Reinforcement Learning: Algorithms

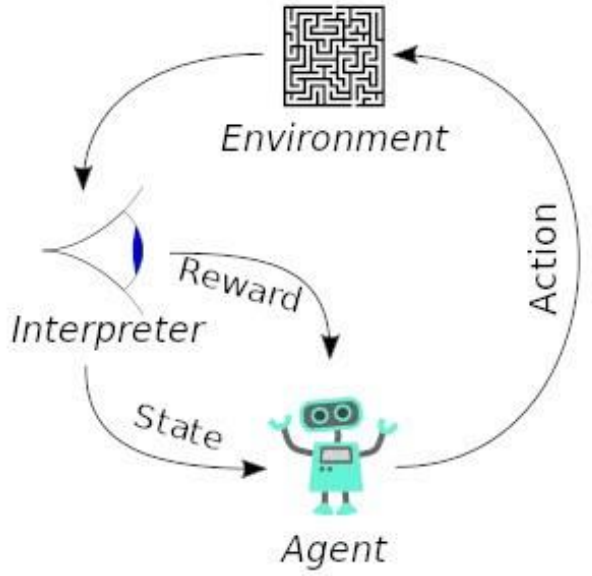
Lecture 13

Dr. Syed Maaz Shahid

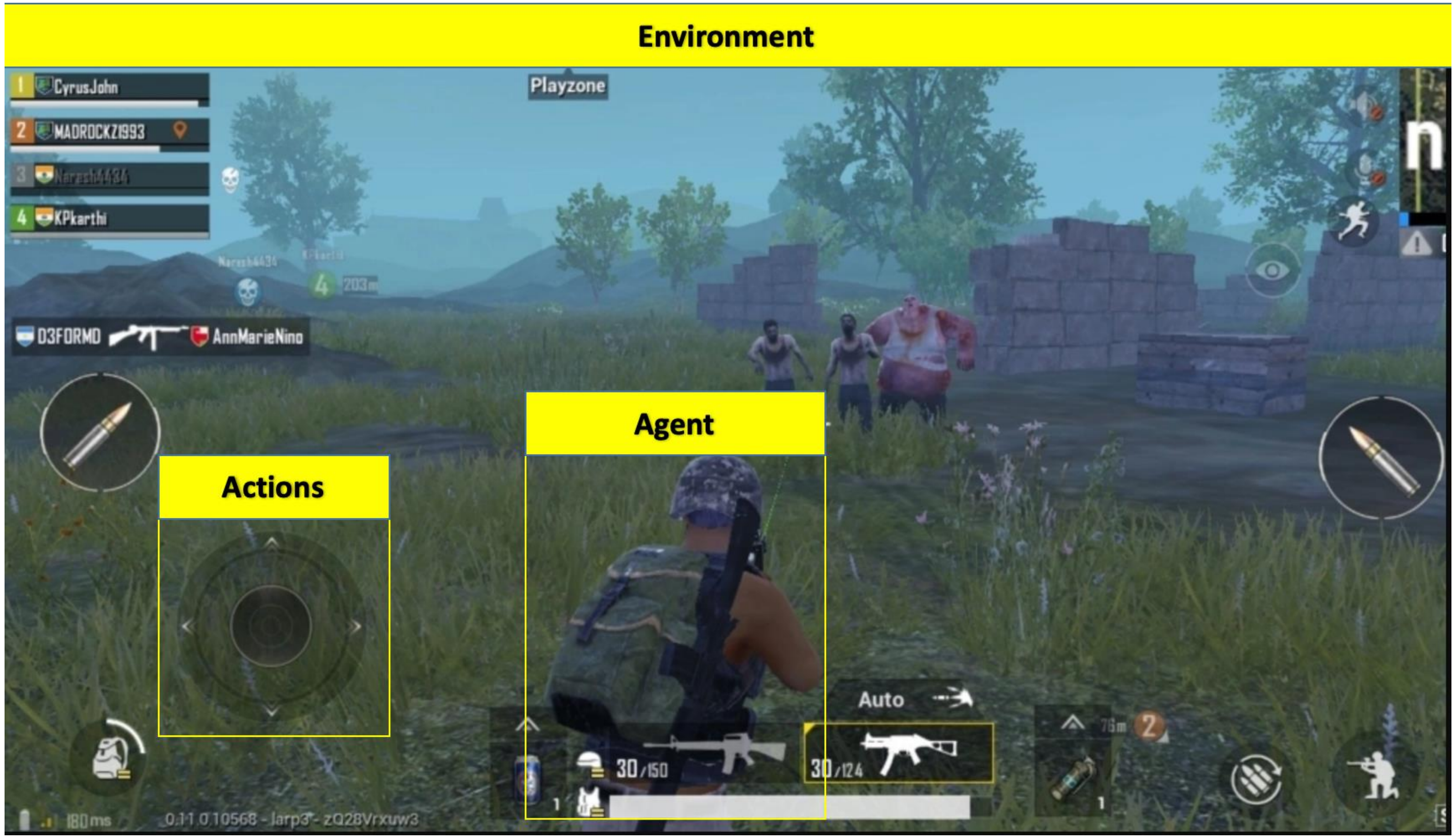
27th May,2024

Reinforcement Learning

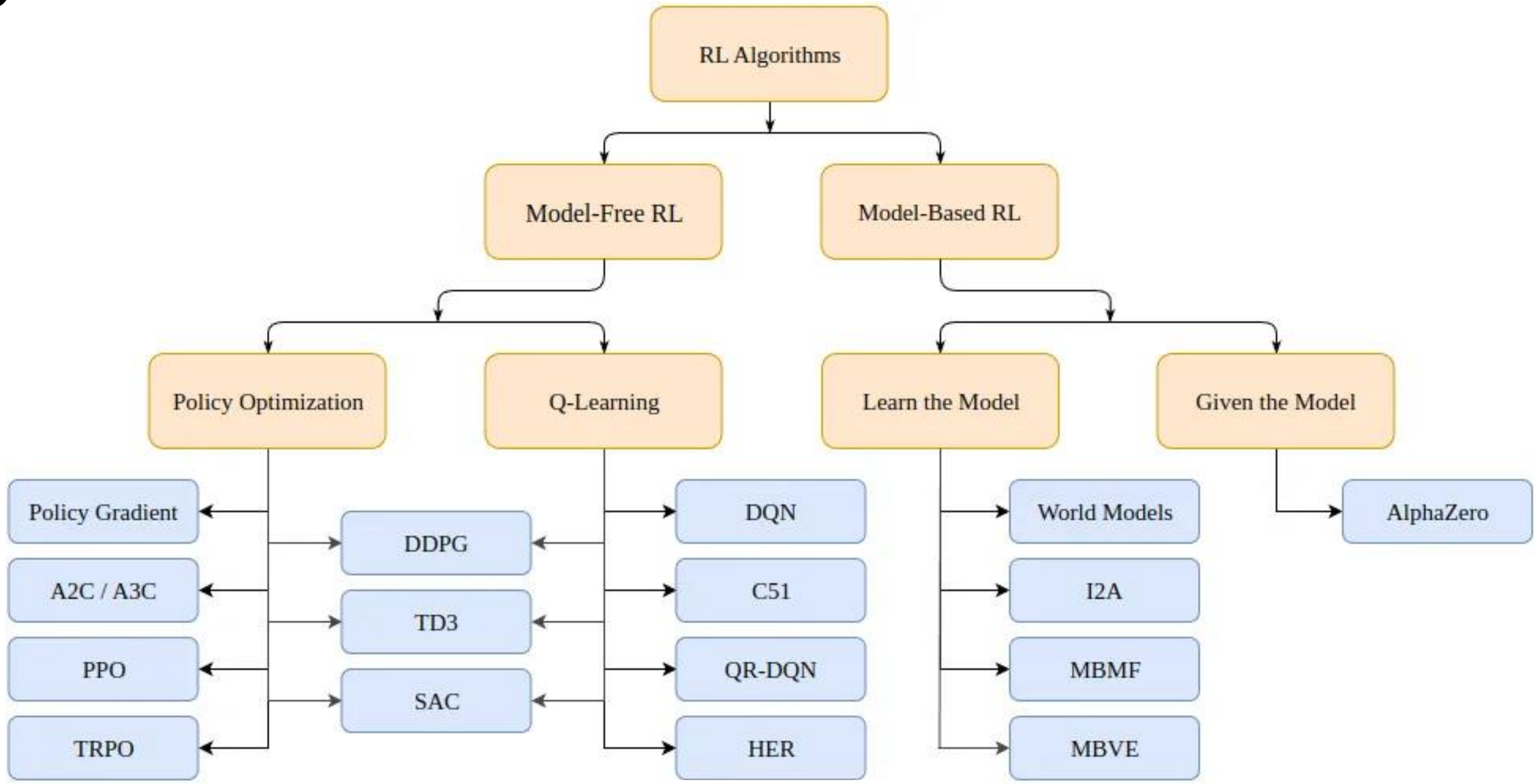
- **Agent/ Policy**
  - The program you train for specific task/Method to map agent's state to actions.
- **Environment/State**
