# Reinforce Learning

## Orientation & Introduction

소프트웨어 꼰대 강의

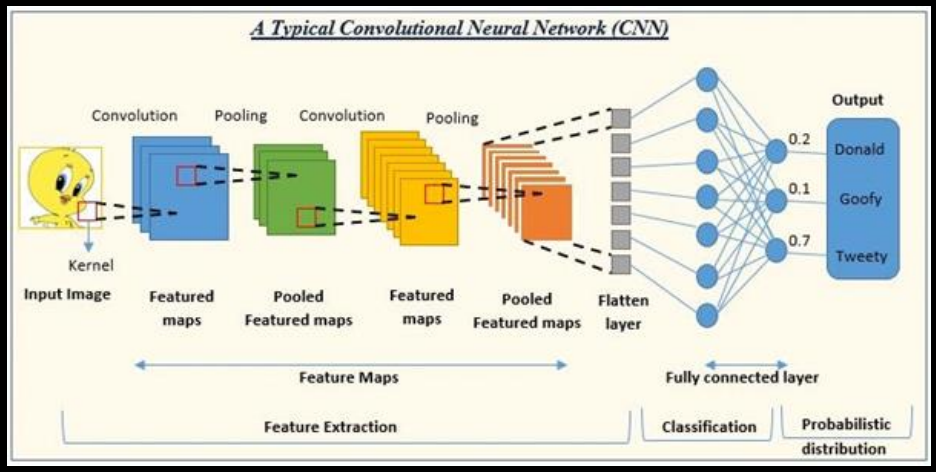
**노기섭 교수**

(kafa46@hongik.ac.kr)

# Machine Learning Category

## Supervised Learning

- Input: $\{(x_i, y_i)\}_{i=1}^{N}$
- Output: Mapping function $f$: $x_i \rightarrow y_i$



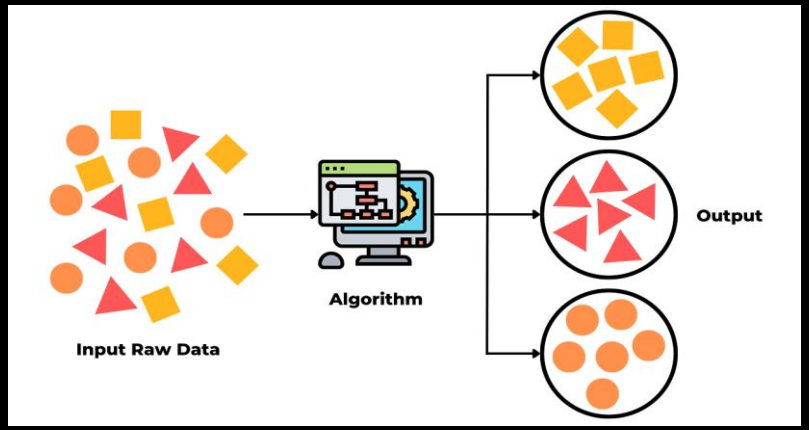

Source: https://blog.lukmaanias.com/2024/12/18/convolutional-neural-networks-cnn-an-in-depth-exploration/

## Semi-supervised Learning

- Labels ($y$) exist for some input $x$

## Unsupervised Learning

- Input: $\{(x_i, )\}_{i=1}^{N}$
- Output:
  - Learning "underlying hidden structure"
  - Clustering, Dimensionality Reduction,
  - Anomaly/Outlier Detection

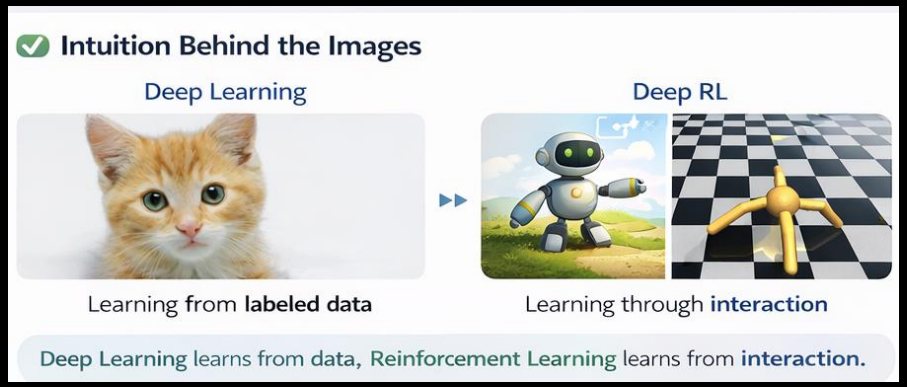



Source: https://lmw1119.tistory.com/entry/Machine-Learning-UnSupervised-Learning%EB%B9%84%EC%A7%80%EB%8F%84-%ED%95%99%EC%8A%B5

