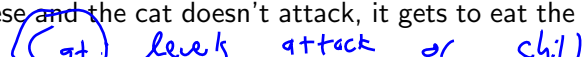
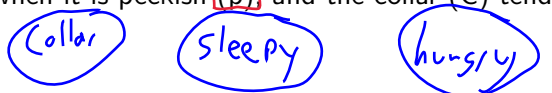
Nov 11 Reinforcement Learning

Suppose there is a mouse trying to figure out whether it should run out of its hole and eat cheese (E) or do nothing (N).

If the mouse hides, nothing happens but it stays hungry. If the mouse runs out to eat the cheese and the cat attacks, the mouse dies (which has a low utility). Otherwise, if the mouse tries to eat the cheese and the cat doesn't attack, it gets to eat the cheese (high utility).



Whether the cat will attack (A) depends on whether the cat is hungry (H) and whether the cat is sleepy (S). The mouse can observe two things, whether the cat is sleepy (S) and whether the cat has a collar (C). The cat is more often sleepy (S) when it's either full (f) or starved (v) than when it is peckish (p), and the collar (C) tends to indicate that the cat is not starved.



Draw the decision diagram according to this decision-making setup.

Announcements and To-Dos

1. Homework posted!
2. Practicum can turn it in up to Friday.

Last time we learned:

1. Beginning Value of Information on Networks

Decision-Making Networks

Definition: A *Decision-Making Network* (or influence diagram) is an extension of a Bayesian network in which nodes are classified as chance, decision, or utility.

1. 

A circular *chance node* is a probabilistic node. All nodes were chance nodes in our prior treatment of Bayesian networks.

2. 

A rectangular *decision node* is a node where an agent can and must select from a set of actions.

3. 

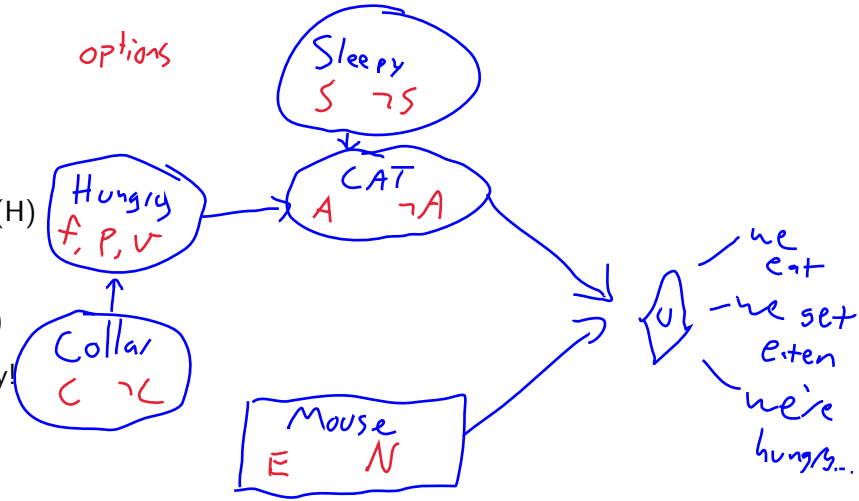
A diamond *utility node* : leaves results of ○ and □ is an outcome node. These reflect the costs and benefits of both our choices and the actions that precede them.

Opening Sol'n

We:

1. Cat Attack (A)
2. is based on (S) and (H)
3. (H) helps predict (S)
4. (C) helps predict (H)

(A) and (M) inform utility!



Suppose we have the following utilities:

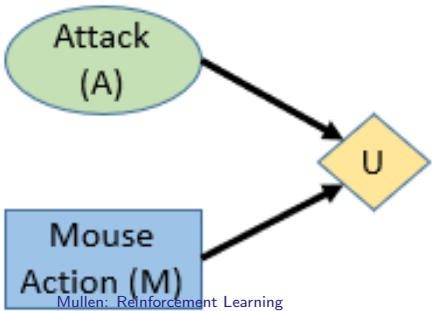
Cat Ready to Attack (A)	Mouse's action (M)	Utility $U(A, M)$
a	E	-20
$\neg a$	E	5
any	N	-2

be eaten

eat

hungry

A	P(A)
+a	0.35
-a	0.65



Cat and Mouse

Example: Suppose we are able to compute a few probabilities.

