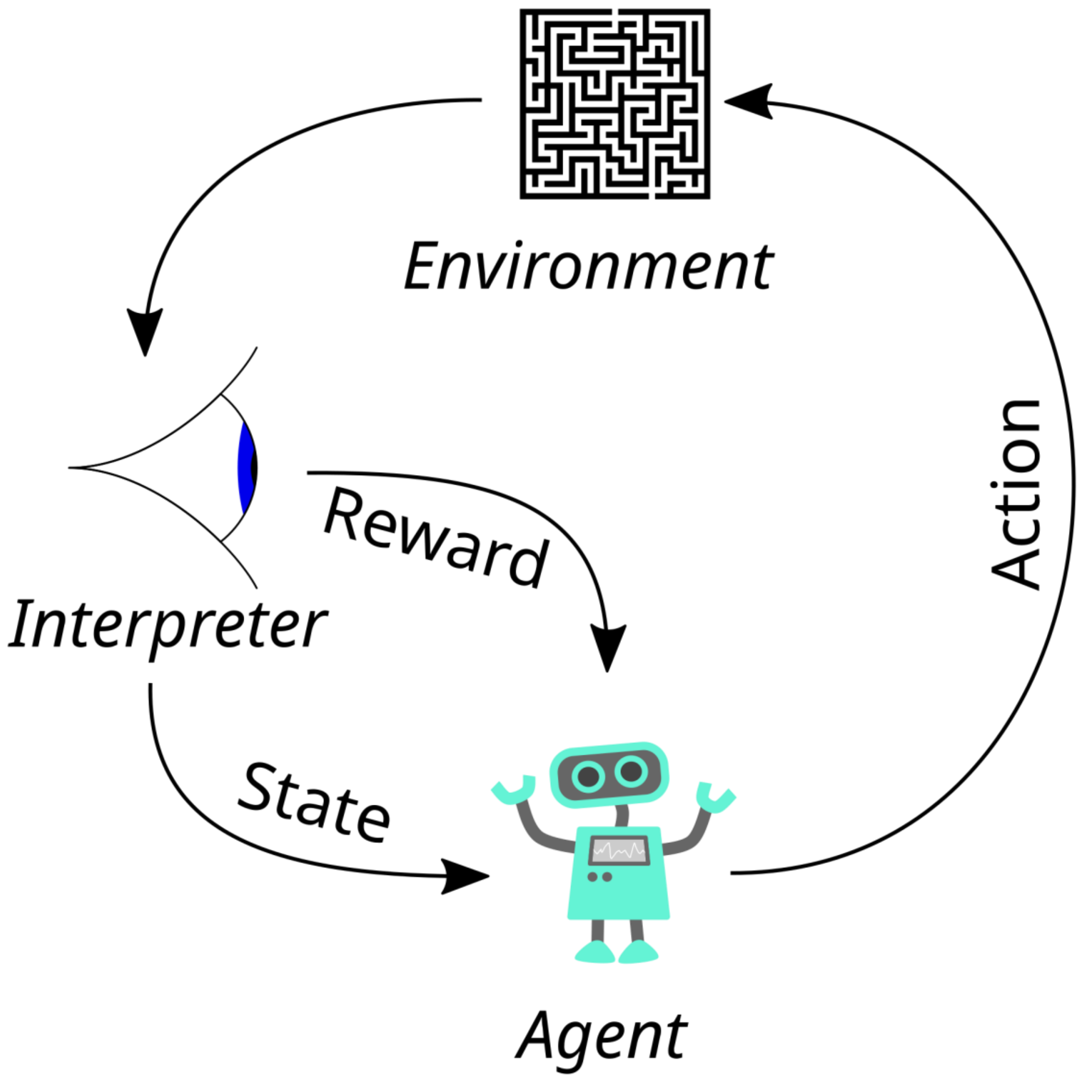
SIT796 Reinforcement Learning

Machine and Reinforcement Learning in History

Presented by:
Dr. Thommen George Karimpanal
School of Information Technology

RL



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

RL Origins

Richard Bellman
(1950s)

Richard Sutton and Andrew Barto
(since 1980's)

Artificial Intelligence Throughout History

Jacquard loom of early 1800s

- Translated card patterns into cloth designs

Charles Babbage's analytical engine (1830s & 40s)

- Programs were cards with data and operations

Ada Lovelace – first programmer

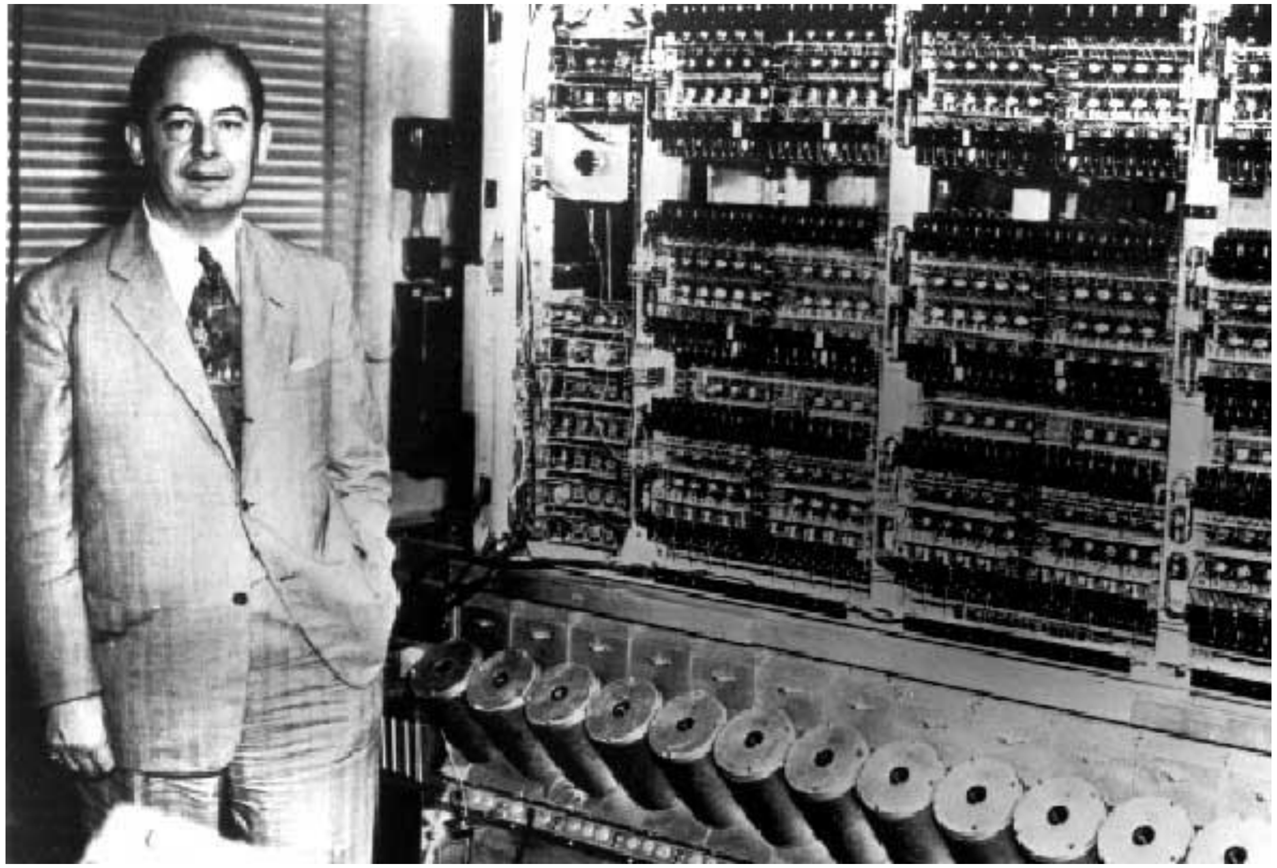
"The engine can arrange and combine its numerical quantities exactly as if they were letters or any other general symbols; And in fact might bring out its results in algebraic notation, were provision made."

The First Computers

1946- John Von Neumann

led a team that built computers with stored programs and a central processor

ENIAC, however, was also programmed with patch cords.



Von Neuman with ENIAC

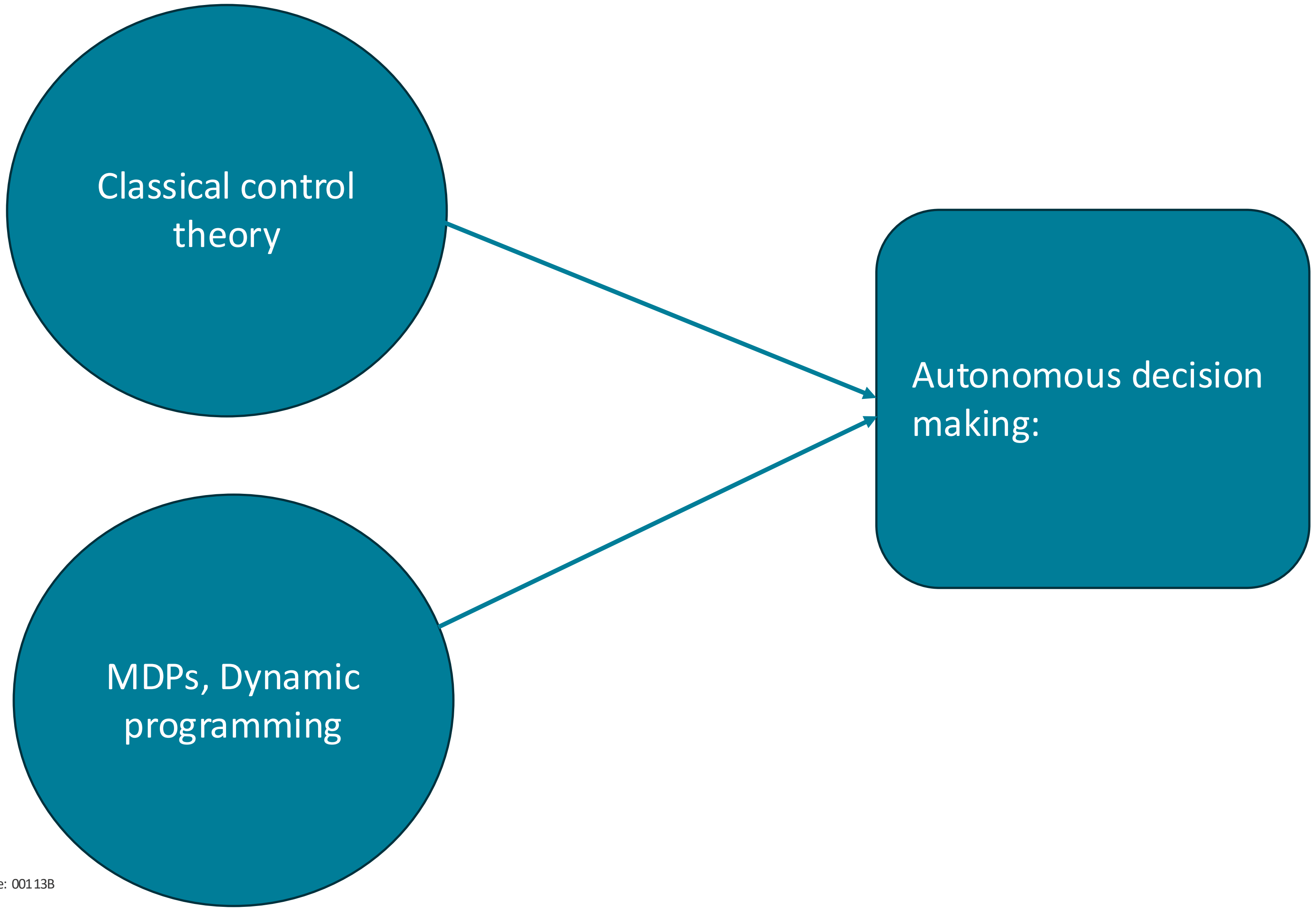
Artificial Intelligence Throughout History

