CS 188: Artificial Intelligence
Spring 2010

Lecture 12: Reinforcement Learning II
2/25/2010

Pieter Abbeel – UC Berkeley

Many slides over the course adapted from either Dan Klein, Stuart Russell or Andrew Moore

Announcements

- W3 Utilities: due tonight
- P3 Reinforcement Learning (RL):
  - Out tonight, due Thursday next week
  - You will get to apply RL to:
    - Gridworld agent
    - Crawler
    - Pac-man

2

Reinforcement Learning

- Still assume a Markov decision process (MDP):
  - A set of states $s \in S$
  - A set of actions (per state) A
  - A model T(s,a,s')
  - A reward function R(s,a,s')
- Still looking for a policy $\pi(s)$ $S \to A$
- New twist: don't know T or R
  - I.e. don't know which states are good or what the actions do
  - Must actually try actions and states out to learn

3

The Story So Far: MDPs and RL

Things we know how to do:

- If we know the MDP
  - Compute V*, Q*, π* exactly
  - Evaluate a fixed policy π
- If we don't know the MDP
  - We can estimate the MDP then solve
  - We can estimate V for a fixed policy π
  - We can estimate Q*(s,a) for the optimal policy while executing an exploration policy

$\arg\max_a Q^*(s,a)$

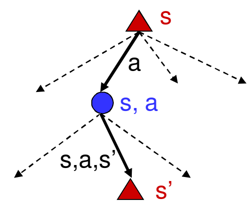
Techniques:

- Model-based DPs
  - Value and policy Iteration
  - Policy evaluation
- Model-based RL
- Model-free RL:
  - Value learning
  - Q-learning

4

Problems with TD Value Learning

- TD value leaning is a model-free way to do policy evaluation
- However, if we want to turn values into a (new) policy, we're sunk:



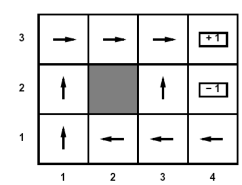
$$\pi(s) = \arg\max_a Q^*(s,a)$$

$$Q^*(s,a) = \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V^*(s')\right]$$

- Idea: learn Q-values directly
- Makes action selection model-free too!

6

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…

7

has access to T, R

Detour: Q-Value Iteration

- Value iteration: find successive approx optimal values
  - Start with $V_0(s) = 0$, which we know is right (why?)
  - Given V_i, calculate the values for all states for depth i+1:

$$V_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s') \right]$$

$Q_{i+1}(s,a)$

$V_{i+1}(s) = \max_a Q_{i+1}(s,a)$

$\max_a Q_i(s',a)$ R

- But Q-values are more useful!
  - Start with $Q_0(s,a) = 0$, which we know is right (why?)
  - Given Q_i, calculate the q-values for all q-states for depth i+1:

$$Q_{i+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q_i(s',a') \right]$$

observe (s,a,s') ⇒ empirical update

sample estimate for $Q(s,a) = R(s,a,s') + \gamma \max_{a'} Q(s',a')$

8

$E_{x \sim P(x)}[f(x)] \doteq \sum_x P(x) f(x)$

sample estimate: $x_i \sim P(x)$

$E[f(x)] = \frac{1}{k} \sum_{i=1}^{k} f(x_i)$

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:

$$Q^*(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q^*(s',a') \right]$$

$$sample = R(s,a,s') + \gamma \max_{a'} Q(s',a')$$

  - Incorporate the new estimate into a running average:

$$Q(s,a) \leftarrow (1-\alpha) Q(s,a) + (\alpha) [sample]$$

$\alpha = 1$

10

Q-Learning Properties

- Amazing result: Q-learning converges to optimal policy
  - If you explore enough ~ all (s,a) visited ∞ often
  - If you make the learning rate small enough $\frac{1}{t}$
  - … but not decrease it too quickly!
  - Basically doesn't matter how you select actions (!)

$\sum_{t=1}^{\infty} \frac{1}{t} = \infty$

$\sum_{t=1}^{\infty} \frac{1}{t^2} < \infty$

- Neat property: off-policy learning
  - learns optimal Q-values, not the values of the policy you are following

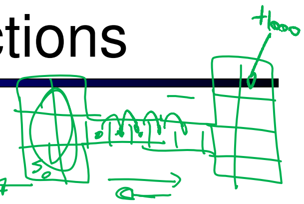
12

Exploration / Exploitation

- Several schemes for forcing exploration
  - Simplest: random actions (ε greedy)
    - Every time step, flip a coin
    - With probability ε, act randomly
    - With probability 1-ε, act according to current policy e.g. $\to \arg\max_a Q(s,a)$
- Regret: expected gap between rewards during learning and rewards from optimal action
  - Q-learning with random actions will converge to optimal values, but possibly very slowly, and will get low rewards on the way
  - Results will be optimal but regret will be large
  - How to make regret small?

