Q-Learning Wrap-Up
Discussion: Bidirectional Search
guaranteed to meet in the middle

Andrea Danyluk
March 6, 2017

Announcements
• Programming Assignment 2 code reviews today
• Turn in reading responses
• Midterm this week
  – Will find it in your CS mailbox by tomorrow at 10am
    (or in mine, if you don’t have a mailbox)
  – Take it out when ready to do it; Complete by 4:30pm
  – Mark start date/time and end date/time; Turn in
    immediately after end
  – Turn in “cheat sheet” with exam
• RL assignment now posted
  – Confirm partners with me by Monday 9am

Today
• Q-Learning Wrap-Up
• Discussion

Pacman
• States?
• Actions?
• Transition Model?
• Rewards?
Demo

Pacman
• States?
• Actions?
• Transition Model?
• Rewards?
Ability to generalize?

Q-Learning in the Real World
• In many cases, too many states
  – Might not be able to hold the Q-values in memory
  – Can’t visit all during training
    • Or even if we can visit them, can’t do so enough
• Want to make use of the power of generalization
Feature-Based Representations

• Describe a state using a vector of features (properties)
  – Features are functions from states to real numbers (sometimes just 0/1)
  – Features capture important properties of the state
  – Pacman examples:
    • Distance to closest ghost [closest food, etc]
    • Number of ghosts [food, etc]
    • Is Pacman in a tunnel?
    • Is Pacman trapped?
  – Can describe a Q state (i.e. $Q(s, a)$) with features, too

Values (utilities) as approximated by evaluation functions

• $V(s) = w_1 f_1(s) + w_2 f_2(s) + ... + w_n f_n(s)$
  – Recall your minimax evaluation functions!
• $Q(s, a) = w_1 f_1(s,a) + w_2 f_2(s,a) + ... + w_n f_n(s,a)$
  – Learn values for the weights $w_1, w_2, ..., w_n$ such that the evaluation function approximates the true value (utility)

Say we have three features:

• PowerPellet <= 1 (1=T, 0=F)
• ScaryGhost <= 1 (1=T, 0=F)
• Food <= 3 (1=T, 0=F)

• Say $w_1 = 0.8$, $w_2 = 0.5$, $w_3 = 0.4$
  Then
  $Q(s_{cur}, E) = .8(0) + .5(0) + .4(1) = .4$

Say we have three features:

• PowerPellet <= 1 (1=T, 0=F)
• ScaryGhost <= 1 (1=T, 0=F)
• Food <= 3 (1=T, 0=F)

• Say $w_1 = 0.8$, $w_2 = 0.5$, $w_3 = 0.4$
  Then
  $Q(s_{cur}, S) = .8(1) + .5(1) + .4(1) = 1.7$

Learning weights for linear $Q$-functions

Before:

$\text{sample} = R(s,a,s') + \gamma \max_a Q(s', a')$
$Q(s, a) = (1-\alpha) \cdot Q(s, a) + \alpha \cdot \text{sample}$
$Q(s, a) = Q(s, a) + \alpha \cdot (\text{sample} – Q(s, a))$

$w_1 = w_1 + \alpha \cdot (\text{sample} \sim \text{current}) [f_1(s,a)]$
$w_2 = w_2 + \alpha \cdot (\text{sample} \sim \text{current}) [f_2(s,a)]$

$w_1 = 0.8 + 0.1(-10 + \gamma(0) - 1.7) = .8 + 0.1(-11.7) = -.37$
$w_2 = 0.5 + 0.1(-10 + \gamma(0) - 1.7) = .5 + 0.1(-11.7) = -.67$
$w_3 = 0.4 + 0.1(-10 + \gamma(0) - 1.7) = .4 + 0.1(-11.7) = -.77$

Learning weights for linear $Q$-functions

Before:

$\text{sample} = R(s,a,s') + \gamma \max_a Q(s', a')$
$Q(s, a) = (1-\alpha) \cdot Q(s, a) + \alpha \cdot \text{sample}$
$Q(s, a) = Q(s, a) + \alpha \cdot (\text{sample} – Q(s, a))$

$w_1 = w_1 + \alpha \cdot (\text{sample} \sim \text{current}) [f_1(s,a)]$
$w_2 = w_2 + \alpha \cdot (\text{sample} \sim \text{current}) [f_2(s,a)]$

$w_1 = 0.8 + 0.1(-10 + \gamma(0) - 1.7) = .8 + 0.1(-11.7) = -.37$
$w_2 = 0.5 + 0.1(-10 + \gamma(0) - 1.7) = .5 + 0.1(-11.7) = -.67$
$w_3 = 0.4 + 0.1(-10 + \gamma(0) - 1.7) = .4 + 0.1(-11.7) = -.77$
Why?

Ordinary Least Squares

- Aim to minimize squared error:
  \[ \frac{1}{2} (\text{current} - \text{obs total reward})^2 \]
- The rate of change of the error wrt each \( w \) parameter is the partial derivative:
  \[
  \begin{align*}
  (w_1 f_1(s,a) + w_2 f_2(s,a) - \text{obs total reward}) f_1(s,a) \\
  (w_1 f_1(s,a) + w_2 f_2(s,a) - \text{obs total reward}) f_2(s,a)
  \end{align*}
  \]

Pros and Cons of Function Approximation

**Pros**
- Makes it practical to handle very large state spaces
- Allows the learner to generalize from states it has visited to states it has not yet seen

**Cons**
- There might not be a good function in the chosen hypothesis space (defined by the choice of features)
- Tradeoff between the size of the hypothesis space and the learning time
- As always, need to take care with learning rate parameter

Why?

Ordinary Least Squares

- The squared error defines a surface in \((n+1)\)-dim space, where \( n \) is the number of parameters.
- To reach the minimum in an online fashion, we “step” along the surface in the direction opposite the gradient
  \[
  \begin{align*}
  w_1 &= w_1 + \alpha(\text{obs total reward} - \text{current}) f_1(s,a) \\
  w_2 &= w_2 + \alpha(\text{obs total reward} - \text{current}) f_2(s,a)
  \end{align*}
  \]

Demo: RL Pacman