An Introduction to Reinforcement Learning

Presenters: Mahyar Ghasedian & Sina Esmaeili

Table of Contents

Preliminary

➢ Definition (Reinforcement Learning)

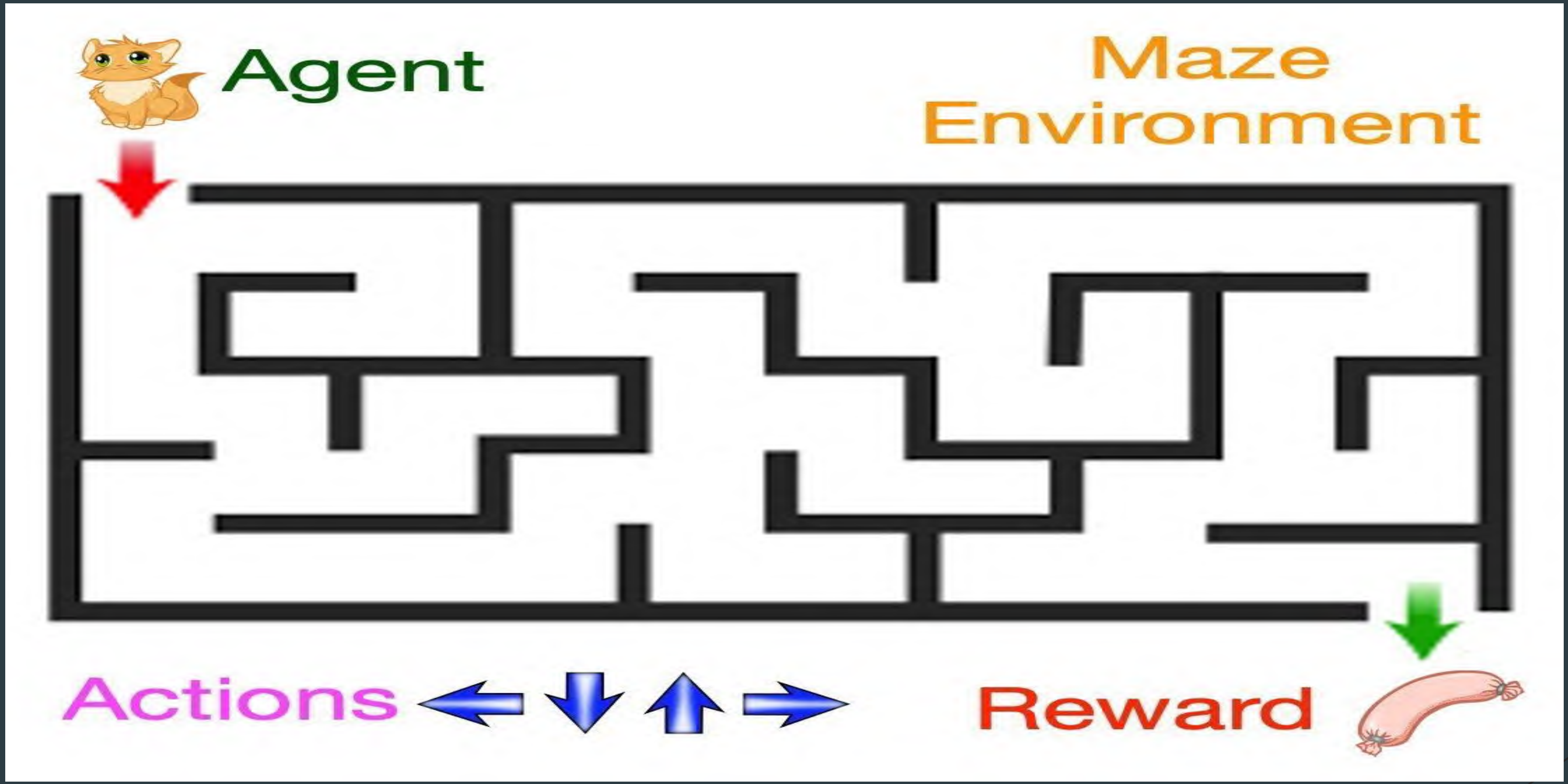
Training: Reinforcement Learning is the science of decision making. It is about learning the optimal behavior in an environment to obtain maximum reward and minimum punishment.

Task: Find a model to maximize the total cumulative reward of the agent.

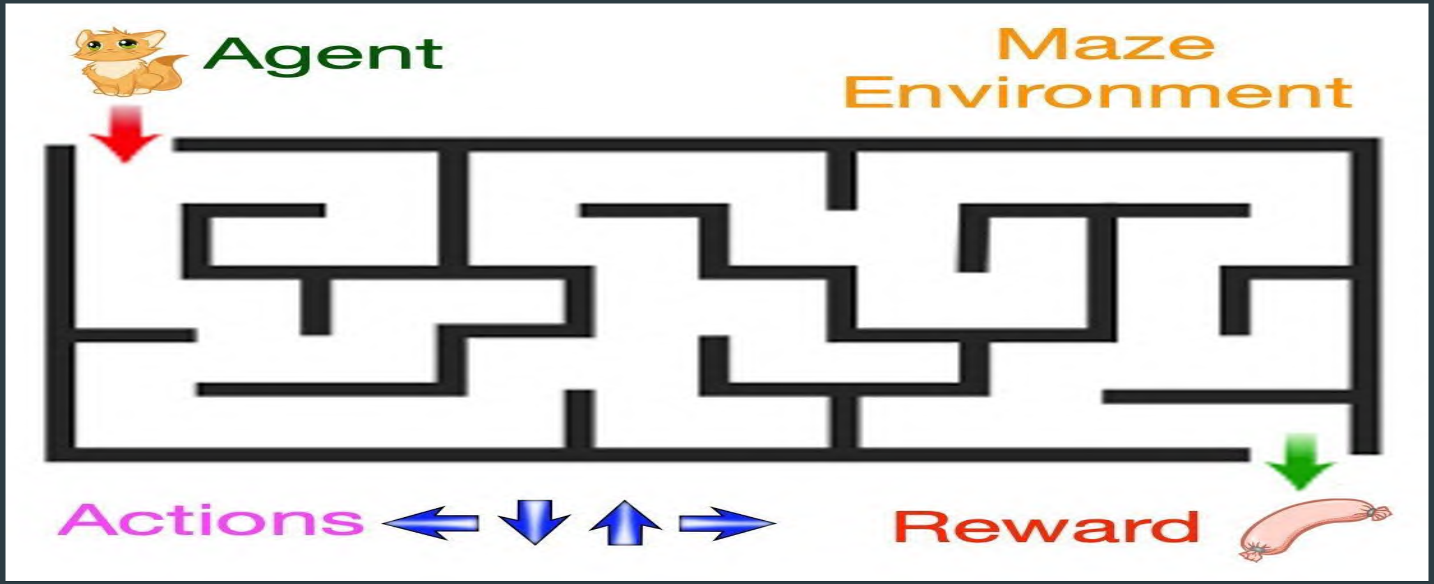
Examples: Teach an artificial intelligence (AI) system to play GO

➢ Training

How is it possible to train a cat to find exit easily?

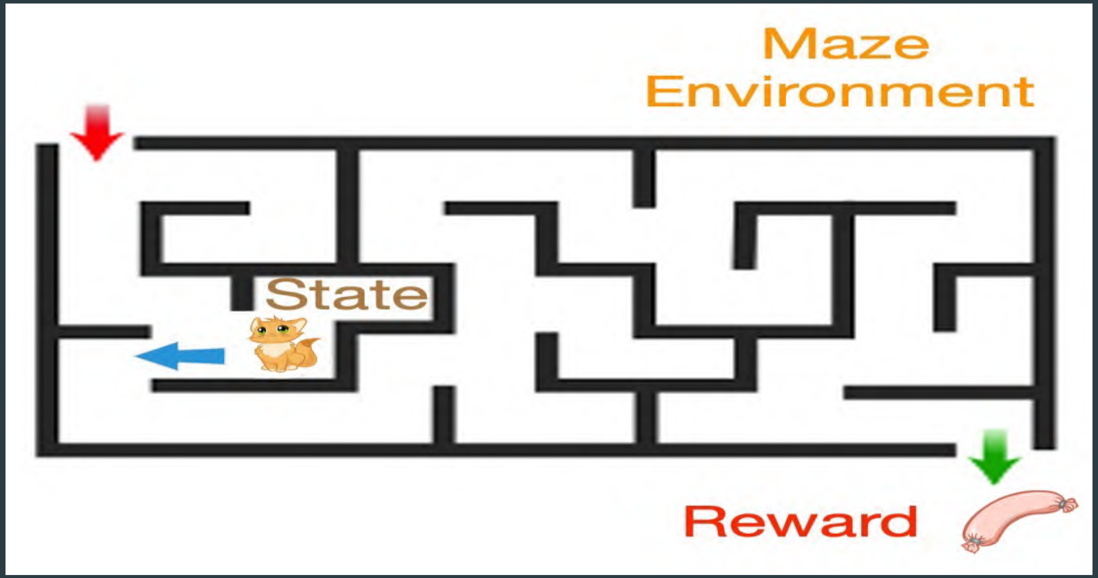


Agent: The learner or decision-maker.
Environment: The Agent’s world in which it lives and interacts.
Goal: The goal of reinforcement learning is to train an agent to complete a task within environment.



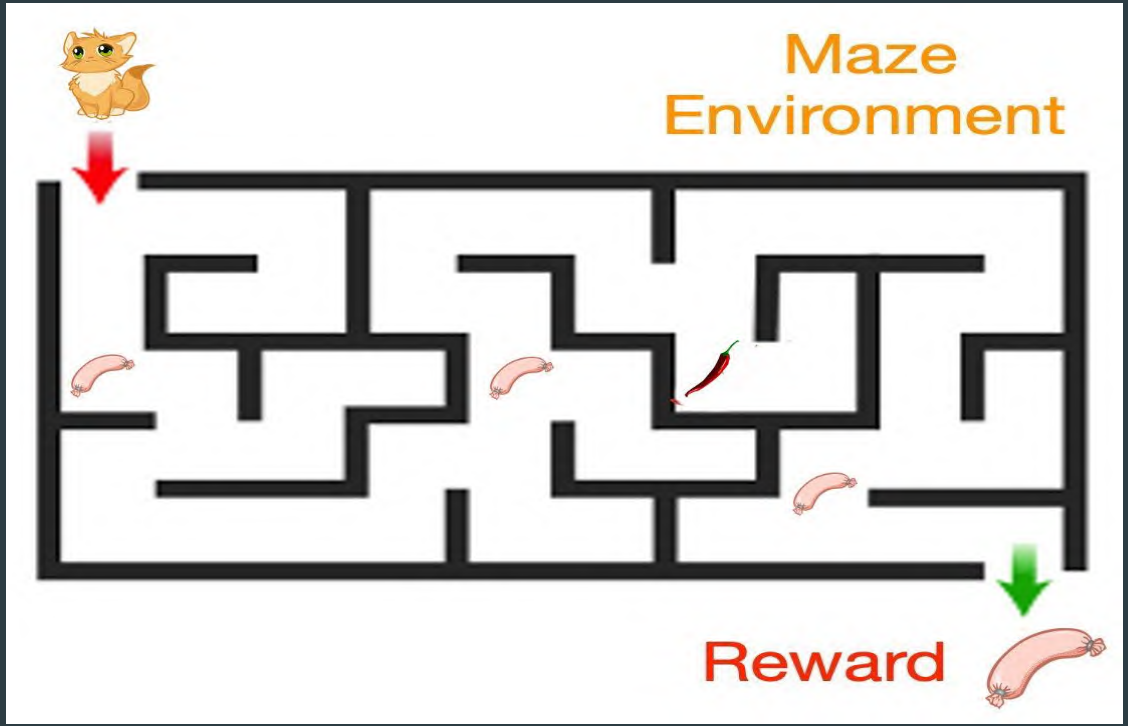
Action: The set of all possible moves an agent can make. The agent can interact with the environment by performing some action but cannot influence the rules or dynamics of it by those actions.

