Assignment 12

Machine Learning, Summer term 2013, Norman Hendrich

Solutions due by July 8

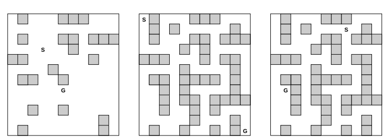
Exercise 12.1 (Incremental first-visit MC, 3 points) To calculate the updates, the first-visit MC policy evaluation algorithm requires the learner to remember all states visited and the collected Returns for all episodes. This requires a lot of memory.

Sketch an improved algorithm that uses an *incremental* update of the state-value function $V(s)$ for every state s visited in one episode using a given policy π. (The idea is similar to the incremental calculation of the average reward (return) described in part 1, slide 35).

Exercise 12.2 (Q-Learning, 3 points) Why is Q-learning considered an *off-policy* control method?

Exercise 12.3 (Labyrinth, programing, 16 points) Implement a Matlab or C/C++ program (or Java or another tool of your choice) to solve an episodic labyrinth task on gridworld with $n \times m$ cells, where the state marked 'S' is the start state s_0 and the state marked 'G' is the goal-state, and 'white' (space ' ') cells can be visited by the agent, while 'grey' ('X') cells mark the walls. The episode ends once the agent has reached the goal state.

The actions are $a \in \{left, right, up, down\}$, and the reward is -1 for every move. Moves outside the gridworld or into a wall don't change the position of the agent (but incur a reward of -1). You can either use the given labyrinth or use a design of your own:



(a) Depending on the number of walls, the labyrinth tasks can be easy or very difficult. It may help to start with a small-size problem (e.g. 5x5 cells) first. Is a random policy good for solving a labyrinth task?

(b) Design a program that estimates the state-value function $V(s)$ for a given labyrinth layout and given goal state. You can use any of the algorithms presented in the lecture.

(c) derive the greedy policy from the value function, and print/plot the trace of the learner through the labyrinth.