- 1950 - Alan Turing
  - Publishes “Computing Machinery and Intelligence
  - Proposes “the imitation game” which will later become known as the “Turing Test.”
- 1951- Marvin Minsky and Dean Edmunds
  - Build SNARC (Stochastic Neural Analog Reinforcement Calculator)
  - This is the first artificial neural network, using 3000 vacuum tubes to simulate a network of 40 neurons.

Artificial Intelligence Throughout History

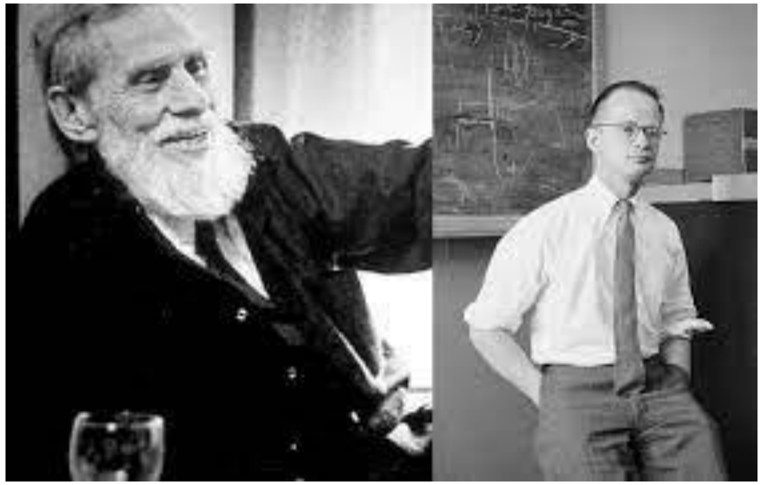
- August 31, 1955 - John McCarthy, Marvin Minsky, Nathaniel Rochester, and Claude Shannon
  - The term “artificial intelligence” is coined
  - They propose a “2 month, 10 man study of artificial intelligence”
  - Submitted to the workshop at Dartmouth College
  - Considered as the official birthdate of the new field
- December 1955 - Herbert Simon and Allen Newell
  - They develop the Logic Theorist, the first artificial intelligence program
  - “The Theorist” proved 38 of the first 52 theorems in Whitehead and Russell's Principia Mathematica.

Artificial Intelligence Throughout History: Decision Making



Artificial Intelligence Throughout History

- McCulloch & Pitts
  - Pitts: self-taught genius, left home at 15
  - McCulloch was impressed by Pitts, homeless at the time
  - Collaborated to write the seminal paper: "A Logical Calculus of Ideas Immanent in Nervous Activity" -First mathematical model of a neural network
  - Research findings that the brain was not solely responsible for image processing
  - Burned his unpublished work on 3D neural networks

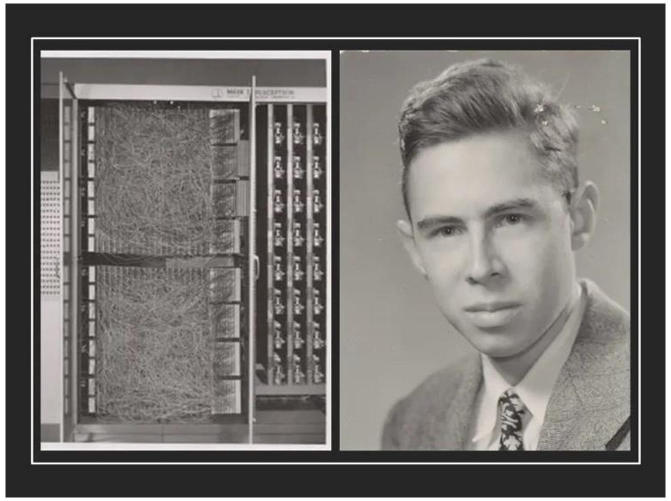

Warren McCulloch (L) and Walter Pitts (R)

Source: https://donaldclarkplanb.blogspot.com/2021/11/mcculloch-pitts-neural-nets.html

Artificial Intelligence Throughout History

Frank Rosenblatt’s Perceptron (1957)

- An electronic device based on a single neuron, able to learn to classify 20x20 images.
- Marvin Minsky showed in his 1969 book *Perceptrons* that perceptrons were fundamentally limited.
- A few years later, this problem was addressed by the development of multi-layer perceptrons.


Frank Rosenblatt
Source: http://csgrad.science.uoit.ca/courses/ist/notebooks/nn-history.html

Marvin Minsky
Source: https://en.wikipedia.org/wiki/Marvin_Minsky

Artificial Intelligence Throughout History

AI Winter (1969-2006)

- Minsky & Papert’s findings in *Perceptrons* (1969) significantly impacted the field
- Funding cuts, fewer and fewer researchers working on AI
- The community moved to more grounded approaches like support vector machines (SVM)
- Meanwhile, a few researchers still persevered
- And PC gaming led to more and more powerful GPUs

Geoff Hinton (Deep learning), Richard Sutton (Reinforcement learning) and Jurgen Schmidhuber (LSTMs)

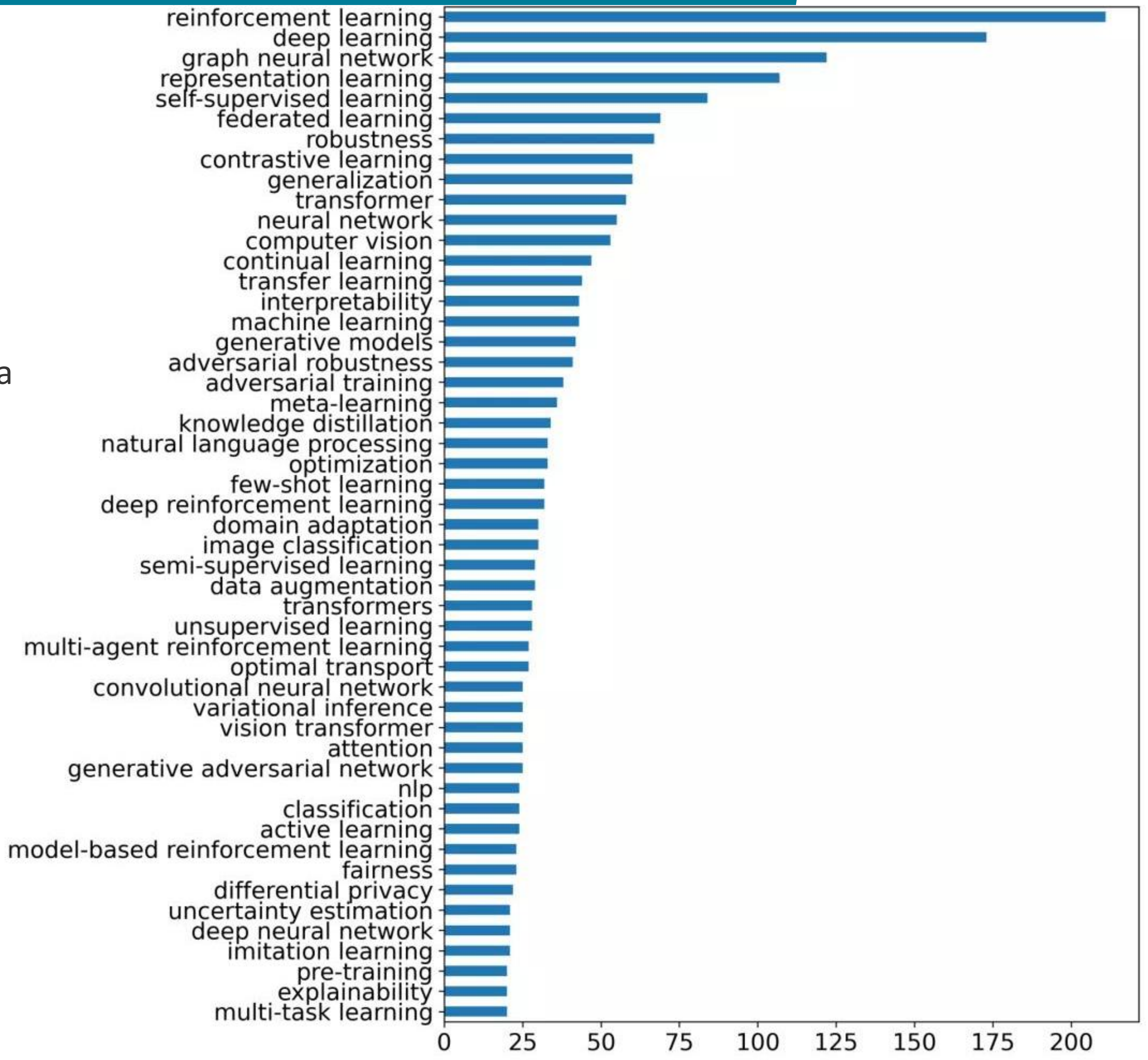
Artificial Intelligence Throughout History

