

Week 1 Report Submissions

Anonymous authors

Paper under double-blind review

Abstract

1 This report focuses on the Markov Decision Process (MDP), the differ-
 2 ence between Reinforcement Learning (RL) and Supervised Learning, the
 3 real-world application of Reinforcement Learning, and the objectives of
 4 Reinforcement Learning.

5 1 Markov Decision Process (MDP)

6 The Markov Decision Process (MDP) is a mathematical model of the decision-making
 7 process, extensively used in the field of Reinforcement Learning (RL) [Contributors \(2024\)](#). It
 8 formally describes a fully observable environment where an agent performs actions, and
 9 outcomes are partly random. In other words, Markov Decision Process (MDP) is a Markov
 10 Reward Process (MRP) with decisions.

11 It is under the assumption that the underlying structure of state transitions follow the
 12 Markov Property. The process is a decision process because it involves decision-making
 13 that influence the state transitions.

14 1.1 Markov Property

15 Markov Property describes the state in which the future is independent of the past, given
 16 the present. A state S_t is Markov if and only if

$$P[S_{t+1} | S_t] = \mathbb{P}[S_{t+1} | S_1, \dots, S_t]$$

17 When the state S_t is Markov, the state captures all relevant information from the history.
 18 Therefore, once the state is known, the history can be discarded (i.e. the state is a sufficient
 19 statistic of the future).

20 1.2 Markov Process (Markov Chain)

21 The Markov Process is a memoryless stochastic process, consisting of a sequence of random
 22 states S_1, S_2, \dots with the Markov Property explained above.

23 A Markov Process (or Markov Chain) is a tuple $\langle S, P \rangle$ where S is a (finite) set of states, P is
 24 a state transition probability matrix such that

$$P = \begin{bmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nn} \end{bmatrix}$$

25 For a Markov state s and successor state s' , the state transition probability is defined by

$$P_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s]$$

26 1.3 Markov Reward Process (MRP)

27 A Markov reward process is a Markov chain with values. The agent does not choose actions.
 28 It is just adding values to sequences of actions.

29 1.3.1 Definition of Markov Reward Process

30 MRP is a tuple $\langle S, P, R, \gamma \rangle$, where S is a finite set of states, P is a state transition probability
31 matrix,

$$P_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s]$$

32 R is a reward function,

$$R_s = \mathbb{E}[R_{t+1} \mid S_t = s]$$

33 γ is a discount factor,

$$\gamma \in [0, 1]$$

34 1.3.2 Return value of Markov Reward Process

35 The return G_t is the total discounted reward from time-step t .

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

36 The discount $\gamma \in [0, 1]$ is the present value of future rewards, and the value of receiving
37 reward R after $k + 1$ time-steps is $\gamma^k R$.

38 MRP is discounted because it represents delayed reward. For example, when applying MRP
39 on financial problem, the immediate rewards are more favourable since more interest can
40 be earned than delayed rewards.

41 1.3.3 Value function of Markov Reward Process

42 The value function $v(s)$ gives the long-term value of state s . The state value function $v(s)$ of
43 an MRP is the expected return starting from state s :

$$v(s) = \mathbb{E}[G_t \mid S_t = s]$$

44 By Bellman Equation for MRP, the value function can be decomposed into two parts:
45 immediate reward R_{t+1} , discounted value of successor state $\gamma v(S_{t+1})$, where

$$\begin{aligned} v(s) &= \mathbb{E}[G_t \mid S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \dots) \mid S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s] \\ &= R_s + \gamma \sum_{s' \in S} P_{ss'} v(s') \end{aligned}$$

46 The value function of the current step = current step rewards + sum of product of probability
47 of next step and reward of next step.

48 1.4 Markov Decision Process (MDP)

49 A Markov Decision Process (MDP) is a Markov Reward Process with the agent making
50 decisions. It is under an environment where all states are Markov (i.e. every state has the
51 Markov property).

