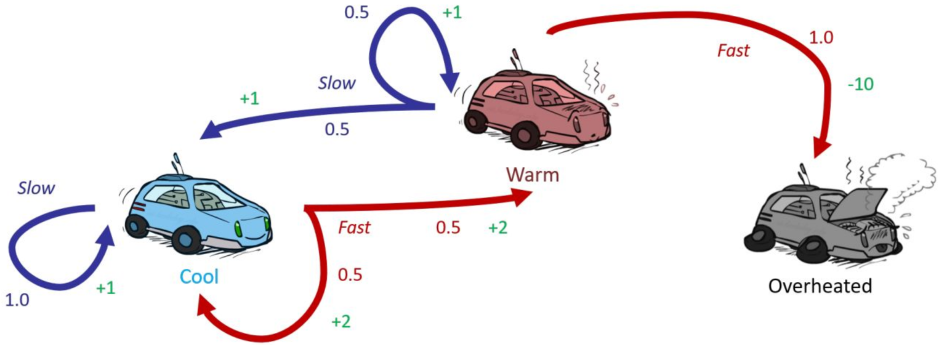
Reinforcement Learning II

CSE 4617: Artificial Intelligence

Isham Tashdeed
Lecturer, CSE

Formalizing RL Problems

- Still assume an MDP
  - A set of states $s \in S$
  - A set of actions per state A
  - A probability model $T(s, a, s')$
  - A reward function $R(s, a, s')$
- Still trying to find the most optimal policy π^*
- We do not know $T(s, a, s')$ and $R(s, a, s')$
- Must actually try out actions and states to learn



MDP vs RL

Known MDP: Offline Solution

Goal	Technique
Compute V*, Q*, π*	Value / policy iteration
Evaluate a fixed policy π	Policy evaluation

Unknown MDP: Model-Based

Goal	Technique
Compute V*, Q*, π*	VI/PI on approx. MDP
Evaluate a fixed policy π	PE on approx. MDP

Unknown MDP: Model-Free

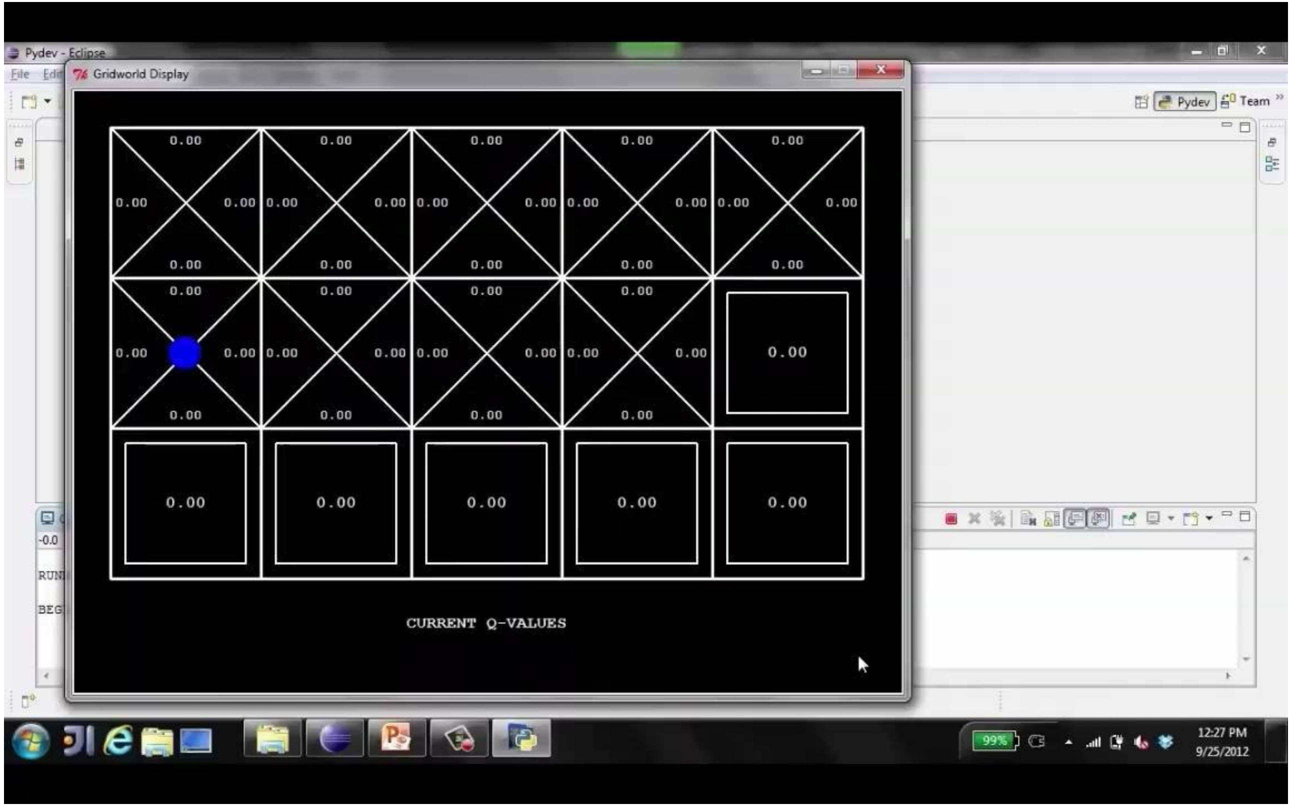
Goal	Technique
Compute V*, Q*, π*	Q-learning
Evaluate a fixed policy π	Value Learning

