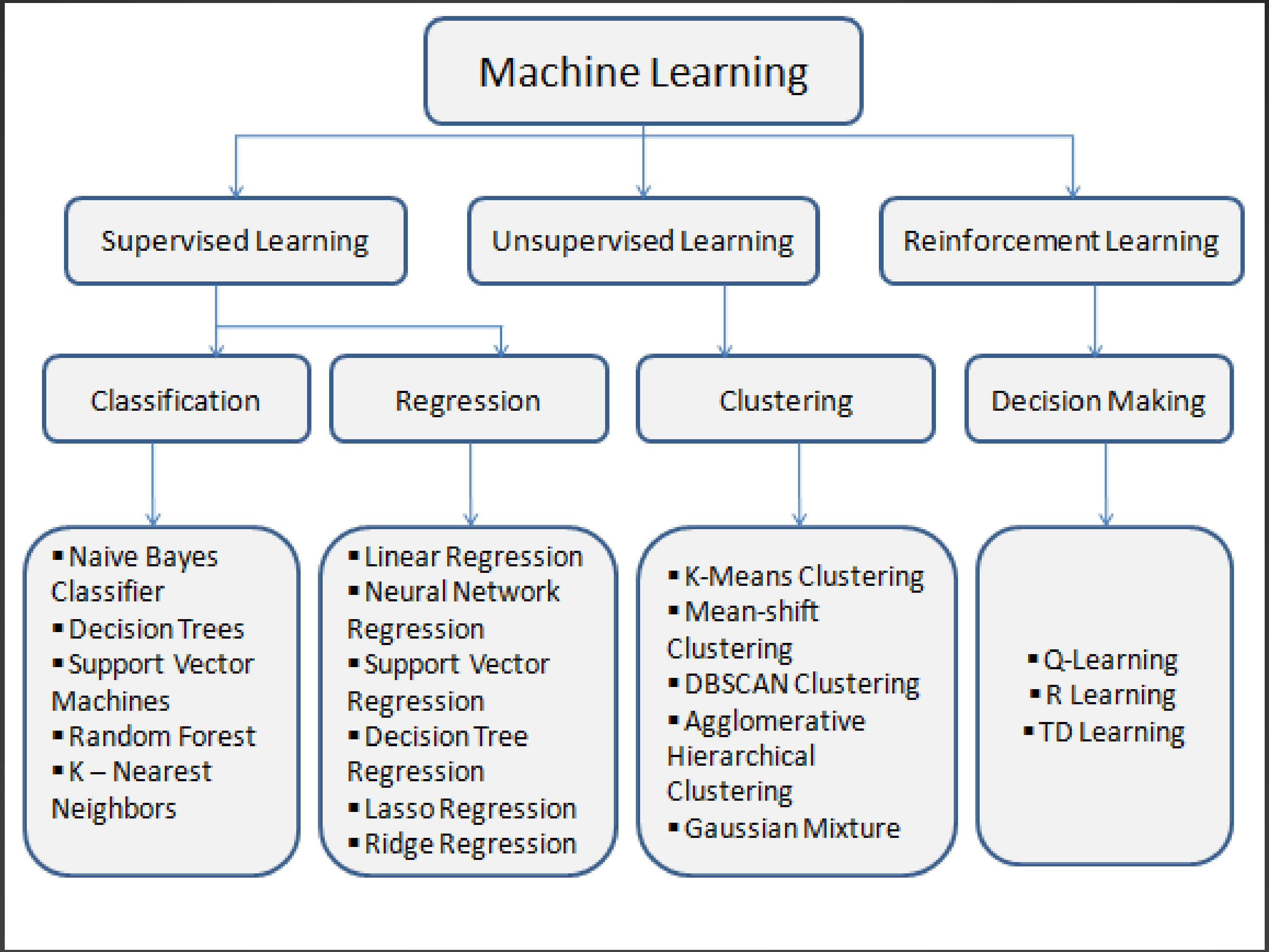
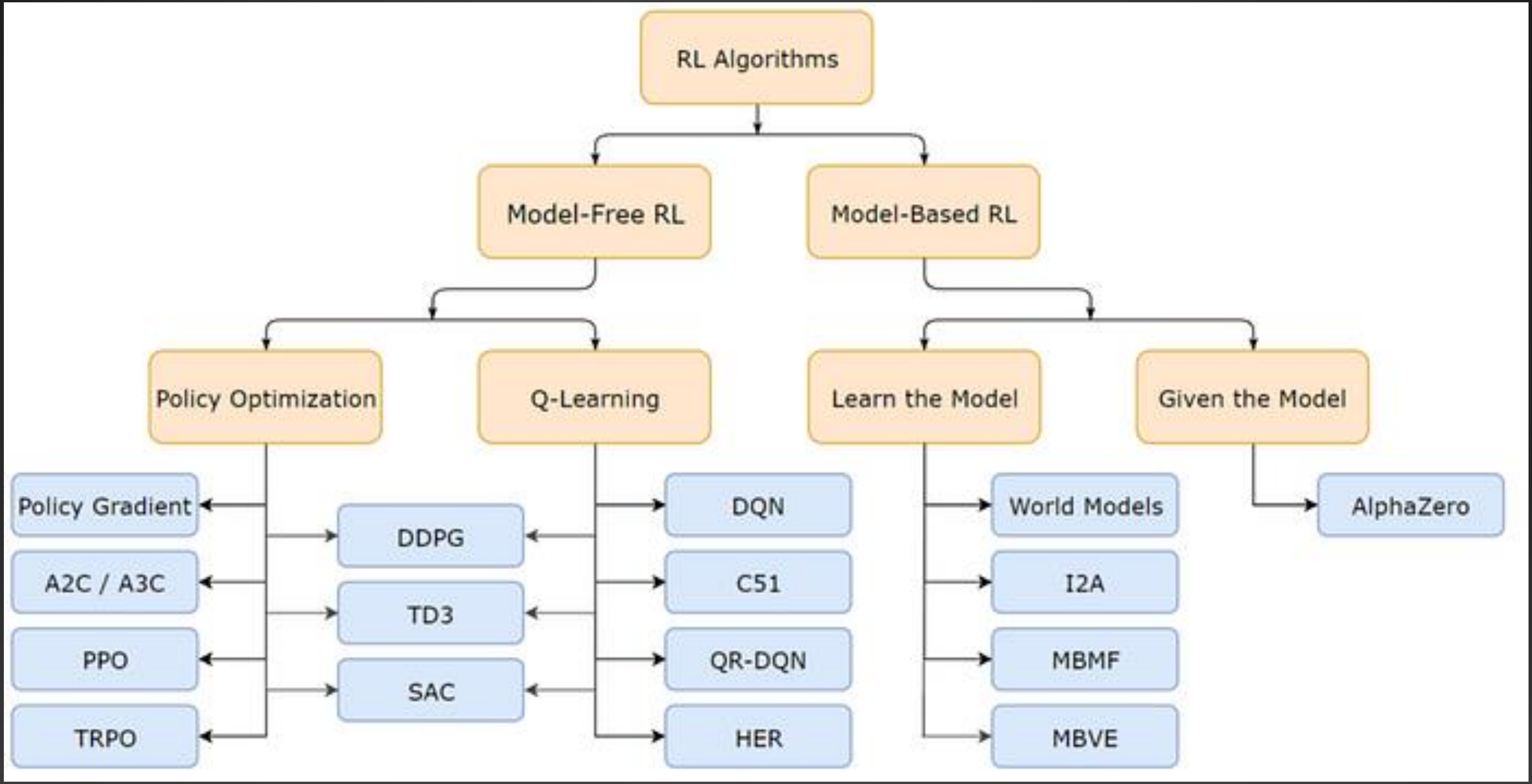
INTRODUCTION TO REINFORCEMENT LEARNING

- WHY RL
- MARKOV MODELS/MDP
- DP
- RL
- DEEP REINFORCEMENT LEARNING

DIFFERENT TYPES OF MACHINE LEARNING

- Supervised
  - Classification
  - Regression
- Unsupervised
  - Clustering
- Semi-Supervised
- Reinforcement Learning


Machine Learning
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Classification
Regression
Clustering
Decision Making
▪ Naive Bayes Classifier
▪ Decision Trees
▪ Support Vector Machines
▪ Random Forest
▪ K – Nearest Neighbors
▪ Linear Regression
▪ Neural Network Regression
▪ Support Vector Regression
▪ Decision Tree Regression
▪ Lasso Regression
▪ Ridge Regression
▪ K-Means Clustering
▪ Mean-shift Clustering
▪ DBSCAN Clustering
▪ Agglomerative Hierarchical Clustering
▪ Gaussian Mixture
▪ Q-Learning
▪ R Learning
▪ TD Learning


RL Algorithms
Model-Free RL
Model-Based RL
Policy Optimization
Q-Learning
Learn the Model
Given the Model
Policy Gradient
A2C / A3C
PPO
TRPO
DDPG
TD3
SAC
DQN
C51
QR-DQN
HER
World Models
I2A
MBMF
MBVE
AlphaZero

WHY REINFORCEMENT LEARNING?

