Reinforcement Learning

Introduction

Jakub Chojnacki

About the Author

Who Am I

Admin

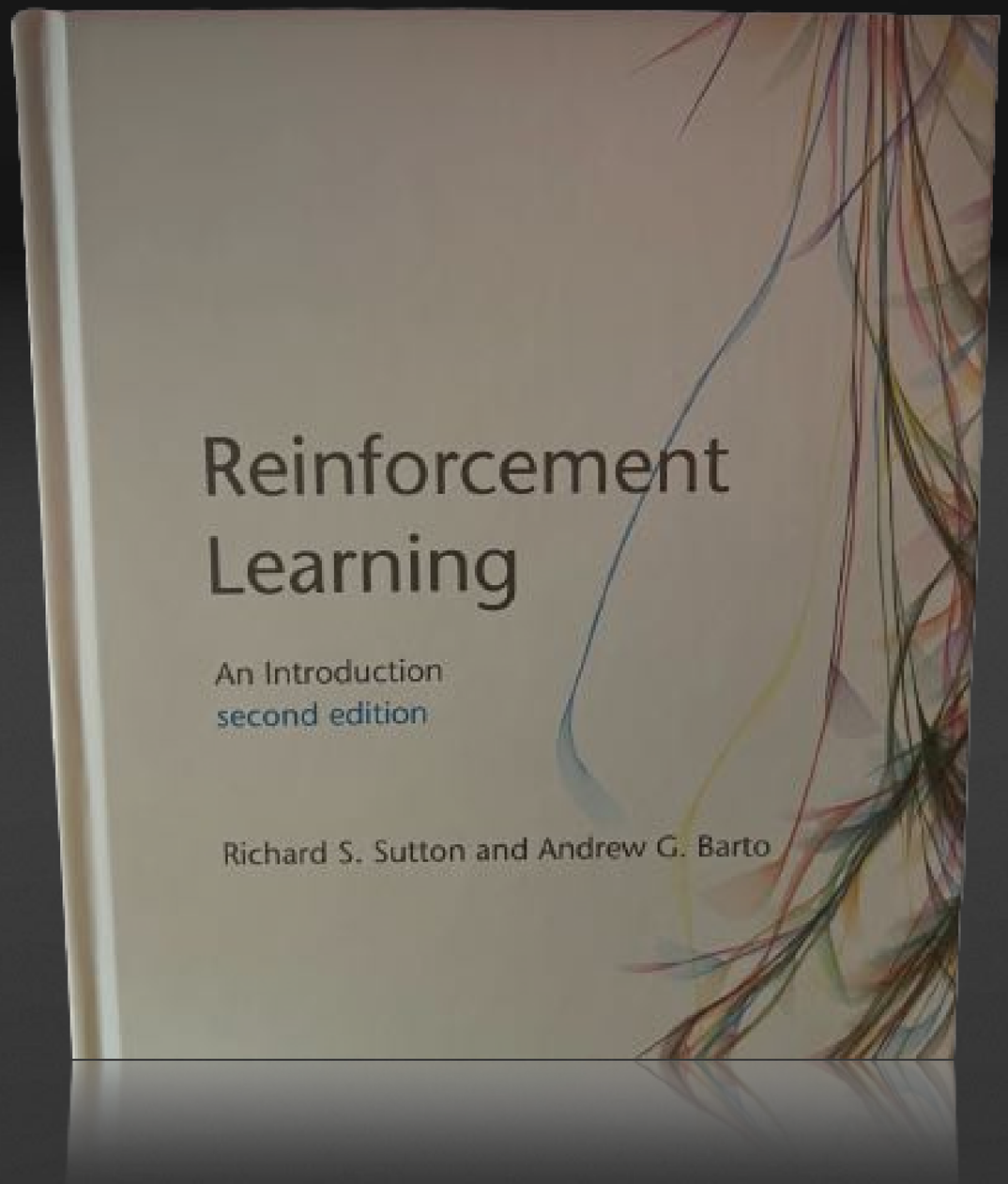
What & When

1. **Introduction** - **[Office: 2.11.2022] & [Online: 3.11.2022] @ 12.00 pm - 1.00 pm**
    1. Deep dive into RL systems
    2. Intuition building
2. **Tabular methods** - **[Office: 9.11.2022] & [Online: 10.11.2022] @ 12.00 pm - 1.00 pm**
    1. Dynamic programming
    2. Monte Carlo methods
    3. Temporal difference methods
3. **Approximate methods** - **[Office: 16.11.2022] & [Online: 17.11.2022] @ 12.00 pm - 1.00 pm**
    1. Intro to Deep Reinforcement Learning
    2. Value function approximation
    3. Policy gradient methods
4. **Modern Deep Reinforcement Learning** - **[Office: 23.11.2022] & [Online: 24.11.2022] @ 12.00 pm - 1.00 pm**
    1. Grokking modern algorithms - { DQN, PPO, DDPG, TD3, SAC, *TQC, *QR-DQN }
    2. Tips and tricks
5. **Coding session** - **[Office: 30.11.2022] & [Online: 1.12.2022] @ 12.00 pm - 1.00 pm**
    1. Learn how to code RL environment using OpenAI API
    2. Choose algorithm, solve the problem and evaluate your agent

Materials

What was used

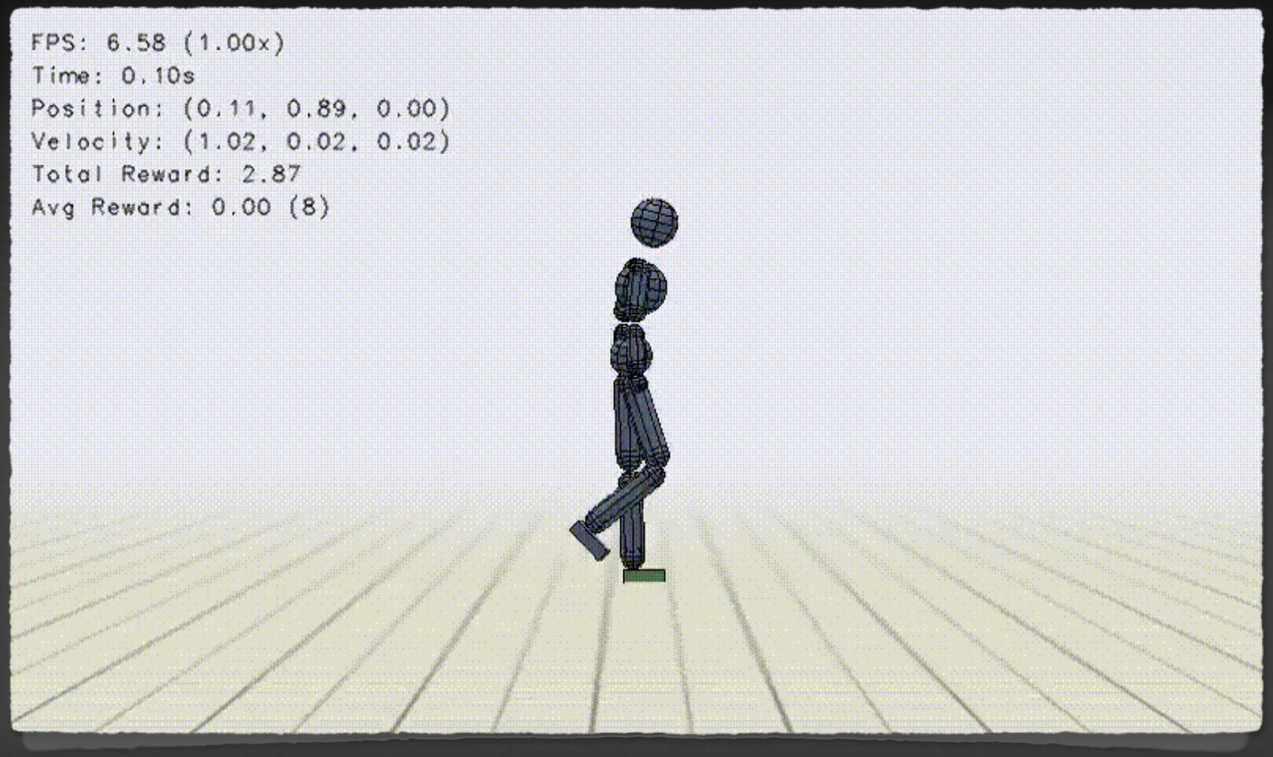
1. Videos: [David Silver] “Introduction to Reinforcement Learning 2015”
2. Book: [Sutton & Barto] “Reinforcement Learning An Introduction 2nd edition”
3. Book: [Miguel Morales] “Grokking Deep Reinforcement Learning”



My intention

Ultimate goal

1. Build **intuition** with respect to Reinforcement Learning field
2. Show how to **think** about the potential projects
3. **Learn** throughout teaching others



Agenda
Topics covered

1. How Do Machines Learn
2. Deep Dive
3. Problem Structure
4. Markov Decision Process
5. Agent Design
6. Value Functions
7. Objective Function

How Do Machines Learn

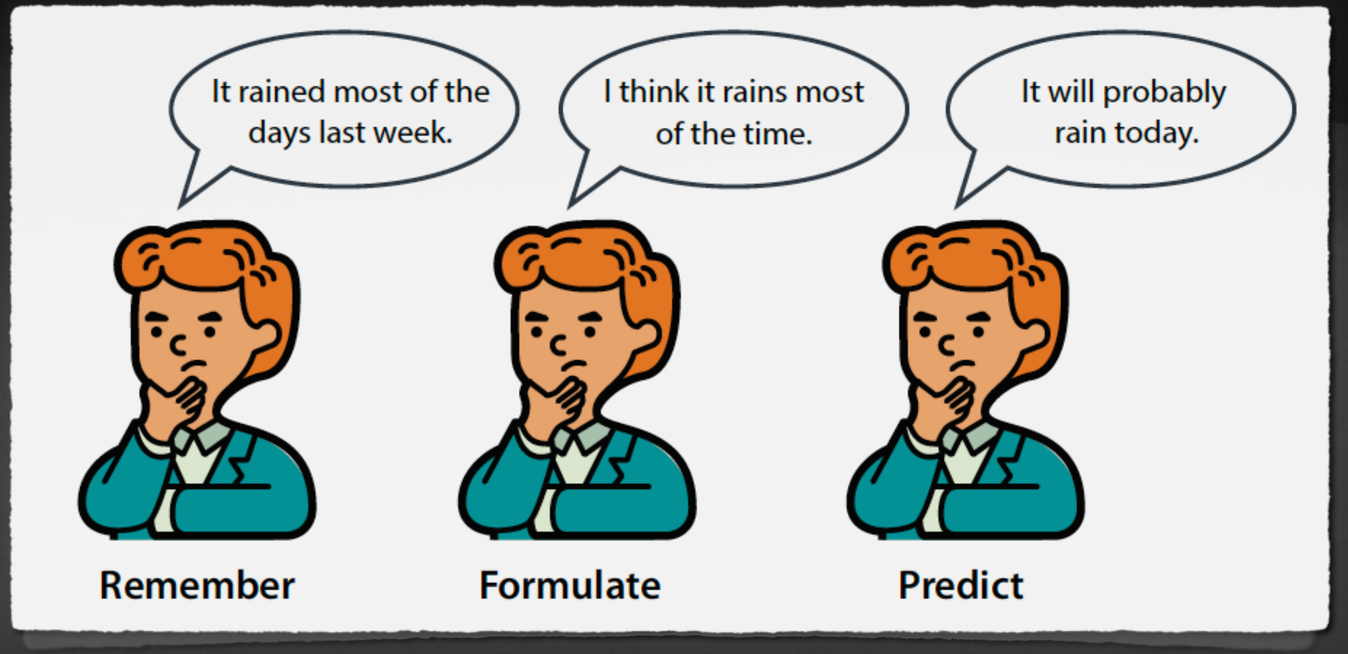
Remember, formulate and predict

Boston Dynamics, Atlas

Framework
Machine Learning Lingo

Given **data** as source of experience

1. **REMEMBER**
   - *Look at the previous data*
2. **FORMULATE**
   - *Build a model, or a rule, w.r.t this data*
3. **PREDICT**
   - *Use the model to make predictions about future data*



Data

Who has the data has the power

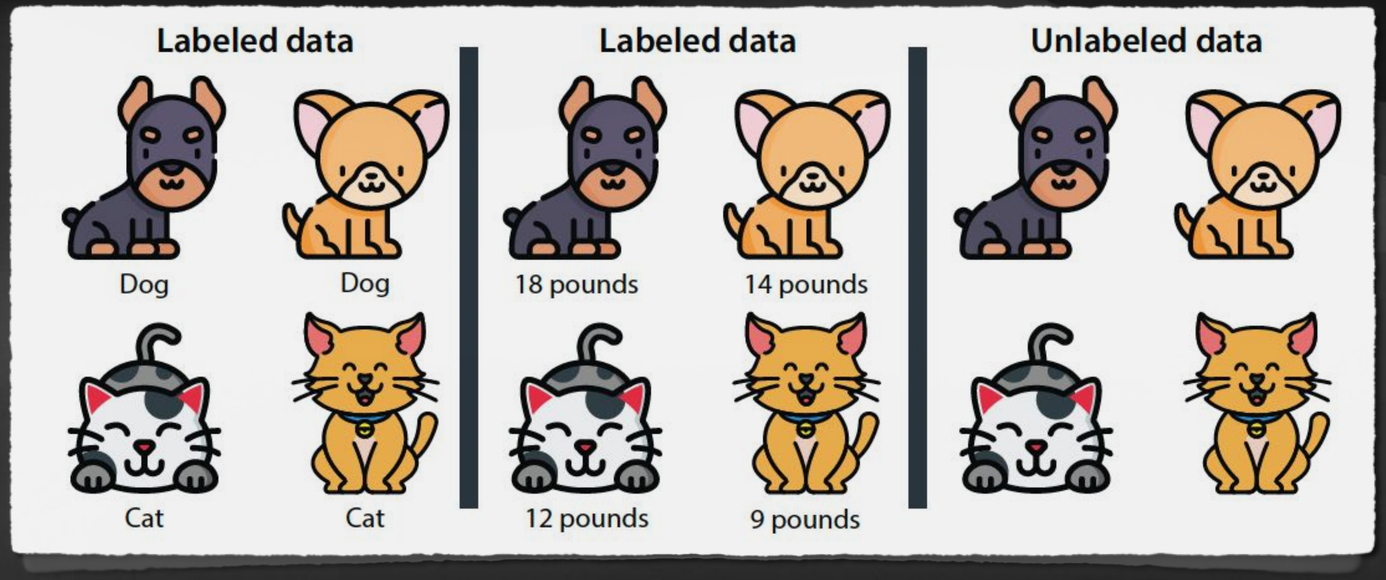
1. Labeled Data
    - Data that comes with a tag, or label
    - Label can be a type or a number
2. Unlabelled Data
    - Does not come with a tag, or label

Numerical data:

- data that used number
- [4], [2.35] or [-199]

Categorical data:

- data that uses categories, or states
- [male / female] or [cat / dog / bird]

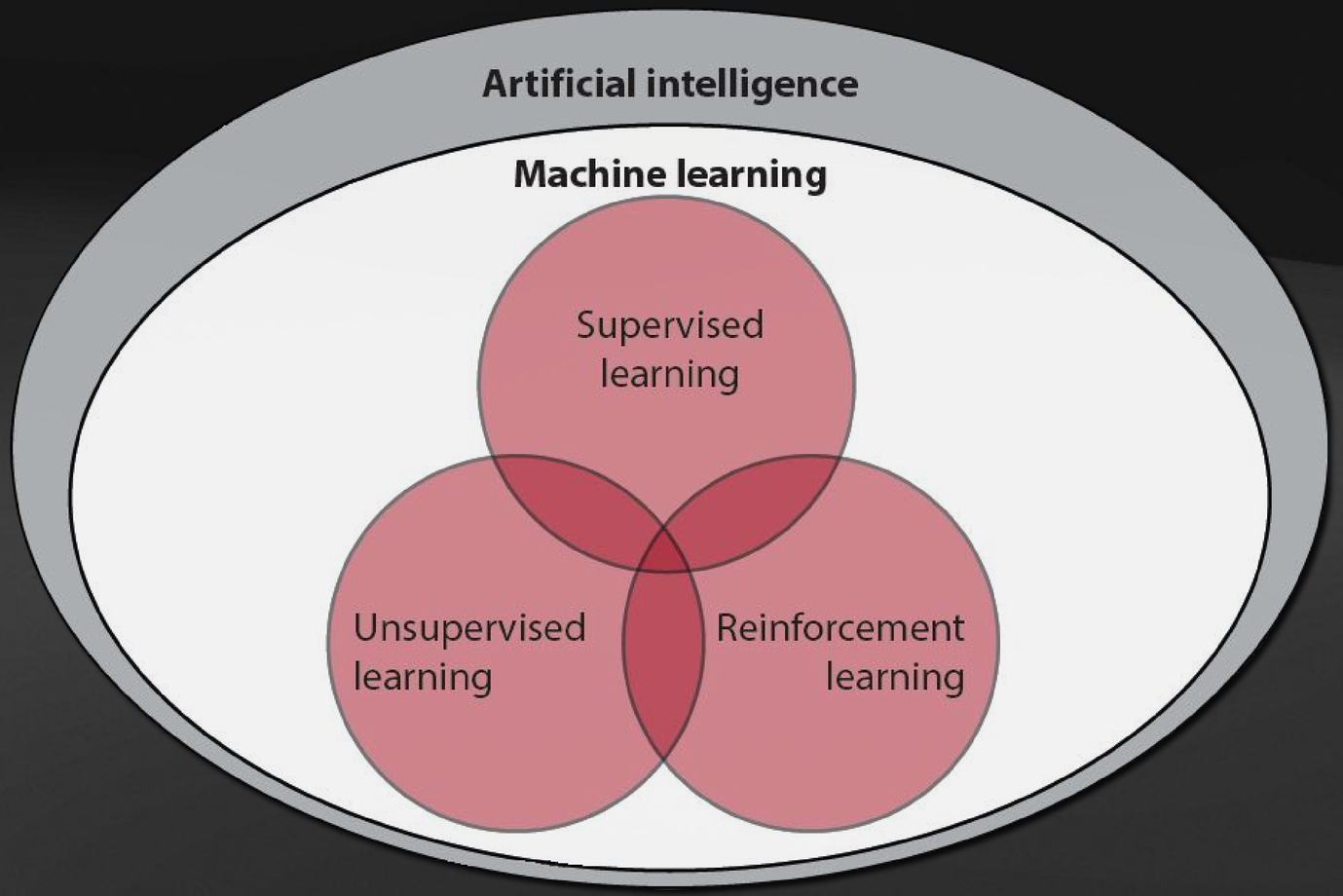


Machine Learning
Algorithms that remember

1. Machine learning is a subset of **artificial intelligence**
2. That gives computers the ability to **learn** without being explicitly programmed to do so

