



# Unit 4 - PLANNING

# Planning and Learning

## Table of Contents



- Planning- Planning problems, Simple planning agent, Planning languages
- Blocks world ,Goal stack planning, Mean Ends Analysis
- Non-linear Planning, Conditional planning, Reactive planning
- Learning- Machine learning, Goals and Challenges of machine learning
- Learning concepts, models, Artificial neural network based learning- Back propagation, Support vector machines,
- Reinforcement learning, Adaptive learning, Multi\_agent based learning, Ensemble learning, Learning for decision making, Distributed learning, Speedup learning

# Planning



- Find a sequence of actions that achieves a given goal when executed from a given initial world state.

That is, given

- a set of operator descriptions (defining the possible primitive actions by the agent),
- an initial state description, and
- a goal state description or predicate,
- compute a plan, which is a sequence of operator instances, such that executing them in the initial state will change the world to a state satisfying the goal-state description.
- Goals are usually specified as a conjunction of subgoals to be achieved



# Planning vs. Problem Solving

- Planning and problem solving methods can often solve the same sorts of problems
- Planning is more powerful because of the representations and methods used.
- States, goals, and actions are decomposed into sets of sentences (usually in first-order logic)
- Search often proceeds through plan space rather than state space (though there are also state-space planners)
- Subgoals can be planned independently, reducing the complexity of the planning problem



# Planning Problem

Get tea, biscuits, and a book.

Given:

- Initial state: The agent is at home without tea, biscuits, book
- Goal state: The agent is at home with tea, biscuits, book



# Planning Problem

States can be represented by predicates States can be represented by predicates such as

At(x), Have(y), Sells(x, y) At(x), Have(y), Sells(x, y)

„ Actions:

- ⟨ Go(y) : Agent goes to y
  - causes At(y) to be true
- ⟨ Buy(z): Agent buys z
  - causes Have(z) to be true
- ⟨ Steal(z): Agent steals z



# Planning Problem

- Actions are given as logical descriptions of preconditions and effects.
  - This enables the planner to make direct connections between states and actions. connections between states and actions.
- The planner is free to add actions to the plan wherever they are required, rather than in an wherever they are required, rather than in an incremental way starting from the initial state. incremental way starting from the initial state.
- „Most parts of the world are independent of most other parts most other parts – hence divide & conquer hence divide & conquer works well.



# Situation Calculus

**Initial state:**

$$\text{At(Home, s0)} \wedge \neg \text{Have(Tea, s0)} \wedge \neg \text{Have(Biscuits, s0)} \wedge \neg \text{Have(Book, s0)}$$

**Goal state:**

$$\exists s \text{ At(Home, s)} \wedge \text{Have(Tea, s)} \wedge \text{Have(Biscuits, s)} \wedge \text{Have(Book, s)}$$

**Operators:**

$$\begin{aligned} \forall a,s \text{ Have(Tea, Result}(a,s)) \Leftrightarrow & [(a = \text{Buy(Tea)}) \wedge \text{At(Tea-shop, s)}]) \\ & \vee (\text{Have(Tea, s)} \wedge a \neq \text{Drop(Tea)})] \end{aligned}$$

Result(a,s) names the situation resulting from executing the action a in the situation s



# Simple Planning Agent

Use percepts to build model of current world state

IDEAL-PLANNER: Given a goal, algorithm generates plan of action

STATE-DESCRIPTION: given percept, return initial state description in format required by planner

MAKE-GOAL-QUERY: used to ask KB what next goal should be



# A Simple Planning Agent

```
function SIMPLE-PLANNING-AGENT(percept) returns an action
    static:      KB, a knowledge base (includes action descriptions)
                p, a plan (initially, NoPlan)
                t, a time counter (initially 0)
    local variables: G, a goal
                    current, a current state description
    TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
    current ← STATE-DESCRIPTION(KB, t)
    if p = NoPlan then
        G ← ASK(KB, MAKE-GOAL-QUERY(t))
        p ← IDEAL-PLANNER(current, G, KB)
    if p = NoPlan or p is empty then
        action ← NoOp
    else
        action ← FIRST(p)
        p ← REST(p)           Like popping from a stack
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t+1
return action
```



# Planning Languages

To represent planning problems we use **Artificial Intelligence planning languages** that describe environment's conditions which then lead to desired goals by generating chain of actions based on these conditions.

**STRIPS** (**ST**anford **R**esearch **I**nstitute **P**roblem **S**olver) -an action language which was a part of the first major planning system with the same name.

**ADL** (**A**ction **D**escription **L**anguage) is one of STRIPS extension which supports negative literals, quantified variables in goals (e.g.  $\exists x \text{ At}(P1, x) \wedge \text{At}(P2, x)$ ), conditional effects and disjunctions in goals (all not allowed in STRIPS)

**PDDL** (**P**lanning **D**omain **D**efinition **L**anguage). - an attempt to standardize planning languages (STRIPS, ADL and much more other representational languages)

# Planning and Learning

## Table of Contents



- Planning- Planning problems, Simple planning agent, Planning languages
- **Blocks world ,Goal stack planning, Mean Ends Analysis**
- Non-linear Planning, Conditional planning, Reactive planning
- Learning- Machine learning, Goals and Challenges of machine learning
- Learning concepts, models, Artificial neural network based learning- Back propagation, Support vector machines,
- Reinforcement learning, Adaptive learning, Multi\_agent based learning, Ensemble learning, Learning for decision making, Distributed learning, Speedup learning



# Blocks World Problem(Planning)

What is the Blocks World? -- The world consists of:

- A flat surface such as a tabletop
- An adequate set of identical blocks which are identified by letters.
- The blocks can be stacked one on one to form towers of apparently unlimited height.
- The stacking is achieved using a robot arm which has fundamental operations and states which can be assessed using logic and combined using logical operations.
- The robot can hold one block at a time and only one block can be moved at a time.



# Blocks World Problem(Planning)

We shall use the **four actions**:

**UNSTACK(A,B)**-- pick up clear block A from block B;

**STACK(A,B)**-- place block A using the arm onto clear block B;

**PICKUP(A)**-- lift clear block A with the empty arm;

**PUTDOWN(A)**-- place the held block A onto a free space on the table.

and the **five predicates**:

**ON(A,B)**-- block A is on block B.

**ONTABLE(A)**-- block A is on the table.

**CLEAR(A)**-- block A has nothing on it.

**HOLDING(A)**-- the arm holds block A.

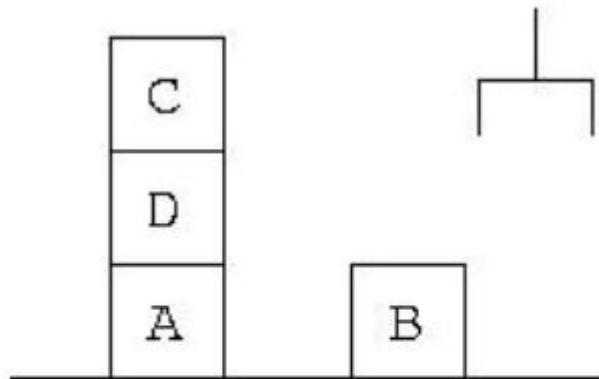
**HANDEMPTY**-- the arm holds nothing



# The Planning System Strips

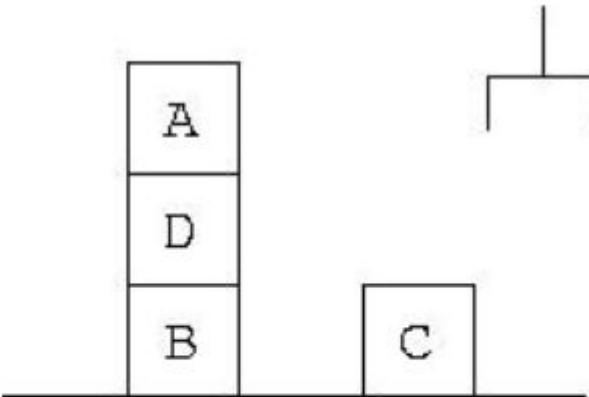
- initial state Init (given by predicate logical formulae)
- goal state goal (given by predicate logical formulae)
- set of rules (given by name, **preconditions**, **delete list**, & **add list** )

## Example: Blocks World



Ontable(A)  
Ontable(B)  
On(D,A)  
On(C,D)  
Clear(C)  
Clear(B)  
Handempty

Initial State



Ontable(B)  
Ontable(C)  
On(D,B)  
On(A,D)  
Clear(A)  
Clear(C)  
Handempty

Goal State



# The Planning System Strips

## Operators of the Blocks World

PICKUP( $x$ )

**preconditions** Clear( $x$ )

Ontable( $x$ )

Handempty

endtr

**delete list**

Clear( $x$ )

Ontable( $x$ )

Handempty

**add list**

Holds( $x$ )

PUTDOWN( $x$ )

**preconditions** Holds( $x$ )

**delete list** Holds( $x$ )

**add list** Clear( $x$ )

Ontable( $x$ )

Handempty

STACK( $x,y$ )

**preconditions** Holds( $x$ )

Clear( $y$ )

**delete list**

Holds( $x$ )

Clear( $y$ )

**add list**

Clear( $x$ )

On( $x,y$ )

Handempty

UNSTACK( $x,y$ )

**preconditions** Clear( $x$ )

On( $x,y$ )

Handempty

**delete list**

Clear( $x$ )

On( $x,y$ )

Handempty

**add list**

Holds( $x$ )

Clear( $y$ )



# The Planning System Strips

## Example Plan Execution

	UNSTACK(C,D)--->	Ontable(A) Ontable(B) On(D,A) On(C,D) Clear(C) Clear(B) Handempty	PUTDOWN(C)--->	Ontable(A) Ontable(B) On(D,A) Clear(B) Holds(C) Clear(D) Clear(C) Ontable(C) Handempty	UNSTACK(D,A)--->	Ontable(A) Ontable(B) On(D,A) Clear(B) Clear(D) Clear(C) Ontable(C) Handempty Holds(D) Clear(A)
STACK(D,B)--->	Ontable(A) Ontable(B) Clear(B) Clear(C) Ontable(C) Holds(D) Clear(A) Clear(D) On(D,B) Handempty	PICKUP(A)--->	Ontable(A) Ontable(B) Clear(C) Ontable(C) Clear(A) Clear(D) On(D,B) Handempty	STACK(A,D)--->	Ontable(B) Clear(C) Ontable(C) Clear(D) On(D,B) Holds(A) Clear(A) On(A,D) Handempty	

### Plan:

(UNSTACK(C,D), PUTDOWN(C), UNSTACK(D,A), STACK(D,B), PICKUP(A), STACK(A,D))



# Goal Stack Planning

Push the original goal to the stack. Repeat until the stack is empty:

- If stack top is a **compound goal**, push its unsatisfied subgoals to the stack.
- If stack top is a **single unsatisfied goal**, replace it by an operator that makes it satisfied and push the operator's precondition to the stack.
- If stack top is an **operator**, pop it from the stack, execute it and change the database by the operation's affects.
- If stack top is a **satisfied goal**, pop it from the stack.

# GOAL STACK PROBLEM



Problem solver relies on

- A database that describes the current situation.
- Set of operators with precondition, add and delete lists.

Let us assume that the goal to be satisfied is:

$$\text{GOAL} = G_1 \wedge G_2 \wedge \dots \wedge G_n$$

Sub-goals  $G_1, G_2, \dots, G_n$  are stacked with compound goal  $G_1 \wedge G_2 \wedge \dots \wedge G_n$  at the bottom.



At each step of problem solving process, the top goal on the stack is pursued.

# GOAL STACK PROBLEM- The Idea!



:Place goal in goal stack

**Goal1**

,Considering top Goal1  
place onto it its  
:sub-goals

**Goals1-2**

**Goals1-1**

**Goal1**

Then try to solve sub-goal  
...Goals1-2, and continue

# GOAL STACK PROBLEM - ALGORITHM



## Algorithm

- Find an operator that satisfies sub goal G1 (makes it true) and replace G1 by the operator.
- If more than one operator satisfies the sub goal then apply some heuristic to choose one.
- In order to execute the top most operation, its preconditions are added onto the stack.
- Once preconditions of an operator are satisfied, then we are guaranteed that operator can be applied to produce a new state.
- New state is obtained by using ADD and DELETE lists of an operator to the existing database.
- Problem solver keeps tract of operators applied.
- This process is continued till the goal stack is empty and problem solver returns the plan of the problem.