How does RL differ?

Supervised Learning :

- Known Ground Truth

Advantages?

No Ground Truth data is needed

Adjust to unknown scenarios

Learn by Interacting with the environment

HOW DO WE LEARN?

We mostly learn from observing the Environment.

We learn from what we sense when we interact with people around us and our environment.

But how do we learn from those inputs? What happens inside our Brain?

One way is to have some form of a feedback to rank or rate our actions!!

REINFORCEMENT, NEGATIVE REINFORCEMENT AND PUNISHMENTS

Reinforcement is the act of strengthening or reinforcing.

Negative Reinforcement is the idea of Reinforcing desired behaviours or actions through removal of negative rewards

You behave bad you deserve to be punished.

Does this really help us learn?

DRAWBACKS

Longer Training Time

Difficulty in defining the reward function.

Not all Problems can be modeled as reinforcement learning problems.

- Must be Discrete.
- Specific number of state spaces.

HOW CAN WE MODEL THIS?

TERMS USED IN REINFORCEMENT LEARNING

- Agent(): An entity that can perceive/explore the environment and act upon it.
- Environment(): A situation in which an agent is present or surrounded by. In RL, we assume the stochastic environment, which means it is random in nature.
- Action(): Actions are the moves taken by an agent within the environment.
- State(): State is a situation returned by the environment after each action taken by the agent.
- Reward(): A feedback returned to the agent from the environment to evaluate the action of the agent.
- Policy(): Policy is a strategy applied by the agent for the next action based on the current state.
- Value(): It is expected long-term retuned with the discount factor and opposite to the short-term reward.
- Q-value(): It is mostly similar to the value, but it takes one additional parameter as a current action (a).

KEY FEATURES OF REINFORCEMENT LEARNING

- In RL, the agent is not instructed about the environment and what actions need to be taken.
- It is based on the hit and trial process.
- The agent takes the next action and changes states according to the feedback of the previous action.
- The agent may get a delayed reward.
- The environment is stochastic, and the agent needs to explore it to reach to get the maximum positive rewards.

APPROACHES TO IMPLEMENT REINFORCEMENT LEARNING

- **Value-based:**
- The value-based approach is about to find the optimal value function, which is the maximum value at a state under any policy. Therefore, the agent expects the long-term return at any state(s) under policy π.
- **Policy-based:**
- Policy-based approach is to find the optimal policy for the maximum future rewards without using the value function. In this approach, the agent tries to apply such a policy that the action performed in each step helps to maximize the future reward.
  - The policy-based approach has mainly two types of policy:
  - Deterministic: The same action is produced by the policy (π) at any state.
  - Stochastic: In this policy, probability determines the produced action.

- **Model-based:** In the model-based approach, a virtual model is created for the environment, and the agent explores that environment to learn it. There is no particular solution or algorithm for this approach because the model representation is different for each environment.

ELEMENTS OF REINFORCEMENT LEARNING

- Policy
- Reward Signal
- Value Function
- Model of the environment

MDPS TO THE RESCUE

Andrey Markov

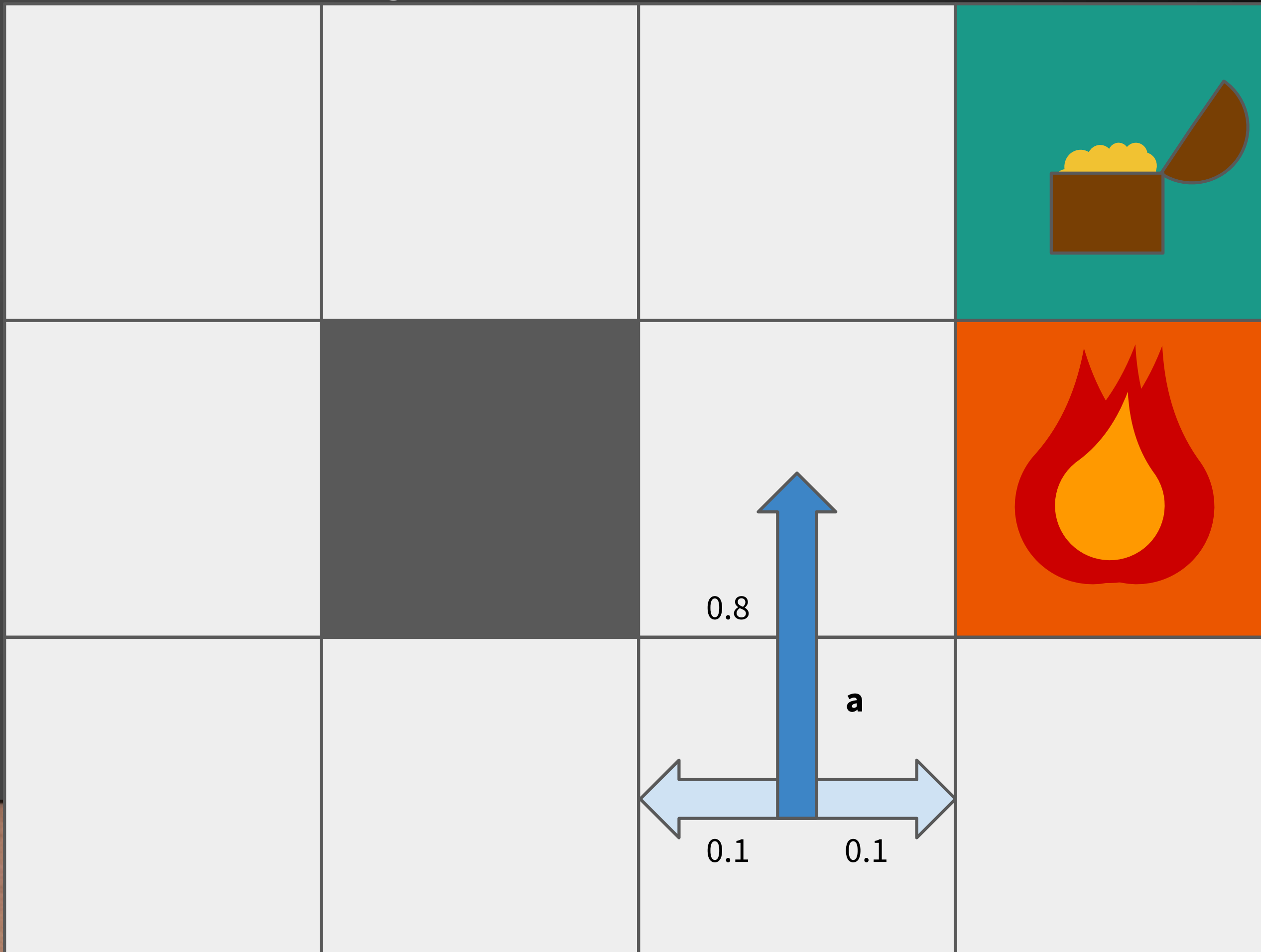
- What is a MDP?
  - Markov Decision Process
  - A mathematical way of representing states, actions and the transition model with the Rewards.
  - Probabilistic Model
  - Discrete
  - Finite

