Intro to Reinforcement Learning

Fall 2023

Dr. Dongchul Kim
Department of Computer Science
UTRGV

Markov Decision Process

Markov Decision Process

Markov Decision Process (MDP) is a mathematical framework to model a decision making problem.

The concept of MDP might appear intricate if approached immediately.

Hence, let's commence with a gradual introduction using a basic model.

Initially, we will elaborate on the **Markov Process**, followed by the **Markov Reward Process**. Lastly, we will delve into the **Markov Decision Process**.

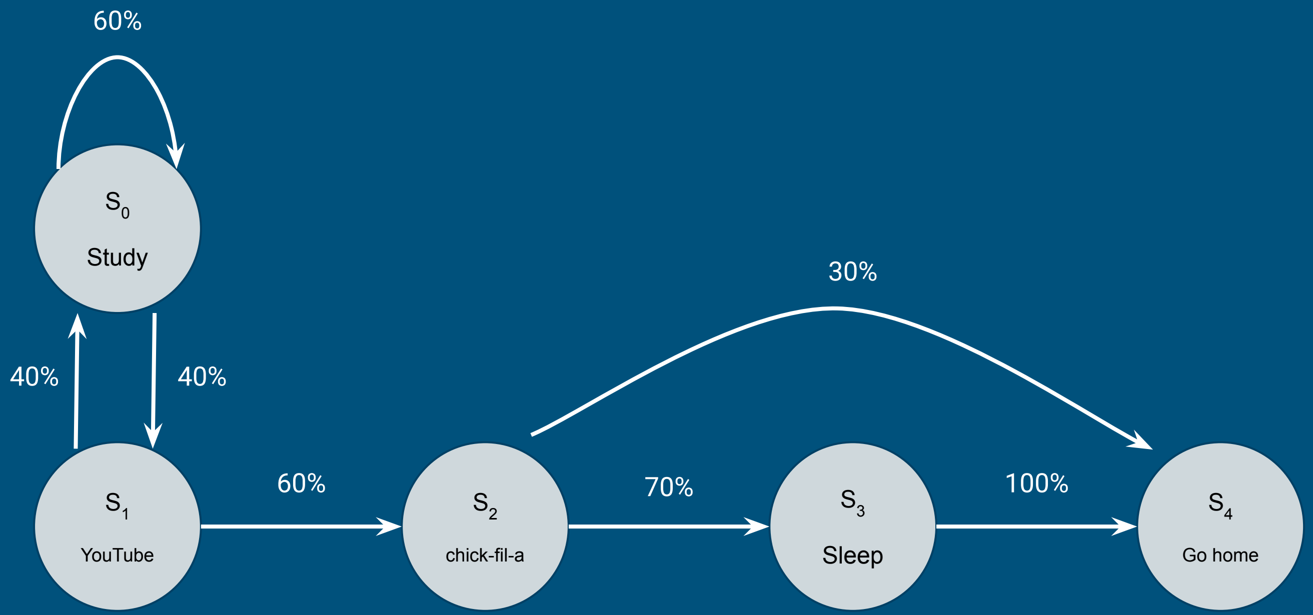
Markov Process

Markov Reward Process

Markov Decision Process

MP

Markov Process



Markov Process

The diagram represents a model of the final exam preparation scenario we discussed earlier using a Markov process. In this process, there are a total of five possible states a student can be in: studying, watching YouTube, sleeping, eating, and going home.

The process always starts in the studying state, and after a certain period of time (e.g. 30 min), it transitions to the next state. Going home is the exit state, after which the Markov process terminates.

Markov Process

A Markov process is defined as a stochastic process in which the next state depends solely on the current state and not on any previous states.

The probability of transitioning from one state to another is predefined and follows a certain probability distribution.

We call it **transition probability matrix P** that describes the probability of transitioning from one state to another.

Markov Process

We can represent the Markov Process for this scenario as $MP = (S, P)$, where $S=\{s_0, s_1, s_2, s_3, s_4\}$, and P is a matrix where $P_{ss'}$ represents the probability of transitioning from state s to state s'.

