4F13 Machine Learning: Coursework #4: Reinforcement Learning

Zoubin Ghahramani & Carl Edward Rasmussen

Due: 4pm Wednesday March 10th, 2010 to Rachel Fogg, room BNO-37

In this coursework, you will implement the *Value Iteration* algorithm for a simple grid world environment.

In order to get started, you need to familiarize yourself with a few provided matlab functions discussed below. At first sight, these functions may seem somewhat complicated - but actually they'll turn out to be helpful to you. First spend a few minutes familiarizing yourself with these functions.

Several matlab functions are provided for you to use. First, examine the `rl.m` script, which specifies a grid world in which you will be learning. We have provided a function `initGridworld.m` which takes the specification in `rl.m` and turns it into a `model` with the following fields: `stateCount` is the number of states in the grid world plus one extra fictitious state; `gamma` is the discount factor; `R` is a `stateCount` by 4 matrix of rewards; `P` is a `stateCount` by `stateCount` by 4 state transition probability matrix.

The grid world has a start state and an absorbing goal state. It also has 'bad' states which have a higher loss and 'obstructed' states, which you cannot visit. Note: in the grid world, a single extra absorbing fictitious state has been added, to which 'obstructed' states transition; this is just for implementational convenience.

There are also two tiny helper functions `rc2s.m` and `s2rc.m`, which convert between the 2 dimensional grid representation and the linear vector representation. Finally, there is a function `plotVP.m` to plot *values* and *policy* for the grid world.

Hand in: Code and plots. Your solution should not exceed 5 pages.

a) 60% : Implement *value iteration* in a function called `valueIteration`. Your function should take the `model` described above and an integer `maxIterations`, giving the maximum number of iterations to run the algorithm for. The function should return a vector of length `stateCount` of values. Initialize the values to be the maximum possible reward for each state before starting value iteration. Your function should return when the maximum number of iterations have been done, or when the values have converged (or change by only a tiny amount). Hand in the code.

b) 20% : Run your value iteration function on the example `model` from `rl.m`. How many iterations does it take to converge? Using the `imagesc` function, show the initial value function, an intermediate function and the final value function on a grid. Explain qualitatively the plots.

c) 20% : Write a function `getPolicy`, which takes the values computed by `valueIteration` and the `model` and returns the optimal policy as a vector of length `stateCount` of actions. Visualize the policy on a grid (optionally using the provided `plotVP.m` function.

d) 0% (optional) : specify a different grid world by modifying the parameter specification in `rl.m`. Test your `valueIteration` algorithm.