TP :Solving MDPs (Planning)

Urtzi AYESTA and Matthieu JONCKHEERE

You may work in pairs, but please submit an individual report answering all the questions, and always explaining your answers. Please do not write only "numbers", explain briefly every step, and provide the code you used. In your report, indicate the name of the person you have worked with.

Submit your report in a single PDF. We will set a deadline during the TP.

Let us consider the following "Grid World" topology.

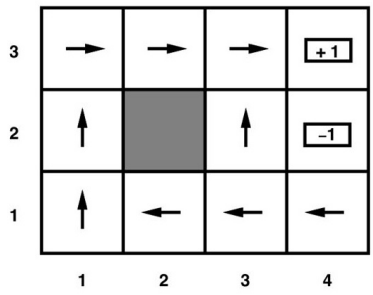

Figure 1: A simple grid world. The reward is -0.04 in every non terminal state. Let us assume a discounting factor $\gamma = 0.9$.

Policy Matrix: Each action is represented by a number : Action (Up) is represented by 0, (Right) by 1, (Down) by 2 and, finally, (Left) by 3. We shall further assume than the grid world is somehow slippery, meaning that a given action may produce a different direction with some (small) probability. For instance, the action 0 (Up) might result in the direction Right with probability 0.1.

Matrix encoding actions and directions: Column 0 represents direction Up, Column 1 represents direction Right, Column 2 represents direction Down and Column 3 represents direction Left. Notice that when bumping with walls, the state in that direction remains the same.

$$\begin{pmatrix} 0.7 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.7 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.7 & 0.1 \\ 0.1 & .10 & 0.1 & 0.7 \end{pmatrix}$$

1 Questions:

Performance Prediction

In this section we are going to assume the following policy

1. Assume the random policy, that is, the policy that takes every possible action with probability 1/4. Compute its value function by solving

$$V = (I - \gamma P)^{-1} R.$$

Since there are 11 possible states in the problem, the vectors R and V have have length 11, and the matrix P has dimension 11x11. There are two absorbing states, i.e., they are visited once, and their respective reward (+1 or -1) is only accrued once. To model this, you can simply put all 0's in all the elements of the respective two lines.

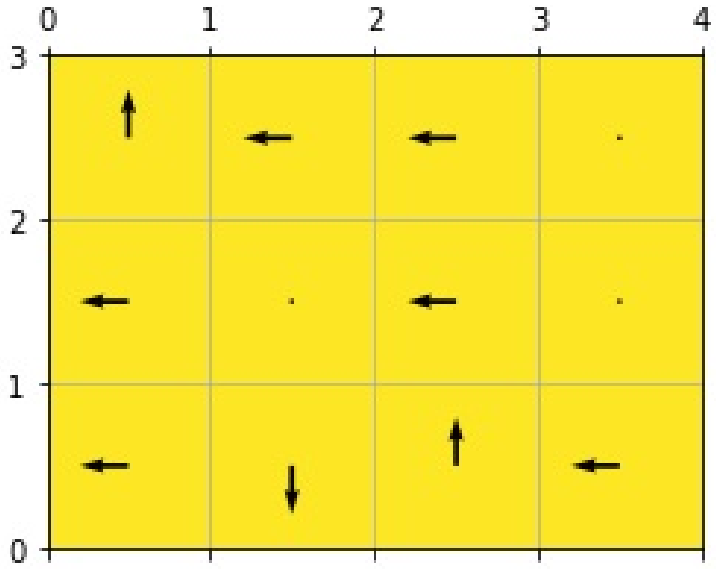

Figure 2: Policy to be considered during the "prediction" exercises.

2. Evaluate now the policy using Iterative Policy Evaluation (lecture 2, 2nd part, slides 11/35), and verify that the algorithm converges to the result obtained in 1. To stop iterating, you can take as a criterion that the difference between two iterations must be smaller than some small δ.

   Due to the contraction principle, the initial vector can be arbitrary.

3. To verify that the Bellman operator is a contraction, take two initial vectors, and calculate the max of their differences. Then, apply the iterative policy evaluation to these 2 vectors as done in the previous item, and plot the maximum of their differences as you keep iterating. Observe what happens with the difference as you iterate, and explain it.

Optimization

1. Write down the Bellman equation that characterizes the optimal policy.

2. Solve numerically the optimal value function by Value Iteration Algorithm (lecture 2, 2nd part, slides 15/35). Verify that the solution you obtain satisfies the Bellman equation.

3. Explain how you can infer the optimal action in every state from the optimal value function $V^*(s)$. Represent in a 2D matrix the optimal policy.

4. Compare the performances obtained with the random policy and the optimal one, how can you conclude that the optimal policy performs better ?

5. **Policy Iteration I:** We are now going to calculate the optimal policy using Policy Iteration (lecture 2, 2nd part, slides 23/35 and 24/35). You can start with the random policy for which you calculated its performance in the **Performance Prediction** section. Carry out a one-step improvement (or greedy step) on the random policy. Represent in a 2D matrix the policy you obtain. How can we verify that it is a better policy than the random one?

6. **Policy Iteration II:** Continue iterating the Prediction and the greedy steps until convergence to the optimal policy.

7. Investigate the structure of the optimal policy for different values of γ, and explain the results. You might use Value Iteration or Policy Iteration.