CS221 Fall 2018 Homework [pacman]

SUNet ID: prabhjot
Name: Prabhjot Singh Rai

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) Minmax algorithm for multiple adversaries is given by:

$$V_{minmax}(s,d) = \begin{cases} \text{Utility}(s), & \text{IsEnd}(s) \\ \text{Eval}(s), & d = 0 \\ max_{a \epsilon Actions(s)} V_{minmax}(Succ(s,a),d), & Player(s) = agent_0 \\ min_{a \epsilon Actions(s)} V_{minmax}(Succ(s,a),d), & Player(s) = agent_1 \\ min_{a \epsilon Actions(s)} V_{minmax}(Succ(s,a),d), & Player(s) = agent_2 \\ ... & \\ min_{a \epsilon Actions(s)} V_{minmax}(Succ(s,a),d-1), & Player(s) = agent_n \end{cases}$$

Where, Utility(s) is the function which gives us the utility at state s, isEnd tells us if s is an end state, Eval(s) is the evaluation function for state s(an estimation for $V_{minmax}(s)$), Player(s) returns whose turn it is.

Here $agent_0$ is the pacman, thus it will be playing max over possible rewards from successor states, whereas $agent_1$ to $agent_n$ are the ghosts, which will try to minimise the rewards.

Since a single depth consists of all the players making a move, we decrease the depth on last agent's/player's turn.

Problem 3

(a) Since we know what policy the ghosts are following, we would like to train our pacman against the specific policy. Therefore, since there are random ghosts, we follow

expectimax policy, given by:

$$V_{exptmax}(s,d) = \begin{cases} \text{Utility}(s), & \text{IsEnd}(s) \\ \text{Eval}(s), & d = 0 \\ max_{a \epsilon Actions(s)} V_{exptmax}(Succ(s,a),d), & Player(s) = agent_0 \\ \frac{\sum_{a \epsilon Actions(s)} V_{exptmax}(Succ(s,a),d)}{length(Actions(s))}, & Player(s) = agent_1 \\ \frac{\sum_{a \epsilon Actions(s)} V_{exptmax}(Succ(s,a),d)}{length(Actions(s))}, & Player(s) = agent_2 \\ ... & \\ \frac{\sum_{a \epsilon Actions(s)} V_{exptmax}(Succ(s,a),d-1)}{length(Actions(s))}, & Player(s) = agent_n \end{cases}$$

Where, Utility(s) is the function which gives us the utility at state s, isEnd tells us if s is an end state, Eval(s) is the evaluation function for state s(an estimation for $V_{minmax}(s)$), Player(s) returns whose turn it is.

Here $agent_0$ is the pacman, thus it will be playing max over possible rewards from successor states, whereas $agent_1$ to $agent_n$ are the ghosts, which we know are following the policy of taking a random decision.

Since a single depth consists of all the players making a move, we decrease the depth on last agent's/player's turn.

Problem 4

(b) The evaluation function takes 5 factors into consideration.

(a) Current Game Score: Keep higher score favorable at any state

(b) Closest Distance from Food: The more closer the pacman is from food, it should be more favorable

(c) Closest Ghost Scared: Pacman should try to be as close to a scared ghost as possible, in order to gain high rewards by eating a scared ghost

(d) Number of capsules left: The number of capsules left should also be considered, as the pacman should try to minimise the capsules on the board so that it encounters more ghosts.

(e) Number of food left: As eating more food is the way to win the game, least number of food on the board is favorable

There were some issues I faced during creating this evaluation function:

(a) Pacman was getting stuck on one side and keep moving between two positions, solved it by adding distance from food variable

(b) Tried to make pacman go away from active ghosts, but that was not yielding high results. Changed the code to add high negative score for loosing but remove distance from ghost as a factor, since this way pacman can pick up food even when ghost is closer to him unless in the next state, ghost is going to eat him. This way, I was able to get much better score.

The weights of the different parameters were decided in order to scale them with score. Moreover, as you can see, highest weight goes to eating closest ghost (since it adds to score the most), than eating the food (closest distance from food). Moreover, eating capsules is given higher weights (more reduction in capsules left, higher the reciprocal) over the food left since pacman should consider eating all the capsules first in order to maximise the profit through creating more scared ghosts than eating the food left to end the game.