Reinforcement Learning : Introduction

"Learning from interaction is a foundational idea underlying nearly all theories of learning and intelligence".

The slides contain

- Excerpts from the book Reinforcement Learning : An Introduction, 2nd edition, Richard S. Sutton and Andrew G. Barto
- Content acquired using ChatGPT
- Excerpts from Stanford CS234 Course Material available on their website : https://web.stanford.edu/class/cs234/modules.html

What is Reinforcement Learning (RL)?

- Reinforcement learning (RL) is a type of machine learning where an agent learns to make decisions by interacting with an environment.
- The agent receives feedback in the form of rewards or punishments based on its actions, helping it learn optimal strategies over time.
- RL is commonly used in areas like robotics, game playing, and autonomous systems.
- Algorithms such as Q-learning and deep reinforcement learning have advanced the field, allowing agents to learn complex behaviors through trial and error.

Reinforcement Learning

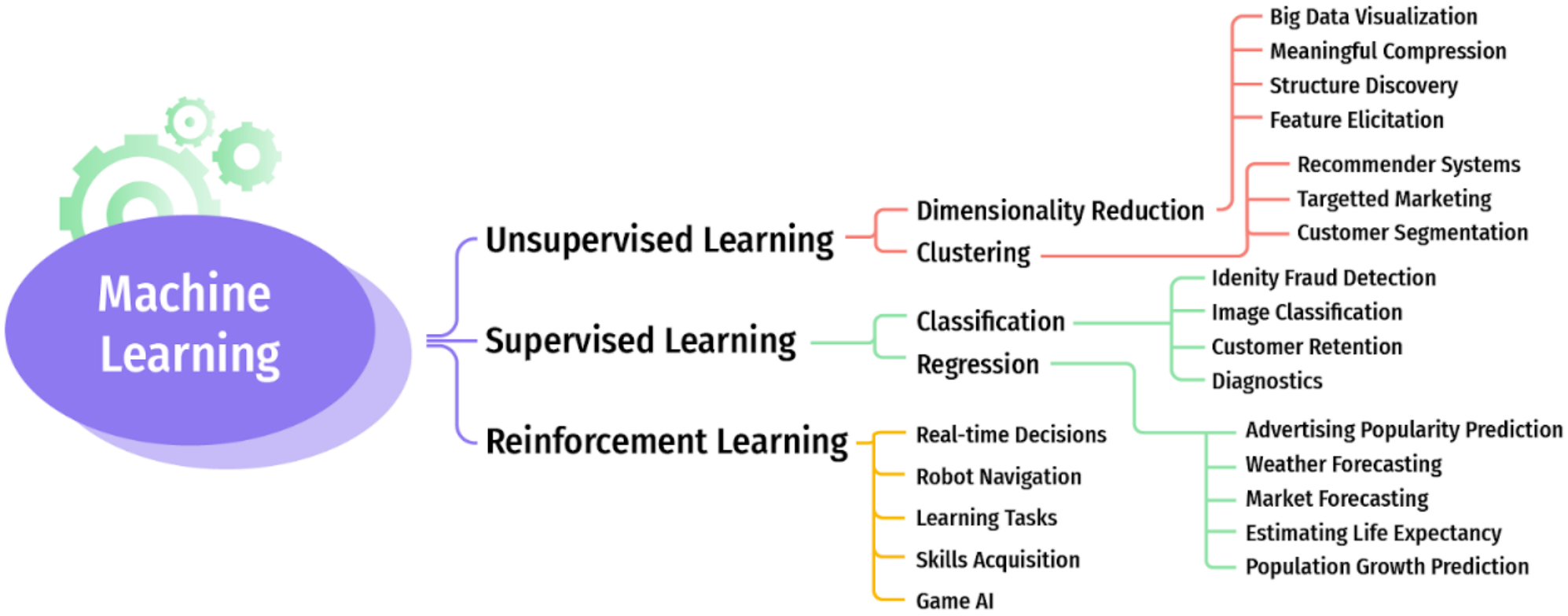
- Reinforcement learning is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them.
- In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards.
- These two characteristics—trial-and-error search and delayed reward—are the two most important distinguishing features of reinforcement learning.

Reinforcement Learning

Reinforcement Learning is simultaneously

- a problem,
- a class of solution methods that work well on the problem, and
- the field that studies this problem and its solution methods.

Types of Machine Learning



Source https://idapgroup.com/blog/types-of-machine-learning-out-there/

Machine Learning (ML) Types

- Supervised Learning:
  - In supervised learning, the algorithm is trained on a labeled dataset, which means the input data is paired with corresponding output labels.
  - The goal is for the algorithm to learn a mapping from inputs to outputs, allowing it to make predictions or decisions on new, unseen data.
  - Algorithms : Linear Regression, Logistic Regression, Decision Trees, Random Forest, Support Vector Machines (SVM), Naive Bayes, K-Nearest Neighbors (KNN), Neural Networks (Deep Learning), Gradient Boosting Algorithms (e.g., XGBoost, LightGBM), Linear Discriminant Analysis (LDA)

Machine Learning (ML) Types

- Unsupervised Learning:
  - Unsupervised learning involves training an algorithm on an unlabeled dataset, where the algorithm needs to find patterns, relationships, or structures in the data without explicit guidance.
  - Clustering and dimensionality reduction are common tasks in unsupervised learning.
  - Algorithms : K-Means Clustering, Hierarchical Clustering, DBSCAN (Density-Based Spatial Clustering of Applications with Noise), Principal Component Analysis (PCA), Independent Component Analysis (ICA), Autoencoders, Generative Adversarial Networks (GANs), t-Distributed Stochastic Neighbor Embedding (t-SNE), Apriori Algorithm, Mean Shift

Machine Learning (ML) Types

- Reinforcement Learning:
  - Reinforcement learning is about training an agent to make sequential decisions by interacting with an environment.
  - The agent receives feedback in the form of rewards or punishments based on its actions, and it learns to optimize its behavior over time.
  - Algorithms : Q-Learning, Deep Reinforcement Learning

RL Involves

- Optimization
- Delayed consequences
- Exploration
- Generalization

RL Involves

- Optimization
  - Optimization in reinforcement learning refers to the process of finding the best policy or strategy that maximizes the cumulative reward over time.
  - Goal is to find an optimal way to make decisions
    - Yielding best outcomes or at least very good outcomes
  - Explicit notion of utility of decisions
  - Example: finding minimum distance route between two cities given network of roads

RL Involves

- Delayed Consequences
  - Delayed consequences highlight the challenge in RL where actions may not immediately lead to rewards, requiring the agent to consider long-term effects and plan accordingly.
  - Decisions now can impact things much later...
    - Choosing an elective subject.
  - Introduces two challenges
    - When planning: decisions involve reasoning about not just immediate benefit of a decision but also its longer term ramifications
    - When learning: temporal credit assignment is hard (what caused later high or low rewards?)

RL Involves

- Exploration
  - Exploration is the agent's strategy of trying out different actions to discover optimal policies, balancing the need for exploiting known good actions and exploring potentially better ones.
  - Learning about the world by making decisions
  - Agent as scientist
    - Learn to ride a bike by trying (and failing)
  - Censored data
    - Only get a reward for decision made
  - Decisions impact what we learn about
    - If we choose to go to X college instead of Y, we will have different later experiences.

RL Involves

- **Generalization**
  - Generalization involves the ability of a reinforcement learning agent to apply learned knowledge from specific experiences to new, unseen situations, enhancing its adaptability and efficiency.