14

Exploration Functions



- When to explore
  - Random actions: explore a fixed amount
  - Better ideas: explore areas whose badness is not (yet) established, explore less over time
- One way: 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)

α

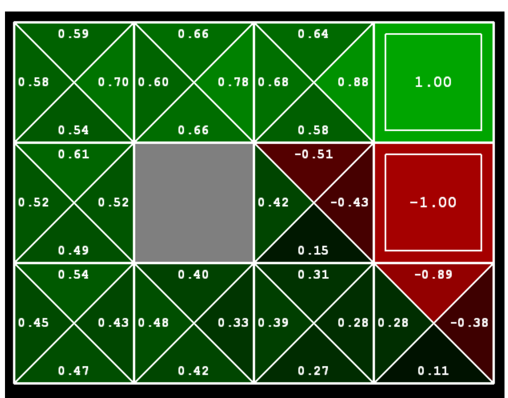
$$Q_{i+1}(s,a) \leftarrow_\alpha R(s,a,s') + \gamma \max_{a'} Q_i(s',a')$$

$$Q_{i+1}(s,a) \leftarrow_\alpha R(s,a,s') + \gamma \max_{a'} f(Q_i(s',a'), N(s',a'))$$

16

Q-Learning

- Q-learning produces tables of q-values:



Q-VALUES AFTER 1000 EPISODES

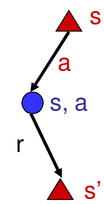
18

Recap Q-Learning

- Model-free (temporal difference) learning
  - Experience world through episodes

$$(s, a, r, s', a', r', s'', a'', r'', s'''' \ldots)$$

  - Update estimates each transition (s, a, r, s')
  - Over time, updates will mimic Bellman updates



Q-Value Iteration (model-based, requires known MDP)

$$Q_{i+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q_i(s',a')\right]$$

Q-Learning (model-free, requires only experienced transitions)

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + (\alpha)\left[r + \gamma \max_{a'} Q(s',a')\right]$$

19

Q-Learning

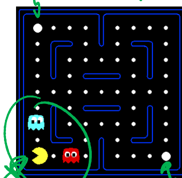
- In realistic situations, we cannot possibly learn about every single state!
  - Too many states to visit them all in training
  - Too many states to hold the q-tables in memory
- Instead, we want to generalize:
  - Learn about some small number of training states from experience
  - Generalize that experience to new, similar states
  - This is a fundamental idea in machine learning, and we'll see it over and over again

20

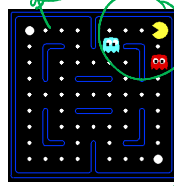
Example: Pacman

states = $2^{81} \times 81 \times 81 \times 81$ ...

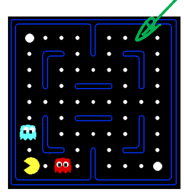
- Let's say we discover through experience that this state is bad:



- In naïve q learning, we know nothing about this state or its q states:



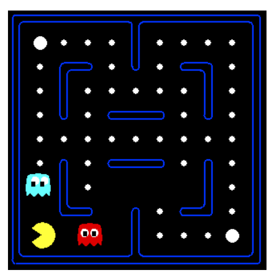
- Or even this one!



22

Feature-Based Representations

- Solution: describe a state using a vector of features (properties)
  - Features are functions from states to real numbers (often 0/1) that capture important properties of the state
  - Example features:
    - Distance to closest ghost
    - Distance to closest dot
    - Number of ghosts
    - 1 / (dist to dot)^2
    - Is Pacman in a tunnel? (0/1)
    - …… etc.
    - Is it the exact state on this slide?
  - Can also describe a q-state (s, a) with features (e.g. action moves closer to food)



23

Linear Feature Functions

- Using a feature representation, we can write a Q function (or value function) for any state using a few weights:

ASSUMPTION which is usually unable to represent the true value function

$$V(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

- Advantage: our experience is summed up in a few powerful numbers
- Disadvantage: states may share features but actually be very different in value!

