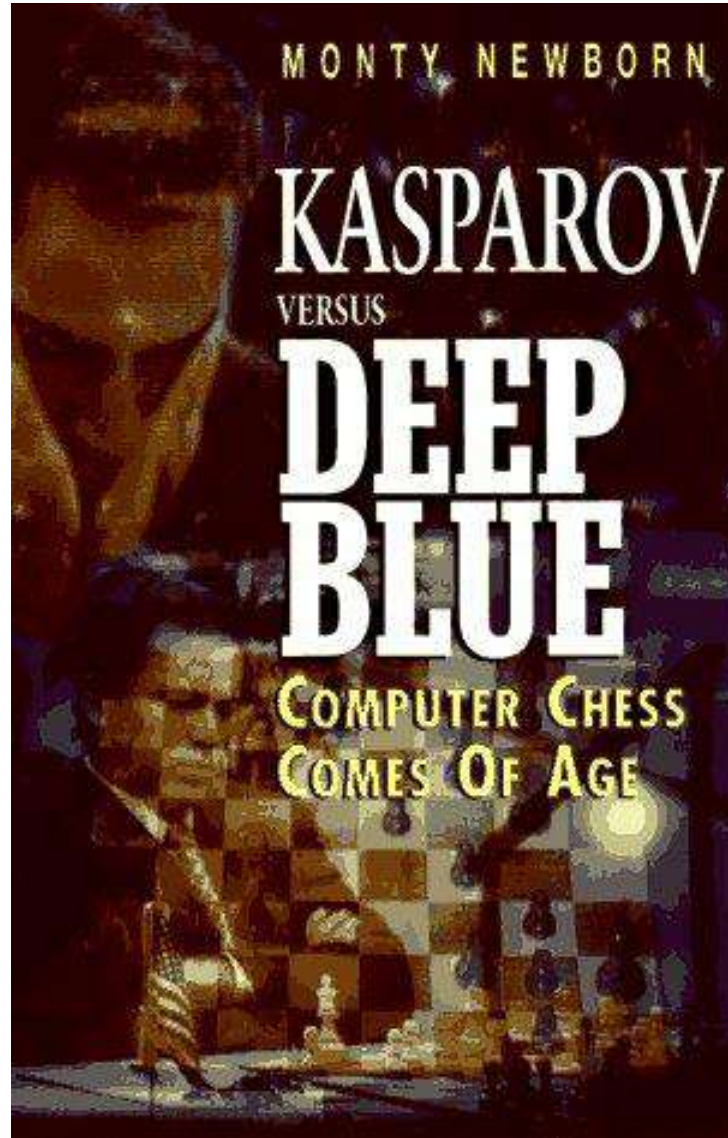


# AI

# State of the Art

“I could feel — I  
could smell — a new  
kind of intelligence  
across the table”  
-Gary Kasparov



Saying Deep Blue  
doesn't really think about  
chess is like saying an  
airplane doesn't really fly  
because it doesn't flap  
its wings.

— Drew McDermott

# Shuttle Repair Scheduling



# Deep Space One

A composite image featuring the Deep Space One spacecraft on the right, which has a large circular antenna and various instruments. To its left is a large, irregularly shaped asteroid. In the upper left corner, there is a bright, glowing sun or star. The background is a dark space filled with stars.

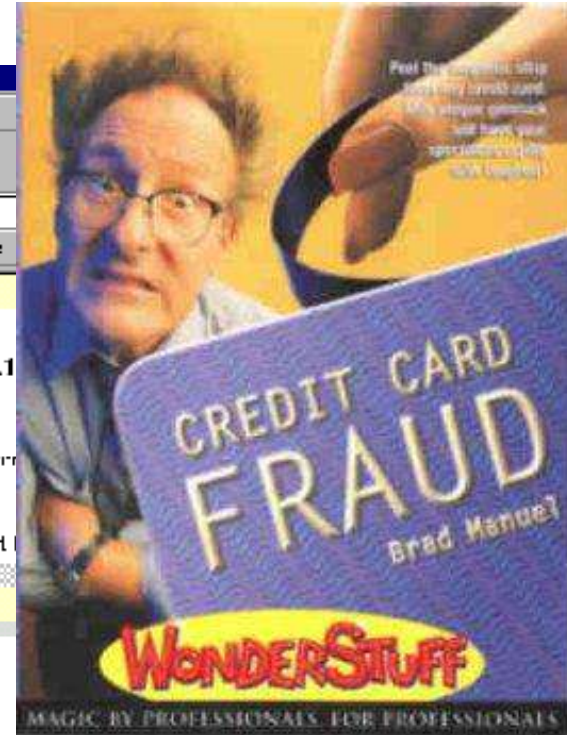
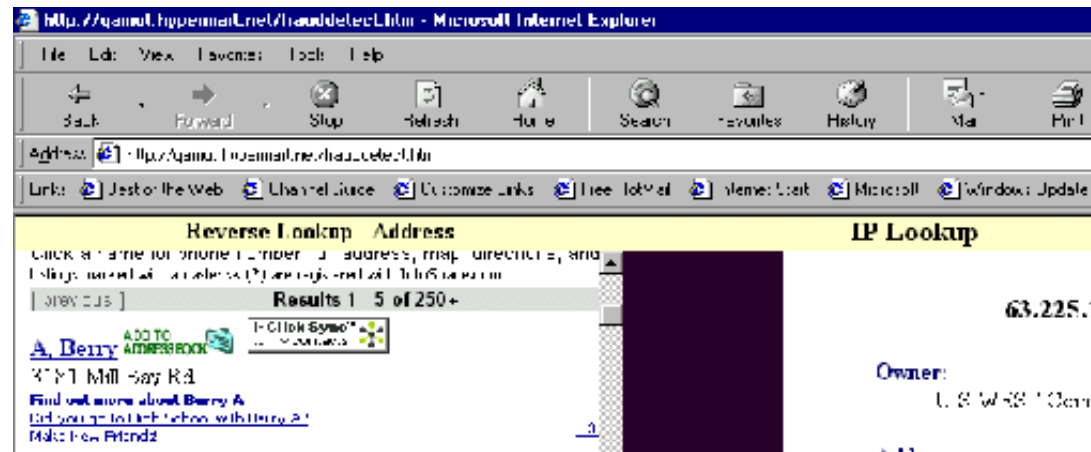
Started: January 1996  
Launch: October 15th, 1998  
Experiment: May 17-21



# Europa Mission ~ 2018



# Credit Card Fraud Detection





# Speech Recognition

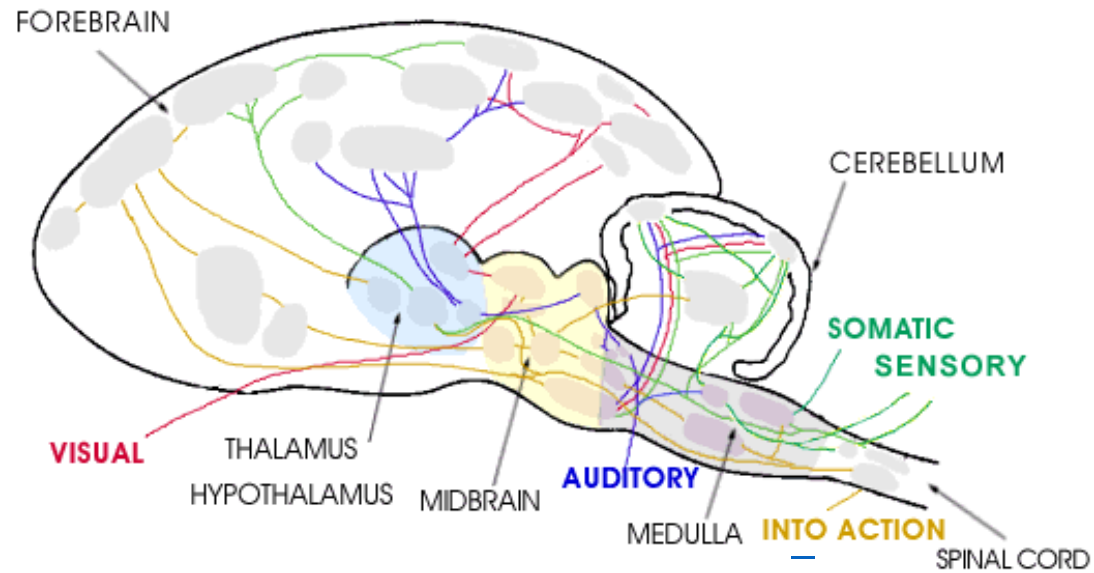


# Autonomous Navigation: NAVLAB 1



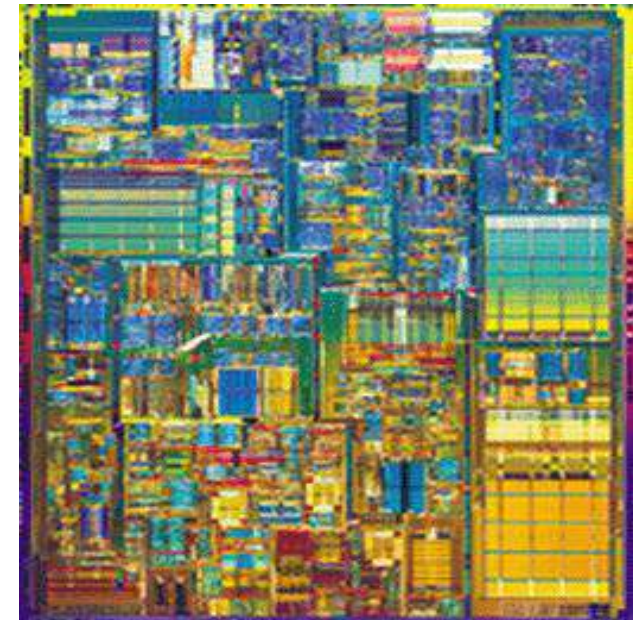


# Hardware



$10^{11}$  neurons  
 $10^{14}$  synapses  
cycle time:  $10^{-3}$  sec

$10^7$  transistors  
 $10^{10}$  bits of RAM  
cycle time:  $10^{-9}$  sec



# What Is Artificial Intelligence (AI)?

- **Artificial intelligence (AI) is the intelligence of a machine or computer that enables it to imitate or mimic human capabilities.**
- AI uses multiple technologies that equip machines to sense, comprehend, plan, act, and learn with human-like levels of intelligence. Fundamentally, AI systems perceive environments, recognize objects, contribute to decision making, solve complex problems, learn from past experiences, and imitate patterns. These abilities are combined to accomplish tasks like driving a car or recognizing faces to unlock device screens.

# Dimensions of the AI Definition

human-like vs. rational

thought  
vs.  
behavior

Systems that think like humans	Systems that think rationally
Systems that act like humans	Systems that act rationally

- **AI Definition:** Artificial Intelligence (AI) refers to the development of computer systems that can perform tasks that typically require human intelligence. These tasks include learning, reasoning, problem-solving, perception, language understanding, and decision-making.
- **Acting Humanly: The Turing Test Approach:** This approach, proposed by Alan Turing in 1950, evaluates a machine's intelligence based on its ability to exhibit behavior indistinguishable from that of a human. If a machine can engage in natural language conversation and convince a human evaluator that it is human, it passes the Turing Test.
- **Thinking Humanly: The Cognitive Modeling Approach:** This approach involves creating computer programs that mimic the cognitive processes observed in humans. Researchers study human cognition through psychology and neuroscience to develop models that simulate human thinking and problem-solving.
- **Thinking Rationally: The "Laws of Thought" Approach:** This approach, inspired by the work of philosophers like Aristotle, focuses on developing AI systems based on principles of formal logic and reasoning. It emphasizes deductive reasoning and logical inference to arrive at rational conclusions.
- **Acting Rationally: The Rational Agent Approach:** In this approach, an AI agent is considered rational if it selects actions that maximize its expected utility or performance measure, given its knowledge and goals. It doesn't necessarily need to mimic human behavior but must make decisions that lead to optimal outcomes in its environment.



# Turing Test

Player A is a computer, Player B is human, and Player C is an interrogator. Interrogator is aware that one of them is machine, but he needs to identify this on the basis of questions and their responses.

The questions and answers can be like:

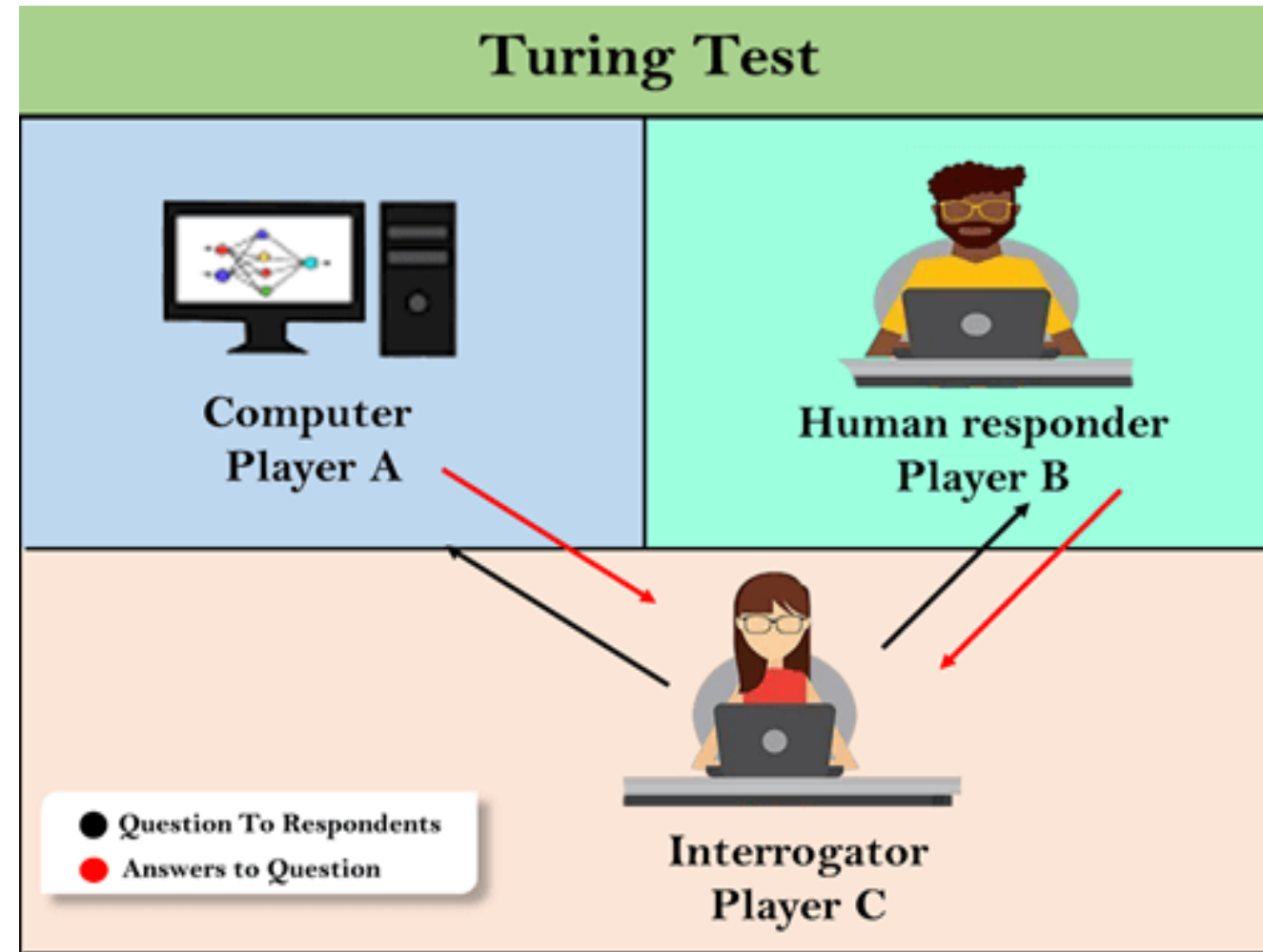
**Interrogator:** Are you a computer?

**PlayerA (Computer):** No

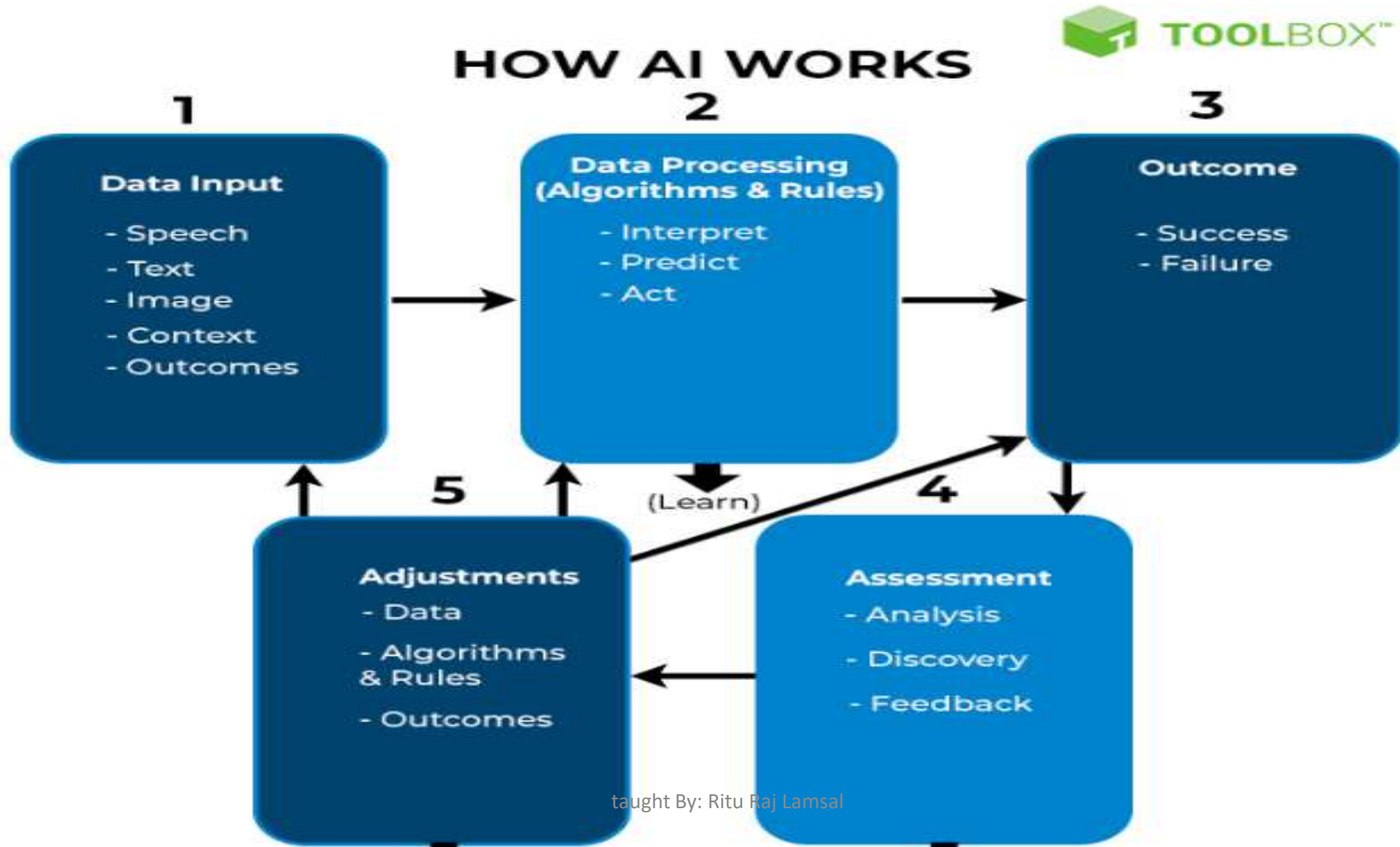
**Interrogator:** Multiply two large numbers such as  $(256896489 * 456725896)$

**Player A:** Long pause and give the wrong answer.

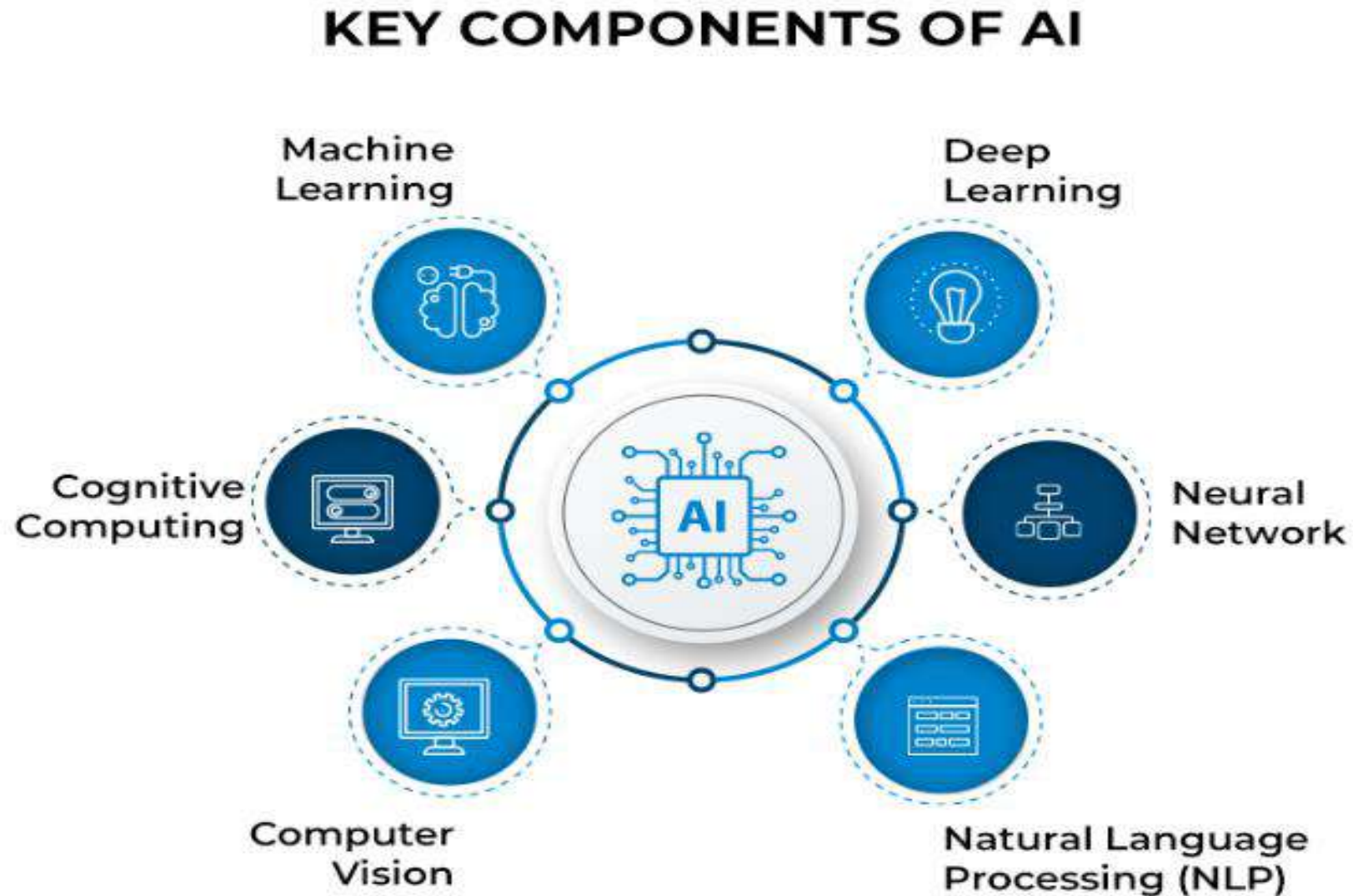
In this game, if an interrogator would not be able to identify which is a machine and which is human, then the computer passes the test successfully, and the machine is said to be intelligent and can think like a human.



# How AI works



# Key Components of AI



# Key Components of AI

1. **Machine learning:** Machine learning is an AI application that automatically learns and improves from previous sets of experiences without the requirement for explicit programming.
2. **Deep learning:** Deep learning is a subset of ML that learns by processing data with the help of artificial neural networks.
3. **Neural network:** [Neural networks](#) are computer systems that are loosely modeled on neural connections in the human brain and enable deep learning.
4. **Cognitive computing:** Cognitive computing aims to recreate the human thought process in a computer model. It seeks to imitate and improve the interaction between humans and machines by understanding human language and the meaning of images.
5. **Natural language processing (NLP):** NLP is a tool that allows computers to comprehend, recognize, interpret, and produce human language and speech.
6. **Computer vision:** Computer vision employs deep learning and pattern identification to interpret image content (graphs, tables, PDF pictures, and videos).



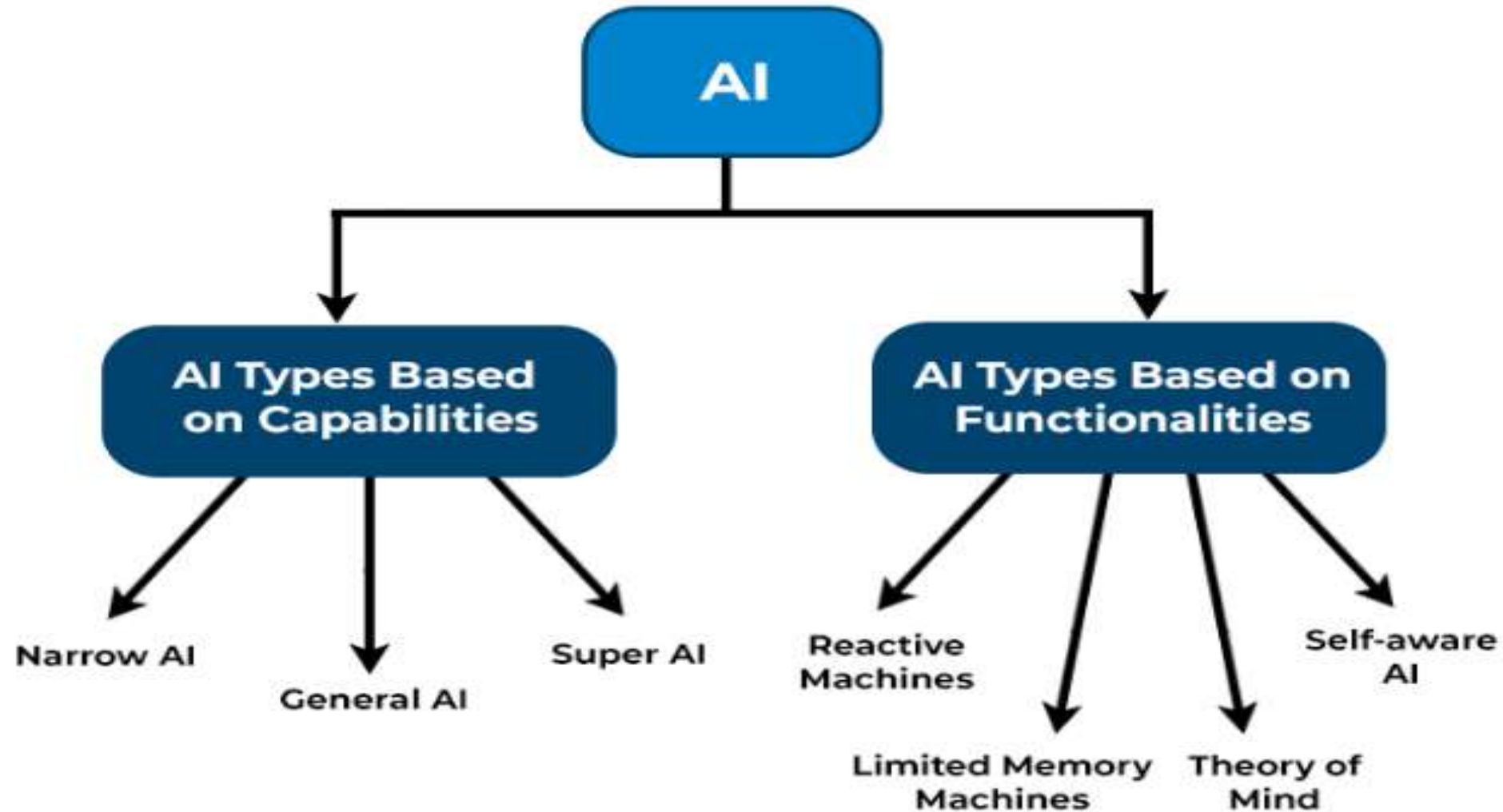
# ***The Foundations of Artificial Intelligence***

- **Philosophy:** Philosophical inquiries into the nature of intelligence and reasoning have laid the groundwork for AI. Early philosophers like Aristotle contemplated concepts of logic and reasoning that later influenced AI development.
- **Mathematics:** Mathematical concepts such as algorithms, probability theory, and formal logic form the basis of AI algorithms and models. Mathematicians like Alan Turing and George Boole made significant contributions to the theoretical foundations of AI.
- **Economics:** Economic principles like decision theory and game theory are applied in AI for reasoning under uncertainty and designing autonomous agents that interact in complex environments.
- **Neuroscience:** Understanding the structure and function of the brain has inspired AI researchers to develop neural networks and computational models of neural activity, leading to advances in machine learning and cognitive science.
- **Psychology:** Psychological theories of learning, memory, perception, and problem-solving have informed the design of AI algorithms and cognitive architectures.
- **Computer Engineering:** The development of digital computers and programming languages has provided the technological infrastructure for implementing AI algorithms and building intelligent systems.
- **Control Theory and Cybernetics:** Principles of feedback control and system dynamics from control theory and cybernetics are applied in AI for designing adaptive and self-regulating systems.
- **Linguistics:** Linguistic theories of syntax, semantics, and pragmatics contribute to natural language processing and understanding in AI systems.

# *The History of Artificial Intelligence*

- **The Gestation of Artificial Intelligence (1943-1955)**: The groundwork for AI was laid during this period, with early developments like McCulloch and Pitts' model of artificial neurons and Alan Turing's work on computability.
- **The Birth of Artificial Intelligence (1956)**: The term "Artificial Intelligence" was coined, and the field was officially established during the Dartmouth Conference, where researchers discussed the potential of creating machines with human-like intelligence.
- **Early Enthusiasm, Great Expectations (1952-1969)**: During this period, researchers explored symbolic AI and developed programs capable of logical reasoning and problem-solving, including the Logic Theorist and General Problem Solver.
- **A Dose of Reality (1966-1973)**: Optimism in AI waned as researchers encountered challenges in natural language understanding, vision, and common-sense reasoning. Funding for AI research decreased during this "AI winter."
- **Knowledge-based Systems: The Key to Power? (1969-1979)**: The focus shifted to knowledge-based systems and expert systems, which used symbolic representations of knowledge to solve specific problems in medicine, engineering, and other domains.
- **AI Becomes an Industry (1980-Present)**: AI technologies began to be commercialized, with applications in areas such as speech recognition, robotics, and financial forecasting.
- **The Return of Neural Networks (1986-Present)**: Neural networks experienced a resurgence in popularity with the development of backpropagation algorithm and increased computational power, leading to breakthroughs in pattern recognition and machine learning.
- **AI Becomes a Science (1987-Present)**: AI research became more interdisciplinary, drawing from fields like cognitive science, linguistics, and neuroscience to develop more human-like and intelligent systems.
- **The Emergence of Intelligent Agents (1995-Present)**: The focus shifted to designing intelligent agents capable of autonomous decision-making and interacting with complex environments, leading to developments in reinforcement learning and multi-agent systems.

# Types of AI



# Types Of AI

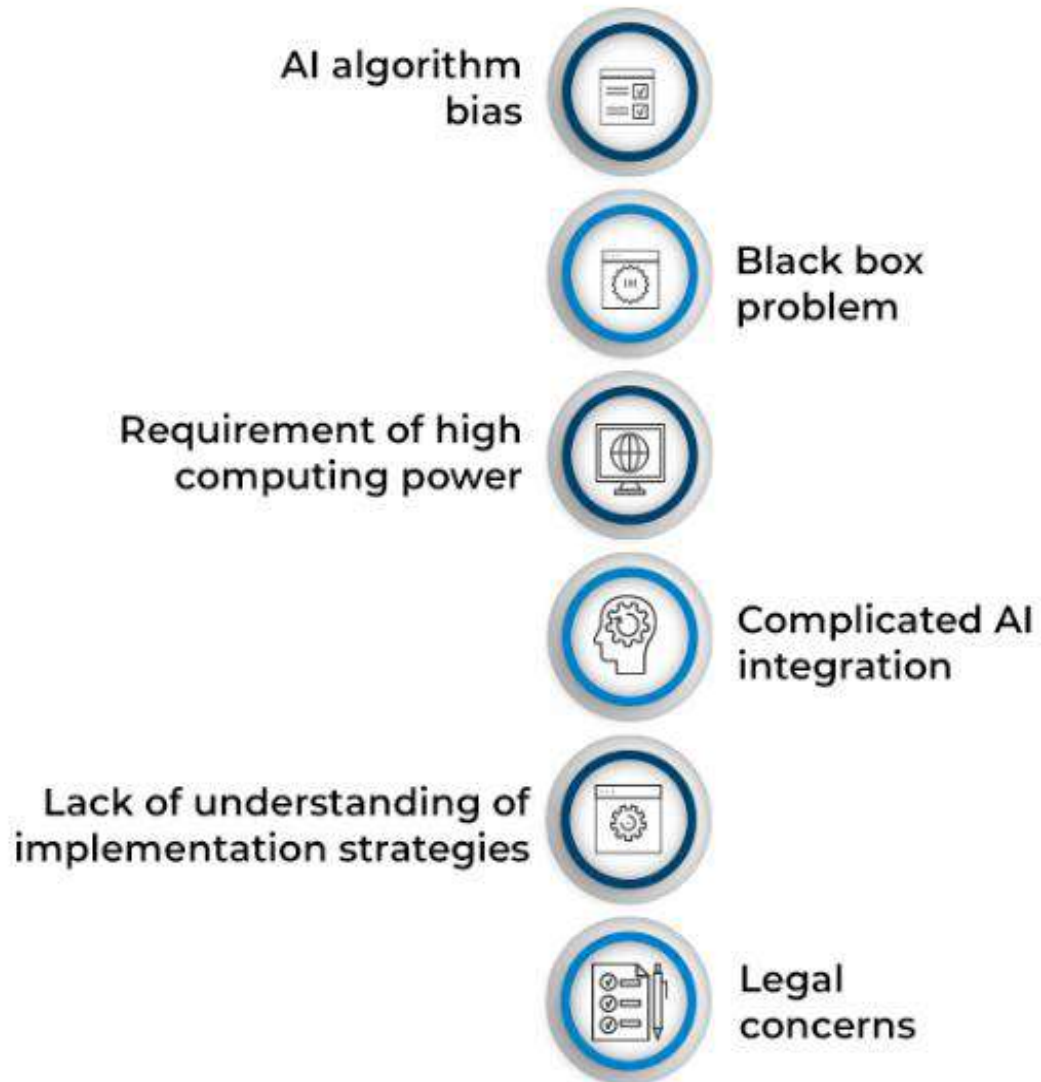
- 1. **Narrow/Weak AI**
  - Narrow AI is a goal-oriented AI trained to perform a specific task. The machine intelligence that we witness all around us today is a form of narrow AI. Examples of narrow AI include Apple's Siri and IBM's Watson supercomputer.
  - Narrow AI is also referred to as weak AI as it operates within a limited and pre-defined set of parameters, constraints, and contexts. For example, use cases such as Netflix recommendations, purchase suggestions on ecommerce sites, autonomous cars, and speech & image recognition fall under the narrow AI category.
- 2. **General AI(Strong AI)**
  - General AI is an AI version that performs any intellectual task with a human-like efficiency. The objective of general AI is to design a system capable of thinking for itself just like humans do. Currently, general AI is still under research, and efforts are being made to develop machines that have enhanced cognitive capabilities.
- 3. **Super AI**
  - Super AI is the AI version that surpasses human intelligence and can perform any task better than a human. Capabilities of a machine with super AI include thinking, reasoning, solving a puzzle, making judgments, learning, and communicating on its own. Today, super AI is a hypothetical concept but represents the future of AI.