Comparison

	AI Planning	SL	UL	RL	IL
Optimization				X	X
Learns from Experience		X	X	X	X
Generalization		X	X	X	X
Delayed consequences				X	X
Exploration				X	

Examples of RL

- A master chess player makes a move.
  - The choice is informed both by planning (anticipating possible replies and counter replies) and by immediate, intuitive judgments of the desirability of particular positions and moves.
- An adaptive controller adjusts parameters of a petroleum refinery's operation in real time.
  - The controller optimizes the yield/cost/quality trade-off on the basis of specified marginal costs without sticking strictly to the set points originally suggested by engineers.

Examples of RL

- A gazelle calf struggles to its feet minutes after being born.
  - Half an hour later it is running at 20 miles per hour.
- A mobile robot decides whether it should enter a new room in search of more trash to collect or start trying to find its way back to its battery recharging station.
  - It makes its decision based on the current charge level of its battery and how quickly and easily it has been able to find the recharger in the past.
- Phil prepares his breakfast.

Examples of RL : Preparing Breakfast

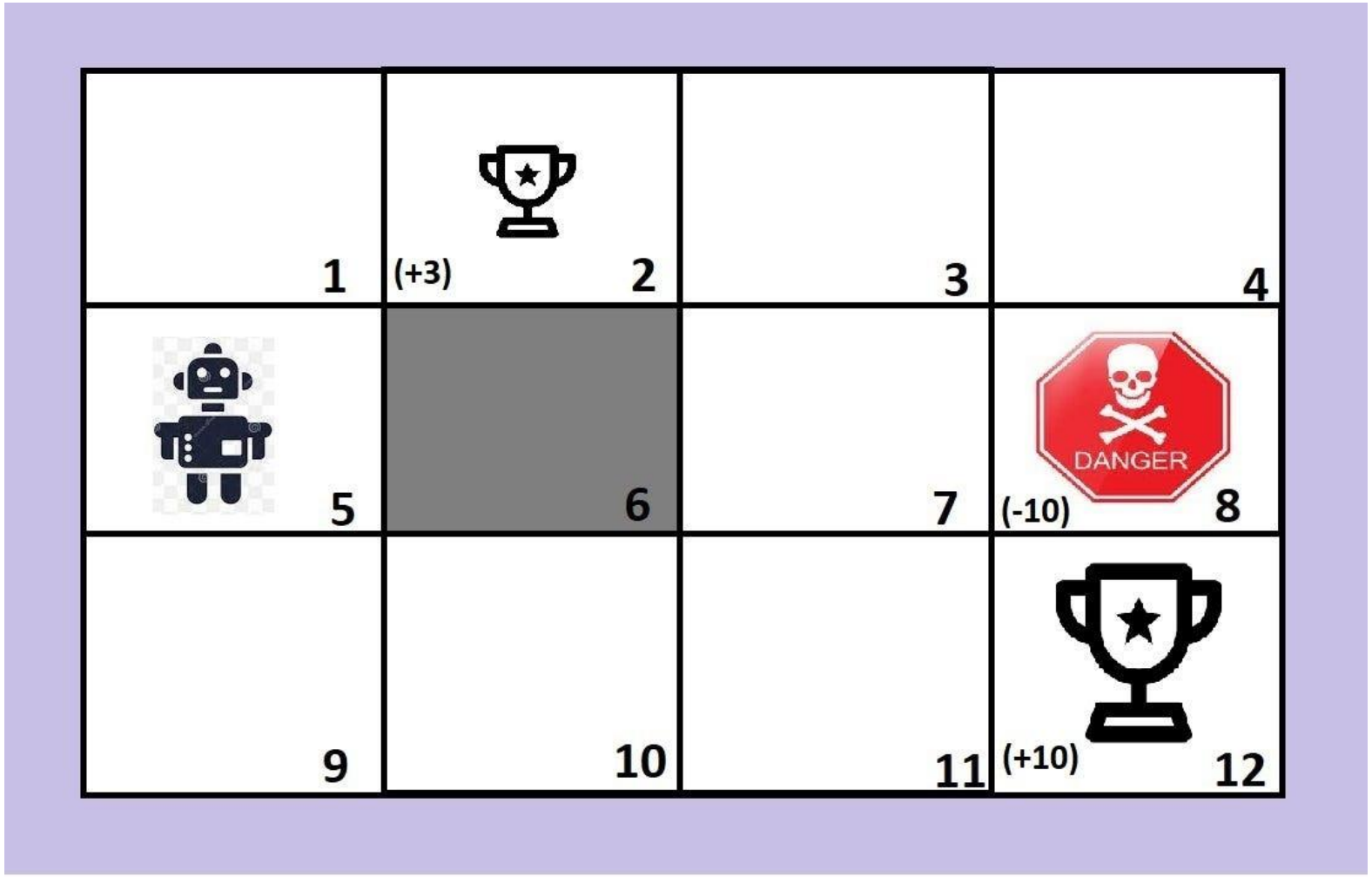
- Closely examined, even this apparently mundane activity reveals a complex web of conditional behavior and interlocking goal–sub goal relationships: walking to the cupboard, opening it, selecting a cereal box, then reaching for, grasping, and retrieving the box.
- Other complex, tuned, interactive sequences of behavior are required to obtain a bowl, spoon, and milk carton.
- Each step involves a series of eye movements to obtain information and to guide reaching and locomotion.

Examples of RL : Preparing Breakfast

- Rapid judgments are continually made about how to carry the objects or whether it is better to ferry some of them to the dining table before obtaining others.
- Each step is guided by goals, such as grasping a spoon or getting to the refrigerator, and is in service of other goals, such as having the spoon to eat with once the cereal is prepared and ultimately obtaining nourishment.
- Whether he is aware of it or not, Phil is accessing information about the state of his body that determines his nutritional needs, level of hunger, and food preferences.

Exploration and Exploitation

Exploration and Exploitation



Elements of Reinforcement Learning

Elements of Reinforcement Learning

- Policy
- Reward
- Value Function
- Model

Elements of RL : Policy

- A policy defines the learning agent’s way of behaving at a given time.
- Roughly speaking, a policy is a mapping from perceived states of the environment to actions to be taken when in those states.
- In some cases the policy may be a simple function or lookup table, whereas in others it may involve extensive computation such as a search process.
- The policy is the core of a reinforcement learning agent in the sense that it alone is sufficient to determine behavior.

Elements of RL : Reward

- A reward signal defines the goal of a reinforcement learning problem.
- On each time step, the environment sends to the reinforcement learning agent a single number called the reward.
- The agent’s sole objective is to maximize the total reward it receives over the long run.
- The reward signal thus defines what are the good and bad events for the agent.
- The reward signal is the primary basis for altering the policy; if an action selected by the policy is followed by low reward, then the policy may be changed to select some other action in that situation in the future.

Elements of RL : Value function

- Whereas the reward signal indicates what is good in an immediate sense, a value function specifies what is good in the long run.
- Roughly speaking, the value of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state.
- Whereas rewards determine the immediate, intrinsic desirability of environmental states, values indicate the long-term desirability of states after taking into account the states that are likely to follow and the rewards available in those states.
- For example, a state might always yield a low immediate reward but still have a high value because it is regularly followed by other states that yield high rewards. Or the reverse could be true.
- To make a human analogy, rewards are somewhat like pleasure (if high) and pain (if low), whereas values correspond to a more refined and farsighted judgment of how pleased or displeased we are that our environment is in a particular state.