24

Function Approximation

ASSUMPTION

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

Known: $f_1, \ldots, f_n$; Unknown: $w_1, w_2, \ldots, w_n$

- Q-learning with linear q-functions:

$$transition \; = (s, a, r, s')$$

$$\text{difference} = \left[r + \gamma \max_{a'} Q(s', a') \right] - Q(s,a)$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha \,[\text{difference}]$$ Exact Q's

$$w_i \leftarrow w_i + \alpha \,[\text{difference}]\, f_i(s,a)$$ Approximate Q's

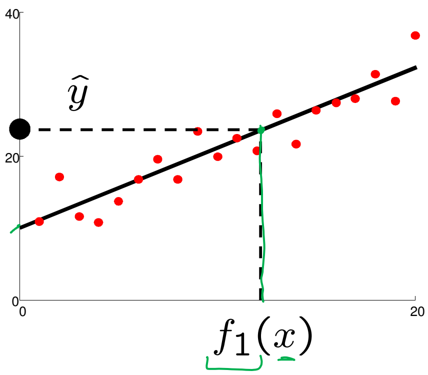
$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a)$

$f_p(s,a) = 0$

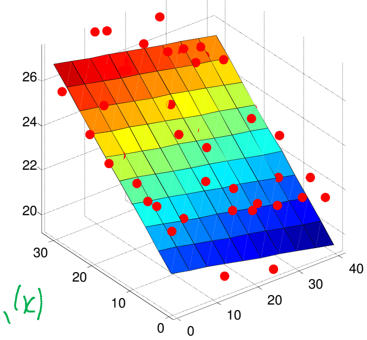
- Intuitive interpretation:
  - Adjust weights of active features
  - E.g. if something unexpectedly bad happens, disprefer all states with that state's features
- Formal justification: online least squares

25

Linear Regression



$\rightarrow f_1(x)$



Prediction

$\hat{y} = w_0 + w_1 f_1(x)$

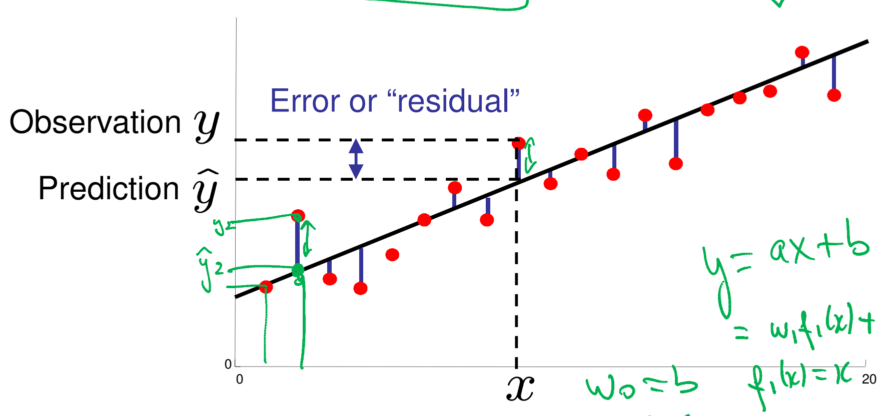
Prediction

$\hat{y}_i = w_0 + w_1 f_1(x) + w_2 f_2(x)$

27

Ordinary Least Squares (OLS)

$$\text{total error} = \sum_i (y_i - \hat{y}_i)^2 = \sum_i \left(y_i - \sum_k w_k f_k(x_i) \right)^2$$



$y = ax + b$

$= w_1 f_1(x) + w_0$

$w_0 = b$

$w_1 = a$

$f_1(x) = x$

28

Minimizing Error

Imagine we had only one point x with features f(x):

$$\text{error}(w) = \frac{1}{2}\left(y - \sum_k w_k f_k(x)\right)^2$$

function

$$\frac{\partial\ \text{error}(w)}{\partial w_m} = -\left(y - \sum_k w_k f_k(x)\right) f_m(x)$$

one entry of the gradient

$$\boxed{w_m \leftarrow w_m + \alpha \underbrace{\left(y - \sum_k w_k f_k(x)\right)}_{\text{error}} f_m(x)}$$

Approximate q update explained:

"target" — "prediction"

$$w_m \leftarrow w_m + \alpha \left[r + \gamma \max_a Q(s', a') - Q(s, a)\right] f_m(s, a)$$

29

Overfitting