CPSC 231 – Fall 2017
Assignment 3: AI Training

Due: *6 Nov 2017, 9:00AM*

One hot topic in today's Computer Science research is Machine Learning. The main idea behind this area is that computers can be trained to learn from their mistakes. Training computers to learn from their mistakes produced programs like Watson `https://en.wikipedia.org/wiki/Watson_(computer)` or AlphaGo `https://en.wikipedia.org/wiki/AlphaGo`. In this assignment, you will be guided through the development of a strategy game; at the end, you will produce a program which is able to beat you.

The elements of the Python language that you need to use are variables, expressions, statements, functions, conditional expressions, and lists. In particular, you need to know that lists are *mutable*, so that, when a function takes some lists as parameters and modifies their elements inside the function, the changes propagate outside the function as well. While the assignment does not require you to use more than these, you may use additional features of Python as you like.

The Game of Nuts

In the game of nuts there is a pile of nuts on a board. There are two players. On their turn, each player picks up 1 to 3 nuts. The one who has to pick the final nut will be the loser.

This assignment is split into three parts, listed here in increasing order of difficulty.

1. Implementing the game as a two-player game.
2. Adding an AI that you can play against.
3. Adding an option for training the AI against another AI, and then playing again with the pitiful human.

Part one: Human versus Human

First, create a game where two players can play against each others. The two examples below demonstrate how the game should behave.

Example 1

```
Welcome to the game of nuts!
How many nuts are there on the table initially (10-100)? 10

There are 10 nuts on the board.
Player 1: How many nuts do you take (1-3)? 3

There are 7 nuts on the board.
Player 2: How many nuts do you take (1-3)? 3

There are 4 nuts on the board.
Player 1: How many nuts do you take (1-3)? 3

There is 1 nut on the board.
Player 2: How many nuts do you take (1-3)? 2
Player 2, you lose.
```

Example 2

```
Welcome to the game of nuts!
How many nuts are there on the table initially (10-100)? 500
Please enter a number between 10 and 100
How many nuts are there on the table initially (10-100)? 3
Please enter a number between 10 and 100
How many nuts are there on the table initially (10-100)? 50

There are 50 nuts on the board.
Player 1: How many nuts do you take (1-3)? 3

There are 47 nuts on the board.
Player 2: How many nuts do you take (1-3)? 55
Please enter a number between 1 and 3
Player 2: How many nuts do you take (1-3)? 3

There are 44 nuts on the board.
Player 1: How many nuts do you take (1-3)? 3
...

There is 1 nut on the board.
Player 1: How many nuts do you take (1-3)? 3
Player 1, you lose.
```

Implement a game with the functionality described above. You are promised that the users will always enter integers when prompted. If you want, you can also address the situation when users may not enter integers, but you don't have to (for that you likely need to use exceptions). You can use the code found here: `https://drive.google.com/file/d/0B-Fi5mZYtAddQmxCbHNieDBxek0/view`

Part two: Human versus AI

Playing against your friends is nice, but you're studying computer science for a reason. We can learn to do some pretty cool stuff. So can the computers – when taught. Let's create an AI for our game.

One way to create an AI for the game of nuts is to mathematically analyze the game and craft an AI based on the analysis. This is doable for this game, but is impossible for many games (like chess or go). Here we create a learning AI that is able to learn a good strategy for the game of nuts by playing the game against us. After this, the mathematical analysis is easier to do because we already know how to play the game optimally.

Consider the functionality of the AI using the following description:

- An AI has a number of hats, one hat for each possible amount of nuts on the table. Initially, each hat contains three balls that are numbered from 1 to 3.
- At every step of the game that the AI plays, the AI takes a random ball out of the hat that matches the amount of nuts currently on the board. When the AI takes a ball out of a hat, it places it next to the hat for waiting, reads the number on the ball, and takes the amount of nuts that the ball indicates.
- If the AI wins the game, it puts two balls of the type to each hat that has a ball next to it. Both balls have the same number. If the AI loses, it will throw away the balls that are next to the hats (note: A hat must always have at least one ball of each number, hence the last ball of a specific number cannot be thrown away and must be put back to the hat).
- As more and more games are played, there will be more balls that indicate a good number of nuts to take. This means that as balls are taken at random, it becomes more likely that the AI is able to play well.

Example

Let us consider an example where there are 10 nuts at the beginning. The hat contents for the AI are as in Figure 1.

hat	1	2	3	4	5	6	7	8	9	10
content	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3

Figure 1: The hat contents for the AI at beginning.