  - The world, real or virtual, in which the agent performs actions.
- **Action**
  - A move made by agent causes a status change in environment.
- **Rewards**
  - The evaluation of an action (positive or negative)/Feedback from the environment.



Reinforcement Learning



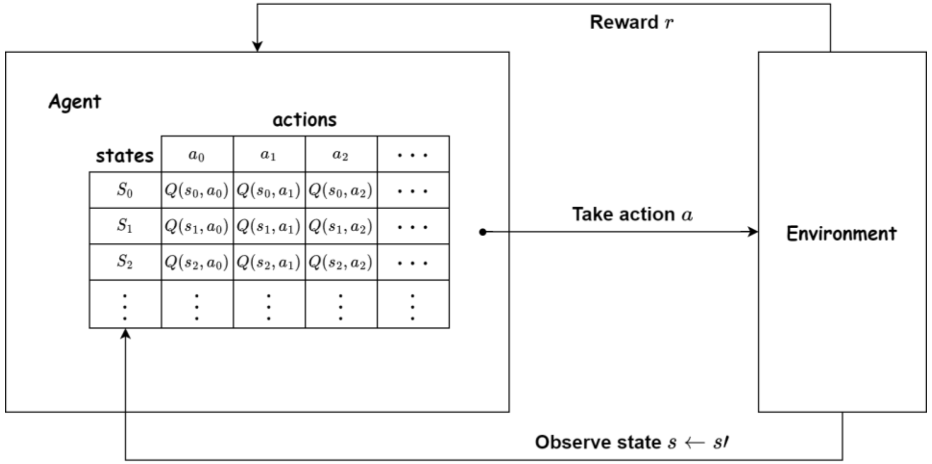
Taxonomy of Reinforcement Learning Al gorithms