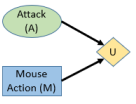
1: If $P(A = a|S = s) = 0.157$, what are the expected utilities of E and N?

$U(\pi = E \mid S = s) = .157 \cdot (-20) + (1-.157)\,5$ (attack | sleepy; given $S = s$; Prob attack; U attack; P; U)

Cat Ready to Attack (A)	Mouse's action (M)	Utility $U(A, M)$
a	E	-20
$\neg a$	E	5
any	N	-2

2: If $P(A = a|S = \neg s) = 0.71$, what are the expected utilities of E and N?

A	P(A)
+a	0.35
-a	0.65



$U(\pi = N \mid S = s) = -2$

$U(\pi = N \mid S = \neg s) = -2$ } hungry

$U(\pi = E \mid S = \neg s) = .71 \cdot (-20) + (.29)(5)$ (Prob outcome; U(outcome); prob; U)

Cat and Mouse

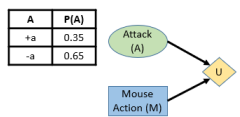
Example: Suppose we are able to compute a few probabilities.

1: If $P(A = a|S = s) = 0.157$, what are the expected utilities of E and N?
Sol: $E(M = E|S = s) = (1 - .157) \cdot 5 + .157 \cdot (-20) = 1.075$
$E(M = N|S = s) = -2$

2: If $P(A = a|S = \neg s) = 0.71$, what are the expected utilities of E and N?
Sol: $E(M = E|S = \neg s) = (1 - .71) \cdot 5 + .71 \cdot (-20) = -12.75$
$E(M = N|S = \neg s) = -2$

negative if Cat awake.

Cat Ready to Attack (A)	Mouse's action (M)	Utility $U(A, M)$
a	E	-20
$\neg a$	E	5
any	N	-2



Cat and Mouse

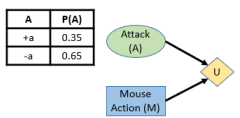
Example: Suppose we are able to compute a few probabilities.

1: If $P(A = a|S = s) = 0.157$, what are the expected utilities of E and N?
Sol: $E(M = E|S = s) = (1 - .157) \cdot 5 + .157 \cdot (-20) = 1.075$
$E(M = N|S = s) = -2$

2: If $P(A = a|S = \neg s) = 0.71$, what are the expected utilities of E and N?
Sol: $E(M = E|S = \neg s) = (1 - .71) \cdot 5 + .71 \cdot (-20) = -12.75$
$E(M = N|S = \neg s) = -2$

Result: If cat is sleeping, agent mouse goes for the cheese and get an *expected* reward of 1.075. If cat is not sleeping, agent doesn't go for the cheese and gets reward of -2. **How valuable is this?**

Cat Ready to Attack (A)	Mouse's action (M)	Utility $U(A, M)$
a	E	-20
$\neg a$	E	5
any	N	-2



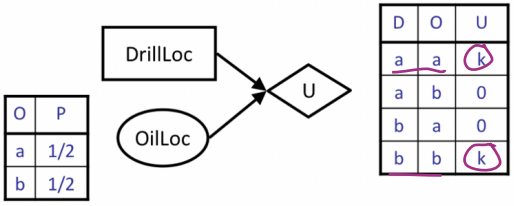
Value of Information

Definition: The *value of information* of a chance node is the increase in expected utility if that node is revealed to us.

1. We can choose different actions with more information
2. Many actions we can take might even include *gathering* information
3. We gather information in hopes that we make better decisions!

Information itself has (expected) utility!

Example: If we drill in location A, what's the utility? What's the value of the information of where the oil O is?

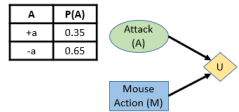


Cat and Mouse

If cat is sleeping, get an *expected* reward of 1.075. If cat is not sleeping, reward of -2.