State: An immediate situation in which the agent finds itself. It can be a specific moment or position in the environment.

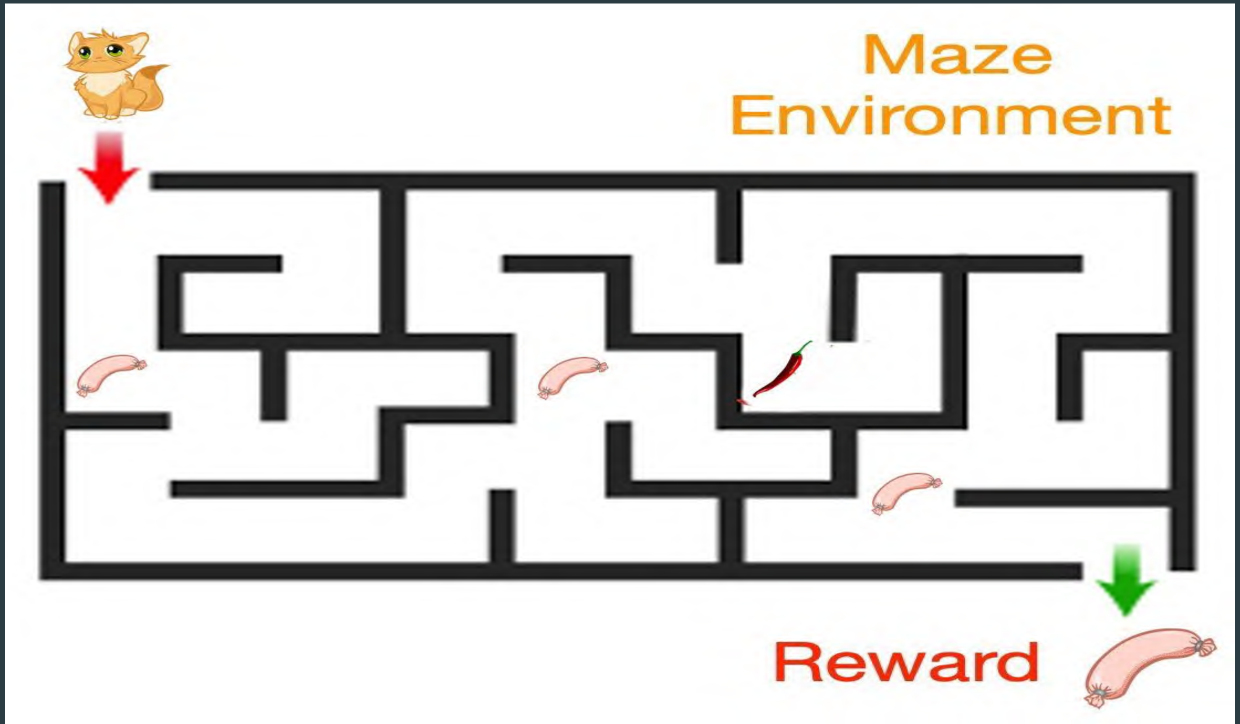


Reward: For every action made, the agent receives a reward from the environment and indicates performance of the agent.

Reward Hypothesis: All goals can be described by the optimization of some function of the sequence of rewards.



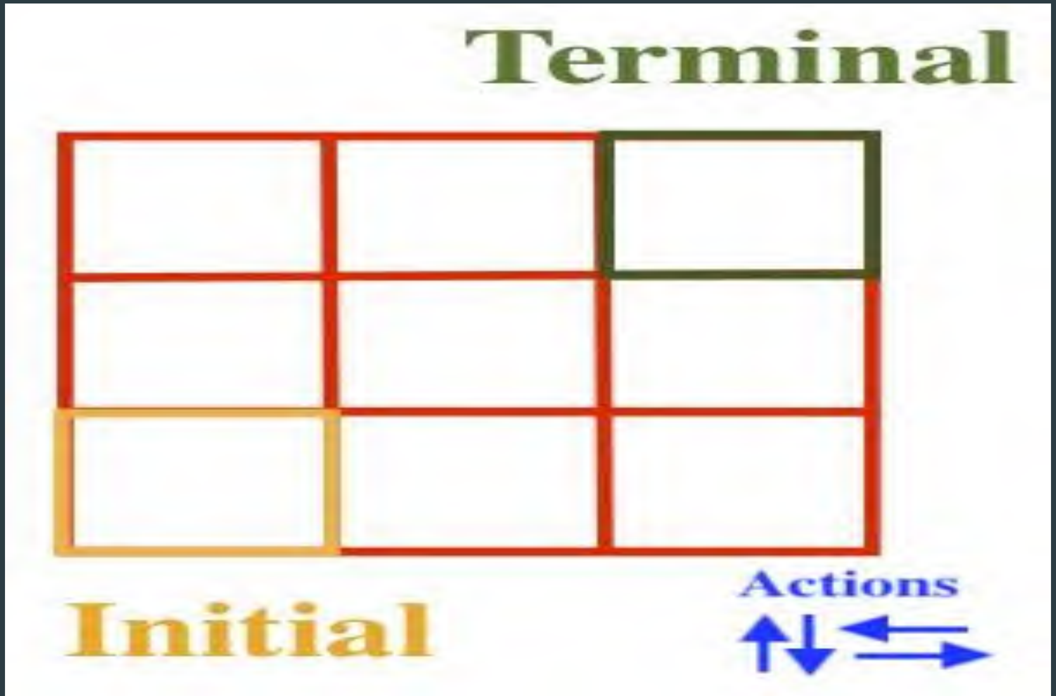
Agent is not so clever to learn alone. Agent's goal is to optimize some function of the sequence of rewards to reach the goal.



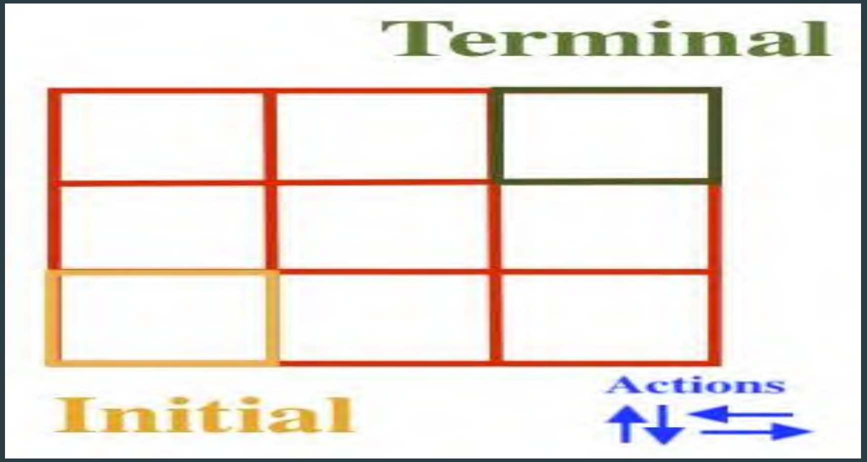
Training an Agent: Assigning Rewards

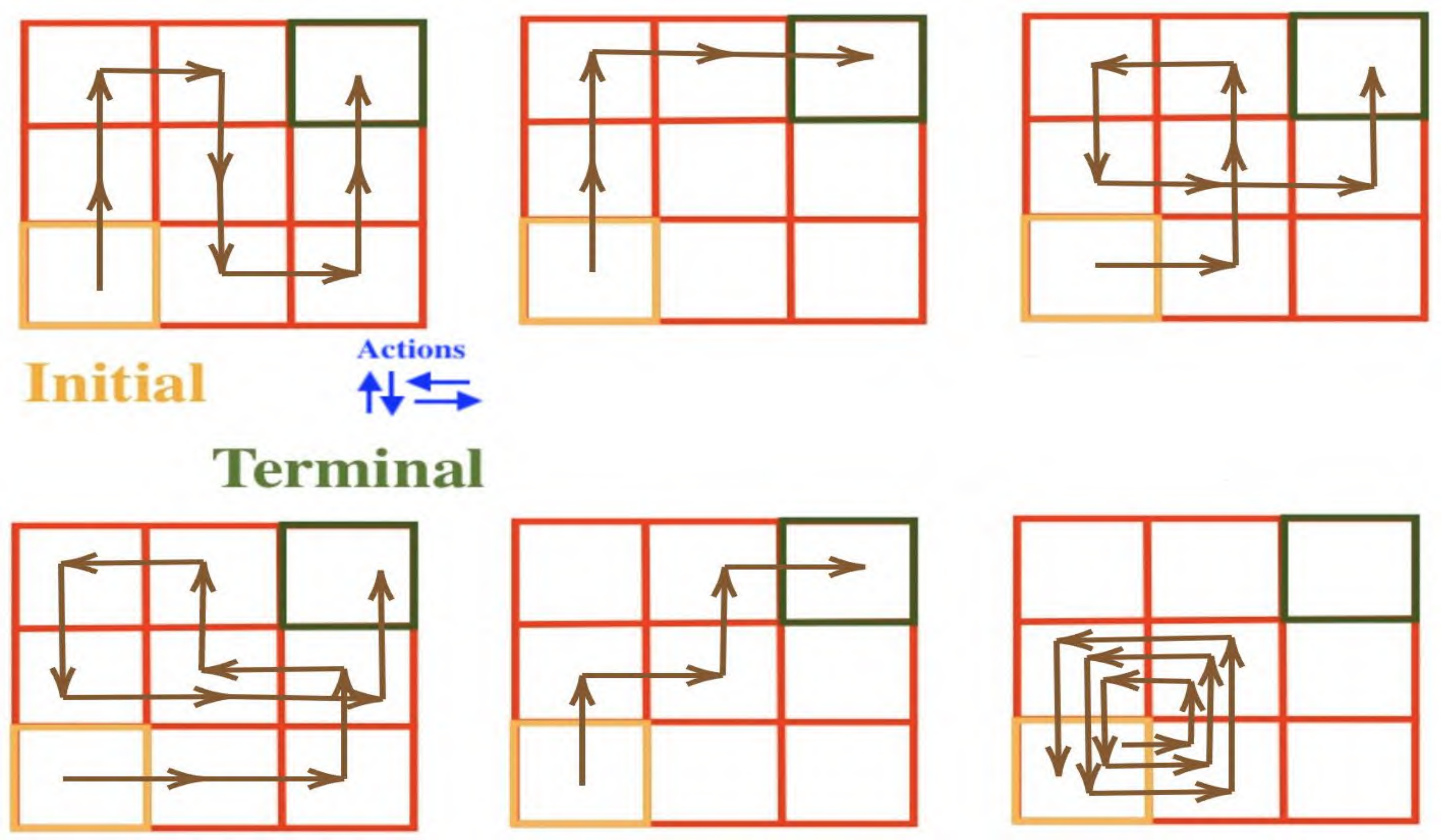
Shortest Path: Train an agent which is in the initial state to reach to the terminal state in the shortest possible time.

How is it possible to train the agent to reach the goal?

