Assignment 2- Artificial Intelligence

muthukumar suresh

February 2015

Contents

1 question 1: Reflex Agent

1.1 my performance measurements

In my Reflex Agent function, I give a cost of:
$\frac{1}{(distance\_to\_closest\_food)} - \frac{10}{distance\_to\_closest\_ghost} + gameState\_score()$
This ensures that the pacman runs more quickly against a ghost rather than going to food.
This performs very well on boards with walls in medium and classic layouts. On small open layouts, it is not than great, around 60 % win rate. The statistics are given below:

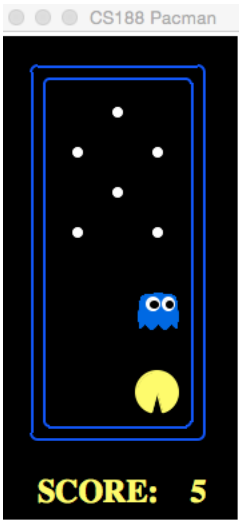

Figure 1: Board for reflex agent test: testClassic

```
command: -p ReflexAgent -l testClassic -n 1 -f
Average Score: 124.51
Win Rate:      61/100 (0.61)
nodes Expanded on average: 174.39
Time Taken on average:6.609015
```

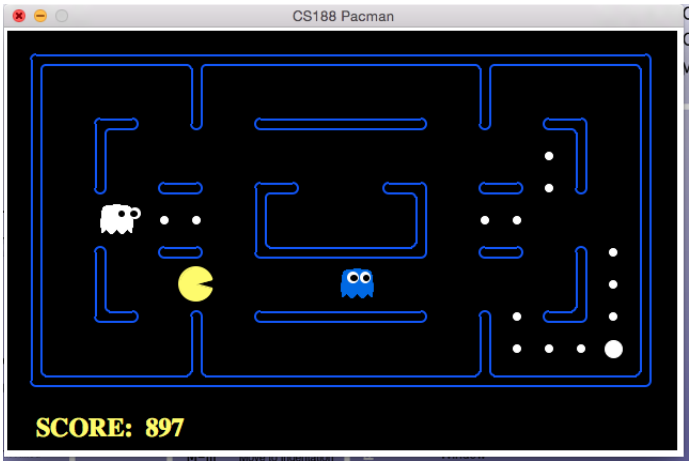

Figure 2: Board for reflex agent test: medium Classic

```
medium classic
command: --frameTime 0 -p ReflexAgent -k 2 -q -n 100
Average Score: 1162.26
Win Rate:      79/100 (0.79)
nodes Expanded on average: 564.14
Time Taken on average:57.866775
```

1.2 Autograder check for reflex agent

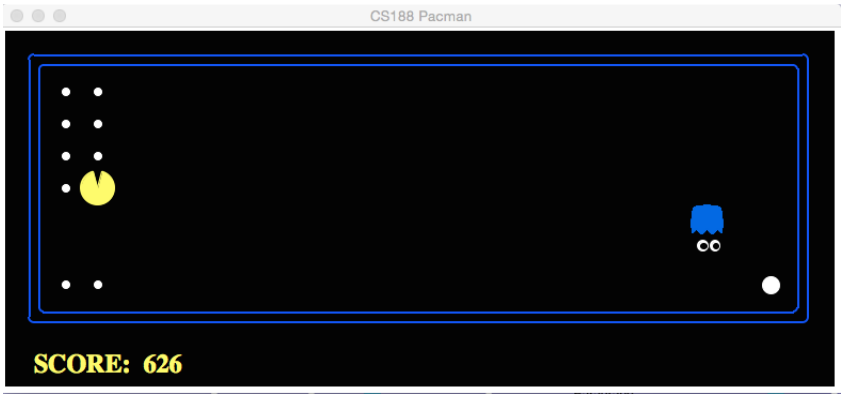

Figure 3: Board for reflex agent test: Autograder board

```
Average Score: 1187.4
Win Rate:      10/10 (1.00)
nodes Expanded: 808.0
Time Taken:6.346675
```

AUTOGRADER OUTPUT:

```
Average Score: 1214.1
Scores:        1213.0, 1125.0, 1149.0, 1182.0, 1211.0, 1203.0, 1216.0, 1204.0, 1419.0, 1219.0
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
*** PASS: test_cases/q1/grade-agent.test (3 of 3 points)
***     1214.1 average score (2 of 2 points)
***         Grading scheme:
***          < 500:  0 points
***         >= 500:  1 points
***         >= 1000:  2 points
***     10 games not timed out (0 of 0 points)
***         Grading scheme:
***          < 10:  fail
***         >= 10:  0 points
***     10 wins (1 of 1 points)
***         Grading scheme:
***          < 1:  fail
***         >= 1:  0 points
***         >= 5:  1 points

### Question q1: 3/3 ###


Finished at 20:24:12

Provisional grades
==================
Question q1: 3/3
------------------
Total: 3/3
```

```
Analysis:
The reflex agent performs well and expands lesser number of nodes than other methods but it is not very smart
at predicting the ghost movement before hand.
It performs well on big boards but on smaller boards it requires complex evaluation function to really perform
optimally
```

