Machine, Data and Learning

Assignment-2 Part-3

Task:

This assignment is linked to Assignment-2 Part-2(value iteration). A python program has to be built for the same scenario. This time, you have to pose the problem as a Linear Programming Problem(LPP) and find the optimal policy by solving the LP.
The same scenario is presented here, but there are subtle differences. Students are advised to go over the scenario again.

Scenario:

'X' is your team number.
Lero, the hero, an amateur archer is on a quest to destroy the Mighty Dragon(MD). Lero can have at most **3 arrows** with him at a time(even **zero** number of arrows is possible). Lero is immortal, but his **stamina** is limited and is maxed out at **100** points(**zero** stamina is possible). MD is a mortal dragon and its **health** is maxed out at **100** points(**zero** health is possible). When MD's health becomes zero, Lero's quest **finishes** and it's a **happy ending but sadly Lero gets NO reward**. Lero can perform 3 actions - **shoot, dodge, and recharge.**

To destroy MD, Lero can **shoot** arrows and at any time step he can **shoot only one arrow** and whenever he does so, his stamina reduces by **50 points** and he **loses** that (**one)** arrow. As Lero is an amateur archer his aim is not so great, so whenever he shoots he can **hit** MD with a probability of **0.5** and **miss** with a probability of **0.5**. When the arrow hits MD, its health reduces by **25 points,** else it remains the same**.** When he has **zero arrows or zero stamina**, he **can't shoot.**

Lero can **dodge** MD's attack. When **he dodges with stamina = 100 points**, his stamina **reduces by 50 points** with a probability of **0.8** and **reduces by 100 points** with a probability of **0.2**. If his stamina is **equal to 50 points**, his stamina **drops down to 0** with a probability of **1.0**. When Lero dodges, he picks up **one** arrow with probability **0.8** and with probability **0.2** he doesn't, but the maximum number of arrows he can have is 3. So if he has 3 arrows and performs a dodge, he will still have 3 arrows. He **can't dodge if his stamina is 0.**

Lero can **recharge** his stamina to **increase it by 50 points**, but this happens only 80% of the time. The rest of the time, it stays the same. If his stamina while recharging is 100 points, it remains the same(there won't be any increment)(not a valid action in such a case).

For each action taken, Lero suffers a **penalty = -10/Y.**
Where

arr = [½, 1, 2]

Y = arr[X mod 3]

In short,

```
if (X mod 3 == 0) {
    Y = ½
    Penalty = -20
} else if (X mod 3 == 1) {
    Y = 1;
    Penalty = -10
} else if (X mod 3 == 2) {
    Y = 2;
    Penalty = -5
}
```

No negative number of arrows, a negative value of MD's health and negative stamina of Lero is possible. All values are greater than or equal to zero always and maxed out with their aforementioned respective limits.

Note: Lero has to pay step cost for all actions that he is taking (DODGE, SHOOT or RECHARGE). Whenever he takes an action, in that case, he has to pay step_cost(a negative number).

NOOP: In terminal states, there is no action available, so you only consider the NOOP action with **0 step cost**. Taking this action gets you back to the same state with a probability of 1. This action is not available in any of the other(non-terminal) states.

Start state to be considered is Lero having 3 arrows and full stamina and MD has full health.

Output format:

You are required to output 6 things. You are required to put all of them in a dictionary with keys as specified.

	Data Type	***Dictionary Key***
The A matrix	list of lists	a
The R array	list of floats	r
The alpha array	list of floats	alpha
The optimised x array	list of floats	x
The derived deterministic policy	list of lists; Each sublist is of the form [state, action] State: A list like in the previous assignment Action: A string from [“NOOP”, “RECHARGE”, “DODGE”, “SHOOT”]	policy
The final optimised objective value	float	objective

For the policy

The **state** is a list of the form

```
[enemy_health, number_of_arrows_remaining, stamina]
```

Increment of the state indices should be done in the order: stamina, number_of_arrows, enemy_health. Eg: [0,0,0] -> [0,0,1] -> [0,0,2] -> [0,1,0] …….

Actions should be in all caps. Possible actions are NOOP, SHOOT, DODGE and RECHARGE.

Sample Dictionary

```
{
   "a": [[1.0, 0.0, 0.0, 0.0],
         [0.0, 1.0, 0.0, 0.0],
         [0.0, 0.0, 1.0, 0.0]],
   "r": [1.0, 1.0, 1.0, 1.0],
   "alpha": [0, 0, 0, 1],
   "x": [1.724, 1.784, 0.645, 1.854],
   "policy": [
       [[0, 0, 0], "NOOP"],
       [[0, 0, 1], "SHOOT"],
       [[0, 0, 2], "DODGE"],
       [[0, 1, 0], "RECHARGE"]],
   "objective": -11.54321
}
```

Saving to the output.json

You need to JSON dump (using json.dump in python) this dictionary into the output.json file. **Please check that you are able to read back the exact same dictionary using json.load, before submitting.**

The output file corresponding to the above sample dictionary can be found in the assignment zip (sample_output.json).

Report:

The report should explain

1. Procedure for making the A matrix.
2. Procedure for finding the policy.
3. Can there be multiple policies? Why? What changes can you make in your code to generate another policy? **(Do not paste code snippets, explain the changes in terms of how they will affect the A matrix, R vector, alpha vector etc.)**

Dimensions of the A matrix:

The A matrix should consist of **only the valid actions for each state**. Eg, for states in which Lero has 0 stamina, the only valid action is "RECHARGE". The other actions should neither appear in the A matrix, nor in any of the alpha, R or X vectors.

The number of **rows (lists) in A matrix must be equal to the number of states**. The dimensions of other values should follow logically from this.

Note:

- Languages allowed: **python3.**
- Libraries allowed: **Numpy, cvxpy.**
- The calculated values should be accurate to **3 decimal places**.
- This will be a **team assignment.**
- The evaluation will be **AUTOMATED,** so kindly stick to the output and submission format. **Failure to stick to the format will result in a direct 0 for the assignment.**
- Marking scheme:
    - Code: 20
    - Automated evaluation (output): 20
    - Report: 10
- **Plagiarism will be penalized.**

Submission Details:

- **Deadline: 11:55 pm, 10th April 2020.**
- **Submission format:**
    - Submit <team_number>.zip file containing 1 folder. The structure is as follows.

      team_<team_number>
      ├─── outputs
      │ └─── output.json
      ├─── solution.py
      └─── team_<team_number>_report.pdf

- **When your code (solution.py) is run, it should create an outputs directory with the output.json file inside it.** We will be running your code like

  ```
  >> python3 solution.py
  ```

  We expect it to create an outputs directory with an output.json file inside it.

- Check that your submission is in the correct format. **Submissions having Incorrect format will get a straight 0.**
- No late submissions will be allowed this time.
- There should be **only 1 submission per team**. If there are multiple we can't guarantee that we will check any submission at all.