OpenAI spinning up

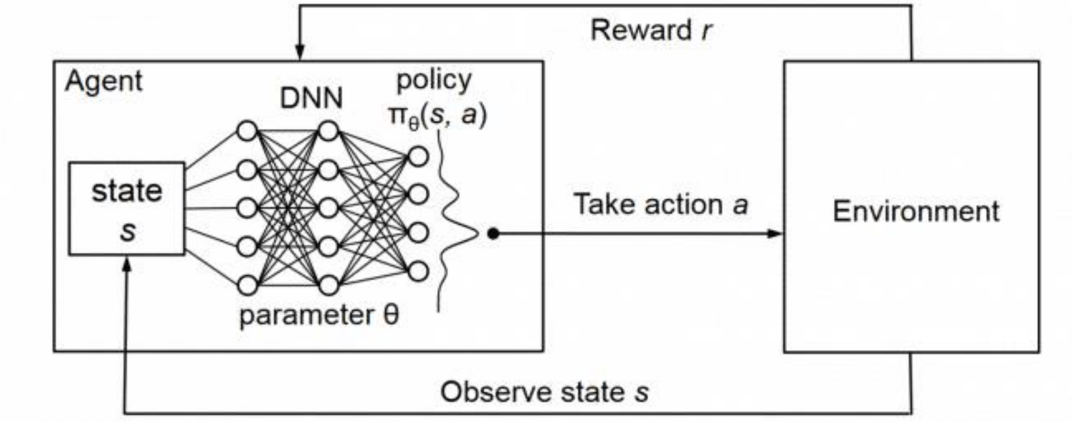
Reinforcement Learning Algorithms

- **Q-Learning:** one of the foundational algorithms in RL.
  - operates in discrete state and action spaces.
  - maintains a **Q-value table** that stores the expected cumulative rewards for each **state-action pair**.
  - *exploration and exploitation* → iteratively updates Q-values based on the agent’s experiences.
  - **Applications:** state and possible moves are well-defined



Reinforcement Learning Algorithms

- **Deep Q Networks (DQNs):** combine Q-learning with deep neural networks.
  - Handles high-dimensional state spaces, such as images or raw sensor data.
  - use **neural networks** to approximate the Q-values for different state-action pairs.
  - network is trained using experience replay ($[S_t, A_t, R_t, S_{t+1}]$)
  - **Applications**: complex environments



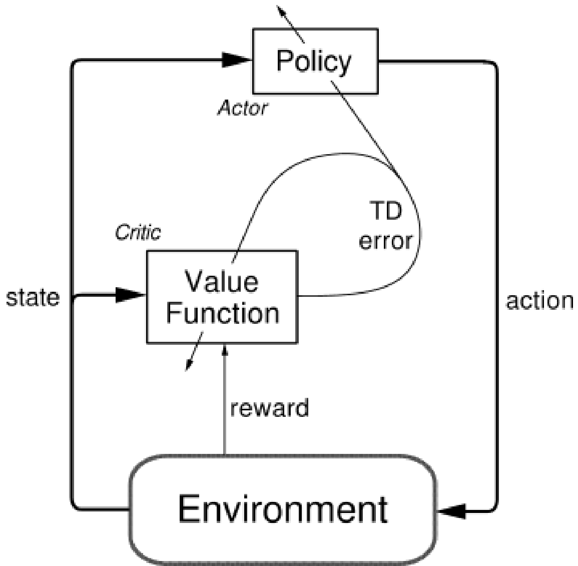
Reinforcement Learning Algorithms

- **Policy Gradient Methods:** directly optimize the policy
  - a mapping of states to actions by updating the parameters of the policy network.
  - use gradient ascent to find the policy that maximizes the expected cumulative reward.
  - **Applications:** continuous action spaces

- *Proximal Policy Optimization (PPO):* policy gradient method.
  - prevents large policy updates that could lead to unstable learning.
  - Used for training robots and autonomous systems

Reinforcement Learning Algorithms

- **Actor-Critic Methods:** combine elements of both policy-based and value-based approaches.
  - *actor-network*, akin to the policy network, selects actions,
  - *critic network* evaluates the policy's performance by estimating the value function.
  - **suitable** where data collection is expensive or time-consuming.



Dynamic Programing

- Dynamic programming (DP) refers to a collection of algorithms that can be used to compute optimal policies → given a **perfect model of the environment** as MDP.

- Key idea → use of value functions to organize and structure the search for good policies.

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.→ **Chapter 4.**

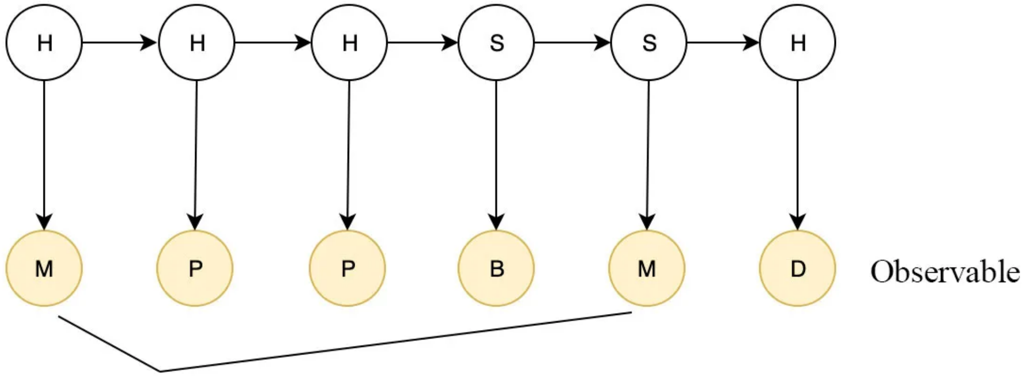
Hidden Markov Model

- An HMM is specified by the following components:
  - states
  - transition probability matrix
  - observation likelihoods (emission probabilities)
  - initial probability distribution over states

- How do we obtain the HMM?

