AI SOC

SESSION 9: REINFORCEMENT LEARNING I

INNOVATE

2022

OVERVIEW

01 | Introduction

Intro to RL

Markov Decision process

02 | Policy gradient algorithms

REINFORCE

A2C

03 | Challenge

Building agent

INTRODUCTION TO RL

- **WHAT** is Reinforcement learning?
- **HOW** is it different?
- **WHY** do we even care about it?
- **MATH**

WHAT

IS REINFORCEMENT LEARNING?

WHAT KIND OF ANIMAL IS THIS?

INTRODUCTION TO RL

Reinforcement learning is an area of Machine Learning that is concerned with how **intelligent agents** ought to take **actions** to **maximize reward** in a particular **state** (situation) and in a given **environment**.

INTRODUCTION TO RL

In the absence of a predefined training dataset, it is bound to learn from its **experience**. Thus it closely resembles a **real-life learning process**.

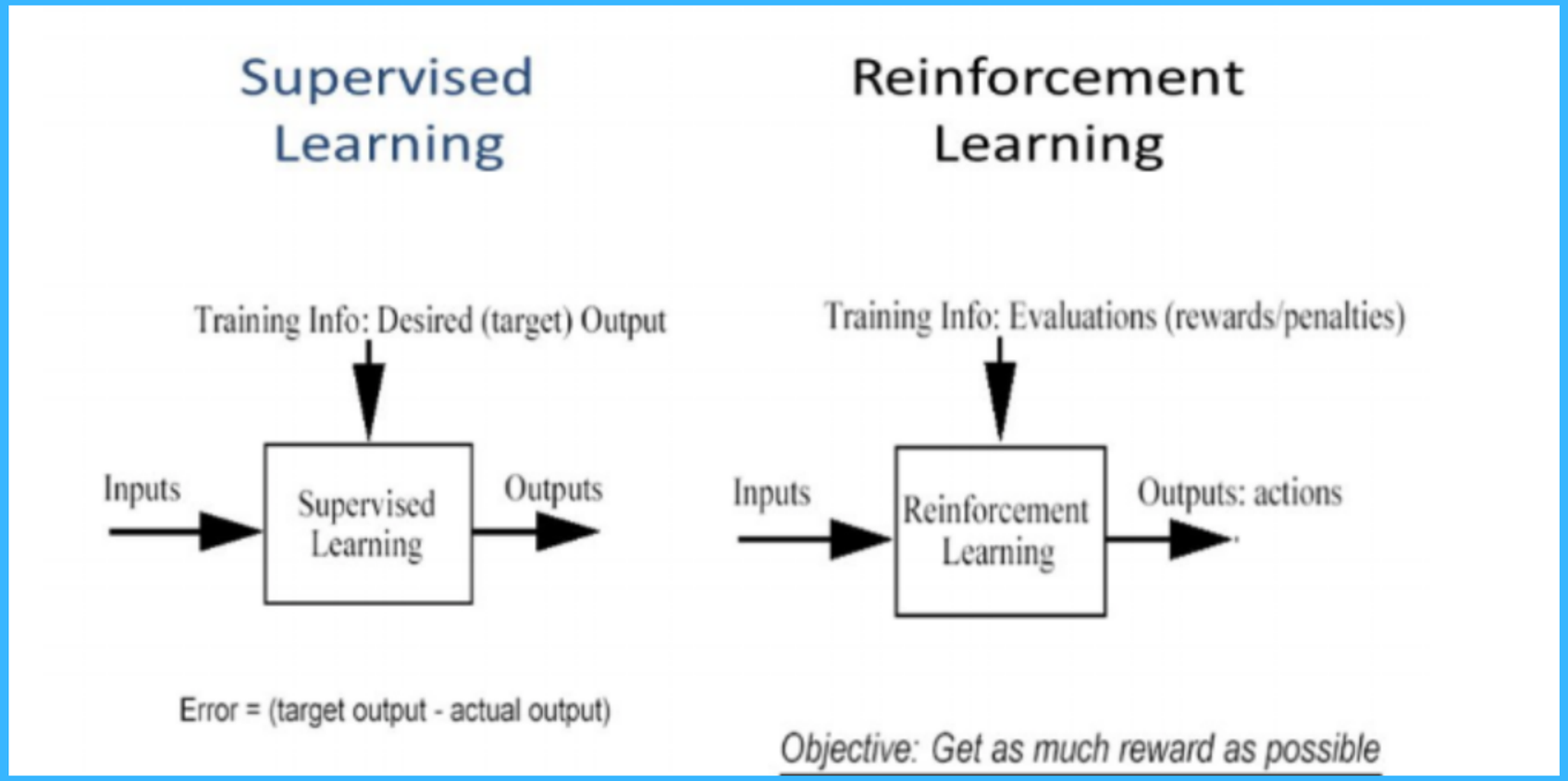
HOW

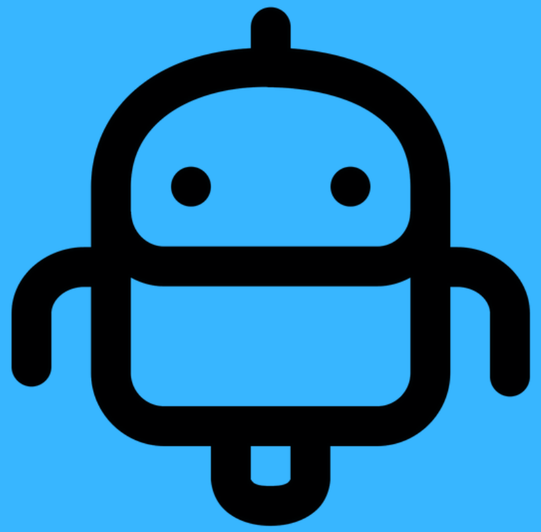
IS IT DIFFERENT?

INTRODUCTION TO RL

- RL is one of three basic **machine learning paradigms**