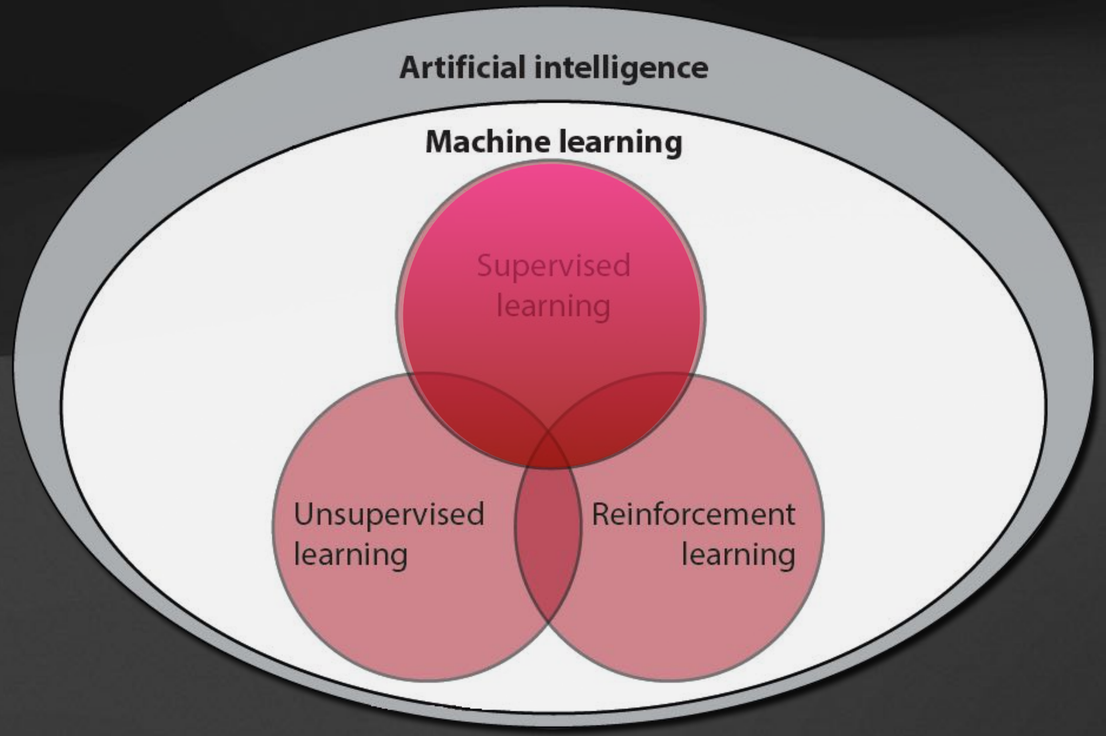
Examples of algorithms used in machine learning

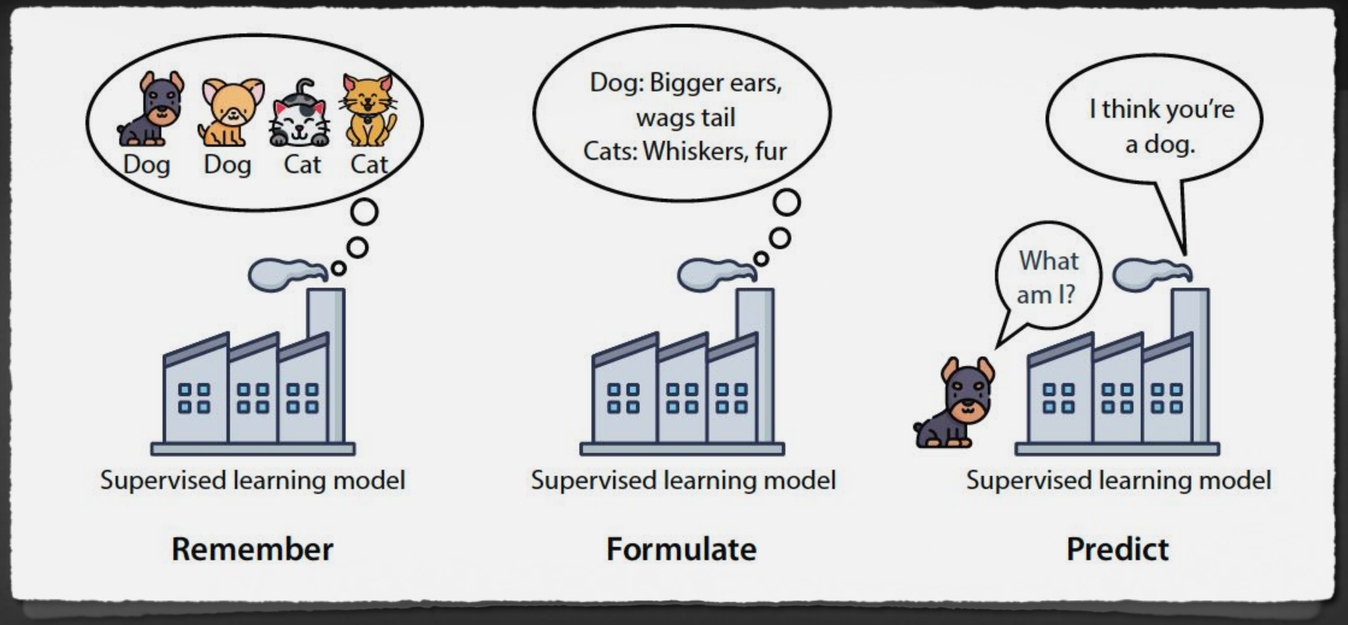
1. Linear regression
2. Logistic regression
3. Decision tree
4. SVM



Supervised Learning

The goal is to GENERALISE



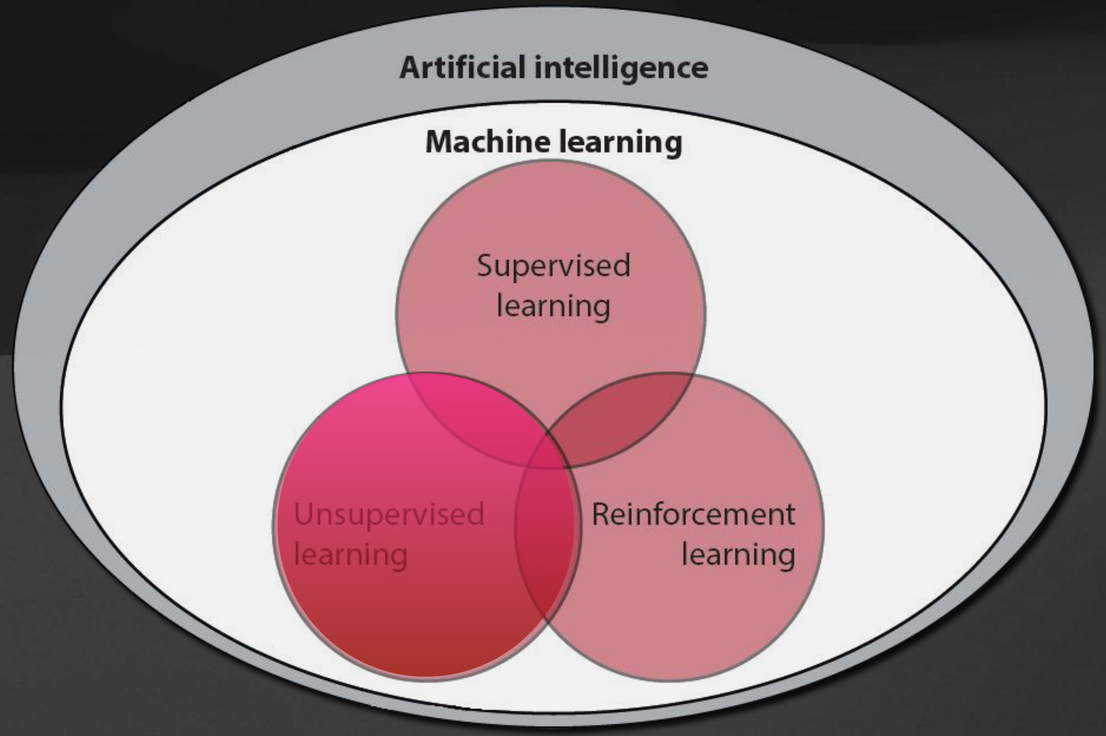


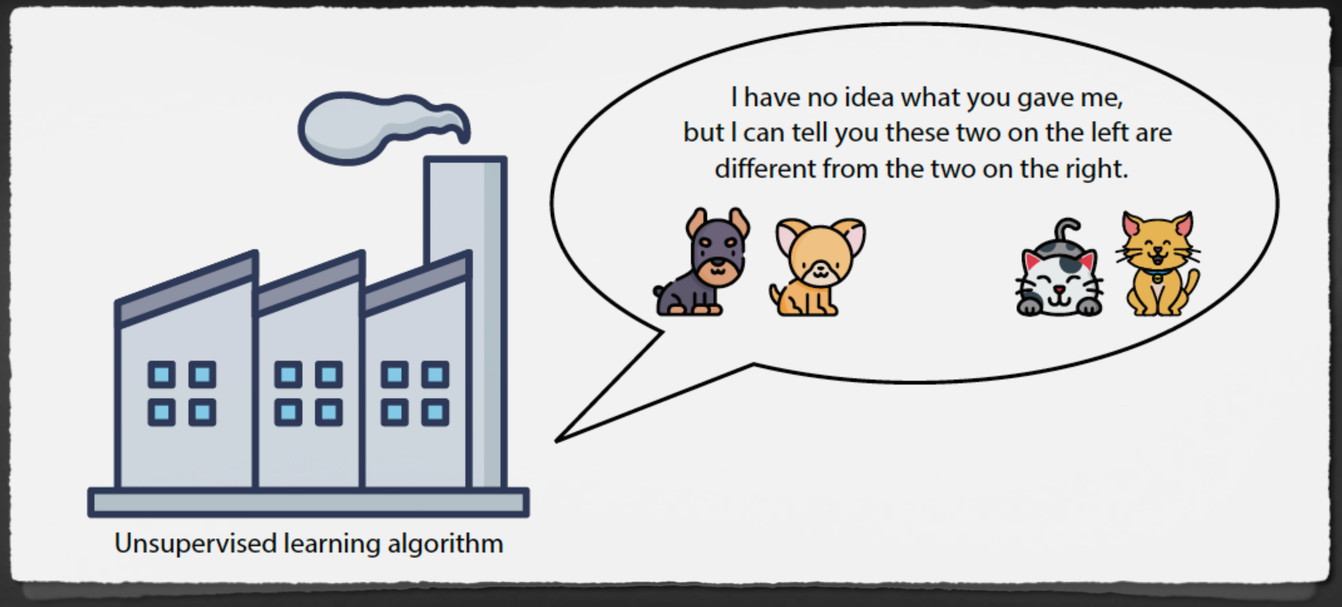
Project examples that use supervised learning

1. Detect surgical instruments and count them before surgery
2. Classify people based on whether they wear a face mask
3. Predict the price of houses given assumptions
4. Detect soldiers and track their position

Unsupervised Learning

The goal is to COMPRESS



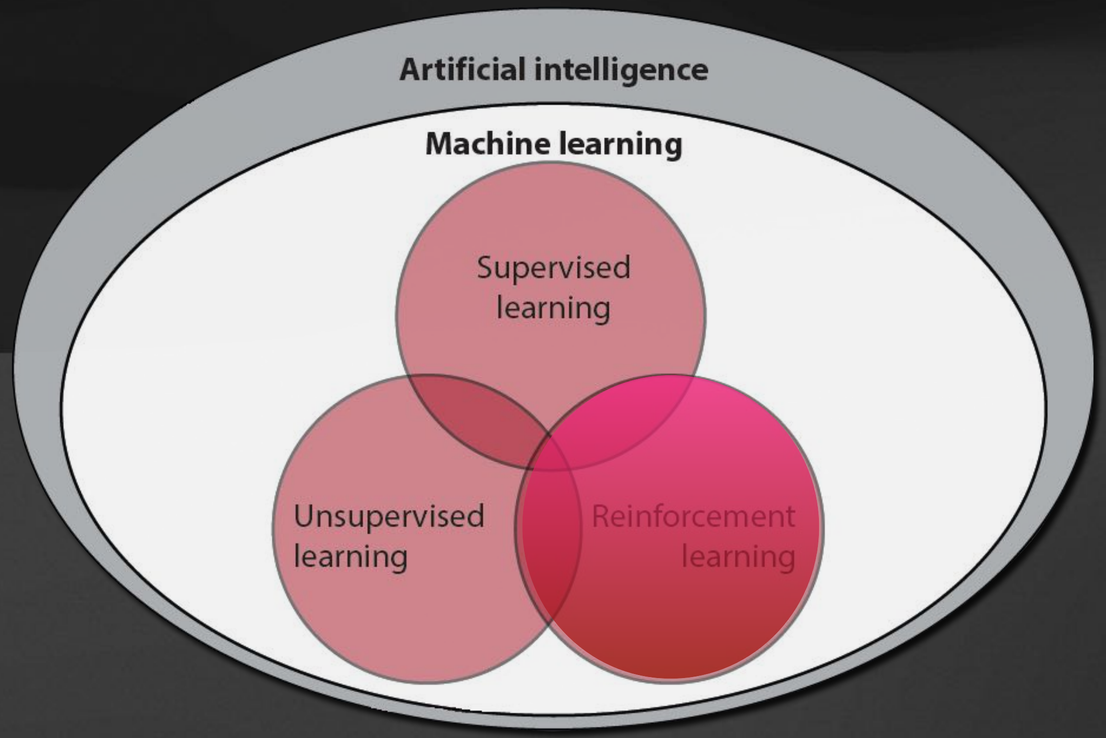


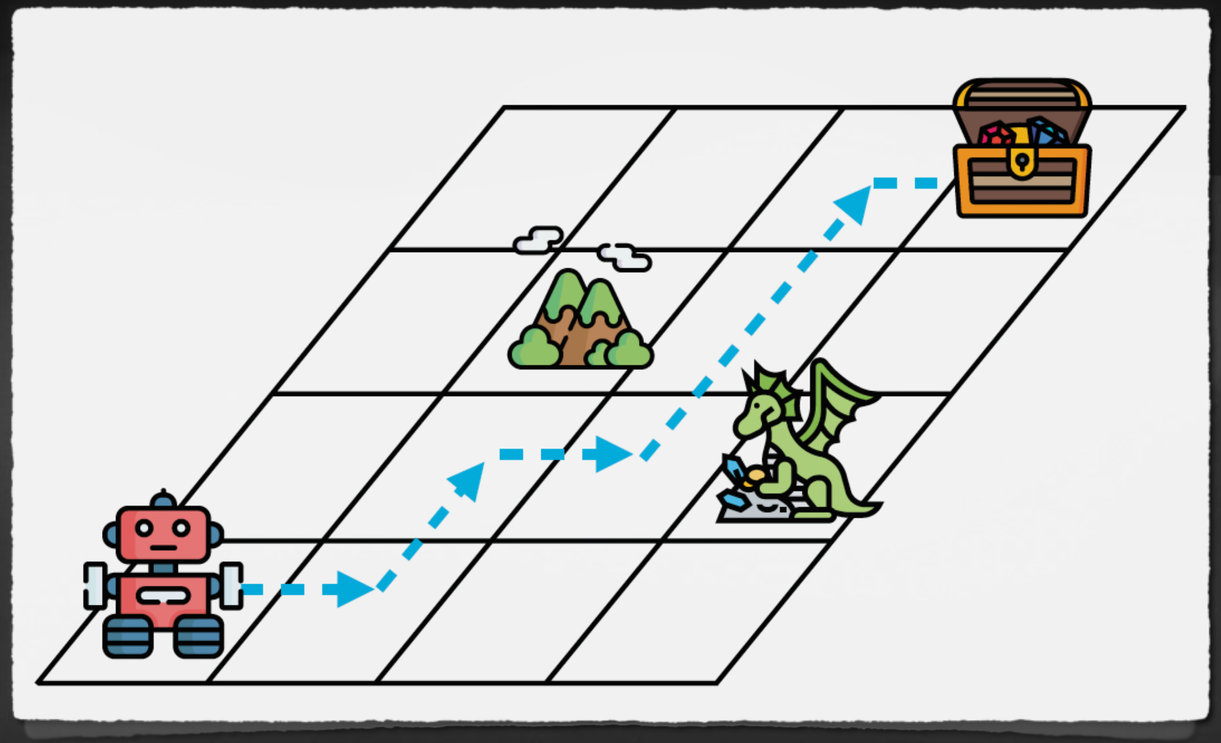
Project examples that use unsupervised learning

1. Detect outliers on data and visualise them
2. Generate images given text prompt
3. Study cancer data
4. Reduce the dimensionality of data