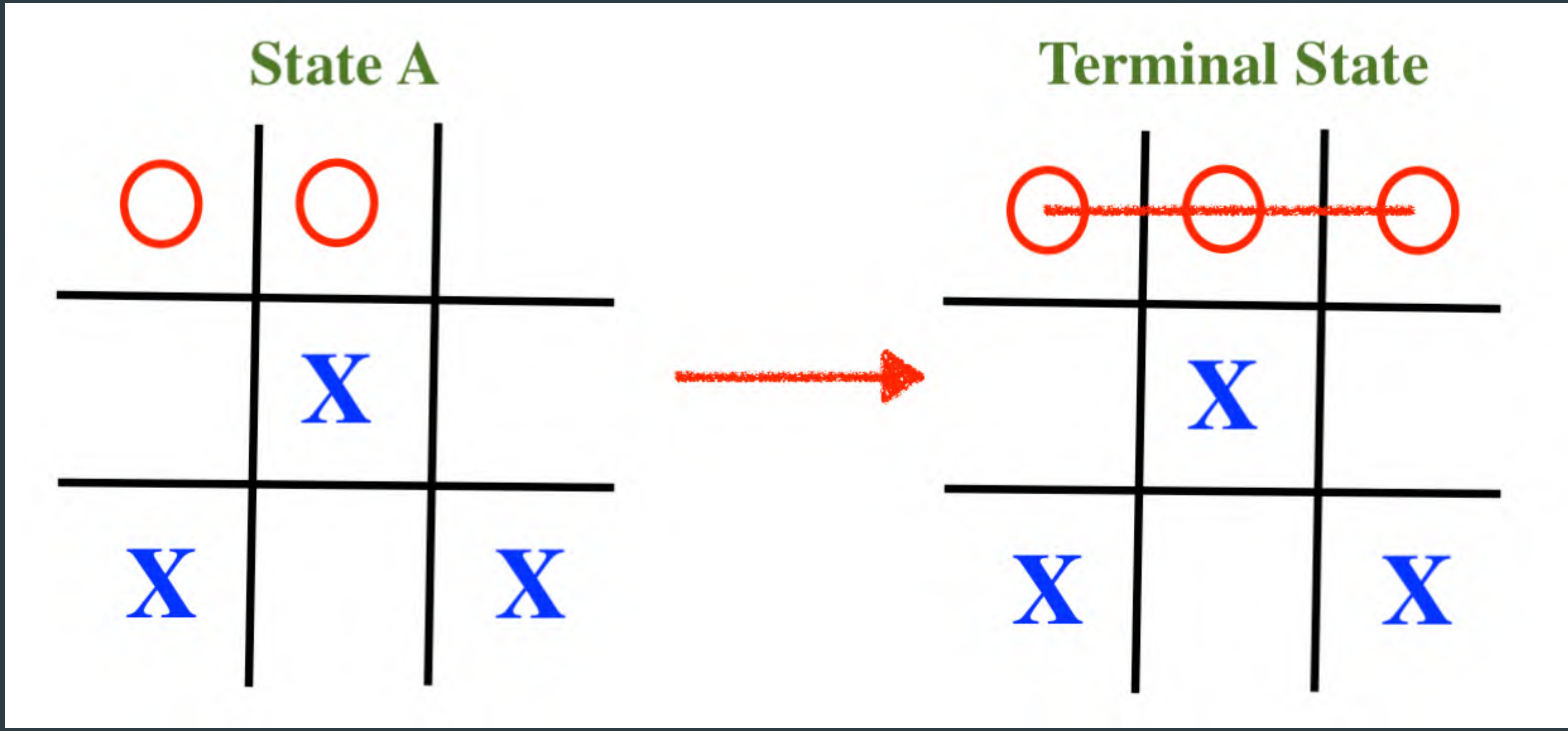


Terminal State is an absorbing state. All actions taken in the absorbing state lead back to that same state.
Agent learns by doing actions and interacting with environment (collecting data: trial and error, former experiences, . . .)
By assigning appropriate rewards, an agent must be able to learn from its own experiences.

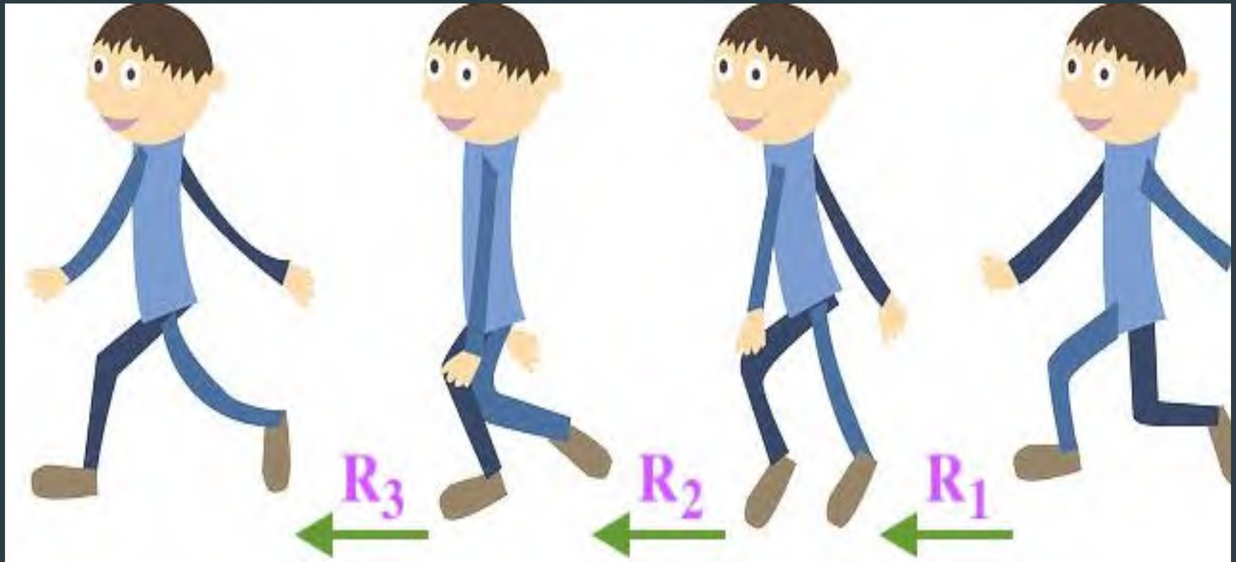


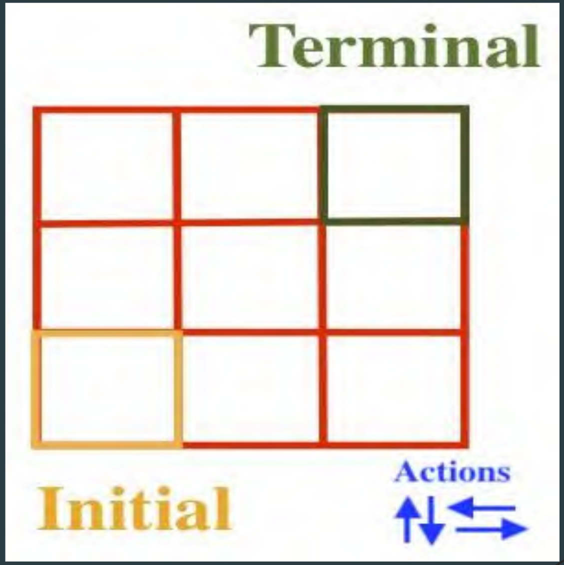

Actions
Initial
Terminal

For instance in Tic-Tac-Toe, collecting data does not mean to have all of wining strategies!!! Also, learning does not mean to copy a wining strategy!



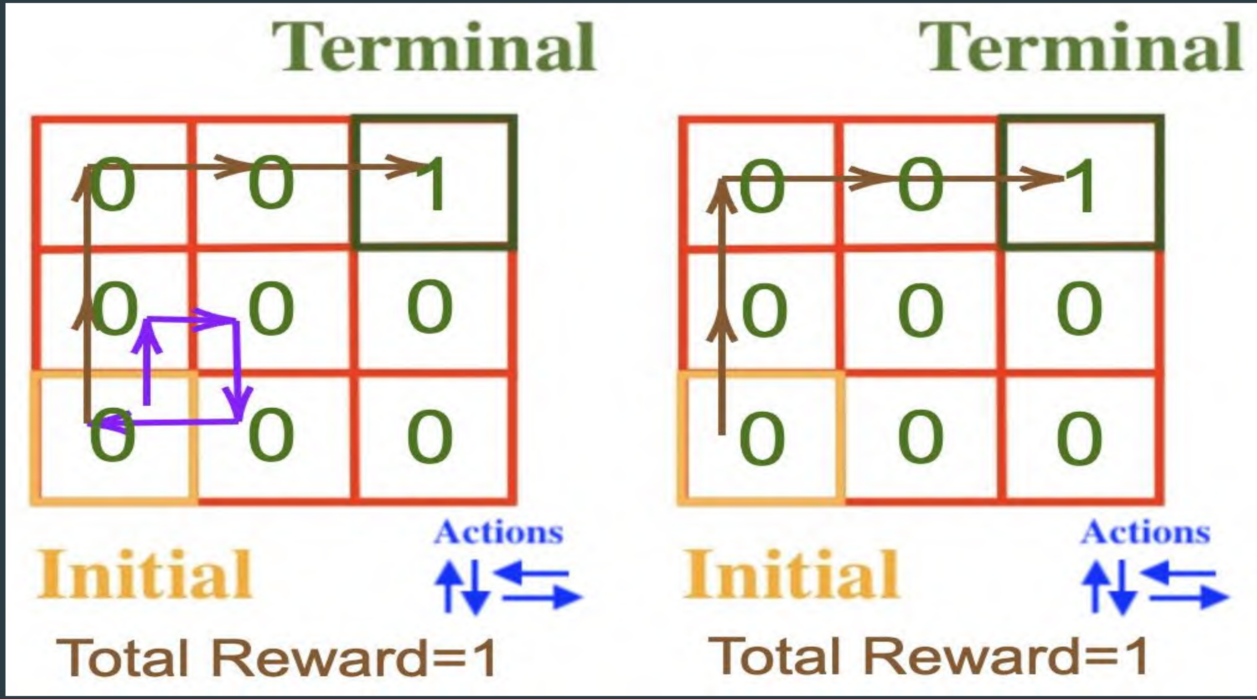
We usually consider the cumulative sum of rewards as a function to reach the goal (Cumulative Sum: $R_1 + R_2 + \ldots + R_n$). In this case, the existence of the terminal state is necessary.

