CS7.301 **13^{th} April, 2020**

MDL Assignment-2 Part-3

Vikrant Dewangan, Roll No.- 2018111024
Mohammad Nomaan Qureshi, Roll No.- 2018111027

Contents

1 File Structure

Upon running the file *solution.py*, we get

```
team71
├── team_71_report.pdf:  contains report
├── output:
│   └── output.json:  Containing the output
└── solution.py:  main solution
```

2 Problem Statement

The same as in Assignment 2, only this time, there is no reward for Lero to reach the terminal state.
Given: Initially, Lero has 3 arrows, full stamina while MD also has full health. Thus,

$$\alpha_{[4,3,2]} = 1.0.$$

Assumption - Order of actions to be {NOOP, SHOOT, DODGE, RECHARGE} .

3 L.P Formulation

The linear programming formulation to MDP is given as -

$$\text{Maximize} \sum V_i$$

such that

$$V_i \leq [R(I, A) + \gamma \sum P(J|I, A) \cdot V_j]$$

where V_i is the value of the i_{th} state.

Now as per our problem, we state the formulation as follows -

$$\text{Max } (\mathbf{r} \cdot \mathbf{x})$$

such that

$$\mathbf{A} \cdot \mathbf{x} = \boldsymbol{\alpha}$$
$$x_i \geq 0 \ , \ \forall x_i \in \mathbf{x}$$

where

$\mathbf{r}$: stands for list of rewards for each (state, action).

$\mathbf{x}$: expected number of times of each action in (state, action).

$\mathbf{A}$: Transition probability matrix.

$\boldsymbol{\alpha}$: Initial probability distribution.

3.1 Preparing the A matrix

Before making the **A** matrix, we first need to know the size of our **r** matrix. The following cases arrive -

1. When enemy health is 0 : Since this is terminal state, the only action performed in this case would be NOOP. Total 12 such cases.
2. When enemy health is not 0 (NOOP action not considered):
   (a) When stamina is 0 : Only option there is to RECHARGE. Total 16 such cases.
   (b) When stamina is 50:
      i. When arrows are zero : SHOOT can't happen. Hence only option is RECHARGE, DODGE. Total 4 such cases.
      ii. When arrows are not zero : All the actions are possible. Total 12 such cases.
   (c) When stamina is 100:
      i. When arrows are zero : SHOOT can't happen. Hence only option is DODGE. Total 4 such cases.
      ii. When arrows are not zero : All the actions are possible except for RECHARGE. Total 12 such cases.

Thus the length of **r** array would be

$$12 \times 1 + 16 \times 1 + 4 \times 2 + 12 \times 3 + 4 \times 1 + 12 \times 2 = 100.$$

Now out of which we know, the reward would be 0 only when either it is at a terminal state, or when having non-finite number of arrows, non-zero stamina and enemy in pen-ultimate health level and action chosen is SHOOT. That is,

$$r_{[i,j,k]} = \begin{cases} 0 \text{ when } i = 0 \\ -5 \text{ otherwise} \end{cases}$$

The dimensions of A matrix are the

$$\begin{aligned} & \text{number of states} \times \text{number of variables} \\ = \; & 60 \times 100 \end{aligned}$$

Each row would correspond to a state and each column would correspond to a variable that is, an action taken in a particular state. For each element of the matrix A_{ij}, it denotes the transition probability of the action j **FROM** state i. It is negative if the action leads Lero into that state and positive if it is originated from it. Using this, we can construct our matrix A -

$$\text{A} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots & & \\ 0 & 0 & 0 & \dots & 0 & 0 & -0.04 \\ 0 & 0 & 0 & \dots & 0.8 & 0 & -0.16 \\ 0 & 0 & 0 & \dots & -0.8 & 1 & 1 \end{pmatrix}$$

Thus the A matrix is obtained.

3.2 Rendering the policy

Following procedure is followed -

- Upon solving the linear programming and obtaining $\mathbf{x}$, we get a linear array of values each being **deterministic** uniformly optimal policy. We get it as follows -

  $$\mathbf{x} = [x_{1a1}, x_{1a2}, x_{2a1}, x_{3a1}, x_{3a2} \dots]$$

  Either of x_{nai} is non zero and all being non zeroes confirming to a deterministic policy.

- For example, we get for the stage [1,1,2], SHOOT is the best option. We iterate over the $\mathbf{x}$ array and check for all actions from a particular state and choose the one with the highest probability.

3.3 Changing the policies

The final policy is dependent on a number of factors, including especially the **order** of policies taken. For example, if we have SHOOT before RECHARGE, then in case of a tie, SHOOT would be given priority before RECHARGE. The way it affects is as follows -

- Changing the order of actions changes the way A matrix is written. Say, if earlier order was NOOP,SHOOT, RECHARGE and then DODGE, if the order is changed, it would lead to different actions within a state reaching to different states.

- If a particular action changes, so does it's reward. In this case the **r** matrx will have it's columns juggled. This will lead to a change in the solution of linear programming to appropriately max out the variables and the results coefficients **x** we get would not be the same. Thus, what we would see is a newer policy implemented.

- Changing the $\boldsymbol{\alpha}$ vector also changes the way we approach the problem. In this problem, we are assuming that $\boldsymbol{\alpha}$ is given to us. However, a different value of $\boldsymbol{\alpha}$ may lead to a completely different policy. Say if I am in a terminal state at the start, then I know I will only perform NOOP action and that will be my optimal policy.