# Goals of AI



## KEY CHALLENGES OF AI



### Key Challenges of AI

Bias refers to racial, gender, communal, or ethnic bias. For example, today's algorithms determine candidates suitable for a job interview or individuals eligible for a loan. If the algorithms making such vital decisions have developed biases over time, it could lead to dreadful, unfair, and unethical consequences.

AI algorithms are like black boxes. We have very little understanding of the inner workings of an AI algorithm. For example, we can understand what the prediction is for a predicting system, but we lack the knowledge of how the system arrived at that prediction. This makes AI systems slightly unreliable.

# Trends in AI

## TOP 5 AI TRENDS

- 1** Computer vision set to grow 
- 2** Boost to autonomous vehicle industry 
- 3** Chatbots and virtual assistants to get smarter 
- 4** Solutions for metaverse 
- 5** Improved language modeling 

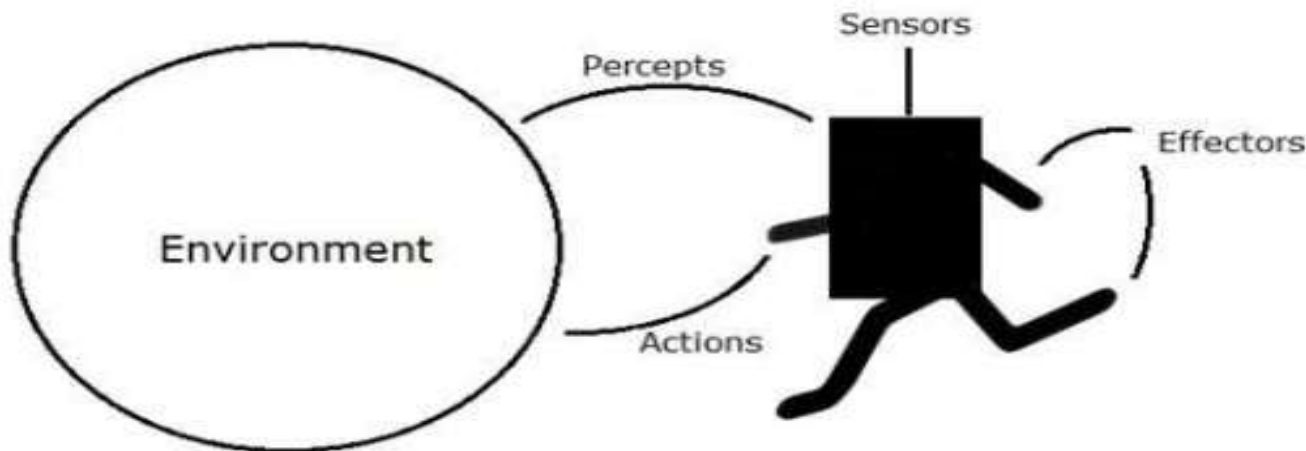




# Agents and Environment

An **agent** is anything that can perceive its environment through **sensors** and acts upon that environment through **effectors**.

- A **human agent** has sensory organs such as eyes, ears, nose, tongue and skin parallel to the sensors, and other organs such as hands, legs, mouth, for effectors.
- A **robotic agent** replaces cameras and infrared range finders for the sensors, and various motors and actuators for effectors.
- A **software agent** has encoded bit strings as its programs and actions.



## Intelligent Agents

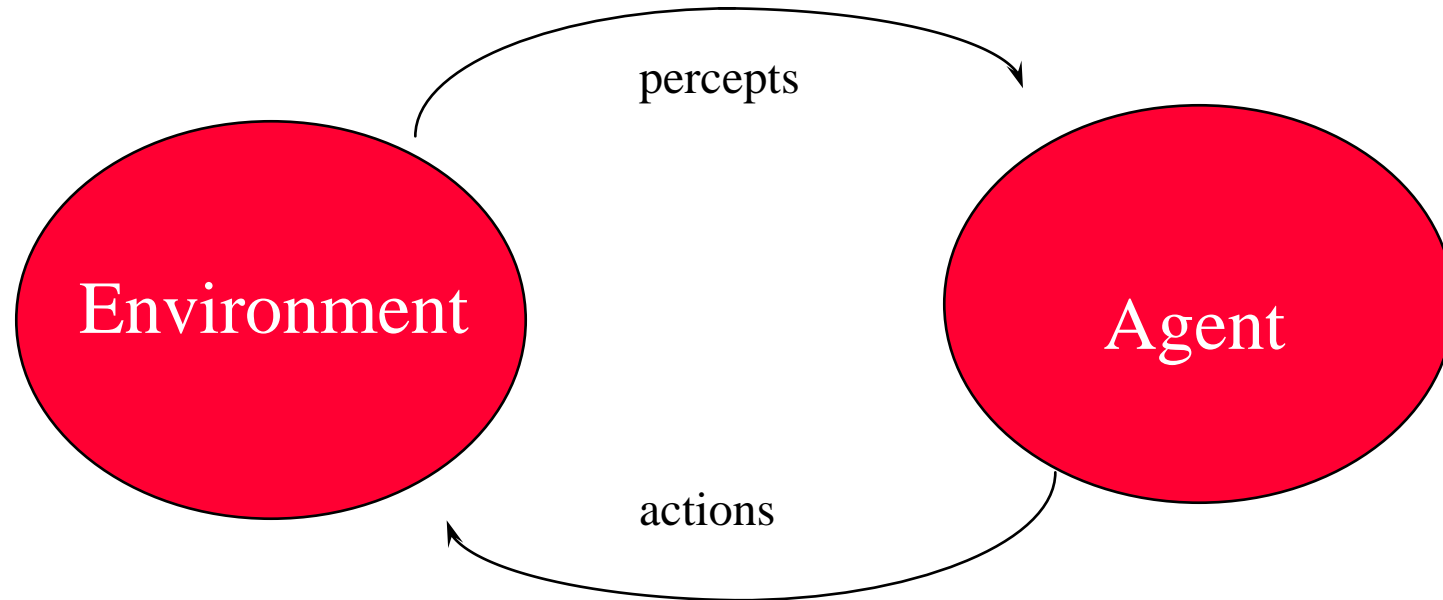
- Intelligent agents are entities capable of perceiving their environment through sensors, processing information, and taking actions to achieve specific goals.
- These agents can be human, robotic, or software-based.

# Agents and Environments

- Definition of Agents and Environments: An agent is any entity capable of perceiving its environment through sensors and acting upon that environment through actuators. The environment encompasses everything outside the agent that can be perceived and affected.
- Interactions between Agents and Environments: Agents interact with their environments by receiving sensory inputs, processing information, and executing actions. This interaction is crucial for the agent to achieve its goals effectively.

# Intelligent Agents

- Have sensors, effectors
- Implement mapping from percept sequence to actions



- Performance Measure

# Description of Agent:

- An agent is an autonomous entity that perceives its environment through sensors, processes this information, and takes actions using actuators to achieve its goals or objectives.
- It operates based on predefined rules, algorithms, or learning mechanisms to make decisions and adapt to changes in its environment.
- Agents can range from simple systems like thermostat controllers to complex AI systems like autonomous vehicles or intelligent virtual assistants.
- The design and effectiveness of an agent depend on its performance measure, the characteristics of its environment, and the capabilities of its actuators and sensors.

- 
- **Performance Measures**: Performance measures define the criteria for evaluating the success of an agent's actions. They are used to assess how well an agent achieves its goals within its environment.
- **Rationality**: Rationality refers to the ability of an agent to select actions that maximize its expected utility or performance measure, given its knowledge and goals. A rational agent makes decisions that lead to optimal outcomes.
- **Omniscience, Learning, and Autonomy**: Rational agents are often constrained by limitations such as incomplete knowledge, the need for learning from experience, and the autonomy to make decisions independently.

## **Rationality**

What is rational at any given time depends on four things:

- The performance measure that defines the criterion of success.
- The agent's prior knowledge of the environment.
- The actions that the agent can perform.
- The agent's percept sequence to date.



# The Nature of Environments

- Task Environment: The task environment defines the context in which an agent operates and includes factors such as the set of possible states, actions, and outcomes.

## Properties of Task Environments:

- Task environments can vary in observability (fully observable or partially observable),
- determinism (deterministic or stochastic),
- dynamicity (static or dynamic),
- and episodicity (episodic or sequential).

# PEAS Description of an Agent

## Performance Measure:

- The criteria used to evaluate the success or efficiency of the agent in achieving its goals within the environment.
- It defines what the agent aims to accomplish and how well it accomplishes it.
- Example: For a self-driving car, the performance measure could be reaching the destination safely and efficiently while obeying traffic laws and minimizing travel time.

## Environment:

- The external context or surroundings in which the agent operates and interacts.
- It includes all the relevant factors and entities that can influence the agent's actions and perceptions.
- Example: The environment for a vacuum-cleaning robot includes the rooms, furniture, obstacles, and dust on the floor.

## Actuators:

- The mechanisms or components through which the agent can affect its environment or perform actions.
- Actuators enable the agent to execute its decisions and manipulate the environment.
- Example: Actuators for a robot could include motors, wheels, arms, grippers, or any other physical devices for movement or manipulation.

## Sensors:

- The devices or sensors that enable the agent to perceive and gather information about its environment.
- Sensors provide input to the agent, allowing it to observe, interpret, and respond to changes in the environment.
- Example: Sensors for a robot might include cameras, infrared sensors, lidar, or touch sensors to detect obstacles, measure distances, or recognize objects.

# PEAS Example

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe: fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

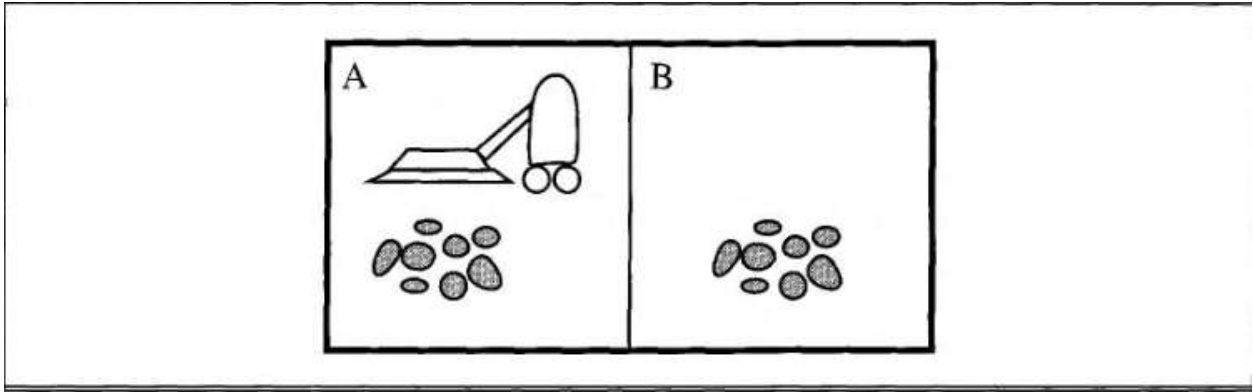
**Figure 2.4** PEAS description of the task environment for an automated taxi.

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, minimize costs, lawsuits	Patient, hospital, staff	Display questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display categorization of scene	Color pixel arrays
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Maximize purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Maximize student's score on test	Set of students, testing agency	Display exercises, suggestions, corrections	Keyboard entry

**Figure 2.5** Examples of agent types and their PEAS descriptions.



**Vacuum World Agent** : This particular world has just two locations: squares A and B. The vacuum agent perceives which square it is in and whether there is dirt in the square. It can choose to move left, move right, suck up the dirt, or do nothing. One very simple agent function is the following: if the current square is dirty, then suck, otherwise move to the other square. A partial tabulation of this agent function is shown in Figure 2.3.



**Figure 2.2** A vacuum-cleaner world with just two locations.

Percept sequence	Action
[A,Clean]	Right
[A,Dirty]	Suck
[B,Clean]	Left
[B,Dirty]	Suck
[A,Clean],[A,Clean]	Right
[A,Clean],[A,Dirty]	Suck
[A,Clean],[A,Clean],[A,Clean]	Right
[A,Clean],[A,Clean],[A,Dirty]	Suck

**Figure 2.3** Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.

# Simple reflex agents

The simplest kind of agent is the simple reflex agent. These agents select actions on the basis of the current percept, ignoring the rest of the percept history. For example, the vacuum agent whose agent function is tabulated in Figure 2.3 is a simple reflex agent, because its decision is based only on the current location and on whether that contains dirt. An agent program for this agent is shown in Figure 2.8.

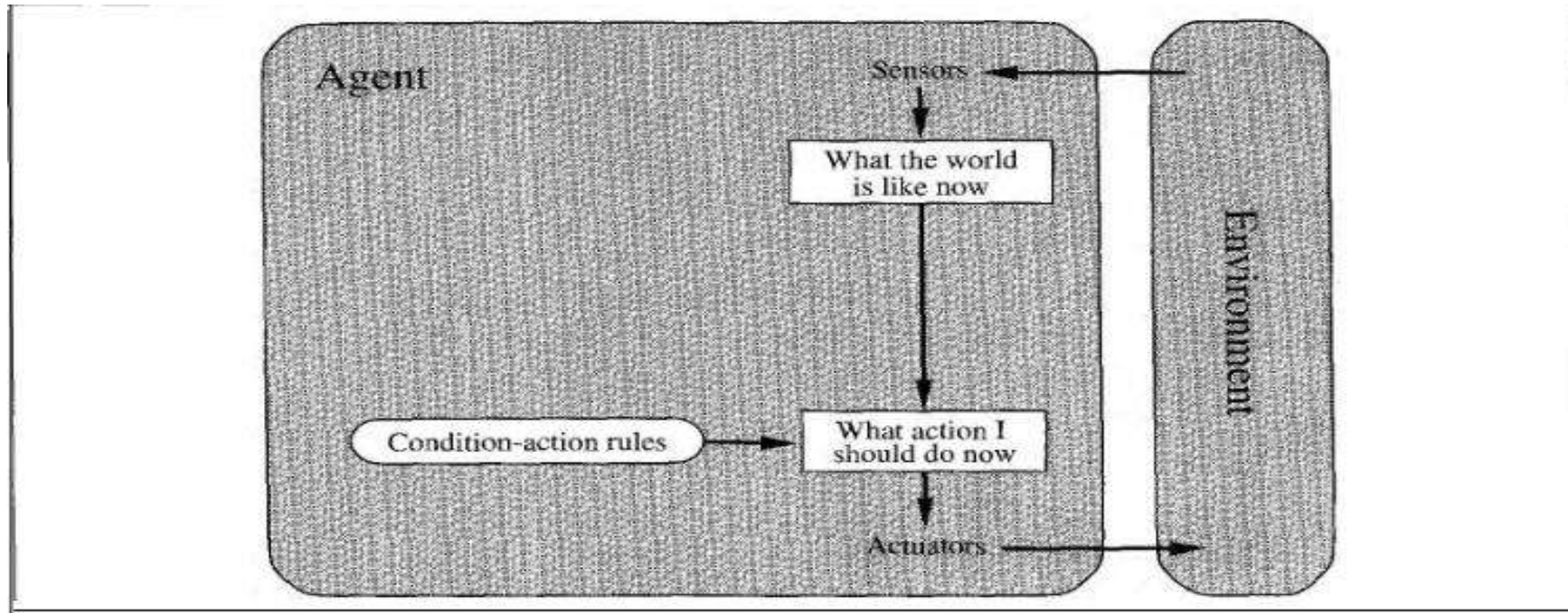
```
function REFLEX-VACUUM-AGENT([location, status]) returns an action
```

```
  if status = Dirty then return Suck  
  else if location = A then return Right  
  else if location = B then return Left
```

**Figure 2.8** The agent program for a simple reflex agent in the two-state vacuum environment. This program implements the agent function tabulated in Figure 2.3.

Imagine yourself as the driver of the automated taxi. If the car in front brakes, and its brake lights come on, then you should notice this and initiate braking. In other words, some processing is done on the visual input to establish the condition we call "The car in front is braking."

# Simple reflex Agent



**Figure 2.9** Schematic diagram of a simple reflex agent.

```
function SIMPLE-REFLEX-AGENT(percept) returns an action
static: rules, a set of condition–action rules

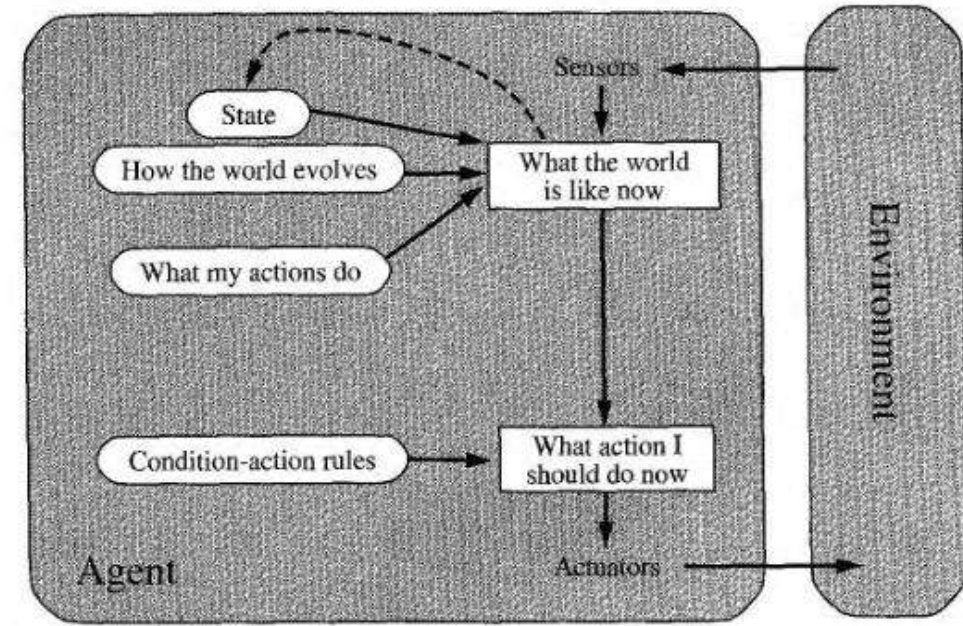
state ← INTERPRET-INPUT(percept)
rule ← RULE-MATCH(state, rules)
action ← RULE-ACTION[rule]
return action
```

**Figure 2.10** A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.



# Model based Reflex Agent

A model-based reflex agent. It keeps track of the current state of the world using an internal model. It then chooses an action in the same way as the reflex agent.



**Figure 2.11** A model-based reflex agent.

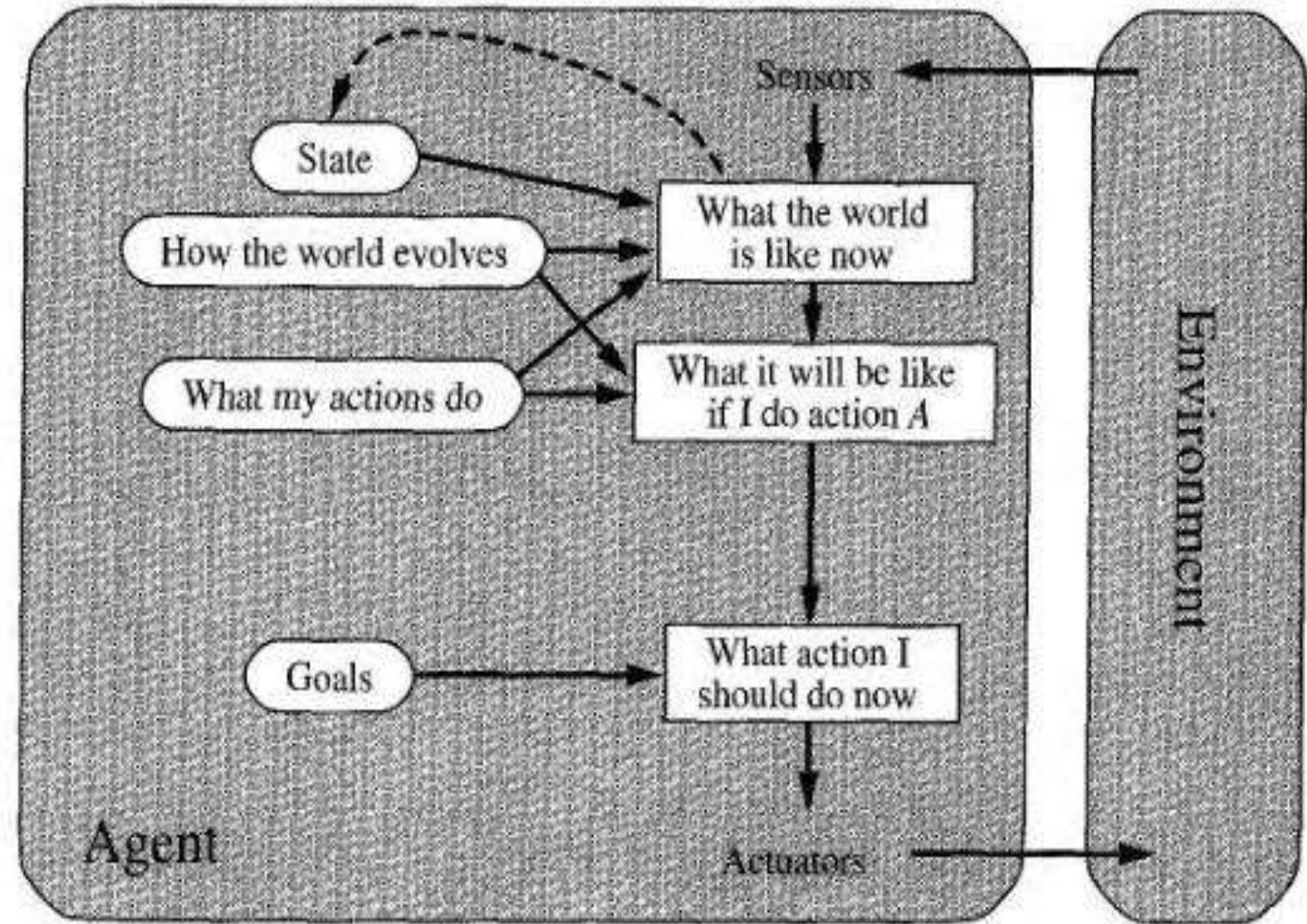
```
function REFLEX-AGENT-WITH-STATE(percept) returns an action
static: state, a description of the current world state
         rules, a set of condition-action rules
         action, the most recent action, initially none

state ← UPDATE-STATE(state, action, percept)
rule ← RULE-MATCH(state, rules)
action ← RULE-ACTION[rule]
return action
```

**Figure 2.12** A model-based reflex agent. It keeps track of the current state of the world using an internal model. It then chooses an action in the same way as the reflex agent.

# model-based, goal - based agent

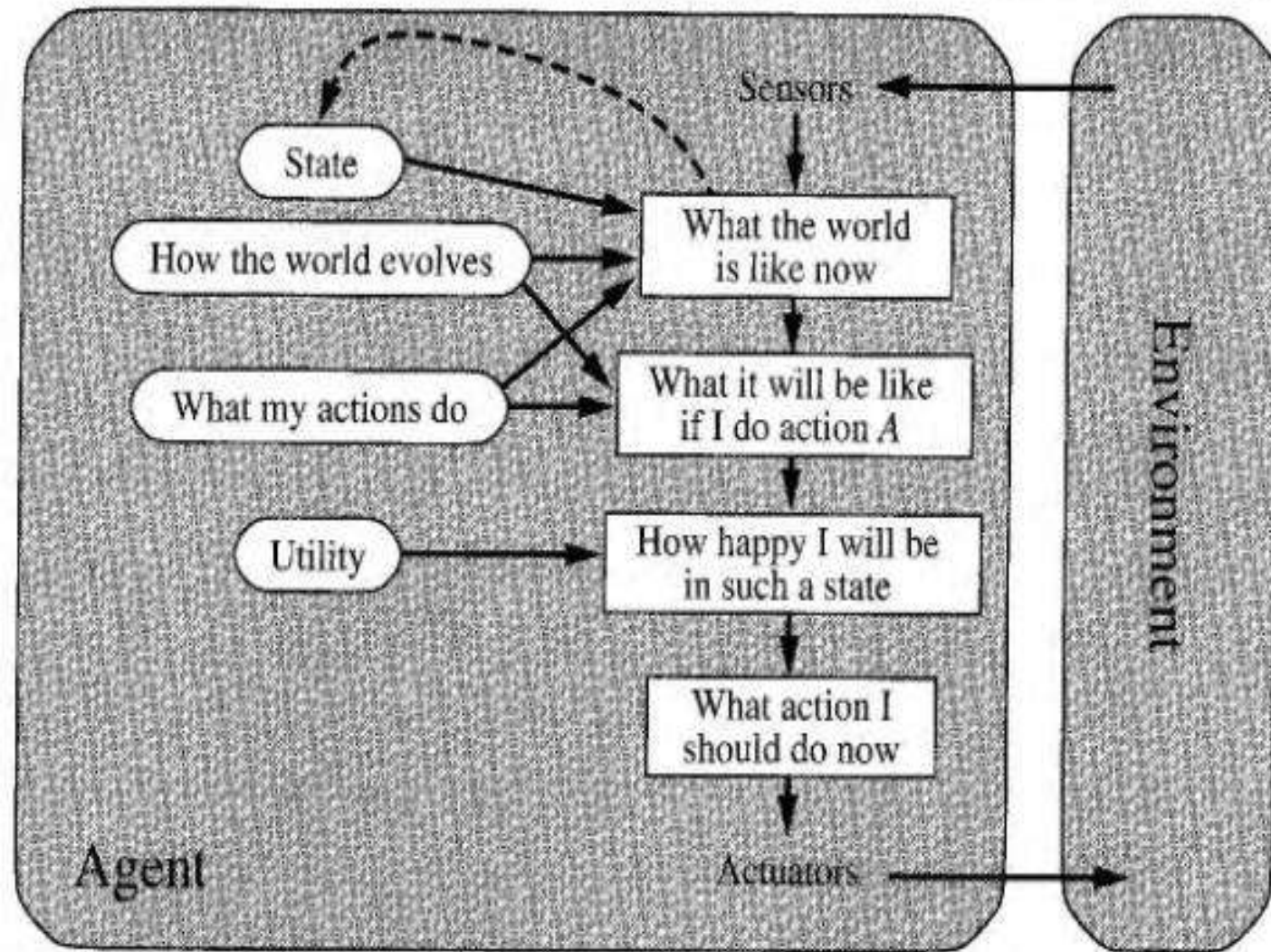
- A model-based, goal - based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals



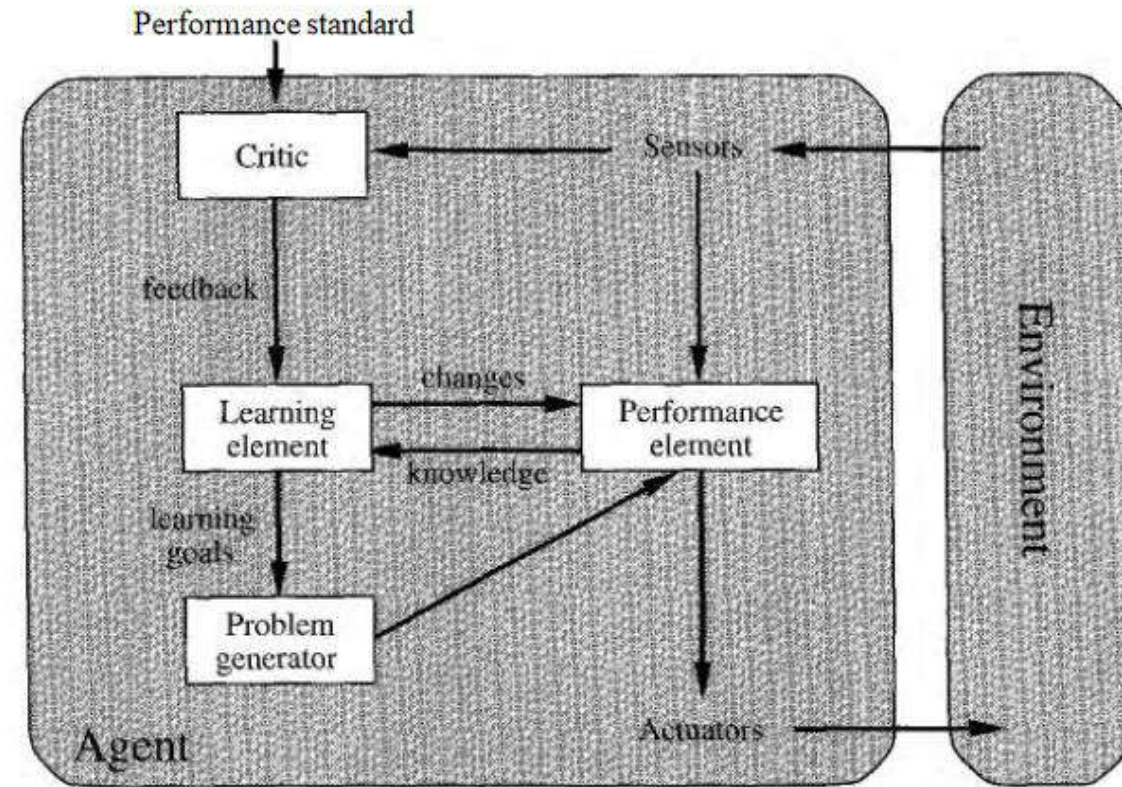


# model-based, utility-based agent.

- A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome



# General Model of learning Agent



**Figure 2.15** A general model of learning agents.



# Summary

- An **agent** is something that perceives and acts in an environment. The **agent function** for an agent specifies the action taken by the agent in response to any percept sequence.
- The **performance measure** evaluates the behavior of the agent in an environment. A **rational agent** acts so as to maximize the expected value of the performance measure, given the percept sequence it has seen so far.
- A **task environment** specification includes the performance measure, the external environment, the actuators, and the sensors. In designing an agent, the first step must always be to specify the task environment as fully as possible.
- Task environments vary along several significant dimensions. They can be fully or partially observable, deterministic or stochastic, episodic or sequential, static or dynamic, discrete or continuous, and single-agent or multiagent.
- The **agent program** implements the agent function. There exists a variety of basic agent-program designs, reflecting the kind of information made explicit and used in the decision process. The designs vary in efficiency, compactness, and flexibility. The appropriate design of the agent program depends on the nature of the environment.
- **Simple reflex agents** respond directly to percepts, whereas **model-based reflex agents** maintain internal state to track aspects of the world that are not evident in the current percept. **Goal-based agents** act to achieve their goals, and **utility-based agents** try to maximize their own expected "happiness."
- All agents can improve their performance through **learning**.







## Searching as a Problem Solving Technique

Problem solving by Searching

- **Searching** is the process of looking for the solution of a problem through a set of possibilities (state space).
- **Search** conditions include :
  - **Current state** - where one is;
  - **Goal state** - the solution reached; check whether it has been reached;
  - **Cost** of obtaining the solution.
- The **Solution** is a path from the current state to the goal state.

# Problem Searching -What is Search?

- Search is the systematic examination of states to find path from the start/root state to the goal state. The set of possible states, together with *operators* defining their connectivity constitute the *search space*.
- The output of a search algorithm is a solution, that is, a path from the initial state to a state that satisfies the goal test.
- In real life search usually results from a lack of knowledge. In AI too search is merely a offensive instrument with which to attack problems that we can't seem to solve any better way.
- **Search techniques fall into three groups:**
  - 1.Methods which find *any* start - goal path,
  - 2.Methods which find the *best* path, and finally
  - 3.Search methods in the face of adversaries.
-

# Solving Problems by Searching:

- 1. Problem-solving through searching involves exploring a space of possible solutions systematically until a satisfactory solution is found.
- It involves
  - defining a problem space, initial state, goal state, operators or actions, and a search strategy.
- Various search algorithms like depth-first search, breadth-first search, and A\* search are employed to navigate the problem space efficiently.