AI Summer (2006-now)

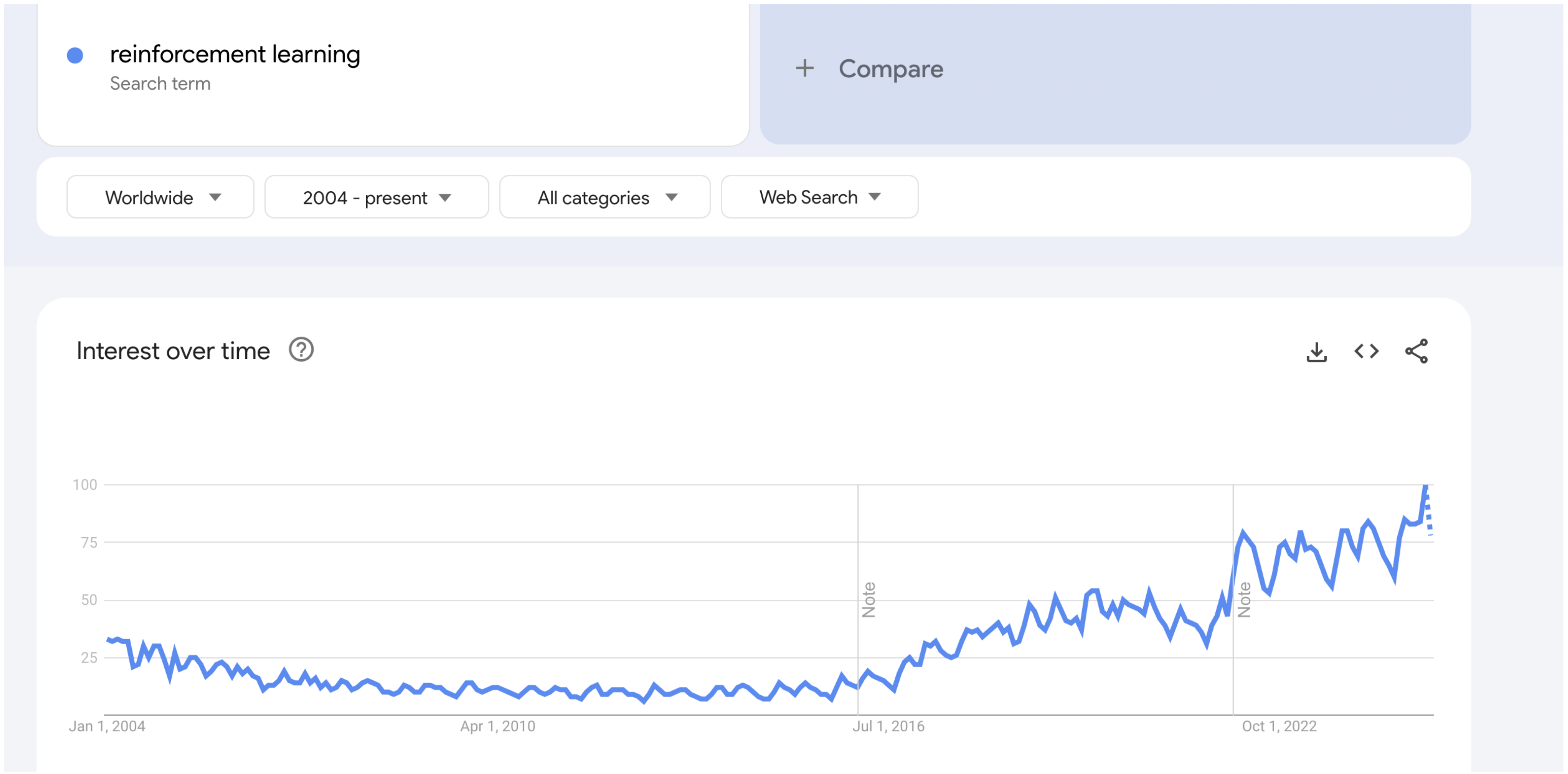
- Deep Belief Nets (2006) by Hinton
- AlexNet (2012) showed significant performance improvements in the ImageNet challenge (database of over 20000 object categories for visual object recognition)
- Deepmind: super-human Atari playing capabilities (2015)
- Transformers, GPT (2021), Sora (2024), DeepSeek (2025)
- A very hot AI summer!

RL: Relevance

Keywords based on ICLR2022 data



RL: popularity



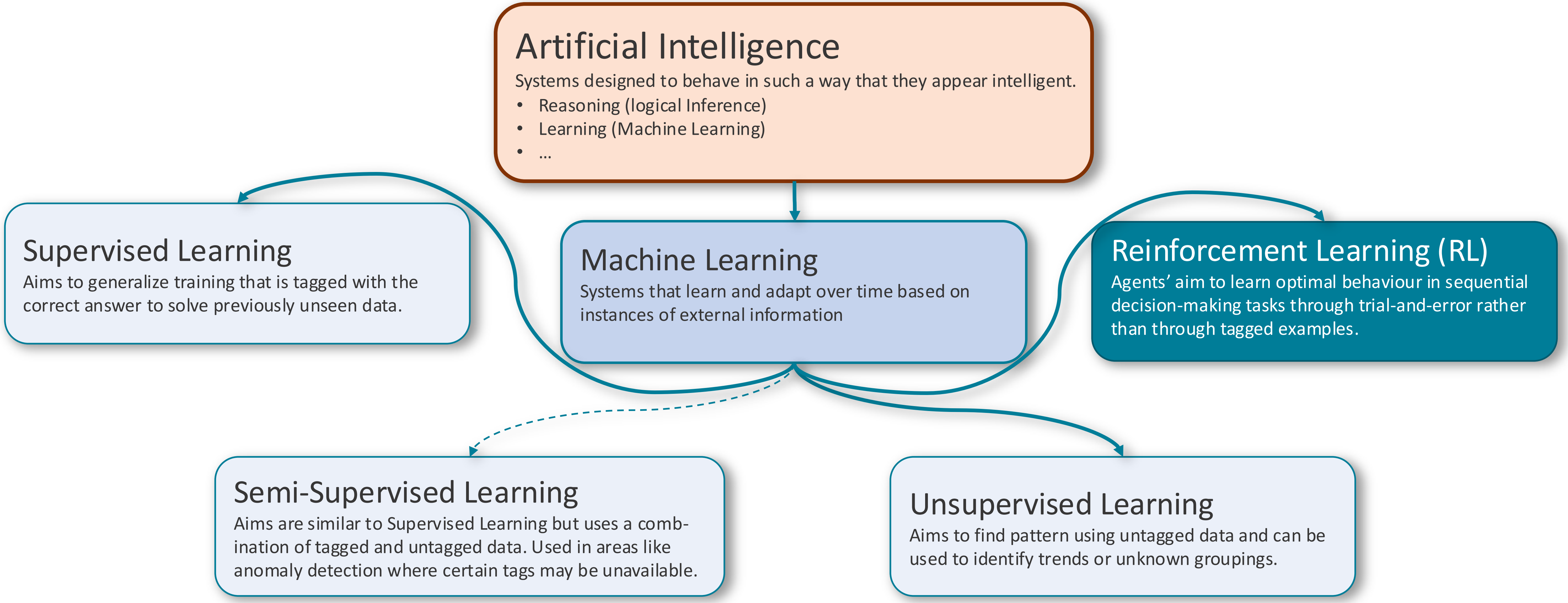
Google trends: Reinforcement Learning

SIT796 Reinforcement Learning

Machine and Reinforcement Learning

Presented by:
Dr. Thommen George Karimpanal
School of Information Technology

Machine and Reinforcement Learning



Artificial Intelligence

Systems designed to behave in such a way that they appear intelligent.

- Reasoning (logical Inference)
- Learning (Machine Learning)
- …

Machine Learning

Systems that learn and adapt over time based on instances of external information

Supervised Learning

Aims to generalize training that is tagged with the correct answer to solve previously unseen data.

Reinforcement Learning (RL)

Agents' aim to learn optimal behaviour in sequential decision-making tasks through trial-and-error rather than through tagged examples.