$P_{ss'}$ is a conditional probability and can be expressed as $P[S_{t+1}=s' | S_t=s]$.

The sum of probabilities for all possible transitions from a given state must always equal 1.

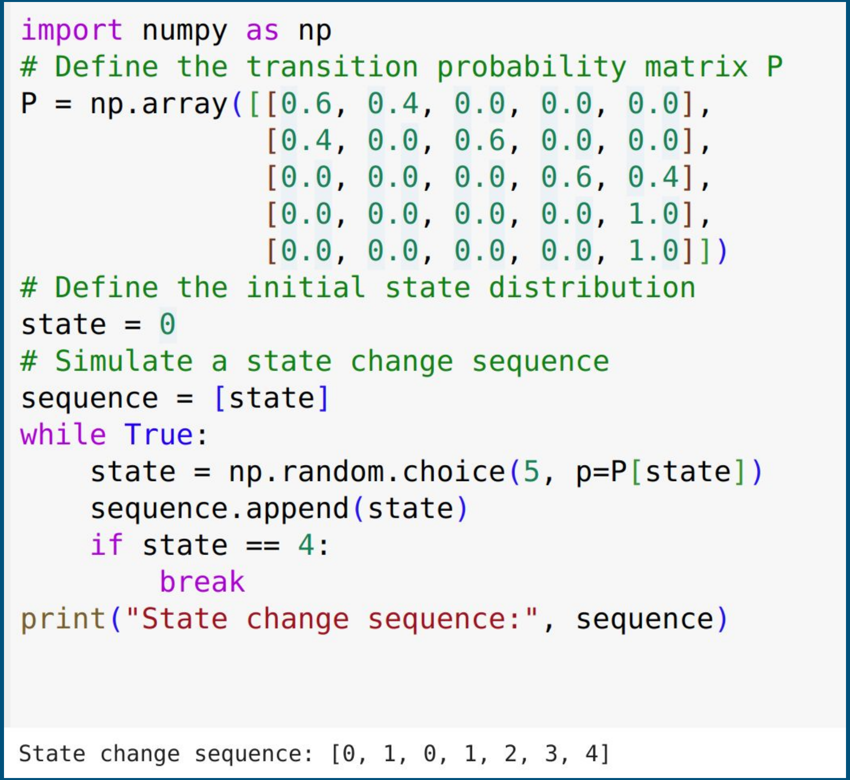
Transition Probability Matrix

	study	youtube	chick-fil-a	sleep	home
study	0.6	0.4			
youtube	0.4		0.6		
chick-fil-a				0.6	0.4
sleep					1.0
home					1.0

Implementation

```
import numpy as np
# Define the transition probability matrix P
P = np.array([[0.6, 0.4, 0.0, 0.0, 0.0],
              [0.4, 0.0, 0.6, 0.0, 0.0],
              [0.0, 0.0, 0.0, 0.6, 0.4],
              [0.0, 0.0, 0.0, 0.0, 1.0],
              [0.0, 0.0, 0.0, 0.0, 1.0]])
# Define the initial state distribution
state = 0
# Simulate a state change sequence
sequence = [state]
while True:
    state = np.random.choice(5, p=P[state])
    sequence.append(state)
    if state == 4:
        break
print("State change sequence:", sequence)
```

Outputs



```python
import numpy as np
# Define the transition probability matrix P
P = np.array([[0.6, 0.4, 0.0, 0.0, 0.0],
              [0.4, 0.0, 0.6, 0.0, 0.0],
              [0.0, 0.0, 0.0, 0.6, 0.4],
              [0.0, 0.0, 0.0, 0.0, 1.0],
              [0.0, 0.0, 0.0, 0.0, 1.0]])
# Define the initial state distribution
state = 0
# Simulate a state change sequence
sequence = [state]
while True:
    state = np.random.choice(5, p=P[state])
    sequence.append(state)
    if state == 4:
        break
print("State change sequence:", sequence)
```

```
State change sequence: [0, 1, 0, 1, 2, 3, 4]
```

Markov Property

What's the origin of the term "Markov process"? Well, the name Markov carries significant meaning. This stems from the fact that every state within a Markov process adheres to the **Markov property** defined as

$$P[s_{t+1} | s_t] = P[s_{t+1} | s_1, s_2, \ldots, s_t]$$

This property can be summed up as follows: "The future is solely influenced by the present." In essence, this signifies that when calculating the probability of the next state, the outcome is contingent on the current state. Any prior states experienced do not impact the probability of the next state.

