Machine, Data and Learning

Assignment-2 Part-2

Task-1:

Code the value iteration algorithm for the given scenario to obtain the optimal policy and the state reward values corresponding to it.

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 and Lero gets a reward of +10**. 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).

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, negative value of MD's health and negative stamina of Lero are possible. All values are greater than or equal to zero always and maxed out with their aforementioned respective limits.

- **Parameters:**
    - **Gamma =** (Discount Factor) 0.99
    - **Delta =** (Convergence or Bellman error) 10^{-3}

Task-2:

For all the parts in this task, take **step cost = -2.5**, irrespective of your team number.

- **Part-1:**
    - Change the step cost of **only shoot action to -0.25**. Other parameters remain the same.
    - What changes do you observe in the policy and why?

- **Part-2:**
    - Change the gamma value from **0.99 to 0.1.** All other parameters remain the same.
    - This is **not** in continuation of part 1. i.e. **step costs for all actions are -2.5.**
    - Explain Lero's weird behaviour.

- **Part-3:**
    - This is in continuation of part 2. **(gamma = 0.1)**
    - This is not in continuation of part 1. i.e. **step costs for all actions are -2.5.**
    - Change the value of delta to $\mathbf{10^{-10}}$.
    - What changes do you observe and why?

Output format:

For each iteration:

print “iteration=<number>(0 based)”

For each state:

print “(enemy_health,number_of_arrows,stamina):Action_chosen=[State Value]”

print <newline>

print <newline>

The **state** is a tuple. It should be printed within round brackets having 3 values - enemy_health, number_of_arrows_remaining, and 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 -> 0,1,0 -> 0,1,1 …….

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

The **value of the state** should be accurate to $\mathbf{10^{-3}}$**.**

Eg.

iteration=0
(0,0,0):SHOOT=[-11.933]
(0,0,1):DODGE=[-11.933]
.
.
.
<newline>
<newline>
iteration=1
(0,0,0):SHOOT=[-11.933]
(0,0,1):DODGE=[-11.933]
.
.
.
. (so on until end of iteration)

For every iteration, print the information about all states in the stated format. Each state information should be in a newline. **Note: There is no <space> character in the output. Only <newlines>.**

Note:

- Languages allowed: **python3.**
- Libraries allowed: **Numpy.**

- 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:
    - Task 1: 46
    - Task 2: 54 (18 for each part).
- **Plagiarism will be penalized.**

Submission Details:

- **Deadline: 11:55 pm - 5th, March 2020.**
- **Submission format:**
    - Submit <team_number>.zip file containing 1 folder. The structure is as follows.

```
team_<team_number>/
├─── outputs
│    ├─── task_1_trace.txt
│    ├─── task_2_part_1_trace.txt
│    ├─── task_2_part_2_trace.txt
│    └─── task_2_part_3_trace.txt
├─── task_1.py
└─── team_<team_number>_report.pdf
```

    - The report should contain inferences from the policy obtained in Task 1 and answers/(observations and conclusions) to all the parts of Task 2.
    - **Note: Trace files contain the output of running the VI algorithm for the respective questions in the Output Format given above.**