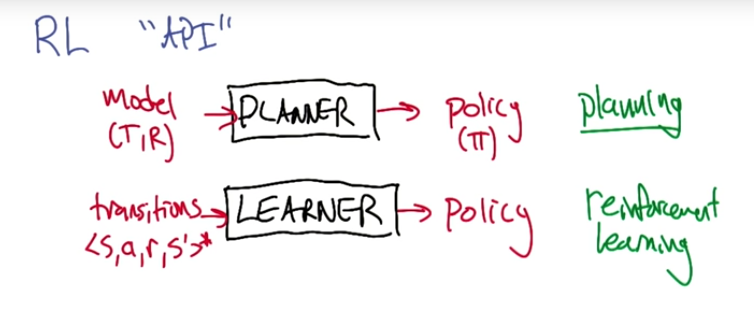
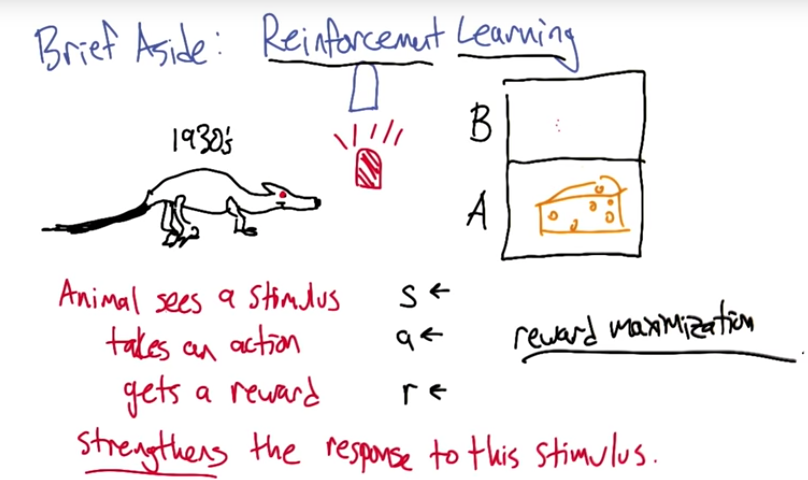
Reinforcement Learning

