Reinforcement Learning

What do you do when you don’t know how the world works?

One option:
• estimate R (reward function) and P (transition function) from data
• solve for optimal policy given estimated R and P

Another option:
• estimate a value function directly

Bandit Problems

• switch on a loser
• always choose the apparent best
• choose the apparent best 90% of the time; choose randomly the other 10%
• consider both the amount of experience you’ve had with each arm and the payoff
• etc...

Arms are like actions in a single-state MDP

Imagine what this problem is like in a multi-state MDP!

Q Function

A different way to write down the recursive value function equation.

\[ Q'(s,a) \] is the expected discounted future reward for starting in state \( s \), taking action \( a \), and continuing optimally thereafter.

\[
Q'(s,a) = R(s) + \sum_{s'} \Pr(s'|s,a) \max_{a'} Q'(s',a') \\
J'(s) = \arg\max_a Q'(s,a)
\]

Q Learning

A piece of experience in the world is \( (s,a,r,s') \)

• Initialize \( Q(s,a) \) arbitrarily
• After each experience, update \( Q \):

\[
Q(s,a) \leftarrow Q(s,a) + \alpha (r + \gamma \max_a Q(s',a) - Q(s,a)) \\
qu(r,s') = r + \gamma \max_a Q(s',a)
\]

Guaranteed to converge to optimal \( Q \) if the world is really an MDP
Lots of issues

- large or continuous state spaces
- slow convergence

Mostly used in large simulations
- TD Gammon
- Elevator scheduling