MACHINE INTELLIGENCE

Introduction to AI and ML

K.S.Srinivas
Department of Computer Science and Engineering

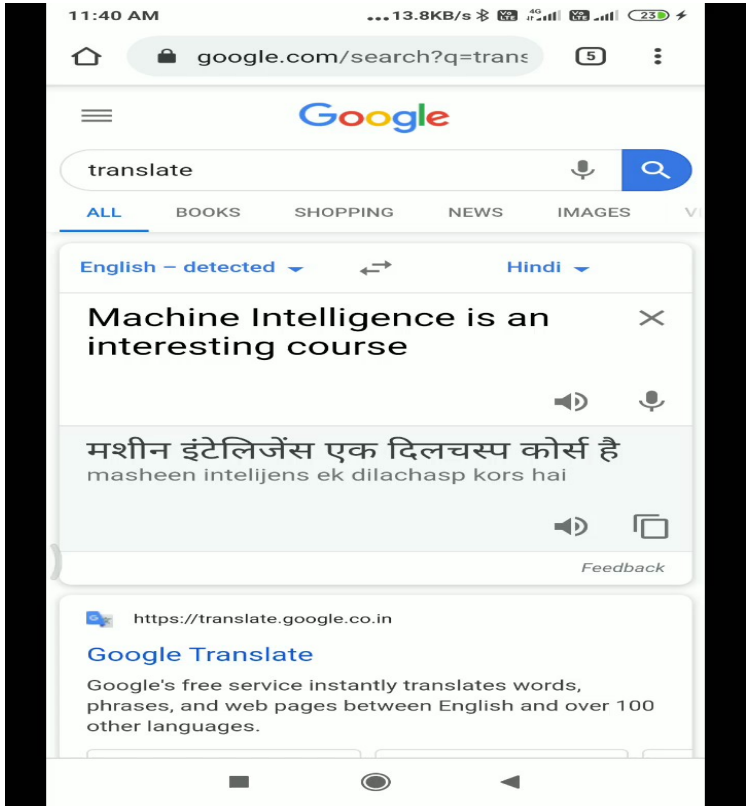
MACHINE INTELLIGENCE

Introduction to AI and ML

K.S.Srinivas
Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Machine Intelligence is Omni Present



MACHINE INTELLIGENCE

Introduction

A news item went as follows: **'Apple buys** machine learning firm **Perceptio** Inc., a startup, in an attempt to bring advanced image-classifying artificial intelligence to smartphones by reducing data **overhead** which is typically required of **conventiona**l methods'.

Source: https://appleinsider.com/articles/15/10/05/apple-buys-machine-learning-firm-perceptio-smartphone-ai

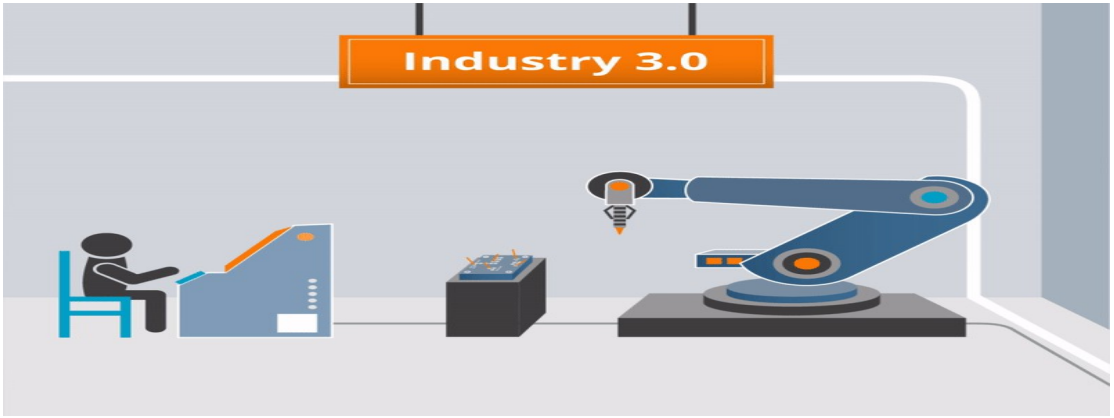
Sundar Pichai, the CEO of software giant Google, on being asked ***what is the next thing*** at the company, said "I can't quite tell exactly but advances in **AI and machine learning**, ***we are making a big bet*** on that. Advances in machine learning will bring a difference in many many fields." while interacting with students at his alma mater IIT-Kharagpur.

Source:http://imorphosis.com/category/artificial-intelligence/

MACHINE INTELLIGENCE

Machine Intelligence a working definition

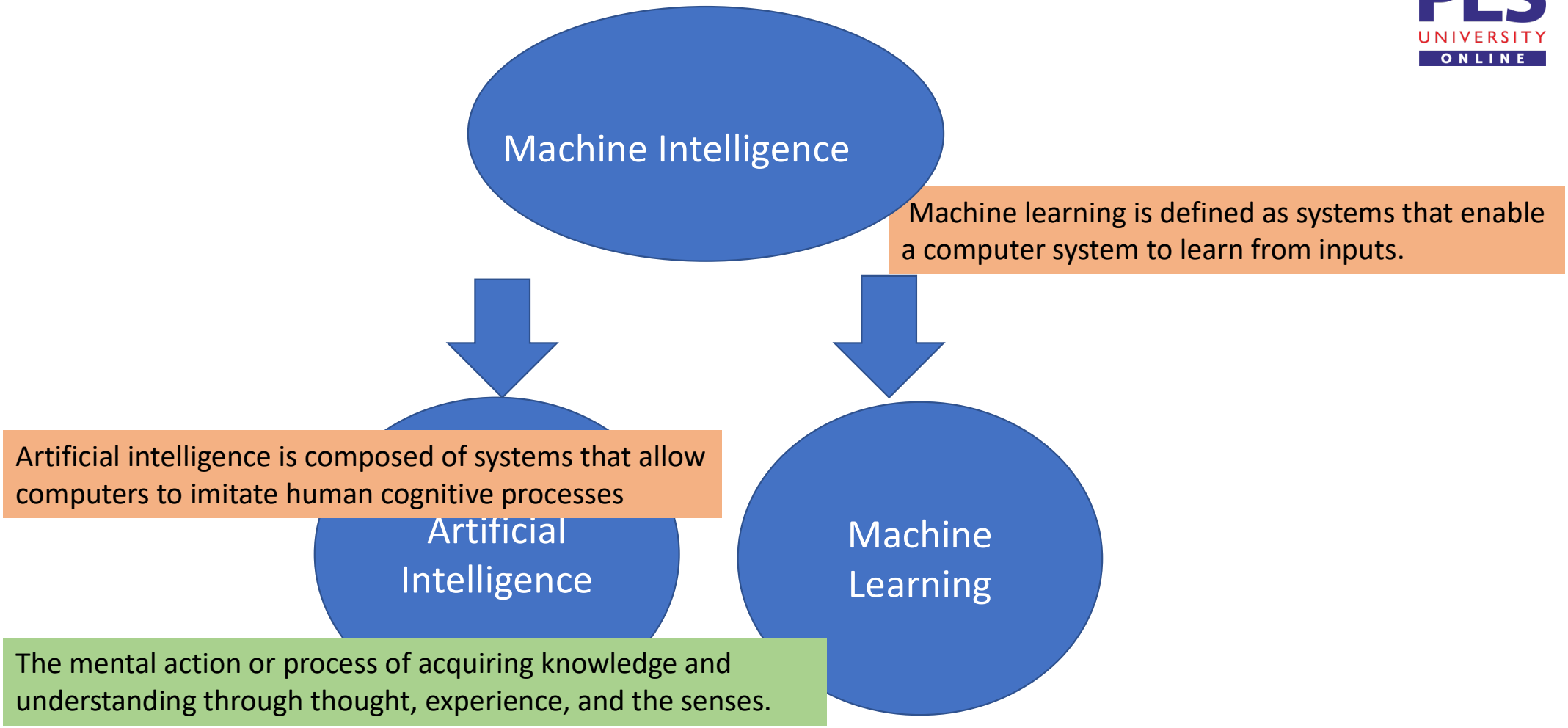
- Machine intelligence “enables a machine to interact with an environment in an intelligent way.”
- Its good to look at this term from the perspective the two other terms that are proliferating today
  - Artificial Intelligence
  - Machine Learning



Source: https://humans-machines-progress.com/reportage/work-4-0-humans-at-its-heart/

MACHINE INTELLIGENCE

Definitions of Artificial Intelligence and Machine Learning



MACHINE INTELLIGENCE

Intelligence – A computer Science Perspective

Intelligence is broadly broken into 3 parts

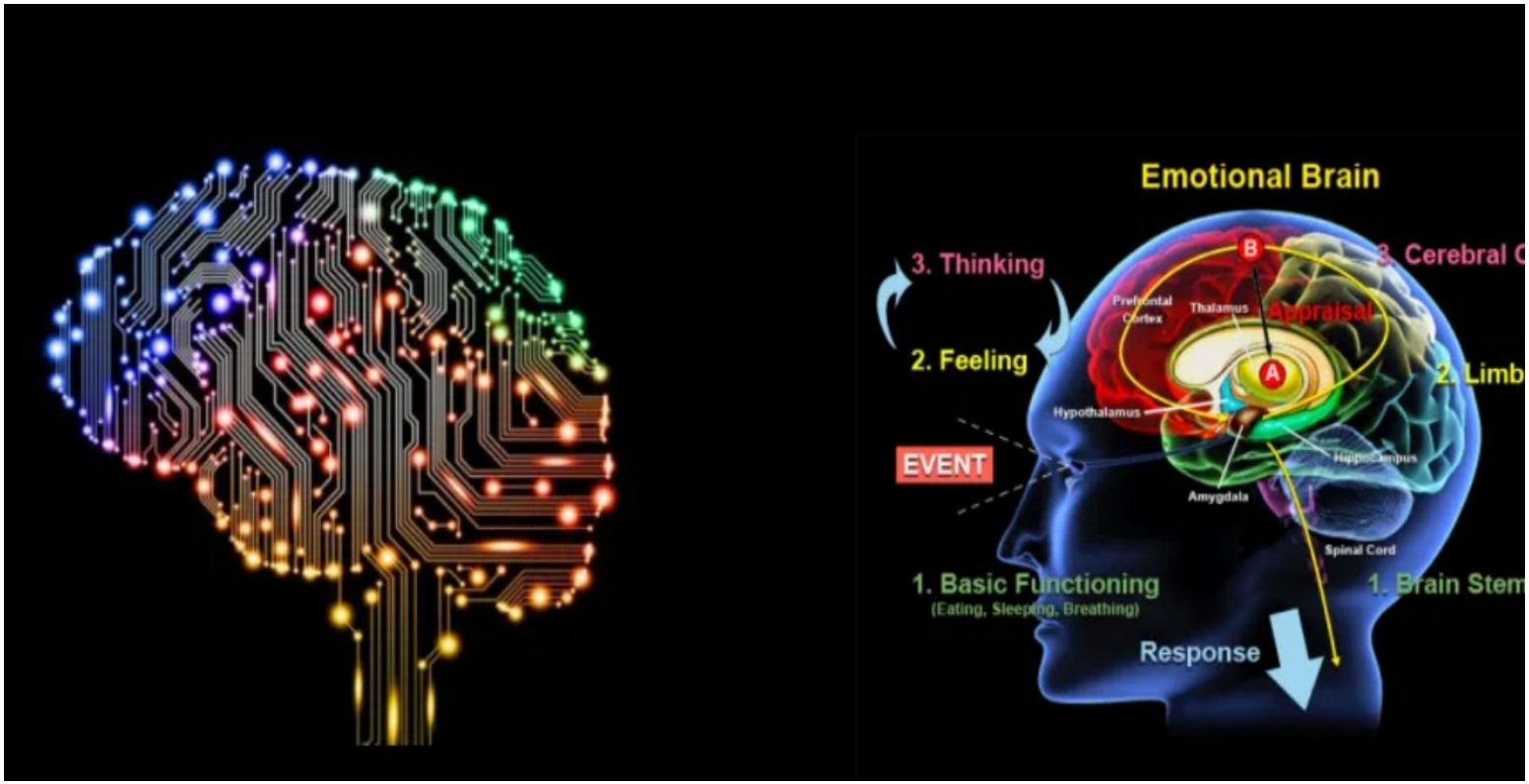
1. Reasoning or Considering – Thinking
2. Seeing, Hearing or Being Understood – Perception
3. Taking Action

We would therefore define Artificial Intelligence as building models that perceive , think and act on the thoughts processed

My definition of AI is any algorithm that is new in computer science. Once the algorithm becomes accepted then it's not AI, it's just a boring algorithm.

MACHINE INTELLIGENCE

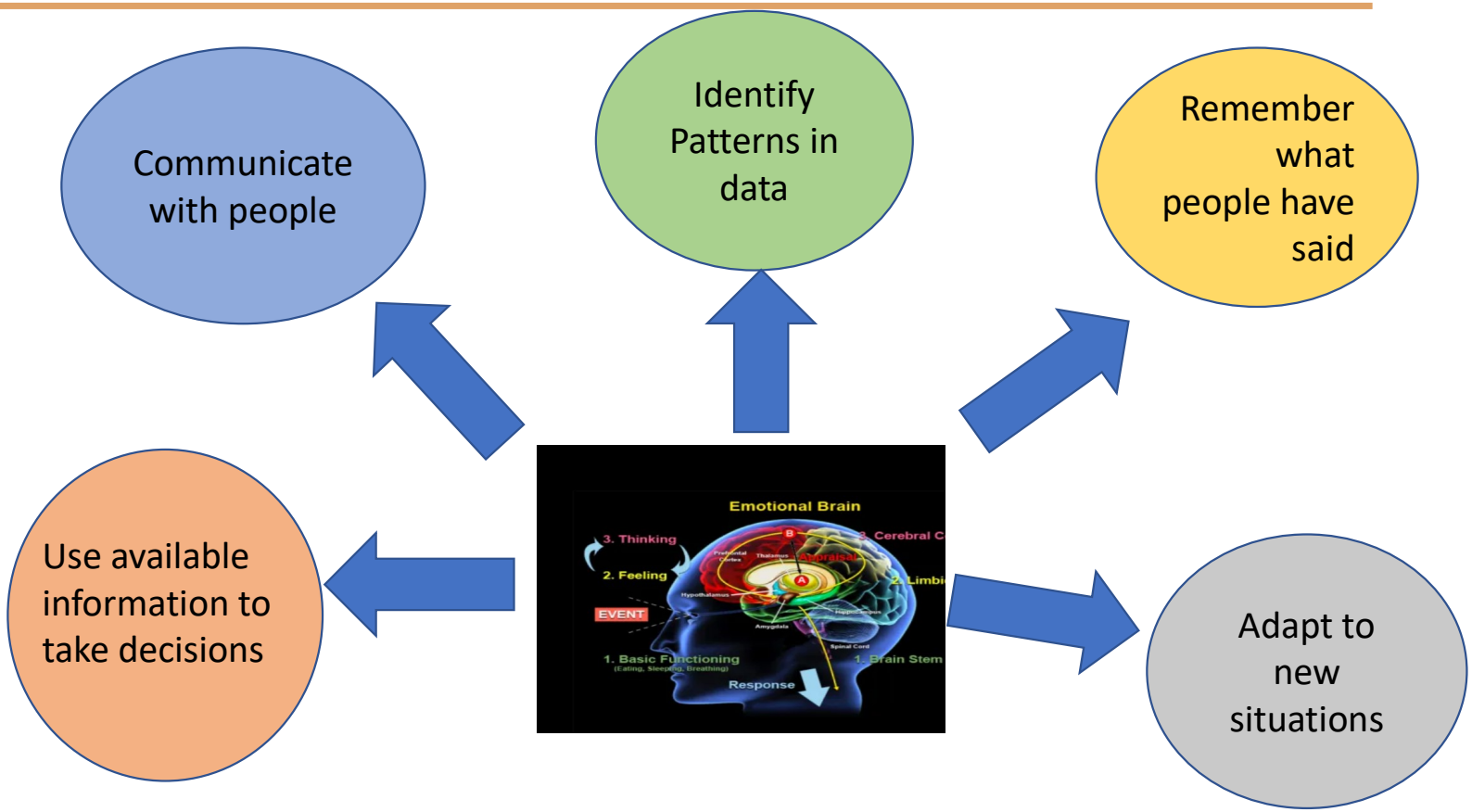
Artificial Intelligence Vs. Human Intelligence



Source:https://techswizard.com/gadget/artificial-intelligence-vs-human-intelligence/

MACHINE INTELLIGENCE

Human Intelligence



MACHINE INTELLIGENCE

Levels of Artificial Intelligence

AI has three different levels:

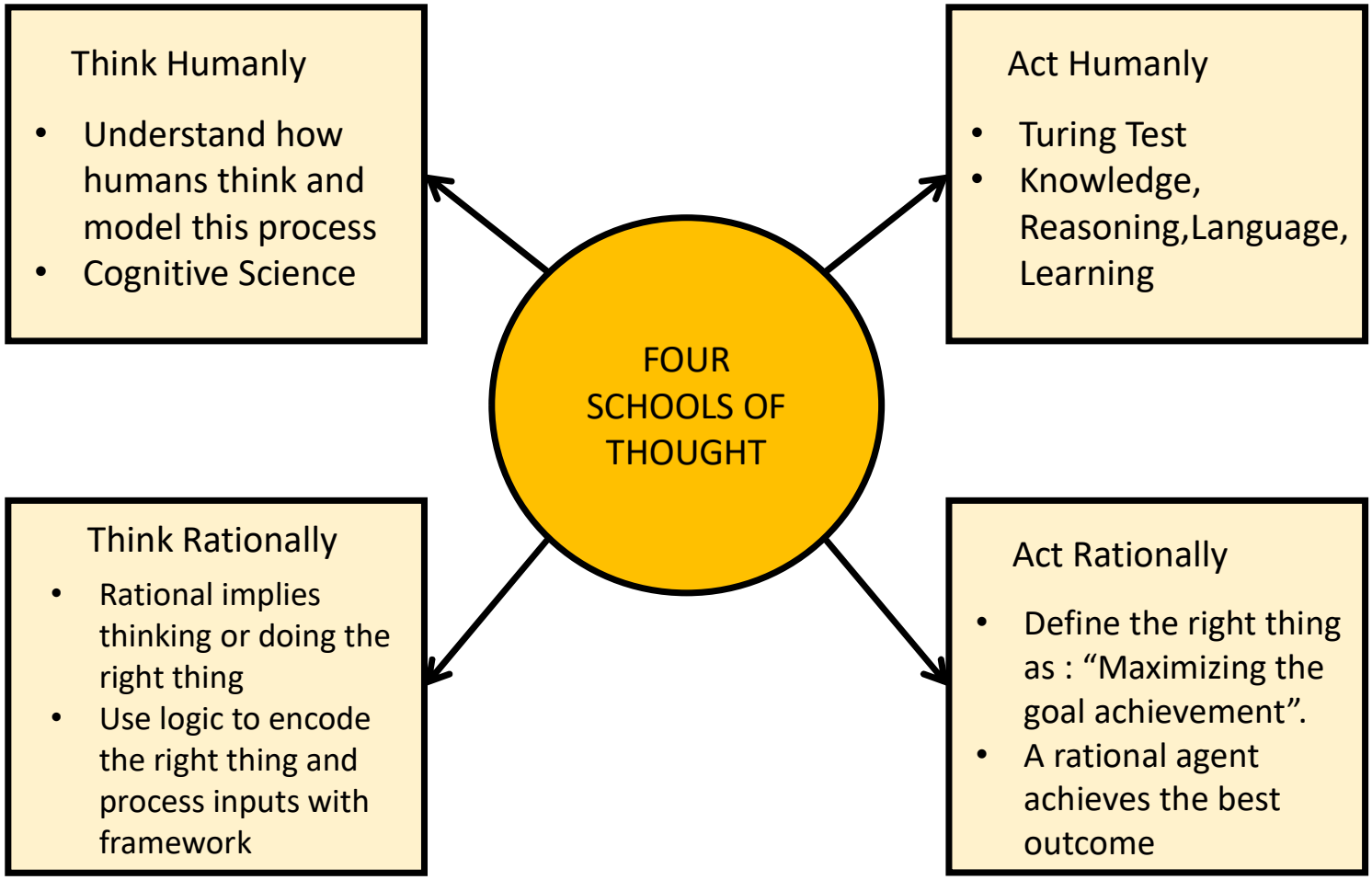
Narrow AI: A artificial intelligence is said to be narrow when the machine can perform a specific task better than a human. The current research of AI is here now

General AI: An artificial intelligence reaches the general state when it can perform any intellectual task with the same accuracy level as a human would

Active AI: An AI is active when it can beat humans in many tasks

MACHINE INTELLIGENCE

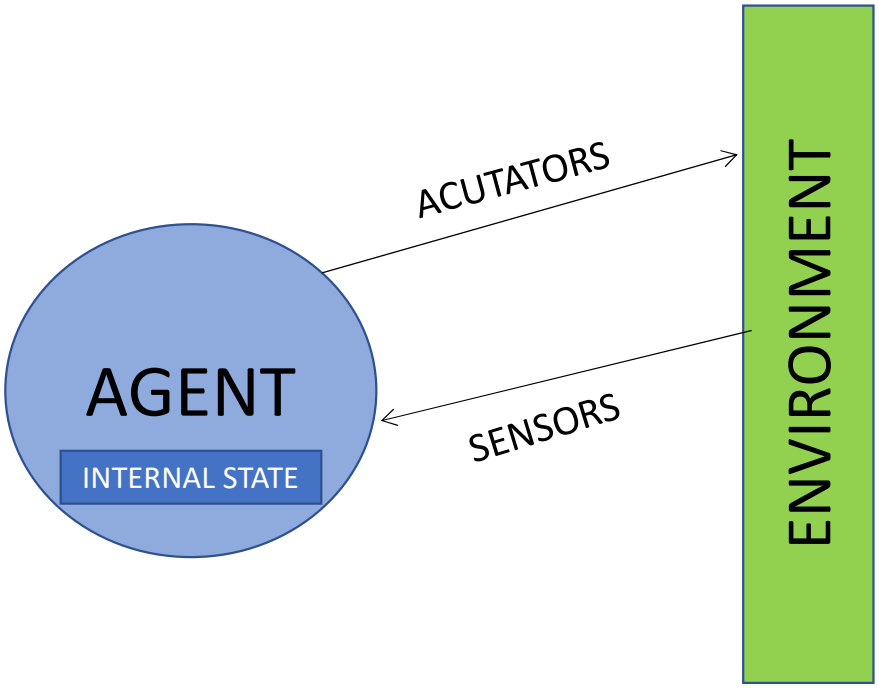
Four Schools of Thought on Artificial Intelligence



MACHINE INTELLIGENCE

Agent – A definition

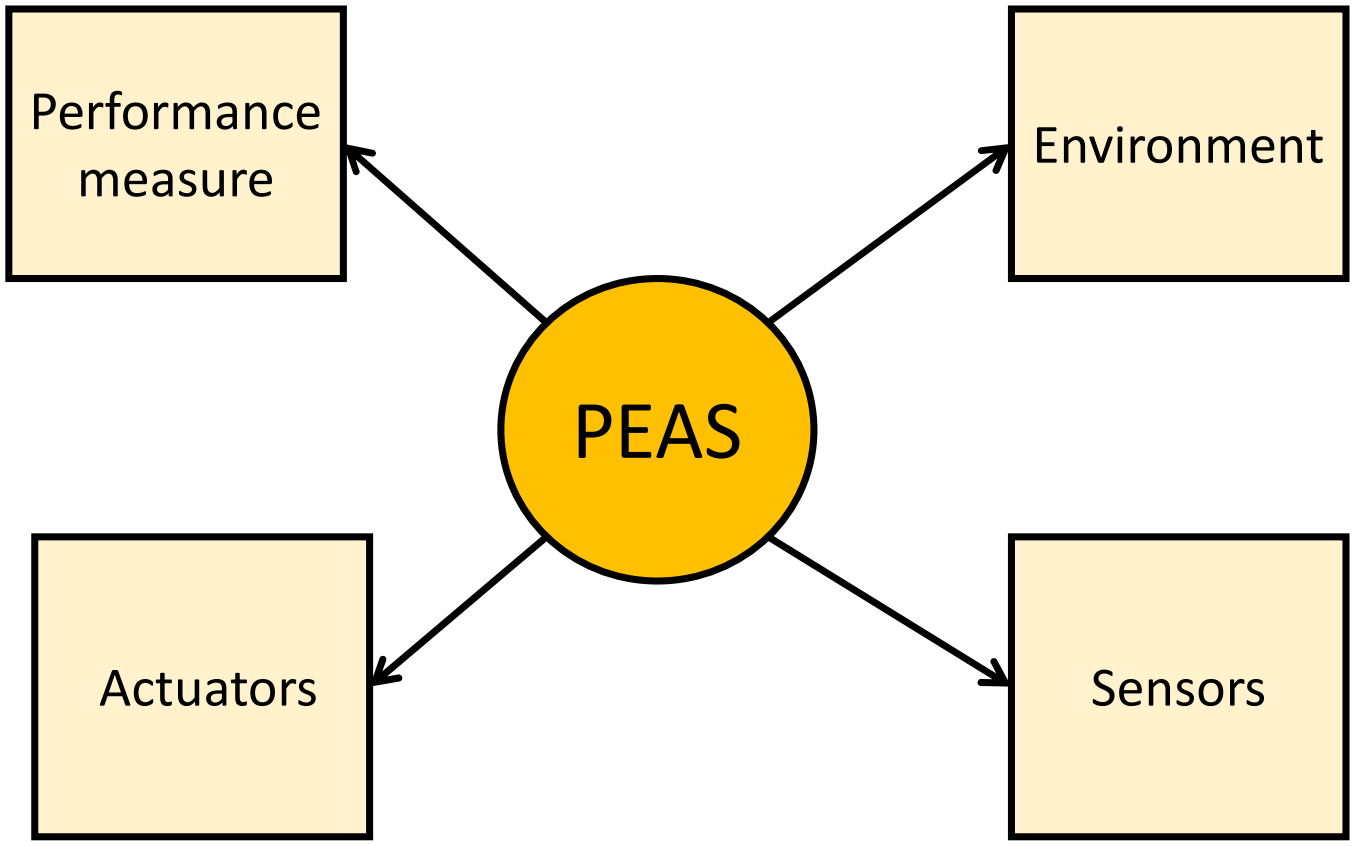
- Agent perceives the environment and acts upon the environment in order to maximize achievement of the required goal.(Actions)
- We will discuss more about agents in the upcoming sessions.



MACHINE INTELLIGENCE

Grouping of Intelligent Agent

In order to group similar time of agents we follow a specific grouping system called PEAS



MACHINE INTELLIGENCE

What would be the PEAS for this example

What Architecture would you choose
What would be your algorithm for preventing this accident

How would you recognize a kerb from a drunk lying on the road?

Performance measure: Safe, fast, legal, comfortable trip, maximize profits
Environment: Roads, other traffic, pedestrians, customers
Actuators: Steering wheel, accelerator, brake, signal, horn
Sensors: Cameras, sonar, speedometer, GPS, odometer, engine sensors, keyboard

MACHINE INTELLIGENCE

Turing Test

Act Humanly

- Turing Test
- Knowledge, Reasoning,Language, Learning

MACHINE INTELLIGENCE

Modelling AI problems

Different types of problems may require different types of approaches

- some problems can be easily represented using state spaces
  ex: Robot navigation through the maze
- Problems that can be solved using Machine Learning techniques
  ex: Face Recognition
- Probabilistic Graphical Models such as Bayes Network,HMMs
  ex: Speech Recognition
- Problem that can be well addressed using deductive logic ,like given a certain proposition and input ,perform logical interface
  ex: imagine a chat bot that encodes some knowledge and can reason with the user

Modelling example

- Problem-

  Suppose you have to reach place A from place B with a route that leads you fast as possible .You are provided with map and info about traffic along routes

- Model-

  Represent the landmarks as nodes of graph,.Edges represent the connection between the landmarks.Edges are annotated with time cost of moving from one landmark to next .

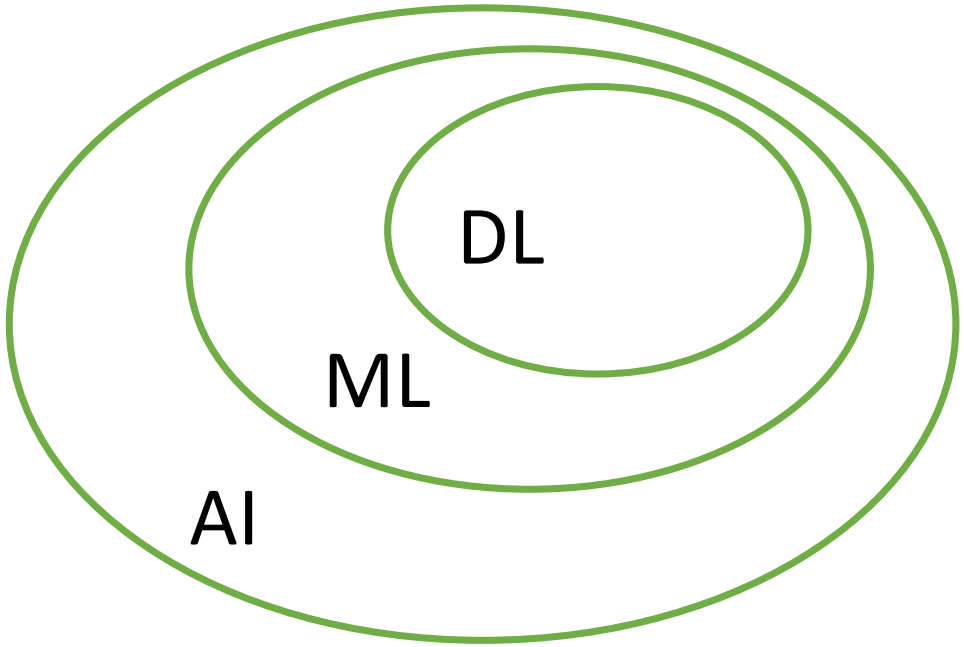
- Algorithm-

  Graph Search algorithm such as BFS,DFS, Uniform Cost Search etc

MACHINE INTELLIGENCE

Artificial Intelligence >>> Machine Learning

- The goal of AI is to build human-like intelligence on machines
- ML is a core approach to achieve this goal
- DL is a suite of techniques that form a sub set of a broad suit of ML techniques



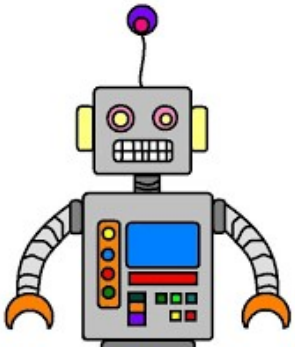
MACHINE INTELLIGENCE

Machine Learning

- Consider the world ,we have humans and we have computers
- Can we get computers to learn from experience too???
- YES -and that is precisely what machine learning means
- but for computers we have a different term for experience that is data

Learn from experience

Learn from ~~experience~~ data

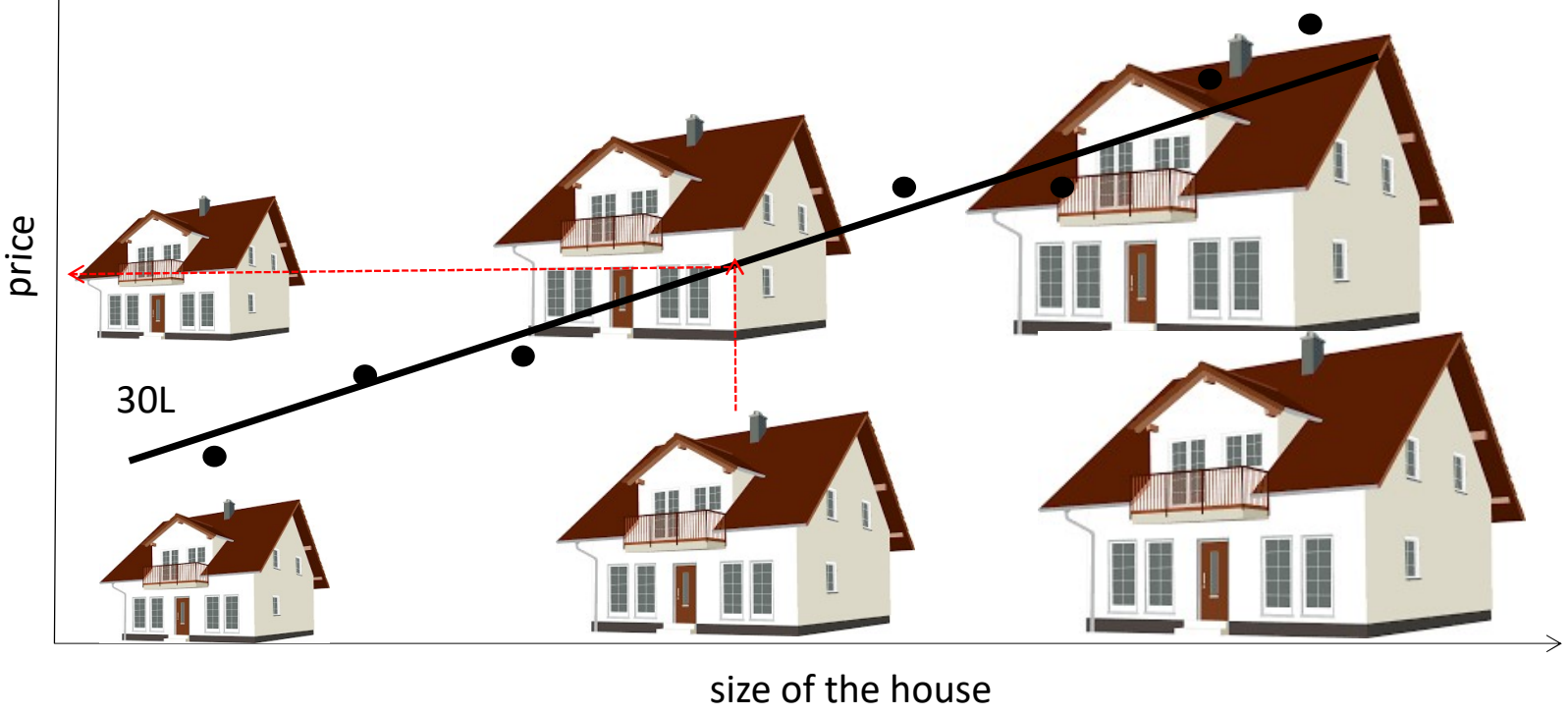


Follow instructions

MACHINE INTELLIGENCE

Learning from Data

- Let us see one example to understand how a machine learns from experience(data)
- consider we have two house with following price and we need to predict the price of the medium sized house
- we will plot them on a graph with some other data ,find a best fit line to predict its price
- this method is called linear regression ,how to find the best fit line ? we will see it in further session.

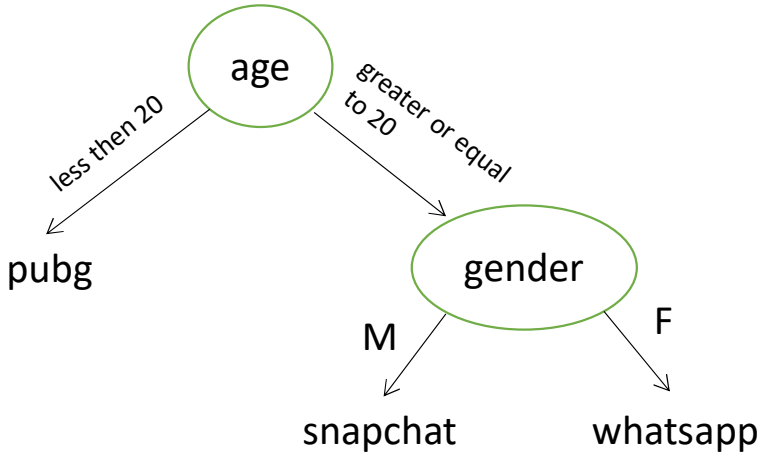


MACHINE INTELLIGENCE

Learning from Data

- we are on a task to built a app recommendation system with some previous data
- what do think can be criteria that influences the recommendation more , gender or age
- There is not much split in gender
- If we use the age split we see people below age 20 downloaded pubg and other downloaded whatsapp and snapchat
- we can decide the following algorithm
- This is known as decision tree learning and we will study this in detail in upcoming sections

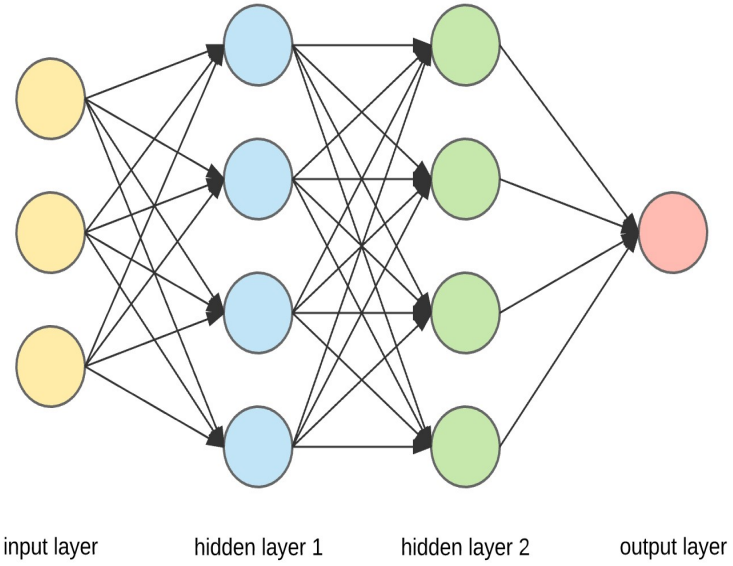
Gender	Age	App
F	15	pubg
F	25	whatsapp
M	32	snapchat
F	40	whatsapp
M	12	pubg
M	14	pubg

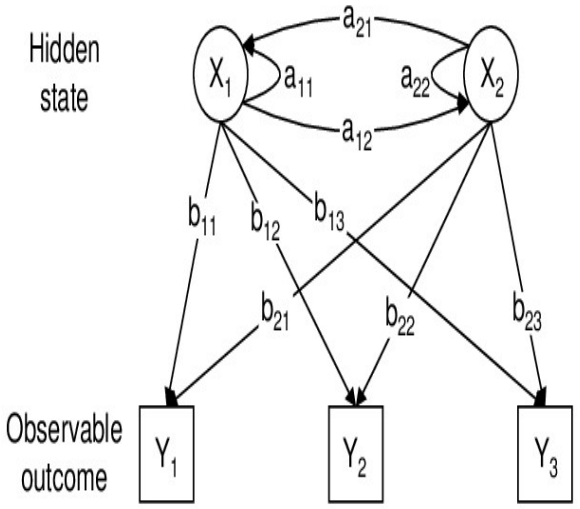


MACHINE INTELLIGENCE

Machine Intelligence is Omni Present

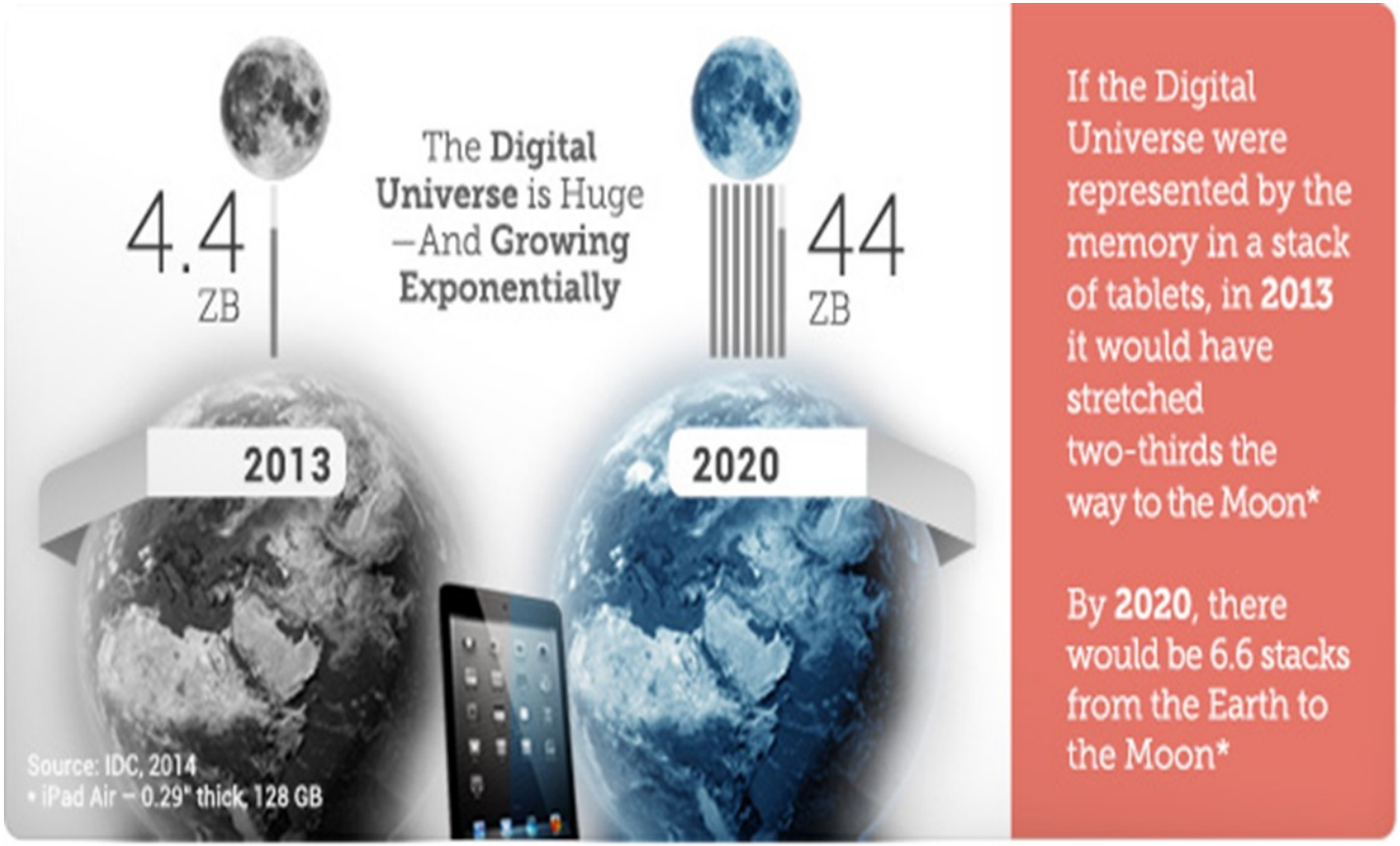
We will be analyzing other various kind of algorithms throughout this course to solve real world problems





MACHINE INTELLIGENCE

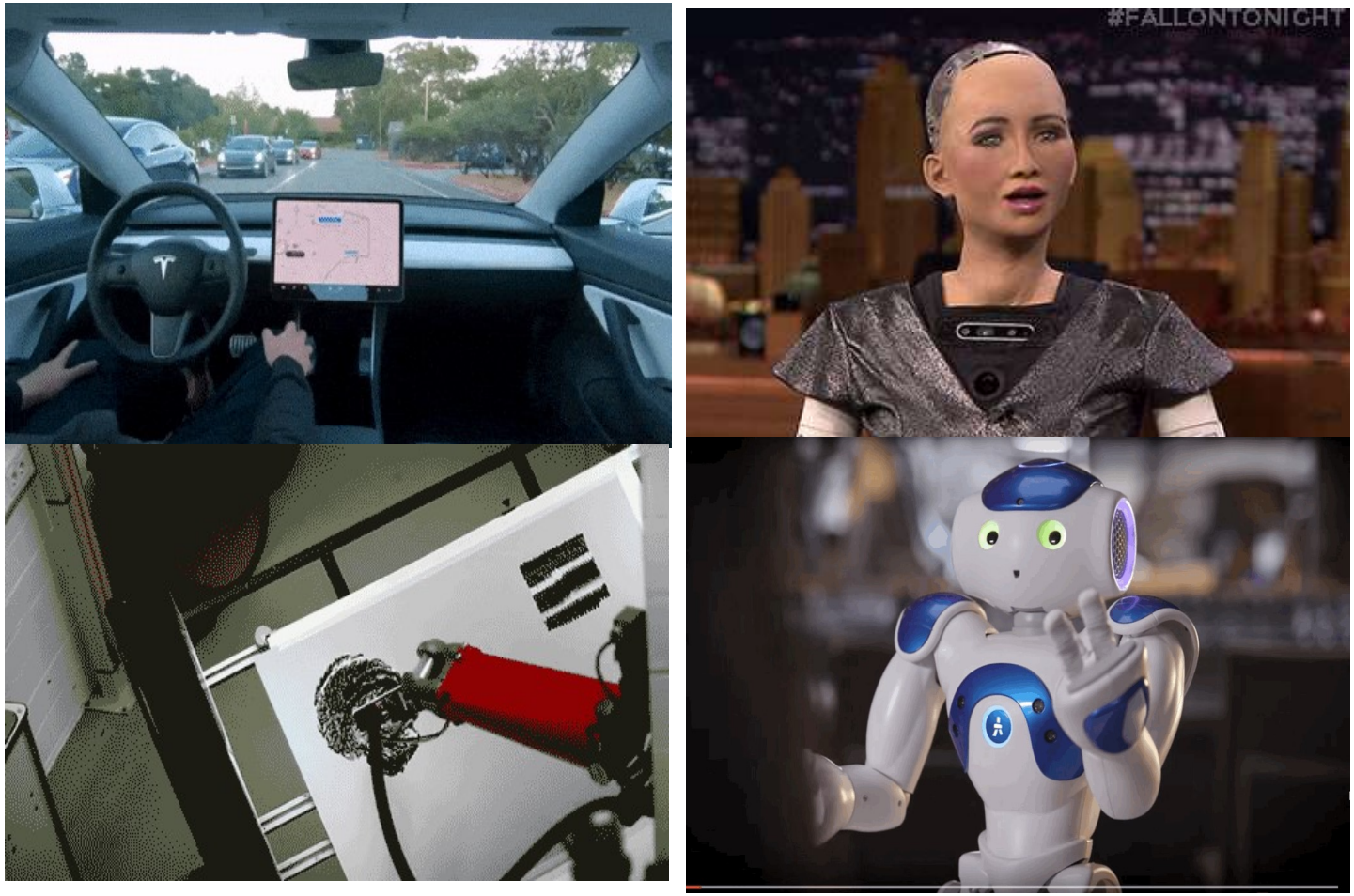
The new dawn of Machine Intelligence



https://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm

MACHINE INTELLIGENCE

Examples of Machine Intelligence



Source: https://giphy.com/search/artificial-intelligence

MACHINE INTELLIGENCE

Issues with Machine Intelligence

- What algorithms can approximate functions well and when ?
- How much training data is sufficient to learn a concept with high confidence?
- When is it useful to use prior knowledge?
- Are some training examples more useful than others?
- What are best tasks for a system to learn?
- What is the best way for a system to represent its knowledge?
- Can the learner automatically alter its representation for improvement ?