2 Minimax Agent

2.1 My performance evaluation

Figure 4: Board for MiniMax test

```
command: -p MinimaxAgent -l minimaxClassic -a depth=4 -n 100 -q -f
Average Score: 142.18
Win Rate:      63/100 (0.63)
nodes Expanded: 16165.37
Time Taken:161.995317
```

trapped classic always fails and it always rushes towards the closest ghost or in a general direction because in all directions, its death is iminent

2.2 Autograder output

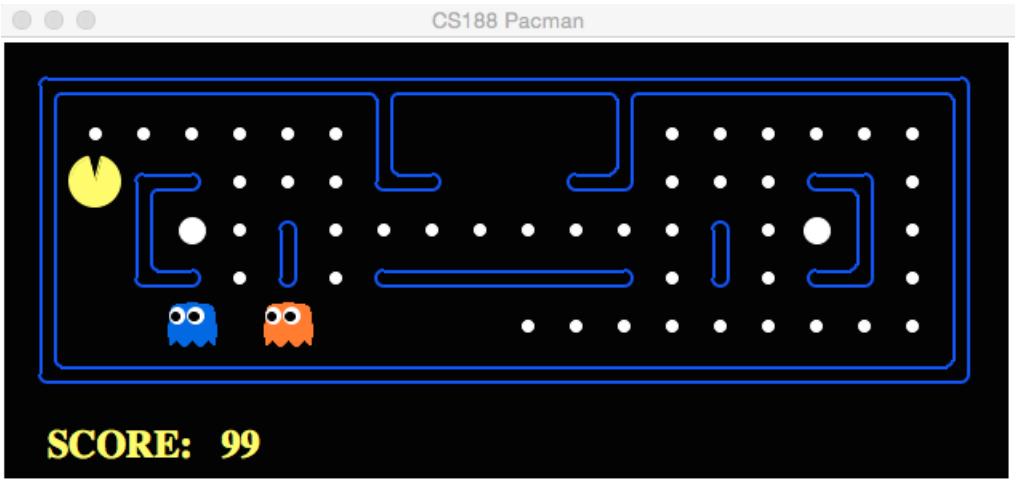

Figure 5: Board for MiniMax Autograder

```
Question q2
===========

*** PASS: test_cases/q2/0-lecture-6-tree.test
*** PASS: test_cases/q2/0-small-tree.test
*** PASS: test_cases/q2/1-1-minmax.test
*** PASS: test_cases/q2/1-2-minmax.test
*** PASS: test_cases/q2/1-3-minmax.test
*** PASS: test_cases/q2/1-4-minmax.test
*** PASS: test_cases/q2/1-5-minmax.test
```

```
*** PASS: test_cases/q2/1-6-minmax.test
*** PASS: test_cases/q2/1-7-minmax.test
*** PASS: test_cases/q2/1-8-minmax.test
*** PASS: test_cases/q2/2-1a-vary-depth.test
*** PASS: test_cases/q2/2-1b-vary-depth.test
*** PASS: test_cases/q2/2-2a-vary-depth.test
*** PASS: test_cases/q2/2-2b-vary-depth.test
*** PASS: test_cases/q2/2-3a-vary-depth.test
*** PASS: test_cases/q2/2-3b-vary-depth.test
*** PASS: test_cases/q2/2-4a-vary-depth.test
*** PASS: test_cases/q2/2-4b-vary-depth.test
*** PASS: test_cases/q2/2-one-ghost-3level.test
*** PASS: test_cases/q2/3-one-ghost-4level.test
*** PASS: test_cases/q2/4-two-ghosts-3level.test
*** PASS: test_cases/q2/5-two-ghosts-4level.test
*** PASS: test_cases/q2/6-tied-root.test
*** Running MinimaxAgent on smallClassic 1 time(s).
Pacman died! Score: -66
Average Score: -66.0
Scores:        -66.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running MinimaxAgent on smallClassic after 39 seconds.
*** Won 0 out of 1 games. Average score: -66.000000 ***
*** PASS: test_cases/q2/7-pacman-game.test

### Question q2: 4/4 ###


Finished at 21:00:51

Provisional grades
==================
Question q2: 4/4
------------------
Total: 4/4
```