MRP

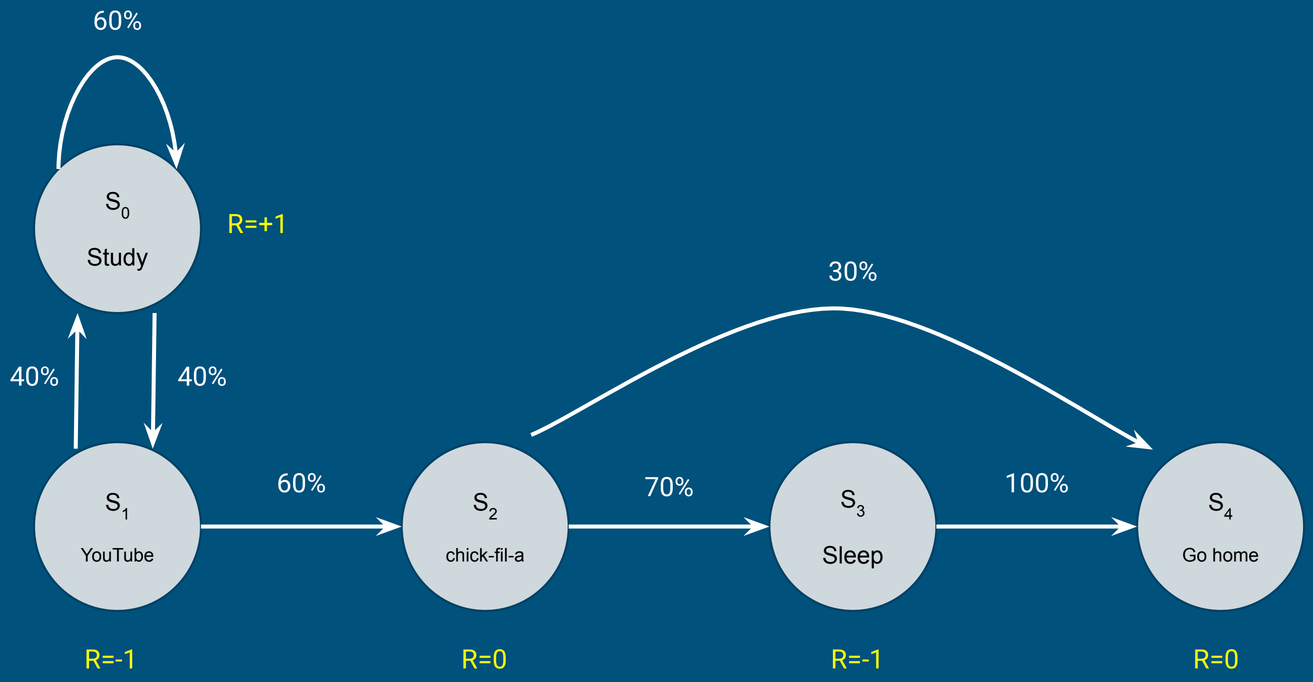
Markov Reward Process

The Markov Reward Process (MRP) is a variation of the Markov process where rewards are added to the states.

The MRP is represented by the tuple (S, P, R, γ) .

The reward R represents the amount of payoff received when transitioning to a certain state s.

Markov Reward Process



Markov Reward Process

Reward

The reward, denoted as R, signifies the reward granted upon reaching a specific state, s. Mathematically, it can be represented as:

$$R = E[R_t | S_t = s]$$

This calculation of the expected reward value is necessary due to potential minor variations in the reward even when in the same state.

In the context of our illustration, we previously considered a scenario where the reward was constant.