At the end of this course you will be able to answer all these questions

THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701

MACHINE INTELLIGENCE

AGENTS AND ITS TYPES

K.S.Srinivas
Department of Computer Science and Engineering

MACHINE INTELLIGENCE

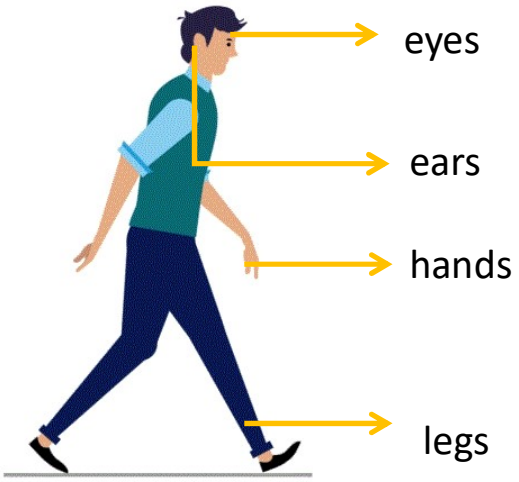
AGENTS AND ITS TYPES

Srinivas K S.
Associate Professor, Department of Computer Science

MACHINE INTELLIGENCE

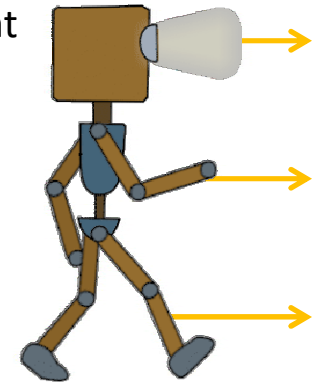
Agent- classic definition

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators



HUMAN AGENT

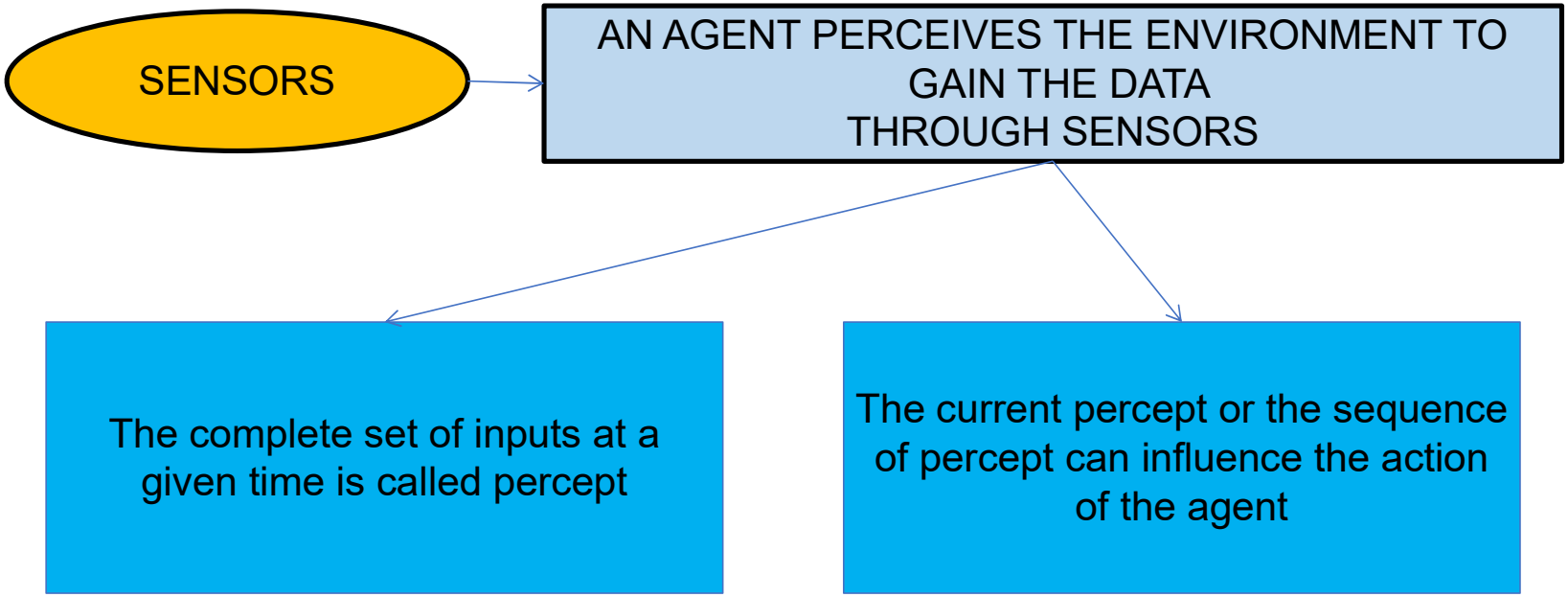
perceives the environment and starts the action



ROBOTIC AGENT

MACHINE INTELLIGENCE

Sensors



MACHINE INTELLIGENCE

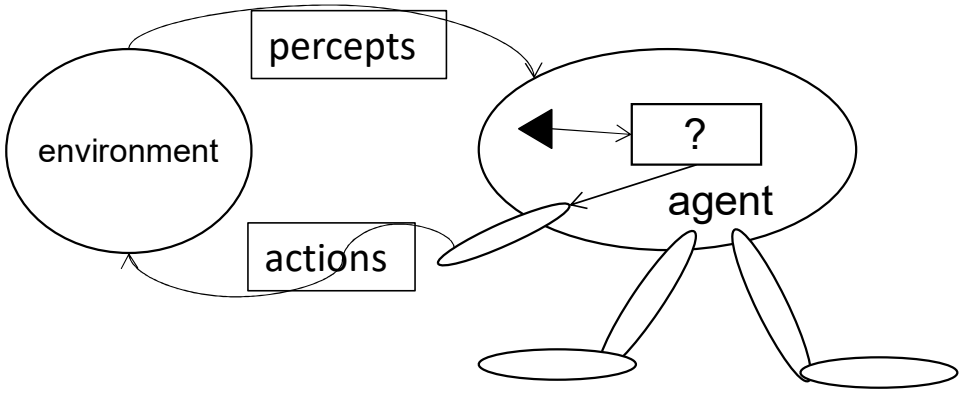
Actuator

ACTUATOR

AN AGENT CAN CHANGE THE STATE OF THE ENVIRONMENT THROUGH ACTUATOR

An operation involving an actuator is called an action

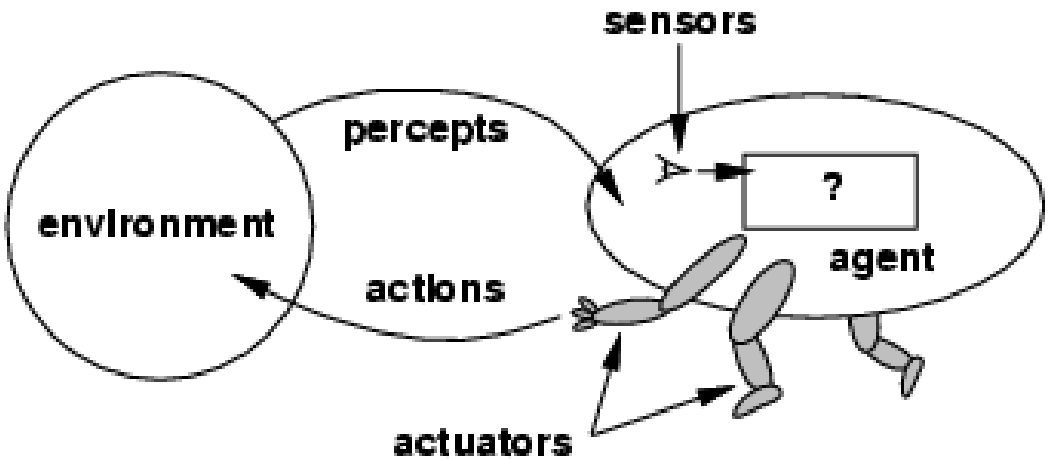
so an agent program implement mapping from percept sequences to actions



- first the agent gets input at a given time called percept
- the agent program maps the percept to an action
- the actuators performs that action to change the state of the environment

MACHINE INTELLIGENCE

Agent -Environment Interaction



- Basically the agent maps from percept history to actions [f: P* → A]
- The agent program runs on the physical architecture to produce f
- agent = architecture + program

MACHINE INTELLIGENCE

Reactive Agent

Each behaviour continually maps perceptual input to action output

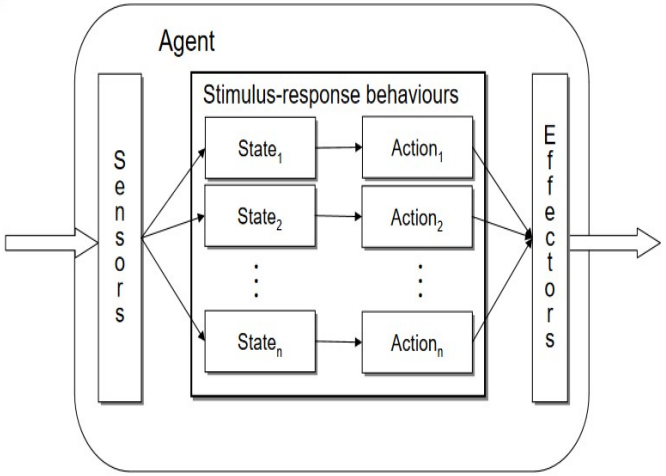
- Reactive behaviour:

  action: S -> A

  - where S denotes the states of the environment, and A the primitive actions the agent is capable of perform.

- Example:

$$\text{action}(s) = \begin{cases} \textit{Heater on}, \text{ if temperature too low} \\ \textit{Heater off}, \text{ otherwise} \end{cases}$$



MACHINE INTELLIGENCE

AI == building Agents???

- AI is about building rational agents
- An agent is something that perceives and acts
- A rational agent always does the right thing.
- An agent should strive to "do the right thing", based on what it can perceive and the actions it can perform. The right action is the one that will cause the agent to be most successful
- E.g., performance measure of a vacuum-cleaner agent could be amount of dirt cleaned up, amount of time taken, amount of electricity consumed, amount of noise generated, etc. – These could be success parameters

MACHINE INTELLIGENCE

Duties of a Rational Agent

For each possible percept sequence, a rational agent should

select an action that is expected to maximize its performance measure

given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

MACHINE INTELLIGENCE

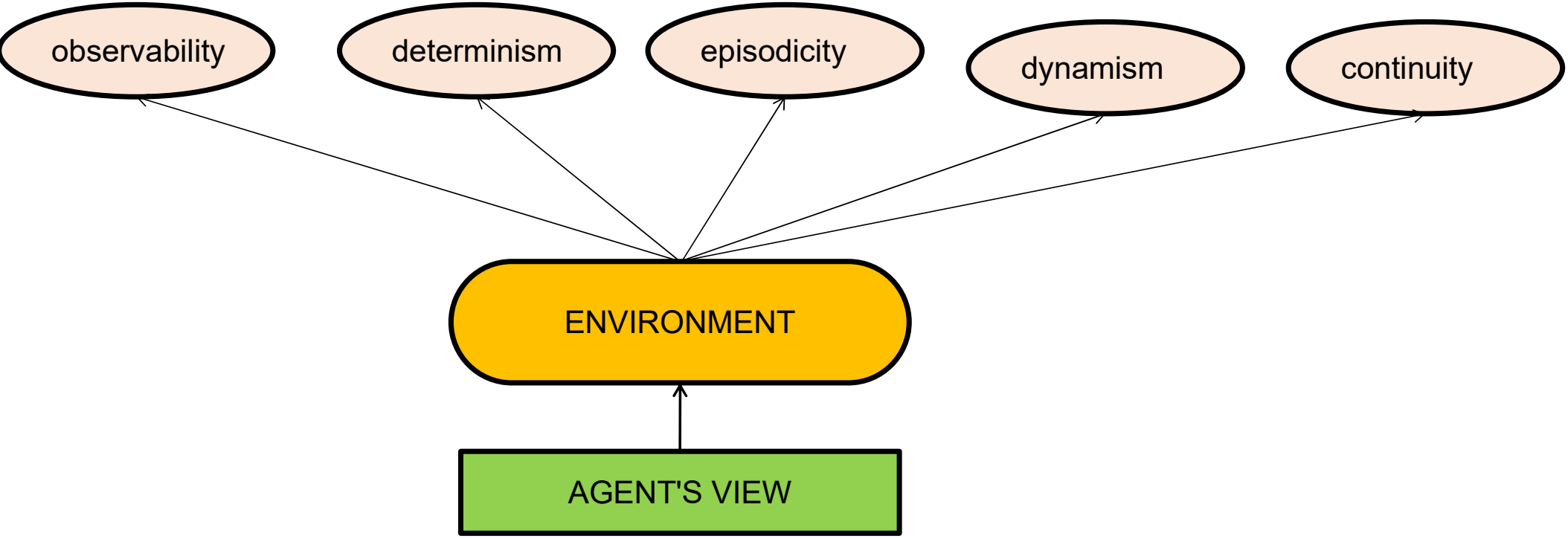
More about Agents

- Rationality is distinct from omniscience (all-knowing with infinite knowledge) – All Knowing
- Agents can perform actions in order to modify future percepts so as to obtain useful information (information gathering, exploration) - Probing
- An agent is autonomous if its behavior is determined by its own experience (with ability to learn and adapt) - Autonomous

MACHINE INTELLIGENCE

Environment from an agents view

- Environment in which agents operate can be defined in different ways
- It is helpful to view the following definitions as referring to the way the environment appears from the point of view of the agent itself.
- Environment can be classified on the following grounds from the view of an agent



MACHINE INTELLIGENCE

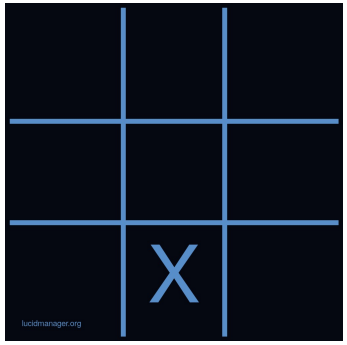
Observability

- **FULLY OBSERVABLE:-**

All of the environment relevant to the action being considered is observable.such environment are comfortable since the agent is freed from the task of keeping track of changes in the environment.

- **PARTIALLY OBSERVABLE:-**

The relevant features of the environment are only partially observable.



MACHINE INTELLIGENCE

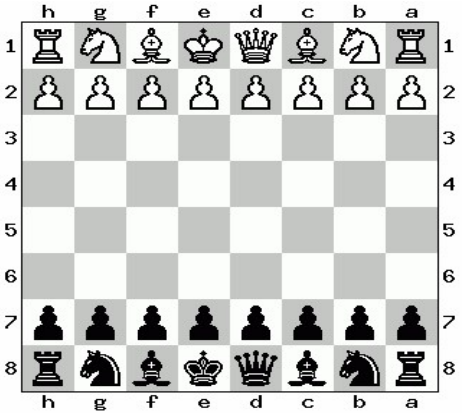
Determinism

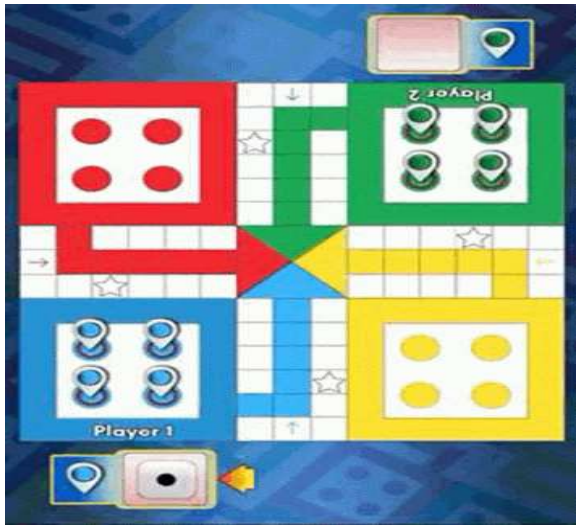
- **DETERMINISTIC:-**

The next state of the environment is completely described by the current state and the agents action

- **STOCHASTIC:-**

If an element of interface or uncertainty occurs then the environment is stochastic. A partially observable environment will appear to be stochastic to the agent





MACHINE INTELLIGENCE

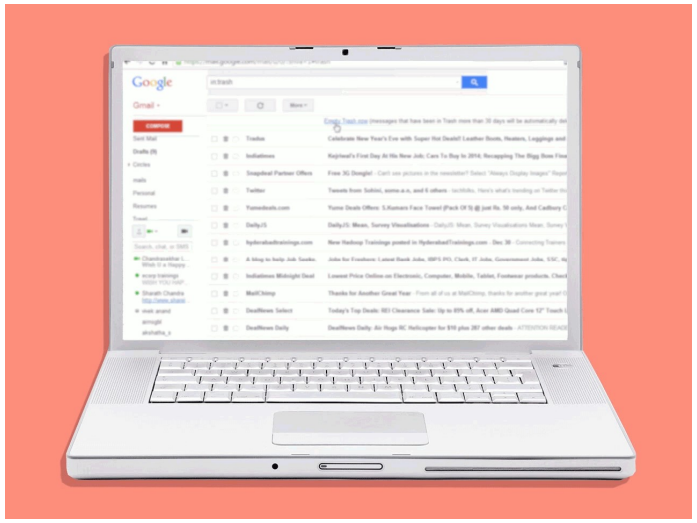
Episodicity

- **EPISODIC:-**

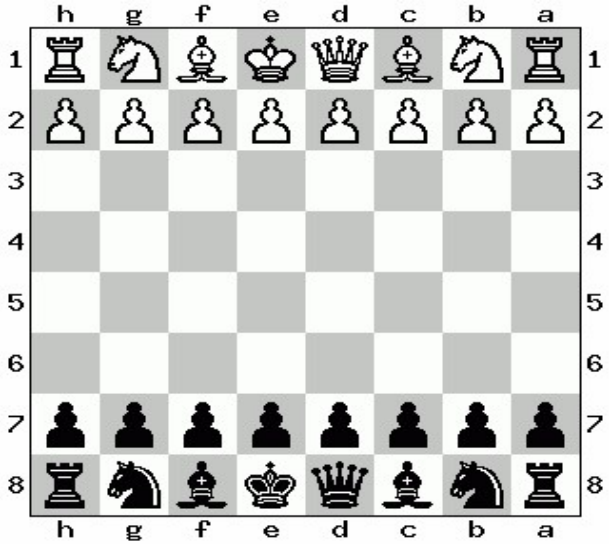
An episodic environment means that subsequent episodes do not depend on what actions occurred in previous episodes

- **SEQUENTIAL:-**

The agent engages in a series of connected episodes.



mail sorting system



MACHINE INTELLIGENCE

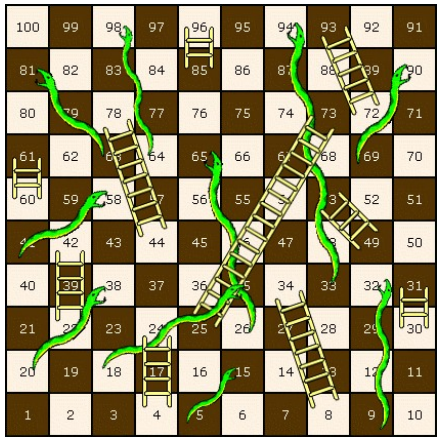
Dynamism

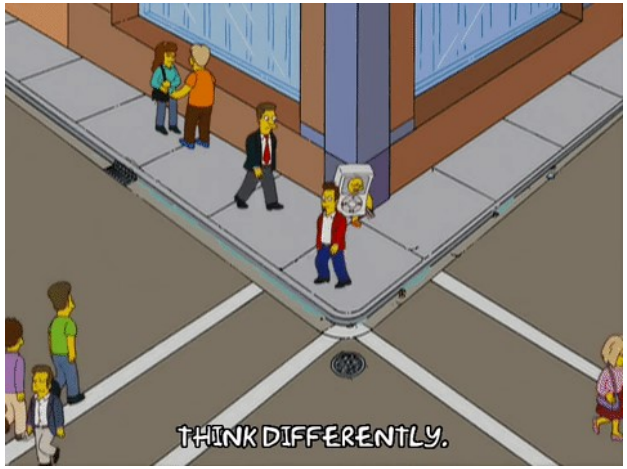
- **STATIC ENVIRONMENT:-**

Does not change from one state to next while the agent is considering its course of action.The only changes to the environment as those caused by the agent itself.

- **DYNAMIC ENVIRONMENT:-**

Changes over time independent of the actions of the agent-and thus if an agent does not respond in a timely manner,this counts as a choice to do nothing.



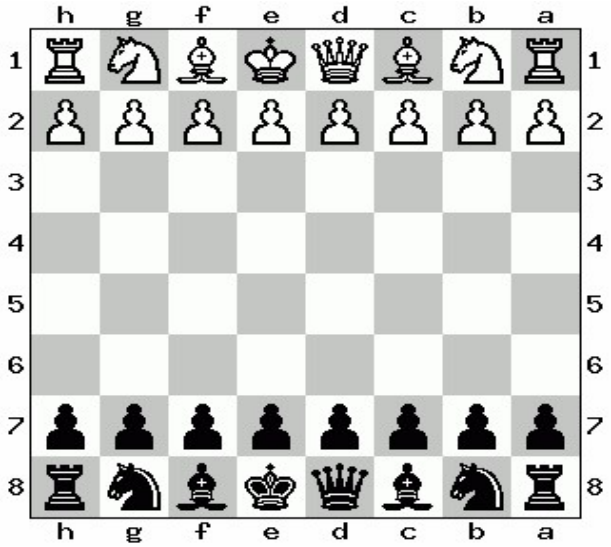


MACHINE INTELLIGENCE

Continuity

- **DISCRETE/CONTINUOUS ENVIRONMENT:-**

If the number of distinct percepts and actions is limited,the environment is discrete,other wise continuous



MACHINE INTELLIGENCE

Analysis Time

- After these discussion lets try to analyse few environment

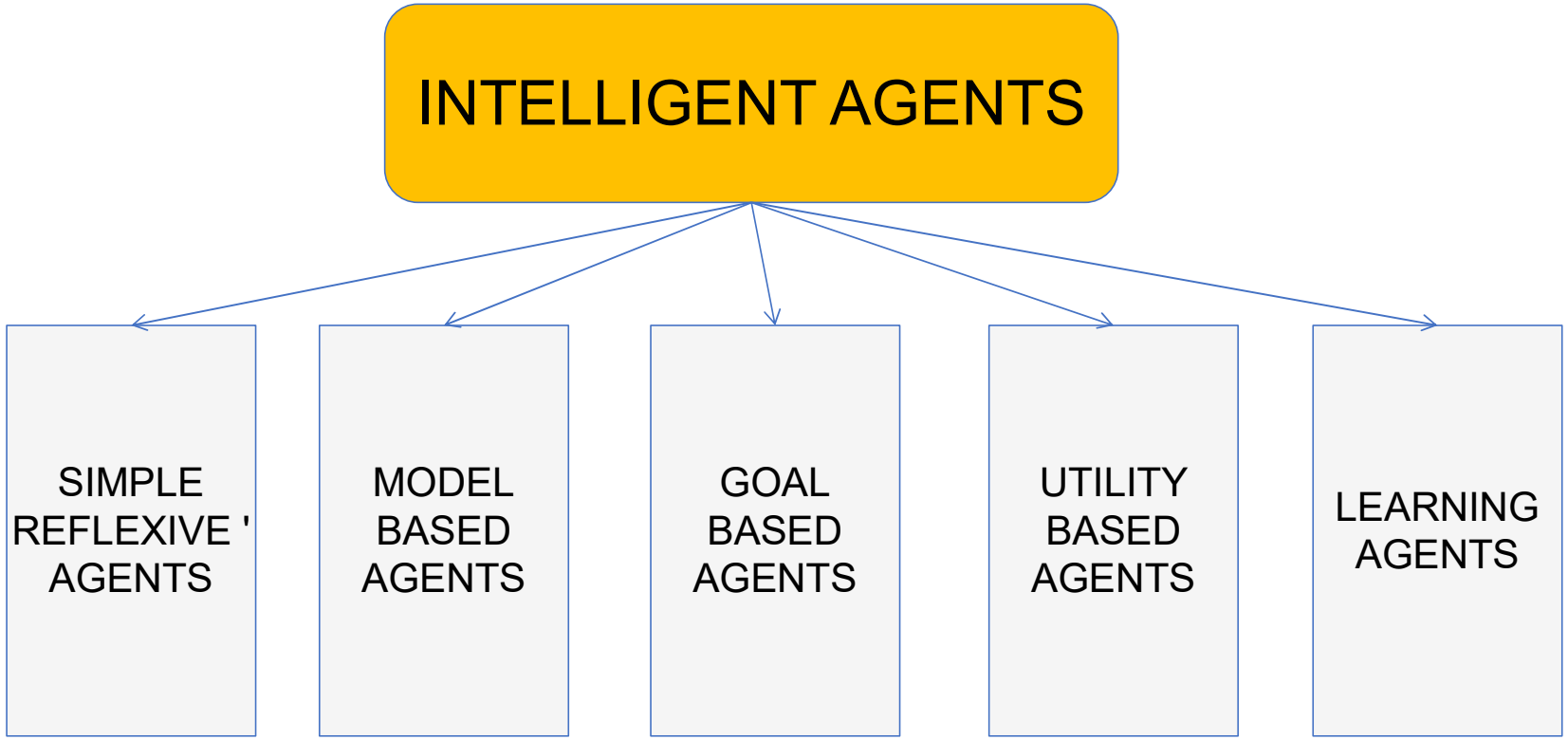
What kind of environment is this?

- FULLY OBSERVABLE
- DETERMINISTIC
- SEQUENTIAL
- STATIC

MACHINE INTELLIGENCE

Classes of Intelligent Agent

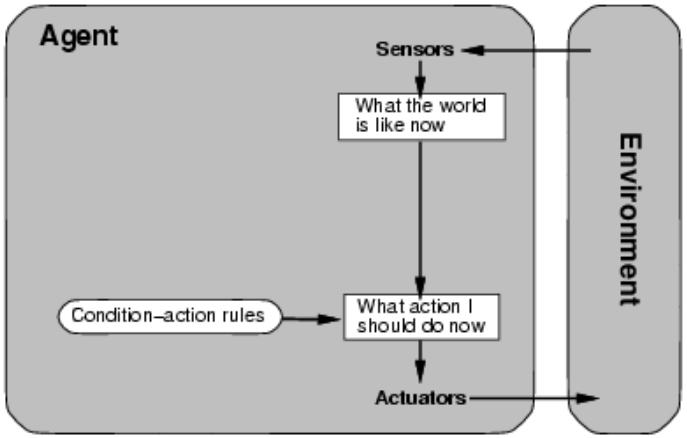
- Intelligent agents are grouped in to five classes based on their degree of perceived intelligence and capability.



MACHINE INTELLIGENCE

Simple Reflex Agent

- Simple reflex agents act only on the basis of the current percept ,ignoring the rest of the percept history. the agent function is based on the condition-action rule: **if condition then action**
- succeeds when the environment is fully observable
- some reflex agents can also contain info on their current state which allows them to disregard conditions




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

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

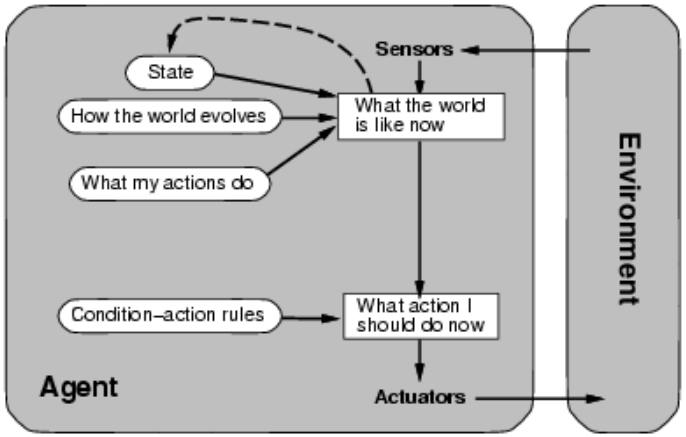
example : metal-detector

it doesn't matter if the previous input detected the metal ,the agent only alerts based on current percept.

MACHINE INTELLIGENCE

Model Based Reflex Agent

- A model-based agent can handle a partially observable environment.
- Its current state is stored inside the agent, maintaining some kind of structure which describes the part of the world which cannot be seen.
- This knowledge about “how the world evolves” is called a model of the world,hence the name “model-based agent”



function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *state*, the agent's current conception of the world state
    model, a description of how the next state depends on current state and action
    rules, a set of condition–action rules
    action, the most recent action, initially none

  state ← UPDATE-STATE(*state*, *action*, *percept*, *model*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← *rule*.ACTION
  return *action*

Model Based Reflex Agent- example

example : self driving cars

lets us consider one scenario

if the car in front of our agent shows red light

using condition action rules would make our agent to stop immediately ,but what if he is trying to slow down,

using model reflex agents it would remember that instead of bringing its speed to zero it should slow down else try to change the lane.

MACHINE INTELLIGENCE

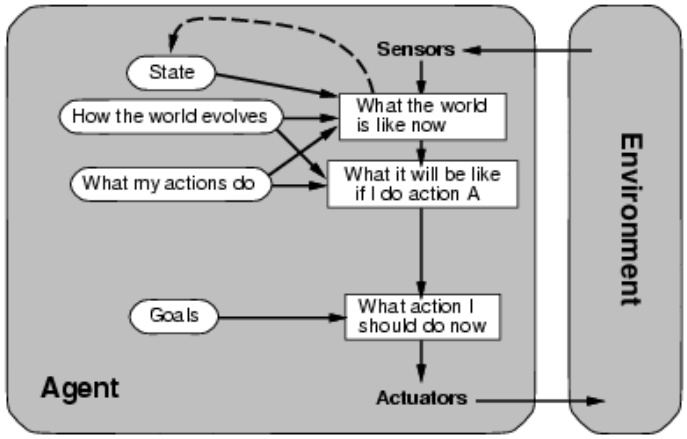
Waymo -Self Driving Car

source: youtube

MACHINE INTELLIGENCE

Goal Based Agent

- Goal-based agents further expand on the capabilities of the model-based agents, by using “goal” information.
- Goal information describes situations that are desirable. This allows the agent a way to choose among multiple possibilities,selecting the one which reaches a goal state.
- search and planning are the sub fields of artificial intelligence devoted to finding action sequences that achieve the agents goals.



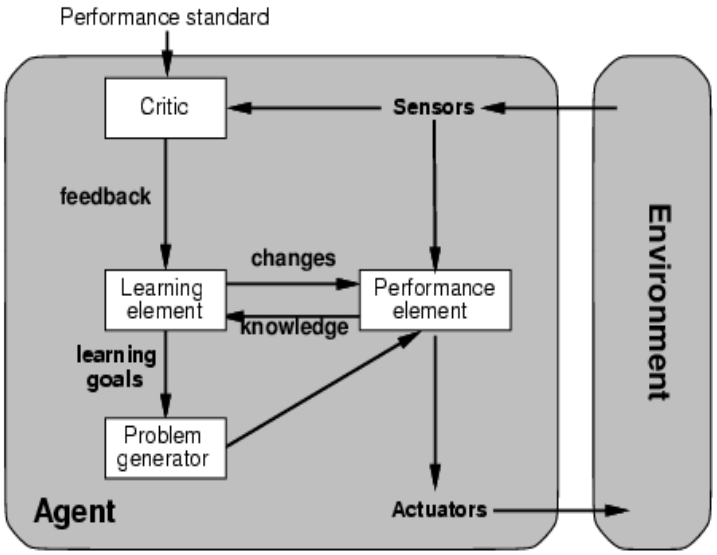
MACHINE INTELLIGENCE

Inside Alibaba's Warehouse

MACHINE INTELLIGENCE

Goal Based Agent

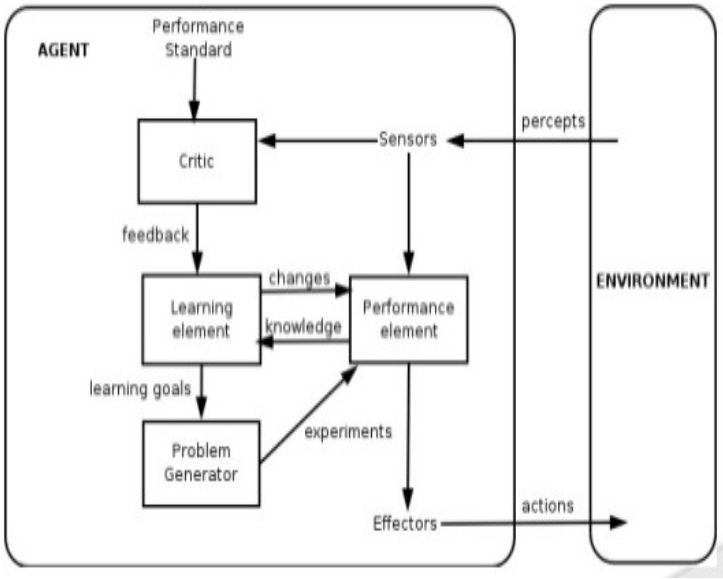
- Goal-based agents only distinguish between goal states and non-goal states
- it is possible to define a measure of how desirable a particular state is. This measure can be obtained through the use of utility function which maps a state to measure of the utility of the state
- A more general performance measure should allow a comparison of different world states accordingly to exactly how happy they would make the agent.



MACHINE INTELLIGENCE

Learning Agent

- Learning has an advantage that it allows the agents to initially operate in unknown environments and to become more competent that its initial knowledge alone might allow.
- the most important distinction is between the “learning element” which is responsible for making improvements and the “performance element” which is responsible for selecting external actions.
- The learning element uses feedback from the critic on how the agent is doing and determines how the performance element should be modified to do better in the future
- The last component of the learning agent is the problem generator.It is responsible for suggesting actions that will lead to new and informative experiences.



THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701

MACHINE INTELLIGENCE

Machine Learning Models

K.S.Srinivas
Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Machine Learning Models

K.S.Srinivas
Department of Computer Science and Engineering

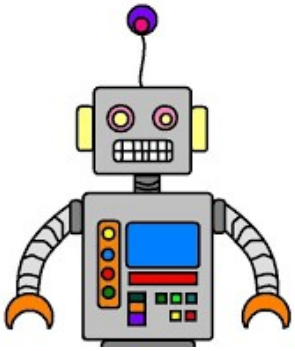
MACHINE INTELLIGENCE

Machine Learning – an intuitive definition

- Consider the world ,we have humans and we have computers
- Can we get computers to learn from experience too???
- YES -and that is precisely what machine learning means
- but for computers we have a different term for experience that is data

Learn from experience

Learn from ~~experience~~ data



Follow instructions

PES UNIVERSITY ONLINE

Machine Learning is

using ***data*** *to*

answer questions

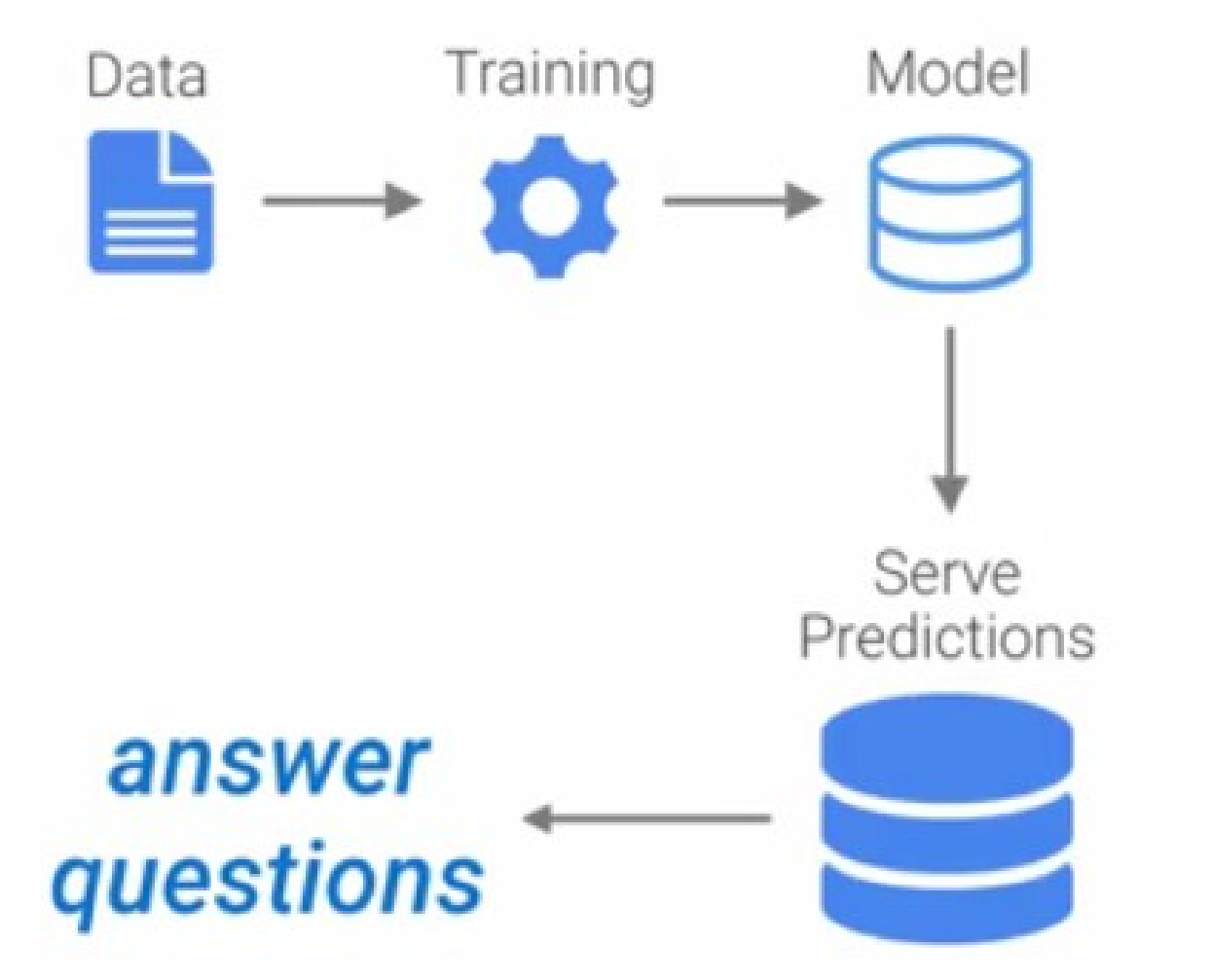
MACHINE INTELLIGENCE

Two broad Phases of Machine Learning

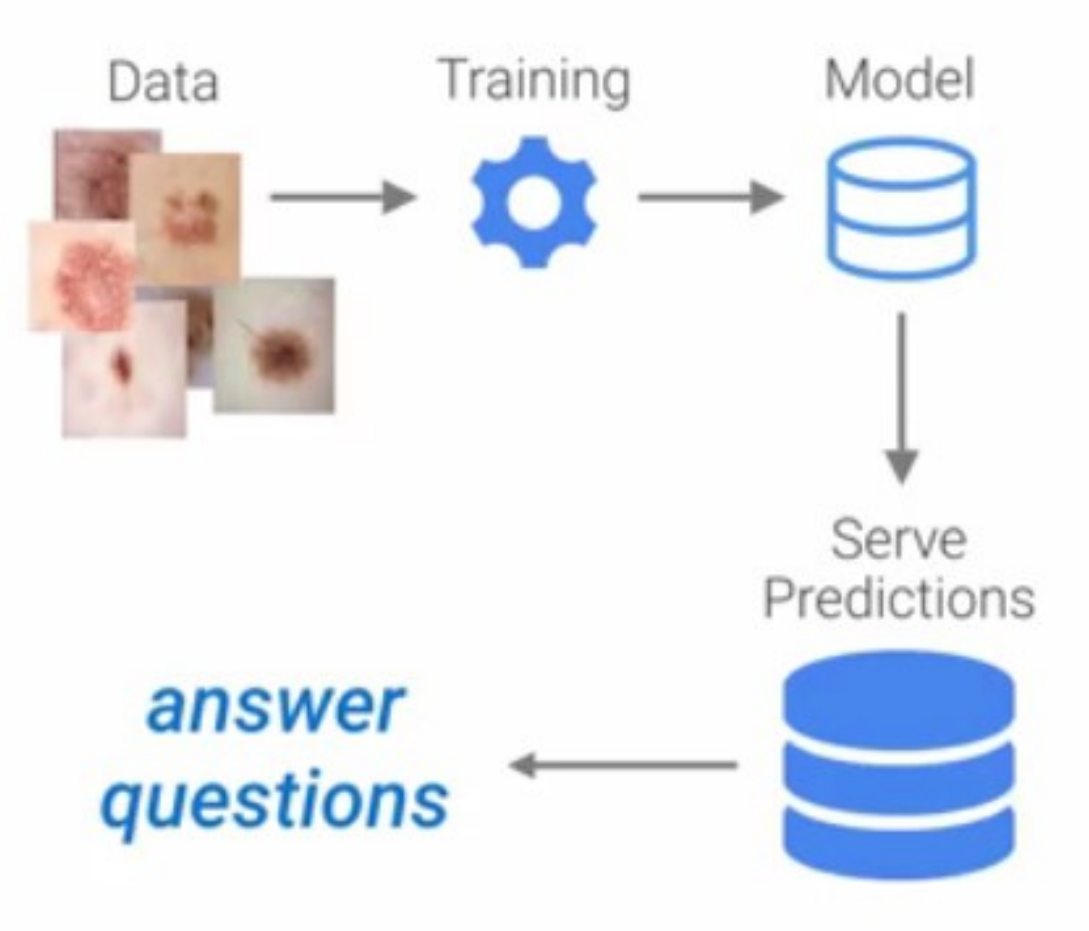
- Training refers to using our data to inform the creation and fine tuning of a predictive model.
- The predictive model is used to serve up predictions on previously unseen data and answer those queries.

