Reinforcement Learning Recap

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

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:
  1. Too many states to visit them all in training
  2. Too many states to hold the q-tables in memory
- Instead, we want to generalize:
  1. Learn about some small number of training states from experience
  2. Generalize that experience to new, similar situations
  3. 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!

Logistics

- PS3 – due 11/12
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
    - Number of dots
    - Number of ghosts
    - 1 / (dist to dot)
    - Is Pacman in a tunnel? (0/1)
    - … etc.
  - 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:
  \[ V(s) = g(f_1(s), f_2(s), ..., f_n(s)) \]
  \[ Q(s, a) = g(f_1(s), f_2(s), ..., f_n(s)) \]

Example: Pacman Features

\[ Q(s, a) = w_1 f_DOT(s, a) + w_2 f_GST(s, a) \]

\[ f_{\text{DIST}}(s, a) = \text{distance to closest food after taking } a \]
\[ f_{\text{GHOST}}(s, \text{NORTH}) = 0.5 \]
\[ f_{\text{GHOST}}(s, a) = \text{distance to closest ghost after taking } a \]
\[ f_{\text{GHOST}}(s, \text{NORTH}) = 1.0 \]

How to use features?

- Using a feature representation, we can write a q function (or value function) for any state using a few weights:
  - 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-learning with linear Q-functions:
  \[ Q(s, a) = w_1 f_1(s, a) + w_2 f_2(s, a) + \ldots + w_n f_n(s, a) \]
  - 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: Q-Pacman

\[ Q(s, a) = 4.0 f_{\text{DIST}}(s, a) - 1.0 f_{\text{GOST}}(s, a) \]

[Diagram: approximate Q-learning pacman (L11D10)]
Video of Demo Approximate Q-Learning -- Pacman

Sidebar: Q-Learning and Least Squares

Linear Approximation: Regression

Optimization: Least Squares

Minimizing Error

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

$$e = \frac{1}{2} \left[ y - \sum_i w_i f_i(x) \right]^2$$

$$\frac{\partial e}{\partial w} = -\left[ y - \sum_i w_i f_i(x) \right] f_i(x)$$

Approximate q update explained:

$$w_k \leftarrow w_k + \alpha \left[ r + \gamma \max_a \{ Q(s', a') - Q(s, a) \} f_a(x) \right]$$

"target" "prediction"

Overfitting: Why Limiting Capacity Can Help
Simple Problem

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

<table>
<thead>
<tr>
<th>Feature Description</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ghost one step away</td>
<td>Pacman dies</td>
</tr>
<tr>
<td>Ghost more than one step away</td>
<td>Pacman lives</td>
</tr>
</tbody>
</table>

Learn: Ghost one step away \(\rightarrow\) Pacman dies!

See a pattern?

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

<table>
<thead>
<tr>
<th>Feature Description</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ghost one step away</td>
<td>Pacman dies</td>
</tr>
<tr>
<td>Ghost more than one step away</td>
<td>Pacman lives</td>
</tr>
</tbody>
</table>

Learn: Ghost one step away \(\rightarrow\) Pacman dies!

What if we add more features?

<table>
<thead>
<tr>
<th>Feature Description</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ghost one step away, score 211</td>
<td>Pacman dies</td>
</tr>
<tr>
<td>Ghost one step away, score 341</td>
<td>Pacman dies</td>
</tr>
<tr>
<td>Ghost more than one step away, score 205</td>
<td>Pacman lives</td>
</tr>
</tbody>
</table>

Learn: Ghost one step away AND score is NOT 301 \(\rightarrow\) Pacman dies!

Normal Programming now resuming...

<table>
<thead>
<tr>
<th>Feature Description</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ghost one step away, score 211</td>
<td>Pacman dies</td>
</tr>
<tr>
<td>Ghost one step away, score 341</td>
<td>Pacman dies</td>
</tr>
<tr>
<td>Ghost one step away, score 231</td>
<td>Pacman dies</td>
</tr>
<tr>
<td>Ghost more than one step away, score 331</td>
<td>Pacman lives</td>
</tr>
</tbody>
</table>

Learn: Ghost one step away AND score is NOT 301 \(\rightarrow\) Pacman dies!
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?

Next

- Probability
  - Random Variables
  - Joint and Marginal Distributions
  - Conditional Distribution
  - Product Rule, Chain Rule, Bayes’ Rule
  - 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)

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
  - B = {true, false} (often write as (+r, -r)}
  - T = {hot, cold}
  - D = [0, ∞)
  - L = possible locations, maybe {(0,0), (0,1), ...}

Probability Distributions

- Associate a probability with each value
  - Temperature:
    - P(T)
      - hot: 0.5
      - cold: 0.5
    - Weather:
      - P(W)
        - sun: 0.6
        - rain: 0.1
        - fog: 0.3
        - meteor: 0.0

Joint Distributions

- A joint distribution over a set of random variables: $X_1, X_2, \ldots, X_n$ specifies a real number for each assignment (or outcome):
  - P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)
  - P(T, W)
    - 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!