# GOAL STACK PROBLEM- Complete solution



```

Ontable(A)
Ontable(B)
On(D,A)
On(C,D)
Clear(C)
Clear(B)
Handempty

UNSTACK(C,D)--->
Ontable(A)
Ontable(B)
On(D,A)
On(E,D)
Clear(E)
Clear(B)
Handempty
Holds(C)
Clear(D)

PUTDOWN(C)--->
Ontable(A)
Ontable(B)
On(D,A)
Clear(B)
Holds(E)
Clear(D)
Clear(C)
Ontable(C)
Handempty
  
```

**UNSTACK(D,A)--->**

```

Ontable(A)
Ontable(B)
On(D,A)
Clear(B)
Clear(D)
Clear(C)
Ontable(C)
Handempty
Holds(D)
Clear(A)
  
```

**STACK(D,B)--->**

```

Ontable(A)
Ontable(B)
Clear(B)
Clear(C)
Ontable(C)
Holds(D)
Clear(A)
Clear(D)
On(D,B)
Handempty
  
```

**PICKUP(A)--->**

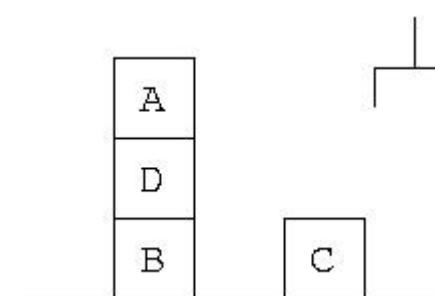
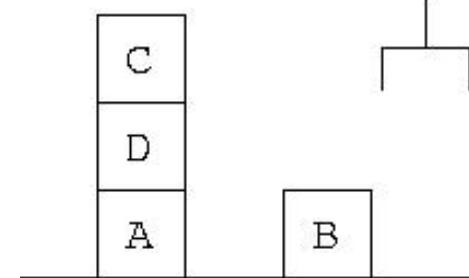
```

Ontable(A)
Ontable(B)
On(D,B)
Clear(C)
Ontable(C)
Clear(A)
Clear(D)
On(D,B)
Handempty
Holds(A)
  
```

**STACK(A,D)--->**

```

Ontable(B)
Clear(C)
Ontable(C)
Clear(D)
On(D,B)
Holds(A)
Clear(A)
On(A,D)
Handempty
  
```



(UNSTACK(C,D),PUTDOWN(C),UNSTACK(D,A),STACK(D,B),PICKUP(A),STACK(A,D))

# GOAL STACK PROBLEM- The Working!



Place on stack original .1

:goal

(On(A,C) & On(C,B)

Stack:

Database

:

(CLEAR(B  
(ON(C,A  
(CLEAR(C  
(ONTABLE(A  
(ONTABLE(B  
HANDEMPTY

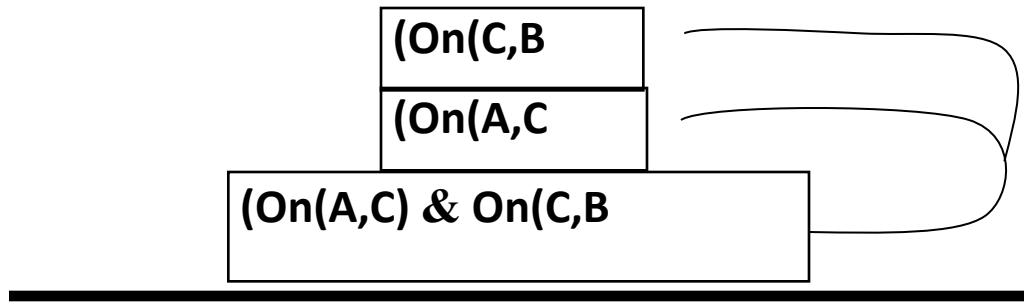
# GOAL STACK PROBLEM-

## The Working!



2. Since top goal is unsatisfied compound goal, list its unsatisfied subgoals on top of it:

:Stack



Database  
(unchanged)

:

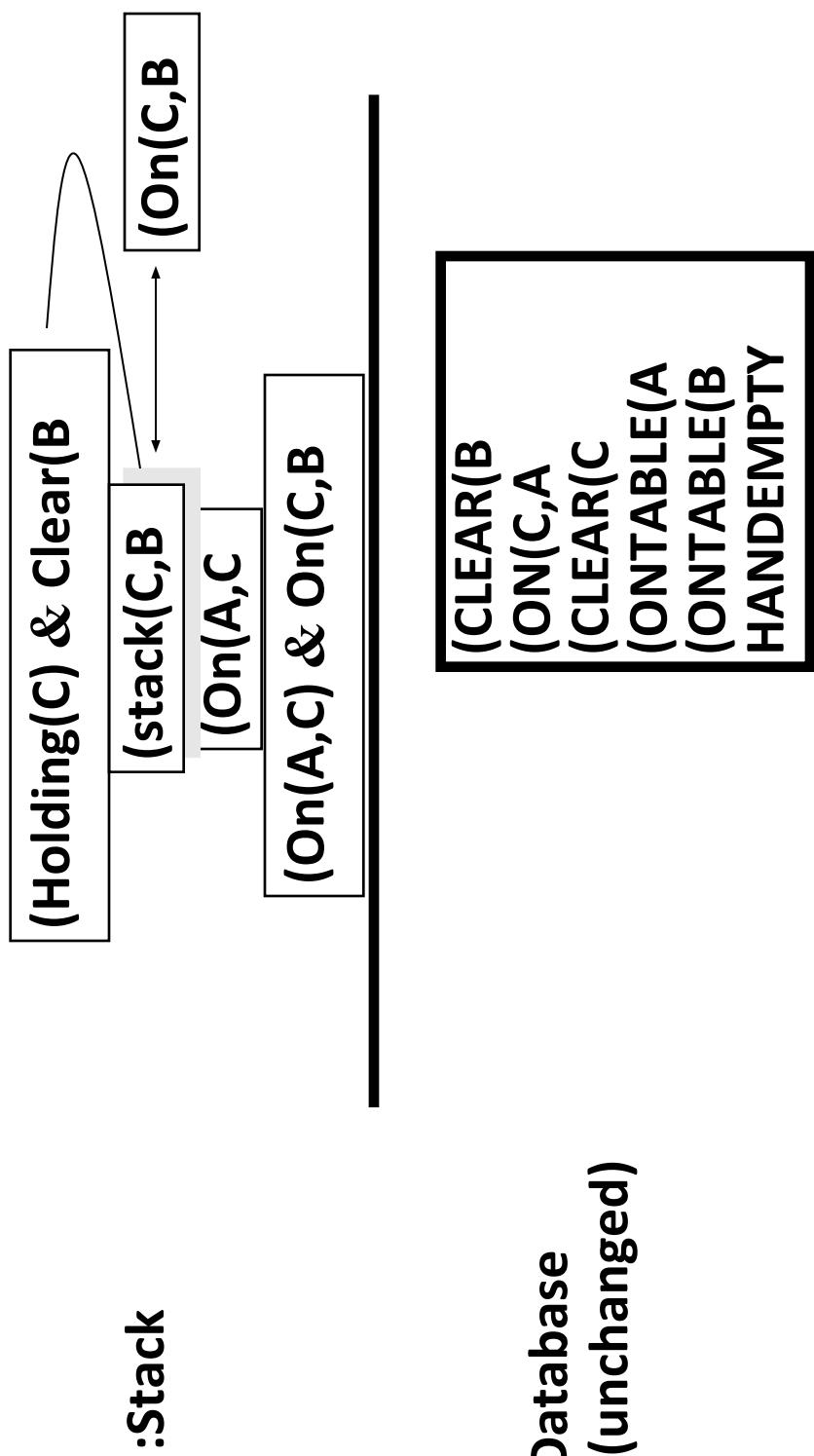
```
(CLEAR(B
(ON(C,A
(CLEAR(C
(ONTABLE(A
(ONTABLE(B
HANDEMPTY
```

# GOAL STACK PROBLEM-

## The Working!



3. Since top goal is unsatisfied single-literal goal, find rule whose instantiated add-list includes the goal, and:
  - a. Replace the goal with the instantiated rule; b. Place the rule's instantiated precondition formula on top of stack

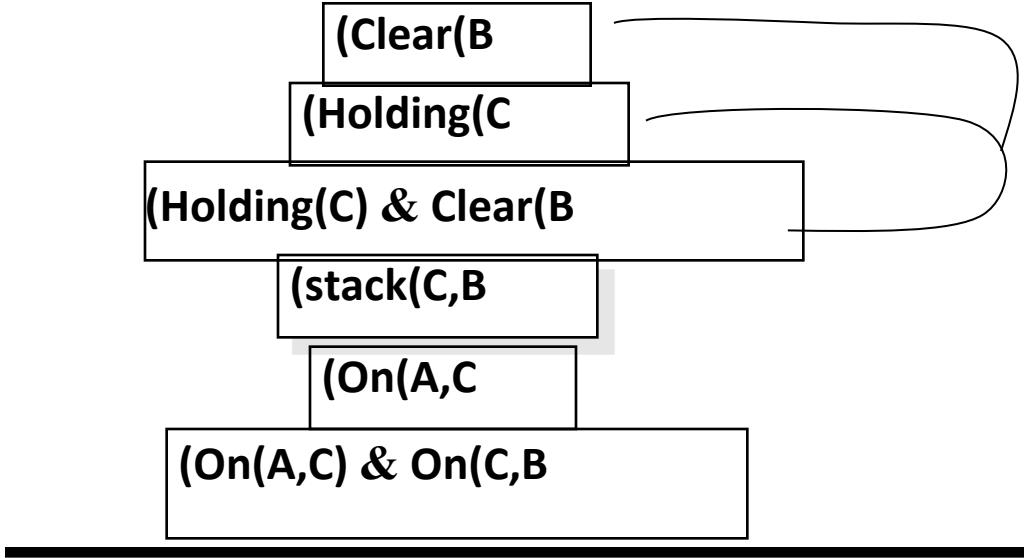


# GOAL STACK PROBLEM- The Working!



4. Since top goal is unsatisfied compound goal, list its subgoals on top of it:

:Stack



Database  
(unchanged)

**(CLEAR(B  
(ON(C,A  
(CLEAR(C  
(ONTABLE(A  
(ONTABLE(B  
HANDEMPTY**

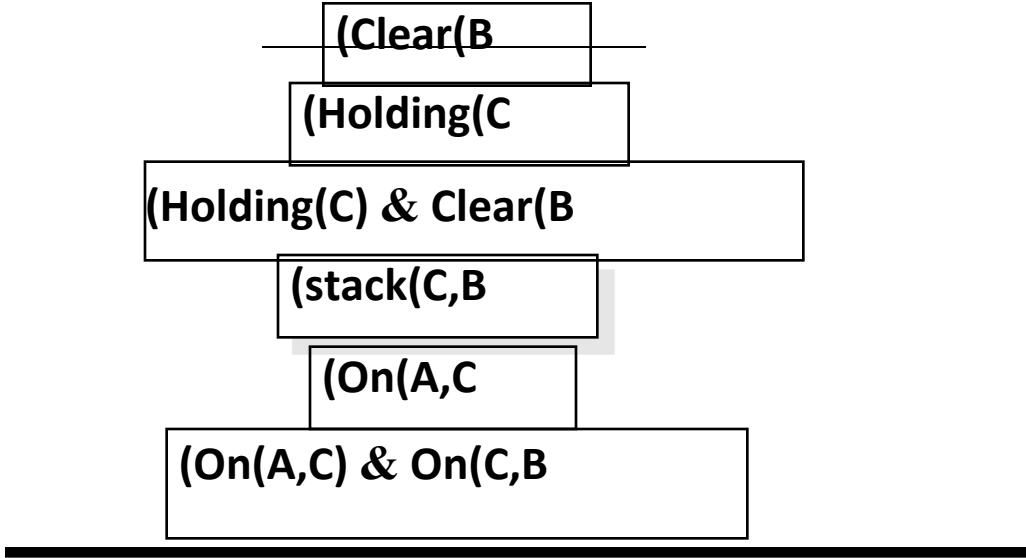
# GOAL STACK PROBLEM-

## The Working!



5. Single goal on top of stack matches data base, so remove it:

:Stack



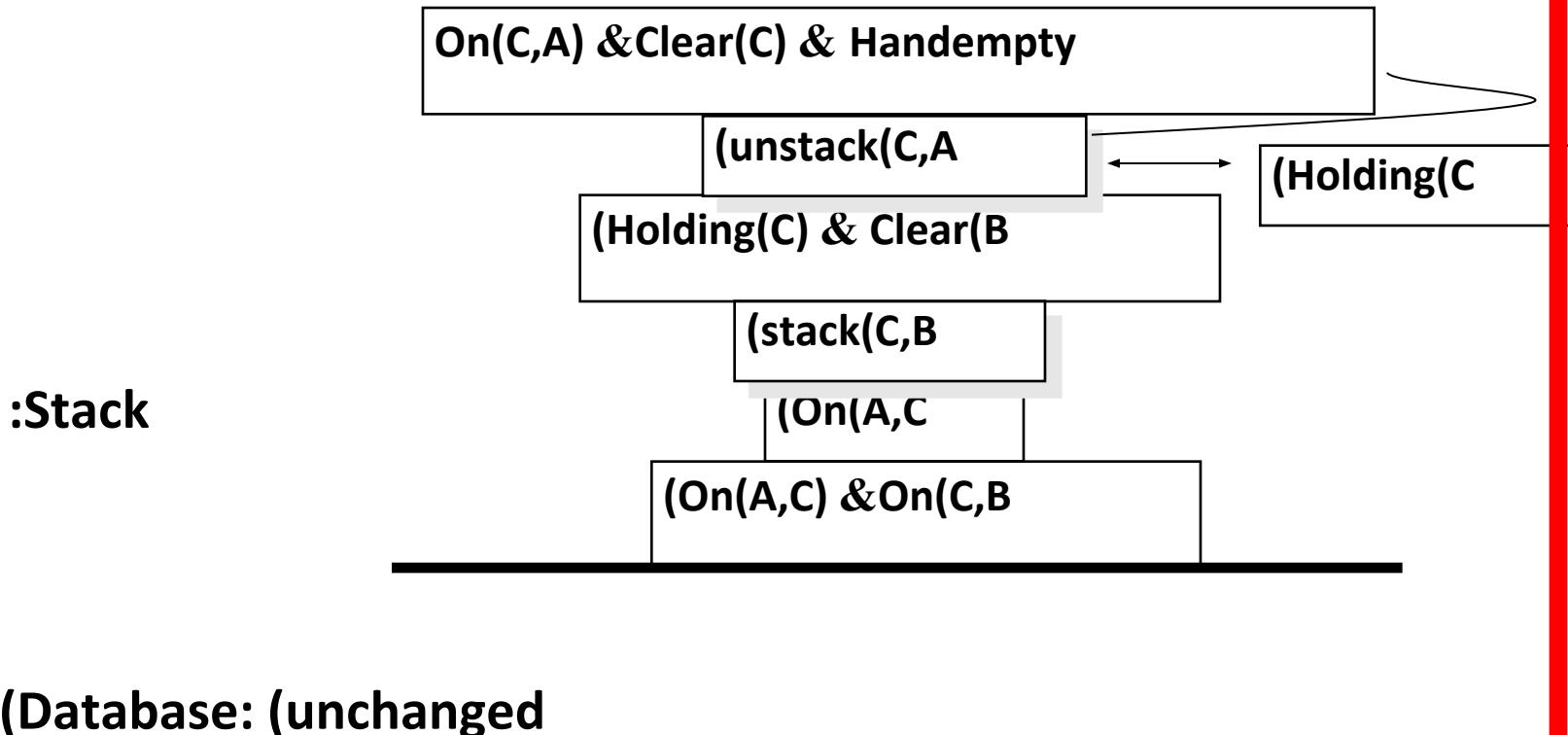
Database  
(unchanged)