Reinforcement Learning

The goal is to ACT





Project examples that use reinforcement learning

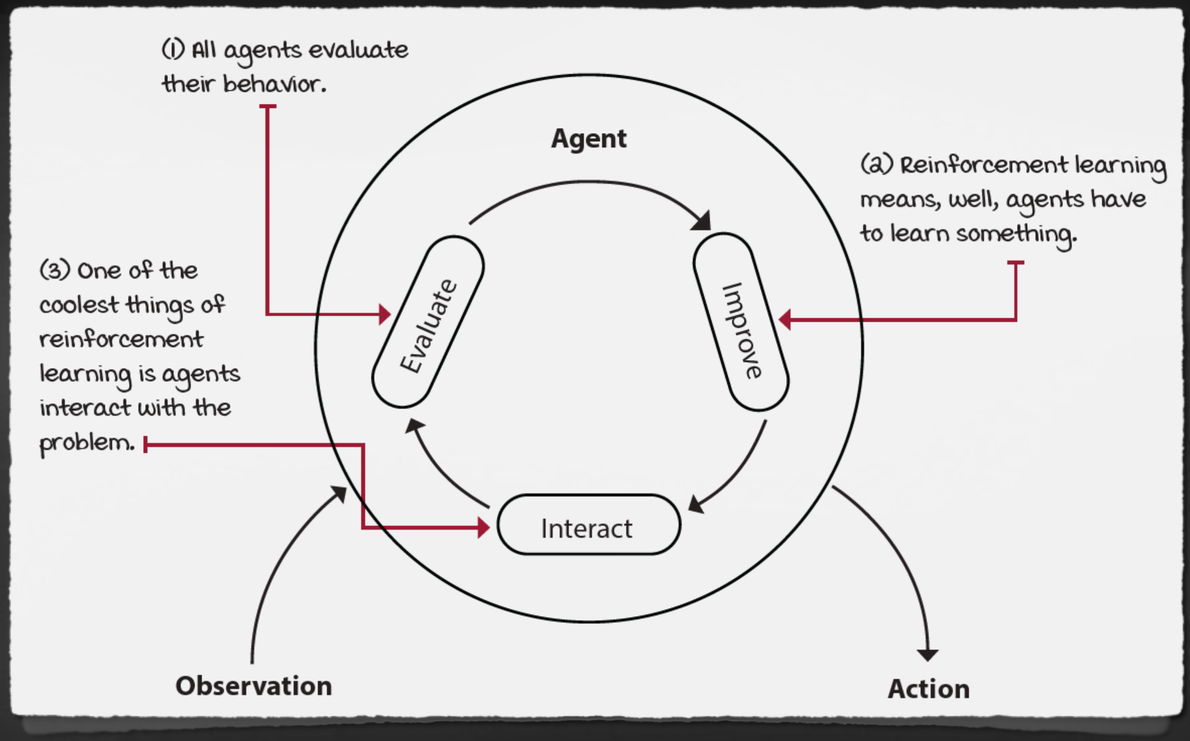
1. Achieve grandmaster level in StarCraft II and bully human players
2. Control a power plant
3. Manage investment portfolio
4. Make a humanoid robot walk

Reinforcement Learning Framework

RL Lingo

Given **agent** as an “actor” and **environment** as data source

1. **INTERACT**
   - *Agent interacts with the environment and gathers data*
2. **EVALUATE**
   - *Given agent’s interactions, evaluate it’s performance*
3. **IMPROVE**
   - *Improve agent’s behaviour based on the evaluation score*

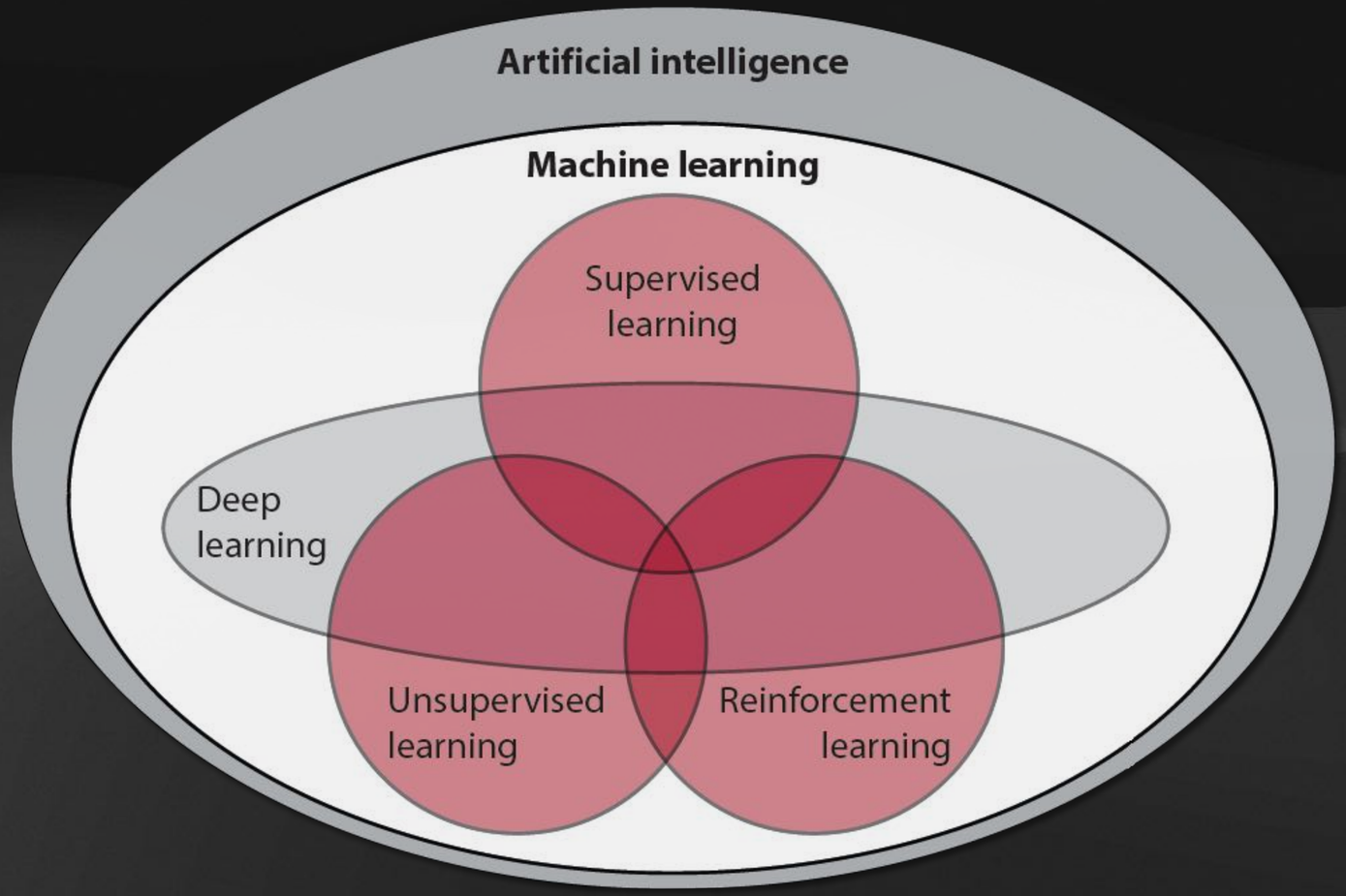


Deep Learning
Getting deeper

1. Deep learning is a subset of **machine learning**
2. That is concerned with replicating the human-brain processes and abilities to remember and generalise given data
3. Relies heavily on **neural networks** and it's evolutions

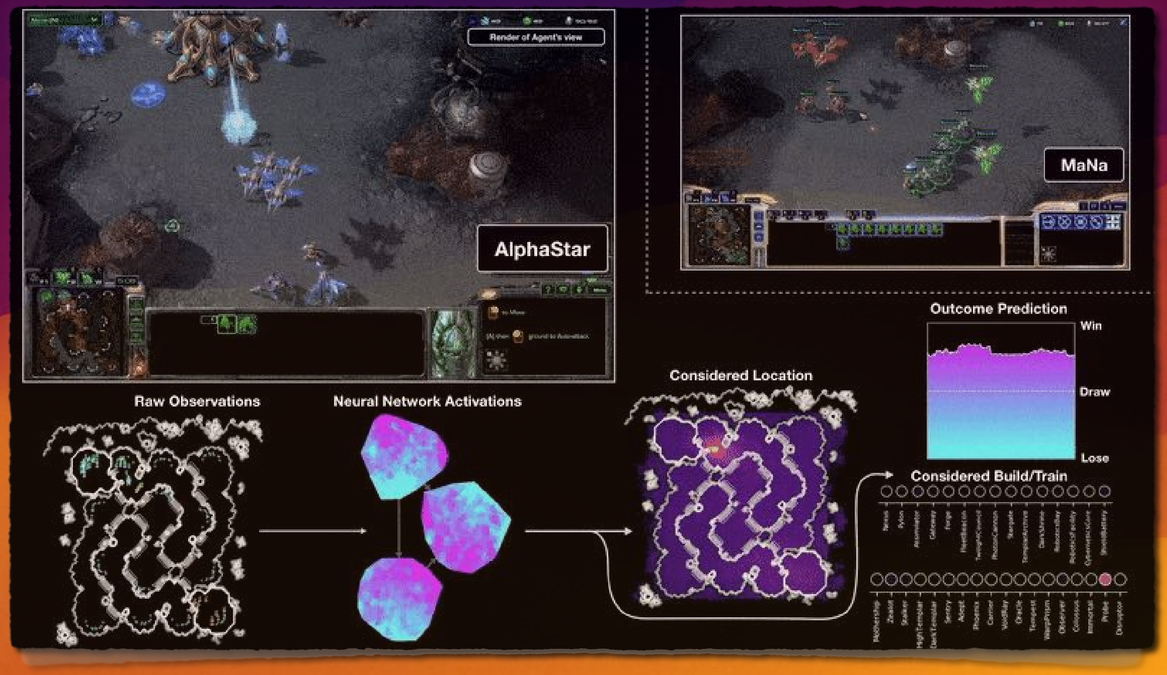
Examples of algorithms used in deep learning

1. CNN [Convolutional Neural Networks]
2. RNN [Recurrent Neural Networks]
3. LSTM [Long Short Term Memory]
4. GAN [Generative Adversarial Networks]



Deep Dive

Reinforcement Learning is about sequential decision making under uncertainty

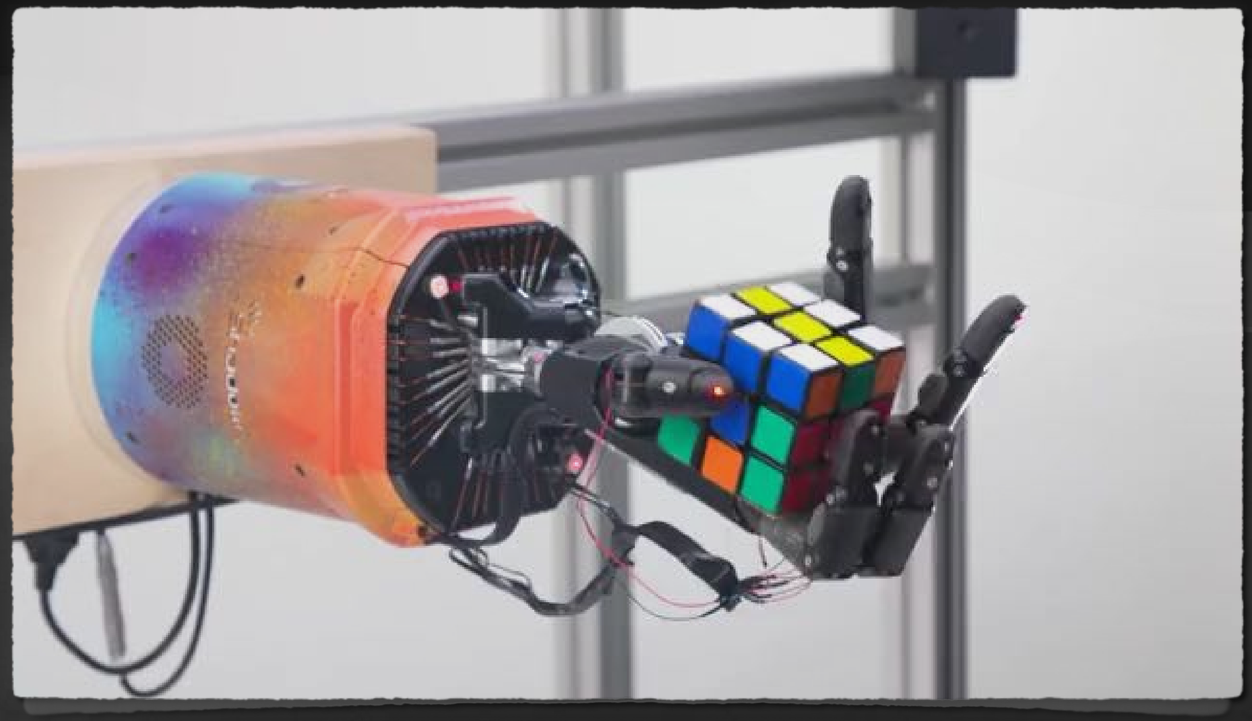


DeepMind, AlphaStar

RL Examples

Act in order to control the environment

1. Play Starcraft II game
2. Control a power plant
3. Control a quadcopter
4. Make a humanoid walk
5. Defeat world champion at Backgammon
6. Defeat stockfish at chess
7. Solve Rubik’s cube
8. Personalise feed on TikTok



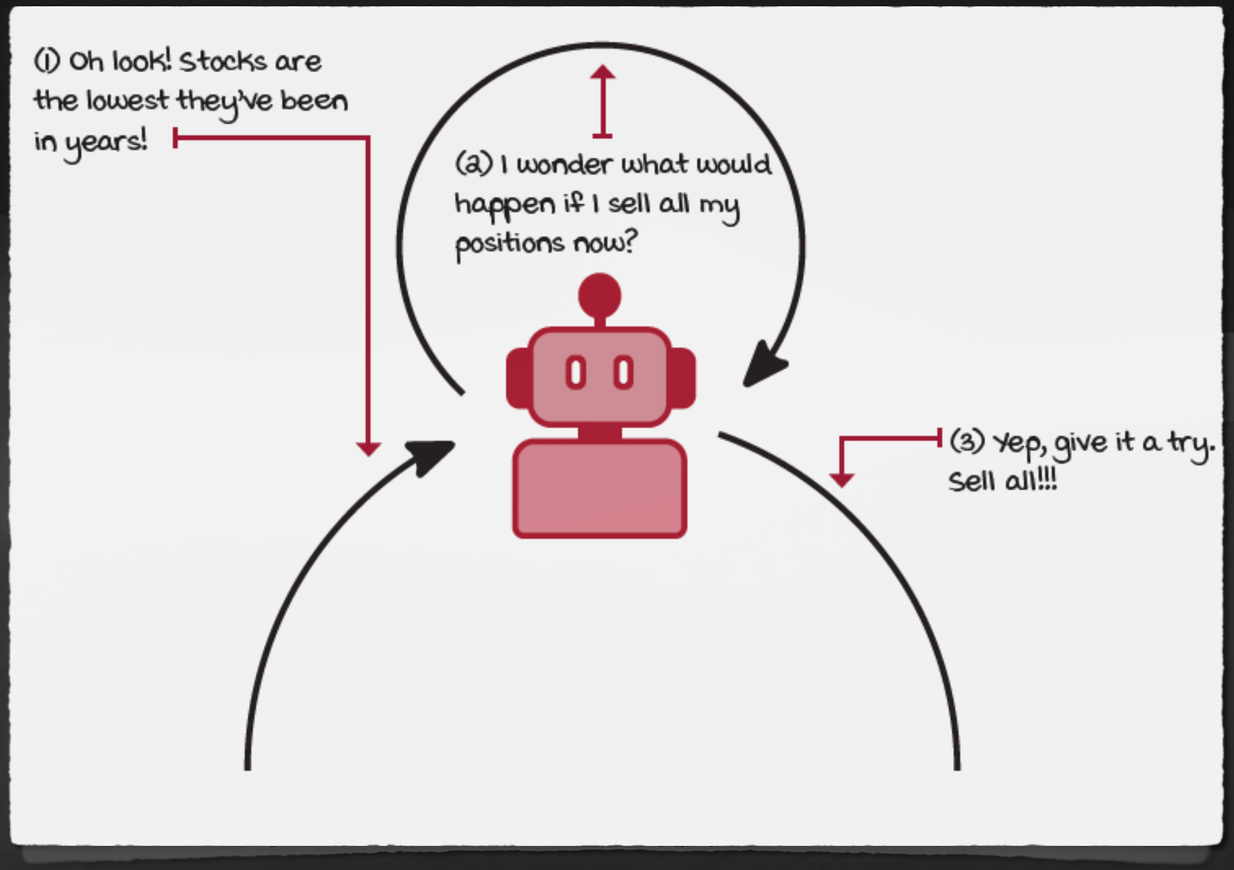
OpenAI Solves Rubik’s Cube

One sentence

To rule them all

1. Deep reinforcement learning (DRL) is a machine learning approach to artificial intelligence,
2. concerned with creating computer programs that can solve problems requiring intelligence,
3. the distinct property of DRL programs is learning through **trial and error** from **feedback**,
4. that's **simultaneously sequential**, **evaluative**, and **sampled** by leveraging powerful non-linear function approximation.