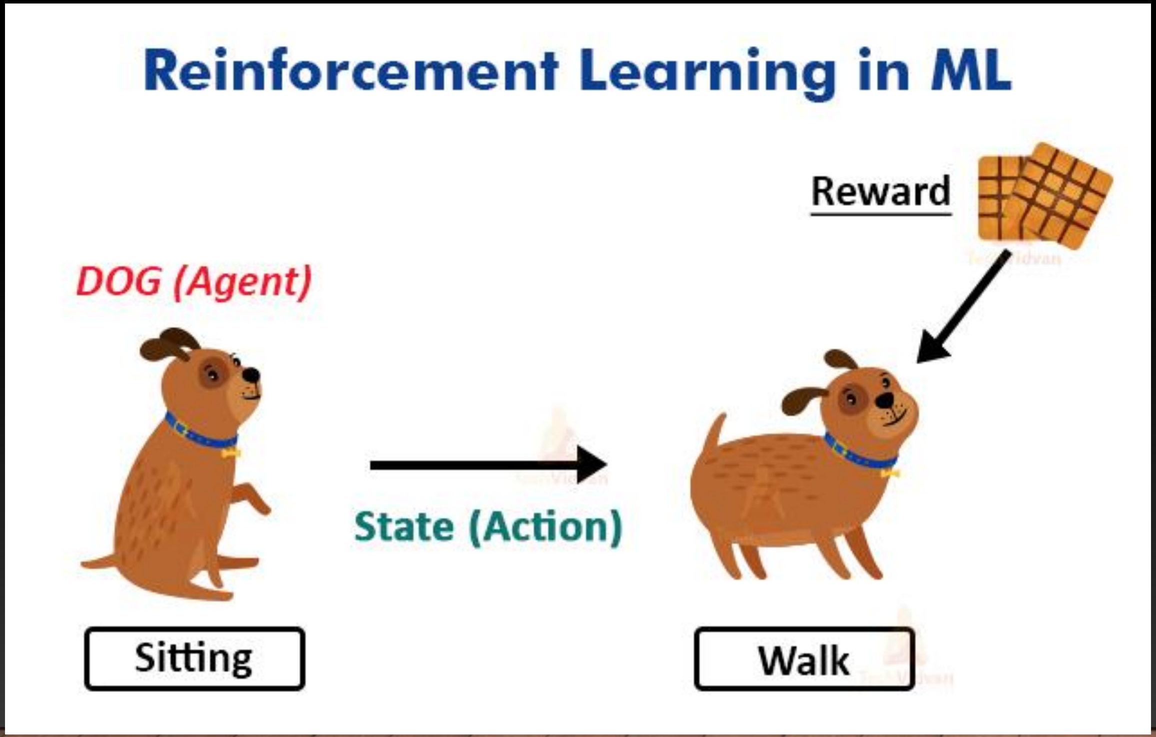
CONVERTING REAL WORLD SCENARIOS TO VIRTUAL MAPS

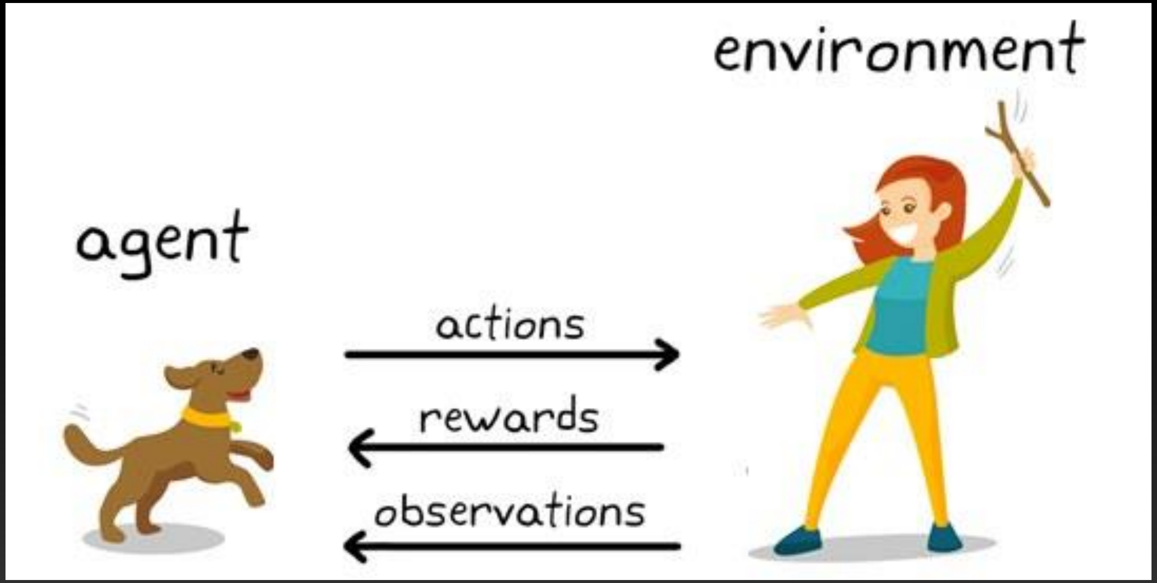
- States : Different Places the Agent can be in within the Environment
- Actions : Things the Agent can do when at a certain State
- Rewards : The Feedback environment gives for Actions taken by the Agent
- Many Frameworks that support building these environments exists.
  - OpenAI Gym

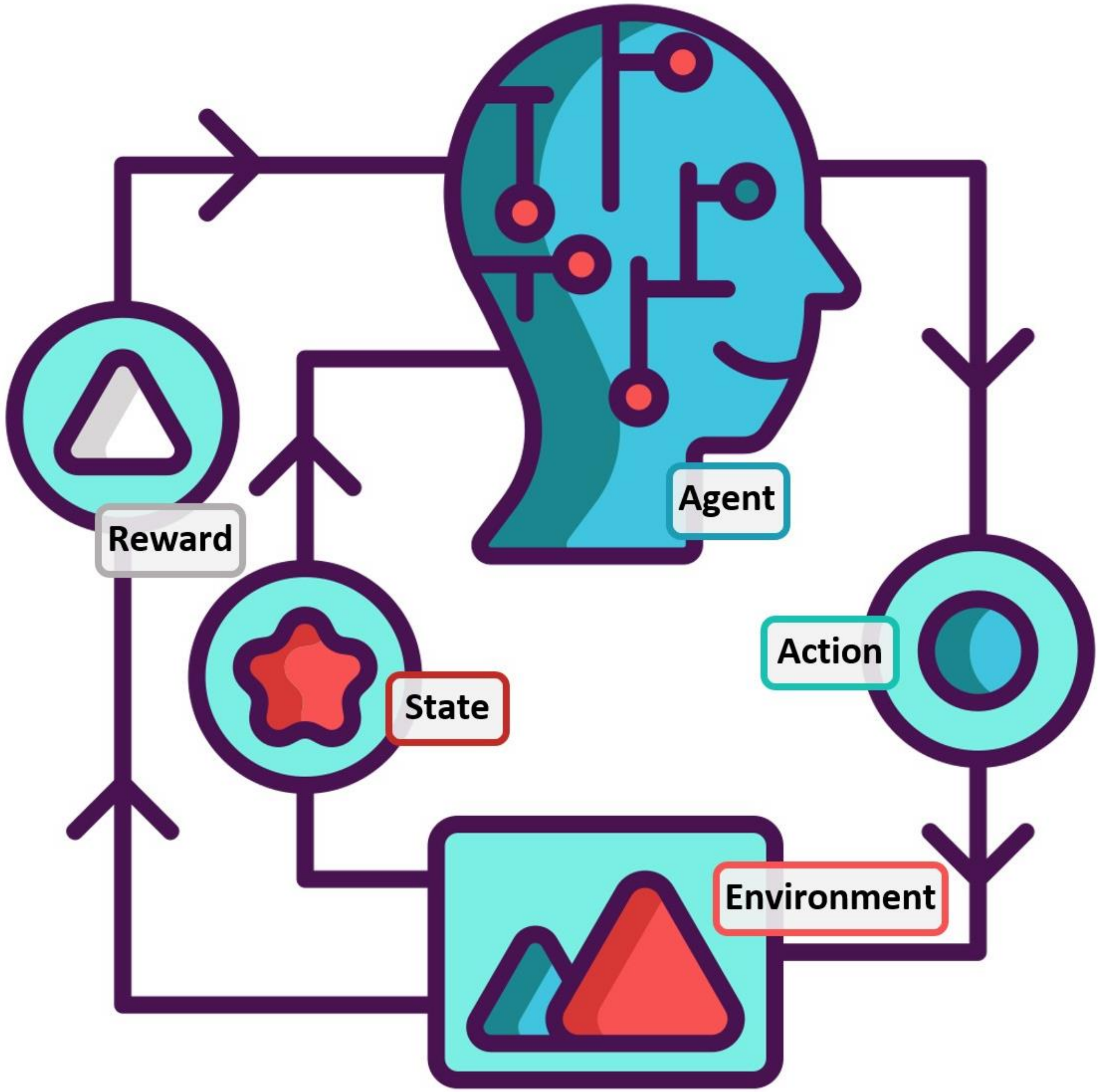
TRANSITIONS AND TRANSITION PROBABILITIES

- Sometimes Actions Do not go as Planned.
  - Case 1 :
    - If we take action a from state s, there is a 100% chance we go to s'
  - Case 2:
    - If we take action a from state s,
      - 0.8 Probability of going to s1
      - 0.1 probability of going to s2
      - 0.1 probability of going to s3
    - Probabilities must add up to 1.0 (100%)




Reinforcement Learning in ML
Reward
DOG (Agent)
State (Action)
Sitting
Walk


environment
agent
actions
rewards
observations


Agent
Reward
State
Action
Environment

SCENARIO 1

- We have a Sweeper Robot(1'x1') that is supposed to Clean a Room(5'x 5'). Robot can move Up, Down, Left or Right. Robot starts at the bottom Mid Square of the room. There is a Dirt Patch towards the center of the room(4,4). Goal of the agent is to Reach the Dirt Patch.