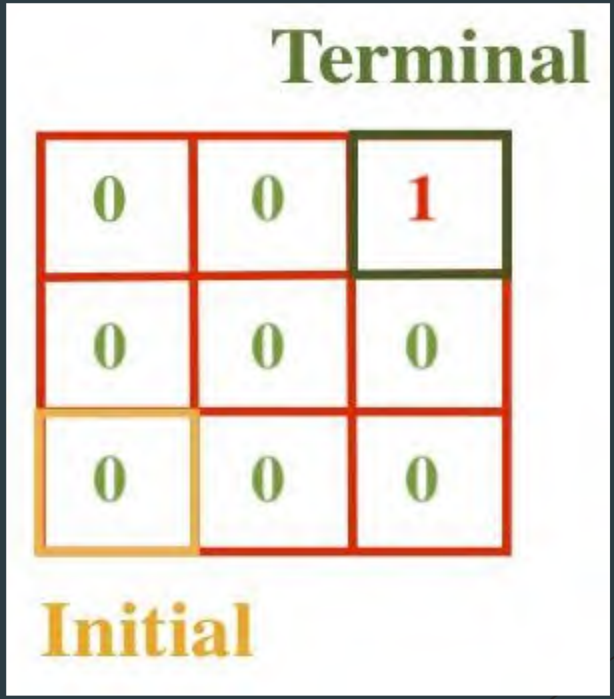




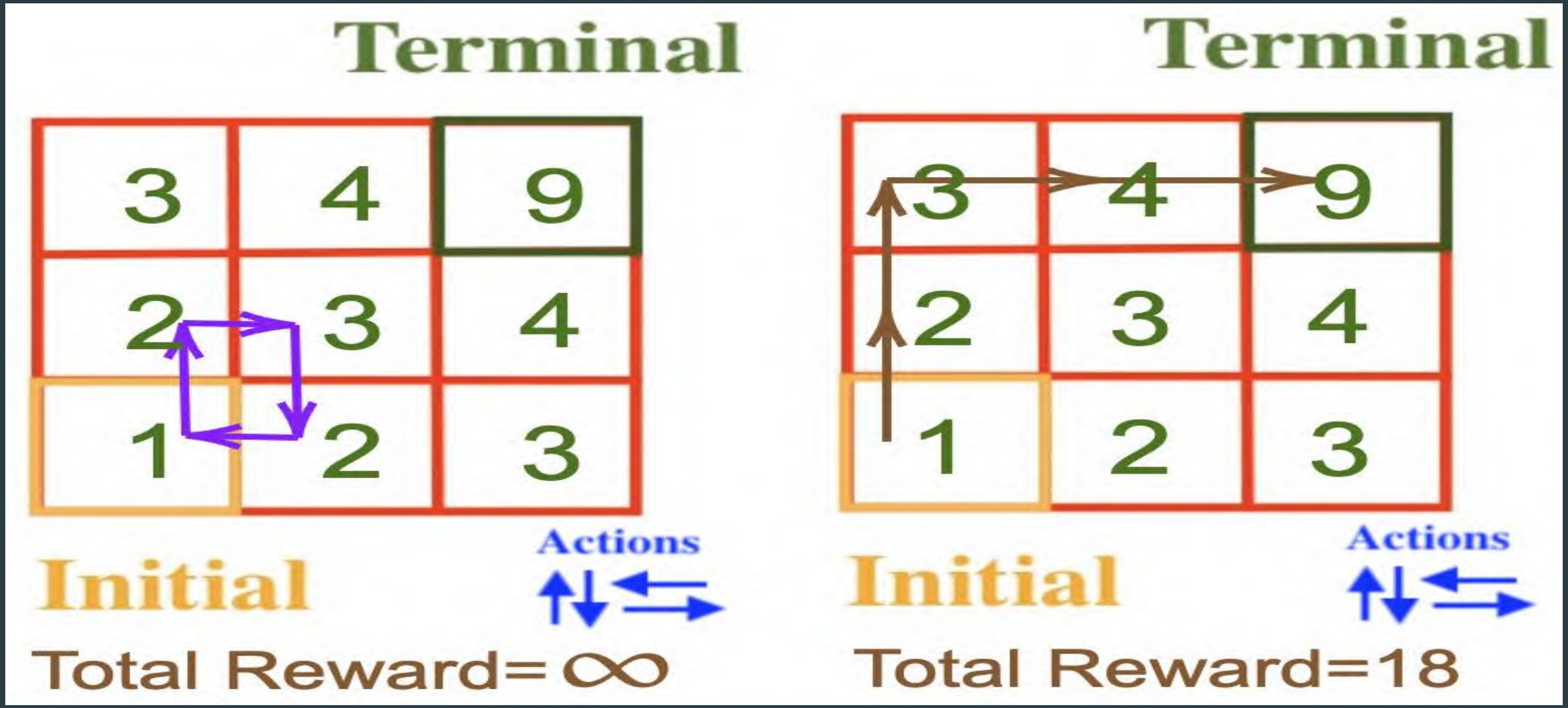
Cumulative Sum: $R_1 + R_2 + \ \dots + R_n$

All reward on JUST terminal state (goal)!



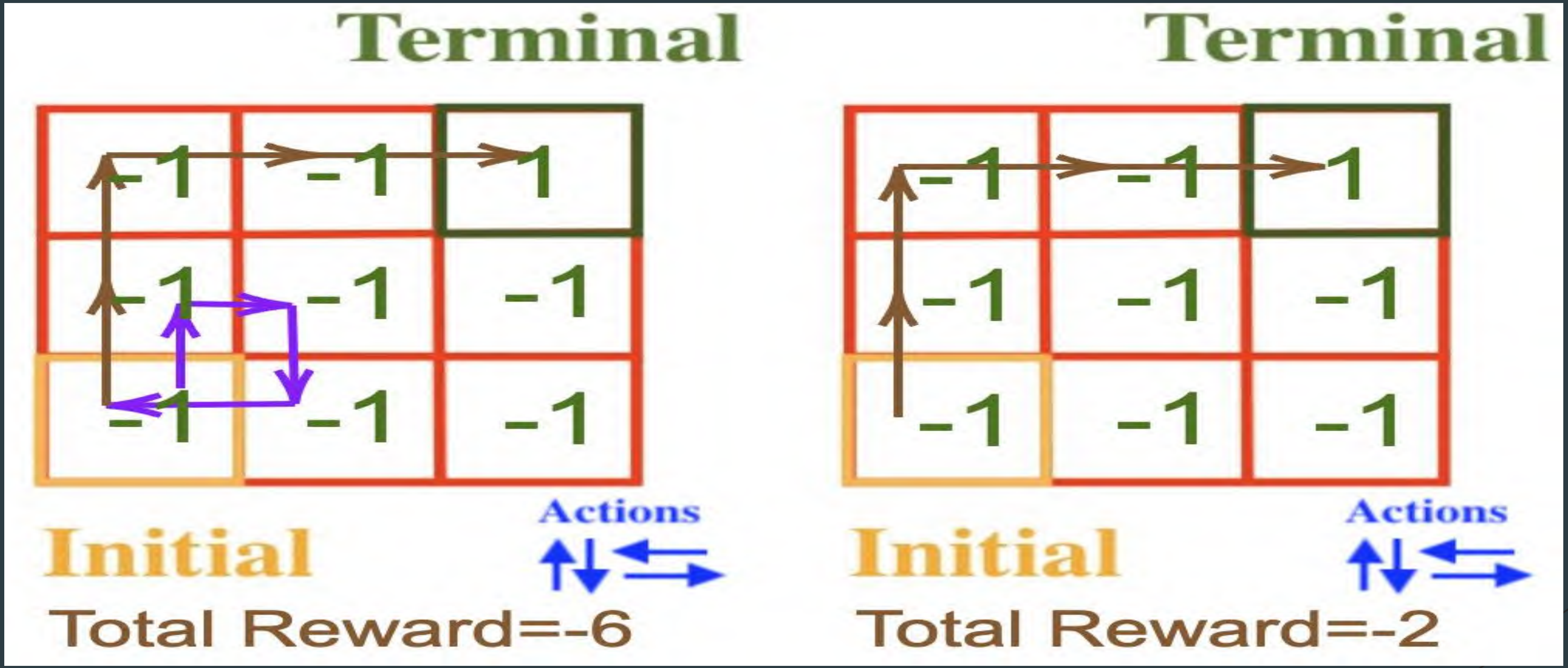


Increasing reward as you approach terminal states.
If there exists a cycle, agent can stay in the cycle forever or if the number of steps is large enough, then the summation will be a very large number!

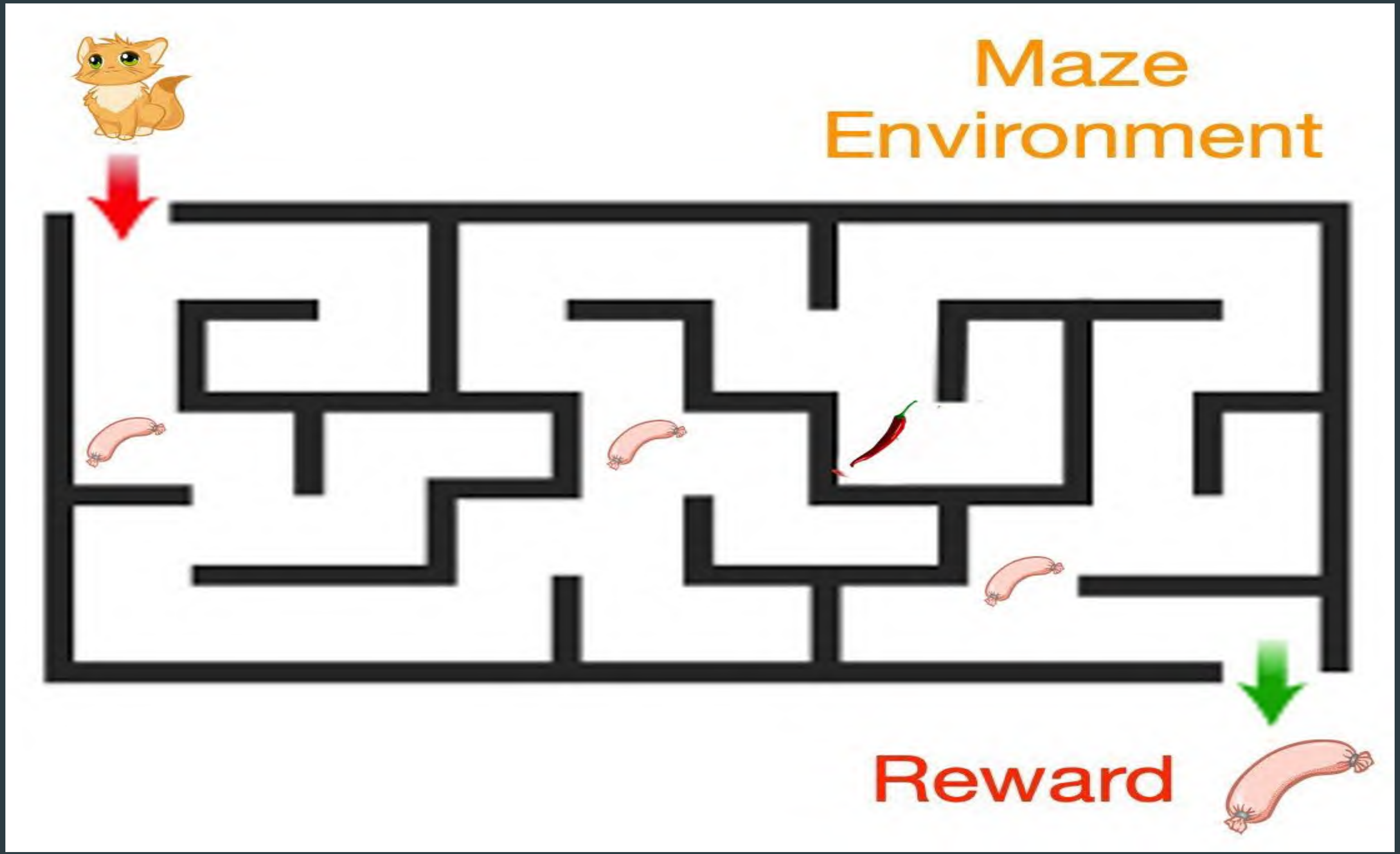


What is the solution?

Negative reward everywhere (except actions of terminal state)



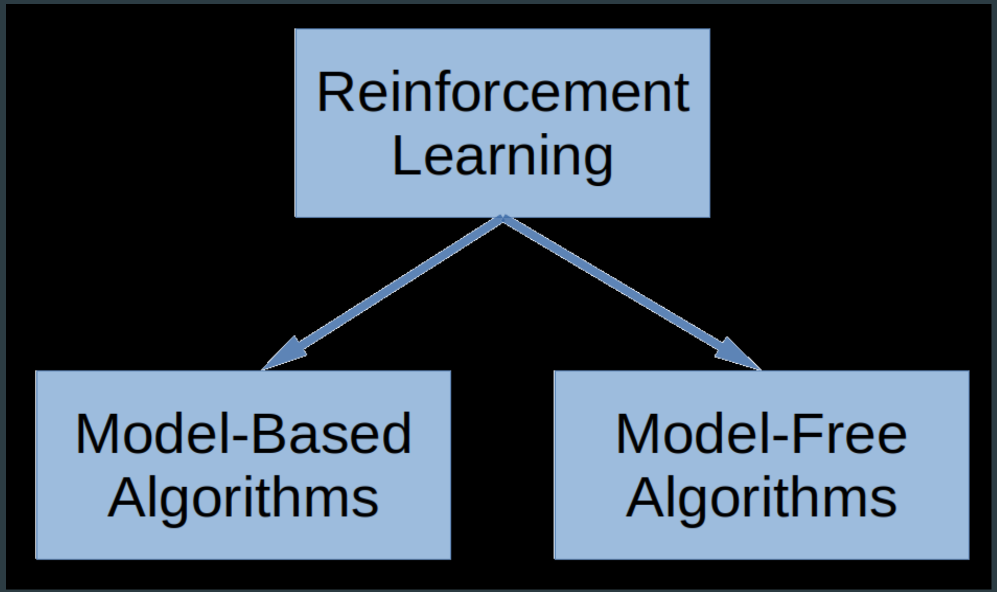
We assign the rewards such that it is possible to an agent. In training an AI, we should consider an train appropriate algorithm.



