CS221 Fall 2016 Assignment5: Pacman

SUNet ID: 06041808
Name : Kai Cheng Chang (Victor)

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) Write the recurrence for $V_{max,min}(s,d)$:

$$V_{max,min}(s,d) = \begin{cases} \mathrm{U}tility(s) & \text{if IsEnd(s)} \\ \mathrm{E}val(s) & \text{if d=0} \\ \max_{a \in Actions(s)} V_{max,min}(Succ(s,a),d) & \text{if Agent=pacman} \\ \min_{a \in Actions(s)} V_{max,min}(Succ(s,a),d) & \text{if Agent=ghost(n), } 1 < n < (N-1) \\ \min_{a \in Actions(s)} V_{max,min}(Succ(s,a),d-1) & \text{if Agent=ghost(N)} \end{cases}$$

Problem 3

(a) Write the recurrence for $V_{max,min}(s,d)$:

$$V_{max,opp}(s,d) = \begin{cases} \mathrm{U}tility(s) & \text{if IsEnd(s)} \\ \mathrm{E}val(s) & \text{if d=0} \\ \max_{a \in Actions(s)} V_{max,opp}(Succ(s,a),d) & \text{if Player(s)=pacman} \\ \sum_{a \in Actions(s)} \pi_{opp}(s,a) V_{max,opp}(Succ(s,a),d) & \text{if Players(s)=ghost(n),} \\ & \text{ n} \in \{1,2,...,N-1\} \\ \sum_{a \in Actions(s)} \pi_{opp}(s,a) V_{max,opp}(Succ(s,a),d-1) & \text{if Players(s)=ghost(N)} \end{cases}$$

$$\pi_{opp}(s,a) = \frac{1}{numberOfLegalActions}$$

Problem 4

(a) Eval(s)=score=

$$\begin{cases} w_{Score} * currentGameState.getScore() + \\ w_{nF} * (1/float(nF)) + w_{nSG} * (1/float(nSG)) + \\ w_{scaredT} * sum(ScaredTimes) - w_{nG} * (1/float(nG)) \end{cases}$$

[**Explanation for each feature**]
<u>Intuition:</u>
Instead of merely considering the score of the current game state, we need to input some guidance for pacman to move toward the "right" direction. The guidance I add here are "distance to Food", "distance to scarred Ghost", "distance to normal Ghost", "total scaredTimes." The high level strategy here is that on one hand, we want pacMan to get close to Food and scarred Ghost; when there is a capsule nearby, eat it, which is rewarded by scaredTimes since once a capsule is eaten, the scaredTimes will increase quite a bit. On the other hand, we want pacMan to escape from Ghost if it is too close to a normal Ghost. Using reciprocal is to indicate the inverse relationship of distance and reward. For example, closer to food is actually better, thus the reciprocal will give us positive guidance.

- currentGameState.getScore():current score shown on the screen
- nF: how far away from the nearest Food (using UCS to search)
- nSG: how far away from the nearest scared Ghost(using A* to search in which h(s)=manhattanDistance(pacPos,scaredGhost)
- sum(ScaredTimes):total current scared time. I use this to let pacMan knows that if eat a capsule, reward will be high.
- nG:how far away from the nearest non-scared Ghost (again, using A* with the same h(s) for nSG)

[**Weight for each feature**]
<u>Hand-tuned coefficients:</u>
$w_Score, w_nF, w_nG, w_nSG, w_scaredT = [1, 2, -0.1, 60, 0.2]$
if sum(scaredTimes)=0 $\implies w_{nSG} = 0$: don't want to get close to that ghost anymore.
if nG == 0: w_{nG}=0 ; nG=1
if nC == 0: w_{nC}=0 ; nC=1
if nF == 0: w_{nF}=0 ; nF=1
if nSG==0 : return float('+inf')
[**Result**]
Pacman emerges victorious: Score: 1760
Pacman emerges victorious: Score: 1762
Pacman emerges victorious: Score: 1743
Pacman emerges victorious: Score: 1745
Pacman emerges victorious: Score: 1770
Pacman emerges victorious: Score: 1764
Pacman emerges victorious: Score: 1558
Pacman emerges victorious: Score: 1751
Pacman emerges victorious: Score: 1764
Pacman emerges victorious: Score: 1544

Average Score: 1716.1
Win Rate: 10/10 (1.00)
Record:Win, Win, Win, Win, Win, Win, Win, Win, Win, Win