Markov Reward Process

The discount factor γ is a value between 0 and 1. It helps to distinguish the value of immediate rewards from those obtained in the distant future by multiplying the value of the expected future reward by γ several times. This makes future rewards less valuable compared to immediate rewards.

For a clearer comprehension of the discount factor, it's important to grasp the concept of "Return," which signifies the accumulation of future rewards.

Markov Reward Process

In the context of MRP, each change in the state results in the acquisition of a corresponding reward. Specifically, when transitioning from initial state S_0 to subsequent states S_1, S_2, ..., and eventually reaching end state T, the associated rewards are denoted as R_0, R_1, ..., R_T respectively.

$$S_0, R_0, S_1, R_1, S_2, R_2, \ldots, S_T, R_T$$

Markov Reward Process

In the realm of reinforcement learning, such a sequence of states and rewards constitutes an "episode."

Within this framework, we can compute the "return" denoted as Gt, which signifies the cumulative sum of anticipated rewards to be obtained from time t onwards. This notion can be mathematically represented as follows:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots$$

Markov Reward Process

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

where R_{t+1} is the immediate reward received after taking an action at time t , γ is the discount factor between 0 and 1 that determines the relative importance of immediate and future rewards, and γ^k is the k_{th} power of γ .

Discount Factor γ (gamma)

- γ is set to a value between 0 and 1.
- Mathematical convenience
- Reflecting uncertainty about the future
- Reflecting people's preferences

Value of state (State Value Function)

In MRP, what states are valuable and how can we define their value? The value of a state is determined by how much reward it is expected to receive in the future, regardless of the rewards received prior to reaching that state.

To obtain the value of a specific state, we need to calculate its *return*. However, since the state transitions in MRP are stochastic, the return will be different for each episode. So how do we evaluate the value of a state?

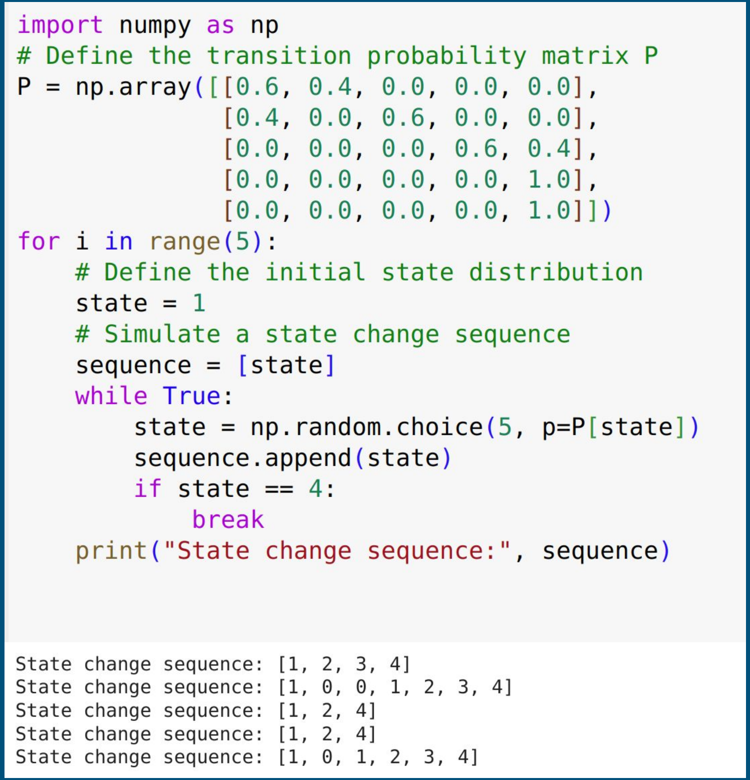
The answer is through expected values. But before diving into that, let's first understand how we sample episodes.

Sampling

In reinforcement learning, sampling is a crucial technique used to obtain certain values, and this method is known as the Monte Carlo approach. To demonstrate this approach, let's consider the MRP example for the final exam preparation that we discussed earlier and sample several episodes from it.

For instance, let's assume that the current state is watching YouTube, and then we can obtain the following results through sampling.