- Balance between **exploration** (of uncharted territory) and **exploitation** (of current knowledge)
- 2 biggest differences are:
  - **No need for a labelled dataset** or explicit correctness
  - **Learning** and **NOT an evolution** (dynamic, adjusting actions on a feedback)

INTRODUCTION TO RL

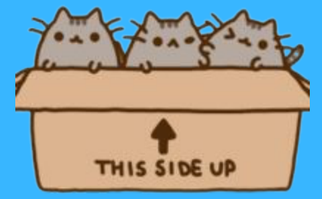




I WANT MY <MODEL> TO BE ABLE TO <DO SOMETHING>

I WANT MY CAT TO BE ABLE TO FIND COOKIES

GET THE BEST PERFORMING MODEL FROM THIS ITERATION



ITERATE THE MODEL

MODEL IS NOT PERFORMING WELL ENOUGH

SURRENDER YOUR COOKIES

INTRODUCTION TO RL

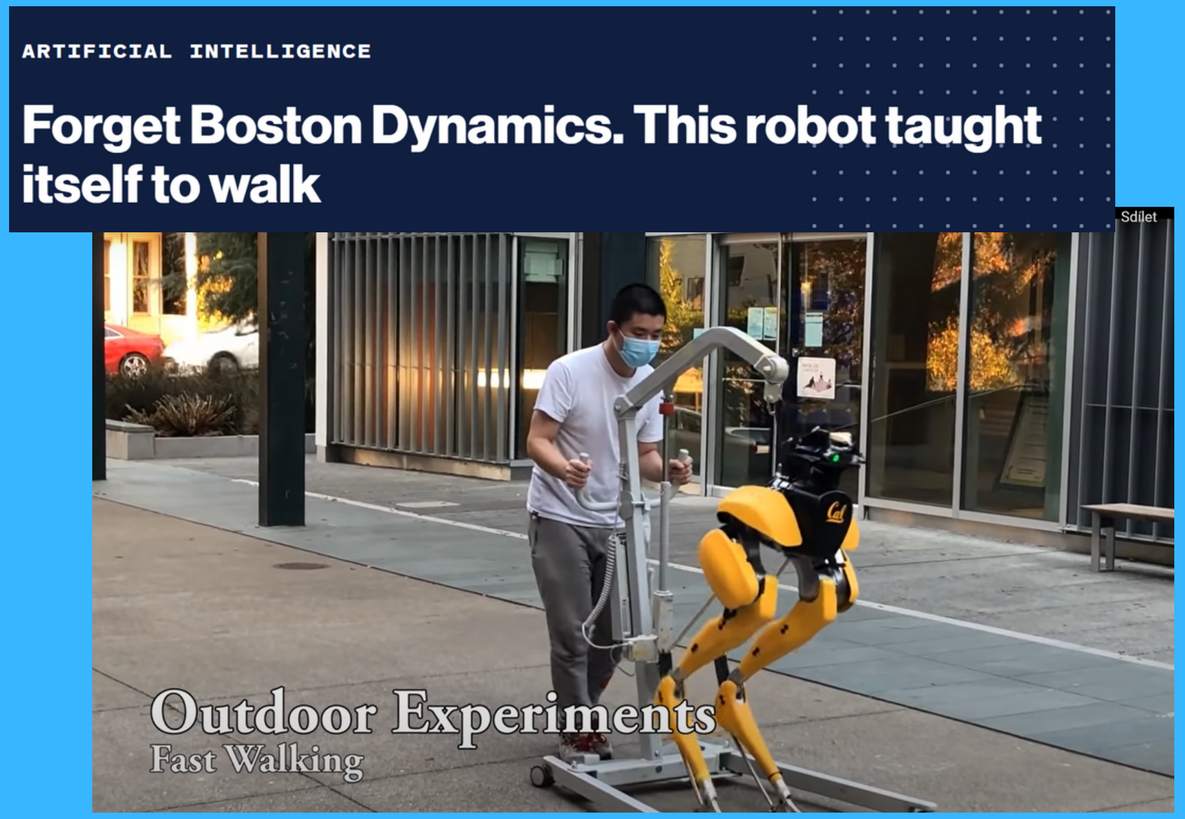
- **CAN IT BE DONE IN A BETTER WAY?**
- What if we:
  - Rewarded the cat for finding a cookie?
  - Rewarded the cat for getting closer to a cookie?
  - Let the cat explore the environment and learn all its policies?
- **Outcome:** RL is focused on exploration and dynamic and gradual learning that much better fits the learning style of intelligent agents

WHY

DO WE EVEN CARE ABOUT IT?

