CS221 Fall 2018 Homework 5

SUNet ID: 05794739
Name: Luis Perez
Collaborators:

By turning in this assignment, I agree by the Stanford honor code and declare that all of this is my own work.

Problem 1

(a) We can write the recurrence as a straight-forward generalization of what was presented in lecture. We'll have:

$$V_{\text{minmax}}(s,d) = \begin{cases} \text{Utility}(s) & \text{IsEnd}(s) \\ \text{Eval}(s) & d = 0 \\ \max_{a \in \text{Actions}(s)} \{V_{\text{minmax}}(\text{Succ}(s,a), d)\} & \text{Player}(s) = a_0 \\ \min_{a \in \text{Actions}(s)} \{V_{\text{minmax}}(\text{Succ}(s,a), d)\} & \text{Player}(s) \in \{a_1, \cdots a_{n-1}\} \\ \min_{a \in \text{Actions}(s)} \{V_{\text{minmax}}(\text{Succ}(s,a), d-1)\} & \text{Player}(s) = a_n \end{cases}$$

(b) In "submission.py"

Problem 2

(a) In "submission.py"

Problem 3

(a) We can write the recurrence as a straight-forward generalization of what was presented in 1a. We'll have:

$$V_{\text{minmax}}(s,d) = \begin{cases} \text{Utility}(s) & \text{IsEnd}(s) \\ \text{Eval}(s) & d = 0 \\ \max_{a \in \text{Actions}(s)} \{V_{\text{minmax}}(\text{Succ}(s,a), d)\} & \text{Player}(s) = a_0 \\ \frac{1}{|\text{Actions}(s)|} \sum_{a \in \text{Actions}(s)} V_{\text{minmax}}(\text{Succ}(s,a), d) & \text{Player}(s) \in \{a_1, \cdots a_{n-1}\} \\ \frac{1}{|\text{Actions}(s)|} \sum_{a \in \text{Actions}(s)} V_{\text{minmax}}(\text{Succ}(s,a), d-1) & \text{Player}(s) = a_n \end{cases}$$

(b) In "submission.py"

Problem 3

(a) In “submission.py”

(b) After further trial, we’ve managed to achieve an average winning score of 1554. We did this by making just a few small tweaks to the algorith from our previous resonse.

- When we’re getting close to winning, we “look ahead” to see if we’re going to end-up in a winning-state. If we are, we add 500 points to the score. The reasoning behind this is to encourage the Pacman to eat the last-pellet, as we noticed frequently that it would fail to do so.

PREVIOUSR RESPONSE 2: Note that I tried a lot of things, many of which I didn’t document here. In general, the final approach taken which achieved an average of 1505 points on the winning games (rather than 1337 as the PREVIOUS RESPONSE approach below), is as follows.

The idea is basically to improve on our previous approach in two ways. The first, we consider which states are better. Our value for a state is given by the sume of the following:

- Current score.
- If we ignore ghost positions, what score will we achieve if we greedily (using BFS) eat the nearest food items as we move across the board from eating the remaining food (ie, we get 10 for each food eaten, -1 for each step taken)
- In order to encourage the eating of scared ghosts, for each scared ghost, if we’re eating it, we receive 200 points. Otherwise, we receive $200 - d$ where d is the number of steps it would take us to get to the ghost to eat him. This is basically just an estimate for the reward we’d receive if we ate the ghosts.

The above is sufficient to achive > 1500 average score, with a win rate of 90%.

PREVIOUS RESPONSE 1:

The idea behind the below is as follows.

Losing is always avoided, significantly, by heavily penalizing all such states. Winning is heavily rewarded (above just a simple score), so that PacMan always tries to win.

For everything else, we use BFS to compute distances to food items, scared ghosts, and normal ghosts from pacman. We then use a simply formula (arrived at after experimentation) to combine these features into a single store.

Intuitively, here are the properties we’re generally looking for:

(a) A higher game score implies a higher evaluation score.

(b) Closer food items imply higher scores.

(c) A lower number of capsules implies a higher score.

(d) A lower number of food items left implies a higher score.

(e) Closer scared ghosts imply a higher score (because we can eat them)

(f) Further active ghosts imply a higher score (because they won't eat us)