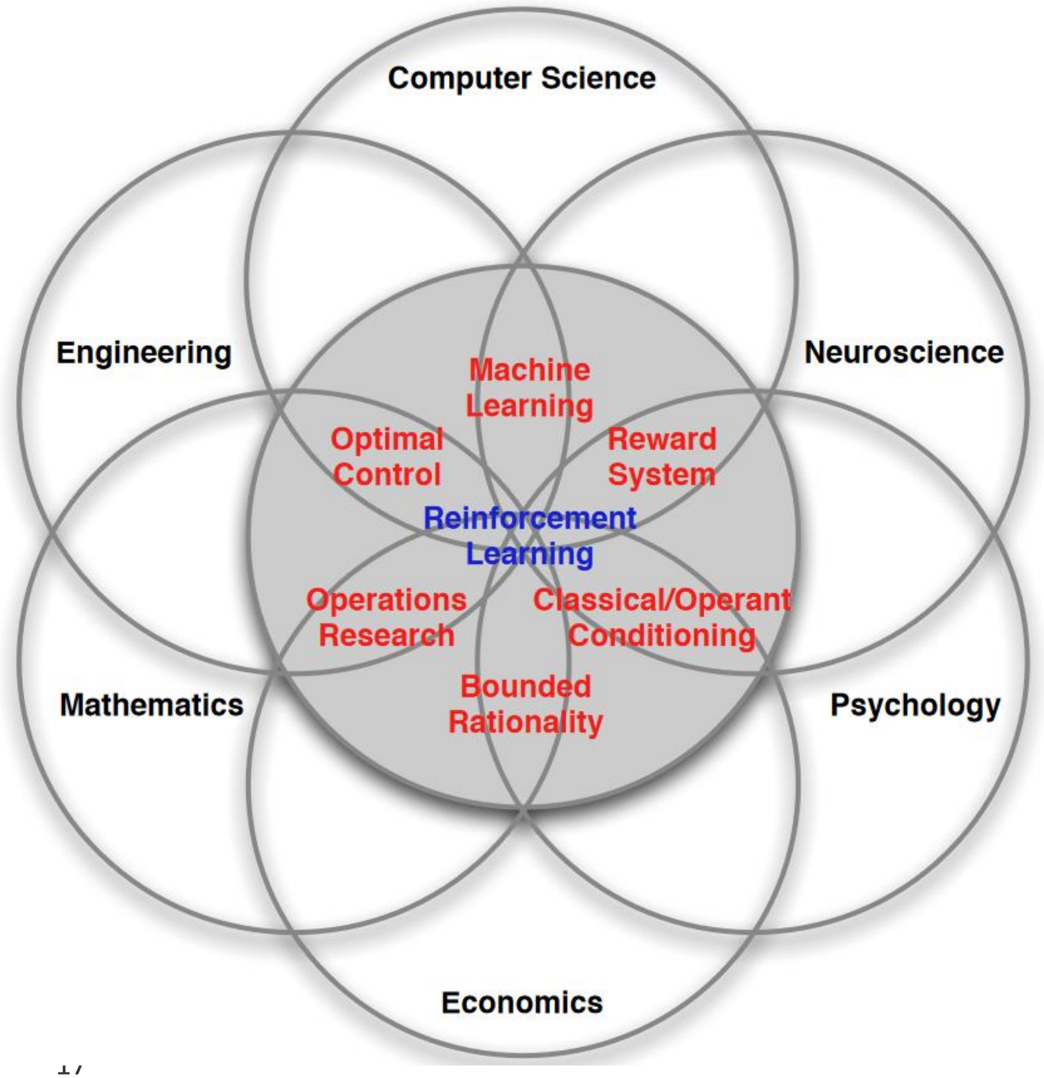
Semi-Supervised Learning

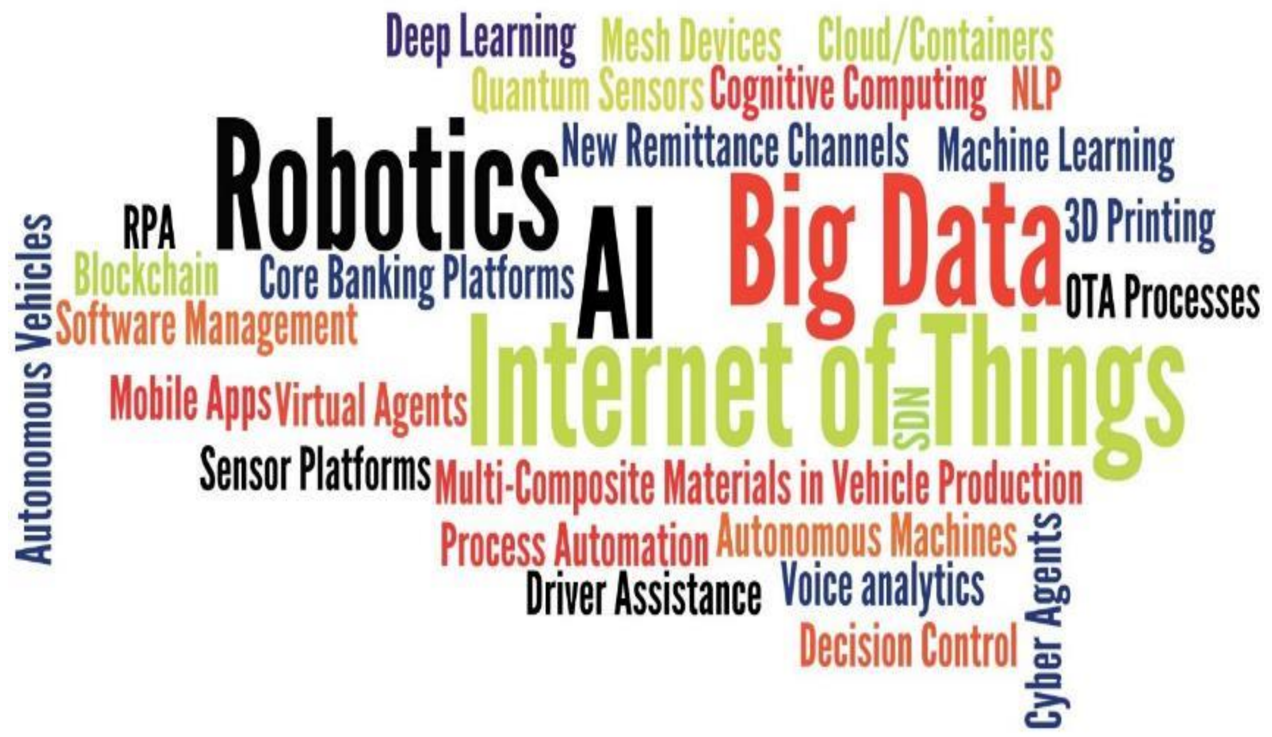
Aims are similar to Supervised Learning but uses a combination of tagged and untagged data. Used in areas like anomaly detection where certain tags may be unavailable.

Unsupervised Learning

Aims to find pattern using untagged data and can be used to identify trends or unknown groupings.

Machine and Reinforcement Learning



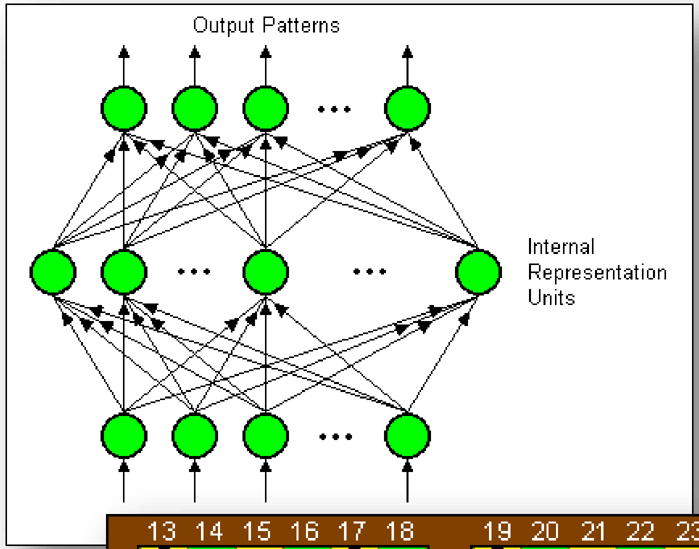


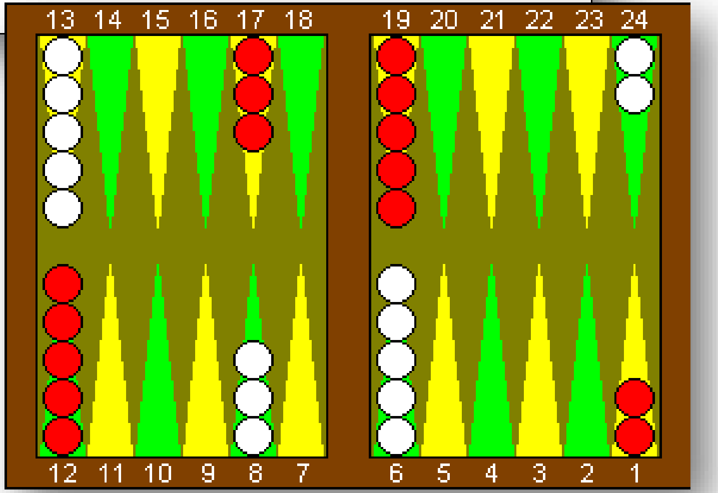
http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/intro_RL.pdf

Examples of Reinforcement Learning (1) – TD Gammon

TD Gammon (1992).

- Finished in the top 10 players of the world
- While not called it at the time – TD Gammon was an early success of Deep RL.
  - Used a multilayer neural network
  - Used Temporal difference learning
- Played unique strategies that expert ended up adopting
  - Eg with a roll of 2-1, 4-1 or 5-1 expert typically used a technique called "slotting" move from point 6 to point 5.
  - Expert, Bill Robertie did a rollout analysis of TD-Gammon's "split" approach of moving from point 23 to 24, and found it was more effective.
  - This is now the standard opening move





https://bkgm.com/articles/tesauro/tdl.html

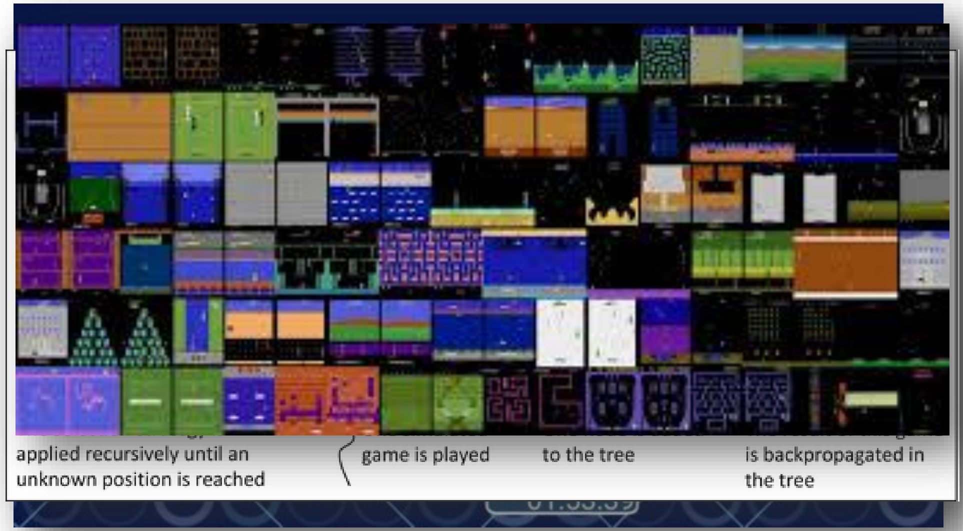
Examples of Reinforcement Learning (2) – DeepMind Games

Agent57

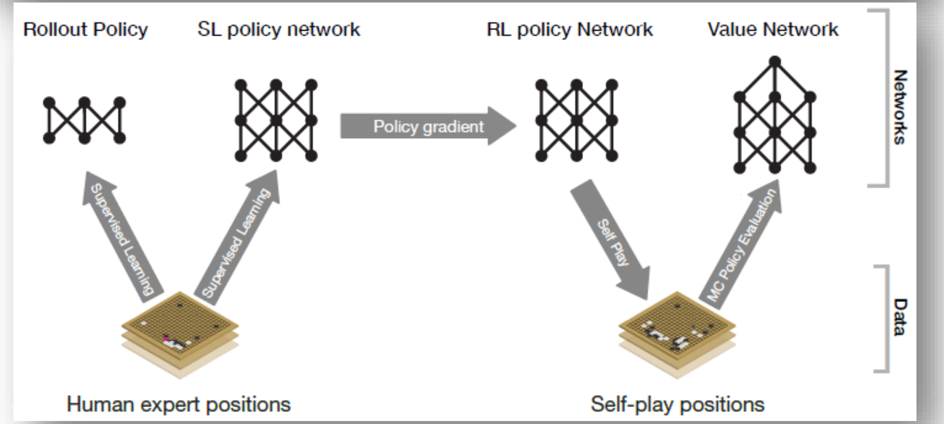
- Could out-perform human benchmarks on all Atari57 games.