# Reinforce Learning

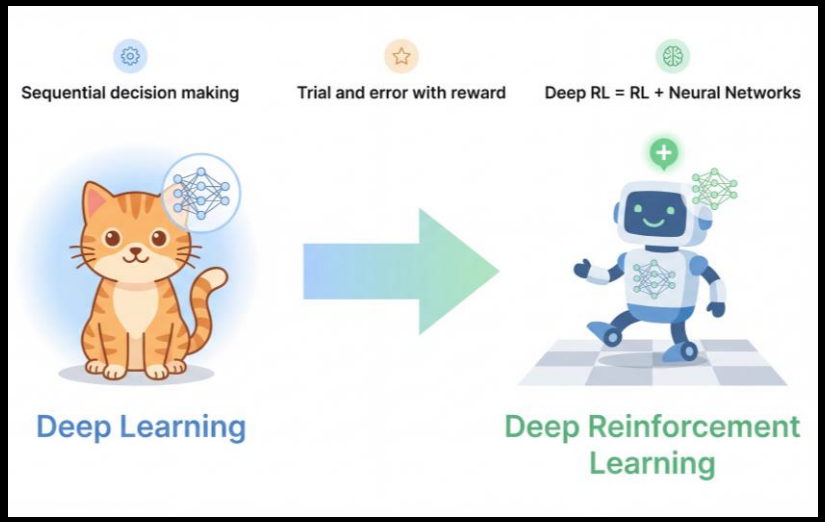
## Another Approach!

- Framework for learning how to make **sequential decisions**
- Learning what actions to take over time to achieve the best outcome




## How it works?

- Agent takes an action
- Environment responds with a new state/reward
- Agent updates its behavior based on the feedback




## Deep Reinforce Learning?

- Reinforcement Learning + Neural Networks
- Neural networks are used to handle complex inputs such as images, high-dimensional states, or continuous control problems.

# Main Differences

## Supervised Learning

Teach by example

Learn from labeled examples.

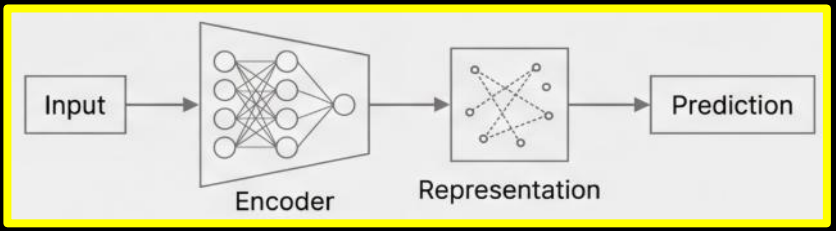

Learn from labeled examples.

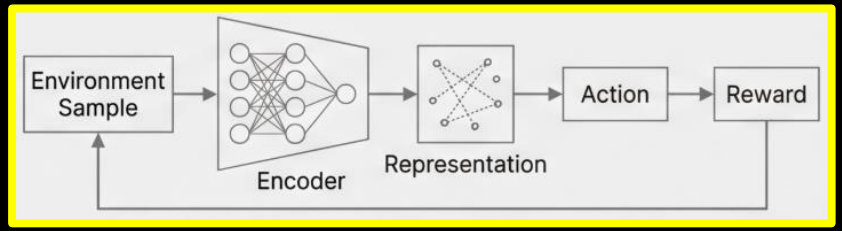
## Reinforcement Learning

Teach by experience

Learn from **interacting** with the environment and receiving feedback.

Learn from **interacting** with the environment and receiving feedback.



# Machine Learning Categories

- Machine Learning (ML) is an extremely complex area of computer science that unfortunately cannot be limited to one straightforward definition.