Juxtaposing Rewards and Value Functions

- Rewards are in a sense primary, whereas values, as predictions of rewards, are secondary.
- Without rewards there could be no values, and the only purpose of estimating values is to achieve more reward.
- It is values with which we are most concerned when making and evaluating decisions. Action choices are made based on value judgments.
- We seek actions that bring about states of highest value, not highest reward, because these actions obtain the greatest amount of reward for us over the long run.
- Unfortunately, it is much harder to determine values than it is to determine rewards. Rewards are basically given directly by the environment, but values must be estimated and re-estimated from the sequences of observations an agent makes over its entire lifetime.

Elements of RL : Model

- Model is something that mimics the behavior of the environment, or more generally, that allows inferences to be made about how the environment will behave.
- For example, given a state and action, the model might predict the resultant next state and next reward.
- Models are used for planning, by which we mean any way of deciding on a course of action by considering possible future situations before they are actually experienced.
- Methods for solving reinforcement learning problems that use models and planning are called model-based methods, as opposed to simpler model-free methods that are explicitly trial-and-error learners—viewed as almost the opposite of planning.
- Modern reinforcement learning spans the spectrum from low-level, trial-and-error learning to high-level, deliberative planning.

Types of RL

Reinforcement Learning : Types

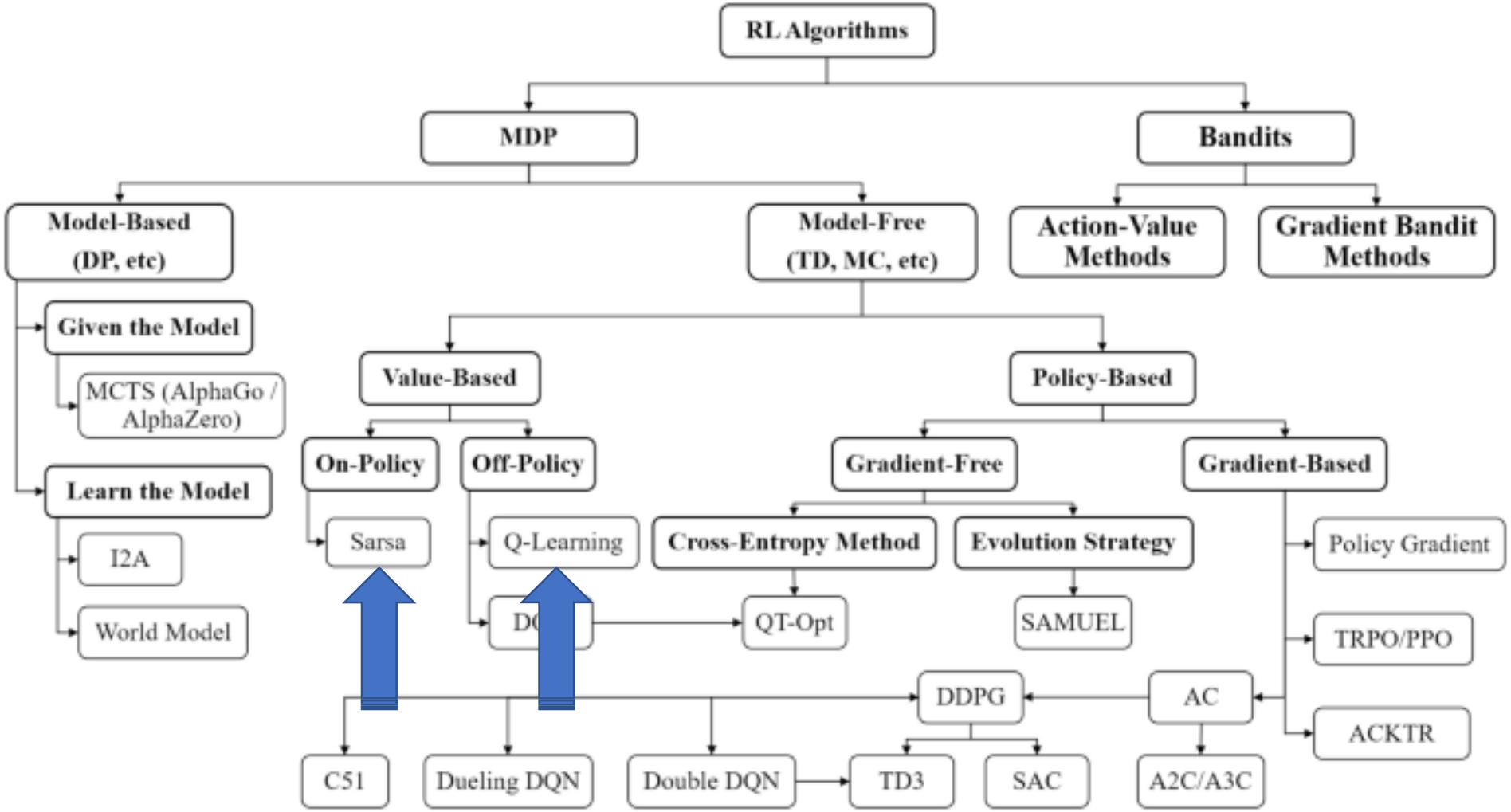
- Model-based methods:
  - These methods use a model of the environment to predict the next state and reward.
  - Examples include Markov decision processes (MDPs) and dynamic programming.
- Model-free methods:
  - These methods do not use a model of the environment and instead learn from experience.
  - Examples include Q-learning and State Action Reward State action (SARSA).

Reinforcement Learning : Types

- Value-based methods:
  - These methods learn to estimate the value of each state or state-action pair.
  - Examples include Q-learning and deep Q-networks (DQNs).
- Policy-based methods:
  - These methods learn a policy that maps states to actions.
  - Examples include policy gradient methods and actor-critic methods.

RL Algorithms

Reinforcement Learning : Algorithms



Source : https://link.springer.com/chapter/10.1007/978-981-15-4095-0_3

Markov Decision Process Introduction

Markov Decision Process (MDP)

- MDP is a mathematical framework used to model decision-making problems where an agent interacts with an environment over a sequence of discrete time steps.
- It provides a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker.
- The MDP framework is widely used in the field of reinforcement learning to formalize problems and algorithms.

Markov Decision Process (MDP)

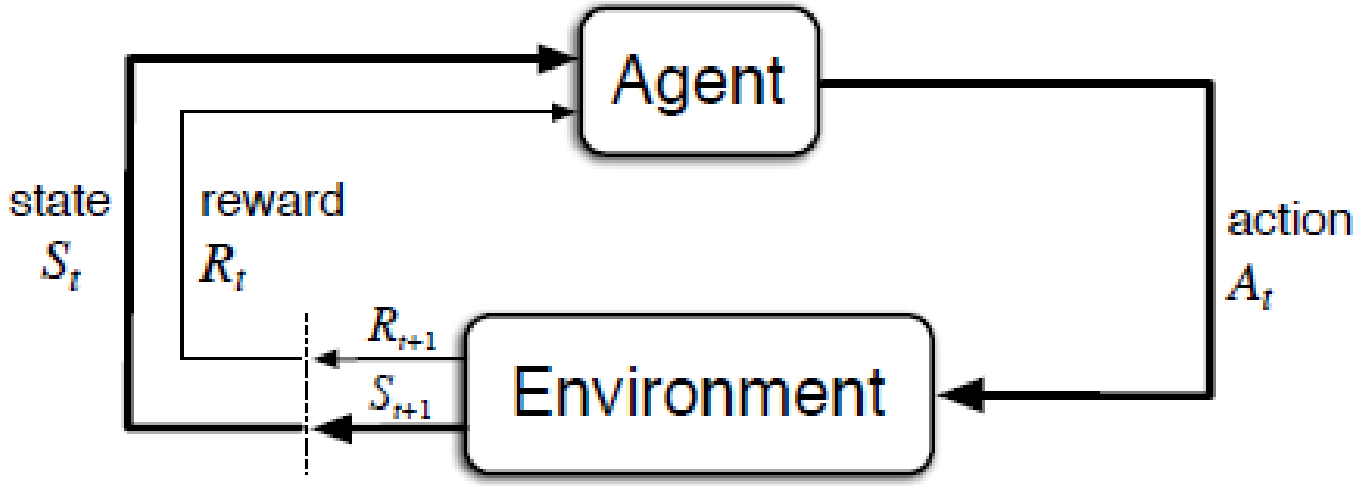


Figure : The agent–environment interaction in a Markov decision process.