CAN WE USE OTHER STRUCTURES INSTEAD OF GRIDS FOR MDP?

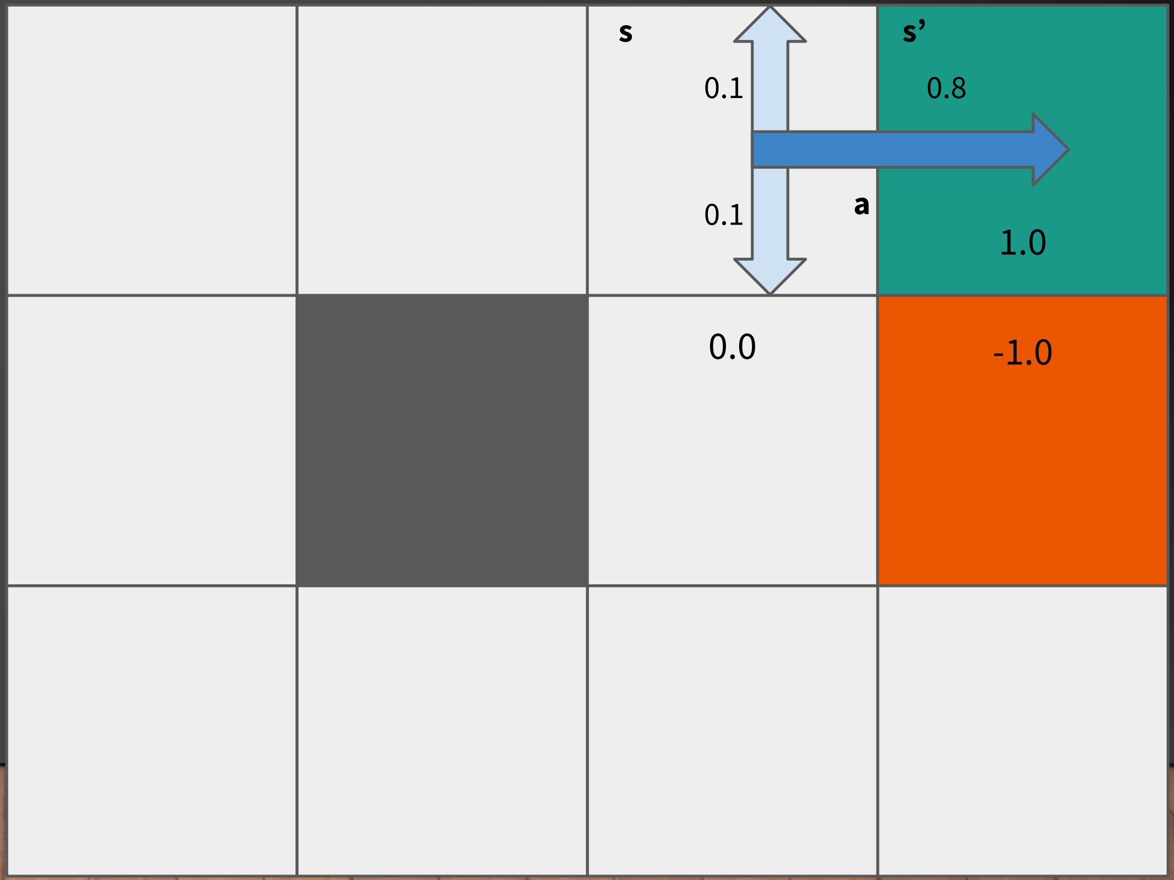
CAN WE USE AN MDP TO LEARN?

- Without AI/ML can we Learn the best policy to take?
  - Use Brute Force methods - NO LEARNING AT ALL
- Dynamic Programming
  - Value Iteration
  - Policy Iteration
- Advantages
  - Faster and efficient than Brute force methods
  - Guaranteed Convergence

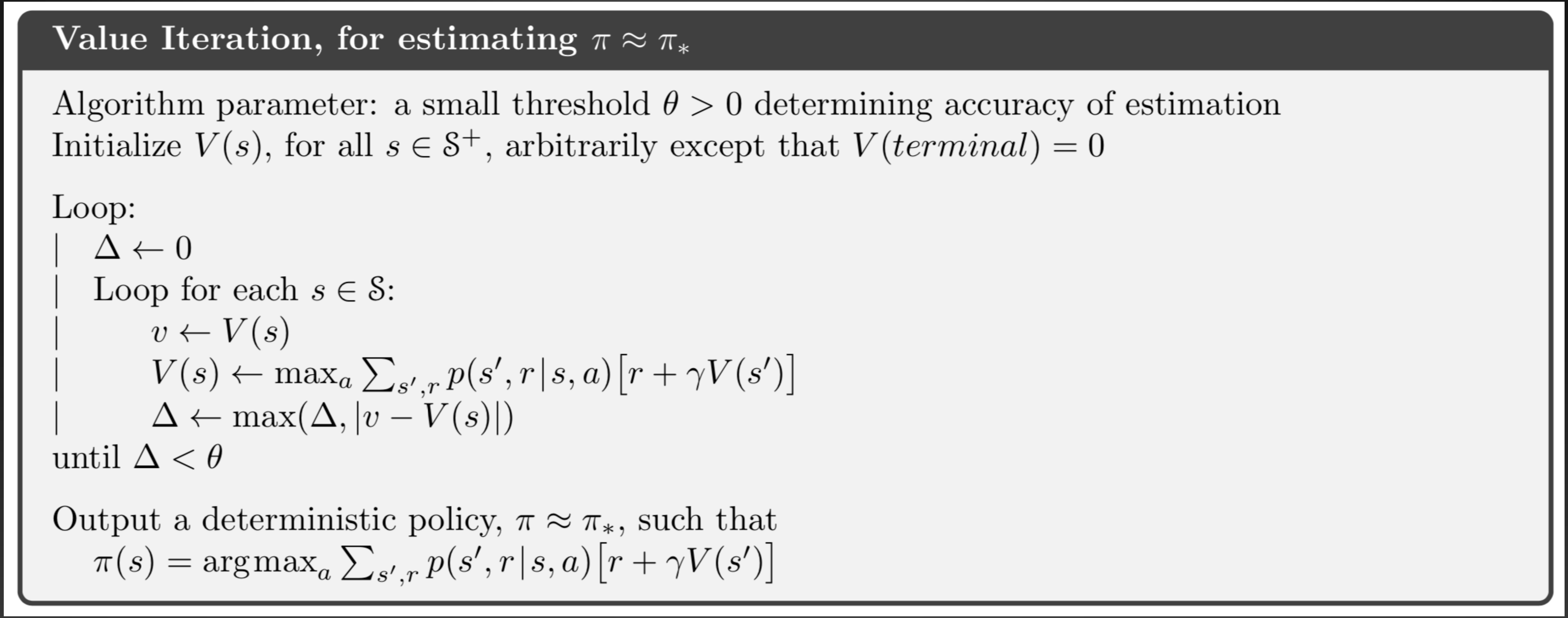
VALUE ITERATION

- Use Bellman Equations to calculate the value of a state.

$$\sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$



VALUE ITERATION



Value Iteration, for estimating $\pi \approx \pi_*$

```
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
|   $\Delta \leftarrow 0$
|   Loop for each $s \in \mathcal{S}$:
|       $v \leftarrow V(s)$
|       $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
|       $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
    $\pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
```