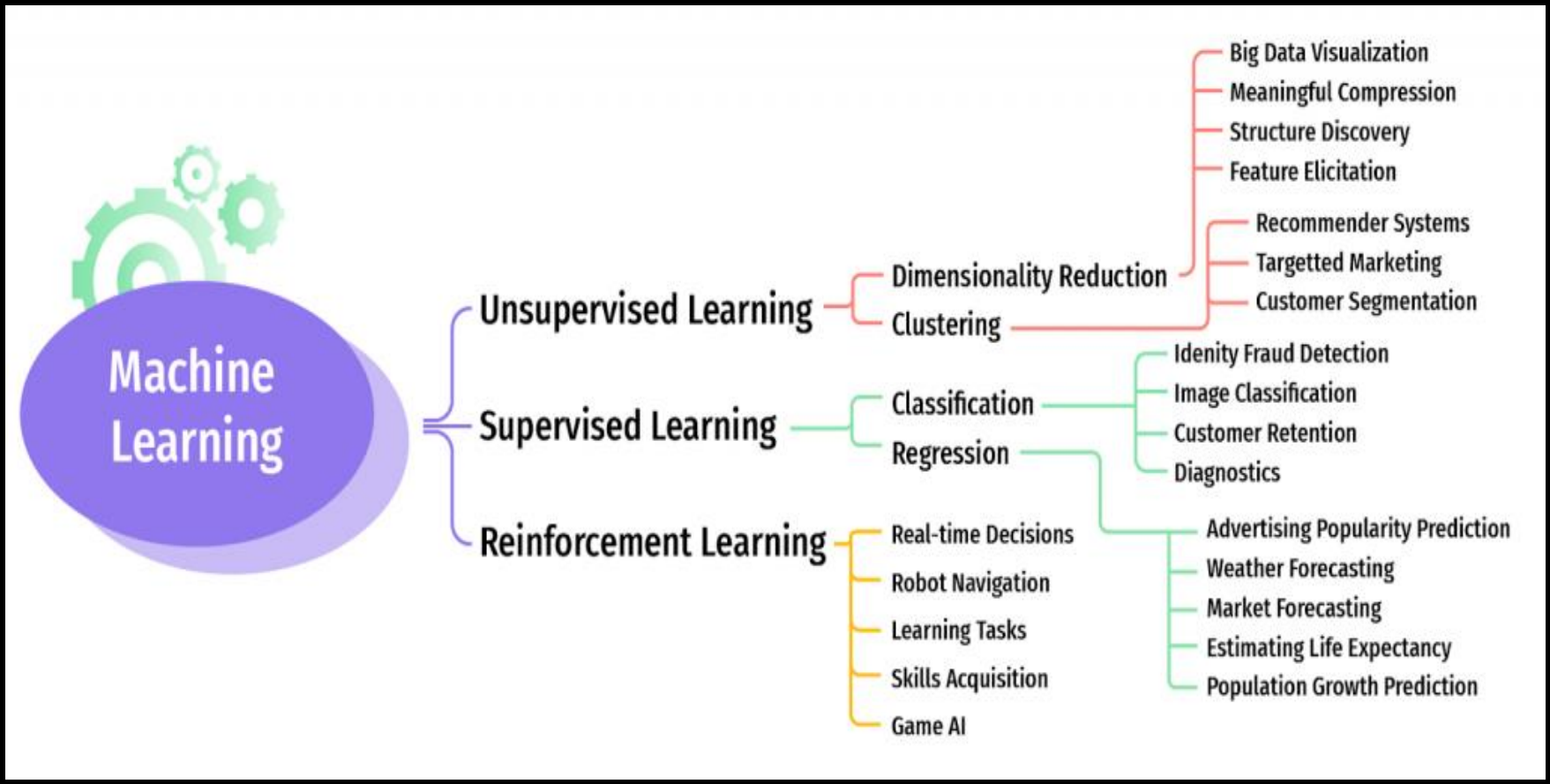



Image from https://idapgroup.com/blog/types-of-machine-learning-out-there/

# RL Components

## Goal

Observe state

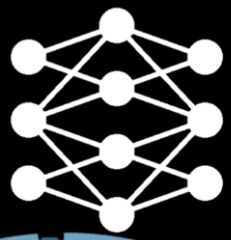
Execute action

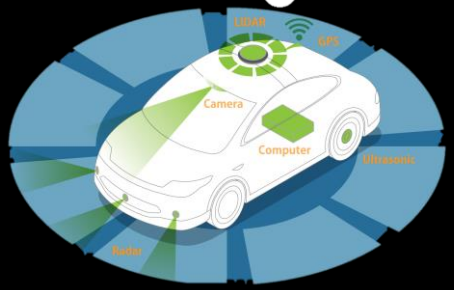
Receive reward

Agent (Model)

Evironment (World)