Hidden Markov Model

internal state {H, S} is not observable or hard to determine



0.2 chance that I go to movie when I am happy.
0.4 chance that I go to movie when I am sad.

π

$P(x_1)$

$P(x_1 = happy) = 0.8$

$P(x_1 = sad\ ) = 0.2$

Initial state distribution π

A

x_{t+1}

x_t		Happy	Sad
	Happy	0.99	0.01
	Sad	0.1	0.9

For example, $P(Happy_{t+1}|Happy_t) = 0.99$

Transition probability matrix $\boldsymbol{A}$ in Markov Process

B

$P(y_t|x_t)$:

	movie	book	party	dinning
Given being happy	0.2	0.2	0.4	0.2
Given being sad	0.4	0.3	0.1	0.2

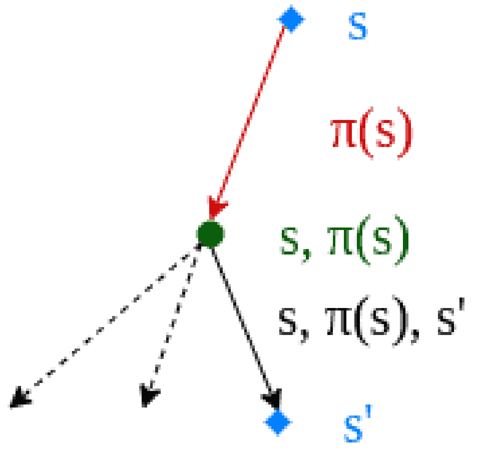
Observation likelihoods or Emission probabilities $\boldsymbol{B}$

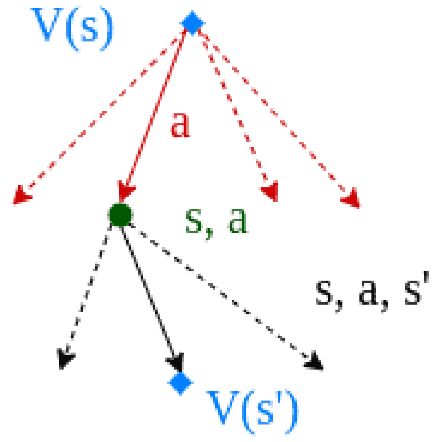
model λ

Dynamic Programing

- Policy Evaluation
- Policy Improvement
- Policy Iteration $\pi_0 \xrightarrow{\mathrm{E}} v_{\pi_0} \xrightarrow{\mathrm{I}} \pi_1 \xrightarrow{\mathrm{E}} v_{\pi_1} \xrightarrow{\mathrm{I}} \pi_2 \xrightarrow{\mathrm{E}} \cdots \xrightarrow{\mathrm{I}} \pi_* \xrightarrow{\mathrm{E}} v_*$
- Value Iteration
- Generalized Policy Iteration

Policy Iteration vs. Value Iteration





- In policy iteration, we start with a fixed policy.
- Evaluates the policy, and the other improves it.

- In value iteration, we begin by selecting the value function.
- Maximizing the utility function for all possible actions.

