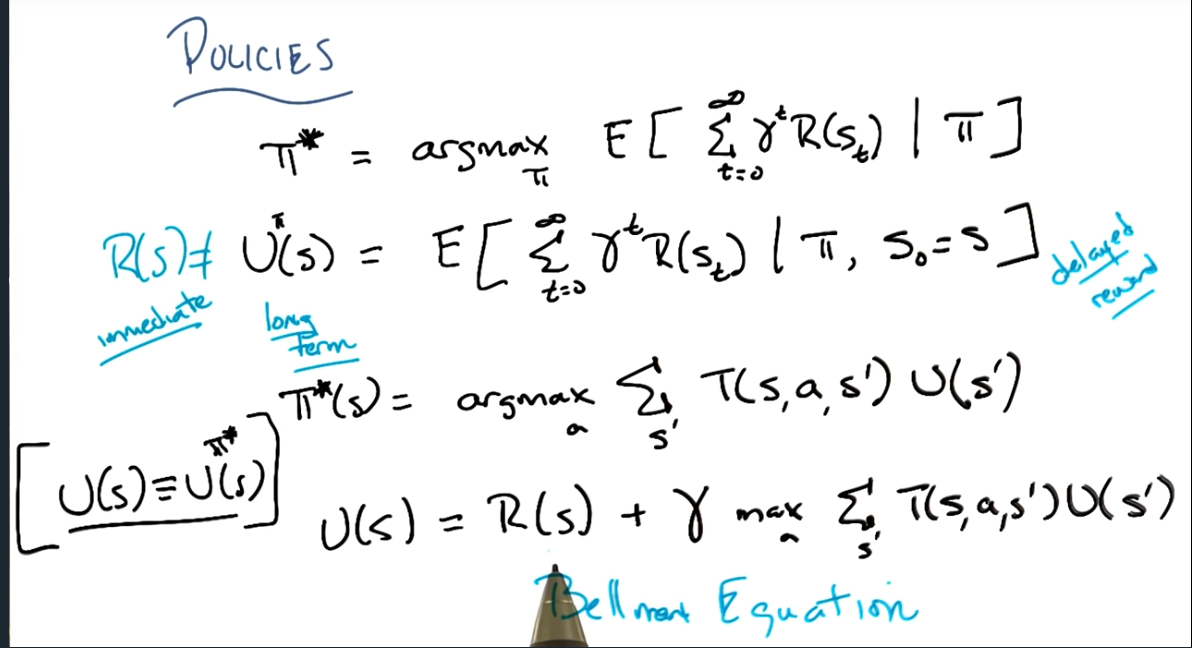
Markov Decision Processes

Decision making and reinforcement learning

* States – represents something and we have a way of knowing which state we are in
* Model – T(s, a s’) ~ Pr(s’ | s,a) – transition model – physics of the world – rules of the game. Produces the probability that you will transition to state s-prime from original state given an action.
* Actions – A(s), A - things you can do in a particular state
* Reward – R(s), R(s,a), R(s,a,s’) – reward of a value entering a state
  + Delayed rewards
  + Minor actions matter
* Policy - – solution to a markov decision process. \* is the optimal policy that maximizes reward
* Markovian property – only the present matters, previous states do not matter. Can make current state into something that “remembers” though.
* MDP doesn’t give a plan. It gives an action given a state. You can “infer” a plan from a policy.
* Delayed reward – can take a fork in the road early on that you would only find out was wrong at the end. Also known as a temporal credit assignment problem
* can use geometric series to get away from infinite reward. Can use gamma^inf where 0 <= gamma <= 1. Allows us to go an “infinite distance in finite time”.
  + Discounted → geometric
  + infinite → finite



**Value Iteration**

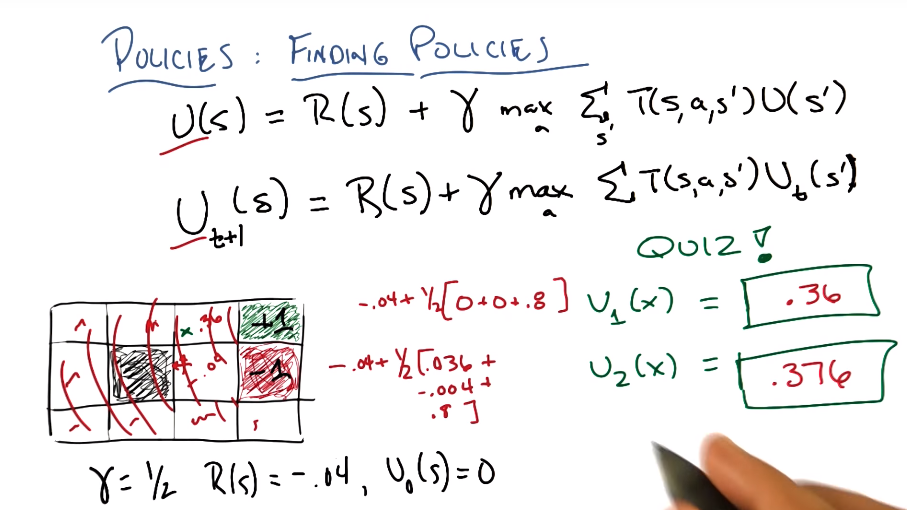
Find the true value of states by iterating.

