Solving Gridworld with policy evaluation

Paulo Henrique dos Anjos
paulodosanjos@usp.br

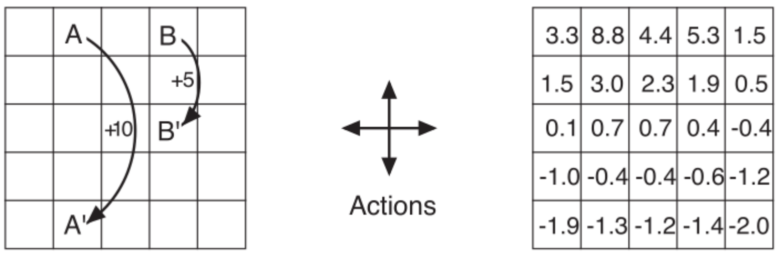
June 17, 2023

1 Introduction to the problem

We use the simple "GridWorld" problem as a model to test our policy evaluation algorithm.

The description of the problem is despicted bellow:

Problem Description. The left figure uses a rectangular grid to illustrate value functions for a simple finite MDP. The cells of the grid correspond to the states of the environment. At each cell, four actions are possible: north, south, east, and west, which deterministically cause the agent to move one cell in the respective direction on the grid. Actions that would take the agent off the grid leave its location unchanged, but also result in a reward of 1. Other actions result in a reward of 0, except those that move the agent out of the special states A and B. From state A, all four actions yield a reward of +10 and take the agent to A'. From state B, all actions yield a reward of +5 and take the agent to B' . Suppose the agent selects all four actions with equal probability in all states. The figure on the right shows the value function, v_π, for this policy, for the discounted reward case with $\gamma = 0.9$. Notice the negative values near the lower edge; these are the result of the high probability of hitting the edge of the grid there under the random policy. State A is the best state to be in under this policy, but its expected return is less than 10, its immediate reward, because from A the agent is taken to A' , from which it is likely to run into the edge of the grid. State B, on the other hand, is valued more than 5, its immediate reward, because from B the agent is taken to B' , which has a positive value. From B' the expected penalty (negative reward) for possibly running into an edge is more than compensated for by the expected gain for possibly stumbling onto A or B.



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

Figure 1: States and value funcion for the gridworld

Our objective is to get the same value function shown above using the policy evaluation algorithm.

2 The algorithm

The policy evaluation algorithm in its simplest form is shown bellow:

```
POLICY-EVALUATION($\pi$)
1  // initialize a array $V[0\,..\,|S|-1] = 0$ for all $s \in \mathcal{S}$
2  repeat
3      $\Delta = 0$
4      for each $s \in \mathcal{S}$
5          $v = V[s]$
6          $V[s] = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma V[s']]$
7          $\Delta = \max(\Delta, |v - V[s]|)$
8
9  until $\Delta < \theta$ (a small positive number)
10 return $V \approx v_\pi$
```

For deterministic policies, we can drop $\sum_a \pi(a|s)$ part on line 6 of POLICY-EVALUATION.

In most cases, computing the $p(s', r|s, a)$ matrix is unfeasible (not in gridworld though). Instead, we "break" the process in two steps, and use the espected reward to simplify computations:

$$v_\pi(s) = \sum_a \pi(a|s)\{r(s,a) + \gamma \sum_{s'} p(s'|s,a) v_\pi(s')$$

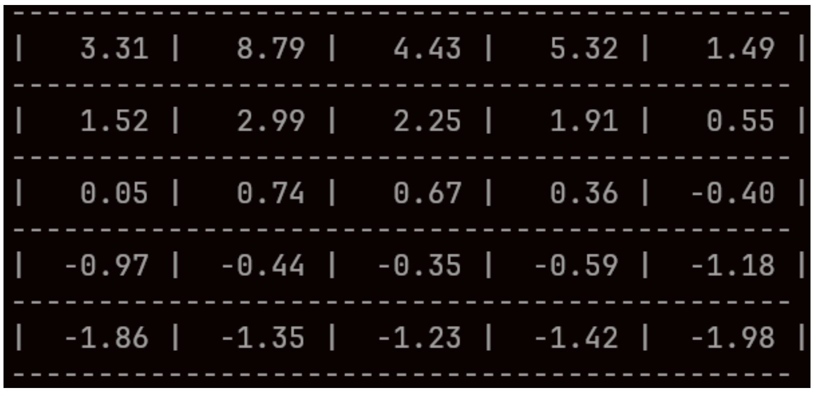
In this form, we are left with a 2D matrix (to store $r(s, a)$ values) and a 3D matrix (to store $p(s', s, a)$ values), which simplifies the preprocessing drastically.

2.1 Implementation

We implemented the algorithm in Python. Check the code in https://github.com/paulohdosanjos/IC/sutton/cap4/gridworld.

2.2 Results

Running main.py, we get the following results:



```
-----------------------------------------------
|   3.31 |   8.79 |   4.43 |   5.32 |   1.49 |
-----------------------------------------------
|   1.52 |   2.99 |   2.25 |   1.91 |   0.55 |
-----------------------------------------------
|   0.05 |   0.74 |   0.67 |   0.36 |  -0.40 |
-----------------------------------------------
|  -0.97 |  -0.44 |  -0.35 |  -0.59 |  -1.18 |
-----------------------------------------------
|  -1.86 |  -1.35 |  -1.23 |  -1.42 |  -1.98 |
-----------------------------------------------
```

Figure 2: Value function for the gridworld problem