

APPROXIMATE Q-LEARNING

CSE 511A: Introduction to Artificial Intelligence

Some content and images are from slides 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>.

1

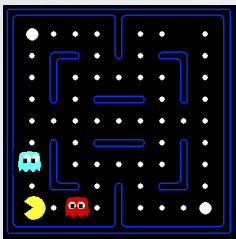
APPROXIMATE Q-LEARNING

- Problem with Q-learning:
 - Needs a lot of memory to store all Q-values
 - Good actions learned for some state cannot be generalized to other similar states

2

APPROXIMATE Q-LEARNING

- Problem with Q-learning:
 - Needs a lot of memory to store all Q-values
 - Good actions learned for some state cannot be generalized to other similar states

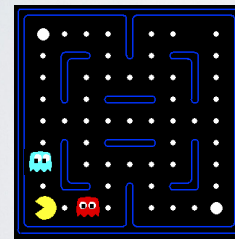


Let's say we learned through experience that this state is bad

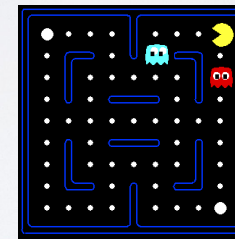
3

APPROXIMATE Q-LEARNING

- Problem with Q-learning:
 - Needs a lot of memory to store all Q-values
 - Good actions learned for some state cannot be generalized to other similar states



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

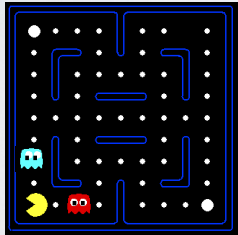


In naive Q-learning, that knowledge doesn't translate to this state

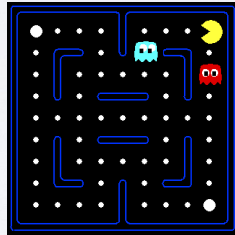
4

APPROXIMATE Q-LEARNING

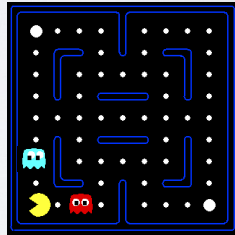
- Problem with Q-learning:
 - Needs a lot of memory to store all Q-values
 - Good actions learned for some state cannot be generalized to other similar states



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



In naive Q-learning, that knowledge doesn't translate to this state



Or worse, not even to this one!

5

APPROXIMATE Q-LEARNING

- Problem with Q-learning:
 - Needs a lot of memory to store all Q-values
 - Good actions learned for some state cannot be generalized to other similar states
 - Takes too long to learn good actions for every state
 - Think about the number of states there is in your Pacman problem
- Goal:
 - Learn what to do on small number of training states
 - Generalize experiences and knowledge learned to new, similar situations
 - Fundamental idea in machine learning, and we will see it over and over again

You do this everyday:
You acquire knowledge in school, learn from textbooks, lectures, etc.,
and then generalize that knowledge after you graduate (ideally :))

6

APPROXIMATE Q-LEARNING

- Main idea: Describe a state using a vector of features (properties)
 - Features are functions from states to real numbers that capture important properties of the state
 - Example features (you probably thought about a lot of these in project 2):
 - Distance to closest ghost
 - Distance to closest dot
 - Number of ghosts
 - $1 / (\text{distance to dot})^2$
 - Is in a tunnel

7

APPROXIMATE Q-LEARNING

- Main idea: Describe a state using a vector of features (properties)
 - Features are functions from states to real numbers that capture important properties of the state
 - Example features (you probably thought about a lot of these in project 2):
 - Distance to closest ghost
 - Distance to closest dot
 - Number of ghosts
 - $1 / (\text{distance to dot})^2$
 - Is in a tunnel
 - Often a good idea to normalize the outputs of the functions so that they are all within the same range (e.g. $[-1, 1]$ or $[0, 1]$). Otherwise, features with larger possible values are more important by default

8

APPROXIMATE Q-LEARNING

- Using a feature representation, we can define the Q-values (or even V-values) for any state using a few weights:

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

- Goal: Learn the weights!
 - Note that there are a lot fewer weights than there are states → Faster to learn!!

