- Proj 1: Q7 (5 pts): Explanation and new code
- Proj 2: Partnered?
  - Mean: 18.9/20
  - Median: 20
  - EC: +1 to those who did it
  - Something incorrect? Let Matt know
Active Learning

- Full reinforcement learning
  - You don’t know the transitions $T(s,a,s’)$
  - You don’t know the rewards $R(s,a,s’)$
  - You can choose any actions you like
  - Goal: learn the optimal policy
  - … what value iteration did!

- In this case:
  - Learner makes choices!
  - Fundamental tradeoff: exploration vs. exploitation
  - This is NOT offline planning! You actually take actions in the world and find out what happens…
Q-Learning

- Q-Learning: sample-based Q-value iteration
- Learn $Q^*(s,a)$ values
  - Receive a sample $(s,a,s',r)$
  - Consider your old estimate: $Q(s,a)$
  - Consider your new sample estimate:
    $$Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q^*(s',a')]$$
    $$sample = R(s,a,s') + \gamma \max_{a'} Q(s',a')$$
  - Incorporate the new estimate into a running average:
    $$Q(s,a) \leftarrow (1 - \alpha)Q(s,a) + (\alpha) [sample]$$
Q-Learning Properties

- Amazing result: Q-learning converges to optimal policy
  - If you explore enough
  - If you make the learning rate small enough
  - … but not decrease it too quickly!
  - Basically doesn’t matter how you select actions (!)

- Neat property: off-policy learning
  - learn optimal policy without following it (some caveats)
Exploration / Exploitation

- Several schemes for forcing exploration
  - Simplest: random actions ($\varepsilon$ greedy)
    - Every time step, flip a coin
    - With probability $\varepsilon$, act randomly
    - With probability $1-\varepsilon$, act according to current policy

- Problems with random actions?
  - You do explore the space, but keep thrashing around once learning is done
  - One solution: lower $\varepsilon$ over time
  - Another solution: exploration functions
Exploration Functions

- **When to explore**
  - Random actions: explore a fixed amount
  - Better idea: explore areas whose badness is not (yet) established

- **Exploration function**
  - Takes a value estimate and a count, and returns an optimistic utility, e.g. \( f(u, n) = u + k/n \) (exact form not important)

\[
Q_{i+1}(s, a) \leftarrow \alpha R(s, a, s') + \gamma \max_{a'} Q_i(s', a') \\
Q_{i+1}(s, a) \leftarrow \alpha R(s, a, s') + \gamma \max_{a'} f(Q_i(s', a'), N(s', a'))
\]
Q-Learning

- Q-learning produces tables of q-values:

![Q-Values after 1000 episodes]