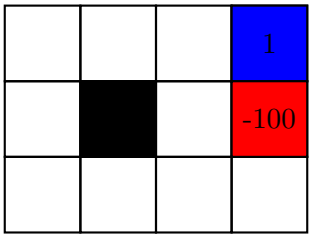
COMP3702/7702 Artificial Intelligence
Semester 2, 2020
Tutorial 9 - Sample Solutions

Example domain

Consider the gridworld below:



States in this environment are the positions on the tiles. The world is bounded by a boundary wall, and there is one obstacle, at (1,1) (using python or zero indexing starting from the bottom left corner). In addition, there are two terminal states, indicated by the coloured squares.

Terminal states: In practice, we would say that the two coloured tiles are *terminal states* of the MDP. For mathematical convenience, we often prefer to deal with an infinite horizon MDP (see more below). To convert this model to an infinite horizon MDP, we will add an extra pseudo-state to the list of MDP states called `'EXIT_STATE'`, which is absorbing (the agent cannot leave it irrespective of its action, ie. $T(EXIT\_STATE, a, EXIT\_STATE) = 1$ for all a).

Actions and Transitions: In this world, an agent can in general choose to move in four directions — *up*, *down*, *left* and *right*, which we might sometimes refer to as `^`, `v`,`<` and `>`, respectively, as in the LaserTank environment. However, the agent moves successfully with only $p = 0.8$, and moves perpendicular to its chosen direction with $p = 0.1$ in each perpendicular direction. If it hits a wall or obstacle, the agent stays where it is. Denote this action `'.'`. In addition, once the agent arrives on a coloured square with a value, it has only one special action available to it; that is, to *exit* the environment and move to the exited state E; denote this action `'e'`.

Rewards: The values stated on the coloured squares are the reward for *exiting* the square and the environment, so the reward is not repeatedly earned; that is, $R([3, 2], exit) = 1$, and $R([3, 1], exit) = -100$. All other states have 0 reward.

Discount factor: $\gamma = 0.9$.

Exercise 9.1

Define and code the transition function for this problem.

a) : What is the probability of arriving in state (1,0) from each a and s, i.e what is $P((1,0)|a,s) \; \forall \; (a,s)$?

$P((1,0)|right,(0,0)) = 0.8$
$P((1,0)|up,(0,0)) = 0.1$
$P((1,0)|down,(0,0)) = 0.1$

$P((1,0)|left,[2,0]) = 0.8$
$P((1,0)|up,[2,0]) = 0.1$
$P((1,0)|down,[2,0]) = 0.1$

$P((1,0)|up,(1,0)) = 0.8$
$P((1,0)|down,(1,0)) = 0.8$
$P((1,0)|left,(1,0)) = 0.2$
$P((1,0)|right,(1,0)) = 0.2$

All other transition probabilities are zero.

b) : What is the one-step probability of arriving in each state s' when starting from (0,0) for each a, i.e what is $P(s'|a,(0,0)) \; \forall \; a,s'$?

$P((1,0)|right,(0,0)) = 0.8$
$P((0,1)|right,(0,0)) = P((0,0)|right,(0,0)) = 0.1$

$P((0,1)|up,(0,0)) = 0.8$
$P((1,0)|up,(0,0)) = P((0,0)|up,(0,0)) = 0.1$

$P((0,0)|left,(0,0)) = 0.9$
$P((0,1)|left,(0,0)) = 0.1$

$P((0,0)|down,(0,0)) = 0.9$
$P((1,0)|down,(0,0)) = 0.1$

All other transition probabilities are zero.

c) : Write a function that can compute the probabilities of arriving in any state $P(s',a,s)$ from an initial state s after taking action a. Rather than hard-coding the probabilities, check if each action moves to a neighbour state or if it results in any collisions with the boundary or the obstacle. Then use this information to compute the transition probabilities. (*Hint:* You may find some of the tricks for 8-puzzle from Tutorial 2 useful. You may also wish to parameterise this function so that p can be varied too (useful for the assignment.)

Notes:

- Set up an mdp class with mdp.mapinfo and mdp.actions sets, among other things
- In addition to directions, the MDP class needs an exited `'EXIT_STATE'`, move for the terminal states, and a no-move action, `'.'`, i.e.for wall collisions and the pseudo-state 'existed'.