(CLEAR(B  
(ON(C,A  
(CLEAR(C  
(ONTABLE(A  
(ONTABLE(B  
HANDEMPTY

# GOAL STACK PROBLEM- The Working!



6. Since top goal is unsatisfied single-literal goal, find rule whose instantiated add-list includes the goal, and: a. Replace the goal with the instantiated rule; b. Place the rule's instantiated precondition formula on top of stack



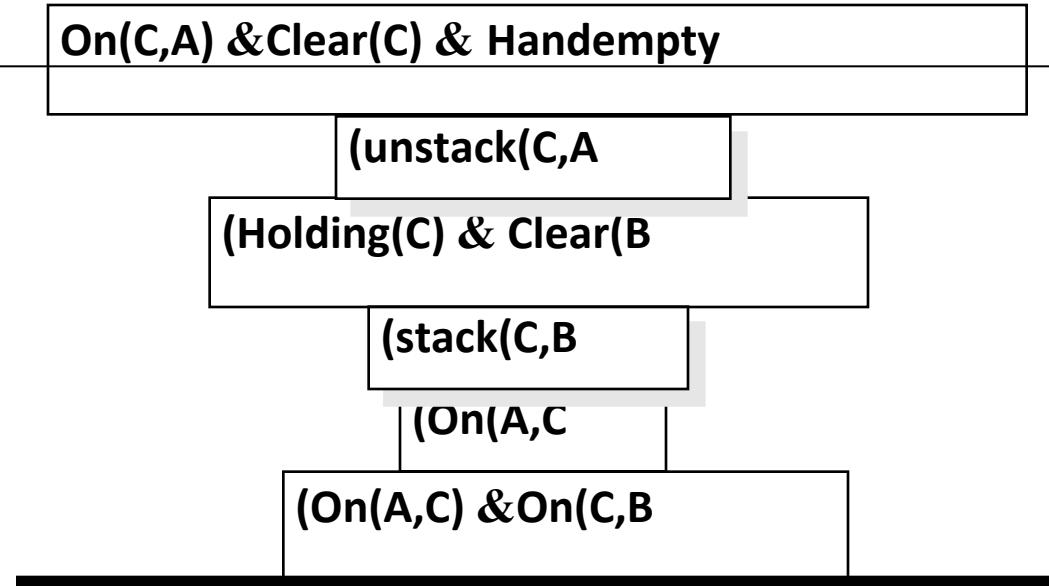
# GOAL STACK PROBLEM-

## The Working!



7. Compound goal on top of stack matches data base, so remove it:

:Stack



Database  
(unchanged)

(CLEAR(B  
(ON(C,A  
(CLEAR(C  
(ONTABLE(A  
(ONTABLE(B  
HANDEMPTY

!STEPS CONTINUES UNTIL STACK IS EMPTY

# Means - Ends Analysis



- Search strategies either reason forward or backward
- Mixed strategy - solve the major parts of problem first and solve the smaller problems that arise when combining them together.
- Such a technique is called "Means - Ends Analysis".
- The means -ends analysis process centers around finding the difference between current state and goal state.
- The problem space of means - ends analysis has
  - an initial state and one or more goal state,
  - a set of operators with a set of preconditions their application and
  - difference functions computes the difference between two states  $a(i)$  and  $s(j)$ .

# Problem Solving using Means - Ends Analysis



The algorithm for MEA consists of the following steps:

- Step 1:** Measure the current state of things by doing as is the study and capture the status at a macro level and to a possible micro level.
- Step 2:** Capture the deficiency in the current state and avenues for improvements (wish list) and define the goal state (to-be state). Define the to-be state at a macro level and to a possible micro level.
- Step 3:** Compare the Current state and Goal state and if they are at the same level the problem is resolved.
- Step 4:** List the differences between the current state and goal state at macro and micro levels.
- Step 5:** Convert the differences into deletions/modifications to current state and new additions.
- Step 6:** Define the action to implement the changes as defined in step-5.
- Step 7:** Implement the changes and measure the actual results with the planned goals.
- Step 8:** Do course correction and achieve the final goal.

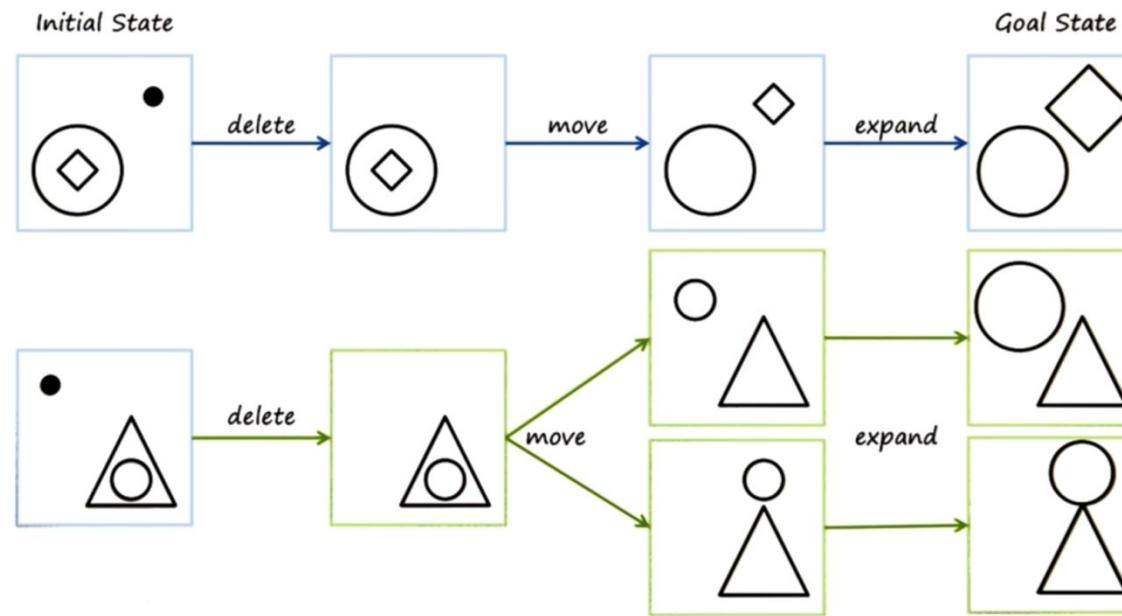
# Problem Solving using Means - Ends Analysis - Example



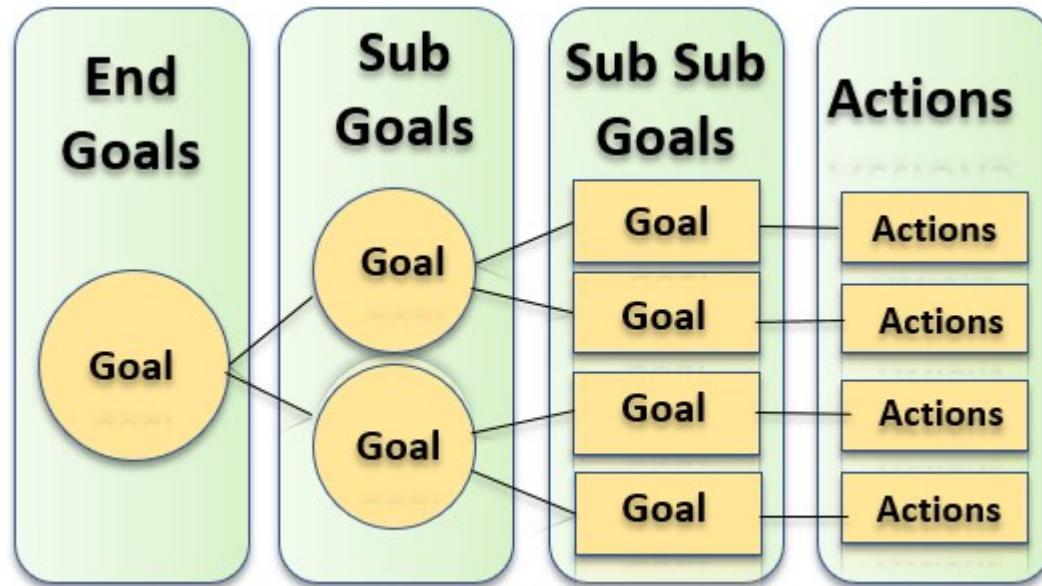
Example 1:

- If in your current state you are hungry, and in your goal state you are not hungry
- Preconditions of visit\_cafeteria: you know where the cafeteria is.  
Postconditions of visit\_cafeteria: you are no longer hungry.

Example 2:



# MEA – To do Exercise!



Example:

An electronics business company aims to get a turnover of Rs. 10 crores per annum. How do you do?

# Planning and Learning

## Table of Contents



- Planning- Planning problems, Simple planning agent, Planning languages
- Blocks world ,Goal stack planning, Mean Ends Analysis
- Non-linear Planning, Conditional planning, Reactive planning
- Learning- Machine learning, Goals and Challenges of machine learning
- Learning concepts, models, Artificial neural network based learning- Back propagation, Support vector machines,
- Reinforcement learning, Adaptive learning, Multi\_agent based learning, Ensemble learning, Learning for decision making, Distributed learning, Speedup learning



## Linear vs. Non-Linear Planning

- Goal Stack planning is **linear**: satisfies subgoals sequentially, one after another.
- **Non-linear planning**: consider interaction among subgoals.

# Partial-order planning



A linear planner builds a plan as a totally ordered sequence of plan steps

A non-linear planner (aka partial-order planner) builds up a plan as a set of steps with some temporal constraints

- constraints of the form  $S_1 < S_2$  if step  $S_1$  must come before  $S_2$ .

One refines a partially ordered plan (POP) by either:

- adding a new plan step , or
- adding a new constraint to the steps already in the plan.

A POP can be linearized (converted to a totally ordered plan) by topological sorting

# Least commitment



- Non-linear planners embody the principle of **least commitment**
  - only choose actions, orderings, and variable bindings that are absolutely necessary, leaving other decisions till later
  - avoids early commitment to decisions that don't really matter
- A linear planner always chooses to add a plan step in a particular place in the sequence
- A non-linear planner chooses to add a step and possibly some temporal constraints



# Non-linear plan

A non-linear plan consists of

- (1) A set of **steps**  $\{S_1, S_2, S_3, S_4 \dots\}$

Each step has an operator description, preconditions and post-conditions

- (2) A set of **causal links**  $\{ \dots (S_i, C, S_j) \dots \}$

Meaning a purpose of step  $S_i$  is to achieve precondition  $C$  of step  $S_j$ .

- (3) A set of ordering constraints  $\{ \dots S_i < S_j \dots \}$

if step  $S_i$  must come before step  $S_j$

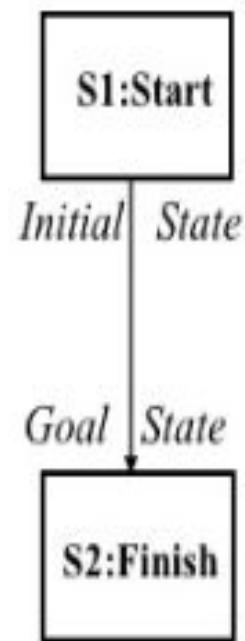
A non-linear plan **is complete** iff

- Every step mentioned in (2) and (3) is in (1)
- If  $S_i$  has prerequisite  $C$ , then there exists a causal link in (2) of the form  $(S_i, C, S_j)$  for some  $S_j$
- If  $(S_i, C, S_j)$  is in (2) and step  $S_k$  is in (1), and  $S_k$  threatens  $(S_i, C, S_j)$  (makes  $C$  false), then (3) contains either  $S_k < S_i$  or  $S_k > S_j$



# The initial plan

Every plan starts the same way



## Trivial example

Operators:

```
Op(ACTION: RightShoe, PRECOND: RightSockOn, EFFECT: RightShoeOn)  
Op(ACTION: RightSock, EFFECT: RightSockOn)  
Op(ACTION: LeftShoe, PRECOND: LeftSockOn, EFFECT: LeftShoeOn)  
Op(ACTION: LeftSock, EFFECT: leftSockOn)
```



