Reinforcement Learning

If something reinforces an idea or an opinion, it provides more proof or support for it and makes it seem true.

It is intent on preserving what ought to be preserved, perfecting what needs to be perfected, and pruning wherever we find practices that ought to be prohibited.

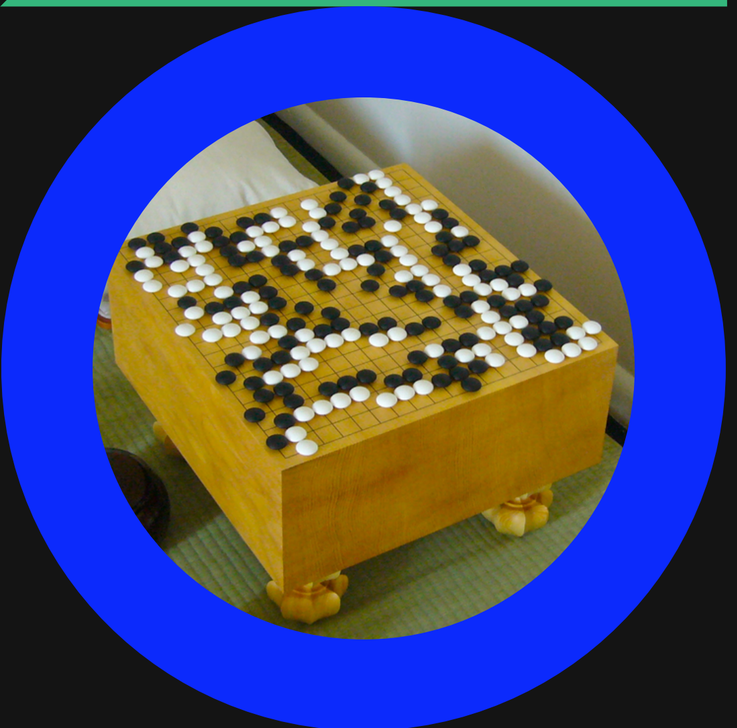
WHAT IS REINFORCEMENT?

REINFORCEMENT LEARNING

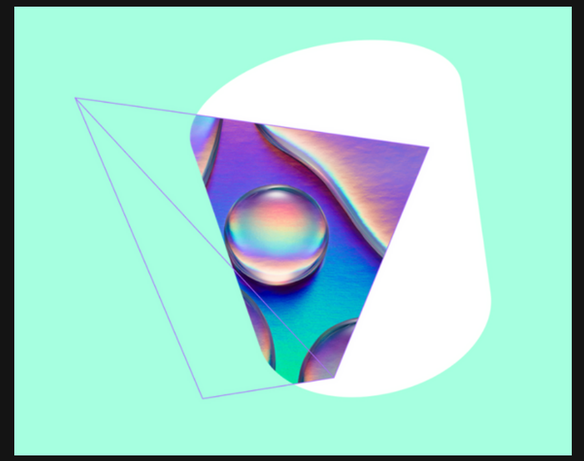
- In RL we are not provided data from which the model learns features.
- Instead, we have an object (called an agent) which is allowed to explore an environment, and it learns the optimal path.
- It generates its own data
- RL is most commonly used to solve a different class of real-world problems, such as a Control task or Decision task, where you operate a system that interacts with the real world.

COOL USES

- Can be succesfully trained to perform very well in complex games such as chess.
- A very successful model is the AlphaGo, trained by DeepMind, a subsidiary of Google.
- In 2015, it became the first model to beat a professional Go player.



Link to the AlphaGo website

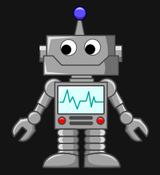


Trial and Error

- As the agent interacts with the environment, it takes several decisions.
- It recieves a feedback from the environment, and improves on these on a trial and error basis.
- Take the primitive example of a baby, which can either touch fire or drink milk, and learns via feedback.

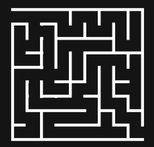
ONTO SOME TECHNICAL TERMS

- One needs to first structure the problem as an Markov Decision Process(MDP)
- It consists of 5 components:



Agent

- It is the system that you operate eg. the robot.
- It is the model you build and train.



Environment

- The real world environment that the system interacts with.
- Contains all external factors, such as wind, friction, etc.

State

- This represents the current ‘state of the world’ at any point.
- State defenition should be self contained.
- No history of states should be required to describe the current state
- There could be finite or infinite states

Action

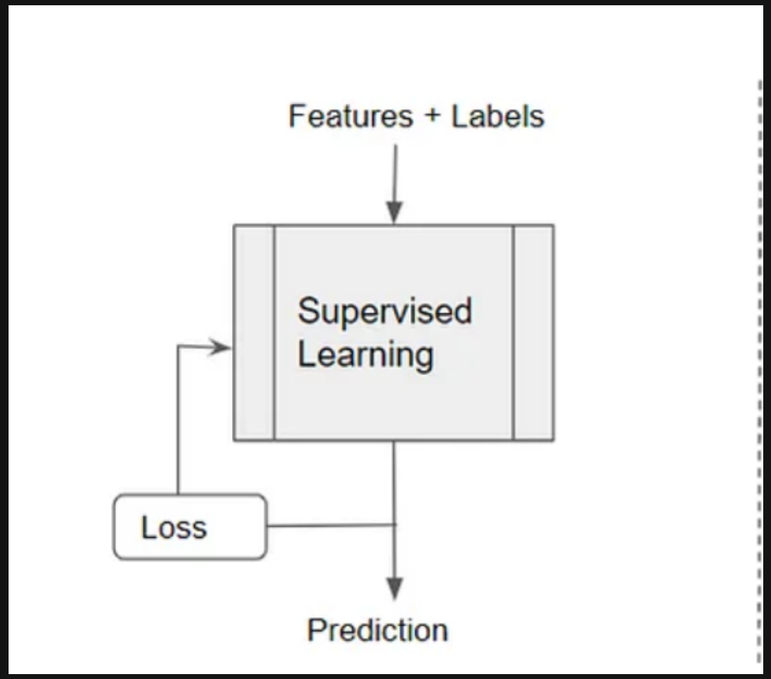
- These are what the robot takes to interact with the environment.
- There could be a finite or infinite set of possible actions.

