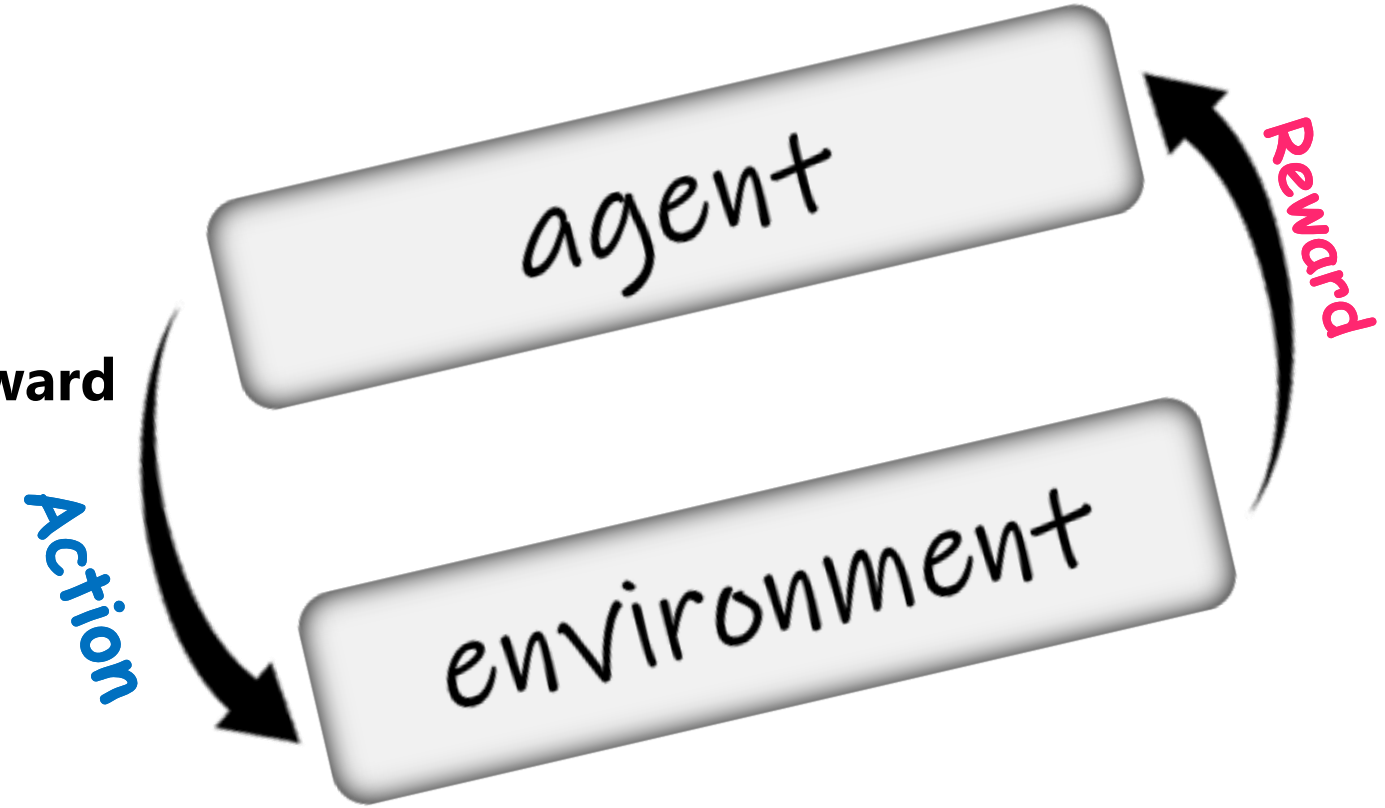
Introduction to Reinforcement learning

What is reinforcement learning?

What is reinforcement learning ?

