MDP Assignment

# Assignment 1

## Reflect on how versions 1 and 2 can deal with this case. See course slides for detail on the MDP.

A diagram of a number of squares

Description automatically generated with medium confidence

#### States:

Where Color1, Color2, and Color3 are discrete variables that represent 3 colors.

#### Actions

Where are discrete actions for each state.

#### Rewards

Where -1,0,1 are discrete rewards.

### Version1

Version 1 is well designed for This MDP scenario because the **environment** defines the stochastic model for the next **state** and **reward** based on the **current state** and **action**.

This means that the table in Figure defines the probability of getting to any **state** and receiving the **reward**.

### Version2

Version 2 is not that suitable for this scenario. In such a version we have

Which represents the probability of the **next state** based on the **current state** and **taken action**. To calculate these probabilities based on Table:

In this version the **reward** is defined as a deterministic function:

But we can’t define this **reward** in a deterministic way for this **environment**.

Hence this case couldn’t be represented with this version.

## Implement a generic stochastic MDP in Python (version 2).

A **stochastic Markov Decision Process (MDP)** is a mathematical model used to optimize decision-making in environments where outcomes depend on actions with some uncertainty.

MDP isxs defined in mathematical terms by a set of states , a set of actions , and ) (or transition probabilities ), and a reward function )).

Here, ), denotes the probability of transitioning to state from state after taking action , and getting reward .

The objective in an MDP is to find a policy, a function from states to actions, that maximizes the Value function() with policy . the Value function() is typically calculated as the expected return . The return is sum of all the future rewards, formalized as:

Where is the discount factor, which shows how we should weight the future rewards. The nearest rewards will be more significant for the model.

The transition probabilities determine the likelihood of moving from one state to another given a specific action, and rewards provide a numerical value for each state transition. The objective of an MDP is to find a policy that maximizes the expected sum of rewards over time:

## Initialization and Configuration

The **MDP** class is initialized with five main components: states, actions, transitions, rewards, and current state.

A screenshot of a computer program

Description automatically generated

1. **States**: Represents the different conditions that can exist within the environment as a list. In this case:
2. **Actions**: Specifies possible actions that can be taken in each state:
3. **Transitions**: Defines the probability of moving from one state to another given a specific action:
4. **Rewards**: Maps each state-action-next state triplet to a numerical reward, quantifying the immediate value of a transition:

## Methods

* **reset()**: Resets the MDP to a random initial state.

A black background with white text

Description automatically generated

* **step(action)**: Takes an action, determines the next state from current state based on the transition probabilities, updates the current state, and returns the new state, reward, and whether the state is terminal.

A computer screen shot of a program

Description automatically generated

* **get\_available\_actions()**: Returns a list of valid actions that can be taken from the current state.

A black background with white text

Description automatically generated