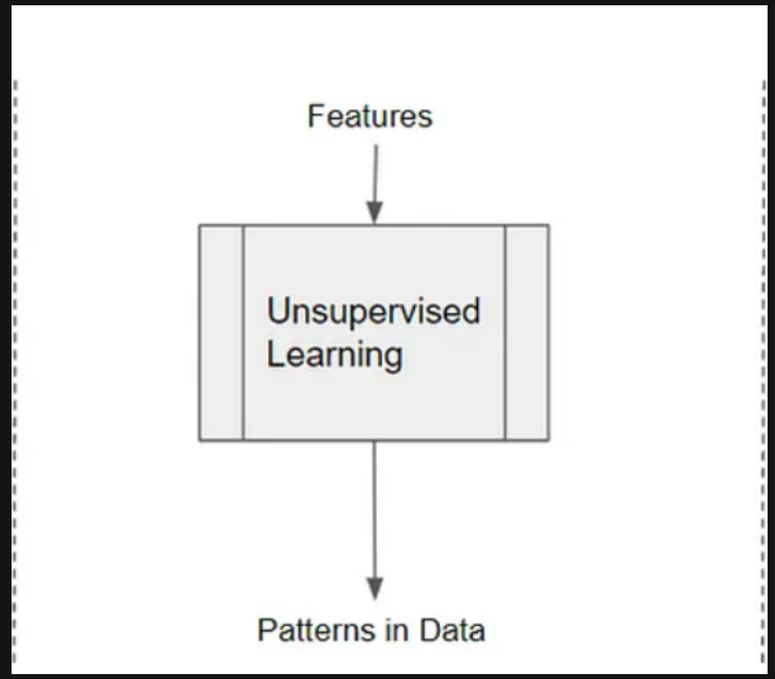
Reward

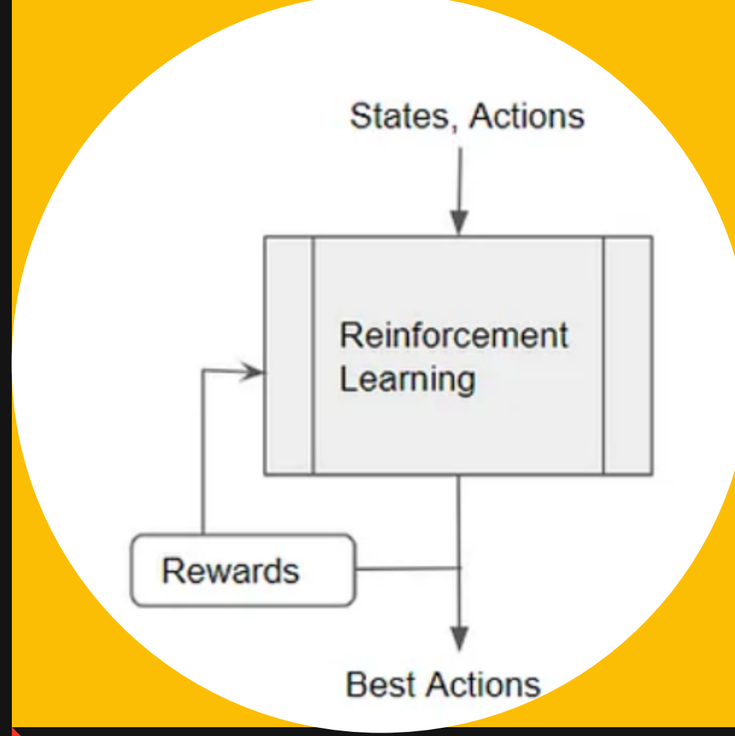
- It is a feedback given by the environment as to how good the action taken was.
- Favorable action leads to highly positive rewards.

ENVIRONMENT

- One of the most important component of an RL model.
- Gives feedback based on a certain action which influences further decision making.







Importance of the rewards

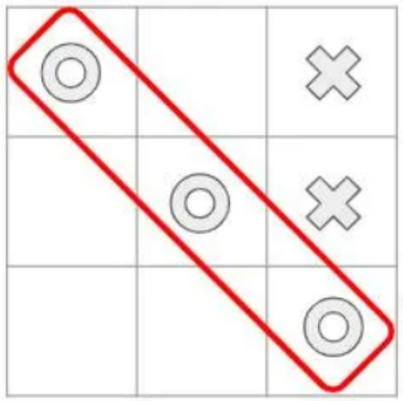
- The most important thing when it comes to RL is the environment, and the rewards that it 'generates' for an action
- It is these rewards that indicate how good or bad a particular decision was, aand helps the agent optimize the same
- Thus, a good amount of emphasis needs to be given to the magnitudes and signs of rewards coded into the environment

- Luckily, there exists a lot of environments on which we simply train our agent, and thus this does not fall within our purview
- We will study further assuming that the environment 'magically' gives out the perfect reward.
- Thus, all we have to do is train the agent to take appropriate decisions based on these rewards.

