MDL ASSIGNMENT-2 REPORT (PART - 2, 3)

Team Number: 44
Team Name: LearningMachinesFromData
Team Members:
T.H.Arjun (Roll number: 2019111012),
Gokul Vamsi Thota (Roll number: 2019111009)

PART 2:

Task 1:

Analysis -

- ***WEST*** - When IJ is in WEST square, he prefers to take the action ***'RIGHT'***. This is because he **wants to get to the centre square** as quickly as possible, just so **he can attack MM better from a closer distance**.

- ***NORTH*** - When IJ is in NORTH square, he prefers to move ***'DOWN'*** **if he doesn't possess any material**, because in this case it is optimal to get closer to MM for attacking him as he cannot make any arrows here, but **if he has some material,** he prefers to ***'CRAFT'*** and makes use of this material to obtain arrows for attacking MM.

- ***EAST*** - When IJ is in EAST square, and MM is in ready state (and might possibly attack), IJ moves ***'LEFT'***. Otherwise he keeps on trying to ***'HIT'*** MM from this square, or trying to ***'SHOOT'*** him (**depends on tradeoff between *'SHOOT'* & *'HIT'* for the loss of health of MM and accuracy**).

If he has some material but no arrows when he is at this square, he tries to go ***'LEFT'*** **and obtain some arrows** (by attempting to reach NORTH state and trying to *'CRAFT'* arrows there) **so that he can return** to this square and attack again, as the action ***'SHOOT'*** **has a higher probability** of success in comparison with the action *'HIT'*. **But in most cases, the action *'HIT'* is preferred as there is no loss of any property of IJ in choosing this action** (whereas he would lose arrows on choosing the action *'SHOOT'*).

- ***SOUTH*** - When IJ is in SOUTH square, he attempts to move ***'UP'*** **to reach CENTRE square, so that he can then further attempt to *'HIT'* or *'SHOOT'* MM** from that position. This is clearly optimal as decreasing MMs health has a high positive reward.

- ***CENTRE*** - When IJ is in CENTRE square, he mostly attempts to take the ***'RIGHT'*** action and **thus reach the EAST square, so that he has better probabilities of success / accuracy of actions *'SHOOT'* and *'HIT'***. If the health of MM is low, then he tries to ***'SHOOT'*** **if sufficient arrows are present** (tradeoff between step cost and accuracy differences with EAST state). He also tries to go ***'UP'*** **from CENTRE, if he is low on arrows**, **such that he can *'CRAFT'* and obtain more arrows in the NORTH square.**

Simulation -

1) **Start state = (W, 0, 0, D, 100):**
   Output of Simulation:
   Step Number: 1 Current State: (W,0,0,D,100) Taking Action: RIGHT
   Step Number: 2 Current State: (C,0,0,D,100) Taking Action: RIGHT
   Step Number: 3 Current State: (E,0,0,D,100) Taking Action: HIT
   Step Number: 4 Current State: (E,0,0,D,50) Taking Action: HIT
   Step Number: 5 Current State: (E,0,0,R,50) Taking Action: HIT
   Step Number: 6 Current State: (E,0,0,R,50) Taking Action: HIT
   Step Number: 7 Current State: (E,0,0,D,75) Taking Action: HIT
   Step Number: 8 Current State: (E,0,0,D,75) Taking Action: HIT
   Step Number: 9 Current State: (E,0,0,D,25) Taking Action: HIT
   Step Number: 10 Current State: (E,0,0,D,25) Taking Action: HIT
   Step Number: 11 Current State: (E,0,0,D,25) Taking Action: HIT

Step Number: 12 Current State: (E,0,0,D,25) Taking Action: HIT
Step Number: 13 Current State: (E,0,0,D,25) Taking Action: HIT
Step Number: 14 Current State: (E,0,0,R,25) Taking Action: HIT
Step Number: 15 Current State: (E,0,0,D,50) Taking Action: HIT
Step Number: 16 Current State: (E,0,0,D,50) Taking Action: HIT
Step Number: 17 Current State: (E,0,0,D,50) Taking Action: HIT
Step Number: 18 Current State: (E,0,0,D,0) Taking Action: NONE

Analysis: We followed the policy, since there are zero arrows and zero material at WEST starting state its better to take the RIGHT action here and move to state E from where you can take HIT action. HIT action is preferred since MM is in Dormant State and even if he changes to READY state he has nothing to lose as he has 0 arrows, and he can keep on 'HIT'-ting MM without losing anything and finally win. This is a very good strategy.

