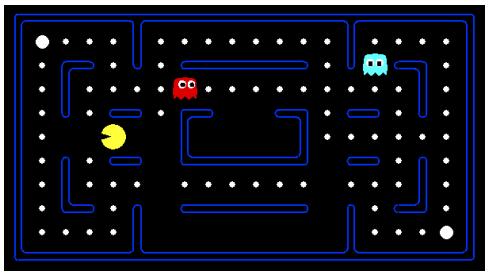
CS188 Fall 2013 Section 5: Reinforcement Learning

1 Learning with Feature-based Representations

We would like to use a Q-learning agent for Pacman, but the state size for a large grid is too massive to hold in memory (just like at the end of Project 3). To solve this, we will switch to feature-based representation of Pacman's state. Here's a Pacman board to refresh your memory:



1. What features would you extract from a Pacman board to judge the expected outcome of the game?

2. Say our two minimal features are the number of ghosts within 1 step of Pacman (F_g) and the number of food pellets within 1 step of Pacman (F_p). For this pacman board:

   Extract the two features (calculate their values).

3. With Q Learning, we train off of a few episodes, so our weights begin to take on values. Right now $w_g = 100$ and $w_p = -10$. Calculate the Q value for the state above.

4. We receive an episode, so now we need to update our values. An episode consists of a start state s, an action a, an end state s', and a reward $R(s, a, s')$. The start state of the episode is the state above (where you already calculated the feature values and the expected Q value). The next state has feature values $F_g = 0$ and $F_p = 2$ and the reward is 50. Assuming a discount of 0.5, calculate the new estimate of the Q value for s based on this episode.

5. With this new estimate and a learning rate (α) of 0.5, update the weights for each feature.

6. Good job on updating the weights. Now let's think about this entire process one step back. What values do we learn in this process (assuming features are defined)? When we have completed learning, how do we tell if Pacman does a good job?

7. In some sense, we can think about this entire process, on a meta level, as an input we control that produces an output that we would like to maximize. If you have a magical function ($F(input)$) that maps an input to an output you would like to maximize, what techniques (from math, CS, etc) can we use to search for the best inputs? Keep in mind that the magical function is a black box.

8. Now say we can calculate the derivative of the magical function, $F'(input)$, giving us a gradient or slope. What techniques can we use now?

2 Odds and Ends

1. When using features to represent the Q-function is it guaranteed that the feature-based Q-learning finds the same optimal $Q*$ as would be found when using a tabular representation for the Q-function?

2. Why is temporal difference (TD) learning of Q-values (Q-learning) superior to TD learning of values?

3. Can all MDPs be solved using expectimax search? Justify your answer.

4. When learning with ϵ-greedy action selection, is it a good idea to decrease ϵ to 0 with time? Why or why not?