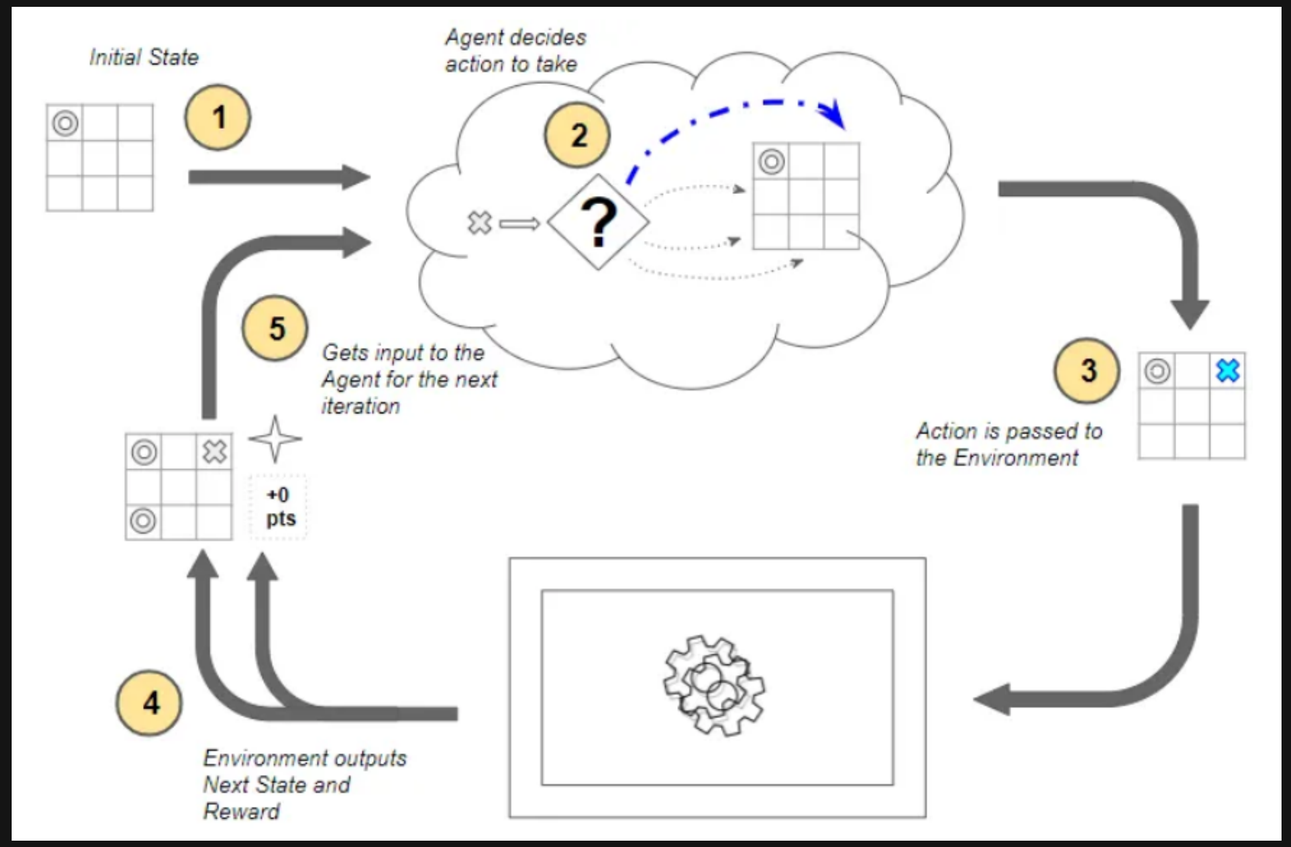
OPERATION OF AN MDP

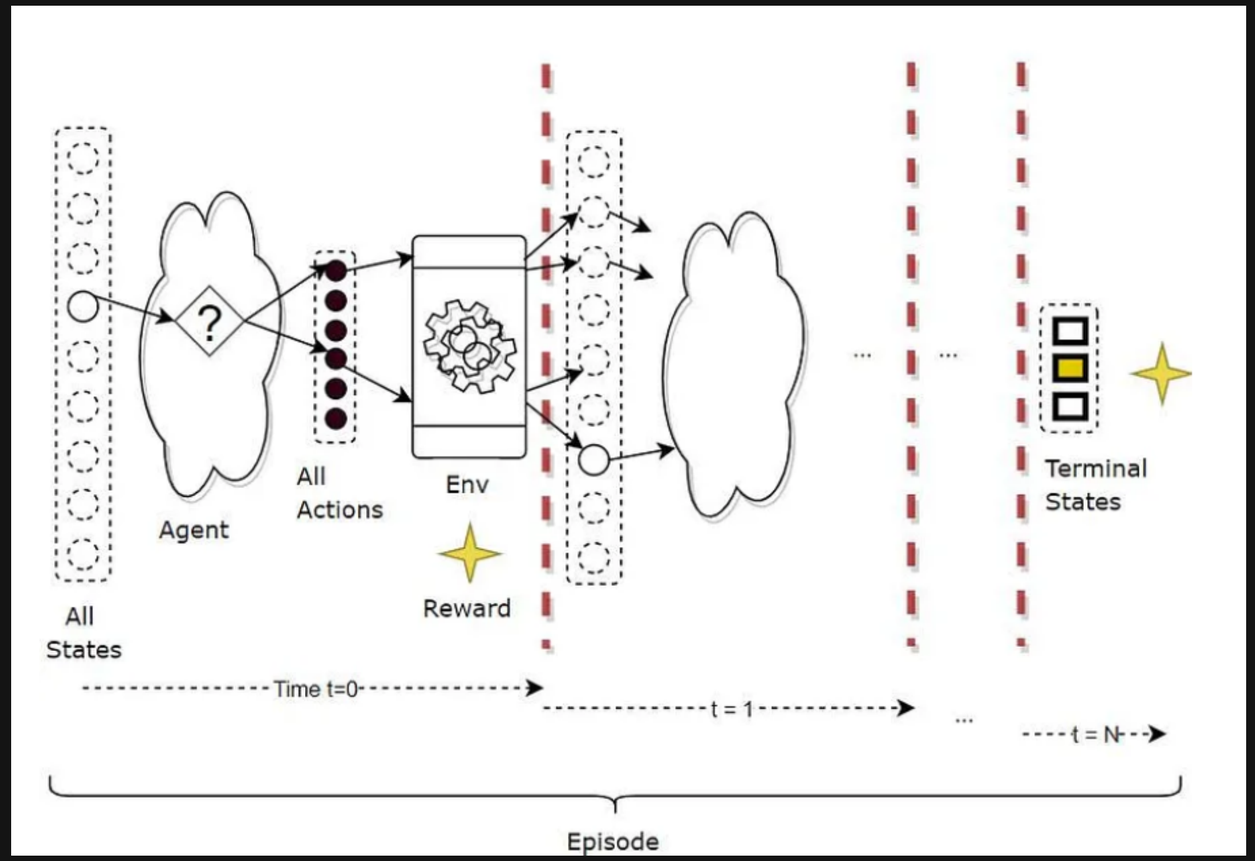
Let us consider a game of tic-tac-toe. Our defenition of the MDP would thus be as follows:

- The agent plays against the environment, the opponent.
- The state at any point, is the current position of all the tokens.
- There are 9 possible actions where the agent can place its token at each of the 9 available squares in the grid.
- If the agent wins it gets a positive reward of +10 points and if it loses it gets a negative reward of -10 points. Each intermediate move gives a neutral reward of 0 points.