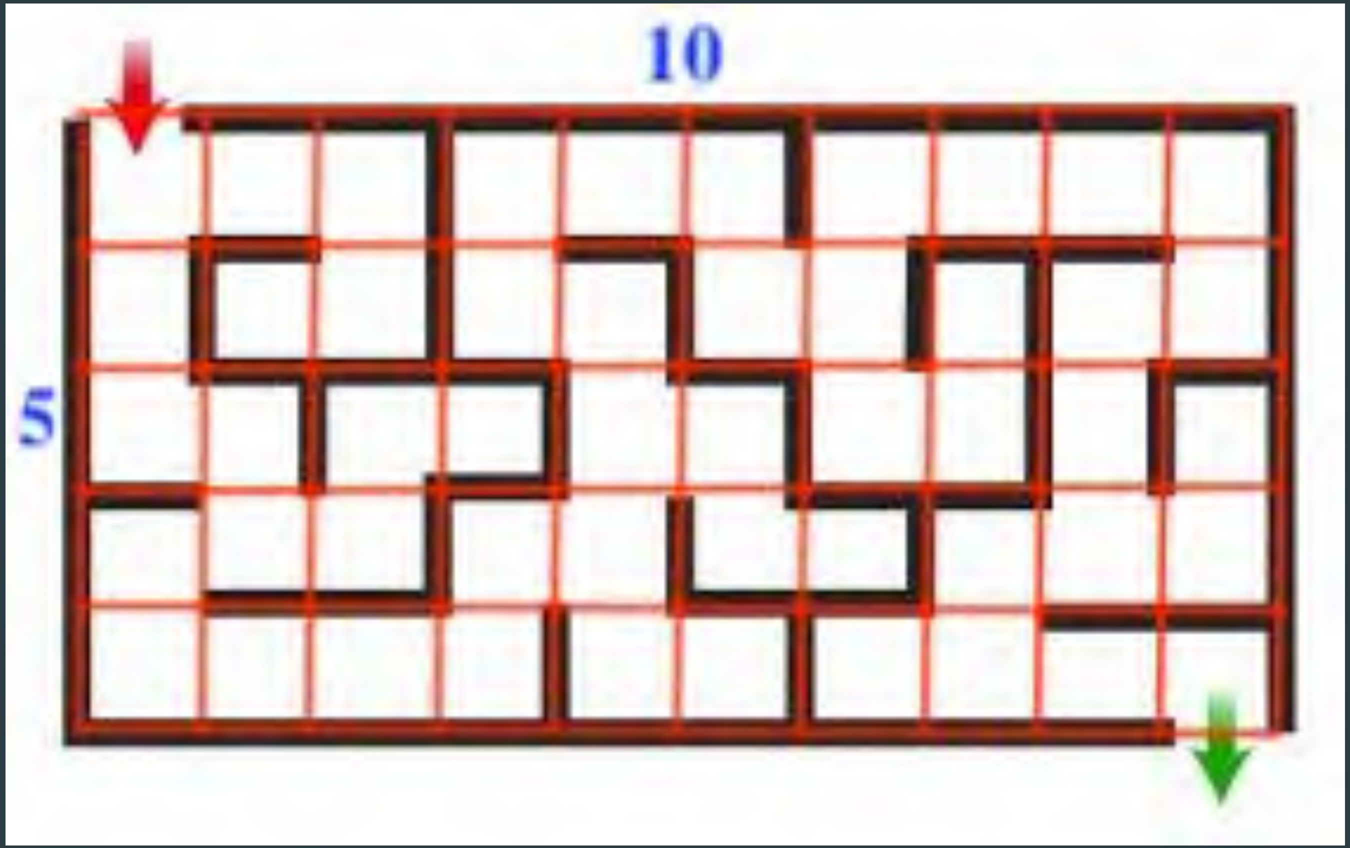
Computational Strategies: Challenges and Limitations

Model-based algorithms use a predictive model of the environment to ask questions of the form "what will happen if I do action a?" to choose the best action a.

Model-free algorithms seek to learn the consequences of their actions through experience.



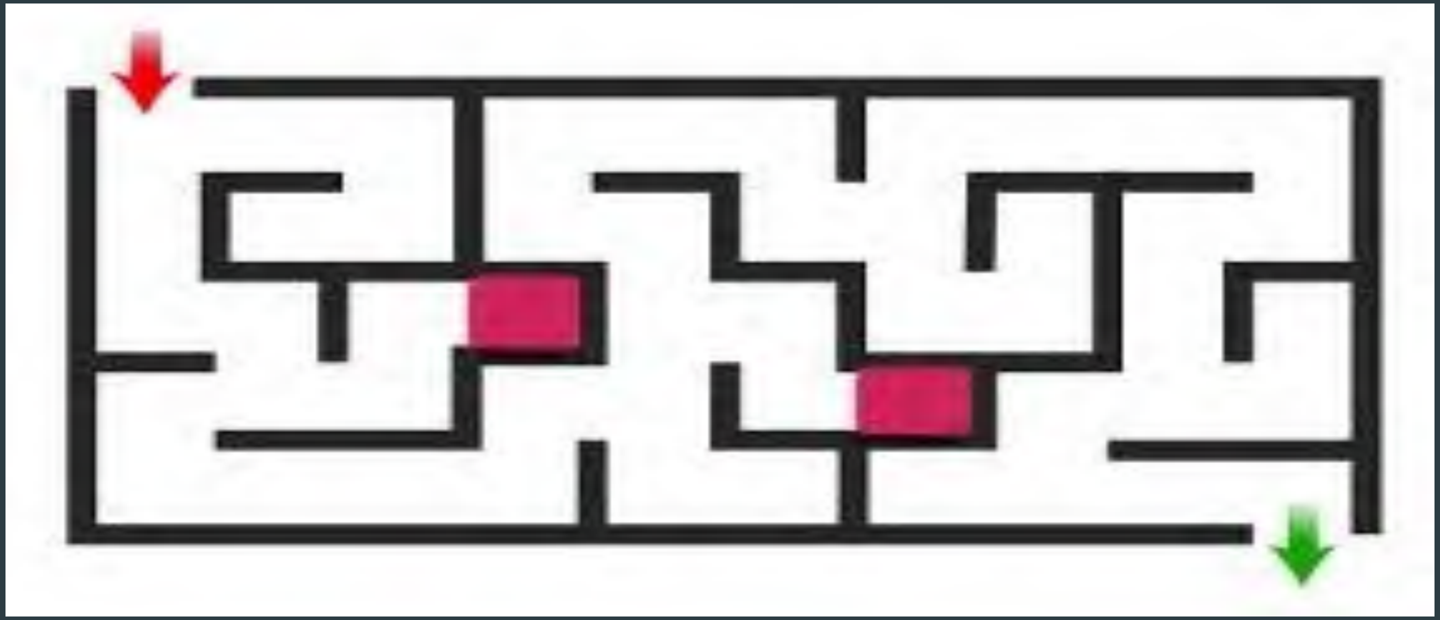
Question: How many states has this environment?
Challenge: The environment is usually large.



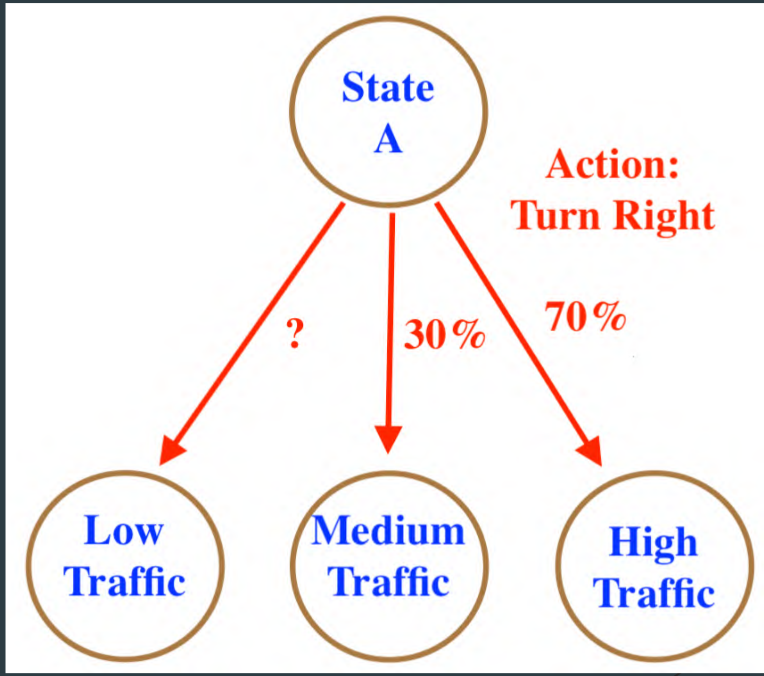
At the beginning of exploring of the environment, if the agent do an action, then the agent moves to an unknown state.

For instance, when we are playing a game and we should choose a door to continue the game and we are not aware about next state.

Is it possible for the agent to distinguish between the red states? If we assume that all of walls are the same and high enough respect to the agent (the agent does not see the objects behind the wall).

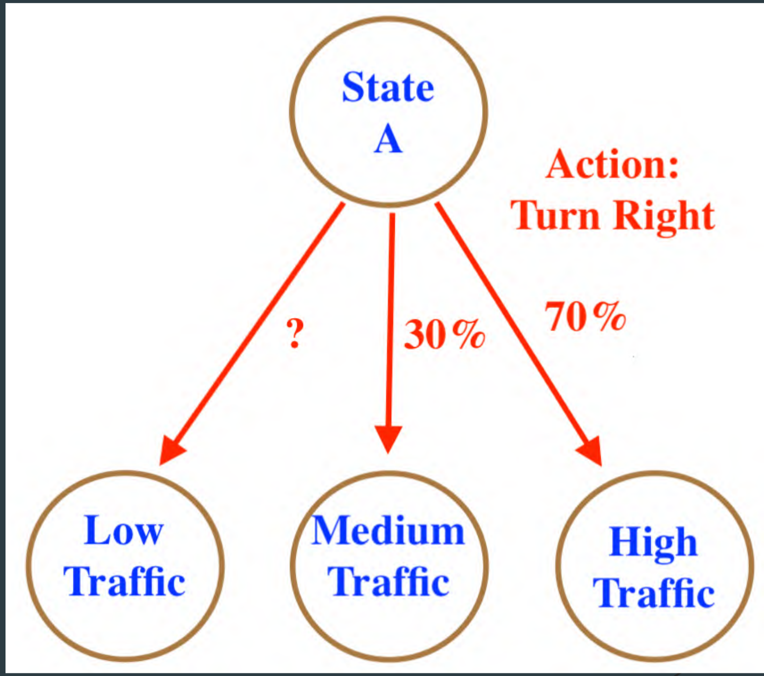


Actions may be stochastic and one may be unlucky to explore all of transitions. For instance, in the following Environment and in the state A, the action Turn Right is stochastic and the state Low Traffic has not been explored as its neighbour so far!