GRID WORLD VISUALIZATION

- https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html

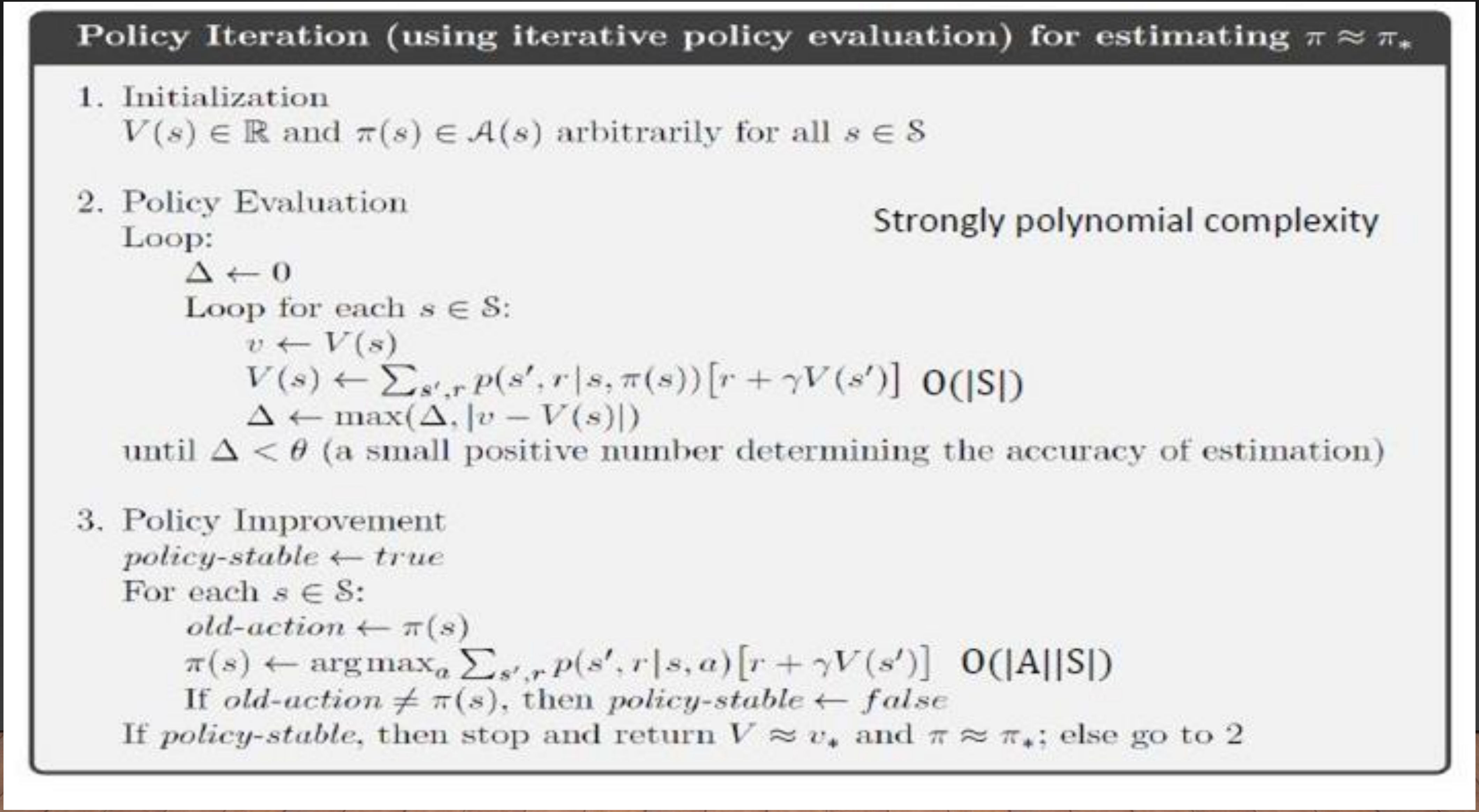
SCENARIO 2

- We have a Sweeper Robot(1’X1’) that is supposed to Clear a Room(5’X 5’). Robot can move Up, Down, Left or Right. Robot starts at the bottom Mid Square of the room. There is a Dirt Patch towards the center of the room(4,4). Goal of the agent is to Reach the Dirt Patch with **MINIMAL Moves**.

SCENARIO 3

- Now assume that the floor is damaged at (4,3), (3,5). If the robot moves into any one of the above sections, it cannot move out. How would you change the MDP model built in Scenario 2 to accommodate this?

POLICY ITERATION



Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

```
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
       $\Delta \leftarrow 0$
       Loop for each $s \in \mathcal{S}$:
           $v \leftarrow V(s)$
           $V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))[r + \gamma V(s')]$   O(|S|)
           $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ true
   For each $s \in \mathcal{S}$:
       old-action $\leftarrow \pi(s)$
       $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$   O(|A||S|)
       If old-action $\neq \pi(s)$, then policy-stable $\leftarrow$ false
   If policy-stable, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2
```

Strongly polynomial complexity

DO YOU THINK DP ALGORITHMS ACTUALLY LEARNS THE ENVIRONMENT/ PROBLEM?

MODEL-BASED VS. MODEL FREE

- Model here Refers to State-Transitions Model
  - Not a ANN Model or a RL Model

- Model-Based Algorithms Require a known State Transition Model
- Model Free Algorithms Interact with the environment and learn the model

DRAWBACKS?

- Requires the Environmental Model to be known
  - Transition Probabilities
- Expensive Computations
  - Iterating over all the sates
  - Convergence takes a longer Time

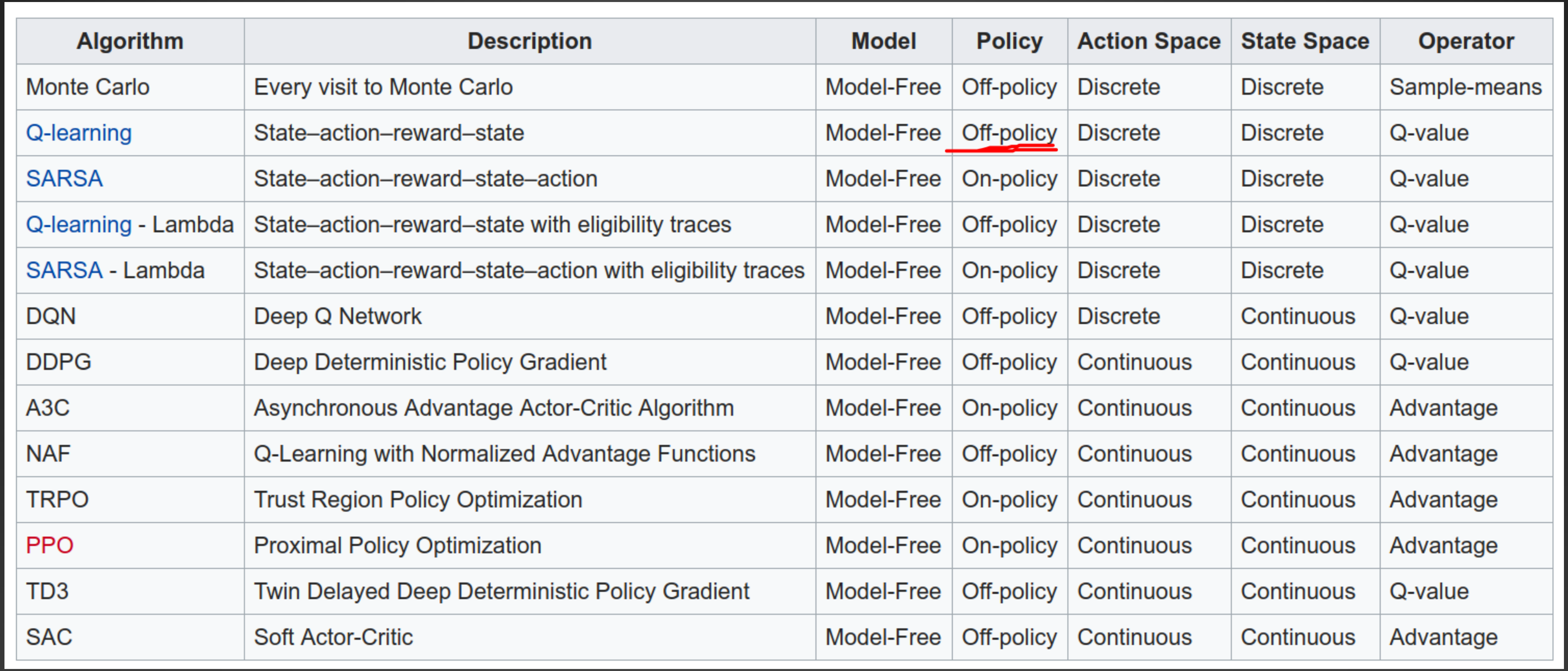
RL TO THE RESCUE

- Can Handle Problems where the Model is not Known
- Can Handle Problems with more State transitions
- Much more efficient than DP methods
- Examples:
  - Q- Learning
  - SARSA (State-Action-Reward-State-Action)