# Well Defined Problem

- **Problem Solving by Search**
- An important aspect of intelligence is *goal-based* problem solving. The solution of many problems (e.g. noughts and crosses, timetabling, chess) can be described by finding a sequence of actions that lead to a desirable goal. Each action changes the *state* and the aim is to find the sequence of actions and states that lead from the initial (start) state to a final (goal) state.
- A well-defined problem can be described by:
- **Initial state**
- **Operator or successor function** - for any state  $x$  returns  $s(x)$ , the set of states reachable from  $x$  with one action
- **State space** - all states reachable from initial by any sequence of actions
- **Path** - sequence through state space
- **Path cost** - function that assigns a cost to a path. Cost of a path is the sum of costs of individual actions along the path
- **Goal test** - test to determine if at goal state

## Real-world problems

We have already seen how the route-finding problem is defined in terms of specified locations and transitions along links between them. Route-finding algorithms are used in a variety of applications, such as routing in computer networks, military operations planning, and airline travel planning systems. These problems are typically complex to specify. Consider a simplified example of an airline travel problem specified as follows:

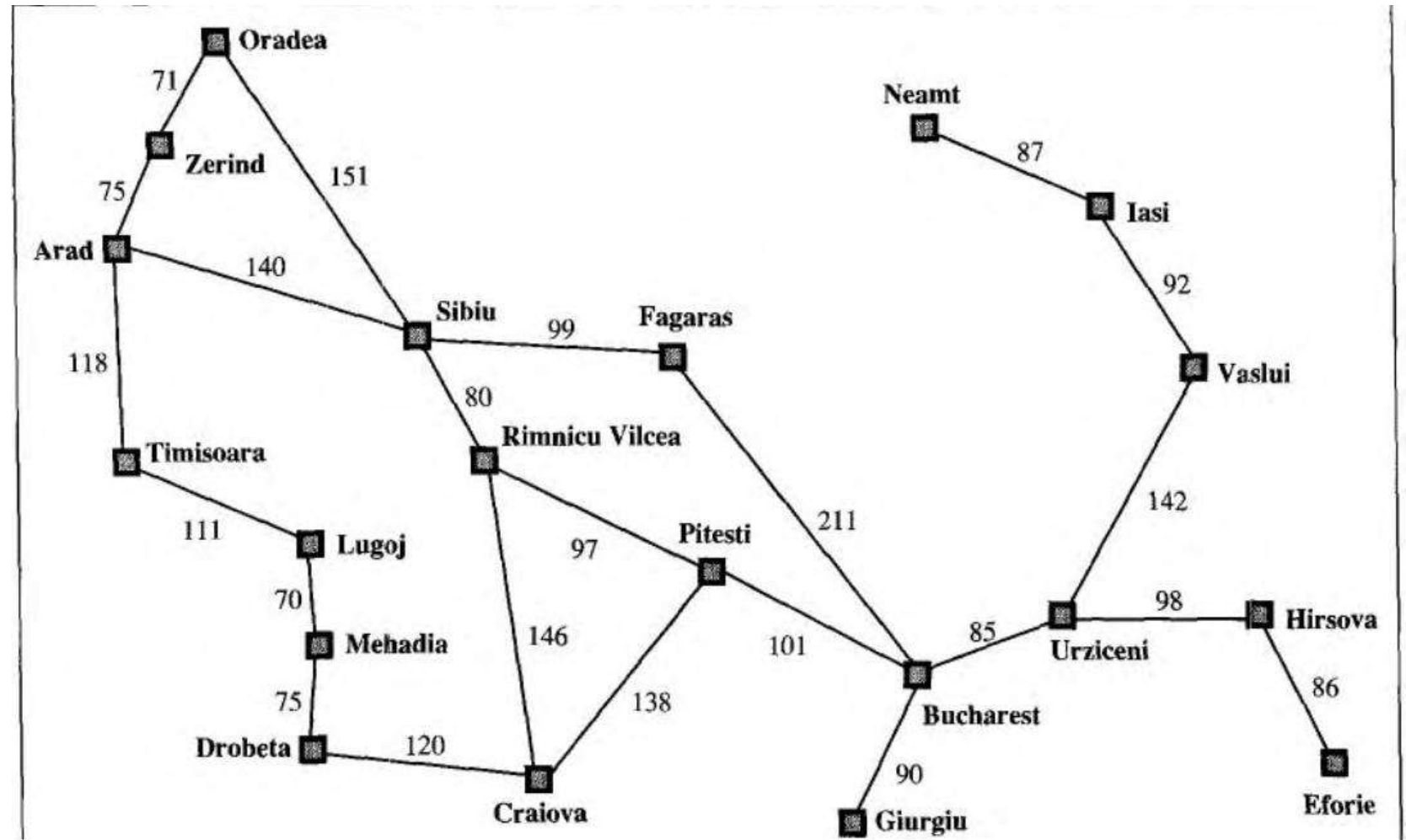
- ◇ States: Each is represented by a location (e.g., an airport) and the current time.
- ◇ Initial state: This is specified by the problem.
- ◇ Successor **function**: This returns the states resulting from taking any scheduled flight (perhaps further specified by seat class and location), leaving later than the current time plus the within-airport transit time, from the current airport to another.
- ◇ Goal test: Are we at the destination by some prespecified time?
- ◇ Path cost: This depends on monetary cost, waiting time, flight time, customs and immigration procedures, seat quality, time of day, type of airplane, frequent-flyer mileage awards, and so on.

# Real World problem: TSP

- Touring problems are closely related to route - finding problems, but with an important difference. The **traveling salesperson problem (TSP)** is a touring problem in which each city must be visited exactly once. The aim is to find the shortest tour.



"Visit every city in Figure 3.2 at least once, starting and ending in Bucharest." As with route finding, the actions correspond to trips between adjacent cities. The state space, however, is quite different. Each state must include not just the current location but also the set of cities the agent has visited. So the initial state would be "In Bucharest; visited {Bucharest}," a typical intermediate state would be "In Vaslui; visited {Bucharest,Urziceni,Vaslui}," and the goal test would check whether the agent is in Bucharest and all 20 cities have been visited



**Figure 3.2** A simplified road map of part of Romania.

# Real World problem

- **A VLSI layout problem** requires positioning millions of components and connections on a chip to minimize area, minimize circuit delays, minimize stray capacitances, and maximize manufacturing yield.
- **Robot navigation** is a generalization of the route- finding problem, Rather than a discrete set of routes, a robot can move in a continuous space with an infinite set of possible actions and states. For a circular robot moving on a flat surface, the space is essentially two- dimensional. When the robot has arms and legs or wheels that must also be controlled, the search space becomes many - dimensional.
- **Automatic assembly** sequencing of complex objects by a robot .Assembly PROTEINDESIGN problem is protein design, in which the goal is to find a sequence of amino acids that will fold into a three -dimensional protein with the right properties to cure some disease.
- increased demand for software robots that perform **Internet searching**, looking for answers to questions, for related information, or for shopping deals.

**function** SIMPLE-PROBLEM-SOLVING-AGENT(*percept*) **returns an** action

**inputs:** *percept*, a percept

**static:** *seq*, an action sequence, initially empty

*state*, some description of the current world state

*goal*, a goal, initially null

*problem*, a problem formulation

*state*  $\leftarrow$  UPDATE-STATE(*state*, *percept*)

**if** *seq* is empty **then do**

*goal*  $\leftarrow$  FORMULATE-GOAL(*state*)

*problem*  $\leftarrow$  FORMULATE-PROBLEM(*state*, *goal*)

*seq*  $\leftarrow$  SEARCH(*problem*)

*action*  $\leftarrow$  FIRST(*seq*)

*seq*  $\leftarrow$  REST(*seq*)

**return** *action*

**Figure 3.1** A simple problem-solving agent. It first formulates a goal and a problem, searches for a sequence of actions that would solve the problem, and then executes the actions one at a time. When this is complete, it formulates another goal and starts over. Note that when it is executing the sequence it ignores its percepts: it assumes that the solution it has found will always work.

# Vacuum World

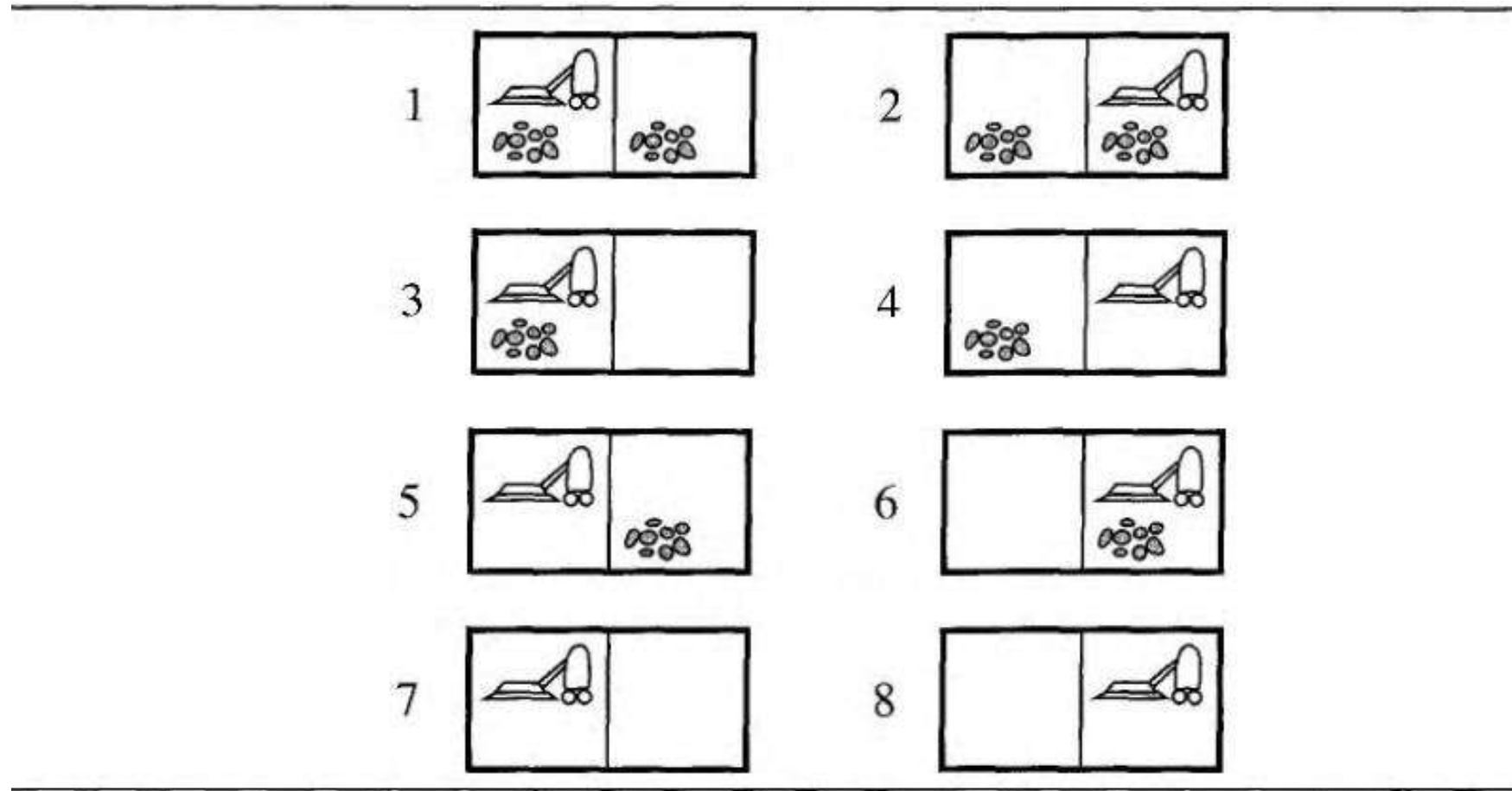
## Toy problems

The first example we will examine is the vacuum world first introduced in Chapter 2. (See Figure 2.2.) This can be formulated as a problem as follows:

- ◇ States: The agent is in one of two locations, each of which might or might not contain dirt. Thus there are  $2 \times 2^2 = 8$  possible world states.
- ◇ Initial state: Any state can be designated as the initial state.
- ◇ Successor function: This generates the legal states that result from trying the three actions (*Left*, *Right*, and *Suck*). The complete state space is shown in Figure 3.3.
- ◇ Goal test: This checks whether all the squares are clean.
- ◇ Path cost: Each step costs 1, so the path cost is the number of steps in the path.



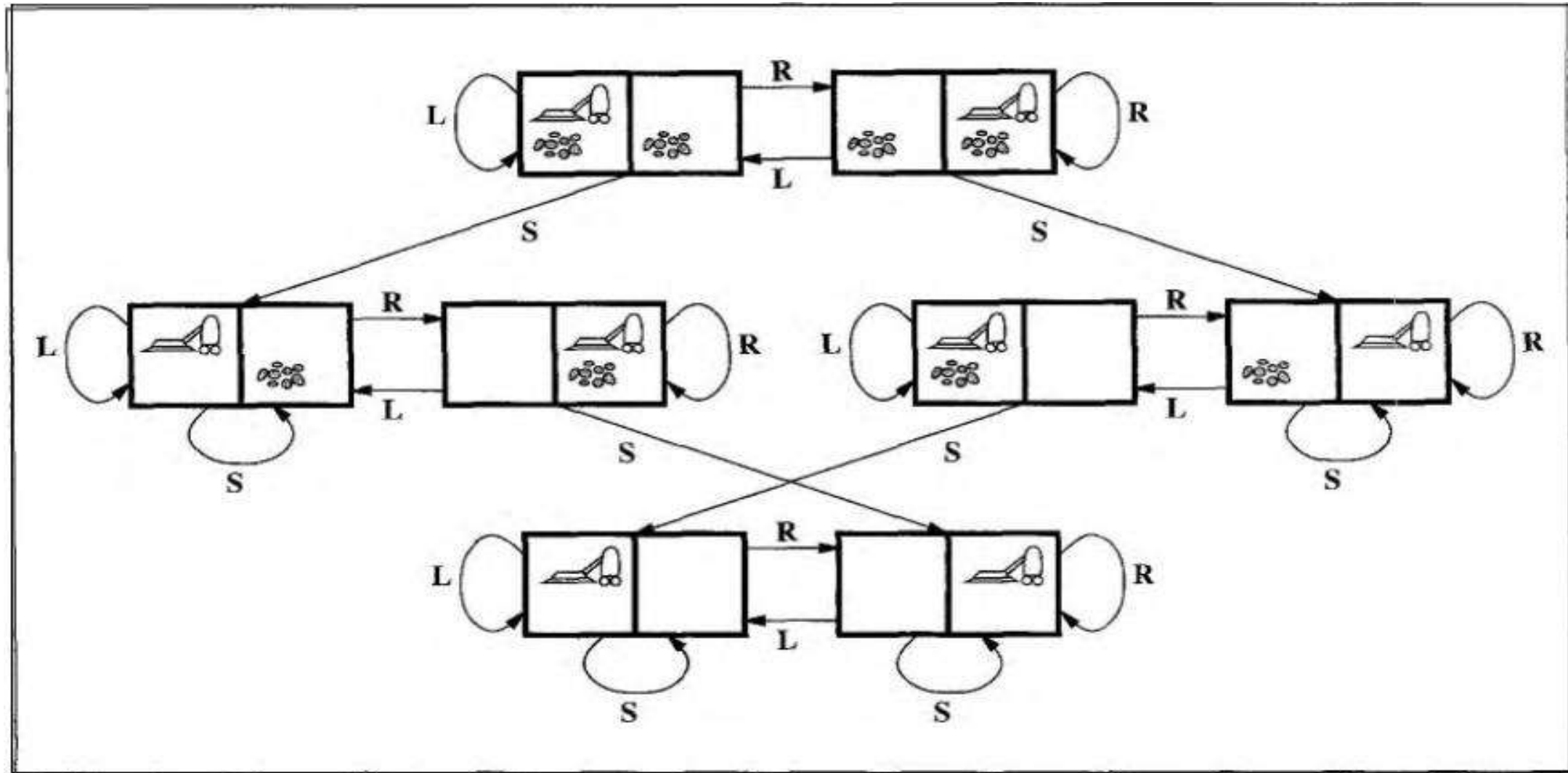
# Possible state



**Figure 3.20** The eight possible states of the vacuum world.

taught By: Ritu Raj Lamsal

# Possible states Vacuum World



**Figure 3.3** The state space for the vacuum world. Arcs denote actions: L = *Left*, R = *Right*, S = Suck.

taught By: Ritu Raj Lamsal



The 8 - puzzle, an instance of which is shown in Figure 3.4, consists of a 3 x 3 board with eight numbered tiles and a blank space. A tile adjacent to the blank space can slide into the space. The object is to reach a specified goal state, such as the one shown on the right of the figure. The standard formulation is as follows:

**States:** A state description specifies the location of each of the eight tiles and the blank in one of the nine squares.

**Initial state:** Any state can be designated as the initial state. Note that any given goal can be reached from exactly half of the possible initial states

- ◇ **Successor function:** This generates the legal states that result from trying the four actions (blank moves Left, Right, Up, or Down).
- ◇ **Goal test:** This checks whether the state matches the goal configuration shown in Figure 3.4. (Other goal configurations are possible.)
- ◇ **Path cost:** Each step costs 1, so the path cost is the number of steps in the path.

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

taught By: Ritu Raj Lamsal.

**Figure 3.4** A typical instance of the 8-puzzle.

## Example Problems

The real art of problem solving is in deciding the description of the states and the operators.

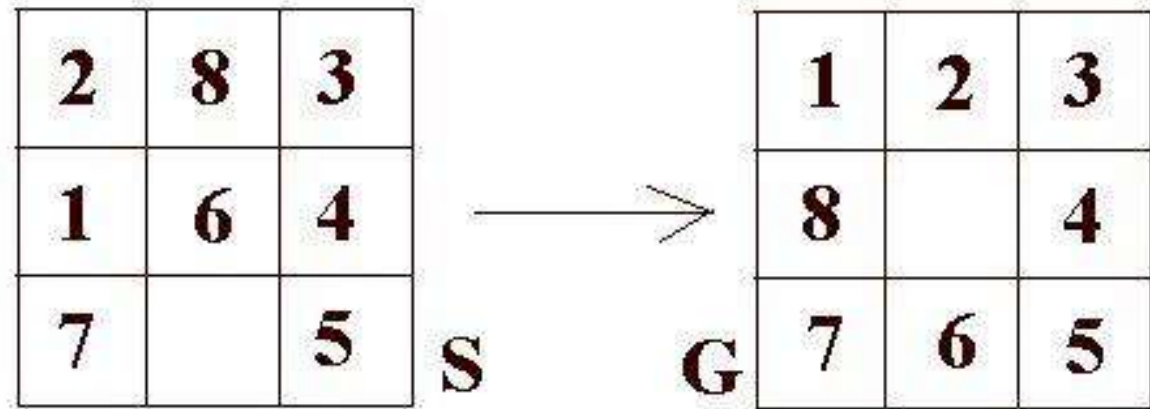
- The 8-puzzle

**State:** location of blank

**Operator:** blank moves left, right, up and down

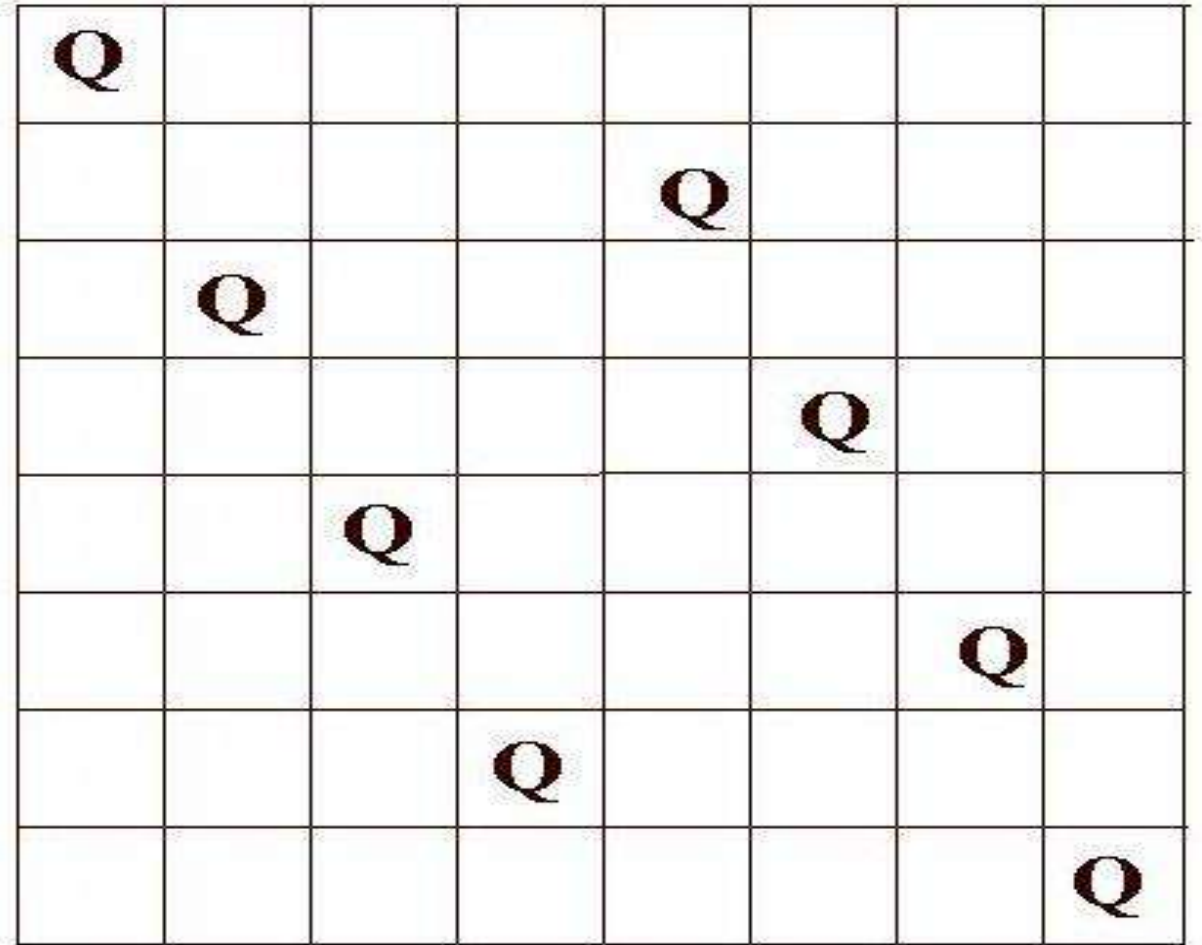
**Goal Test:** match G

**Path Cost:** each step costs 1 so cost is length of path



# 8 queens problem

Aim of problem is to place 8 queens on chess board so none attacks another, diagram almost shows a solution, with only two queens attacking each other.



# One solution to 8 Queens Problem

Q							
				Q			
							Q
					Q		
		Q					
						Q	
	Q						
			Q				

taught By: Ritu Raj Lamsal

# Mission and Cannibals Problem

## Missionaries and Cannibals Problem

- Three missionaries and three cannibals wish to cross the river.
- They have a small boat that will carry up to two people.
- Everyone can navigate the boat.
- If at any time the Cannibals outnumber the Missionaries on either bank of the river, they will eat the Missionaries.
- Find the smallest number of crossings that will allow everyone to cross the river safely.

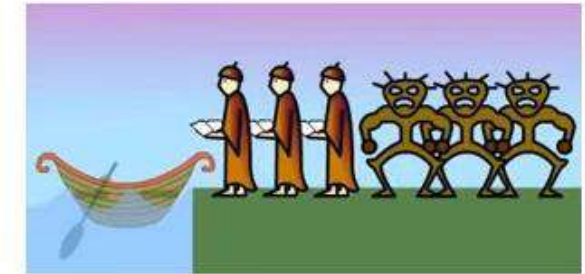


taught By: Ritu Raj Lamsal



# Missionaries and cannibals

## Solution to problem



- 0 Initial setup:
- 1 Two cannibals cross over:
- 2 One comes back:
- 3 Two cannibals go over again:
- 4 One comes back:
- 5 Two missionaries cross:
- 6 A missionary & cannibal return:
- 7 Two missionaries cross again:
- 8 A cannibal returns:
- 9 Two cannibals cross:
- 10 One returns:
- 11 And brings over the third: -

MMMCCC B

MMMC

MMMCC B

MMM

MMMC B

MC

MMCC B

CC

CCC B

C

CC B

—

B CC

C

B CCC

CC

B MMCC

MC

B MMMC

MMM

B MMMCC

MMMC

B MMMCCC

## Measuring problem-solving performance

The output of a problem-solving algorithm is either *failure* or a solution. (Some algorithms might get stuck in an infinite loop and never return an output.) We will evaluate an algorithm's performance in four ways:

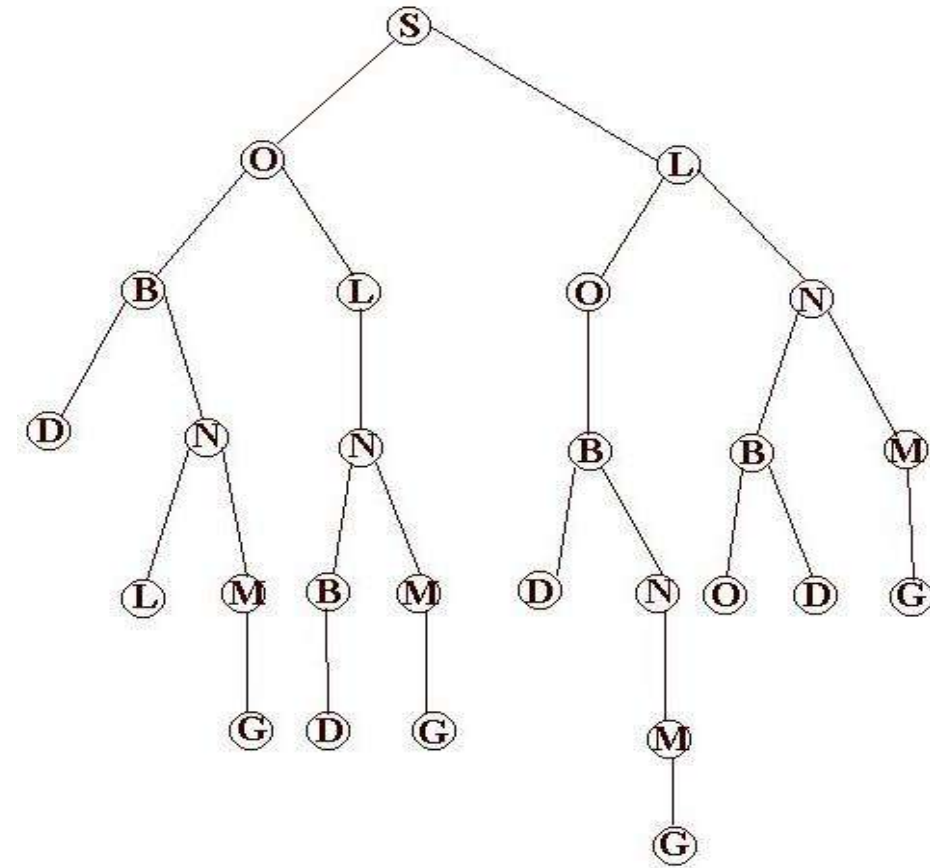
- ◇ **Completeness:** Is the algorithm guaranteed to find a solution when there is one?
- ◇ **Optimality:** Does the strategy find the optimal solution, as defined on page 62?
- ◇ **Time complexity:** How long does it take to find a solution?
- ◇ **Space complexity:** How much memory is needed to perform the search?

# Searching for solutions: Graphs or nets versus trees

The map of all paths within a state-space is a **graph of nodes** which are connected by **links**. Now if we trace out all possible paths through the graph, and terminate paths before they return to nodes already visited on that path, we produce a **search tree**.

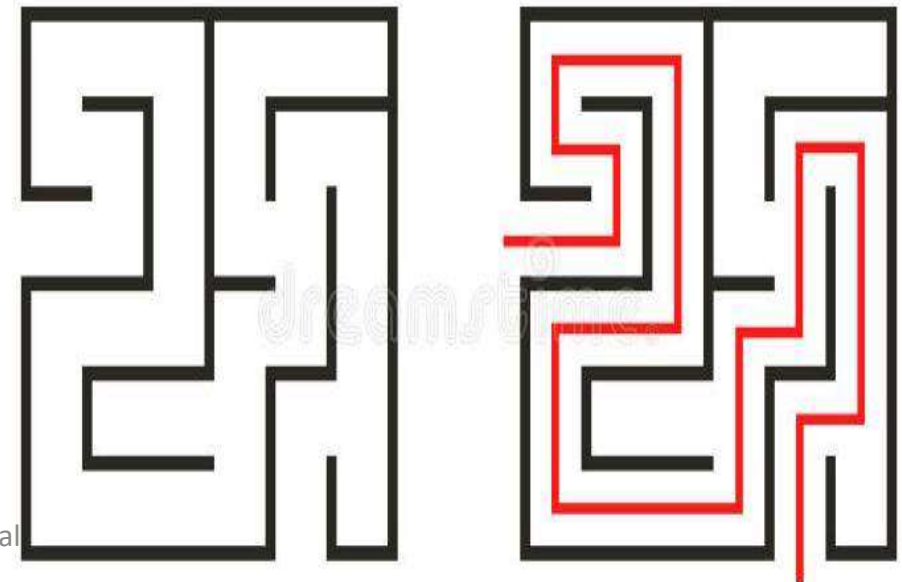
Like graphs, trees have nodes, but they are linked by **branches**. The start node is called the *root* and nodes at the other ends are *leaves*. Nodes have *generations of descendants*. **The first generation are children**. These children **have a single parent node**, and the list of nodes back to the root is their *ancestry*. A node and its descendants form a *subtree* of the node's parent. If a node's subtrees are unexplored or only partially explored, the node is *open*, otherwise it is *closed*. If all nodes have the same number of children, this number is the *branching factor*.

**The aim of search is *not* to produce complete physical trees in memory, but rather explore as little of the virtual tree looking for root-goal paths.**



# Uninformed and Informed search

- Two fundamental strategies used in artificial intelligence for finding solutions to problems in a search space. **Uninformed Search:** Uninformed search, also known as blind search, explores the search space without considering any additional information about the problem other than the current state and the available actions.
- Example: Consider a simple maze where you need to find the path from the entrance to the exit. In uninformed search, you might use techniques like depth-first search (DFS) or breadth-first search (BFS) to explore the maze without knowing anything about the locations of obstacles or the shortest path.



# DFS and BFS

- **Depth-First Search (DFS):** In DFS, you explore as far as possible along each branch before backtracking. This means you go as deep as possible down one path before trying other paths.
- **Breadth-First Search (BFS):** In BFS, you systematically explore all neighbor nodes at the present depth prior to moving on to the nodes at the next depth level.



# Informed Search

**2. Informed Search:** Informed search, also known as heuristic search, uses additional information about the problem to guide the search process. This information is typically provided by a heuristic function, which estimates how close a state is to the goal state.

Some types of informed search includes: Greedy Best Search, Hill Climbing, Simulated annealing , Genetic Algorithms

- Example: Let's consider the same maze problem. In informed search, you might use techniques like A\* search, which incorporates a heuristic function to guide the search towards the goal.
- A\* Search: A\* search evaluates nodes based on two cost functions:  $g(n)$ , the cost of reaching a node, and  $h(n)$ , the estimated cost of reaching the goal from that node. The function  $f(n) = g(n) + h(n)$  is used to prioritize nodes for expansion. A\* is considered optimal if the heuristic function  $h(n)$  is admissible (never overestimates the cost to reach the goal) and consistent (satisfies the triangle inequality).

# Uninformed Search: DFS

The depth-first algorithm is:

1. Form a one element queue Q consisting of the root node.
2. Until the Q is empty or the goal has been reached, determine if the first element in the Q is the goal.
  - 2a. If it is, do nothing.
  - 2b. If it isn't, remove the first element from the Q and add the first element's children, if any, to the FRONT of the Q.
3. If the goal is reached, success; else failure.

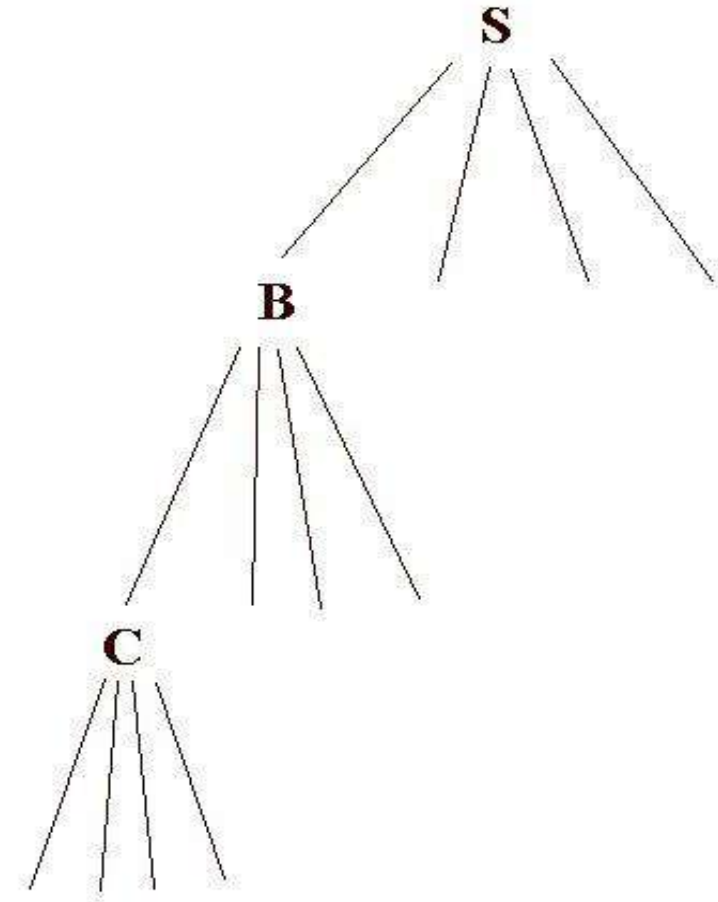
Depth-first search is simple, but is very unsatisfactory if the tree is unbalanced, with some leaves being at the end of much longer branches than others. It lacks discrimination.