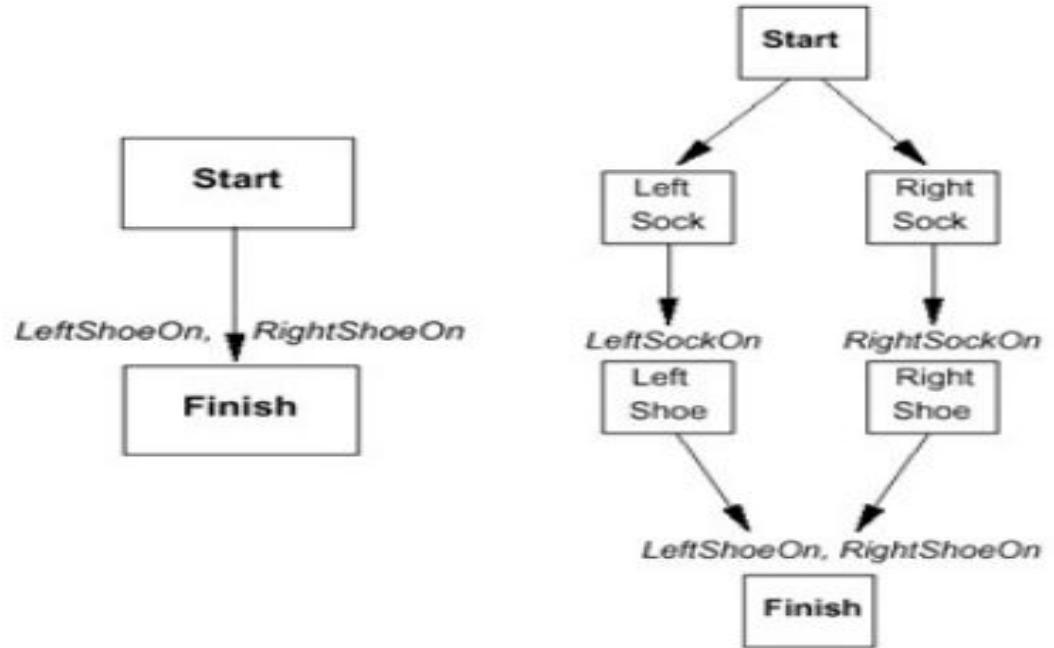
Steps: {S1:[Op(Action:Start)],  
S2:[Op(Action:Finish,  
Pre: RightShoeOn ^ LeftShoeOn)]}

Links: {}

Orderings: {S1 < S2}



# Solution



A plan is complete iff every precondition is achieved

A precondition is achieved iff it is the effect of an earlier step  
and no possibly intervening step undoes it

# Conditional Planning



- The agent might not know what the initial state is
- The agent might not know the outcome of its actions
- The plans will have branches rather than being straight line plans, includes conditional steps
  - if *<test>* then *plan A* else *plan B*
- **Full observability:** The agent knows what state it currently is, does not have to execute an observation action Simply get plans ready for all possible contingencies
- Actions sometimes fail → disjunctive effects
- Example: moving left sometimes fails
  - Action (Left, PRECOND: AtR, EFFECT: AtL  $\vee$  AtR )

# Conditional Planning



- Conditional effects: effects are conditioned on secondary preconditions

Action (Suck, PRECOND: ;,

EFFECT: (when AtL: CleanL )  $\wedge$  (when AtR: CleanR))

Actions may have both disjunctive and conditional effects: Moving sometimes dumps dirt on the destination square only when that square is clean

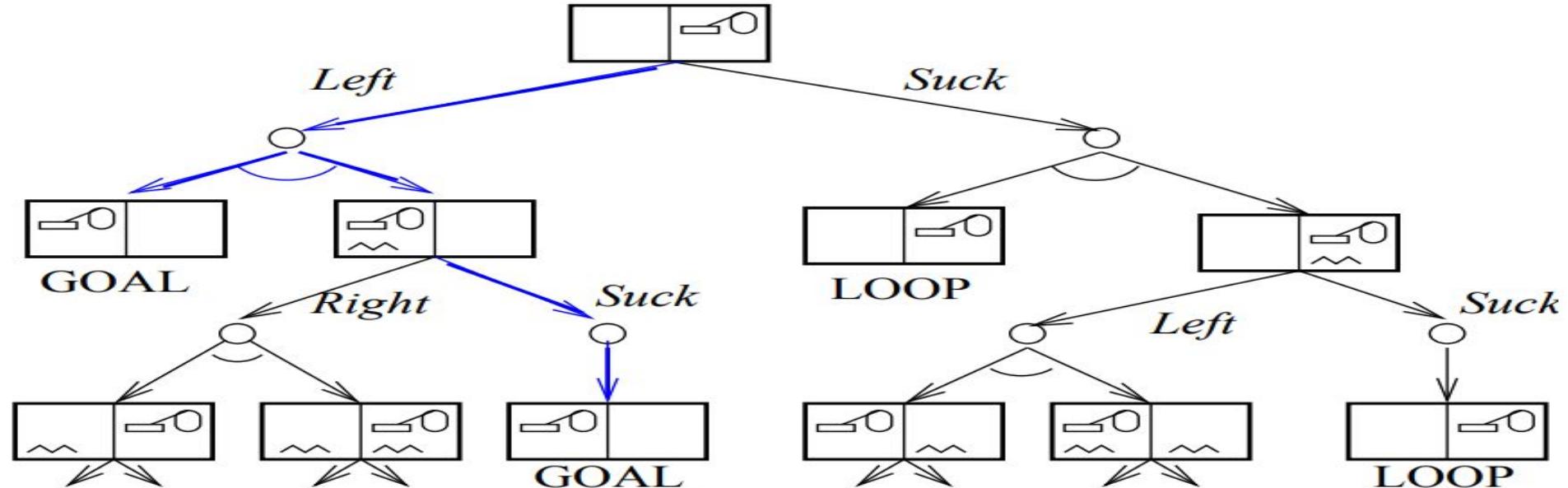
Action (Left, PRECOND: AtR;, EFFECT: AtL  $\vee$  (AtL  $\wedge$  when CleanL:  $\neg$  CleanL))

Example : The vacuum world

# Conditional Planning



## **Perform and-or search**



In the “double-Murphy” vacuum world, the plan is:

```

[ Left,
  if AtL ∧ CleanL ∧ CleanR
    then []
  else Suck
]

```

# Partially observable environments



The agent knows only a certain amount of the actual state (e.g., local sensing only, does not know about the other squares)

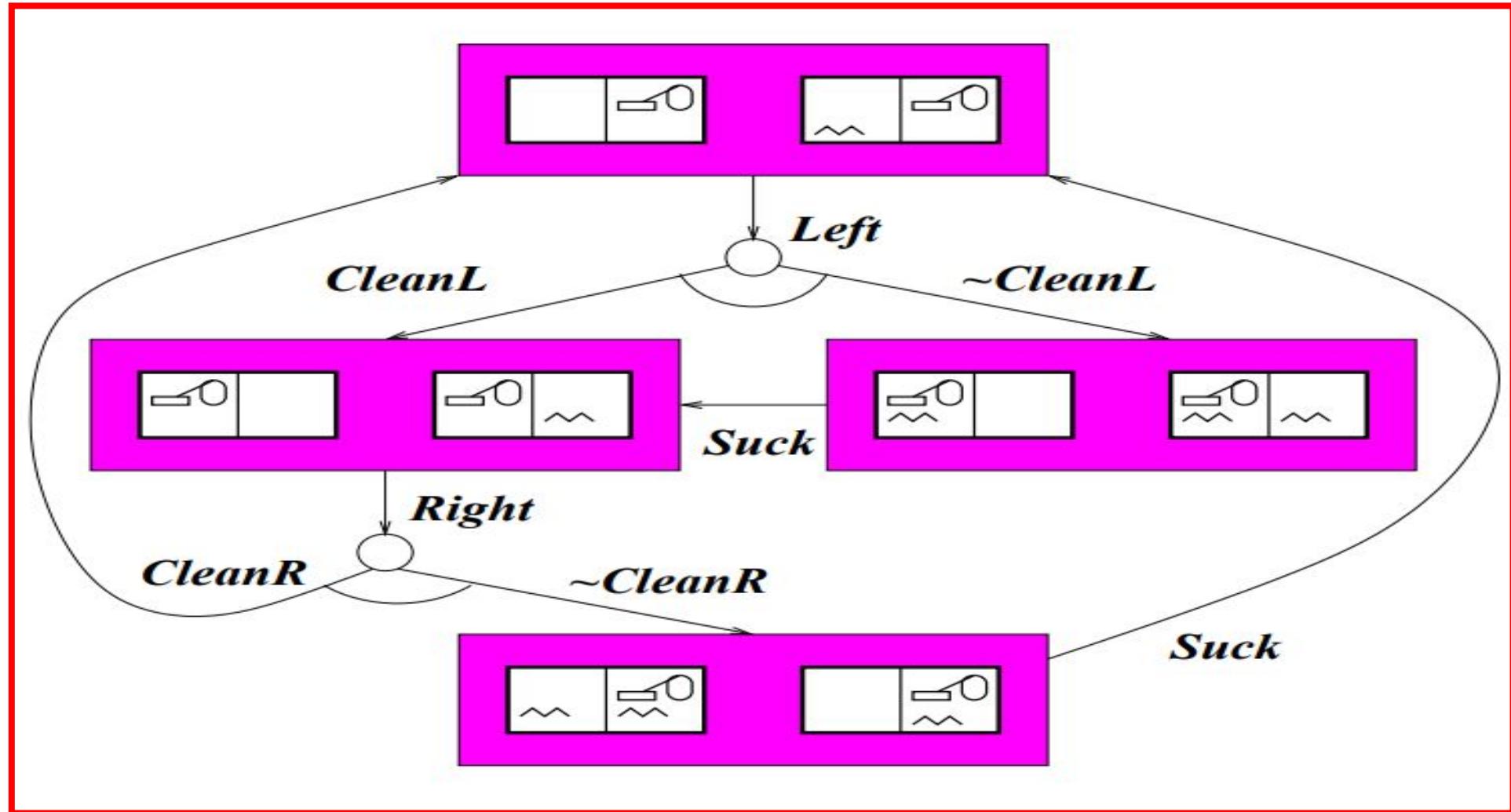
**Automatic sensing:** at every time step the agent gets all the available percepts

**Active sensing:** percepts are obtained only by executing specific sensory actions

**Belief state:** The set of possible states that the agent can be in  
“Alternate double Murphy world”: dirt can sometimes be left behind when the agent leaves a clean square



# Example –Part of the Search



# Reactive planning



- Reactive planning is planning under uncertainty.
- A group of techniques for action selection by autonomous agents.
- Differs from classical planning in two aspects:
  - operate in a timely fashion and hence can cope with highly dynamic and unpredictable environments.
  - compute just one next action in every instant, based on the current context.
- Reactive planners often (but not always) exploit reactive plans, which are stored structures describing the agent's priorities and behaviour.
- Makes use of the if-then rules.
- The rule which is at present in execution is said to be active whereas the others are inactive.

# REACTIVE PLAN REPRESENTATION



- Finite State Machines (FSMs) and Behavior trees
- Fuzzy approaches (Utility Systems)
- Connectionists approaches

# Planning and Learning

## Table of Contents



- Planning- Planning problems, Simple planning agent, Planning languages
- Blocks world ,Goal stack planning, Mean Ends Analysis
- Non-linear Planning, Conditional planning, Reactive planning
- **Learning- Machine learning, Goals and Challenges of machine learning**
- Learning concepts, models, Artificial neural network based learning- Back propagation, Support vector machines
- Reinforcement learning, Adaptive learning, Multi\_agent based learning, Ensemble learning, Learning for decision making, Distributed learning, Speedup learning

# Learning



An agent is learning if it improves its performance on future tasks after making observations about the world.

Any component of an agent can be improved by learning from data.

The improvements, and the techniques used to make them, depend on four major factors:

- Which component is to be improved.
- What prior knowledge the agent already has.
- What representation is used for the data and the component.
- What feedback is available to learn from.



# Machine Learning Defined

The oft quoted and widely accepted formal definition of machine learning as stated by field pioneer Tom M. Mitchell is:

**A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E**

## Another Definition of Machine learning :

Machine learning is a subfield of computer science, but is often also referred to as predictive analytics, or predictive modeling. Its goal and usage is to build new and/or leverage existing algorithms to learn from data, in order to build generalizable models that give accurate predictions, or to find patterns, particularly with new and unseen similar data.



# Goals of Machine Learning

The goal of ML, in simple words, is to understand the nature of (human and other forms of) learning, and to build learning capability in computers.

To be more specific, there are three aspects of the goals of ML.

- (1) To make the computers smarter, more intelligent. The more direct objective in this aspect is to develop systems (programs) for specific practical learning tasks in application domains.
- (2) To develop computational models of human learning process and perform computer simulations. The study in this aspect is also called cognitive modeling.
- (3) To explore new learning methods and develop general learning algorithms independent of applications.



# Challenges of Machine Learning

In Machine Learning, we can hope to discover ways for machine to learn which are better than ways human learn (the opportunity), and that there are amply amount of difficulties to be overcome in order to make machines learn (the challenge).

Dimension	Human Learning	Machine Learning
Speed	Slow	Slow - hope to find tricks for machine to learn fast
Ability to transfer	No copy mechanism	Easy to copy
Require repetition	Yes	Yes/No
Error-prone	Yes	Yes
Noise-tolerant	Yes	No

# Planning and Learning

## Table of Contents



- Planning- Planning problems, Simple planning agent, Planning languages
- Blocks world ,Goal stack planning, Mean Ends Analysis
- Non-linear Planning, Conditional planning, Reactive planning
- Learning- Machine learning, Goals and Challenges of machine learning
- Learning concepts, models, Artificial neural network based learning- Back propagation, Support vector machines
- Reinforcement learning, Adaptive learning, Multi\_agent based learning, Ensemble learning, Learning for decision making, Distributed learning, Speedup learning

# Learning concepts



- **Supervised Learning** - Involves learning a function that maps an input to an output based on example input-output pairs
- **Unsupervised Learning** - The machine learns from data for which the outcomes are not known. It's given input samples, but no output samples
- **Semi-Supervised Learning** - A model uses unlabeled data to gain a general sense of the data's structure, then uses a small amount of labeled data to learn how to group and organize the data as a whole. Sometimes referred to as “weak learning.”
- **Reinforcement Learning** – Allows the machine or agent to learn its behaviour based on feedback from the environment

# Supervised Learning



If shape of object is rounded and depression at top having color Red then it will be labelled as –Apple.