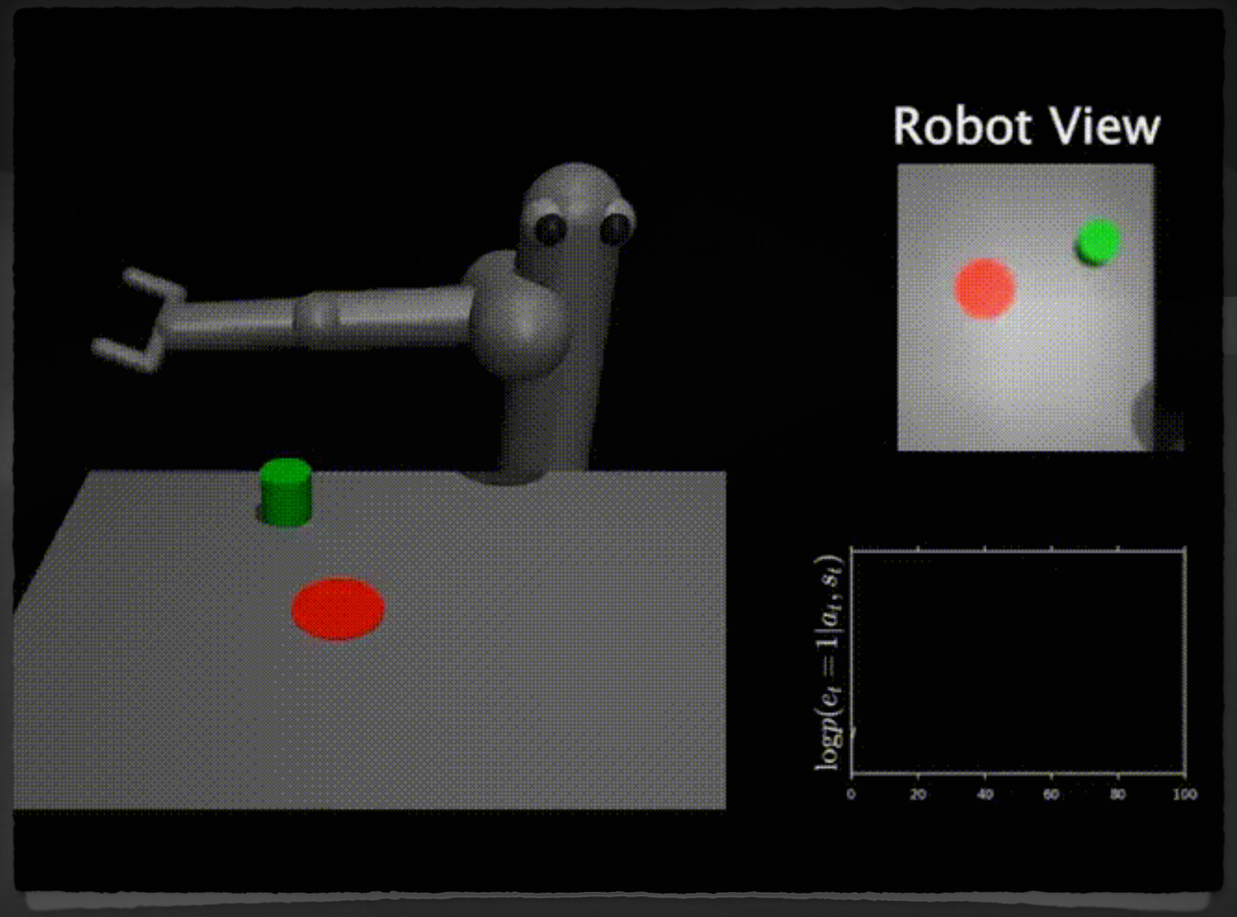
~ Miguel Morales, Grokking Deep Reinforcement Learning



Interact-Evaluate-Improve

One sentence
Unpack magic words

1. **trial and error** from **feedback**
   - The agent has **no access** to the **model** of environment and underlying **reward function**
   - The agent has to **try actions** in order to find its impact on environment and it's dynamics
   - The environment's **reward function** is used to **punish** or **reward** agent's interactions
2. **simultaneously sequential**,
   - Actions have **delayed** consequences
   - The data is **sequential** and **correlated**
3. **evaluative**, and **sampled,**
   - Balance **exploration** and **exploitation** of information
   - Resources are **limited**, thus the agent has to use **approximation** methods

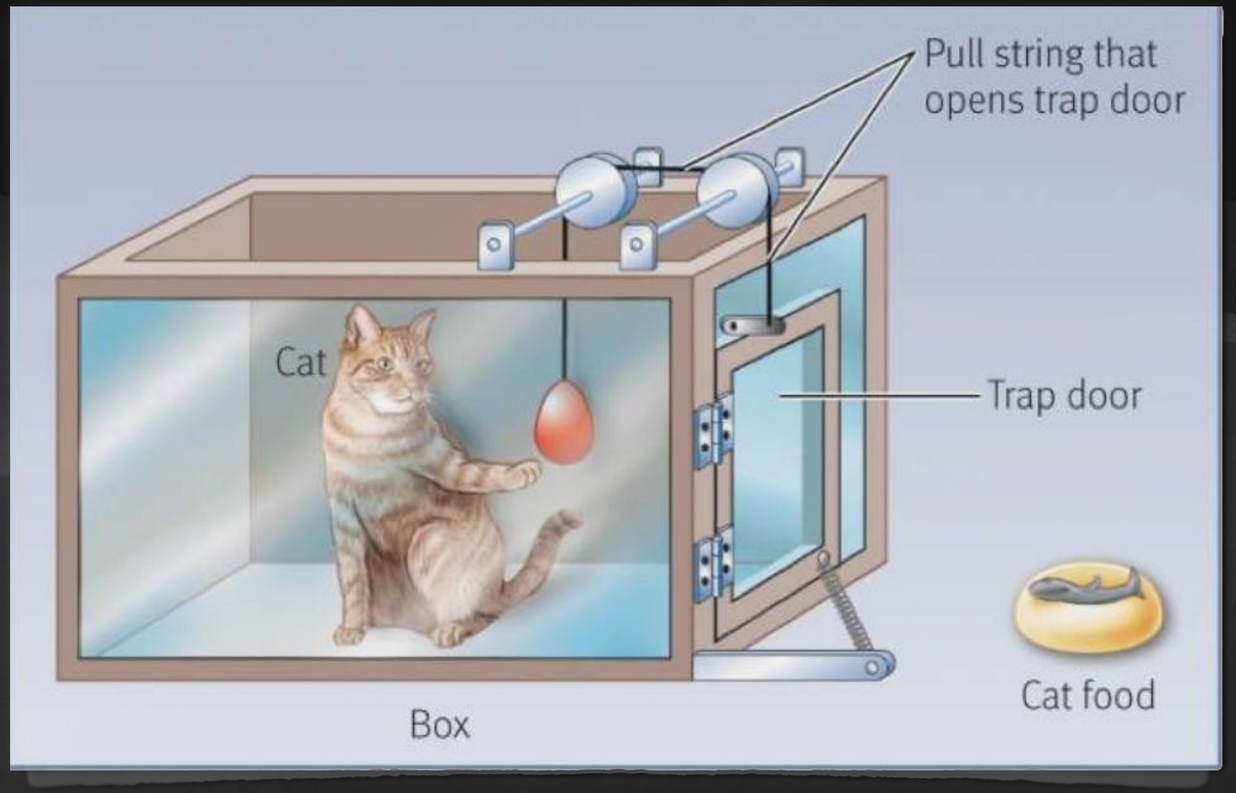


Institute of Robotics, UC Berkeley

Trial and error

Life is one big data stream

1. In this example the cat's brain is our **agent**
2. Everything else is considered an **environment**
3. The **goal** of our agent is to eat delicious **fish**
4. The agent has **no knowledge** of the underlying structure of environment and it's internal state
5. Agent's role is to learn the combinations of **actions** given **observations**
6. In such way to influence the environment to transmit the final reward **[open trap door]**

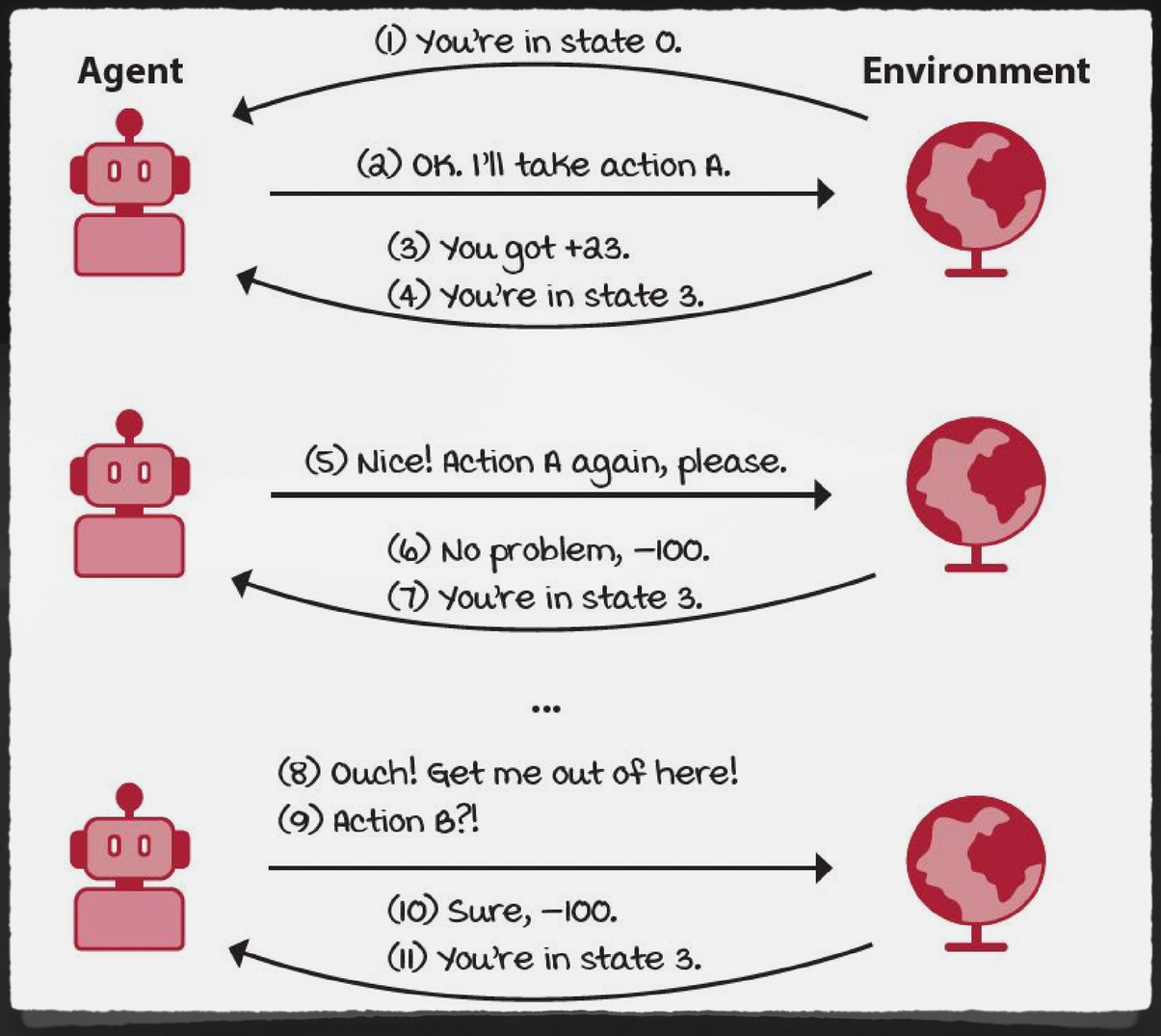


Law of Effect, Edward Throndlike 1991

Sequential feedback

Temporal credit assignment

1. The temporal credit assignment problem is the challenge of determining which **state** and/or **action** is responsible for reward,
2. **Actions** has delayed consequences,
3. Sacrificing queen in chess might look like a bad move in **short term**
4. But in **long term** might win you the game



Evaluative feedback

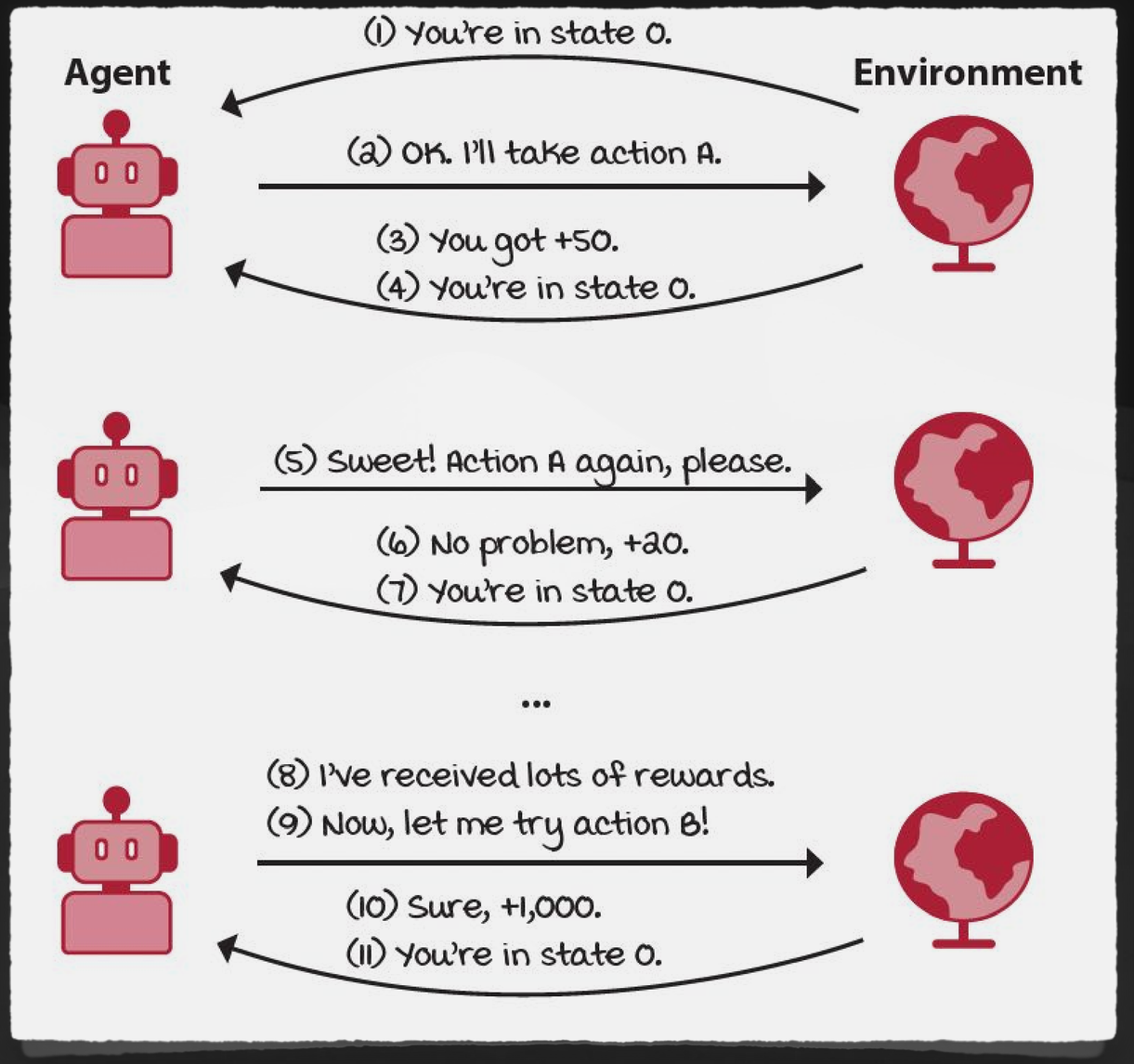
Exploration vs Exploitation trade-off

1. The agent must be able to balance the **gathering [exploration]** of information with the **utilisation [exploitation]** of current information,

2. The **rewards** assigned to you'r **actions** are initially unknown,

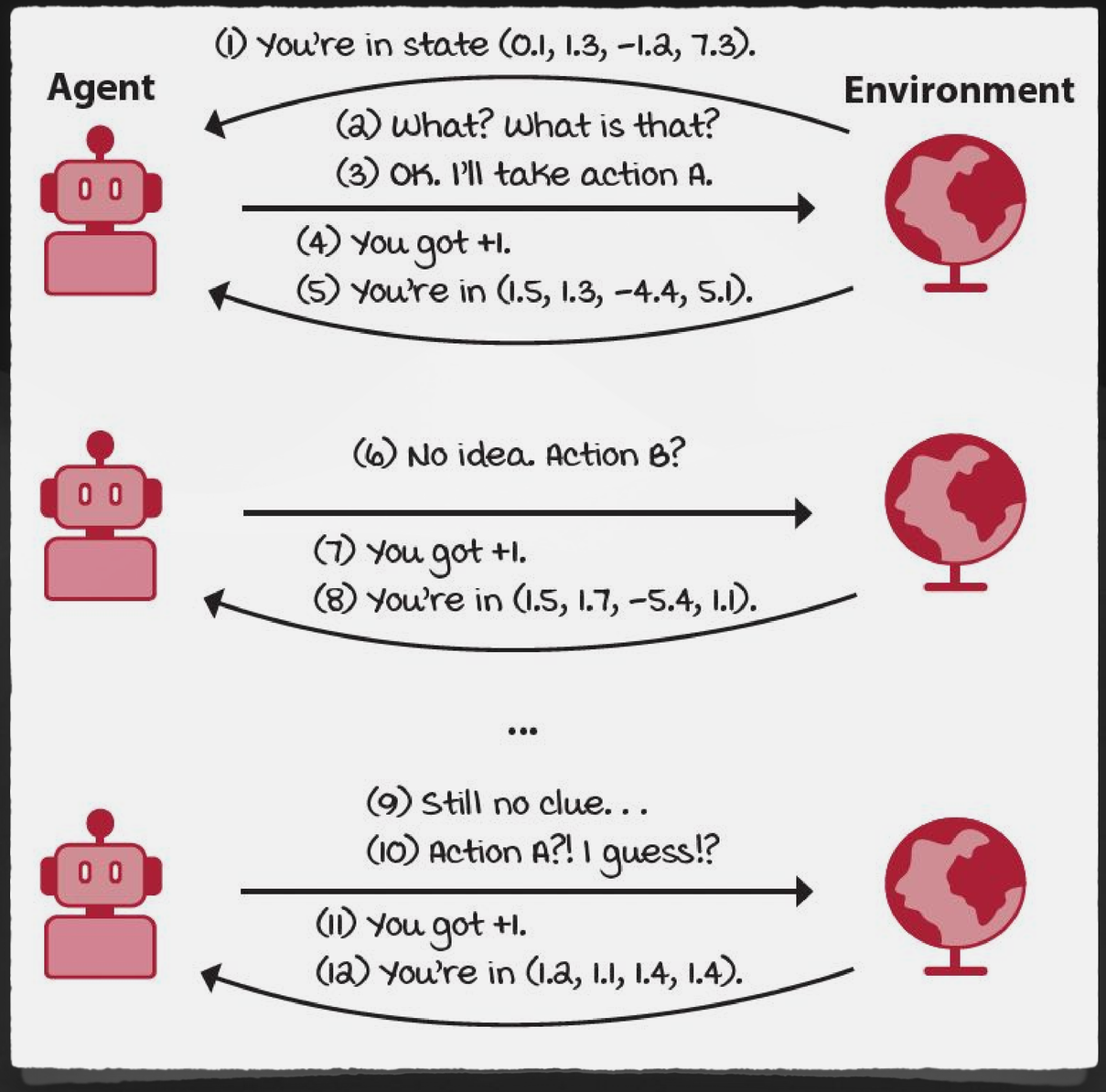
3. **Exploration** allows our agent to learn something new about the environment,

4. **Exploitation** allows our agent to learn from the gathered information,



Sampled feedback
Generalisation problem

1. The reward received by the agent is merely a **sample**
2. The agent has no access to the **reward function**
3. In real world **state** and **action spaces** are commonly large or even infinite
4. Trying to learn from sparse and week feedback becomes a challenge
5. Therefore, the agent has to be designed in such to be able to learn from **sampled feedback** and **generalise [function approximation]**



Let’s see how difficult it is to interpret feedback

A Chinese farmer gets a horse, which soon runs away.
A neighbour says, “So, sad. That’s bad news.”
The farmer replies, “Good news, bad news, who can say?”