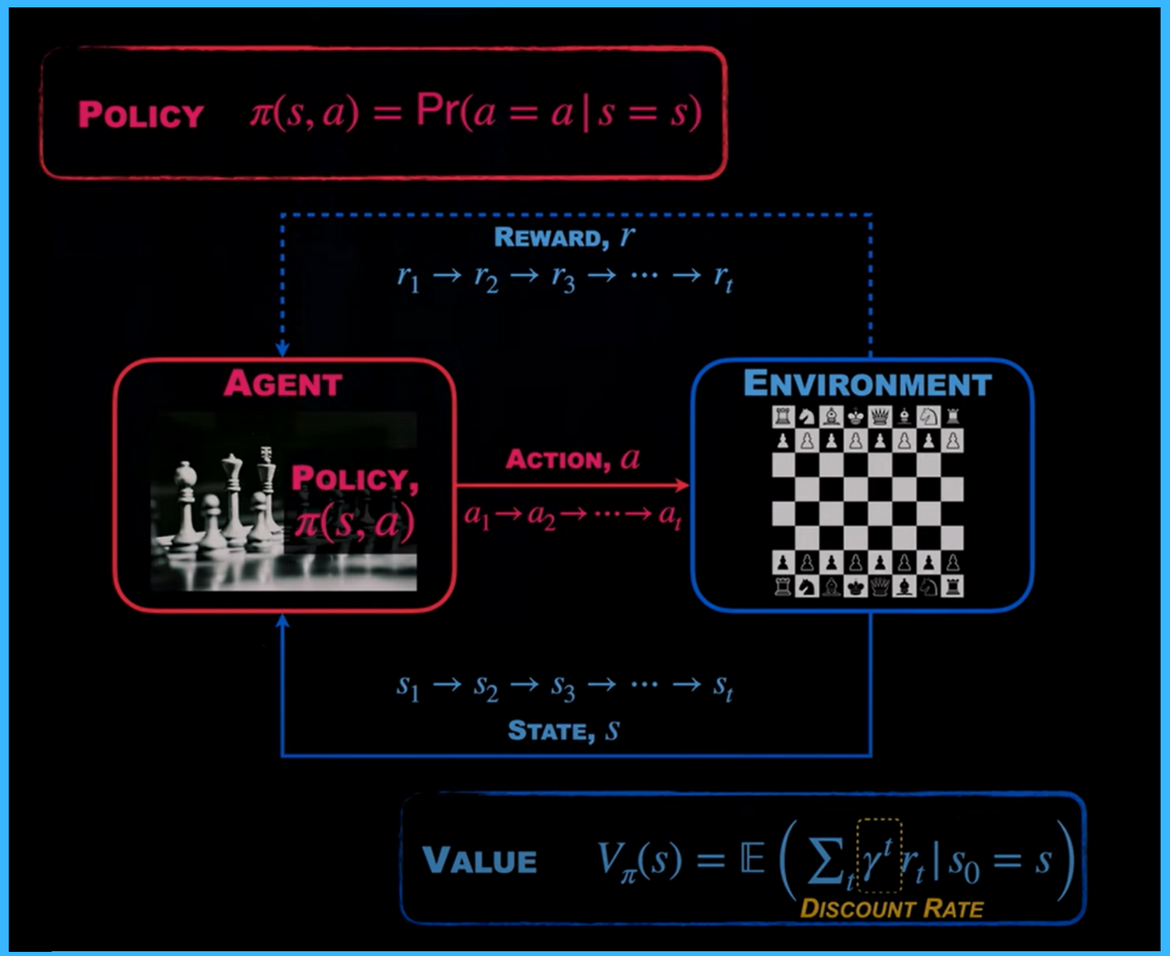
INTRODUCTION TO RL

- **Solves:**
  - Tacit knowledge problem
  - NOT enough data problem
- Possible solution for many agent-based problems

