CSC 350/500 – Project 4: Value Iteration

Due Date: Sunday, April 27, 2025

Introduction

In this project, you will implement use value iteration to investigate the game Pig (described below) and develop a policy for gameplay. I supply code in Java and Python that applies to Piglet, which we discussed in class; you can choose which language to use for your implementation with Pig.

This is a group project. A group can have up to 5 students. A group must be either all undergraduate students or all graduate students, no mixing.

File to Edit

You should only need to make a copy of PigletSolver.java or pigletsolver.py, renamed as PigSolver.java or pigsolver.py.

Getting Help

You are not alone! If you find yourself stuck on something, contact me for help sooner rather than later. I want these projects to be rewarding and instructional, not frustrating or demoralizing. But I do not know how or when to help unless you ask.

What to Submit

Zip all project files for your chosen language and your answers for written questions. One person per group should submit on Blackboard by the due date. Any submission after the due date is subject to the late penalties described in the syllabus.

How You Are Graded

Your grade is determined based on the implementation of the algorithms, answers to written questions, and your presentation of the final policy. The grading rubric for this assignment is on the final page of this document.

The Game of Pig

The object of the dice game Pig is to be the first player to reach 100 points. Each player's turn consists of repeatedly rolling a 6-sided die. After each roll, the player is faced with two choices: *roll* again, or *hold* (decline to roll again).

- If the player rolls a 1, the player scores nothing and it becomes the opponent's turn.
- If the player rolls a number of other than 1, the number is added to the player's *turn total* (the sum of the rolls during the turn) and the player's turn continues.
- If the player holds, the turn total is added to the player's score, and it becomes the opponent's turn.

For such a simple dice game, one might expect to see a simple strategy, such as in Blackjack (e.g. "stand on 17" under certain circumstances). As we shall see, this simple dice game yields a more complex and intriguing optimal policy.

Piglet: A Simplified Version

In class, we discussed Piglet, a simpler version of Pig that uses coins instead of dice. In Piglet, we play to 10 points. Each player's turn consists of repeatedly flipping a coin. After each flip, the player is faced with two choices: *flip* again, or *hold* (decline to flip again).

- If the player flips tails, the player scores nothing and it becomes the opponent's turn.
- If the player flips heads, add 1 to the player's *turn total* and the player's turn continues.
- If the player holds, the turn total is added to the player's score, and it becomes the opponent's turn.

Maximizing Probability of Winning in Piglet

In class, we defined the following math for maximizing our probability of winning a game of Piglet:

Let $P_{i,j,k}$ be the player's probability of winning if the player's score is *i*, the opponent's score is *j*, and the player's turn total is *k*. In the case where $i + k \geq 10$, we have $P_{i,j,k} = 1$ because the player can simply hold and win. In the general case where $0 \leq i, j < 10$ and $k < 10 - i$, the probability of a player who plays optimally (an *optimal player*) winning is:

$$P_{i,j,k} = \max\left(P_{i,j,k,flip}, P_{i,j,k,hold}\right),$$

where $P_{i,j,k,flip}$ and $P_{i,j,k,hold}$ are the probabilities of winning for flipping or holding, respectively. That is, the optimal player will always choose the action yielding the higher probability of winning. These probabilities are:

$$P_{i,j,k,flip} = \frac{1}{2}\left(\overbrace{(1 - P_{j,i,0})}^{\text{TAILS}} + \overbrace{(P_{i,j,k+1})}^{\text{HEADS}}\right)$$

$$P_{i,j,k,hold} = 1 - P_{j,i+k,0}$$

The probability of winning when flipping is $\frac{1}{2}$ times your probability of winning after getting heads, and $\frac{1}{2}$ times your probability of winning after getting tails. When you flip tails or hold, your probability of winning is the same as the probability that the other player will not win beginning with their next turn.

With these equations, we were able to apply value iteration to identify when it was optimal to flip and when it was optimal to hold in Piglet, taking whichever action gave us a higher probability of winning from any given state.