An MDP iterates over timesteps

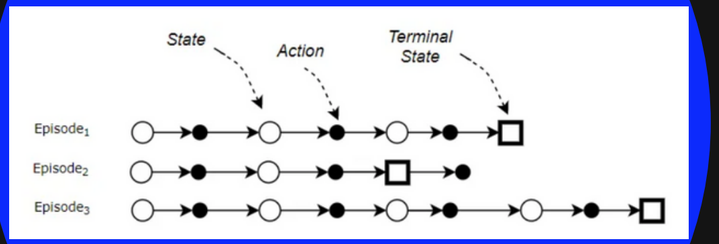




So the execution of the MDP can be described as a trajectory of occurrences (in terms of state, action, reward) over a sequence of time-steps.

(s3,a3,r4,s4,a4,r5,s5,a5,r6, s6)

CLASSIFICATION OF TASKS



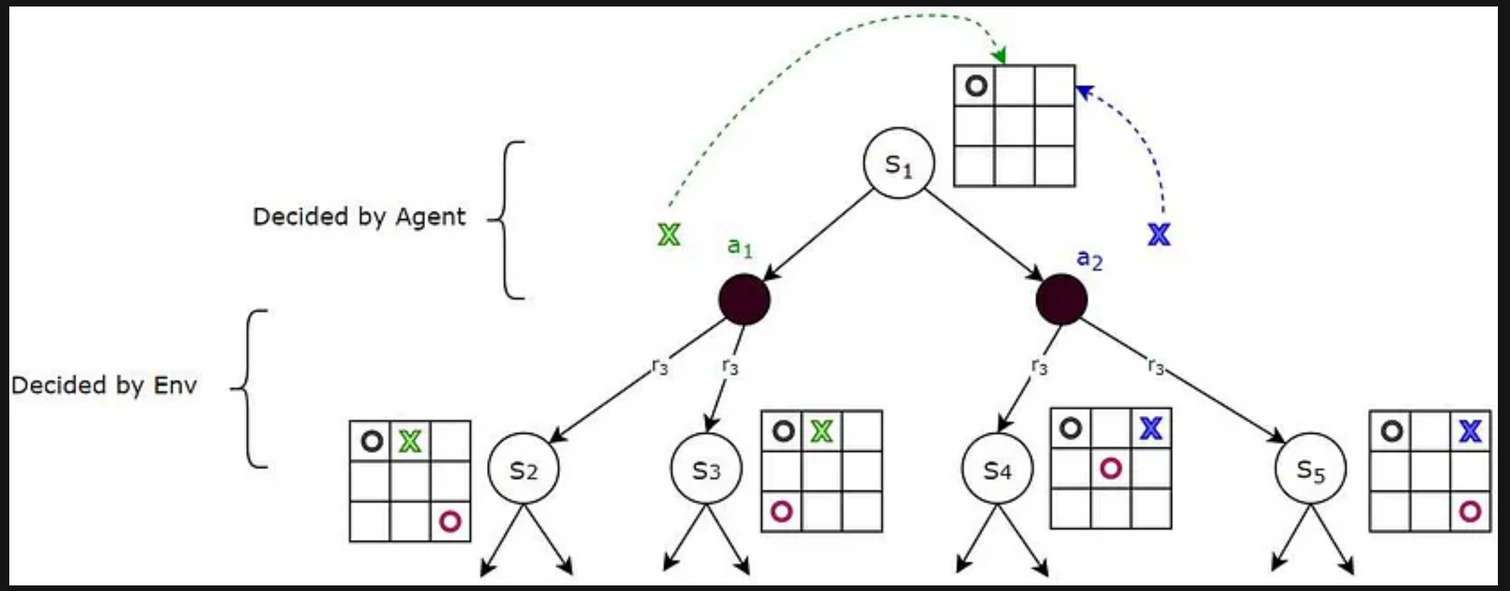
Episodic tasks

- Have a well defined end or Terminal state
- Each round is an episode
- Each episode is independent of each other

Continuing Tasks

- Have no end
- Can go on forever, or until the system is stopped
- Eg. A robot managing a process

Transitions



- Given a state, the agent determines the next action
- Given a state-action pair, the environment decides the next state
- Notice that the states are self-contained, the history of the game is not required to analyze the current state

THE BIGGER PICTURE

- After analyzing all the above, we must raise the question, what is our job?
- Despite the importance given to the environment, we have no control over it.
- We must focus ourselves to the actions of the agent.
- We need to train the agent to improve on its decision making process such that, given a state, it can chose the optimal further action.

