MDL Assignment-2 Part 2 REPORT

Team Number : 22

Team Name : Room543

Mihir Bani, 2019113003

Amul Agrawal, 2019101113

Contents:

Part-2: Value Iteration (VI)

Overview

The Bellman equation is the basis of the value iteration algorithm for solving MDPs. The Bellman equation looks like this:

$$V(s) = \max_a \left(R(s,a) + \gamma \sum_{s'} P(s,a,s')V(s') \right)$$

Here,

$V(s)$ is the new value of state *s*.

a is a action that is possible from state s

$R(s,a)$ is the reward for taking action *s* from state *a*.

$P(s,a,s')$ is the probability of moving from state *s* to *s'* using action *a*

$V(s')$ is the old value of state *s'*

γ is the Bellman coefficient.

Basically this equation tries to pick the best action that is possible from the current state.

The Pseudo code for the same is:

```
function VALUE-ITERATION($mdp, \epsilon$) returns a utility function
  inputs: $mdp$, an MDP with states $S$, actions $A(s)$, transition model $P(s' \mid s, a)$,
          rewards $R(s)$, discount $\gamma$
        $\epsilon$, the maximum error allowed in the utility of any state
  local variables: $U$, $U'$, vectors of utilities for states in $S$, initially zero
                   $\delta$, the maximum change in the utility of any state in an iteration

  repeat
     $U \leftarrow U'$; $\delta \leftarrow 0$
     for each state $s$ in $S$ do
        $U'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\, U[s']$
        if $|U'[s] - U[s]| > \delta$ then $\delta \leftarrow |U'[s] - U[s]|$
  until $\delta < \epsilon(1-\gamma)/\gamma$
  return $U$
```

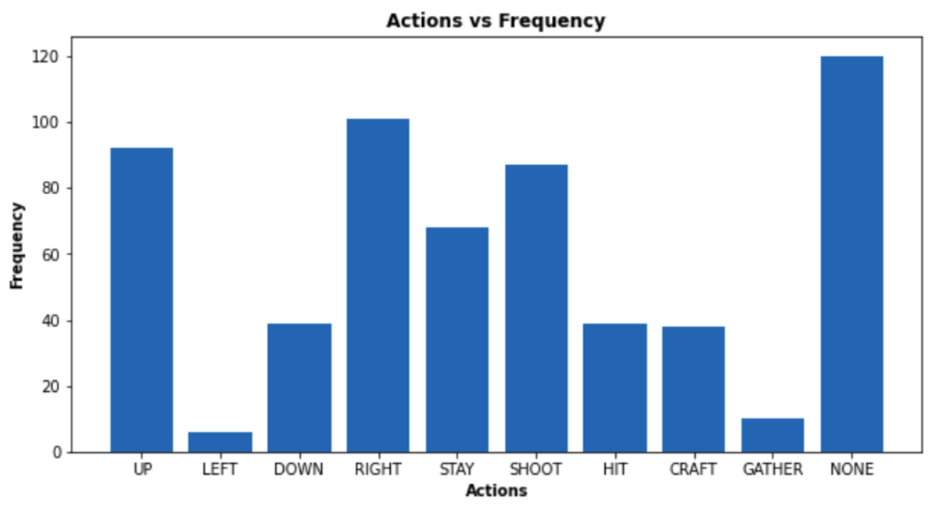
Task 1

```
STEP COST = -10
GAMMA = 0.999
```

```
BELLMAN ERROR = 0.001
```

Comments on the Policy

1. Graph showing the frequency distribution of actions in the optimal policy. The NONE action is taken by all states which have MM health as zero.