Suppose is *costs* 1.5 units of utility to figure out if the cat is sleeping, and without any other information the cat is sleeping ($P(S = s)$) is 0.642. Should we try to get this information? A few things to compare!

Cat Ready to Attack (A)	Mouse's action (M)	Utility $U(A, M)$
a	E	-20
$\neg a$	E	5
any	N	-2



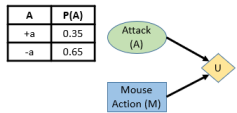
Cat and Mouse

If cat is sleeping, get an *expected* reward of 1.075. If cat is not sleeping, reward of -2.

Suppose is *costs* 1.5 units of utility to figure out if the cat is sleeping, and without any other information the cat is sleeping ($P(S = s)$) is 0.642. Should we try to get this information?
A few things to compare!

1. *If* we investigate, we get our reward of 1.075 a 0.642 proportion of the time. *If* we investigate, we get our reward of -2 the rest (0.358 proportion) of the time. Each of these *also* incur a utility of -1.5.

Cat Ready to Attack (A)	Mouse's action (M)	Utility $U(A, M)$
a	E	-20
$\neg a$	E	5
any	N	-2

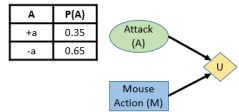


Cat and Mouse

If cat is sleeping, get an *expected* reward of 1.075. If cat is not sleeping, reward of -2.

Suppose is *costs* 1.5 units of utility to figure out if the cat is sleeping, and without any other information the cat is sleeping ($P(S = s)$) is 0.642. Should we try to get this information?
A few things to compare!

Cat Ready to Attack (A)	Mouse's action (M)	Utility $U(A, M)$
a	E	-20
$\neg a$	E	5
any	N	-2



2. *If* we don't investigate, we get our reward of -2 every time.

Cat and Mouse

If cat is sleeping, get an *expected* reward of 1.075. If cat is not sleeping, reward of -2.

Suppose is *costs* 1.5 units of utility to figure out if the cat is sleeping, and without any other information the cat is sleeping ($P(S = s)$) is 0.642. Should we try to get this information?

A few things to compare!

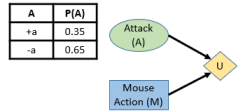
1. *If* we investigate, we get our reward of 1.075 a 0.642 proportion of the time. *If* we investigate, we get our reward of -2 the rest (0.358 proportion) of the time. Each of these *also* incur a utility of -1.5.
2. *If* we don't investigate, we get our reward of -2 every time.

So we compare -2 to $-1.5 + (0.642 \cdot 1.075 + 0.358 \cdot (-2)) = -1.52$ and choose to investigate!

$-1.52 > -2$

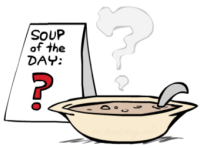
Cat Ready to Attack (A)	Mouse's action (M)	Utility $U(A, M)$
a	E	-20
$\neg a$	E	5
any	N	-2