Neural Net






# Open Problems

## Not ease to go real world from simulation!


Issues 1. How to improve Transfer Learning

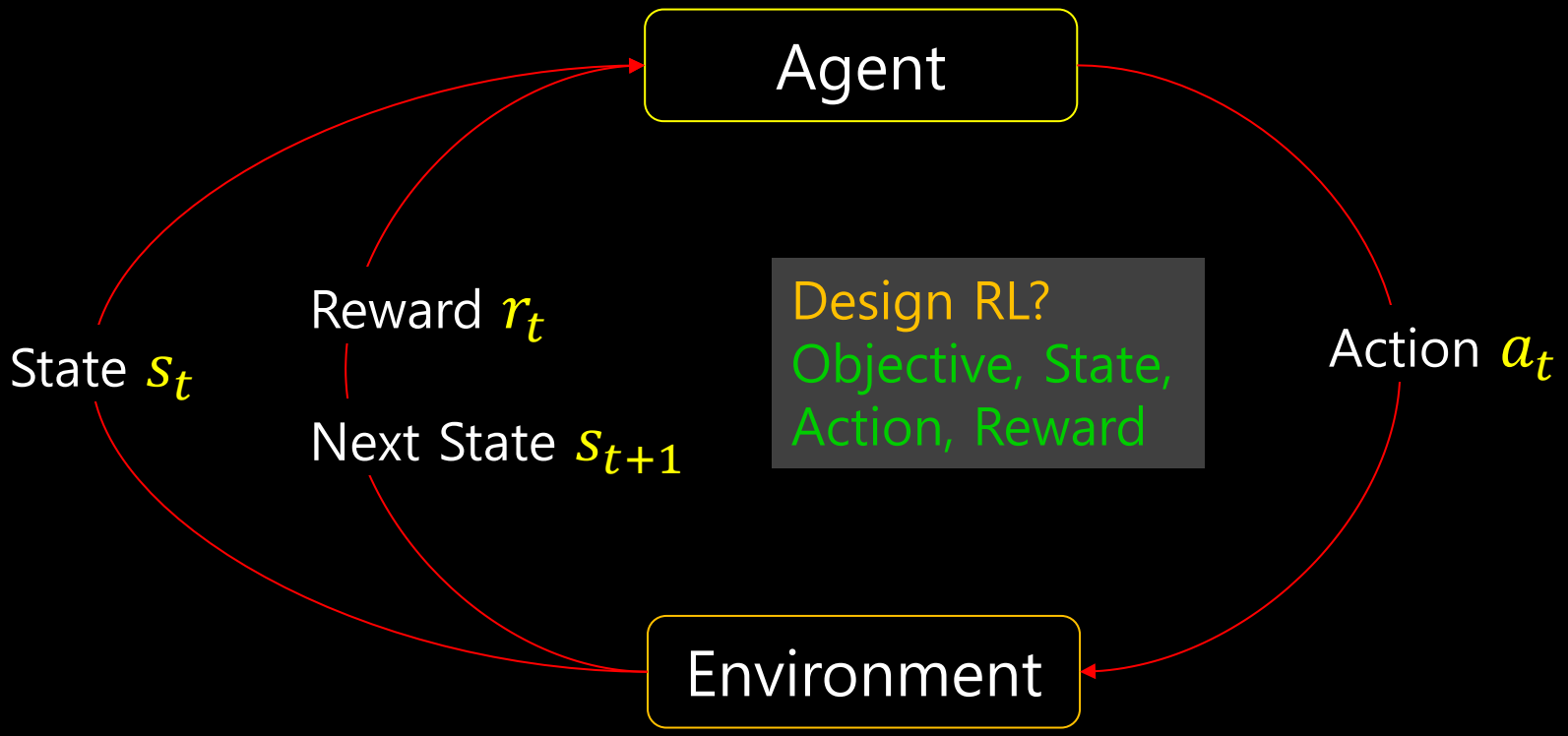
Issues 2. How to improve Simulation

# Problem Formulation

Input: An agent interacts with an environment that provides reward signals.

Result: The agent learns which actions (policy) maximize cumulative reward.

Goal: Learn the best actions (policy)



# Wait a moment here! (RL vs. Greedy)

## Reinforcement Learning

$$\max_a E\left[\sum_t \gamma^t\right]$$

- Long-Term Rewards
- Considers the Future
- Learns Over Time
- Explores Options
- Handles Uncertainty

