Aston University
Machine Learning

Reinforcement Learning (1)

Learning Outcomes:
In the current unit, we studied Markov Decision Processes and methods for evaluating policies on them. In this lab, we will put our knowledge into practice using Python.

Instructions:
Attached to the lab task on Blackboard is a file grid_world.py. This is an implementation of the grid world MDP discussed in lectures. You will be working with the code during this lab.

Task 0:
To get started, familiarise yourself with the code in grid_world.py. You don't need to understand any of the implementation details, but should be familiar with:

- **`Move`**: You will use this enum to specify a move in grid world. For instance, if you wanted to specify that an agent should move up in a given position, you would recommend the move **`Move.up`**.
- **`random_pos`**: This returns a (valid) random position in grid world. A position is simply a two element array, with the first element being the x coordinate in the grid and the second being the y coordinate. As in the lecture slides, positions start at 1 (so the bottom right position would be `[4,1]`).
- **`run_policy`**: Given a starting position and a policy, run policy generates a sample sequence of positions and rewards from that starting position under that policy. It returns two values: the first being an array of the positions visited and the second being the corresponding reward received in those positions.

To generate a sequence, you will also need to define a policy. In this implementation, a policy is simply a function which takes a position as argument and returns a **`Move`** enum containing the policy's recommended move in that position. I have defined a sample policy (**`sample_policy`**) as an example. The sample policy is suboptimal. Based on what was discussed in lectures, can you implement the optimal policy?

Task 1:
Write a function **`calculate_discounted_utility`** which takes two parameters: a sequence of rewards and a discount factor γ and returns the discounted utility of the reward sequence under that discount factor.

Test your function by testing it with some sequences and values of γ which you have calculated by hand.

Task 2:
Use Python to implement **temporal difference learning** to estimate the expected utility of states of grid world under a given policy and discount factor.

You will need to:

- starting at a random state, sample a sequence for the given policy,
- use the update equation discussed in lectures to set and update the expected utility of each state encountered in the sequence based on observed transitions and rewards,
- iterate the above until differences between utility estimates become small.

Validate your implementation by running it against the optimal policy for grid world with a discount factor of 1. Does it match the values given in the lecture slides?