The horse comes back and brings another horse with him.
The neighbour says, “How lucky. That’s good news.”
The farmer replies, “Good news, bad news, who can say?”

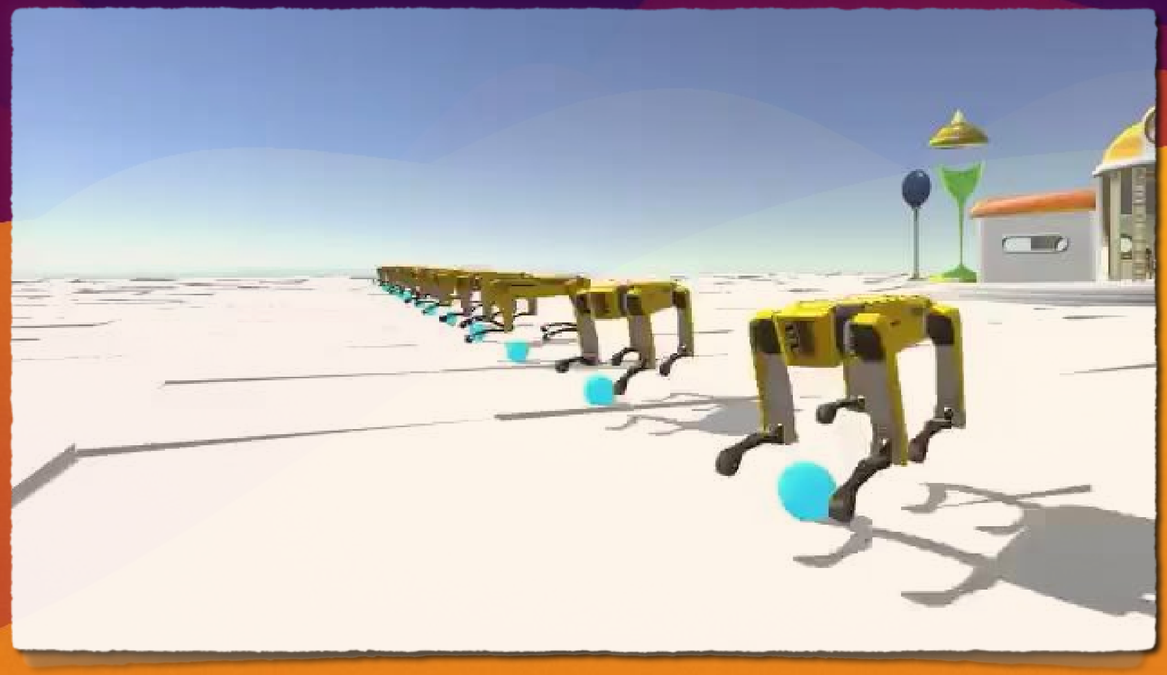
The farmer gives the second horse to his son, who rides it, then is thrown and badly breaks his leg.
The neighbour says, “So sorry for your son. This is definitely bad news.”
The farmer replies, “Good news, bad news, who can say?”

In a week or so, the emperor’s men come and take every healthy young man to fight in a war.
The farmer’s son is spared.

So, good news or bad news? Who can say?

Problem Structure

The goal of reinforcement learning agent is to act

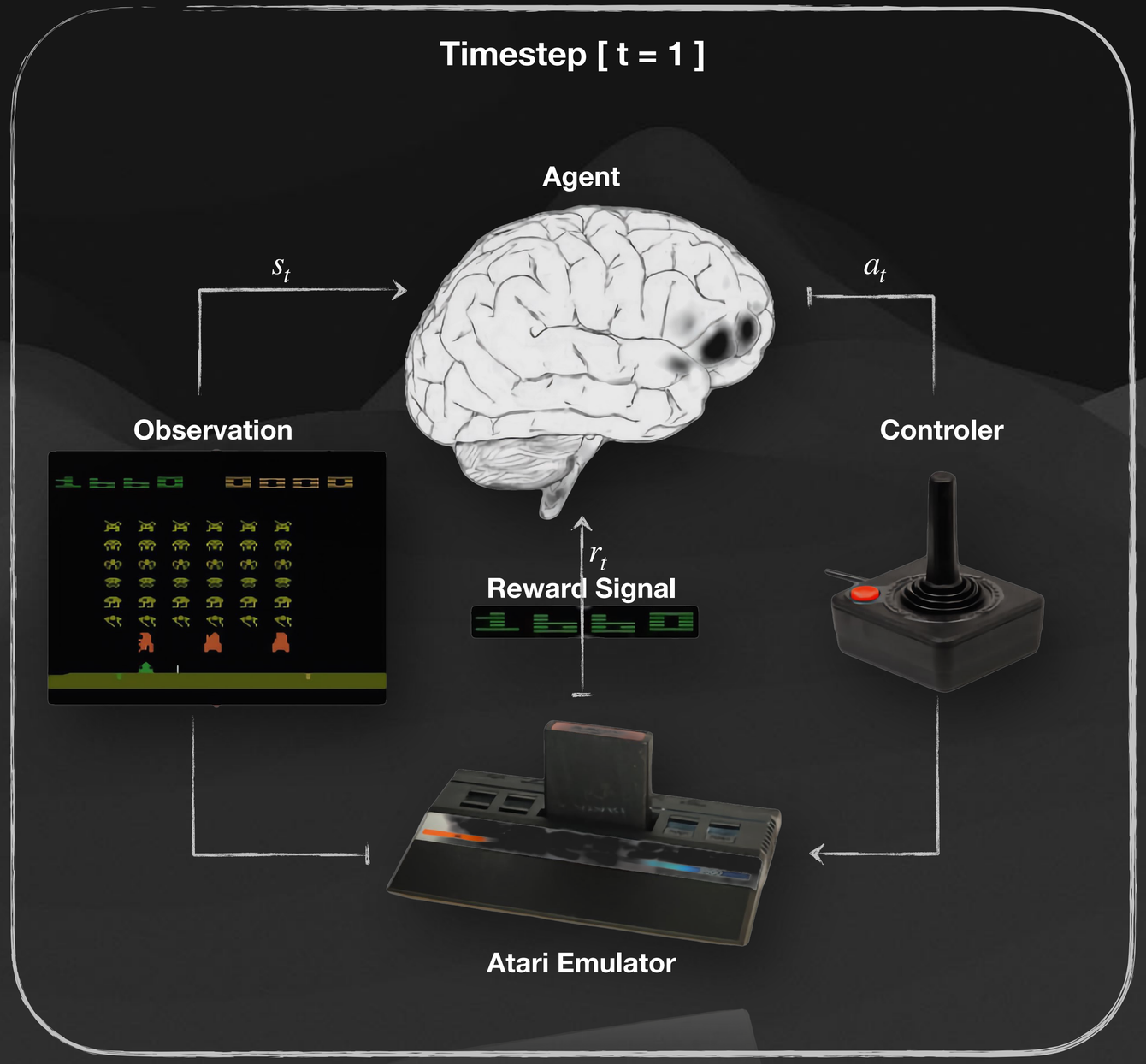


Boston Dynamics, Quadrupedal Robot Sim Training

Agent and Environment

Observe and act

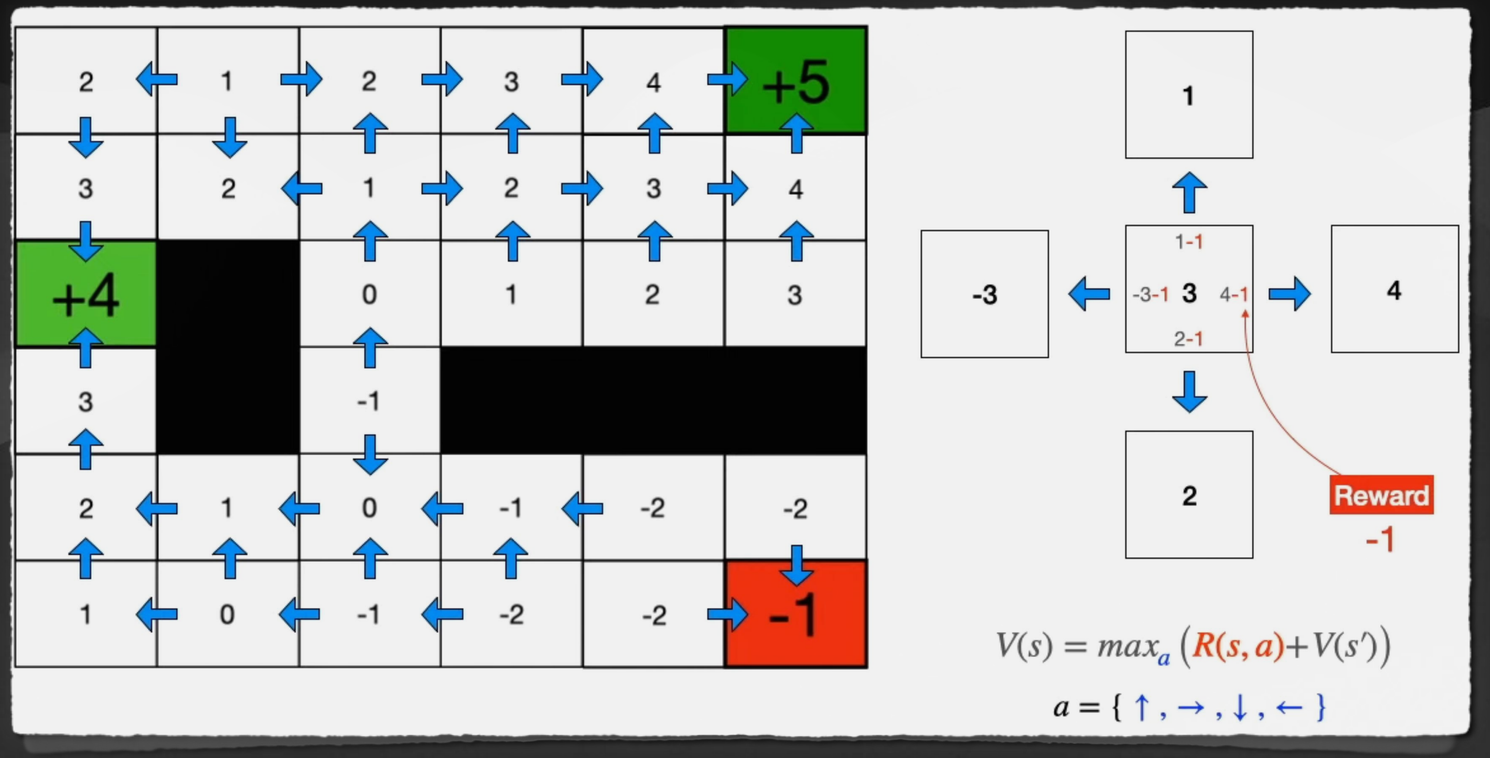
1. An **agent** is a **solution** to a problem,
2. An **environment** is a **representation** of a problem
3. An **agent** is a decision maker **only and nothing else,**
4. That means if you’r training a robot to pick up objects, the robot arm **is not** part of the agent,
5. Only the **code** that makes **decision** is referred to as **agent**.



Reward and Return

How well agent is doing

1. Reward [r_t] refers to a **scalar** feedback signal per timestep
2. Return [G_t] refers to the **total discounted reward** starting from timestep t
3. Agent role is to **maximise return**



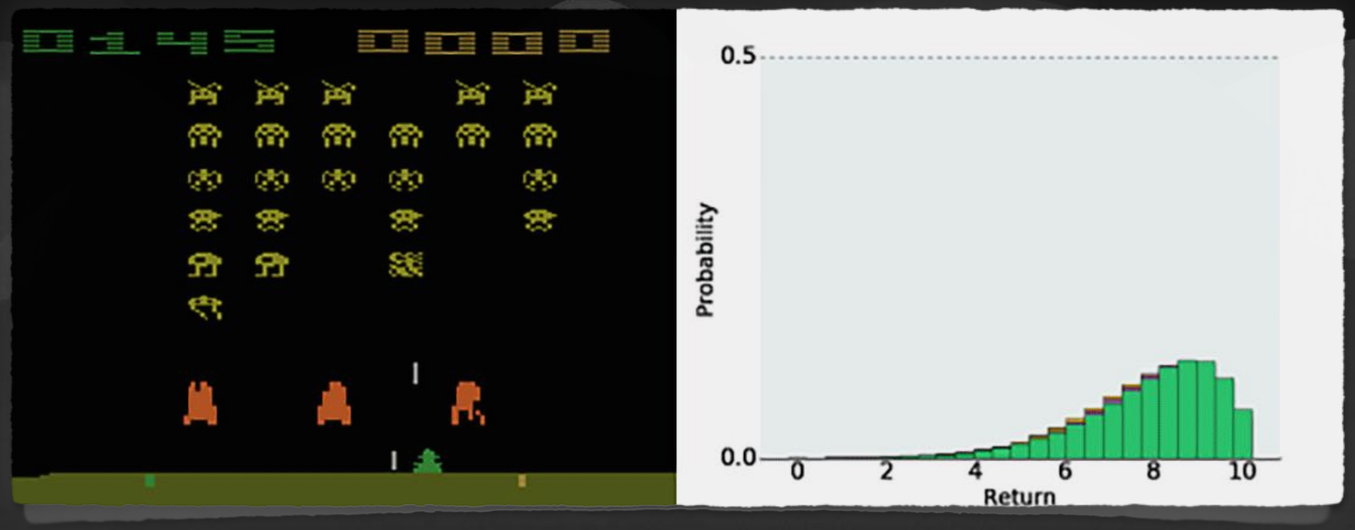
Return Definition

Maths and intuition

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{k-1} R_{t+k}$$

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

where $\gamma \in [0, 1]$



1. The return [G_t] is the total discounted reward from timestep t
2. The **discount factor** [γ] is the present value of future rewards
    - γ close to 0 leads to **myopic** evaluation
    - γ close to 1 leads to **farsighted** evaluation
3. Intuition behind discounting
    - Future rewards are uncertain
    - 10 dollars now is worth more than 10 dollars in 2 years

Reward Examples

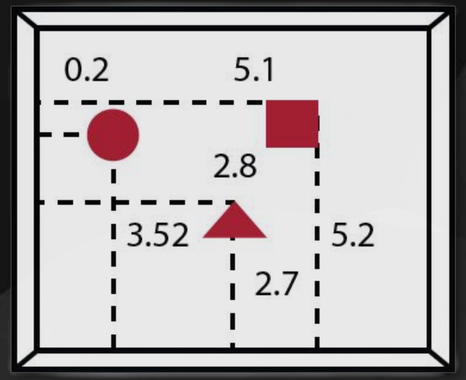
Learning signal per timestep

1. Defeat the world champion at Backgammon
   - Positive reward [+ 1] for winning
   - Negative reward [- 1] for loosing
2. Control a power plant
   - Positive reward [+ %power produced] for producing power
   - No reward [+ 0] for standby
   - Negative reward [- %power exceeded] for exceeding safety thresholds
3. Make a humanoid robot walk
   - Positive reward [+ distance traveled in meters] for forward motion
   - Negative reward [- 100] for falling over

