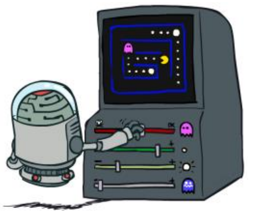
CS 473: Artificial Intelligence

Reinforcement Learning III



Travis Mandel (filling in for Dan) / University of Washington

[Most slides were taken from Dan Klein and Pieter Abbeel / CS188 Intro to AI at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.]

Logistics

- PS3 – due 11/12

2

Reinforcement Learning Recap

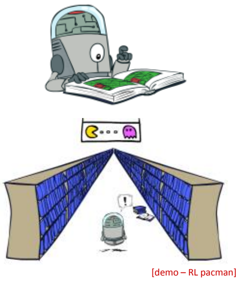
- Model-based approach
- Model-free approaches
  - TD-learning
  - Tabular Q-Learning
  - Epsilon-Greedy, Exploration Functions
  - TODAY: Approximate Linear Q-Learning

4

Approximate Q-Learning

Generalizing Across States

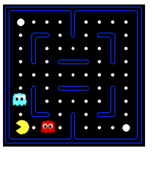
- Basic Q-Learning keeps a table of all q-values
- In realistic situations, we cannot possibly learn about every single state!
  - Too many states to visit them all in training
  - Too many states to hold the q-tables in memory
- Instead, we want to generalize:
  - Learn about some small number of training 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



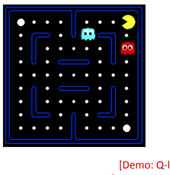
[demo – RL pacman]

Example: Pacman

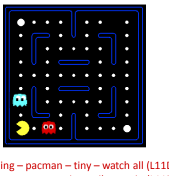
Let's say we discover through experience that this state is bad:



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



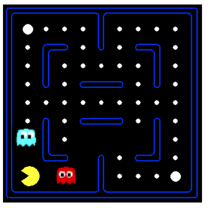
Or even this one!



[Demo: Q-learning – pacman – tiny – watch all (L11D5)]
[Demo: Q-learning – pacman – tiny – silent train (L11D6)]
[Demo: Q-learning – pacman – tricky – watch all (L11D7)]

Feature-Based Representations

- Solution: describe a state using a **vector of features** (aka "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)



How to use features?

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

$$V(s) = g(f_1(s), f_2(s), \dots, f_n(s))$$

$$Q(s,a) = g(f_1(s), f_2(s), \dots, f_n(s))$$

How to use features?

- 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

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

- Q-learning with linear Q-functions:

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

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

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

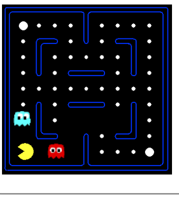
$w_i \leftarrow w_i + \alpha$ [difference] $f_i(s,a)$ Approximate Q's

- Intuitive interpretation:
  - Adjust weights of active features
  - E.g., if something unexpectedly bad happens, blame the features that were on: disprefer all states with that state's features
- Formal justification: in a few slides!

Example: Pacman Features

$$Q(s,a) = w_1 f_{DOT}(s,a) + w_2 f_{GST}(s,a)$$

s



$$f_{DOT}(s,a) = \frac{1}{\textit{distance to closest food after taking } a}$$

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

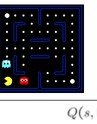
$$f_{GST}(s,a) = \textit{distance to closest ghost after taking } a$$

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

Example: Q-Pacman

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

s

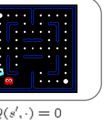


$f_{DOT}(s, \text{NORTH}) = 0.5$

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

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

s'



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

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

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

difference $= -501$

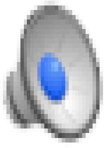
$w_{DOT} \leftarrow 4.0 + \alpha\,[-501]\,0.5$

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

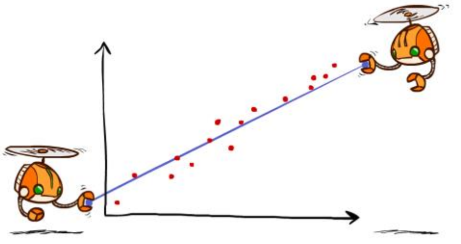
$$Q(s,a) = 3.0 f_{DOT}(s,a) - 3.0 f_{GST}(s,a)$$

[Demo: approximate Q-learning pacman (L11D10)]

