MDL ASSIGNMENT-2 REPORT (PART - 3)

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

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'*** when MM has **high health** and is in **READY state**, as he is safe here. When MM is in **Dormant State** he moves ***'RIGHT'*** as he achieves better accuracy while attacking from closer to MM (such as EAST and CENTRE squares). He might also do this when he is **low on arrows or materials** so that he can go to NORTH or SOUTH via centre. Otherwise when there are sufficient arrows to attack, he prefers to ***'SHOOT'.*** He considers it best to stay away from attacks of MM, and thereby avoid unnecessarily boosting the health of MM in that way.

- When IJ is in SOUTH square, he chooses to ***'STAY'*** for the same reason, or he would prefer to ***'GATHER'*** if he is low on materials. Or when MM changes to Dormant State he prefers to go **'UP',** so that he can attempt to attack better from CENTRE or EAST square, or ***'CRAFT'*** arrows with the material gathered by reaching NORTH square**.**

- When in **NORTH** square also, he chooses to go ***'DOWN'*** when MM is in **Dormant State,** as he can get better accuracy from EAST and CENTRE. When MM is in **'READY' state,** he prefers to ***'STAY'*** here to avoid unnecessary hassles with MM by reaching CENTRE square. When he **has materials** and MM is in **Ready State** he stays here and ***'CRAFTS' .*** He chooses to craft even when he is full on arrows as this is clearly an optimal action compared to STAY as he can stay away without failure from MM, (although STAY and CRAFT would yield the same output when the upper limit of arrows is reached). If his ***'STAY'* fails** or in some cases or when **MM has low health** he moves to CENTRE or EAST square to attack MM in a better way.

- When at ‘EAST’ square, he prefers to ***‘HIT’*** MM when he doesn’t have any arrows, and ***‘SHOOT’*** when he has high arrows (tradeoff between accuracy and health loss, accuracy is given more priority if sufficient arrows are present, and HIT is preferred otherwise (as there is no loss to any attribute of IJ in choosing this action))

- 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.

- At CENTRE he prefers to take ***‘LEFT’*** action to stay away from MM when he is in **Ready state.** Or ***‘DOWN’*** when he is low on materials to ***‘GATHER’*.** Otherwise he choses ***‘RIGHT’*** to get closer so that he can ***‘SHOOTS’* .** He goes **‘UP’** when he is **low on arrows** so that he can **‘CRAFT’** and gain more arrows.

- 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.

- The alpha vector also had a key role to play in obtaining this policy. Here we are starting from state ("C", 2, 3, "R", 100), which has MM in Ready state, he can go to Dormant State by doing an attack. Hence, IJ decides to take LEFT from here . This is the opportunistic move by IJ as he plays safe and wants to get away. He doesn't care how much time it takes as there is no depreciation. He wants to play safe.

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.