \gamma \max\{Q_1(1,1), Q_1(1,2)\} - Q_1(1,1)\right)$$
  $$Q_2(1,1) = 8 + \frac{1}{1}\left(2 + \frac{1}{2}8 - 8\right) = 6$$
  $$Q_2(1,2) = Q_1(s = 1, a = 2) = 4.$$
- Second update:
  $$Q_3(1,2) = Q_2(1,2) + \alpha_2 \left(4 + \gamma \max\{Q_2(1,1), Q_2(1,2)\} - Q_2(1,2)\right)$$
  $$Q_3(1,2) = 4 + \frac{1}{2}\left(4 + \frac{1}{2}6 - 4\right) = 5.5$$