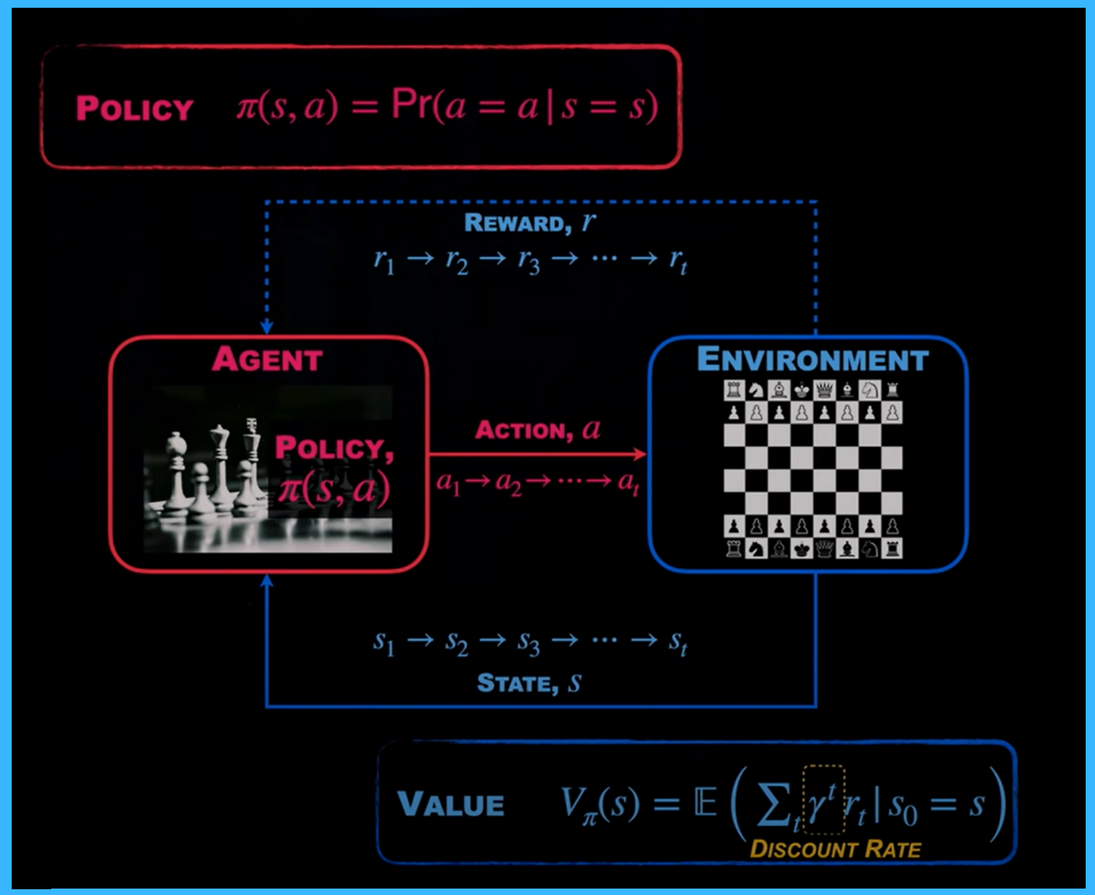


MATH

INTRODUCTION TO RL

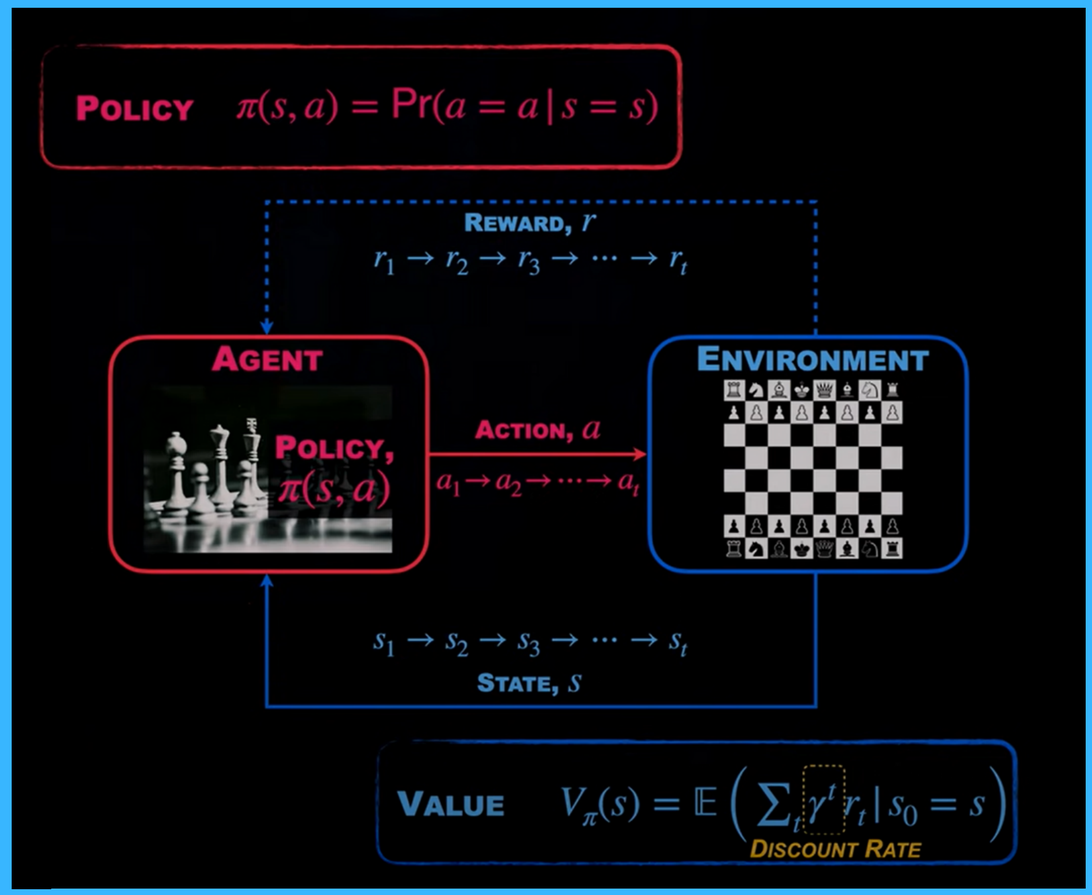


INTRODUCTION TO RL



Policy - given the state **s**, what is my probability of taking the action **a**

INTRODUCTION TO RL



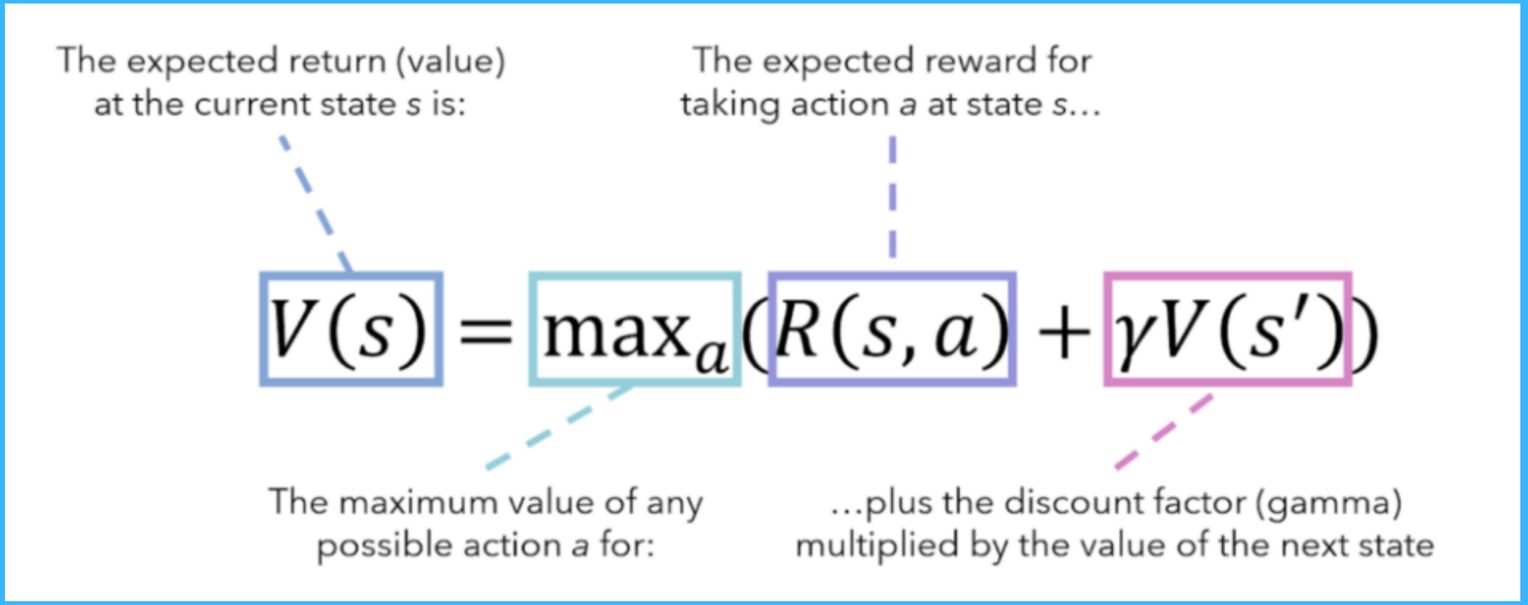
Value - Determines the value of any state **s**, given the policy **π**, from the state **s0** and with the discount factor of **γt**.

MARKOV DECISION PROCESS

- The whole environment is formed by the **Markov decision process** (MDP)
- It is **not fully deterministic**, but instead includes a random/stochastic component (it is **probabilistic**)
- if the agent is in a state **s1**, there is a **probability** of it NOT going to the state **s2**