MACHINE INTELLIGENCE

Machine Learning – As a flow

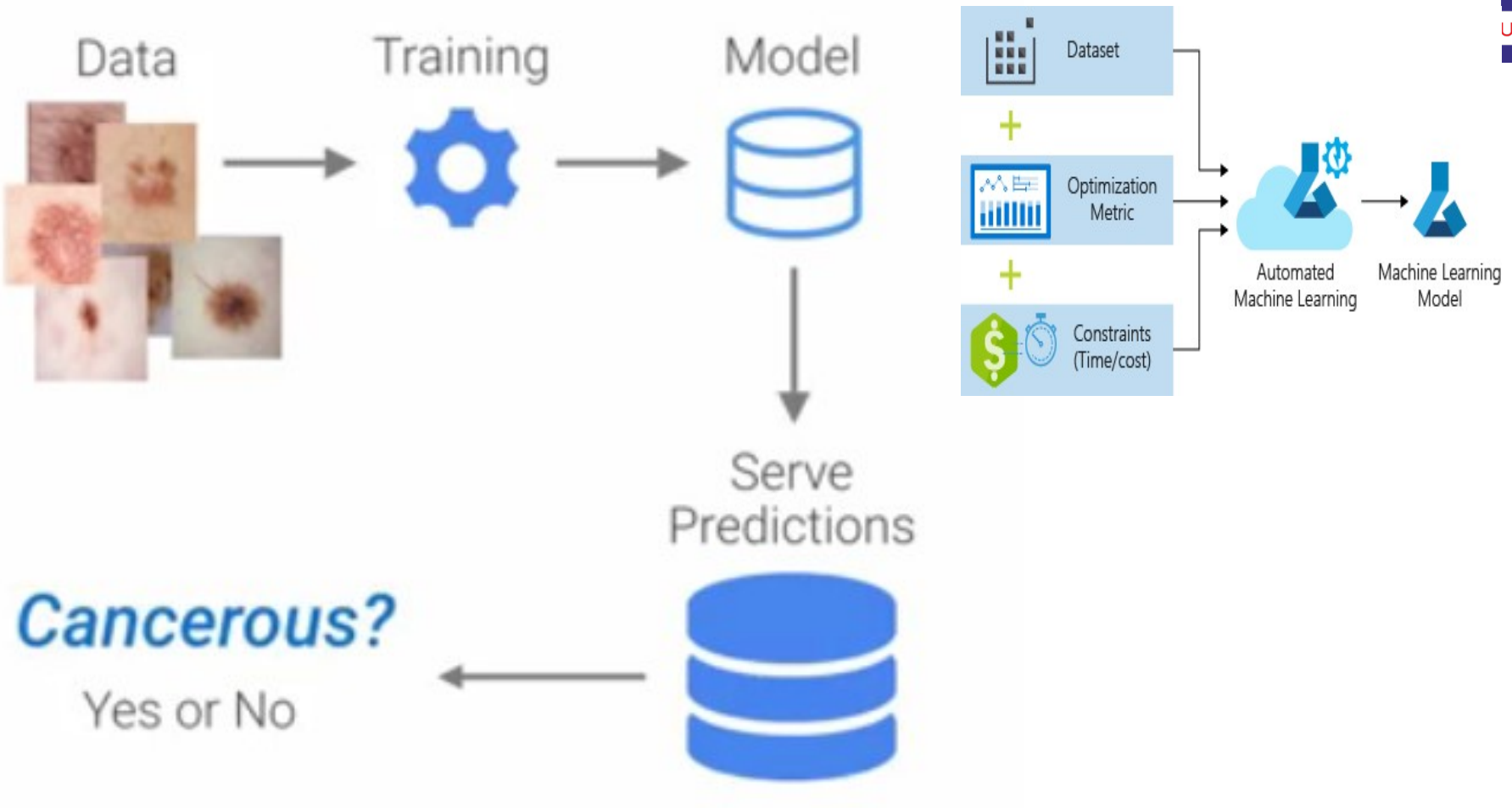


Machine Learning



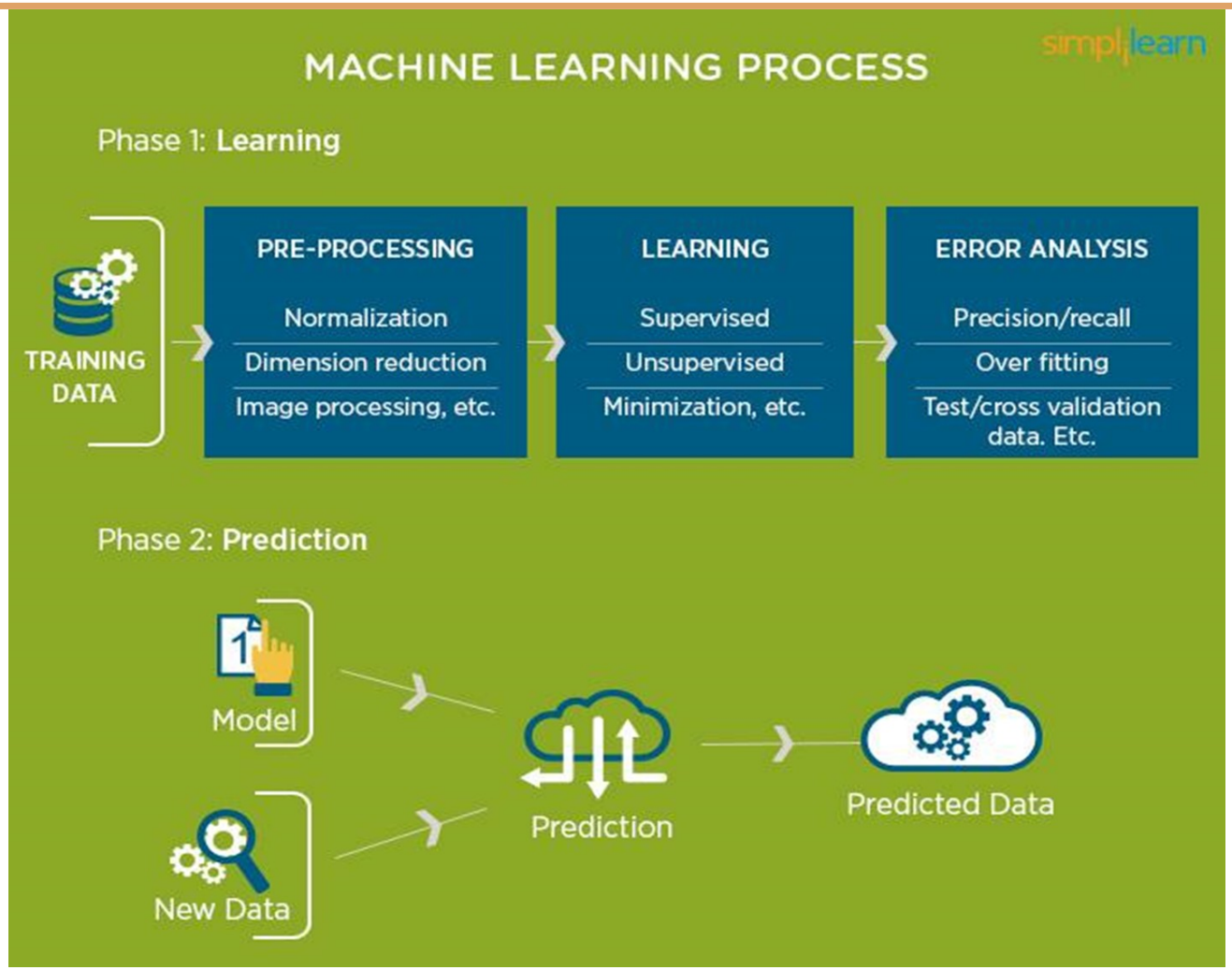
MACHINE INTELLIGENCE

Inputs to a machine learning model



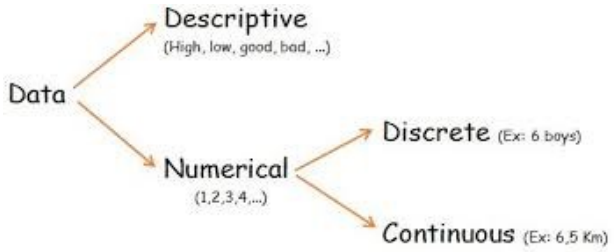
MACHINE INTELLIGENCE

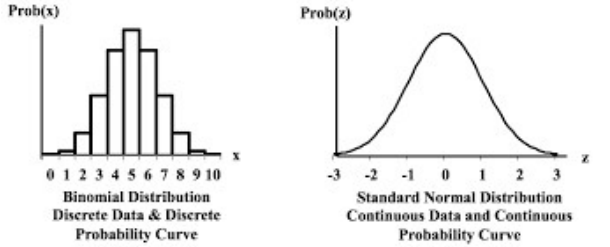
Metrics to Machine Learning Algorithm Evaluation



MACHINE INTELLIGENCE

Data Types



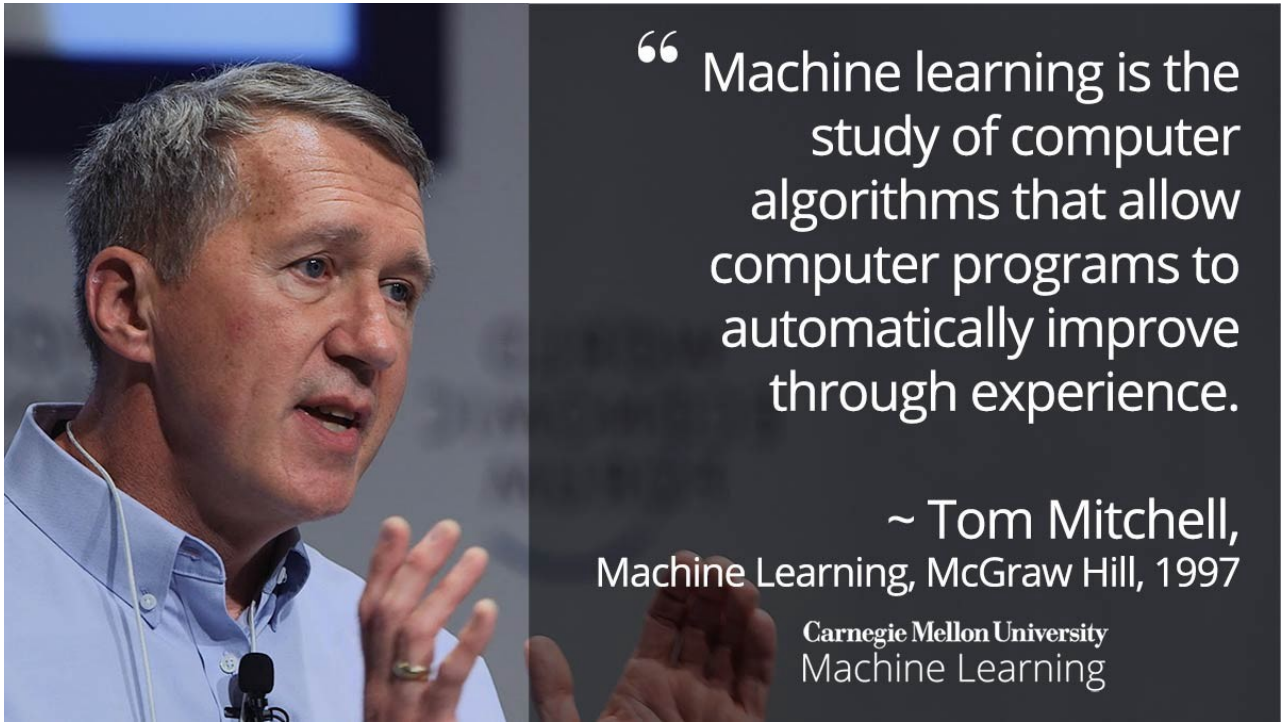


7 Steps to Machine Learning

- Gathering data
- Data preparation
- Choosing a model
- Training
- Evaluation
- Parameter tuning
- Prediction

MACHINE INTELLIGENCE

Formal Definition of Machine Learning



PTE Machine Learning Model Definition

"Learning is any process by which a system improves performance from experience."

- Herbert Simon

Definition by Tom Mitchell (1998):

Machine Learning is the study of algorithms that

- improve their performance P
- at some task T
- with experience E.

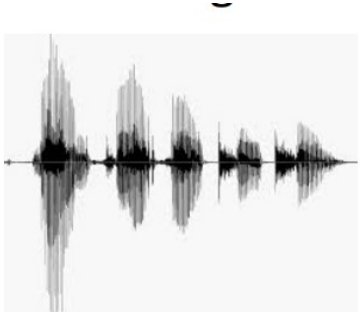
A well-defined learning task is given by <P,T,E>.

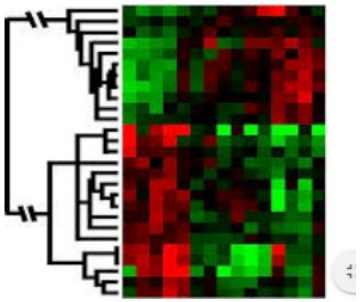
MACHINE INTELLIGENCE

Machine Learning usage

ML is used when:

- Human expertise does not exist (navigating on Mars)
- Humans can’t explain their expertise (speech recognition)
- Models must be customized (personalized medicine)
- Models are based on huge amounts of data (genomics)

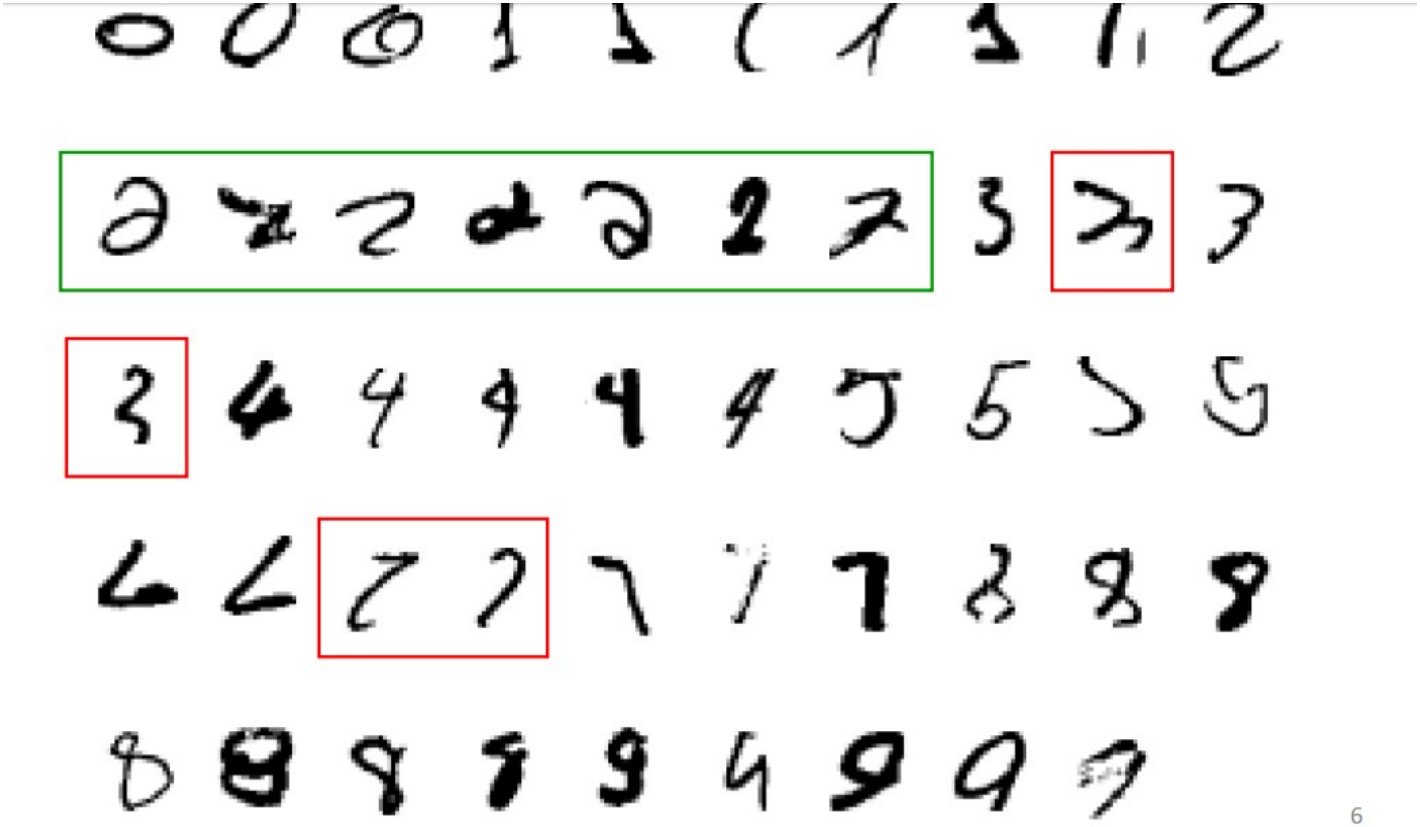





Example of a Machine Learning Task

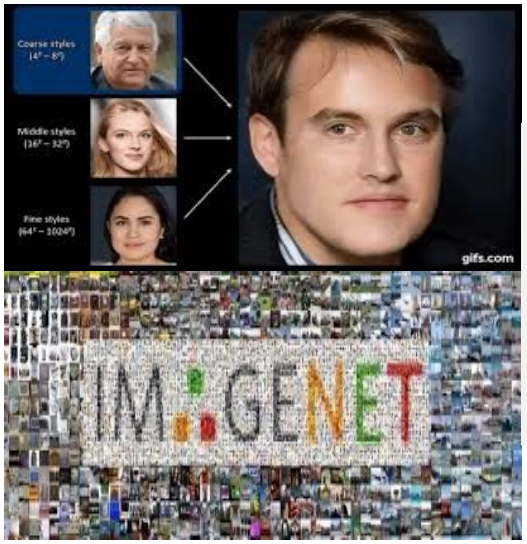
A classic example of a task that requires machine learning:

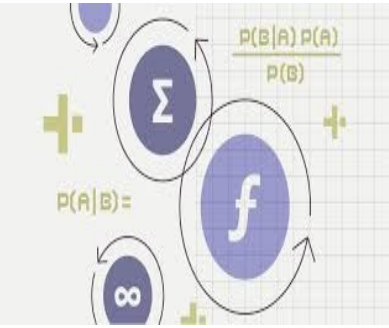
It is very hard to say what makes a 2:



Machine Learning Classes

Some more examples of tasks that are best solved by using a learning algorithm:

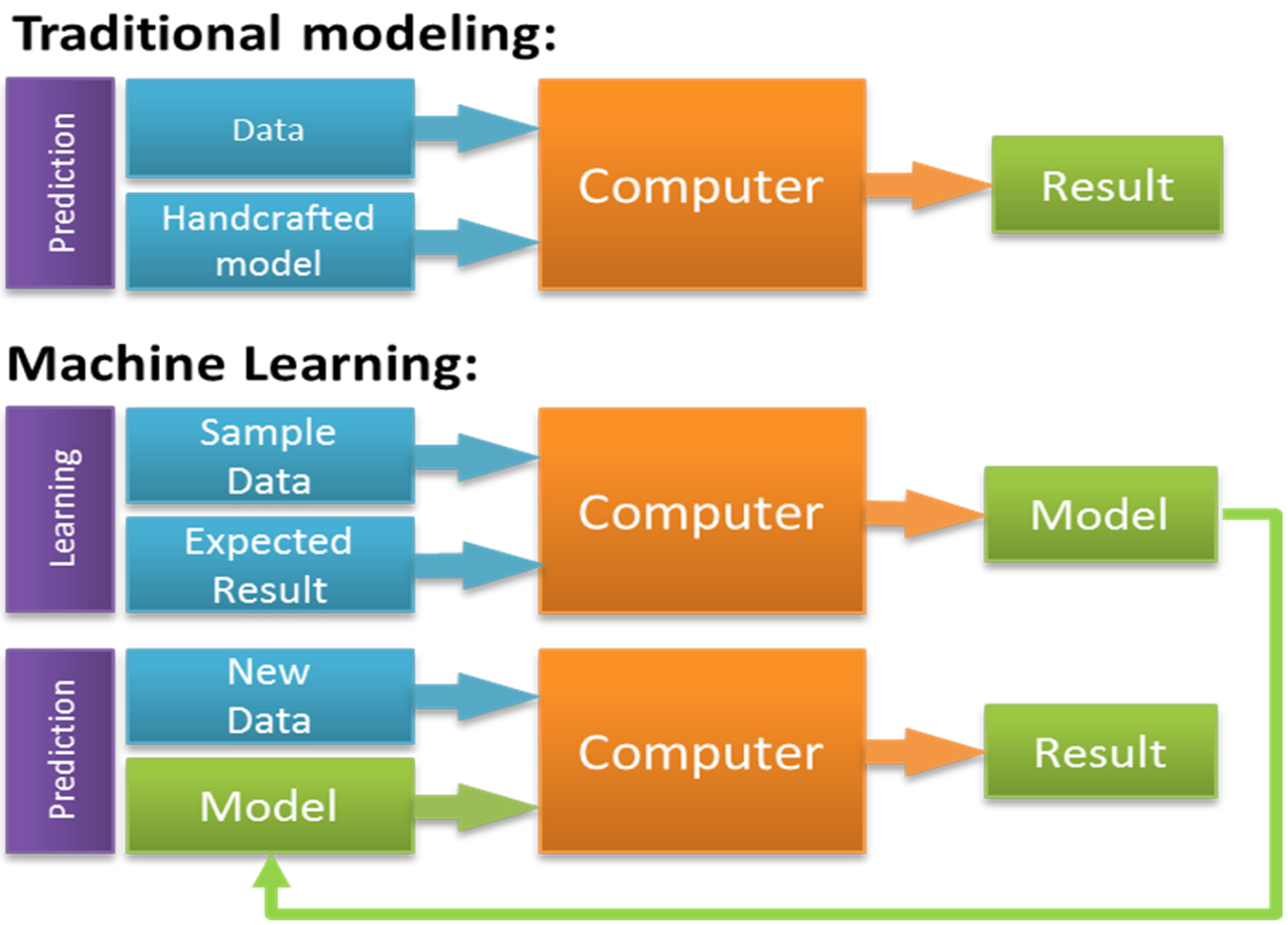




- Recognizing patterns:
  - Facial identities or facial expressions
  - Handwritten or spoken words
  - Medical images
- Generating patterns:
  - Generating images or motion sequences
- Prediction:
  - Future stock prices or currency exchange rates

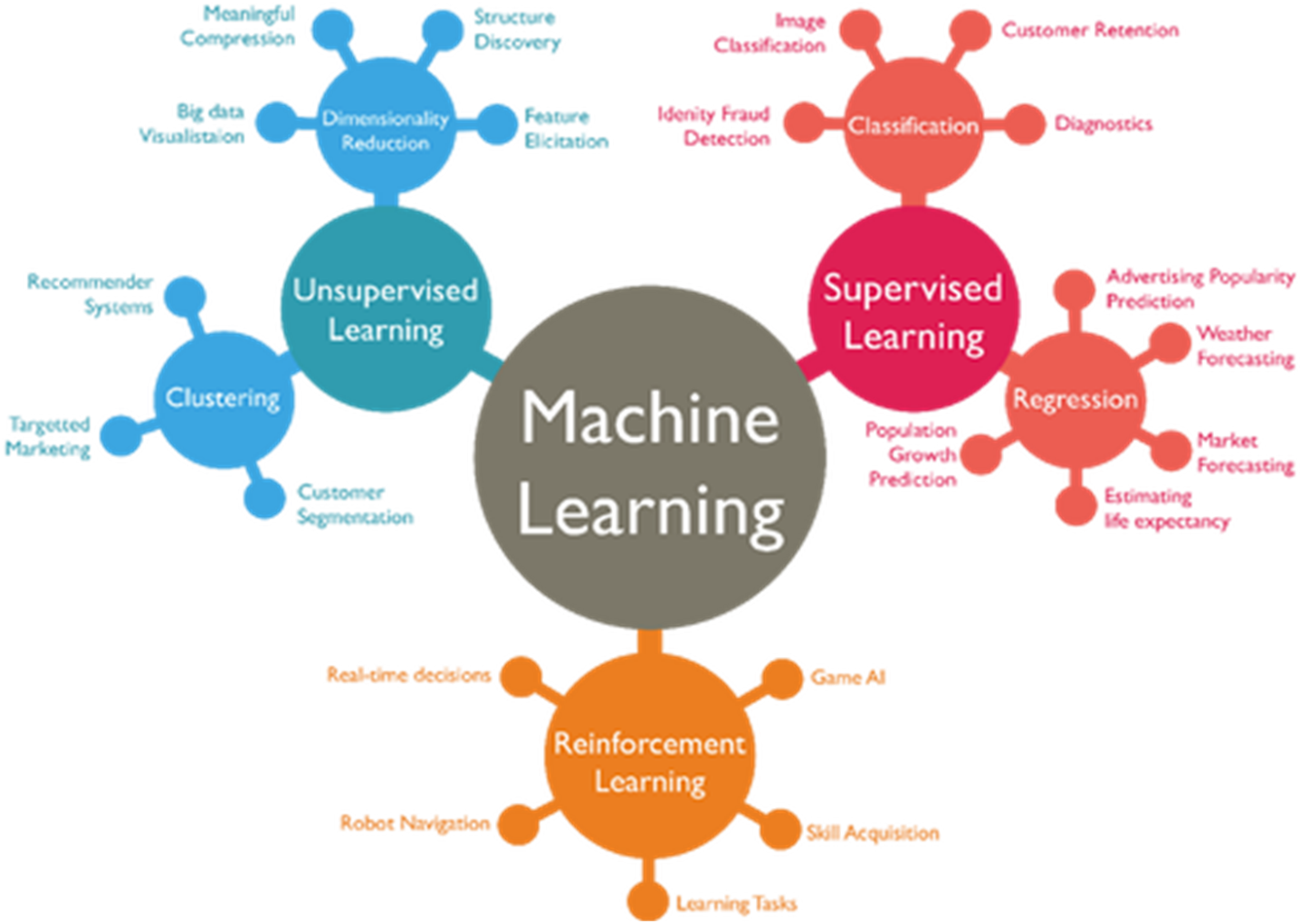
MACHINE INTELLIGENCE

Traditional Models are Different



MACHINE INTELLIGENCE

Three Broad Categories of Machine Learning



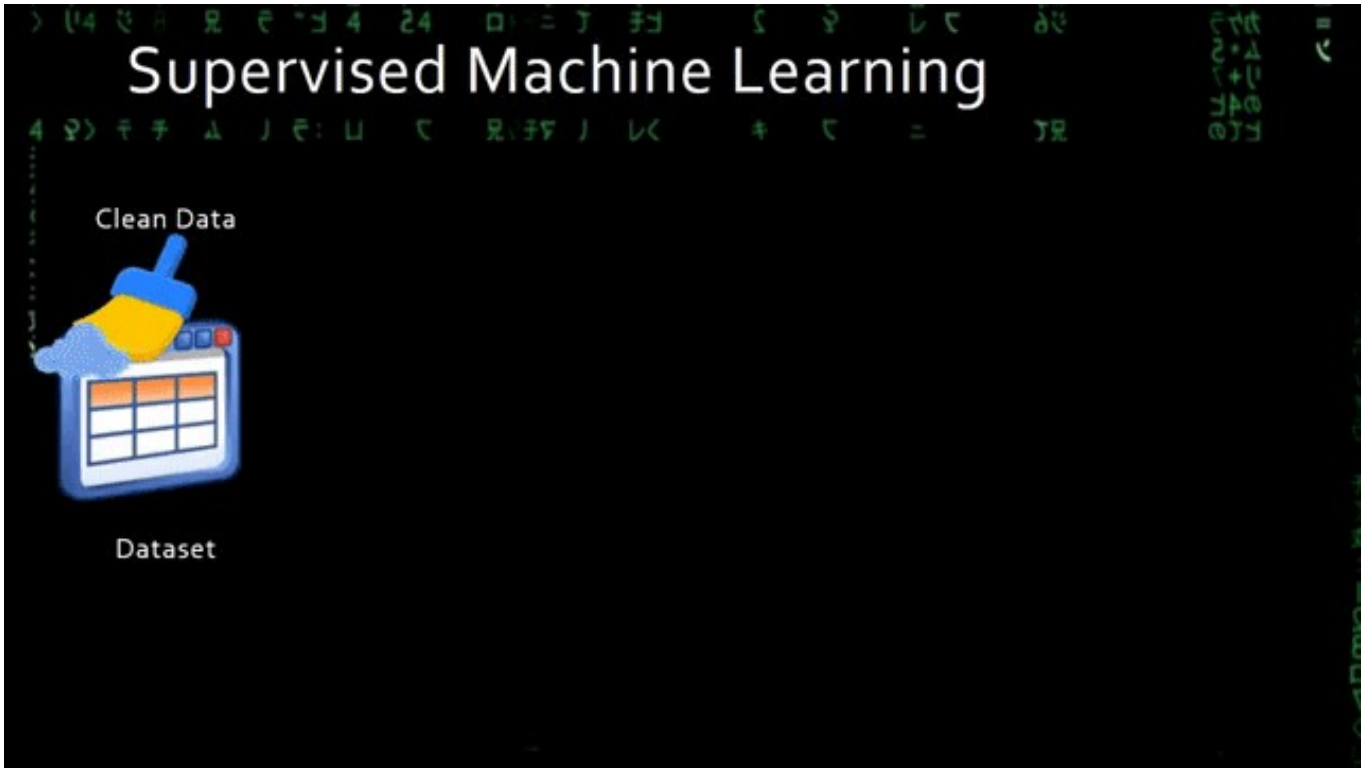
MACHINE INTELLIGENCE

Types of Machine Learning

- Supervised (inductive) learning-
  - Given: training data +desired outputs (labels)
- Unsupervised learning-
  - Given:training data (without desired outputs)
- Reinforcement learning
  - Rewards from sequence of actions

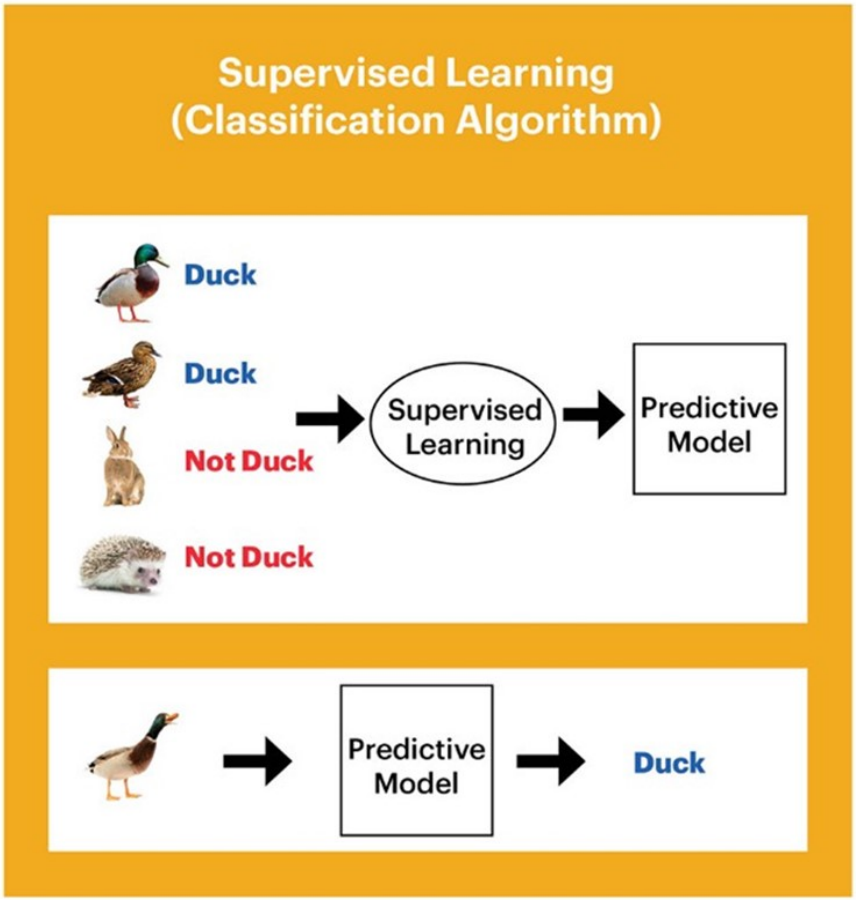
MACHINE INTELLIGENCE

Supervised Learning



MACHINE INTELLIGENCE

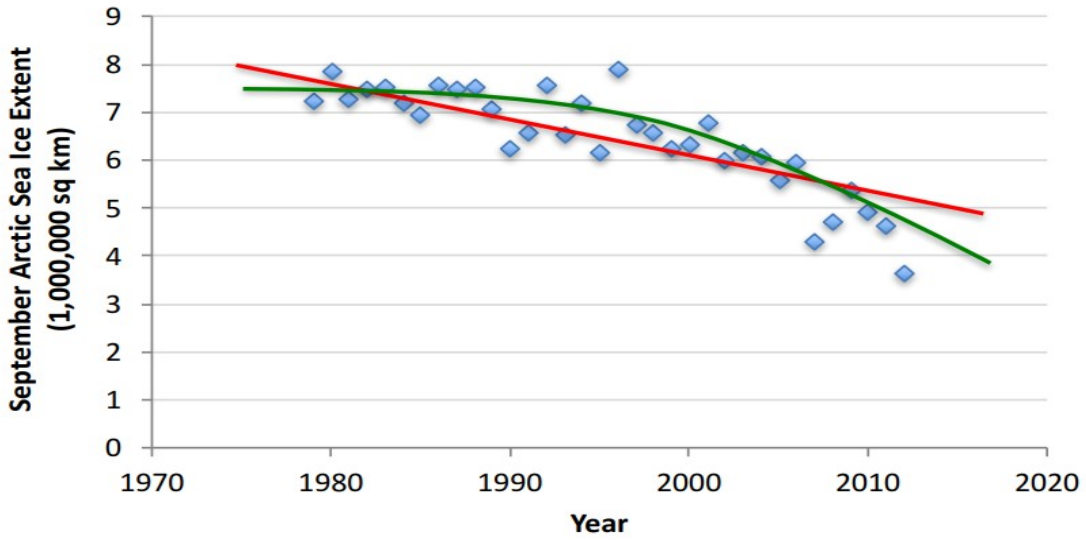
Supervised Learning



Formal Definition of Supervised Learning

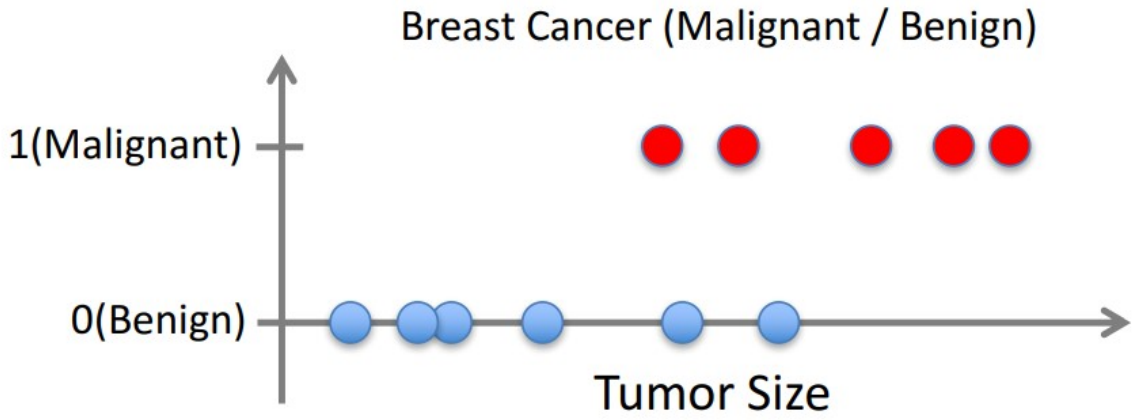
REGRESSION:

- Given $(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)$
- Learn a function f(x) to predict y given x
  - y is real-valued == regression



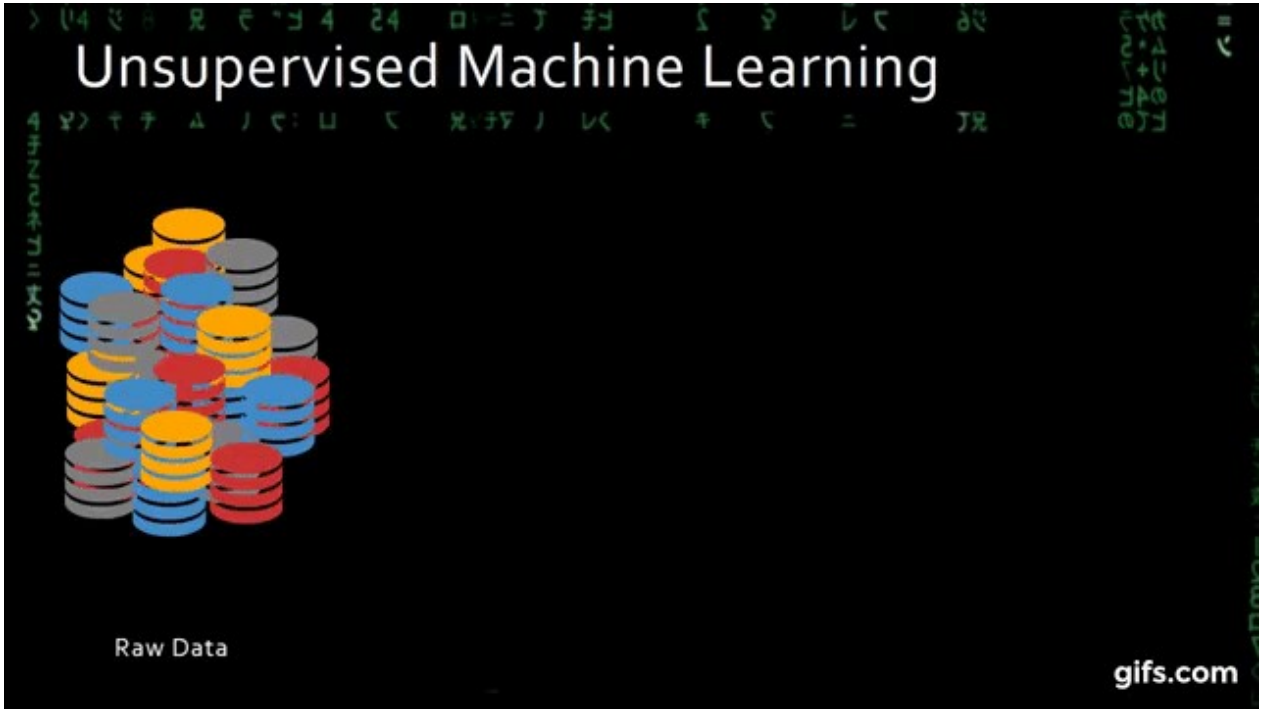
CLASSIFICATION:

- Given $(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)$
- Learn a function f(x) to predict y given x
  - y is categorical== classification



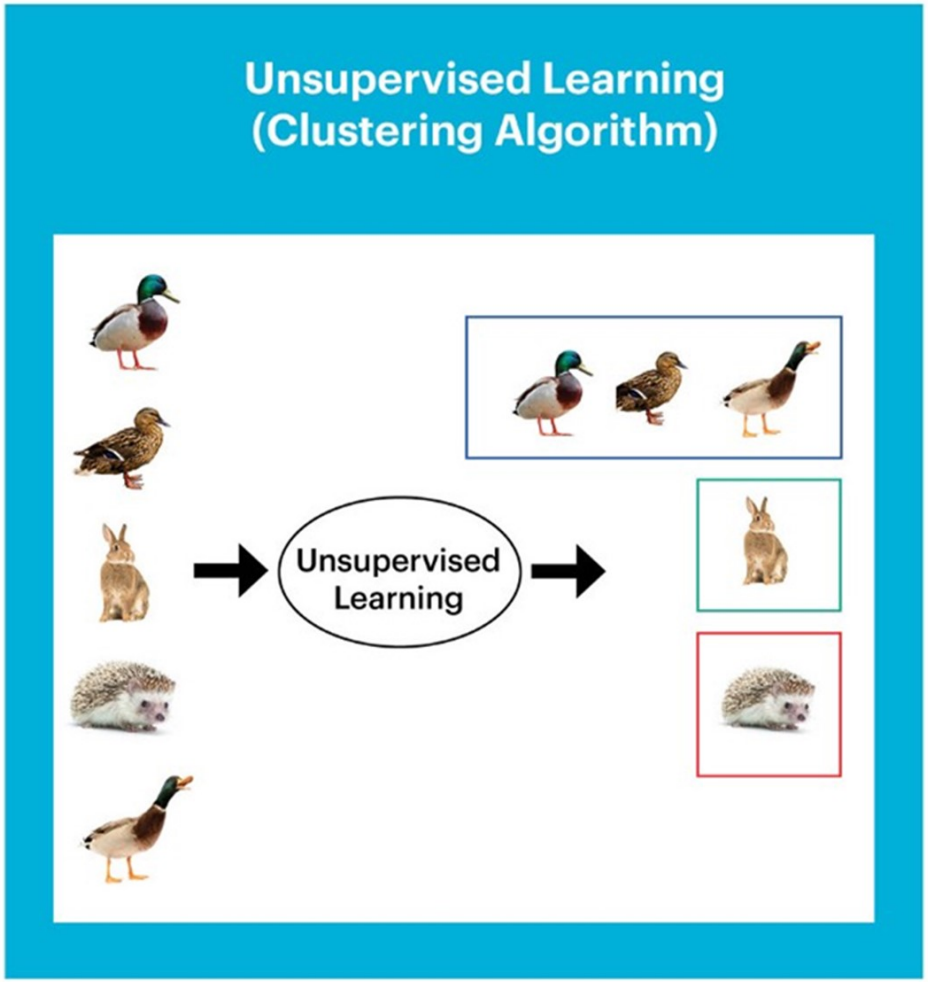
MACHINE INTELLIGENCE

Machine Intelligence is Omni Present



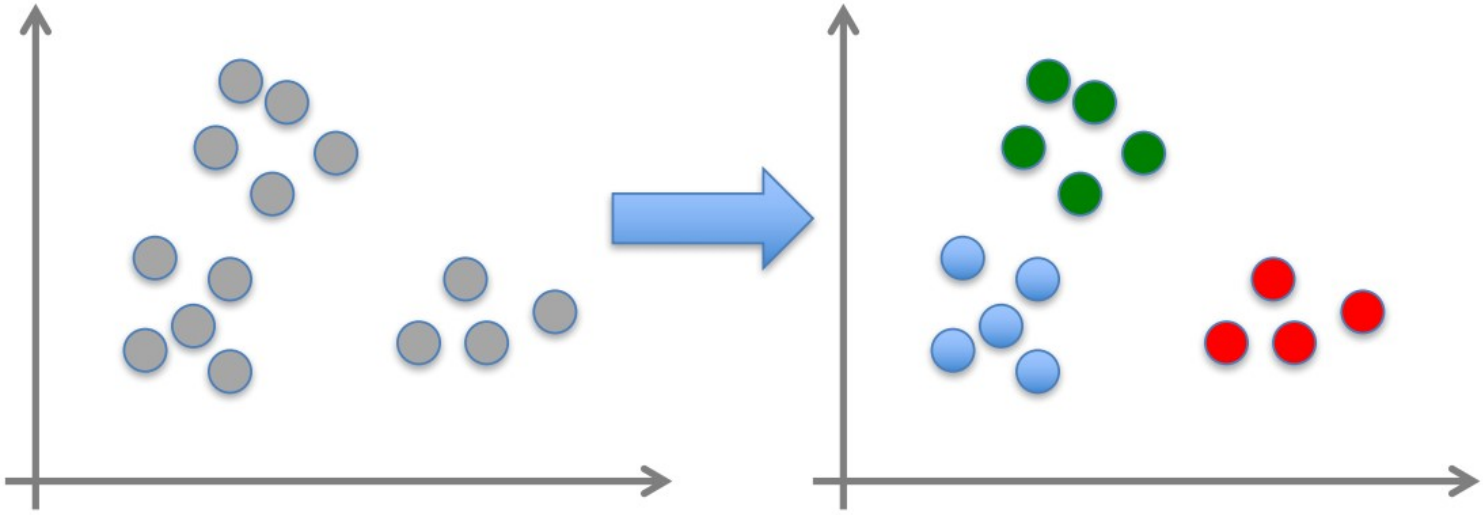
MACHINE INTELLIGENCE

Unsupervised Learning



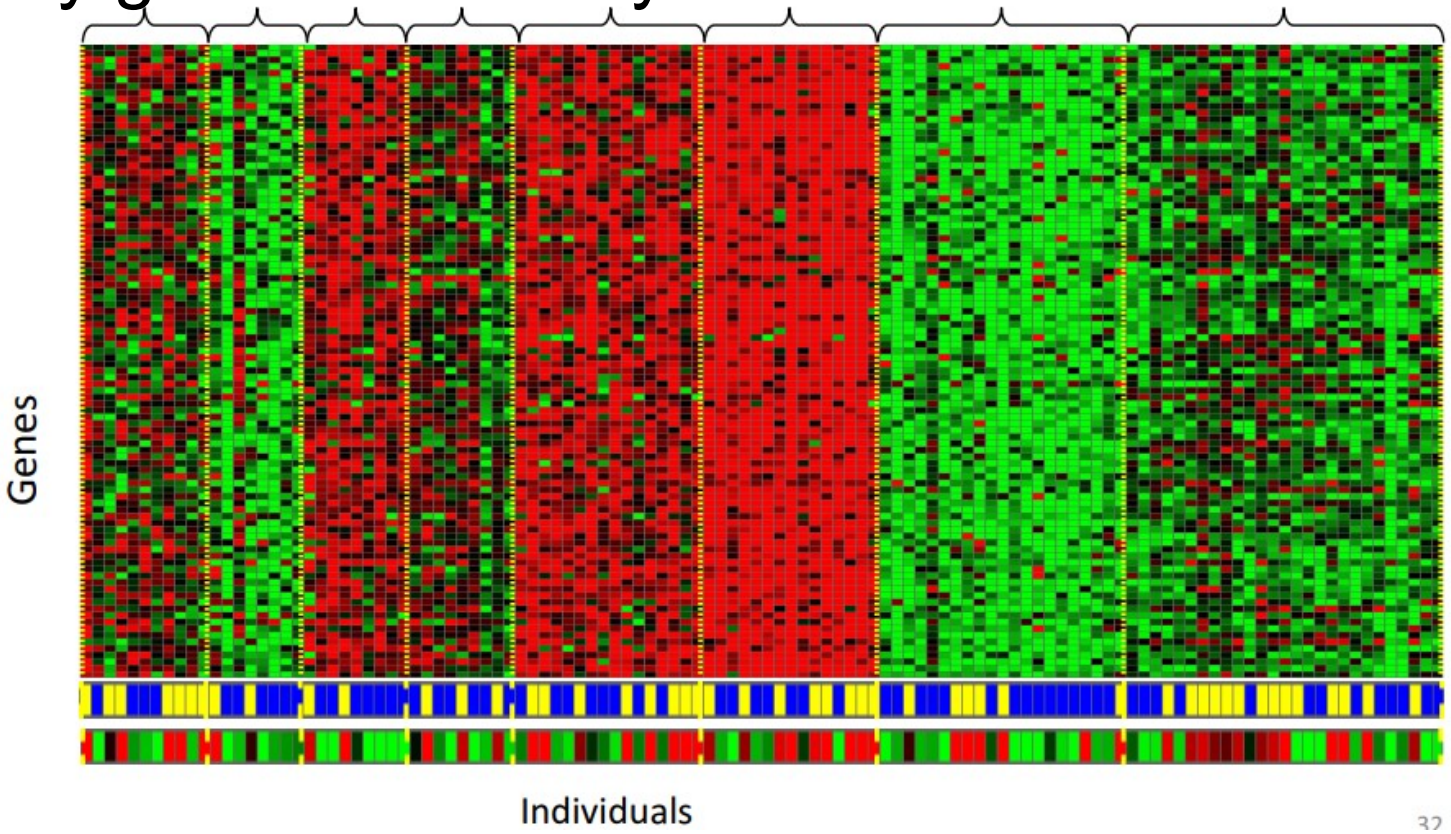
Unsupervised Learning

- Given x_1, x_2, ...,x_n (without labels)
- Output hidden structure behind the x's
  - E.g., clustering