WE NEED TO UNDERSTAND THE FOLLOWING

RETURN

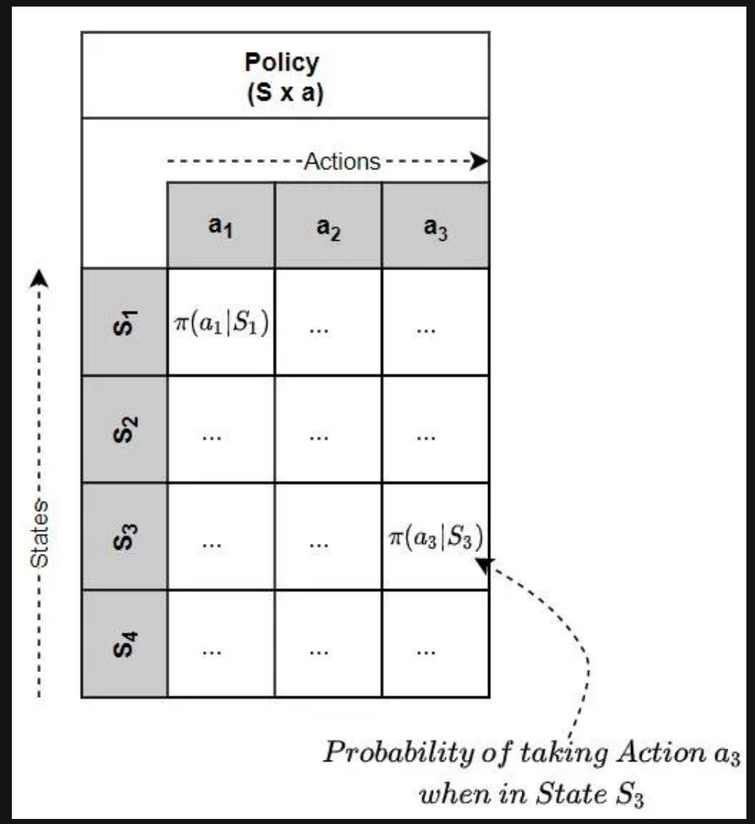
- Return is the cumulative reward reward accumulated over the task
- It is calculated using discounted rewards
- This ensures that for very long episodes and continuing tasks the return does not grow infinitely

$$\text{Return} = r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots + \gamma^n r_n$$

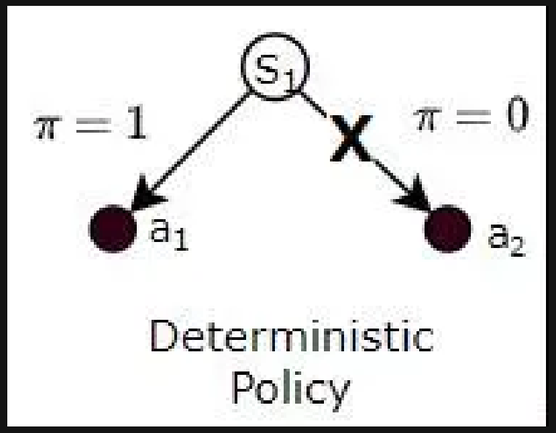
- Immediate rewards are more valuable than further ones.
- Rewards that lead to highest total return are better

POLICY

- Policy is the strategy followed to pick an action.
- Examples include always picking the next action at random, picking the next state that gives the highest known reward, etc.
- For convenience, we can approximate a policy as a lookup table

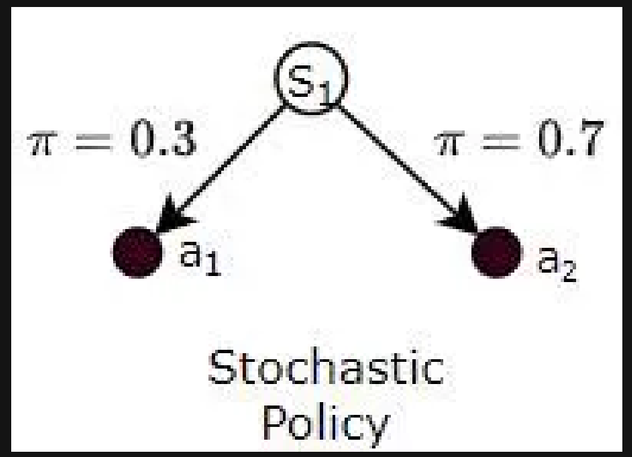


- Policies can be deterministic or stochastic



- The action of a deterministic policy is fixed

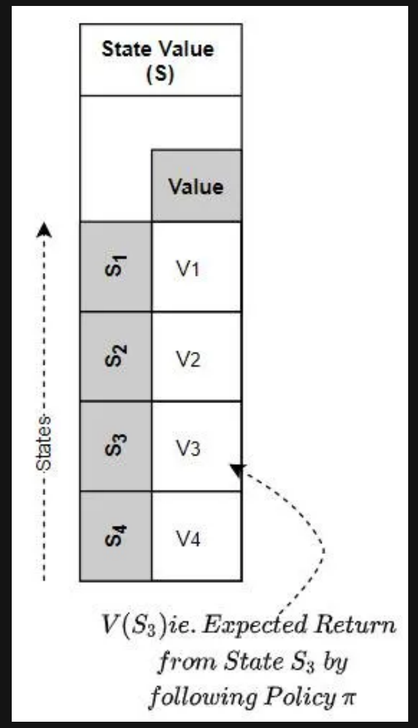
- On the other hand, in a stochastic process, actions are chosen based on a probability



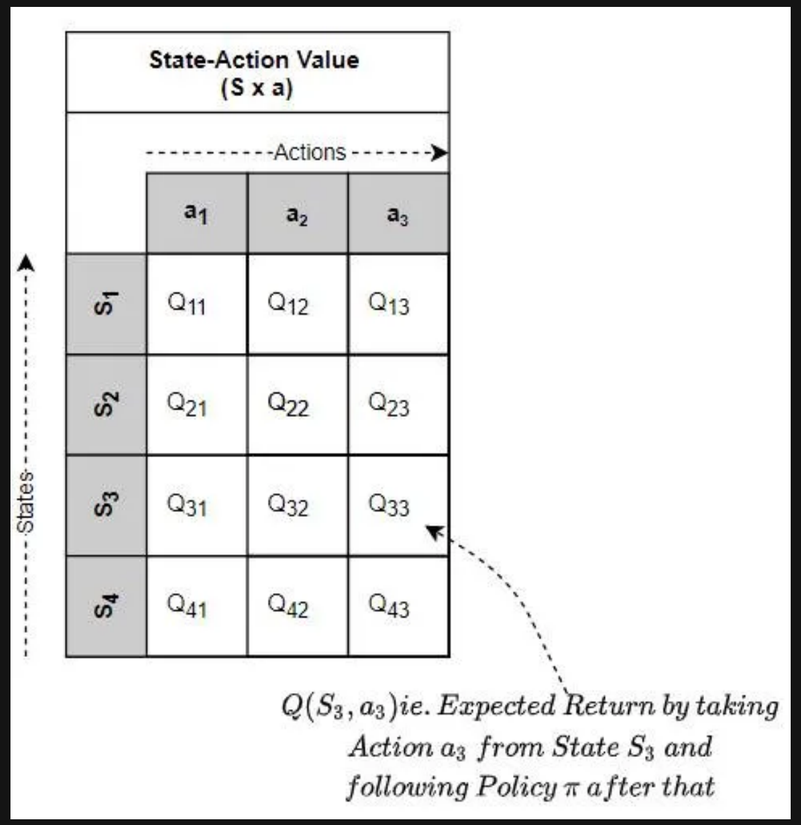
- It is this policy that we aim to learn through the training process

