CS 343H: Honors AI

Lecture 14:

Reinforcement Learning, part 3

3/3/2014

Kristen Grauman

UT Austin

Slides courtesy of Dan Klein, UC Berkeley

Announcements

- Midterm this Thursday in class
  - Can bring one sheet (two sided) of notes
  - Covers everything so far *except* for reinforcement learning (up through and including lecture 11 on MDPs)

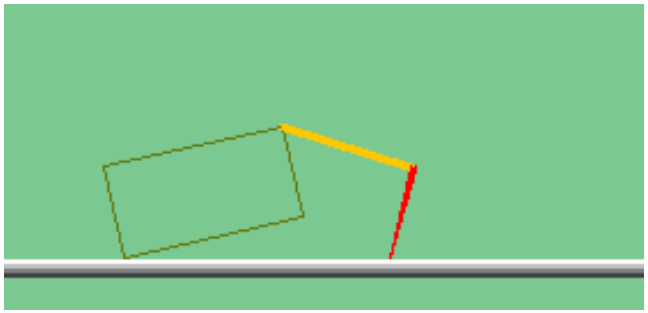
Outline

- Last time: Active RL
  - Q-learning
  - Exploration vs. Exploitation
  - Exploration functions
  - Regret

- Today: Efficient Q-learning
  - Approximate Q-learning
  - Feature-based representations
  - Connection to online least squares
  - Policy search main idea

Reinforcement Learning

- Still assume an 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 $\pi(s)$
- New twist: don't know T or R



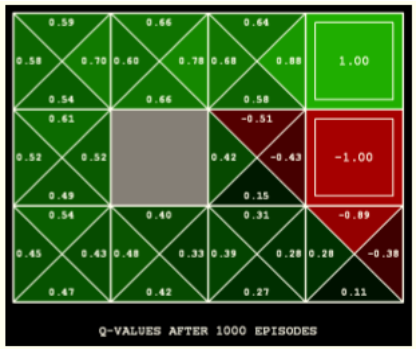
- **Big idea**: Compute all averages over T using sample outcomes

Recall: Q-Learning

- Q-Learning: sample-based Q-value iteration

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

- Learn Q(s,a) values as you go
  - Receive a sample (s,a,s',r)
  - Consider your old estimate: $Q(s,a)$
  - Consider your new sample estimate:



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

  - 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, even if you're acting suboptimally!

- This is called off-policy learning.

- Caveats:
  - If you explore enough
  - If you make the learning rate small enough
  - … but not decrease it too quickly!
  - Basically in the limit it doesn't matter how you select actions (!)

The Story So Far: MDPs and RL

Things we know how to do:

- If we know the MDP: offline
  - Compute V*, Q*, π* exactly
  - Evaluate a fixed policy π
- If we don't know the MDP: online
  - 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

Techniques:

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

Recall: Exploration Functions

- When to explore?
  - Random actions: explore a fixed amount
  - Better idea: explore areas whose badness is not (yet) established, eventually stop exploring.
- Exploration function
  - Takes a value estimate and a visit count n, and returns an optimistic utility, e.g. $f(u, n) = u + k/n$

Regular Q-Update

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

Modified Q-Update

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

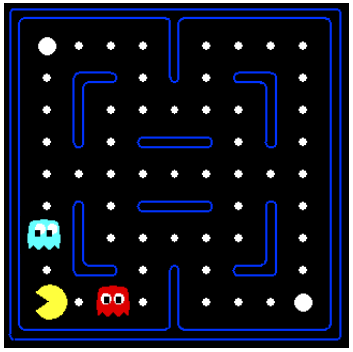
  - Note: this propagates the "bonus" back to states that lead to unknown states as well!