- Relationship between Action, effect, reward and penalty
- Reward-Penalty System
- Suitable actions to maximize reward
- Making decisions sequentially
- Markov Decision Process



Formulating Basic RL problem

Environment

- Physical world for operation

State and Actions

- situation of agent & what action to take

Reward

- Feedback from environment

Policy

Method to map agents state to action

Value

- Future reward an agent would receive for action in particular state

Criteria	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Input Data	Input data is labelled.	Input data is not labelled.	Input data is not predefined.
Problem	Learn pattern of inputs and their labels.	Divide data into classes.	Find the best reward between a start and an end state.
Applications	Deal with regression and classification problems.	Deals with clustering and associative rule mining problems.	Deals with exploration and exploitation problems.
Algorithms Used	Decision trees, linear regression, K-nearest neighbors	K-means clustering, k-means clustering, agglomerative clustering	Q-learning, SARSA, Deep Q Network
Examples	Image detection, Population growth prediction	Customer segmentation, feature elicitation, targeted marketing, etc	Drive-less cars, self-navigating vacuum cleaners, etc

Sequential Decision Making

- Time really matters in RL
- The ”sequence” in which you make your decisions (moves) will decide the path you take
- Hence the final outcome.

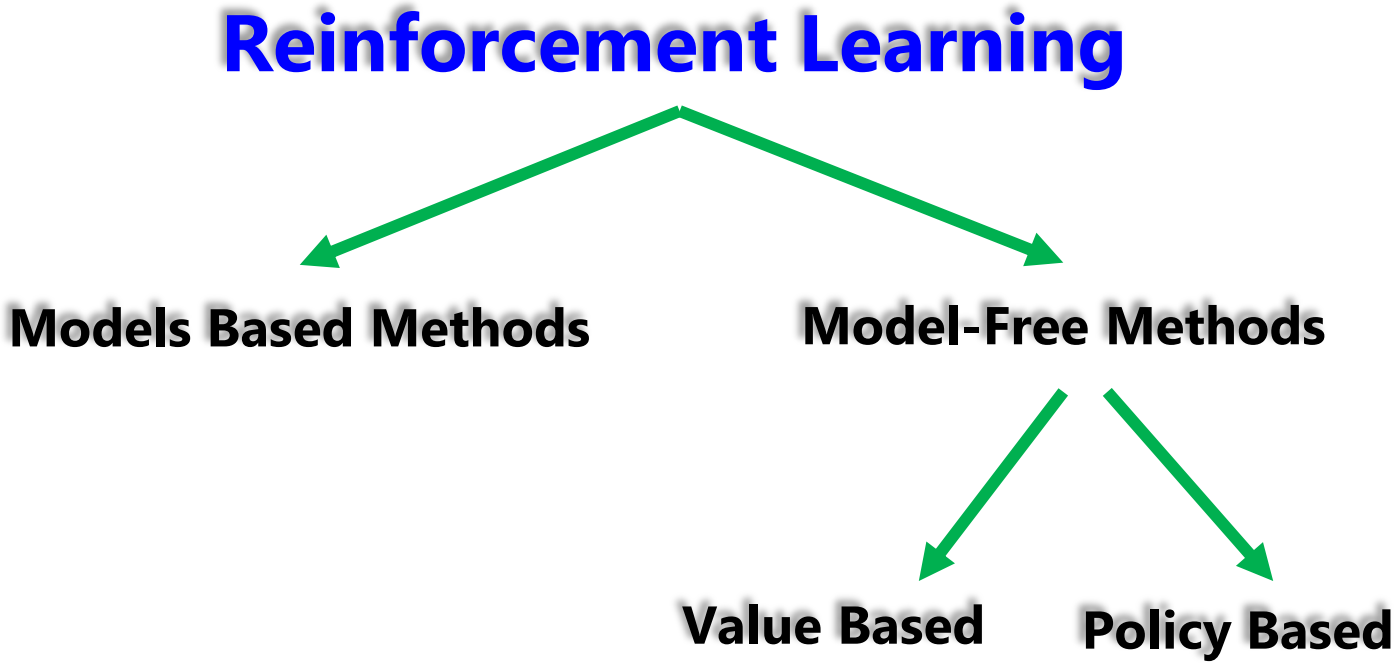
Categories in RL

- **Positive:** occurs due to a particular behavior, increases the strength and the frequency of the behavior.