***Alpha Go (2015)*.**

- Used Monte Carlo Tree search (Coulom 2006) and learnt from both self and human games.
- Defeated Lee Sedol 4 games to 1 (2016). First program to beat a 9-dan Go champion on a 19x19 board without a handicap.
- Televised internationally, and a documentary movie released.



applied recursively until an unknown position is reached
game is played
to the tree
is backpropagated in the tree



Sutton and Barto (2018)

https://images.techhive.com/images/article/2016/03/go-game-screen-shot-2016-03-08-at-8.17.43-pm-pst-100649230-large.jpg

Winands M.H.M. (2017) Monte-Carlo Tree Search in Board Games. In: Nakatsu R., Rauterberg M., Ciancarini P. (eds) Handbook of Digital Games and Entertainment Technologies. Springer, Singapore. https://doi.org/10.1007/978-981-4560-50-4_27

https://lh3.googleusercontent.com/If132Z_OUQW9jZmdgbaWWJK6cRTwGs5-tAbB27nO_MsB_-sqNYNXOSXrZ8frMu_EuVWOj-6uji7nniRgOLUQ4uazhVnAvRsFc3y2A=w1440

Examples of Reinforcement Learning (2) – DeepMind Games

https://www.youtube.com/watch?v=WXuK6gekU1Y

Examples of Reinforcement Learning

AlphaZero (2018).

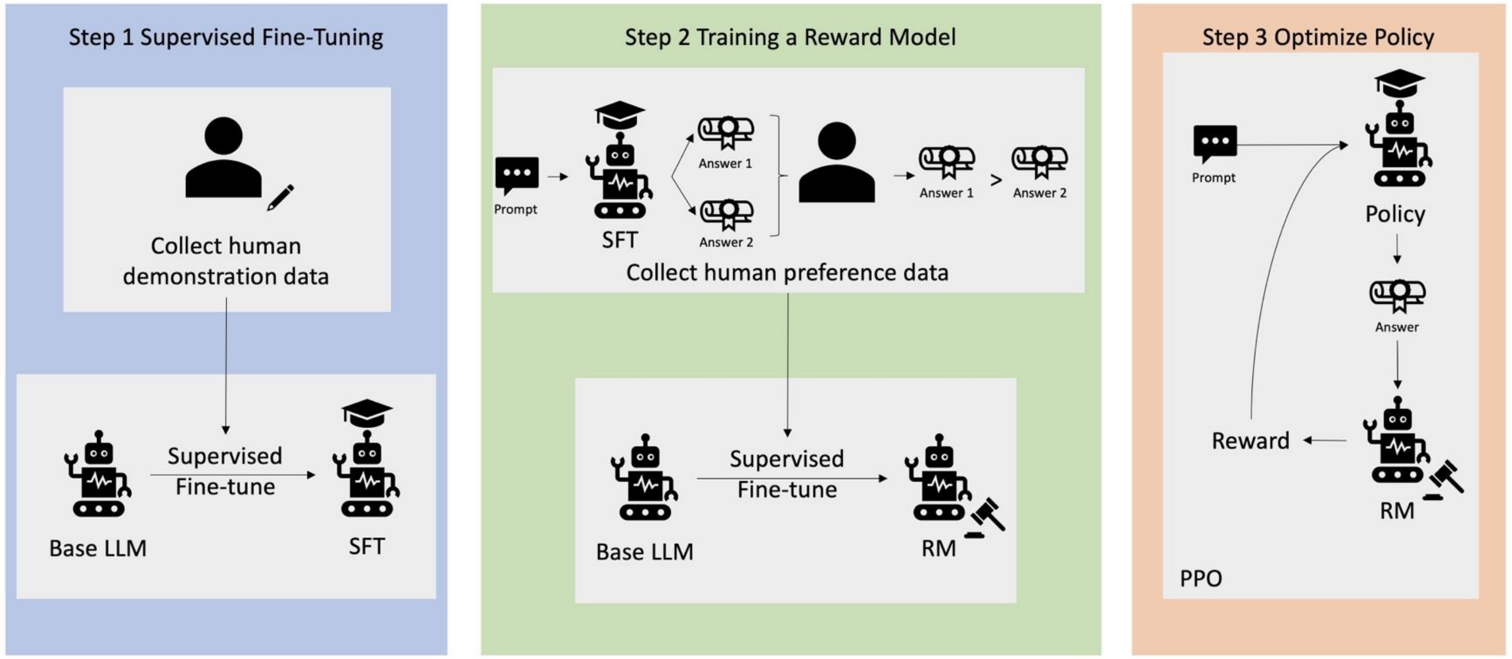
- Can now play different games eg chess and shogi

MuZero (2019).

- Can learn any game without being told anything about the rules
  - Learnt Go better than Alpha Zero. Better on Atari

ChatGPT (2023)

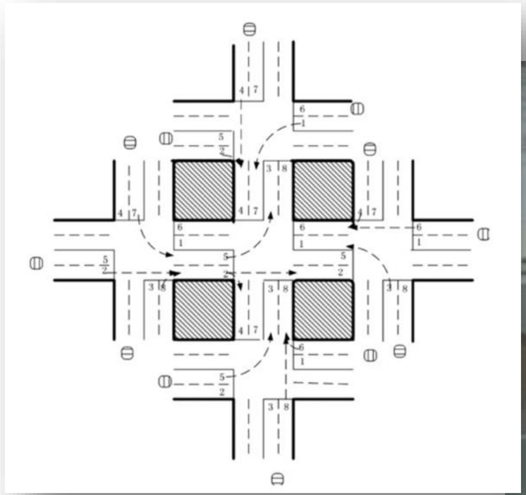
Uses reinforcement learning to decide the more human-like response (RLHF)



Examples of Reinforcement Learning (4)

Many other application areas

- Robotic control, autonomous cars, drones, autonomous underwater vehicles, …
- Energy plant , manufacturing, warehouse operations, logistics, …
- Trading and finance, healthcare, recommendation, marketing,…
- Decision support, natural language processing, video captioning, …



http://web.eecs.utk.edu/~itamar/Papers/IET_ITS_2010.pdf

SIT796 Reinforcement Learning

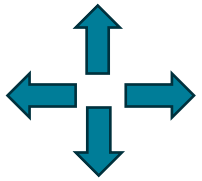
Reinforcement Learning Overview

Presented by:
Dr. Thommen George Karimpanal
School of Information Technology

An Example:

The agent (robot) knows nothing about the environment

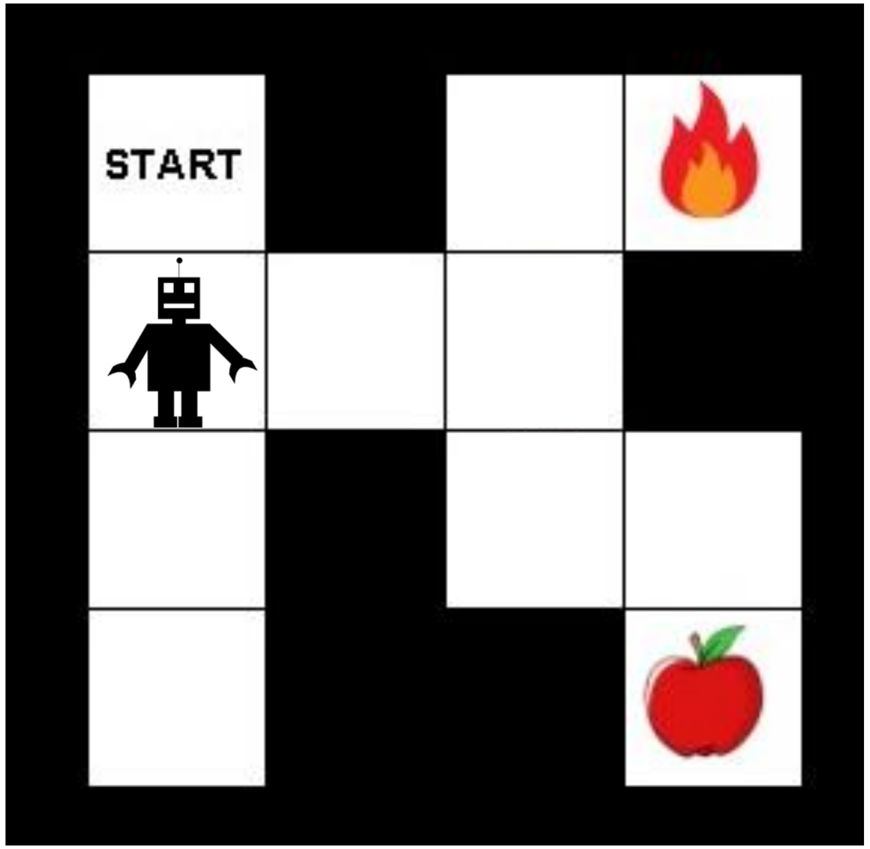
Possible actions: left, right, up, down



What should it do?

Try out different actions and see what happens

Trial and error!