PES UNIVERSITY ONLINE

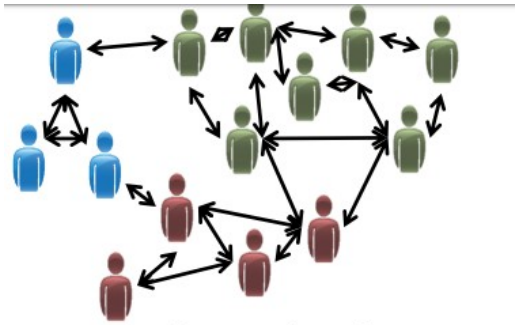
Genomics application: group individuals by genetic similarity



MACHINE INTELLIGENCE

Example of Unsupervised Learning Applications

Organize computing clusters



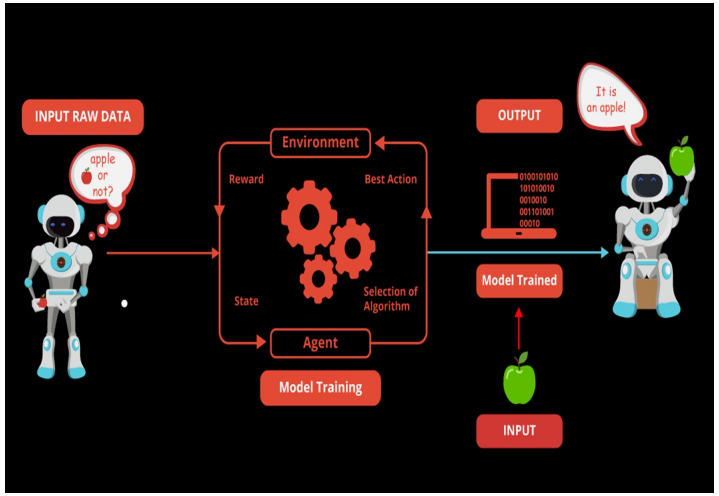
Social network analysis

Market segmentation

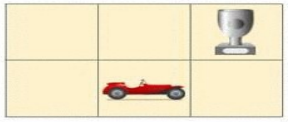
Astronomical data analysis

MACHINE INTELLIGENCE

Reinforcement learning



Game Board:



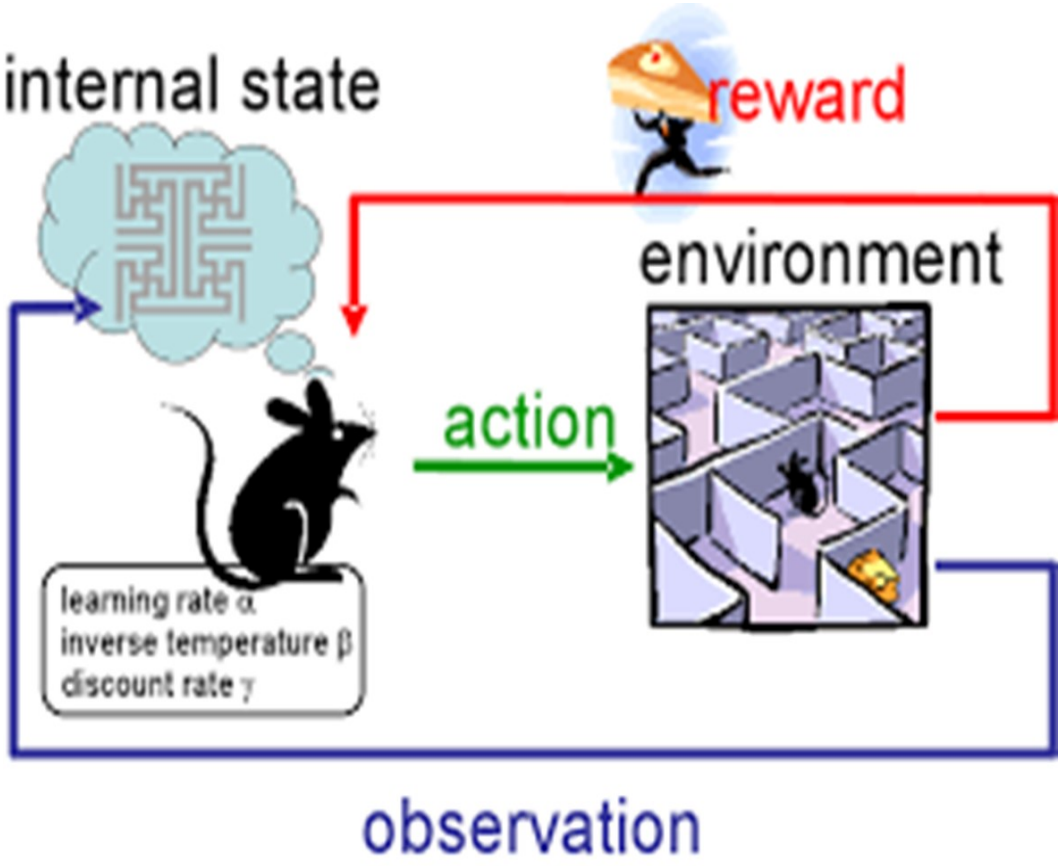
Current state (s): 0 0 0 / 0 1 0

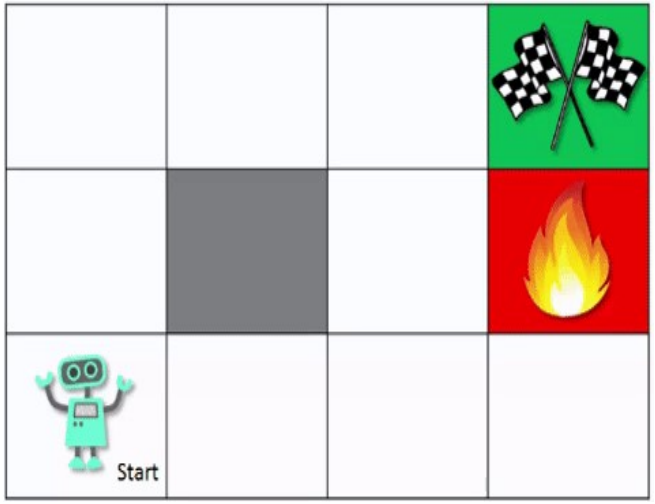
Q Table: $\gamma = 0.95$

	0 0 0 1 0 0	0 0 0 0 1 0	0 0 0 0 0 1	1 0 0 0 0 0	0 1 0 0 0 0	0 0 1 0 0 0
⇧	0.2	0.3	1.0	-0.22	-0.3	0.0
⇩	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
⇨	0.21	0.4	-0.3	0.5	1.0	0.0
⇦	-0.6	-0.1	-0.1	-0.31	-0.01	0.0

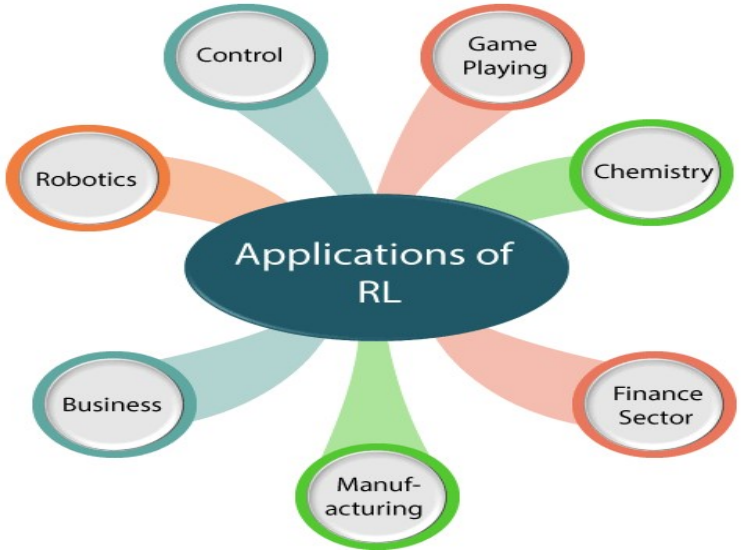
MACHINE INTELLIGENCE

Reinforcement Learning





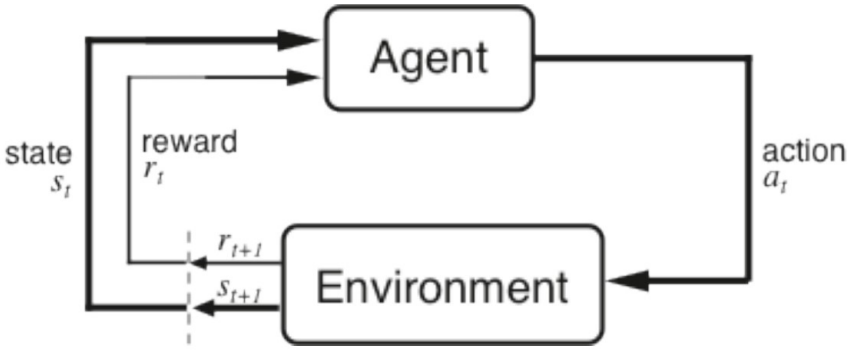
- Given a sequence of states and actions with (delayed) rewards,output a policy.
  - Policy is a mapping from states -> actions ,that tells you what to do in a given state



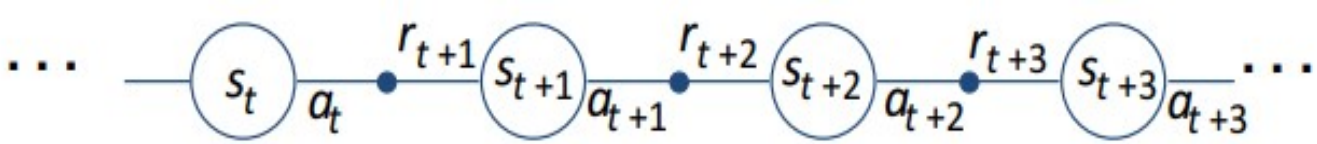
MACHINE INTELLIGENCE

Reinforcement learning - learning process

- The Agent-Environment Interface

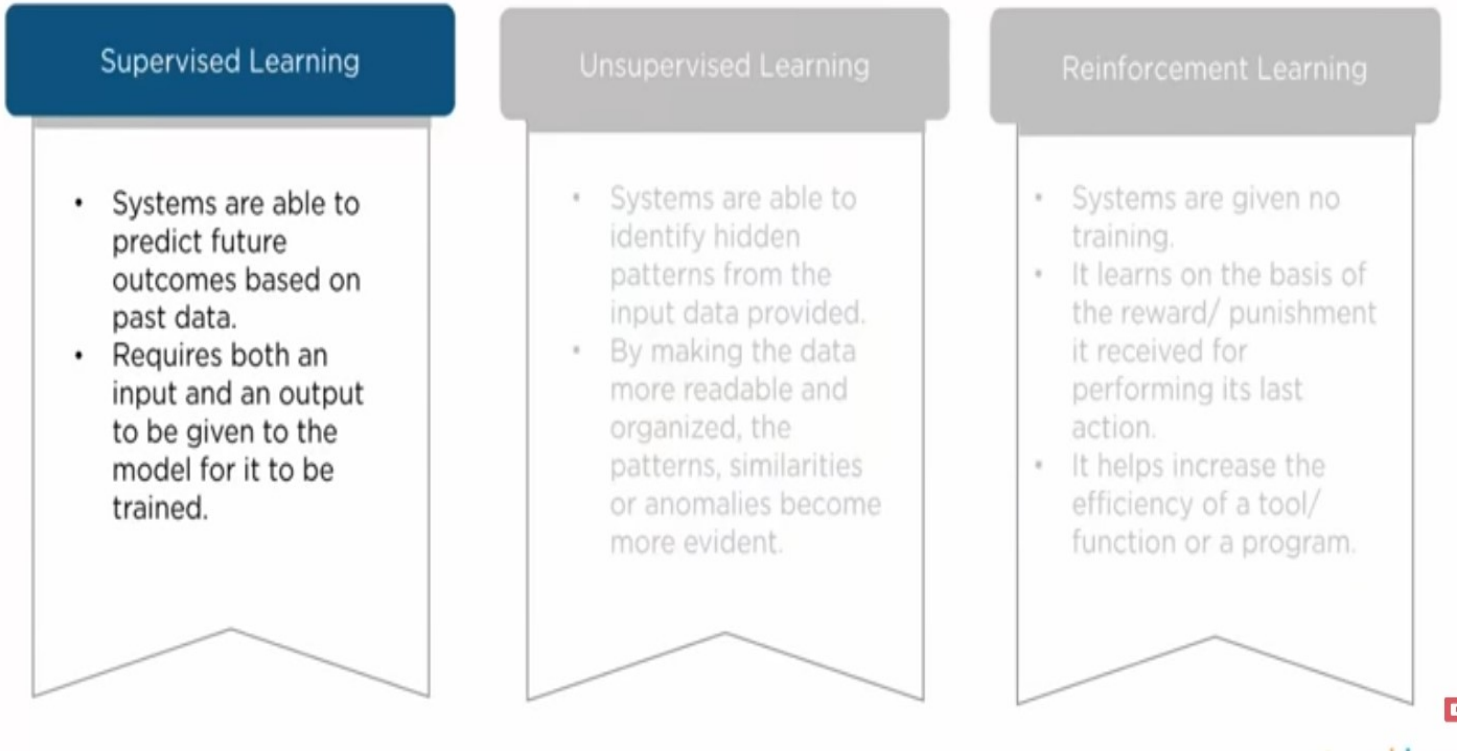


- Agent and environment interact at discrete time steps : t = 0, 1, 2, K
- Agent observes state at step t: s_t □ S
- produces action at step t : a_t □ $A(s_t)$
- gets resulting reward : r_t +1 □ □
- and resulting next state : s_t +1



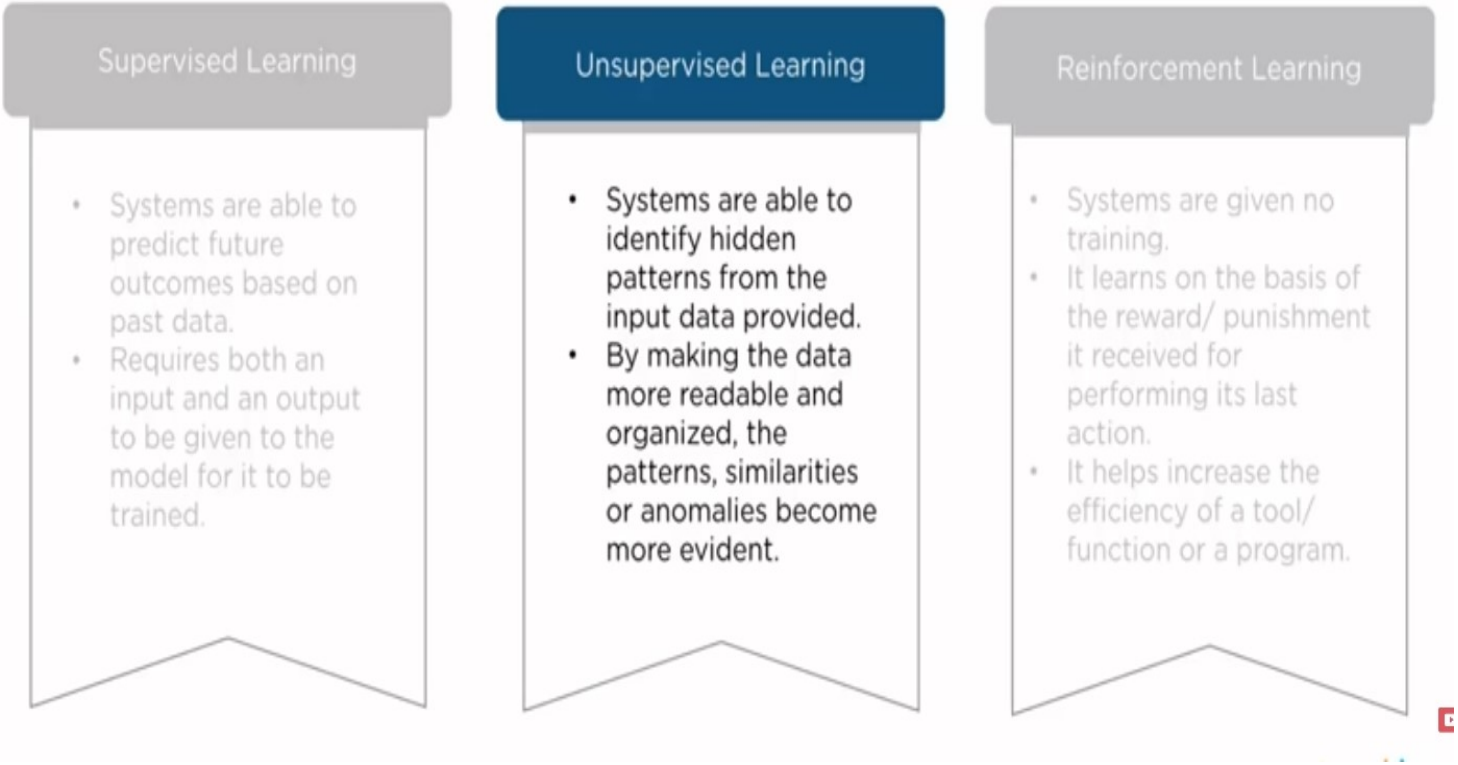
MACHINE INTELLIGENCE

ML Model Summary



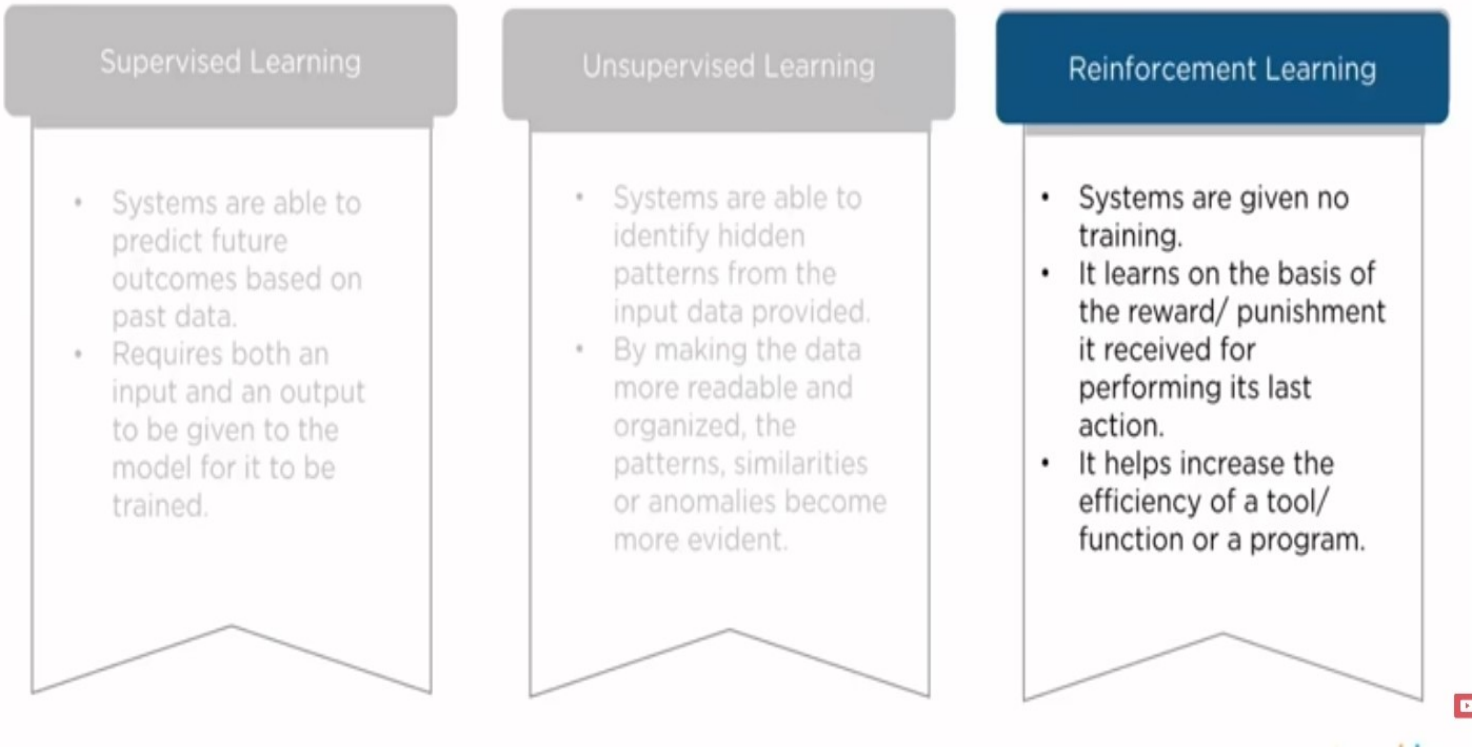
MACHINE INTELLIGENCE

ML Model Summary



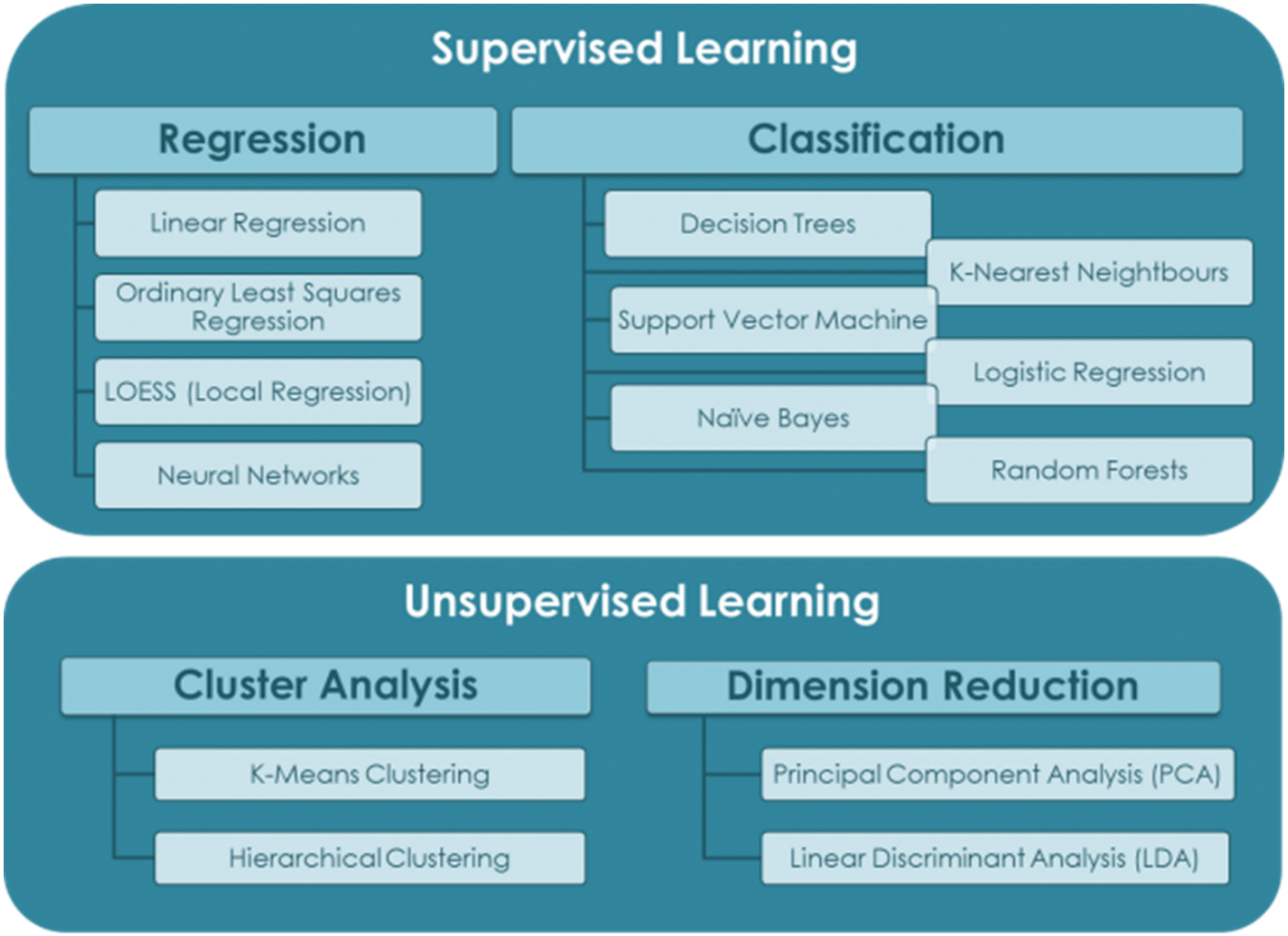
MACHINE INTELLIGENCE

ML Model Summary



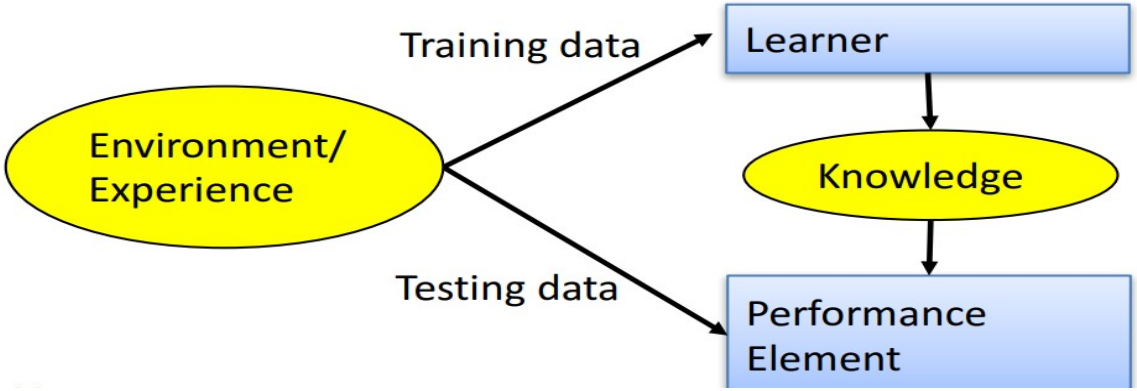
MACHINE INTELLIGENCE

Subtypes within Models



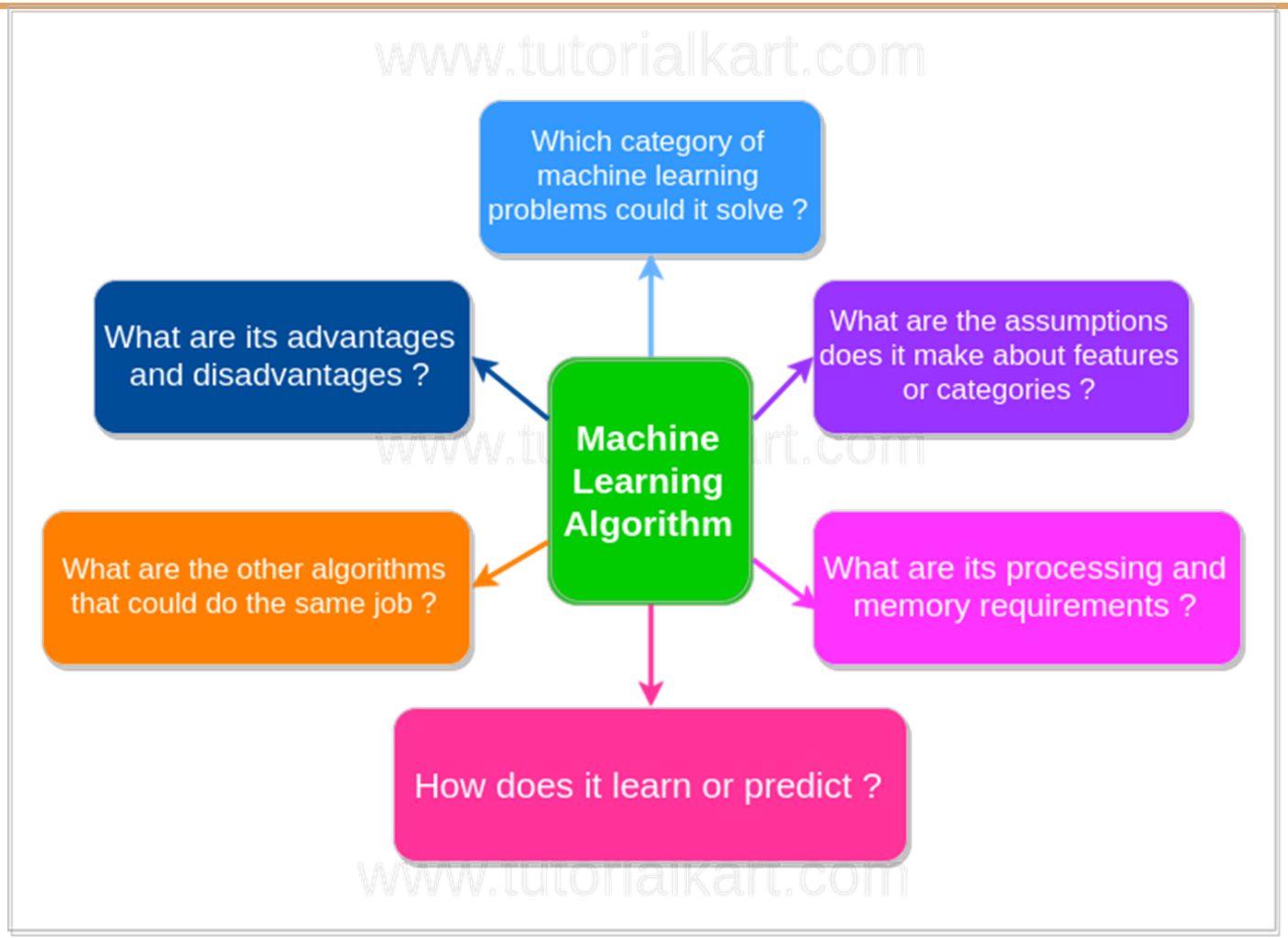
Designing a learning system

- Choose the training experience
- Choose exactly what is to be learned
  - i.e. the target function
- Choose how to represent the target function
- Choose a learning algorithm to infer the target function from the experience



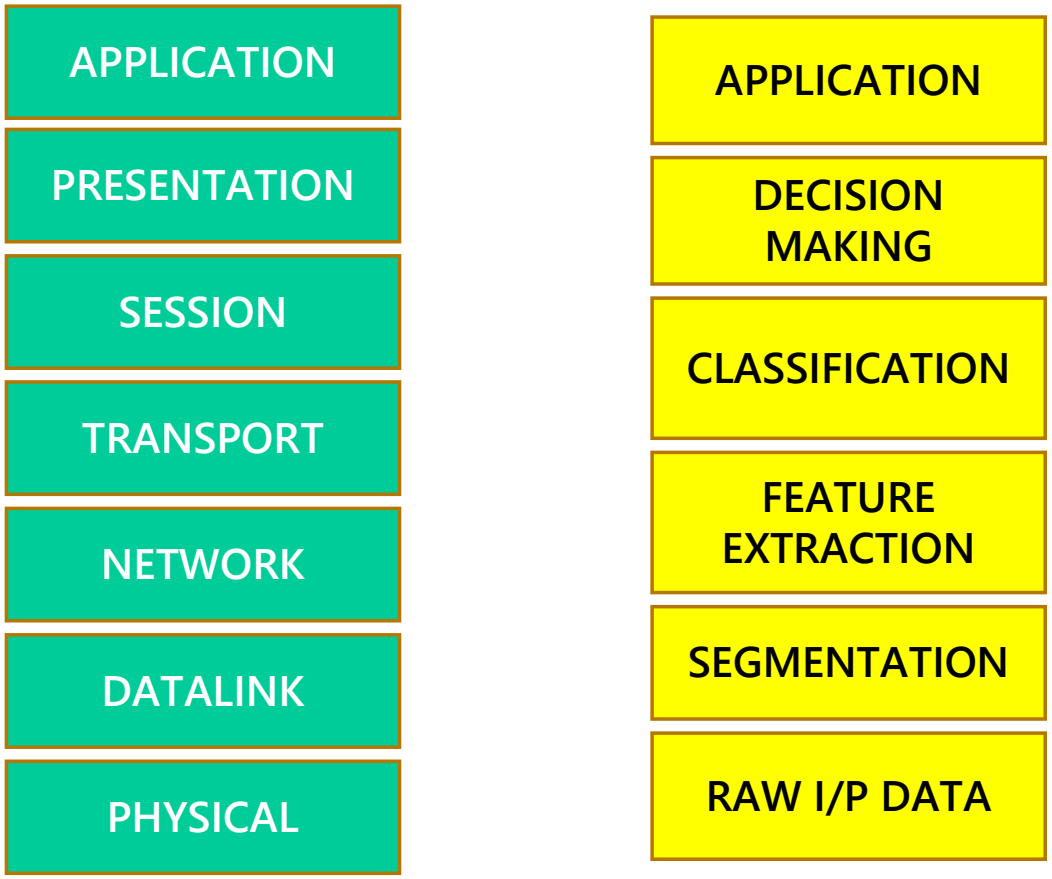
MACHINE INTELLIGENCE

Machine Learning – Choosing the right algorithm



MACHINE INTELLIGENCE

OSI Analogy



THANK YOU

K.S.Srinivas

srinivasks@pes.edu

+91 80 2672 1983 Extn 701

MACHINE INTELLIGENCE

SEARCH STRATEGIES

K.S.Srinivas
Department of Computer Science and Engineering

MACHINE INTELLIGENCE

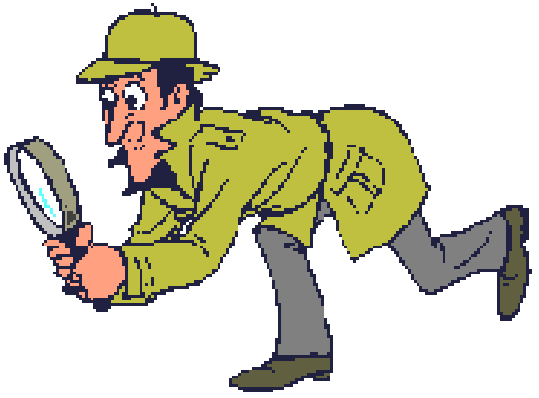
SEARCH STRATEGIES

Srinivas K S.
Associate Professor, Department of Computer Science

MACHINE INTELLIGENCE

What is a search strategy?

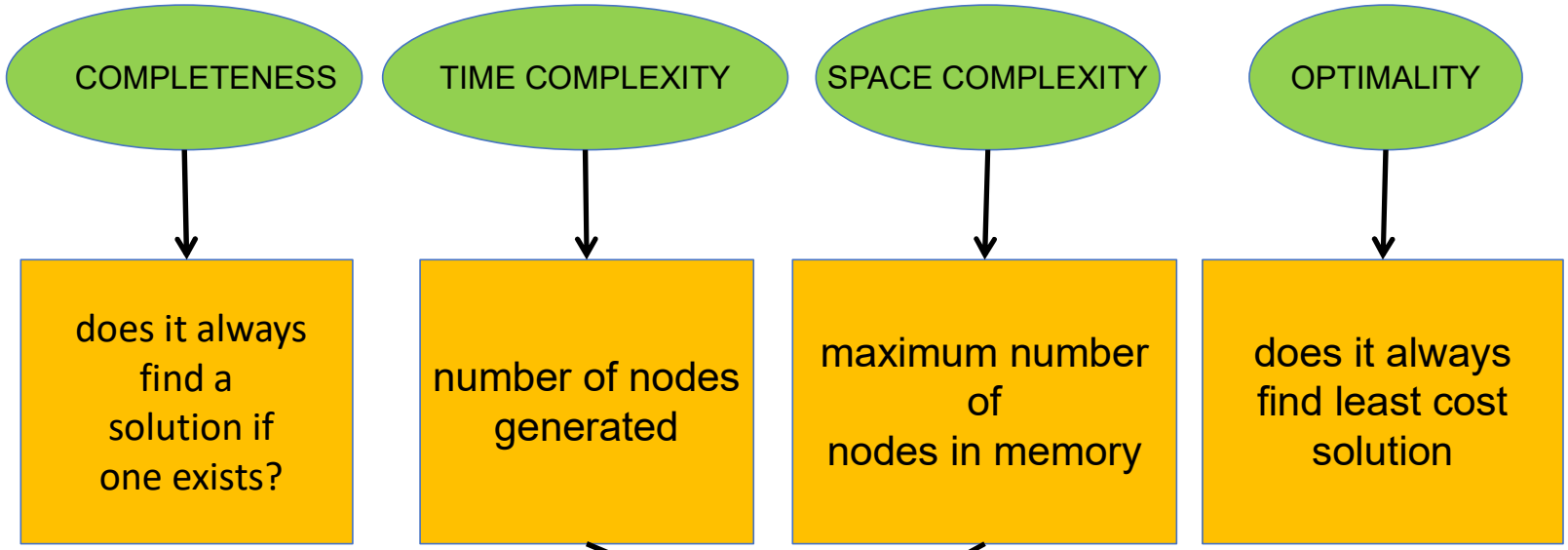
- Whenever you are given a search field problem you are provided with graph and the goal but not the path to select from the frontier.
- This is the job of search strategy.
- A search strategy is defined by picking the order of node expansion.
- Are all strategy equally good?
- what are the testing parameter?