Q-Learning

Learn $Q(s, a)$ as you go

- Receive a sample (s, a, s', r)
- Consider the old estimate $Q(s, a)$
- New sample estimate $\rightarrow$ $sample = R(s, a, s') + \gamma \max_{a'} Q(s', a')$
- Incorporate the new estimate into the running average
  - $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha\,(sample)$

Q-Learning



Pydev - Eclipse
Gridworld Display
0.00
CURRENT Q-VALUES
Pydev
Team
12:27 PM
9/25/2012

Q-Learning Properties

- Q-learning converges to optimal policy → even if you're acting suboptimally!
- This is called off-policy learning

Cons:

- You have to explore enough
- You have to eventually make the learning rate (α) small enough
- But not decrease the learning rate (α) too quickly

How to Explore?

Schemes for forcing exploration

- Random action (ε-greedy):
  - Every time step, flip a coin
  - With a very small probability (ε), stop following the established policy and act randomly
  - With a large probability (1-ε), keep following the established policy
  - Setting ε to a fixed value ensures sufficient exploration but the agent will not get a chance to use what it has learned
  - Gradually decrease ε

Exploration Functions

Explore areas whose **badness/goodness** is not (yet) established, eventually stop exploring

- Takes a value estimate u and a visit count n, and returns an optimistic utility: $f(u, n) = u + k/(n+1)$
- Modified Q-update:

$$Q(s,a) \leftarrow_{\alpha} R(s,a,s') + \gamma \max_{a'} f(Q(s',a'), N(s',a'))$$

Regret

Regret

- Even if you learn the optimal policy, you still make mistakes along the way!
- Regret is the difference between the total expected rewards and optimal expected rewards
- Gives a notion of how quickly an agent learns
- Minimizing regret means you will optimally learn how to be optimal!
- Random exploration and exploration functions both end up optimal, but random exploration has higher regret

Approximate Q-Learning

Generalizing Across States

- Simple Q-Learning keeps a table of all q-values → Or in a dictionary!
- In realistic situations, we cannot possibly learn about every single state! → Too many states!
  - Too many states to visit them all in training
  - Too many states to hold the q-tables in memory
- A better idea is to generalize to unseen states
  - Learn about a small number of states
  - Generalize the experience to novel yet similar states
  - This idea can be borrowed from machine learning