A	P(A)
+a	0.35
-a	0.65

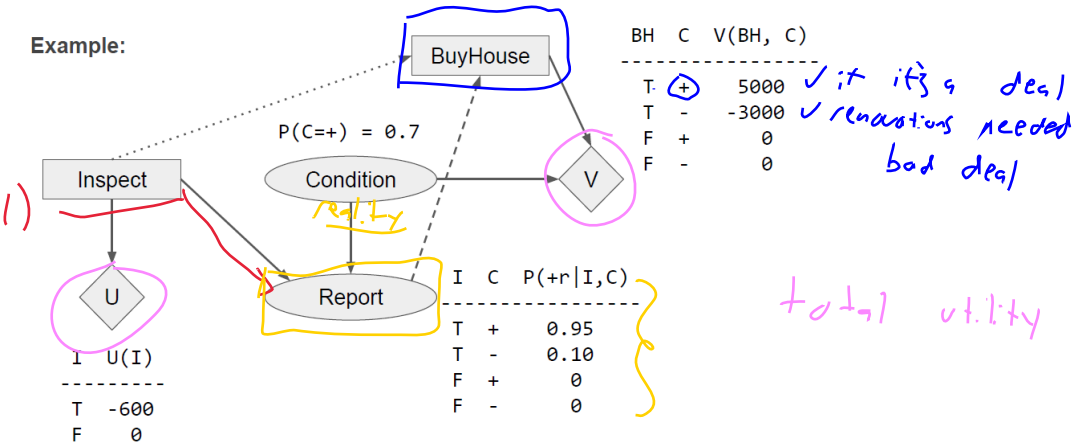


Quick Examples

- The soup of the day is either clam chowder or split pea, but you wouldn't order either one. What's the value of knowing which it is?
- There are two kinds of plastic forks at a picnic. One kind is slightly sturdier. What's the value of knowing which?
- You're playing the lottery. The prize will be $0 or $100. You can play any number between 1 and 100 (chance of winning is 1%). What is the value of knowing the winning number?

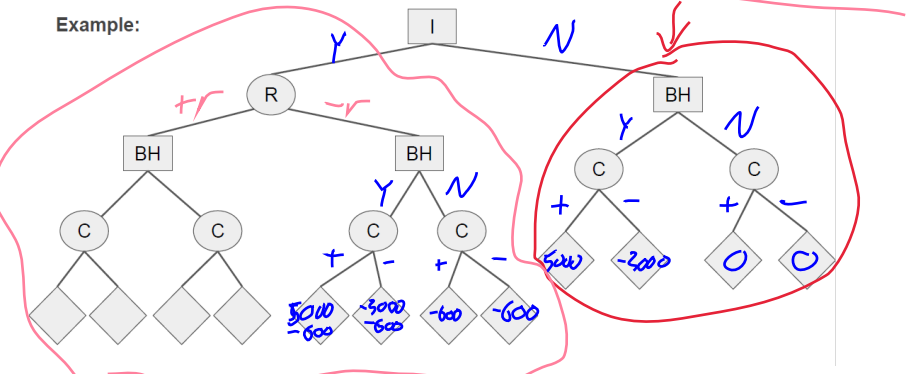


but what if we need a good fork?



Example: Suppose we need to decide *both* whether to buy a house and whether or not to get a house inspection, prior to that.

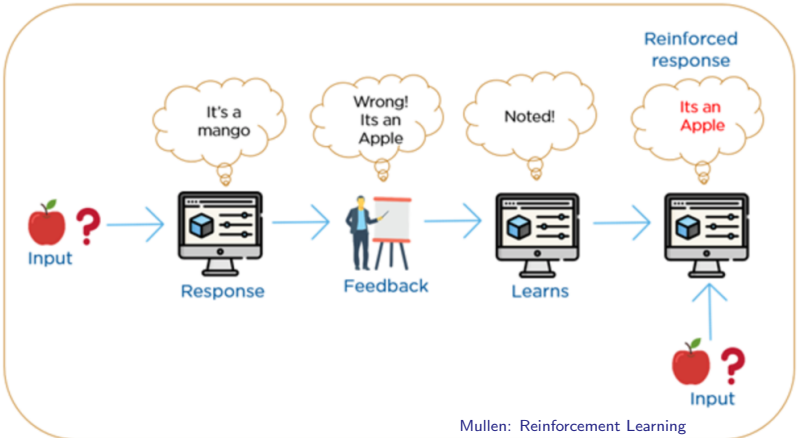
Generalizing to Sequential Decisions



Example: We might try to represent this as a tree: The I node is a decision either way, but the R node only provides information when $I = True$.
The utility nodes end up as leaves. Here each leaf is *cumulative:* the utility/cost from the I decision as well as the utility from the BH decision.

Reinforcement Learning

The Markov Decision Process was an example of *Reinforcement Learning*, where agents use an additive reward structure to give incentive to their actions. The utilities cumulatively *reinforce* a decision.



Reinforcement Learning

Definition: *Reinforcement Learning*, is the act of training agents to make a sequence of decisions to achieve some notion of utility or rewards.

