Python checkers game which demonstrates machine learning

August 7, 2017

Abstract

This program demonstrates a checkers program written in python in which one player learns from each game it plays. It avoids moves which result in it losing one of its pieces or ultimately lose the game. Past moves are stored in a mysql database. These are then considered before it makes a move.

1 Objective

To demonstrate machine learning within a checkers game written in python

2 Functional Requirements

Random moves for one player The dumb player function describes a computer player which chooses its moves entirely at random.

Machine learning for player The smart player function describes a computer player which chooses its moves based on past experiences.

Ascii Board The game play is displayed via a simple Ascii board. This can prove where the moves are to.

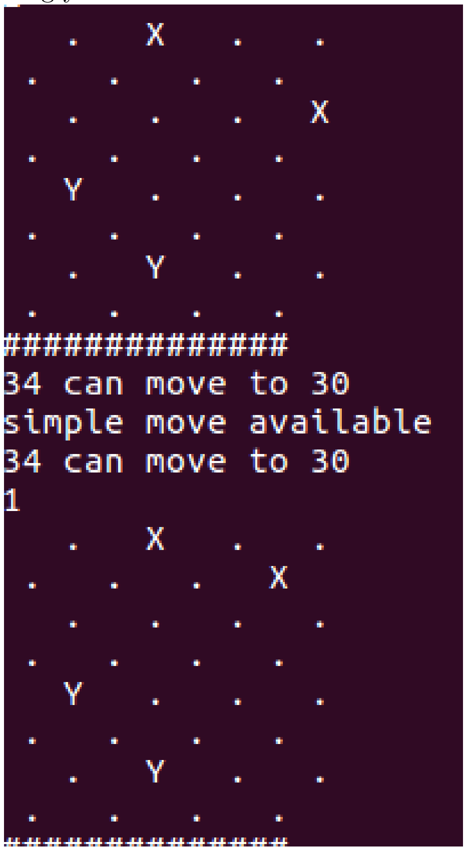
Learns to win This program can show that smart player learns to win against the dumb player.

3 Development Methodology

This program was developed using python 2.7. It was developed using a text editor. A link to a mysql database was used to store previous moves. It was not tested in an IDE. Some outputs were stored in a text file so the games could be manually reviewed. Print statements in addition to the board display were added so that variables could be viewed during execution. The program was tested several times at 150 games at a time to demonstrate that the smart player was better than the dumb player. A number of integers were adjusted to give a decent learning response. These were weights assigned to memory. A victory gave each move made by the smart player a +1 rating. A loss gave each move made by that player a -1 rating. Finally any move which resulted in the player losing a piece was given a -10 rating. A random number generator was made to choose move with heavier ratings towards good moves was developed. However the simpler choice of choosing the best move each time was finally chosen for better results.

4 User Interface

In order to run one game please enter "python checkers.py". When running you can now view the board.



```
   .   X   .   .
 .   .   .   .
   .   .   .   X
 .   .   .   .
   Y   .   .   .
 .   .   .   .
   .   Y   .   .
 .   .   .   .
##############
34 can move to 30
simple move available
34 can move to 30
1
   .   X   .   .
 .   .   .   X
   .   .   .   .
 .   .   .   .
   Y   .   .   .
 .   .   .   .
   .   Y   .   .
 .   .   .   .
##############
```

Sample output of the board.

At the end of each game the winner will be displayed.
In order to run a batch of 150 games please enter "python trigger-mulitple.py"
You may find it necessary to redirect the output to a text file in order to review later. The total amount of games won by each player will be stored in the mysql database.

5 Software Testing

This program was ran through numerous times to observe the results. Bugs were spotted and removed.
Occasionally the program does not exit when there is a stalemate. The parameters were adjusted to give a more complete learning algorithm.

6 Results and Conclusions

This program successfully demonstrates machine learning within a python program. Checkers is quite a difficult example for this because of the sheer amount of possible combinations of pieces and moves. This makes developing a simple lookup table of all these impossible to generate within 150 games. However a subset of the lookup table can be generated to give the smart player an edge over a dumb player.