Generalizing across states

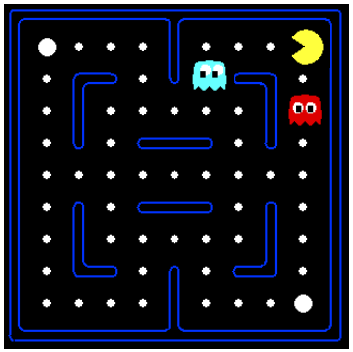
- Basic Q-Learning keeps a table of all q-values
- In realistic situations, we cannot possibly learn about every single state!
  - Too many states to visit them all in training
  - Too many states to hold the q-tables in memory
- Instead, we want to generalize:
  - Learn about some small number of training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

Example: Pacman

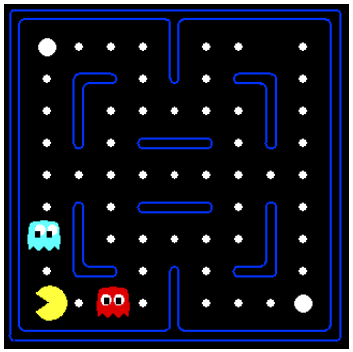
- Let's say we discover through experience that this state is bad:



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

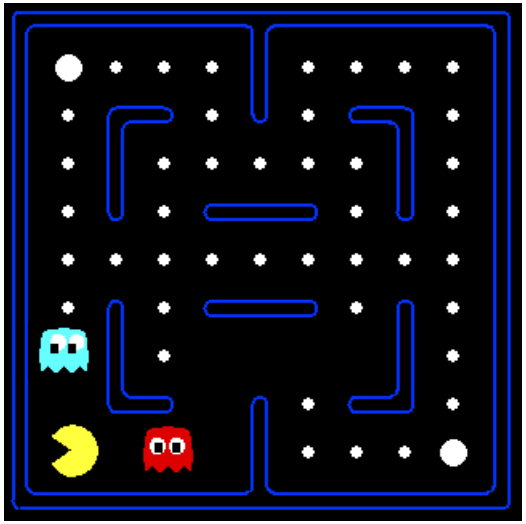


- Or even this one!



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)



Linear Value Functions

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

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

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

- Advantage: our experience is summed up in a few powerful numbers

- Disadvantage: states may share features but actually be very different in value!

Approximate Q-learning

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

- Q-learning with linear q-functions:

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

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

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

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

- Intuitive interpretation:
  - Adjust weights of active features
  - E.g. if something unexpectedly bad happens, we start to prefer <u>less</u> all states with that state's features