- **Negative:** strengthening of behavior because a negative condition is stopped or avoided.

Introduction to Reinforcement learning

Types of Reinforcement Learning


Reinforcement Learning
Models Based Methods
Model-Free Methods
Value Based
Policy Based

Model Based Methods

- Learning Optimal Behaviour indirectly
- Learns a Model of the environment by taking action and observes the outcome
- Learns Optimal policies by interaction with the environment
- Given a state and action, we can predict the next state and reward
- Planning is the primary tool to decide the course of action
- Models & Planning
- Goal based

Model Free Methods

- Response to local actions not considered
- Used when intricate environments are difficult to model
- Can react to new unseen changes
- Relies on stored state and action pairs
- The action values are estimates of the highest return the agent can expect
- Rat in a maze

Model Based RL

Learn The Model

- I2A
- MBVE

Give The Model

- Alpha Zero

Model Free RL

Policy Optimization

- Policy Gradients
- A2C
- A3C

Value Based

- Q learning
- SARSA

Bellman equation

$$V(s) = max_a(R(s,a) + \gamma V(s^{|}))$$

V(s)	: Value for being in certain state
$V(s^{|})$	: Value for being in next state after action
$R(s,a)$	: Reward after a in s
Γ	: Discount factor

Introduction to Reinforcement learning

Applications

Applications of Q-Learning

Gaming and Artificial Intelligence

Robotics

Autonomous Vehicles

Introduction to Reinforcement learning

What is Markov Decision Process ?

A sequential Decision problem for a accessible, stochastic environment with a markovian state transition model with a reward R(S) is called a Markov Decision Process

To elaborate, the problem is fully observable and its outcomes are stochastic, it has following components

- **States S**
- **Actions A**
- **Transition Model T(S, A, $S^{|}$)**
- **Reward function R(S)**
- **Policy π**

Markov Decision Process

State

- Represented as set of tokens
- Represents every state the agent can be in

Model

- Provides action's effect in a state
- T(S,A,$S^{|}$) defines transition in state S, taking action 'A' and taking to state $S^{|}$
- For stochastic actions, P($S^{|}$|S,A) represents the probability of reaching $S^{|}$ if A is taken in S.
- Only based on current state and no on history

Markov Decision Process

Action

- A(S) defines set of actions that can be taken in S

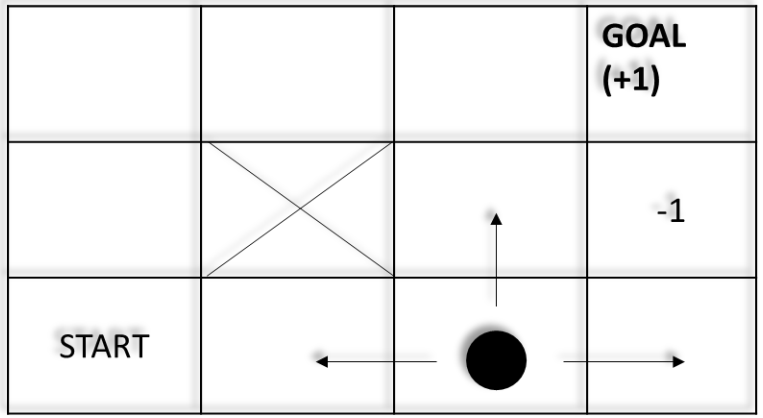
Reward

- Real-Valued reward function
- R(S) is reward for being in state S
- R(S,A) is reward for being in state S by taking an action A
- R(S,A, $S^{|}$) is reward for being in S, by taking A and ending up at $S^{|}$

Markov Decision Process

Policy

- Solution to markov decision process
- Mapping from S to A
- Indicates A to be taken on S
- From START to GOAL
- Restrictions: Block Zone & Penalty Zone
- Possible actions: UP, DOWN, LEFT, RIGHT
- Possibilities:
  - RIGHT, RIGHT, UP, UP, RIGHT
  - UP, UP, RIGHT, RIGHT, RIGHT



Markov Decision Process

Introduction to Reinforcement learning

Markov Property

"If the agent is present in the current

state S1, performs an action a1 and move

to the state s2, then the state transition

from s1 to s2 only depends on the current

state and future action and states do not

depend on past actions, rewards, or

states."