Limitation: The agent usually has no chance to run several actions sequentially in a specific state to explore the environment.

Limitation: Agent usually cannot restart to explore the environment form an arbitrary state.



Challenge: Partial Observability

In this talk, we assume full observability: the new state resulting from executing an action will be known to the system.

An example for partial observability: in some card games ,the agent cannot see the hands of the opponent for the most part of the game.

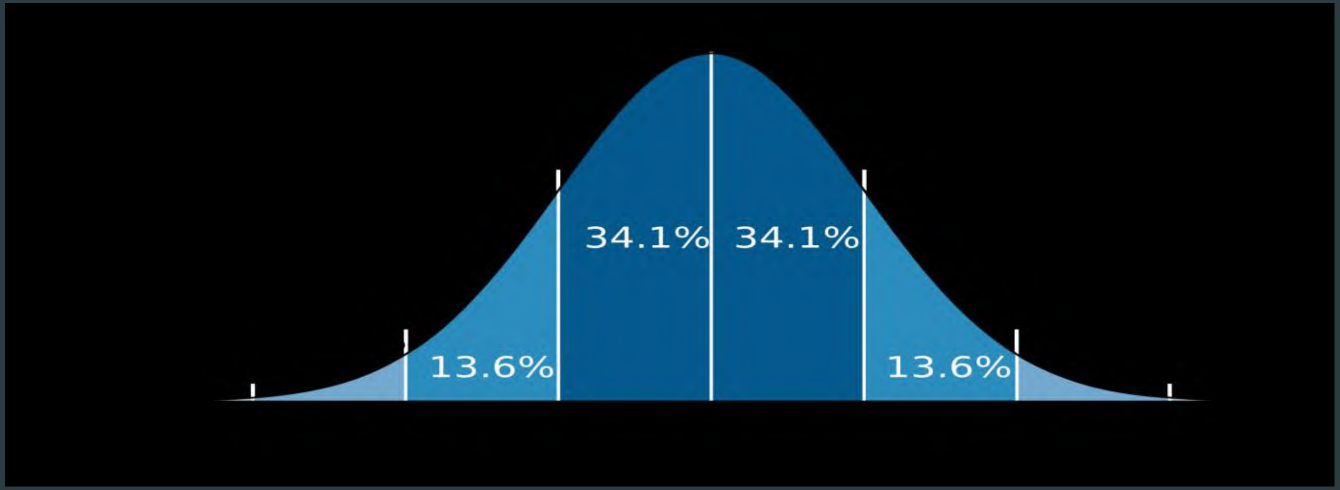
The Smallest Environment: Multi-Armed Bandits

Question: Is there any challenge for small environments?

"Bandit" is someone who robs people, especially one of a group of people who attack travellers.

"Bandit" in "Multi-Armed Bandits" comes from "one-armed bandit" machines used in a casino.

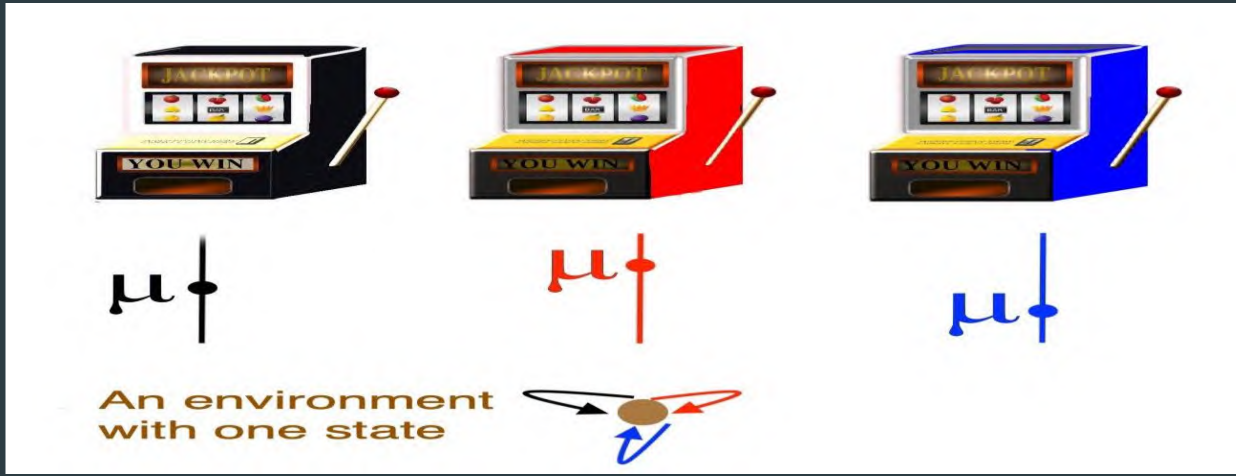
Question: Does a teacher always give the same reward when a student solve a kind of problem?
The rewards and punishments are often non-deterministic, and there are invariably stochastic elements governing the underlying situation.
Question: If the reward has normal distribution with unknown mean, how do we find a approximation of the mean of distribution?
The law of large numbers states that as a sample size grows, its mean gets closer to the average of the whole population.



Given a bandits with K arms where each arm has a fixed but unknown probability distributions with mean μ. Pulling any arm, gives you a stochastic reward. In this talk, we usually consider Gaussian Bandits.

Gaussian Bandits: The rewards come from a normal distribution.

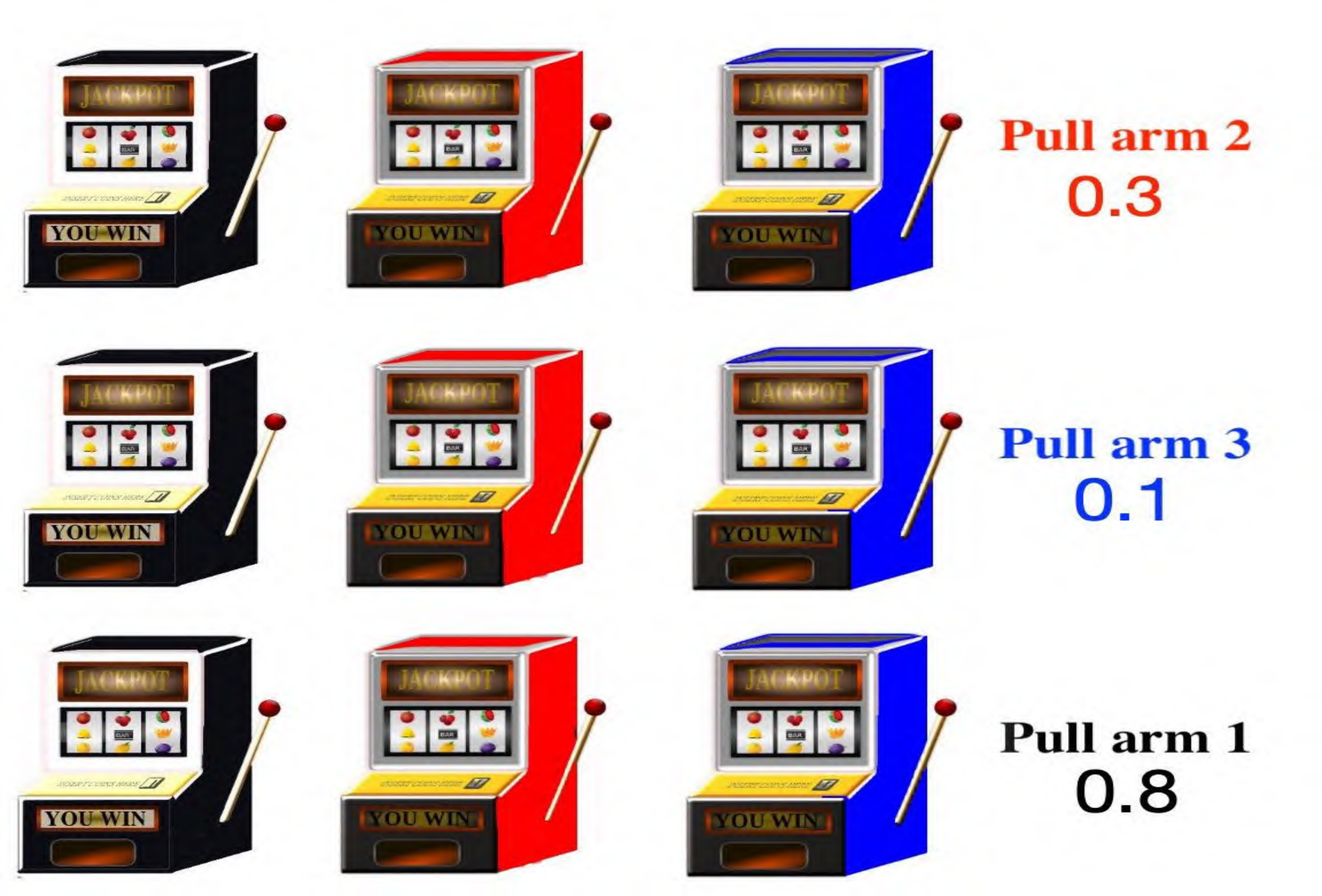
Bernoulli Bandits: At each round, we receive a binary reward (0 – 1).



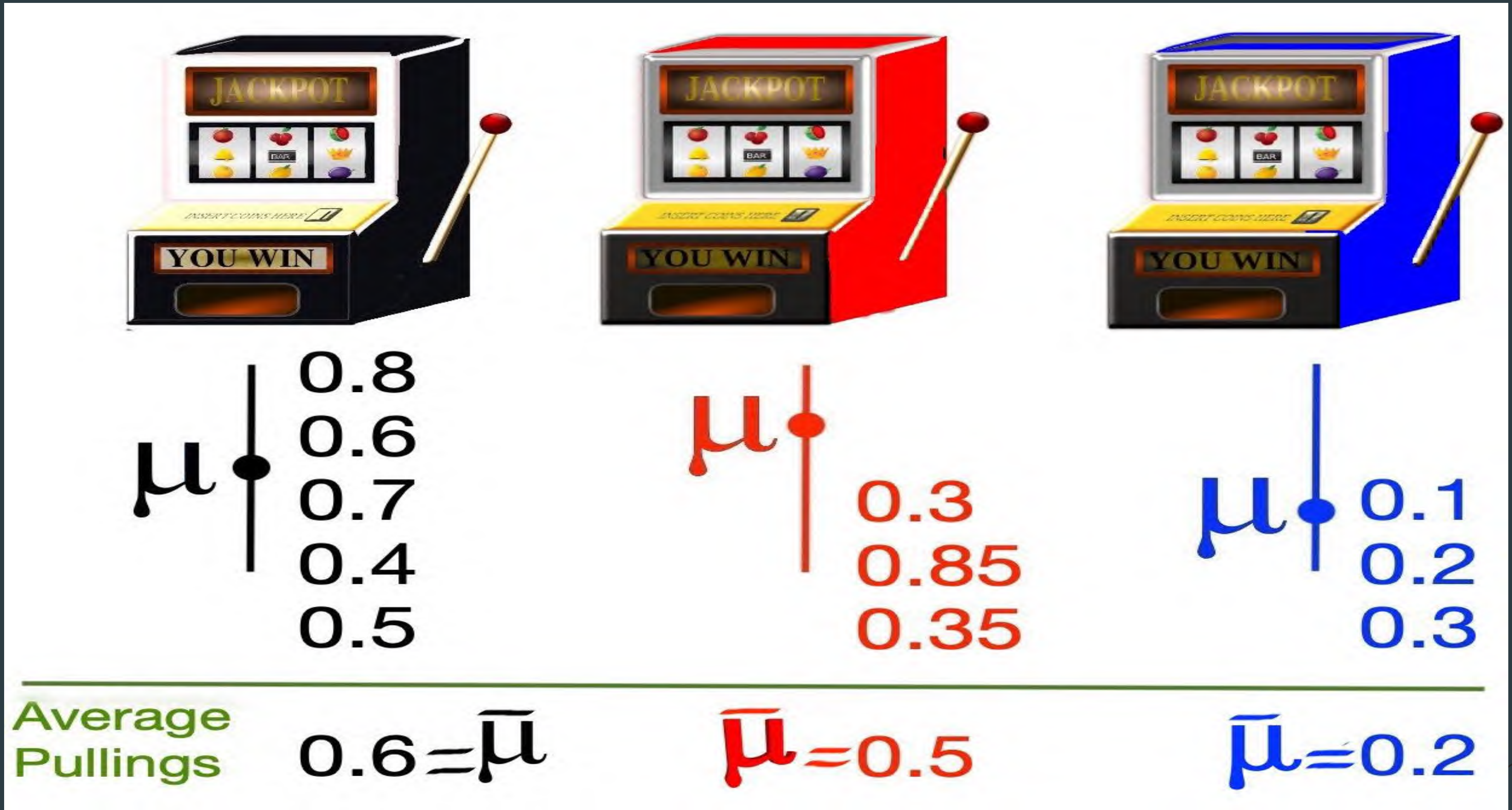
Our Goal can be one of the following items:

- Pull the arms one-by-one in sequence such that we maximize our total reward collected in the long run.
- Best arm identification: minimize the error probability at time T.
- Best arm identification: minimize the total number of stages used to return the best arm with probability $1 - \delta$.