Sampling



```python
import numpy as np
# Define the transition probability matrix P
P = np.array([[0.6, 0.4, 0.0, 0.0, 0.0],
              [0.4, 0.0, 0.6, 0.0, 0.0],
              [0.0, 0.0, 0.0, 0.6, 0.4],
              [0.0, 0.0, 0.0, 0.0, 1.0],
              [0.0, 0.0, 0.0, 0.0, 1.0]])
for i in range(5):
    # Define the initial state distribution
    state = 1
    # Simulate a state change sequence
    sequence = [state]
    while True:
        state = np.random.choice(5, p=P[state])
        sequence.append(state)
        if state == 4:
            break
    print("State change sequence:", sequence)
```

```
State change sequence: [1, 2, 3, 4]
State change sequence: [1, 0, 0, 1, 2, 3, 4]
State change sequence: [1, 2, 4]
State change sequence: [1, 2, 4]
State change sequence: [1, 0, 1, 2, 3, 4]
```

Episode 1

$$\begin{tikzcd} S_1 \arrow[r] & S_2 \arrow[r] & S_3 \arrow[r] & S_4 \end{tikzcd}$$

Episode 2

$$\begin{tikzcd} S_1 \arrow[r] & S_0 \arrow[r] & S_0 \arrow[r] & S_1 \arrow[r] & S_2 \arrow[r] & S_3 \arrow[r] & S_4 \end{tikzcd}$$

Episode 3

$$\begin{tikzcd} S_1 \arrow[r] & S_2 \arrow[r] & S_4 \end{tikzcd}$$

Episode 4

$$\begin{tikzcd} S_1 \arrow[r] & S_2 \arrow[r] & S_4 \end{tikzcd}$$

Episode 5

$$\begin{tikzcd} S_1 \arrow[r] & S_0 \arrow[r] & S_1 \arrow[r] & S_2 \arrow[r] & S_3 \arrow[r] & S_4 \end{tikzcd}$$

State Value Function

To determine the value of a particular state, we need to define a state value function. This function takes the state as input and gives the output as the expected value of the return. As mentioned earlier, since the return varies for each episode, we use the expected value to calculate the state value.

The formula for state value function is $v(s)=E(G_t \mid S_t=s)$, where $v(s)$ is the value of state s, G_t is the return obtained after time t, and $S_t=s$ means that the current state is s.

For example, let's find the value of state s_1 (YouTube).

Episode 1

$$\begin{tikzcd}
S_1 \arrow[r] & S_2 \arrow[r] & S_3 \arrow[r] & S_4 \\
-1 & +0 * \gamma & -1 * \gamma^2 & +0 * \gamma^3
\end{tikzcd}$$

Episode 2

$$\begin{tikzcd}
S_1 \arrow[r] & S_0 \arrow[r] & S_0 \arrow[r] & S_1 \arrow[r] & S_2 \arrow[r] & S_3 \arrow[r] & S_4 \\
-1 & +1 * \gamma & +1 * \gamma^2 & -1 * \gamma^3 & +0 * \gamma^4 & -1 * \gamma^5 & +0 * \gamma^6
\end{tikzcd}$$

Episode 3

$$\begin{tikzcd}
S_1 \arrow[r] & S_2 \arrow[r] & S_4 \\
-1 & +0 * \gamma & +0 * \gamma^2
\end{tikzcd}$$

Episode 4

$$\begin{tikzcd}
S_1 \arrow[r] & S_2 \arrow[r] & S_4 \\
-1 & +0 * \gamma & +0 * \gamma^2
\end{tikzcd}$$

Episode 5

$$\begin{tikzcd}
S_1 \arrow[r] & S_0 \arrow[r] & S_1 \arrow[r] & S_2 \arrow[r] & S_3 \arrow[r] & S_4 \\
-1 & +1 * \gamma & +0 * \gamma^2 & -1 * \gamma^3 & 0 * \gamma^4 & +0 * \gamma^5
\end{tikzcd}$$

Episode 1

$$\begin{tikzcd} S_1 \arrow[r] & S_2 \arrow[r] & S_3 \arrow[r] & S_4 \end{tikzcd}$$