Completeness: Yes

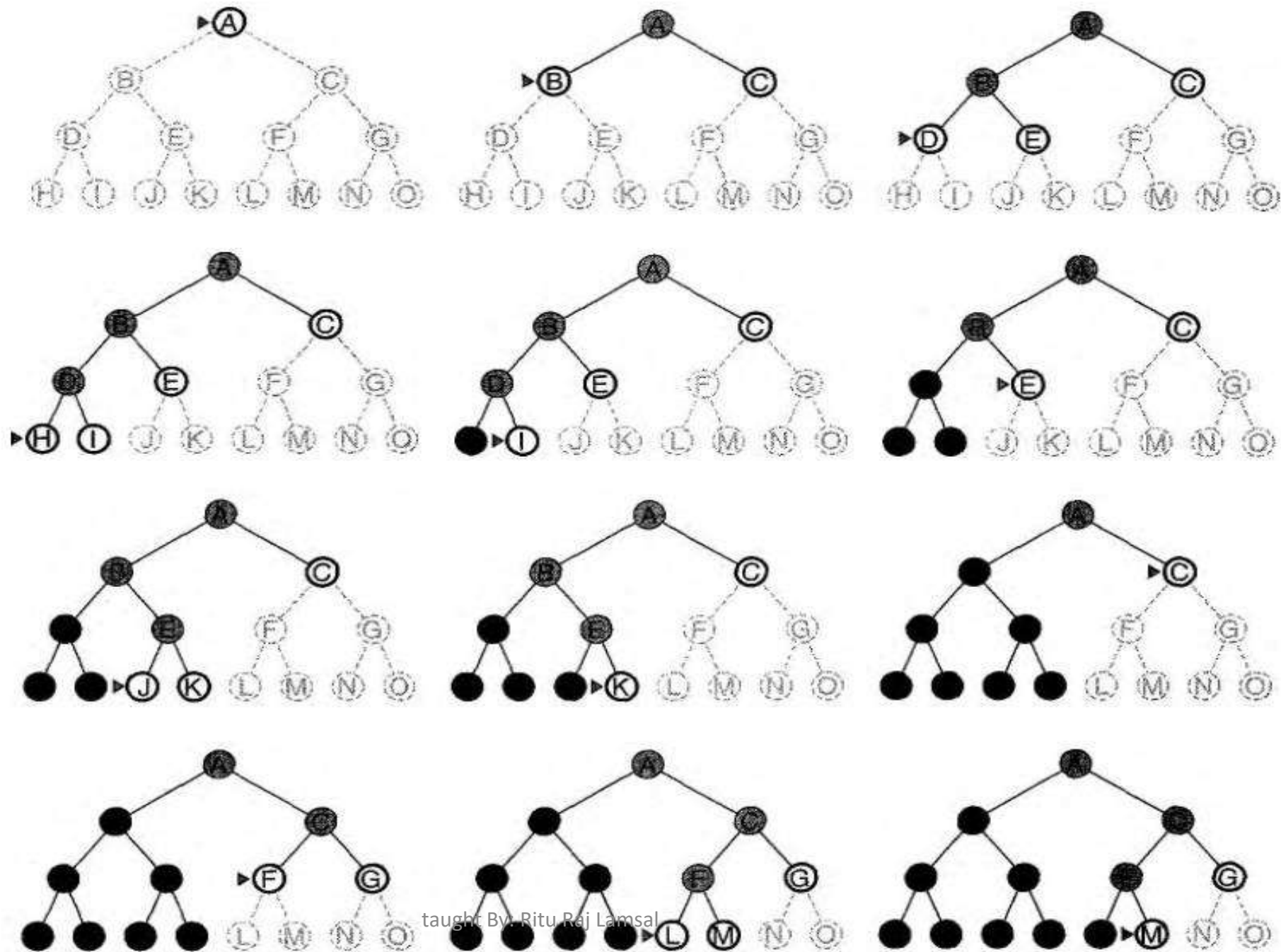
Time efficiency: No

Space efficiency: Yes

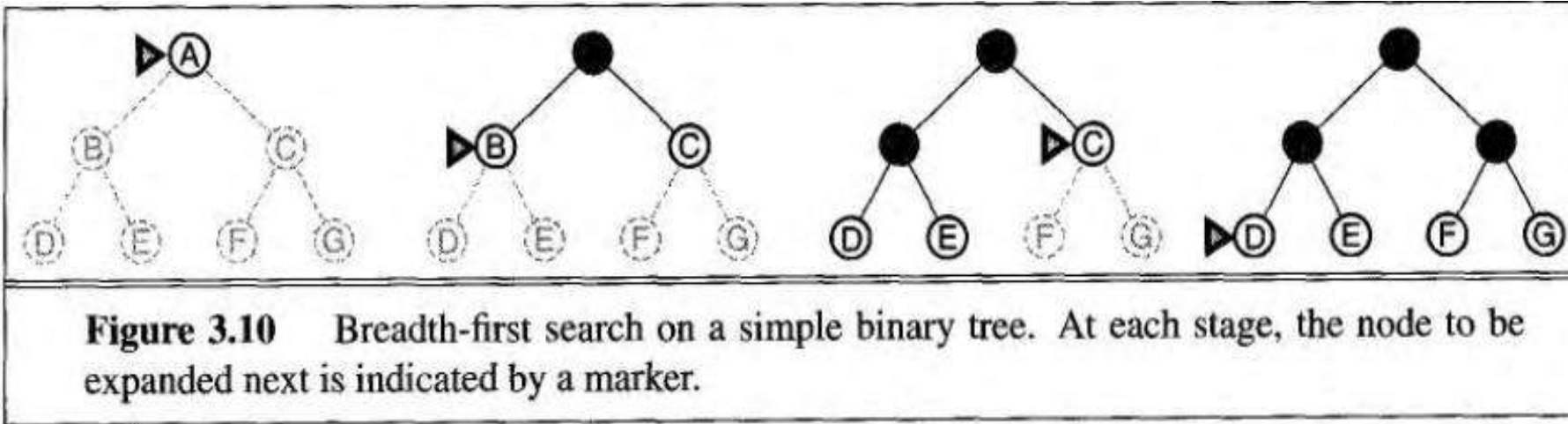
Optimality: No



# DFS



# Breadth First Search



Depth	Nodes	Time	Memory
2	1100	.11 seconds	1 megabyte
4	111,100	11 seconds	106 megabytes
6	$10^7$	19 minutes	10 gigabytes
8	$10^9$	31 hours	1 terabytes
10	$10^{11}$	129 days	101 terabytes
12	$10^{13}$	35 years	10 petabytes
14	$10^{15}$	3,523 years	1 exabyte

**Figure 3.11** Time and memory requirements for breadth-firstsearch. The numbers shown assume branching factor  $b = 10$ ; 10,000 nodes/second; 1000 bytes/node.

# Breadth first Search

The breadth-first algorithm is:

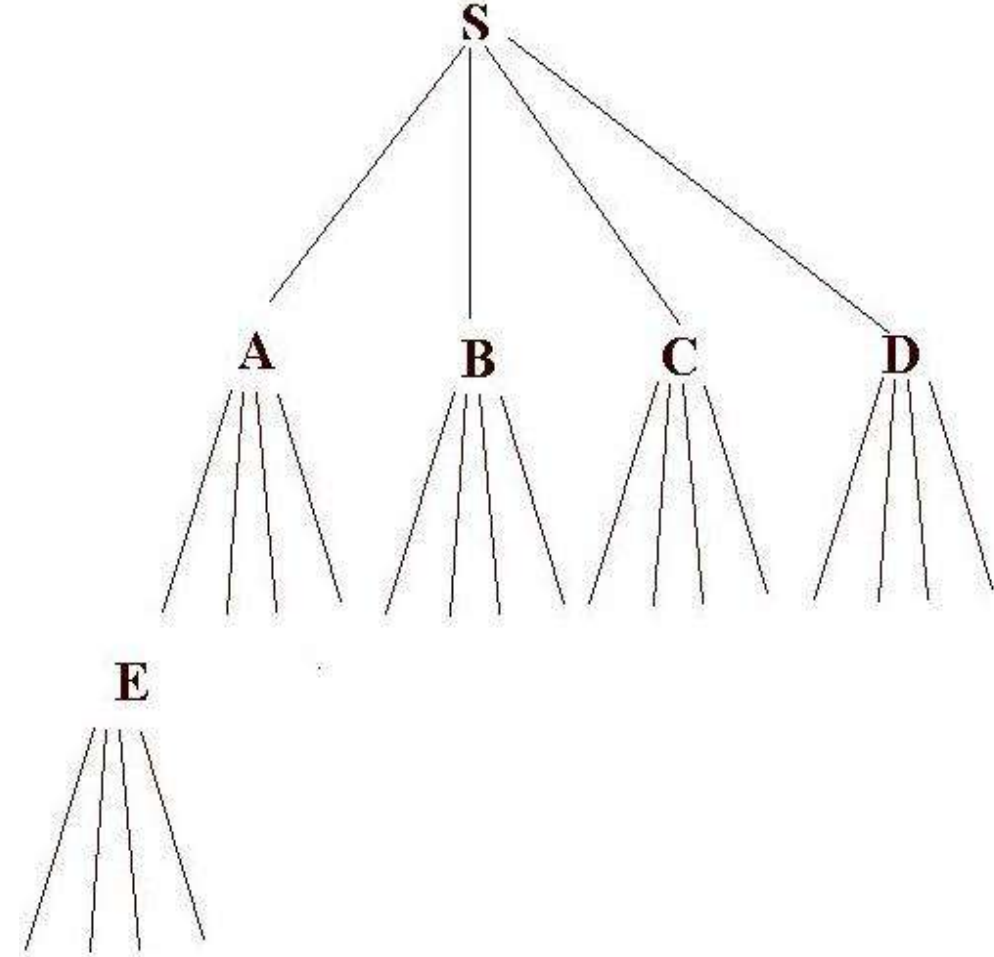
1. Form a one element queue Q consisting of the root node.
2. Until the Q is empty or the goal has been reached, determine if the first element in the Q is the goal.
  - 2a. If it is, do nothing.
  - 2b. If it isn't, remove the first element from the Q, and add the first element's children, if any, to the BACK of the Q.
3. If the goal is reached, success; else failure.

Completeness: Yes

Time efficiency: No

Space efficiency: No

Optimality: No





## Comparing uninformed search strategies

Figure 3.17 compares search strategies in terms of the four evaluation criteria set forth in Section 3.4.

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes <sup>a</sup>	Yes <sup>a,b</sup>	No	No	Yes <sup>a</sup>	Yes <sup>a,d</sup>
Time	$O(b^{d+1})$	$O(b^{1+\lceil C^*/\epsilon \rceil})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^{d+1})$	$O(b^{1+\lceil C^*/\epsilon \rceil})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes <sup>c</sup>	Yes	No	No	Yes <sup>c</sup>	Yes <sup>c,d</sup>

**Figure 3.17** Evaluation of search strategies.  $b$  is the branching factor;  $d$  is the depth of the shallowest solution;  $m$  is the maximum depth of the search tree;  $\ell$  is the depth limit. Superscript caveats are as follows: <sup>a</sup> complete if  $b$  is finite; <sup>b</sup> complete if step costs  $\geq \epsilon$  for positive  $\epsilon$ ; <sup>c</sup> optimal if step costs are all identical; <sup>d</sup> if both directions use breadth-first search.

# Informed Search

- Methods use an **evaluation** function to guide the search towards goal. Information about the state space will be used to guide the search. Knowledge is used in the queueing function to determine which node to expand next. This process uses an evaluation function:
- Evaluation function  $f(n)$
- Cost:  $g(n)=C$
- heuristic:  $h(n)=E$  (estimate of distance to goal)
- The evaluation function returns a number describing the desirability of expanding the node and uses these numbers to guide the search towards goal.

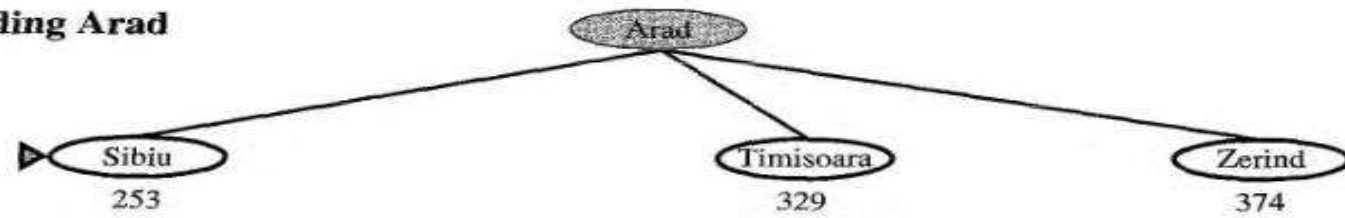
- Best First search
- simulated annealing (Inspired by statistical physics)
- Genetic algorithms and evolutionary biology
- A\* algorithm

# Greedy best first search

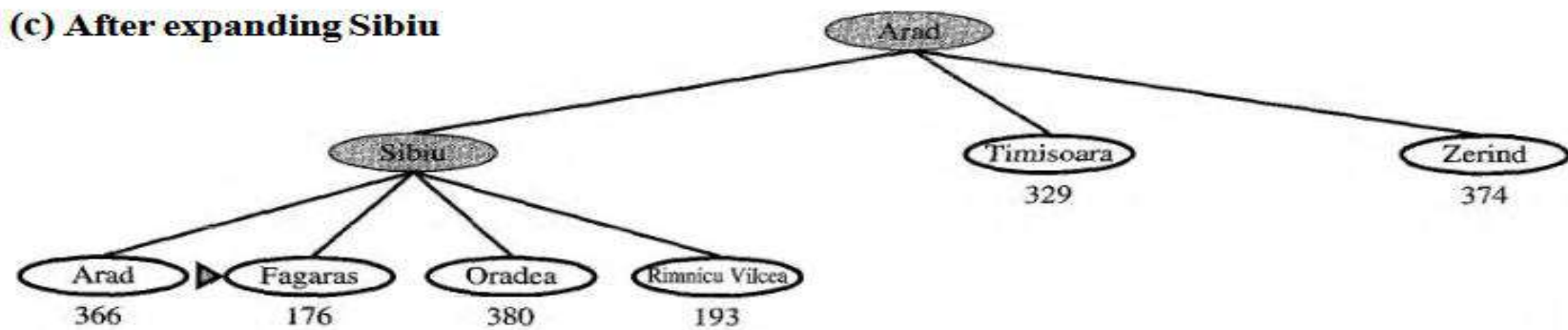
(a) The initial state



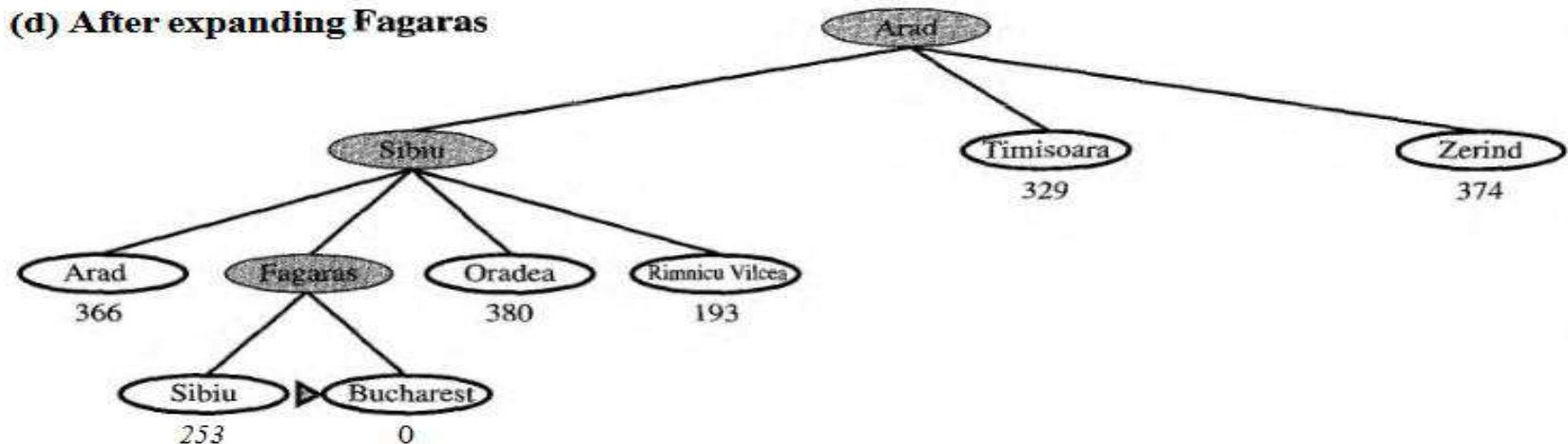
(b) After expanding Arad



(c) After expanding Sibiu



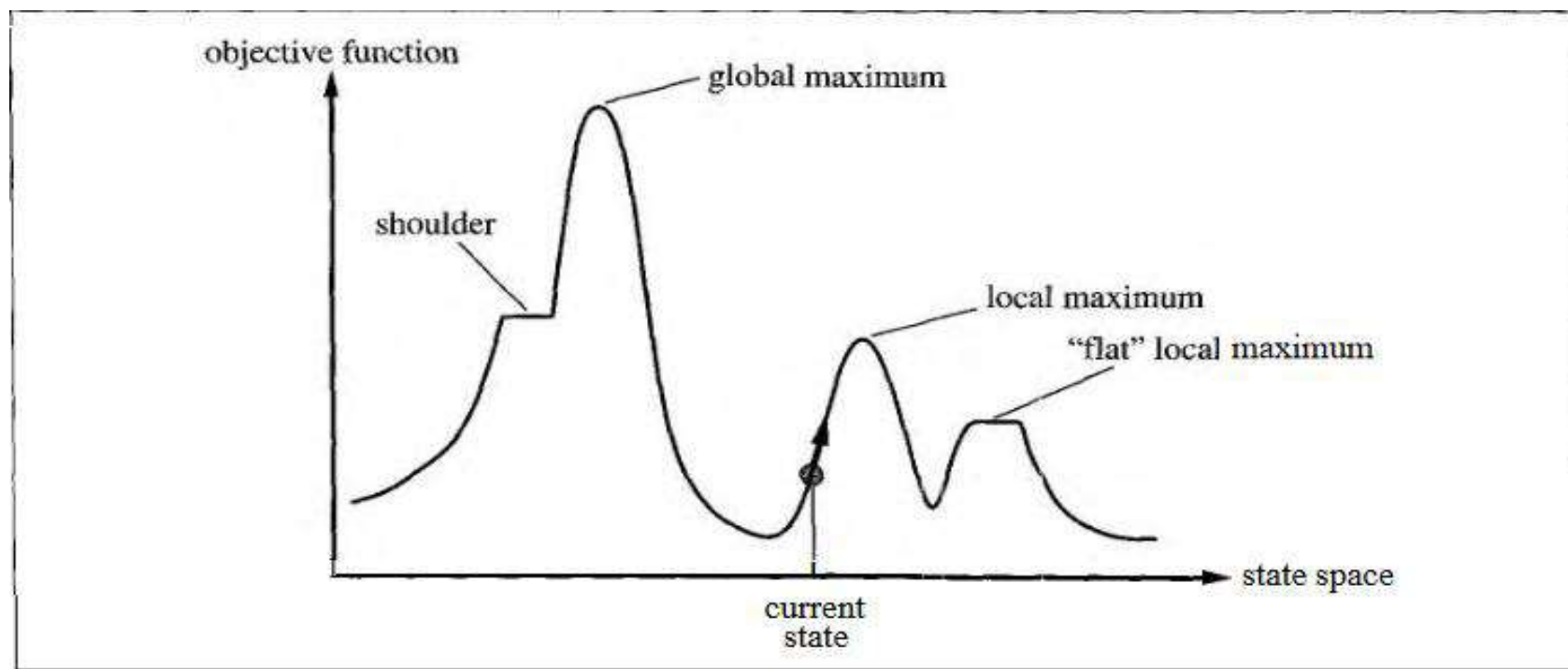
(d) After expanding Fagaras



**Figure 4.2** Stages in a greedy best first search for Bucharest using the straight-line distance heuristic  $h_{SLD}$ . Nodes are labeled with their h-values.

## Hill-climbing search

The **hill-climbing** search algorithm is shown in Figure 4.11. It is simply a loop that continually moves in the direction of increasing value—that is, uphill. It terminates when it reaches a “peak” where no neighbor has a higher value. The algorithm does not maintain a search tree, so the current node data structure need only record the state and its objective function value. Hill-climbing does not look ahead beyond the immediate neighbors of the current state. This resembles trying to find the top of Mount Everest in a thick fog while suffering from amnesia.



**Figure 4.10** A one-dimensional state space landscape in which elevation corresponds to the objective function. The aim is to find the global maximum. Hill-climbing search modifies the current state to **try** to improve it, as shown by the arrow. The various topographic features are defined in the text.



2b. If it isn't, remove the first element from the Q, sort the first element's children, if any, by estimating remaining distance, and add this sorted list to the FRONT of the Q. 3. If the goal is reached, success; else failure.



## **A\* search: Minimizing the total estimated solution cost**

The most widely-known form of best-first search is called **A\*** search (pronounced "A-star search"). It evaluates nodes by combining  $g(n)$ , the cost to reach the node, and  $h(n)$ , the cost to get from the node to the goal:

$$f(n) = g(n) + h(n)$$

Since  $g(n)$  gives the path cost from the start node to node  $n$ , and  $h(n)$  is the estimated cost of the cheapest path from  $n$  to the goal, we have

$$f(n) = \text{estimated cost of the cheapest solution through } n$$

Thus, if we are trying to find the cheapest solution, a reasonable thing to try first is the node with the lowest value of  $g(n) + h(n)$ . It turns out that this strategy is more than just reasonable: provided that the heuristic function  $h(n)$  satisfies certain conditions, A\* search is both complete and optimal.

# Simulated Annealing

- In metallurgy, annealing is the process used to temper or harden metals and glass by heating them to a high temperature and then gradually cooling them, thus allowing the material to coalesce into a low-energy crystalline state.
- A hill -climbing algorithm that never makes "downhill" moves towards states with lower value (or higher cost) is guaranteed to be incomplete, because it can get stuck on a local maximum.
- it seems reasonable to try to combine hill climbing with a random walk in some way that yields both efficiency and completeness. Simulated annealing is such an algorithm

# Genetic Algorithm

- Genetic algorithms (GAs) are optimization algorithms inspired by the process of natural selection and genetics. They are used to find approximate solutions to optimization and search problems where traditional methods may be impractical or inefficient. Here are the basics of genetic algorithms:
  - 1.Population:** A genetic algorithm starts with a population of candidate solutions, often represented as a set of chromosomes or individuals. Each chromosome represents a potential solution to the problem.
  - 2.Fitness Function:** A fitness function evaluates how good each chromosome is as a solution to the problem. It assigns a numerical value (fitness score) to each chromosome based on its quality or how well it satisfies the optimization criteria. The fitness function guides the search process by providing a measure of the solution's quality.

# GA cont..

**3.Selection:** The selection process determines which chromosomes are selected to reproduce and create offspring for the next generation. Chromosomes with higher fitness scores are more likely to be selected, mimicking the principle of "survival of the fittest" in natural selection.

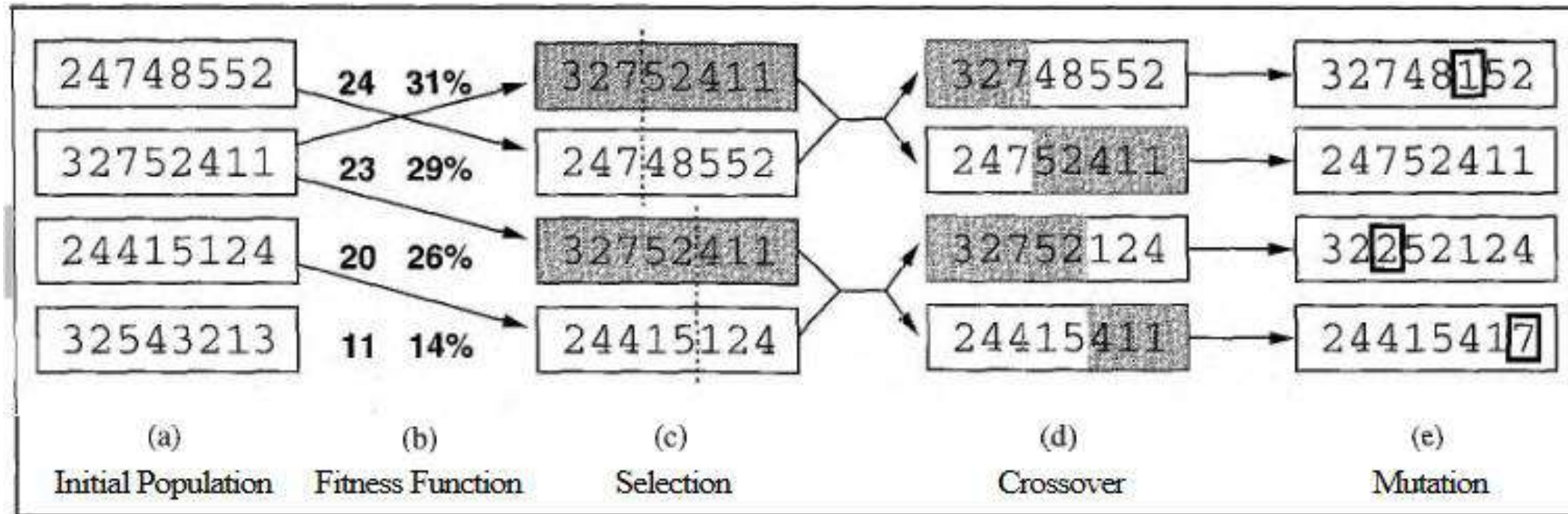
**4.Crossover (Recombination):** Crossover is the process of combining genetic information from two parent chromosomes to produce offspring. It mimics the biological process of genetic recombination during reproduction. Different crossover techniques, such as one-point crossover, two-point crossover, and uniform crossover, can be used to generate diverse offspring.

**5.Mutation:** Mutation introduces random changes or variations to the genetic information of individual chromosomes. It helps maintain diversity in the population and prevents the algorithm from getting stuck in local optima. Mutation typically involves randomly changing or flipping certain bits or components of a chromosome with a low probability.

**6.Termination Criteria:** Genetic algorithms iterate through generations of populations until a termination condition is met. Termination criteria can include reaching a maximum number of generations, finding a satisfactory solution, or reaching a predefined fitness threshold.



# Genetic Algorithm



**Figure 4.15** The genetic algorithm. The initial population in (a) is ranked by the fitness function in (b), resulting in pairs for mating in (c). They produce offspring in (d), which are subject to mutation in (e).

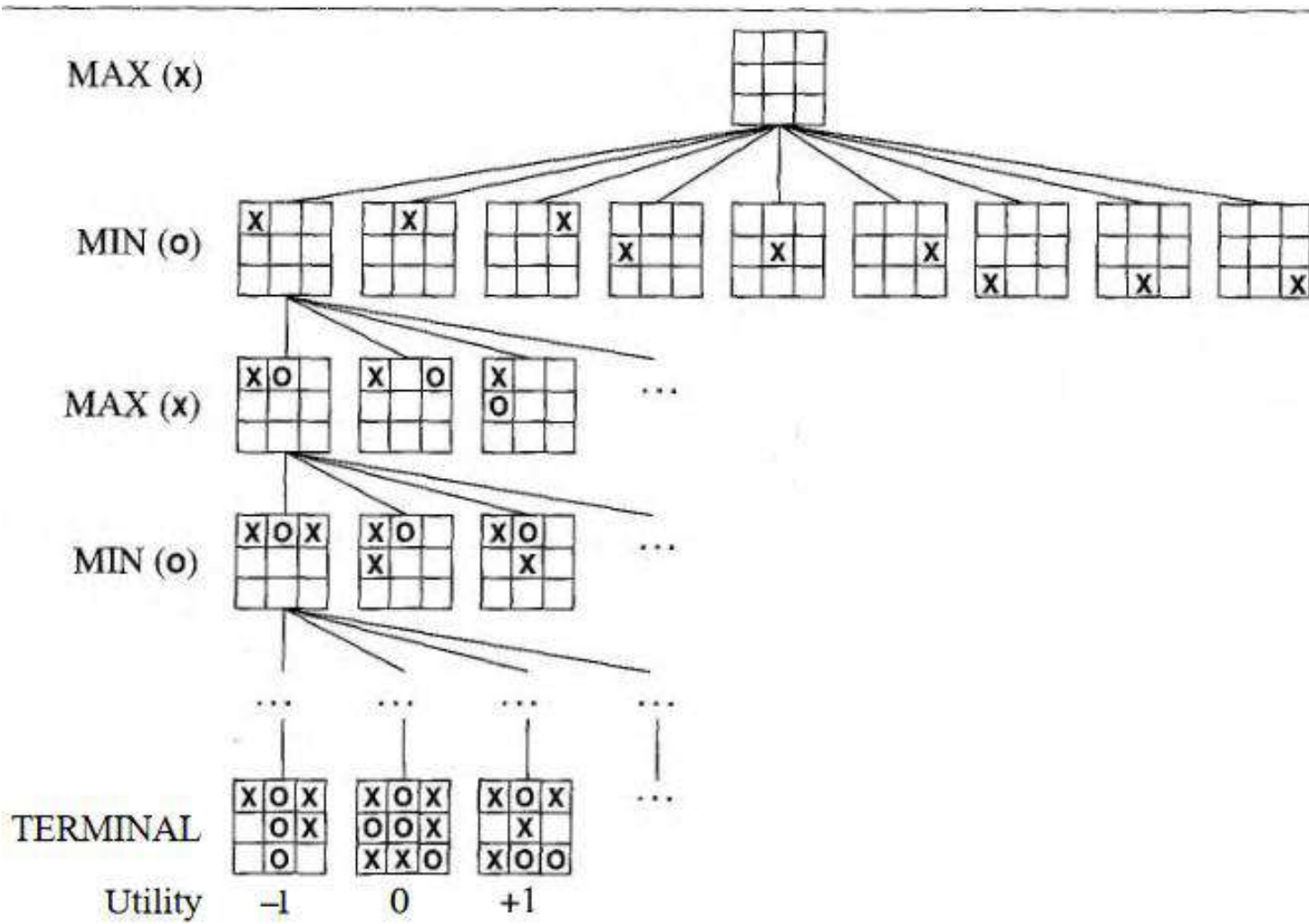
# Adversarial search

- Adversarial search in artificial intelligence is used to resolve two-person games in which one player seeks to maximize their score while the other tries to minimize it.
- Chess, checkers, and tic-tac-toe are the three most popular games that can be solved via adversarial search.
- In many real-world applications, such as financial decision-making and military operations, adversarial search is used. To give players clever opponents in video games, it is also deployed.

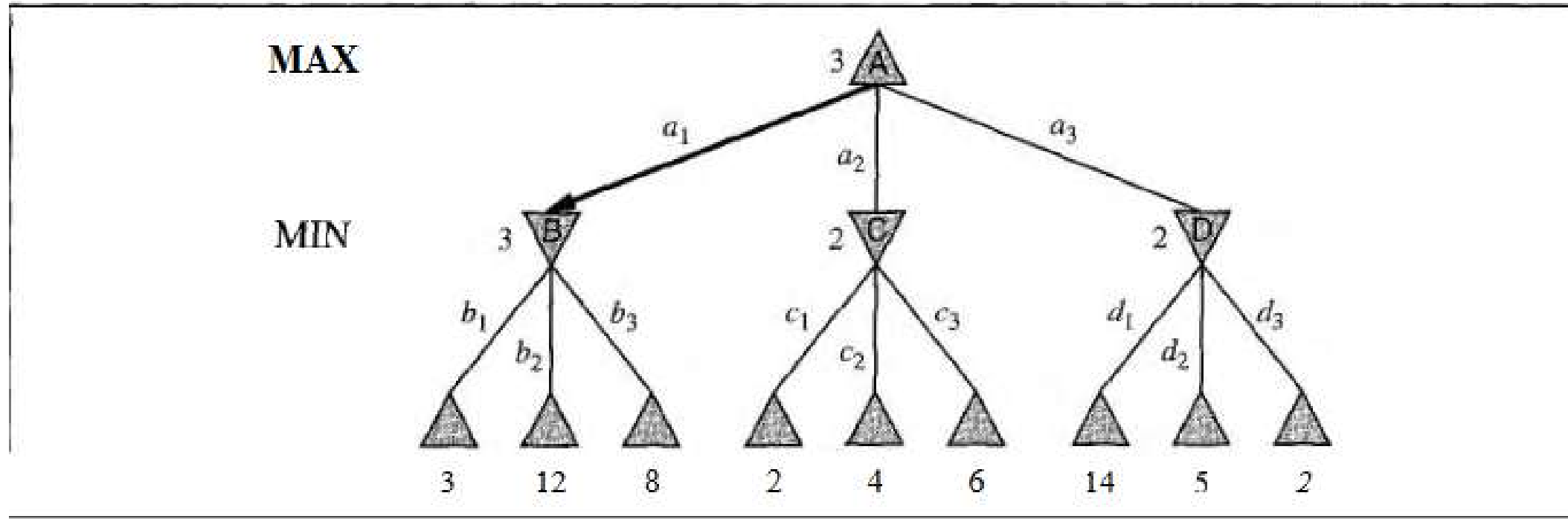
- The objective of the adversarial search is to create intelligent algorithms that can choose the best course of action under conflicting circumstances while taking into account the moves of rivals and their potential retaliations. To find the optimum move or action to make, adversarial search algorithms often search through a game tree, which represents all feasible game states and their transitions.

# Partial Search Tree for game Tic – Tac-Toe

A (partial) search tree for the game of tic -tac -toe. The top node is the initial state, and M A X moves first, placing an X in an empty square. We show part of the search tree, giving alternating moves by MIN (O) and MAX, until we eventually reach terminal states, which can be assigned utilities according to the rules of the game.



# 2-Ply Game Tree ( Best Move- Max and MIN)



**Figure 6.2** A two-ply game tree. The  $\triangle$  nodes are "MAX nodes," in which it is MAX's turn to move, and the  $\nabla$  nodes are "MIN nodes." The terminal nodes show the utility values for MAX; the other nodes are labeled with their minimax values. MAX's best move at the root is  $a_1$ , because it leads to the successor with the highest minimax value, and MIN's best reply is  $b_1$ , because it leads to the successor with the lowest minimax value.



# Alpha-beta pruning

- Alpha-beta pruning is a technique used in game trees to reduce the number of nodes that need to be evaluated in the search for the best move. It's a clever way to avoid examining branches of the game tree that cannot possibly affect the final decision. Here's a simple explanation:
- Imagine you're playing a game like chess, where you have to consider many possible moves ahead. The game tree represents all possible moves for both players, branching out with each move.
- Alpha-beta pruning works by keeping track of two values, called alpha and beta, during the search:
- Alpha represents the best score that the maximizing player (e.g., the AI trying to win) has found so far on the path from the root of the tree down to the current node.
- Beta represents the best score that the minimizing player (e.g., the opponent) has found so far on the path from the root of the tree down to the current node.

# Working of alpha-beta pruning

- 1.The algorithm begins by traversing the game tree, evaluating nodes and updating alpha and beta values as it goes.
- 2.When it reaches a node, it evaluates whether the node's score can affect the final decision. If it's determined that the current node cannot possibly lead to a better outcome than what's already known (based on alpha and beta values), it prunes the subtree below that node. This means that the algorithm doesn't need to explore that part of the tree further.
- 3.By pruning unnecessary branches, the algorithm can significantly reduce the number of nodes it needs to evaluate, making the search more efficient.

# Unit III

# Knowledge Representation using Logic

- **Knowledge Representation using Logic**
- The aim of this section is to show how logic can be used to form representations of the world and how a process of inference can be used to derive new representations about the world and how these can be used by an intelligent agent to deduce what to do.
- We require:
- *A formal language* to represent knowledge in a computer tractable form.
- *Reasoning* - Processes to manipulate this knowledge to deduce non-obvious facts.