2) **Start state = (C, 2, 0, R, 100):**

Output of Simulation:

Step Number: 1 Current State: (C,2,0,R,100) Taking Action: RIGHT
Step Number: 2 Current State: (E,2,0,R,100) Taking Action: LEFT
Step Number: 3 Current State: (C,2,0,R,100) Taking Action: RIGHT
Step Number: 4 Current State: (E,2,0,R,100) Taking Action: LEFT
Step Number: 5 Current State: (C,2,0,R,100) Taking Action: RIGHT
Step Number: 6 Current State: (C,2,0,D,100) Taking Action: UP
Step Number: 7 Current State: (E,2,0,R,100) Taking Action: LEFT
Step Number: 8 Current State: (E,2,0,D,100) Taking Action: LEFT
Step Number: 9 Current State: (C,2,0,R,100) Taking Action: RIGHT
Step Number: 10 Current State: (E,2,0,R,100) Taking Action: LEFT
Step Number: 11 Current State: (C,2,0,R,100) Taking Action: RIGHT
Step Number: 12 Current State: (E,2,0,R,100) Taking Action: LEFT
Step Number: 13 Current State: (C,2,0,R,100) Taking Action: RIGHT
Step Number: 14 Current State: (C,2,0,D,100) Taking Action: UP
Step Number: 15 Current State: (N,2,0,D,100) Taking Action: CRAFT
Step Number: 16 Current State: (N,1,1,D,100) Taking Action: CRAFT
Step Number: 17 Current State: (N,0,2,D,100) Taking Action: DOWN
Step Number: 18 Current State: (C,0,2,D,100) Taking Action: RIGHT
Step Number: 19 Current State: (E,0,2,D,100) Taking Action: HIT

Step Number: 20 Current State: (E,0,2,D,100) Taking Action: HIT
Step Number: 21 Current State: (E,0,2,D,100) Taking Action: HIT
Step Number: 22 Current State: (E,0,2,D,50) Taking Action: SHOOT
Step Number: 23 Current State: (E,0,1,R,50) Taking Action: HIT
Step Number: 24 Current State: (E,0,0,D,75) Taking Action: HIT
Step Number: 25 Current State: (E,0,0,D,75) Taking Action: HIT
Step Number: 26 Current State: (E,0,0,R,75) Taking Action: HIT
Step Number: 27 Current State: (E,0,0,D,100) Taking Action: HIT
Step Number: 28 Current State: (E,0,0,D,100) Taking Action: HIT
Step Number: 29 Current State: (E,0,0,D,100) Taking Action: HIT
Step Number: 30 Current State: (E,0,0,D,100) Taking Action: HIT
Step Number: 31 Current State: (E,0,0,D,50) Taking Action: HIT
Step Number: 32 Current State: (E,0,0,D,50) Taking Action: HIT
Step Number: 33 Current State: (E,0,0,D,50) Taking Action: HIT
Step Number: 34 Current State: (E,0,0,D,50) Taking Action: HIT
Step Number: 35 Current State: (E,0,0,D,50) Taking Action: HIT
Step Number: 36 Current State: (E,0,0,D,50) Taking Action: HIT
Step Number: 37 Current State: (E,0,0,D,50) Taking Action: HIT
Step Number: 38 Current State: (E,0,0,D,0) Taking Action: NONE

Analysis:

Initially he moves towards MM hoping that he remains in R state, but now since he didn't change he has chances that he changes in the next move so he gets scared and moves away. He keeps on doing this scared tactic until MM goes to Dormant State. IJ also goes to N state to CRAFT arrows until MM goes Dormant so that he can attack with Arrows from East with high accuracy when MM is in Dormant State. He comes to East square when MM is in Dormant State and Shoots and HITS him and defeats him.

Task 2:

Case 1:

In this case, when IJ chooses the action ‘LEFT’ at EAST square, he would end up at WEST square if successful. This change would not affect the policy which was computed earlier, as this change would not put IJ in any advantageous position (disadvantageous rather). As taking ‘LEFT’ action was not optimal based on previous policy, it would definitely not be optimal now, and hence, is not preferred. Thus, the policy remains unaffected with this change. He won’t opt to ‘SHOOT’ at both these positions, as the accuracy is bad for the same. This is because he is greedy with respect to obtaining the reward quickly to avoid step cost as much as possible.