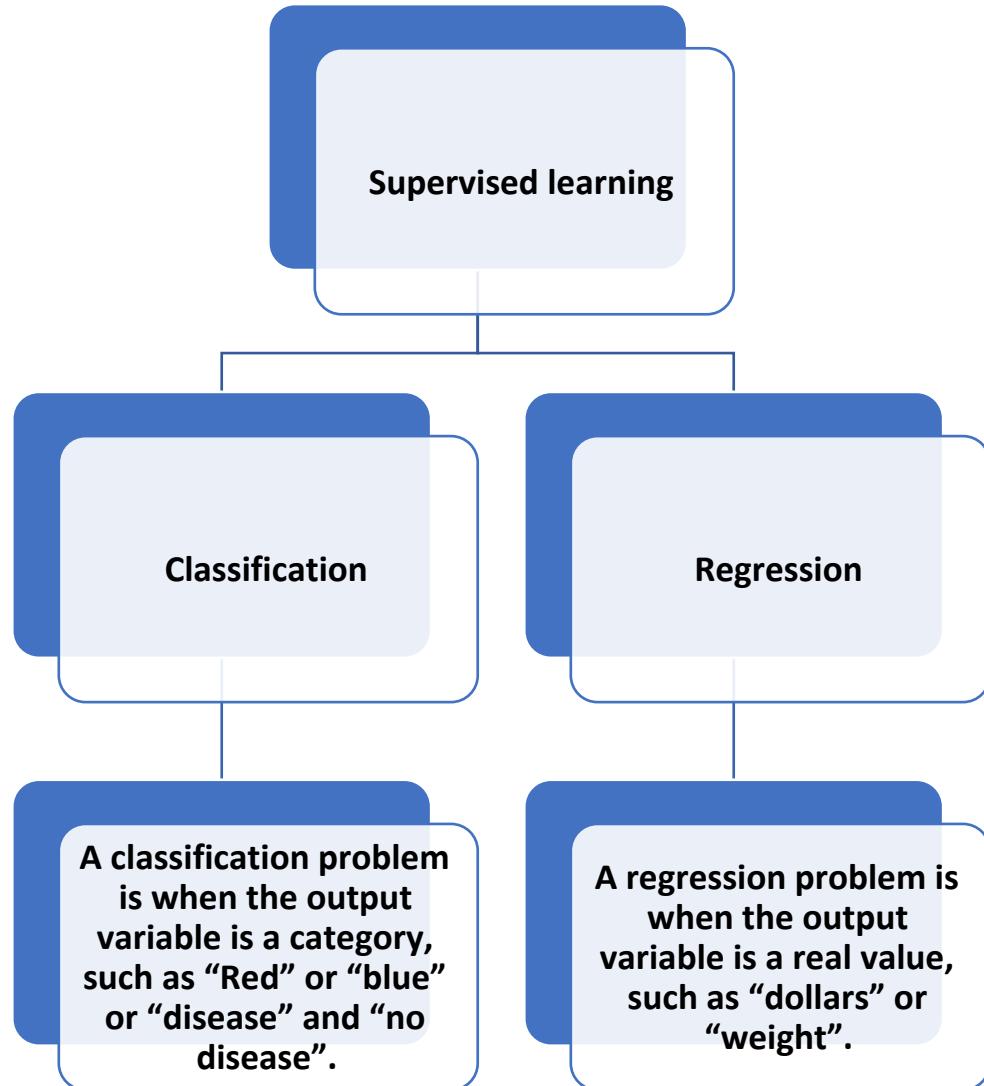


If shape of object is rounded and depression at top having color Green then it will be labelled as –Green Apple.



If shape of object is long curving cylinder having color Green-Yellow then it will be labelled as –Banana.

# Supervised Learning Contd...



# CLASSIFICATION

Which of the following are classification tasks?

- Find the gender of a person by analyzing his writing style
- Predict the price of a house based on floor area, number of rooms etc.
- Predict whether there will be abnormally heavy rainfall next year
- Predict the number of copies of a book that will be sold this month

# CLASSIFICATION

Which of the following are classification tasks?

- Find the gender of a person by analyzing his writing style
- Predict the price of a house based on floor area, number of rooms etc.
- Predict whether there will be abnormally heavy rainfall next year
- Predict the number of copies of a book that will be sold this month

# REGRESSION

Which ONE of the following are regression tasks?

- Predict the age of a person
- Predict the country from where the person comes from
- Predict whether the price of petroleum will increase tomorrow
- Predict whether a document is related to science

# REGRESSION

Which ONE of the following are regression tasks?

- Predict the age of a person
- Predict the country from where the person comes from
- Predict whether the price of petroleum will increase tomorrow
- Predict whether a document is related to science

# Un-Supervised Learning



stock.com • 22567

# Un -Supervised Learning

Which of the following are examples of unsupervised learning?

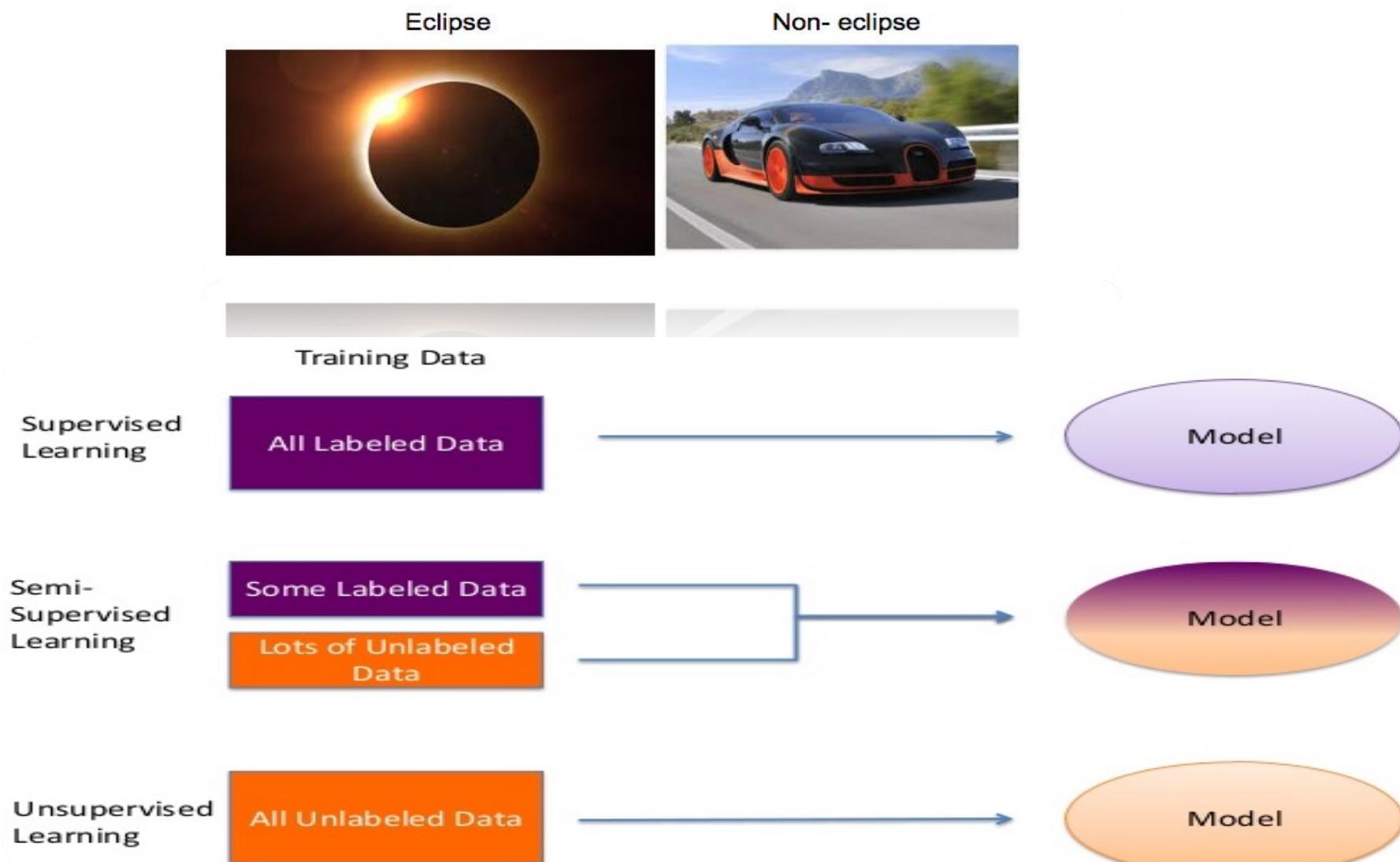
- Group news articles based on text similarity
- Make clusters of books on similar topics in a library
- Filter out spam emails
- Segment online customers into two classes based on their age group – below 25 or above 25

# Un -Supervised Learning

Which of the following are examples of unsupervised learning?

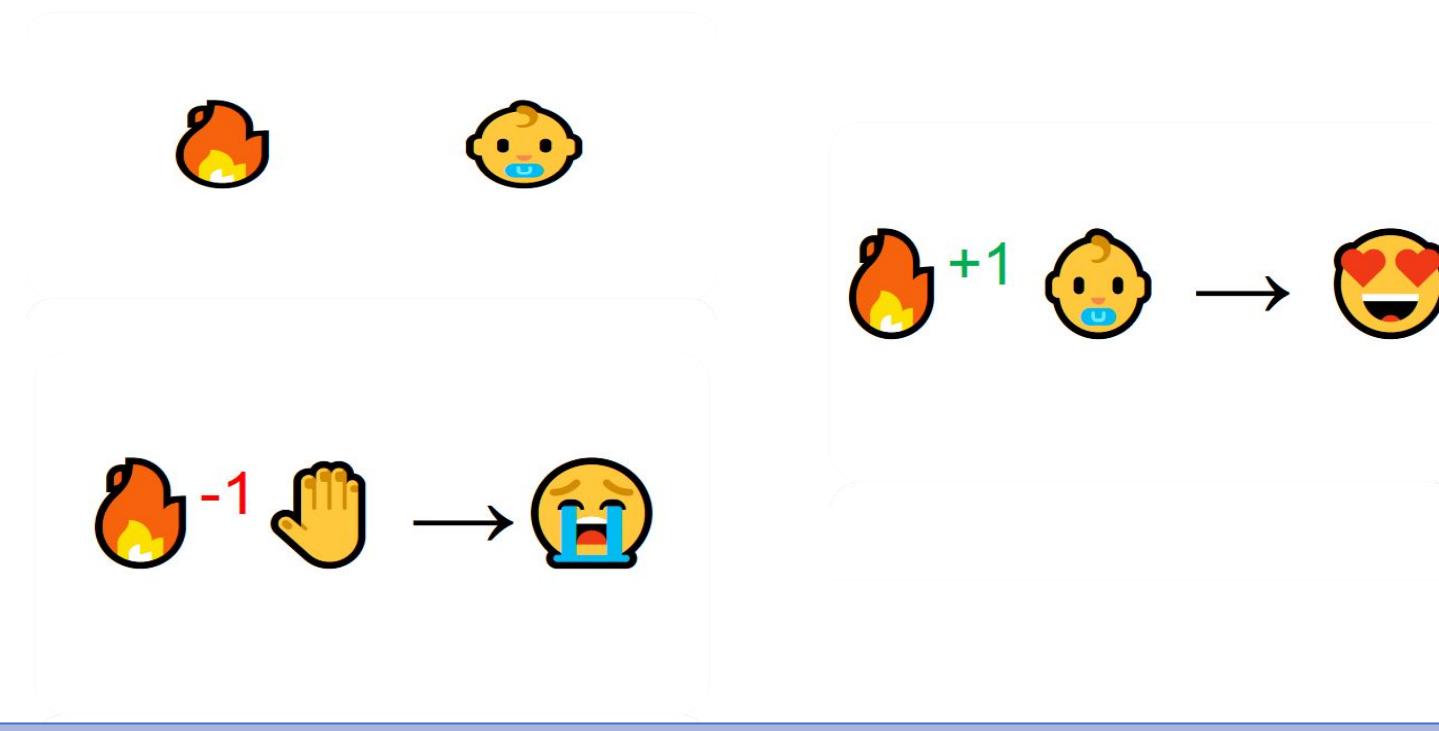
- Group news articles based on text similarity
- Make clusters of books on similar topics in a library
- Filter out spam emails
- Segment online customers into two classes based on their age group – below 25 or above 25

# Semi Supervised Learning



# Reinforcement Learning

The idea behind Reinforcement Learning is that an agent will learn from the environment by interacting with it and receiving rewards for performing actions.



That's how humans learn, through interaction. Reinforcement Learning is just a computational approach of learning from action.

# CATEGORICAL VS CONTINUOUS

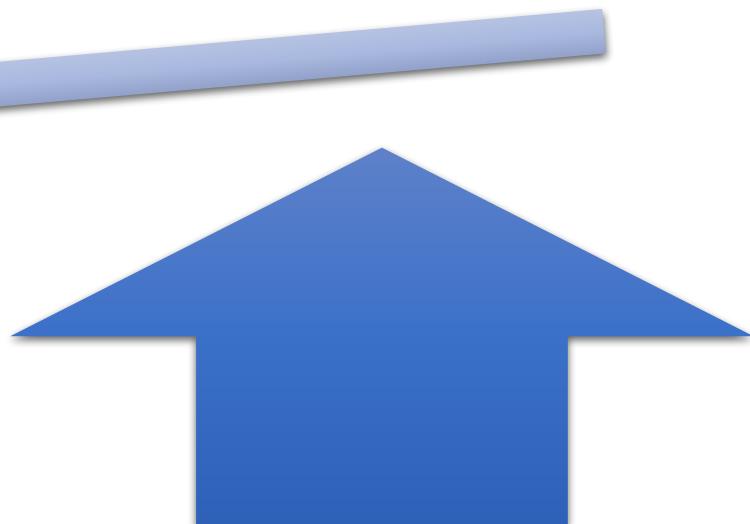


## CONTINOUS:

- Infinite number of values
- Age, height, weight, price

## CATEGORICAL:

- Finite number of categories or classes
- Gender, age group, Presence or absence of something



# Introduction to simple linear regression

Relation between variables where changes in some variables may “explain” or possibly “cause” changes in other variables.