Doesn’t give you the answer but gives you something that’s closer and closer to the answer

Once you have the true utilities, we know how to define the optimal policy in terms of the utilities. Only need to look at the state you are in, then look at all the states you might get to, figure out the expectation that you’re going to get given an action, pick whichever is the max and you’re done.

Solving for the utilities, or the true value of a state, is effectively the same thing as solving for the optimal policy.

Value iteration works because eventually value propagates out from its neighbors.



What’s propagating out are the true utilities of each state.

A policy is a function from states → actions, NOT a mapping from states → utilites (that’s what U is). If we have U, we can figure out pi, but U is much more information than we need to figure out pi.

If we have a U that is not the correct utility, but has the ordering of the actions correct, then we are actually doing really well, it doesn’t matter if we have the right utilities.

We don’t actually care about the actual utilities, even though by having the correct utilities we have the correct policy. What we really care about is the policy.

Pi is more of a classifier, mapping inputs to discrete classes, and U’s are more like regression where it’s mapping states to continuous values.

You only want a utility that’s good enough to get you to your pi. Gives us a hint that might go faster in practice…

What might be faster and just as good as doing value iterations?

**Policy Iteration**

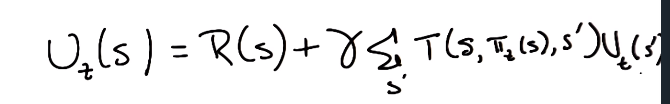
1.) Start with an initial policy, which is a guess

2.) Evaluate how good the policy is (U\_t)

- evaluate it by calculating the Utility

3.) Improve the policy by updating the policy t+1 to be the policy that takes the actions that maximize the expected utility

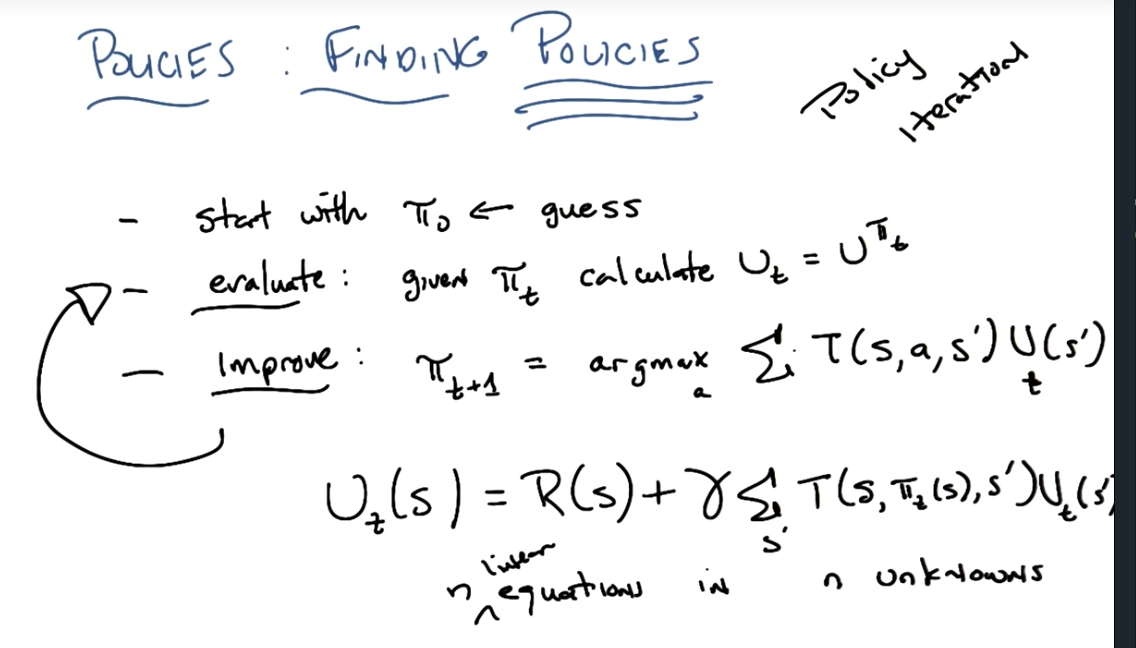
Calculating U\_t → use Bellman’s equation”

Only difference from before is to not use the “max” which means we have linear equations which lets us compute the answer in a reasonable amount of time by doing matrix inversions and regressions and other “hand-wavy” things.

Pro: Don’t have to do as many iterations.

Just keep improving and evaluating until your policy doesn’t change.

PI makes larger jumps because it is making jumps in policy space instead of value space.



Recap:

- Utilites are long term, rewards are instantaneous

- value functions (utilities)

- discounting (infinite → finite)

- Solve Bellman equation using value iteration and policy iteration

- MDP != Reinforcement Learning

- we know the rewards, states, actions, and transitions. We have discounts and have been solving MDPs

- Reinforcement learning = you don’t necessarily know the rewards, transitions, or even the states and the actions