ON-POLICY VS OFF-POLICY

- On policy algorithms follow the existing policy when calculating the next Q-value update
- Off-Policy algorithms use a Greedy approach and uses the max Q-value when computing the Q-update

COMPARISON OF RL ALGORITHMS



Algorithm	Description	Model	Policy	Action Space	State Space	Operator
Monte Carlo	Every visit to Monte Carlo	Model-Free	Off-policy	Discrete	Discrete	Sample-means
Q-learning	State–action–reward–state	Model-Free	Off-policy	Discrete	Discrete	Q-value
SARSA	State–action–reward–state–action	Model-Free	On-policy	Discrete	Discrete	Q-value
Q-learning - Lambda	State–action–reward–state with eligibility traces	Model-Free	Off-policy	Discrete	Discrete	Q-value
SARSA - Lambda	State–action–reward–state–action with eligibility traces	Model-Free	On-policy	Discrete	Discrete	Q-value
DQN	Deep Q Network	Model-Free	Off-policy	Discrete	Continuous	Q-value
DDPG	Deep Deterministic Policy Gradient	Model-Free	Off-policy	Continuous	Continuous	Q-value
A3C	Asynchronous Advantage Actor-Critic Algorithm	Model-Free	On-policy	Continuous	Continuous	Advantage
NAF	Q-Learning with Normalized Advantage Functions	Model-Free	Off-policy	Continuous	Continuous	Advantage
TRPO	Trust Region Policy Optimization	Model-Free	On-policy	Continuous	Continuous	Advantage
PPO	Proximal Policy Optimization	Model-Free	On-policy	Continuous	Continuous	Advantage
TD3	Twin Delayed Deep Deterministic Policy Gradient	Model-Free	Off-policy	Continuous	Continuous	Q-value
SAC	Soft Actor-Critic	Model-Free	Off-policy	Continuous	Continuous	Advantage