Case 2:

In this case, the step cost when ‘IJ’ chooses to ‘STAY’ in the same square is 0. Here, the policy would change. Evidently, IJ Prefers 0 step cost over negative step costs since the goal of IJ is to maximize expected reward, and hence would rather prefer to ‘STAY’ in the same square and not gain any reward, in comparison with taking an action which changes his state but makes him lose reward. He is now more selective in picking his actions, and would filter out the possible actions from current state such that he would choose only those actions which are worthy of the negative step cost (actions which would assist in gaining higher reward by attacking MM). Such actions would typically be chosen if the health of MM is low and IJ has a decent chance to reduce his health significantly. Otherwise ‘ IJ’ would choose to ‘STAY’ (especially when MM attacks), preferring the 0 reward, over negative costs and a depreciated reward for destroying MM. But if he somehow reaches a square close to MM (if ‘STAY’ fails), he tries to kill MM. IJ also moves closer to MM if he has a lot of arrows and health of MM is low.

Case 3:

In this case, Gamma is reduced to 0.25 from the high value of 0.999. The algorithm now converges faster (in lesser iterations) because of the decreasing impact of the values of next states along iterations. Subtle changes in policy are observed here. We start to observe ‘SHOOT’ as one of the optimal actions at WEST even if it has low accuracy. This typically occurs when health of MM is low and IJ has sufficient arrows (as he is more interested in the current gain of the positive reward). North is approached in a similar way as the initial policy. If he is in EAST square, IJ chooses to ‘HIT’ if health of MM is high and IJ wants to deal as much damage as possible quickly, otherwise he chooses to ‘SHOOT’ as it has better accuracy. IJ further prefers to ‘GATHER’ at SOUTH, if he is low on Material, so that he can ‘CRAFT’ at NORTH.

PART - 3:

Procedure of constructing A matrix:

- The matrix A is a list of lists, where there is a unique list that corresponds to a particular state, of the given MDP.

- Let's consider S to be the set of all possible states of the given MDP, and let's consider that every state S_i has a set of valid actions which can be given by the list a_i for that state, and each element of this list of actions can be represented as a_{ij} where i corresponds to state S_i and j corresponds to the j^{th} action in the list $a_{i.}$

- The list / row for a state S_i would be represented by the i^{th} list / row of the matrix A. Thus, if we define the procedure for constructing the i^{th} list of this matrix for a state S_i, then the same procedure can be followed for obtaining a list for all states and thus the A matrix will be completely obtained. Note that the number of rows in A matrix would be the same

as the number of states of the given system. Thus let's define the procedure for constructing the i^{th} row of the matrix for state S_i.

Steps to follow for constructing i^{th} row of matrix A:

1) Iterate through all possible states, and visit them one by one.

2) For the current state which is being visited, consider all possible actions which can be taken from this state. In other words, if we consider the current state being visited to be S_k, consider the list of actions a_k, and iterate through all possible actions a_{kj} in this list a_k.

3) If the current state being visited (say S_k) is not S_i (the state for which we are computing corresponding list in A), then, find the probability of reaching the state S_i from S_k (in the next step) on taking the action a_{kj} (for each action in a_k), and append the negative of this probability to the list A_i. The negative sign indicates that in this transition, the final state is S_i (It can be interpreted as a directed edge which is represented by a_{kj} and probability that it would lead to state S_i, that is drawn from state S_k to state S_i).

4) If the current state being visited (S_k) is S_i , we should discard the probability of taking some action a_{kj} after which the resultant state would still be S_i, as this a self loop, and a self loop should be avoided as it would give rise to an unbounded LP. Thus, we consider all the set of valid actions a_{kj} that can be taken from S_i (which is S_k), and append to A_i only those probabilities for each action, with which the resulting state would not be S_i itself. Here, the probabilities appended would be positive because the transition occurs outward from S_i (It can be interpreted as directed edges represented by a_{kj} and probability that it would lead to some fixed state other than S_i, that is drawn outwards from state S_i to that some other fixed state).

5) Note that, if $S_i = S_k$ and S_i is the terminal state (here, when the health of MM is 0), 1 is appended to A_i, as it would take it to state S_i (remain in the same final state) with absolute certainty.