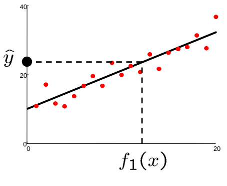
Video of Demo Approximate Q-Learning -- Pacman

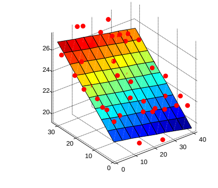


Sidebar: Q-Learning and Least Squares



Linear Approximation: Regression



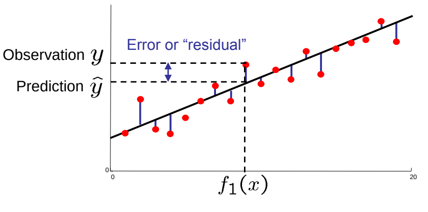


Prediction:
$\hat{y} = w_0 + w_1 f_1(x)$

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

Optimization: Least Squares

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



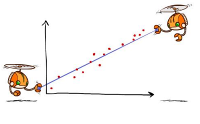
Minimizing Error

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

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

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

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

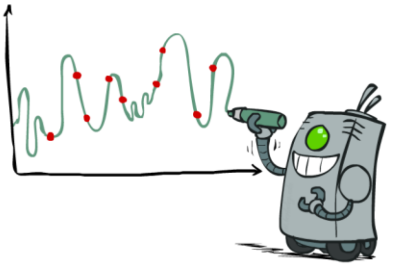


Approximate q update explained:

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

"target" "prediction"

Overfitting: Why Limiting Capacity Can Help



Simple Problem

Given: Features of current state
Predict: Will Pacman die on the next step?

Just one feature. See a pattern?

- Ghost one step away, pacman dies
- Ghost one step away, pacman dies
- Ghost one step away, pacman dies
- Ghost one step away, pacman dies
- Ghost one step away, pacman lives
- Ghost more than one step away, pacman lives
- Ghost more than one step away, pacman lives
- Ghost more than one step away, pacman lives
- Ghost more than one step away, pacman lives
- Ghost more than one step away, pacman lives
- Ghost more than one step away, pacman lives

Learn: Ghost one step away → pacman dies!

See a pattern?

- Ghost one step away, pacman dies
- Ghost one step away, pacman dies
- Ghost one step away, pacman dies
- Ghost one step away, pacman dies
- Ghost one step away, pacman lives
- Ghost more than one step away, pacman lives
- Ghost more than one step away, pacman lives
- Ghost more than one step away, pacman lives
- Ghost more than one step away, pacman lives
- Ghost more than one step away, pacman lives
- Ghost more than one step away, pacman lives

Learn: Ghost one step away → pacman dies!

What if we add more features?

- Ghost one step away, score 211, pacman dies
- Ghost one step away, score 341, pacman dies
- Ghost one step away, score 231, pacman dies
- Ghost one step away, score 121, pacman dies
- Ghost one step away, score 301, pacman lives
- Ghost more than one step away, score 205, pacman lives
- Ghost more than one step away, score 441, pacman lives
- Ghost more than one step away, score 219, pacman lives
- Ghost more than one step away, score 199, pacman lives
- Ghost more than one step away, score 331, pacman lives
- Ghost more than one step away, score 251, pacman lives

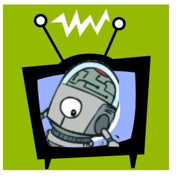
Learn: Ghost one step away AND score is NOT 301 → pacman dies!

What if we add more features?

- Ghost one step away, score 211, pacman dies
- Ghost one step away, score 341, pacman dies
- Ghost one step away, score 231, pacman dies
- Ghost one step away, score 121, pacman dies
- Ghost one step away, score 301, pacman lives
- Ghost more than one step away, score 205, pacman lives
- Ghost more than one step away, score 441, pacman lives
- Ghost more than one step away, score 219, pacman lives
- Ghost more than one step away, score 199, pacman lives
- Ghost more than one step away, score 331, pacman lives
- Ghost more than one step away, score 251, pacman lives

Learn: Ghost one step away AND score is NOT 301 → pacman dies!

Normal Programming now resuming…



That's all for Reinforcement Learning!

- Very tough problem: How to perform any task well in an unknown, noisy environment!
- Traditionally used mostly for robotics, but becoming more widely used
- Lots of open research areas:
  - How to best balance exploration and exploitation?
  - How to deal with cases where we don't know a good state/feature representation?