Example: Pacman with approx. Q-learning

$$Q(s,a) = 4.0 f_{DOT}(s,a) - 1.0 f_{GST}(s,a)$$

$$f_{DOT}(s, \text{NORTH}) = 0.5$$

$$f_{GST}(s, \text{NORTH}) = 1.0$$

$$Q(s,a) = +1$$

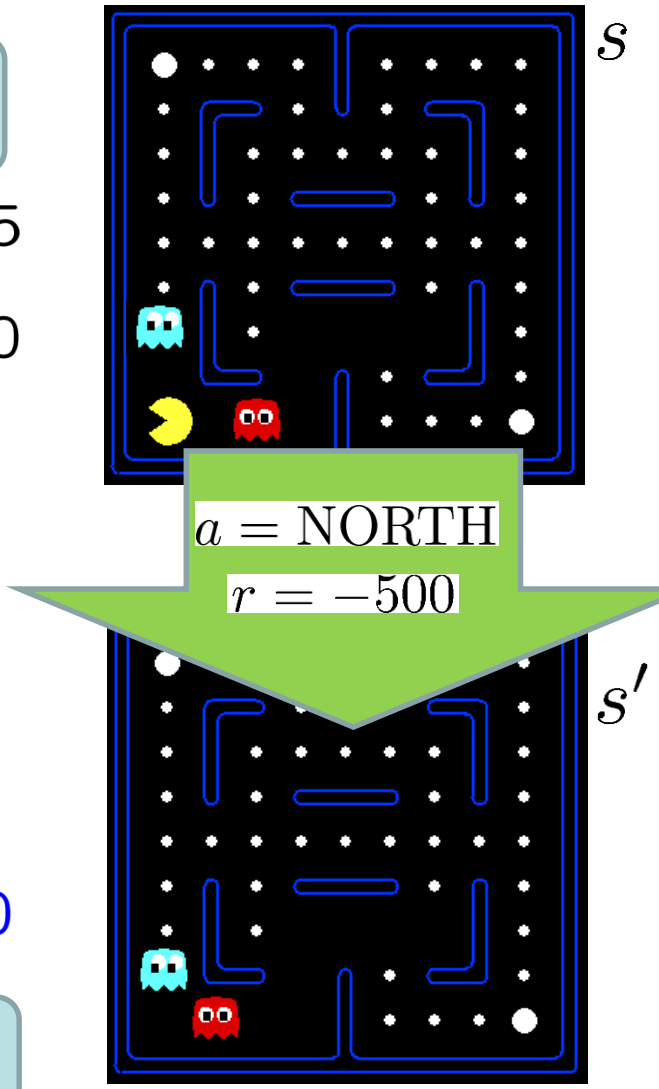
$$R(s,a,s') = -500$$

$$\left[r + \gamma \max_{a'} Q(s',a')\right] - Q(s,a) \longrightarrow difference = -501$$

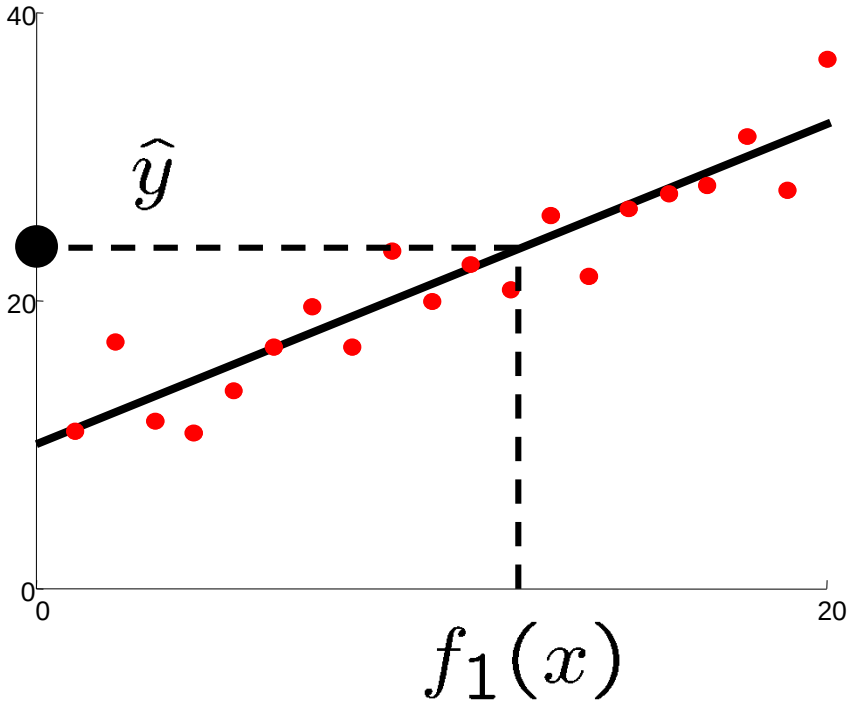
$$w_{DOT} \leftarrow 4.0 + \alpha\,[-501]\,0.5$$

