LSINF2275 Project I
« Data mining and decision making »
Markov Decision Processes

Professor: Marco Saerens marco.saerens@uclouvain.be
Address: Université catholique de Louvain
LSM & ICTEAM
Place des Doyens 1
B-1348 Louvain-la-Neuve
Belgique
Telephone: 010 47.92.46.
Fax : 010 47.83.24.
Teaching assistants: Bertrand Lebichot, Pierre Leleux

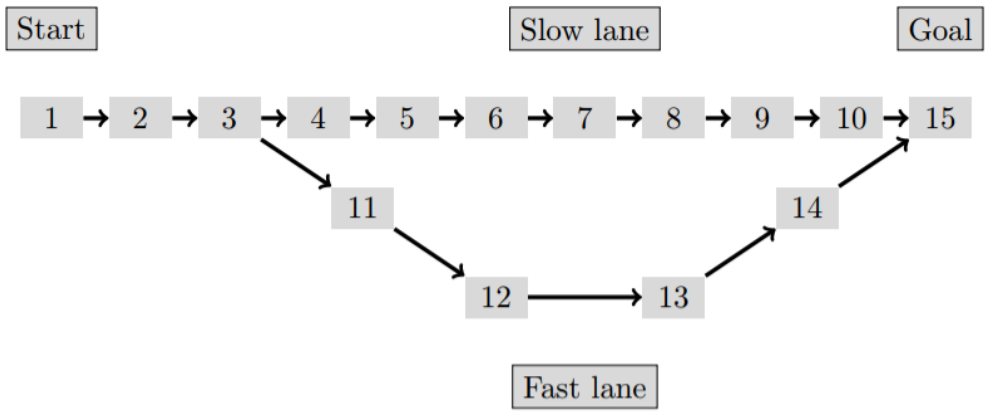
Objective

The objective of this project is to put into practice some of the techniques introduced in the data mining and decision making lectures. This will be done through the study of a practical case which requires the use of a software related to the statistical processing of data (such as *Matlab*). This work aims at applying algorithms solving « Markov decision processes » in the framework of a Snakes and Ladders game (i.e., a « Jeu de l'oie »).

Problem statement

The practical realization of the project will be carried out in groups of 2 students exactly. You are asked to determine the solution of the following problem.

Let us assume that you are playing a Snakes and Ladders game which is made of 15 squares. Square number 1 is the initial square (start) and square 15 is the winning square (arrival). If you reach square 15, you win. A schematic representation of this game is shown below:



To move forward, you can choose between **three different dices**:

- The « security » dice which can only take two possible values: 1 or 0. It allows you to move forward by 0 or 1 square, with a probability of 1/2. With that dice, you are invincible/immune, which means that you cannot fall into any trap.
- The « normal » dice, which allows the player to move by 0, 1 or 2 squares with a probability of 1/3. If you land on a trapped square using this dice, you have a 50% chance of triggering the trap.
- The « risky » dice which randomly allows the player to move by 0, 1, 2 or 3 squares, each of these actions having a uniform probability of 1/4. When you use that dice, you are always exposed to the traps if they are on the square you reach (100% trigger probability).

At square 3, there is a junction with two possible paths. If the player just passes through square 3 without stopping, he will continue to square 4 or beyond (5, 6, etc), just as if the other path did not exist. Conversely, if the player stops on square 3, he will have, on the following turn, an equal probability of going to square 4 along the normal path or to square 11 on the shortcut. For instance, for the security dice, if the player is on square 3 and the result is 0, the player stays on square 3 with probability 1. If the result is 1, he reaches square 4 or 11, each with probability 0.5. Both paths ultimately reach the final square.

You should also define « trap » squares on the game – feel free to try this. Recall that the player is only sensitive to trap nodes when drawing the normal or the risky dice. Then, if the player stops on one of these squares and triggers the trap, he will have to restart from the 1st square at the following turn or to go backward a given number of steps. For instance, if we turn node 7 into a trap square, we could, e.g., say that if the player stops on that square after drawing the risky dice, he will trigger the trap and has to go backward for 3 steps. In that case, he returns to square 4.