The game may proceed as follows.

1. Player takes 3 nuts, there are 7 nuts remaining.
2. AI randomly picks up ball 2 from hat 7. This means that the AI takes 2 nuts, and there are 5 nuts remaining.
3. Player takes 1 nut, there are 4 nuts remaining.
4. AI randomly picks up ball 3 from hat 4. This means that AI takes 3 nuts, and there is 1 nut remaining.
5. Player has to take the final nut and loses.

Now, the situation with the AI is as in Figure 2.

hat	1	2	3	4	5	6	7	8	9	10
content	1, 2, 3	1, 2, 3	1, 2, 3	1, 2	1, 2, 3	1, 2, 3	1, 3	1, 2, 3	1, 2, 3	1, 2, 3
beside				3			2			

Figure 2: The hat contents for the AI at the end of a game.

As the AI wins the game, it will put the balls that are next to the hats back to the hats with extra balls. The situation is now as depicted in Figure 3.

hat	1	2	3	4	5	6	7	8	9	10
content	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3, 3	1, 2, 3	1, 2, 3	1, 2, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3

Figure 3: The updated hat contents for the AI after winning a game.

Now the AI will more likely take 3 nuts in the case of 4 nuts remaining on the board, and 2 nuts in case there are 7 nuts remaining on the board.

Your task is to modify the human versus human version of the game so that the player can choose to play against an AI that works as described above. After each game, the AI will update the contents of its hats. The AI will play relatively randomly at first, but you will notice that it will start to learn a strategy as you play against it.

Note that you do not want to store the balls as separate numbers, all you need is to store how many balls are available for each possible move. In fact, you are required to store counts (otherwise your program might run out of memory). For instance, the initial starting position from Figure 1 can be stored as in Figure 4.

hat	1	2	3	4	5	6	7	8	9	10
content	1, 1, 1	1, 1, 1	1, 1, 1	1, 1, 1	1, 1, 1	1, 1, 1	1, 1, 1	1, 1, 1	1, 1, 1	1, 1, 1

Figure 4: The hat contents for the AI at beginning. Each move (1, 2 or 3) has the same chance of being selected.

Consequently, the situation from Figure 3 can be represented as in Figure 5.

hat	1	2	3	4	5	6	7	8	9	10
content	1, 1, 1	1, 1, 1	1, 1, 1	1, 1, 2	1, 1, 1	1, 1, 1	1, 2, 1	1, 1, 1	1, 1, 1	1, 1, 1

Figure 5: The updated hat contents for the AI after winning a game (using counters).

The following example displays how the program should behave after you have finished this step. Note that, in order to complete this part, you also need to add an option of playing again.

Example 3

```
Welcome to the game of nuts!
How many nuts are there on the table initially (10-100)? 10
Options:
 Play against a friend (1)
 Play against the computer (2)
Which option do you take (1-2)? 2

There are 10 nuts on the board.
Player 1: How many nuts do you take (1-3)? 3

There are 7 nuts on the board.
AI selects 2

There are 5 nuts on the board.
Player 1: How many nuts do you take (1-3)? 3

There are 2 nuts on the board.
AI selects 2
AI loses.
Play again (1 = yes, 0 = no)? 1

There are 10 nuts on the board.
Player 1: How many nuts do you take (1-3)? 1

There are 9 nuts on the board.
AI selects 1

There are 8 nuts on the board.
Player 1: How many nuts do you take (1-3)? 3

There are 5 nuts on the board.
AI selects 3

There are 2 nuts on the board.
Player 1: How many nuts do you take (1-3)? 2
You lose.
Play again (1 = yes, 0 = no)? 1

There are 10 nuts on the board.
Player 1: How many nuts do you take (1-3)? 3

There are 7 nuts on the board.
AI selects 2

There are 5 nuts on the board.
Player 1: How many nuts do you take (1-3)? 3

There are 2 nuts on the board.
AI selects 2
AI loses.
Play again (1 = yes, 0 = no)? 0
```

Part three: Human versus Trained AI

In the previous part we created an AI that is able to learn from playing against the player. As we play against it, we notice that it takes a considerable amount of time before the AI is able to perform against a human player. In this assignment, you need to modify the program so that the player can choose to play either against a naive AI or a pre-trained AI.

In order to pre-train an AI, you need to create a program that allows two AIs to battle against each others – say a 100000 times (after the training is working, try out different numbers as well!) – and after that the player will be set to play against the AI that is ready to battle the player. Note that the human player always plays first. To train the AIs, initialize two distinct AIs and let them play. When the game ends, both AIs have to update their tables according to the result of the game (the AI which lost needs to throw away balls, and the AI which won needs to add balls corresponding to the winning moves).