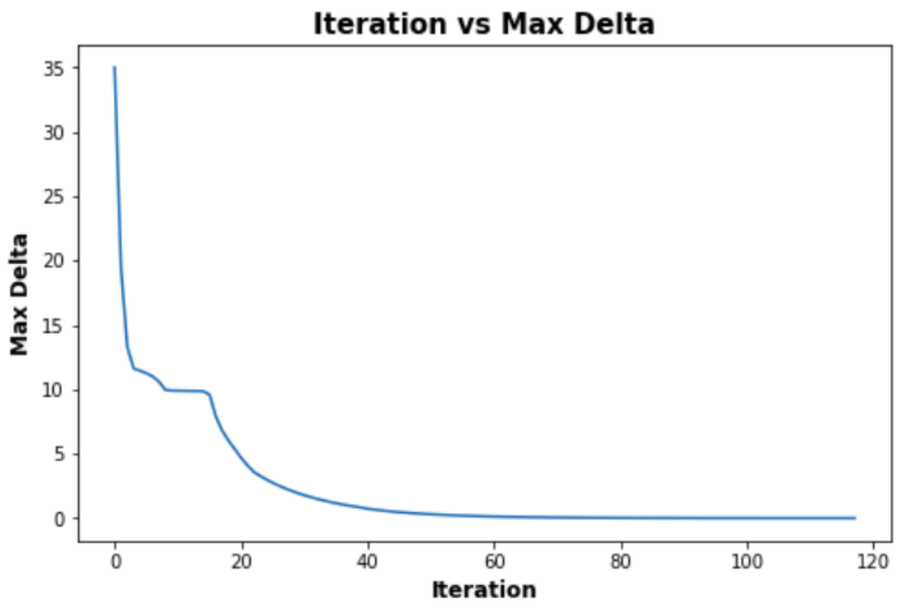
2. **Centre Square:**

    1. If MM is Dormant:
       Goes RIGHT to East square
    2. If MM is Ready:
        1. Materials > 0 and Arrows < 3:
           Goes UP to the North Square
        2. else
           Goes DOWN to the South Square
3. **East Square**
    1. if MM health is 100, he tries HIT action.
    2. if MM health is not 100, he SHOOT all his arrows first.
4. **North Square**

   Takes the first possible action in the following sequence.
    1. Materials > 0 and Arrows < 3:
       He CRAFTs arrows.
    2. MM is Ready:
       He STAYs in the same square.
    3. MM is Dormant:
       He moves DOWN.
5. **South Square**
    1. MM is Ready:
       He GATHERS material if he has zero materials otherwise he STAYS.
    2. MM is Dormant:
       He moves UP.
6. **West Square**
    1. MM is Dormant:
       Moves RIGHT
    2. MM is Ready:
       Either STAYS or SHOOTS

Convergence

It took 118 iterations for Value iteration to converge. Here is the graph for the same. Delta is the maximum change in the value of state on current iteration w.r.t to the previous iteration.
$Delta = abs(V_t(s) - V_{t+1}(s))$



Simulations

tuple: (Position IJ, # Materials, # Arrows, MM state, MM Health)

1. **(W, 0, 0, D, 100)**

```
Simulation from ('W', 0, 0, 'D', 100)
CURR: ('W', 0, 0, 'D', 100)  ACTION: RIGHT  MM_ATTACKED: NOT ATTACKED
CURR: ('C', 0, 0, 'D', 100)  ACTION: RIGHT  MM_ATTACKED: NOT ATTACKED
CURR: ('E', 0, 0, 'R', 100)  ACTION: HIT    MM_ATTACKED: NOT ATTACKED
CURR: ('E', 0, 0, 'R', 100)  ACTION: HIT    MM_ATTACKED: NOT ATTACKED
CURR: ('E', 0, 0, 'R', 100)  ACTION: HIT    MM_ATTACKED: NOT ATTACKED
CURR: ('E', 0, 0, 'R', 100)  ACTION: HIT    MM_ATTACKED: SUCCESS
CURR: ('E', 0, 0, 'D', 100)  ACTION: HIT    MM_ATTACKED: NOT ATTACKED
CURR: ('E', 0, 0, 'D', 100)  ACTION: HIT    MM_ATTACKED: NOT ATTACKED
CURR: ('E', 0, 0, 'D', 100)  ACTION: HIT    MM_ATTACKED: NOT ATTACKED
CURR: ('E', 0, 0, 'R', 50)   ACTION: HIT    MM_ATTACKED: SUCCESS
CURR: ('E', 0, 0, 'D', 75)   ACTION: HIT    MM_ATTACKED: NOT ATTACKED
CURR: ('E', 0, 0, 'R', 75)   ACTION: HIT    MM_ATTACKED: SUCCESS
CURR: ('E', 0, 0, 'D', 100)  ACTION: HIT    MM_ATTACKED: NOT ATTACKED
CURR: ('E', 0, 0, 'D', 50)   ACTION: HIT    MM_ATTACKED: NOT ATTACKED
CURR: ('E', 0, 0, 'R', 50)   ACTION: HIT    MM_ATTACKED: NOT ATTACKED
CURR: ('E', 0, 0, 'R', 0)    ACTION: NONE   MM_DIED
```