https://en.wikipedia.org/wiki/Reinforcement_learning

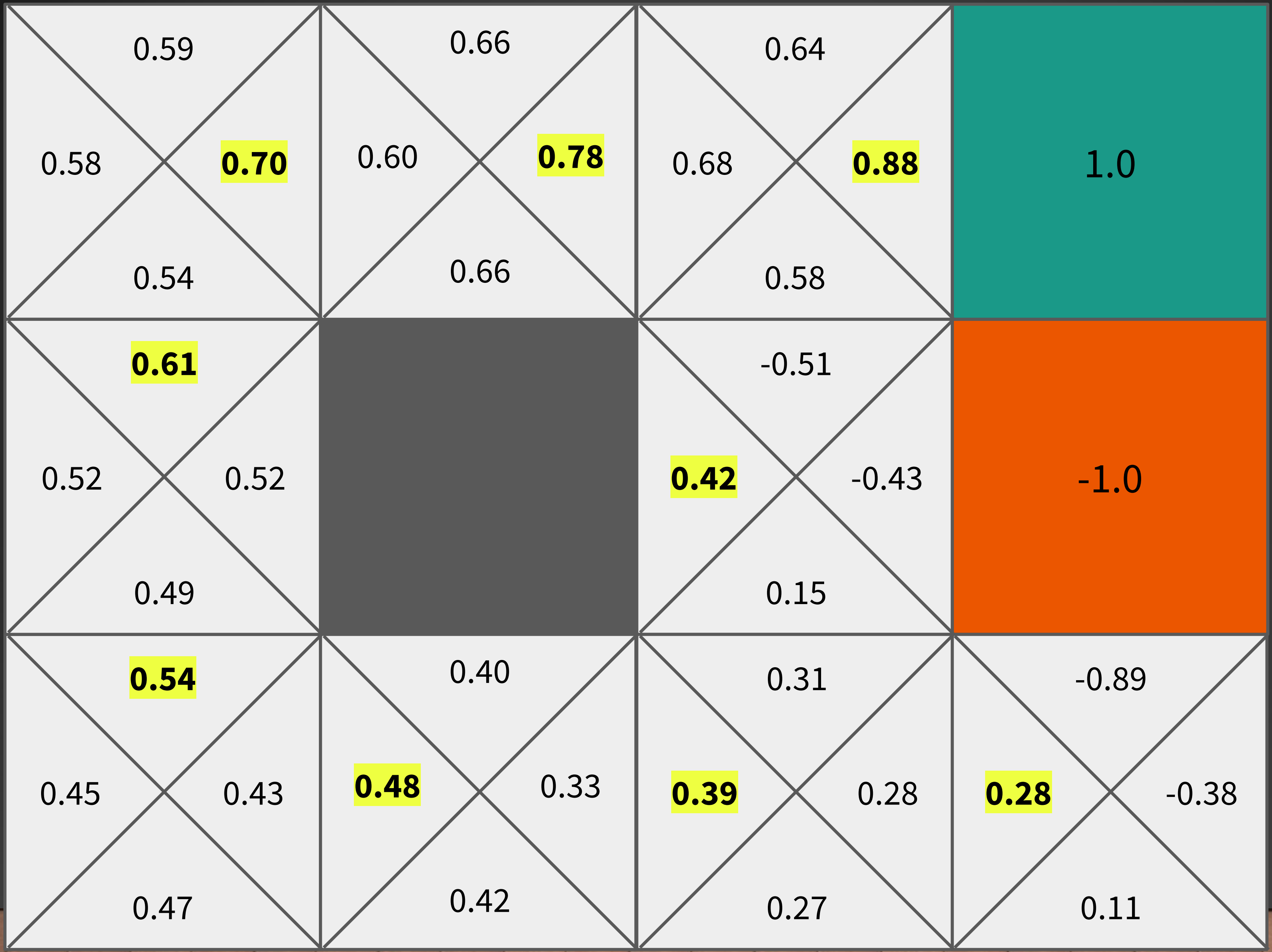
Q LEARNING

- Update State-Action Values (Q-values) instead of the policy

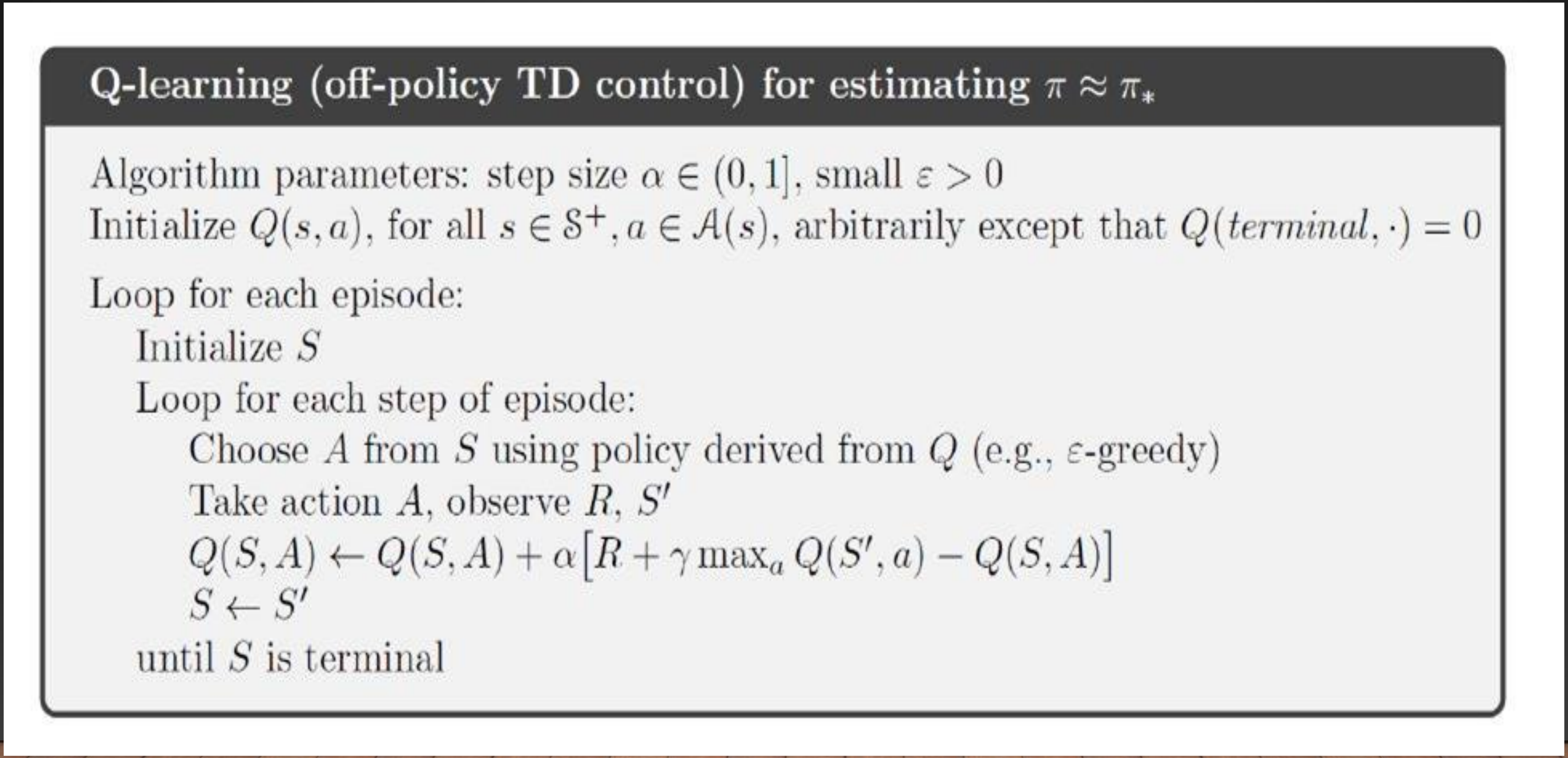
- Use a Bellman Equation to update the Q values

$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha(\hat{Q}(s,a) - Q_k(s,a))$$

$$\hat{Q}(s,a) = r + \gamma \max_{a'} Q_k(s',a')$$


0.59
0.58
0.70
0.54
0.66
0.60
0.78
0.66
0.64
0.68
0.88
0.58
1.0
0.61
0.52
0.52
0.49
-0.51
0.42
-0.43
0.15
-1.0
0.54
0.45
0.43
0.47
0.40
0.48
0.33
0.42
0.31
0.39
0.28
0.27
-0.89
0.28
-0.38
0.11

Q - LEARNING



Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

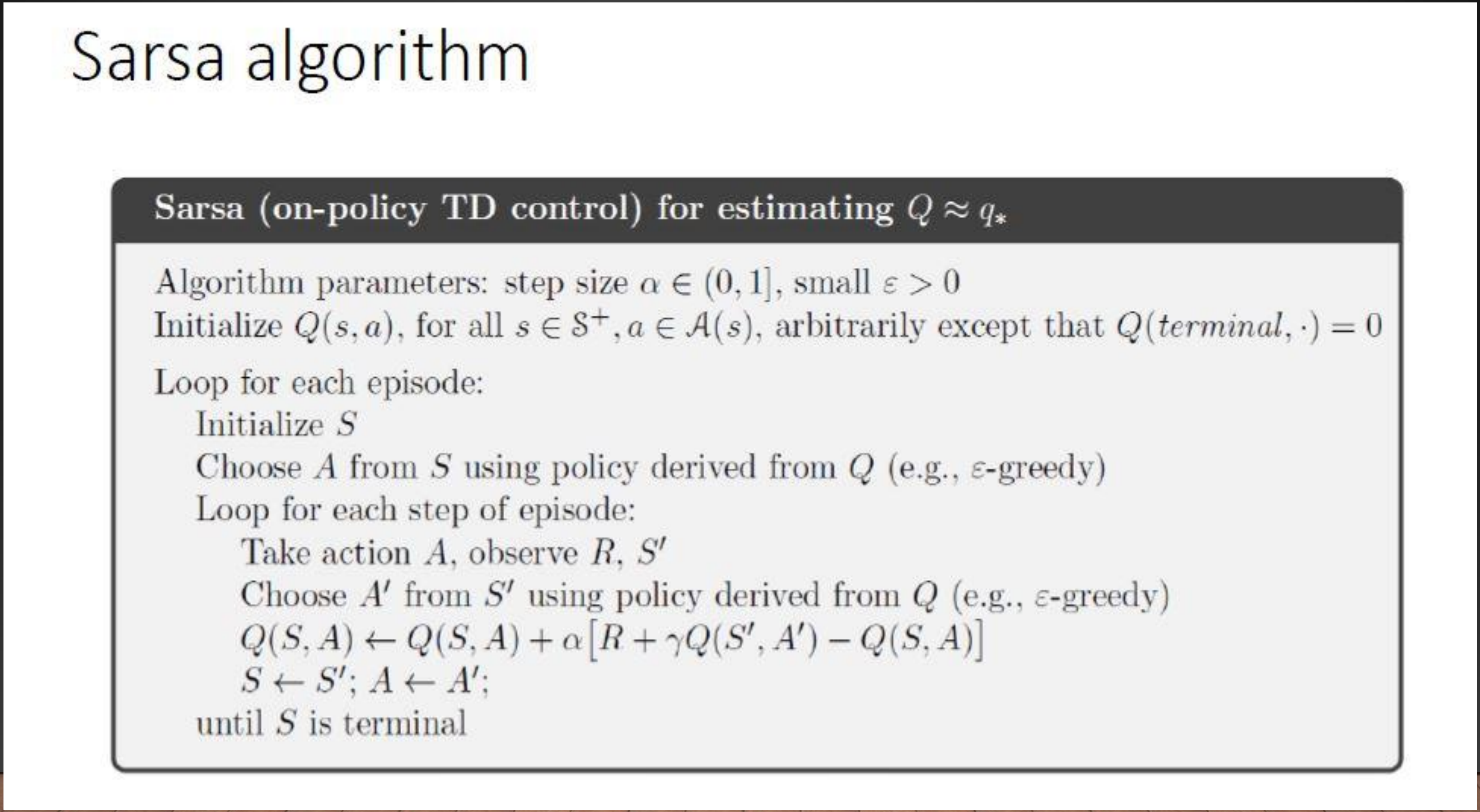
```
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
    Initialize $S$
    Loop for each step of episode:
        Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
        Take action $A$, observe $R$, $S'$
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$
    until $S$ is terminal
```

Q-LEARNING IN ACTION

- https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_td.html

SARSA



Sarsa algorithm

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

```
Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$
Initialize $Q(s,a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
    Initialize $S$
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
    Loop for each step of episode:
        Take action $A$, observe $R$, $S'$
        Choose $A'$ from $S'$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
        $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
        $S \leftarrow S'$; $A \leftarrow A'$;
    until $S$ is terminal
```

DO YOU THINK CHOOSING THE NEXT STATE FROM THE EXISTING POLICY IS ALWAYS BENEFICIAL?

WHAT COULD BE A DISADVANTAGE OF THIS?

EXPLORATION VS. EXPLOITATION

- Q and SARSA both chose the s value from the policy.
- If the initially policy or actions chosen are not very good this may make the algorithm converge a lot later or converge to a local minima

- Exploration can be introduced to help solve this issue.
- It forces the Agent to go to a random state instead of following the policy.

How Would the Implementation Work? What do you want to Change?

DELAYED REWARDS

- In some games such as chess we receive the rewards at the end (delayed-rewards).
  - Which moves were crucial for the win/loss?
  - Which move should come First and to which move should we give the reward and how much?
  - Extremely popular research are with may unexplored areas
- Self Driving cars

DRAWBACKS

- How to Store the Q table?
  - How much memory would be needed for large state spaces
  - Is searching the Q-table efficient?
- Learning process is expensive for the agent
  - Convergence Time is longer
  - Visiting all possible states takes a very long time for problems with larger state spaces

HOW CAN WE FIX THE Q-TABLE ISSUE?

CAN WE REPLACE THE Q- TABLE WITH SOMETHING?