Motivation – Learning from Experience

We learn a number of skills by trial and error

But how and what do we actually learn?

https://directadvicefordads.com.au/new-dads/teaching-baby-to-walk/
https://www.kidsafensw.org/safety/road-safety/bikes-and-wheeled-toys/
https://www.euroschoolindia.com/blogs/how-to-play-chess/

Motivation – Learning from Experience

Two key challenges

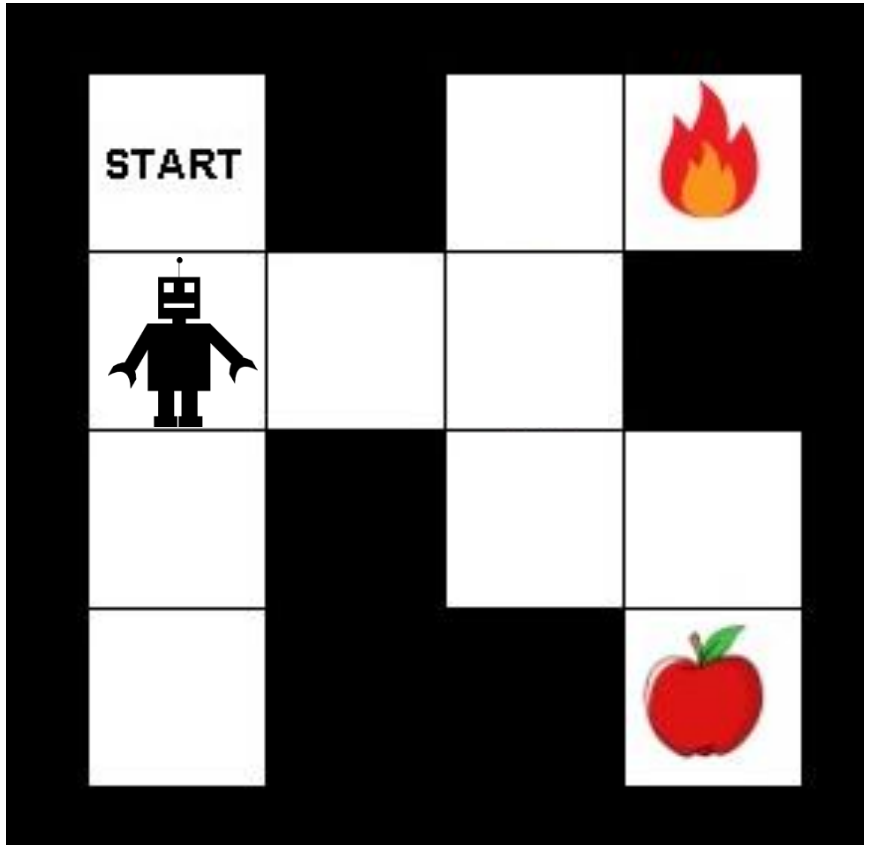
- How to act in a way that is beneficial in the long term and not just short term?

- Temporal credit assignment: which of the previous actions were responsible for an agent’s good/bad performance? Hard problem!

Motivation – Learning from Experience

Reinforcement Learning Involves:

- Optimization: Find an optimal way of making decisions
- Generalization: How well do these decisions apply to similar situations
- Exploration: Use past experience to execute optimal actions, but also ensure that the agent is exposed to new experiences
- Delayed rewards: Consequences of current decisions may be experienced only in the future



SIT796 Reinforcement Learning

The Problem

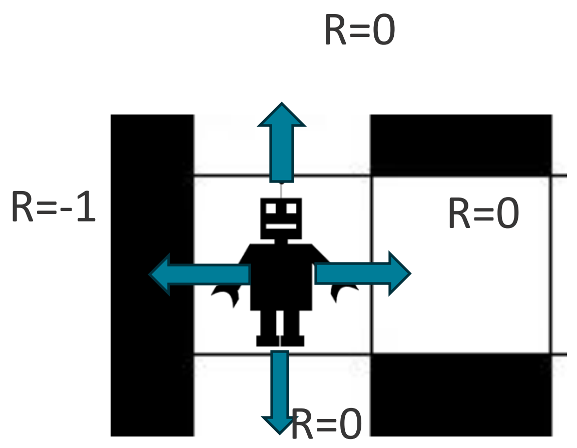
Presented by:
Dr. Thommen George Karimpanal
School of Information Technology

DEAKIN UNIVERSITY

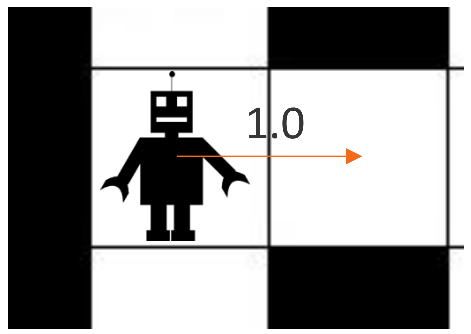
The Problem

Formally, the RL problem is formulated as a Markov Decision Process (MDP)

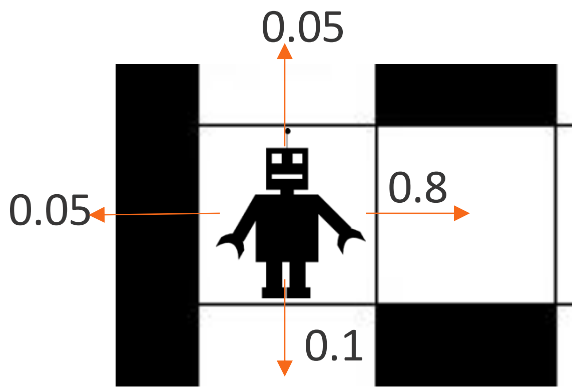
- An MDP is a tuple $M = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{R}\}$
  - $\mathcal{S}$ - The set of possible states
  - $\mathcal{A}$ - The set of actions the agent can take
  - $\mathcal{T}$ - transition probabilities
  - γ - The discount rate or the discount factor.
  - $\mathcal{R}$ - A reward distribution function conditioned.



Reward function



Deterministic transition

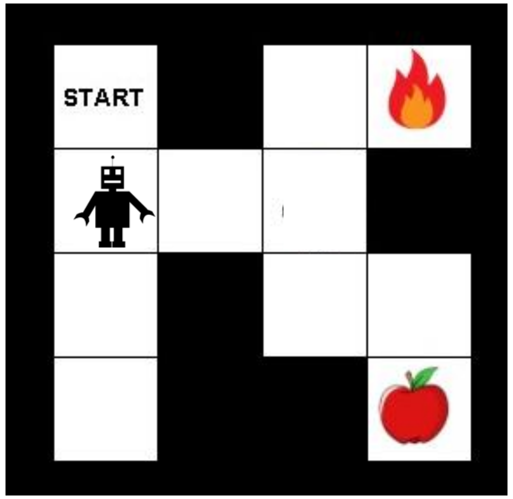


Probabilistic transition

The Environment

An agent is the learner and decision-maker – the environment is everything outside the agent that the agent does not directly control

- The boundary between agent and the environment may not be the physical boundary.
- For example, a robot’s sensors, motors and actuators, while physically part of the robot, are often treated as part of the environment.
- This is the same as separating the human mind (agent) from the eyes, ears, skin, muscles and bones (environment)
- A simple rule is that anything that cannot be changed arbitrarily by the agent is part of its environment



https://data61.csiro.au/~/media/D61/Images/weaver-outdoor-incline.jpg?mw=1600&hash=8E899FC8780FE865A065C9C229D0438106914166

State

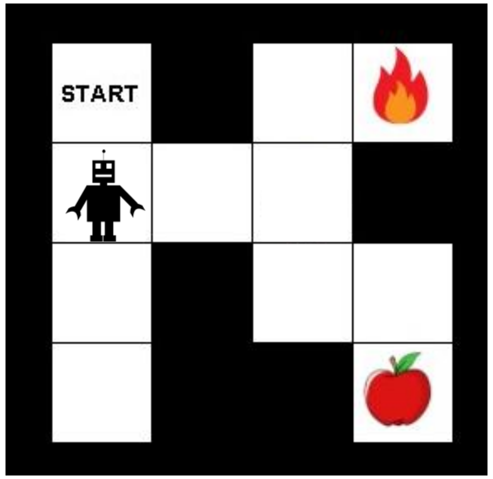
The state is the agent‘s internal representation

- May only be a small portion of the environment.

For example:

- Agents generally operate in discrete time and hence perceive the environment as snap shots – like a film.
- Location of chess pieces / number of pieces it attacks / places the piece can move.

The choice of state representation can significantly affect the agent’s ability to learn a solution.