CS 473: Artificial Intelligence

Probability

Instructor: Travis Mandel --- University of Washington

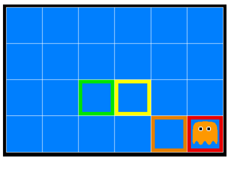
[These slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.]

Next

- Probability
  - Random Variables
  - Joint and Marginal Distributions
  - Conditional Distribution
  - Product Rule, Chain Rule, Bayes' Rule
  - Inference
  - Independence
- You'll need all this stuff A LOT for the next few weeks, so make sure you go over it now!

Inference in Ghostbusters

- A ghost is in the grid somewhere
- Sensor readings tell how close a square is to the ghost
  - On the ghost: red
  - 1 or 2 away: orange
  - 3 or 4 away: yellow
  - 5+ away: green

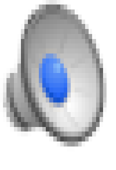


- Sensors are noisy, but we know P(Color | Distance)

P(red | 3)	P(orange | 3)	P(yellow | 3)	P(green | 3)
0.05	0.15	0.5	0.3

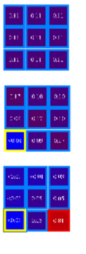
[Demo: Ghostbuster – no probability (L12D1)]

Video of Demo Ghostbuster – No probability



Uncertainty

- General situation:
  - **Observed variables (evidence)**: Agent knows certain things about the state of the world (e.g., sensor readings or symptoms)
  - **Unobserved variables**: Agent needs to reason about other aspects (e.g. where an object is or what disease is present)
  - **Model**: Agent knows something about how the known variables relate to the unknown variables
- Probabilistic reasoning gives us a framework for managing our beliefs and knowledge



Random Variables

- A random variable is some aspect of the world about which we (may) have uncertainty
  - R = Is it raining?
  - T = Is it hot or cold?
  - D = How long will it take to drive to work?
  - L = Where is the ghost?
- We denote random variables with capital letters
- Like variables in a CSP, random variables have domains
  - R in {true, false} (often write as {+r, -r})
  - T in {hot, cold}
  - D in [0, ∞)
  - L in possible locations, maybe {(0,0), (0,1), …}

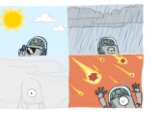
Probability Distributions

- Associate a probability with each value
  - Temperature:

$P(T)$

T	P
hot	0.5
cold	0.5

  - Weather:



$P(W)$

W	P
sun	0.6
rain	0.1
fog	0.3
meteor	0.0

Probability Distributions

- Unobserved random variables have distributions

$P(T)$

T	P
hot	0.5
cold	0.5

$P(W)$

W	P
sun	0.6
rain	0.1
fog	0.3
meteor	0.0

Shorthand notation:

$$P(hot) = P(T = hot),$$
$$P(cold) = P(T = cold),$$
$$P(rain) = P(W = rain),$$
$$\dots$$

OK *if* all domain entries are unique

- A distribution is a TABLE of probabilities of values
- A probability (lower case value) is a single number

$$P(W = rain) = 0.1$$

- Must have: $\forall x \;\; P(X = x) \geq 0$ and $\sum_x P(X = x) = 1$

Joint Distributions

- A *joint distribution* over a set of random variables: $X_1, X_2, \dots X_n$ specifies a real number for each assignment (or *outcome*):

$$P(X_1 = x_1, X_2 = x_2, \dots X_n = x_n)$$
$$P(x_1, x_2, \dots x_n)$$

  - Must obey: $P(x_1, x_2, \dots x_n) \geq 0$

$$\sum_{(x_1, x_2, \dots x_n)} P(x_1, x_2, \dots x_n) = 1$$

$P(T, W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

- Size of distribution if n variables with domain sizes d?
  - For all but the smallest distributions, impractical to write out!

Probabilistic Models

- A probabilistic model is a joint distribution over a set of random variables
- Probabilistic models:
  - (Random) variables with domains
  - Assignments are called *outcomes*
  - Joint distributions: say whether assignments (outcomes) are likely
  - *Normalized:* sum to 1.0
  - Ideally: only certain variables directly interact
- Constraint satisfaction problems:
  - Variables with domains
  - Constraints: state whether assignments are possible
  - Ideally: only certain variables directly interact

Distribution over T,W

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

Constraint over T,W

T	W	P
hot	sun	T
hot	rain	F
cold	sun	F
cold	rain	T