Explanatory variables are termed the **independent** variables and the variables to be explained are termed the **dependent** variables.

Regression model estimates the nature of the relationship between the independent and dependent variables.

- Change in dependent variables that results from changes in independent variables, ie. size of the relationship.
- Strength of the relationship.
- Statistical significance of the relationship.

# Examples

## Dependent variables

Retail price of petrol  
in Chennai

Employment income

Quantity of product  
sold

## Independent variables

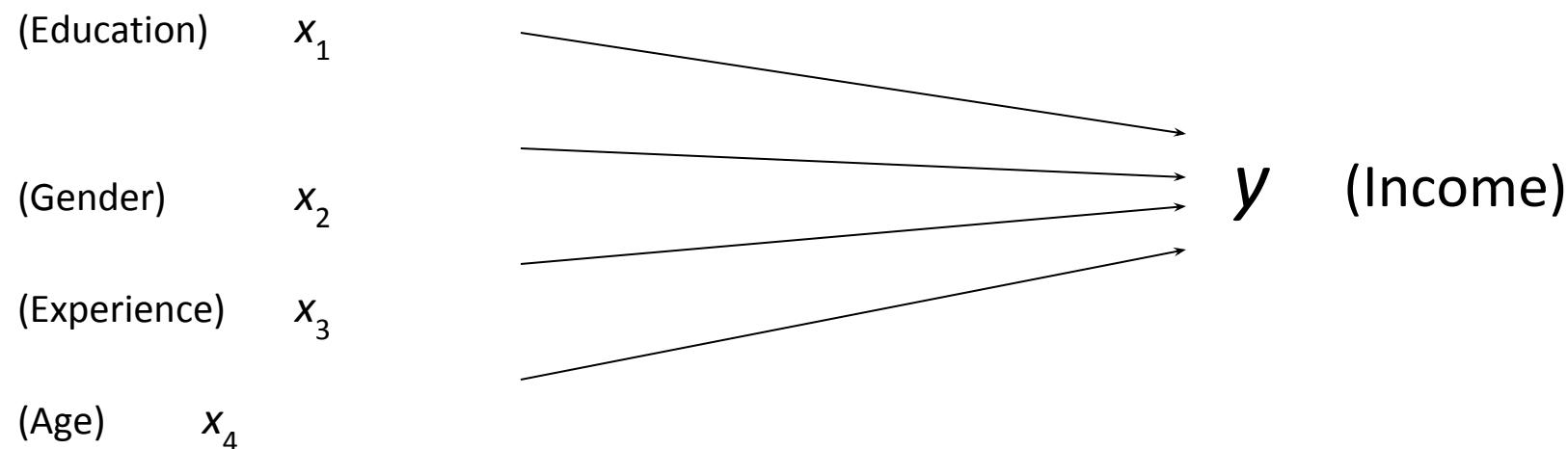
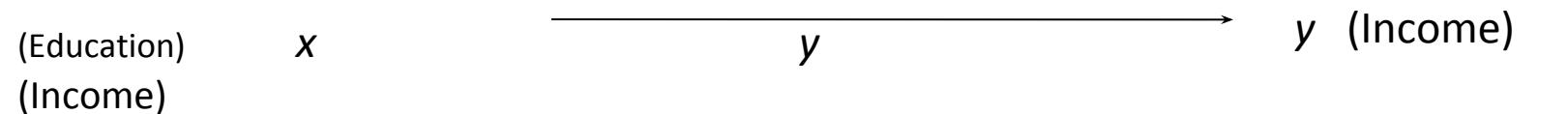
The price of crude oil.

Hours of work,  
education, occupation,  
sex, age, region, years  
of experience,  
unionization status,  
etc

Price

# Bivariate and multivariate models

## Bivariate or simple regression model



Model with simultaneous relationship

Price of wheat       $\longleftrightarrow$       Quantity of wheat produced

# Bivariate or simple linear regression

$x$  is the independent variable

$y$  is the dependent variable

The regression model is

$$y = \beta_0 + \beta_1 x + \varepsilon$$

The model has two variables, the independent or explanatory variable,  $x$ , and the dependent variable  $y$ , the variable whose variation is to be explained.

The relationship between  $x$  and  $y$  is a linear or straight line relationship.

Two parameters to estimate – the slope of the line  $\beta_1$  and the  $y$ -intercept  $\beta_0$  (where the line crosses the vertical axis).

$\varepsilon$  is the unexplained, random, or error component. Much more on this later.

# Regression line

$$y = \beta_0 + \beta_1 x + \varepsilon$$

The regression model is

- Data about  $x$  and  $y$  are obtained from a sample.
- From the sample of values of  $x$  and  $y$ , estimates  $b_0$  of  $\beta_0$  and  $b_1$  of  $\beta_1$  are obtained using the least squares or another method.

The resulting estimate of the model is

- The symbol  $\hat{y}$  is termed “ $y$  hat” and refers to the predicted values of the dependent variable  $y$  that are associated with values of  $x$ , given the linear model.

# Uses of regression

Amount of change in a dependent variable that results from changes in the independent variable(s) – can be used to estimate elasticities, returns on investment in human capital, etc.

Attempt to determine causes of phenomena.

Prediction and forecasting of sales, economic growth, etc.

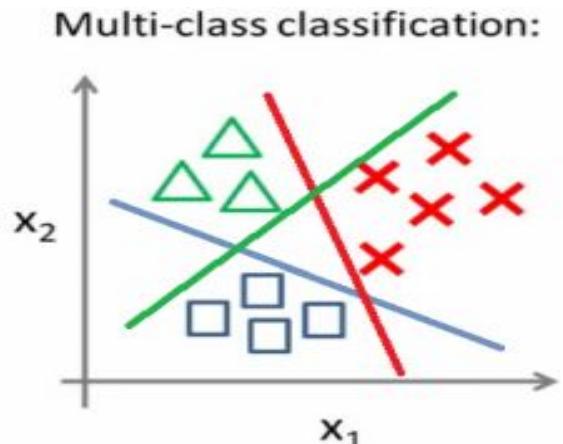
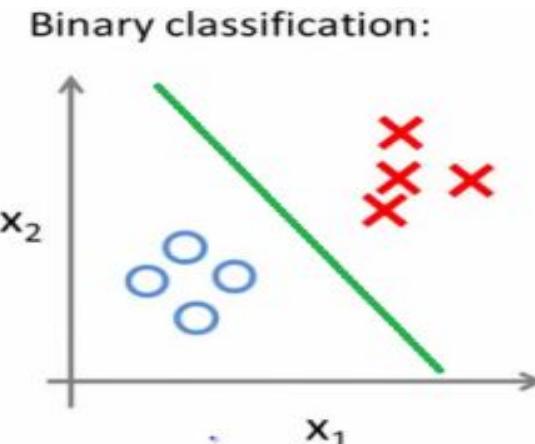
Support or negate theoretical model.

Modify and improve theoretical models and explanations of phenomena.

# Classification Techniques

## What is Classification?

It is a process of categorizing a given set of feature vectors into two or more classes.

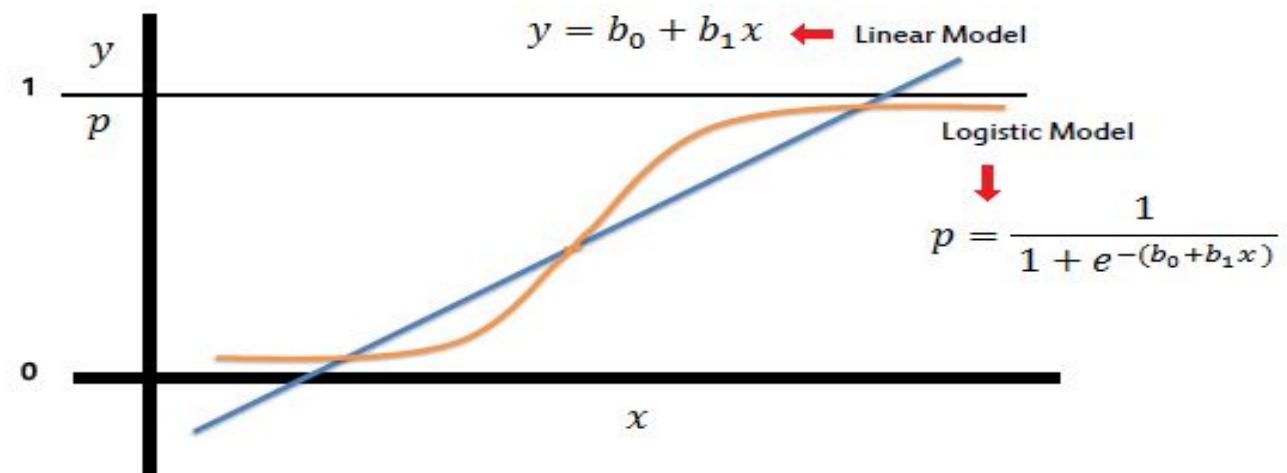


## Classification Techniques:

- Logistic Regression
- Artificial Neural Networks (ANN)
- Support Vector Machine (SVM)
- Deep Neural Network (DNN)
- Convolution Neural Network (1D-CNN and 2D-CNN)

# Logistic Regression

- **Logistic Regression** is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable.
- In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts  $P(Y=1)$  as a function of  $X$ .



# Types of Logistic Regression

1. **Binary Logistic Regression:** The categorical response has only two possible outcomes. E.g.: Spam or Not
2. **Multinomial Logistic Regression:** Three or more categories without ordering. E.g.: Predicting which food is preferred more (Veg, Non-Veg, Vegan)
3. **Ordinal Logistic Regression:** Three or more categories with ordering. E.g.: Movie rating from 1 to 5

# Math behind Binary Logistic Regression

$$z = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3$$

$$\hat{p} = \frac{e^{b_0+b_1x}}{1+e^{b_0+b_1x}} = \frac{e^z}{1+e^z}$$

$$\hat{p} = f(z)$$

## Variables

- N - number of observations
- M - number of possible class labels (dog, cat, fish)
- log - the natural logarithm
- y - a binary indicator (0 or 1) of whether class label  $c$  is the correct classification for observation  $o$
- p - the model's predicted probability that observation  $o$  is of class  $c$

## Binary Classification

In binary classification ( $M=2$ ), the formula equals:

$$-(y \log(p) + (1 - y) \log(1 - p))$$

Demo

# Clustering Techniques

## What is Clustering?

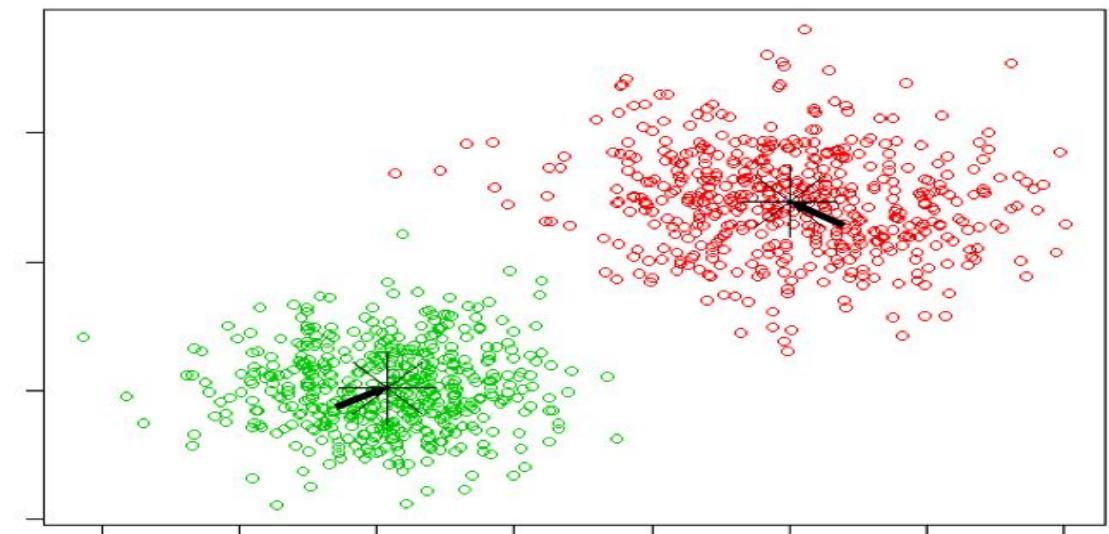
- It is a process of dividing the given set of feature vectors into two or more groups (clusters). The data points in each cluster are somehow related to each other.