MACHINE INTELLIGENCE

Parameters to define a good strategy

- The goodness of your strategy is subjected to following criteria



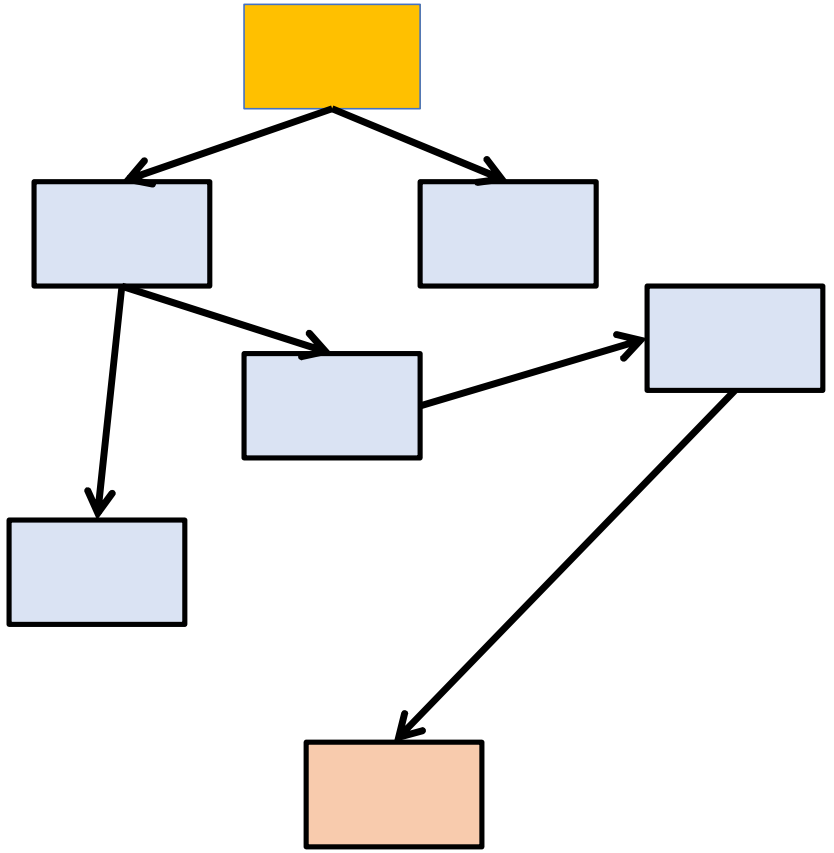
These two criteria are measured in terms of

- *b:* maximum branching factor of the search tree
- *d:* depth of the least-cost solution
- *m*: maximum depth of the state space (may be ∞)

MACHINE INTELLIGENCE

A simple search strategy

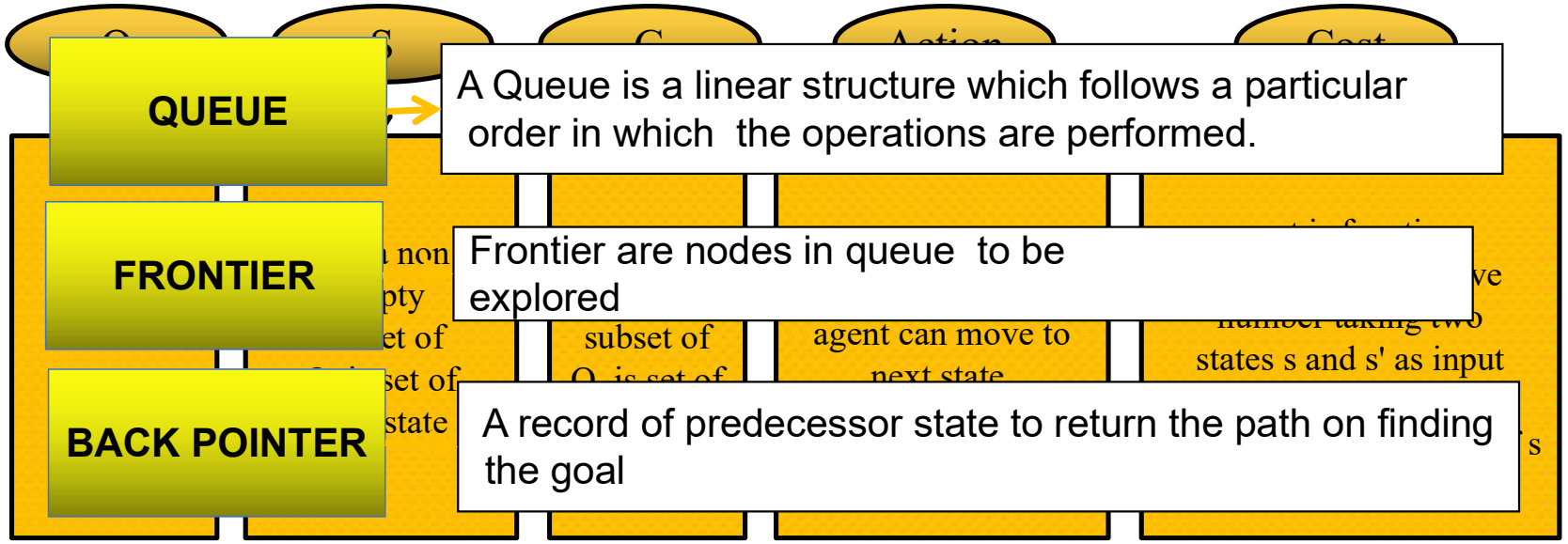
- Define initial state
- Find all possible actions from the state
- Take some step(defined) by the algorithm
- Get to the new state
- Test if the new state is goal
- Repeat this iteratively
- The problem is in defining the state which is absurdly complex in the real world



MACHINE INTELLIGENCE

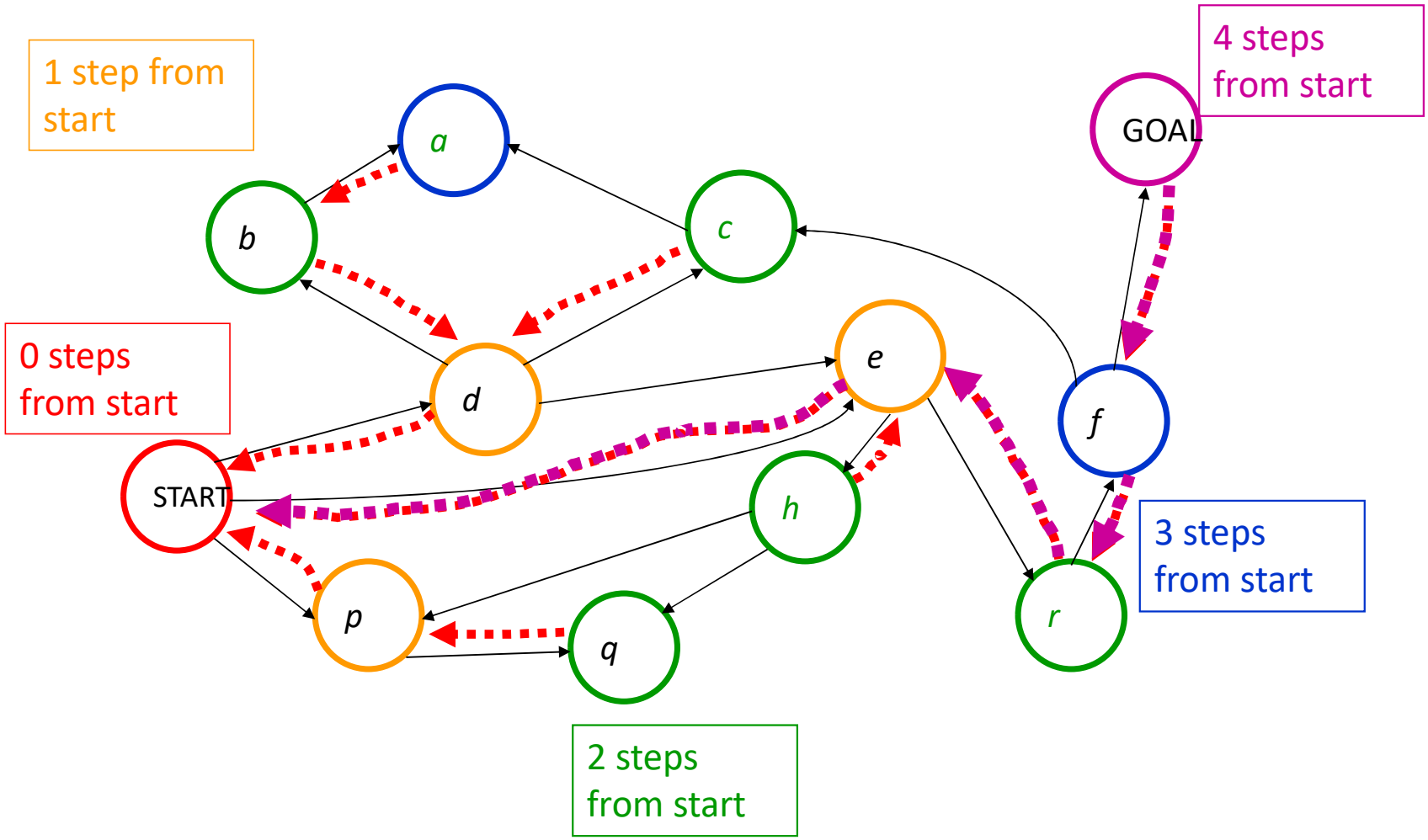
Formalizing a search problem

- Before we move on to how to solve our search problem using different search strategy ,let us see how do we prepare our set up to solve the search problem.
- we begin with defining basic components
- A search problem has 5 basic components
- some more definition that we will be using to solve our searching problem



MACHINE INTELLIGENCE

Tracking with back pointers



MACHINE INTELLIGENCE

Lets analyse

Now that we have learned how to formalize our problem let us try to formalize this problem

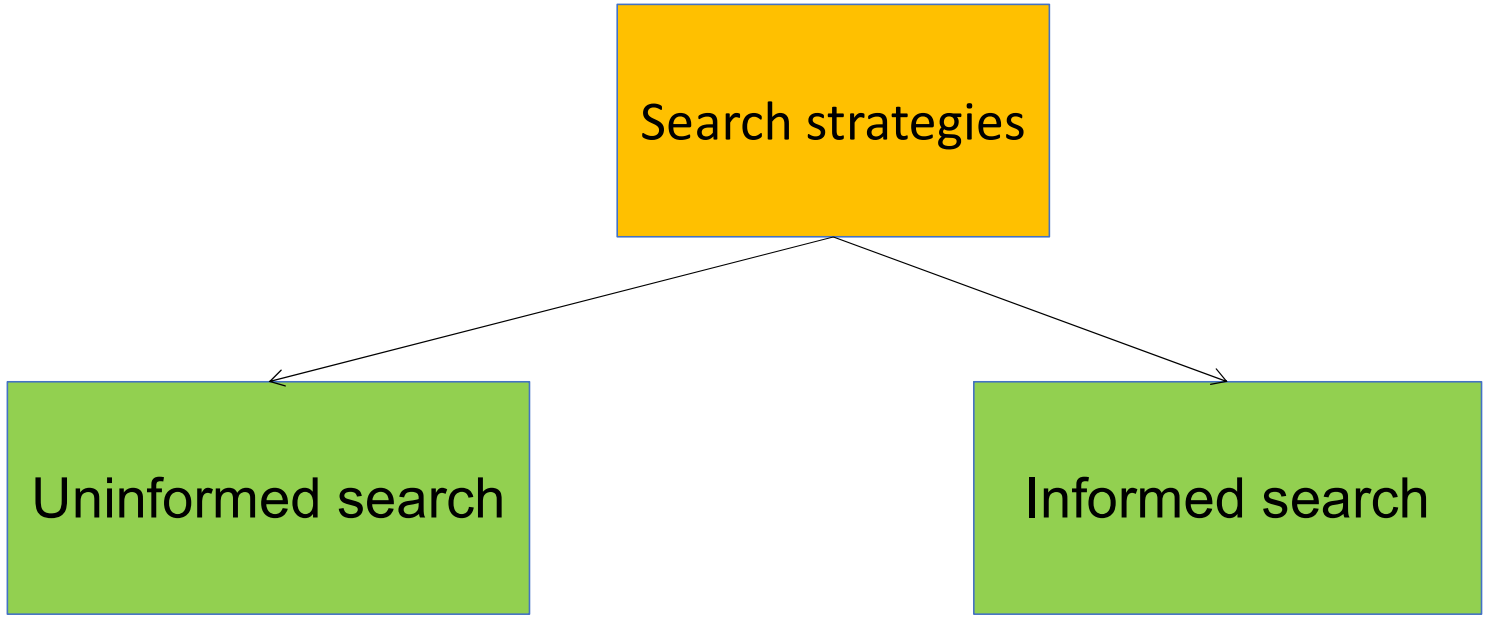
shop 16	15	14	13
9	10	shop 11	12
8	7	6	5
1Home	2	3	4

The women needs to go to shop from her home .Formalize the shop search problem defining all the 5 components

states:	all the 16 locations (boxes)
start_state:	home
actions:	up left down right
end state:	either of the shops
cost:	1 per move

MACHINE INTELLIGENCE

Types of search strategies



THANK YOU

K.S.Srinivas

srinivasks@pes.edu

+91 80 2672 1983 Extn 701

MACHINE INTELLIGENCE

UNINFORMED SEARCH STRATEGIES

K.S.Srinivas
Department of Computer Science and Engineering

MACHINE INTELLIGENCE

UNINFORMED SEARCH STRATEGIES

Srinivas K S.
Associate Professor, Department of Computer Science

MACHINE INTELLIGENCE

Uniformed Search Strategy

- Uninformed search strategies use only the information available in the problem definition
- The term means that the strategies have no additional information about states beyond that provided in the problem definition.
- All they can do is generate successors and distinguish a goal state from a non-goal state. All search strategies are distinguished by the order in which nodes are expanded
- We will be discussing 5 types of uninformed search
- Breadth-first search
- Uniform-cost search
- Depth-first search
- Depth-limited search
- Iterative deepening search

MACHINE INTELLIGENCE

Breadth-First Search

- Breadth-first search is a simple strategy in which the root node is expanded first, then all the successors of the root node are expanded next, then their successors, and so on.
- In general, all the nodes are expanded at a given depth in the search tree before any nodes at the next level are expanded.
- This is achieved very simply by using a FIFO queue for the frontier.
- let us see the pseudo code and simultaneously explore a problem

MACHINE INTELLIGENCE

Problem +algo

function BREADTH-FIRST-SEARCH(problem) **returns** a solution, or failure

node ←a node with STATE = problem.INITIAL-STATE,PATH-COST =0

if problem.GOAL-TEST(node.STATE) then **return** SOLUTION(node)

frontier ←a FIFO queue with node as the only element

explored ←an empty set

loop do

if EMPTY?(frontier) then **return** failure

node ←POP(frontier) /* chooses the shallowest node in frontier */

add node.STATE to explored

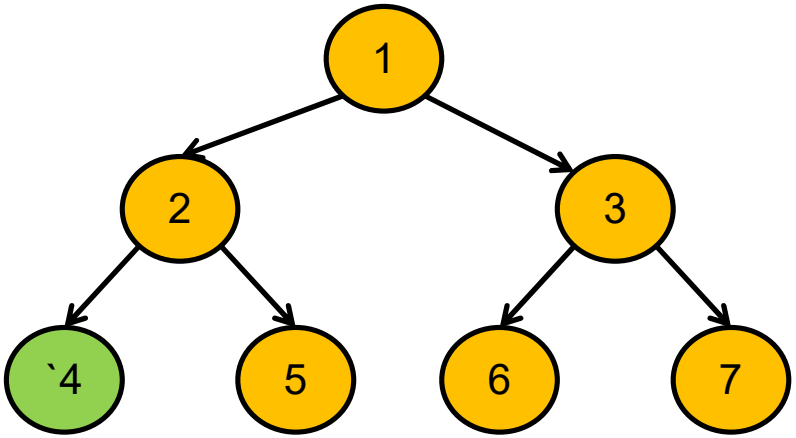
for each action in problem.ACTIONS(node.STATE) **do**

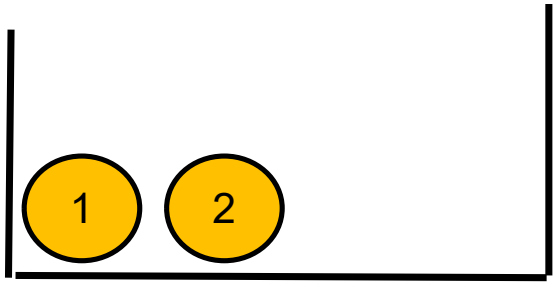
child ←CHILD-NODE(problem,node,action)

if child.STATE is not in explored or frontier then

if problem.GOAL-TEST(child.STATE) then **return** SOLUTION(child)

frontier ←INSERT(child,frontier)





MACHINE INTELLIGENCE

How good is it???

COMPLETENESS

we can easily say its complete.If the shallowest goal node is at some finite depth d ,breadth-first search will eventually find it after generating all shallower nodes with finite branching factor b.

TIME COMPLEXITY

for a uniform tree where every state b has b successor. root would generate b nodes at first level each of which generates b more nodes.In the worst case scenario for a tree of depth d the number of nodes generated would be $b+b^2+b^3+.....+b^d=O(b^d)$

SPACE COMPLEXITY

For breadth-first graph every node generated remains in memory. There will be $O(b^{d-1})$ in the explored set and $O(b^d)$ in the queue.

OPTIMALITY

the shallowest goal node is not necessarily the optimal one technically, breadth-first search is optimal if the path cost is a non decreasing function of the depth of the node. The most common such scenario is that all actions have the same cost.

MACHINE INTELLIGENCE

Uniform Cost Search

- when all the step cost are equal bfs is optimal because it always expands the shallowest unexploded node
- uniform-cost search is an extension bf when the cost is not uniform.
- uniform-cost search expands the node n with the lowest path cost g(n)
- This is done by storing the frontier as a priority queue ordered by g.
- lets see the pseudo code and then explore a problem

MACHINE INTELLIGENCE

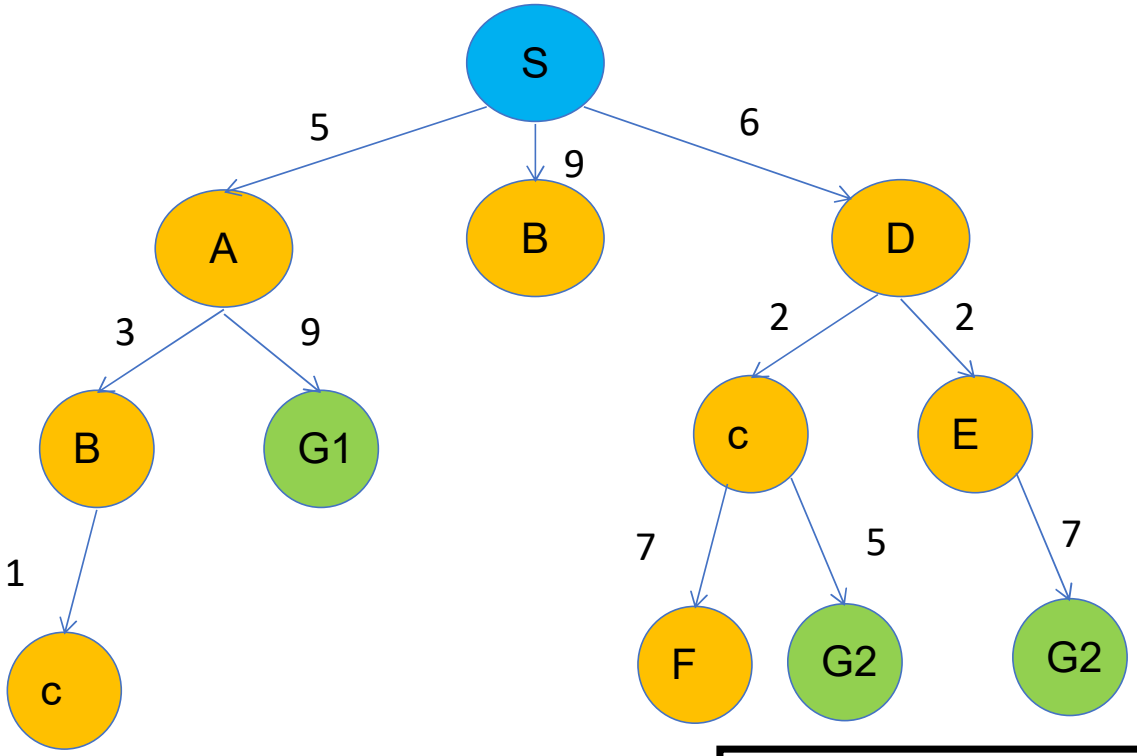
Pseudo-code

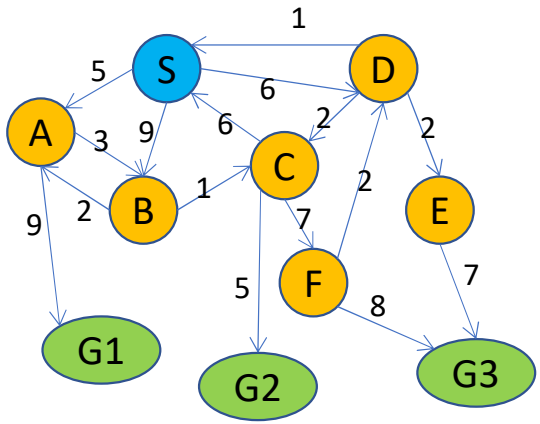
function UNIFORM-COST-SEARCH(problem) **returns** a solution, or failure
node ←a node with STATE = problem.INITIAL-STATE,PATH-COST =0
frontier ←a priority queue ordered-by PATH-COST, with node as the only element
explored ←an empty set
loop do
if EMPTY?(frontier) **then** return failure
node ←POP(frontier) /* chooses the lowest-cost node in frontier */
if problem.GOAL-TEST(node.STATE) **then return** SOLUTION(node)
add node.STATE to explored
for each action in problem.ACTIONS(node.STATE) **do**
child ←CHILD-NODE(problem,node,action)
if child.STATE is not in explored **or** frontier **then** frontier ←INSERT(child,frontier)
else if child.STATE is in frontier with higher PATH-COST then replace that frontier node with child

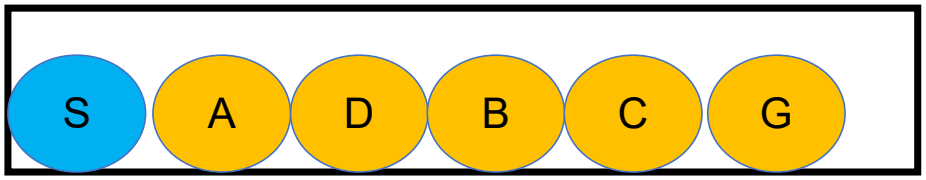
MACHINE INTELLIGENCE

Problem

[redacted] the goal from S







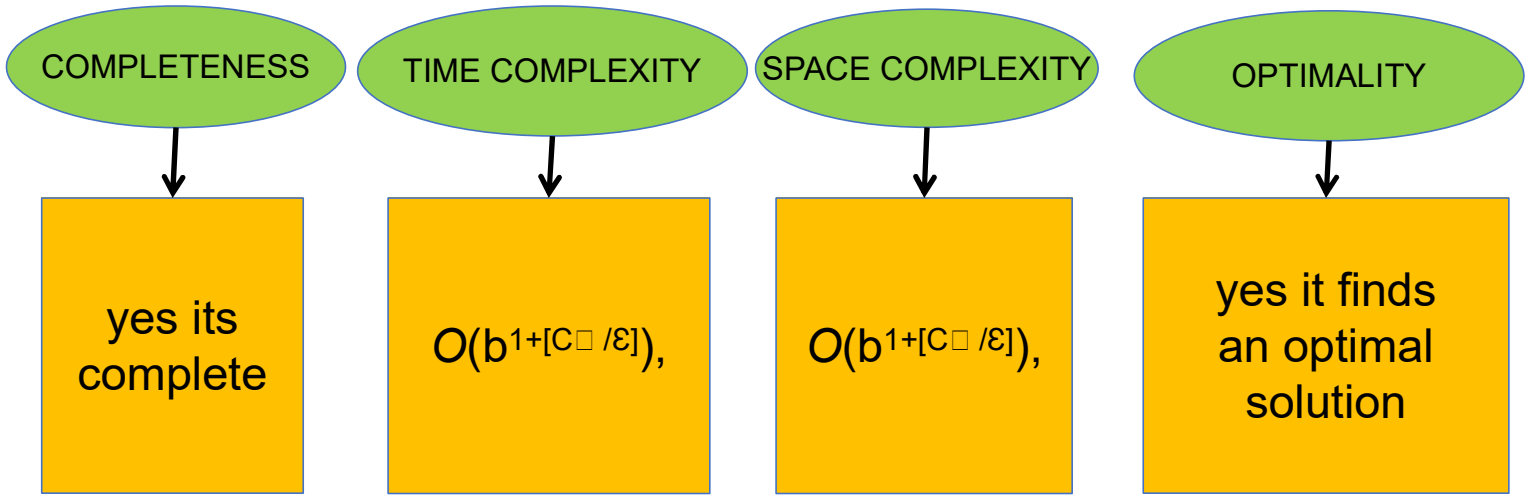
MACHINE INTELLIGENCE

Analyzing

- It is easy to see that uniform-cost search is optimal in general. First, we observe that whenever uniform-cost search selects a node n for expansion, the optimal path to that node has been found.
- Uniform-cost search does not care about the number of steps a path has, but only about their total cost. Therefore, it will get stuck in an infinite loop if there is a path with an infinite sequence of zero-cost actions. Completeness is guaranteed provided the cost of every step exceeds some small positive constant ε
- Uniform-cost search is guided by path costs rather than depths, so its complexity is not easily characterized in terms of b and d. Instead, let C^* be the cost of the optimal solution,and assume that every action costs at least ε. Then the algorithm's worst-case time and space complexity is $O(b^{1+\lfloor C^*/\varepsilon \rfloor})$, which can be much greater than b^d.

MACHINE INTELLIGENCE

How good is it??



MACHINE INTELLIGENCE

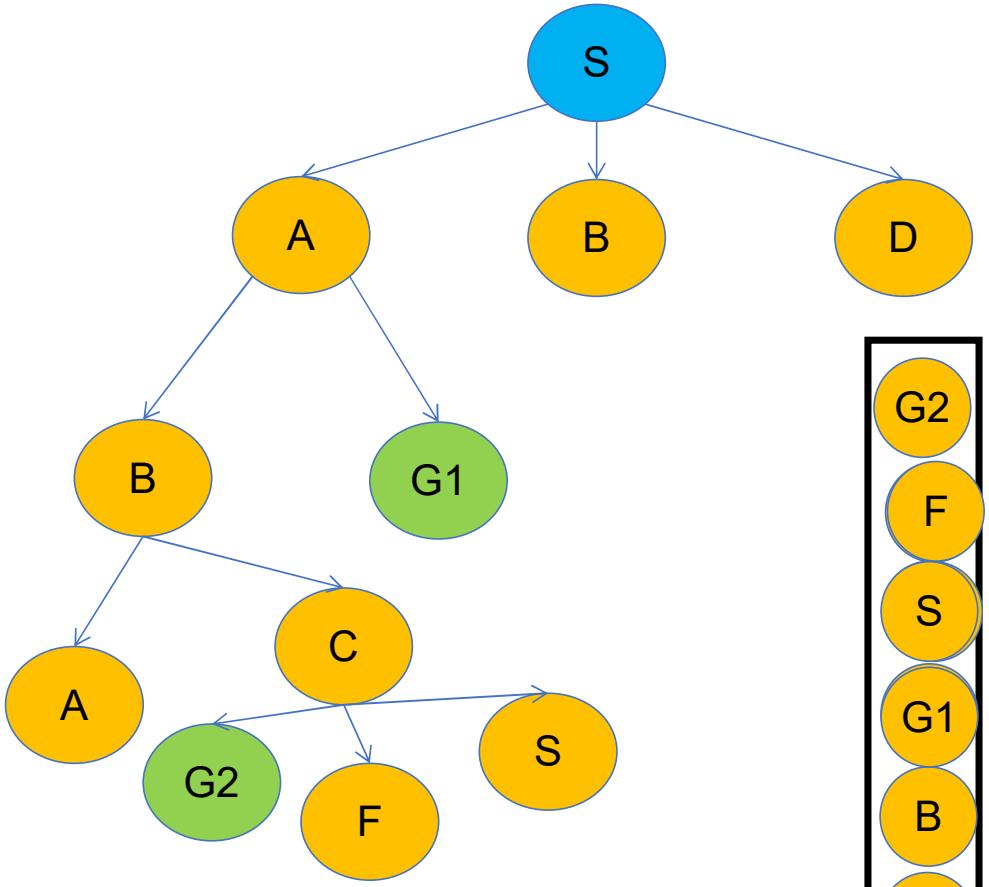
Depth-First Search

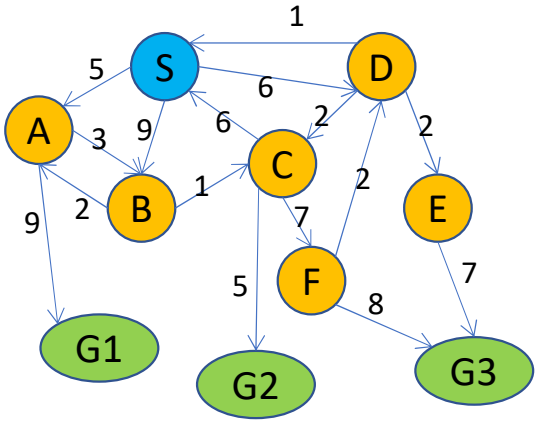
- Depth-first search always expands the deepest node in the current frontier of the search tree.
- depth-first search uses a LIFO queue.
- A LIFO queue means that the most recently generated node is chosen for expansion .
- it is common to implement depth-first search with a recursive function that calls itself on each of its children in turn.
- lets see how it actually does the job for us.

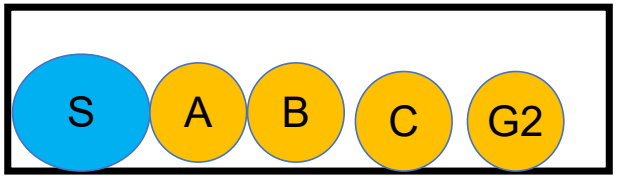
MACHINE INTELLIGENCE

Problem

[redacted] the goal from S

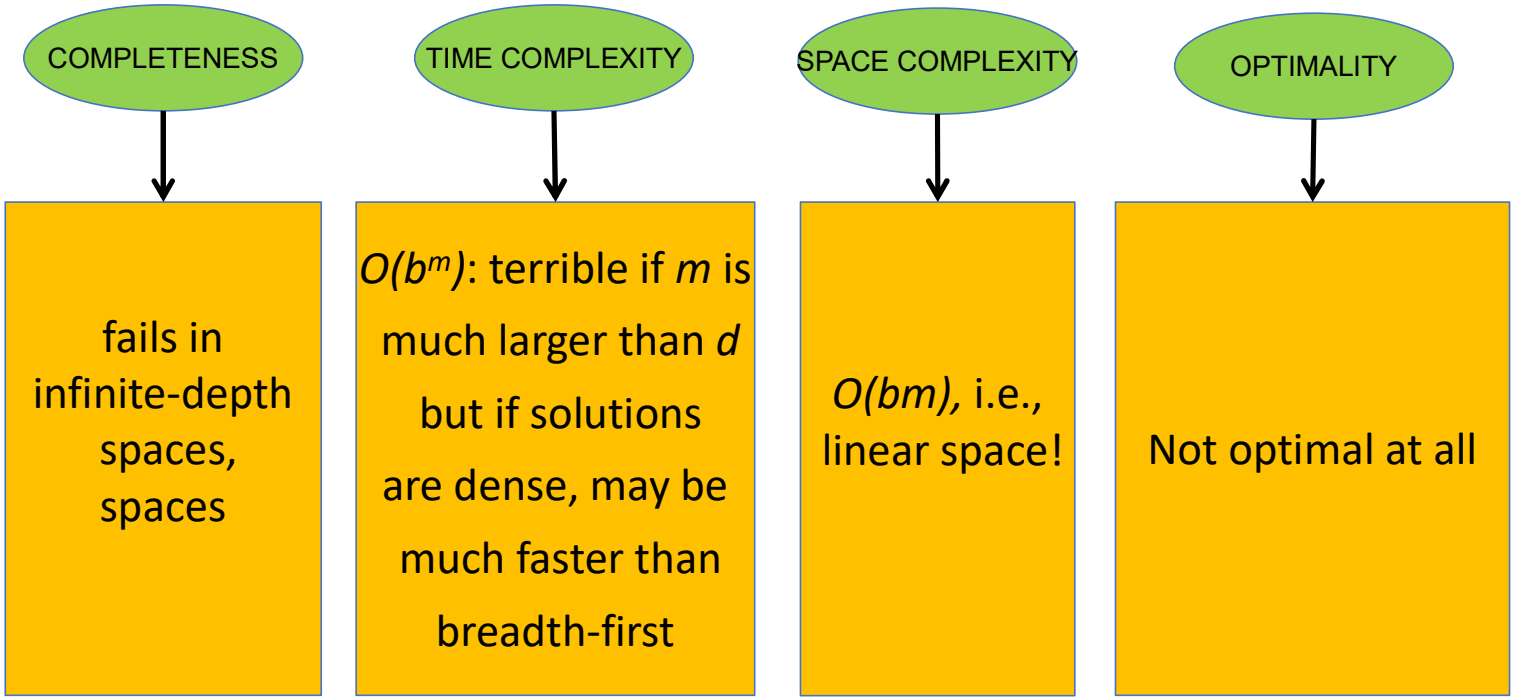






MACHINE INTELLIGENCE

How good is it???



MACHINE INTELLIGENCE

Depth-limit Search

- The embarrassing failure of depth-first search in infinite state spaces can be alleviated by supplying depth-first search with a predetermined depth limit l .
- That is, nodes at depth l are treated as if they have no successors. This approach is called depth-limited search.
- let us see the pseudo code for depth-limit search

MACHINE INTELLIGENCE

Pseudo-code


```
function DEPTH-LIMITED-SEARCH(problem,limit) returns a solution, or failure/cutoff
        return RECURSIVE-DLS(MAKE-NODE(problem.INITIAL-STATE),problem,limit)
function RECURSIVE-DLS(node,problem,limit) returns a solution, or failure/cutoff
        if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
       else if limit =0 then return cutoff
       else cutoff occurred?←false
       for each action in problem.ACTIONS(node.STATE)
                do child ←CHILD-NODE(problem,node,action)
                result ←RECURSIVE-DLS(child,problem,limit − 1)
                if result = cutoff then cutoff occurred?←true
               else if result != failure then return result
       if cutoff occurred? then return cutoff else return failure
```

MACHINE INTELLIGENCE

Iterative deepeining Search

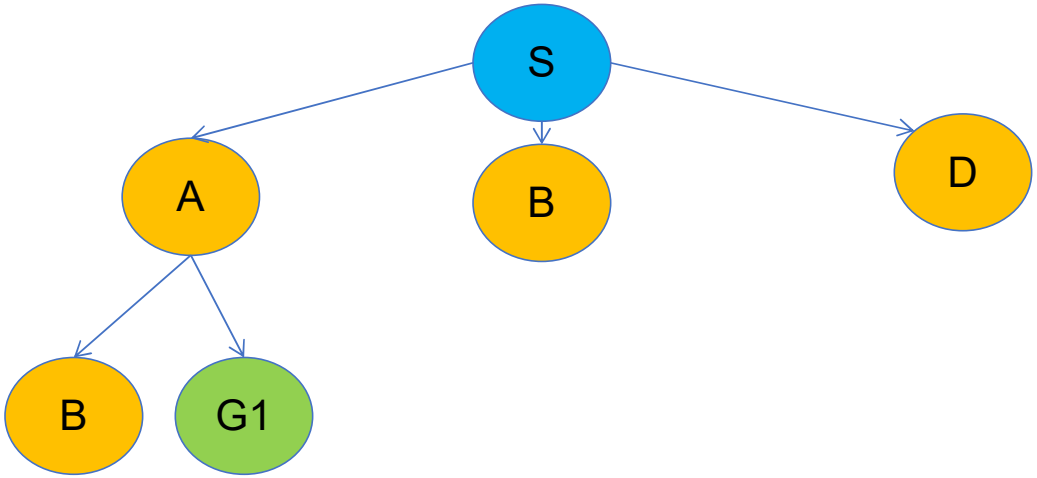
- Iterative deepening search (or iterative deepening depth-first search) is a general strategy, often used in combination with depth-first tree search, that finds the best depth limit.

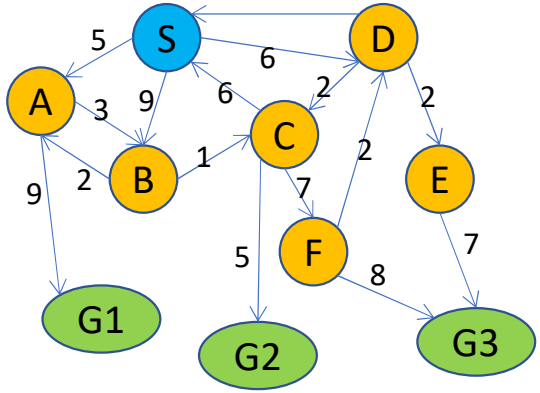
```
function ITERATIVE-DEEPENING-SEARCH( problem) returns a solution, or fail-
ure
    inputs: problem, a problem
    for depth ← 0 to ∞ do
        result ← DEPTH-LIMITED-SEARCH( problem, depth)
        if result ≠ cutoff then return result
```

MACHINE INTELLIGENCE

Problem

[redacted] from S





limit l =2

Calculations

Number of nodes generated in a depth-limited search to depth *d* with branching factor *b*:

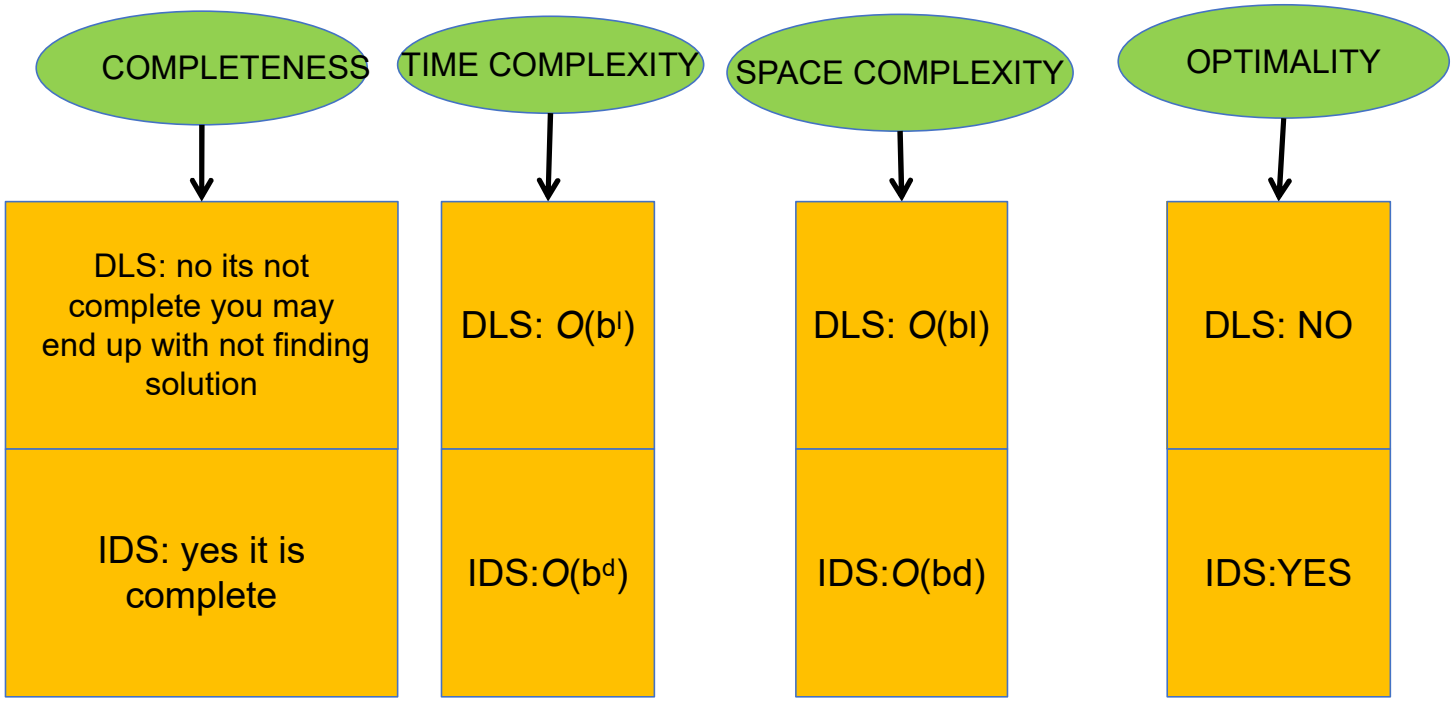
$$N_{DLS} = b^0 + b^1 + b^2 + \ldots + b^{d-2} + b^{d-1} + b^d$$

Number of nodes generated in an iterative deepening search to depth *d* with branching factor *b*:

$$N_{IDS} = (d+1)b^0 + d\ b\text{^}1 + (d-1)b\text{^}2 + \ldots + 3b^{d-2} + 2b^{d-1} + 1b^d$$

MACHINE INTELLIGENCE

How good is it??

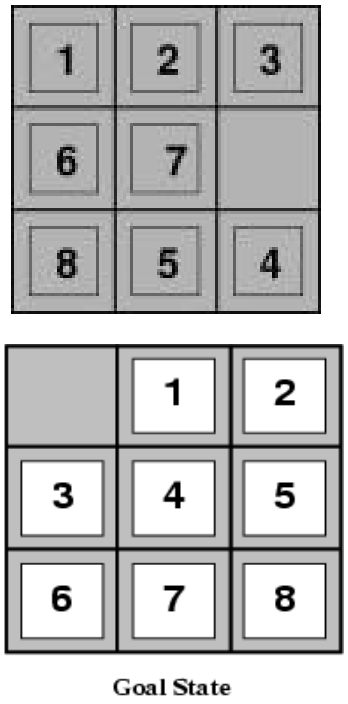


MACHINE INTELLIGENCE

Applications

- Now let us look at some of the real world example that our search algorithms include.

8-puzzle



Goal State

Can you try to formalize this search problem by defining the 5 components we discussed earlier?

STATES	Locations of tiles
START STATE	Given position of tiles
GOAL STATE	GIVEN
ACTIONS	move blank left, right, up, down
COST	1 per move

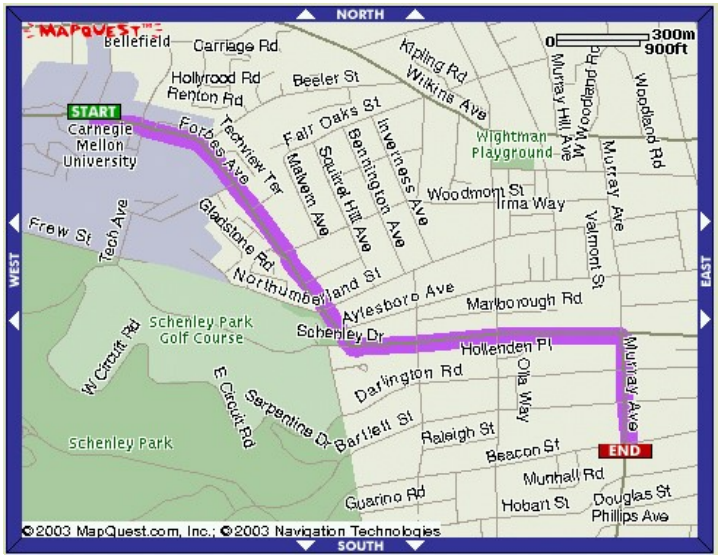
- MAZE PROBLEM

can you guess what kind of search algorithm it may be using? DFS

MACHINE INTELLIGENCE

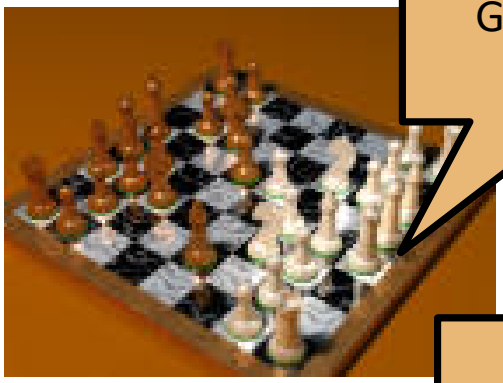
Applications

some more applications

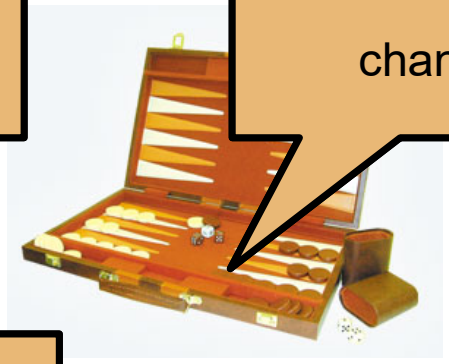


MACHINE INTELLIGENCE

Our definition excludes

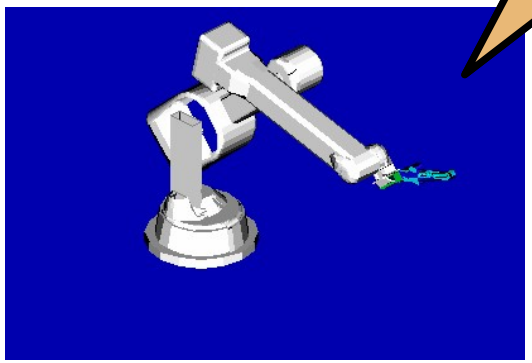


Game against adversary



chance

Continuum (infinite number) of states



All of the above, plus distributed team control

THANK YOU

K.S.Srinivas

srinivasks@pes.edu

+91 80 2672 1983 Extn 701

MACHINE INTELLIGENCE

INFORMED SEARCH STARTEGIES

K.S.Srinivas
Department of Computer Science and Engineering

MACHINE INTELLIGENCE

INFORMED SEARCH STARTEGIES

Srinivas K S.
Associate Professor, Department of Computer Science

Informed Search

- Informed Search is the one that uses problem-specific knowledge beyond the definition of the problem itself.
- example-for a path-search in a graph with a geometrical representation-give preference to the neighbour which are closest to the target based on the Eucledian distance
- can find solutions more efficiently than an uninformed strategy.
- The general approach we consider is called best-first search
- best-first search is an algorithm in which the node is selected based on the evaluation function ***f*(n).**
- the evaluation function **f(n)** is a function that takes a node as a input and gives out a positive number similar to the g(n) function we used in UCS.
- so the node with the lowest evaluation is expanded first.

ARE WE DOING UNIFORM COST SEARCH AGAIN

The implementation of best-first graph search is identical to that for uniform-cost search. except for the use of f instead of g to order the priority queue. The choice of f determines the search strategy. .

Tip touch on Heuristic Function

Most best-first algorithms include as a component of f a heuristic function, denoted h(n):

h(n)= estimated cost of the cheapest path from the state at node n to a goal state

Heuristic functions are the most common form in which additional knowledge of the problem is imparted to the search algorithm.

basically Heuristic is what you call as अनुमान

for now

Lets now convince ourselves for the below mentioned about HEURISTIC FUNCTION

it is a arbitrary, non negative, problem-specific functions, with one constraint: if n is a goal node, then *h*(n)=0.