- **Why logic?**
- The challenge is to design a language which allows one to represent all the necessary knowledge. We need to be able to make statements about the world such as describing things - people, houses, theories etc; relations between things and properties of things. **Logic** makes statements about the world which are true (or false) if the state of affairs it represents is the case (or not the case). Compared to natural languages (expressive but context sensitive) and programming languages (good for concrete data structures but not expressive) logic combines the advantages of natural languages and formal languages. Logic is:
  - concise
  - unambiguous
  - context insensitive
  - expressive
  - effective for inferences



# A logic is defined by the following:

- 1.**Syntax** - describes the possible configurations that constitute sentences.
  - 2.**Semantics** - determines what facts in the world the sentences refer to i.e. the interpretation. Each sentence makes a claim about the world.
  - 3.**Proof theory** - set of rules for generating new sentences that are necessarily true given that the old sentences are true. The relationship between sentences is called **entailment**. The semantics link these sentences (representation) to facts of the world. The proof can be used to determine new facts which follow from the old.
- We will consider two kinds of logic: **propositional logic** and **first-order logic** or more precisely first-order **predicate calculus**. Propositional logic is of limited expressiveness but is useful to introduce many of the concepts of logic's syntax, semantics and inference procedures.

# Propositional Logic

Propositional symbols are used to represent facts. Each symbol can mean what we want it to be. Each fact can be either true or false.

Propositions are combined with logical connectives to generate sentences with more complex meaning. The connectives are:

$\wedge$  AND

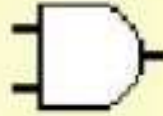
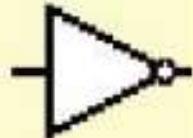
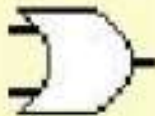
$\vee$  OR

$\neg$  NOT

$\Rightarrow$  implies

$\Leftrightarrow$  mutual implication

The meaning (semantics) of the connectives is represented by truth tables. The following are the truth tables for the connectives:

Expression	Truth Table	Electrical Equivalent
$E \wedge F$	$E \downarrow F \rightarrow$ True False True True False False False False	
$\neg E$	$E$ True False False True	
$E \vee F$	$E \downarrow F \rightarrow$ True False True True True False True False	
$E \Rightarrow F$	$E \downarrow F \rightarrow$ True False True True False False True True	-

NOTE(1):

$$E \Rightarrow F \Leftrightarrow \neg E \Rightarrow F .$$

NOTE(2): 'False True' is a True sentence/assertion.

taught By: Ritu Raj Lamsal

# Implication

- A sentence of the form:  $E \Rightarrow F$   
is called an implication with premise (antecedent)  $E$  and conclusion (consequent)  $F$ . Implications are also known as if-then statements or rules.
- Its truth table does not quite fit our intuitive understanding of "E implies F" since propositional logic does not require any relation of causation or relevance between  $E$  and  $F$ . It is better to think of "E F" as saying: "if  $E$  is true then I am claiming that  $F$  is true (otherwise I am making no claim)".
- The following commutative, distributive, associative rules apply, as do De Morgan's theorems.

Commutative	$E \wedge F \Leftrightarrow F \wedge E$
	$E \vee F \Leftrightarrow F \vee E$
Distributive	$E \wedge (F \vee G) \Leftrightarrow (E \wedge F) \vee (E \wedge G)$
	$E \vee (F \wedge G) \Leftrightarrow (E \vee F) \wedge (E \vee G)$
Associative	$E \wedge (F \wedge G) \Leftrightarrow (E \wedge F) \wedge G$
	$E \vee (F \vee G) \Leftrightarrow (E \vee F) \vee G$
De Morgan's	$\neg(E \wedge F) \Leftrightarrow \neg E \vee \neg F$
	$\neg(E \vee F) \Leftrightarrow \neg E \wedge \neg F$
Negation	$\neg(\neg E) \Leftrightarrow E$

## Logical Proofs

A sentence is **valid** or necessarily true if and only if it is true under all possible interpretations in all possible worlds. If a sentence is valid, it is so by virtue of its logical structure independent of what possible interpretations are like.

A sentence is **satisfiable** if and only if there is some interpretation in some world for which it is true. A sentence that is not satisfiable is **unsatisfiable**.

## Making Inferences

Reasoning and inference are generally used to describe any process by which conclusions are reached. Inference is often of three types:

### 1. Abduction

Abduction is a process to generate explanations. It aims to give a hypothetical explanation. It is only a *plausible* inference.

If there is an axiom  $E \Rightarrow F$  and an axiom  $F$ , then  $E$  does NOT logically follow. This is called Abduction and is not a sound rule of inference.

### 2. Induction

Hypothesises a general rule from observations.

### 3. Deduction or rational inference

These are inferences which are made by the **sound rules of inference**. This means that the conclusion is true in all cases where the premise is true. These rules preserve the truth.



- **Sound rules of inference**

- Looking at the truth tables you will easily see that the sound rules are:

1. **Modus ponens** (or implication-elimination) From an implication and the premise of the implication you can infer the conclusion.

2. If there is an axiom  $E \rightarrow F$  and an axiom  $E$ , then  $F$  follows logically.

3. **Modus tollens** If there is an axiom  $E \rightarrow F$  and an axiom  $\neg F$ , then  $\neg E$  follows logically.

4. **Resolution**. If there is an axiom  $E \rightarrow F$  and an axiom  $F \rightarrow G$  then  $E \rightarrow G$  follows logically.

If there is an axiom  $E \vee F$  and an axiom  $F \vee G$  then  $E \vee G$  follows logically.

In fact, resolution can subsume both modus ponens and modus tollens. It can also be generalized so that there can be any number of disjuncts in either of the two resolving expressions, including just one. (Note, disjunct are expressions connected by  $\vee$ , conjuncts are those connected by  $\wedge$ .) The only requirement is that one expression contains the negation of one disjunct from the other.

To verify soundness we can construct a truth-table with one line for each possible model of the proposition symbols in the premise, and show that in all models where the premise is true, the conclusion is also true.

Example of truth table for resolution:

Example of truth table for resolution:

A	B	C	$A \vee B$	$\neg(B \vee C)$	$A \vee C$
F	F	F	F	T	F
F	F	T	F	T	T
F	T	F	T	F	F
F	T	T	T	T	T
T	F	F	T	T	T
T	F	T	T	T	T
T	T	F	T	F	T
T	T	T	T	T	T





- **1. Knowledge and Knowledge Representation:**
- Knowledge refers to information, facts, and expertise acquired through learning and experience.
- Knowledge representation involves organizing and structuring knowledge in a way that can be effectively utilized by intelligent systems.
- It aims to capture, store, and manipulate knowledge to support reasoning, problem-solving, and decision-making tasks.

- **2. Knowledge Representation System:**

- A knowledge representation system is a framework or methodology used to represent knowledge in a structured and meaningful way.
- It includes formal languages, data structures, and algorithms for encoding and processing knowledge.
- Knowledge representation systems enable machines to understand and reason about the world, facilitating intelligent behavior.



- **3. Knowledge Representation Techniques:**
- **Simple Relational Knowledge:**
  - Utilizes relationships between entities to represent knowledge.
  - Techniques include tables, graphs, conceptual dependency, and scripts.
  - Each technique provides a different way to organize and represent knowledge based on relational structures.
- **Table:**
  - Represents knowledge using rows and columns in a tabular format.
  - Each row corresponds to an entity or object, and each column represents an attribute or property of the entity.
  - Tables are suitable for representing structured data with well-defined attributes and relationships.

- **Graph:**

- Represents knowledge as a network of nodes and edges.
- Nodes represent entities, and edges represent relationships between entities.
- Graphs allow for complex and flexible representations of interconnected knowledge.

- **Conceptual Dependency:**

- Represents knowledge using semantic structures called conceptual dependencies.
- Conceptual dependencies capture the meaning of actions, events, and relationships between entities.
- It provides a high-level representation of knowledge suitable for natural language understanding and reasoning.

- **Scripts:**

- Represent knowledge in the form of structured sequences of events or actions.
- Scripts define typical sequences of events and the roles of participants involved.
- They facilitate reasoning about common scenarios and predicting future actions based on past experiences.

# 4. Inheritable Knowledge:

- **Semantic Networks:**

- Semantic networks represent knowledge using nodes and arcs to denote concepts and relationships between them.
- Nodes represent entities or concepts, and arcs represent relationships or connections between nodes.
- Semantic networks provide a graphical representation of knowledge and are used for organizing and structuring semantic information.

- **Frames:**

- Frames represent knowledge as a collection of attributes or slots associated with a specific object or concept.
- Each frame consists of slots representing properties or characteristics of the object and fillers representing specific values or instances.
- Frames are used for representing structured knowledge and capturing domain-specific information.

- **Ontologies:**

- Ontologies represent knowledge using a formal, explicit specification of concepts, relationships, and constraints within a domain.
- They define a shared understanding of a domain and provide a common vocabulary for describing entities and their relationships.
- Ontologies are used for knowledge sharing, interoperability, and reasoning in various domains, such as the Semantic Web, healthcare, and manufacturing.

# 5. Procedural Knowledge:

- **Rule-Based Representation:**

- Rule-based representation represents knowledge in the form of rules or conditional statements that specify actions or decisions based on conditions.
- Rules consist of antecedents (conditions) and consequents (actions) and are used to model procedural knowledge.
- Rule-based systems use inference mechanisms to apply rules and make decisions or perform actions based on input data or facts.

- These knowledge representation techniques complement each other and are often used in combination to represent different aspects of knowledge in intelligent systems. Semantic networks, frames, and ontologies focus on organizing and structuring declarative knowledge, while rule-based representation deals with procedural knowledge and decision-making logic.



- **6. Inferential Knowledge:**
- **Propositional Logic:**
  - Propositional logic represents knowledge using propositions, which are statements that can be either true or false.
  - It employs logical operators such as AND, OR, and NOT to construct complex statements from simpler propositions.
  - Propositional logic is used for reasoning about the truth values of statements and making logical deductions.
- **First-Order Logic:**
  - First-order logic extends propositional logic by introducing variables, quantifiers, and predicates to represent more complex knowledge.
  - It allows for the representation of relationships between objects and properties of objects using quantified statements.
  - First-order logic is widely used in artificial intelligence for knowledge representation, reasoning, and automated theorem proving.
- **Modal Logic:**
  - Modal logic extends propositional logic to reason about necessity, possibility, and other modalities.
  - It introduces modal operators such as  $\Box$  (necessity) and  $\Diamond$  (possibility) to express statements about possible worlds and their relationships.
  - Modal logic is used in areas such as philosophy, computer science, and artificial intelligence to reason about knowledge, belief, and possibility.
- **Temporal Logic:**
  - Temporal logic extends propositional logic to reason about temporal aspects of knowledge, such as time and sequence of events.
  - It introduces temporal operators such as F (eventually) and G (always) to express statements about temporal properties and relationships.
  - Temporal logic is used in areas such as formal verification, model checking, and real-time systems to specify and verify temporal properties of systems.

- **7. Fuzzy Logic:**

- Fuzzy logic extends classical binary logic by allowing degrees of truth between true and false.
- It employs fuzzy sets and fuzzy logic operators to represent and reason with vague or imprecise information.
- Fuzzy logic is used in control systems, expert systems, and decision-making applications where uncertainty and ambiguity are present.

- **8. Neural Networks:**

- Neural networks represent knowledge using interconnected nodes (neurons) organized in layers.
- They learn from data through a process called training, adjusting the connections (weights) between neurons to minimize prediction errors.
- Neural networks are used in various applications, including pattern recognition, image classification, natural language processing, and reinforcement learning.

# Machine Learning

A compressive Foundation

# Machine Learning and Its Types

- Machine learning is a part of artificial intelligence that allows models to continuously learn by using past experiences, exploring data, and identifying patterns with little human intervention.

# Types of Machine Learning:

**1.Supervised Learning**

**2.Unsupervised Learning**

**3.Reinforcement Learning**

- Each type reflects a different approach to learning from data and serves a distinct purpose, tailored to specific problems and scenarios in various fields.

*Supervised learning is a method in which we teach the machine using labelled data*



*In unsupervised learning the machine is trained on unlabelled data without any guidance*



*In Reinforcement learning an agent interacts with its environment by producing actions & discovers errors or rewards*

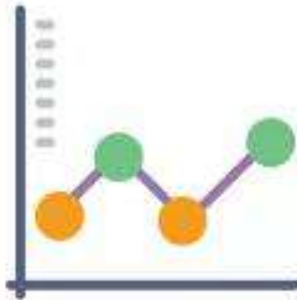




# Problem Type

## Supervised Learning

### Regression

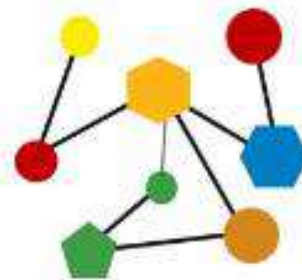


### Classification

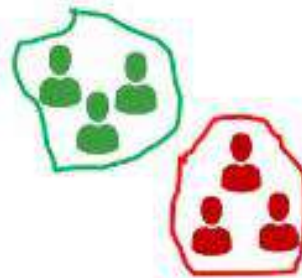


## Unsupervised Learning

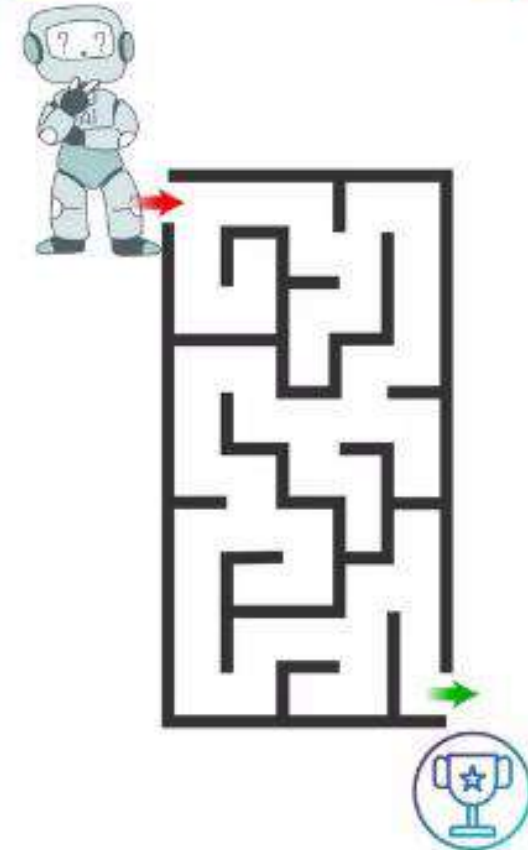
### Association



### Clustering



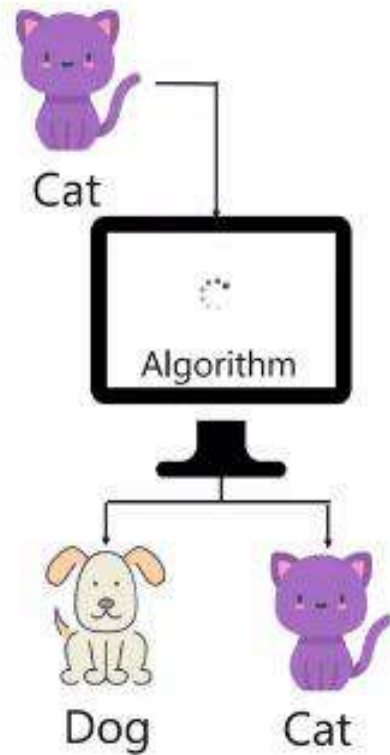
## Reinforcement Learning



# Types of data

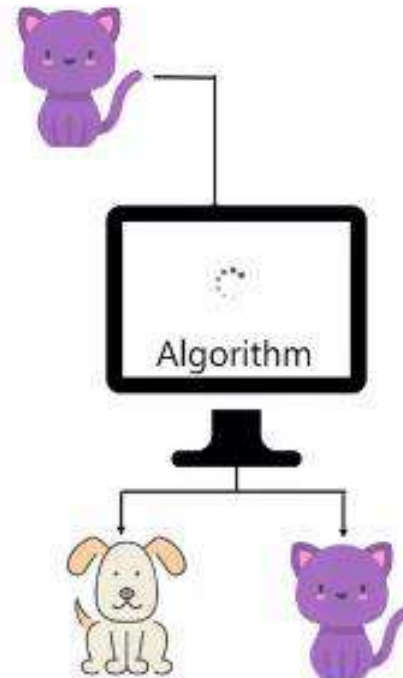
## Supervised Learning

Labelled Data



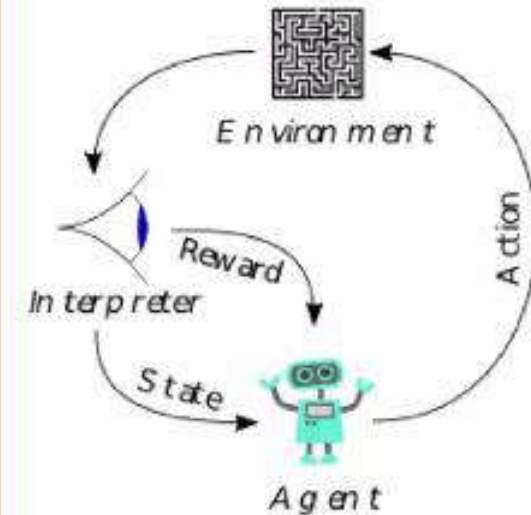
## Unsupervised Learning

Unlabelled Data



## Reinforcement Learning

No Predefined Data



# AIM

## Supervised Learning

Forecast outcomes



## Unsupervised Learning

Discover underlying patterns



Taught BY: Ritu Raj Lamsal

## Reinforcement Learning

Learn series of action





### Supervised Learning

Map labelled input to known output

Labelled Input

Training

Algorithm

Known Output

### Unsupervised Learning

Understand patterns and discover output

Unlabelled Input

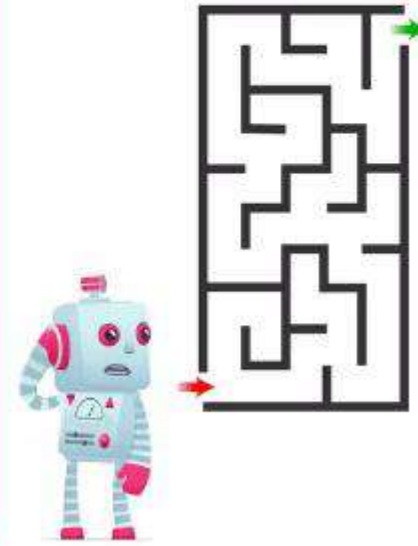
Explore patterns & trends

Algorithm

Output

### Reinforcement Learning

Follow Trail and Error method



# Algorithms

## Supervised Learning

Linear Regression

Logistic Regression

Support Vector  
Machine

K Nearest  
Neighbour

Random Forest

## Unsupervised Learning

K- Means

Apriori

C- Means

## Reinforcement Learning

Q- Learning

SARSA

# Supervised Machine Learning for the Beginners

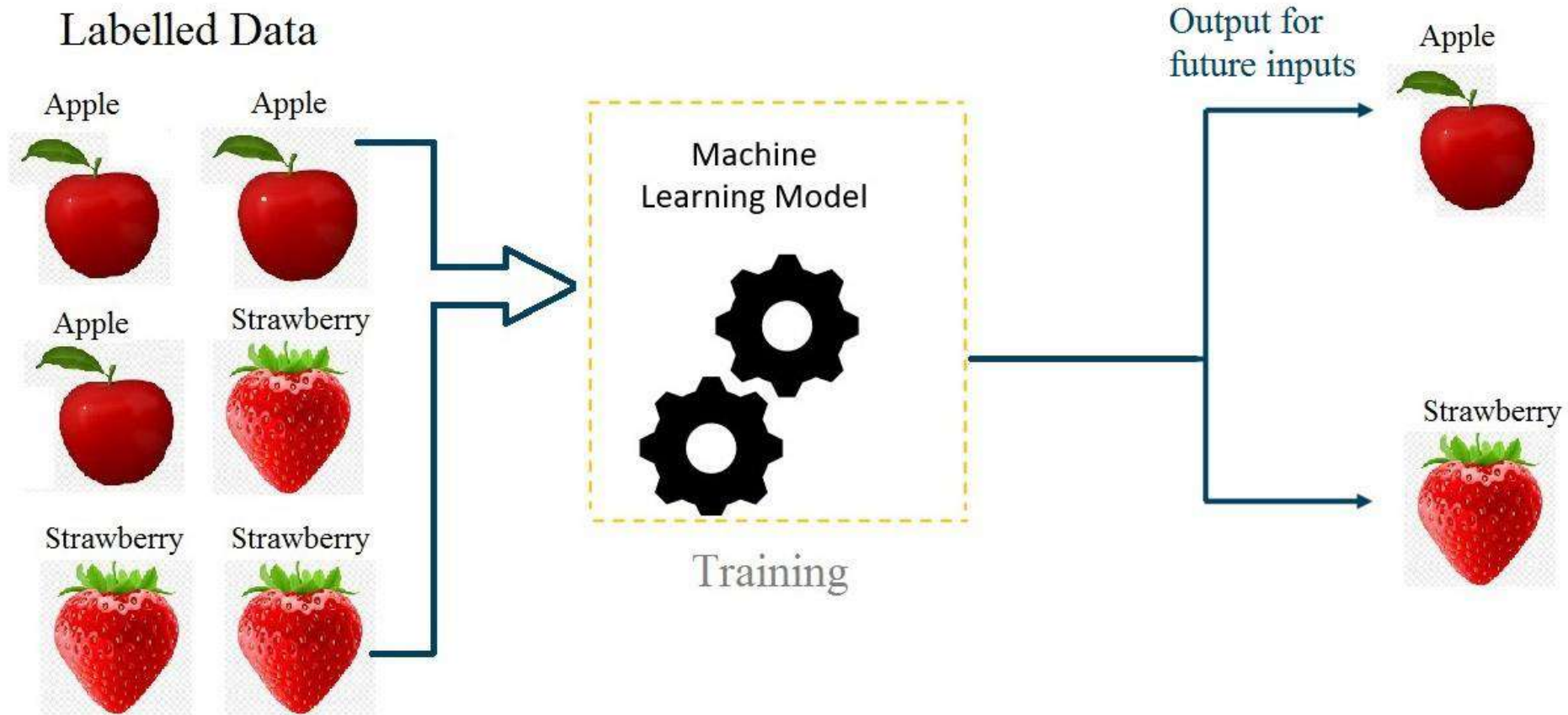
- Imagine teaching a child to recognize and name fruits – an apple, for example. You show them several pictures of apples, and each time, you say, “This is an apple.” The child sees various apples – green ones, red ones, big ones, and small ones. They hear the word “apple” associated with the picture each time. Eventually, the child starts to identify what characteristics make an apple – perhaps the round shape, the stem at the top, or a particular range of colors.
- Now, when you show the child a new picture of an apple, even one they’ve never seen before, they identify it correctly. Why? Because through repeated examples (the training), they’ve learned to associate specific features (shape, color, stem) with the label “apple.”

- In machine learning, this process is akin to supervised learning.
- The **child** is the **algorithm**.
- The **pictures of apples** are the **data**.
- Your repeated saying, “This is an apple,” is the **labeling**.
- The model training is the child’s learning process, identifying features like shape and color.
- The child’s ability to name a new fruit correctly is the **prediction**.



- So, supervised learning is about learning a function from labeled training data that allows us to predict unseen or future data. Just like the child learning from labeled pictures of fruit!

# Supervised learning



# Examples:Supervised

- Image classification (e.g., identifying objects in photographs),
- spam detection (e.g., classifying emails as spam or non-spam),
- sentiment analysis (e.g., determining the sentiment of text).

# Types of Supervised Learning

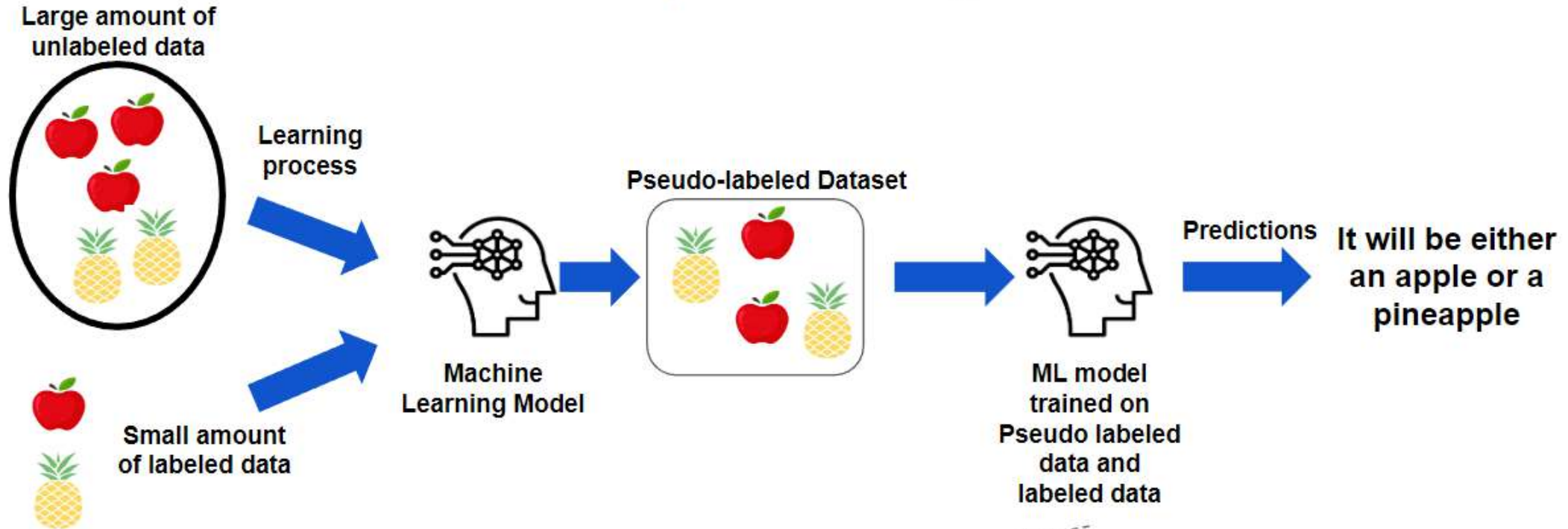
**1.Classification:** In classification tasks, the output variable is a category, such as “spam” or “not spam” in email filtering. The model is trained to categorize the input data into discrete labels. Classification can be binary, with only two categories, or multi-class, with more than two categories of outputs. Examples include image recognition (where inputs are images and outputs are labels), disease diagnosis, and sentiment analysis.

**2.Regression:** In regression tasks, the output variable is a continuous value or a real number, such as “house price” or “temperature”. The model predicts a quantity given the input data. This could involve predicting prices, forecasting weather, estimating values, etc. Regression analysis is about predicting a numerical value based on historical data.

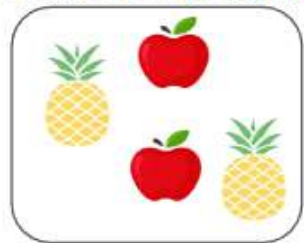
- For classification:
  - **Logistic Regression**
  - **Decision Trees**
  - Support Vector Machines (SVM)
  - **K-Nearest Neighbors (KNN)**
  - **Random Forests**
  - **Neural Networks**
  - Naive Bayes
- For regression:
  - **Linear Regression**
  - Polynomial Regression
  - Support Vector Regression (SVR)
  - Decision Trees and Random Forest Regression
  - Lasso Regression
  - Ridge Regression
  - Elastic Net
  - **Neural Networks**

# Semi Supervised

## Semi-Supervised Learning



Data to be predicted



Trained Machine learning Model



Two groups are identified. Now a human could label group 1 as apples and group 2 as pineapples



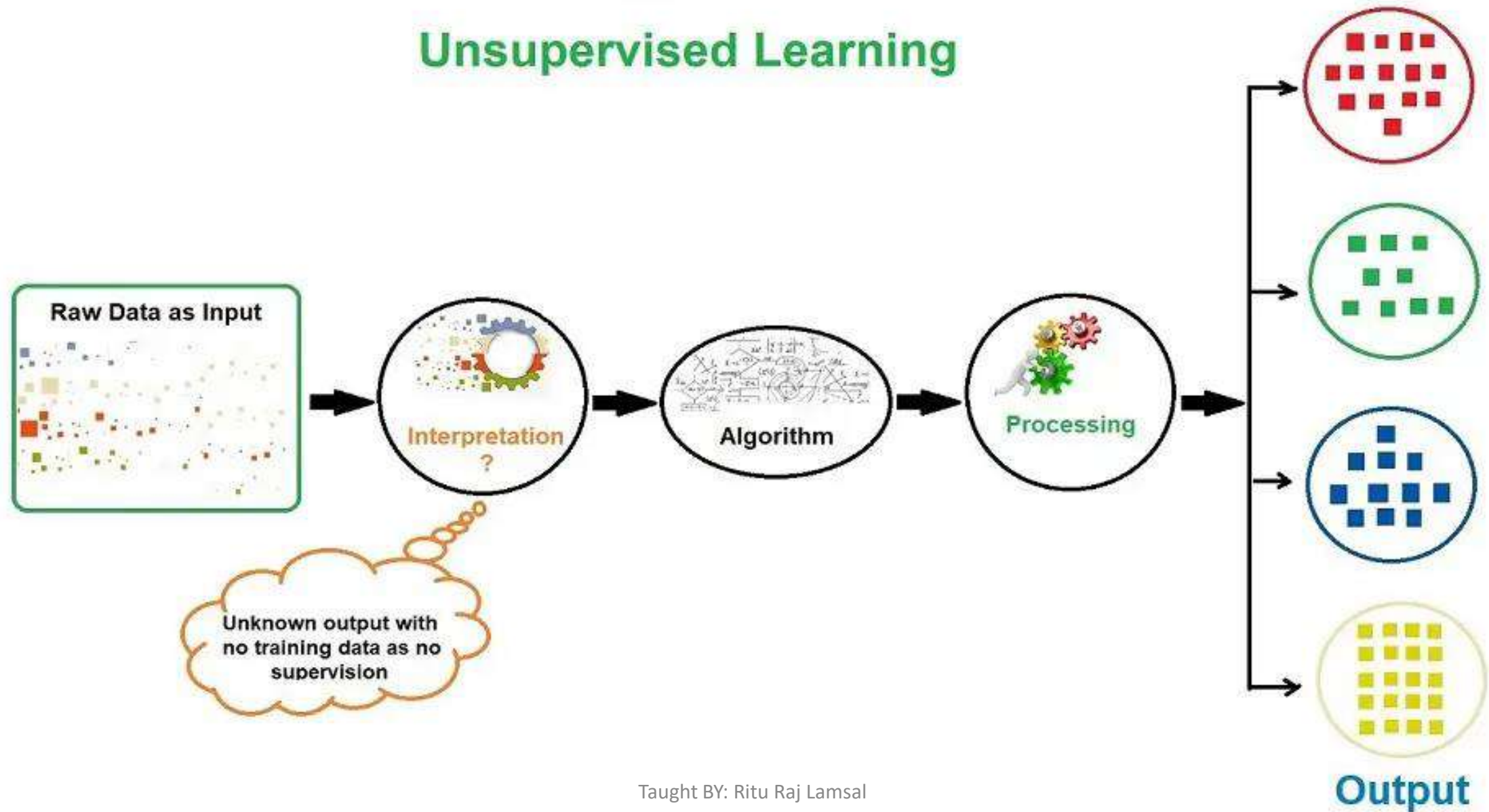
# Unsupervised Machine Learning for Beginners

- Imagine walking into a room full of various toys for the first time – balls, teddy bears, dolls, and cars. Nobody tells you what each item is or how they're grouped.
- However, after exploring the toys, you might naturally start organizing them. Perhaps you group them based on similarities and differences that you observe – all the balls together because they are round, all teddy bears together because they are soft and fluffy, etc.
- You don't know the names or purposes of these toys (since no one told you), but you're finding structure in the collection based on inherent characteristics.
- In unsupervised learning:
- **You** are the **algorithm**.
- The **toys** are the **data**.
- Grouping **similar toys** represents **clustering**, one of the key techniques in unsupervised learning.

- In a practical machine-learning scenario, imagine having a dataset of various flowers without any labels indicating their types. Unsupervised learning algorithms explore the dataset and might group (or cluster) the flowers together based on similar features, such as petal size, shape, or color, without knowing the names of the flower types.