Markov Property

Introduction to Reinforcement learning

State Transition Matrix

- **The state transition probability tells us, given we are in state *s* what the probability the next state *s´* will occur.**
- **We can also define all state transitions in terms of a *State Transition Matrix P***
- **Each row tells us the transition probabilities from one state to all possible successor states.**

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

State Transition Matrix

Introduction to Reinforcement learning

Markov Process

- **Markov Process is a memoryless process with a sequence of random states S_1, S_2, ..., S_t that uses the Markov Property.**
- **Markov process is also known as Markov chain**
- **which has a tuple (S, P) on state S and transition function P.**
- **These two components (S and P) can define the dynamics of the system.**

Markov Process

Introduction to Reinforcement learning

Markov Reward Process

- **A *Markov Reward Process* is a Markov chain with reward values.**
- **Our goal is to maximize the *return*. The return G_t is the total discount reward from time-step *t*.**

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

Markov Reward Process tuple is $\langle S, P, R, \gamma \rangle$
: S is a finite set of states
: P is a state transition probability matrix, $P_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s]$
: R is a reward function, $R_s = [R_{t+1} | S_t = s]$
: γ is a discount factor, $\gamma \in [0, 1]$

Markov Reward Process

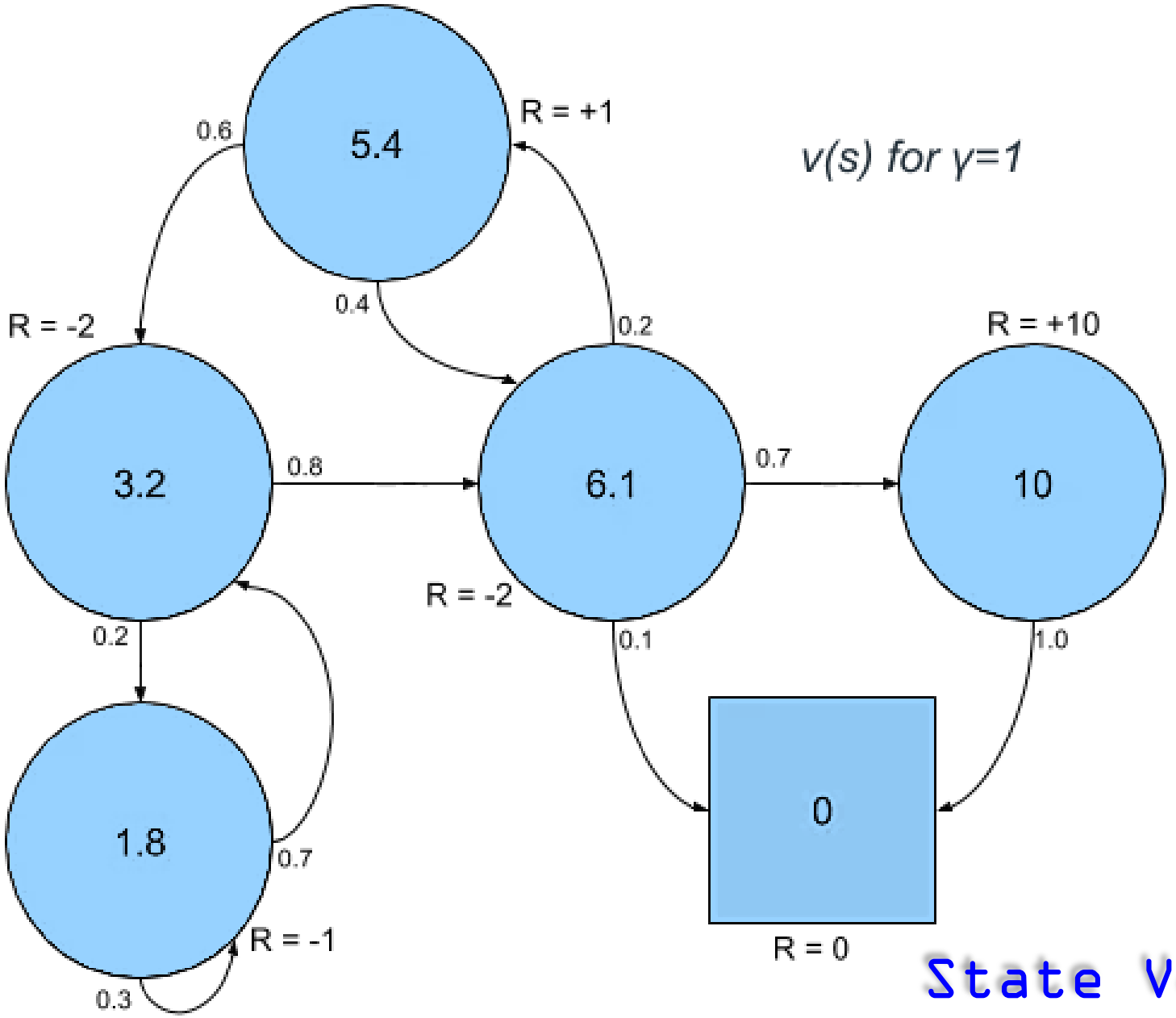
Introduction to Reinforcement learning

State Value Function

- ***State Value Function v(s)*: gives the long-term value of state *s*. It is the expected return starting from state *s***

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

State Value Function



State Value Function

Bellman Equation

- **One of the essential building blocks of reinforcement learning is the Bellman equation.**
- **The main objective of reinforcement learning is to maximize the long-term reward.**
- **The equation shows us what long-term gain we can anticipate**

Bellman Equation

$$V(s)=\max_a(R(s,a)+\gamma V(s'))$$

The equation has three parts –

- The max function, which selects the action that maximizes the reward (max a)