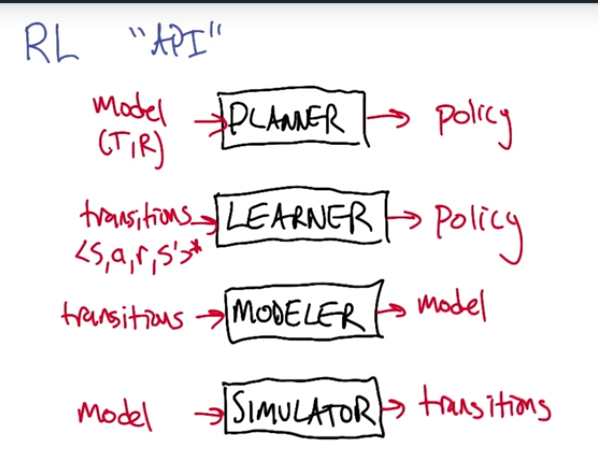


Reinforcement learning = reward maximization

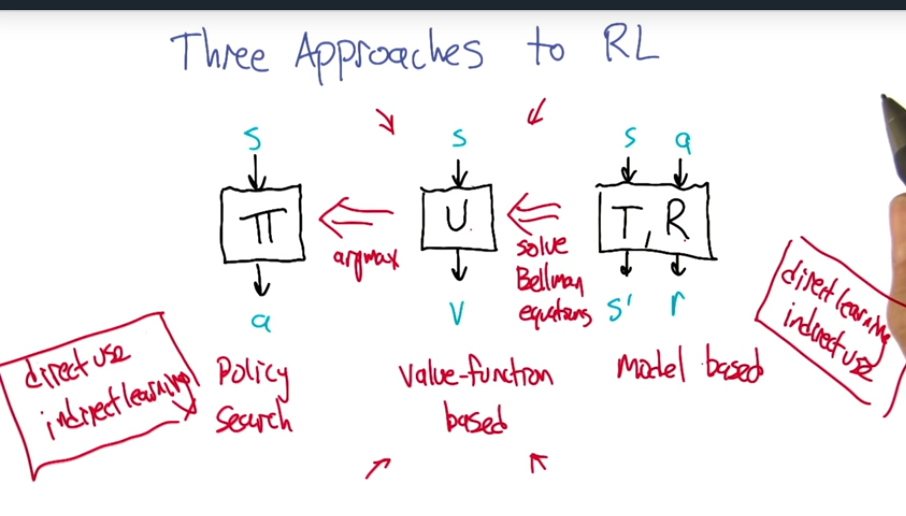


Our case really has nothing to do with strengthening

Both VI and PI take in models and produce policies (i.e. they are planners)



* 3 approaches to RL
  + Policy search algorithm – maps states to actions
    - Pro: working on the data that you’ll be using
    - Con: function is indirect, data doesn’t tell you what action to choose
  + Value function based - Maps state to values
    - Pro: if we’re acting in the world, we can observe the values and the learning is not so indirect
    - Con: hard to use “U” directly, but can use *argmax*
  + Model based learning
    - Direct learning, can solve as a supervised learning problem
    - Indirect use



Value-functioned based functions is the “goldilocks” method where we don’t have fully direct use or learning but it is close for each.

**Q-function** – value for arriving in *s*, leaving via *a*, proceeding optimally thereafter. We are tying the algorithms hands by forcing it to leave by action *a.* This allows us to compare the values of different actions without having to stare at the model to do it.

U(s) = max reward of action of all Q(s,a) (returns scalar)

Pi(s) = argmax (max reward of action of all Q(s,a) (returns action)

How do evaluate Q-function? Q-learning

**Q-learning** – estimating Q-function from transitions

* Don’t have R(s) and T, so must figure out a different way to come up with optimal solution

If we start Q\_hat anywhere, then we will converge on the solution of the Bellman equation, which is the optimal solution to the MDP

* This is only true if we visit *s,a* infinitely often…

Q-learning is a family of algorithms

* How to initialize Q\_hat?
* How do we decay our alpha\_t?
* How to choose our actions?
  + Always choose the same a\_0? – terrible choice and violates Q-learning convergence of visiting each state infinitely often. We aren’t learning anything.
  + Choose randomly? – seems kind of good since we’ll visit all states and all actions and can learn Q, but we are not taking advantage of *learning* Q and choosing actions based off what we’ve learned. What’s the point of learning a Q function if we are never using it?
  + Use Q\_hat to choose actions? = seems like a good idea. We will use what we’ve learned, but what happens if we have a bad initial value which sends us down a bad path?
    - We can get into a local min by way of “greediness”, which is about equivalent of always choosing the same a\_0
    - We can use random restarts? – problem is this will be slow
    - Can do something similar to simulated annealing? – is a mixture of choosing randomly and using what we know that seems to be the best. This means that we’ll take a random action sometimes. This allows us to explore the whole space which increases the chances we find the true Q.

**Epsilon-greedy exploration** – If GLIE (greedy in the limit with infinite exploration – that we start off more random and then get more and more greedy)

* Q\_hat -> Q and pi\_hat -> pi\_\*
  + We learn stuff and we use it too
  + **Exploration – exploitation dilemma**
    - Exploitation – using what you know
    - Exploration – getting the data that you need so you can learn
    - This is a tradeoff because there is only one agent working in the world.
    - Model learning + planning interact with each other

**Recap**

* You can learn how to solve an MDP
  + Don’t know T&R, but just have the ability to interact with the environment and receive transitions <s,a,r,s’>
* Q-learning: converge, family of algorithms
* Exploration-exploitation: learn and use
  + Initialization of Q function is another form of exploration
  + Optimism in the face of uncertainty!
* Approaches to RL
* Connection to planning