### Applications of Clustering

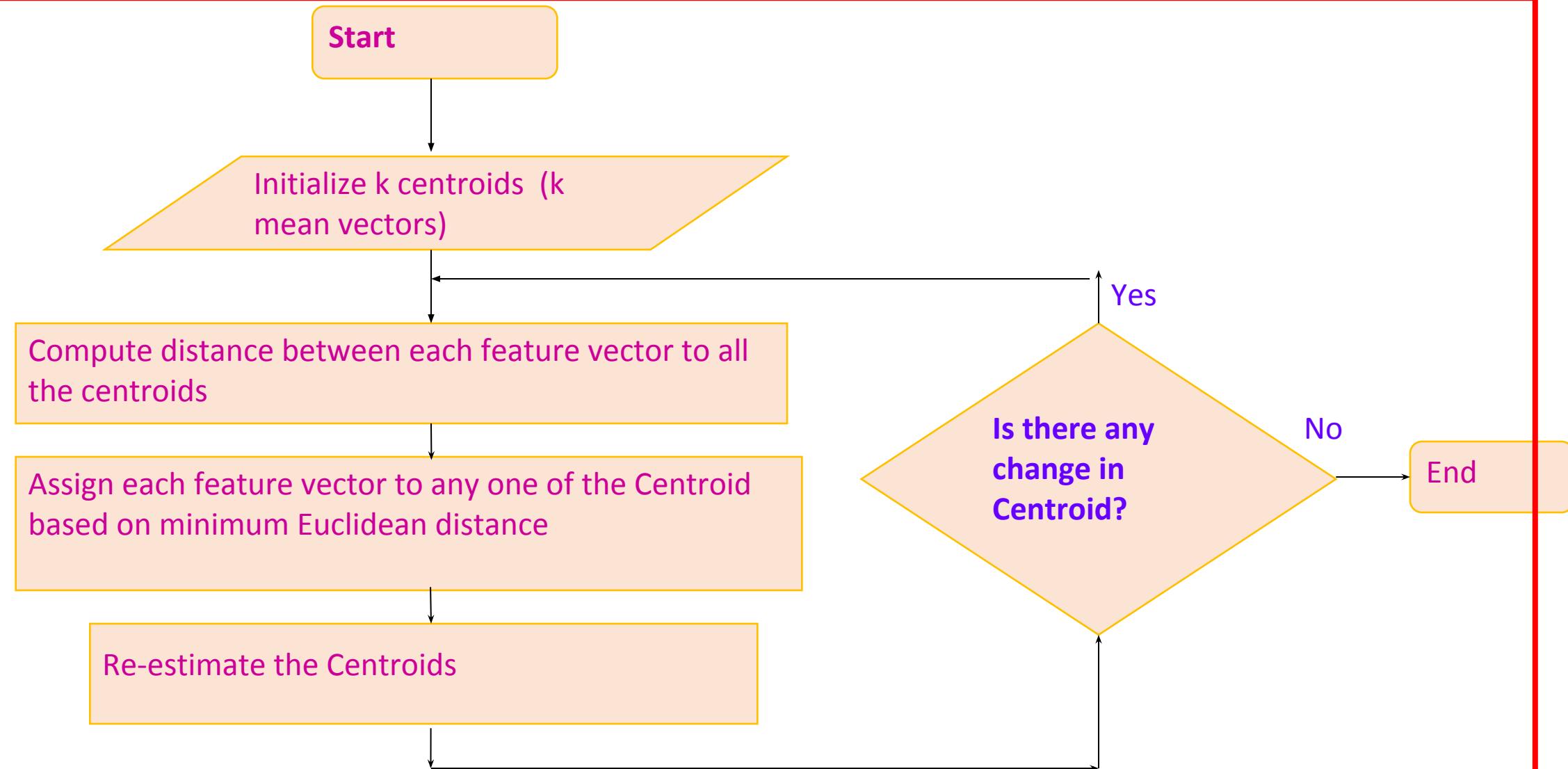
- Recommendation engines
- Market segmentation
- Social network analysis
- Search result grouping
- Image segmentation

## Clustering Techniques:

- k-means Clustering
- Hierarchical Clustering



# K-means Clustering Algorithm

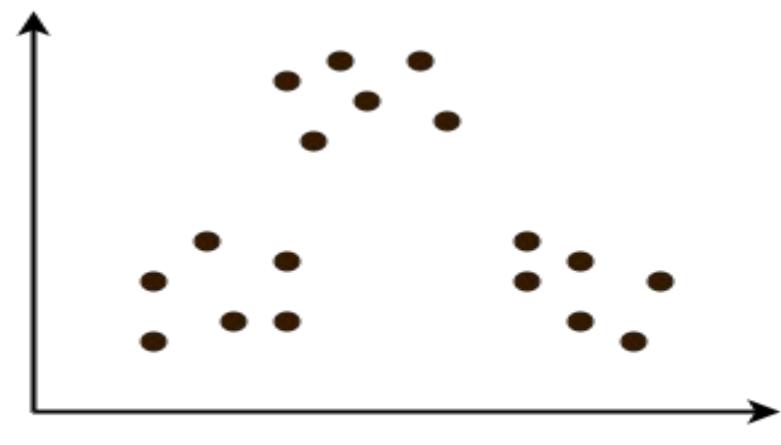


# K-means Clustering Algorithm

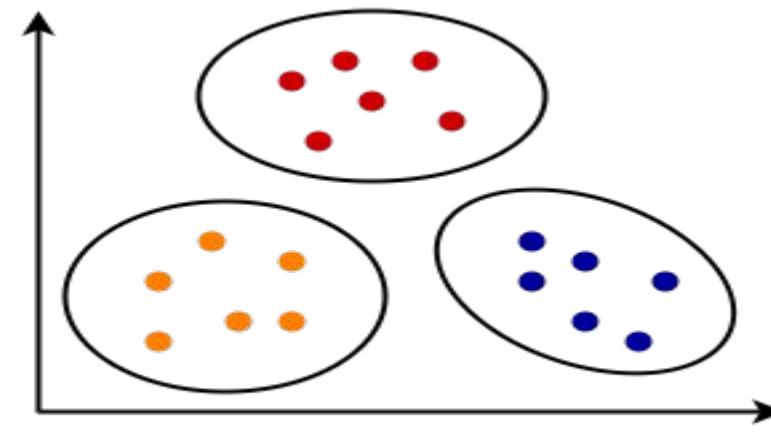
- K-Means clustering is an unsupervised iterative clustering technique.
- It partitions the given data set into k predefined distinct clusters.
- A cluster is defined as a collection of data points exhibiting certain similarities.

# K-means Clustering Algorithm

- It partitions the data set such that-
- Each data point belongs to a cluster with the nearest mean.
- Data points belonging to one cluster have high degree of similarity.
- Data points belonging to different clusters have high degree of dissimilarity.

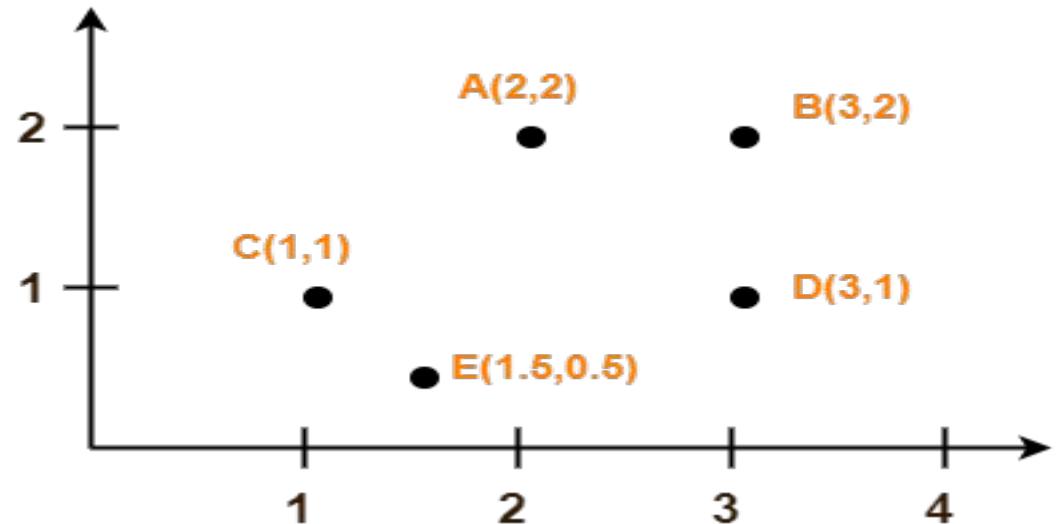


**Before K-Means**



**After K-Means**

# Use K-Means Algorithm to create two clusters



C1			C1		
A	2	2	1.5	1.5	1.166667
C1	1	1			
E	1.5	0.5			
C2			C2		
B	3	2	3	3	1.5
D	3	1			

## Demo

# Models



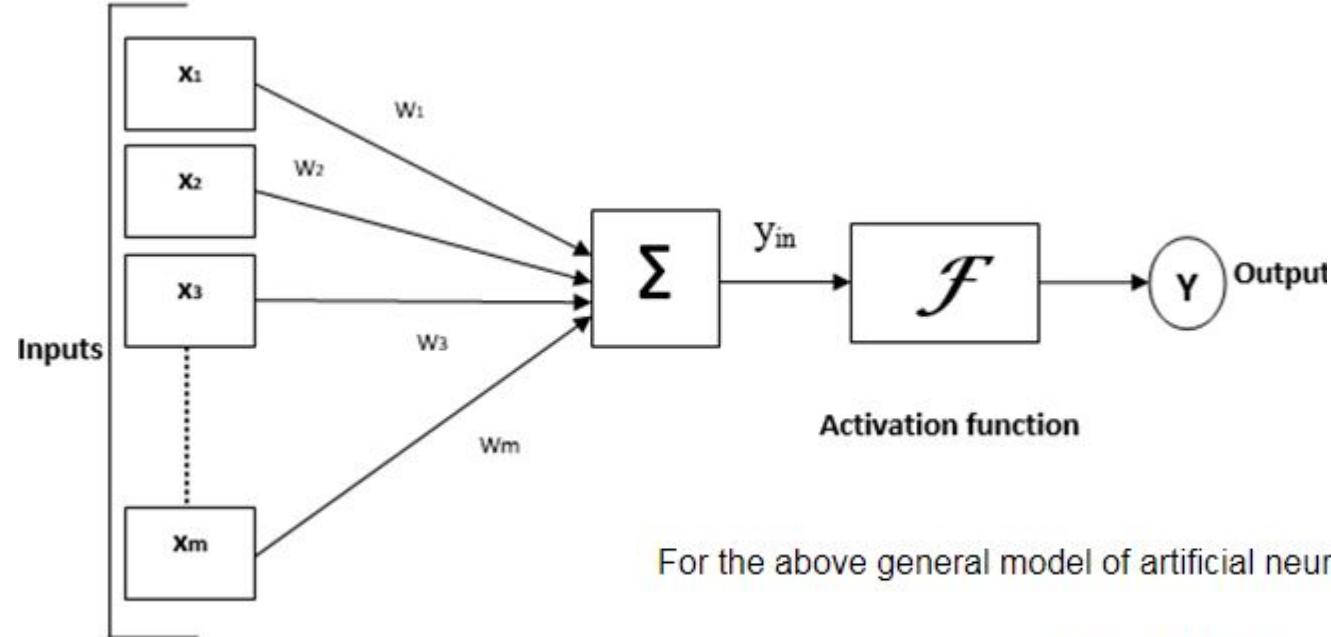
- Artificial neural networks
- Decision trees
- Support-vector machines
- Regression analysis
- Bayesian networks
- Genetic algorithms

# Artificial neural network based learning



- Artificial Neural Network ANN is an efficient computing system whose central theme is borrowed from the analogy of biological neural networks.
- ANNs are also named as “artificial neural systems” acquires a large collection of units that are interconnected in some pattern to allow communication between the units. These units, also referred to as **nodes or neurons**, are simple processors which operate in parallel.
- Every neuron is connected with other neuron through a connection link. Each connection link is associated with a weight that has information about the input signal. This is the most useful information for neurons to solve a particular problem because the weight usually excites or inhibits the signal that is being communicated.
- Each neuron has an internal state, which is called an activation signal. Output signals, which are produced after combining the input signals and activation rule, may be sent to other units.

# Model of Artificial Neural Network



For the above general model of artificial neural network, the net input can be calculated as follows –

$$\text{i.e., Net input } y_{in} = \sum_i^m x_i \cdot w_i$$

The output can be calculated by applying the activation function over the net input.

$$Y = F(y_{in})$$

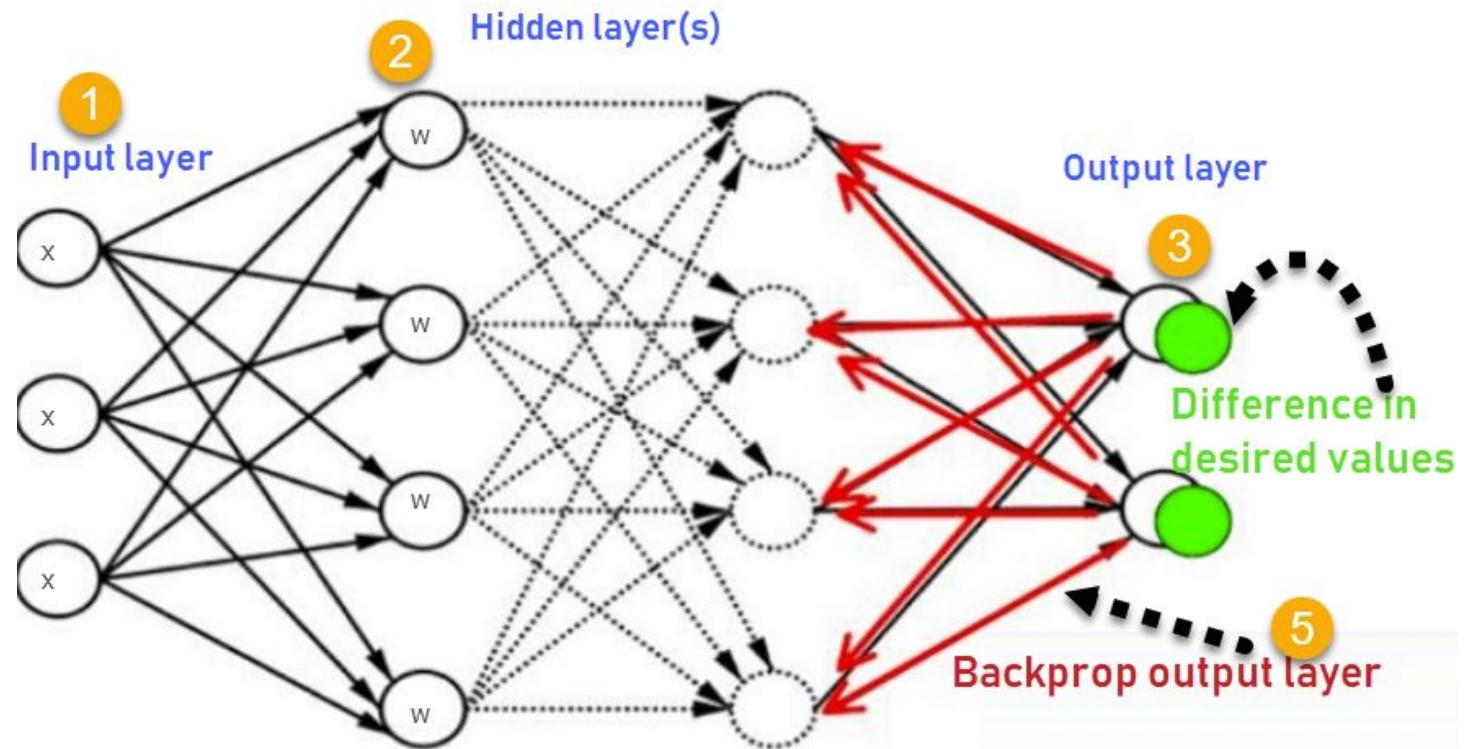


# Back Propagation Method

- Back propagation is a common method for training a neural network.
  - It is the method of fine-tuning the weights of a neural net based on the error rate obtained in the previous epoch (i.e., iteration).
  - Proper tuning of the weights allows you to reduce error rates and to make the model reliable by increasing its generalization
- **TWO STEPS:**
- **Forward pass phase:** computes ‘functional signal’, feed forward propagation of input pattern signals through network
  - **Backward pass phase:** computes ‘error signal’, propagates the error backwards through network starting at output units (where the error is the difference between actual and desired output values)



# Back Propagation Algorithm



# Back Propagation Algorithm



1. Inputs X, arrive through the preconnected path
2. Input is modeled using real weights W. The weights are usually randomly selected.
3. Calculate the output for every neuron from the input layer, to the hidden layers, to the output layer.
4. Calculate the error in the outputs

$$\text{Error}_B = \text{Actual Output} - \text{Desired Output}$$

5. Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.