Markov Decision Process (MDP)

- Key components of an MDP include:
  - **States (S):** A finite set of possible situations or configurations the system can be in. At each time step, the system is in a specific state.
  - **Actions (A):** A finite set of possible actions the agent can take. The available actions may depend on the current state.
  - **Transition Probabilities (P):** The probabilities of moving from one state to another based on the chosen action. It defines the dynamics of the system.
  - **Rewards (R):** Immediate numerical values that the agent receives as a consequence of taking a specific action in a particular state. The goal is to maximize the cumulative reward over time.
  - **Policy (π):** A strategy or mapping from states to actions that guides the agent's decision-making. It represents the agent's behavior in the environment.
  - **Discount Factor (γ):** A parameter that influences the agent's preference for immediate rewards over future rewards. It determines the importance of future rewards in the decision-making process.
  - **Value Function (V or Q):** The expected cumulative reward an agent can achieve starting from a particular state (V) or state-action pair (Q) following a given policy.

Markov Property in MDP

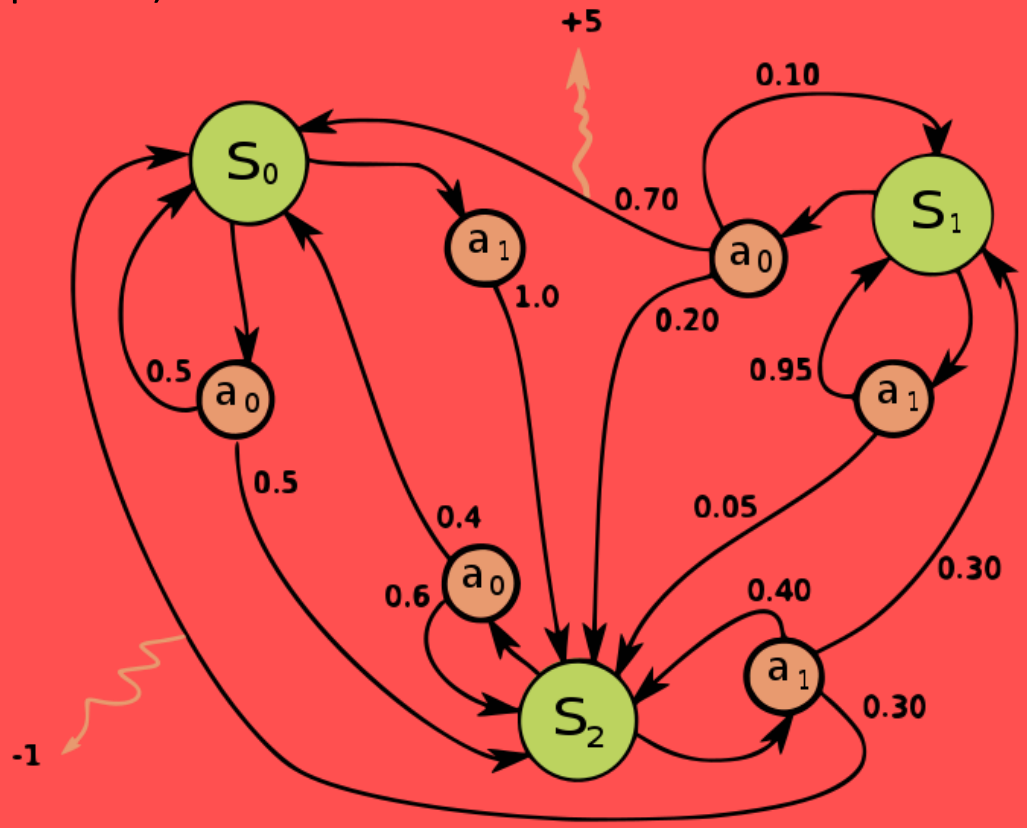
- The dynamics of an MDP satisfy the Markov property, meaning the future state depends only on the current state and action, and not on the sequence of states and actions that preceded it.
  - making the decision-making process memoryless.

MDP (from Wikipedia)

A Markov decision process is a 4-tuple (S, A, P_a, R_a), where:

- S is a set of states called the *state space*,
- A is a set of actions called the *action space* (alternatively, A_s is the set of actions available from state s),
- $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ is the probability that action a in state s at time t will lead to state s' at time $t + 1$,
- $R_a(s, s')$ is the immediate reward (or expected immediate reward) received after transitioning from state s to state s', due to action a

MDP (from Wikipedia)



Example of a simple MDP with three states (green circles) and two actions (orange circles), with two rewards (orange arrows).

Reinforcement Learning : Algorithms

- Q-Learning
- State Action Reward State action (SARSA)

Q-Learning

$$Q(s_t,a_t) \leftarrow (1-\alpha) \cdot Q(s_t,a_t) + \alpha \cdot (r_{t+1} + \gamma \cdot \max_a Q(s_{t+1},a))$$

Q-Learning Algorithm

- Create a table Q(s, a) to represent the Q-values for each state-action pair. Initialize the Q-values arbitrarily, often with zeros.
- For each time step t, observe the current state s_t.
- Select an action a_t using the exploration-exploitation strategy.
  - Common strategies include ε-greedy, where with probability ε, a random action is selected, and with probability 1-ε, the action with the highest Q-value is chosen.
- Execute the selected action and observe the immediate reward r_{t+1} and the new state s_{t+1}.
- Update the Q-value of the current state-action pair using the equation:

$$Q(s_t,a_t) \leftarrow (1-\alpha) \cdot Q(s_t,a_t) + \alpha \cdot (r_{t+1} + \gamma \cdot max_a Q(s_{t+1},a))$$

Q-Learning : Learning Rate (α)

- **Role:** The learning rate (α) determines the extent to which the Q-values are updated in each iteration. It controls the weight given to new information compared to the existing knowledge in the Q-table.
- **Range:** $0 < \alpha \leq 1$.
- **Effect:** A higher learning rate gives more weight to recent experiences, potentially leading to faster convergence but also making the algorithm more sensitive to noise. A lower learning rate makes the algorithm more stable but slower to adapt to changes in the environment.

Q-Learning : Discount Factor (γ)

- **Role:** The discount factor (γ) determines the importance of future rewards in the Q-value update. It reflects the agent's preference for immediate rewards over delayed rewards.
- **Range:** $0 \leq \gamma \leq 1$.
- **Effect:** A higher discount factor values future rewards more, encouraging the agent to consider long-term consequences. A lower discount factor makes the agent focus more on immediate rewards.