- The discount factor is a hyperparameter that can be modified to highlight the long-term benefit or to have the model focus on low-hanging fruits and promote the best short-term solution. (gamma)
- The function that computes the reward based on the selected action and the current state (R(s, a))

Bellman Optimality Equation

- **The optimal action-value function q∗ satisfies the fundamental property of the optimal action-value function.**
- **this equation can be used to find optimal q∗ in order to find optimal policy π**
- **Reinforcement learning algorithm can find the action *a* that maximizes *q∗(s, a).***

$$q_*(s,a) = E\left[R_{t+1} + \gamma \max_{a'} q_*(s',a')\right]$$

Bellman Optimality Equation

Partially Observable Markov Decision Process (POMDP)

- **It is a mathematical model used to describe an AI decision-making problem in which the agent does not have complete information about the environment.**
- **The agent must use its observations and past experience to make decisions that will maximize its expected reward.**

POMDP

Belief State

- **Belief State is a probability distribution over the possible states must be maintained by the agent at all times. The belief state allows the agent to act optimally in this uncertain environment under the POMDP model.**

Belief State

The Power of Dynamic Programming

What is Dynamic Programming?

1

Definition & Overview

Dynamic programming is a problem-solving technique that involves breaking down large problems into smaller subproblems, solving each subproblem only once, and storing the results for future use.

2

Principles

Dynamic programming relies on two principles: overlapping subproblems and optimal substructure. These principles enable efficient solutions to complex problems by reusing previously computed results.

3

Techniques & Approaches

Two common techniques for implementing dynamic programming are memoization and tabulation. Memoization stores the results of each subproblem in memory, while tabulation stores results in a table or array. Both can be used to solve a wide variety of optimization problems.

4

Applications

Dynamic programming has widespread applications, including solving the Fibonacci sequence, Knapsack problem, Traveling Salesman problem, Longest Common Subsequence, and Edit Distance.