Yang_Jinxin_1168646_homework-4-Q1

brianyang1106@gmail.com

December 3, 2012

1 Description

Write a code to use reinforcement learning to generate a solution for the given wumpus world. The solution must present the best action to perform at each section of the board and calculate the expected reward of the optimal solution.

The output of program contains three parts:

- Utility (initial and final)
- Policy (show as arrows)
- Path

For the directory of this homework-4-Q1:

- *src*: *Wumpus* is a Java project file folder, you can just import this directory as "Existing Projects into Workspace" by Eclipse (both Windows and Linux works well) and run *src*→*wumpus*→*Wumpus.java* as "Java Application"
- *outputs*: the direct outputs from eclipse (the output is too long to paste in the report)
- *graphs*: all the screenshots

I will go though the logic of my program to explain the work I did.

2 Data Structure

After analyzing the problem, I extracted three basic data structures:

- Agent: current position, last block
- Block: basic blocks in the Wumpus world, coordinates, utility, name, policy (Pit, Wumpus, etc.)
- Wumpus (main function): initialize the Block and Agent, keep track of path, iteration process

3 Data Generation

Except *Pit*, *Gold*, *Wumpus* and *Start*, all the blocks should have a initial Utility, I generate them (0∼5000) using random function. Here is my initial utility:

```
Initial World

     Pit       .    Gold       .     Pit
       .       .       .       .       .
       .       .  Wumpus       .       .
     Pit       .       .       .       .
       .       .       .       .   Start
```

Figure 1. Initial World

```
Initial Utilities

  -500  2648  5000  3737  -500

  1782  1467  4791  4353  4441

  2564   890 -1000  3160  2151

  -500  1902  2522  2200  4721

  2479  4549  1596  1927     0
```

Figure 2. Initial Utility

4 Iteration

First, it will take several iterations to let the utilites converge:

- start from every block in the Wumpus world
- every block will find its best choice of next block it will move into
- update current block's utility with $U = U' + R$

After 7 iterations, it will converge:

```
after 7 iterations Utilities

  -500  4990  5000  4990  -500

  4970  4980  4990  4980  4970

  4960  4970 -1000  4970  4960

  -500  4960  4950  4960  4950

  4940  4950  4940  4950     0
```

Figure 3. Utility converge

Then starting from `world[4,4]`, agent will do following steps until reach the Gold block:

- check which block it will go next, which means determine the right direction
- agent moves forward (simply change the position of agent class)

once setting a new direction of agent, `printPolicy()` function will print the matrix with policy, you can see the arrow is changing the direction just like these:

```
Pit      .   Gold     .   Pit
  .      .      .     .     .
  .      . Wumpus     .     .
Pit      .      .     .     *
  .      .      .     .     .
```

Current position

Figure 4. Start

```
Pit      .   Gold     .   Pit
  .      .      .     .     .
  .      . Wumpus     .     .
Pit      .      .     .     ^
  .      .      .     .     *
```

Current Policy

Figure 5. 1 iteration

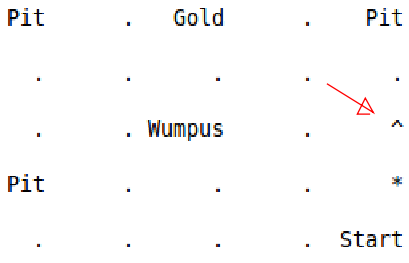

Figure 6. 2 iteration

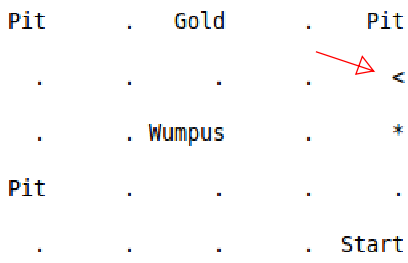

Figure 7. 3 iteration

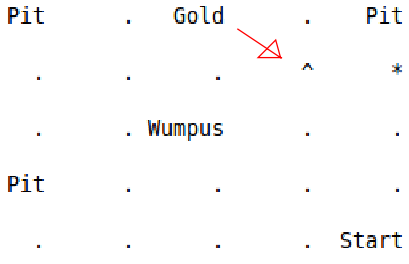

Figure 8. 4 iteration

Figure 9. 5 iteration

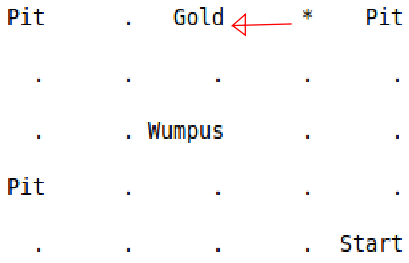

Figure 10. 6 iteration

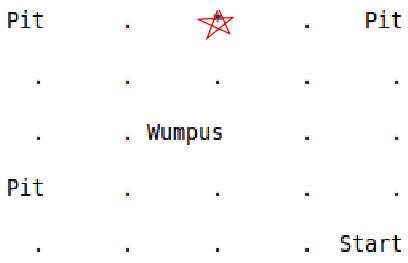

Figure 11. Catch the gold!

5 Final Results

After grabing the gold successfully, the final reward will be:

```
Final Utilities

  -500  4990  5000  4990  -500

  4970  4980  4990  4980  4970

  4960  4970 -1000  4970  4960

  -500  4960  4950  4960  4950

  4940  4950  4940  4950     0

Total Reward: 4840
```

Figure 12. Final Utility and Reward

Here is the optimal path output:

```
Path:

(4,4) -> (3,4) -> (2,4) -> (1,4) ->
(1,3) -> (0,3) -> (0,2) -> Gold!
```

Figure 13. Path

If your want to see the full output of one example, there is one in */outputs/output*.