Generalized Policy Iteration

- Letting policy evaluation and policy improvement processes interact

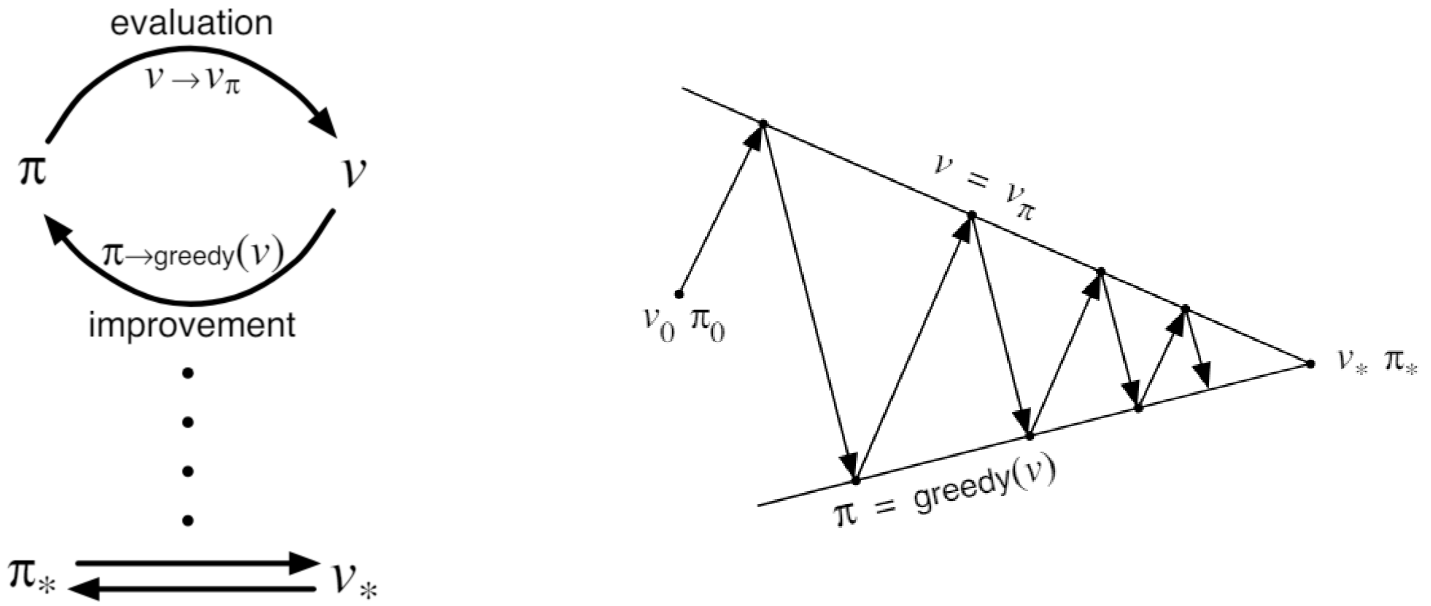
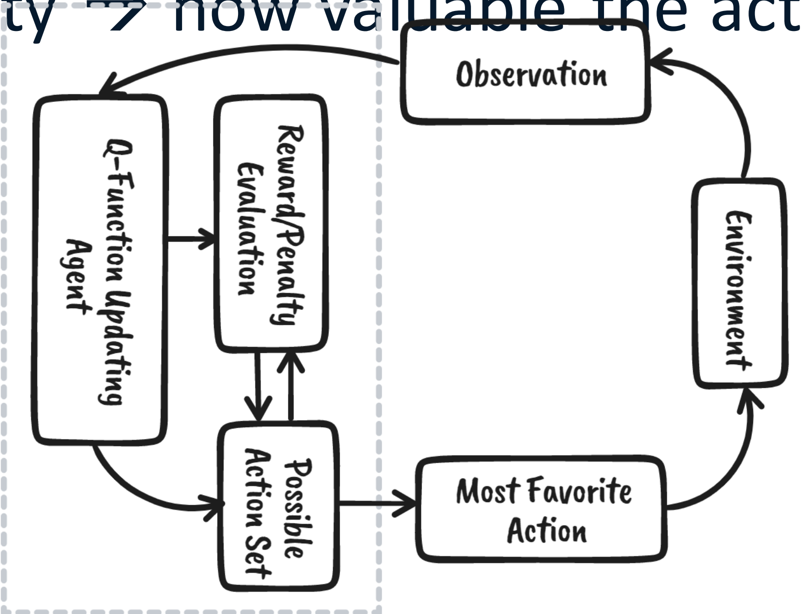


Fig: Generalized policy iteration: Value and policy functions interact until they are optimal and thus consistent with each other.

Q-learning

- Q-learning is a **model-free**, **value-based**, **off-policy** algorithm → find best series of actions based on agent's current state.

- The “Q” stands for quality → how valuable the action is in maximizing future rewards.



Model-based vs. Model-free Algorithms

- The **model-based** algorithms use transition and reward functions to estimate the optimal policy and create the model.

- The **model-free** algorithms learn the consequences of their actions through the experience without transition and reward function.

Value-based vs. Policy-based Methods

- The **value**-based method trains the value function to learn which state is more valuable and take action.

- The **policy**-based methods train the policy directly to learn which action to take in a given state.

Off-policy vs. On-policy

- In the **off**-policy, the algorithm evaluates and updates a policy that differs from the policy used to take an action.

- The **on**-policy algorithm evaluates and improves the same policy used to take an action.

Key Terminologies in Q-learning

- **States(s):** the current position of the agent in the environment.
- **Action(a):** a step taken by the agent in a particular state.
- **Rewards:** for every action, the agent receives a reward and penalty.
- **Episodes:** the end of the stage, where agents can’t take new action.
- **Q($\boldsymbol{S_{t+1}}$, a):** expected optimal Q-value of doing action in particular state.
- **Q($\boldsymbol{S_t}$, $\boldsymbol{A_t}$):** it is the current estimation of Q(St+1, a).
- **Q-Table:** the agent maintains the Q-table of sets of states and actions.
- **Temporal Differences(TD):** used to estimate expected value of Q(S_{t+1}, a) by using the current state and action and previous state and action.

Q-Table

- The agent will use a Q-table to take the best possible action based on the expected reward for each state.

- Q-table is a data structure of sets of actions and states,
  - Q-learning algorithm to update values in the table.

- The Q-function uses the **Bellman equation** and takes state(s) and action(a) as input.

$$\underbrace{\boxed{Q^{\pi}(s_t, a_t)}}_{\substack{\text{Q-Values for the state} \\ \text{given a particular state}}} = \underbrace{\boxed{E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots}}_{\substack{\text{Expected discounted} \\ \text{cumulative reward}}} \underbrace{\boxed{s_t, a_t]}}_{\text{Given the state and action}}$$

Bellman Equation

- Concept that comes from the field of dynamic programming.
- Value function to be defined as the sum of the immediate reward and the subsequent reward.

$$\underbrace{V_\pi(s)}_{\text{Value of state s}} = \underbrace{\mathbf{E}_{\boxed{\pi}}[R_{t+1}}_{\text{Expected value of immediate reward}} + \underbrace{\gamma * V_\pi(S_{t+1})}_{\text{+ the discounted value of next\_state}} \,|\, \underbrace{S_t = s]}_{\text{If the agent starts at state s}}$$