Training to Reach the Goal: One can use the rewards of bandits to train the agent.


Pull arm 2
0.3
Pull arm 3
0.1
Pull arm 1
0.8

- The multi-armed bandit problem:



Various Algorithms for Multi-Armed Bandits

Challenge: The Exploration-Exploitation Dilemma

Exploitation: make the best decision given current information
Exploration: gather more information

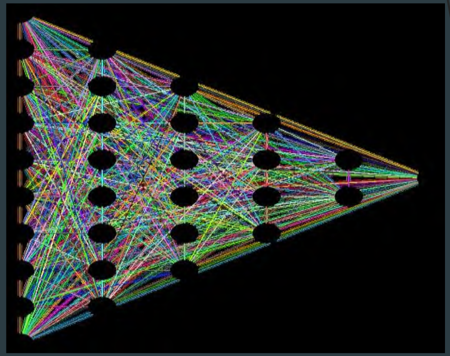
The best long-term strategy may involve short- term sacrifices
Gather enough information to make the best overall decision

Exploit: Eat your favorite food

Explore: Try a new food

Hyperparameter Tuning in Machine Learning:

Exploit: Use the best known Hyperparameters
Explore: Check new Hyperparameters



Skill Improvement:

Exploit: Use current skills
Explore: Learn new skills

Online Advertising:

Exploit: Show the most successful advert
Explore: Show a new advert

The agent needs an algorithm to reach his/her goal!

Goal: maximize the cumulative sum of rewards

Greedy Algorithm: Based on Exploitation

1. Pull all arms once! (Exploration)
2. Choose the best arm!
3. Pull the best arm again (Exploitation)
4. Recompute the average reward of arms
5. Go to step 2 and continue

What is the best algorithm?

There are K-arms.

$\mu^* = \max_{1 \le i \le K} \mu_i$ denotes the optimal mean of arms.

In each time step t, the agent has to play an arm $I_t \in \{1, 2, \ldots, K\}$.

Let $x_{I_t}(t)$ be the obtained reward at time t where $1 \le t \le T$.

The aims is to maximize the cumulative sum of rewards.

The rewards are stochastic, so we consider the expectation of them

$$\sum_{t=1}^{T} E\left(x_{I_t}(t)\right) = \sum_{t=1}^{T} \mu_{I_t}$$

Instead, one can minimize the pseudo regret

$$R(T) = T\mu^* - \sum_{t=1}^{T} \mu_{I_t}$$

Greedy Algorithm: Explore-First with parameter N

1. Pull each of K arms N times! (Exploration)
2. Choose the best arm!
3. Pull the best arm again (Exploitation)
4. Recompute the average reward of arms
5. Go to step 2 and continue

- Theorem: If T is known, Explore-first achieves regret:

$$R(t) \leq T^{\frac{2}{3}} \times O(K \log T)^{\frac{1}{3}}.$$

The performance of the exploration phase is terrible.

- Theorem (P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire): Fix time horizon T and the number of arms K. For any bandit algorithm, there exists a problem instance such that

$$R(t) \geq \Omega(\sqrt{KT}).$$

ϵ-Greedy Algorithm:

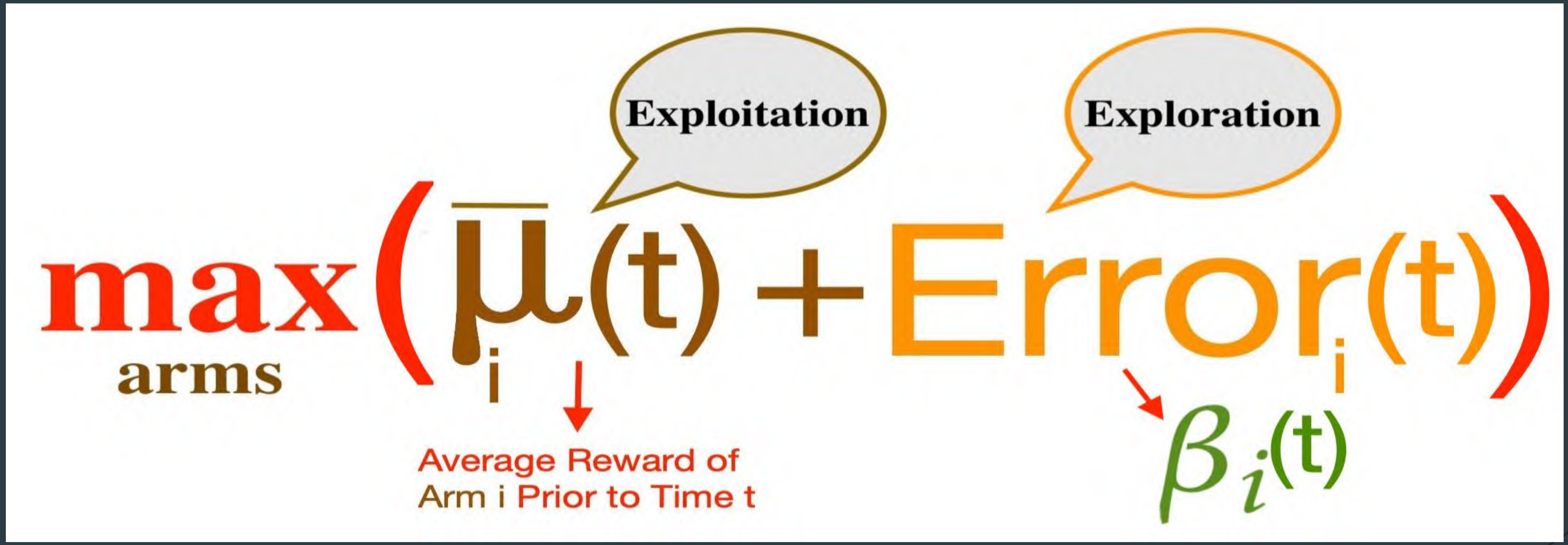
Explore with probability $\epsilon_T = T^{-\frac{1}{3}}(K \log T)^{\frac{1}{3}}$

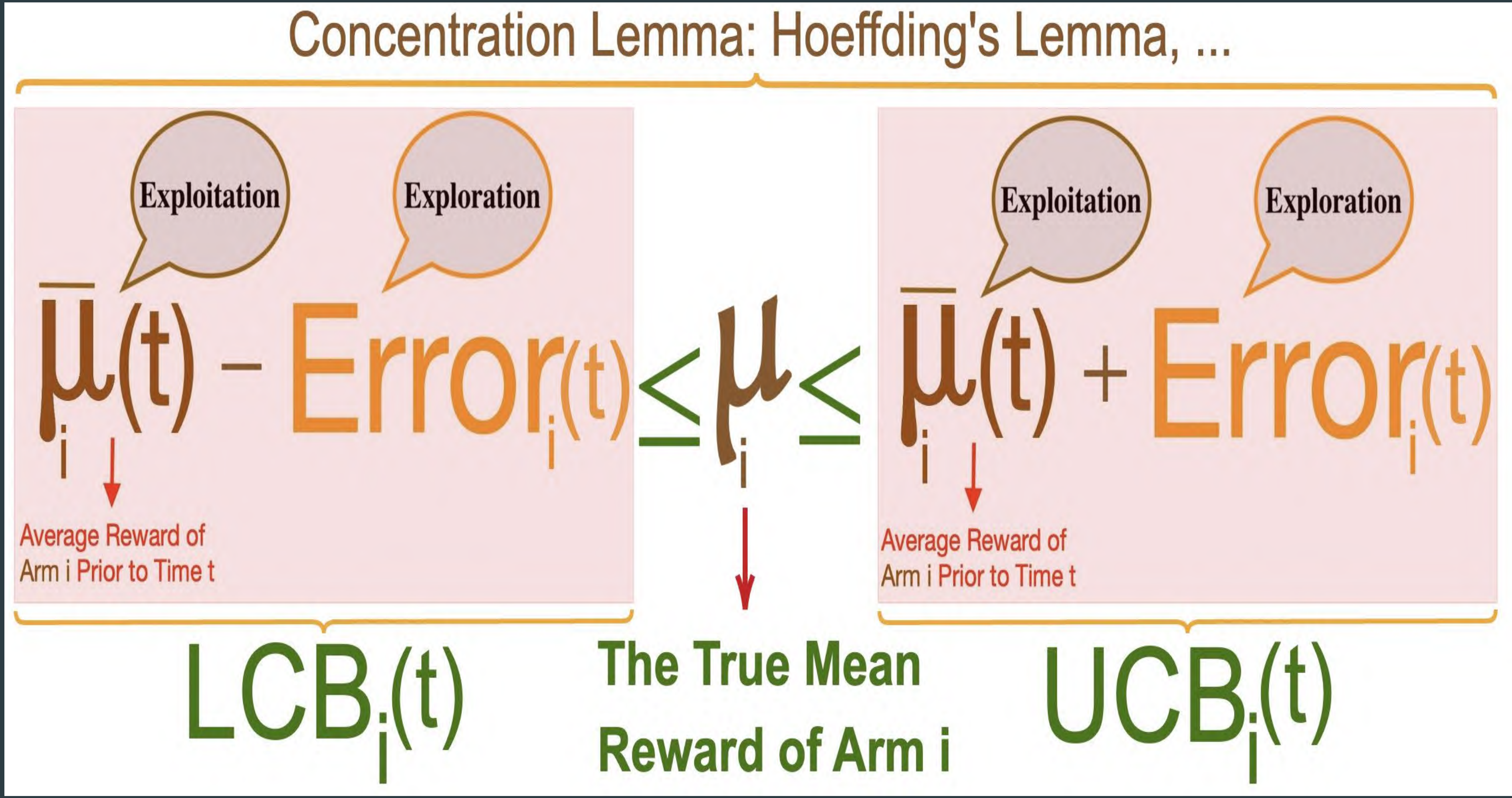
Exploit with probability $1 - \epsilon_T$

1. Pull each of K arms once!
2. At time step T, pull the best arm with probability $1 - \epsilon_T$, otherwise pull an arbitrary arm with probability ϵ_T
3. Recompute the average reward of arms
4. Go to step 2 and continue

