MDL Assignment-2 Part-3

Jaidev Shriram (2018101012), Rohan Grover (2018101017)

April 2020

1 Introduction

The assignment aims to solve a Markov Decision Process (MDP) via linear programming. Linear programming is a method of solving MDPs in which we try to satisfy the objective function while following a set of constraints. In this assignment, we will try to maximise the vector product of the matrix r and x, where r is our reward matrix which we have defined as the step cost for every state action pair. The constraints followed are:

$$x >= 0 \tag{1}$$

$$Ax = \alpha \tag{2}$$

where the α vector is the initial probability of starting in every state. Hence, it is 0 for all elements except for the state [4, 3, 2] i.e. our starting state, where it's value is 1. The A matrix will be defined in the following sections.

2 Procedure for Making the A Matrix

The A matrix is a 2-D matrix with the following logic:

1. Each column represents a state action pair. Example: jth column may be (0, 1, 1) and NOOP or (1, 1, 1) and DODGE.
2. The rows represent all the possible states.

3. Hence, values in the matrix at index (i, j) correspond to a state that on taking an action (which corresponds to column j) reaches state i.

The above is the logic for understanding the matrix. To create the matrix we use the following logic:

1. For terminal states, for all elements such that start state and destination state is also the terminal state, then the matrix value is 1 for these corresponding elements.
2. For non terminal states, the elements represent negated probability of transition from state s, to state s' on taking action a.
3. For non terminal states where the start and end states are the same, add 1 to the corresponding element.

3 Procedure for Finding the Policy

When we solve this linear program, we get our vector x which is a row vector with elements that correspond to the state action pairs. To obtain our policy, we must determine what action to take from every state. This can be done by simply choosing the state-action pair which has the largest x value for any particular state.

For example, for state A, we have actions a1, a2. If our solved LP returns a vector with x values such as [0.1, 0.7] which correspond to (state A, action a1) and (state A, action a2) respectively, then our policy for state A would be action a2 since it had a higher x value.

4 Can There be Multiple Policies?

No. Multiple policies aren't possible for the same transitions and reward matrices. The linear program is a deterministic algorithm and will generate the same x vectors if A, α , r are the same. However, we can have multiple policies depending on how we choose our max x value. This is explained in the next section.

5 Generating Other Policies

To generate other policies, we must tweak our existing vectors in some form.

5.1 Re-order the actions

As mentioned earlier, we can change our policy by changing the way we choose our best action from our x values.

If we choose the first highest or the last highest or any one of the highest x values for the same state and same maximum value, policy can vary. For instance, if SHOOT and DODGE at state [1, 1, 1] both have x values 1, 1. Then, both actions are equally good and either may be chosen.

If we stick to choosing the first highest x value, then changing the order of actions will also have an impact. Like in the previous example, if DODGE was before SHOOT, then DODGE would be chosen with a first highest value method. Hence, equally good but different policies may result due to varying action order.

The ordering of actions will determine the columns in the A vector, x vector, and R vector. Hence, changing the order will also reorder the columns of these vectors. For the sake of explanation, let x be a row vector that has not been transposed yet. Now, the jth column in A, x, and R all correspond to the same (state, action) pair. Hence, reordering the actions, will move the jth column to another index. Hence, these vectors will also change.

5.2 Changing Step Cost

A change that can be made is using a non-uniform step cost. Hence, we can reduce the cost of being in a low stamina state for instance, which may help our agent be more risky when reaching the terminal state.

This change essentially changes the external world.

5.3 Changing the Initial Probabilities

At the moment, it is guaranteed that the agent starts at [4, 3, 2] but if we let the start state be one of many - with attached probabilities, our answer will change.

For instance, [4, 0, 2] and [4, 3, 2] could have equal probability of starting in a scenario where Lero may or may not remember to take his arrows to the fight.

This is once again, changing the outer world and introducing some new logic into the problem.

5.4 Changing the Transition Probabilities

This is an obvious way to change policies but this also means changing the outer world which this agent is expected to navigate.