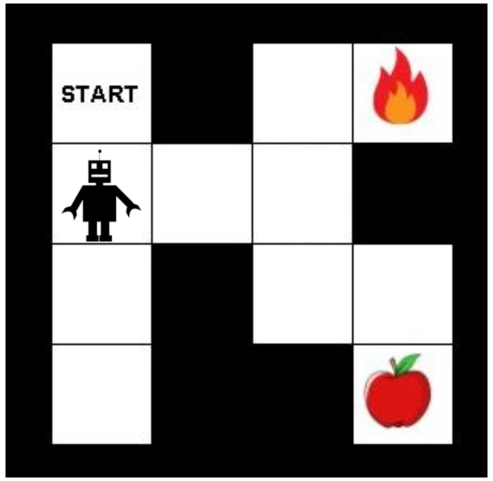


Markov Property: All of the agent’s history is represented by the state

Actions

These are things the agent can do in the environment

Can be discrete or continuous



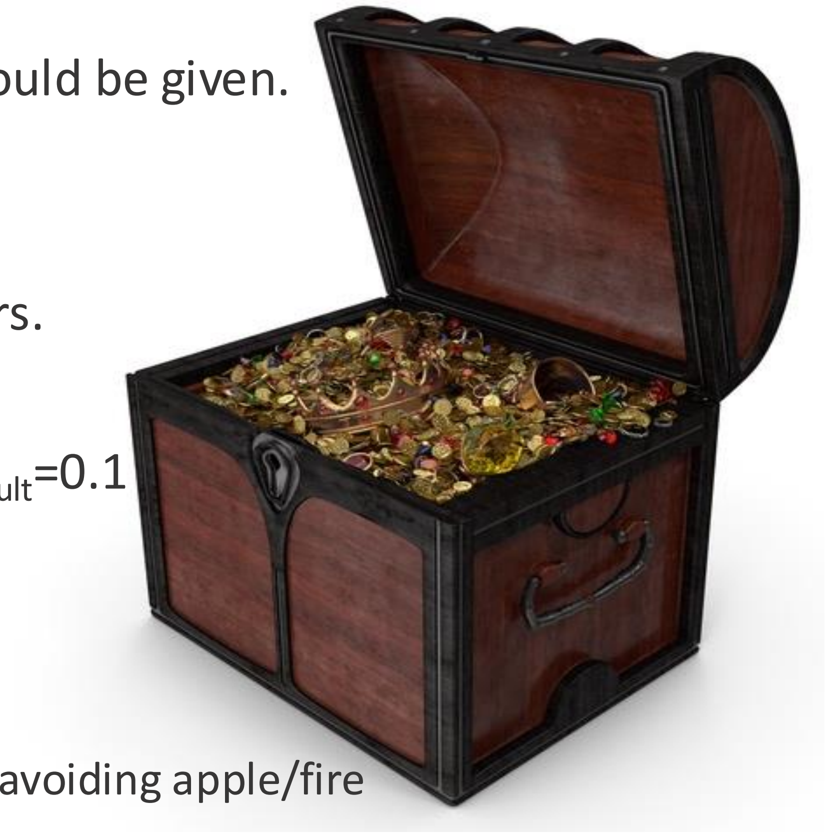
Reward Function

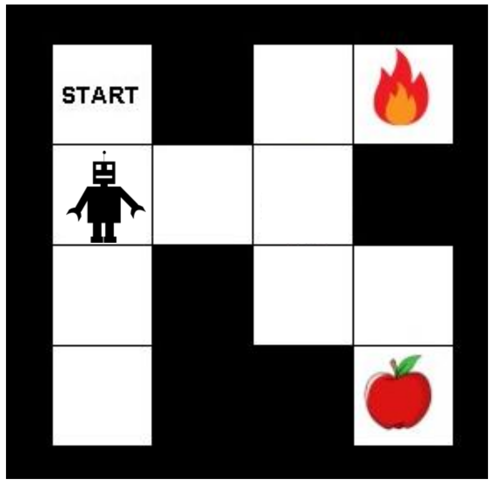
The ***Reward Function*** defines the ***Goal*** of the ***Problem*** being learnt.

The reward function determines how much and when/where reward should be given.

- Note: the reward function is external to the agent.
- Therefore, the agent cannot alter it

Improperly designed reward functions can lead to undesirable behaviours.





Eg: R_{apple}=1 (terminal), R_{fire}=-1 (terminal), $R_{default}$=0.1

Agent can collect an accumulate infinite rewards by avoiding apple/fire states!

https://atlas-content-cdn.pixelsquid.com/stock-images/small-wooden-treasure-chest-RBewxr1-600.jpg

Value Function

The ***Value Function*** is an ***estimate*** of the ***total reward*** that an agent can expect to accumulate in the future in a given state.

Many ways to store/record a value function:

- State-value function *V(s)*: indicates how valuable it is to be in state *s*
- Action-value function (or Q-value) *Q(s,a)*: indicates how valuable it is to be in state *s* and take action *a*

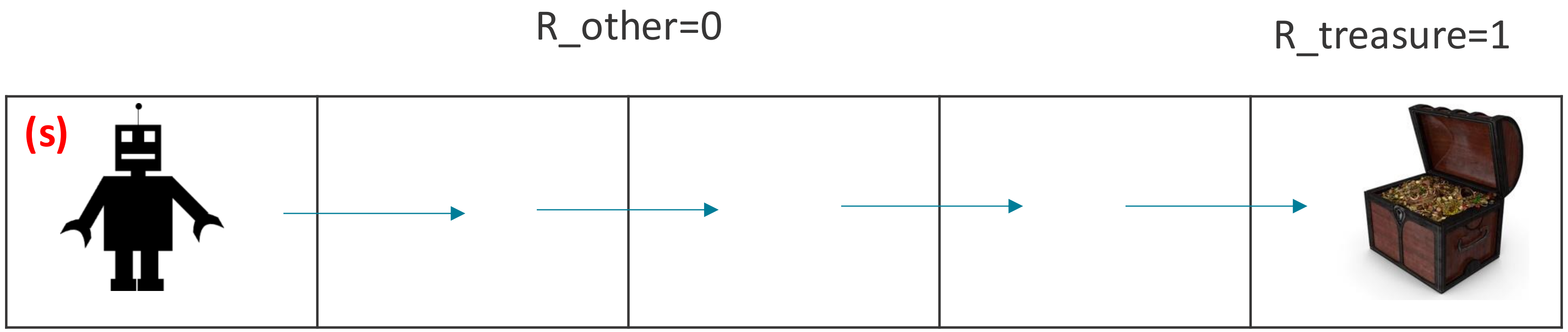
RL aims to estimate the value/action-value function.

https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcROCpO4FHAnt5qvpb_VOfRhOnUHyTZYQTCM5A&usqp=CAU
https://atlas-content-cdn.pixelsquid.com/stock-images/small-wooden-treasure-chest-RBewxr1-600.jpg

Discount Factor

$100 now or $100 after 1 year?   Of course, now!

We value things in the future less



No discounting: $V(s)=0+0+0+1=1$

With discounting (discount factor=0.9): $V(s)=0+0.9*0+(0.9)^{2}*0+(0.9)^{3}*1= 0.729$

https://atlas-content-cdn.pixelsquid.com/stock-images/small-wooden-treasure-chest-RBewxr1-600.jpg

Agent’s Policy

The value function provides a means to map an agent’s states to the actions

- This mapping is called the agent’s policy, π_t.
- Where $\pi_t(s, a)$ is the probability of selecting action a at state s at time t.

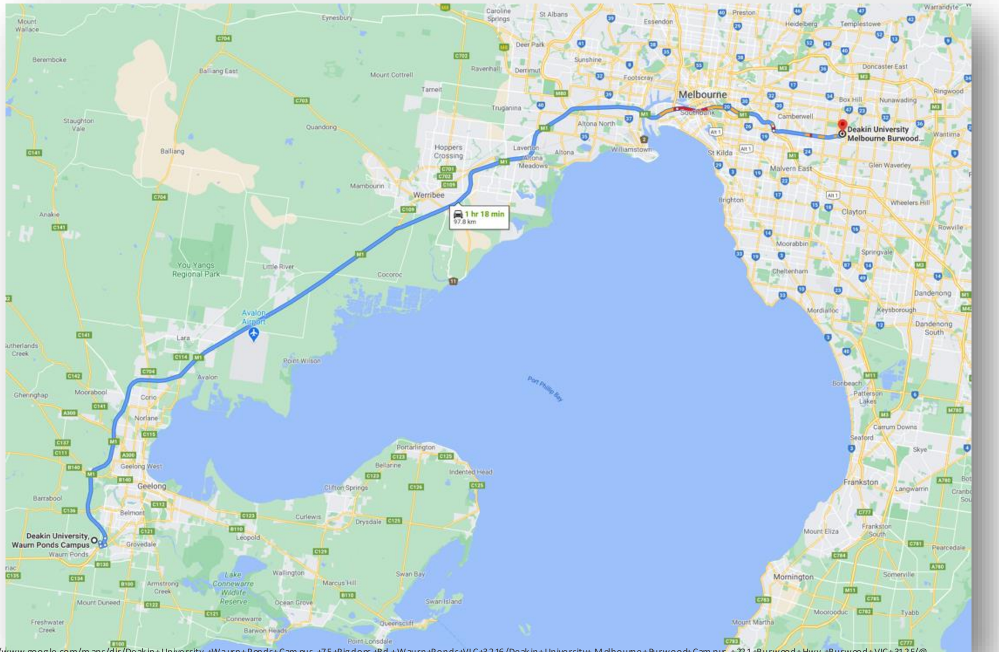
You can think of the policy as being the way the agent has chosen to behave for a given state

- It is this mapping (policy) that RL methods are attempting to learn.
- Once learnt these probabilities should be the same as the optimal probability for each state, denoted $\pi_t^*(s, a)$

RL methods update their policy – attempting to maximise the total amount of reward over the long run.

After the value function is learnt, the best (optimal) action is simply the one that corresponds to the highest value.

But should the agent simply choose the action with the highest value all the time?


Melbourne
Deakin University Melbourne Burwood...
Deakin University, Waurn Ponds Campus
1 hr 18 min
Geelong
Werribee
Port Phillip Bay
Mornington
Frankston

https://www.google.com/maps/dir/Deakin+University,+Waurn+Ponds+Campus,+75+Pigdons+Rd,+Waurn+Ponds+VIC+3216/Deakin+University+Melbourne+Burwood+Campus,+221+Burwood+Hwy,+Burwood+VIC+3125/@-38.0099452,144.5760167,11.17z/data=!4m14!4m13!1m5!1m1!1s0x6ad41329860 9d90f:0xc0f7d39b2ebe86b2!2m2!1d144.298789!2d-38.19691!1m5!1m1!1s0x6ad640592c2ddceb:0x805bd52f251bd12!2m2!1d145.1149861!2d-37.8474187!3e0

SIT796 Reinforcement Learning

Action Selection

Presented by:
Dr. Thommen George Karimpanal
School of Information Technology

Action Selection (Exploration vs Exploitation)

A fundamental issue in Reinforcement Learning is the trade-off between:

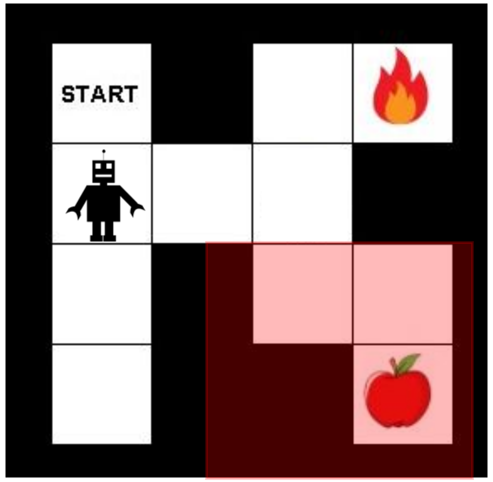
- Exploration: Search to see if there are other, potentially better, ways to do your task
- Exploitation: Keep doing what you have already learnt is best
  - Select the action with the highest value function

We will discuss a number of ways to address this dilemma:

Multi-armed Bandits

greedy, ε-greedy, Upper Confidence Bounds, and Soft-max

Optimistic Initialisation



https://drek4537l1klr.cloudfront.net/morales/v-4/Figures/Images_01-04_118.jpg

Learning (Updating the Value Function)

Learning is the process of updating/changing/improving our understanding of the what is the best policy

- Tells the system the correct answer regardless of the answer actually given by the system.