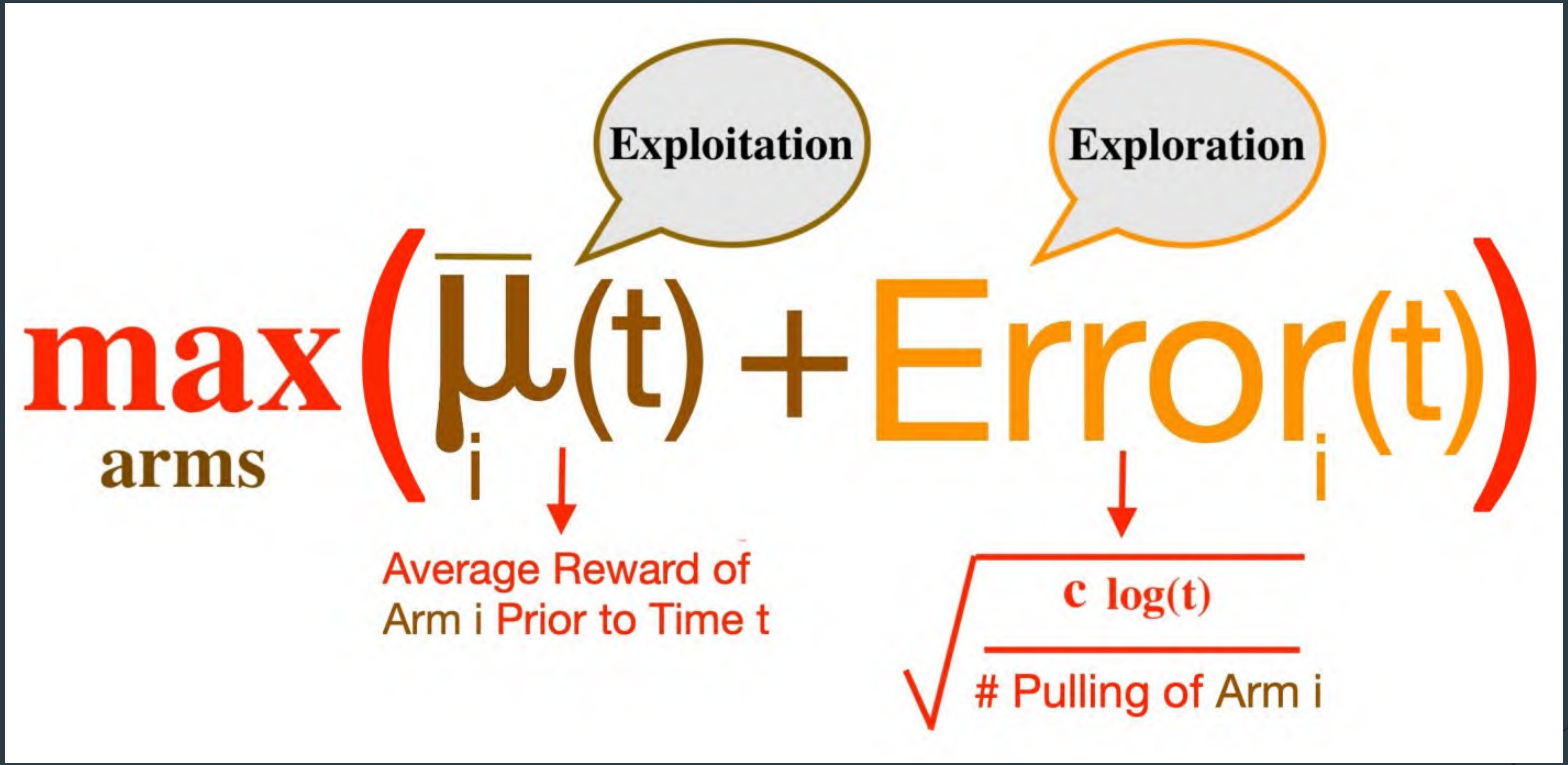
Advanced Algorithms: Adaptive Exploration

1. Pull each of K arms once!
2. Pull the arm whose empirical average reward prior to time t plus its error has the maximum value and repeat this step!




Concentration Lemma: Hoeffding's Lemma, ...
Exploitation
Exploration
$\bar{\mu}_i(t) - \text{Error}_i(t) \leq \mu_i \leq \bar{\mu}_i(t) + \text{Error}_i(t)$
Average Reward of Arm i Prior to Time t
Average Reward of Arm i Prior to Time t
$LCB_i(t)$
The True Mean Reward of Arm i
$UCB_i(t)$

Upper Confidence Bound Bandit (UCB1):



- **Theorem** (P. Auer, N. Cesa-Bianchi, and P. Fischer): for all rounds $t \leq T$, Algorithm achieves regret:

$$R(t) = O(\sqrt{Kt \log T}).$$

Bayesian Algorithms:

1. Pull each of K arms once!
2. Pull the arm whose empirical average reward prior to time t plus its error has the maximum value and repeat this step!
3. Bayesian: $\max_i\{\bar{\mu}_i(t) + \bar{\sigma}_i(t) \times \Phi^{-1}(1-\alpha_t)\}$, where $\bar{\mu}_i(t)$ and $\bar{\sigma}_i(t)$ are the Bayesian mean and Bayesian variance. Also, Φ^{-1} stands for quantile function of the standard normal random variable!

Question: is it possible to introduce an algorithm with finite regret? NO!

1. In practice, we are often happy to perform a task just good enough. For example, when driving to work we will be content with a strategy that will let us arrive just in time.

2. We only care about whether an arm with mean reward $\geq S$ is chosen, where S is the level of satisfaction we aim at.

3. Modify the classic notion of regret and consider, the satisficing (pseudo-)regret with respect to S (short S-regret) defined as:

S-Regret:

$$R_S(T) = \sum_{t=1}^{T} \max_{I_t}\{S - \mu_{I_t}, 0\}$$

Regret:

$$R(T) = T\mu^* - \sum_{t=1}^{T} \mu_{I_t}$$

➢ Theorem (T. Michel, H.H., R. Ortner):

If $S < \mu^*$, there exists a constant C such that for any T,

$$R_S(T) \leq C.$$

ALGORITHM

Input: K, S

▷ Play each arm once.

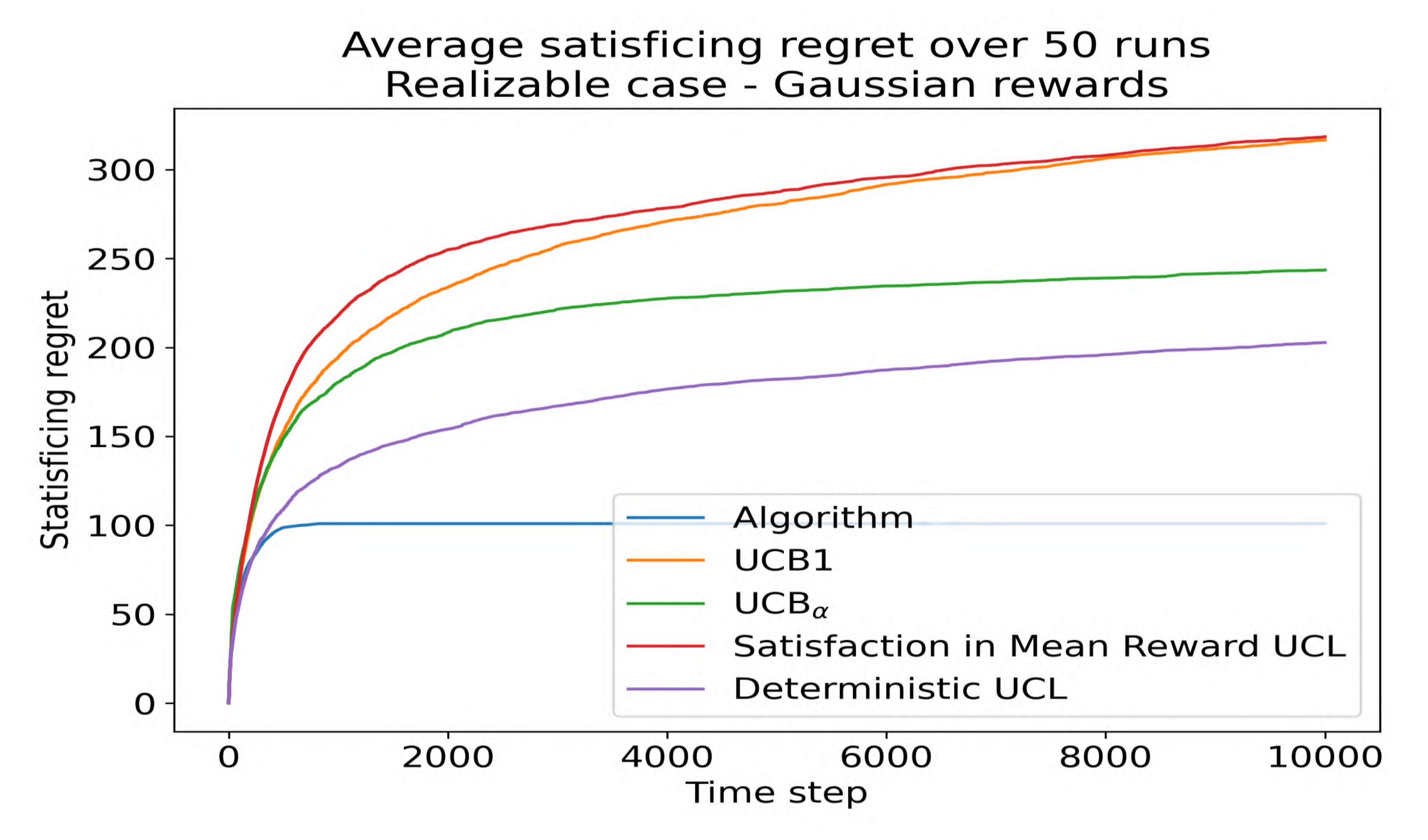
▷ **For** each further step t **do**

If ∃ arm i with $\widehat{\mu_i} \geq S$ play

$$A_t \in arg \max_{1 \leq i \leq K} \left\{ \frac{UCB_i(t) - max\{S, LCB_i(t)\}}{\beta_i(t)} \right\}$$

Else If ∃ arm i with $UCB_i(t) \geq S$ choose A_t uniformly at random from $\{i \mid UCB_i \geq S\}$.

Else choose $A_t \in arg \max_{1 \leq i \leq K} UCB_i(t)$.


Average satisficing regret over 50 runs
Realizable case - Gaussian rewards
Statisficing regret
Time step
0
50
100
150
200
250
300
0
2000
4000
6000
8000
10000
Algorithm
UCB1
UCB$_\alpha$
Satisfaction in Mean Reward UCL
Deterministic UCL

Exploration versus Exploitation: Based on the goal, the optimal algorithms may move towards more exploration than exploitation.

Assume that there are just two arms.
The distribution of the rewards of each arm is Gaussian with unknown mean and the same variance.
The goal is the best arm identification: minimize the error probability at time 2T.

Algorithm:

1. Pull each arm T times!

Applications and Variants

1. K treatments for a given symptom (with unknown effect)
What treatment should be allocated to the next patient, based on responses observed on previous patients?

2. K adds that can be displayed
Which add should be displayed for a user, based on the previous clicks of previous (similar) users?

1. Assign to each person a feature or a context.
2. Imagine that each machine responds differently to each person.
3. You need to find the best strategy for the given context.
4. K treatments and K ads can be considered as contextual bandit.
5. Variants: Adversarial Bandit, Infinite-Armed Bandit, Non-Stationary Bandit, and so on

tanQ 4 @enshen!!!

Goodbye!!!!