MDL | Assignment 2 | Part 3

Linear Programming

In this part of the assignemnt, we solved an MDP using Linear Programming. The construction of the matrices is given below along with the analysis of the results

Construction of Matrix A:

A is a $2D$ Matrix, where each row corresponds to a unique state and each column corresponds to a unique and valid $(state, action)$ pair, i.e A_{ij} refers to the $i^{th} state$ and j^{th} $(state, action)$ pair.

The A matrix is constructed in the `generate_A()` function.
The **pseudocode** for the construction of A is as follows:

```
def generate_A():
    for every valid (state,action) in STATE_ACTIONS:
        ## No Op case
        if action == ACTION_NONE:
            A[state_row][state_action_col] = +1
            continue

        ## Else
        A[state_row][state_action_col] = 1
        for every possible new_state:
            A[new_state_row][state_action_col] -= probability[new_state]
```

Finding the Optimal Policy

The matrix X stores the number of times a particular action has to be chosen in a particular state for every such valid (state, action) pair. Since our objective is to maximise Rx to increase the total Reward, a higher value of x signifies that choosing that action a higher number of times yeilds a better reward. Thus, for each state, the element of x which corresponds to the (state,action) pair which has a highest value is the best action.

The Policy

In the West Square:

– If MM is Dormant and IJ has sufficient arrows, then IJ takes a `RIGHT` action in most of the cases. In other times, IJ either takes a `SHOOT` action or very rarely `STAY`
– If MM is Ready, IJ prefers to `STAY` or `SHOOT` depending on the health fo MM. He takes a `RIGHT` action very rarely, when he has no arrows

In the North Square:

– If MM is Dormant and IJ has arrows, IJ moves `DOWN` . If IJ is out of arrows, he mostly uses `CRAFT` when he has materials. `STAY` is used very rarely when he has no materials and no arrows.
– If MM is Ready, IJ stays in the North, by either taking a `CRAFT` (when he has sufficient materials and fewer arrows), or `STAY` action otherwise.

In the East Square:

– IJ always attacks in the East Square. IJ either `SHOOT` or `HIT` depending on the health of MM, the state of MM and the number of arrows

In the South Square:

– If MM is Dormant, IJ chooses go `UP` in most of the cases. If IJ has less materials and the health of MM is also less, then he choose to `GATHER` .
– If MM is Ready, IJ usally stays either by `GATHER` or `STAY` depending on the number of materials and Health of MM. He chooses to go `UP` to center square very rarely, mostly when he has no arrows and materials and health of MM is high.

In the Center Square:

– If MM is Dormant, IJ takes a `RIGHT` action most of the times, as he has a higher chance of success for attacking. He goes `UP` and craft arrows if he has materials and no arrows. Very rarely, he chooses to `SHOOT` .

– If MM is Ready, if IJ has less materials, he moves `DOWN` . If he has less arrows and is not out of materials, he goes `UP` . Otherwise depending on MM ’s health, he attacks with `SHOOT` or goes `LEFT` to get away from MM ’s attack zone. Very rarely, he performs the `RIGHT` action

Multiple Policies:

Yes, Multiple Policies are possible

- If for a particular state, the values in x, corresponding to more than one different valid (state, action) pair are equal and highest, either of the actions can be chosen to generate the policy for that state. In our code, we pick the first positioned action in such cases. However, having another process to pick the best action might lead to another equally good policy.
- If we change the order of columns(which refers to a valid state_action pair) in Matrix A and r simultaneosly, this will lead to a re-organisation the generated x matrix. So the the process of the above mentioned tie-breaking being the same(for our case its picking the first action which corresponds to highest element in x for a particular state) might pick different action leading to different policy.
- Also, if we modify the transition functions or rewards in the MDP, it leads to change in A and r matrix respectively, leading to a different set of solution policies.