End state is reached.

As expected from the policy, we can see here that the optimal policy tries to take IJ to the East square, as that square has higher chances of successful attack by IJ. He didn't try to gather materials and then craft arrows out of it because he would have to take multiple extra steps (atleast 6 extra steps, meaning 6*10 penalty, which is even greater than reward) for it, and step cost is quite high for it.

2. **(C, 2, 0, R, 100)**

```
Simulation from ('C', 2, 0, 'R', 100)
CURR: ('C', 2, 0, 'R', 100)  ACTION: UP      MM_ATTACKED: NOT ATTACKED
CURR: ('N', 2, 0, 'R', 100)  ACTION: CRAFT   MM_ATTACKED: NOT ATTACKED
CURR: ('N', 1, 1, 'R', 100)  ACTION: CRAFT   MM_ATTACKED: NOT ATTACKED
CURR: ('N', 0, 3, 'R', 100)  ACTION: STAY    MM_ATTACKED: UNSUCCESS
CURR: ('N', 0, 3, 'D', 100)  ACTION: DOWN    MM_ATTACKED: NOT ATTACKED
CURR: ('C', 0, 3, 'D', 100)  ACTION: RIGHT   MM_ATTACKED: NOT ATTACKED
CURR: ('E', 0, 3, 'D', 100)  ACTION: SHOOT   MM_ATTACKED: NOT ATTACKED
CURR: ('E', 0, 2, 'D', 75)   ACTION: SHOOT   MM_ATTACKED: NOT ATTACKED
CURR: ('E', 0, 1, 'D', 50)   ACTION: SHOOT   MM_ATTACKED: NOT ATTACKED
CURR: ('E', 0, 0, 'D', 25)   ACTION: HIT     MM_ATTACKED: NOT ATTACKED
CURR: ('E', 0, 0, 'D', 0)    ACTION: NONE    MM_DIED
```

End State is reached.

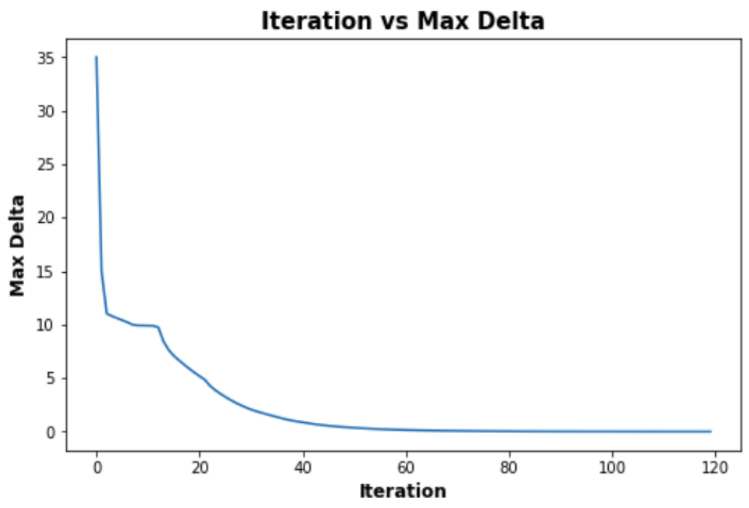
The results of the simulation match our expectations. IJ having 2 materials in hand first moves to North square to craft arrows as he has none of them at the moment. He then uses both of his materials to craft arrows. Now in order to protect himself from attack by MM on the Centre square, he waits on the North square. After MM turns Dormant he moves to the East square. Now here, he first shoots all his arrows and then hits MM by his blade as expected.