Value

- Given a policy, value represents the average return over many episodes
  It is of two types:
- State Value - The value after a policy is followed starting from a certain state



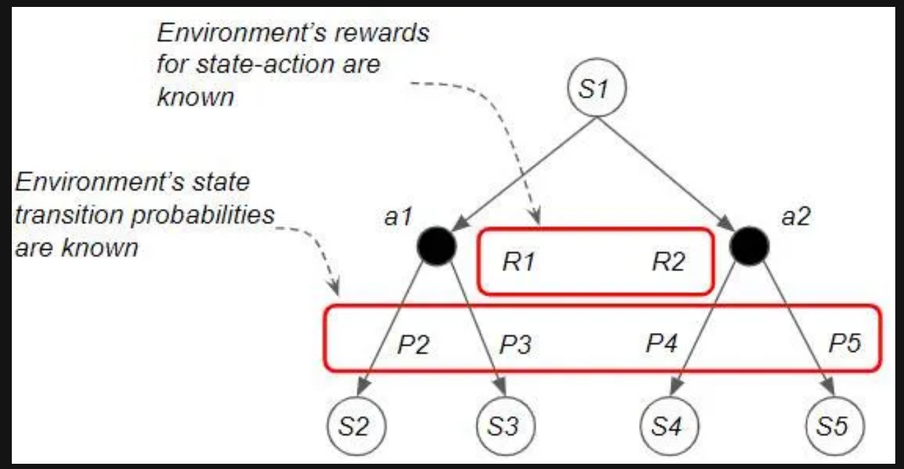
- State-Action Value (Q value) - It represents the value returned given a certain state and a certain action taken from given state.



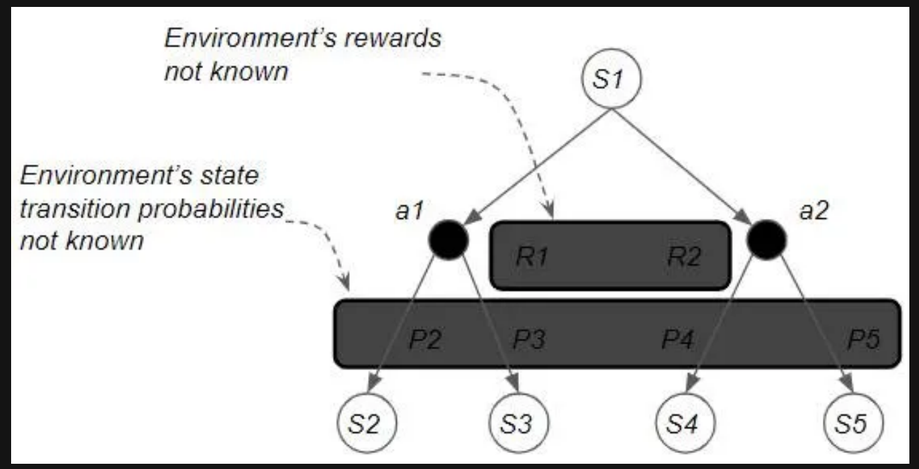
- We compare values of policies to find the optimal policy

Solution Categories

- Model Based - These assume that the internal operation of the environment is known.
- Given a state, the next state and the reward can be predicted
- This is known as planning.
- Model Free - These treat the environment as an external black box, whose workings are not known.
- This is known as Reinforcement Learning.