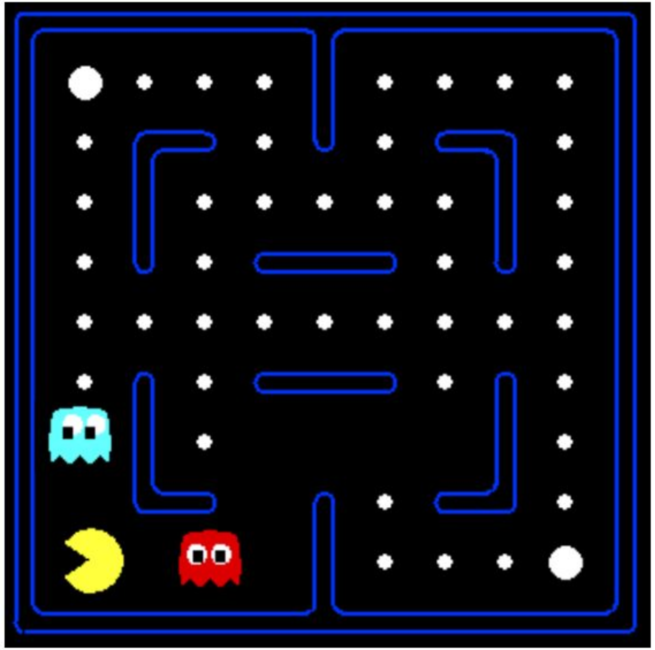
Generalizing Across States

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



- Describe a state using a vector of features
- Features are functions from states to real numbers (often [0, 1]) that capture important properties of the state
- They can be hand crafted → Similar to evaluation functions of a state
    - Distance to closest ghost
    - Distance to closest food
    - Number of ghosts
    - 1 / (distance to food)^2
    - Is Pacman in a tunnel? (0/1) → True/False
- Or these vectors can be learned → Deep Learning

Linear Value Functions

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

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

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

- Q-learning will learn the weights (w_1, w_2 ... w_n) to approximate the values of the states
- If the number of features are not adequate, your agent will not be able to tell states apart, even though they are very different from each other

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:

$$\text{transition } = (s, a, r, s')$$

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

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

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

- If something unexpectedly bad happens, blame the features that were on: disprefer all states with that state's features

Approximate Q-Learning

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

s

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

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

$a = \text{NORTH}$
$r = -500$

s'

$$Q(s, \text{NORTH}) = +1$$

$$r + \gamma \max_{a'} Q(s', a') = -500 + 0$$

$$Q(s', \cdot) = 0$$

$$\text{difference} = -501$$

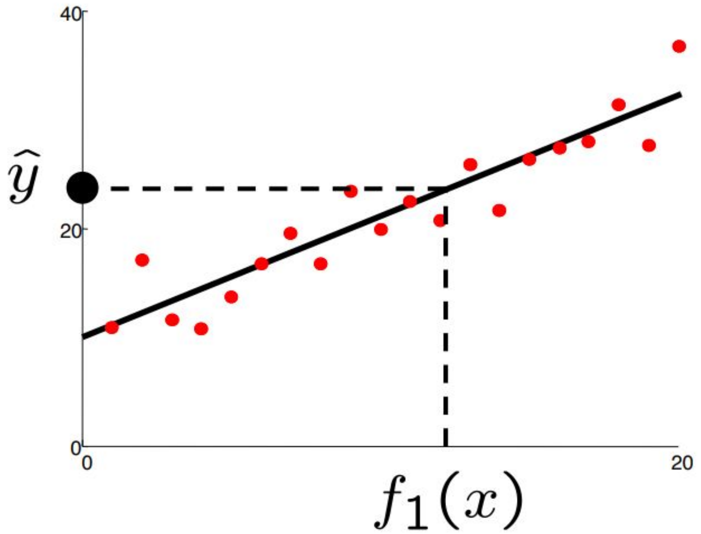
$$w_{DOT} \leftarrow 4.0 + \alpha\,[-501]\,0.5$$

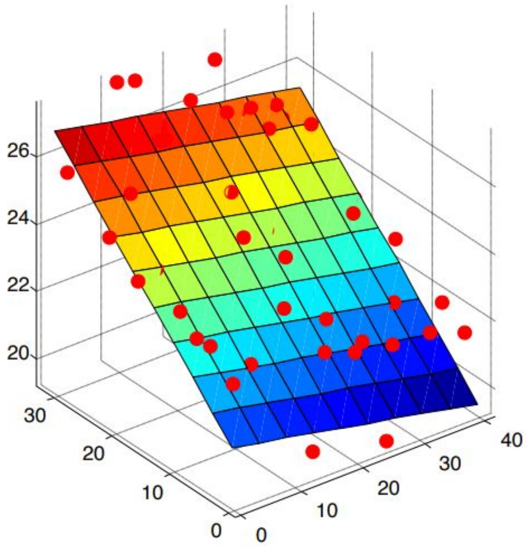
$$w_{GST} \leftarrow -1.0 + \alpha\,[-501]\,1.0$$