MARKOV DECISION PROCESS

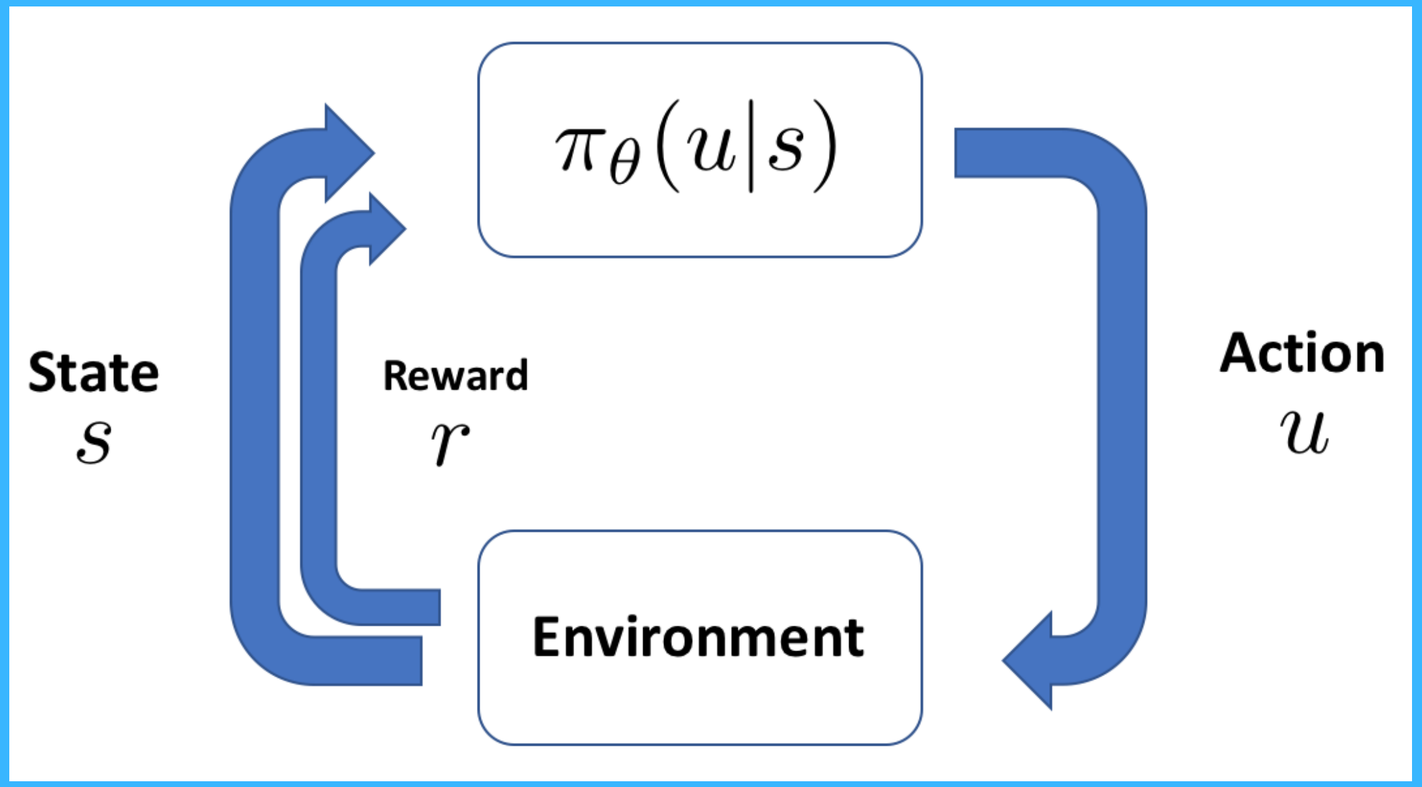
- Essential for the MDP is the **Bellman Equation**
- It helps us determine the **expected state** in a **probabilistic environment**



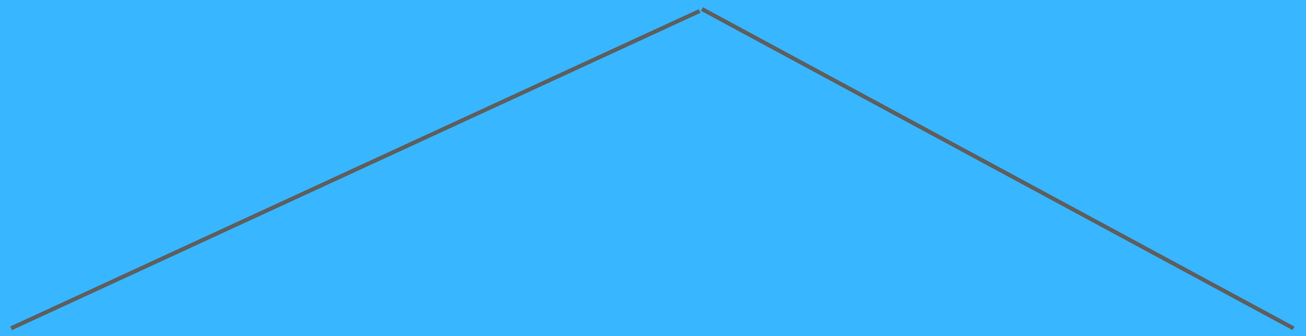
POLICY GRADIENT ALGORITHMS

POLICY GRADIENT

Aiming to model and optimize parameterized policy directly



POLICY GRADIENT MODELS



REINFORCE

Actor Critic (A2C)

