Lab - Week 4

Markov Decision Problems

This tutorial is intended for you to familiarise yourself with MDPs. We will work in detail through an MDP problem that you know from the seminar and you will learn how to use an MDP library to solve it.

1. Installing an MDP Solver

For this lab you will have to use an external MDP solver to perform value iteration. If you are programming in Python, the recommended choice is *PyMDPtoolbox*. You will only have to use value-iteration from this solver. *Warning*: *q-learning appears to be broken in the current version of PyMDP-toolbox. You will not have to use q-learning in this tutorial.* There are other libraries available but the advantage of using this solver is that its interface is very simple and perfectly aligned with how MDPs are discussed in the lecture and in the book: It implements the mathematical definition directly without any additional coding complications.

Tasks

1. Install PyMDPtoolbox on your computer. There are various ways to do this but if you have multiple Python installations on your machine (as is common) you must ensure that the toolbox is actually installed for the Python version on your system used by Anaconda. This is best done from the command line using the command
   ```
   python3 -m pip install pymdptoolbox
   ```
2. Test that your installation works correctly. A simple way is to run the example as outlined on the PyMDPtoolbox home page from within an Anaconda Jupyter notebook. If the result is correct, your installation should work ok.

```
import mdptoolbox.example
P, R = mdptoolbox.example.forest()
vi = mdptoolbox.mdp.ValueIteration(P, R, 0.9)
vi.run()
vi.policy # result is (0, 0, 0)
```

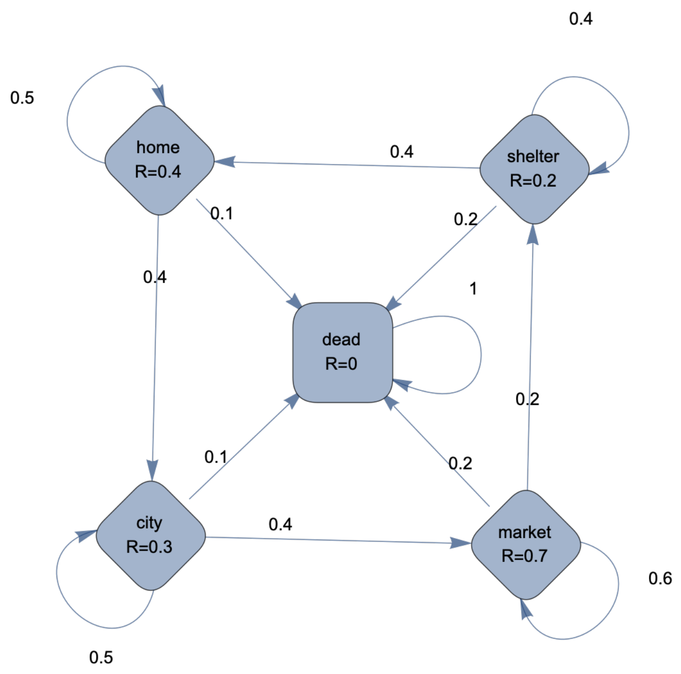
2. Defining and Solving an MDP

Review the “Seven Lives of Cats” example from the seminar in Week 3.

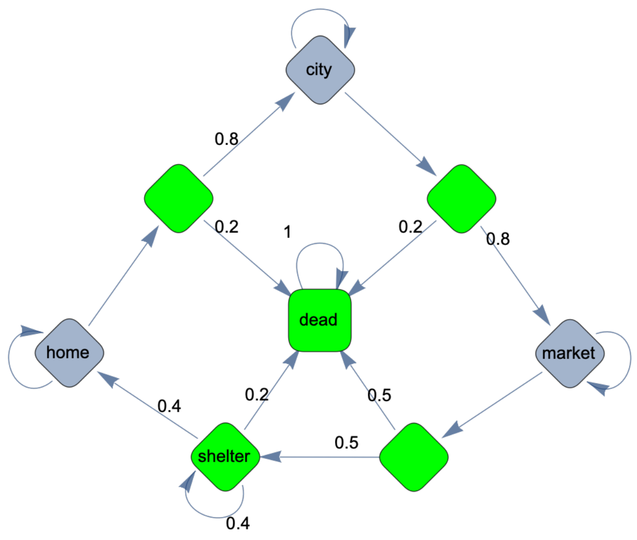
Tasks

1. Before we turn our attention to the MDP, let us review the simple **Markov Chain**. Define the states for the cat problem as depicted below. Ignore the rewards in the nodes for now.

Out[ ]=



2. Define the Markov Chain transition matrix (where the cat does not make any decisions yet, we are just capturing the statistical behaviour of our assumed cats). Write this Matrix down in Python (or Mathematica).
3. We are now turning our attention to the **Markov Reward Process**. Write down the reward matrix for the rewards given in the diagram in Python or Mathematica.
4. Determine the values for the example by solving the Bellman expectation equation for this problem in closed form. If you are using Python, you will have to use the Numpy functions `matmul`, `identity` and `linalg.inv`.
5. We are now switching to a **Markov Decision Process**. The cat has two actions available and these are the same in each of the states “Home”, “City”, and “Market”. It can either “stay” (ie. remain in the current state) or “run” (ie. exit the state). When it runs, it has no control over the target state reached. This happens probabilistically according to the transitions in the diagram (see Seminar Week 3).



6. Points to discuss:
   - 6.1. How to capture the fact that the cat cannot actively influence the situation when it is in a shelter (whether it is adopted or not is an external decision, the cat does not have any real influence on this).
   - 6.2. how to handle the fact that there is now return from being dead for a normal cat (more on this below) given that our solver can only handle infinite episodes .
7. Write down the MDP transition matrix, which in contrast to the Markov Chain transition matrix above must also capture the influence of actions.
8. Write down the MDP reward matrix, which must also capture the actions. Note that the cat should only receive the reward depicted in the diagram for each state when it stays in this state, not when it transitions to a new state. This models the fact that we understand the rewards as the benefit of "being in a state" (rather than directly as the reward of an action).
9. Discuss how you would model the fact that cats are said to have seven lives. Importantly, they have exactly 7. Not 1, not 2, not 8, not infinitely many! How can you model the above decision problem using an MDP under this assumption?

If Value Iteration has not yet been discussed in the seminar preceding your lab, delay solving the last two questions until after this has happened. In this case, solve an alternative task first: perform the first step of an "iterative policy evaluation" for the above problem. To do so you first fix a policy (arbitrarily), initialise the state values arbitrarily, and then interpret the Bellman state value equation as an update equation. If you iterate this update often enough, it will converge on the true state value. Your tutor will explain the necessary steps in detail.

10. Perform the first iteration of value iteration for this problem.
11. Use the PyMDPtoolbox that you have installed before to compute the optimal policy for the cat by value iteration. Your tutors can help you with the coding but first attempt to work out how to do this by using the documentation at https://pymdptoolbox.readthedocs.io/en/latest/api/mdp.html#mdptoolbox.mdp.ValueIteration