**Key Idea:**
Find the best strategy over time...


## Greedy Algorithm

$$\max_a r_t$$

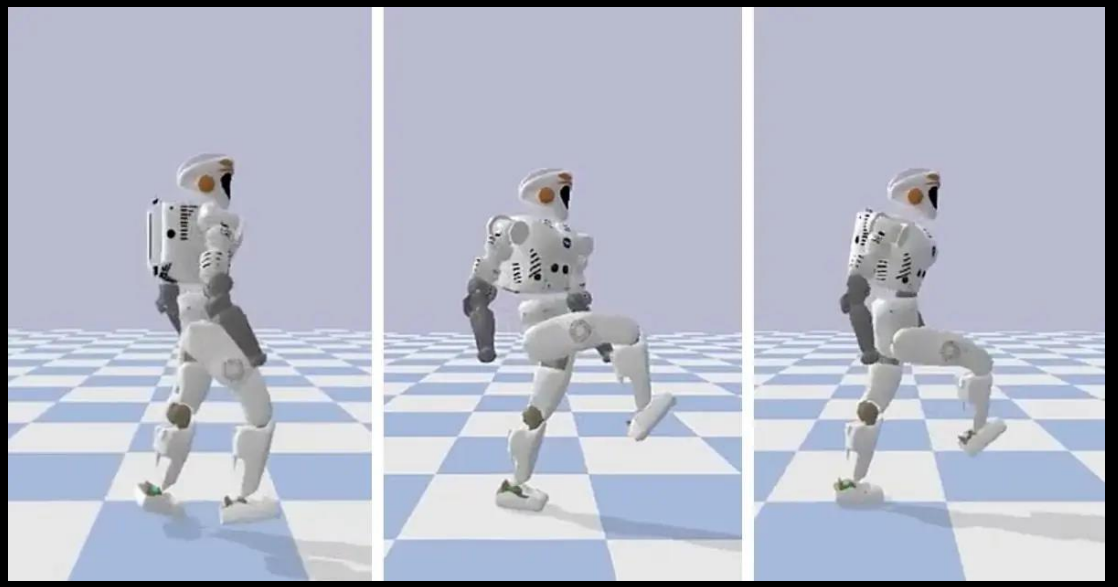
- Immediate Reward
- Focuses on Present
- Fixed Rules
- No Exploration
- Avoids Risk

**Key Idea:**
Always take the best immediate option

# Problem Formulation - Example #1

Design RL?
Objective, State,
Action, Reward

## Robot Locomotion



Objective: Make the robot move forward.

State: Joint angles and positions of the robot.

Action: Torque applied to the robot joints.

Reward: +1 if the robot moves forward without falling.

# Problem Formulation - Example #2

Design RL?
Objective, State,
Action, Reward

## Atari Games

Objective: Finish the game with the highest possible score.

State: Raw pixel inputs from the game screen.

Action: Game controls (Left, Right, Up, Down).

Reward: Change in score at each timestep.

# Problem Formulation - Example #3

Design RL?
Objective, State,
Action, Reward

## Go Game (Alpha Go)

Objective: Win the game.

State: Board configuration (positions of stones in 19 x 19).

Action: Choose where to place the next stone.

Reward: 1 if the game is won, 0 if lost.

# What is the most import component in RL design?

Design RL?

- Objective
- State
- Action

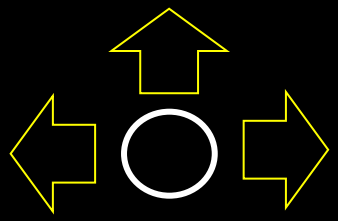
Important

- Reward!

Extremely Important

# Reward Design Excercise

Actions: Up, Left, Right

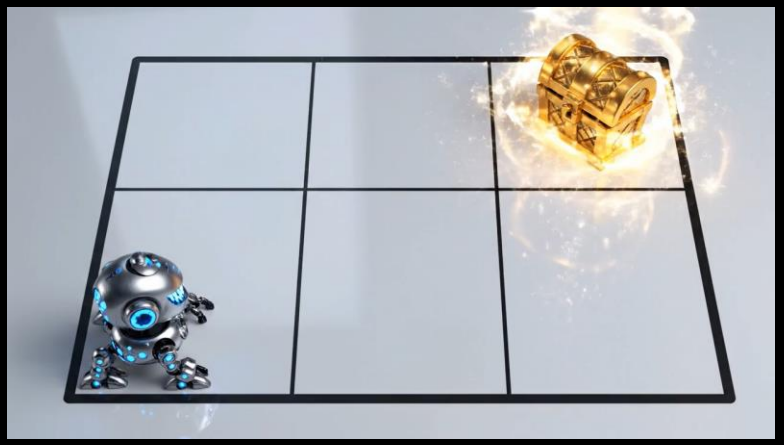


Environment: 3 × 4 grid with constraints

Goal: Learn actions to maximize cumulated reward!