Moreover, according to the fact that we are working, in principle, with a Markov model of order one, if the player is currently on a trap and the value of the dice is zero, he reaches again the same square and the trap is re-activated.

We ask you to determine the optimal strategy, i.e., for each square, which dice should be played, to reach the goal square in a minimal number of turns, on average (the total cost is the number of turns to reach the goal square). You should determine this solution in two different scenarios:

- You should exactly stop on the arrival square to win. The game board is designed as a circle, which means that, if you pass the last square, you reach the 1st square again and go on.
- You win as soon as you have passed the arrival square (i.e. if you are **on** square 15 or further – see details later).

You should implement, in *Matlab, Octave, Python, Julia* or *R*, the following function:

```
[Expec, Dice] = markovDecision(List, Circle)
```

This function (here we used the Matlab conventions) launches the Markov Decision Process algorithm to determine the optimal strategy regarding the choice of the dice in the Snakes and Ladders game, using the « value-iteration » method.

The function should have as inputs/outputs:

In :

List: Vector of 15 values representing the 15 squares of the Snakes and Ladders game.

The value for each square is:

= 0 if it is an ordinary square.

= 1 if it is a « trap1 » square (go back to square 1)

= 2 if it is a « trap2 » square (go back three steps)

etc.

Circle: an indicator value (0 or 1) specifying if the player has to reach exactly the goal state to succeed (1) or still wins if he oversteps the final square (0).

Out :

Expec: Vector that contains the expectation of the cost when staring from each of the 15 squares of the game.

Dice: Vector that contains the choice of the dice to use for each of the 15 squares of the game, that is, the optimal strategy (1 for the « security » dice, 2 for the « normal » dice and 3 for the « risky » dice).

In addition, it is also interesting to compare this strategy with other (sub-optimal) strategies, e.g. the use of dice 1, dice 2 or dice 3 only, a mixed random strategy, or a purely random choice, etc. Launch/simulate an important number of games and compare empirically the performance of each of these strategies with the optimal strategy (or policy) obtained by value iteration. You should then report (i) the theoretical expected cost, obtained by value iteration, and compare it with the empirical average cost when playing (simulating) a large number of games with the optimal strategy, and (ii) compare the empirical average cost obtained by the optimal strategy with the other (sub-optimal) strategies mentioned before (dice 1 only, etc). You could test this with a few different configurations of traps, depending on your time.

Additional experiments

You could also define other kinds of traps: (3) « prison » squares; if the player stops on this square and triggers the trap, he must wait one turn before playing again, (4) « cost-sensitive » squares which provide an additional penalty, etc. Be creative!

It would also be interesting to investigate reinforcement learning and study to which extend this algorithm is competitive with value iteration (both in terms of computation time and average cost to reach the goal state), but this is left for the second project.

Report

Please do not forget to mention your affiliation on the cover page, together with your name (SINF, INFO, MAP, STAT, BIR, DATS, etc).

You are asked to write a report (in English) of maximum 5 pages (without the code which should be in a different file), preferably in latex. This report will contain

- a description of your game (traps, etc).

- a brief explanation of the method used to determine the optimal strategy regarding the choice of the dice, and a clear description of your Markov model.
- a short description of your implementation along with results/comparisons comparing the optimal strategy and the empirical results obtained for the Snakes and Ladders game with, or without, trap squares at relevant places. Please include graphics when relevant.
- a discussion of the simulation results comparing the different strategies. Try also to interpret intuitively the results and the optimal strategy.
- bibliographical references.

This report must be uploaded on April 01 (it's not a joke), 2018, before 23:55 together with the code (all files zipped together) on Moodle in the section « Assignments ». Do not forget to comment your code. This first project accounts for 3 points on 20. The weight of the second project will be 4 on 20. Thus, the final oral exam amounts to 13/20.

Good Work !