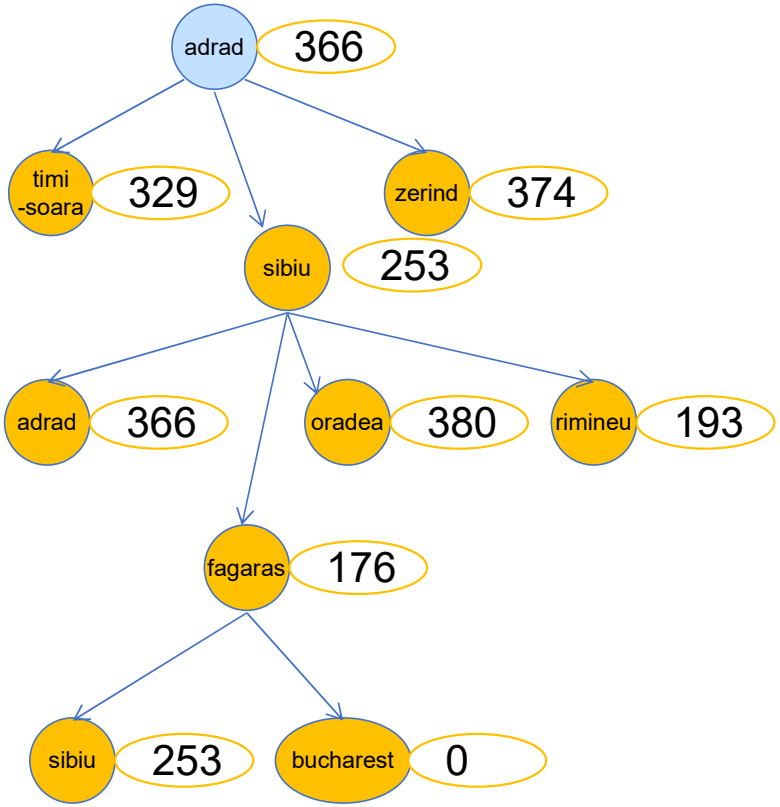
Greedy Best First Search

- Greedy best-first search tries to expand the node that is closest to the goal, on the grounds that this is likely to lead to a solution quickly. Thus, it evaluates nodes by using just the heuristic function; that is, f(n)=h(n).
- Let us analyse the PSEUDO CODE

```
Init-PriQueue(PQ)
Insert-PriQueue(PQ,START,h(START))
while (PQ is not empty and PQ does not contain a goal state)
   (s , h ) := Pop-least(PQ)
   foreach s' in succs(s)
   if s' is not already in PQ and s' never previously been visited
      Insert-PriQueue(PQ,s',h(s'))
```

- Let us now directly travel to Romania and solve some problem for better understanding

Adrad-->>>Bucharest



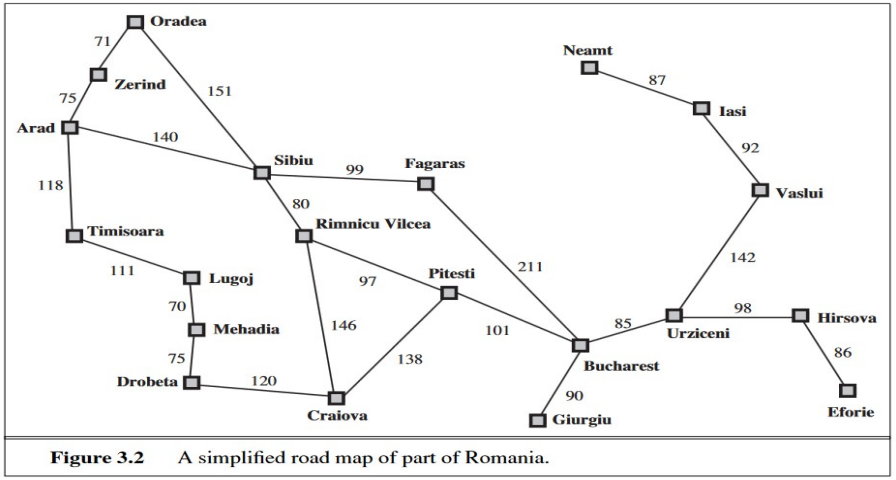


Figure 3.2 A simplified road map of part of Romania.

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Lets Analyse

- For this particular problem, greedy best-first search using hSLD finds a solution without ever expanding a node that is not on the solution path; hence, its search cost is minimal.
- It is not optimal, however: the path via Sibiu and Fagaras to Bucharest is 32 kilometers longer than the path through Rimnicu Vilcea and Pitesti. This shows why the algorithm is called “greedy”—at each step it tries to get as close to the goal as it can.
- Greedy best-first tree search is also incomplete even in a finite state space, much like depth-first search.

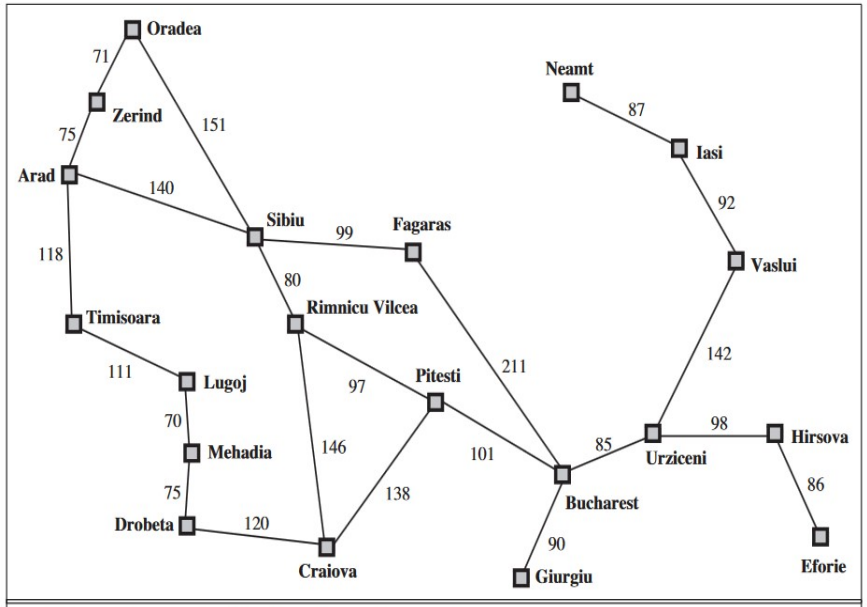


Figure 3.2 A simplified road map of part of Romania.

Lets Analyse

- Consider the problem of getting from Iasi to Fagaras
- The heuristic suggests that Neamt be expanded first because it is closest to Fagaras, but it is a dead end
- The solution is to go first to Vaslui—a step that is actually farther from the goal according to the heuristic—and then to continue to Urziceni, Bucharest, and Fagaras.
- The algorithm will never find this solution, however, because expanding Neamt puts Iasi back into the frontier, Iasi is closer to Fagaras than Vaslui is, and so Iasi will be expanded again, leading to an infinite loop.
- The worst-case time and space complexity for the tree version is O(bm),wherem is the maximum depth of the search space. With a good heuristic function, however, the complexity can be reduced substantially. The amount of the reduction depends on the particular problem and on the quality of the heuristic

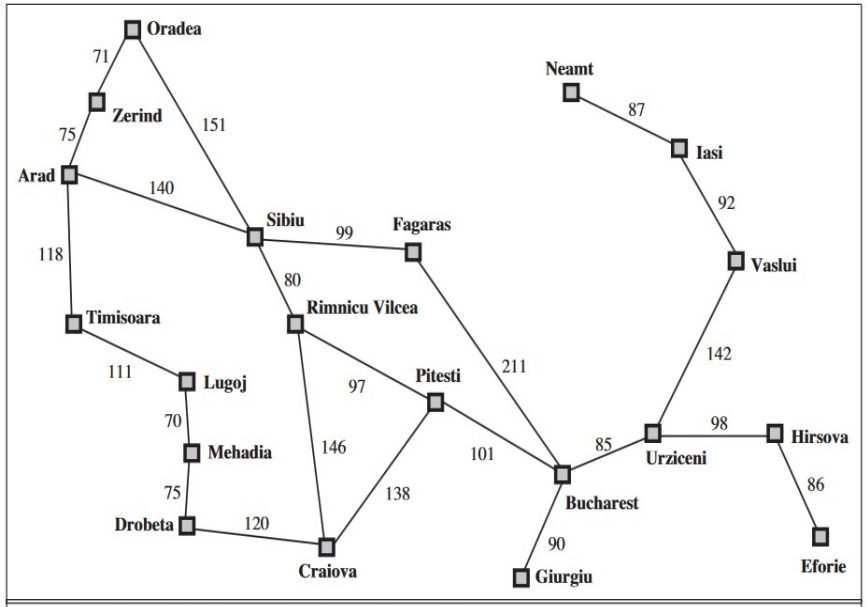
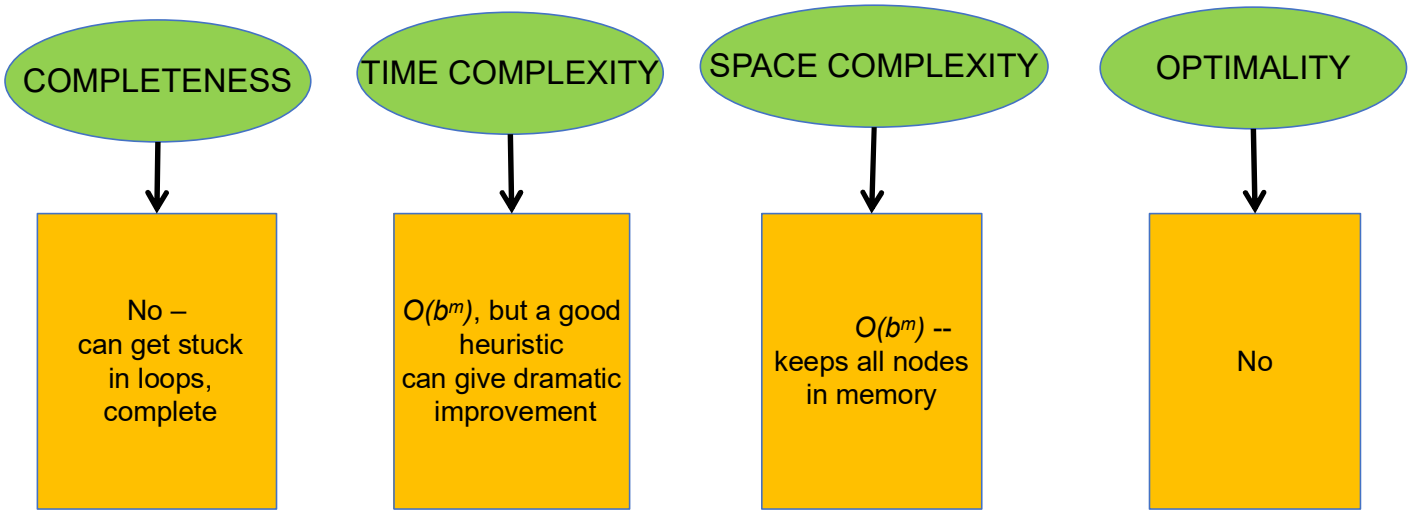


Figure 3.2 A simplified road map of part of Romania.

How good is it???



A* Search

- The most widely known form of best-first search is called A* search
- The most important point to notice is the function f(n)

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

- 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)= estimated cost of the cheapest solution through n.

- The algorithm is identical to **UNIFORM-COST-SEARCH** except that A* uses g + h instead of g.
- 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.

Conditions for optimality

Admissibility

- The first condition we require for optimality is that h(n) be an admissible heuristic.
- An admissible heuristic is one that never overestimates the cost to reach the goal.
- Because g(n) is the actual cost to reach n along the current path, and f(n)=g(n)+h(n), we have as an immediate consequence that f(n) never overestimates the true cost of a solution along the current path through n.
- Admissible heuristics are by nature optimistic because they think the cost of solving the problem is less than it actually is.

EXAMPLE:
straight-line distance h_{SLD} Straight-line distance is admissible because the shortest path between any two points is a straight line, so the straight line cannot be an overestimate.

Conditions for optimality

Consistency

- A heuristic h(n) is consistent if, for every node n and every successor n' of n generated by any action a, the estimated cost of reaching the goal from n is no greater than the step cost of getting to n' plus the estimated cost of reaching the goal from n'

- consistent heuristic is also admissible

- Consistency is therefore a stricter requirement than admissibility,

h(n) <= c(n,a,n')+h(n')
this is form of triangle inequality, which stipulates that each side of a triangle cannot be longer than the sum of the other two sides. Here, the triangle is formed by n, n', and the goal Gn closest to n.

Pseudo-Code

- lets analyse the pseudo code and jump to solve a prob

The A* Algorithm

Reminder: $g(n)$ is cost of shortest known path to n

Reminder: $h(n)$ is a heuristic estimate of cost to a goal from n

- Priority queue PQ begins empty.
- V (= set of previously visited ($state$,f,$backpointer$)-triples) begins empty.
- Put S into PQ and V with priority $f(s) = g(s) + h(s)$
- Is PQ empty?
  - Yes? Sadly admit there's no solution
  - No? Remove node with lowest $f(n)$ from queue. Call it n.
  - If n is a goal, stop and report success.
  - "expand" n : For each n' in **successors**(n)....
    - Let $f' = g(n') + h(n') = g(n) + cost(n,n') + h(n')$
    - **If** n' not seen before, or n' previously expanded with $f(n') > f'$, or n' currently in PQ with $f(n') > f'$
    - **Then** Place/promote n' on priority queue with priority f' and update V to include ($state$=n', f', $BackPtr$=n).
    - **Else** Ignore n'

= $h(s)$ because $g(\text{start}) = 0$

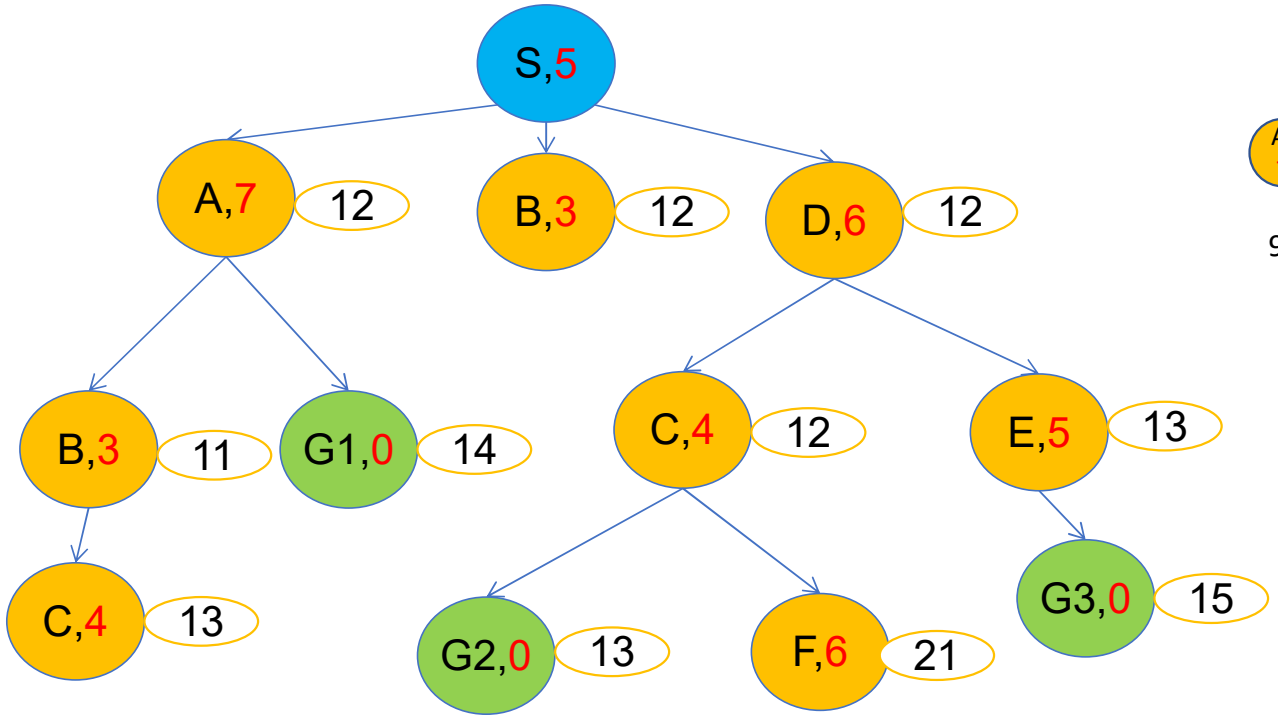
use sneaky trick to compute $g(n)$

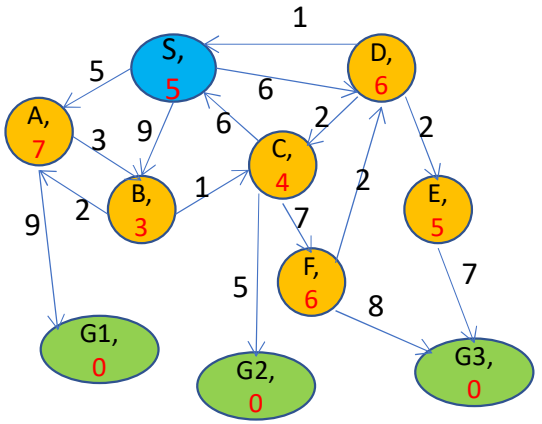
Problem

now we add A to visited along wit its A* score

Fo[illegible] goal from S

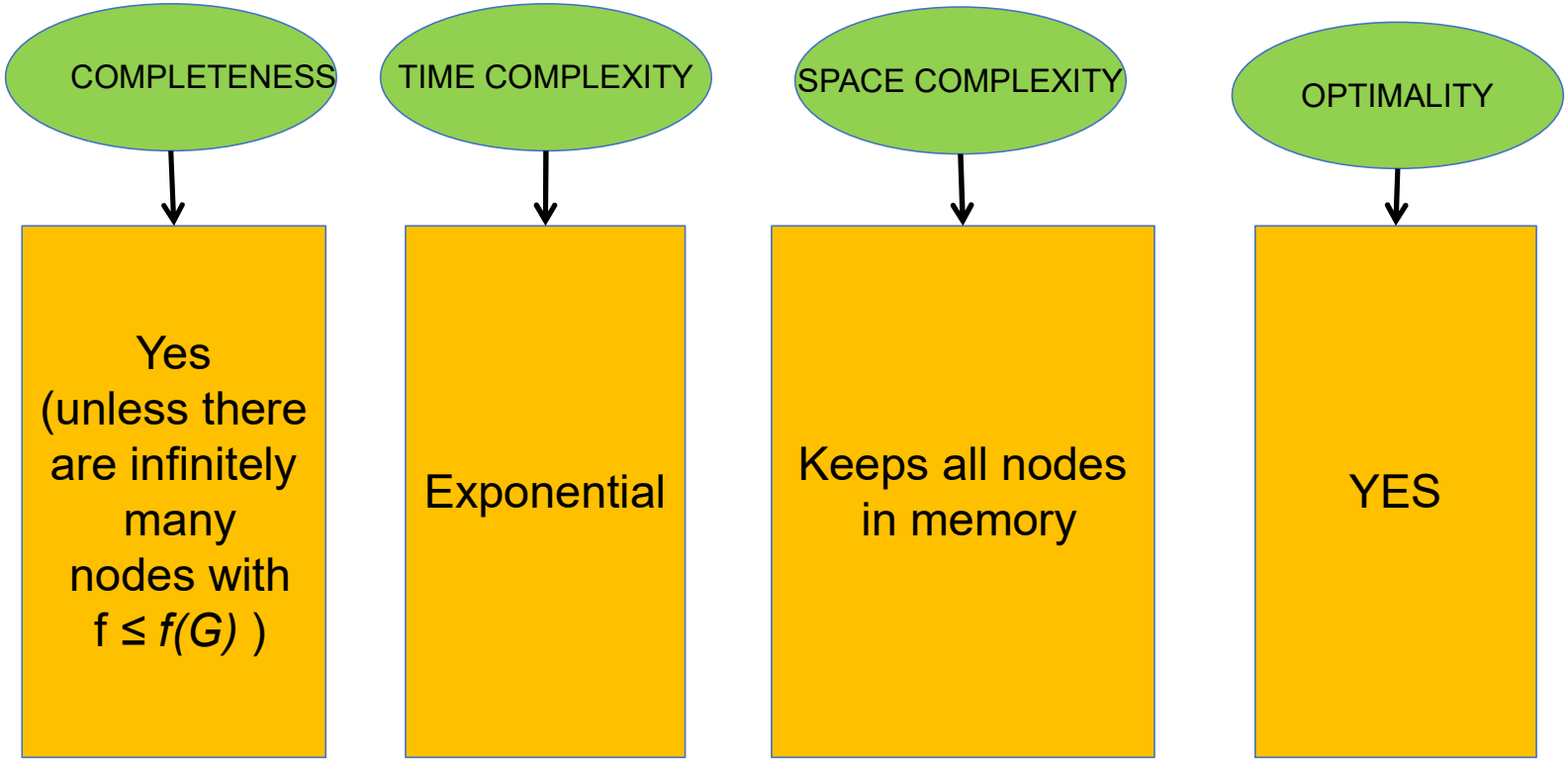
[illegible] as visited





S,0,5	A,7,12	B,3,11	D,6,12	C,4,12	E,5,13

How good is it??



Heuristic is the controller of A*

- The selection of the heuristic function is crucial in determining the efficiency of A*.
- Using h≡ 0 assures admissibility but results in a uniform-cost search and is thus usually inefficient.
- Setting h equal to the highest possible lower bound on h expands the fewest nodes consistent with maintaining admissibility.
- In the Eight-puzzle, for example, the function h(n)=W(n) (where W (n) is the number of tiles in the wrong place) is a lower bound on h (n), but it does not provide a very good estimate of the difficulty (in terms of number of steps to the goal) of a tile configuration. A better estimate is the function h(n)=P(n), where P(n) is the sum of the (**Manhattan**) distances that each tile is from "home" (ignoring intervening pieces).
- In selecting an h function, we must take into consideration the amount of effort involved in calculating it.
- There is usually a trade off between the benefits gained by an accurate h function and the cost of computing it.
- Another possibility is to modify the relative weights of g and h in the evaluation function . That is, we use f=g+w.h , where w is a positive number. Very large values of w overemphasize the heuristic component, whereas very small values of w give the search a predominantly breadth-first character.

Heuristic is the controller of A*

- A* ability to vary its behavior based on the heuristic and cost functions can be very useful.
- For most games, you don't really need the best path between two points. You need something that's close. What you need may depend on what's going on in the game, or how fast the computer is. Using a function that guarantees it never overestimates the cost means that it will sometimes underestimate the cost by quite a bit
- Suppose your game has two types of terrain, Flat and Mountain, and the movement costs are 1 for flat land and 3 for mountains, A* is going to search three times as far along flat land as it does along mountainous land. This is because it's possible that there is a path along flat terrain that goes around the mountains. You can speed up A*'s search by using 1.5 as the heuristic distance between two map spaces. A* will then compare 3 to 1.5, and it won't look as bad as comparing 3 to 1. It is not as dissatisfied with mountainous terrain, so it won't spend as much time trying to find a way around it.

Choice of a Heuristic

- On a grid, there are well-known heuristic functions to use.
- On a square grid that allows 4 directions of movement, use Manhattan distance (L1).
- On a square grid that allows 8 directions of movement, use Diagonal distance (L∞).
- On a square grid that allows any direction of movement, you might or might not want Euclidean distance (L2). If A* is finding paths on the grid but you are allowing movement not on the grid, you may want to consider other representations of the map.
- On a hexagon grid that allows 6 directions of movement, use Manhattan distance adapted to hexagonal grids.
- If you want to search for any of several goals, construct a heuristic h'(x) that is the minimum of h1(x), h2(x), h3(x), ... where h1, h2, h3 are heuristics to each of the nearby spots.

THANK YOU

K.S.Srinivas

srinivasks@pes.edu

+91 80 2672 1983 Extn 701

MACHINE INTELLIGENCE

Concept Learning

K.S.Srinivas
Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Concept Learning

Srinivas K S.
Associate Professor, Department of Computer Science

MACHINE INTELLIGENCE

Concept Learning

- what is a concept?
- we will try to understand it practically

does not belong to the concept

Concept	Data object	
C	x	
		label
x- belongs to concept c		1
or		
not belongs to concept c		0

CONCEPT?
YES

MACHINE INTELLIGENCE

Concept Learning

- what is a concept?
- we will try to understand it practically

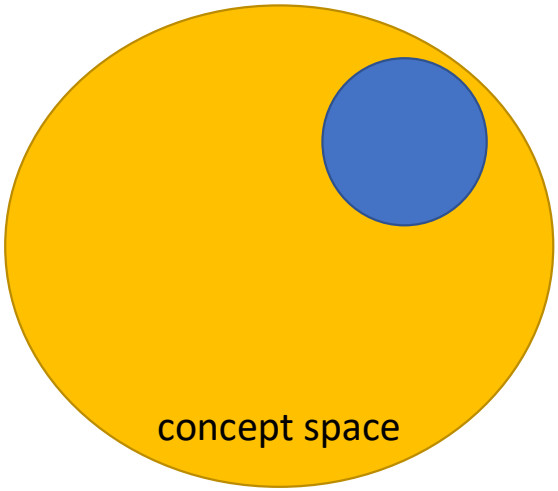
each object x is is defined by specific features
over job is to make machine learn this unknown
through which we can decide if it belongs to the
concept (shape, size, colour attributes (&
and once we know it we can label any new data
accordingly.....this the concept learning task

Concept	Data object	
C	x	
		label
X = {	belongs to concept c (oval,large,dark),.....(circular,small,light) or not belongs to concept c	1 0

shape-oval,circular
size-large,small
colour-dark,light

MACHINE INTELLIGENCE

Concept Learning

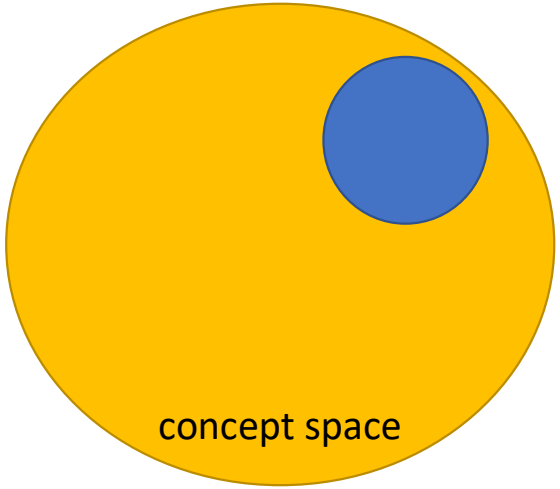


2^8 concepts.....how???

suppose our data is defined using d binary attributes then there are 2^d let us say it as b ,then we have 2^b concepts

MACHINE INTELLIGENCE

Inductive bias



X_1 -> (1,0,1)
X_2 -> (0,0,1)

circular ^ dark

the reason we call this concept
is only if both of this is true for a data object
then the object belongs to the concept

we will see what it is

(logical AND) of few of these
basically a subset of all possible feature

0	1
oval	circular
small	large
light	dark

Inductive bias

__ ^ __ ^ __

circular ^ dark can be represented like this

circular ^ _?_ ^ dark

? means anything,that is any value there is true

since we have three attributed here we can mention our conjunctive concept something like this

? ^ ? ^ ? is a concept that accepts everything ,we call it accept all

Ø is a concept that rejects everything ,we call it reject all

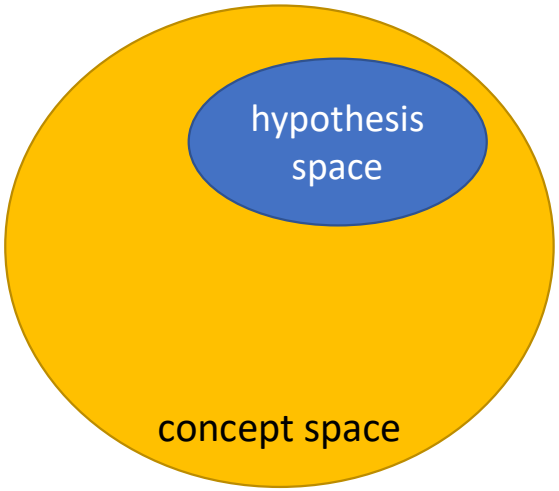
MACHINE INTELLIGENCE

Inductive bias

for every position we can have three value that is two among the binary value and one **?**
also we have one concept of **Ø**
so we have (3 x 3 x3)+1=28 conjunctive concepts

In general if we have 'n' features then we will have 3n+1 conjunctive concepts for a object with 3 binary attributes

if u carefully observe we had total of 256 concepts in our concept space but now we have only 28 concepts in our conjunctive concept space
i.e we have shrunk our search space
this shrunk space is what we call as hypothesis space



MACHINE INTELLIGENCE
Find S algorithm

Concept- Days on which person enjoys sports

Attributes-
Sky-sunny,rainy
Temp-warm,cold
Humidity-Normal,High
Wind- strong,weak
Water-warm,cool
Forecast-same,change

with this description we will move forward to our first algorithm that is Find S algorithm

MACHINE INTELLIGENCE

Find S algorithm

1. start with h=Ø
2. use next input {x,c(x)}
3. if c(x)=0 ,go to step 2
4. h <--- h ^ x (pairwise-and)
5. if more examples .Go to step 2
6. stop

Let us look at the pseudo code of this algorithm

Pairwise -and rules

$$a_h \wedge a_x = \begin{cases} a_x & : \text{if } a_h = \emptyset \\ a_x & : \text{if } a_h = a_h \\ ? & : \text{if } a_h \neq a_x \\ ? & : \text{if } a_h = ? \end{cases}$$

MACHINE INTELLIGENCE

Find S algorithm-Problem

step1: start with h=Ø

h={Ø,Ø,Ø,Ø,Ø,Ø}

step2: use next input {x,c(x)}

x={sunny,warm,normal,strong,warm,same}
c(x)=yes==1

step3: if c(x)=0 ,go to step 2
step4: h <--- h ^ x (pairwise-and)

$h_0=h_0$^ x
h_0={sunny,warm,normal,strong,warm,same}

step5: if more examples .Go to step2

h(0)=sunny
h(1)=warm
h(2)=normal
h(3)=strong
h(4)=warm
h(5)=same

we will take a concept step by step and rule we need to learn this concept with the training data below

sky	temp	humidity	wind	water	forecast	enjoy
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	same	yes

$$a_h \wedge a_x = \begin{cases} a_x & : \text{if } a_h = \emptyset \\ a_x & : \text{if } a_h = a_h \\ ? & : \text{if } a_h \neq a_x \\ ? & : \text{if } a_h = ? \end{cases}$$

MACHINE INTELLIGENCE

Find S algorithm-Problem

step2: next input {x,c(x)}

x={sunny,warm,high,strong,warm,same}
c(x)=yes==1

step3: if c(x)=0 ,go to step 2
step4: h <--- h ^ x (pairwise-and)

$h_0=h_0$^ x
h_0={sunny,warm,?,strong,warm,same}

step5: if more examples Go to step 2

h(0)=sunny
h(1)=warm
h(2)=?
h(3)=strong
h(4)=warm
h(5)=same

sky	temp	humidity	wind	water	forecast	enjoy
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	same	yes

$$a_h \wedge a_x = \begin{cases} a_x & : \text{if } a_h = \emptyset \\ a_x & : \text{if } a_h = a_h \\ ? & : \text{if } a_h \neq a_x \\ ? & : \text{if } a_h = ? \end{cases}$$

MACHINE INTELLIGENCE

Find S algorithm-Problem

step2: next input {x,c(x)}

x={rainy,cold,high,strong,warm,change}
c(x)=yes==0

step3: if c(x)=0 ,go to step 2

sky	temp	humidity	wind	water	forecast	enjoy
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	same	yes

$$a_h \wedge a_x = \begin{cases} a_x & : \text{if } a_h = \emptyset \\ a_x & : \text{if } a_h = a_h \\ ? & : \text{if } a_h \neq a_x \\ ? & : \text{if } a_h = ? \end{cases}$$

MACHINE INTELLIGENCE

Find S algorithm-Problem

step2: next input {x,c(x)}

x={sunny,warm,high,strong,cool,same}
c(x)=yes==1

step3: if c(x)=0 ,go to step 2
step4: h <--- h ^ x (pairwise-and)

$h_0=h_0$^ x
h_0={sunny,warm,?,strong,?,same}

step5: if more examples Go to step 2
step6: stop

h(0)=sunny
h(1)=warm
h(2)=?
h(3)=strong
h(4)=?
h(5)=same

sky	temp	humidity	wind	water	forecast	enjoy
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	same	yes

$$a_h \wedge a_x = \begin{cases} a_x & : \text{if } a_h = \emptyset \\ a_x & : \text{if } a_h = a_h \\ ? & : \text{if } a_h \neq a_x \\ ? & : \text{if } a_h = ? \end{cases}$$

MACHINE INTELLIGENCE

Find S algorithm-prediction

C={sunny,warm,?,strong,?,same}

x={sunny,warm,high,strong,warm,same}

c(x)=1

suppose now we give the test data set and we can pass the data through our concept and we give our final hypothesis as our concept that we use the concept learnt to predict the labels that is if the person will play a sport or no

MACHINE INTELLIGENCE

Version space

a hypothesis is said to be consistent wrt to training data set if it classifies all the object of training data set to their corresponding classes

X	C(X)
x_1	$c(x_1)$
x_2	$c(x_2)$

so on.....

a hypothesis is said to be consistent if $h(x_i)=C(x_i)$

at any given point let H be our hypothesis space
so a version space is a subset of H
but all concepts in version space VS are consistent wrt to training set

i.e VS={h : h $\in$ **H** and h is consistent with $D_{training}$

this ensures our algorithm learns only the best hypothesis

[illegible] the hypothesis space may never find the right concept that is to be learned and always predict wrong.

THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701

MACHINE INTELLIGENCE

Concept Learning

K.S.Srinivas
Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Concept Learning

Srinivas K S.
Associate Professor, Department of Computer Science

MACHINE INTELLIGENCE

Concept Learning

- what is a concept?
- we will try to understand it practically

does not belong to the concept

Concept	Data object	
C	x	
		label
x- belongs to concept c		1
or		
not belongs to concept c		0

CONCEPT?
YES

MACHINE INTELLIGENCE

Concept Learning

- what is a concept?
- we will try to understand it practically

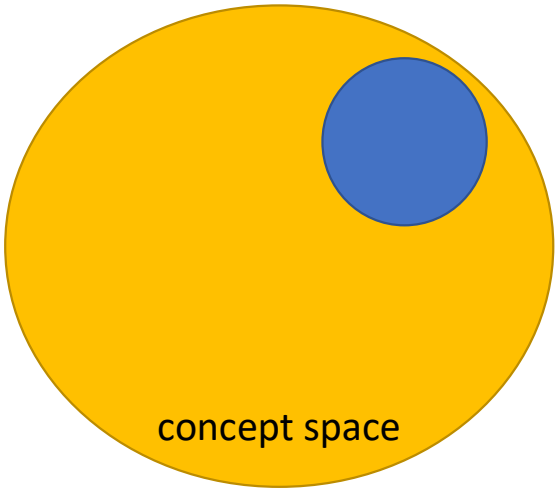
each object x is is defined by specific features
over job is to make machine learn this unknown
through which we can decide if it belongs to the
concept (shape, size, colour attributes (&
and once we know it we can label any new data
accordingly.....this the concept learning task

Concept	Data object	
C	x	
		label
X = {	belongs to concept c (oval,large,dark),.....(circular,small,light) or not belongs to concept c	1 0

shape-oval,circular
size-large,small
colour-dark,light

MACHINE INTELLIGENCE

Concept Learning

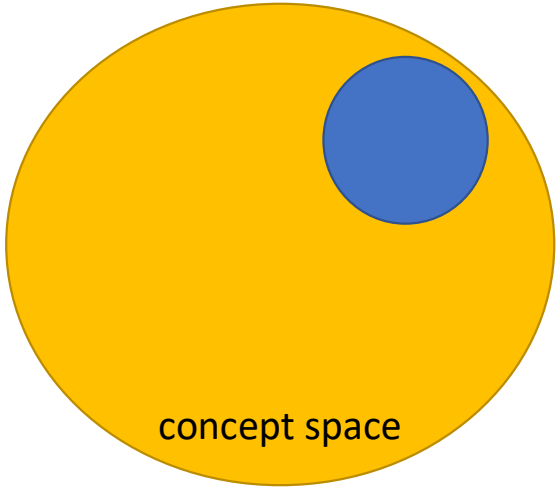


2^8 concepts.....how???

suppose our data is defined using d binary attributes then there are 2^d let us say it as b ,then we have 2^b concepts

MACHINE INTELLIGENCE

Inductive bias



X_1 -> (1,0,1)
X_2 -> (0,0,1)

circular ^ dark

the reason we call this concept
is only if both of this is true for a data object
then the object belongs to the concept

we will see what it is

(logical AND) of few of these
basically a subset of all possible feature

0	1
oval	circular
small	large
light	dark

Inductive bias

__ ^ __ ^ __

circular ^ dark can be represented like this

circular ^ _?_ ^ dark

? means anything,that is any value there is true

since we have three attributed here we can mention our conjunctive concept something like this

? ^ ? ^ ? is a concept that accepts everything ,we call it accept all

Ø is a concept that rejects everything ,we call it reject all

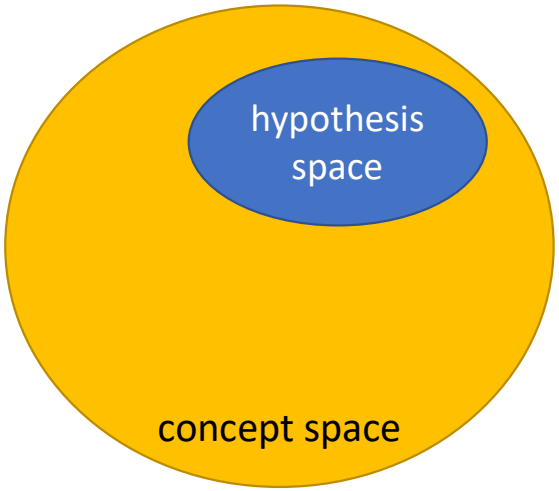
MACHINE INTELLIGENCE

Inductive bias

for every position we can have three value that is two among the binary value and one **?**
also we have one concept of **Ø**
so we have (3 x 3 x3)+1=28 conjunctive concepts

In general if we have 'n' features then we will have 3n+1 conjunctive concepts for a object with 3 binary attributes

if u carefully observe we had total of 256 concepts in our concept space but now we have only 28 concepts in our conjunctive concept space
i.e we have shrunk our search space
this shrunk space is what we call as hypothesis space



MACHINE INTELLIGENCE

Find S algorithm

<u>**Concept-**</u> Days on which person enjoys sports

<u>**Attributes-**</u>

Sky-sunny,rainy
Temp-warm,cold
Humidity-Normal,High
Wind- strong,weak
Water-warm,cool
Forecast-same,change

[illegible] that is Find S algorithm

MACHINE INTELLIGENCE

Find S algorithm

1. start with h=Ø
2. use next input {x,c(x)}
3. if c(x)=0 ,go to step 2
4. h <--- h ^ x (pairwise-and)
5. if more examples .Go to step 2
6. stop

Let us look at the pseudo code of this algorithm

Pairwise -and rules

$$a_h \wedge a_x = \begin{cases} a_x & : \text{if } a_h = \emptyset \\ a_x & : \text{if } a_h = a_h \\ ? & : \text{if } a_h \neq a_x \\ ? & : \text{if } a_h = ? \end{cases}$$

MACHINE INTELLIGENCE

Find S algorithm-Problem

step1: start with h=Ø

h={Ø,Ø,Ø,Ø,Ø,Ø}

step2: use next input {x,c(x)}

x={sunny,warm,normal,strong,warm,same}
c(x)=yes==1

step3: if c(x)=0 ,go to step 2
step4: h <--- h ^ x (pairwise-and)

$h_0=h_0$^ x
h_0={sunny,warm,normal,strong,warm,same}

step5: if more examples .Go to step2

h(0)=sunny
h(1)=warm
h(2)=normal
h(3)=strong
h(4)=warm
h(5)=same

we will take a concept step by step and rule we need to learn this concept with the training data below

sky	temp	humidity	wind	water	forecast	enjoy
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	same	yes

$$a_h \wedge a_x = \begin{cases} a_x & : \text{if } a_h = \emptyset \\ a_x & : \text{if } a_h = a_h \\ ? & : \text{if } a_h \neq a_x \\ ? & : \text{if } a_h = ? \end{cases}$$

MACHINE INTELLIGENCE

Find S algorithm-Problem

step2: next input {x,c(x)}

x={sunny,warm,high,strong,warm,same}
c(x)=yes==1

step3: if c(x)=0 ,go to step 2
step4: h <--- h ^ x (pairwise-and)

$h_0=h_0$^ x
h_0={sunny,warm,?,strong,warm,same}

step5: if more examples Go to step 2

h(0)=sunny
h(1)=warm
h(2)=?
h(3)=strong
h(4)=warm
h(5)=same

sky	temp	humidity	wind	water	forecast	enjoy
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	same	yes

$$a_h \wedge a_x = \begin{cases} a_x & : \text{if } a_h = \emptyset \\ a_x & : \text{if } a_h = a_h \\ ? & : \text{if } a_h \neq a_x \\ ? & : \text{if } a_h = ? \end{cases}$$

MACHINE INTELLIGENCE

Find S algorithm-Problem

step2: next input {x,c(x)}

x={rainy,cold,high,strong,warm,change}
c(x)=yes==0

step3: if c(x)=0 ,go to step 2

sky	temp	humidity	wind	water	forecast	enjoy
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	same	yes

$$a_h \wedge a_x = \begin{cases} a_x & : \text{if } a_h = \emptyset \\ a_x & : \text{if } a_h = a_h \\ ? & : \text{if } a_h \neq a_x \\ ? & : \text{if } a_h = ? \end{cases}$$

MACHINE INTELLIGENCE

Find S algorithm-Problem

step2: next input {x,c(x)}

x={sunny,warm,high,strong,cool,same}
c(x)=yes==1

step3: if c(x)=0 ,go to step 2
step4: h <--- h ^ x (pairwise-and)

$h_0=h_0$^ x
h_0={sunny,warm,?,strong,?,same}

step5: if more examples Go to step 2
step6: stop

h(0)=sunny
h(1)=warm
h(2)=?
h(3)=strong
h(4)=?
h(5)=same

sky	temp	humidity	wind	water	forecast	enjoy
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	same	yes

$$a_h \wedge a_x = \begin{cases} a_x & : \text{if } a_h = \emptyset \\ a_x & : \text{if } a_h = a_h \\ ? & : \text{if } a_h \neq a_x \\ ? & : \text{if } a_h = ? \end{cases}$$

MACHINE INTELLIGENCE

Find S algorithm-prediction

C={sunny,warm,?,strong,?,same}

x={sunny,warm,high,strong,warm,same}

c(x)=1

suppose now we give the test data set and we can pass the data through our concept and we give our final hypothesis as our concept that we use the concept learnt to predict the labels that is if the person will play a sport or no

MACHINE INTELLIGENCE

Version space

a hypothesis is said to be consistent wrt to training data set if it classifies all the object of training data set to their corresponding classes

X	C(X)
x_1	$c(x_1)$
x_2	$c(x_2)$

so on.....

a hypothesis is said to be consistent if $h(x_i)=C(x_i)$

at any given point let H be our hypothesis space
so a version space is a subset of H
but all concepts in version space VS are consistent wrt to training set

i.e VS={h : h $\in$ **H** and h is consistent with $D_{training}$

this ensures our algorithm learns only the best hypothesis

[illegible] the hypothesis space may never find the right concept that is to be learned and always predict wrong.

THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701

MACHINE INTELLIGENCE

Performance metrics

K.S.Srinivas
Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Performance metrics

Srinivas K S.
Associate Professor, Department of Computer Science

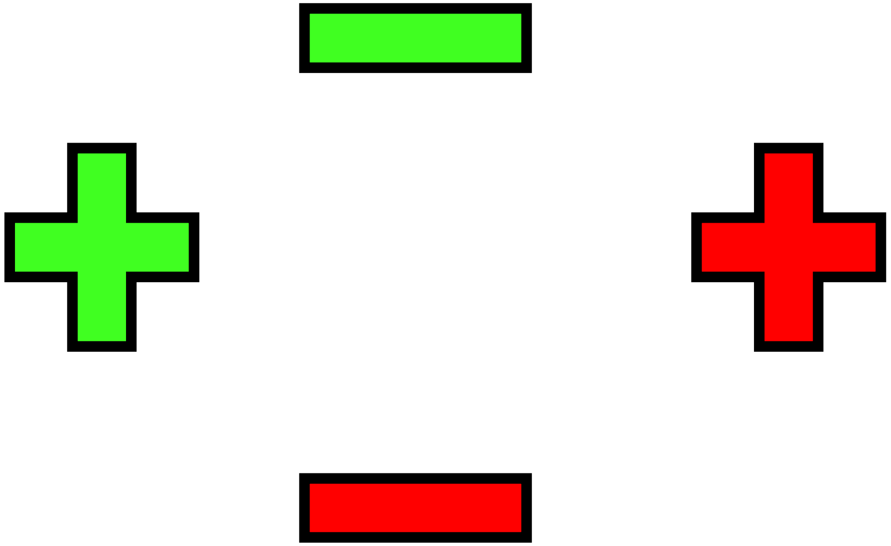
1. Accuracy
2. Precision
3. Recall
4. Specificity
5. Receiver Operating Characteristics (ROC)
6. Area Under Curve (AUC)

After the initial feature Engineering and using implementing a model, we get some output in terms of a class or probability.