A starter file is provided on BB to get you going, called `GridWorld_starter.py`

d) : Write a new function to processes the gridworld and store all of the transition probabilities, using the transition-probability computing function you have developed above. The aim is to have functions that can be used on an arbitrary gridworld (i.e. do not hard-code your function just for this problem instance!).

Exercise 9.2

Use the starter code provided, and see see `http://aima.cs.berkeley.edu/python/mdp.html` for hints, but note that our gridworld has some differences and additional elements compared to AIMAs:

- The 'exited' state `'EXIT_STATE'`
- `'EXIT_STATE'`, move in the terminal states, and
- a no-move action, `'.'`, i.e. for wall collisions and the pseudo-state `EXIT_STATE`.

Exercise 9.3

a) Let $P^\pi \in R^{|S|\times|S|}$ be a matrix containing probabilities for each transition under the policy π, where:

$$P^\pi_{ij} = P(s_{t+1} = j \mid s_t = i, a_t = \pi(s_t))$$

What is the size of P^π in this gridworld, when the special pseudo-state `EXIT_STATE` is included?

If you exclude the obstacle at (1,1) as an (unreachable) state, but include `EXIT_STATE`, then $dim(P^\pi) = 12 \times 12$. (Also, you could include the obstacle, so that $dim(P^\pi) = 13 \times 13$, but the transition probability to (1,1) is 0 from all states.)

b) Set the policy to move *right* everywhere, $\pi(s) = \text{RIGHT}\ \forall\ s \in S$. Calculate the row of P^{RIGHT} corresponding to an initial state at the bottom left corner, (0,0) of the gridworld.

Let the state indices be ordered:
$((0,0),(1,0),(2,0),(3,0),(0,1),(2,1),(3,1),(0,2),(1,2),(2,2),(3,2),$`EXIT_STATE`$)$

The row is: $P^{\text{RIGHT}}_{((0,0),j)} = \begin{bmatrix}0.1 & 0.8 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$

b) Write a function that calculate the row of P^{RIGHT} corresponding to an initial state at the bottom left corner, (0,0) of the gridworld for any action or deterministic $\pi((0,0))$.

Reuse code answers from Exercise 9.1 to compute these values.

c) Turn this into a function that computes P^π for any deterministic π. Note that

$$P([3,1], a, \text{EXIT\_STATE}) = P([3,2], a, \text{EXIT\_STATE}) = P(\text{EXIT\_STATE}, a, \text{EXIT\_STATE}) = 1$$

for all a, so the rows of P^π for these states s are all zeros except for a 1 corresponding to the transition to $s' = \text{EXIT\_STATE}$.

As above, reuse code answers from Exercise 9.1 to compute these values.

d) Compute

$$V^{\text{RIGHT}} = (I - \gamma P^{\text{RIGHT}})^{-1} r.$$

To do this, define r as the reward for landing on a square, which holds because $R(s, a, s') = R(s)$ for the red and blue squares at [3,1] and [3,2], respectively. (*Hint*: Rather than explicitly computing the matrix inverse, you may want to use `numpy.linalg.solve()` for large $|S|$ as it implements the LU decomposition to solve for V^π using forward and backward substitution, so can result in a big speed-up.)

Using the same indexing as above: $r = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & -100 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}^T$ (i.e. transposed to a column vector).

Using your pre-defined P^π, code to solve for v is:

```
import numpy as np
# you should already have these defined
size_S = len(P)
discount_factor = 0.9

A = np.identity(size_S) - discount_factor*np.array(P)
r = np.array([0.0 for s in range(size_S)])
r[7]  = -100
r[11] = 1
v = np.linalg.solve(A, r)
```

e) Using the policy evaluation above, implement PI for this problem.

Use the starter code provided, and see `http://aima.cs.berkeley.edu/python/mdp.html` for hints, but again note that our gridworld has some differences and additional elements compared to AIMAs.

f) How many iterations does PI take to converge?

If it takes more than 5 iterations, you are doing something wrong!

How long does each iteration take? It depends on the machine.

Exercise 9.4

See the material in:

- http://www.diego-perez.net/papers/MCTSSurvey.pdf and
- https://jeffbradberry.com/posts/2015/09/intro-to-monte-carlo-tree-search/