Q-Learning

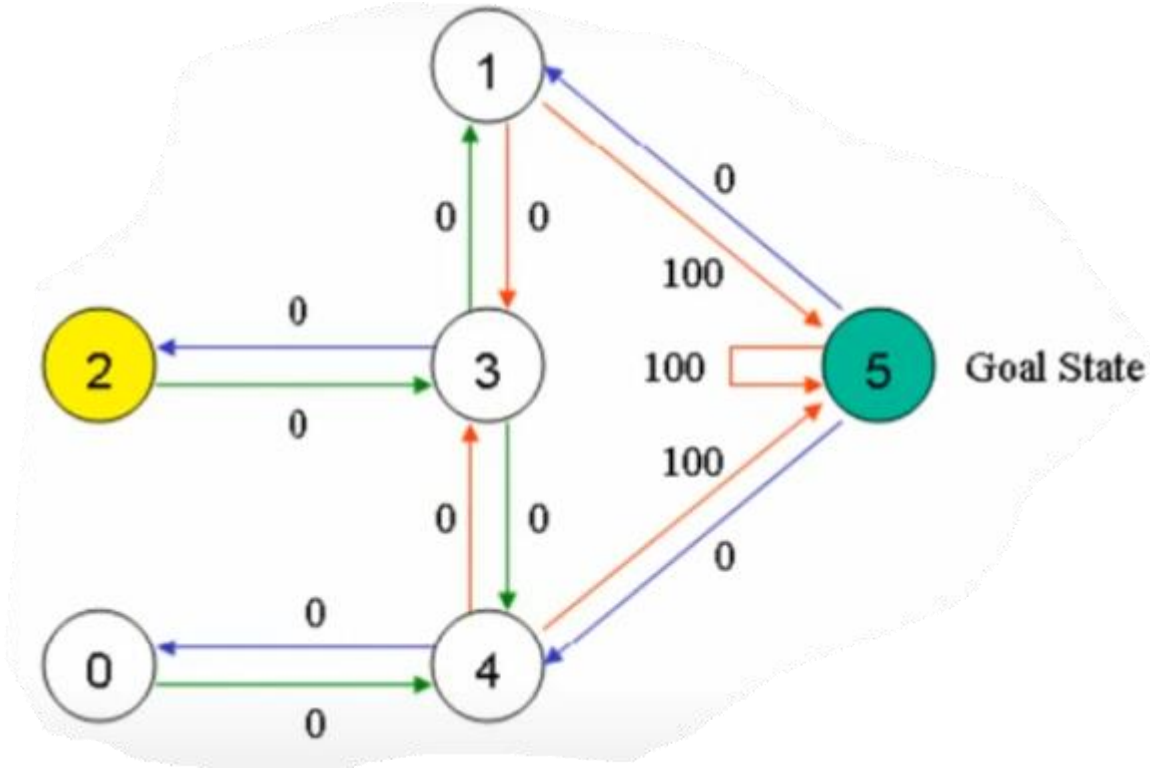
- $Q(s_t,a_t) \leftarrow (1-\alpha) \cdot Q(s_t,a_t) + \alpha \cdot (r_{t+1} + \gamma \cdot max_a Q(s_{t+1},a))$

- Assume $\alpha = 1$
  - What does it signify?
  - $Q(s_t,a_t) \leftarrow r_{t+1} + \gamma \cdot max_a Q(s_{t+1},a)$

Q-Learning : Example

Consider Rewards as

- -1 for ‘No Connection’
- 0 for ‘Non Goal State’
- 100 for ‘Goal State’

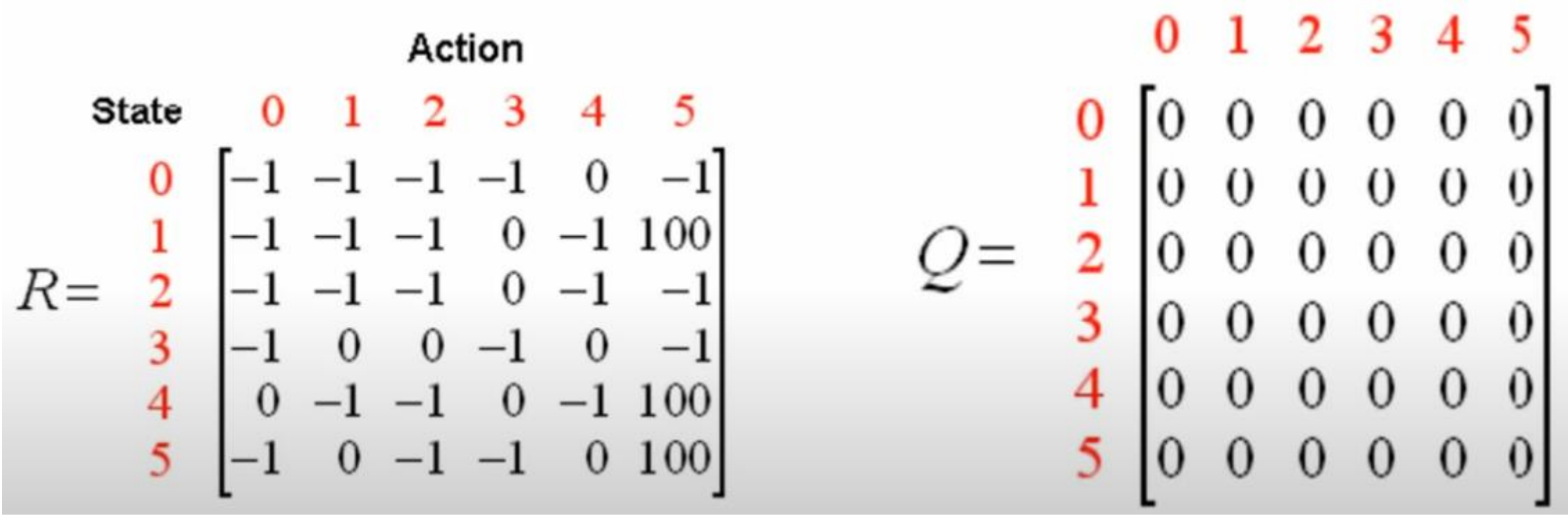


$R=$

State \ Action	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						

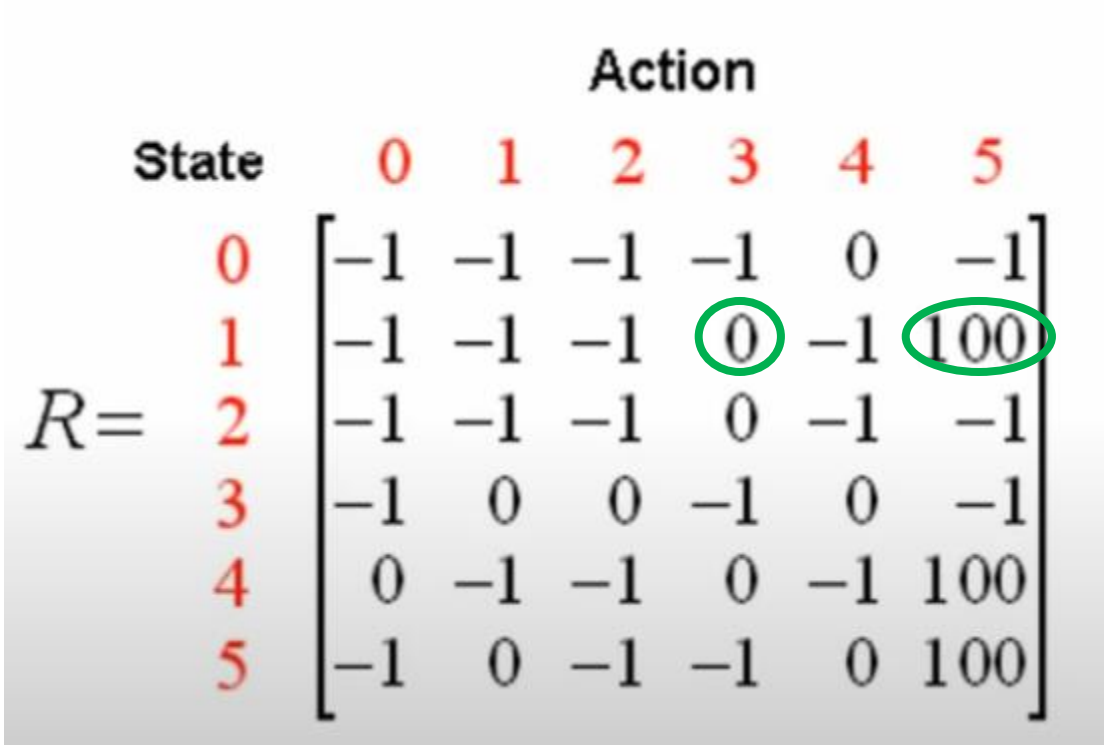
Q-Learning

- Initialize Q matrix to Zero



Q-Learning

- Consider discount factor (γ) = 0.8
- Initial State as 1
- We can go to ??
  - 3 or 5
- How to select one of it?