And uses the policy to choose its actions for all time steps

Value Function

- Value function $v_\pi(s)$ is a prediction of future reward

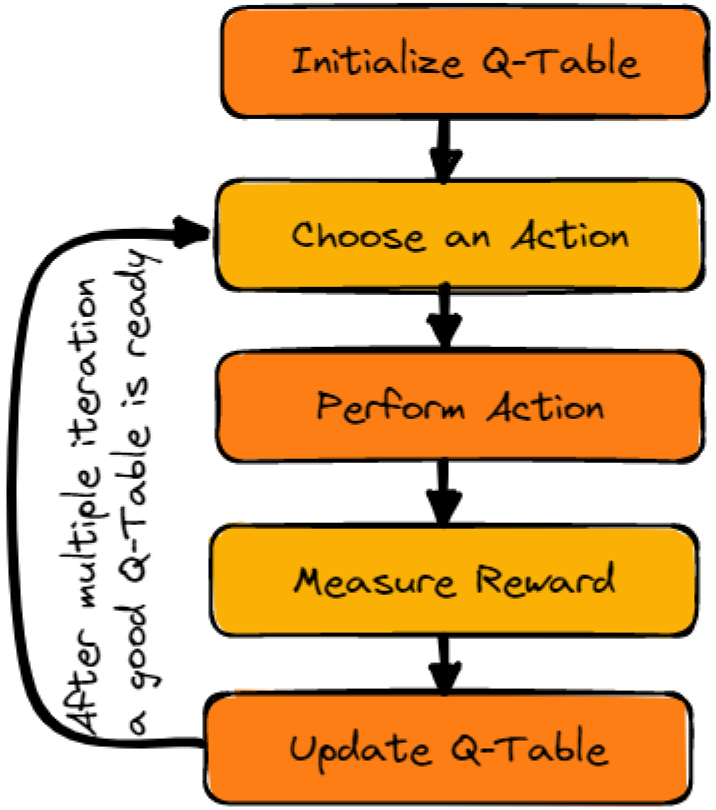
- Used to evaluate the goodness/badness of states

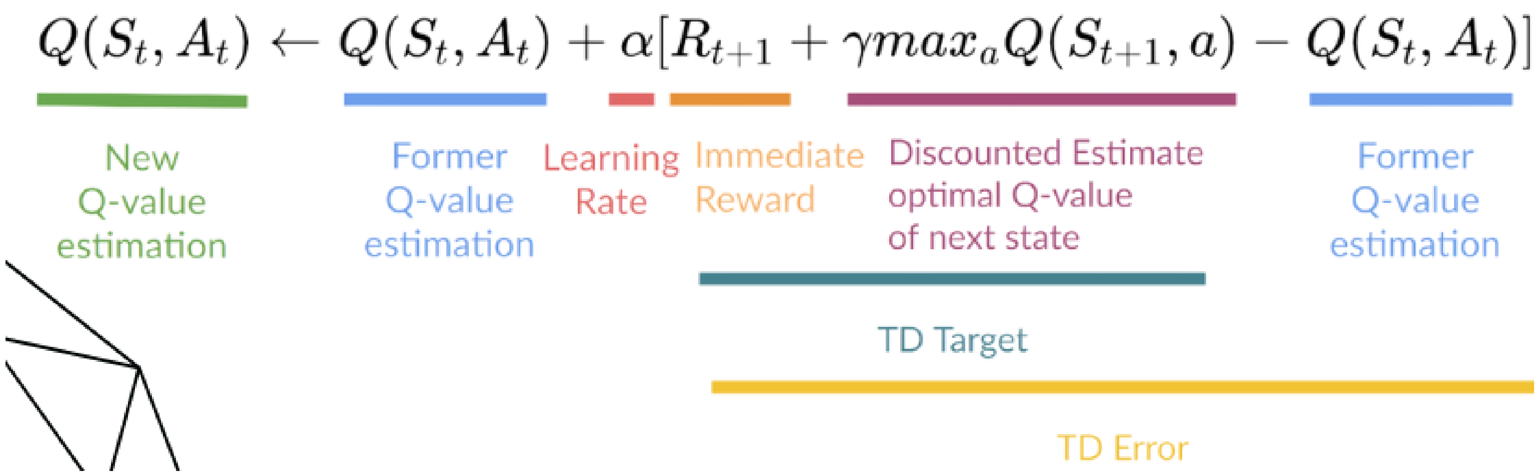
- By following a policy π, the value function is defined as

$$v_\pi(s) = E[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots | S_t = s]$$

  - $0 \leq \gamma \leq 1$: *discount rate*
  - γ close to 1*:* rewards further in the future count more→ agent is fars ighted