States and Observations

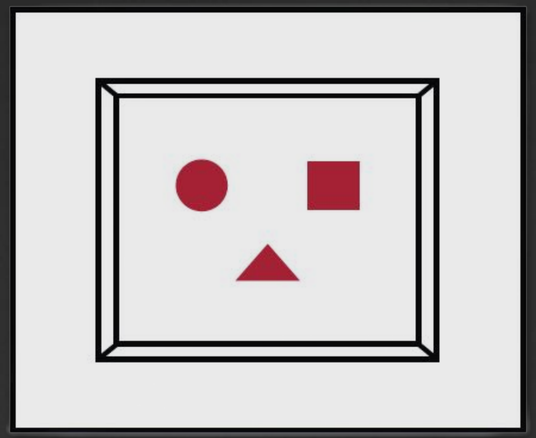
Quantified unit of information

1. States are the **perfect** and **complete** information related to the task at hand
   - “environment internal description”



true location and dynamics

2. Observations are the information the **agent receives**, this could be **noisy** or **incomplete information**
   - “information derived from state”



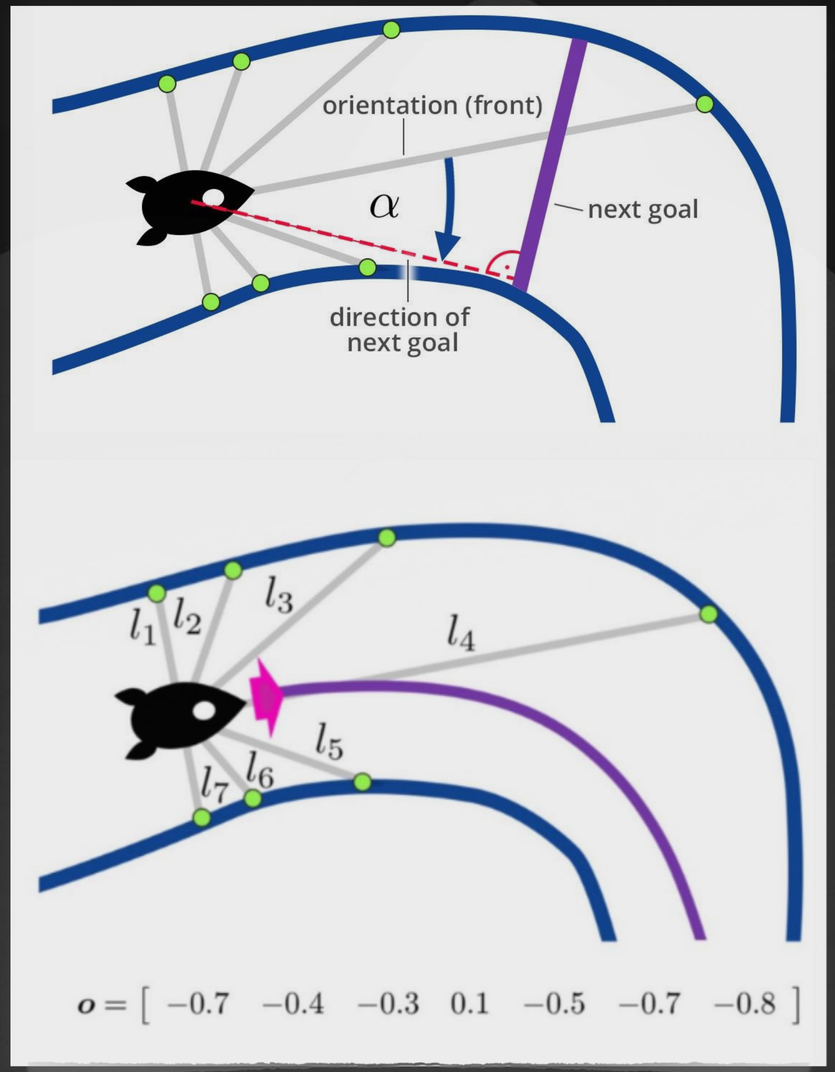
just an image

3. In most cases agents have to deal with incomplete and noisy informations - **observations**

Observation Examples

Data flow for the agent

1. **Discrete** observation space
   - 3x3 grid world position
   - Chess board positions
2. **Continuous** observation space
   - Helicopter angular position, velocity and remaining fuel
   - Robotic arm joint positions and effector orientation
   - Atari 210X160@60Hz 3 channel video
   - Power plant energy levels and demand



RocketMeister Environment

Markov Decision Process

The future is independent of the past given the present

DeepMind, Agent57

~ David Silver

Markov Decision Process

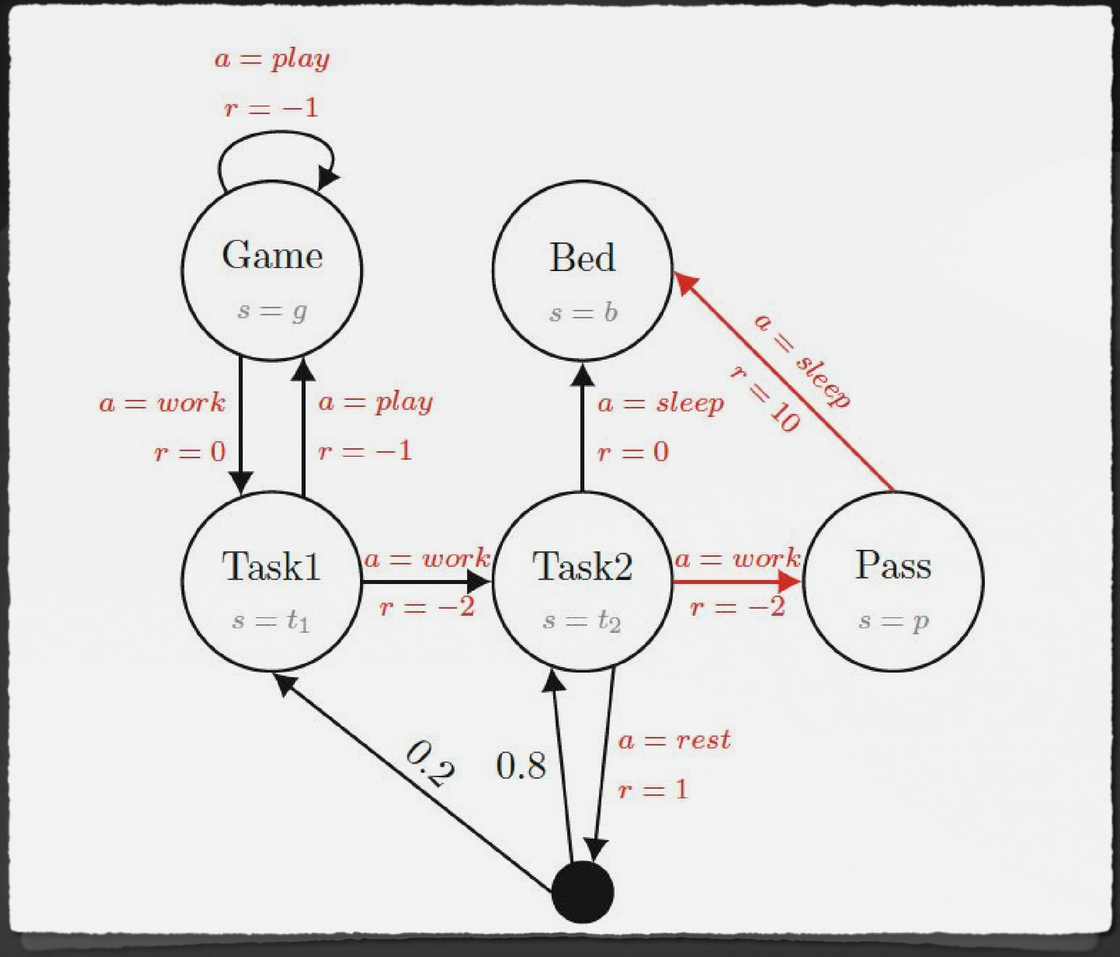
Formal description of environment

1. Markov decision process **[MDP]** is the description of problem to be solved

2. This formalism describes fully observable environment

   - We know the transition dynamics matrix $P^a_{ss'}$
   - The current state completely characterises the process
   - Agent has access to environment's state

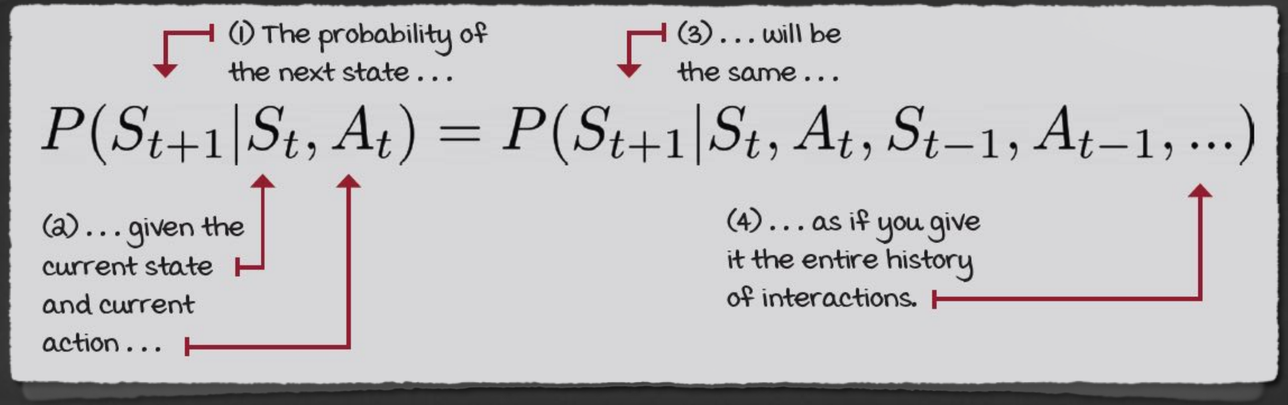
3. It allows us to model virtually any complex sequential decision making problem in a way that RL agents can interact with and learn to solve **solely** through **experience**



Markov Property

The memoryless property of MDPs

1. **“The future is independent of the past given the present”**
   - The state captures all relevant information from the history
   - Once the state is known, the history may be thrown away
2. The probability of the next state, given current state and action, is independent of the history of interaction
   - Completely keeping the Markov assumption is impractical, perhaps impossible
   - State **[observation]** has to be designed in such way to include all important informations for the agent to make decisions



MDP Parameters

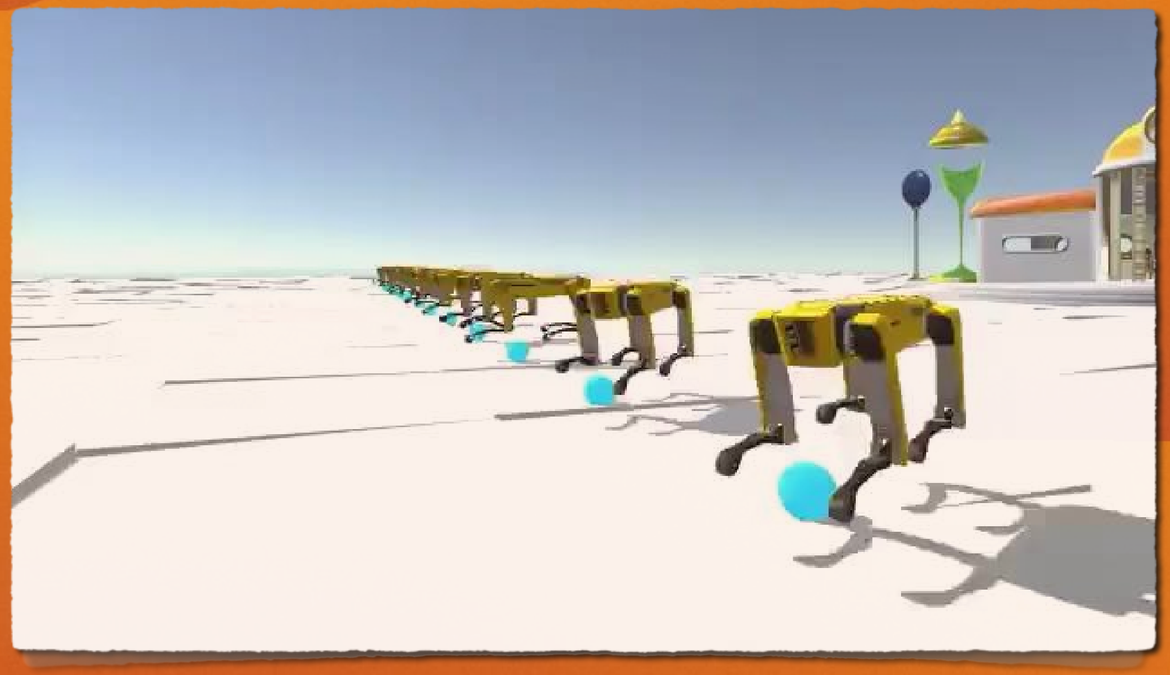
With respect to Markov Property

A *Markov Decision Process* is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}^{a}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}^{a}_{s} = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$
- γ is a discount factor $\gamma \in [0, 1]$.

Agent Design

What to look for



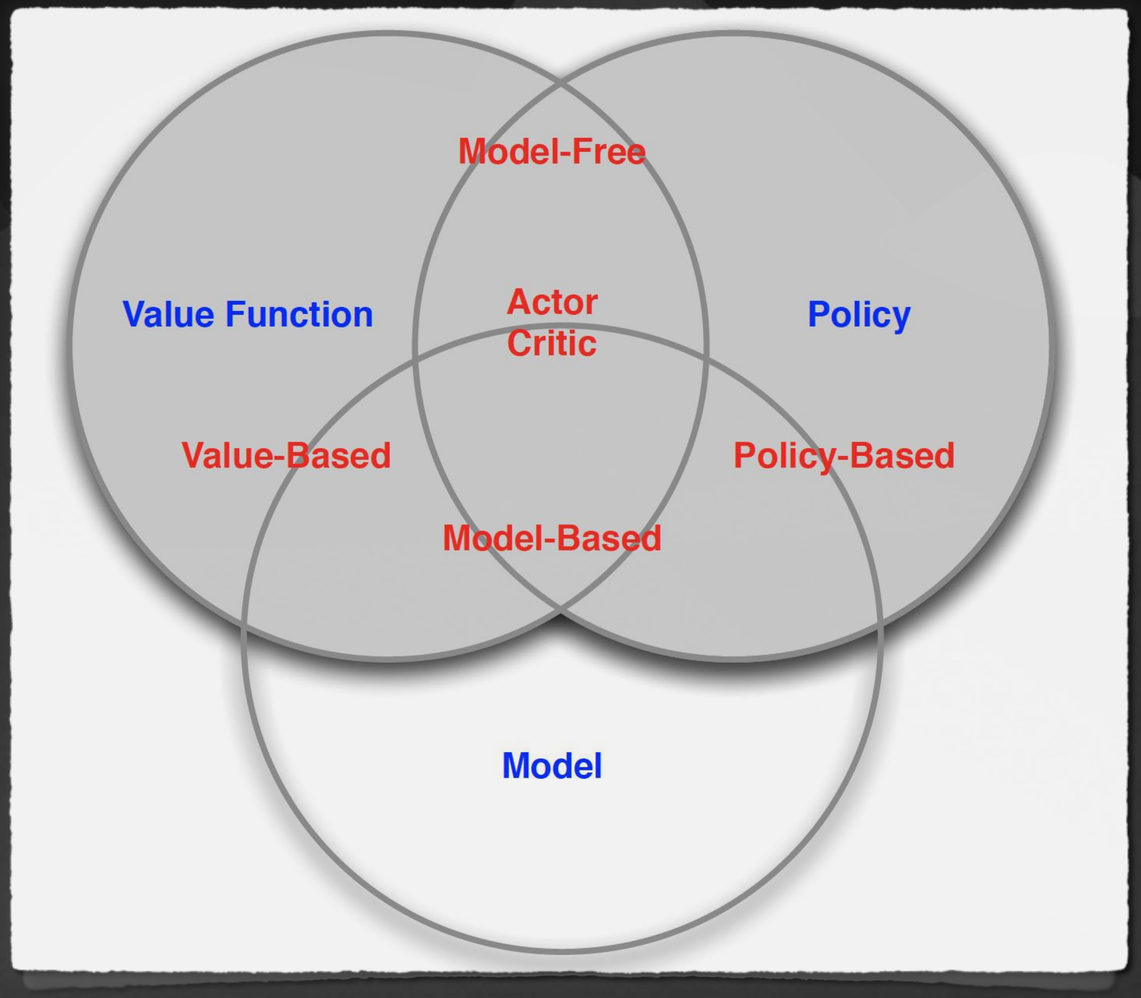
Boston Dynamics, Quadrupedal Robot Sim Training

Agent internals

Agent categories

Agent might be designed to learn

1. **Value based**
    - Value function
2. Policy based
    - Policy
3. ***Actor Critic**
    - Policy
    - Value function
4. Model based
    - Policy and/or value function
    - Model
5. Model free
    - Policy and/or value function