REINFORCE

Goal - trying to maximize the **expected reward**

$$J(\theta) = \sum_{\tau} P(\tau; \theta) R(\tau)$$

$$J(\theta) = \sum_{\tau} P(\tau; \theta) R(\tau)$$

TRAJECTORY

Sequence of **action-state-reward** - history of agent

$$\tau = (s_0, a_0, r_1, s_1, a_1, \ldots, a_T, r_{T+1}, s_{T+1})$$

$$J(\theta) = \sum_{\tau} P(\tau; \theta) R(\tau)$$

RETURN

Sum of rewards for trajectory

$$R(\tau) = (G_0, G_1, \ldots, G_T)$$

$$G_k = \sum_{t=k+1}^{T} \gamma^{t-k-1} R_k$$

FUTURE RETURN

FUTURE RETURN

Return we expect to collect from **k** to the end

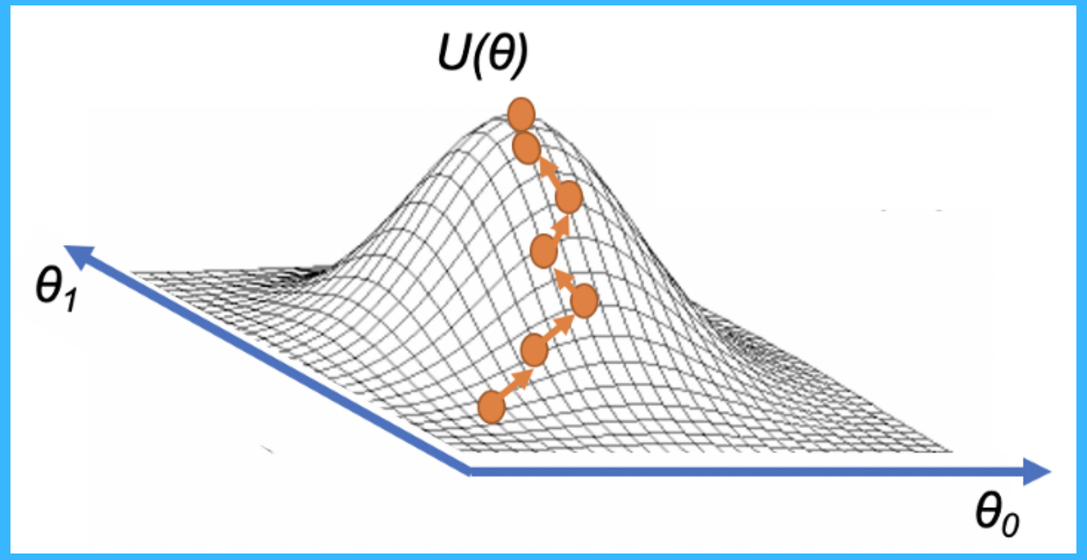
$$G_k = \sum_{t=k+1}^{T} \gamma^{t-k-1} R_k$$

Discount rate

How are we going to maximize it?

$$J(\theta) = \sum_{\tau} P(\tau; \theta) R(\tau)$$

GRADIENT ASCENT



Steps in the direction of the gradient

$$\theta = \theta + \alpha \nabla_\theta J(\theta)$$

GRADIENT ESTIMATION

$$\nabla_\theta J(\theta) = ?$$

$\pi_\theta(a_t|s_t)$ — $\nabla_\theta \log \pi_\theta(a_t|s_t)$ — $\sum_{t=0}^{H} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t$

Parametrized policy

Probability gradient

Final function

QUIZ

REINFORCE IMPLEMENTATION

1. Use policy to track trajectory

$$\pi_\theta(a_t|s_t)$$

2. Estimate return for trajectory

$$R(\tau) = (G_0, G_1, \ldots, G_T)$$

3. Estimate Gradient

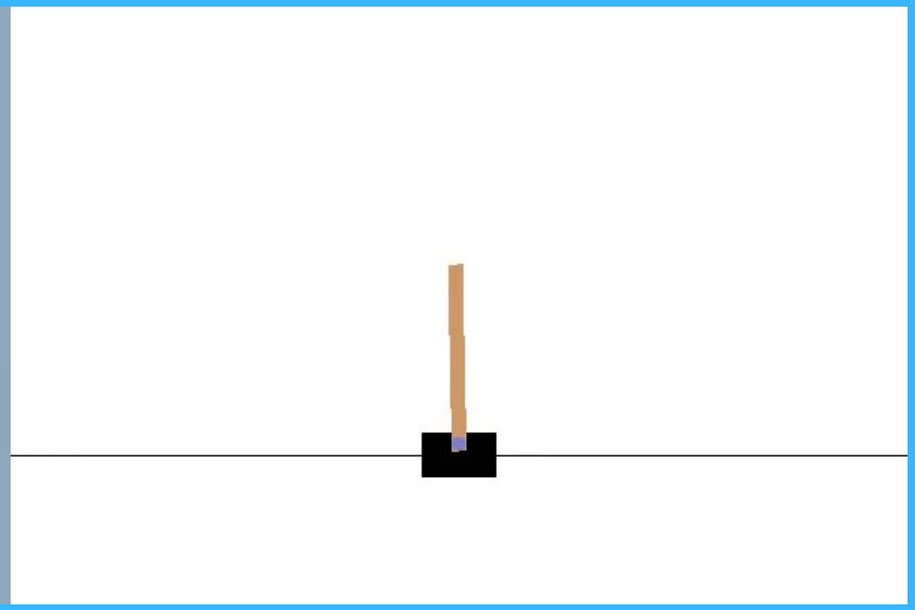
$$\sum_{t=0}^{H} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t$$