$-1 \quad +0 * 0.9 \quad -1 * 0.9^2 \quad +0 * 0.9^3 \quad = -1.81$

Episode 2

$$\begin{tikzcd} S_1 \arrow[r] & S_0 \arrow[r] & S_0 \arrow[r] & S_1 \arrow[r] & S_2 \arrow[r] & S_3 \arrow[r] & S_4 \end{tikzcd}$$

$-1 \quad +1 * 0.9 \quad +1 * 0.9^2 \quad -1 * 0.9^3 \quad +0 * 0.9^4 \quad -1 * 0.9^5 \quad +0 * 0.9^6 \quad = -0.6095$

Episode 3

$$\begin{tikzcd} S_1 \arrow[r] & S_2 \arrow[r] & S_4 \end{tikzcd}$$

$-1 \quad +0 * 0.9 \quad +0 * 0.9^2 \quad = -1$

Episode 4

$$\begin{tikzcd} S_1 \arrow[r] & S_2 \arrow[r] & S_4 \end{tikzcd}$$

$-1 \quad +0 * 0.9 \quad +0 * 0.9^2 \quad = -1$

Episode 5

$$\begin{tikzcd} S_1 \arrow[r] & S_0 \arrow[r] & S_1 \arrow[r] & S_2 \arrow[r] & S_3 \arrow[r] & S_4 \end{tikzcd}$$

$-1 \quad +1 * 0.9 \quad -1 * 0.9^2 \quad +0 * 0.9^3 \quad -1 * 0.9^4 \quad +0 * 0.9^5 \quad = -1.5661$

MDP

Markov Decision Process

MDP stands for Markov Decision Process, which builds on Markov Processes (MP) and Markov Reward Processes (MRP) by adding the element of **action**.

While in MP and MRP the next state is determined by the transition probability, MDP involves an agent that selects an action.

MDP

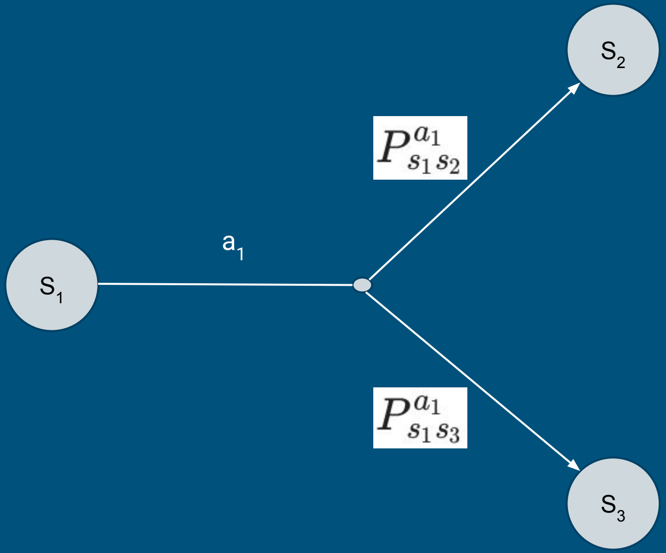
The MDP is defined by the tuple (S, A, P, R, gamma), where A is a set of possible actions.

P is the transition probability matrix, which differs from that of MP and MRP in that it incorporates the action component.

Specifically, $P^a_{ss'}$ represents the probability that the next state will be *s'* given the current state *s* and the action *a* chosen by the agent.

$$P[S_{t+1} = s' | S_t = s, A_t = a]$$

MDP

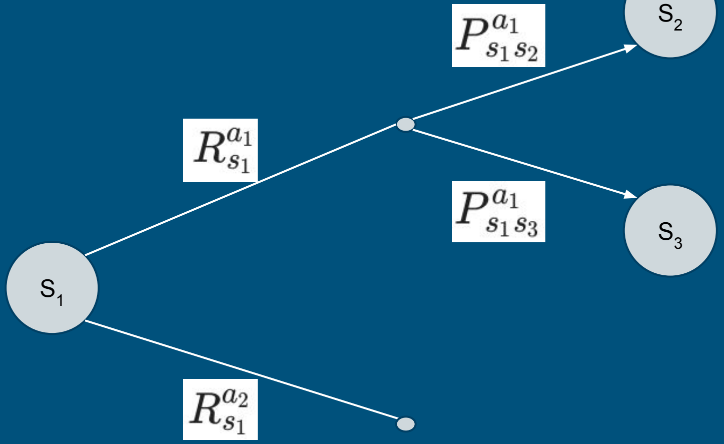


MDP

In MDP, the reward associated with taking a specific action in a given state is defined as R^a_s, which represents the expected reward received at time $t+1$, given that the agent is in state s and chooses action a.