Confusion Matrix

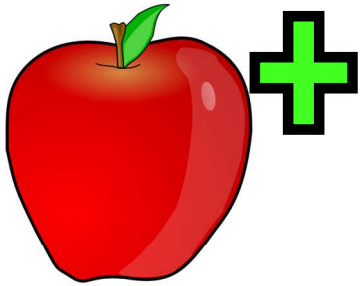
1. True Positive
2. True Negatives
3. False Positive
4. False Negatives

[illegible] confusion matrix.
It is a simple way to lay out ,how many predicted categories or classes were correctly predicted and how many were not.

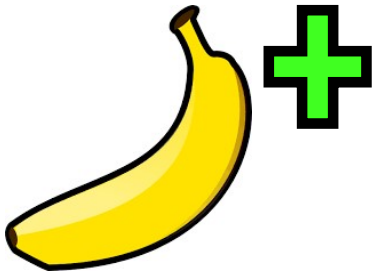


MACHINE INTELLIGENCE

Confusion Matrix



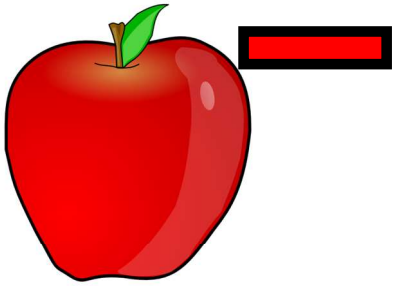
class A correctly predicted as class A



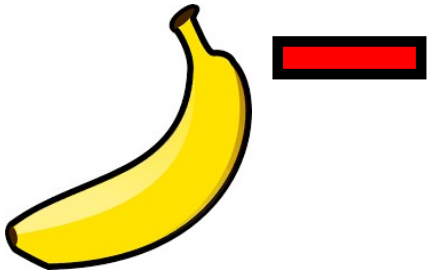
class B correctly predicted as class B

suppose our classification model has two class class A(**apple**) and class B (**all other fruits**)

essentially the confusion matrix is keeping track of



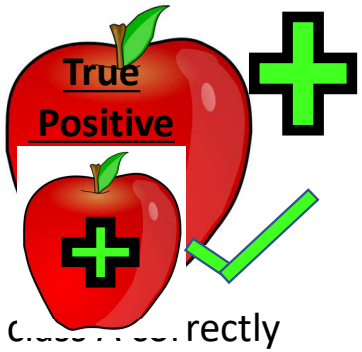
class A incorrectly predicted as class B



class B incorrectly predicted as class A

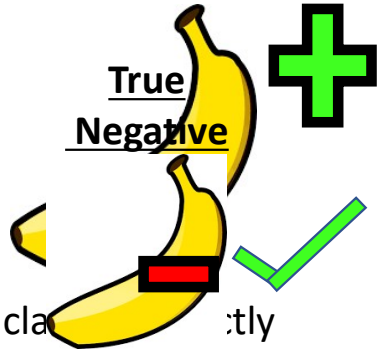
MACHINE INTELLIGENCE

Confusion Matrix



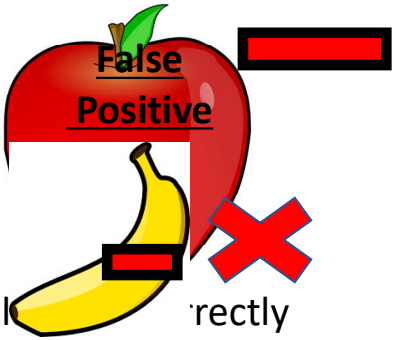
rectly predicted as class A

a positive class A predicted as class A



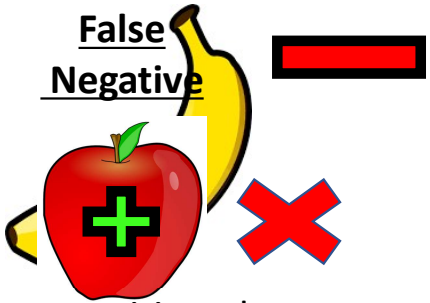
predicted as class B

a negative class B predicted as class B



predicted as class B

B predicted as positive class A



class B incorrectly predicted as class A

a positive class A predicted as negative class B

suppose our classification model is trying to classify fruits (**apple**) and class B (**all other fruits**)

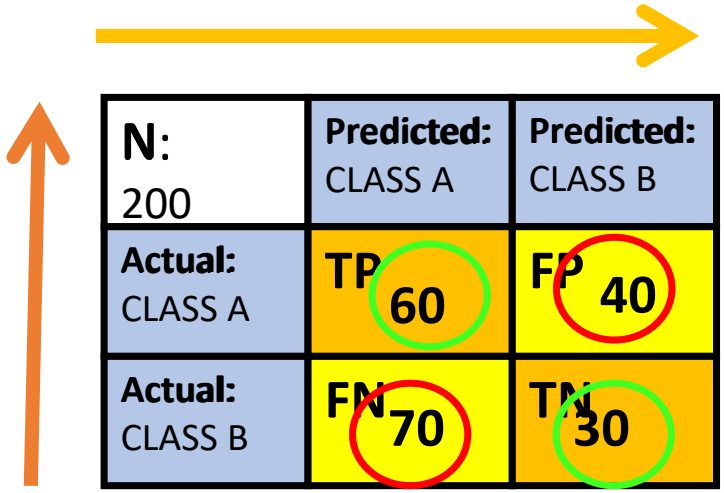
essentially the confusion matrix is keeping track of our false positive and false negative are as follows

GOAL

As many predictions as possible
More true then false

MACHINE INTELLIGENCE

Confusion Matrix



N: 200	Predicted: CLASS A	Predicted: CLASS B
Actual: CLASS A	TP 60	FP 40
Actual: CLASS B	FN 70	TN 30

NOTE:
FP and FN are type 1 and type 2 error respectively (same as what you learn't in IDS course in 3rd sem)

we draw a matrix grid
with predictions from our model as below
along x axis we represent predicted values and along y axis we represent actual values

N	class A	class B
200	100	100

- 60 of the data object were correctly predicted as positive class A as in the data set
- 30 of the data object were correctly predicted as negative class B as in the data set

- most of the time it doesn't matter as the confusion matrix will tell us how many predictions were properly made from each class
- but if your classification is binary or of type one vs all you can assign the target class as +ve class

MACHINE INTELLIGENCE

Accuracy

Accuracy is given by,

$$\textbf{Accuracy=} \frac{TP + TN}{(TP + TN + FP + FN)}$$

Accuracy in classification problems is the number of correct predictions made by the model over all kinds predictions made.

Accuracy is generally a good measure when the target variable classes are nearly balanced.

MACHINE INTELLIGENCE

Precision and Recall

Precision: How many +ve cases did we catch?

Recall: How many did we miss? (sensitivity)

Precision is given by,

Precision= $\frac{TP}{(TP+FP)}$

Recall is given by,

Recall= $\frac{TP}{(TP+FN)}$

MACHINE INTELLIGENCE

Specificity and F1 score

Specificity : How many -ve cases did we catch?

Recall: The harmonic mean of precision and recall.

Specificity is given by,

Specificity= $\frac{TN}{(TN+FP)}$

F1 score is given by,

F1 score= $\frac{2*(recall*precision)}{(recall+precision)}$

MACHINE INTELLIGENCE

Metrics calculations

N: 200	Predicted: CLASS A	Predicted: CLASS B	CLASSIFICA-TION OVERALL	
Actual: CLASS A	TP 60	FP 40	100	precision
Actual: CLASS B	FN 70	TN 30	100	
TRUTH OVERALL	130	70	200	
	recall	specificity		accuracy

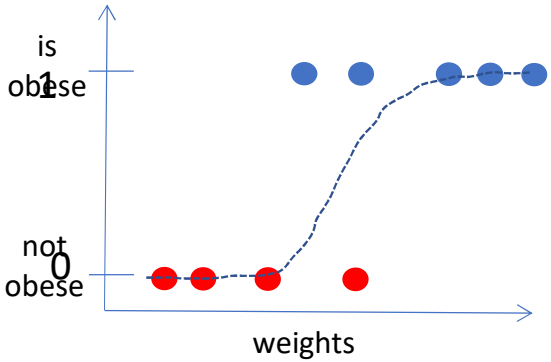
consider the following confusion matrix
Let us calculate all the metrics we discussed till now

+

——— = 0.45

MACHINE INTELLIGENCE

ROC -Receiver Operating Characteristics



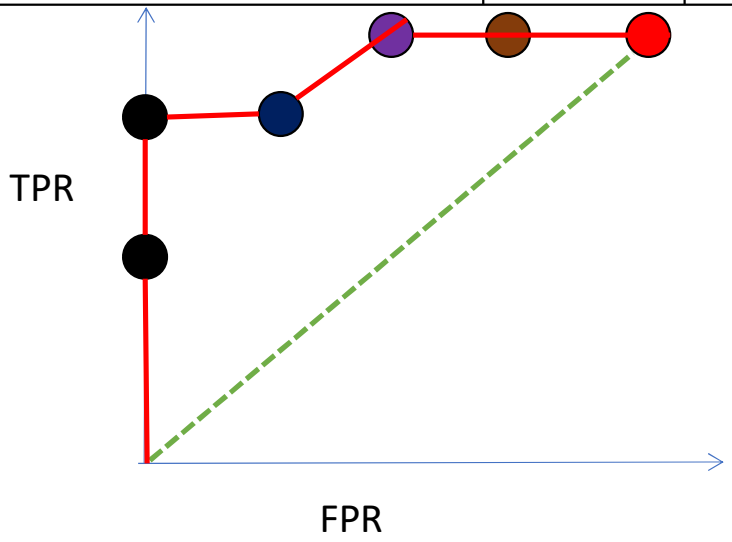
if we need to convert probability to classification, one method is setting up a threshold say at 0.5 and say all sample with probability more than 0.5 is obese and vice versa

- the y axis has two categories- obese and not obese
- the blue dots represent sample who are obese
- the red dots represent sample that are not obese
- along x axis we have weights

MACHINE INTELLIGENCE

ROC -Receiver Operating Characteristics

Threshold	TPR	FPR
0 (all sample classified obese)	1	1
0.3	1	0.75
0.4	1	0.5
0.6	0.75	0.25
0.7	0.75	0
0.9	0.5	0

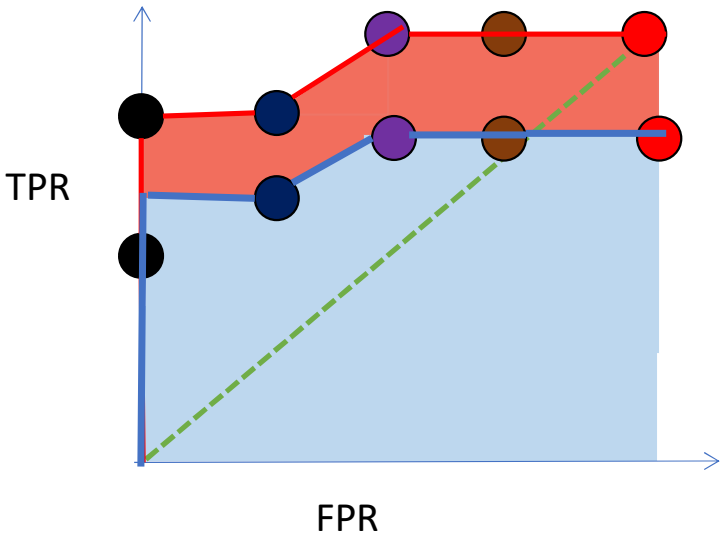


[illegible]
proportion of samples that were incorrectly classified as obese
this means the new threshold is better then the first one

- y axis represents true positive rate (TPR) that is sensitivity
- x axis represents False positive rate (FPR) that is specificity

MACHINE INTELLIGENCE

AUC -Area Under The Curve



THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701

MACHINE INTELLIGENCE

DECISION TREE -ID3 ALGORITHM

K.S.Srinivas
Department of Computer Science and Engineering

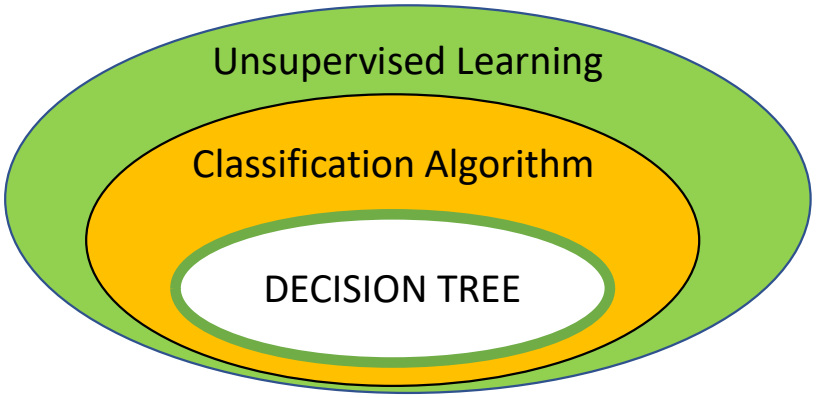
MACHINE INTELLIGENCE

DECISION TREE -ID3 ALGORITHM

Srinivas K S.
Associate Professor, Department of Computer Science

What is a Decision Tree algorithm?

- A type of classification algorithm
- Comes under unsupervised learning technique

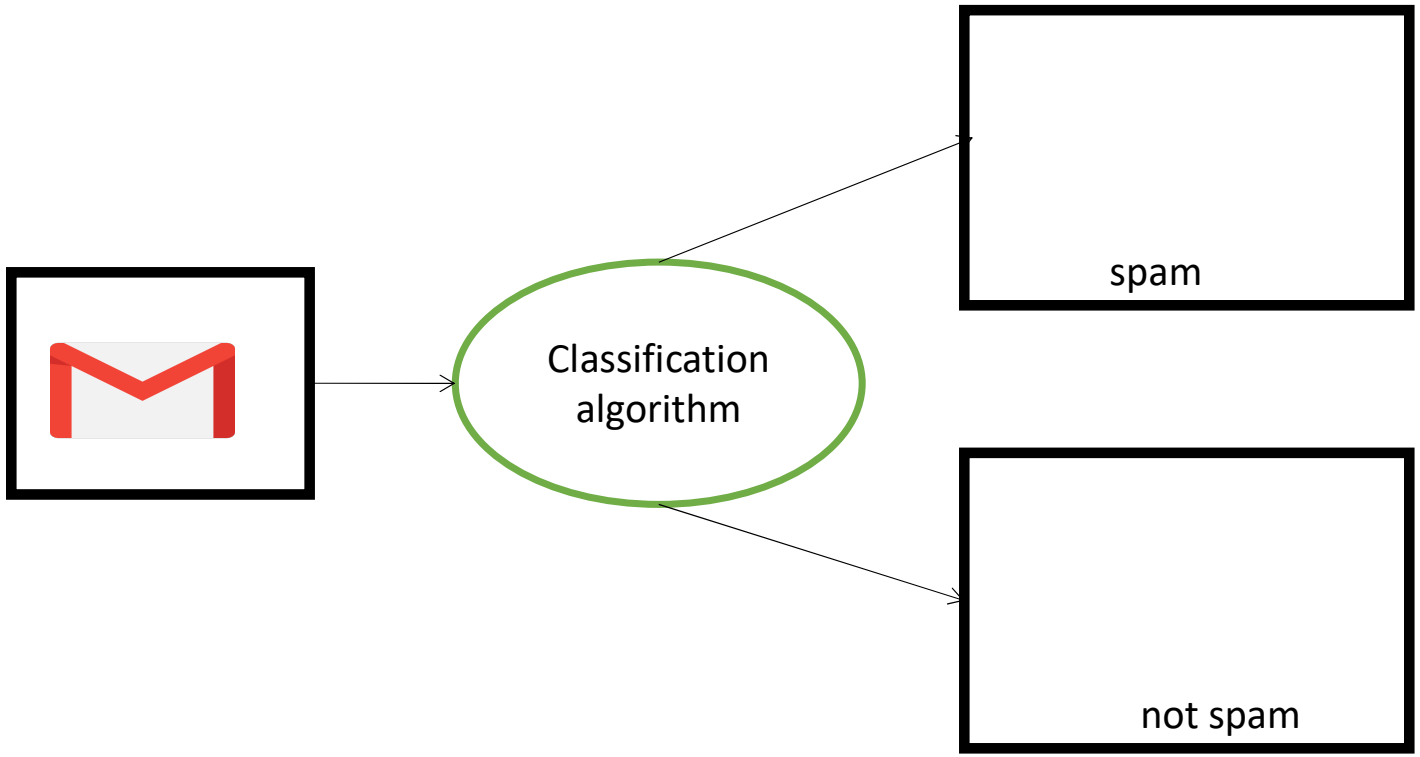


MACHINE INTELLIGENCE

What is classification algorithm?

"Classification is the process of dividing the data sets into different categories or groups by adding label"

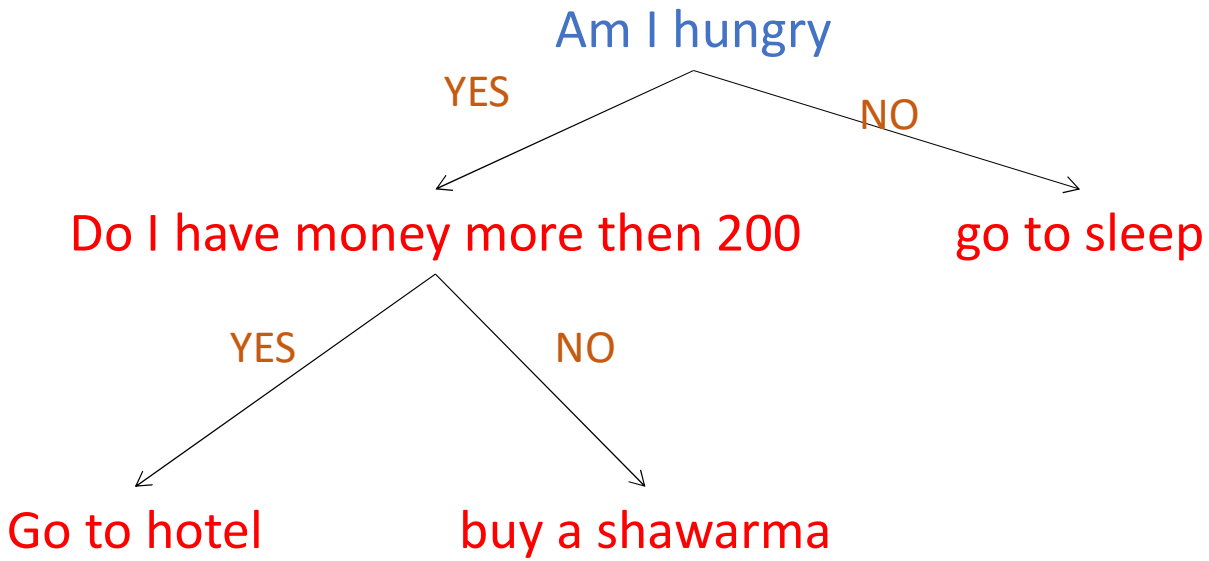
example: classification of emails as spam or not spam based on certain conditions



MACHINE INTELLIGENCE

What is a Decision Tree?

- Graphical representation of all the possible solutions to a decision
- Decisions are based on some conditions
- Decision made can be easily explained



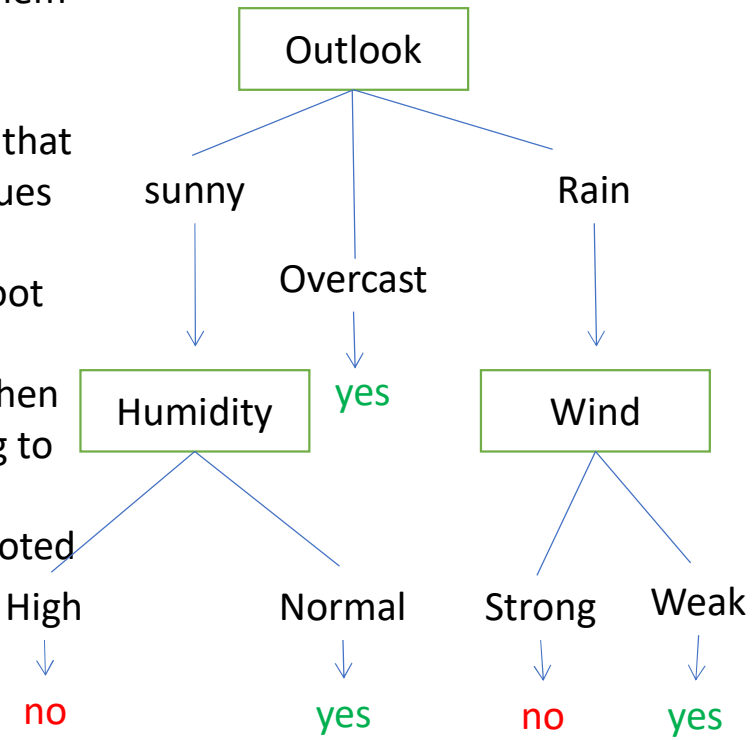
MACHINE INTELLIGENCE

Decision Tree Learning definition-Tom Mitchell

"Decision tree learning is a method for approximating discrete-valued target function,in which the learned function is represented by a decision tree."

- Decision trees classify instances by sorting them down the tree from the root to some leaf node ,which provides classification of the instances and each branch descending from that node corresponds to one of the possible values of this attribute.
- An instances is classified by starting at the root node of the tree
- testing the attribute specified by this node then moving down the tree branch corresponding to value of the attribute.
- This process is then repeated for sub tree rooted at the new node

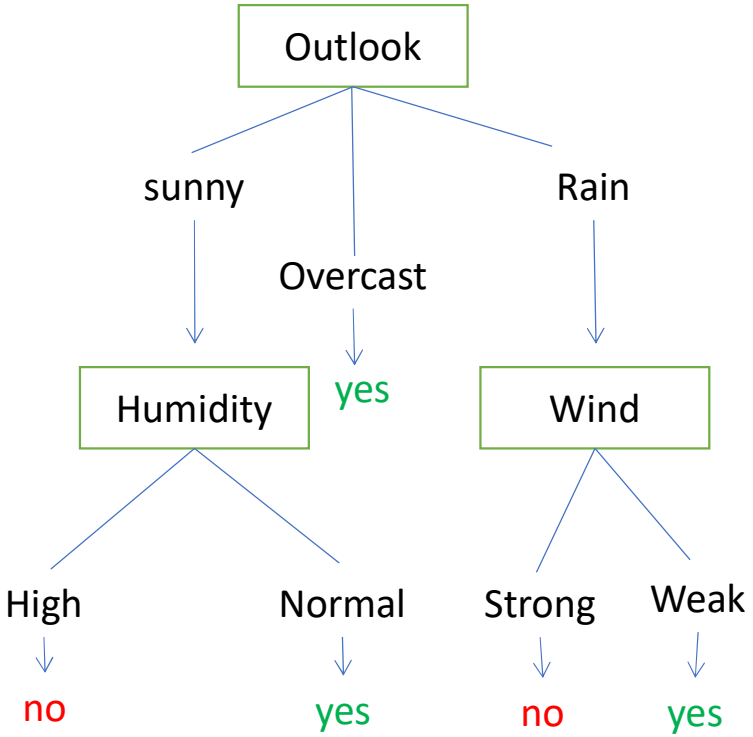
Decision tree for concept to play Tennis



MACHINE INTELLIGENCE

Decision Function

- Based on this decision tree lets try to think of the function that our machine may have learn't



f(outlook,humidity,wind)= {

1 **if** (outlook=sunny and humidity=Normal)
or
(outlook=overcast)
or
(outlook=Rain and Wind=weak)

0 for any other inputs

}

MACHINE INTELLIGENCE

Appropriate Problem for Decision Tree Learning

- **Instances are Represented by attribute-value pairs**
  example: Temperature and their values
- **The target function has discrete output values**
  example the previous example of concept of playing tennis
- **Disjunctive descriptions may be required**
  As noted above, decision trees naturally represent disjunctive expressions.
- **The training data may contain errors**
  Decision tree learning methods are robust to errors, both errors in classifications of the training examples and errors in the attribute values that describe these examples.
- **The training data may contain missing attribute values**
  Decision tree methods can be used even when some training examples have unknown values

MACHINE INTELLIGENCE

ID3 Algorithm

- In this course we will look at the basic ID3 algorithm for learning decision trees
- Later we will examine the hypothesis space search performed by this learning algorithm.
- We will then head forward to characterize the inductive bias of this ID3 algorithm
- At last we will see the problem of over fitting the training data , also check strategies to deal with it.

MACHINE INTELLIGENCE

ID3 Approach

- Our basic algorithm ,ID3 learns decision trees by constructing them top-down, beginning with question .

  ”which attribute should be tested at the root of the tree?”
- The simple answer **statistical test**
- We evaluate each instance attribute using statistical test to determine how well it alone classifies the training example.
- The best attribute is selected and used as the test at the root node of the tree.
- A descendant of the root node is then created for each possible value of this attribute, and the training examples are sorted to the appropriate descendant node.
- The entire process is then repeated using the training examples associated with each descendant node to select the best attribute to test at that point in the tree. This forms a greedy search for an acceptable decision tree, in which the algorithm never backtracks to reconsider earlier choices.
- We will define a statistical property, called information gain, that measures how well a given attribute separates the training examples according to their target classification.

MACHINE INTELLIGENCE

Entropy

- In order to define information gain precisely, we begin by defining a measure commonly used in information theory, called entropy
- characterizes the (im)purity of an arbitrary collection of examples.
- Given a collection S, containing positive **(p)** and negative **(n)** examples of some target concept, the entropy of S relative to this Boolean classification is

$$\text{Entropy}(S) = \frac{-p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{n+p}\log_2\left(\frac{n}{n+p}\right)$$

uncertainty due to positive examples in data set

uncertainty due to positive examples in data set

- More generally, if the target attribute can take on c different values, then the entropy of S relative to this c-wise classification is defined as

$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

where p_i is the proportion of S belonging to class i.

MACHINE INTELLIGENCE

Entropy Calculation

consider the following dataset of people accepting a job

salary	Location	job acceptance
Tier1	MUM	YES
Tier 2	BLR	YES
Tier 1	BLR	NO
Tier 1	HYD	NO
Tier 2	MUM	YES
Tier 1	HYD	NO
Tier 1	HYD	NO

$$\text{Entropy}(S) = \frac{-p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{n+p}\log_2\left(\frac{n}{n+p}\right)$$

Now let us calculate total number of positive and negative points

number of positive samples p=3

number of negative samples n = 4

$$\text{Entropy}(S) = \frac{-3}{3+4}\log_2\left(\frac{3}{3+4}\right) - \frac{4}{4+3}\log_2\left(\frac{4}{4+3}\right)$$

$$= -0.428 \times (-1.222) - 0.5714 \times -0.8$$

$$= 0.98$$

MACHINE INTELLIGENCE

Entropy Calculation

$$\text{Entropy}(S) = \frac{-p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{n+p}\log_2\left(\frac{n}{n+p}\right)$$

now we first calculate the entropy for salary = Tier1 which we need to split our data

number of positive points = 1

number of negative points = 7

$$\text{Entropy}(\text{Salary} = \text{Tier1}) = \frac{-1}{1+7}\log_2\left(\frac{1}{1+7}\right) - \frac{7}{7+1}\log_2\left(\frac{7}{1+7}\right)$$

$$= 0.375 - (-0.16856)$$

$$= 0.543$$

salary	Location	job acceptance
Tier1	MUM	YES
Tier 1	BLR	NO
Tier 1	HYD	NO
Tier 1	HYD	NO
Tier 1	HYD	NO
Tier 1	HYD	NO
Tier 1	HYD	NO

salary	Location	job acceptance
Tier 2	BLR	YES
Tier 2	MUM	YES

salary	entropy
Tier1	0.543

MACHINE INTELLIGENCE

Entropy Calculation

$$\text{Entropy}(S) = \frac{-p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{n+p}\log_2\left(\frac{n}{n+p}\right)$$

we will calculate entropy of salary=Tier2

number of positive points = 2

number of negative points = 0

$$\text{Entropy}(\text{Salary} = \text{Tier2}) = \frac{-2}{2+0}\log_2\left(\frac{2}{2+0}\right) - \frac{0}{2+0}\log_2\left(\frac{0}{2+0}\right)$$

$= 0 - 0$

$= 0$

salary	Location	job acceptance
Tier1	MUM	YES
Tier 1	BLR	NO
Tier 1	HYD	NO
Tier 1	HYD	NO
Tier 1	HYD	NO
Tier 1	HYD	NO
Tier 1	HYD	NO

salary	Location	job acceptance
Tier 2	BLR	YES
Tier 2	MUM	YES

salary	entropy
Tier1	0.543
Tier2	0

Note: ENTROPY =0 when all samples are of one class
ENTROPY=1 when all class have equal samples

MACHINE INTELLIGENCE

Average Information

The statistical term Average Information of a attribute is given by

$$I(\text{Attribute}) = \sum \frac{p_i + n_i}{p+n} Entropy(A)$$

Lets us understand this by using the previous calculations we did

salary	entropy
Tier1	0.543
Tier2	0

now,

$$I(\text{SALARY}) = \frac{p_{tier1} + n_{tier1}}{p+n} Entropy(salary = tier1) + \frac{p_{tier2} + n_{tier2}}{p+n} Entropy(salary = tier2)$$

$$I(\text{SALARY}) = \frac{1+7}{3+7} \times 0.543 + \frac{2+0}{3+7} \times 0$$

=0.4344

MACHINE INTELLIGENCE

Information Gain

- Given entropy as a measure of the impurity in a collection of training examples, we can now define a measure of the effectiveness of an attribute in classifying the training data
- The measure we will use, called **information gain**
- Is simply the expected reduction in entropy caused by partitioning the examples according to this attribute
- More precisely Information gain G(S,A) of an attribute A relative to collection of example S is defined by

G(S, A)=ENTROPY(S)-I(A)

that is differences of entropy of the collection of example S and information gain of the attribute A

MACHINE INTELLIGENCE

Information Gain- Calculation

from our previous calculation we have the following data

Entropy(S)	0.98
I(salary)	0.4344

let us calculate the Information gain G(S, Salary)

G(S, A)=ENTROPY(S)-I(A)

G(S,SALARY)=ENTROPY(S)-I(SALARY)

G(S,SALARY)=0.98-0.4344
=0.5456

MACHINE INTELLIGENCE

The ID3 Algorithm

Steps to create a decision tree using the ID3 algorithm

1. COMPUTE THE **ENTROPY** FOR DATA-SET **ENTROPY(S)**
2. FOR EVERY ATTRIBUTE
   - CALCULATE ENTROPY FOR ALL OTHER VAUES **ENTROPY(A)**
   - TAKE **AVERAGE INFORMATION ENTROPY** FOR THE CURRENT ATTRIBUTE
   - CLACULATE **GAIN** FOR THE CURRENT ATTRIBUTE
3. PICK THE **HIGHEST GAIN ATTRIBUTE**
4. **REPEAT** UNTIL WE GET THE TREE WE DESIRED

MACHINE INTELLIGENCE

The ID3 Algorithm- problem

Let us now use all the previous knowledge and create a decision tree of the following data set the play tennis is the target attribute here

Outlook	Temp	Humidity	Windy	Play tennis
Sunny	High	High	Weak	No
Sunny	High	High	Strong	No
Overcast	High	High	Weak	Yes
Rainy	Medium	High	Weak	Yes
Rainy	Cool	Normal	Weak	Yes
Rainy	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Medium	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rainy	Medium	Normal	Weak	Yes
Sunny	Medium	Normal	Strong	Yes
Overcast	Medium	High	Strong	Yes
Overcast	High	Normal	Weak	Yes
Rainy	Medium	High	Strong	No

Lets create decision tree for this by following the steps

step 1: COMPUTE THE **ENTROPY** FOR DATA-SET

ENTROPY(S)

$$\text{Entropy(S)} = \frac{-\,p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{n+p}\log_2\left(\frac{n}{n+p}\right)$$

number of positive points = 9

number of negative points = 5

$$\text{Entropy(S)} = \frac{-9}{9+5}\log_2\left(\frac{9}{9+5}\right) - \frac{5}{5+9}\log_2\left(\frac{5}{5+9}\right)$$

=0.94

MACHINE INTELLIGENCE

The ID3 Algorithm- problem

2. FOR EVERY ATTRIBUTE
   - CALCULATE ENTROPY FOR ALL OTHER VAUES **ENTROPY(A)**
   - TAKE **AVERAGE INFORMATION ENTROPY** FOR THE CURRENT ATTRIBUTE
   - CLACULATE **GAIN** FOR THE CURRENT ATTRIBUTE

we will first check outlook attribute and create sub table for outlook =sunny,outlook =rainy, and outlook =overcast

for outlook=sunny

number of positive points = 2

number of negative points = 3

$$Entropy(outlook = sunny) = \frac{-2}{2+3}\log_2(\frac{2}{2+3}) - \frac{3}{2+3}\log_2(\frac{3}{2+3}) = 0.971$$

Similarly doing for rainy and overcast we have the following results

$$Entropy(outlook = rainy) = \frac{-3}{2+3}\log_2(\frac{3}{2+3}) - \frac{2}{2+3}\log_2(\frac{2}{2+3}) = 0.971$$

Outlook	Play tennis
Sunny	No
Sunny	No
Sunny	No
Sunny	Yes
Sunny	Yes

property	value
Entropy(s)	0.94
G(outlook)	0.247

now we calculate Average information of the attribute outlook

$$I(outlook) = \frac{3+2}{9+5} * 0.971 + \frac{2+3}{9+5} * 0.971 + \frac{4+0}{9+5} * 0 = 0.693$$

finally we calculate Information gain for the attribute outlook

G(S,outlook)=0.94-0.693=0.247

MACHINE INTELLIGENCE

The ID3 Algorithm- problem

we will do the same procedures for other table and the obtain the following result

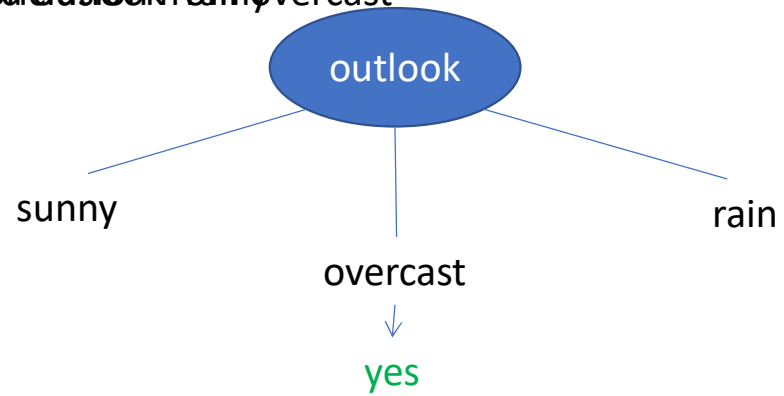
property	value
Entropy(s)	0.94
G(outlook)	0.247
G(temp)	0.029
G(humidity)	0.152
G(windy)	0.048

MACHINE INTELLIGENCE

The ID3 Algorithm- problem

3. PICK THE **HIGHEST GAIN ATTRIBUTE**
4. **REPEAT** UNTIL WE GET THE TREE WE DESIRED

we will now create our tree using the 3rd and 4th step



property	value
Entropy(s)	0.94
G(outlook)	0.247
G(temp)	0.029
G(humidity)	0.152
G(windy)	0.048

MACHINE INTELLIGENCE

The ID3 Algorithm- problem

with outlook=sunny our data would look something like this

first we calculate entropy of the data set

p=2 n=3

$$\text{Entropy}(S_{\text{sunny}}) = \frac{-2}{2+3}\log_2\left(\frac{2}{2+3}\right) - \frac{3}{3+2}\log_2\left(\frac{3}{3+2}\right) = 0.97$$

next we need to consider each attribute and calculate its gain

Outlook	Temp	Humidity	Windy	Play tennis
Sunny	High	High	Weak	No
Sunny	High	High	Strong	No
Sunny	Medium	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Sunny	Medium	Normal	Strong	Yes

MACHINE INTELLIGENCE

The ID3 Algorithm- problem

considering temperature attribute

Temperature	p	n	Entropy
cool	1	0	0
high	0	2	0
medium	1	1	1

Outlook	Temp	Play tennis
Sunny	High	No
Sunny	High	No
Sunny	Medium	No
Sunny	Cool	Yes
Sunny	Medium	Yes

Average Information Entropy: I(Temp)=0.4

Gain : G(Temp)=0.571

MACHINE INTELLIGENCE

The ID3 Algorithm- problem

considering humidity attribute

Temperature	p	n	Entropy
normal	2	0	0
high	0	3	0

Outlook	Humidity	Play tennis
Sunny	High	No
Sunny	High	No
Sunny	High	No
Sunny	Normal	Yes
Sunny	Normal	Yes

Average Information Entropy: I(Temp)=0

Gain : G(Temp)=0.971

MACHINE INTELLIGENCE

The ID3 Algorithm- problem

considering windy attribute

Temperature	p	n	Entropy
strong	1	1	1
weak	1	2	0.918

Outlook	Windy	Play tennis
Sunny	Weak	No
Sunny	Strong	No
Sunny	Weak	No
Sunny	Weak	Yes
Sunny	Strong	Yes

Average Information Entropy: I(Temp)=0.951

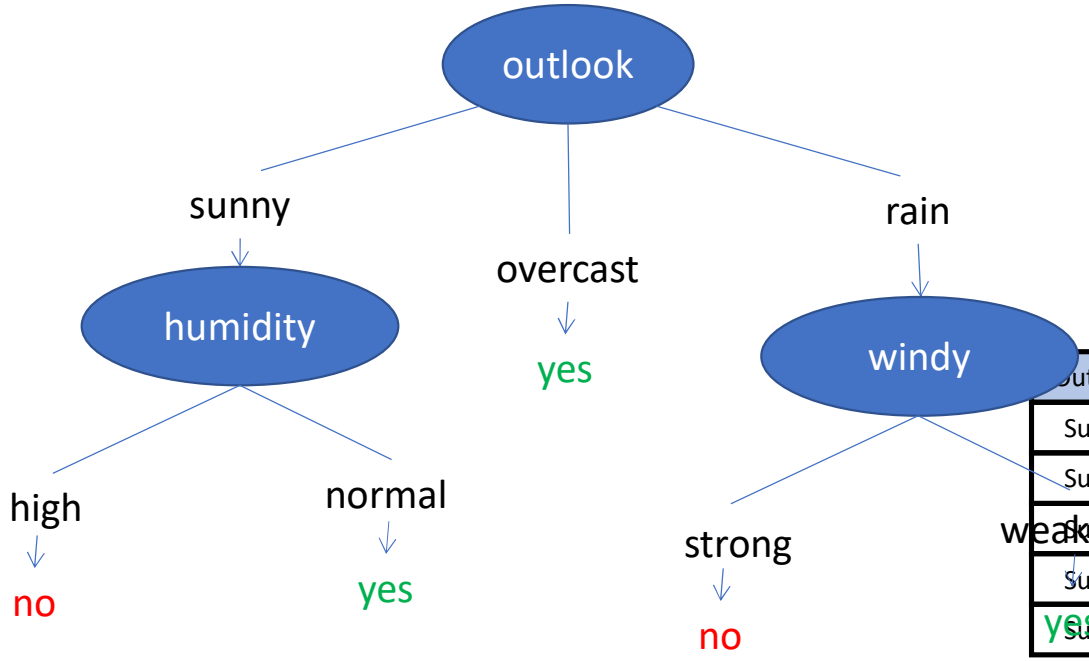
Gain : G(Temp)=0.020

MACHINE INTELLIGENCE

The ID3 Algorithm- problem

after the previous calculations we have the following data for the data with outlook=sunny

property	value
Entropy(S_{sunny})	0.97
G(temp)	0.571
G(humidity)	0.971
G(windy)	0.02



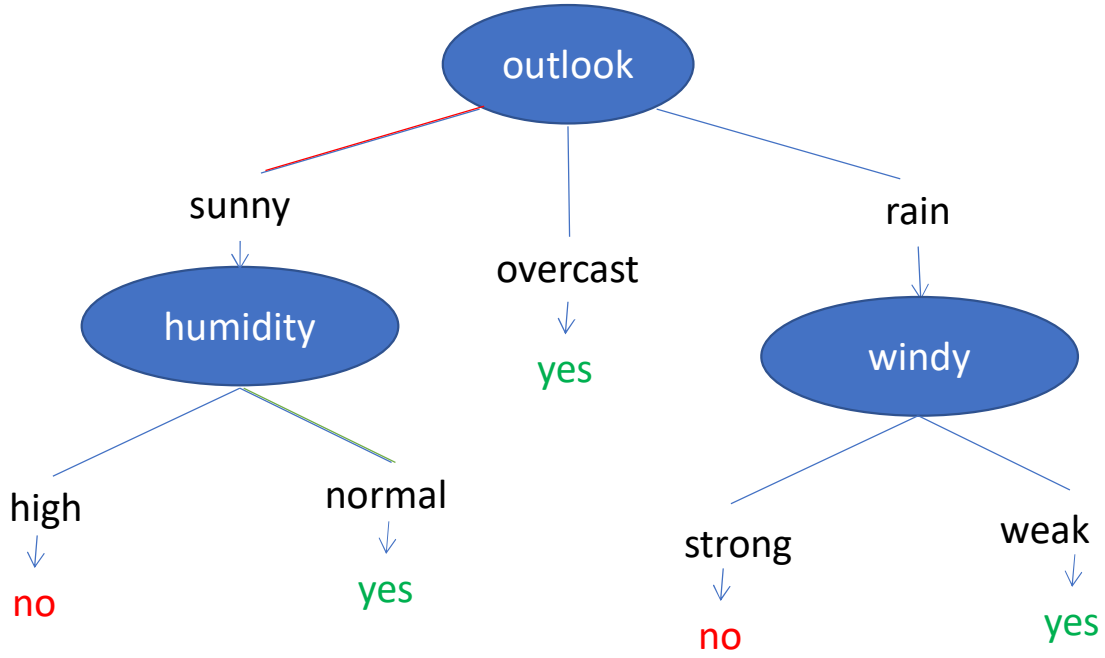
Outlook	Humidity	Play tennis
Sunny	High	No
Sunny	High	No
Sunny	High	No
Sunny	Normal	Yes
Sunny	Normal	Yes

MACHINE INTELLIGENCE

Decision Tree analysis

Let us see how decision tree helps us predicting if a player will play tennis or not

on a given day let this be the report of the weather forecast
*"the day would be **sunny** with **normal humidity** and **weak wind**"*



THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701

MACHINE INTELLIGENCE

DECISION TREE -ID3 ALGORITHM

K.S.Srinivas
Department of Computer Science and Engineering

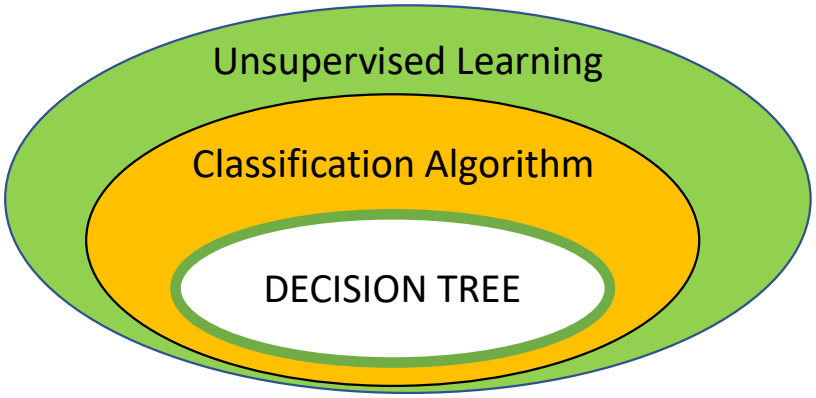
MACHINE INTELLIGENCE

DECISION TREE -ID3 ALGORITHM

Srinivas K S.
Associate Professor, Department of Computer Science

What is a Decision Tree algorithm?