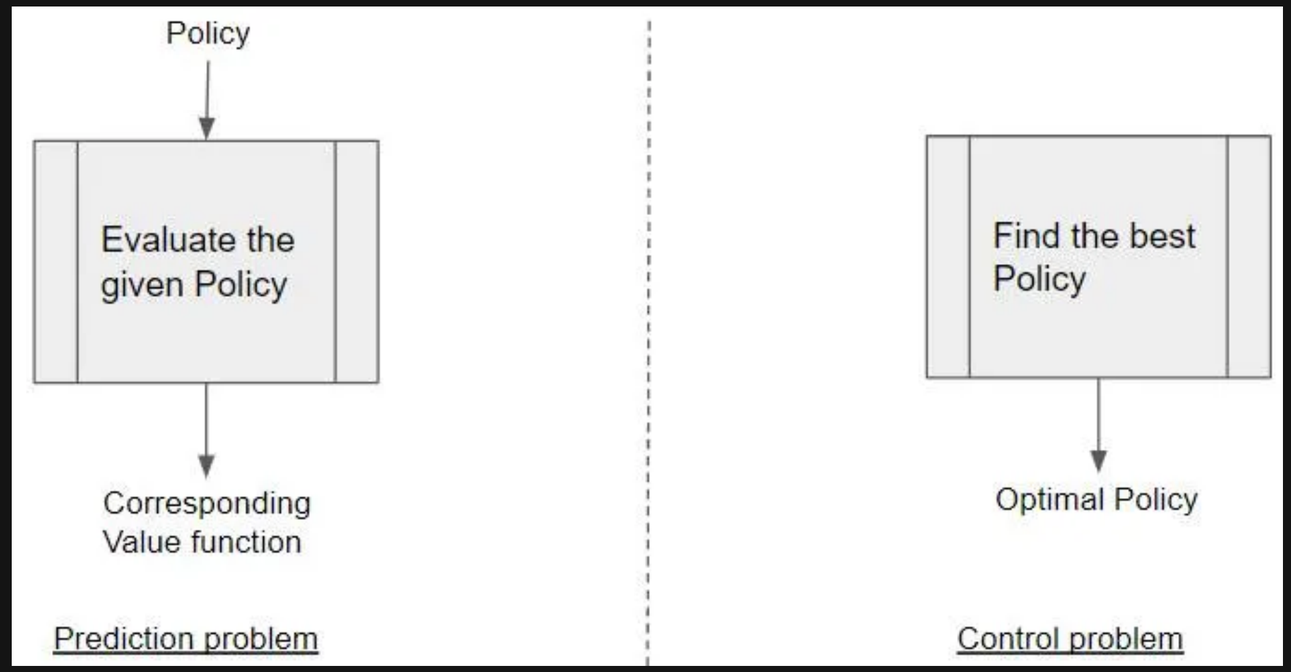
Model Based



Model Free

Another Classification

- Prediction - In a prediction problem, we are to find the Value given a certain policy
- Control - In a control problem, the agent explores the policy space to find the optimal policy

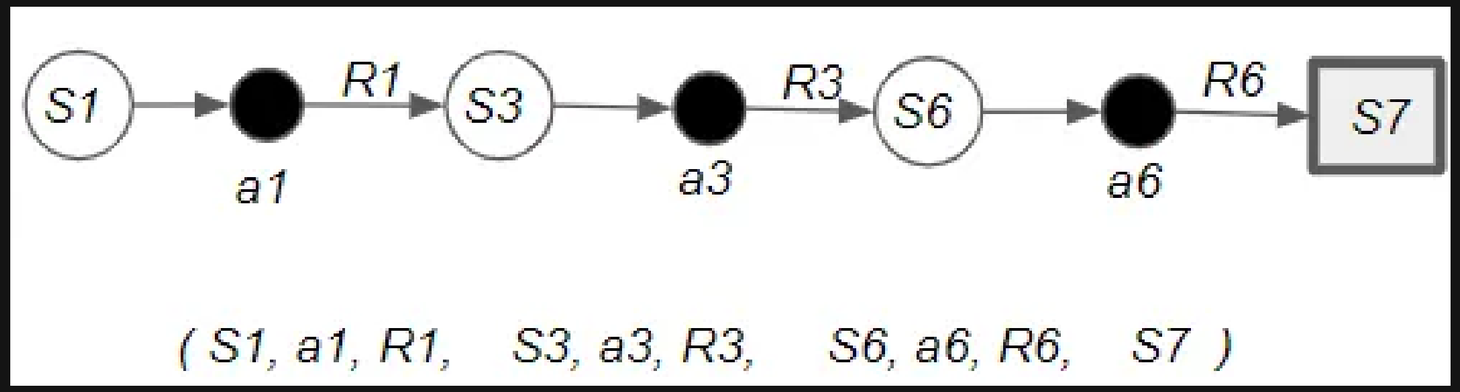


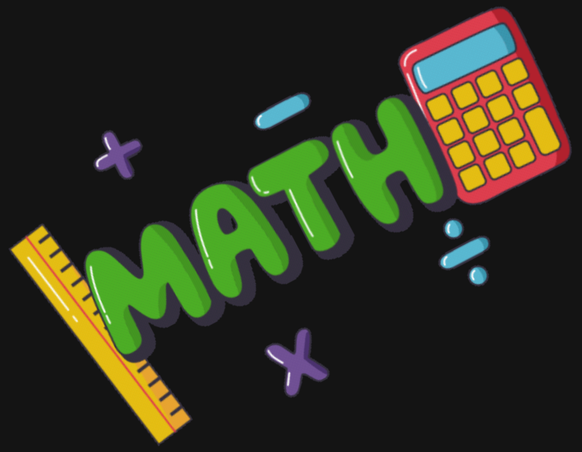
- Evidently, most of the problems we deal with in RL are Model-Free control problems

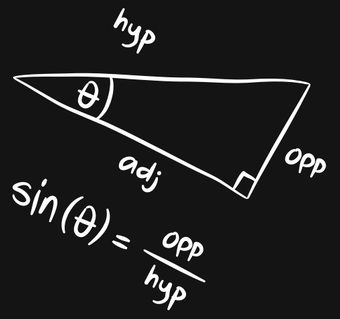
Model Free Learning

- As we have no understanding of how the environment reacts, the agent can only gain this understanding by interaction with the environment
- Thus, it takes actions, gets corresponding rewards, and tries to better its actions based on these.
- It learns on a trial and error basis

- One can think that it generates its own training data by going through various trajectories of steps.







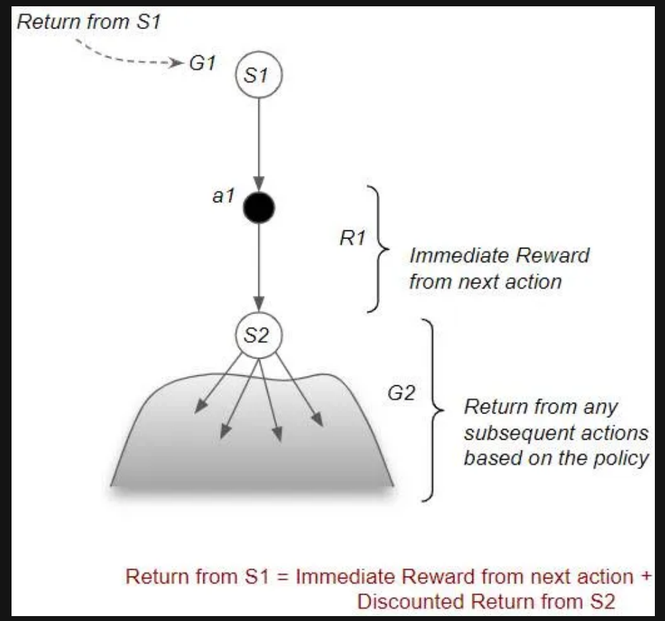
BELLMAN EQUATIONS

- In order to mathematically analyze any RL system, one must first study the Bellman Equations
- They are fairly intuitive, recursive relationships