# Unsupervised

## Unsupervised Learning



- **Types of Unsupervised Machine Learning**

- within unsupervised learning, there are several categories based on the nature of the task being performed. Here are the primary ones:

**1. Clustering:** This is the task of grouping a set of objects in such a way that objects in the same group (a cluster) are more similar to each other than to those in other groups. Common clustering algorithms include:

1. K-Means

2. Hierarchical Clustering

3. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

4. Gaussian Mixture Models

5. Mean Shift

**2. Association:** Association rules are used to discover relationships between variables in large databases. It's a rule-based machine learning method for discovering interesting relations between variables in large databases. An example algorithm is:

1. Apriori

2. Eclat

3. FP-Growth

**3. Dimensionality Reduction:** This reduces the number of random variables under consideration and can be divided into feature selection and feature extraction. Techniques include:

1. Principal Component Analysis (PCA)
2. Singular Value Decomposition (SVD)
3. t-Distributed Stochastic Neighbor Embedding (t-SNE)
4. Autoencoders

**4. Anomaly Detection:** The identification of rare items, events, or observations that raise suspicions by differing significantly from most of the data. Techniques include:

1. Isolation Forest
2. One-Class SVM
3. Local Outlier Factor (LOF)
4. Autoencoders (when used for reconstruction error)

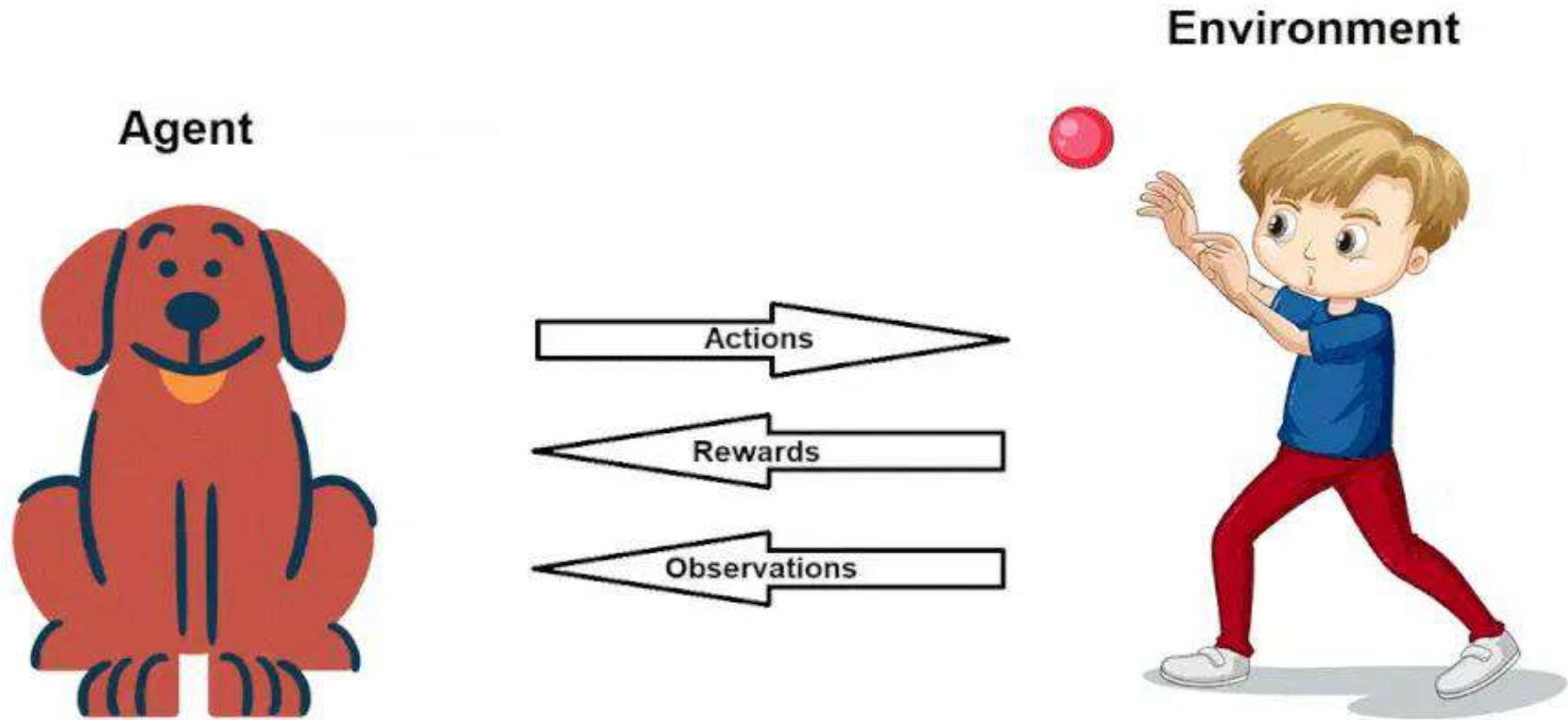
**5. Neural Network-Based Approaches:** With advancements in deep learning, there are unsupervised neural network architectures designed for unsupervised tasks:

1. Generative Adversarial Networks (GANs)
2. Self-Organizing Maps (SOM)
3. Deep Belief Networks (DBNs)
4. Variational Autoencoders (VAEs)





# Reinforcement



# Reinforcement Machine Learning for Beginners

- Imagine training a dog to fetch a ball. The first time you throw the ball, the dog might not know what to do, and it might not retrieve it.
- However, if it brings the ball back, you treat the dog. Quickly, the dog realizes: “Ah, when I fetch the ball, I get a reward!” From then on, it’s likely to fetch the ball to earn more treats. It might refine its approach – fetching it faster or returning it more accurately to your hand – to secure its reward more efficiently.

- The **dog** represents the **learning algorithm or agent**.
- The **environment** is where you and the dog are playing.
- The **action** is fetching the ball.
- The **reward** is the treat – positive if the ball is fetched, perhaps nonexistent, or even negative if not.

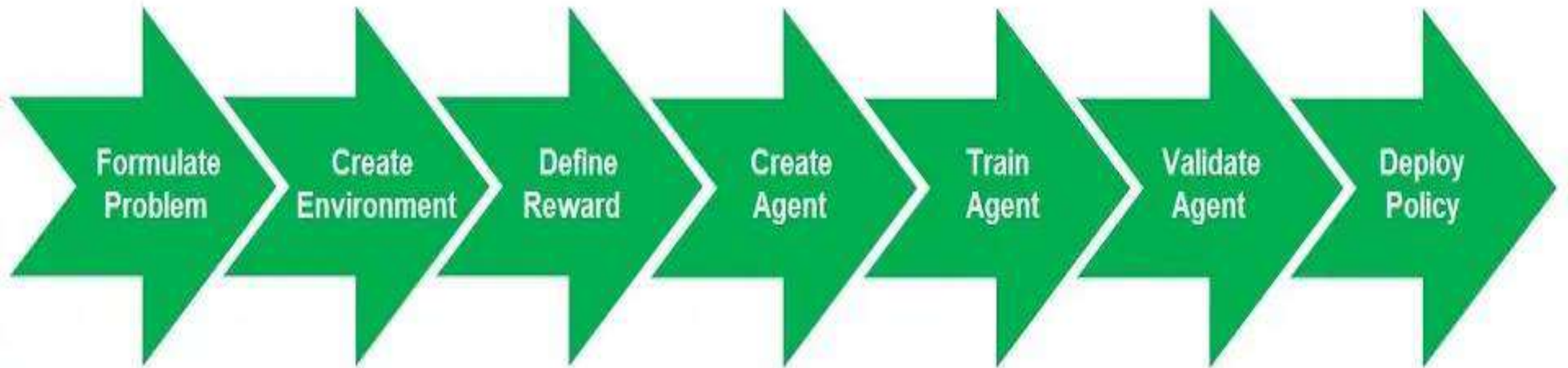
- The agent isn't told which actions to take but is given a reward or penalty based on its chosen actions. Over time, through trial and error, the agent learns the best actions to take in various states to maximize its cumulative reward.
- In a computer science context, consider a chess-playing algorithm:
- The **agent** is the algorithm that plays chess.
- The **environment** is the chessboard.
- The **action** is moving a chess piece.
- The **reward** might be positive for taking an opponent's piece and even larger for winning but negative for losing pieces or the game.
- The algorithm (agent) learns by playing many games (interacting with the environment), gradually improving its strategy (policy) to make moves (actions) that increase its chance of winning (maximizing cumulative reward).

- **Technical Aspects of Reinforcement Machine Learning**
- **Reinforcement Learning** is the third type of Machine Learning and is the training of machine learning models to make a sequence of decisions. It concerns how intelligent agents are to act in an environment to maximize cumulative reward.
- **Reinforcement Learning** stems from Machine Learning. It aims to train a model to return an optimum solution. It uses a sequence of solutions and/or decisions created for a specific problem.

- **Reinforcement Learning terminology**

- Below is a list of terminologies that will be important for you to understand the overall concept of Reinforcement Learning.
- **Actions:** Actions are all the possible steps the model can take, yielding a reward from the Environment. Actions are based on the policy.
- **Agent:** This is the learning phase of Reinforcement Learning. The aim of the Agent is to maximize the rewards the Environment gives it.
- **Environment:** In layman's terms, this is the Agent's home where it lives and interacts. The Agent performs actions within the Environment but cannot influence the dynamics of the Environment due to these actions.
- **Episode:** This is each of the repeated attempts by the Agent to help it learn the Environment. All the states come in between an initial state and a terminal state.
- **Policy:** This determines how an agent behaves and acts as a mapping method between the Agent's present state and actions.
- **Return:** This is the sum of all the rewards the Agent expects to receive. The Agent does this following the policy from the state to the end of the episode.
- **Reward:** This numerical value comes from the Environment and is sent to the Agent. It is a response to the Agent's actions in the Environment. There are three types of rewards: positive (desired action), negative (undesired action), and zero (no action).
- **State:** This is the current configuration of the Environment in which the Agent uses to choose to make an action.

## Reinforcement Learning





# Types of Problems

## Classification Problems

- is a supervised learning task where the goal is to categorize data points into predefined classes or categories.
- Example: Identifying handwritten digits (0-9) from images.

## Regression problems

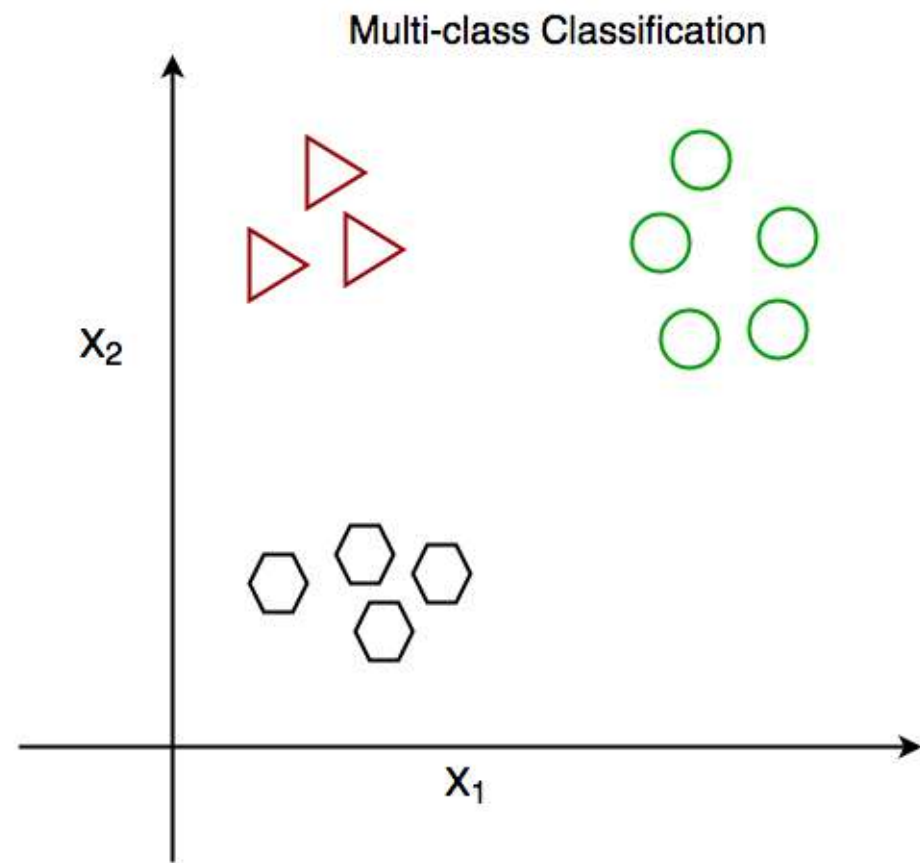
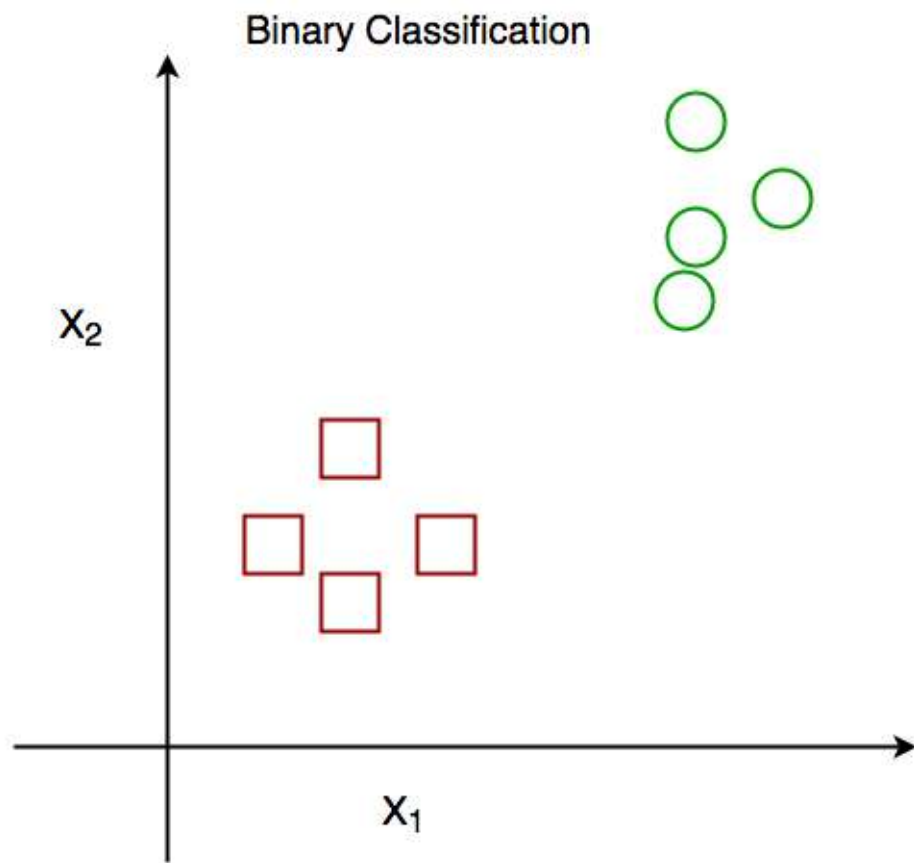
- is a supervised learning task where the goal is to predict continuous numerical values based on input features.
- Example: Predicting house prices based on features such as size, location, and number of bedrooms.

- **Object Detection Problem**

- Definition: Object detection is a computer vision task where the goal is to detect and localize objects within an image or video frame.
- Example: Identifying and locating multiple objects (e.g., cars, pedestrians) in a street scene.

- **Object Segmentation Problem**

- Definition: Object segmentation is a computer vision task where the goal is to partition an image into meaningful segments or regions corresponding to objects.
- Example: Segmenting individual instances of objects in an image (e.g., segmenting people from the background).

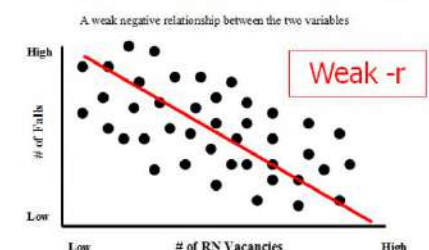
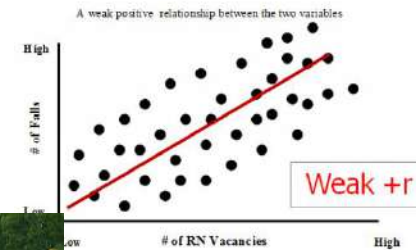
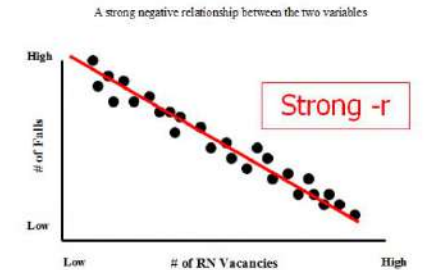
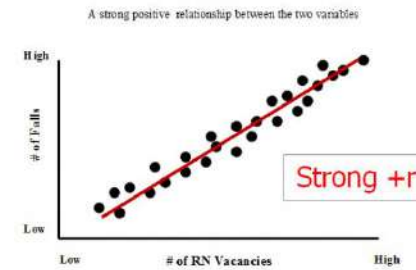
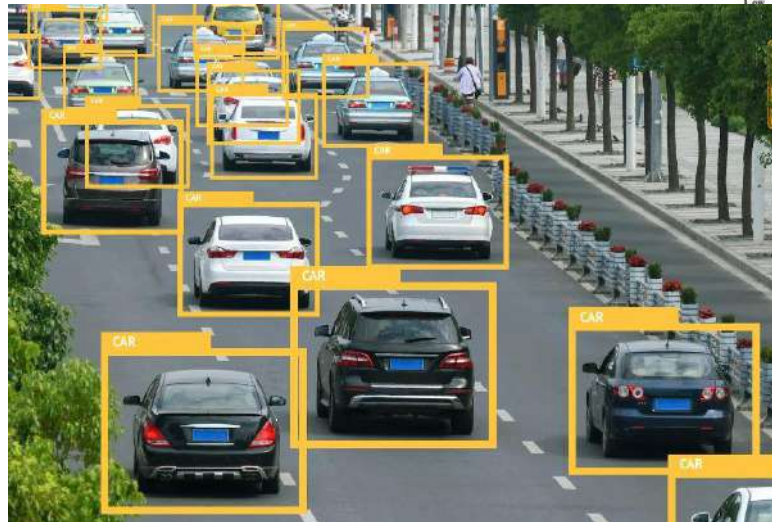


# Handwritten, Scatterplot, object detection,

7 → 7 5 → 5

8 → 8 3 → 3

2 → 2 4 → 4



**Classification**



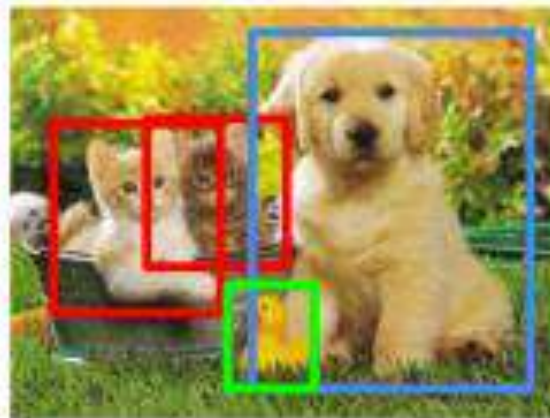
CAT

**Classification  
+ Localization**



CAT

**Object Detection**



CAT, DOG, DUCK

**Instance  
Segmentation**



CAT, DOG, DUCK

Single object

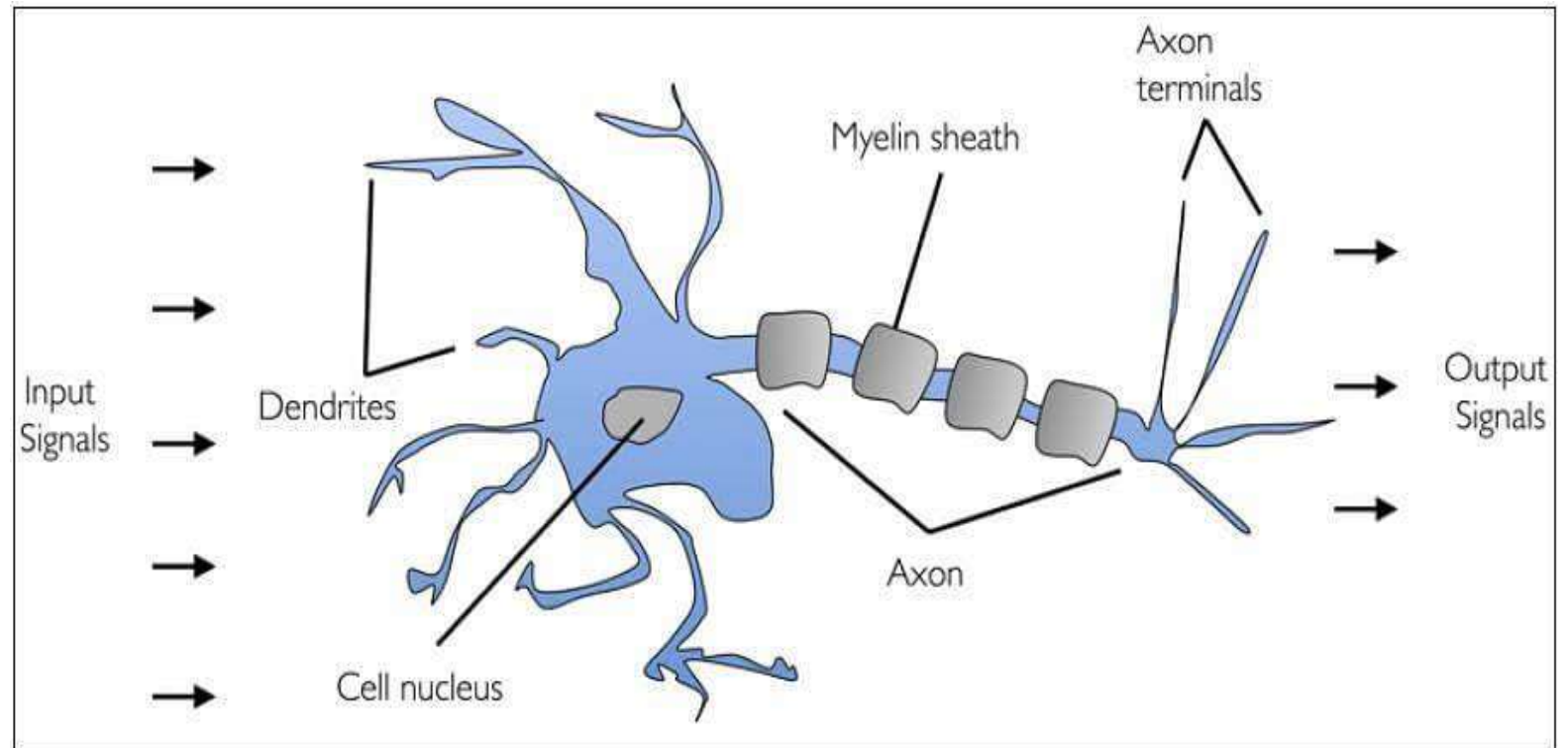
Multiple objects

- Cell nucleus or Soma processes the information received from dendrites. Axon is a cable that is used by neurons to send information. Synapse is the connection between an axon and other neuron dendrites.

## Biological Neuron

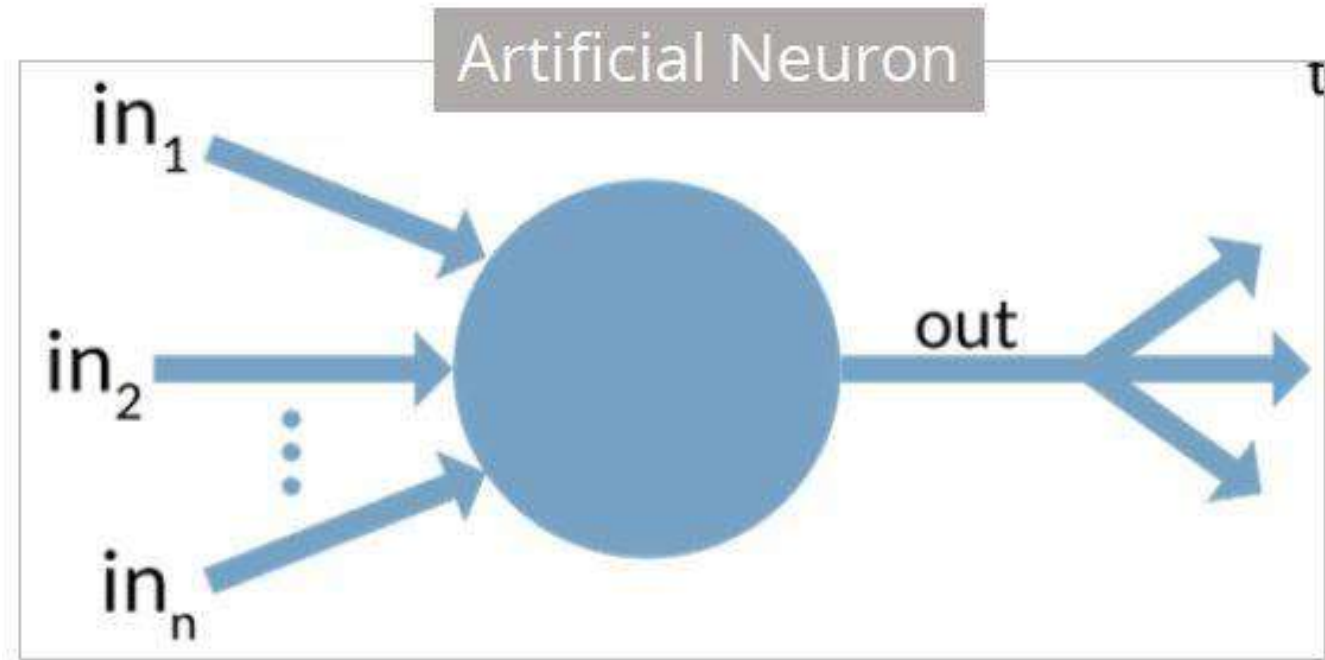
A human brain has billions of neurons. Neurons are interconnected nerve cells in the human brain that are involved in processing and transmitting chemical and electrical signals. Dendrites are branches that receive information from other neurons.

Cell nucleus or Soma processes the information received from dendrites. Axon is a cable that is used by neurons to send information. Synapse is the connection between an axon and other neuron dendrites.

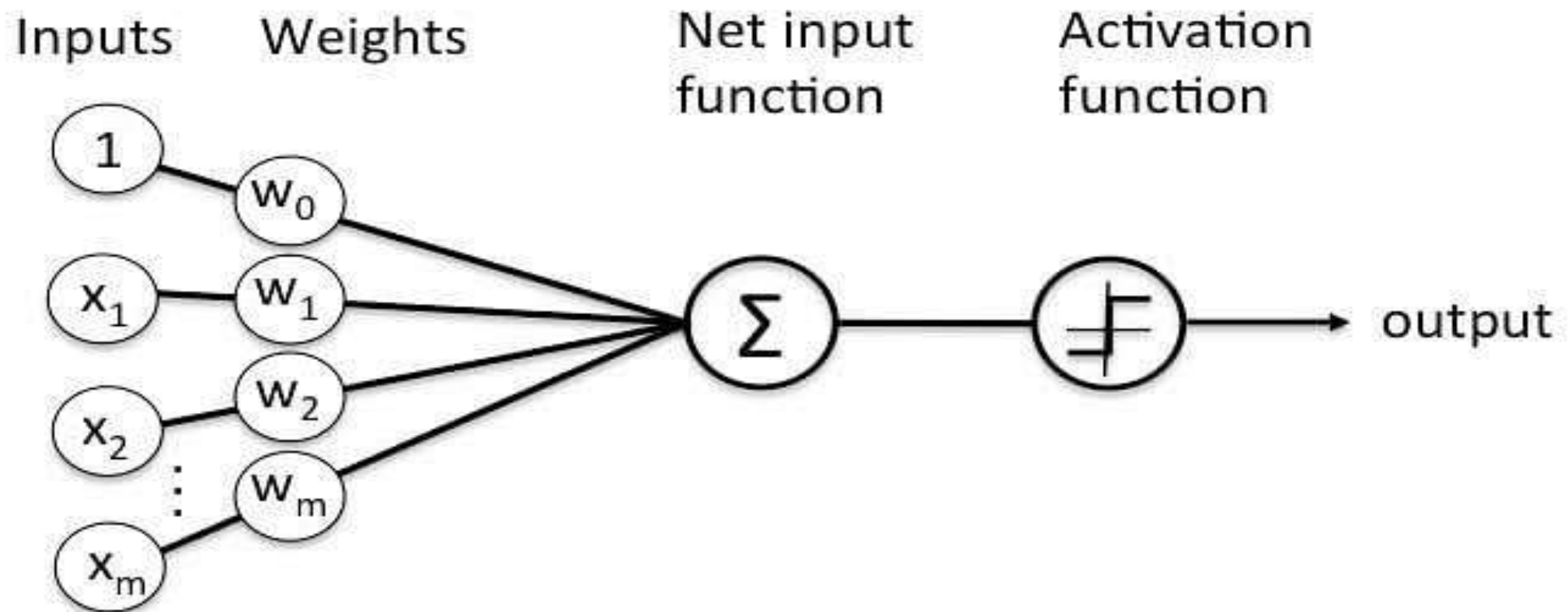




An artificial neuron is a mathematical function based on a model of biological neurons, where each neuron takes inputs, weighs them separately, sums them up and passes this sum through a nonlinear function to produce output.

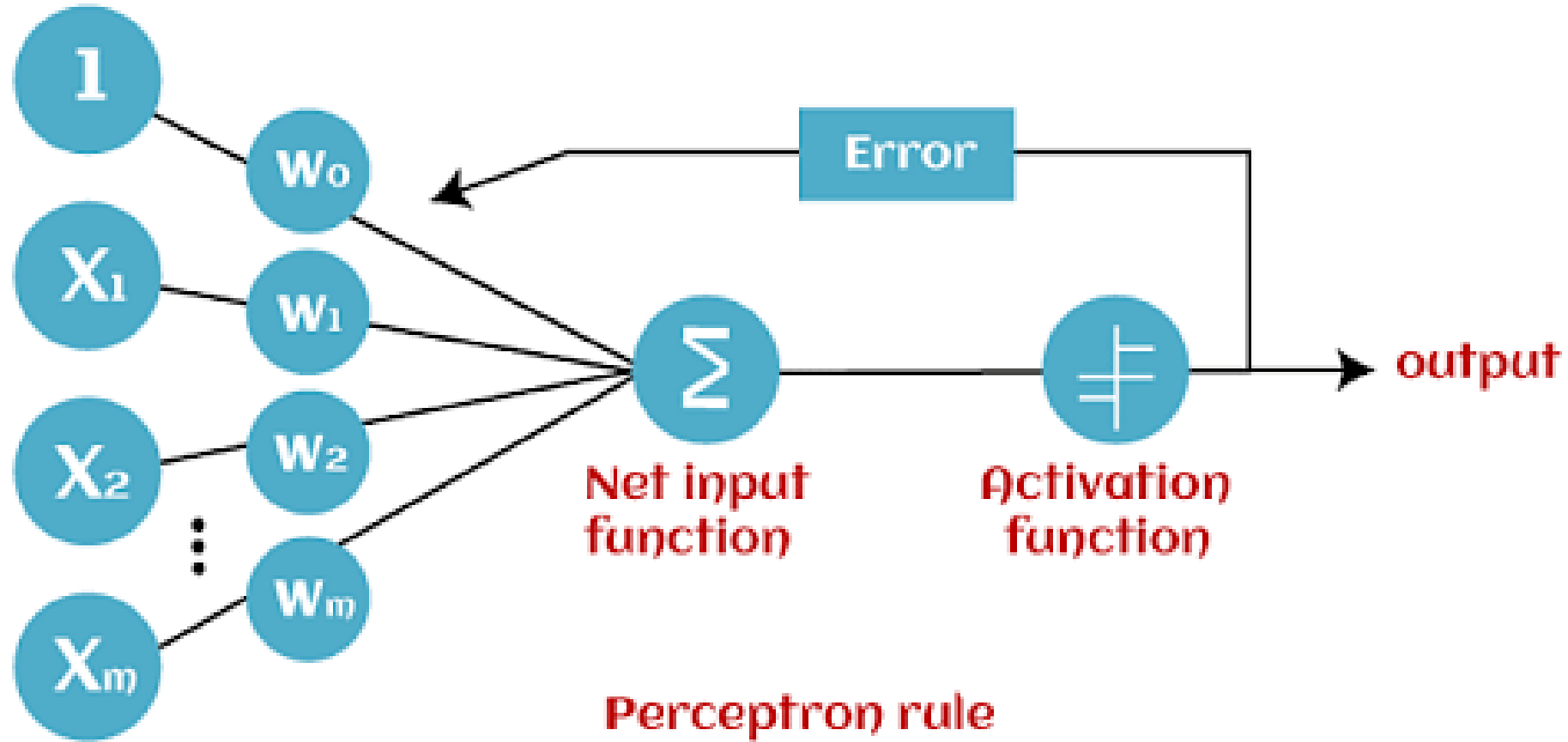


- A Perceptron is an algorithm for supervised learning of binary classifiers. This algorithm enables neurons to learn and processes elements in the training set one at a time.

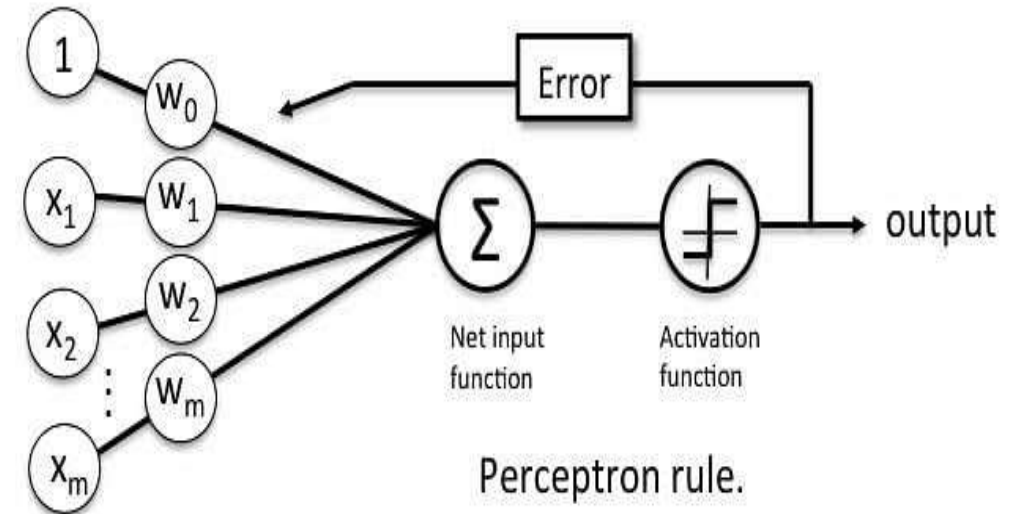


- Perceptron is a type of artificial neural network, which is a fundamental concept in machine learning. The basic components of a perceptron are:
  1. Input Layer: The input layer consists of one or more input neurons, which receive input signals from the external world or from other layers of the neural network.
  2. Weights: Each input neuron is associated with a weight, which represents the strength of the connection between the input neuron and the output neuron.
  3. Bias: A bias term is added to the input layer to provide the perceptron with additional flexibility in modeling complex patterns in the input data.
  4. Activation Function: The activation function determines the output of the perceptron based on the weighted sum of the inputs and the bias term. Common activation functions used in perceptron's include the step function, sigmoid function, and ReLU function.
  5. Output: The output of the perceptron is a single binary value, either 0 or 1, which indicates the class or category to which the input data belongs.
  6. Training Algorithm: The perceptron is typically trained using a supervised learning algorithm such as the perceptron learning algorithm or backpropagation. During training, the weights and biases of the perceptron are adjusted to minimize the error between the predicted output and the true output for a given set of training examples.
  7. Overall, the perceptron is a simple yet powerful algorithm that can be used to perform binary classification tasks and has paved the way for more complex neural networks used in deep learning today.