Q-Learning

- Assume 5 is Selected
  - Calculate Q (1,5)
  - Which actions from 5 ??
  - 1, 4 or 5
- Calculations
  - $Q(s_t,a_t) \leftarrow r_{t+1} + \gamma \cdot max_a Q(s_{t+1},a)$
  - Q (1,5) = 100 + 0.8 Max [Q(5,1), Q(5,4), Q(5,5)]
  - Q (1,5) = 100 + 0.8 Max [0 , 0, 0]
  - Q (1,5) = 100 + 0.8 * 0
  - Q (1,5) = 100

$$R = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} \text{Action} & & & & & \\ 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array} \quad \text{State}$$

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

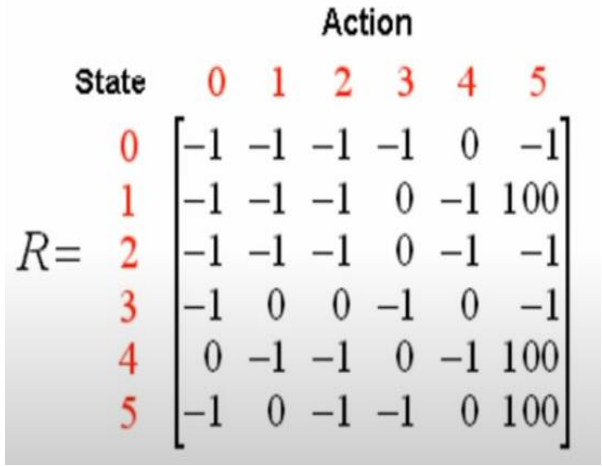
Q-Learning

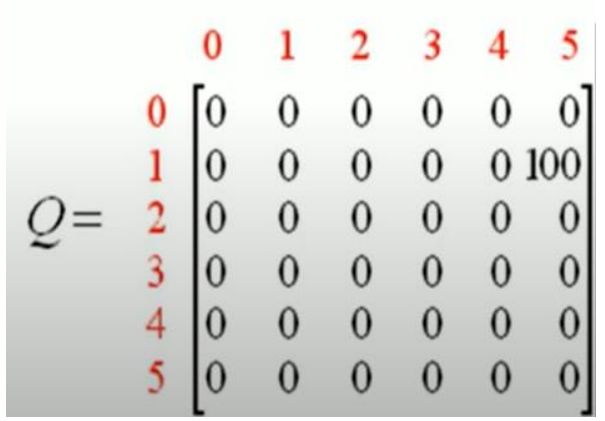
$$R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

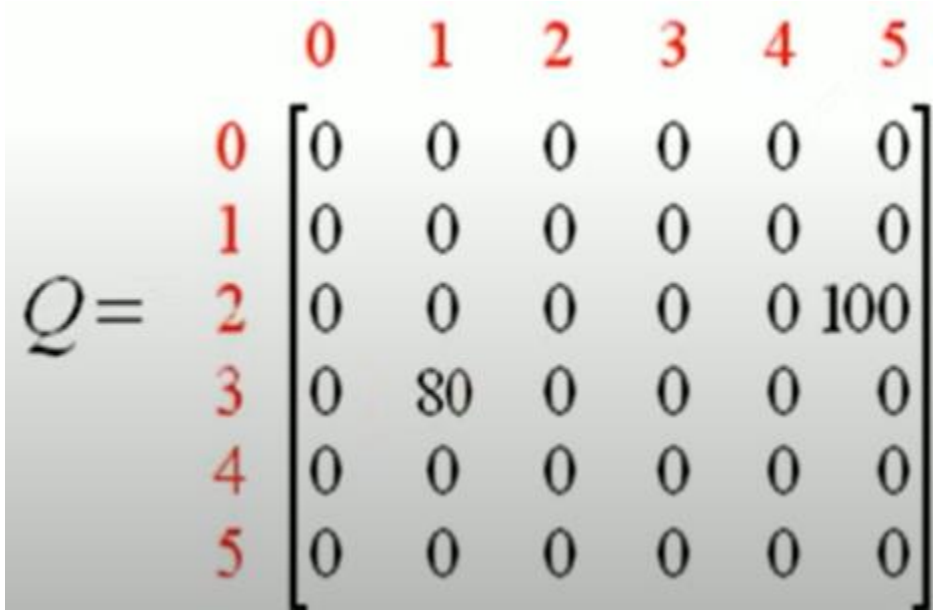
- Q(1,5) is calculated as 100
- Next state selected is 5.
- We can’t go anywhere from 5, since it is goal state.
- One episode is over.
- For next episode, randomly choose the starting stage.
  - Let us say 3
  - From 3 we can go to ??
    - 1, 2 or 4
    - Assume 1 is selected, Calculate Q (3,1)

Q-Learning

- Q (3,1)
  - = R(3,1) + 0.8 * Max [Q(1,3), Q(1,5)]
  - = 0 + 0.8 * Max [0, 100]
  - = 0.8 * 100
  - = 80