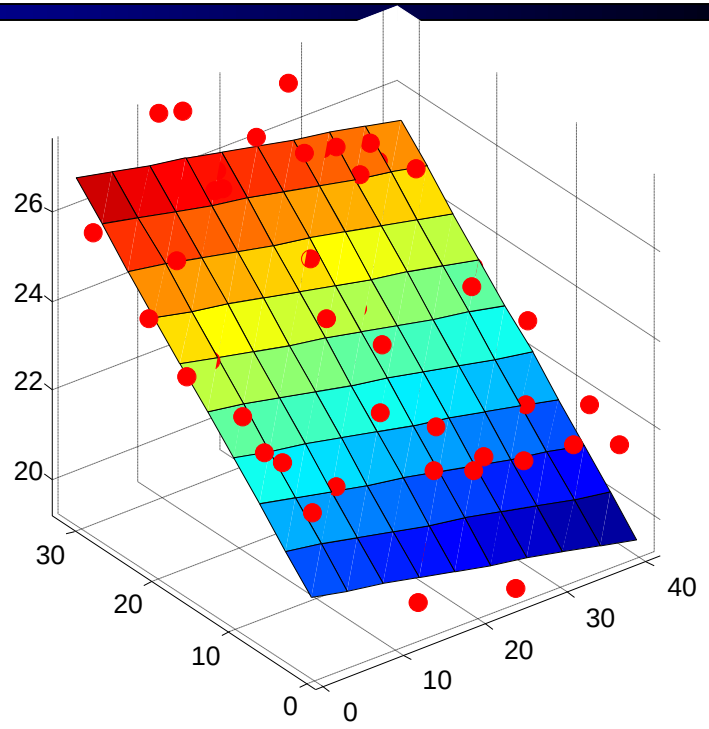
$$w_{GST} \leftarrow -1.0 + \alpha\,[-501]\,1.0$$