9

APPROXIMATE Q-LEARNING

- Using a feature representation, we can define the Q-values (or even V-values) for any state using a few weights:

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

- Goal: Learn the weights!
 - Note that there are a lot fewer weights than there are states → Faster to learn!!
 - The individual weights reflect how important its feature is compared to the others

10

APPROXIMATE Q-LEARNING

- Using a feature representation, we can define the Q-values (or even V-values) for any state using a few weights:

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

- Goal: Learn the weights!
 - Note that there are a lot fewer weights than there are states → Faster to learn!!
 - The individual weights reflect how important its feature is compared to the others
 - States that have the same (or similar) features will now have the same value
 - Can be both good and bad
 - If you define features appropriately, then its good
 - If you define features inappropriately (e.g., you ignore an important feature like where the ghost is in Pacman or you only store the x-coordinate of a ghost's location), then its bad

11

APPROXIMATE Q-LEARNING

- Using a feature representation, we can define the Q-values (or even V-values) for any state using a few weights:

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

- Goal: Learn the weights!
 - Note that there are a lot fewer weights than there are states → Faster to learn!!
 - The individual weights reflect how important its feature is compared to the others
 - States that have the same (or similar) features will now have the same value

$$\begin{aligned} \text{difference} &= [r + \gamma \max_{a'} Q(s', a')] - Q(s, a) \\ w_i &= w_i + \alpha \cdot \text{difference} \cdot f_i(s, a) \end{aligned}$$

12

APPROXIMATE Q-LEARNING

- In summary,

- Approximate Q-learning:

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

$$w_i = w_i + \alpha \cdot \text{difference} \cdot f_i(s, a)$$

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

- Exact Q-learning:

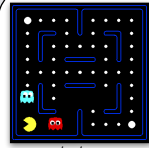
$$Q(s, a) = Q(s, a) + \alpha \cdot \text{difference}$$

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

13

APPROXIMATE Q-LEARNING

$$\text{Q-function before update: } Q(s, a) = 4.0 f_{DOT}(s, a) - 1.0 f_{GST}(s, a)$$



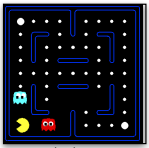
$f_{DOT}(s, N) = 0.5$
 $f_{GST}(s, N) = 1.0$
 $Q(s, N) = 1$

state s

14

APPROXIMATE Q-LEARNING


$$\text{Q-function before update: } Q(s, a) = 4.0 f_{DOT}(s, a) - 1.0 f_{GST}(s, a)$$



$f_{DOT}(s, N) = 0.5$
 $f_{GST}(s, N) = 1.0$
 $Q(s, N) = 1$

state s

➔



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

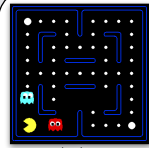
state s'

$a = N$
 $r = -500$

15

APPROXIMATE Q-LEARNING

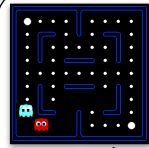
$$\text{Q-function before update: } Q(s, a) = 4.0 f_{DOT}(s, a) - 1.0 f_{GST}(s, a)$$



$f_{DOT}(s, N) = 0.5$
 $f_{GST}(s, N) = 1.0$
 $Q(s, N) = 1$

state s

➔



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

state s'

$a = N$
 $r = -500$

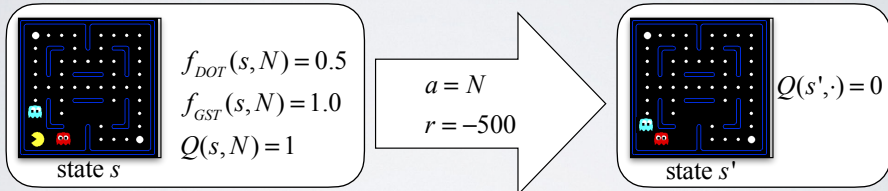
$$\text{Sample value: } r + \gamma \max_{a'} Q(s', a') = -500 + 0$$

$$\text{Difference: } \text{sample} - Q(s, a) = -501$$

16

APPROXIMATE Q-LEARNING

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



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

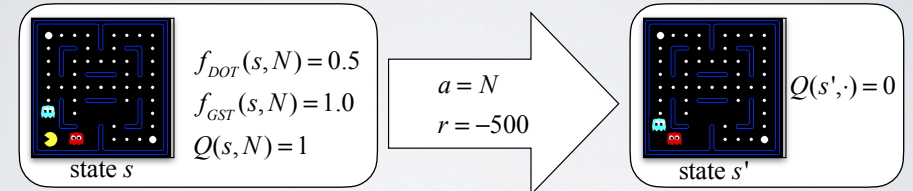
Difference: $sample - Q(s, a) = -501$

Weight updates: $w_{DOT} = 4.0 + \alpha \cdot (-501) \cdot 0.5$
 $w_{GST} = -1.0 + \alpha \cdot (-501) \cdot 1.0$

17

APPROXIMATE Q-LEARNING

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



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

Difference: $sample - Q(s, a) = -501$

Weight updates: $w_{DOT} = 4.0 + \alpha \cdot (-501) \cdot 0.5$
 $w_{GST} = -1.0 + \alpha \cdot (-501) \cdot 1.0$

Q-function after update: $Q(s, a) = 3.0 f_{DOT}(s, a) - 3.0 f_{GST}(s, a)$

18