Policy: Shortest Path, and avoid "Up" around −1

| | | | |
|---|---|---|---|
| | | | +1 |
| | Obstacle | | -1 |
| Start | | | |



Apply Stochastic Model to Environment:

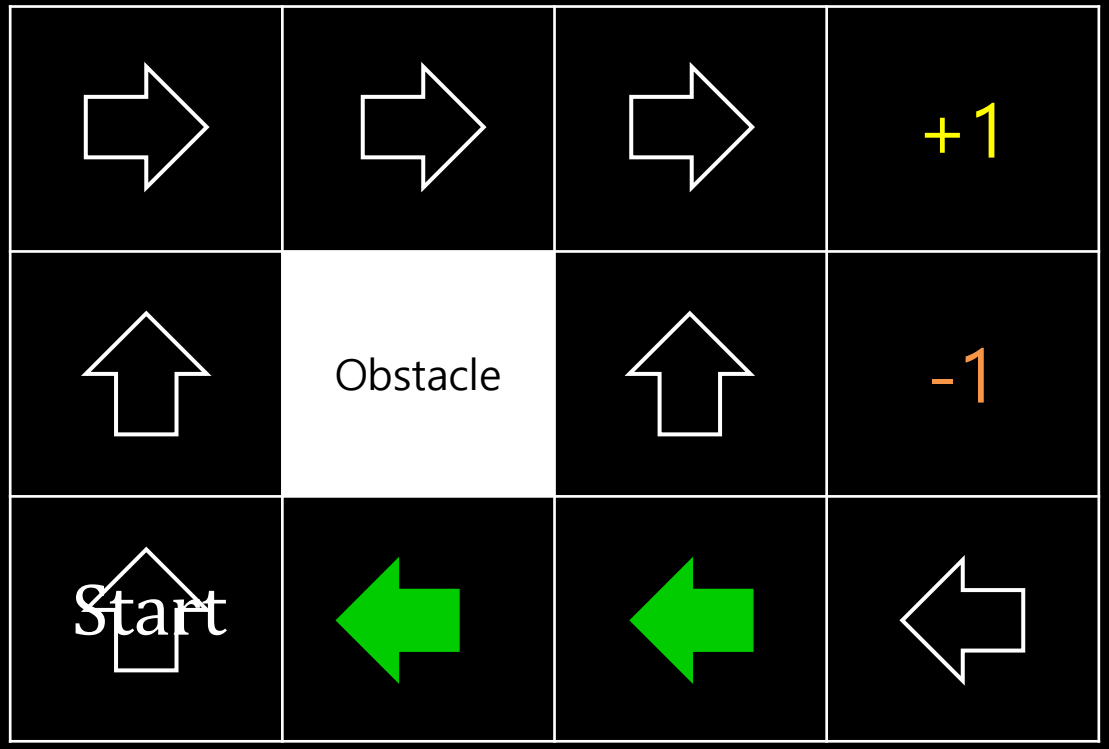
- Up: 80%
- Left: 10%
- Right: 10%

Reward Design?