The following example shows how the game would work with the trained AI option.

Example 4

```
Welcome to the game of nuts!
How many nuts are there on the table initially (10-100)? 10
Options:
 Play against a friend (1)
 Play against the computer (2)
 Play against the trained computer (3)
Which option do you take (1-3)? 3
Training AI, please wait...

There are 10 nuts on the board.
Player 1: How many nuts do you take (1-3)? 3

There are 7 nuts on the board.
AI selects 2

There are 5 nuts on the board.
Player 1: How many nuts do you take (1-3)? 1

There are 4 nuts on the board.
AI selects 3

There is 1 nut on the board.
Player 1: How many nuts do you take (1-3)? 1
You lose.
Play again (1 = yes, 0 = no)? 1

There are 10 nuts on the board.
Player 1: How many nuts do you take (1-3)? 2

There are 8 nuts on the board.
AI selects 3

There are 5 nuts on the board.
Player 1: How many nuts do you take (1-3)? 2
```

```
There are 3 nuts on the board.
AI selects 2

There is 1 nut on the board.
Player 1: How many nuts do you take (1-3)? 1
You lose.
Play again (1 = yes, 0 = no)? 0
```

Hand In

1. Electronic copy of your program using D2L, as per your TAs instructions. The TA will run your program to test that it functions correctly.

Marking

The following incremental list will be used to asses your work. The total for this assignment is 10 points

1. Human versus human: 1.5 points
2. Human versus AI: 3 points
3. Human versus trained AI: 3.5 points
4. Code is clear and easy to read: 1 point
5. Code is modular: 1 point.

Additional things you may want to consider.

1. If the program produces run-time errors, your grade will be 0.
2. If the program is long and hard to understand, you may receive a lower score, even if the functionality is OK. If the code is incomprehensible, the TA will not grade it (thus, you get 0).
3. Other possibilities to get a score of 0 for your assignment: using no comments, using less than 4 functions, or using global variables.
4. Aim for about 200 lines of commented code, and about $10 - 15$ functions.
5. Comments are mandatory at the beginning of the program, at the beginning of each function (say what the parameters are and what the function is doing), and also before difficult blocks of code. At the beginning of the program, please write your name and which parts of the assignment you solved successfully (this will save the TAs some time).
6. To make sure the AI is running correctly, you need to print the tables that the AI is using (after the end of the game). Once you get the training running, please comment the print statements, **but mark clearly** that section in your code: the TAs need to uncomment your print statements to grade your code.
7. To create the table from Figure 4 you can use the following code. Here n is the number of nuts initially.

```
hats = []
    for i in range(n):
        row = [1, 1, 1]
        hats += [row]
```

Late work

After the deadline and up to 24hrs late: -2.5pts. After 24hrs and up to 48hrs late: -5pts. No assignment will be accepted beyond 48hrs after the deadline.

Collaboration

Although it can be helpful to you to discuss you program with other people, and that is a reasonable thing to do and a good way to learn, the work you hand in must ultimately be *your own work*. This is essential for you to benefit from the learning experience, and for the instructors and TAs to grade you fairly. Handing in work that is not your original work, but is represented as such, is *plagiarism* and academic misconduct. Penalties for academic misconduct are outlined in the university calendar.

Here are some tips to avoid plagiarism in your programming assignments.

1. Cite all sources of code that you hand in that are not your original work. You can put the citation into comments in your program. For example, if you find and use code found on a web site, include a comment that says, for example,

   `# the following code is from http://stackexchange.com`.

   Use the complete URL so that the marker can check the source.

2. Citing sources will avoid accusations of plagiarism and penalties for academic misconduct. However, you may still get a low grade if your work is not the product of your own efforts.

3. Discuss and share ideas with other programmers as much as you like, but make sure that when you write your code that it is your own. A good rule of thumb is to wait 20 minutes after talking with somebody before writing your code. If you find you are exchanging code by electronic means, writing code while sitting and discussing with a fellow student, typing what you see on another person's console, then you can be sure that your code is not substantially your own, and your sources must then be cited to avoid plagiarism.

4. We **will** be looking for plagiarism in your code, possibly using automated software designed for the task. For example, see *Measures of Software Similarity* (MOSS - `https://theory.stanford.edu/~aiken/moss/`).

Remember, if you are having trouble with an assignment, it is always better to go to your TA and/or instructor to get help, than it is to plagiarize.