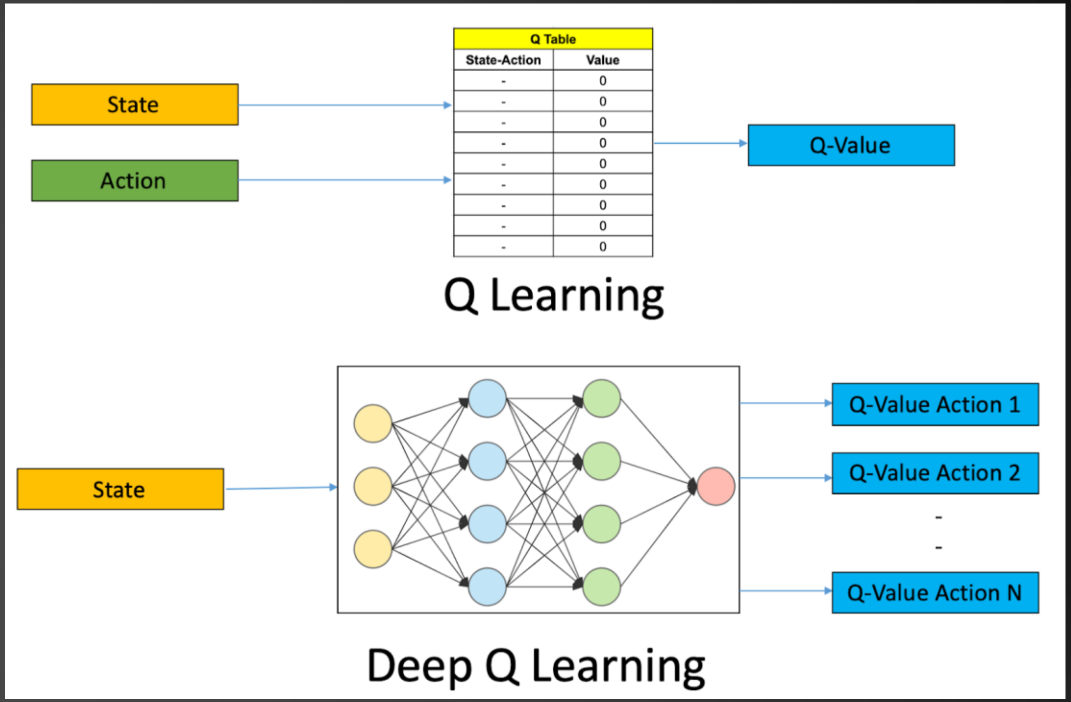
VALUE FUNCTION APPROXIMATORS

- We need a supervised learning technique that can capture nonlinear patterns
- Linear regression with an augmented basis expansion
- Decision-trees
- Nearest Neighbour
- Neural networks
  - E.g.: We learn the value of $Q^*(s,a)$ in the Bellman equation using a neural network

DEEP RL VS RL

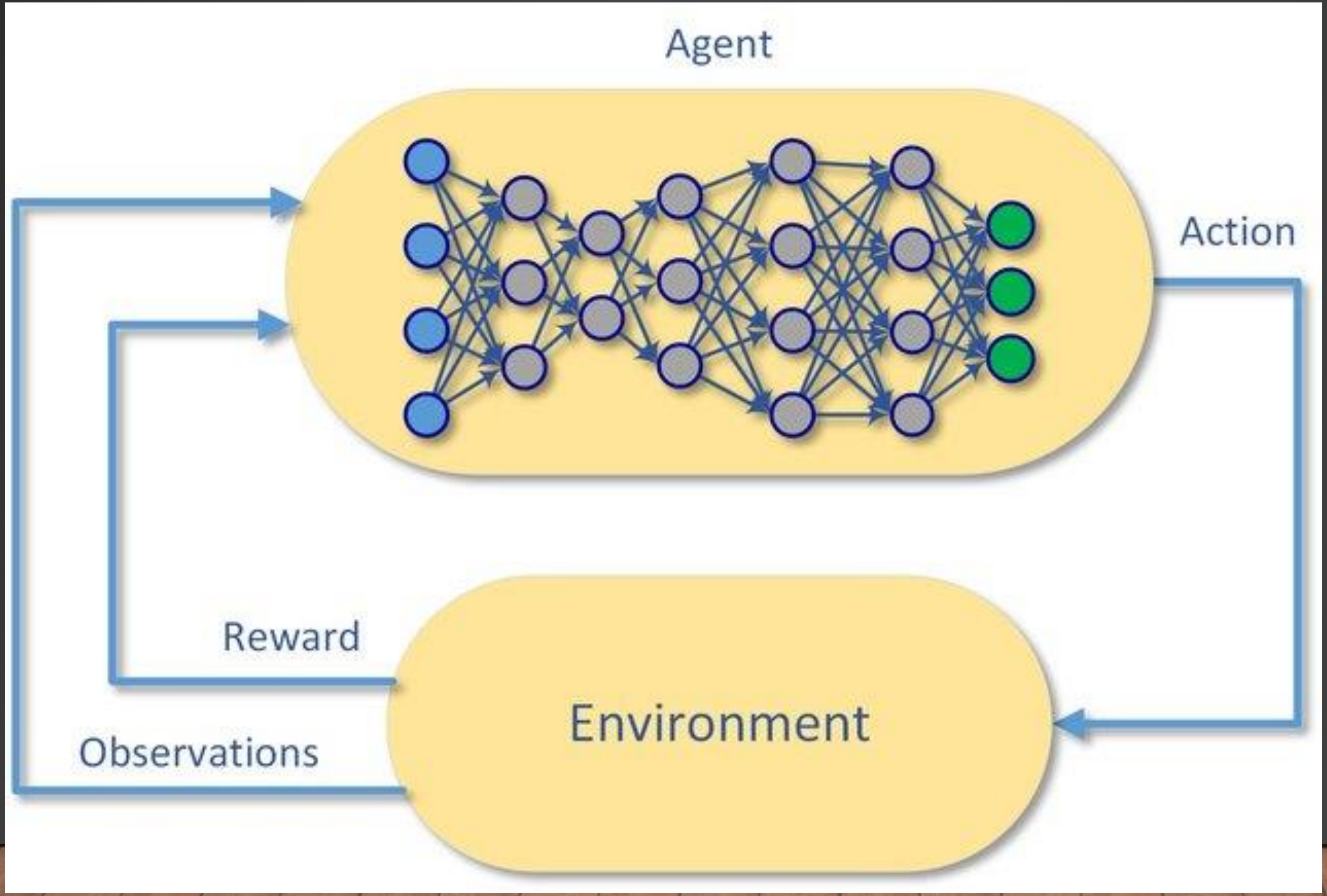
- In RL we use array to store the Q-values
- In Deep RL this is replaced with an Artificial Neural Network that acts as a function Approximator.
- Can give the Q value or the next Action as the output based on the Architecture of the ANN.
- This Allows us to use complicated structures as the State.
  - Images
  - Graphs
  - sequences

WHAT CAN WE USE AS THE ANN



https://medium.com/@novacek_48158/connect-x-with-dqn-and-pbt-be11915dd860

- Convolutional Networks – (DQN)
  - Games
- Pointer Networks
  - Combinatorial Optimization
- GPN
  - Graph based Combinatorial Optimization
- Any ANN as long as it serves as the function Approximator for Q*(s,a)


Agent
Action
Reward
Environment
Observations

DEEP Q NETWORK (DQN)

- https://towardsdatascience.com/deep-q-network-with-pytorch-146bfa939dfe

POLICY GRADIENT METHODS

- In Deep Q-learning, a neural network is used to approximate $Q^*(s, a)$.
- In policy gradient methods, neural networks are instead used to represent a policy $\pi\_\theta(a \mid s)$.
- Rather than encouraging the policy to achieve positive rewards, we want to encourage it to do better than expected.
- Recent actor-critic methods for continuous action spaces
  - Trust Region Policy Optimization (TRPO)
  - Proximal Policy Optimization (PPO) - outperforms TRPO

ACTOR CRITIC METHODS

- Actor
- Critic

EXAMPLE VIDEO LINKS

- https://www.youtube.com/watch?v=Lu56xVlZ40M
- https://www.youtube.com/watch?v=t33jvL7ftd4
- https://www.youtube.com/watch?v=gn4nRCC9TwQ

REFERENCE

- MIT Lecture : https://www.youtube.com/watch?v=zR11FLZ-O9M
- Stanford Lecture : https://www.youtube.com/watch?v=lvoHnicueoE