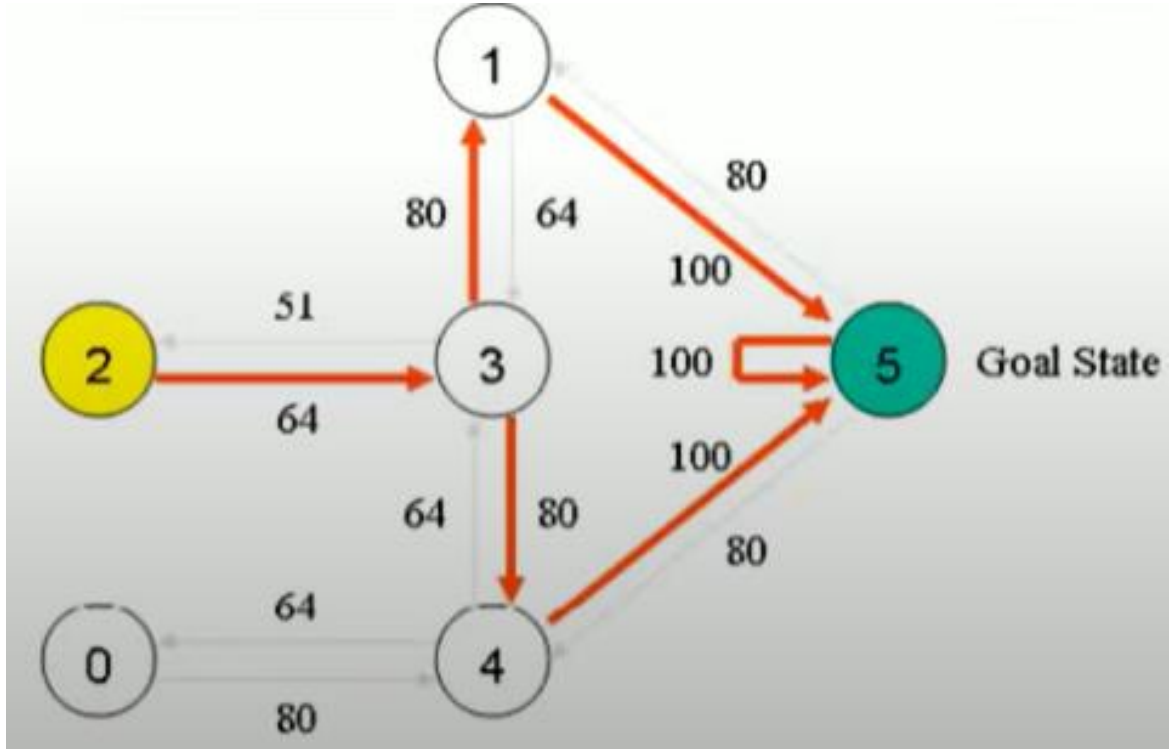
Q-Learning

- The agent learns more through further episodes.
- It will finally reach to convergence values in Matrix Q as

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \end{array}$$

Q-Learning

- Once fully learnt (convergence happened) the agent may simply follow the action with highest reward at given state to reach the Goal State



SARSA

State-Action-Reward-State-Action

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \cdot (r_{t+1} + \gamma \cdot Q(s_{t+1},a_{t+1}) - Q(s_t,a_t))$$

SARSA

- SARSA (State-Action-Reward-State-Action) is another reinforcement learning algorithm, similar to Q-learning, used for learning optimal policies in Markov Decision Processes (MDPs).
- Like Q-learning, SARSA is a model-free algorithm, meaning it does not require knowledge of the underlying dynamics of the environment.

SARSA Overview

1. **Initialize Q-values**:
   - Start by initializing the Q-values for all state-action pairs.
   - This is typically done by creating a Q-table and initializing its entries with arbitrary values.
2. **Exploration-Exploitation Tradeoff**:
   - Define an exploration-exploitation strategy, similar to Q-learning, to balance exploration of new actions and exploitation of learned knowledge.
   - Common strategies include ε-greedy, where with probability ε, a random action is selected, and with probability 1-ε, the action with the highest Q-value is chosen.

SARSA Overview

3. **Learning Process:**
   - For each time step *t*, observe the current state s_t.
   - Select an action a_t using the exploration-exploitation strategy.
   - Execute the selected action and observe the immediate reward r_{t+1} and the new state s_{t+1}.
   - Select a new action a_{t+1} using the same exploration-exploitation strategy.
   - Update the Q-value of the current state-action pair using the SARSA update rule:
   - $$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \cdot (r_{t+1} + \gamma \cdot Q(s_{t+1},a_{t+1}) - Q(s_t,a_t))$$
     - α is the learning rate ($0 < \alpha \leq 1$), determining the extent to which new information replaces old information.
     - γ is the discount factor ($0 \leq \gamma \leq 1$), which discounts the importance of future rewards.
4. **Repeat:**
   - Continue the learning process for multiple episodes or until convergence.

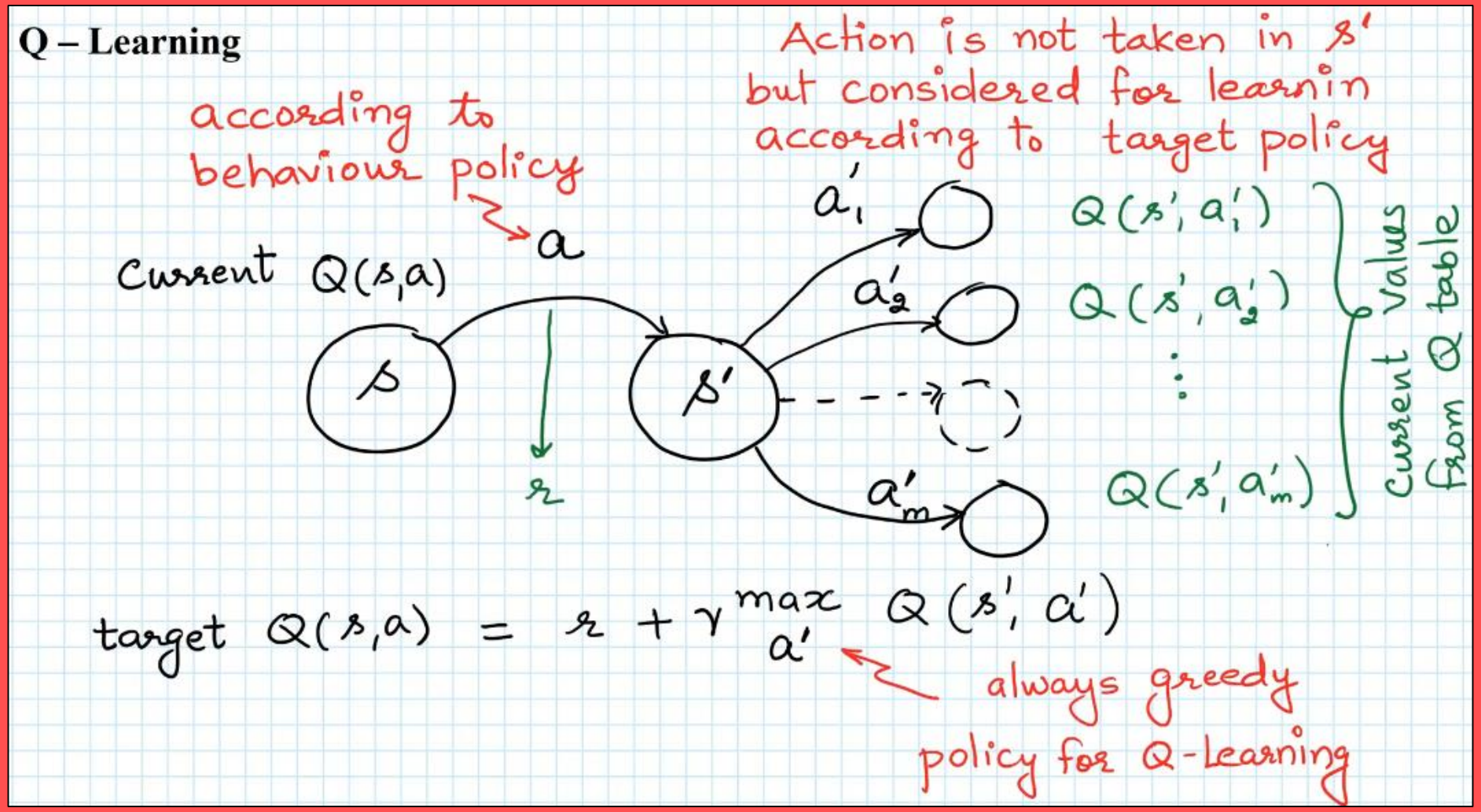
SARSA Overview

- SARSA is an on-policy algorithm, meaning it learns the policy that it follows during training.
- It is suitable for scenarios where the policy needs to be continuously updated based on the agent's actions.