It is often split into:

1. *Passive* Learning, where the agent learns the utilities of the states by taking a *fixed* and given policy.
2. *Active* Learning, where the agent must also learn to update their actions or policies along the way.

Many reinforcement learning problems are couched as MDPs, where actions may not only include moving from state to state, but also things like "investigating" to get more information, as our mouse might have.

Unfortunately, as we get more and more chance nodes and possible states to explore, the exact methods suggested in our prior processes can become prohibitive.

Passive Learning

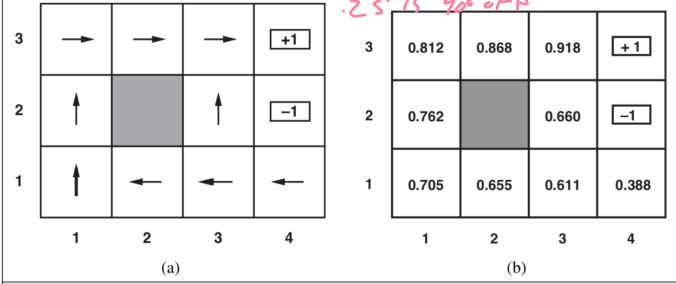


Figure 21.1 (a) A policy π for the 4×3 world; this policy happens to be optimal with rewards of $R(s) = -0.04$ in the nonterminal states and no discounting. (b) The utilities of the states in the 4×3 world, given policy π.

Example: The *Policy Evaluation* step of the MDP policy iteration algorithm was an example of passive learning. In it, we:

1. Learn how good the policy π is by computing and estimating U^{π}
2. This was a problem with a full observable environment with a state-based representation.

In more general problems, we *don't know* the transition probability $P(s'|a, s)$, but we *do* have access to the sets of actions a and some sensor that tells us *which* states are nearby/possible.

Passive Learning

More generally, passive learning can be tackled by a variety of approaches:

1. Direct Estimation:
   Try many policies, see which is doing best.
2. Adaptive Dynamic Programming:
   Try to learn the transition *model* as we go, and then make better decisions.
3. Temporal-difference Learning:
   Try to make sure that the utilities of *neighboring* states follow the Bellman equations.

Recall the Bellman equations are the system that state that under optimal policies, the utility of each state is given by the utility of the *best action* a from that state:

$$U(s) = R(s) + \sum_{s'} P(s'|a, s)U(s')$$

Direct Estimation

→ random / exhaustive

Direct Estimation: Try many policies, see which is doing best.

1. Sample many action sequences beginning from state s. We call these *training episodes* or *trials*.
2. Move according to a policy $\pi(s)$, where S_t might be a random variable.

↓ transition to or start.

3. This is the same as sampling from a Bayes Net, where our transition probabilities $P(s'|a, s)$ behave the same as a conditional probability table.
4. We add up all our rewards along the way. In the end, the value of the policy is estimated by the utility achieved!

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right]$$

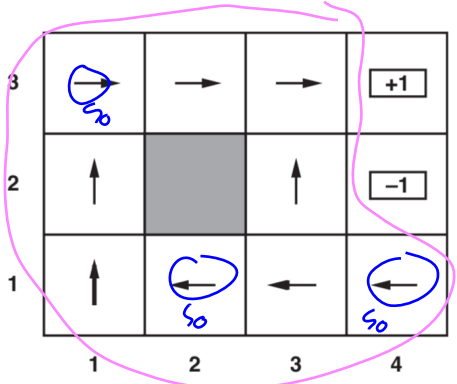
discount

accumulated rewards

future states S_t

5. This can be done **model-free**. We can often *simulate* results of our actions without writing down or computing explicit probabilities $P(s'|a, s)$.

Direct Estimation



run to completion / a long time.

one π

Example: We could estimate the utility of this policy by:

1. Performing trials using the policy shown.
2. Averaging the observed rewards to get the utilities for each state given that policy.