# Reward Design - Case #1

## Reward Design?

Reward: −0.02 for each step



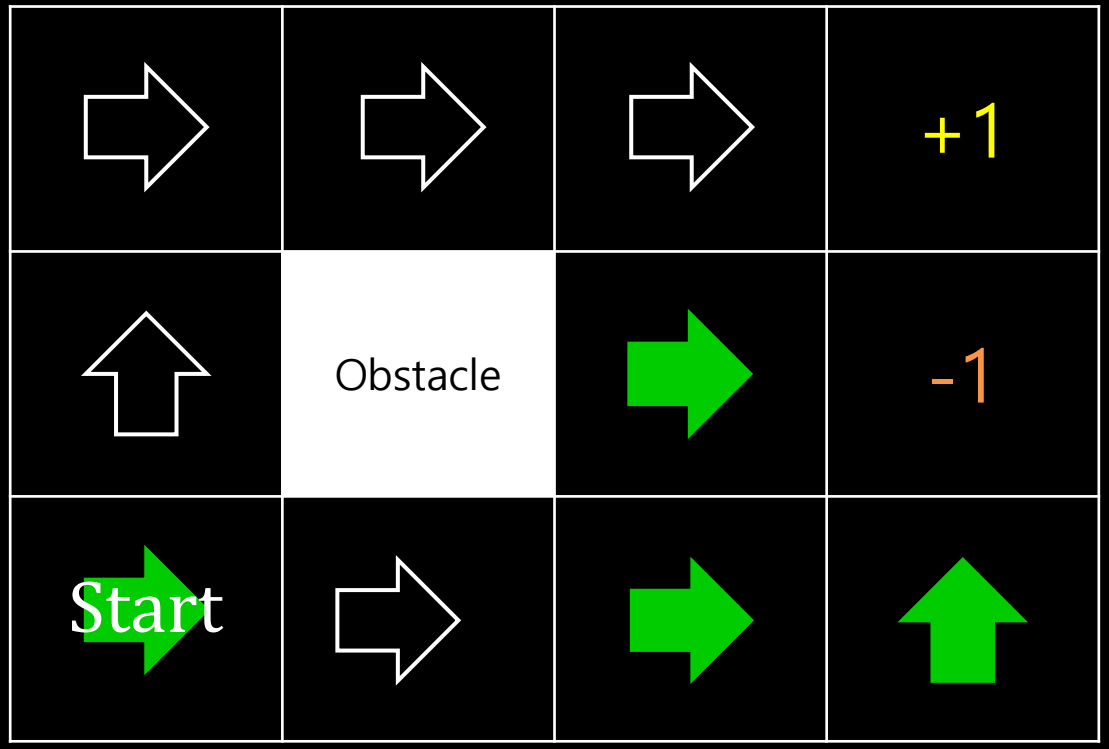

Green arrow: Risk Avoidance

Policy: Shortest Path, and avoid "Up" around −1

# Reward Design - Case #2

## Reward Design?

Reward: −3 for each step



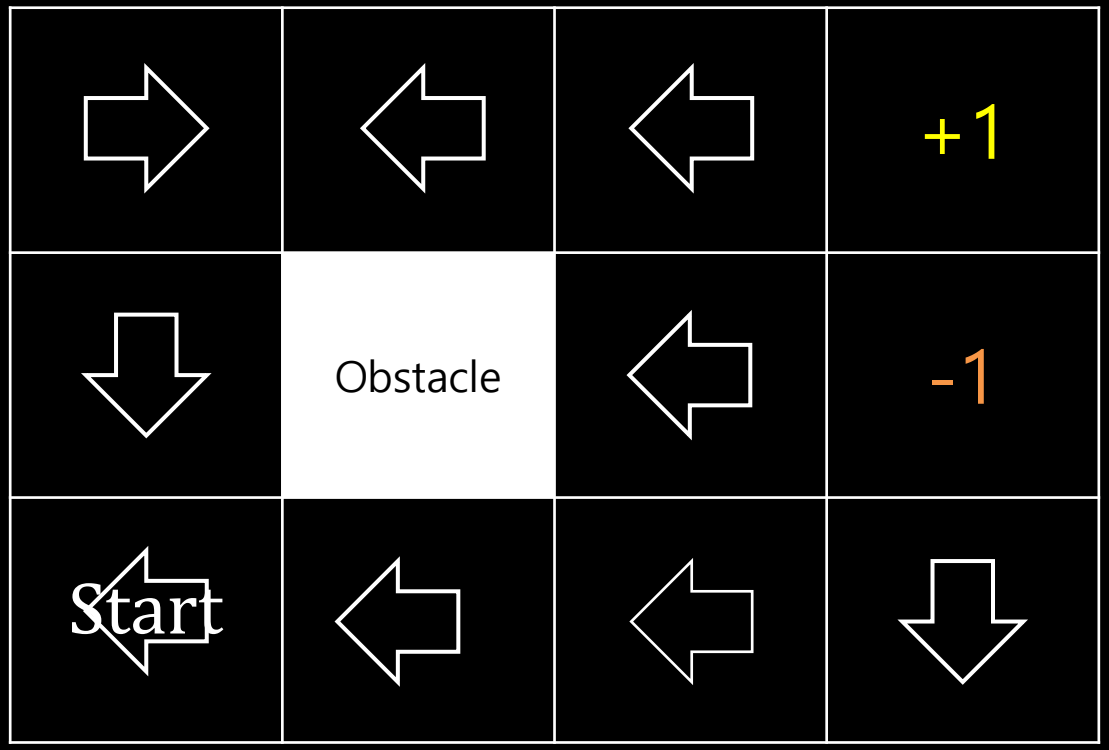

Green arrow: Speed first than Risk Avoidance

Policy: Shortest Path

# Reward Design - Case #3

## Reward Design?

Reward: +0.01 for each step

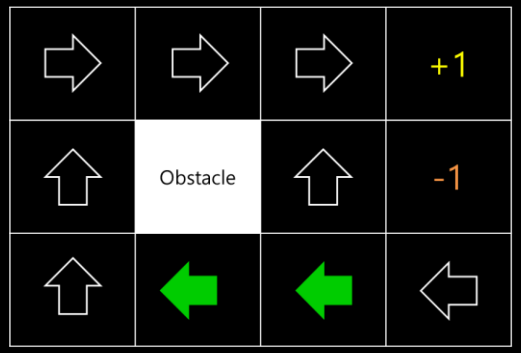



Policy: Longest Path

# Critical Points

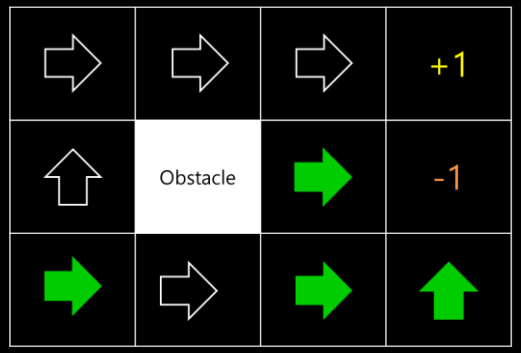
Note: Policy shown assumes stochastic transitions