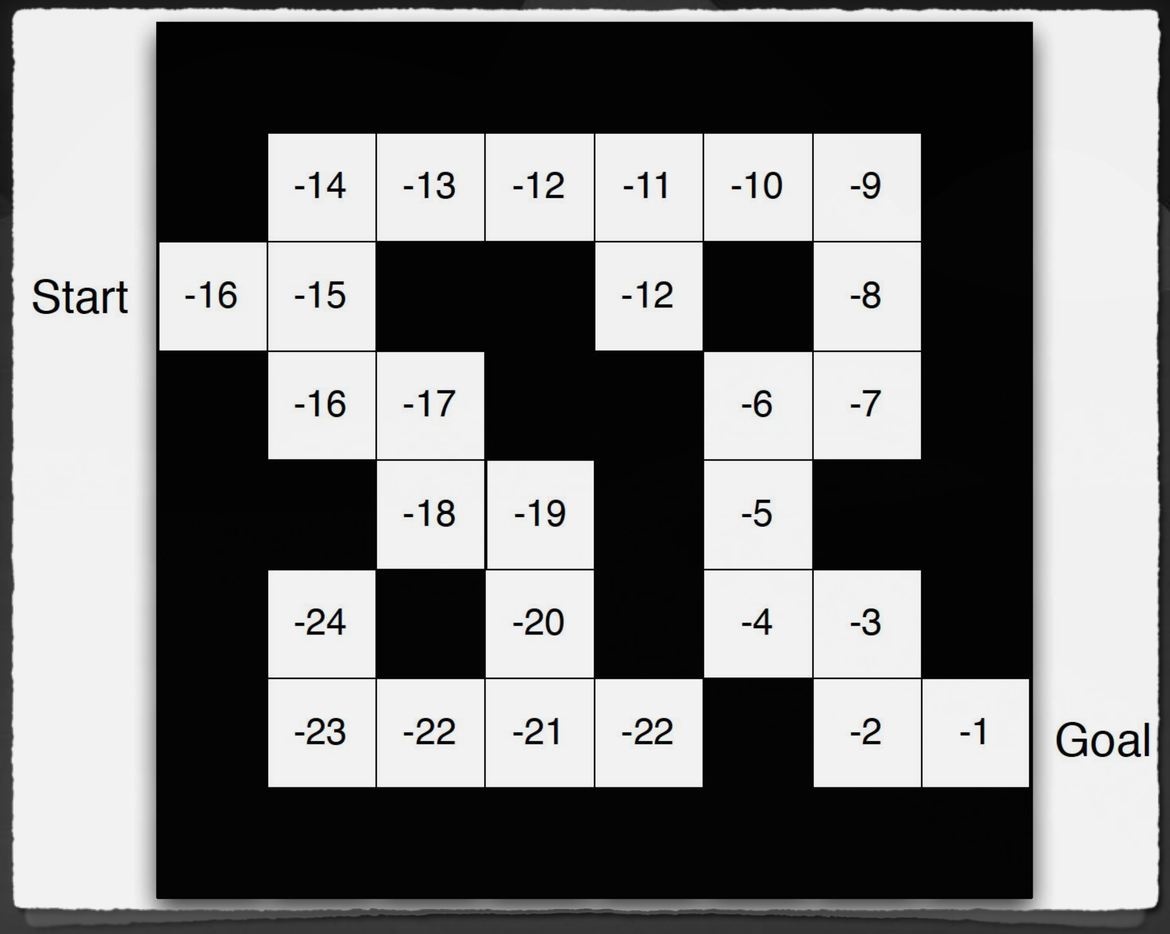


Value function

How good is each state and/or action

1. Value function is a prediction of **future reward**
2. Used to evaluate the goodness / badness of states
3. And therefore to select between actions
4. The value function $v_\pi(s)$ gives the **long-term** value of state s while following the policy π

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$



Numbers represent $v_\pi(s)$ for each state s and immediate reward

Policy

Mapping from states to actions

1. Agent's behaviour function
2. Policies are universal **plans**
3. **It is a map from states to actions**

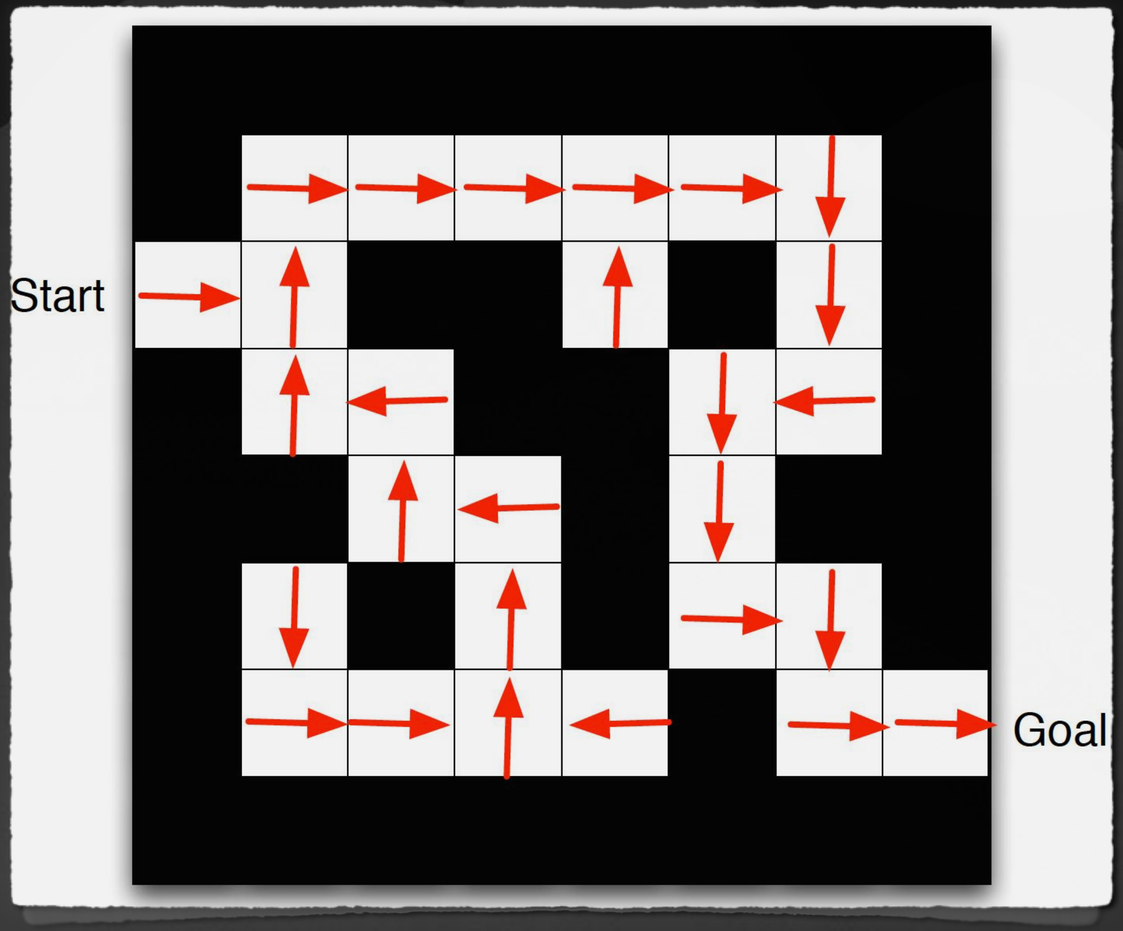
There are two types of policies

1. Deterministic

$$a = \pi(s)$$

2. Stochastic

$$\pi(s \mid a) = P[A_t = a \mid S_t = s]$$



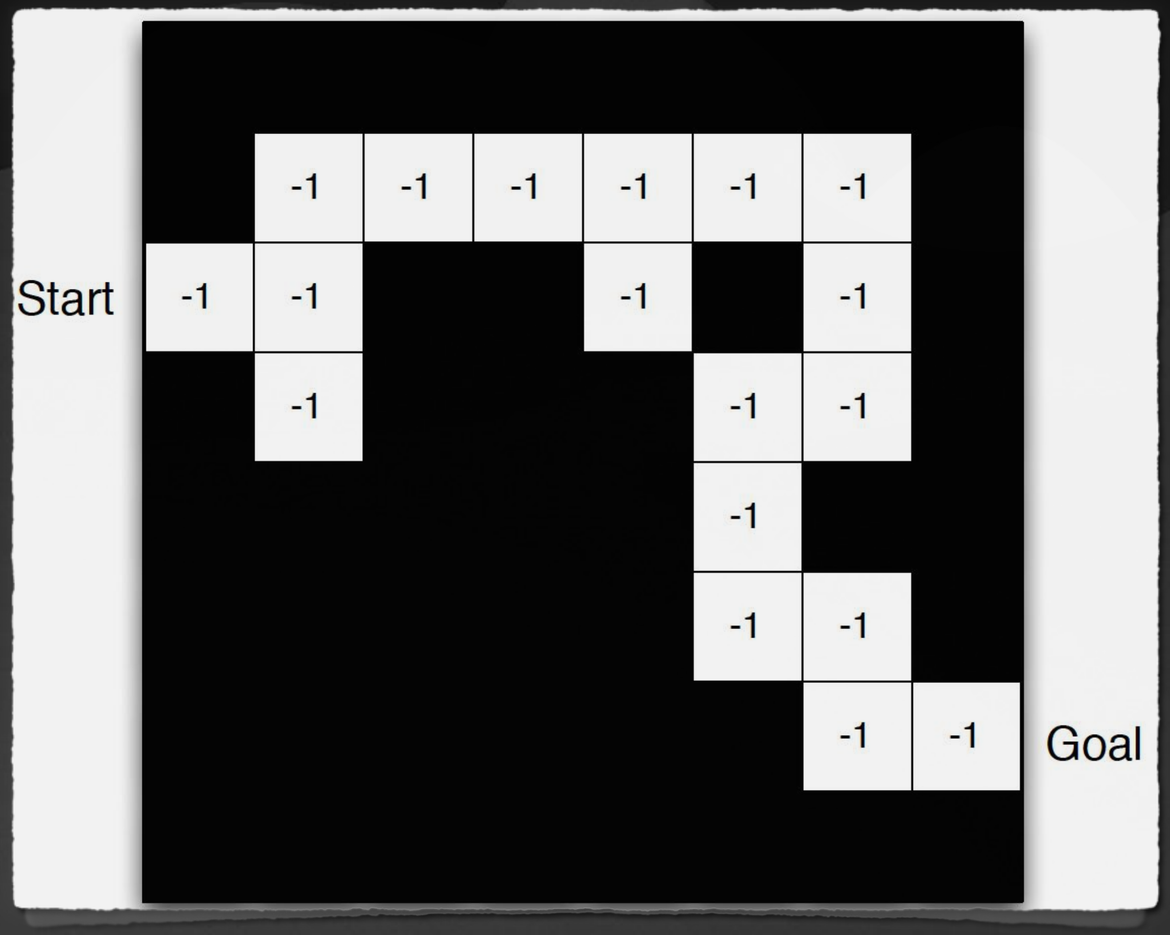
Arrows represent deterministic policy for each state s

Model

Learning underlying structure of environment

1. Agent may have an internal model of the environment
   - Dynamics - how actions change state
   - Rewards - how much reward from each state
2. Agent may use learnt model for planning
   - Generate rollouts
   - Learn from both real experience and rollouts

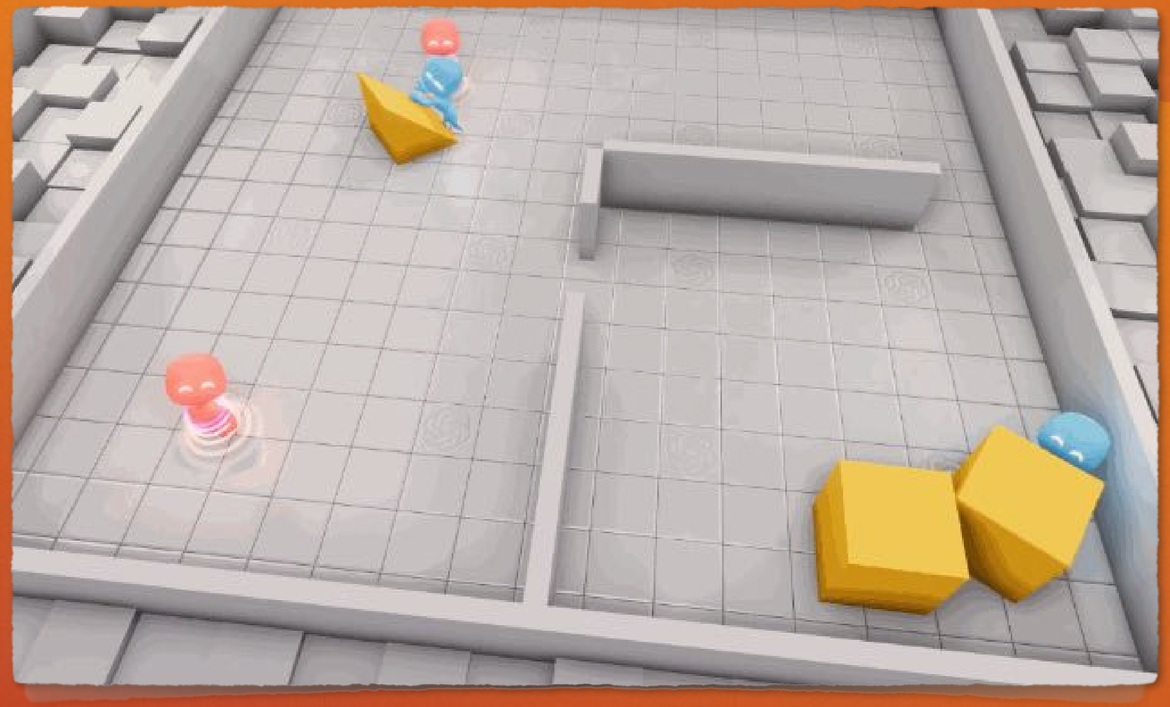
The model might be imperfect



Numbers represent immediate reward from each state s

Value Functions

How to measure goodness



OpenAI, Multi-Agent Competition

Value functions

Engine to compare policies

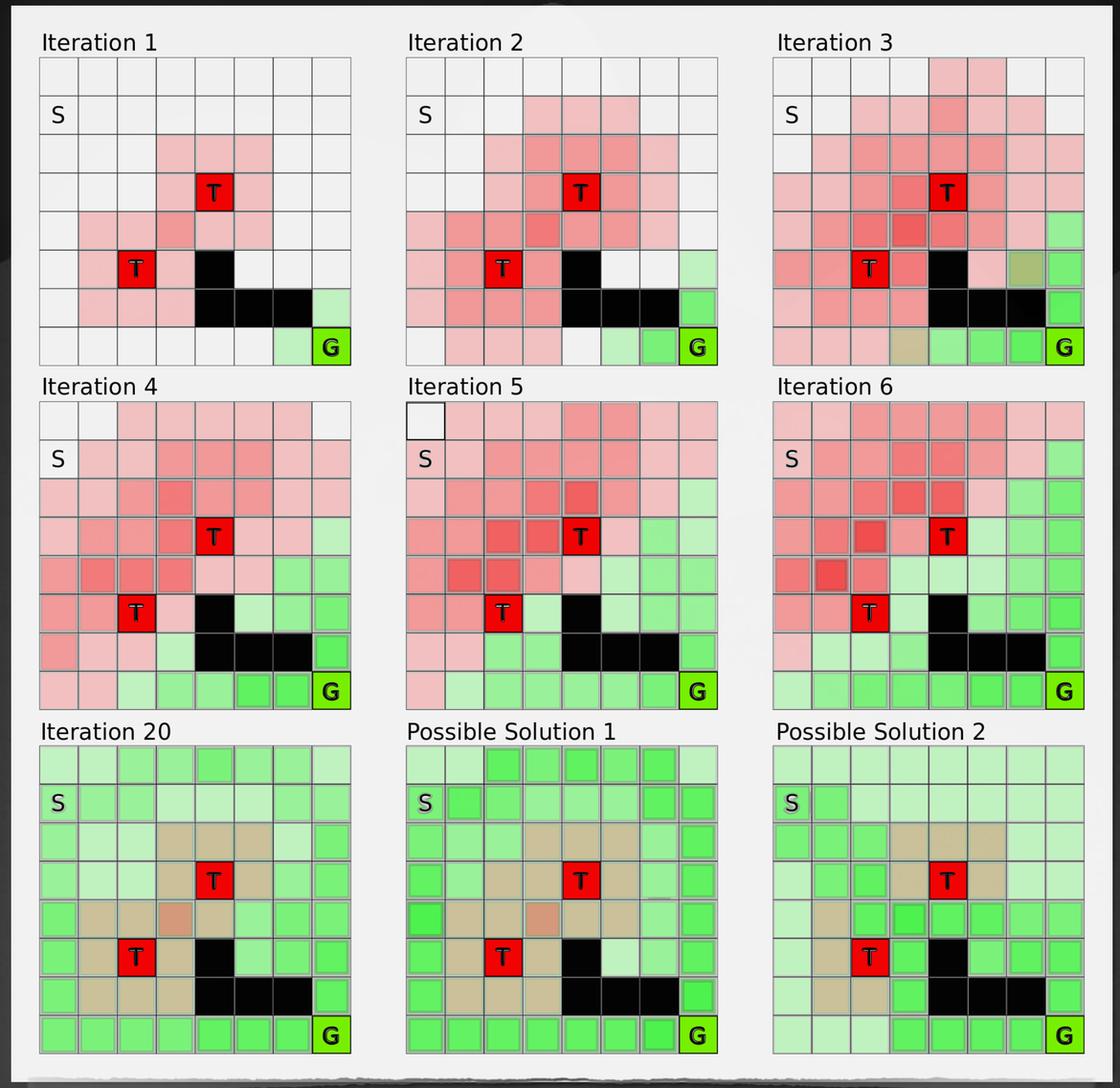
1. There are three main value functions
   - State value function

     How good it is for the agent to be in a given **state [s] under **policy** π*
   - Action value function

     How good is to take **action [a] and then follow the **policy** π*
   - Action advantage function

     *How much **better** is to take **action [a]** than average w.r.t **policy** π*
2. **Finding optimal value function solves the reinforcement learning problem**
   - **Optimal state value function [v_*]**
   - **Optimal action value function [q_*]**

*__How good__ - future rewards that can be expected **[expected return]**



Value iteration method in gridworld maze

State value function

How much juice can we extract from environment

1. The state value function $v_\pi(s)$ is the expected return starting from state s, and then following policy π

2. State value function requires a model of environment or MDP in order to improve policy

   $$\pi'(s) = argmax[R_s^a + P_{ss}^a v(s')]$$

3. The value function can be decomposed into two parts

   - Immediate reward R_{t+1} **[r]**
   - Discounted value of successor state $\gamma v(s')$