Task 2

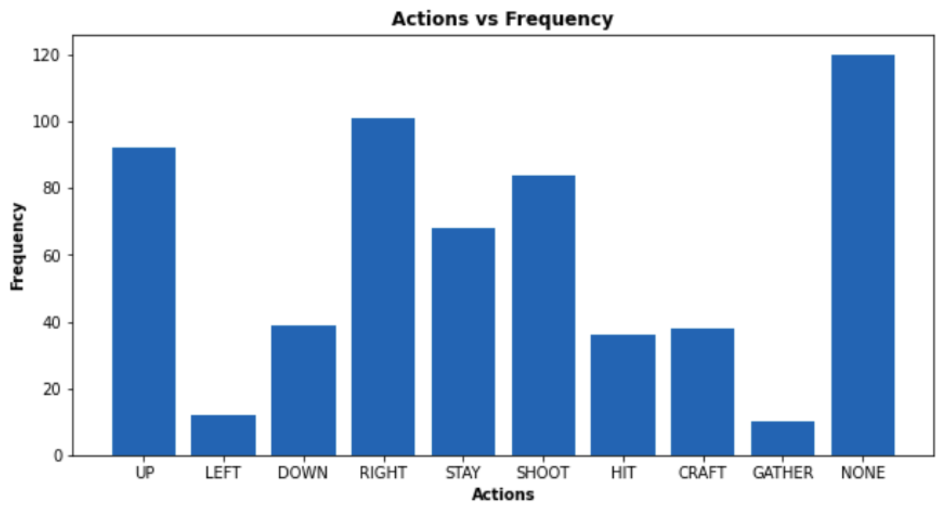
Case 1

> Left Action From East Square takes IJ to West Square.

1. **Convergence**
   It took 121 iterations to converge.



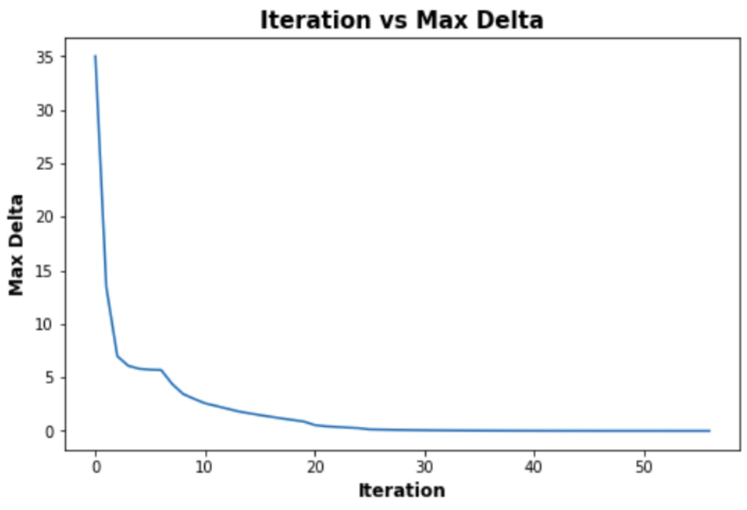
2. **Change In Policy**
   No change in policy. This was also expected as going to West square serves no benifit to IJ as we neither can't CRAFT, GATHER there nor we have good chances of successful attack. Since due to high step cost, IJ avoids going to LEFT from East square.