3 Question 3: Alpha-Beta Agent

3.1 My performance Evaluation

On the same board as we tested MiniMax, if we use Alpha-beta:

```
-p AlphaBetaAgent -l minimaxClassic -a depth=4 -n 100 -q -f
Average Score: 142.18
nodes Expanded: 11040.25
Time Taken:108.772155
```

We can see on average 4000 nodes are expanded lesser than when we use Minimax algorithm.
One more comparison of Alpha-beta and MiniMax:

```
Command: -p AlphaBetaAgent -a depth=3 -l smallClassic -f -q
Pacman died! Score: -657
Average Score: -657.0
Scores:        -657.0
Win Rate:      0/1 (0.00)
Record:        Loss
nodes Expanded: 820934.0
Time Taken:93.785307

Command: -p MinimaxAgent -a depth=3 -l smallClassic -f -q
```

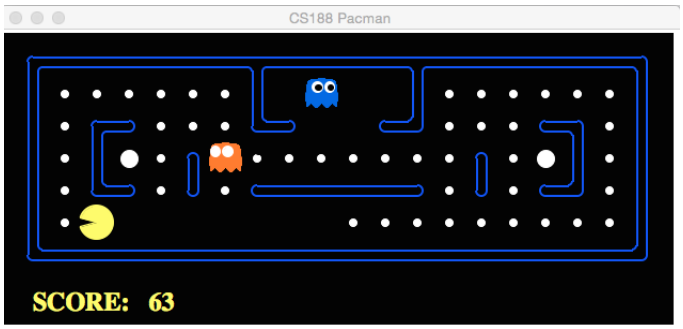

Figure 6: Board for alphabeta test 2: smallclassic

```
Pacman died! Score: -657
Average Score: -657.0
Scores:        -657.0
Win Rate:      0/1 (0.00)
Record:        Loss
nodes Expanded: 918147.0
Time Taken:105.0459
```

As we can see as we give it a more complex game, Alpha-beta gives the exact result but lesser number of nodes are expanded.

3.2 Autograder output

```
Question q3
===========

*** PASS: test_cases/q3/0-lecture-6-tree.test
*** PASS: test_cases/q3/0-small-tree.test
*** PASS: test_cases/q3/1-1-minmax.test
*** PASS: test_cases/q3/1-2-minmax.test
*** PASS: test_cases/q3/1-3-minmax.test
*** PASS: test_cases/q3/1-4-minmax.test
*** PASS: test_cases/q3/1-5-minmax.test
*** PASS: test_cases/q3/1-6-minmax.test
*** PASS: test_cases/q3/1-7-minmax.test
*** PASS: test_cases/q3/1-8-minmax.test
*** PASS: test_cases/q3/2-1a-vary-depth.test
*** PASS: test_cases/q3/2-1b-vary-depth.test
*** PASS: test_cases/q3/2-2a-vary-depth.test
*** PASS: test_cases/q3/2-2b-vary-depth.test
*** PASS: test_cases/q3/2-3a-vary-depth.test
*** PASS: test_cases/q3/2-3b-vary-depth.test
*** PASS: test_cases/q3/2-4a-vary-depth.test
*** PASS: test_cases/q3/2-4b-vary-depth.test
*** PASS: test_cases/q3/2-one-ghost-3level.test
*** PASS: test_cases/q3/3-one-ghost-4level.test
*** PASS: test_cases/q3/4-two-ghosts-3level.test
*** PASS: test_cases/q3/5-two-ghosts-4level.test
*** PASS: test_cases/q3/6-tied-root.test
*** Running AlphaBetaAgent on smallClassic 1 time(s).
Pacman died! Score: -66
Average Score: -66.0
Scores:        -66.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running AlphaBetaAgent on smallClassic after 10 seconds.
*** Won 0 out of 1 games. Average score: -66.000000 ***
*** PASS: test_cases/q3/7-pacman-game.test
```

```
### Question q3: 4/4 ###
```

4 conclusion

In this project, we compared the performance of 3 agents:

- Reflex agent: performs quite well for big boards where there are lot of escape routes and expands only a few nodes. but it is not very smart. It requires a complicated evaluation function to solve it properly with any reasonable performance.
- Minimax agent: has a lookahead involved, it performs quite poorly for depth of 2 and 3 but performs quite well for depths of 4 or more (63 %) even with a very poor evaluation function.
- AlphaBeta agent: optimizes the Minimax agent by pruning the search tree and gives much quicker performance compared to Minimax.