$$Q(s,a) = 3.0 f_{DOT}(s,a) - 3.0 f_{GST}(s,a)$$



Linear approximation: Regression





Prediction

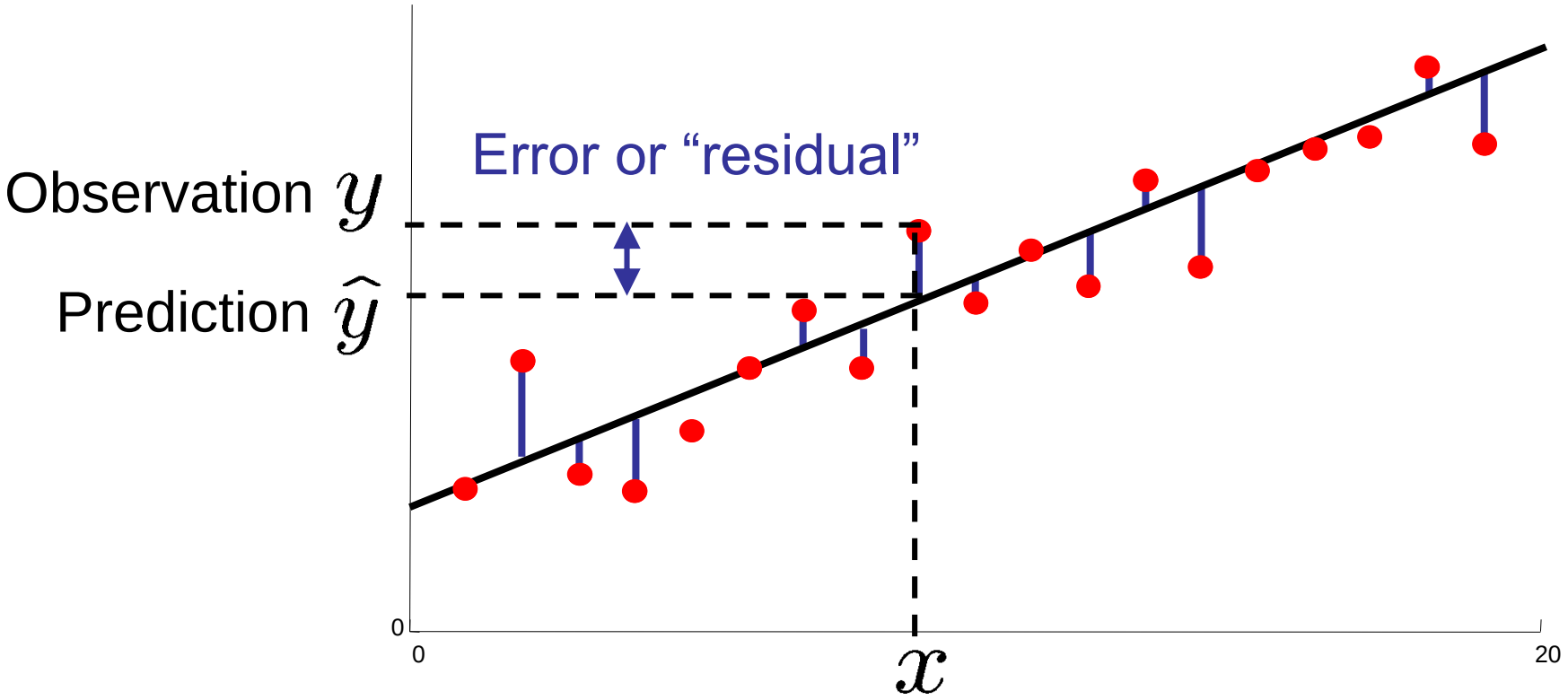
$$\widehat{y} = w_0 + w_1 f_1(x)$$

Prediction

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

Optimization: Least squares

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



Minimizing Error

Imagine we had only one point x with features f(x), target value y, and weights w:

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

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

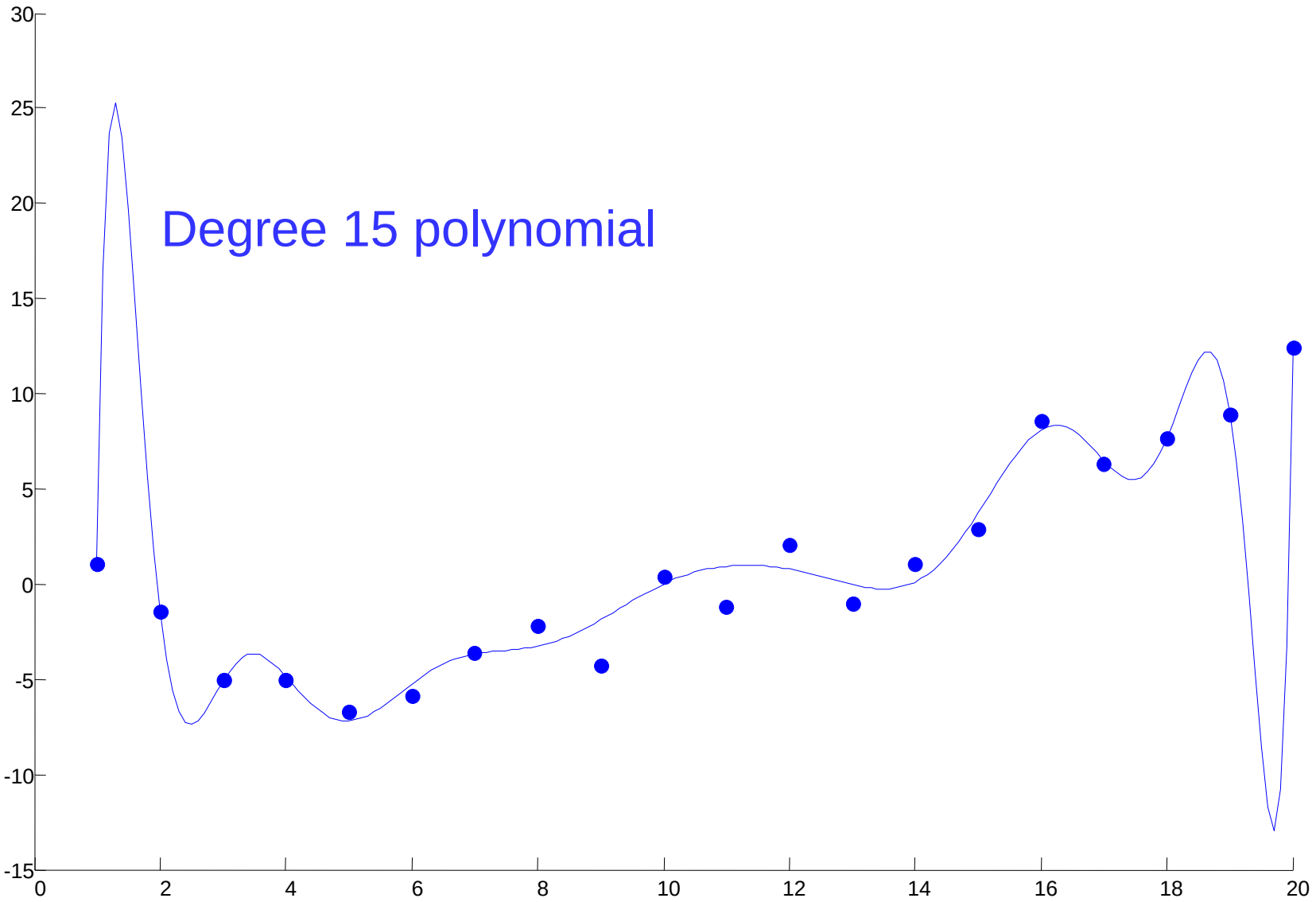
$$w_m \leftarrow w_m + \alpha\left(y - \sum_k w_k f_k(x)\right) f_m(x)$$