Thus, we constructed the i^{th} row of the matrix A successfully. Thus, we can follow the same procedure for every other state and obtain the matrix A.

Procedure of constructing the policy:

- The vector x is obtained on solving the LP of **maximizing *(r.x)*** under the **constraints *(A.x = alpha and x >= 0)***

- Let's consider S to be the set of all possible states of the given MDP, and let's consider that every state S_i has a set of valid actions which can be given by the list a_i for that state, and each element of this list of actions can be represented as a_{ij} where i corresponds to state S_i and j corresponds to the j^{th} action in the list a_i.

- The policy can be constructed from this x vector in the following way:

    1) Iterate over the corresponding entries for every state S_i, one by one, in the x vector. Further, for every state, iterate over all possible actions a_{ij} that can be taken from that state. Both these iterations are through corresponding entries in x vector.

    2) Say the current state (in iteration process) is S_i. Observe the values of the x vector which correspond to different actions that can be taken from this state (a_i).

    3) Compute the action a_{ij} of the list a_i that has the highest corresponding value in x vector, and make note of this action. In other words, compute argmax(x_{ij}) among all possible actions a_{ij}.

    4) This action (say a_{best}) is considered to be the most optimal action that can be taken at this state S_i.

    5) Continue with the iteration process and thus obtain the optimal action for every state. In this way, the policy is completely constructed.

Analysis of obtained policy:

- When IJ is in **WEST** square, he tries to ***'STAY'*** there as long as possible as he doesn't want to get into trouble with MM. He considers it best to stay away from attacks of MM, and thereby avoid unnecessarily boosting the health of MM in that way.

- When he is in **SOUTH** square also, he chooses to ***'STAY'*** for the same reason, or he would prefer to ***'GATHER'*** if he is low on materials.

- When in **NORTH** square also, he chooses to ***'STAY'*** there itself for the same reason.

- If his ***'STAY'*** **fails** or in some cases when **MM has low health** he **moves to CENTRE or EAST square**.

- When at **CENTRE or EAST** square , IJ appears to panic about the fact that he is now vulnerable to the attack of MM, and thus **tries to defeat MM as soon as possible**, by preferring actions like ***'SHOOT'*** and ***'HIT'*** whenever possible, **or tries to move away if MM is in the ready** state.

- This change in policy from PART 2 can be attributed to the absence of Gamma, the depreciation factor. The presence of gamma factor would discount the reward or the fulfillment of the reward obtained in the future. But as it is absent, the time step at which the reward is obtained is immaterial for IJ as the value isn't depreciated.

- IJ is now thinking of the future, as a reward in the future has the same value as getting it now. Thus, he now chooses to focus on the long term benefit instead of taking risks to obtain reward

quickly. Hence, he stays away from MM until the right opportunity presents itself, strikes at the right time devoid of risk, and then would again prefer to stay away and safe until the next opportunity. Clearly, this policy is very logical and opportunistic.

Multiple Policies

There can be multiple policies.

1) There can be cases where multiple actions have the same benefit. Thus any one of them can be taken as optimal
2) If there are many maximums then any of them can be chosen. So it depends on the selection procedure while selecting optimal policy. This is done after solving LP so the structures of A,x,R,alpha remain unchanged.
3) There can be cases where changing order of actions doesn't matter, so any order can lead to the same final state, in these cases the optimal policy can change even though it leads to the same final state. For example SHOOT, SHOOT, STAY will be the same as STAY , SHOOT , SHOOT leading to the same final states. But the policy is SHOOT in one and STAY in another. Thus it depends on the order in which the policy is looped through or analysed.
4) The optimal policy also depends on the order in which actions are chosen (preference of actions) as the structure of A matrix depends on this order. This changes the structure of A matrix. R matrix would change as order of action changed, but alpha would remain the same, x would change as A and R changed. Objective would remain the same.
5) Selecting a different start state can also result in generating a different policy to the MDP. That is, any change in the alpha vector can change the policy. But A,R remain unchanged in this case while x changes, along with objective.
6) Changing the reward/step-cost/penalty for each state can also result in a change in the policy. Since if the penalty is higher (step cost is lower) IJ will want to and try to reach the terminal state faster with a greater reward. Here the R vector will be different. A will remain the same, alpha will remain the same.

7) Changing values of parameters such as the probabilities of taking various different actions will also change policy as the whole problem will be altered in this case. In this case A will change.