# How Perceptron works



- A Perceptron accepts inputs, moderates them with certain weight values, then applies the transformation function to output the final result. The image below shows a Perceptron with a Boolean output.
- A Boolean output is based on inputs such as salaried, married, age, past credit profile, etc. It has only two values: Yes and No or True and False. The summation function “ $\Sigma$ ” multiplies all inputs of “x” by weights “w” and then adds them up as follows:



$$w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n$$

# Types of Perceptron:

1. Single layer: Single layer perceptron can learn only linearly separable patterns.

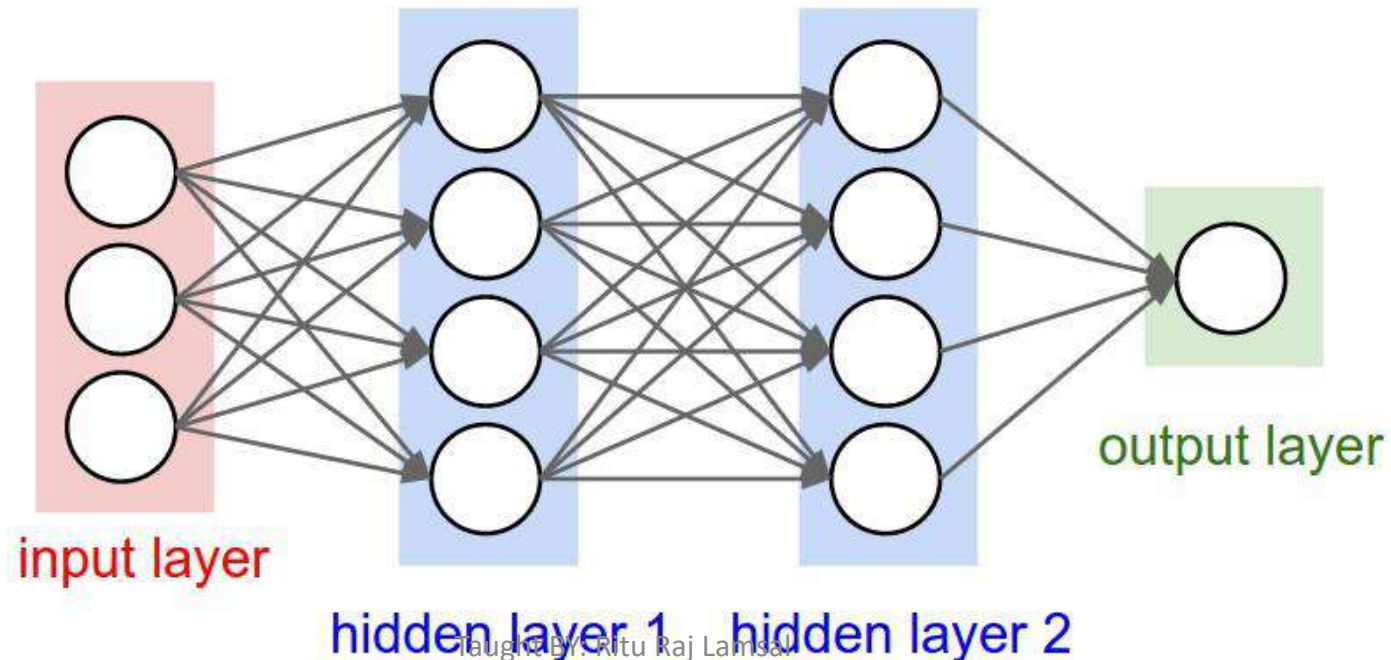
( AND , OR, function are linearly separable where as Xor is not)

1. Multilayer: Multilayer perceptron's can learn about two or more layers having a greater processing power.

- The Perceptron algorithm learns the weights for the input signals in order to draw a linear decision boundary.

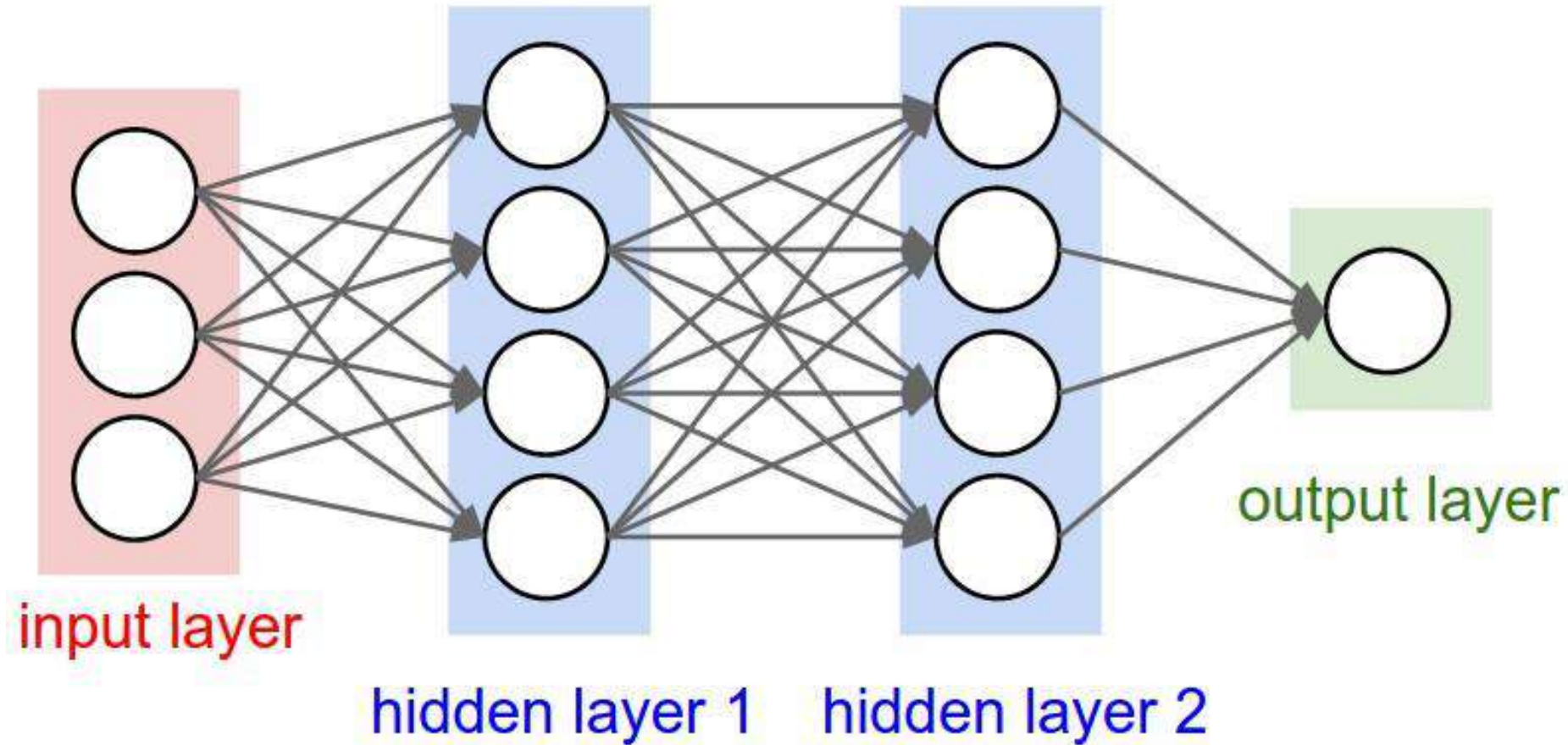
# Artificial Neural Networks (ANNs)"

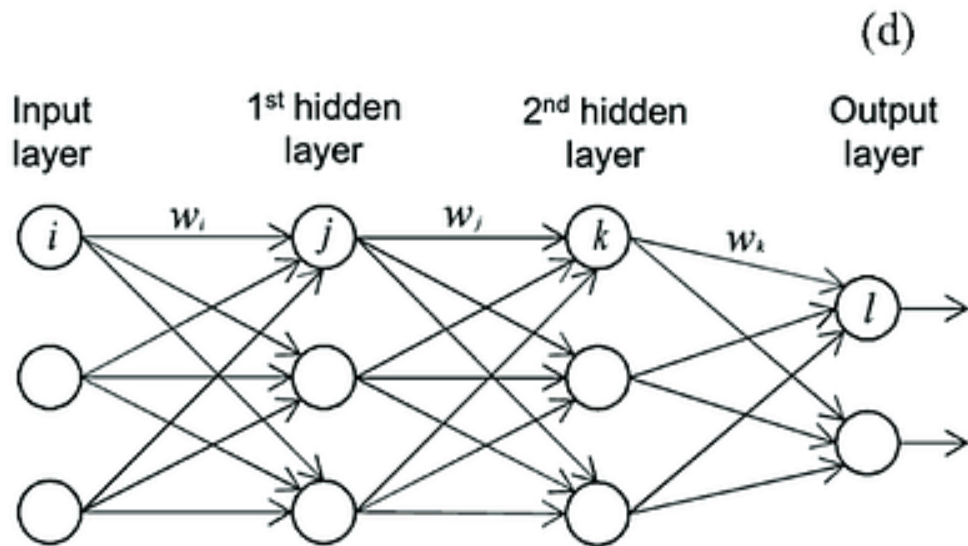
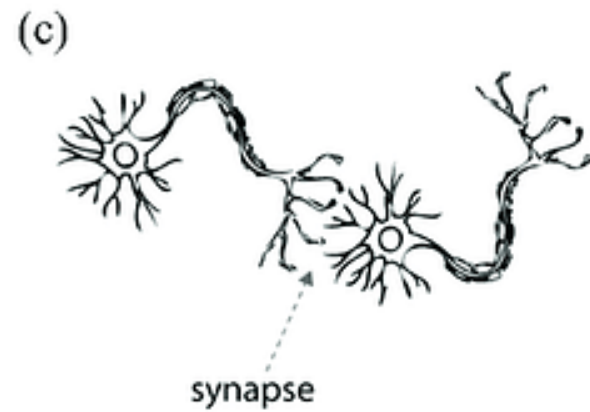
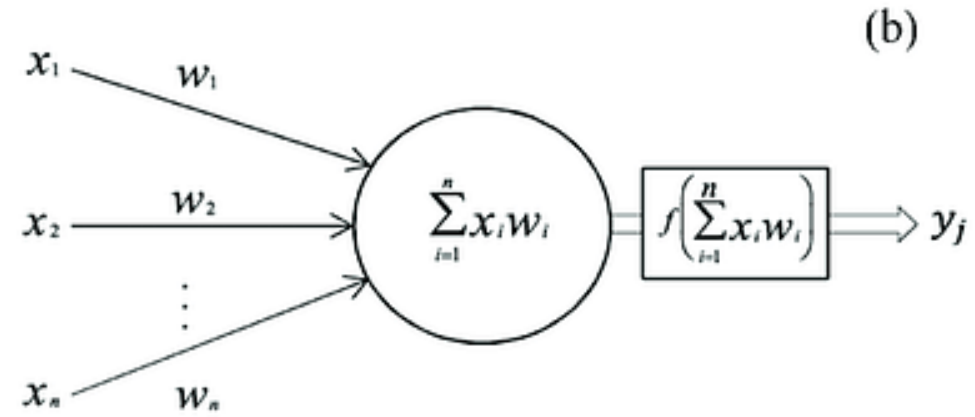
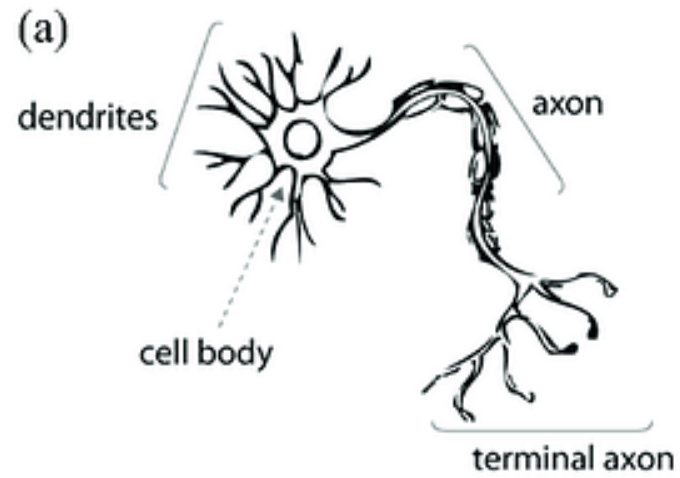
- represent a class of machine learning models inspired by the biological structure and functioning of the human brain. ANNs consist of interconnected nodes, or artificial neurons, organized into layers. These networks are capable of learning complex patterns and relationships from data through a process called training.





# Architecture of ANN

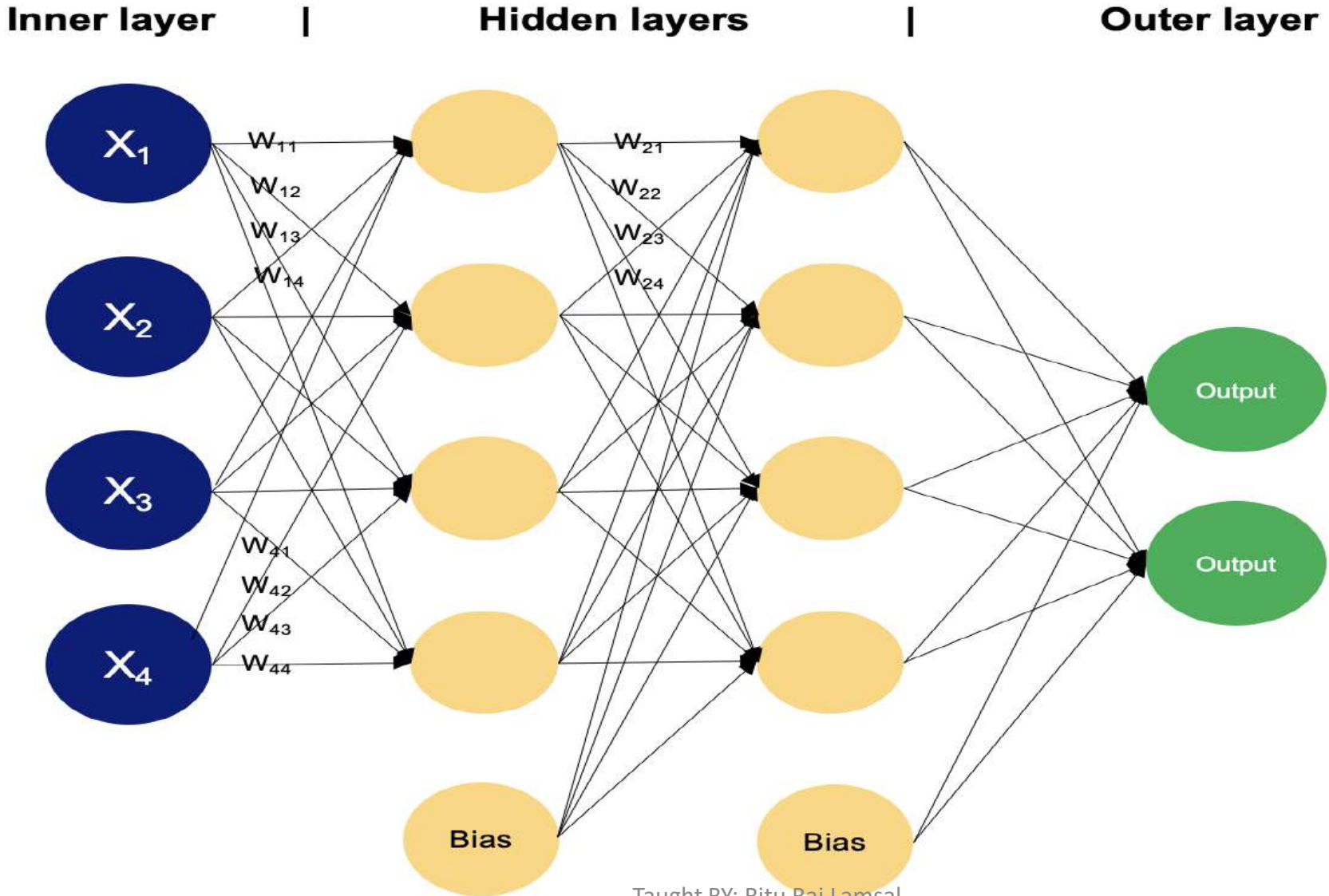




# Multi-layer Perceptron (MLP)

- One of the most widely used architectures of ANNs is the "Multi-layer Perceptron (MLP)." MLPs consist of an input layer, one or more hidden layers, and an output layer. Each neuron in one layer is connected to every neuron in the subsequent layer, and each connection is associated with a weight that adjusts during the training process. MLPs are versatile and can be applied to various tasks, including classification, regression, and pattern recognition.

# Architecture of MLP



- 1. Neurons and Layers:** The basic building block of an MLP is the neuron, which is a computational unit that receives input, performs a weighted sum of its inputs, applies an activation function to the sum, and then outputs the result. Neurons are organized into layers: an input layer, one or more hidden layers, and an output layer.
- 2. Feedforward Architecture:** MLPs use a feedforward architecture, meaning that information flows in one direction—from the input layer, through the hidden layers, to the output layer—without any cycles or loops.
- 3. Weights and Bias:** Each connection between neurons in adjacent layers is associated with a weight, which determines the strength of the connection. Additionally, each neuron typically has a bias term, which allows the network to learn offset values. Both weights and biases are parameters that the network learns during training.
- 4. Activation Functions:** Neurons in MLPs typically use activation functions to introduce nonlinearity into the model, allowing it to learn complex patterns in the data. Common activation functions include the sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU).

**4.Forward Propagation:** During forward propagation, the input data is fed into the input layer, and the activations propagate through the network, layer by layer, until the output layer produces a prediction. Each neuron's output is calculated based on the weighted sum of its inputs and the activation function.

**5.Backpropagation:** Backpropagation is the process of updating the weights of the network to minimize the difference between the predicted output and the actual output (the error). It involves computing the gradient of the loss function with respect to the weights using the chain rule of calculus and adjusting the weights in the direction that reduces the error.

**6.Training:** Training an MLP involves iteratively presenting training examples to the network, calculating the prediction error, and updating the weights using backpropagation. This process continues until the model's performance on a validation set converges or reaches a satisfactory level.

- **Applications:** MLPs are used for a wide range of tasks, including classification, regression, pattern recognition, and function approximation. They have been successfully applied in areas such as image recognition, natural language processing, and financial forecasting.





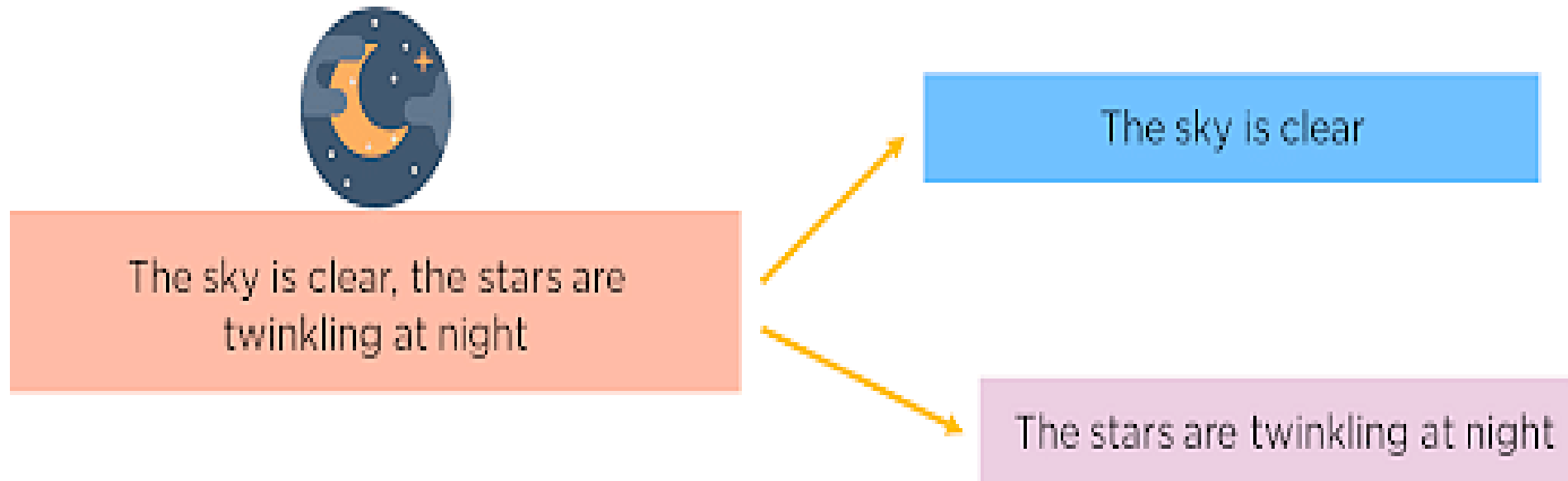
# Natural Language Processing

- Natural Language Processing or NLP refers to the branch of [Artificial Intelligence](#) that gives the machines the ability to read, understand and derive meaning from human languages.

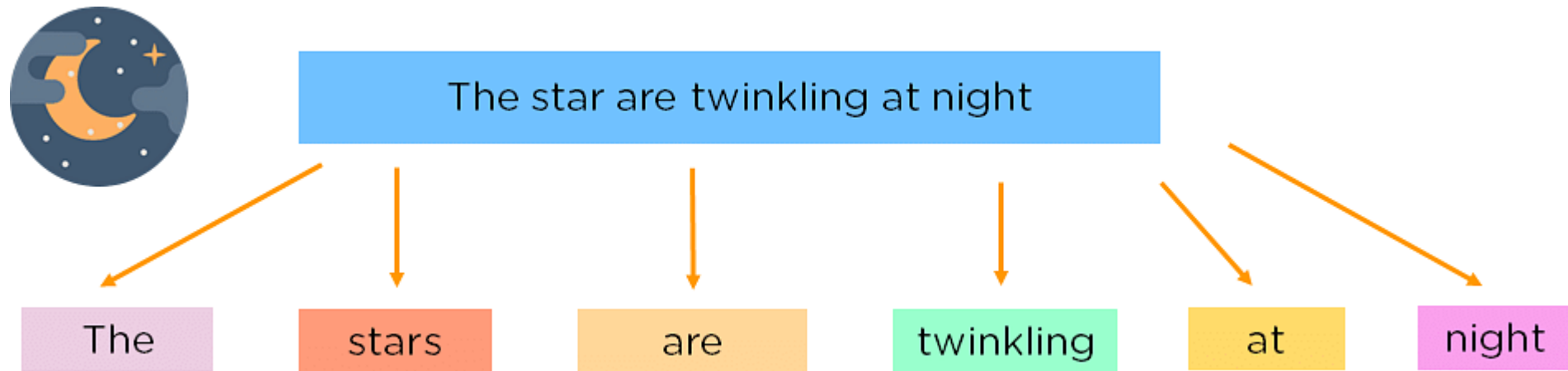
# Steps in NLP preprocessing

- **Segmentation:**

You first need to break the entire document down into its constituent sentences. You can do this by segmenting the article along with its punctuations like full stops and commas.



- Tokenizing:
- For the algorithm to understand these sentences, you need to get the words in a sentence and explain them individually to our algorithm. So, you break down your sentence into its constituent words and store them. This is called tokenizing, and each word is called a token.



- Removing Stop Words:
- You can make the learning process faster by getting rid of non-essential words, which add little meaning to our statement and are just there to make our statement sound more cohesive. Words such as was, in, is, and, the, are called stop words and can be removed.

The star are twinkling at night



stars

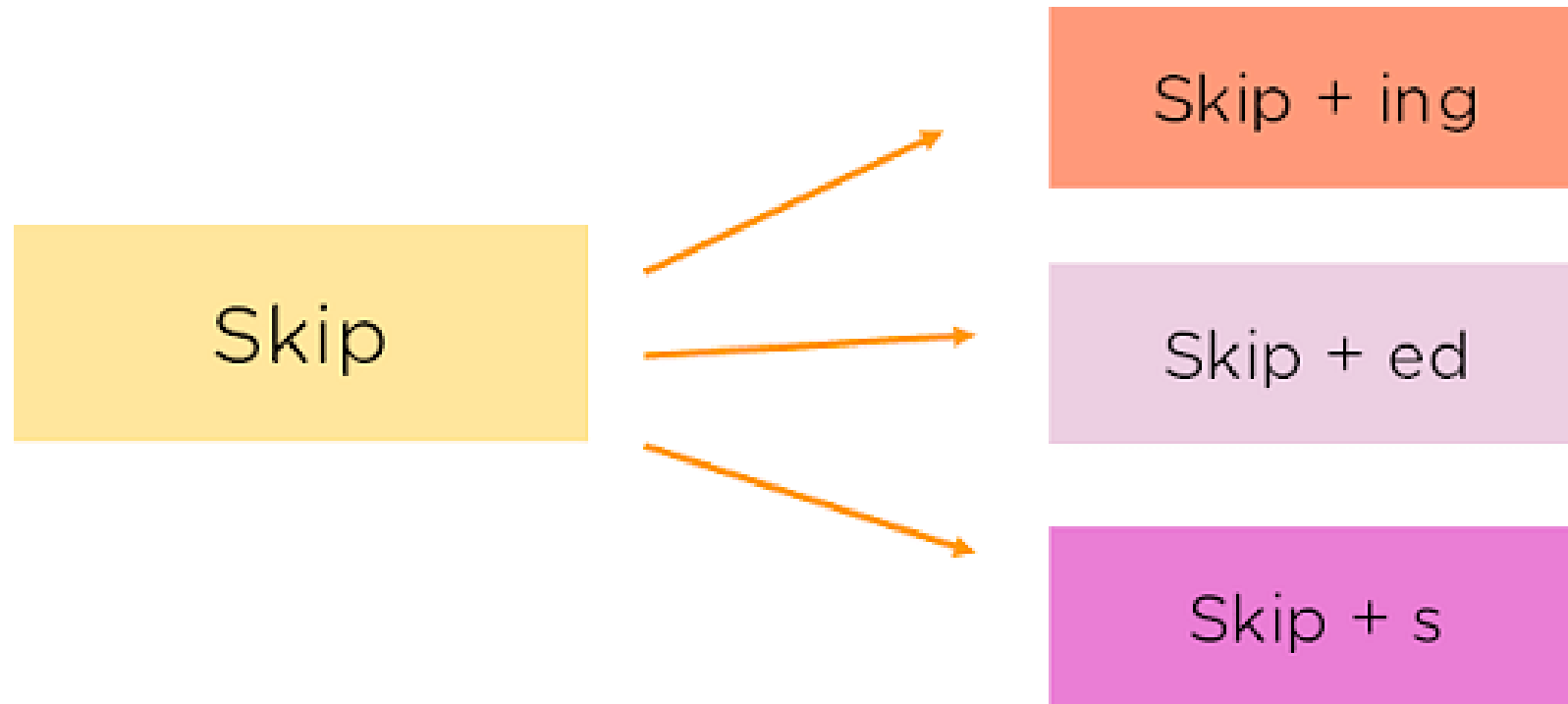


twinkling

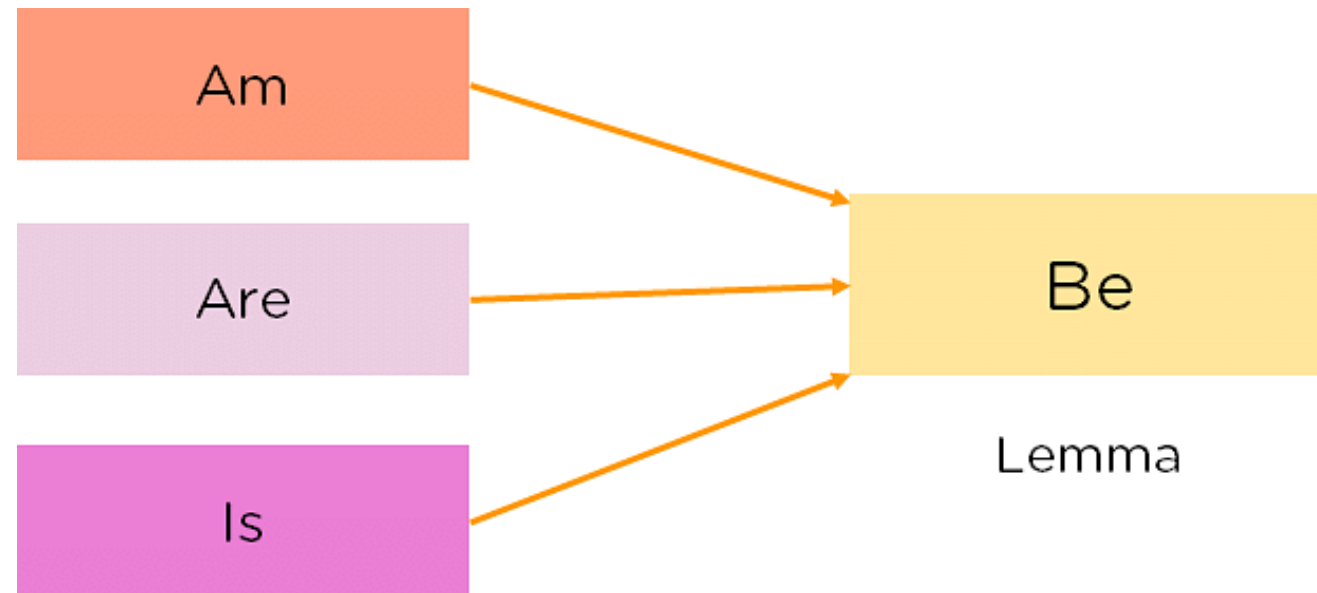


night

- Stemming:
- It is the process of obtaining the Word Stem of a word. Word Stem gives new words upon adding affixes to them

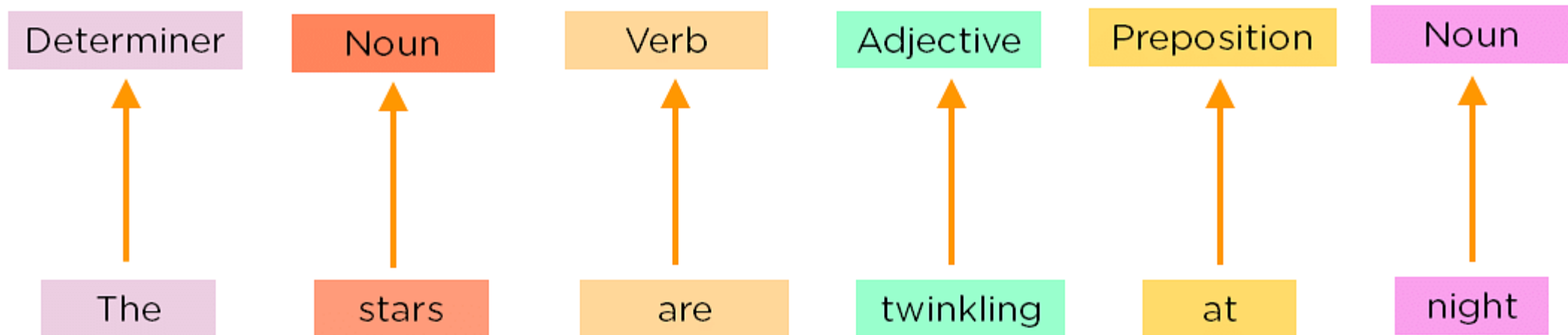


- Lemmatization:
- The process of obtaining the Root Stem of a word. Root Stem gives the new base form of a word that is present in the dictionary and from which the word is derived. You can also identify the base words for different words based on the tense, mood, gender, etc.





- Part of Speech Tagging:
- Now, you must explain the concept of nouns, verbs, articles, and other parts of speech to the machine by adding these tags to our words. This is called 'part of'.



# Applications of NLP

- NLP is one of the ways that people have humanized machines and reduced the need for labor. It has led to the automation of speech-related tasks and human interaction. Some applications of NLP include :
- Translation Tools: Tools such as Google Translate, Amazon Translate, etc. translate sentences from one language to another using NLP.
- Chatbots: Chatbots can be found on most websites and are a way for companies to deal with common queries quickly.
- Virtual Assistants: Virtual Assistants like Siri, Cortana, Google Home, Alexa, etc can not only talk to you but understand commands given to them.
  - Targeted Advertising: Have you ever talked about a product or service or just googled something and then started seeing ads for it? This is called targeted advertising, and it helps generate tons of revenue for sellers as they can reach niche audiences at the right time.
- Autocorrect: Autocorrect will automatically correct any spelling mistakes you make, apart from this grammar checkers also come into the picture which helps you write flawlessly.

# Image

- An image is defined as a two-dimensional function,  $F(x,y)$ , where  $x$  and  $y$  are spatial coordinates, and the amplitude of  $F$  at any pair of coordinates  $(x,y)$  is called the **intensity** of that image at that point. When  $x,y$ , and amplitude values of  $F$  are finite, we call it a **digital image**.  
In other words, an image can be defined by a two-dimensional array specifically arranged in rows and columns.  
Digital Image is composed of a finite number of elements, each of which elements have a particular value at a particular location. These elements are referred to as *picture elements, image elements, and pixels*.

# Types of an image

- 1.BINARY IMAGE**– The binary image as its name suggests, contain only two pixel elements i.e 0 & 1,where 0 refers to black and 1 refers to white. This image is also known as Monochrome.
- 2.BLACK AND WHITE IMAGE**– The image which consist of only black and white color is called BLACK AND WHITE IMAGE.
- 3.8 bit COLOR FORMAT**– It is the most famous image format.It has 256 different shades of colors in it and commonly known as Grayscale Image. In this format, 0 stands for Black, and 255 stands for white, and 127 stands for gray.
- 4.16 bit COLOR FORMAT**– It is a color image format. It has 65,536 different colors in it.It is also known as High Color Format. In this format the distribution of color is not as same as Grayscale image.

# Image classification

- Image classification is the process of categorizing images into predefined classes or categories based on their visual content.
- It's a fundamental task in computer vision and has numerous real-world applications, including object detection, face recognition, medical image analysis, and autonomous vehicles.

# Steps in Image classification process

- 1.Data Collection:** Gathering a large dataset of labeled images, where each image is associated with a specific class or category.
- 2.Preprocessing:** This step involves preparing the data for training by resizing images to a uniform size, normalizing pixel values, and augmenting the dataset with techniques like rotation, flipping, or adding noise to increase its diversity.
- 3.Feature Extraction:** Extracting meaningful features from the images that can be used to distinguish between different classes. This can be done using various techniques such as convolutional neural networks (CNNs), which are particularly effective at capturing hierarchical features in images.
- 4.Model Training:** Training a classification model using the labeled dataset and the extracted features. Popular algorithms for image classification include CNN architectures like AlexNet, VGG, ResNet, and Inception, which have been pretrained on large datasets like ImageNet and then fine-tuned for specific tasks.
- 5.Evaluation:** Assessing the performance of the trained model using metrics such as accuracy, precision, recall, and F1 score on a separate validation dataset. This helps determine how well the model generalizes to unseen data.
- 6.Deployment:** Deploying the trained model into production environments where it can classify new images in real-time.

# Audio identification

- Audio identification, also known as audio recognition or audio fingerprinting, is the process of automatically identifying or recognizing audio signals based on their unique characteristics.