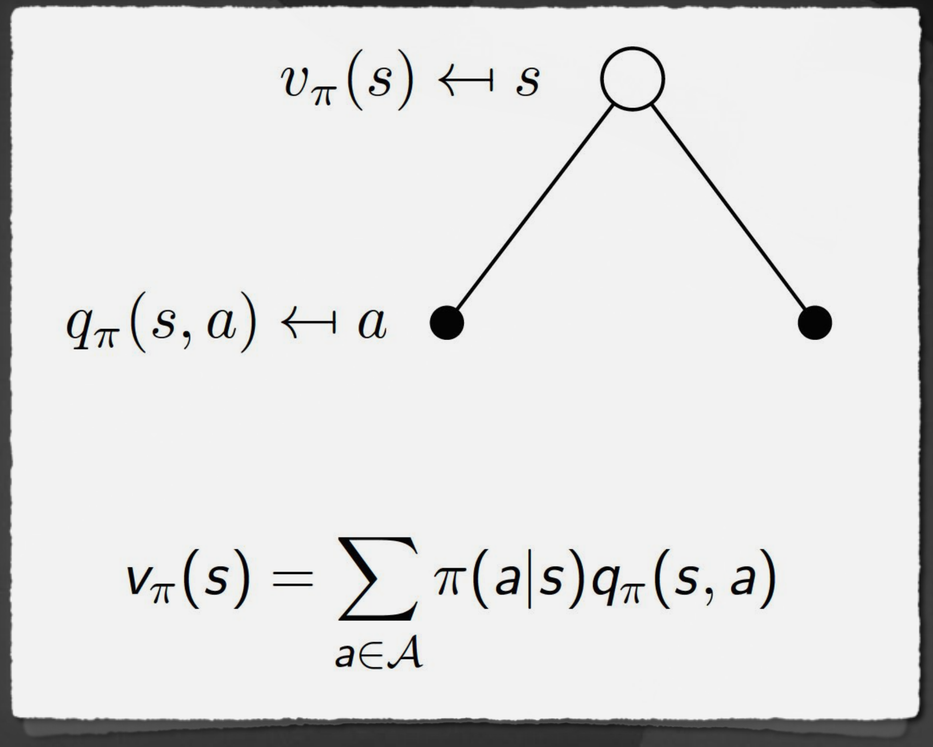
$$
\begin{aligned}
G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\
&= R_{t+1} + \gamma \big(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots \big) \\
&= R_{t+1} + \gamma G_{t+1}
\end{aligned}
$$

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \,|\, s, a) \Big[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s'] \Big] \\
&= \sum_a \pi(a|s) \sum_{s', r} p(s', r \,|\, s, a) \Big[r + \gamma v_\pi(s') \Big], \quad \text{for all } s \in \mathcal{S},
\end{aligned}
$$

This equation is called Bellman expectation equation for state value functions

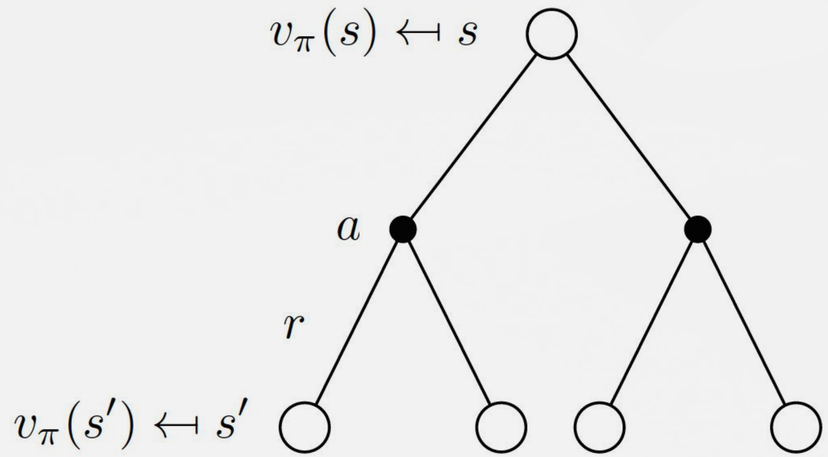
State value function

Backup diagram



$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

State value and action value function relationship



$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

Bellman expectation equation for v_π

Action value function

Control engine

1. The action value function is the expected return if the agent follows policy π after taking action a in state s.

2. Action value function does not require the transition dynamics of environment in order to improve policy

   $$\pi'(s) = argmaxQ(s,a)$$

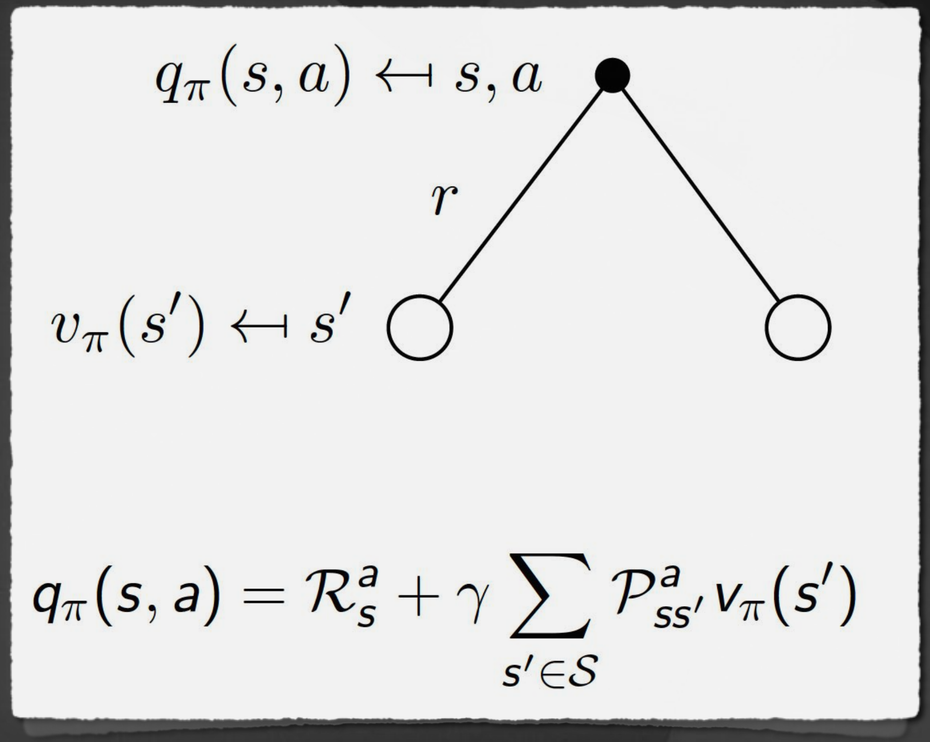
3. Action value function is used to compare actions and improve our policies

$$
\begin{aligned}
& q_\pi(s,a) \\
& = \mathbb{E}_{a\sim\pi(\cdot|s),s'\sim p(\cdot|s,a)}[R(\tau_{t:T})|S_t = s,\, A_t = a] \\
& = \mathbb{E}_{a\sim\pi(\cdot|s),s'\sim p(\cdot|s,a)}[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots + \gamma^T R_T|S_t = s,\, A_t = a] \\
& = \mathbb{E}_{a\sim\pi(\cdot|s),s'\sim p(\cdot|s,a)}[R_t + \gamma(R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T)|S_t = s,\, A_t = a] \\
& = \mathbb{E}_{S_{t+1}\sim p(\cdot|S_t,A_t)}[R_t + \gamma \mathbb{E}_{a\sim\pi(\cdot|s),s'\sim p(\cdot|s,a)}[R_{\tau_{t+1:T}}]|S_t = s] \\
& = \mathbb{E}_{S_{t+1}\sim p(\cdot|S_t,A_t)}[R_t + \gamma \mathbb{E}_{A_{t+1}\sim\pi(\cdot|S_{t+1})}[q_\pi(S_{t+1},\, A_{t+1})]|S_t = s] \\
& = \mathbb{E}_{s'\sim p(\cdot|s,a)}[R(s,a) + \gamma \mathbb{E}_{a'\sim\pi(\cdot|s')}[q_\pi(s',a')]]
\end{aligned}
$$

This equation is called Bellman expectation equation for action value functions

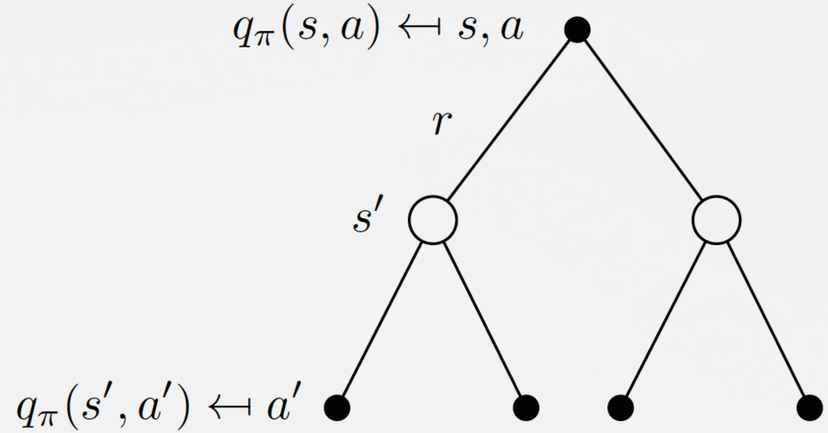
Action value function

Backup diagram



$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

Action value and state value function relationship



$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a')$$

Bellman expectation equation for q_π

Action advantage function

How much better is to take some action

$$a_\pi(s,a) = q_\pi(s,a) - v_\pi(s)$$

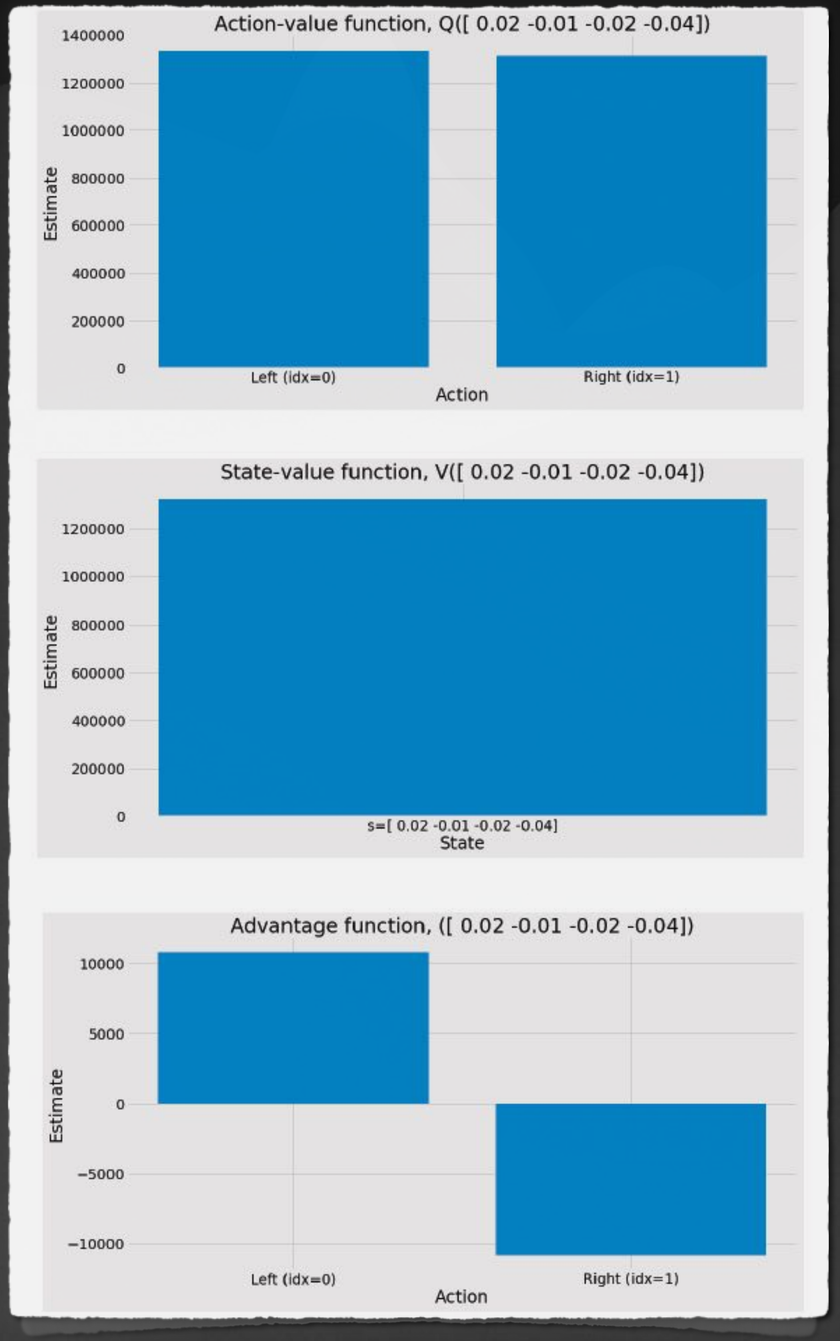
1. The action advantage function describes how much better is to take **action a** instead of choosing **default** action under policy π

2. The action advantage function describes **relation** between actions and states

   - Taking worst action in good state

   - Could be better than taking the best action in a bad state

3. The bottom line is that values of actions depend on values of states



Thank You

Appendix

The objective
Finding optimal value

1. The optimal value function specifies the best possible performance in the MDP
2. An MDP is “solved” when we know the optimal value function
3. The optimal state value function $v_*(s)$
   - Is the maximum value function over all policies

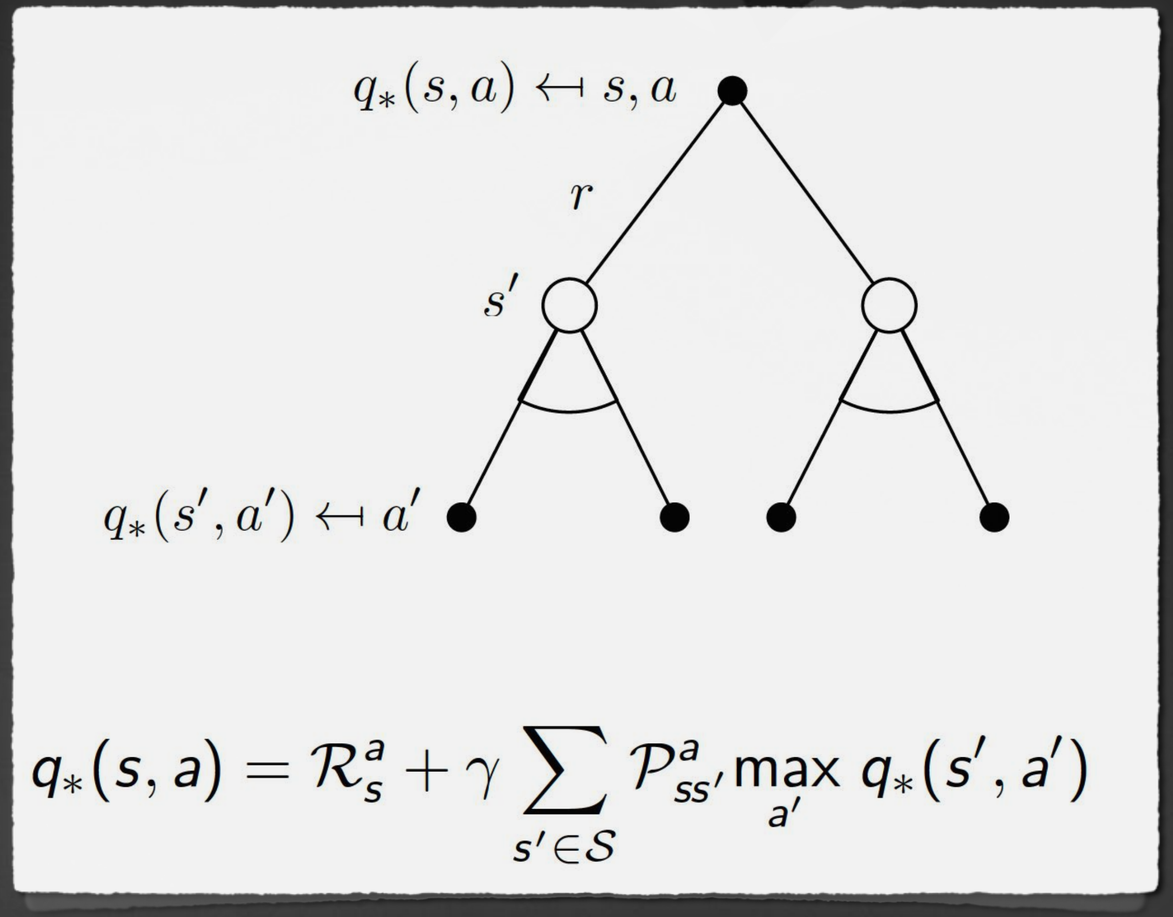
$$v_*(s) = \max_\pi v_\pi(s)$$

4. The optimal action value function $q_*(s, a)$
   - Is the maximum action value function over all policies

$$q_*(s, a) = \max_\pi q_\pi(s, a)$$

There can be only one unique optimal value function

There can be many optimal policies



Bellman optimality equation for action value functions

Optimal policy

Finding optimal behaviour

1. An optimal policy is a policy that for every state can obtain expected returns greater than or equal to any other policy

2. There is always a deterministic optimal policy for any MDP

3. If we know $q_*(s, a)$, we immediately have the optimal policy

$$\pi_*(s \mid a) = \begin{cases} 1 & if \;\; a = argmax \; q_*(s, a) \\ 0 & otherwise \end{cases}$$

For any Markov Decision Process

1. There exists an optimal policy π_*, that is better than or equal to all other policies

$$\pi_* \geq \pi, \forall \pi$$

2. All optimal policies achieve the optimal state value function

$$v_{\pi_*}(s) = v_*(s)$$

3. All optimal policies achieve the optimal action value function

$$q_{\pi_*}(s, a) = q_*(s, a)$$