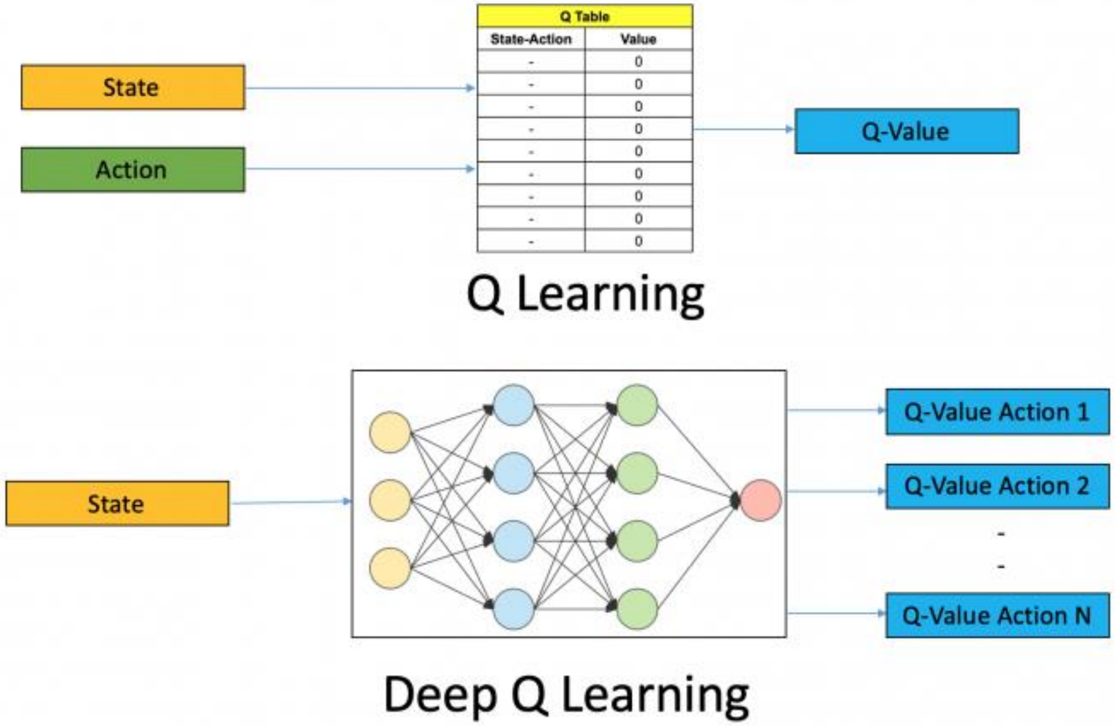
Q-learning algorithm





Deep Q-Networks (DQN)

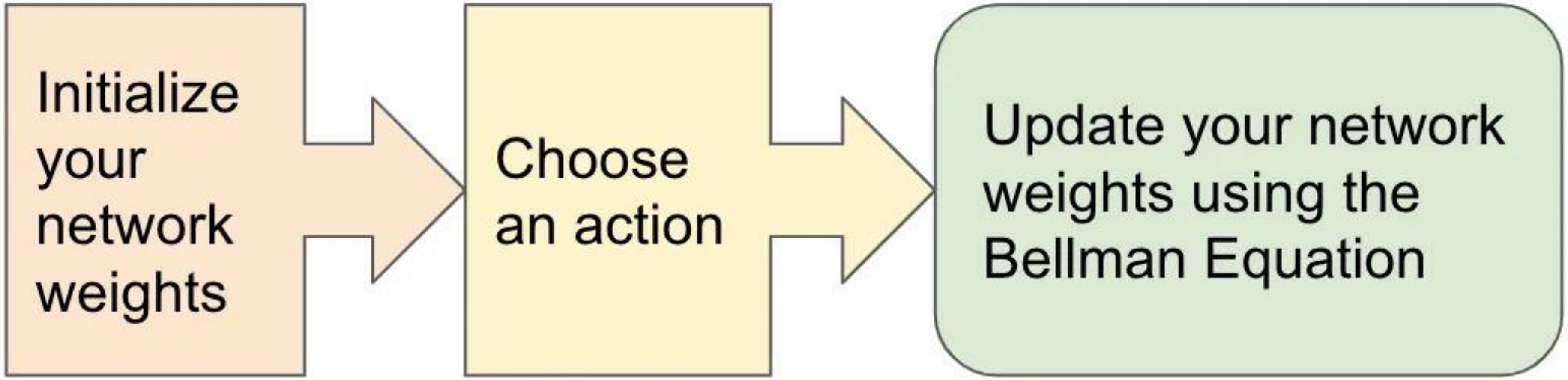
- Rather than mapping a **state-action** pair to a **Q-value**, a neural network maps input states to **(action, Q-value)** pairs.



Why DQN?

- For example: Learning to play chess.
  - Impossible to memorize every possible board configuration and the best move for each.
  - Better to general strategies and principles (like controlling the center of the board and protecting the king).
- Neural networks allows DQNs to handle environments with large or continuous state spaces.