Direct Estimation

Example: Direct Estimation. Recall our notebook from last Friday. In it, there is a betting game that has these rules:

1. You start with some amount of money.
2. You can wager any amount in increments of 1, with the maximum amount permitted = however much money you currently have.
3. You flip a fair coin. If heads, you win your wager and your money increases by that amount. If tails, you lose your wager and your money decreases by that amount.
4. You start playing this game with 5. You will quit if you have at least 10 at any point, or if you lose all of your money. Your reward is the difference between your ending money and $\$5$.

Direct Estimation

Example: Direct Estimation. In this game, suppose that we wish to evaluate a policy by *direct estimation.*

1. We choose a policy. Say we start with $\pi(s) = 1 \forall s$, or we always wager $\$1$.
2. We start at some state. Maybe $s_0 = 2$. Then we play the game a few times, and observe a chain of results for one *training episode*:

$$2 \to 3 \to 2 \to 1 \to 0$$

We then tally our utilities. If we had $R(s) = -.01$ for non terminal results, we've now seen no less than *4* samples! For example, we in this trial were at state "1," got our -0.01 reward, then lost and got our -5 terminal state.

From right to left:

$$U(1) = -5.01, \quad U(2) = -5.02, \quad U(3) = -5.03, \quad U(2) = -5.04$$

Direct Estimation

After performing *a lot* of simulations from a variety of initial states, we can simply tabulate the rewards observed *given* the various policies undertaken. Upside:

1. Easy to write.
2. Intuitive.

Downside:

1. Really really expensive to explore the way that different policies at every state interact.
2. Doesn't formally take advantage of the fact that the transition probabilities $P(s'|a,\, s)$ are fixed: we require a huge sample because we're not exploiting this dependence!

Adaptive Dynamic Programming Estimation

Definition: *Adaptive Dynamic Programming* for reinforcement learning samples from the state space and updates estimates of $P(s'|s,\, a)$ as it runs.

This is *model-based*, because we're now possibly *assuming* things like the underlying Markov Property, that $P(s'|a,\, s)$ depends only on the current state s and not some notion of continuity or momentum based on earlier states.

One twist: we no longer need a fully observable state space: instead we can let our agent *explore* the space!

ADP RL

1. Initialize:
   a) $Na(s, a)$: the number of times we took action a from state s.
   b) $Nt(s', s, a)$: the number of times *when* we took action a from state s we ended up at state s'.
   c) Set $P(s'|a, s) = Nt(s', s, a)/Na(s, a)$ to estimate the transition model: the probability that action a in state s yields state s'.
   d) $U(s)$: utility for each state, initialized somehow.

ADP RL

2. Train. Run *many* training episodes, where for each:
   a) Start as some random state s_0.
   b) Sample a next state s_1. Store the *percept* $(s_1, R(s_1))$.
   c) If s_1 is a new state, add its key to the list of states and the counting dictionaries. Store $R(s_1))$ in the rewards (vector/dictionary).
   d) Update the counters Na and Nt.
   e) Update the the estimate CPT $P(s'|s, a)$.
   f) Update $U(s)$ by *policy evaluation*. For example, use our current estimates of $P(s'|a,\ s)$ and $U(s')$ to choose a best policy for s and find its utility.

ADP RL Example

Suppose again we're playing our Roulette-like game, and observe a chain of results for one *training episode*:

$$2 \to 3 \to 2 \to 1 \to 0$$

Now we tally up:

1. Counts of *actions*: $Na[(2, a = 1)] = 2$, $Na[(3, a = 1)] = 1$, $Na[(1, a = 1)] = 1$
2. Counts of *results*: $Nt[(3, 2, a = 1)] = 1$, $Na[(2, 3, a = 1)] = 1$, $Nt[(1, 2, a = 1)] = 1$, $Na[(0, 1, a = 1)] = 1$
3. Find some probabilities! Now we have:

$$P[(3, 2, a = 1)] = \frac{Nt[(3, 2, a = 1)]}{Na[(2, a = 1)]} = \frac{1}{2}$$

4. We could now update $U(2)$ as a function of $U(3)$ and $U(1)$, but right now our only policy is "1." We need more training episodes!

