Foundations of machine Learning II

Project: Bandits and Wumpus

Flavie Vampouille
MSc in DS&BA
ESSEC Business School & CentralSupélec

Q1) Why is it interesting to give a negative reward for the default event?

It would avoid the agent staying in the same zone and encourage him to move further along, for if he retraces his steps or turns round in the same zone he will lose a point at each movement. Thus giving a negative reward for the default event should enable the solution to be reached more quickly.

Q2) What do you observe in terms of cumulative reward for the random agent?

The random agent just randomly take one of the 8 possible actions (UP, DOWN, LEFT, RIGHT, FLASH_UP, FLASH_DOWN, FLASH_LEFT, FLASH_RIGHT) without using the signals (smell and breeze) or the knowledge about the previous positions. The reward is basically -1 at each step until he randomly met the Wumpus, a pit or the treasure.

Q3) Propose and implement a new agent and compare the result with the cumulative reward of the random agent. Discuss about the robustness of your agent with the possible behaviors of the Wumpus.

The new agent I propose (WumpusNewAgent.py) react to the smell signal, which indicates that the Wumpus is close. When this signal is activated, the agent flash the light in one randomly chosen direction and after that he moves in the same direction if the Wumpus is not killed, or on a randomly chosen direction if the Wumpus is killed.
Dealing with the Wumpus allow a better reward than for the random agent, as the Wumpus is often either killed or just avoided. But the agent is still often wandering in the same restricted area, retracing is steps, and could either fall in the pit or find the treasure with the same probability. So the cumulative reward could again be high (find the treasure) or low (fall in the pit, or sometimes get killed by the Wumpus after using all flashlights).

Q4) What is the main drawback of the greedy policy?

The greedy agent I propose (WumpusGreedy.py) found the treasure after some try but the trajectories are not optimal (he loses points in unnecessary moves) and he often doesn't kill the Wumpus (which would allow more points). To work, the epsilon value has to be small (0.1 is a good one). So the agent learn to win the game but the cumulative reward could be better. He gets stuck in a local optimum and don't try to improve it.

Q5) UCB Agent

The UCB Agent (WumpusUCB.py) achieved the best cumulative reward possible. After some training (however the agent needs a lot of training, to be sure he achieved the best cumulative reward possible, something like 100 iterations is needed) he directly goes to kill the Wumpus and goes to the treasure after that, obtaining a reward of 100, which is the best possible.

Q) Q Learning Agent

I add a Q Learning method in the Greedy Agent (WumpusQLearning.py). It appears that the agent learns quite fast an optimum solution. However, the cumulative reward is not the best possible (100 in killing the Wumpus before catching the treasure) but is the faster one (98 in going directly to the treasure in two step).