DQN Algorithm

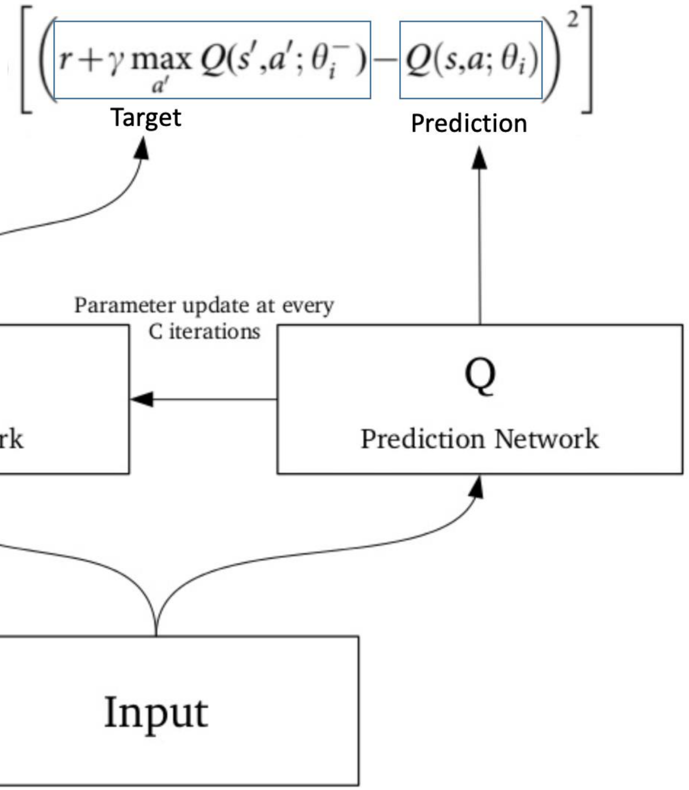


DQN Algorithm

- DQN algorithm has 2 networks: **main network** & **target network**
- Both networks have the same architecture but different weights.
- Every N steps, the weights from the **main network** are copied to the **target network**.

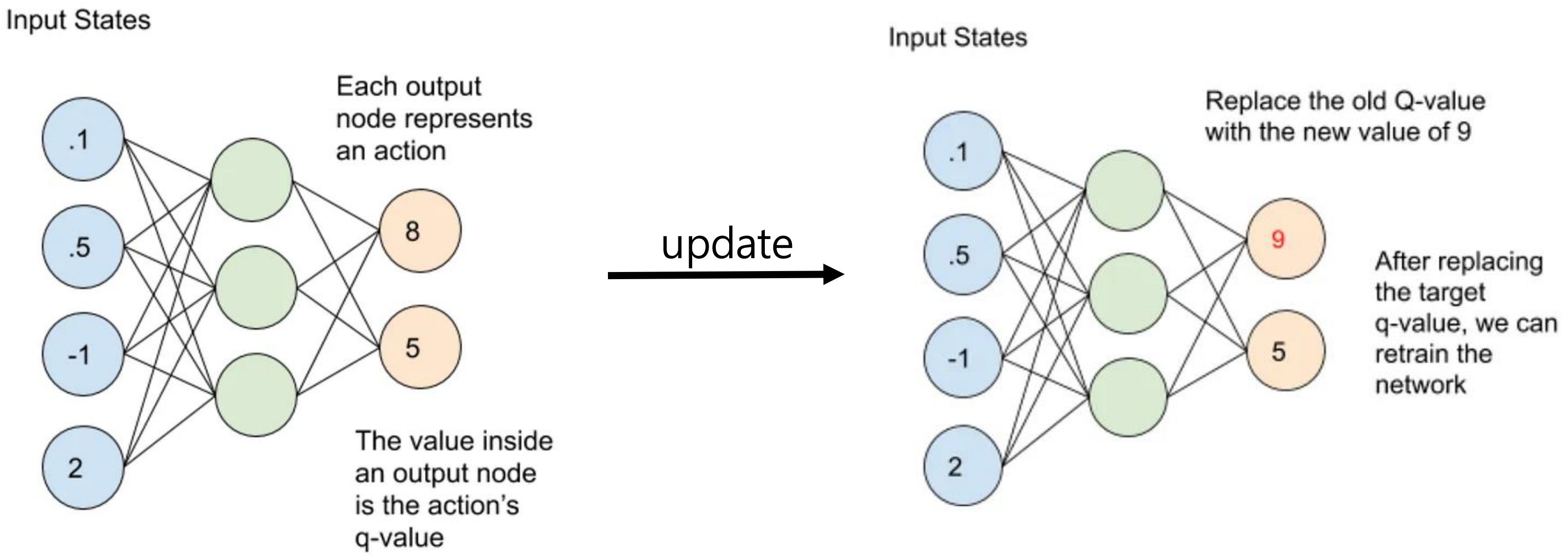
DQN Algorithm

- More stable training because it keeps the target function fixed (for a while)



DQN Algorithm

- How to map States to (Action, Q-value) pairs



DQN Algorithm

- **Experience Replay** is act of storing and replaying game states ($[S_t, A_t, R_t, S_{t+1}]$) that RL algorithm is able to learn from.
  - Deep Q-Learning uses Experience Replay to learn in small batches in order to avoid skewing the dataset.

- For example, if we're teaching a self-driving car how to drive,
  - first part of road is just a straight line, agent might not learn how to deal with any curves → This is where experience replay comes in.

  - The initial experiences of driving in a straight line don't get put through the neural network right away.

DQN Algorithm-Action Selection Policies

- **Epsilon ϵ-Greedy** to balance exploration and exploitation by choosing between exploration and exploitation randomly.

$$\text{Action at time(t)} \begin{cases} \max Q_t(a) & \text{with probability } 1-\epsilon \\ \text{any action (a)} & \text{with probability } \epsilon \end{cases}$$

Actor-critic (A2C) Method

- A2C a Temporal Difference(TD) version of Policy gradient.
  - It has two networks: Actor and Critic.
  - Actor decided which **action** should be taken and critic inform the actor how **good** was the action and how it should **adjust**.
  - Learning of the actor is based on **policy gradient** approach.
  - Critics evaluate the action produced by the actor by computing the **value** function.

- Generative Adversarial Network (GAN) similar to A2C?
  - *Discriminator* and *Generator* participate in a game.

Next Lecture

- RL algorithm for real-world problem in wireless networks
- Monte Carlo Methods and Temporal-difference Learning