Approximate q update explained:

"target" "prediction"

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

Overfitting: why limiting capacity can help



Quiz: feature-based reps

Consider the following feature based representation of the Q-function:

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

with

$$f_1(s,a) = 1/(\text{distance to nearest dot after having executed action } a \text{ in state } s)$$

$$f_2(s,a) = (\text{distance to nearest ghost after having executed action } a \text{ in state } s)$$

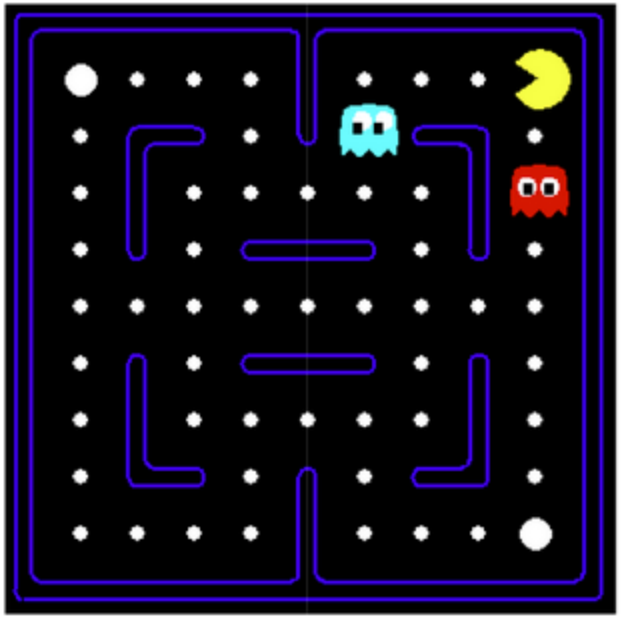
$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a)$$

Quiz: feature-based reps (part1)

$f_1(s,a) = 1/(\text{distance to nearest dot after having executed action } a \text{ in state } s)$

$f_2(s,a) = (\text{distance to nearest ghost after having executed action } a \text{ in state } s)$

- Assume w_1=1, w_2=10.
- For the state s shown below, assume that red and blue ghosts are both sitting on top of a dot.



Q(s,West) = ?

Q(s, South) = ?

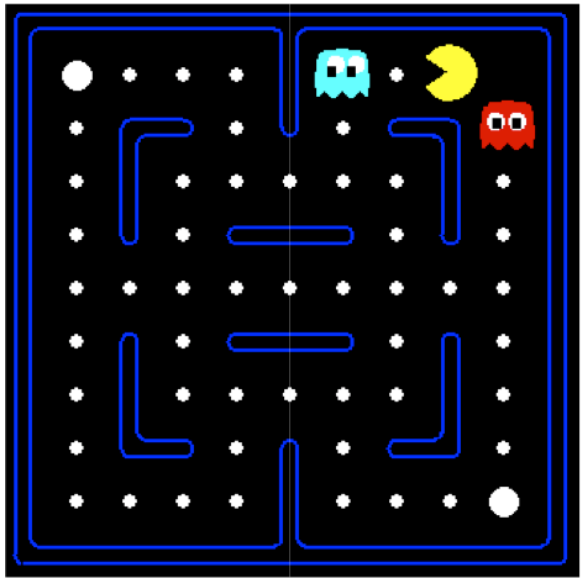
Based on this approx. Q function, the action chosen would be ?