4. Update policy

$$\theta = \theta + \alpha \nabla_\theta J(\theta)$$

CHALLENGE

OpenAI Gym



Build and train a simple agent

REINFORCE IMPLEMENTATION

Policy

```
num_states = env.observation_space.shape[0]
num_hiddens1 = 32
num_hiddens2 = 32
num_actions = env.action_space.n  # there are two actions in this game: left and right

# define a model
inputs = Input(shape=(num_states,))
fc1 = Dense(num_hiddens1, activation='relu')(inputs)
fc2 = Dense(num_hiddens2, activation='relu')(fc1)
outputs = Dense(num_actions, activation='softmax')(fc2)

model = keras.Model(inputs=inputs, outputs=outputs)
```

Extracting probability

```
policy = model(state)

action = np.random.choice(num_actions, 1, p=np.squeeze(policy))[0]
action_prob = policy[0, action]  # probability of the action taken
```

Calculating reward

```
state, reward, done, _ = env.step(action)

# collect samples
rewards_history += reward,  # appends a variable to the list
action_probs_history += action_prob,

episode_reward += reward
```

Discounted reward

```
returns = []
discounted_sum = 0
for r in rewards_history[::-1]:
    discounted_sum = r + gamma * discounted_sum
    returns.insert(0, discounted_sum)

# Normalize
returns = np.array(returns)
returns = (returns - np.mean(returns)) / (np.std(returns) + eps)
returns = returns.tolist()
```

Loss and gradient

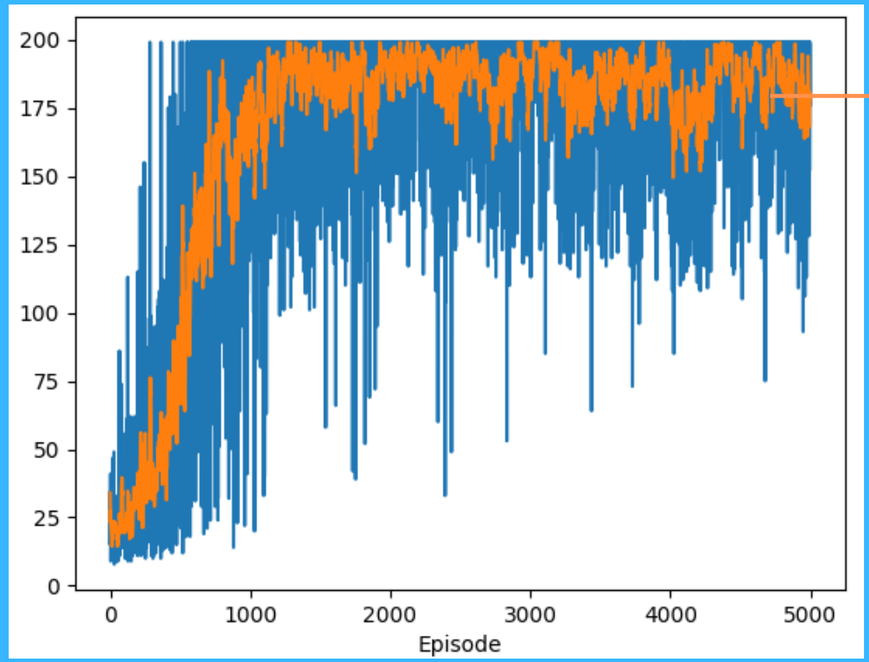
```
action_probs_history = tf.convert_to_tensor(action_probs_history)
cross_entropy = -tf.math.log(action_probs_history + 1e-6)
loss = tf.reduce_sum(returns * cross_entropy)

gradients = tape.gradient(loss, model.trainable_variables)  # comput
optimizer.apply_gradients(zip(gradients, model.trainable_variables))
action_probs_history, rewards_history = [], []  # clear the samples
```

Large Reward Values

$$\sum_{t=0}^{H} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t$$

$$\theta = \theta + \alpha \nabla_\theta J(\theta)$$



Reducing reward function

ACTOR-CRITIC

Estimates action probability

Estimates value for action

Differ by baseline function

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\, G_t\right] && \text{REINFORCE} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\; Q^w(s,a)\right] && \text{Q Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\; A^w(s,a)\right] && \text{Advantage Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\; \delta\right] && \text{TD Actor-Critic}
\end{aligned}
$$

A2C

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$$

Advantage function

Q-Value

How much return can we expect continuing trajectory?

Value function

How much return can we expect from the current state?

$$A(s_t, a_t) = r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t)$$

QUIZ

A2C
IMPLEMENTATION

THANK YOU

SEE YOU NEXT WEEK!