Using a Model

ADP is not model-free because it relies heavily on the Markov property. This tends to lead it to be much more accurate than direct estimation *when that assumption is correct*.

Our final method today is **Temporal-difference** learning, which is:

1. Model-free, because it doesn't estimate P.
2. Idea: instead of working on updating probabilities, work directly with the *utilities* and iterate with a modification on Bellman equations:

$$U_{i+1}(s) = R(s) + \gamma \sum_{s'} \underbrace{P(s'|a,\, s)}_{don't\, know\, this}\; U(s')$$

Temporal-Difference

$$U_{i+1}(s) = R(s) + \gamma \sum_{s'} \underbrace{P(s'|a,\, s)}_{don't\, know\, this}\; U(s')$$

Ignoring P for now, suppose that we always go where we want. P is 0 or 1, and

$$U(s) = R(s) + \gamma U(s')$$

Now define the *difference* between these two sides:

$$\Delta U = R(s) + \gamma U(s') - U(s)$$

Why this?

1. If we could always go where we want, should always be zero for the *actual* utility values.
2. What happens to this value of our *current* estimate of $U^{\pi}(s)$ is too high?
3. What happens to this value of our *current* estimate of $U^{\pi}(s)$ is too low?

Temporal-Difference

$$U_{i+1}(s) = R(s) + \gamma \sum_{s'} \underbrace{P(s'|a,\, s)}_{don't\, know\, this}\; U(s')$$

Ignoring P for now, suppose that we always go where we want. P is 0 or 1, and

$$U(s) = R(s) + \gamma U(s')$$

Now define the *difference* between these two sides:

$$\Delta U = R(s) + \gamma U(s') - U(s)$$

Why this?

1. If we could always go where we want, should always be zero for the *actual* utility values.
2. What happens to this value of our *current* estimate of $U^\pi(s)$ is too high? $\boxed{\Delta U < 0}$
3. What happens to this value of our *current* estimate of $U^\pi(s)$ is too low? $\boxed{\Delta U > 0}$

Temporal-Difference

$$\Delta U = R(s) + \gamma U(s') - U(s)$$

estimates how far off our value of $U(s)$ is from its true value.

Note that this estimate will be off by more if the probabilities of moving where we *don't* intend to is high!

Algorithm:
Update $U(s)$ by adding $\alpha \Delta U$:

$$U^{\pi}_{i+1}(s) = U^{\pi}_{i}(s) + \alpha \left[R(s) + \gamma U^{\pi}_{i}(s') - U^{\pi}_{i}(s) \right]$$

where α: the *learning rate*.

Temporal-Difference Learning Rates

α: the *learning rate*

1. The idea behind the learning rate is twofold: it helps include the missing information of P *not* being 0 or 1, and it adjusts to how much information we have.
2. Typically, we make α *decrease* with the number of trials. We might write $\alpha = \alpha(N(s))$ to explicitly state that alpha at state s depends on the number of visits to s.
3. This way, we don't make large changes to a state's utility if we've visited it many times!
4. Formal requirements for convergence of U_i to true utilities under policy π:

$$\sum_{n=0}^{\infty} \alpha(n) = \infty \qquad \textbf{AND} \qquad \sum_{n=0}^{\infty} (\alpha(n))^2 < \infty$$

Temporal-Difference Learning Rates

α: the *learning rate*

1. The idea behind the learning rate is twofold: it helps include the missing information of P *not* being 0 or 1, and it adjusts to how much information we have.
2. Typically, we make α *decrease* with the number of trials. We might write $\alpha = \alpha(N(s))$ to explicitly state that alpha at state s depends on the number of visits to s.
3. This way, we don't make large changes to a state's utility if we've visited it many times!
4. Formal requirements for convergence of U_i to true utilities under policy π:

$$\sum_{n=0}^{\infty} \alpha(n) = \infty \qquad \textbf{AND} \qquad \sum_{n=0}^{\infty} (\alpha(n))^2 < \infty$$

Examples: Things like $\alpha(n) = \frac{1}{n}$

Moving Forward

- Coming up:
  1. Reinforcement learning continues!