Quiz: feature-based reps (part2)

$$f_1(s,a) = 1/(\text{distance to nearest dot after having executed action } a \text{ in state } s)$$

$$f_2(s,a) = (\text{distance to nearest ghost after having executed action } a \text{ in state } s)$$

- Assume w_1=1, w_2=10.
- For the state s shown below, assume that red and blue ghosts are both sitting on top of a dot.
- Assume Pacman moves West, resulting in s' below.
- Reward for this transition is r=+10 – 1 = 9 (+10 for food, -1 for time passed)



Q(s',West) = ?

Q(s', East) = ?

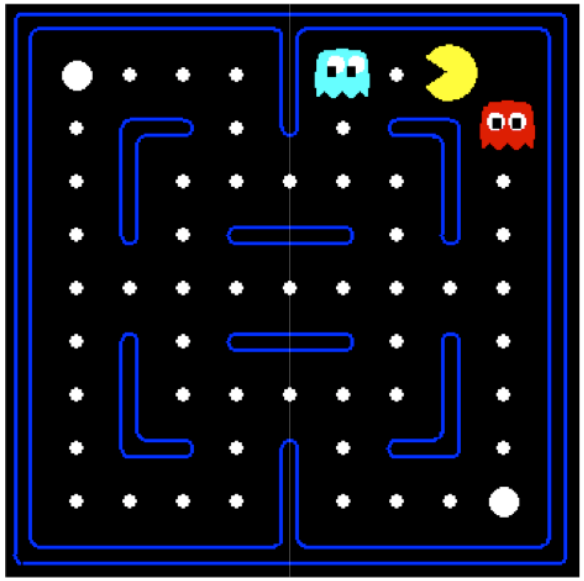
What is the sample value (assuming γ= 1)?

Quiz: feature-based reps (part3)

$$f_1(s,a) = 1/(\text{distance to nearest dot after having executed action } a \text{ in state } s)$$

$$f_2(s,a) = (\text{distance to nearest ghost after having executed action } a \text{ in state } s)$$

- Assume w_1=1, w_2=10.
- For the state s shown below, assume that red and blue ghosts are both sitting on top of a dot.
- Assume Pacman moves West, resulting in s' below. Alpha = 0.5
- Reward for this transition is r=+10 – 1 = 9 (+10 for food, -1 for time passed)



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

$$w_1 \leftarrow w_1 + \alpha(\text{difference}) f_1(s,a) =$$

$$w_2 \leftarrow w_2 + \alpha(\text{difference}) f_2(s,a) =$$

Policy Search

- **Problem**: Often the feature-based policies that work well (win games, maximize utilities) aren't the ones that approximate V / Q best
  - E.g. your value functions from project 2 were probably horrible estimates of future rewards, but they still produced good decisions
  - Q-learning's priority: get Q-values close (modeling)
  - Action selection priority: get **ordering** of Q-values right (prediction)
  - We'll see this distinction between modeling and prediction again later in the course

- **Solution**: learn the policy that maximizes rewards rather than the value that predicts rewards

- **Policy search**: start with an ok solution (e.g., Q learning), then fine-tune by hill climbing on feature weights.

Policy Search

- Simplest policy search:
  - Start with an initial linear value function or q-function
  - Nudge each feature weight up and down and see if your policy is better than before
- Problems:
  - How do we tell the policy got better?
  - Need to run many sample episodes!
  - If there are a lot of features, this can be impractical
- Better methods exploit lookahead structure, sample wisely, change multiple parameters…

Take a Deep Breath…

- We're done with search and planning!
- Next, we'll look at how to reason with probabilities
  - Diagnosis
  - Tracking objects
  - Speech recognition
  - Robot mapping
  - … lots more!
- Last part of course: machine learning