$$X = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Return

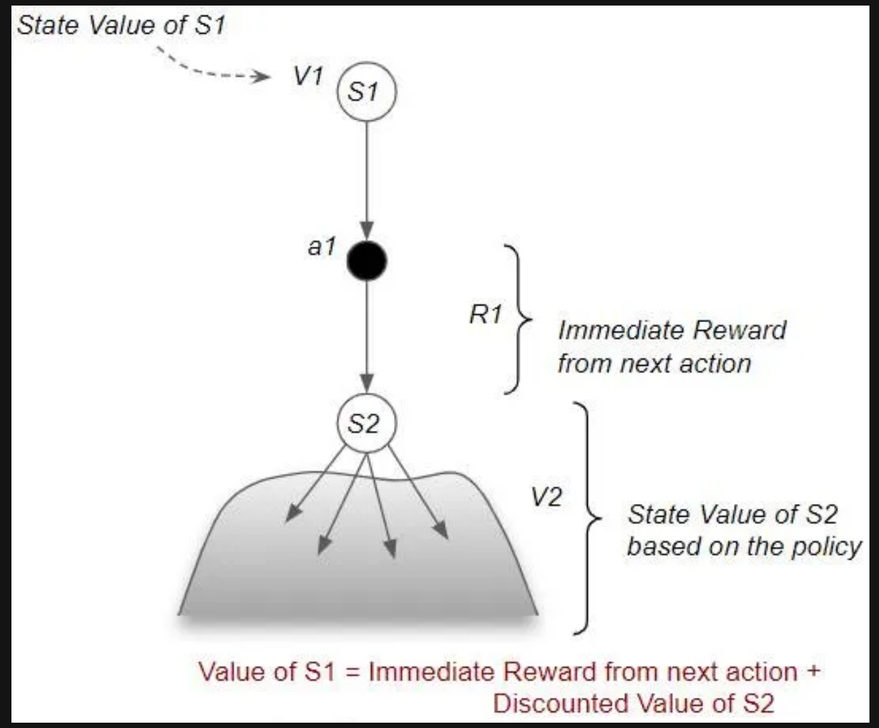
- The Bellman equation for return can be thought as follows:
- Following a policy, the return from a state can be thought of as the sum of the reward in going to the next state and the discounted return from the next state.



$$G_1 = R_1 + \gamma G_2$$

$$G_t = R_t + \gamma G_{t+1}$$

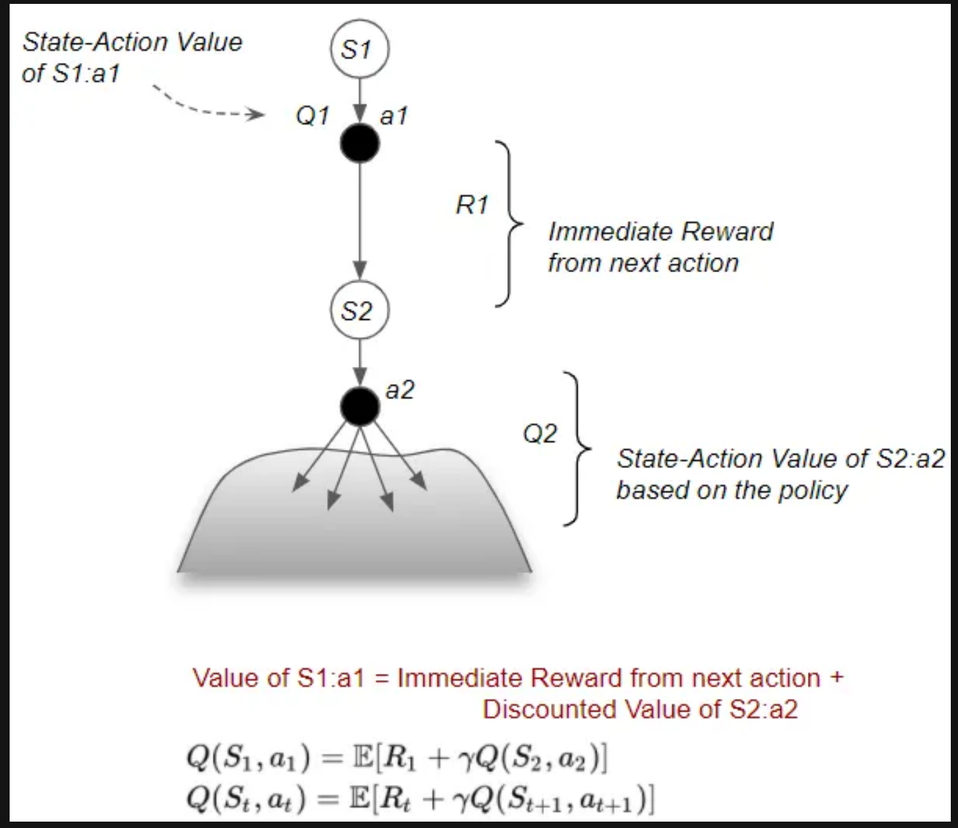
State-Value



$$V(S_1) = \mathbb{E}[R_1 + \gamma V(S_2)]$$
$$V(S_t) = \mathbb{E}[R_t + \gamma V(S_{t+1})]$$

- Here, the expectation value of reward over many iterations gives the value

STATE-ACTION VALUE



$$Q(S_1, a_1) = \mathbb{E}[R_1 + \gamma Q(S_2, a_2)]$$
$$Q(S_t, a_t) = \mathbb{E}[R_t + \gamma Q(S_{t+1}, a_{t+1})]$$

- The same logic as the above two can be applied here