Case 2

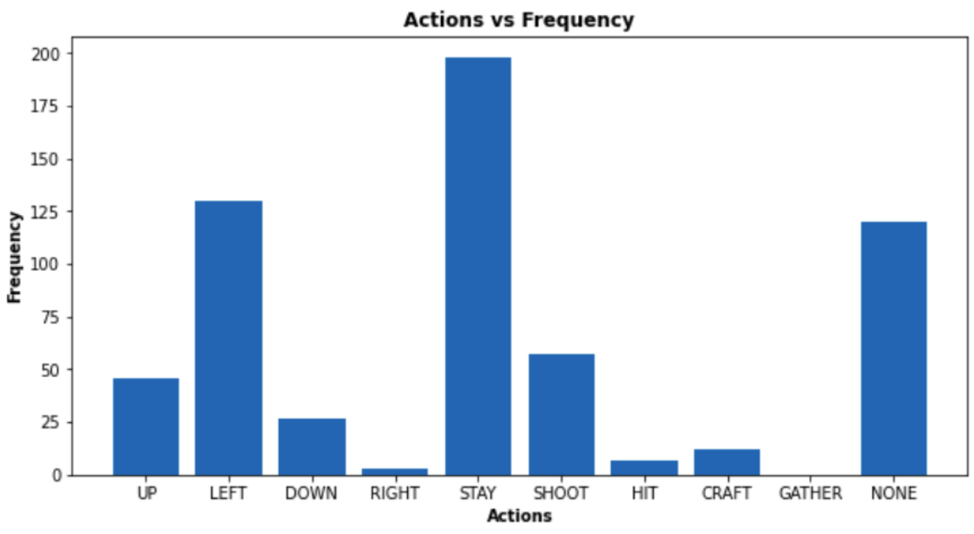
> The step cost of the STAY action is now zero.

1. **Convergence**
   It took 58 iterations for the Value Iteration to converge from here. It converged faster as in the optimal policy from most states it just tries to go to West State and stay there. He goes to west square because MM can't attack him there.



2. **Change in Policy**

- Increase in frequency of STAY action



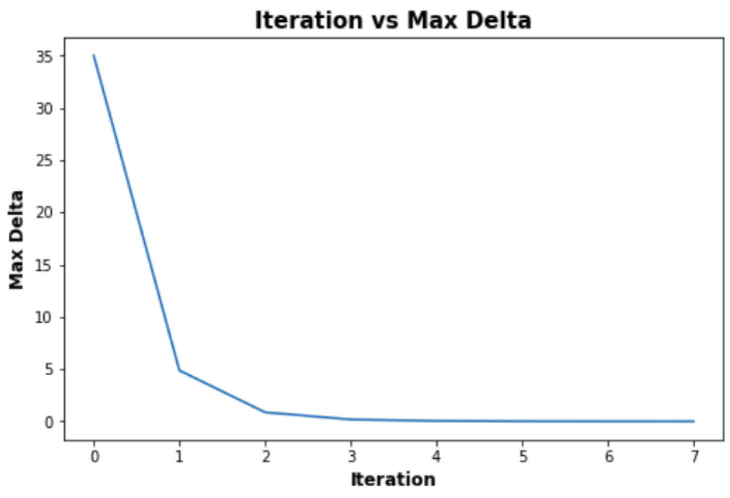
- 2. In this new policy, **IJ always tries to go to West Square and STAY**. This completely makes sense as the STEP COST of other actions is too high for their rewards. So what he does is he goes to West Square (as MM can't attack him there) and Just STAYS there enduring no STEP COST.
- 3. IJ decides to **STAY when MM is ready and he is on North or South square**, whereas he decides to **move to Centre when MM is dormant and he is on North or South square.** This makes sense as STAY action has zero cost and he avoids going to center square when MM is ready.