SARSA vs Q-Learning

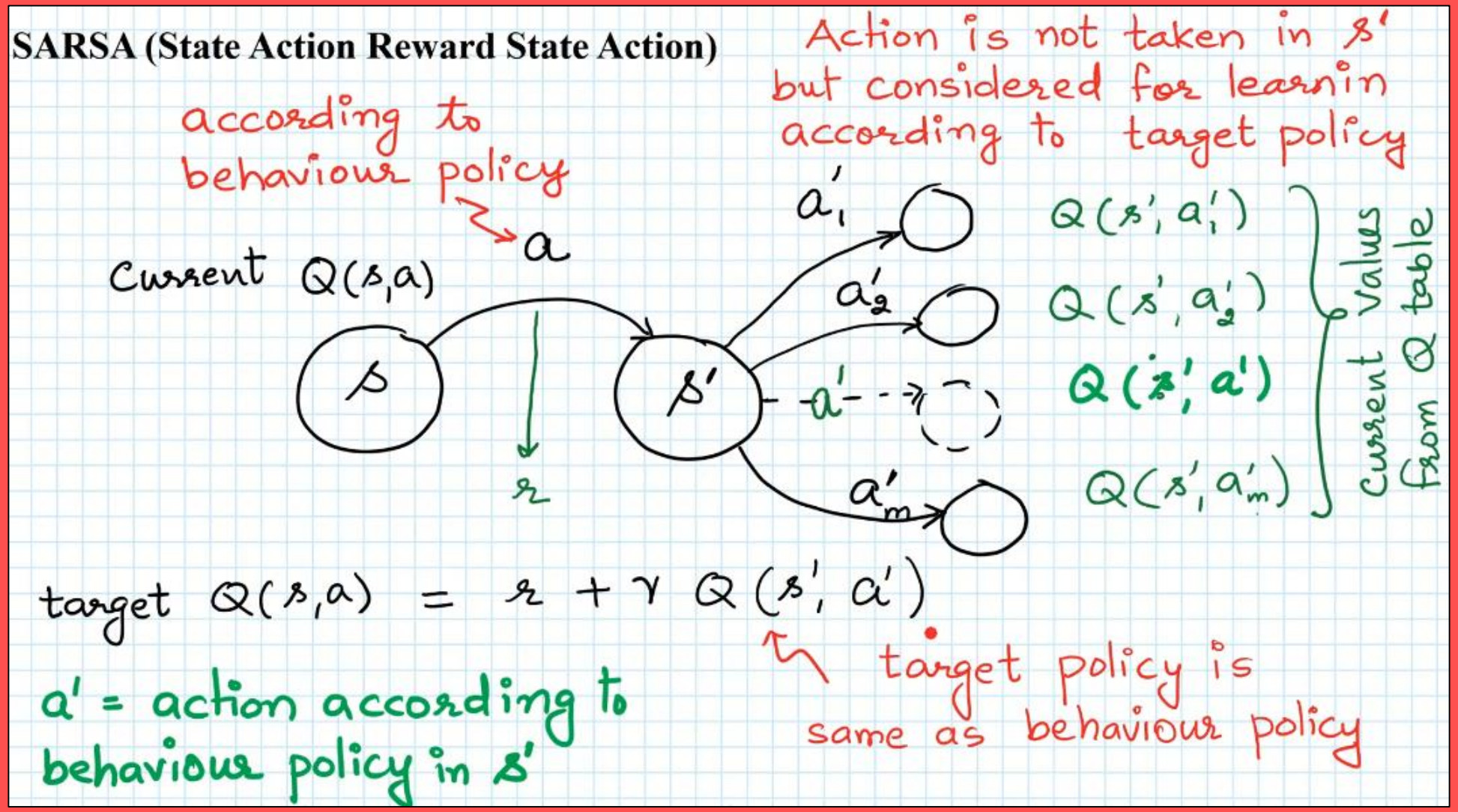
- *Q-Learning :* $Q(s_t,a_t) \leftarrow (1-\alpha) \cdot Q(s_t,a_t) + \alpha \cdot (r_{t+1} + \gamma \cdot \max_a Q(s_{t+1},a))$
- SARSA : $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \cdot (r_{t+1} + \gamma \cdot Q(s_{t+1},a_{t+1}) - Q(s_t,a_t))$
- The $(1-\alpha)$ term is not explicitly present in the SARSA update because it's implicitly accounted for in the form of α itself. The SARSA update rule directly multiplies the learning rate (α) with the temporal difference error ($r_{t+1} + \gamma \cdot Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)$).
- In Q-learning, you often see the $(1-\alpha)$ term explicitly because the Q-value update involves both the current Q-value and the maximum Q-value of the next state (which is part of the temporal difference error), and $(1-\alpha)$ scales the current estimate down, leaving room for the updated estimate.
- In summary, the absence of $(1-\alpha)$ in the SARSA update is due to how the learning rate α is incorporated directly into the update rule.

SARSA vs Q-Learning



Credit : Dr. Pankaj Kumar Porwal Source : https://www.youtube.com/watch?v=FhSaHuC0u2M

SARSA vs Q-Learning



Credit : Dr. Pankaj Kumar Porwal Source : https://www.youtube.com/watch?v=FhSaHuC0u2M

SARSA vs Q-Learning

Q – Learning (Off policy)

Updated Q Value | Current Q Value | Target Q Value | Current Q Value

$$Q(s,a) = Q(s,a) + \alpha \left[r + \max_{a'} \gamma Q(s',a') - Q(s,a) \right]$$

α = Learning Rate

Target policy is always Greedy Policy

SARSA (State Action Reward State Action) Algorithm (On policy)

Updated Q Value | Current Q Value | Target Q Value | Current Q Value

$$Q(s,a) = Q(s,a) + \alpha \left[r + \gamma Q(s',a') - Q(s,a) \right]$$

α = Learning Rate

Target Policy is always same as Behaviour Policy

Credit : Dr. Pankaj Kumar Porwal

Source : https://www.youtube.com/watch?v=FhSaHuC0u2M

SARSA vs Q-Learning

- On-Policy vs Off-Policy:
  - **Q-learning**: Off-policy. Q-learning learns the optimal Q-values regardless of the policy followed to generate the data. It estimates the maximum expected future rewards for each state-action pair.
  - **SARSA**: On-policy. SARSA learns the Q-values for the policy it is currently following. It updates Q-values based on the actual actions taken and the subsequent state-action pairs.

SARSA vs Q-Learning

- **Update Rule:**
  - **Q-learning:** The update rule is based on the maximum Q-value of the next state, irrespective of the action taken. The Q-value for a state-action pair is updated using the maximum Q-value of the next state.

    $Q(s_t,a_t) \leftarrow (1-\alpha) \cdot Q(s_t,a_t) + \alpha \cdot (r_{t+1} + \gamma \cdot max_a Q(s_{t+1},a))$

  - **SARSA:** The update rule considers the Q-value of the next state-action pair based on the policy being followed. It takes into account the action actually taken in the next state.

    $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \cdot (r_{t+1} + \gamma \cdot Q(s_{t+1},a_{t+1}) - Q(s_t,a_t))$

SARSA vs Q-Learning

- Policy Used for Exploration:
  - **Q-learning**: Typically uses an epsilon-greedy strategy for exploration. It exploits the current best action with high probability and explores randomly with a small probability.
  - **SARSA**: Also often uses epsilon-greedy, but it explores and updates its policy simultaneously, as it's an on-policy algorithm.
- Usage:
  - **Q-learning**: Can be more effective in situations where exploration is crucial, and the learned policy may deviate significantly from the behavior policy.
  - **SARSA**: Tends to be more stable in environments where the agent should follow a specific policy closely, such as in applications where safety is a primary concern.

End