Introduction to Machine Learning (67577)

Problem Set 6 – Reinforcement Learning

Due: 6.7.2017

Submission guidelines

- The assignment is to be submitted via Moodle only.
- Files should be zipped to ex_N_First_Last.zip (In this case N stands for 6, First is your first name, Last is your last name. In English, of course).
- The .zip file size should not exceed 10Mb.
- All answers should be submitted as a single pdf file. We encourage you to typeset your answers (either LaTeX, Lyx or Word), but it is not mandatory. Note that if you choose to scan a handwritten solution, then answers written in incomprehensible handwriting or scanned in poor quality will be counted as wrong answers. Moreover, the scans should be integrated with the plots to a single coherent pdf file.
- All of your Python code files used in the practical question should be attached in the zip file. There should be one .py file, other then that there are no special requirements or guidelines for your code. Note that if your code doesn't run on the CS environment you will not get points for this question.

1 Shortest Path Problems

Let $G = (U, E)$ be a directed weighted graph with weights $w : E \to \mathbb{R}$ and $u \in U$ a source vertex, assume that all the cycles in the graph have positive weight. In this question we will show the connection between Value Iteration and the Bellman-Ford dynamic programming algorithm for shortest paths from u to all other vertices.

Algorithm 1 Bellman-Ford

For $v \in U$:
    Set $dist(v) = \infty$
$dist(u) = 0$
For i from 1 to $|U| - 1$:
    For $(l, v) \in E$:
        If $dist(v) > dist(l) + w(l, v)$
            Set $dist(v) = dist(l) + w(l, v)$

Algorithm 2 Value Iteration

Set $V_0 = 0$
While V has not converged
    $V_{t+1}(s) = \max_{a \in \mathcal{A}} \rho(s, a) + \mathbb{E}_{s' \sim \tau(s,a)}[V_t(s')] \quad \forall s \in \mathcal{S}$

The value function we will use for our MDP is the (non-discounted) infinite horizon value:

$$V_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} r(s_t, a_t) \mid s_0 = s\right]$$

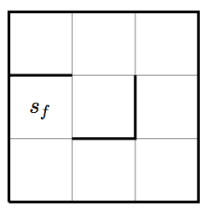
1. Provide a definition for $\mathcal{A}, \tau, \rho$ so that $V^*(s)$, the optimal value function of the corresponding MDP $\langle \mathcal{S} = U, \mathcal{A}, \tau, \rho \rangle$, equals minus the shortest weighted distance from u to s.
   Instructions:
   - Define actions and (deterministic) rewards so that only sequences of actions and states that give backward paths in G (i.e. paths that walk in opposite directions of edges in G) obtain rewards that are larger than $-\infty$. Such paths should get rewards that equal minus their weighted length. Also define an action that allows us to stay at state u with reward 0, in case the agent is at state u (u is then called an absorbing state).
   - Define τ as a deterministic transition function that changes the state according to the given action. Under this definition, letting the agent start at state s and act according to π, gives an infinite walk on the reversed graph.
   - Prove that under the above definitions, $V_\pi(s) \neq -\infty$ if and only if the infinite walk defined above arrives at node u and stays there indefinitely. Whenever $V_\pi(s) \neq -\infty$, prove that it is minus the

weight of the reversed path taken from s to u. Conclude that the optimal value function V^* satisfies the requirements we set.

2. Consider the Bellman-Ford algorithm and Value Iteration as applied to the MDP you defined in the previous article. Change the text of Value Iteration that's marked in red and define an iteration order over E and $\mathcal{S}$ for both algorithms, so that the iterations of both algorithms are equivalent (in terms of the calculations each of them preforms). Explain the equivalence.

2 Maze

Consider a model of the following maze:



Each tile is a state and the tile marked s_f is an absorbing state (i.e. all actions taken from s_f give reward 0 and stay at this state). The actions Right, Up, Left, Down take you to the appropriate next state, with success probability 0.8 if there's no wall, 0 if there is a wall. The action fails and we remain in the same state, with probability 0.2 if there's no wall, 1 if there is a wall. The reward is -1 if an action succeeds and -6 if it fails (hence the reward is a stochastic function and ρ is its expectation as defined in class).

- Open the file valueIteration.py found on the course website, read its documentation. Write a file named maze.py with a main function that constructs the model of the maze, then runs on it valueIteration from the provided file with $\gamma = 0.75$ until convergence. Submit your code.

- Plot a heatmap of the value function after $1, 2, 3, 4$ iterations and submit the plots with your answers in the pdf file. To produce the plots use the num_iters parameter of valueIteration and the command plt.imshow(V.reshape((3,3)),cmap='hot') from the pyplot package (from matplotlib import pyplot as plt).

3 Patience, Dear

We are given the following model:

\$\$\$					s_0	\$

The agent can move Left or Right and the actions always succeed. Moving to the extreme states resets the game to state s_0. Moving to the rightmost state gives reward 1, while moving to the leftmost state gives reward 6. Other actions reward nothing (cost 0).

- Write a file named patience.py with a main function that constructs the above model, then runs valueIteration on it until convergence with $\gamma = 0.75$. Submit your code.
- Run valueIteration with $\gamma = 0.5, 0.75, 0.85$, describe the optimal policy for each of these values (i.e. where does the optimal policy lead the agent at each state).
- Run valueIteration with γ ranging from 0.5 to 0.99 in strides of 0.01. Plot the value of s_0 in each of these 50 runs and submit your plots in the pdf.