Artificial Intelligence
Exercise 2

Marc Toussaint
Machine Learning & Robotics lab, U Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany

12. November 2018

1 Programmieraufgabe: Schach

Implementieren Sie ein Schach spielendes Programm. Der grundlegende Python code ist dafür in Ihren Repositories. Wir haben auch bereits die Grundstruktur eines UCT Algorithmus implementiert, so dass Sie nur die einzelnen Funktionen implementieren müssen. Die Implementierung von möglichen Erweiterungen steht Ihnen frei.

Evaluations-Funktion statt Random Rollouts: Letztes Jahr stellte sich heraus, dass der Erfolg naiver UCT Algorithmen bescheiden ist. Um die Baumsuche deutlich zu vereinfachen kann man die Evaluations-Funktion nutzen, um neue Blätter des Baums zu evaluieren (und den backup zu machen), statt eines random rollouts. Aber: Die Evaluations-Funktion ist deterministisch, und könnte die Suche fehlleiten. Als nächsten Schritt kann man deshalb sehr kurze random rollouts nehmen, die schon nach wenigen Schritten enden und mit der Evaluations-Funktion bewertet werden.

Ziel: Wir 'be-punkten' diese Aufgabe automatisiert indem wir den Schach-Agenten 10 mal gegen einen Random-Spieler antreten lassen. Ziel ist es nach Punkten zu gewinnern. (Sieg - 1 Punkt, Unentschieden - 0.5 Punkte, Niederlage - 0 Punkte).

Turnier: Außerdem planen wir alle Schach-Agenten in einem Turnier gegeneinander antreten zu lassen. Das Gewinnerteam darf sich über eine kleine Belohnung freuen!

Ihr Algorithmus soll auf folgendes Interface zugreifen:

```
class ChessPlayer(object):
    def __init__(self, board, player):
        # The game board is the board at the beginning, player is
        # either chess.WHITE or chess.BLACK.
        pass

    def inform_move(self, move):
        # after each move (also your own) this function is called to inform
        # the player of the move played (which can be a different one than you
        # chose, if you chose a illegal one.
        pass

    def get_next_move(self):
        # yields the move that you want to play next.
        pass
```

Sie können Ihre Implementierung testen mit

```
$ python2 interface.py --human --white --secs 2
```

um als Mensch gegen Ihren Spieler zu spielen. Oder mit

```
$ python2 interface.py --random --white --secs 2
```

um einen zufällig spielenden Spieler gegen ihr Programm antreten zu lassen.

2 Votieraufgabe: Bayes

a) Box 1 contains 8 apples and 4 oranges. Box 2 contains 10 apples and 2 oranges. Boxes are chosen with equal probability. What is the probability of choosing an apple? If an apple is chosen, what is the probability that it came from box 1?

b) The blue M&M was introduced in 1995. Before then, the color mix in a bag of plain M&Ms was: 30% Brown, 20% Yellow, 20% Red, 10% Green, 10% Orange, 10% Tan. Afterward it was: 24% Blue , 20% Green, 16% Orange, 14% Yellow, 13% Red, 13% Brown.

A friend of mine has two bags of M&Ms, and he tells me that one is from 1994 and one from 1996. He won't tell me which is which, but he gives me one M&M from each bag. One is yellow and one is green. What is the probability that the yellow M&M came from the 1994 bag?

c) The Monty Hall Problem: I have three boxes. In one I put a prize, and two are empty. I then mix up the boxes. You want to pick the box with the prize in it. You choose one box. I then open *another* one of the two remaining boxes and show that it is empty. I then give you the chance to change your choice of boxes—should you do so?

d) Given a joint probability $P(X, Y)$ over 2 binary random variables as the table

	Y=0	Y=1
X=0	.06	.24
X=1	.14	.56

What are $P(X)$ and $P(Y)$? Are X and Y independent?

3 Präsenzaufgabe: Bandits

Assume you have 3 bandits. You have already tested them a few times and received returns

- From bandit 1: 8 7 12 13 11 9
- From bandit 2: 8 12
- From bandit 3: 5 13

For the returns of each bandit separately, compute a) the mean return, the b) standard deviation of returns, and c) standard deviation of the mean estimator.

Which bandid would you choose next? (Distinguish cases: a) if you know this is the last chance to pull a bandit; b) if you will have many more trials thereafter.)