52 1.4.1 Definition of Markov Decision Process (MDP)

53 MDP is a tuple $\langle S, A, P, R, \gamma \rangle$, where S is a finite set of states, A is a finite set of actions, P is
54 a state transition probability matrix, which is defined as $P_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s]$, R is a
55 reward function with the reward R_s at state s being $R_s = \mathbb{E}[R_{t+1} \mid S_t = s]$, and $\gamma \in [0, 1]$ is
56 a discount factor.

1.4.2 Policy of Markov Decision Process (MDP)

A policy π is a distribution over actions given states,

$$\pi(a|s) = \mathbb{P}[A_t = a \mid S_t = s]$$

The policy fully defines the behaviour of the agent (i.e. the policy is the action taken by the agent). The MDP policies depend only on the current state instead of the history, and the policies are stationary (time-independent).

Given an MDP $M = \langle S, A, P, R, \gamma \rangle$ and a policy π , the state and reward sequence S_1, R_2, S_2, \dots is a Markov reward process $\langle S, P^\pi, R^\pi, \gamma \rangle$ where

$$P_{s,s'}^\pi = \sum_{a \in A} \pi(a|s) P_{ss'}^a$$

$$R_s^\pi = \sum_{a \in A} \pi(a|s) R_s^a$$

1.4.3 Value function of Markov Decision Process (MDP)

The state-value function $v_\pi(s)$ of an MDP is the expected return starting from state s , and then following policy π :

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi [G_t \mid S_t = s] \\ &= \mathbb{E}_\pi [R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\ &= \sum_{a \in A} \pi(a|s) \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right) \end{aligned}$$

The action-value function $q_\pi(s, a)$ is the expected return starting from state s , taking action a , and then following policy π :

$$\begin{aligned} q_\pi(s, a) &= \mathbb{E}_\pi [G_t \mid S_t = s, A_t = a] \\ &= \mathbb{E}_\pi [R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a] \\ &= R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a'|s') q_\pi(s', a') \end{aligned}$$

1.4.4 Optimal value function and policy

The optimal state-value function $v_*(s)$ is the maximum value function attainable over all policies (assuming no positive cycles):

$$v_*(s) = \max_{\pi} v_\pi(s)$$

The optimal action value function $q_*(s, a)$ is the maximum action-value function attainable over all policies (assuming no positive cycles):

$$q_*(s, a) = \max_{\pi} q_\pi(s, a)$$

To find the optimal policy, a partial ordering over the policies is defined:

$$\pi \geq \pi' \text{ if } v_\pi(s) \geq v_{\pi'}(s), \forall s$$

For any Markov Decision Process: There exists an optimal policy π_* that is better than or equal to all other policies, $\pi_* \geq \pi, \forall \pi$. All optimal policies achieve the optimal value function and the optimal action-value function. An optimal policy can be found by maximizing over $q_*(s, a)$:

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \operatorname{argmax}_{a \in A} q_*(s, a) \\ 0 & \text{otherwise} \end{cases}$$

2 Differences between Reinforcement Learning and Supervised Learning

Both Reinforcement Learning and Supervised Learning are schemes of Machine Learning (ML). They are different in ultimate goals and learning signals, data and outputs, as well as use cases.

2.1 Ultimate goals and learning signals

Supervised Learning aims to learn a mapping from inputs to outputs. $f : \mathcal{X} \rightarrow \mathcal{Y}$ To achieve this, A loss function [GeeksforGeeks \(2023\)](#) is used in Supervised Learning, to measure the inaccuracy of the model's prediction, compared to the true label. For exmaple, one value function used is the mean square error:

$$\mathcal{L}(f(x), y) = (f(x) - y)^2$$

The mapping from inputs to outputs is learnt by minimizing the loss function. Training is typically done using gradient descent.

On the other hand, in Reinforcement Learning, the agent learns a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that maximizes expected return:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

through interaction with the environment.