- 1.Audio Representation:** Audio signals are represented as waveforms, which are essentially variations in air pressure over time. These waveforms can be converted into digital representations using techniques like analog-to-digital conversion (ADC), resulting in discrete samples of the audio signal.
- 2.Feature Extraction:** Extracting relevant features from the audio signal is crucial for identification. Common features include spectrograms, which represent the frequency content of the audio signal over time, and Mel-frequency cepstral coefficients (MFCCs), which capture the spectral characteristics of the audio signal.
- 3.Fingerprinting:** Audio fingerprinting involves creating a compact and robust representation of the audio signal that can be used for identification. This is typically achieved by extracting key features from the audio signal and converting them into a fingerprint, which is essentially a unique identifier for the audio snippet.

- 1.Database Creation:** A database of reference audio fingerprints is created by analyzing and fingerprinting a large collection of audio files. Each fingerprint is associated with metadata such as song title, artist, and album.
- 2.Matching:** When a new audio sample needs to be identified, its fingerprint is computed and compared against the fingerprints in the reference database using algorithms like locality-sensitive hashing (LSH) or dynamic time warping (DTW). The closest match or matches are then retrieved from the database.

- 1.Applications:** music recognition (identifying songs based on short audio clips), content-based audio retrieval (finding audio files similar to a query), copyright protection (detecting unauthorized use of copyrighted audio content), and audio surveillance (identifying specific audio events or patterns in recordings).
- 2.Challenges:** dealing with noise and distortions in the audio signal, handling variations in recording conditions and quality, scalability to large audio databases, ensuring robustness against deliberate attacks or tampering.

# Fundamentals of Knowledge Representation in AI

- **Knowledge Representation (KR)** is a fundamental characteristic of artificial intelligence (AI) focused on how information and knowledge are stored and manipulated within an AI system.
- Knowledge Representation enables AI to understand, reason, and draw conclusions similar to human cognition.

## **Types of Knowledge Representation**

- **Logical Representation**
- **Semantic Networks**
- **Frames**
- **Production Rules**
- **Ontologies**
- **Bayesians Networks**

## 1. Logical Representation

- Uses formal logic to represent knowledge.
- **Propositional Logic:** Represents facts using statements (propositions) that can be true or false.
- **Predicate Logic:** Extends propositional logic by dealing with objects and their relationships.

## **2.Semantic Networks**

- Represents knowledge in a graph of nodes (concepts) and edges (relationships).
- Useful for visualizing hierarchical relationships.



### 3. Frames

- Structured representations of stereotypical situations.
- Each frame consists of slots (attributes) and fillers (values).

## 4. Production Rules

- Uses if-then rules to represent knowledge.
- **If Condition (antecedent) Then Action (consequent).**
- Useful in expert systems.

## 5. Ontologies

- Formal representation of a set of concepts and their relationships within a domain.
- Facilitates sharing and reuse of knowledge.

## 6. Bayesian Networks

- Probabilistic graphical models representing variables and their conditional dependencies.
- Useful for reasoning under uncertainty.

### 3. Components of Knowledge Representation

1. **Syntax:** Defines the symbols and structure used to represent knowledge.
2. **Semantics:** Defines the meaning of the symbols and structures.
3. **Pragmatics:** Defines how the symbols and structures are used in practice.

## 4. Important Concepts in KR

- **Entities:** Objects, individuals, or things that exist in the domain.
- **Attributes:** Properties or characteristics of entities.
- **Relations:** Connections between entities.
- **Events:** Actions or occurrences involving entities.
- **Facts:** Specific assertions about entities, attributes, relations, or events.

## 5. Key Challenges in KR

- **Representation Adequacy:** Ensuring the representation captures all necessary knowledge.
- **Inferencing:** Enabling the system to draw conclusions from the represented knowledge.
- **Efficiency:** Ensuring that knowledge representation and inferencing are computationally feasible.
- **Scalability:** Ability to handle large amounts of knowledge.

## 6. Applications of KR

- **Expert Systems:** Systems that mimic human experts in specific domains.
- **Natural Language Processing (NLP):** Understanding and generating human language.
- **Robotics:** Enabling robots to understand and interact with their environment.
- **Decision Support Systems:** Helping humans make informed decisions.



# Fundamentals of Artificial Intelligence

## Chapter 01: Introduction to A.I.

Roberto Sebastiani

DISI, Università di Trento, Italy – [roberto.sebastiani@unitn.it](mailto:roberto.sebastiani@unitn.it)  
[http://disi.unitn.it/rseba/DIDATTICA/fai\\_2020/](http://disi.unitn.it/rseba/DIDATTICA/fai_2020/)

Teaching assistant: **Mauro Dragoni** – [dragoni@fbk.eu](mailto:dragoni@fbk.eu)  
<http://www.maurodragoni.com/teaching/fai/>

M.S. Course “Artificial Intelligence Systems”, academic year 2020-2021

Last update: Tuesday 8<sup>th</sup> December, 2020, 15:17

Copyright notice: *Most examples and images displayed in the slides of this course are taken from*

*[Russell & Norvig, “Artificial Intelligence, a Modern Approach”, Pearson, 3<sup>rd</sup> ed.], including explicitly figures from the above-mentioned book, and their copyright is detained by the authors.*

*A few other material (text, figures, examples) is authored by (in alphabetical order):*

*Pieter Abbeel, Bonnie J. Dorr, Anca Dragan, Dan Klein, Nikita Kitaev, Tom Lenaerts, Michela Milano, Dana Nau, Maria Simi, who detain its copyright. These slides cannot can be displayed in public without the permission of the author.*

# Outline

- 1 AI: Fiction vs. Reality
- 2 What is AI?
- 3 Foundations and History of AI
- 4 AI: State of the Art

# Outline

- 1 AI: Fiction vs. Reality
- 2 What is AI?
- 3 Foundations and History of AI
- 4 AI: State of the Art

There is plenty of AI in fiction ...

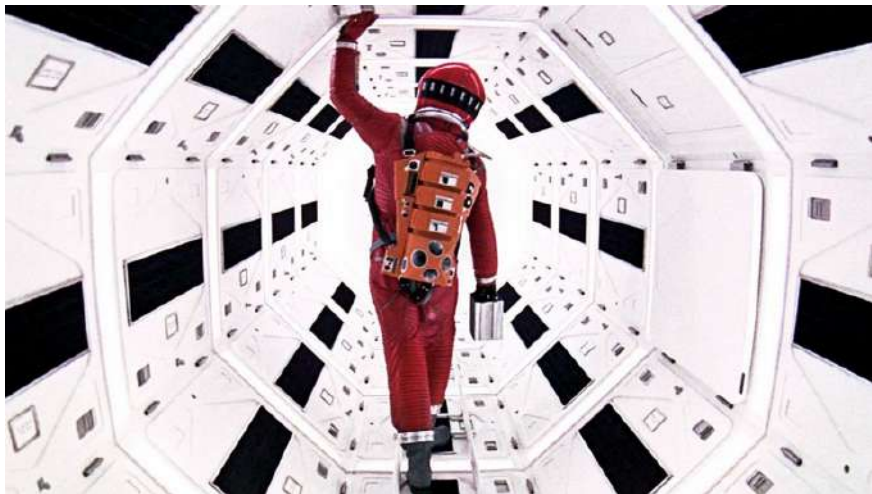
# AI in Fiction



“Metropolis”, 1927, by Fritz Lang

... and many others ...

## AI in Fiction



"2001, Space Odyssey", 1968, by Stanley Kubrick

... and many others ...

# AI in Fiction



“Star Wars”, 1977, by George Lucas

... and many others ...

## AI in Fiction



“Blade Runner”, 1982, by Ridley Scott

... and many others ...



# AI in Fiction



“Terminator”, 1984, by James Cameron

... and many others ...

## AI in Fiction



“A.I., Artificial Intelligence”, 2001, by Steven Spielberg

... and many others ...

# AI in Fiction



"I, Robot", 2004, by Alex Proyas

... and many others ...

# AI in Fiction



“Wall-E”, 2008, by Andrew Stanton

... and many others ...

## AI in Fiction



“Ex Machina”, 2015, by Alex Garland

... and many others ...

# AI in Fiction



“Blade Runner, 2049”, 2017, by Denis Villeneuve

... and many others ...

... and many others ...

(see, e.g., <https://www.looper.com/198685/the-stunning-evolution-of-ai-in-movies/>)

Many AI fantasies from fiction are becoming reality ...



... self-driving cars, ...



©WATMO Inc.

... autonomous vacuum cleaners, ...



©iRobot Inc.

# AI in Reality

... soccer-playing robots, ...



©Sony

# AI in Reality

.. acrobatic humanoid robots, ...



©Boston Dynamics

... autonomous trading bots, ...



..., vocal assistants, ...



©Amazon

# AI in Reality

... image/face recognition tools, ...



# AI in Reality

... world-champion beating chess players, ...





# AI in Reality

... world-champion beating go players, ...



# AI in Reality

... AI fighter pilots, ...



... and many others ...

# Outline

- 1 AI: Fiction vs. Reality
- 2 What is AI?**
- 3 Foundations and History of AI
- 4 AI: State of the Art

# Intelligence vs. Artificial Intelligence

## Intelligence

For thousands of years, we have tried to **understand** how we think:

- how can a “handful of matter” **perceive**, **understand**, **predict**, and **manipulate** a world far larger and more complicated than itself?
- involves many disciplines, including **logic**, **psychology**, **cognitive science**, **neuroscience**, **philosophy**, **ethics**, **linguistics**, ...

## Artificial Intelligence

The field of **Artificial Intelligence (AI)** goes further still:

- it attempts not just to **understand**, but also to **build** intelligent entities
- involves all the above disciplines, but also **mathematics**, **computer science**, **engineering**, **economics**, **control theory & cybernetics**, **electronics**, ...

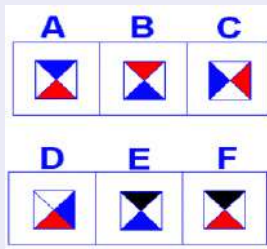
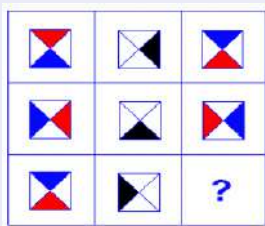
# What is Intelligence?

## Intelligence (from Wikipedia)

*“**Intelligence** has been defined in many ways: the capacity for **logic**, **understanding**, **self-awareness**, **learning**, **emotional knowledge**, **reasoning**, **planning**, **creativity**, **critical thinking**, and **problem-solving**. More generally, it can be described as the ability to **perceive** or **infer information**, and to **retain** it as **knowledge** to be **applied** towards **adaptive behaviors** within an **environment** or **context**. (...)”*

# What is Intelligence? [cont.]

## Example: simple puzzle



(Courtesy of Michela Milano, UniBO)

- What is the solution of this puzzle?  
⇒ (I'd say) **B**: result of column-by-column clockwise rotation
- What have you done for solving it?
  - 1 read & recognize figures ⇒ perceive information
  - 2 recognize patterns, problem and candidate solutions  
⇒ retain knowledge
  - 3 choose solution ⇒ infer other knowledge

# What is Artificial Intelligence?

## Different definitions due to different criteria

Historically, four approaches, along two orthogonal dimensions:

- thought processes & reasoning

vs.

behavior & action

- Success according to human standards

vs.

success according to an ideal concept of intelligence: rationality.

- human-centered approach: involves observations and hypotheses about human behavior
- rationalist approach involves a combination of mathematics and engineering.

The four groups have both disparaged and helped each other.



# What is Artificial Intelligence? [cont.]

## Thinking Humanly

“The exciting new effort to make computers think . . . *machines with minds*, in the full and literal sense.” (Haugeland, 1985)

“[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . .” (Bellman, 1978)

## Acting Humanly

“The art of creating machines that perform functions that require intelligence when performed by people.” (Kurzweil, 1990)

“The study of how to make computers do things at which, at the moment, people are better.” (Rich and Knight, 1991)

## Thinking Rationally

“The study of mental faculties through the use of computational models.”  
(Charniak and McDermott, 1985)

“The study of the computations that make it possible to perceive, reason, and act.”  
(Winston, 1992)

## Acting Rationally

“Computational Intelligence is the study of the design of intelligent agents.” (Poole *et al.*, 1998)

“AI . . . is concerned with intelligent behavior in artifacts.” (Nilsson, 1998)

# Thinking Humanly: The cognitive modeling approach

Problem: How do humans think?

- Idea: develop a **theory of the mind**
    - ⇒ express the theory as computer programs
      - e.g. Newell & Simon's **General Problem Solver** (1961)
  - Requires scientific theories of brain activities (**cognitive model**)
  - Inter-disciplinary field: **Cognitive Science**
    - combines computer models from AI and experimental techniques from psychology
    - construct precise and testable theories of the human mind
  - AI and Cognitive Science nowadays distinct
    - A.I: find an algorithm performing well on a task
    - C.S: find a good model of human performance
- although they fertilize each other (e.g. in computer vision)

# Acting Humanly: The Turing Test Approach

Problem: When does a system behave intelligently?

## The Turing Test

- **Alan Turing** “Computing Machinery and Intelligence” (1950)
- Operational test of intelligence (aka “**The Imitation game**”):
  - A human, a computer, an interrogator in a different room.
  - The interrogator should classify the human and the machine.
  - Can the computer mislead the interrogator and be classified as a human?
- “behave intelligently”  $\iff$  “behave humanly”



# Acting Humanly: The Turing Test Approach

Problem: When does a system behave intelligently?

## The Turing Test

- **Alan Turing** “Computing Machinery and Intelligence” (1950)
- Operational test of intelligence (aka “**The Imitation game**”):
  - A human, a computer, an interrogator in a different room.
  - The interrogator should classify the human and the machine.
  - Can the computer mislead the interrogator and be classified as a human?
- “behave intelligently”  $\iff$  “behave humanly”

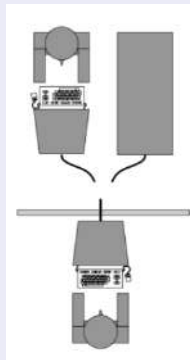


# Acting Humanly: The Turing Test Approach

Problem: When does a system behave intelligently?

## The Turing Test

- **Alan Turing** “Computing Machinery and Intelligence” (1950)
- Operational test of intelligence (aka “**The Imitation game**”):
  - A human, a computer, an interrogator in a different room.
  - The interrogator should classify the human and the machine.
  - Can the computer mislead the interrogator and be classified as a human?
- “behave intelligently”  $\iff$  “behave humanly”



# Acting Humanly: The Turing Test Approach [cont.]

## Capabilities for passing the Turing Test

- **natural language processing** to enable it to communicate successfully in English (or other)
- **knowledge representation** to store what it knows or hears
- **automated reasoning** to answer questions and to draw new conclusions
- **machine learning** to adapt to new circumstances and to detect and extrapolate patterns

For **Total Turing test** (with physical interaction wrt. interrogator):

- **computer vision** to perceive objects
- **computer speech** to communicate orally
- **robotics** to manipulate objects and move about

- These disciplines compose most of AI
- Turing Test is still relevant in AI (although not fundamental)

# Acting Humanly: The Turing Test Approach [cont.]

## Some successes with Turing test

- (2014) a chatbot by Eugene Goostman, mimicking the answer of a 13 years old boy, has succeeded the test.
  - chatbots are now frequently available
- vocal assistants are now of common use
  - e.g. Alexa (Amazon), Siri (Apple), Cortana (Microsoft), ...

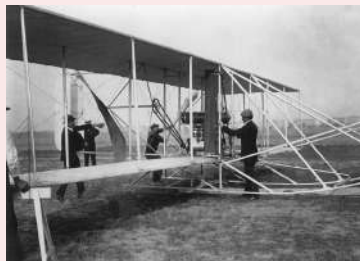
## Limitations of Turing Test

- not reproducible, constructive or amenable to mathematical analysis
- AI researchers devoted little effort to make systems pass the Turing Test
- [ Do humans always pass the Turing test? (See e.g. [here](#)) ]
- **Should we really emulate humans to achieve intelligence?**
- Shouldn't we study the underlying principles of intelligence instead?

# Acting Humanly: The Turing Test Approach [cont.]

## Metaphorical Example

Successful flight machines have not been developed by imitating birds, rather by studying engines and aerodynamics.



(see e.g. [this video](#)).



# Thinking Rationally: The “Laws of Thought” Approach

Problem: Can we capture the laws of thought?

- Aristotle: What are **correct** argument and thought processes?
  - codify “right thinking” i.e. irrefutable reasoning processes (**syllogisms**): (e.g. “all men are mortal; Socrates is a man; therefore, Socrates is mortal”)  
⇒ **Logic** and **Logical inference**
- The **Logician tradition** in AI hopes to create intelligent systems using logic-based inference systems
  - “**algorithm = logic + control**”
  - logic programming, automated-deduction systems, ...
  - logics: propositional, first-order, modal & description, temporal, ...
- Two main limitations:
  - not easy to state informal knowledge into the formal terms of logic
  - problems undecidable or computationally very hard (NP-hard)
- **Logical reasoning** is currently part of many fields of AI
  - **problem solving, knowledge representation & reasoning, planning,**
  - does not exhaustively cover AI

# Acting Rationally: The Rational-Agent Approach

Problem: Can we make systems “do the right thing”?

## Rational Agents

- An **agent** is an entity that **perceives** and **acts**
  - persists over a prolonged time period
- A **rational agent** acts so as
  - **to achieve the best outcome** (maximize goal achievement), or
  - **to achieve the best expected outcome** (under uncertainty)
- Rational agents need all skills needed for the Turing Test!
- Thinking rationally is **sometimes part of** being a rational agent
  - e.g. planning an action
  - sometimes action without thinking (e.g. reflexes)
- Two advantages over previous approaches:
  - **More general than law of thoughts approach** (correct inference is just one of several possible mechanisms for achieving rationality)
  - **More amenable to scientific development than human-emulation approaches** (rationality mathematically well defined & general)

# Acting Rationally: The Rational-Agent Approach [c.]

This course concentrates on **general principles of rational agents** and on the **components for constructing them**. (Following AIMA book.)

## Remark

- achieving **perfect rationality** is not feasible in complex environments
  - computational demands too high
  - however, good working hypothesis and starting point for analysis

⇒ dealing with **limited rationality**

- acting appropriately when not enough time to do all computations

# AI Systems Classification

## Weak vs. Strong AI

- **Weak AI**: Is it possible to build systems that **act as if they were intelligent**?
- **Strong AI**: Is it possible to build systems that **are intelligent**? (i.e., that have conscious minds, wills and sentiments?)

## General AI vs. Narrow AI

- **General AI** refers to systems able to cope with any generalized task which is asked of it, much like a human.
- **Narrow AI** refers to systems able to handle one particular task.  
⇒ AI system displays a certain degree of intelligence only in a particular narrow field to perform highly specialized tasks

# AI Systems Classification [cont.]

## Symbolic Approach vs. Connectionist Approach

- **Top-down, or Symbolic Approach:**
  - Symbolic representation of knowledge
  - Logics, ontologies, rule based systems, declarative architecture
  - Human-understandable models
- **Bottom up, or Connectionist Approach:**
  - Based on Neural networks.
  - Knowledge is not symbolic and it is “encoded” into connections between neurons.
  - Concepts are learned by examples
  - Non understandable by humans

# Outline

- 1 AI: Fiction vs. Reality
- 2 What is AI?
- 3 Foundations and History of AI**
- 4 AI: State of the Art

# The Foundations of Artificial Intelligence

Different fields have contributed to AI in the form of **ideas**, **viewpoints** and **techniques**

- **Philosophy**: Logic, reasoning, mind as a physical system, foundations of learning, language and rationality
- **Mathematics**: Formal representation and proof, computation, (un)decidability, (in)tractability, probability
- **Economics**: formal theory of rational decisions, game theory
- **Neuroscience**: physical substrate for mental activities
- **Psychology**: adaptation, phenomena of perception and motor control
- **Computer Science & Engineering**: algorithms, data structures, efficient implementations
- **Control Theory & Cybernetics**: homeostatic systems, stability, optimal agent design
- **Linguistics**: knowledge representation, grammar

# Brief History of Artificial Intelligence

## The Gestation of AI (1943-1955)

- 1943: **Warren Mc Culloch** and **Walter Pitts**: a model of artificial Boolean neurons to perform computations
  - First steps toward connectionist computation and learning
  - **Marvin Minsky** and **Dann Edmonds** (1951) constructed the first neural network computer
- 1950: **Alan Turing**: “Computing Machinery and Intelligence”
  - Turing Test
  - First complete vision of AI



# Brief History of Artificial Intelligence [cont.]

## The Birth of AI (1956) and Era of Great Expectations

- **Darmouth Workshop** (1956) brought together top minds on automata theory, neural nets and the study of intelligence
  - **Allen Newell** and **Herbert Simon**: The **Logic Theorist**
    - first **nonnumerical thinking program** used for theorem proving
    - proved theorems from Russell & Whitehead Principia Mathematica
- **The era of great expectations** (1952-1969)
  - **Newell** and **Simon** introduced the **General Problem Solver (GPS)**
    - could handle a (limited) number of logical puzzles
    - imitation of human problem-solving: strategy to address subgoals
    - Idea: any system (human or machine) exhibiting intelligence must operate by **manipulating data structures composed of symbols**
  - **John McCarthy**
    - Invented **LISP** (and time-sharing)
    - Logic-oriented **Advice Taker**, decoupling knowledge and reasoning
  - **Marvin Minsky**
    - addressed **microworlds**, problems in limited domain that appear to require intelligence to solve (e.g. blocks-world, geometric problems)
  - **S. Winograd** and **J.D. Cowan**, et al.: early work on **neural networks**

# Brief History of Artificial Intelligence [cont.]

## Collapse in AI research (1966 - 1973)

- Progress was slower than expected.
  - enthusiast predictions turned unrealistic
- Some systems lacked scalability
  - computational intractability due to combinatorial explosion in search
- Fundamental limitations on techniques and representations
  - [Minsky](#) & [Papert](#) (1969): important limitations to neural networks

# Brief History of Artificial Intelligence [cont.]

## AI Revival via knowledge-based systems (1969-1970)

- **General-purpose**  $\implies$  **domain specific systems**
  - narrow domains, exploiting domain-specific knowledge
  - E.g. **DENDRAL**: successful in inferring molecular structure from information by mass-spectrometer (**Buchanan** et al. 1969)
- **Expert systems** applied to areas of human expertise
  - e.g., **MYCIN**: diagnose blood infections (**Feigenbaum** et al.)
  - based on 450 domain-specific rules from experts & textbooks
  - a calculus for **uncertainty**
- Several progresses in Natural language processing
  - incorporate domain knowledge in NLP

## AI becomes an industry (1980-present)

- commercial expert system **R1** at DEC (**McDermott**, 1982)
  - helped configure orders for computer system (saves: 40M\$/year)
- followed a period of national and industry investments in AI
- followed a period of expert systems industry busts ("**AI Winter**")

# Brief History of Artificial Intelligence [cont.]

## The return of neural networks (1986-present)

- (re)invented the **back-propagation** learning algorithm
  - applied to many learning problems in computer science and psychology

⇒ revival of **connectionist models** for intelligent systems  
(vs. symbolic or logicist approaches)

# Brief History of Artificial Intelligence [cont.]

## AI adopts the scientific method (1987-present)

- A “gentle revolution” in AI content and methodology
  - build on existing theories than to propose brand-new ones
  - base claims on rigorous theorems or hard experimental evidence rather than on intuition
  - show relevance to real-world applications rather than toy example
- AI has finally come firmly under the scientific method
  - hypotheses must be subjected to rigorous empirical experiment
  - results must be analyzed statistically for their importance

⇒ general increase in technical depth
- Resurgence of **probability**, focus on **uncertainty**
  - (speech & handwriting recognition): **hidden Markov models**
  - neural networks benefited from statistics, pattern recognition, and machine learning ⇒ **data mining**
  - rigorous reasoning with uncertainty: **Baynesian networks**
  - Similar “gentle revolutions” occurred in **robotics**, **computer vision**, and **knowledge representation**.

# Brief History of Artificial Intelligence [cont.]

## The emergence of intelligent agents (1995-present)

- renewed interest in the “whole agent” problem:  
“How does an agent act/behave embedded in real environments with continuous sensory inputs?”
- Es: AI in the internet domain “-bots”
  - Decision support systems, robotic agents, natural language
- Need for interaction between sensing and reasoning  
⇒ reasoning and planning systems must handle uncertainty
- AI forced into much closer contact with other fields
  - e.g. [control theory](#), [economics](#)

# Brief History of Artificial Intelligence [cont.]

## The availability of very large data sets (2001-present)

**Big data** and **massive computing power** (e.g. GPUs) have enabled deep networks to be properly trained and to work properly

- Until recently: emphasis on **algorithms**
- Recent works in AI: emphasis on **data**  
(for **machine learning** & **deep learning**)

⇒ **learning methods** rather than **hand-coded knowledge engineering** used to express the knowledge a system needs

- ⇒ **Large amount and variety of AI applications**  
(**speech and image recognition**, **spam filtering**, **robotics**, **machine translation**, **autonomous vehicles**, **game playing**, ...)
- many AI applications are now deeply embedded in the infrastructure of every industry

# Brief History of Artificial Intelligence [cont.]

## The Deep-Learning Tsunami (2015-present)

- “Deep Learning waves have lapped at the shores of computational linguistics for several years now, but 2015 seems like the year when the full force of the tsunami hit the major Natural Language Processing (NLP) conferences.” [C. Manning]
- Previous successes in the fields of image classification and speech...
- Experts in the field (LeCun, Hinton, Bengio) agree on the fact that there will be important developments in text and video understanding, machine translation, question answering ... [Turing award]
- Google masters GO: Deep-learning software defeats human professional for the first time. AlphaGo. Nature 529, 445-446 (28 January 2016). In March 2016, Lee Sedol defeated.



# Main AI Research Venues

- Major AI Journals

- Artificial Intelligence
- Computational Intelligence
- Journal of Artificial Intelligence Research
- IEEE Transactions on Pattern Analysis and Machine Intelligence
- IEEE Intelligent Systems
- [ area-specific journals ]

- Main AI Conferences

- International Joint Conference on AI (IJCAI)
- National Conference on AI (AAAI)
- European Conference on AI (ECAI)
- [ area-specific conferences ]

- Main professional societies for AI

- American Association for Artificial Intelligence (AAAI)
- ACM Special Interest Group in Artificial Intelligence (SIGART)
- Society for Artificial Intelligence and Simulation of Behaviour (AISB)

# Outline

- 1 AI: Fiction vs. Reality
- 2 What is AI?
- 3 Foundations and History of AI
- 4 AI: State of the Art**

# AI is everywhere ...

- Search engines
- Route planning (e.g. maps, traffic)
- Logistics (e.g. packages, inventory, airlines)
- Medical diagnosis, machine diagnosis
- Automated help desks
- Spam/fraud detection
- Smarter devices, e.g. cameras
- Product recommendations
- Assistants, smart homes
- ... Lots more!

# What can AI Systems Currently Do?

... classify incoming e-mails as spam (or not), ...



<http://www.resilientystems.co.uk/>

# What can AI Systems Currently Do?

... predict stock price evolution, ...

Apple Inc. (NASDAQ:AAPL)

Add to portfolio

More results

**105.67** -0.46 (-0.43%)

Mar 24 Close  
NASDAQ real-time data - Disclaimer  
Currency in USD

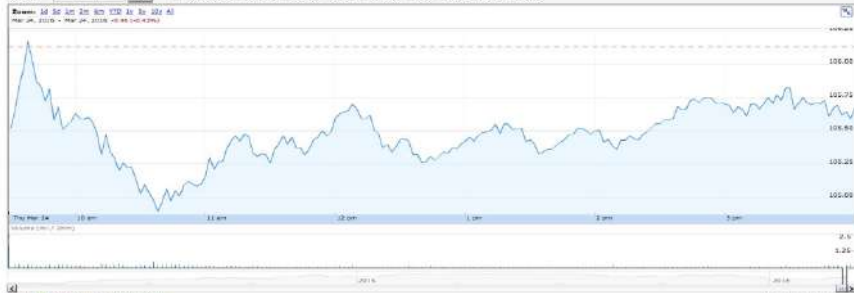
Range 104.89 - 106.25  
52 week 92.00 - 134.54  
Open 105.47  
Vol / Avg 26.13M/25.90M  
Mkt cap 569.36B  
P/E 11.22  
Div/yield 0.52/1.97  
EPS 9.41  
Shares 5.54B  
Beta 0.97  
Inst. own 59%

Q+1 9.36

Dow Jones 17,515.73 0.58%  
Nasdaq 4,773.50 0.10%  
Technology 4,773.50 0.18%  
AAPL 105.67 -0.43%



Compare: ☐ S&P 500 ☐ Nasdaq ☐ SPY ☐ Dow Jones ☐ Nasdaq ☐ SNBK ☐ MSFT ☐ SSKNF ☐ VZ ☐ HPQ ☐ IBM ☐ HTCKF



Gettings | Technicals | [Link to this view](#)

Volume selected for 15 mins.  
Prices are not from all markets.  
Source: Reuters

# What can AI Systems Currently Do?

... understanding handwriting, ...

80322-4129 80206

40004 14310

37872 05453

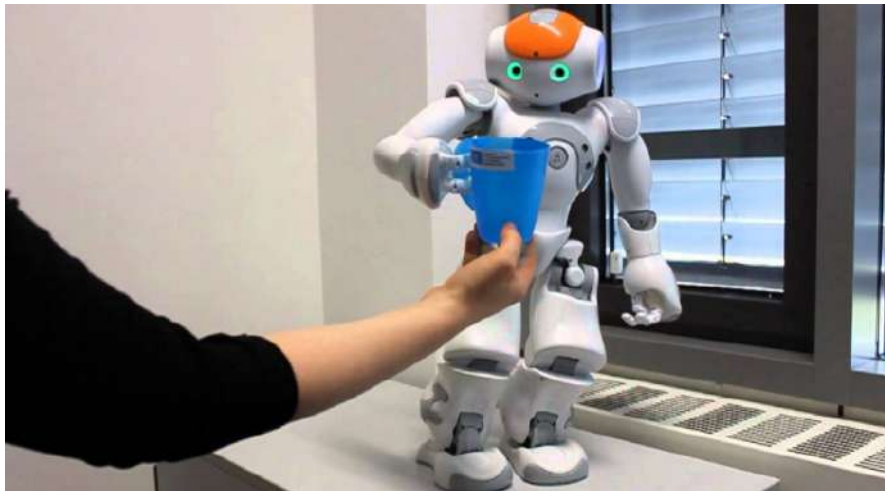
~~33~~02 75216

35460 44209

[LeCun et al. 1989]

# What can AI Systems Currently Do?

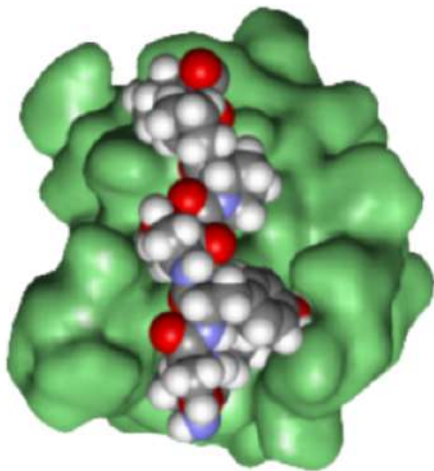
... learn to grab a cup, ...



<http://www.informatik.uni-bremen.de/>

# What can AI Systems Currently Do?

... design a molecule with given properties, ...



<http://pande.stanford.edu/>



# What can AI Systems Currently Do?

... translate text from Chinese to English, ...



©Google Inc.

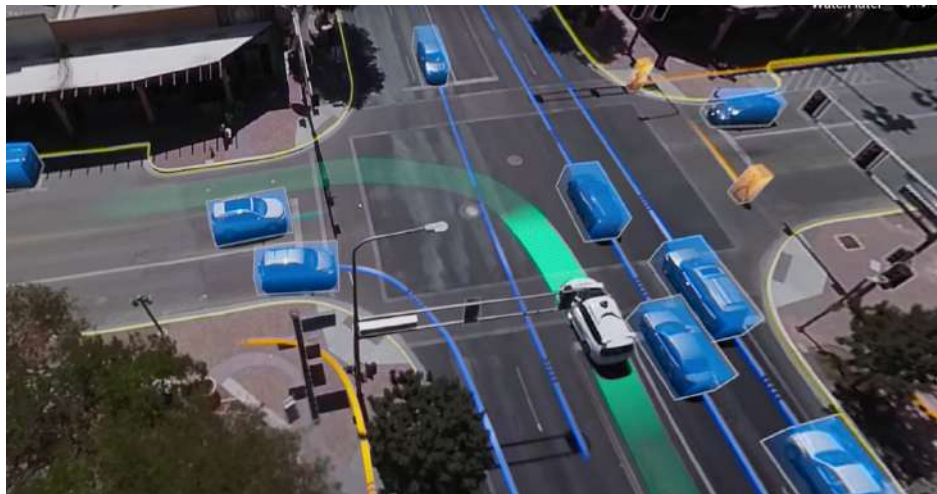
# What can AI Systems Currently Do?

... convert a voice into text, ...



# What can AI Systems Currently Do?

... predict traffic trajectories, ...



# What can AI Systems Currently Do?

... automatically writing the caption of a figure, ...



"man in black shirt is playing guitar."



"construction worker in orange safety vest is working on road."



"two young girls are playing with lego toy."



"boy is doing backflip on wakeboard."



"girl in pink dress is jumping in air."



"black and white dog jumps over bar."



"young girl in pink shirt is swinging on swing."



"man in blue wetsuit is surfing on wave."

[Karpathy & Fei-Fei, 2015; Donahue et al., 2015; Xu et al, 2015;...]

# What can AI Systems Currently Do?

... driving autonomously, ...



©Google Inc.

# What can AI Systems Currently Do?

... run & jump on two legs, ...



© Boston Dynamics

# What can AI Systems Currently Do?

... beat a top-gun pilot in a simulated F16 dogfight, ...



## Quiz: What can AI Systems Currently Do?

- Play a decent game of Jeopardy? YES
- Win against any human at chess? YES
- Win against the best humans at Go? YES
- Play a decent game of tennis? YES
- Grab a particular cup and put it on a shelf? YES
- Unload any dishwasher in any home? NO
- Drive safely along the highway? YES
- Drive safely in Naples' center on rush hour? NO
- Buy groceries on the web? YES
- Buy groceries at next corner shop? NO
- Discover and prove a new mathematical theorem? NO
- Perform a surgical operation? NO
- Translate spoken Chinese into spoken English in real time? YES
- Write an intentionally funny story? NO