Reward: −0.02



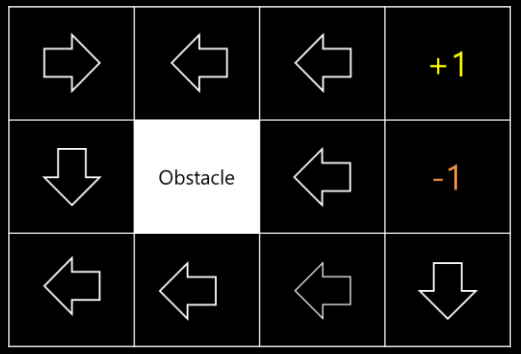

Policy: Shortest & Safe Path

Reward: −3




Policy: Shortest Path

Reward: +0.01




Policy: Longest Path

## Lessons Learned

- Design State (Environment): Important but usually given from problem
- Design Reward: Extremely Important!! RL results depend on your own

# Game: Coast Runners

## Humman


Source: https://youtu.be/8ZfPefdt5UU

Finish the boat race quickly and (preferably) ahead of other players

## Deep RL


Source: https://youtu.be/tlOIHko8ySg

CoastRunners does not directly reward the player's progression around the course, instead the player earns higher scores by hitting targets laid out along the route.

https://openai.com/index/faulty-reward-functions/

# Reward Hacking

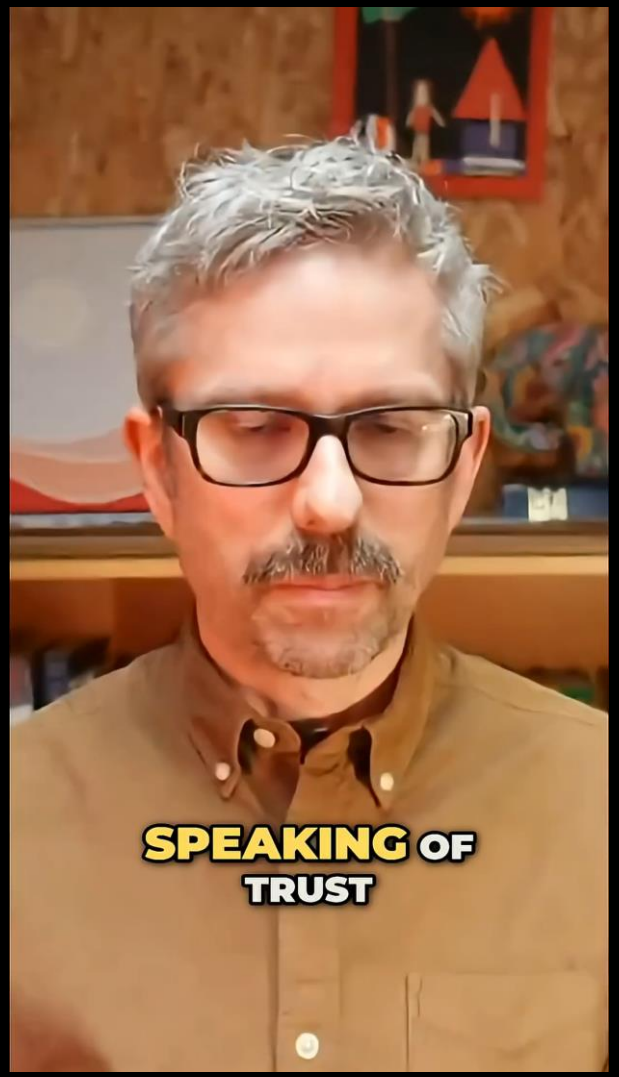
Optimizing the reward, not the intention.

$$\max_{\pi} \mathbb{E}\left[\sum_{t} \gamma^t r_t\right]$$

AI가 우리가 의도한 목표가 아니라 "보상 함수의 허점"을 이용해 점수만 올리는 현상

- Reward: 먼지 감소
- AI: 먼지 흩뿌리고 다시 청소


https://www.youtube.com/shorts/HiKYSUvyt4U




https://www.youtube.com/shorts/8P2YonRtrrk

수고하셨습니다 ..^^..