Report

Roll No : 2018113003

Making the A Matrix

The formula for making the A matrix is :

$$A_{ij} = \delta_{ij} - Prob(i, j)$$

Where δ is the kronecker delta which returns 1 if i = j otherwise returns 0.

In this i is the initial state and j is (state,action). So for the delta function, only the state part of it is considered. Hence, the delta function is very easy to compute. The probability function can be computed from the problem given earlier.

The only thing to consider is to take valid j's. That is there are some states on which a particular action is not valid, hence need to remove it.

Code to run valid actions

```
possible_act = []
for j in action:
    for i in possible_states:
        if j == 'NOOP':
            if i[0] == 0:
                possible_act.append((i,j))
        elif j == 'SHOOT':
            if i[0] != 0 and i[1] != 0 and i[2] != 0:
                possible_act.append((i,j))
        elif j == 'DODGE':
            if i[0] != 0 and i[2] != 0:
                possible_act.append((i,j))
        else:
            if i[0] != 0 and i[2] != 2:
                possible_act.append((i,j))
```

This is the code that removes invalid actions for a state. After having the states and the (state,action) pair, computation of A is as simple as running the formula given for A_{ij} for all i and j.

Procedure for finding the policy

After computation of A matrix, the LP formed is :

$$Max(R^T X)$$

Under Constraint :

$$AX = \alpha, \;\; X > 0$$

After the LP is solved, we have x (in X matrix solved)value for every (state,action) pair. Hence, for the policy, find the action with maximum value for a given state. Hence the policy is :

$$Policy[state] = Max(X[state, action])$$

Computation of the policy equation as given above

```
for i in range(len(possible_act)):
    state, act = possible_act[i]
    if policy[state][1] < x[i]:
        policy[state] = (act, x[i])
```

Policy Change

A different policy under exact conditions can only be produced if they have the same x value. If that is true (two actions having the same value), then different policies will be generated under the same conditions.

In this condition, finding different policies would imply changing the order of searching under actions. So if ‘SHOOT’ value is stored as maxima earlier in one policy and in the other policy, we come across the same state with the same value but ‘DODGE’ as the action, that would be stored and thus changing the policy.

Otherwise, change in policy implies change in the solution provided by the LP, thus a change in A,α or r (reward function).

This means that, the probability of the occurrence of an action, or the reward for a state or the initial state must change for a different policy to exist.

If there is difference in probabilities, the difference will be seen in the A matrix, if the difference is in α, the initial state must be different and if the difference is in r, the reward per action must be different.