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

Least Squares Approximation

Linear Regression





Prediction:

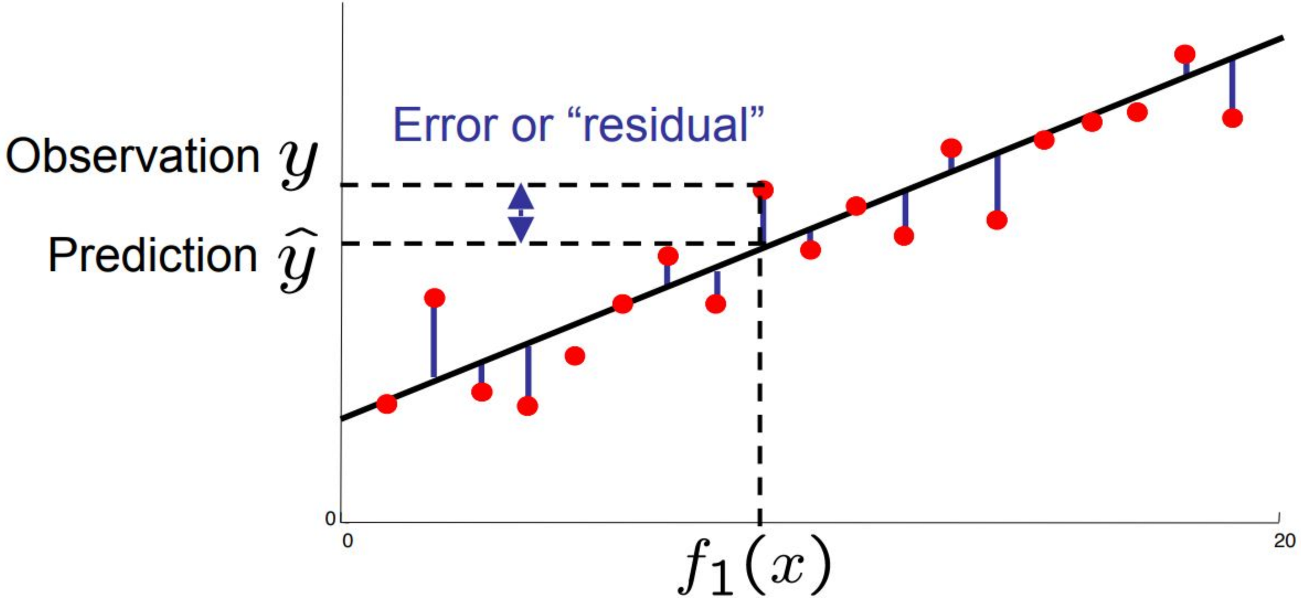
$$\hat{y} = w_0 + w_1 f_1(x)$$

Prediction:

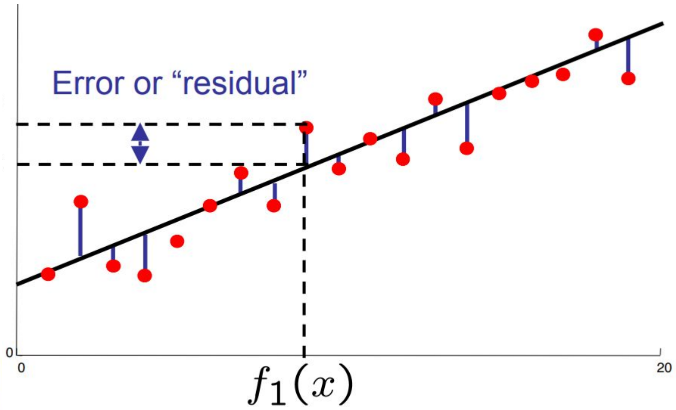
$$\hat{y}_i = w_0 + w_1 f_1(x) + w_2 f_2(x)$$

Linear Regression

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



Linear Regression



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

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

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

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

"target" "prediction"

Thank you