- A type of classification algorithm
- Comes under unsupervised learning technique

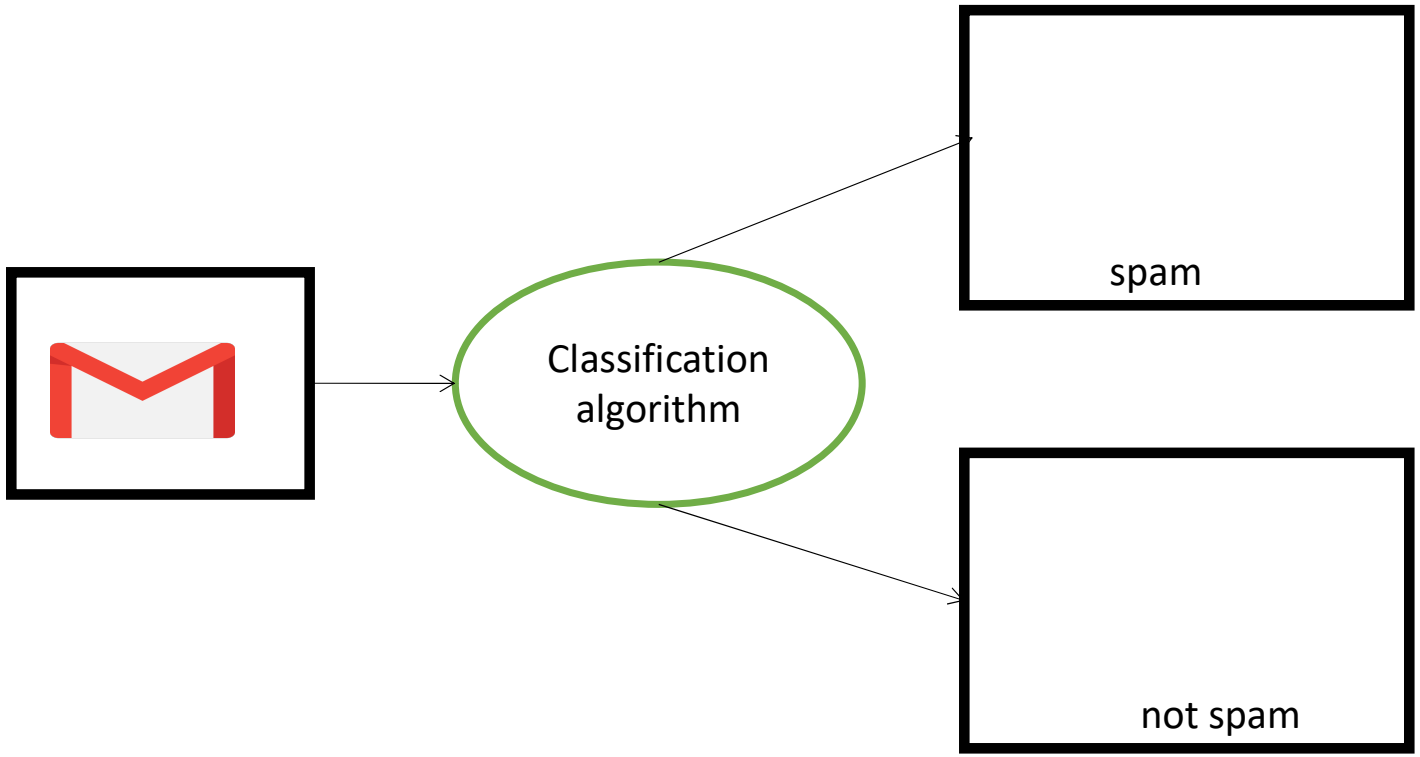


MACHINE INTELLIGENCE

What is classification algorithm?

"Classification is the process of dividing the data sets into different categories or groups by adding label"

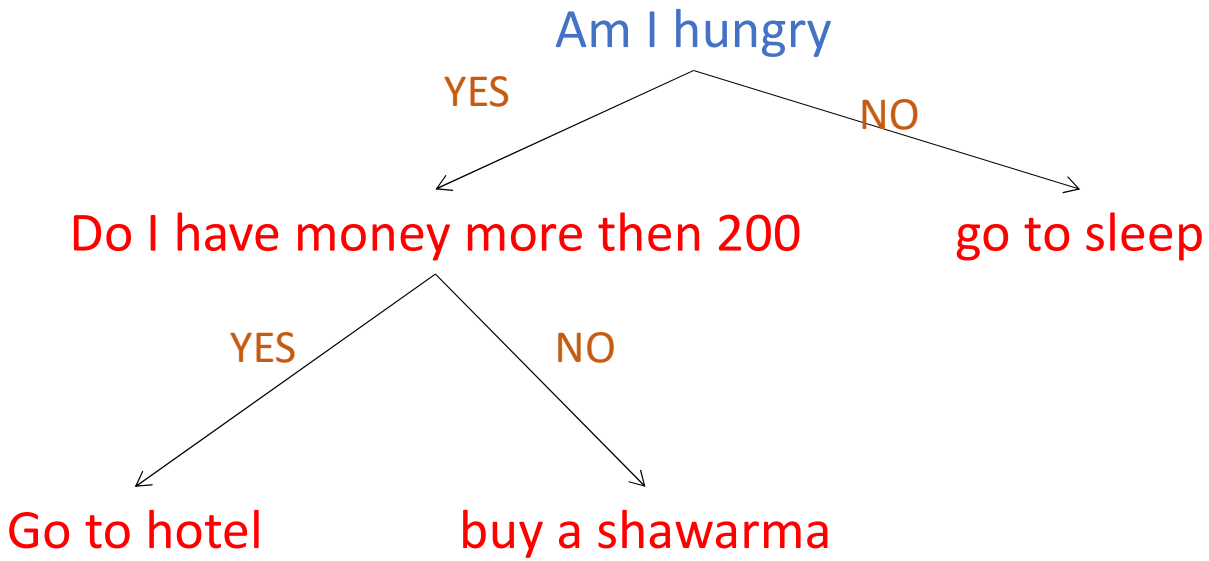
example: classification of emails as spam or not spam based on certain conditions



MACHINE INTELLIGENCE

What is a Decision Tree?

- Graphical representation of all the possible solutions to a decision
- Decisions are based on some conditions
- Decision made can be easily explained



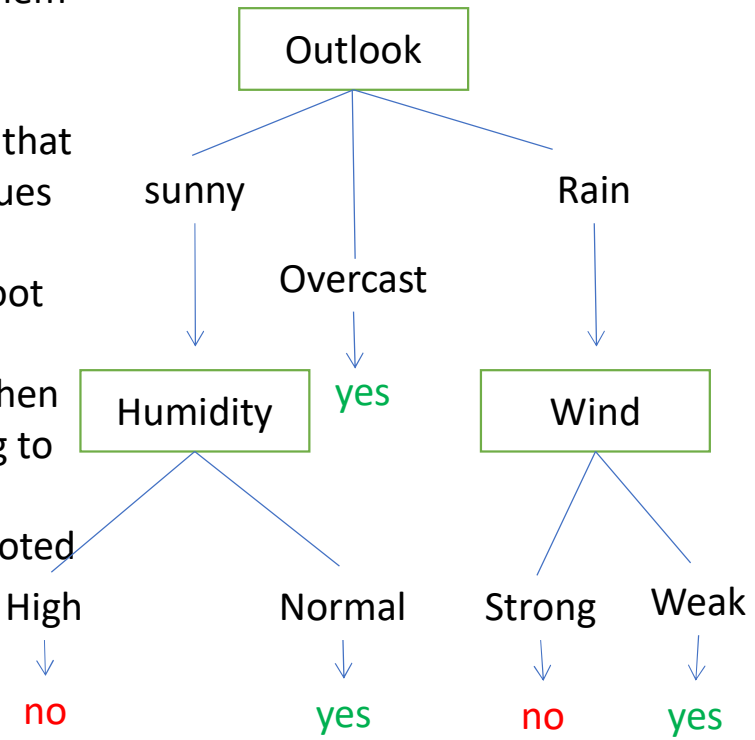
MACHINE INTELLIGENCE

Decision Tree Learning definition-Tom Mitchell

"Decision tree learning is a method for approximating discrete-valued target function,in which the learned function is represented by a decision tree."

- Decision trees classify instances by sorting them down the tree from the root to some leaf node ,which provides classification of the instances and each branch descending from that node corresponds to one of the possible values of this attribute.
- An instances is classified by starting at the root node of the tree
- testing the attribute specified by this node then moving down the tree branch corresponding to value of the attribute.
- This process is then repeated for sub tree rooted at the new node

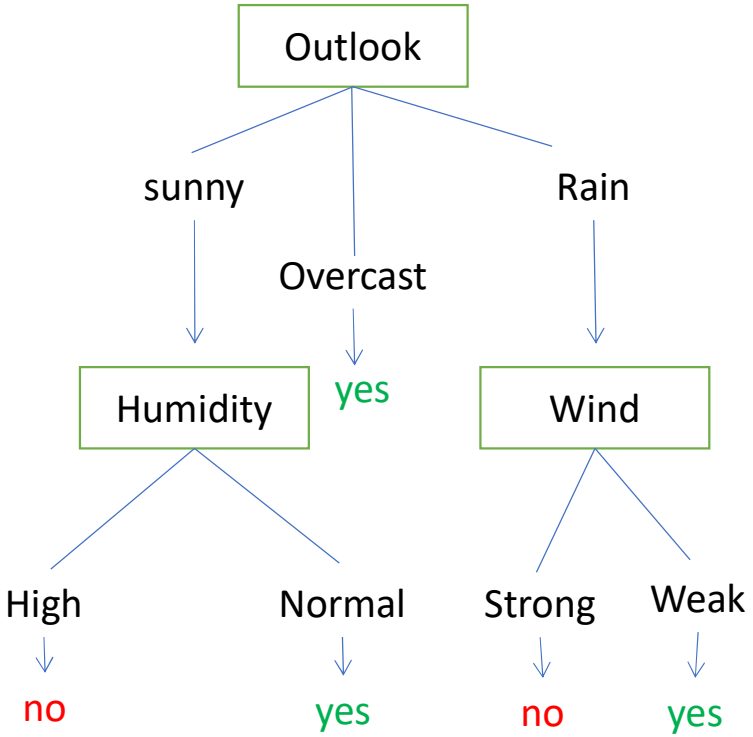
Decision tree for concept to play Tennis



MACHINE INTELLIGENCE

Decision Function

- Based on this decision tree lets try to think of the function that our machine may have learn't



f(outlook,humidity,wind)= {

1 **if** (outlook=sunny and humidity=Normal)
or
(outlook=overcast)
or
(outlook=Rain and Wind=weak)

0 for any other inputs

}

MACHINE INTELLIGENCE

Appropriate Problem for Decision Tree Learning

- **Instances are Represented by attribute-value pairs**
  example: Temperature and their values
- **The target function has discrete output values**
  example the previous example of concept of playing tennis
- **Disjunctive descriptions may be required**
  As noted above, decision trees naturally represent disjunctive expressions.
- **The training data may contain errors**
  Decision tree learning methods are robust to errors, both errors in classifications of the training examples and errors in the attribute values that describe these examples.
- **The training data may contain missing attribute values**
  Decision tree methods can be used even when some training examples have unknown values

MACHINE INTELLIGENCE

ID3 Algorithm

- In this course we will look at the basic ID3 algorithm for learning decision trees
- Later we will examine the hypothesis space search performed by this learning algorithm.
- We will then head forward to characterize the inductive bias of this ID3 algorithm
- At last we will see the problem of over fitting the training data , also check strategies to deal with it.

MACHINE INTELLIGENCE

ID3 Approach

- Our basic algorithm ,ID3 learns decision trees by constructing them top-down, beginning with question .

  ”which attribute should be tested at the root of the tree?”
- The simple answer **statistical test**
- We evaluate each instance attribute using statistical test to determine how well it alone classifies the training example.
- The best attribute is selected and used as the test at the root node of the tree.
- A descendant of the root node is then created for each possible value of this attribute, and the training examples are sorted to the appropriate descendant node.
- The entire process is then repeated using the training examples associated with each descendant node to select the best attribute to test at that point in the tree. This forms a greedy search for an acceptable decision tree, in which the algorithm never backtracks to reconsider earlier choices.
- We will define a statistical property, called information gain, that measures how well a given attribute separates the training examples according to their target classification.

MACHINE INTELLIGENCE

Entropy

- In order to define information gain precisely, we begin by defining a measure commonly used in information theory, called entropy
- characterizes the (im)purity of an arbitrary collection of examples.
- Given a collection S, containing positive **(p)** and negative **(n)** examples of some target concept, the entropy of S relative to this Boolean classification is

$$\text{Entropy}(S) = \frac{-p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{n+p}\log_2\left(\frac{n}{n+p}\right)$$

uncertainty due to positive examples in data set

uncertainty due to positive examples in data set

- More generally, if the target attribute can take on c different values, then the entropy of S relative to this c-wise classification is defined as

$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

where p_i is the proportion of S belonging to class i.

MACHINE INTELLIGENCE

Entropy Calculation

consider the following dataset of people accepting a job

salary	Location	job acceptance
Tier1	MUM	YES
Tier 2	BLR	YES
Tier 1	BLR	NO
Tier 1	HYD	NO
Tier 2	MUM	YES
Tier 1	HYD	NO
Tier 1	HYD	NO

$$\text{Entropy}(S) = \frac{-p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{n+p}\log_2\left(\frac{n}{n+p}\right)$$

Now let us calculate total number of positive and negative points

number of positive samples p=3

number of negative samples n = 4

$$\text{Entropy}(S) = \frac{-3}{3+4}\log_2\left(\frac{3}{3+4}\right) - \frac{4}{4+3}\log_2\left(\frac{4}{4+3}\right)$$

$$= -0.428 \times (-1.222) - 0.5714 \times -0.8$$

$$= 0.98$$

MACHINE INTELLIGENCE

Entropy Calculation

$$\text{Entropy}(S) = \frac{-p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{n+p}\log_2\left(\frac{n}{n+p}\right)$$

now we first calculate the entropy for salary = Tier1 which we need to split our data

number of positive points = 1

number of negative points = 7

$$\text{Entropy}(\text{Salary} = \text{Tier1}) = \frac{-1}{1+7}\log_2\left(\frac{1}{1+7}\right) - \frac{7}{7+1}\log_2\left(\frac{7}{1+7}\right)$$

$$= 0.375 - (-0.16856)$$

$$= 0.543$$

salary	Location	job acceptance
Tier1	MUM	YES
Tier 1	BLR	NO
Tier 1	HYD	NO
Tier 1	HYD	NO
Tier 1	HYD	NO
Tier 1	HYD	NO
Tier 1	HYD	NO

salary	Location	job acceptance
Tier 2	BLR	YES
Tier 2	MUM	YES

salary	entropy
Tier1	0.543

MACHINE INTELLIGENCE

Entropy Calculation

$$\text{Entropy}(S) = \frac{-p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{n+p}\log_2\left(\frac{n}{n+p}\right)$$

we will calculate entropy of salary=Tier2

number of positive points = 2

number of negative points = 0

$$\text{Entropy}(\text{Salary} = \text{Tier2}) = \frac{-2}{2+0}\log_2\left(\frac{2}{2+0}\right) - \frac{0}{2+0}\log_2\left(\frac{0}{2+0}\right)$$

$= 0 - 0$

$= 0$

salary	Location	job acceptance
Tier1	MUM	YES
Tier 1	BLR	NO
Tier 1	HYD	NO
Tier 1	HYD	NO
Tier 1	HYD	NO
Tier 1	HYD	NO
Tier 1	HYD	NO

salary	Location	job acceptance
Tier 2	BLR	YES
Tier 2	MUM	YES

salary	entropy
Tier1	0.543
Tier2	0

Note: ENTROPY =0 when all samples are of one class
ENTROPY=1 when all class have equal samples

MACHINE INTELLIGENCE

Average Information

The statistical term Average Information of a attribute is given by

$$I(\text{Attribute}) = \sum \frac{p_i + n_i}{p+n} Entropy(A)$$

Lets us understand this by using the previous calculations we did

salary	entropy
Tier1	0.543
Tier2	0

now,

$$I(\text{SALARY}) = \frac{p_{tier1} + n_{tier1}}{p+n} Entropy(salary = tier1) + \frac{p_{tier2} + n_{tier2}}{p+n} Entropy(salary = tier2)$$

$$I(\text{SALARY}) = \frac{1+7}{3+7} \times 0.543 + \frac{2+0}{3+7} \times 0$$

=0.4344

MACHINE INTELLIGENCE

Information Gain

- Given entropy as a measure of the impurity in a collection of training examples, we can now define a measure of the effectiveness of an attribute in classifying the training data
- The measure we will use, called **information gain**
- Is simply the expected reduction in entropy caused by partitioning the examples according to this attribute
- More precisely Information gain G(S,A) of an attribute A relative to collection of example S is defined by

G(S, A)=ENTROPY(S)-I(A)

that is differences of entropy of the collection of example S and information gain of the attribute A

MACHINE INTELLIGENCE

Information Gain- Calculation

from our previous calculation we have the following data

Entropy(S)	0.98
I(salary)	0.4344

let us calculate the Information gain G(S, Salary)

G(S, A)=ENTROPY(S)-I(A)

G(S,SALARY)=ENTROPY(S)-I(SALARY)

G(S,SALARY)=0.98-0.4344
=0.5456

MACHINE INTELLIGENCE

The ID3 Algorithm

Steps to create a decision tree using the ID3 algorithm

1. COMPUTE THE **ENTROPY** FOR DATA-SET **ENTROPY(S)**
2. FOR EVERY ATTRIBUTE
   - CALCULATE ENTROPY FOR ALL OTHER VAUES **ENTROPY(A)**
   - TAKE **AVERAGE INFORMATION ENTROPY** FOR THE CURRENT ATTRIBUTE
   - CLACULATE **GAIN** FOR THE CURRENT ATTRIBUTE
3. PICK THE **HIGHEST GAIN ATTRIBUTE**
4. **REPEAT** UNTIL WE GET THE TREE WE DESIRED

MACHINE INTELLIGENCE

The ID3 Algorithm- problem

Let us now use all the previous knowledge and create a decision tree of the following data set the play tennis is the target attribute here

Outlook	Temp	Humidity	Windy	Play tennis
Sunny	High	High	Weak	No
Sunny	High	High	Strong	No
Overcast	High	High	Weak	Yes
Rainy	Medium	High	Weak	Yes
Rainy	Cool	Normal	Weak	Yes
Rainy	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Medium	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rainy	Medium	Normal	Weak	Yes
Sunny	Medium	Normal	Strong	Yes
Overcast	Medium	High	Strong	Yes
Overcast	High	Normal	Weak	Yes
Rainy	Medium	High	Strong	No

Lets create decision tree for this by following the steps

step 1: COMPUTE THE **ENTROPY** FOR DATA-SET

ENTROPY(S)

$$\text{Entropy(S)} = \frac{-\,p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{n+p}\log_2\left(\frac{n}{n+p}\right)$$

number of positive points = 9

number of negative points = 5

$$\text{Entropy(S)} = \frac{-9}{9+5}\log_2\left(\frac{9}{9+5}\right) - \frac{5}{5+9}\log_2\left(\frac{5}{5+9}\right)$$

=0.94

MACHINE INTELLIGENCE

The ID3 Algorithm- problem

2. FOR EVERY ATTRIBUTE
   - CALCULATE ENTROPY FOR ALL OTHER VAUES **ENTROPY(A)**
   - TAKE **AVERAGE INFORMATION ENTROPY** FOR THE CURRENT ATTRIBUTE
   - CLACULATE **GAIN** FOR THE CURRENT ATTRIBUTE

we will first check outlook attribute and create sub table for outlook =sunny,outlook =rainy, and outlook =overcast

for outlook=sunny

number of positive points = 2

number of negative points = 3

$$Entropy(outlook = sunny) = \frac{-2}{2+3}\log_2(\frac{2}{2+3}) - \frac{3}{2+3}\log_2(\frac{3}{2+3}) = 0.971$$

Similarly doing for rainy and overcast we have the following results

$$Entropy(outlook = rainy) = \frac{-3}{2+3}\log_2(\frac{3}{2+3}) - \frac{2}{2+3}\log_2(\frac{2}{2+3}) = 0.971$$

Outlook	Play tennis
Sunny	No
Sunny	No
Sunny	No
Sunny	Yes
Sunny	Yes

property	value
Entropy(s)	0.94
G(outlook)	0.247

now we calculate Average information of the attribute outlook

$$I(outlook) = \frac{3+2}{9+5} * 0.971 + \frac{2+3}{9+5} * 0.971 + \frac{4+0}{9+5} * 0 = 0.693$$

finally we calculate Information gain for the attribute outlook

G(S,outlook)=0.94-0.693=0.247

MACHINE INTELLIGENCE

The ID3 Algorithm- problem

we will do the same procedures for other table and the obtain the following result

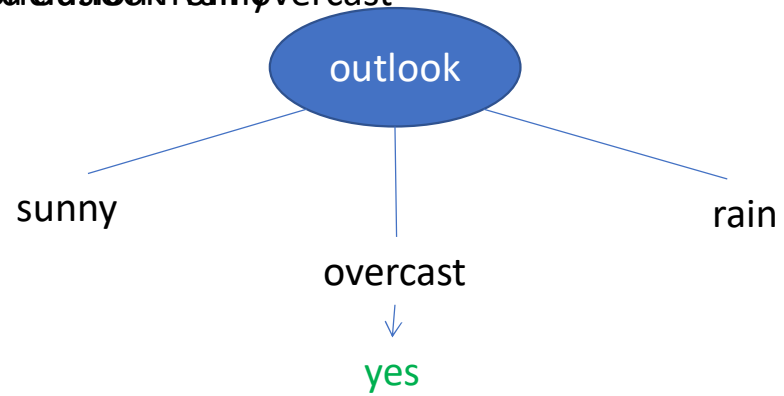
property	value
Entropy(s)	0.94
G(outlook)	0.247
G(temp)	0.029
G(humidity)	0.152
G(windy)	0.048

MACHINE INTELLIGENCE

The ID3 Algorithm- problem

3. PICK THE **HIGHEST GAIN ATTRIBUTE**
4. **REPEAT** UNTIL WE GET THE TREE WE DESIRED

we will now create our tree using the 3rd and 4th step



property	value
Entropy(s)	0.94
G(outlook)	0.247
G(temp)	0.029
G(humidity)	0.152
G(windy)	0.048

MACHINE INTELLIGENCE

The ID3 Algorithm- problem

with outlook=sunny our data would look something like this

first we calculate entropy of the data set

p=2 n=3

$$\text{Entropy}(S_{\text{sunny}}) = \frac{-2}{2+3}\log_2\left(\frac{2}{2+3}\right) - \frac{3}{3+2}\log_2\left(\frac{3}{3+2}\right) = 0.97$$

next we need to consider each attribute and calculate its gain

Outlook	Temp	Humidity	Windy	Play tennis
Sunny	High	High	Weak	No
Sunny	High	High	Strong	No
Sunny	Medium	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Sunny	Medium	Normal	Strong	Yes

MACHINE INTELLIGENCE

The ID3 Algorithm- problem

considering temperature attribute

Temperature	p	n	Entropy
cool	1	0	0
high	0	2	0
medium	1	1	1

Outlook	Temp	Play tennis
Sunny	High	No
Sunny	High	No
Sunny	Medium	No
Sunny	Cool	Yes
Sunny	Medium	Yes

Average Information Entropy: I(Temp)=0.4

Gain : G(Temp)=0.571

MACHINE INTELLIGENCE

The ID3 Algorithm- problem

considering humidity attribute

Temperature	p	n	Entropy
normal	2	0	0
high	0	3	0

Outlook	Humidity	Play tennis
Sunny	High	No
Sunny	High	No
Sunny	High	No
Sunny	Normal	Yes
Sunny	Normal	Yes

Average Information Entropy: I(Temp)=0

Gain : G(Temp)=0.971

MACHINE INTELLIGENCE

The ID3 Algorithm- problem

considering windy attribute

Temperature	p	n	Entropy
strong	1	1	1
weak	1	2	0.918

Outlook	Windy	Play tennis
Sunny	Weak	No
Sunny	Strong	No
Sunny	Weak	No
Sunny	Weak	Yes
Sunny	Strong	Yes

Average Information Entropy: I(Temp)=0.951

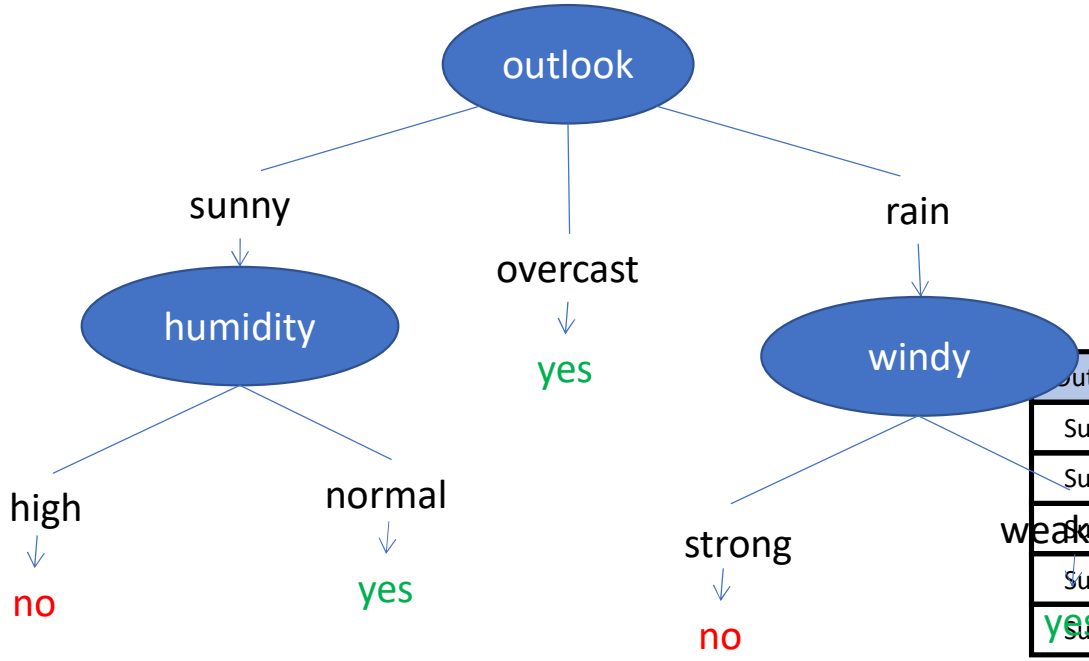
Gain : G(Temp)=0.020

MACHINE INTELLIGENCE

The ID3 Algorithm- problem

after the previous calculations we have the following data for the data with outlook=sunny

property	value
Entropy(S_{sunny})	0.97
G(temp)	0.571
G(humidity)	0.971
G(windy)	0.02



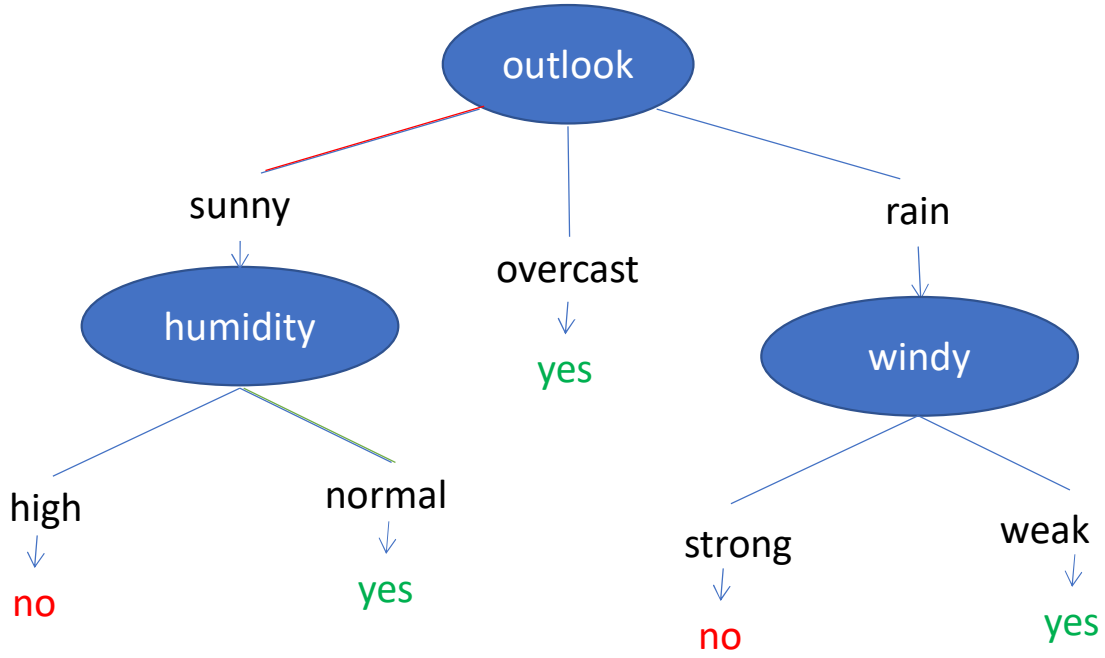
Outlook	Humidity	Play tennis
Sunny	High	No
Sunny	High	No
Sunny	High	No
Sunny	Normal	Yes
Sunny	Normal	Yes

MACHINE INTELLIGENCE

Decision Tree analysis

Let us see how decision tree helps us predicting if a player will play tennis or not

on a given day let this be the report of the weather forecast
*"the day would be **sunny** with **normal humidity** and **weak wind**"*



THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701

MACHINE INTELLIGENCE

Hypothesis Search and Inductive bias-ID3

K.S.Srinivas
Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Hypothesis Search and Inductive bias-ID3

Srinivas K S.
Associate Professor, Department of Computer Science

MACHINE INTELLIGENCE

HYPOTHESIS SPACE SEARCH

- As with other inductive learning methods, ID3 can be characterized as searching a space of hypotheses for one that fits the training examples. The hypothesis space searched by ID3 is the set of possible decision trees
- ID3 performs a simple-to- complex, hill-climbing search through this hypothesis space, beginning with the empty tree, then considering progressively more elaborate hypotheses in search of a decision tree that correctly classifies the training data. .
- The evaluation function that guides this hill-climbing search is the information gain measure

MACHINE INTELLIGENCE

HYPOTHESIS SPACE SEARCH

- 1D3's hypothesis space of all decision trees is a complete space of finite discrete-valued functions, relative to the available attribute
- Because every finite discrete-valued function can be represented by some decision tree, ID3 avoids one of the major risks of methods that search incomplete hypothesis spaces (such as methods that consider only conjunctive hypotheses): that the hypothesis space might not contain the target function.
- ID3 maintains only a single current hypothesis as it searches through the space of decision trees.
- By determin- ing only a single hypothesis, ID^ loses the capabilities that follow from explicitly representing all consistent hypotheses. For example, it does not have the ability to determine how many alternative decision trees are con- sistent with the available training data, or to pose new instance queries that optimally resolve among these competing hypotheses

MACHINE INTELLIGENCE

HYPOTHESIS SPACE SEARCH

- ID3 in its pure form performs no backtracking in its search. Once it,se- lects an attribute to test at a particular level in the tree, it never backtracks to reconsider this choice.
- ID3 uses all training examples at each step in the search to make statistically based decisions regarding how to refine its current hypothesis. This contrasts with methods that make decisions incrementally, based on individual train- ing examples (e.g., FIND-S or CANDIDATE-ELIMINATION).
- One advantage of using statistical properties of all the examples (e.g., information gain) is that the resulting search is much less sensitive to errors in individual training examples. ID3 can be easily extended to handle noisy training data by mod- ifying its t

MACHINE INTELLIGENCE

INDUCTIVE BIAS

- Given a collection of training examples, there are typically many decision trees consistent with these examples. Describing the inductive bias of ID3 there- fore consists of describing the basis by which it chooses one of these consis- tent hypotheses over the others.
- Which of these decision trees does ID3 choose?
- It chooses the first acceptable tree it encounters in its simple-to-complex, hill- climbing search through the space of possible trees.
- Roughly speaking, then , the ID3 search strateg
  1. selects in favor of shorter trees over longer ones
  2. selects trees that place the attributes with highest information gain closest to the root.

Approximate inductive bias of ID3: Shorter trees are preferred over larger trees

A closer approximation to the inductive bias of ID3: Shorter trees are preferred over longer trees. Trees that place high information gain attributes close to the root are preferred over those that do not.

THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701

MACHINE INTELLIGENCE

Issues in Decision Tree Learning and solutions to it

K.S.Srinivas
Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Issues in Decision Tree Learning and solutions to it

Srinivas K S.
Associate Professor, Department of Computer Science

MACHINE INTELLIGENCE

Issues in Decision Tree Learning

- The most important issues in decision tree comes at two place
- over fitting of data
- handling continuous attributes
- we will discuss each of the problem and try to find an appropriate solutions for it

MACHINE INTELLIGENCE

Over fitting of data

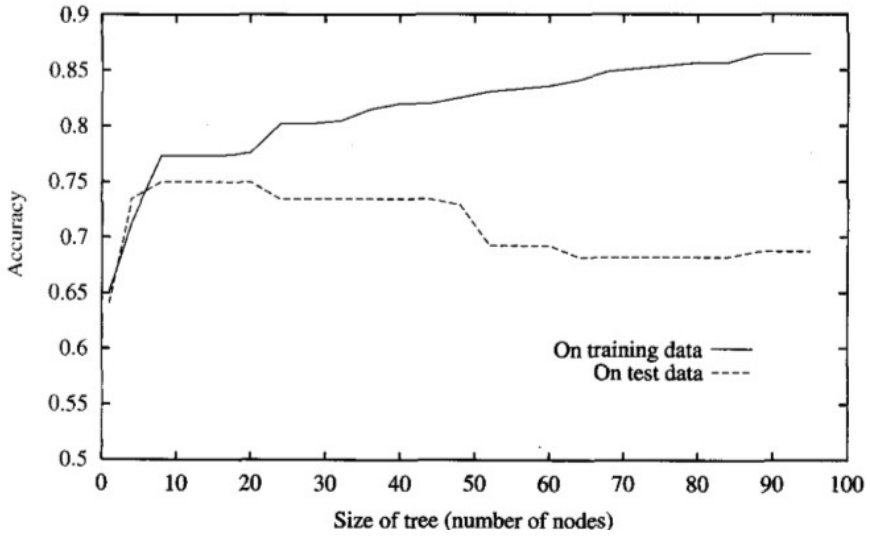
- The ID3 algorithm grows each branch of the tree just deeply enough to perfectly classify the training examples
- It can lead to difficulties when there is noise in the data, or when the number of training examples is too small to produce a representative sample of the true target function. In either of these cases, this simple algorithm can produce trees that over fits the training examples.
- We will say that a hypothesis over fits the training examples if some other hypothesis that fits the training examples less well actually performs better over the entire distribution of instances

- **Definition:**Given a hypothesis space H, a hypothesis h € H is said to over fit the training data if there exists some alternative hypothesis h' € H, such that h has smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances.

MACHINE INTELLIGENCE

Over fitting of data

- The graph illustrates the impact of over fitting in a typical application of decision tree learning.
- In this case, the ID3 algorithm is applied to the task of learning which medical patients have a form of diabetes
- The horizontal axis of this plot indicates the total number of nodes in the decision tree, as the tree is being constructed. The vertical axis indicates the accuracy of predictions made by the tree.
- The solid line shows the accuracy of the decision tree over the training examples, whereas the broken line shows accuracy measured over an independent set of test examples (not included in the training set).
- Predictably, the accuracy of the tree over the training examples increases monotonically as the tree is grown. How- ever, the accuracy measured over the independent test examples first increases, then decreases. As can be seen, once the tree size exceeds approximately 25 nodes
- further elaboration of the tree decreases its accuracy over the test examples despite increasing its accuracy on the training examples.



MACHINE INTELLIGENCE

Avoiding Over fitting of data

- One way this can occur is when the training examples contain random errors or noise.
- There are several approaches to avoiding over fitting in decision tree learning. These can be grouped into two classes:
    1. approaches that stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data,
    2. approaches that allow the tree to over fit the data, and then post-prune the tree.
- Although the first of these approaches might seem.more direct, the second approach of post-pruning over fit trees has been found to be more successful in practice.
- This is due to the difficulty in the first approach of estimating precisely when to stop growing the tree.

MACHINE INTELLIGENCE

Avoiding Over fitting of data

- Regardless of whether the correct tree size is found by stopping early or by post-pruning, a key question is what criterion is to be used to determine the correct final tree size.
- Approaches include:
    1. Use a separate set of examples, distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree.
    2. Use all the available data for training, but apply a statistical test to estimate whether expanding (or pruning) a particular node is likely to produce an improvement beyond the training set. For example, Quinlan (1986) uses a chi-square test to estimate whether further expanding a node is likely to improve performance over the entire instance distribution, or only on the current sample of training data.
    3. Use an explicit measure of the complexity for encoding the training examples and the decision tree, halting growth of the tree when this encoding size is minimized.

MACHINE INTELLIGENCE

Dealing with continuous value

- Our initial definition of ID3 is restricted to attributes that take on a discrete set of values.
- First, the target attribute whose value is predicted by the learned tree must be discrete valued.
- Second, the attributes tested in the decision nodes of the tree must also be discrete valued
- This second restriction can easily be re- moved so that continuous-valued decision attributes can be incorporated into the learned tree.
- This can be accomplished by dynamically defining new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals.
- In particular, for an attribute A that is continuous-valued, the algorithm can dynamically create a new Boolean attribute A_c, that is true if A < c and false otherwise. The only question is how to select the best value for the threshold c.

MACHINE INTELLIGENCE

Dealing with continuous value

- As an example, suppose we wish to include the continuous-valued attribute Temperature in describing the training example days in the learning task in the given table

TEMP	40	48	60	72	80	90
Play Sport	no	no	yes	yes	yes	no

- What threshold-based Boolean attribute should be defined based on Temperature?
- Clearly, we would like to pick a threshold, c, that produces the greatest information gain.
- By sorting the examples according to the continuous attribute A, then identifying adjacent examples that differ in their target classification, we can generate a set of candidate thresholds midway between the corresponding values of A.
- It can be shown that the value of c that maximizes information gain must always lie at such a boundary
- These candidate thresholds can then be evaluated by computing the information gain associated with each.

MACHINE INTELLIGENCE

Dealing with continuous value

TEMP	40	48	60	72	80	90
Play Sport	no	no	yes	yes	yes	no

- In the current example, there are two candidate thresholds, corresponding to the values of Temperature at which the value of Play Tennis changes: (48 + 60)/2, and (80 + 90)/2.
- The information gain can then be computed for each of the candidate attributes
- $\text{Temperature}_{>54}$ and $\text{Temperature}_{>85}$, and the best can be selected $\text{Temperature}_{>54.}$
- This dynamically created Boolean attribute can then compete with the other discrete-valued candidate attributes available for growing the decision tree.
- Further extension to this can be splitting data into multiple intervals rather then just two intervals based on a single threshold.

THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701

MACHINE INTELLIGENCE

Issues in Decision Tree Learning and solutions to it

K.S.Srinivas
Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Issues in Decision Tree Learning and solutions to it

Srinivas K S.
Associate Professor, Department of Computer Science

MACHINE INTELLIGENCE

Issues in Decision Tree Learning

- The most important issues in decision tree comes at two place
- over fitting of data
- handling continuous attributes
- we will discuss each of the problem and try to find an appropriate solutions for it

MACHINE INTELLIGENCE

Over fitting of data

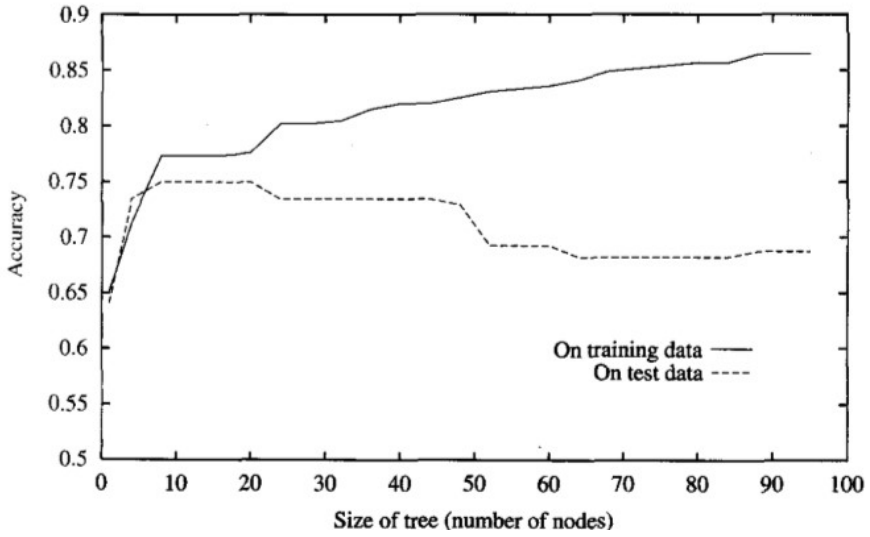
- The ID3 algorithm grows each branch of the tree just deeply enough to perfectly classify the training examples
- It can lead to difficulties when there is noise in the data, or when the number of training examples is too small to produce a representative sample of the true target function. In either of these cases, this simple algorithm can produce trees that over fits the training examples.
- We will say that a hypothesis over fits the training examples if some other hypothesis that fits the training examples less well actually performs better over the entire distribution of instances

- **Definition:**Given a hypothesis space H, a hypothesis h € H is said to over fit the training data if there exists some alternative hypothesis h' € H, such that h has smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances.

MACHINE INTELLIGENCE

Over fitting of data

- The graph illustrates the impact of over fitting in a typical application of decision tree learning.
- In this case, the ID3 algorithm is applied to the task of learning which medical patients have a form of diabetes
- The horizontal axis of this plot indicates the total number of nodes in the decision tree, as the tree is being constructed. The vertical axis indicates the accuracy of predictions made by the tree.
- The solid line shows the accuracy of the decision tree over the training examples, whereas the broken line shows accuracy measured over an independent set of test examples (not included in the training set).
- Predictably, the accuracy of the tree over the training examples increases monotonically as the tree is grown. How- ever, the accuracy measured over the independent test examples first increases, then decreases. As can be seen, once the tree size exceeds approximately 25 nodes
- further elaboration of the tree decreases its accuracy over the test examples despite increasing its accuracy on the training examples.



MACHINE INTELLIGENCE

Avoiding Over fitting of data

- One way this can occur is when the training examples contain random errors or noise.
- There are several approaches to avoiding over fitting in decision tree learning. These can be grouped into two classes:
    1. approaches that stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data,
    2. approaches that allow the tree to over fit the data, and then post-prune the tree.
- Although the first of these approaches might seem.more direct, the second approach of post-pruning over fit trees has been found to be more successful in practice.
- This is due to the difficulty in the first approach of estimating precisely when to stop growing the tree.

MACHINE INTELLIGENCE

Avoiding Over fitting of data

- Regardless of whether the correct tree size is found by stopping early or by post-pruning, a key question is what criterion is to be used to determine the correct final tree size.
- Approaches include:
    1. Use a separate set of examples, distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree.
    2. Use all the available data for training, but apply a statistical test to estimate whether expanding (or pruning) a particular node is likely to produce an improvement beyond the training set. For example, Quinlan (1986) uses a chi-square test to estimate whether further expanding a node is likely to improve performance over the entire instance distribution, or only on the current sample of training data.
    3. Use an explicit measure of the complexity for encoding the training examples and the decision tree, halting growth of the tree when this encoding size is minimized.

MACHINE INTELLIGENCE

Dealing with continuous value

- Our initial definition of ID3 is restricted to attributes that take on a discrete set of values.
- First, the target attribute whose value is predicted by the learned tree must be discrete valued.
- Second, the attributes tested in the decision nodes of the tree must also be discrete valued
- This second restriction can easily be re- moved so that continuous-valued decision attributes can be incorporated into the learned tree.
- This can be accomplished by dynamically defining new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals.
- In particular, for an attribute A that is continuous-valued, the algorithm can dynamically create a new Boolean attribute A_c, that is true if A < c and false otherwise. The only question is how to select the best value for the threshold c.

MACHINE INTELLIGENCE

Dealing with continuous value

- As an example, suppose we wish to include the continuous-valued attribute Temperature in describing the training example days in the learning task in the given table

TEMP	40	48	60	72	80	90
Play Sport	no	no	yes	yes	yes	no

- What threshold-based Boolean attribute should be defined based on Temperature?
- Clearly, we would like to pick a threshold, c, that produces the greatest information gain.
- By sorting the examples according to the continuous attribute A, then identifying adjacent examples that differ in their target classification, we can generate a set of candidate thresholds midway between the corresponding values of A.
- It can be shown that the value of c that maximizes information gain must always lie at such a boundary
- These candidate thresholds can then be evaluated by computing the information gain associated with each.

MACHINE INTELLIGENCE

Dealing with continuous value

TEMP	40	48	60	72	80	90
Play Sport	no	no	yes	yes	yes	no

- In the current example, there are two candidate thresholds, corresponding to the values of Temperature at which the value of Play Tennis changes: (48 + 60)/2, and (80 + 90)/2.
- The information gain can then be computed for each of the candidate attributes
- $\text{Temperature}_{>54}$ and $\text{Temperature}_{>85}$, and the best can be selected $\text{Temperature}_{>54.}$
- This dynamically created Boolean attribute can then compete with the other discrete-valued candidate attributes available for growing the decision tree.
- Further extension to this can be splitting data into multiple intervals rather then just two intervals based on a single threshold.

THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701