The use of expected value is important because the reward can vary even if the same action is taken in the same state.

MDP



Policy Function

A policy, or policy function, is a function that determines which action the agent should select in each state.

$$\pi(a|s)=P[A_t=a|S_t=s]$$

For instance, suppose that in state s_3 , the possible actions are a_0 , a_1 , and a_2 . The policy function determines the probabilities assigned to each of these actions. As an example, $\pi(a_0|s_3)=0.2$, $\pi(a_1|s_3)=0.5$, and $\pi(a_2|s_3)=0.3$, and the sum of the probabilities must always be 1

State Value Function

The state value function in MDP is very similar to the value function in MRP, with the main difference being the incorporation of the policy.

In MDP, the action is determined based on the policy instead of the transition probability matrix.

$$v_\pi(s) = E_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2_{r+3} + \ldots . | S_t = s]$$
$$= E_\pi[G_t | S_t = s]$$

Action Value Function

The action value function, which is a critical concept in MDP, assesses the value of a specific action that is taken in a particular state.

$$q_\pi(S, A) = E_\pi[G_t | S_t = s, A_t = a]$$

This equation expresses the expected value of the return you can expect to receive when you choose action *a* from state *s* and then follow the policy π .

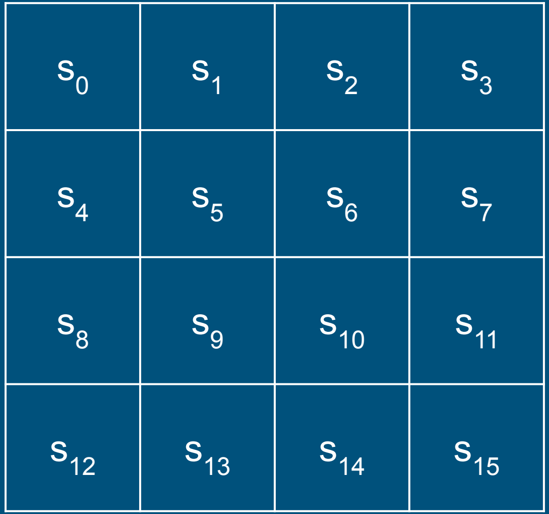
Prediction and Control

Prediction and Control

Given MDP, there are two tasks.

1. **Prediction**: Problem to evaluate state values given policy
2. **Control**: Problem to find optimal policy, π^*

Grid World


s_0 s_1 s_2 s_3
s_4 s_5 s_6 s_7
s_8 s_9 s_{10} s_{11}
s_{12} s_{13} s_{14} s_{15}

Start: s_0

End: s_{15}

Reward: -1 per step

Policy: Uniform random

Prediction

$s_{11} \text{->} s_{15}$ return = -1

$s_{11} \text{->} s_{10} \text{->} s_{14} \text{->} s_{15}$ return = -3

$s_{11} \text{->} s_7 \text{->} s_6 \text{->} s_{10} \text{->} s_{11} \text{->} s_{15}$ return = -5

$s_{11} \text{->} s_7 \text{->} s_3 \text{->} s_2 \text{->} s_3 \text{->} s_2 \text{->} s_6 \text{->} s_{10} \text{->} s_9 \text{->} s_{13} \text{->} s_{14} \text{->} s_{10} \text{->} s_{14} \text{->} s_{15}$ return = -13

Control

Optimal policy: π^* is the policy which return is largest.

The optimal value function, denoted as v*, is the value function that emerges when adhering to the optimal policy.