To get the optimal policy, we want to learn the **optimal policy** π^* that maximizes expected return from any state s :

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{\pi}[G_t \mid S_t = s]$$

To do this, we define:

- **State-Value Function:**

$$V^{\pi}(s) = \mathbb{E}_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right]$$

- **Action-Value Function (Q-function):**

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right]$$

- **Bellman Optimality Equations** define recursive relationships:

$$V^*(s) = \max_a \left[R(s, a) + \gamma \sum_{s'} P(s' \mid s, a) V^*(s') \right]$$

2.2 Data and outputs

Supervised Learning includes a pre-labeled dataset (i.e. pairs of inputs and their respective correct outputs)

$$D = \{(x_i, y_i)\}_{i=1}^N$$

This requires large amount of manual effort to label the data.

The output will be a prediction of the input value (e.g., classification or regression of the input).

On the other hand, Reinforcement Learning does not include labels. It allows the agent to interact with an environment to maximize cumulative data rewards. The output returns a policy that maps from states to actions.

2.3 Use cases

Supervised learning does not require exploration since data given is fixed. It can be used in the scenario of prediction and classification. For example, in the case of spam email detection (using KNN algorithm). On the other hand, reinforcement learning requires exploration (i.e. agent must explore actions). It can be used in cases such as recommendation systems and Artificial Intelligence (AI) gaming.

3 Real-world case of Reinforcement Learning

Reinforcement Learning is widely used in the areas of robotics, healthcare, gaming and Natural Language Processing (NLP) [Scribbr \(2023\)](#). One such example is training the self-driving car. In this case, the agent is the Artificial Intelligence (AI) used in the self-driving car. It is able to take actions such as accelerating and decelerating. The environment that the agent is in is the road, including cars, pedestrians, other vehicles and traffic signals. The tuple $\langle S, A, P, R, \gamma \rangle$ represents the following respectively:

- State ($s \in S$): each state may include the current speed, the distance to the nearest car or pedestrian, the color of the traffic light, the weather conditions, the time of the day and the current condition of the car.
- Actions ($a \in A$): at each state, the agent can choose actions including accelerating, decelerating, braking and turning left or right. Such actions can change the state of the environment.
- State transition possibility matrix ($P(s'|s, a)$): The possibility matrix describes how the agent make actions. For example, if the agent take the action of braking, it leads to a decelerating state.
- Reward Function ($R(s, a)$): The reward function can be designed and guides the agent's behaviour. For example, ten points to be added for proceeding at green light while ten points to be deducted for running a red light. If it collides with another vehicle or hits a pedestrian, one hundred points will be deducted. If it successfully reaches the destination, one hundred points will be awarded.

3.1 Value function

The value function ($V^\pi(s)$) determines how good a state s is under the policy. For example, a state with clear roads nearer to the destination has high value, while a state with traffic congestion has low value.

3.2 Policy

The policy ($\pi(a|s)$) determines which action to be taken in a state. For example, the agent should decelerate when it is in a state that is close to the pedestrian (e.g., 50 meters from the pedestrian). The optimal policy (π^*) is the best driving strategy to be taken in this environment. It maximizes the total expected reward

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \right]$$

By Reinforcement Learning, the car can learn to follow traffic rules, avoid traffic incidences and reach the destination safely and efficiently.

149 4 Objective of Reinforcement Learning

150 The main objective of Reinforcement Learning (RL) is to find an optimal policy π that
151 maximizes the reward (i.e. maximizes the state-value function and action-value function):

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \right]$$

152 References

153 Wikipedia Contributors. Markov decision process, 2024. URL [https://en.wikipedia.org/
154 wiki/Markov_decision_process](https://en.wikipedia.org/wiki/Markov_decision_process).

155 GeeksforGeeks. Common loss functions in machine learning, 2023. URL [https://www.
156 geeksforgeeks.org/ml-common-loss-functions/](https://www.geeksforgeeks.org/ml-common-loss-functions/).

157 Scribbr. What are some real-life applications of reinforcement learn-
158 ing?, 2023. URL [https://www.scribbr.com/frequently-asked-questions/
159 what-are-some-real-life-applications-of-reinforcement-learning/](https://www.scribbr.com/frequently-asked-questions/what-are-some-real-life-applications-of-reinforcement-learning/).