Case 3

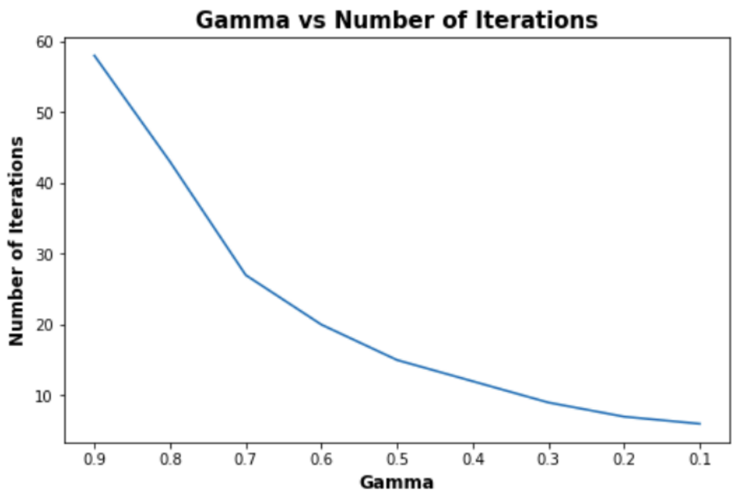
> Change the value of gamma to 0.25

1. **Convergence**

It took only 8 iterations to converge. This was expected as when the gamma value decreases the VI will put more weight to short-term gains as compared to long-term benefits.



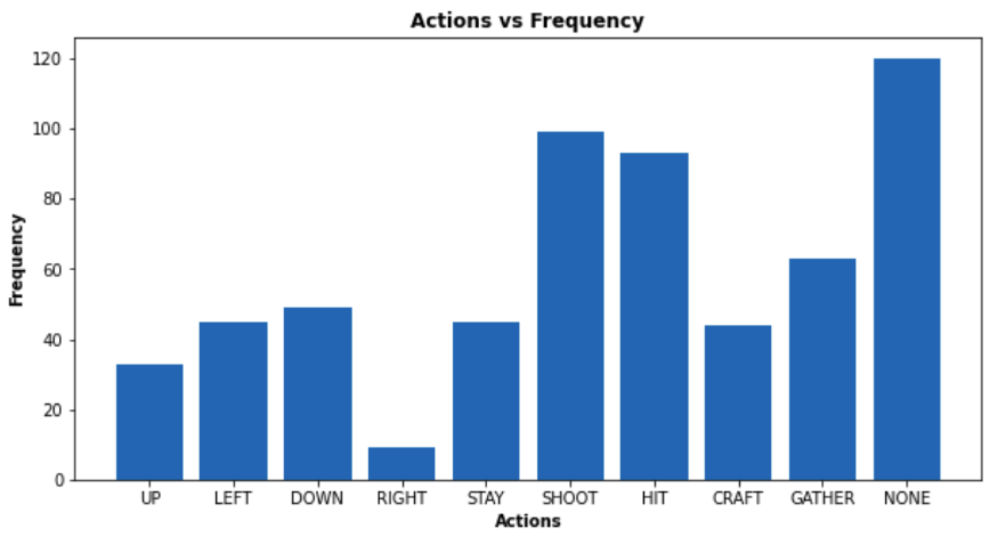
Plotted a graph for Number of Iterations VI takes to converge vs Gamma



2. **Change in Policy**

   The policy changes so as to maximize the short-term benefits. Some of the decisions are as follows:

   - We can see low number of right movements as going to East or Centre increases chances of getting attacked by MM in the near future. We can say that in this policy IJ tries to show risk avert behaviour.



- **Tries to Hit when MM health is 50 and shoot when MM health is 25**. This is because by doing so MM health will drop to zero and IJ will receive reward from this. He neglects the fact that shooting arrows have higher chances of success because to him it doesn't matter as even in case of success MM health drops to 25, which gives no reward whereas even if low probability, hitting with blade will kill MM in near future and he will receive reward.
- **He tries to go to West Square.** He does this because the west square is one of the safest squares (MM can't attack and also he can move with 100% probability from there). Low value of gamma weakens his foresight and he just tries to come to a safe state as soon as possible.
- Over here **most simulations end with IJ not killing MM**. IJ moves to a safe square and just stays there and decides not to kill MM.