One approach is to use the Value Function.

- Recall: The Value Function is an estimate of our future expected reward.
- So if after executing an action we find we received more/less reward that we expected then we should update our value function

Therefore, if our value for a state, s, at time, t, was $V(s_t)$, then we can update it using the ***Bellman equation***.

$$V(s_t) \leftarrow V(s_t) + \alpha[V(s_{t+1}) - V(s_t)],$$

Where α is set to a value between $0 < \alpha \leq 1$. This is called the step-size parameter (or learning rate)

α is sometimes set high initially and reduced overtime.

https://drek4537l1klr.cloudfront.net/morales/v-4/Figures/Images_01-04_118.jpg

Tasks

A task in RL is an MDP {S,A,R,T}

- Represents one instance (epoch) of the RL problem

Examples

- Game of Chess
- Stand up for as long as possible
- Minimise customer wait times
- Extinguish a fire

Two primary types of tasks:
Episodic tasks – is one that has a defined terminal condition when the task is completed. Eg: Chess, reach destination, solve problem

Continuing Tasks – tasks with no terminal condition. Eg: moving in a circular path, learning mathematics

http://anji.sourceforge.net/polebalance.htm
https://media.wired.com/photos/5f592bfb643fbe1f6e6807ec/16:9/w_2400,h_1350,c_limit/business_chess_1200074974.jpg

Prediction vs Control

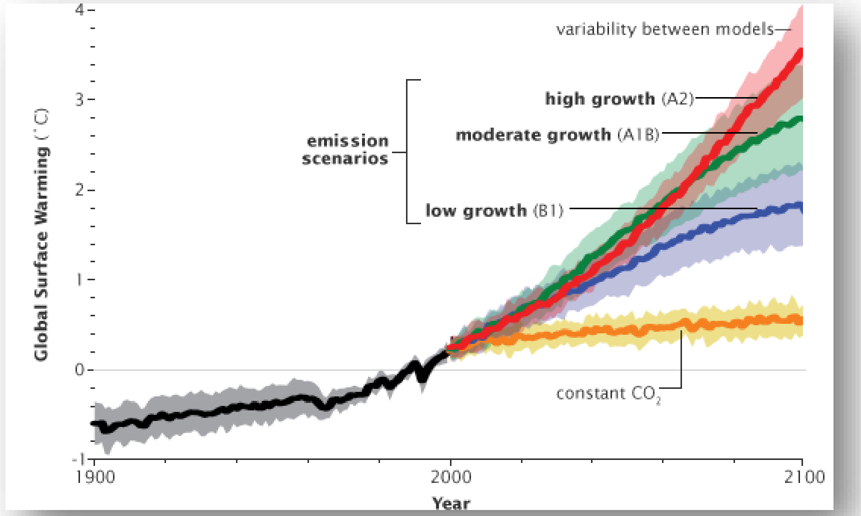
Reinforcement Learning is used to solve two classes of problem

The Prediction Problem

- Aims to learn what value a particular state has $V_t(s)$ or the value of an action when taken from a state $V_t(s,a)$ usually given some example policy.
- The values learnt are not used to decide on what the agent will do – just to predict what the outcome will be for different policies
- For instance, predicting global warming given different potential policies.

The Control Problem

- Aims to learn a control policy that identifies the best action to take given a particular state $\pi_t(s,a)$.
- Used in decision-making tasks
- Such methods need to continually update and improve the policy based on past experiences.
- For instance, controlling a robot for picking up rubbish.



https://earthobservatory.nasa.gov/ContentFeature/GlobalWarming/images/ipcc_scenarios.png
https://robotschampion.com/wp-content/uploads/2018/10/micropsi-industries_technology_02_micropsi_technologies_welcome_ai.jpg

A Simple Maze Example (1)

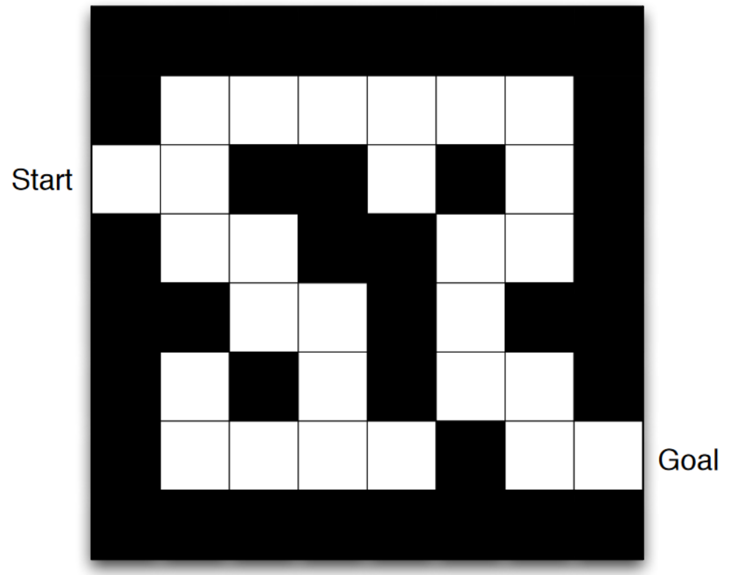
Episodic task. Using a policy provided the aim is to learn a prediction of the expected value for each state.

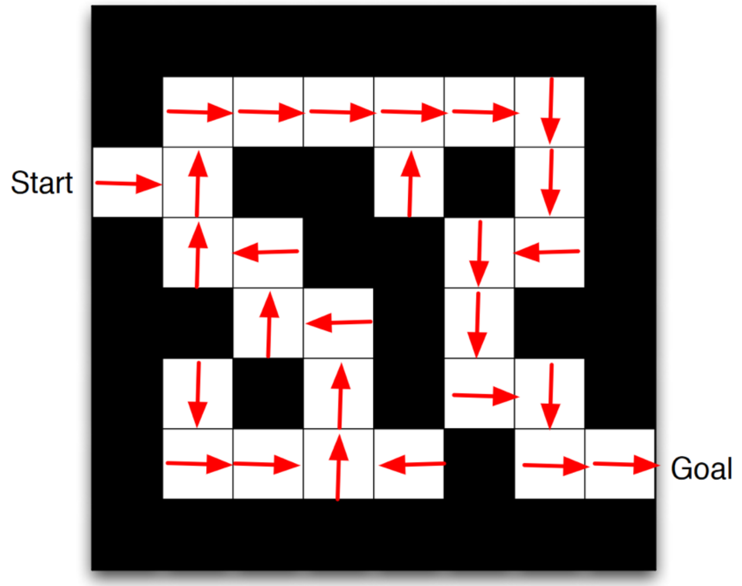
Reward: -1 per time-step

Actions: up, down, left, right

State: agent’s location

Optimal Policy $\pi^*(s,a)$ represented for each state s with a red arrow.

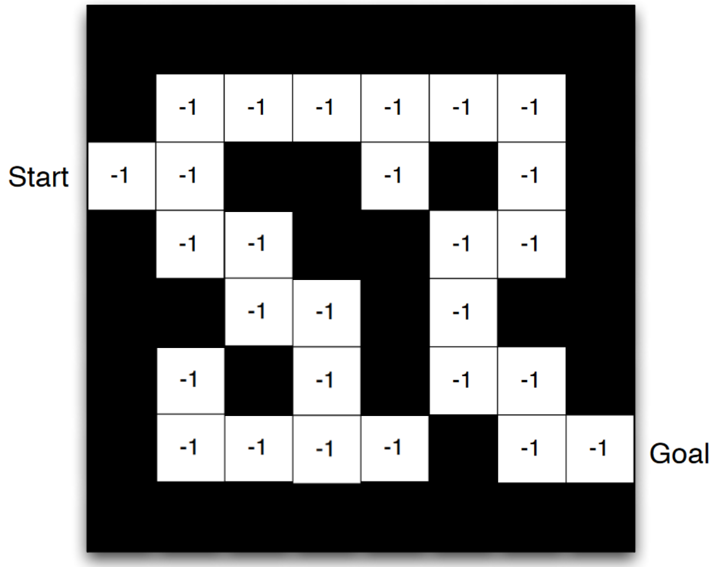


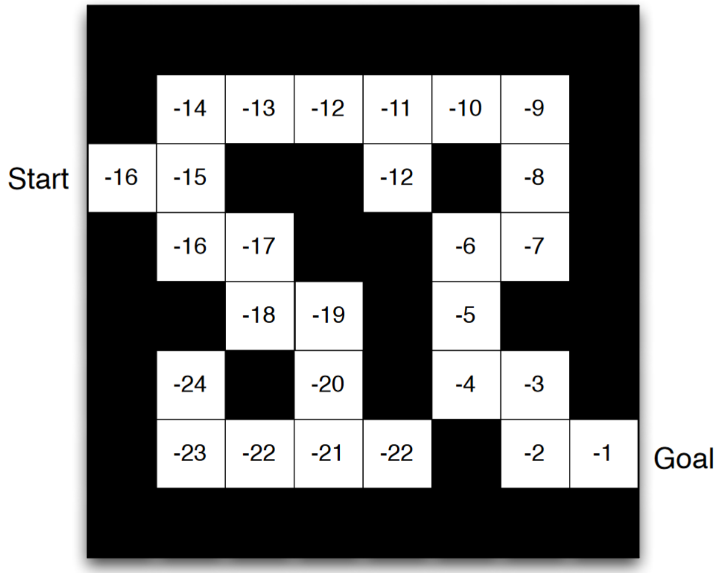


A Simple Maze Example (2)

Reward Function: defines the amount of immediate reward.

Value Function: Stores a number representing the value (expected sum of rewards under a policy) for each state, $v_\pi(s)$





Readings

This was a quick overview of Reinforcement Learning.

- Intention was to introduce the main terminology and the RL learning process.
- Following topics will delve deeper into the topics discussed here.
- Ensure you understand what was discussed here before doing the following topics

For more detailed information see Sutton and Barto (2018) Reinforcement Learning: An Introduction

- *Chapter 1: Introduction*
- http://incompleteideas.net/book/RLbook2020.pdf