Maximizing Probability of Winning in Pig

We now need to apply the same approach to Pig as we did to Piglet. We can use the same definition of $P_{i,j,k}$ as we did before, except that now our goal is to play to 100 and instead of choosing between *flip* and *hold*, we now choose between *roll* and *hold*.

$$P_{i,j,k} = \max\left(P_{i,j,k,roll}, P_{i,j,k,hold}\right)$$

Just as we defined $P_{i,j,k,flip}$ and $P_{i,j,k,hold}$ for Piglet, you now need to define $P_{i,j,k,roll}$ and $P_{i,j,k,hold}$ for Pig. The questions below lead you through this process.

Assignment Questions

Question 1 – Definitions (3 points)

Complete the definitions for $P_{i,j,k,roll}$ and $P_{i,j,k,hold}$. Use the definitions from Piglet as a guide. In order to help you make progress on this project, I am happy to verify your answers before submission for this question if you bring them to me (in-person, e-mail, etc.).

You may write these out by hand or use some other tool to write the equations if you do not feel like fighting with Microsoft Word.

$$P_{i,j,k,roll} =$$

$$P_{i,j,k,hold} =$$

Question 2 – Probabilities by Hand (3 points)

To be sure you understand the game and the probability concepts above, solve for the following win probabilities. Note that each set below is dependent on the solution of the previous set. That is, you cannot solve (b) without solving (a) first, and you cannot solve (c) without solving all of (b). You will also need to have solved Question 1 first.

You may include your answers to this question wherever you answered Question 1.

(a) $P_{99,99,0}$
(b) $P_{98,99,0}$ and $P_{99,98,0}$
(c) $P_{97,99,0}$, $P_{97,99,2}$, and $P_{99,97,0}$

Question 3 – Solve Piglet Using Value Iteration (3 points)

I have given you PigletSolver.java, the code we used to apply the Value Iteration algorithm to Piglet in class. (I have also given you pigletsolver.py, a Python equivalent).

Modify this code so that it solves Pig instead. Change the filename to PigSolver.java or pigsolver.py. Hint: You will need to use your solution to Question 1.

Question 4 – Probability of Winning at the Start (3 points)

Use your code from Question 3 to answer these questions. Place these answers with your answers to Question 1 and Question 2.

(a) What is that probability that the first player wins assuming both players play optimally?

(b) On the first turn, what is the lowest turn total where the player should *hold*, rather than roll?

(c) If Player 1 scores no points on their first turn, what is the probability that the second player wins (assuming optimal play from both players)? When should the second player *hold* on their first turn?

Question 5 – Policy Output (6 points)

As given, PigletSolver prints a large table that indicates what turn total (what *k* value) a player should *hold* at for the *i* and *j* values at the start of their turn. For Piglet, this results in a reasonably-sized 10x10 table, but for Pig, we would get a 100x100 table which would be challenging to read clearly.

Summarize and/or visualize the policy **and** describe it qualitatively in your own words. Consider a table (easier to read than the 100x100) or a graph. Your summary and/or visualization may be generated by your code, or they may be generated after running your code, whichever you prefer.

Rubric

Question	3 points	2 points	1 point	0 points
Q1 – Definitions	Both equations are correct	Minor errors in one equation	Minor errors in both equations	Major errors in both equations
Q2 – Probabilities	Correct answers for all 3 parts	Correct answers for 2 out of 3 parts	Correct answer for 1 out of 3 parts	No correct answers
Q3 – Solve Pig Using Value Iteration	At most, minor bugs in code	Major bug in value iteration code	Attempted to make changes that address value iteration	No attempt
Q4 – Probabilities of Winning at Start	Correct answers for all 3 parts	Correct answers for 2 out of 3 parts	Correct answer for 1 out of 3 parts	No correct answers
Q5a – Summary or Visualization	Summary or visualization is clearly legible and makes the policy readily apparent	Summary or visualization makes the policy fairly apparent	Summary or visualization is difficult to follow	Not included, or no improvement on text-based table generated for Piglet
Q5b – Qualitative Description	Clear qualitative description of the policy	Description is good, but missing pieces	Description attempted, but inaccurate or missing major details	Not included