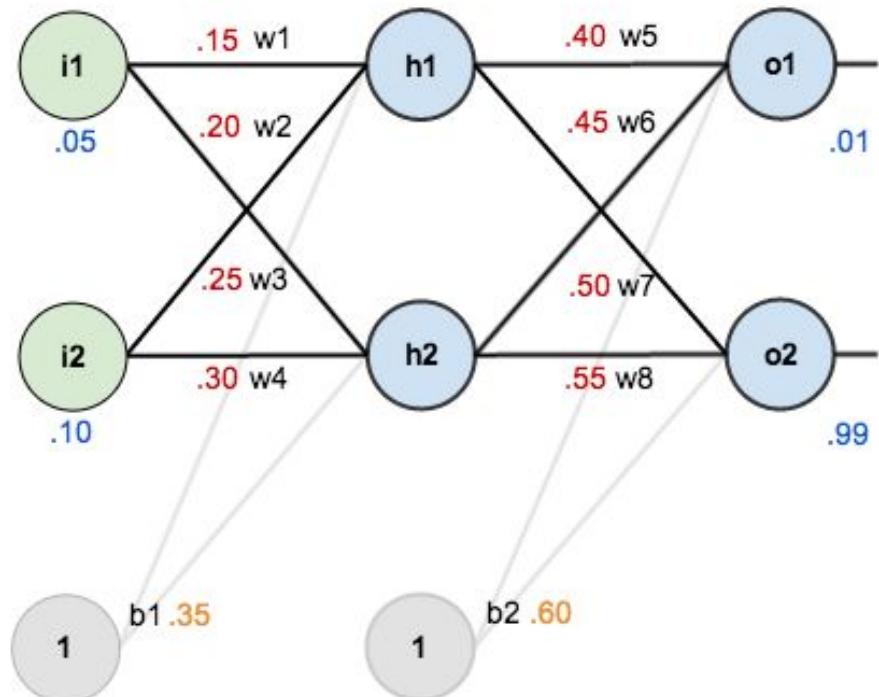
Keep repeating the process until the desired output is achieved



# Back Propagation Algorithm

A neural network with two inputs, two hidden neurons, two output neurons. Additionally, the hidden and output neurons will include a bias.

The initial weights, the biases, and training inputs/outputs are given as follows:



*The goal of back propagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs.*

we're going to work with a single training set:  
given inputs 0.05 and 0.10,  
we want the neural network to output 0.01 and  
0.99.(Expected / Desired)



# The Forward Pass

- First, we'll feed **the inputs 0.05 and 0.10** forward through the network.
- We figure out the *total net input* to each hidden layer neuron, *squash* the total net input using an *activation function* (here we use the *logistic function*), then repeat the process with the output layer neurons.

Here's how we calculate the total net input for  $h_1$ :  $net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

We then squash it using the logistic function to get the output of  $h_1$ :

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$

Carrying out the same process for  $h_2$  we get:  $out_{h2} = 0.596884378$



# The Forward Pass contd...

We repeat this process for the output layer neurons, using the output from the hidden layer neurons as inputs.

Here's the output for  $o_1$ :  $net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$

$$net_{o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$$

And carrying out the same process for  $o_2$  we get:  $out_{o2} = 0.772928465$

## Calculating the Total Error

We can now calculate the error for each output neuron using the squared error function and sum them to get the total error:

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$



# The Forward Pass contd...

the target output for  $o_1$  is 0.01 but the neural network output 0.75136507, therefore its error is:

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

Repeating this process for  $o_2$  (remembering that the target is 0.99) we get:

$$E_{o2} = 0.023560026$$

The total error for the neural network is the sum of these errors:

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$



# The Backward Pass

Our goal with back propagation is to **update each of the weights** in the network so that they cause the actual output to be closer the target output, thereby minimizing the error for each output neuron and the network as a whole.

## Output Layer

Consider  $w_5$ . We want to know how much a change in  $w_5$  affects the total error, aka  $\frac{\partial E_{total}}{\partial w_5}$ .

By applying the chain rule we know that:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

First, how much does the total error change with respect to the output?

$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$



# The Backward Pass contd...

Next, how much does the output of  $o_1$  change with respect to its total net input?

The partial derivative of the logistic function is the output multiplied by 1 minus the output:  $out_{o1} = \frac{1}{1+e^{-net_{o1}}}$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

Finally, how much does the total net input of  $o_1$  change with respect to  $w_5$ ?

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992$$

Putting it all together:  $\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$



# The Backward Pass contd...

To decrease the error, we then subtract this value from the current weight (optionally multiplied by some learning rate, eta, which we'll set to 0.5):

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$

We can repeat this process to get the new weights  $w_6$ ,  $w_7$ , and  $w_8$ :

$$w_6^+ = 0.408666186$$

$$w_7^+ = 0.511301270$$

$$w_8^+ = 0.561370121$$

We perform the actual updates in the neural network *after* we have the new weights leading into the hidden layer neurons (ie, we use the original weights, not the updated weights, when we continue the backpropagation algorithm below).



# The Backward Pass contd...

## Hidden Layer

Next, we'll continue the backwards pass by calculating new values for  $w_1, w_2, w_3$ , and  $w_4$ .

Big picture, here's what we need to figure out:  $\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$

We're going to use a similar process as we did for the output layer, but slightly different to account for the fact that the output of each hidden layer neuron contributes to the output (and therefore error) of multiple output neurons. We know that  $out_{h1}$  affects both  $out_{o1}$  and  $out_{o2}$  therefore the  $\frac{\partial E_{total}}{\partial out_{h1}}$  needs to take into consideration its effect on the both output neurons:

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

Starting with  $\frac{\partial E_{o1}}{\partial out_{h1}}$ :  $\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}}$

We can calculate  $\frac{\partial E_{o1}}{\partial net_{o1}}$  using values we calculated earlier:

$$\frac{\partial E_{o1}}{\partial net_{o1}} = \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = 0.74136507 * 0.186815602 = 0.138498562$$

And  $\frac{\partial net_{o1}}{\partial out_{h1}}$  is equal to  $w_5$ :  $net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$

$$\frac{\partial net_{o1}}{\partial out_{h1}} = w_5 = 0.40$$

Plugging them in:  $\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}} = 0.138498562 * 0.40 = 0.055399425$



# The Backward Pass contd...

Following the same process for  $\frac{\partial E_{o2}}{\partial out_{h1}}$ , we get:  $\frac{\partial E_{o2}}{\partial out_{h1}} = -0.019049119$

Therefore:  $\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}} = 0.055399425 + -0.019049119 = 0.036350306$

Now that we have  $\frac{\partial E_{total}}{\partial out_{h1}}$ , we need to figure out  $\frac{\partial out_{h1}}{\partial net_{h1}}$  and then  $\frac{\partial net_{h1}}{\partial w}$  for each weight:

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}}$$

$$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1}) = 0.59326999(1 - 0.59326999) = 0.241300709$$

We calculate the partial derivative of the total net input to  $h_1$  with respect to  $w_1$  the same as we did for the output neuron:

$$net_{h1} = w_1 * i_1 + w_3 * i_2 + b_1 * 1 \quad \frac{\partial net_{h1}}{\partial w_1} = i_1 = 0.05$$

Putting it all together:  $\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$

$$\frac{\partial E_{total}}{\partial w_1} = 0.036350306 * 0.241300709 * 0.05 = 0.000438568$$



# The Backward Pass contd...

We can now update  $w_1$ :  $w_1^+ = w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 * 0.000438568 = 0.149780716$

Repeating this for  $w_2$ ,  $w_3$ , and  $w_4$

$$w_2^+ = 0.19956143$$

$$w_3^+ = 0.24975114$$

$$w_4^+ = 0.29950229$$

Finally, we've updated all of our weights! When we fed forward the 0.05 and 0.1 inputs originally, the error on the network was 0.298371109. After this first round of backpropagation, the total error is now down to 0.291027924. It might not seem like much, but after repeating this process 10,000 times, for example, the error plummets to 0.0000351085. At this point, when we feed forward 0.05 and 0.1, the two outputs neurons generate 0.015912196 (vs 0.01 target) and 0.984065734 (vs 0.99 target).

# Planning and Learning

## Table of Contents



- Planning- Planning problems, Simple planning agent, Planning languages
- Blocks world ,Goal stack planning, Mean Ends Analysis
- Non-linear Planning, Conditional planning, Reactive planning
- Learning- Machine learning, Goals and Challenges of machine learning
- Learning concepts, models, Artificial neural network based learning- Back propagation, **Support vector machines**
- Reinforcement learning, Adaptive learning, Multi\_agent based learning, Ensemble learning, Learning for decision making, Distributed learning, Speedup learning

# Support vector machines



- Support Vector Machines is considered to be a classification approach, it but can be employed in both types of classification and regression problems.
- It can easily handle multiple continuous and categorical variables.
- SVM constructs a hyperplane in multidimensional space to separate different classes.
- SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error.
- The core idea of SVM is to find a maximum marginal hyperplane(MMH) that best divides the dataset into classes.



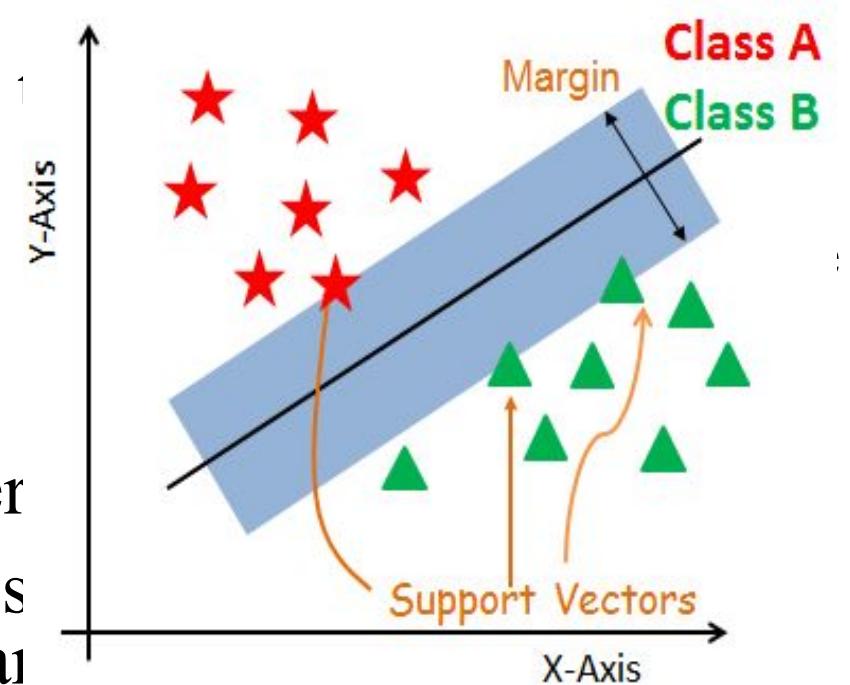
# Support vector machines

- **Support Vectors** - The data points, which are hyperplane. These points will define calculating margins.

These points are more relevant to the construction classifier

- **Hyperplane** - A decision plane which separates objects having different class member

- **Margin** - A margin is a gap between the two lines class points. This is calculated as the perpendicular support vectors or closest points. If the margin is larger in between the classes, then it is considered a good margin, a smaller margin is a bad margin.

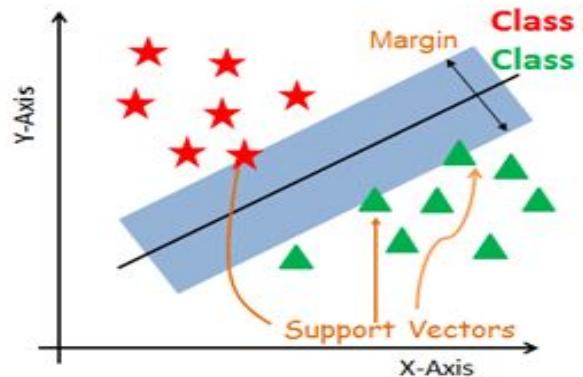
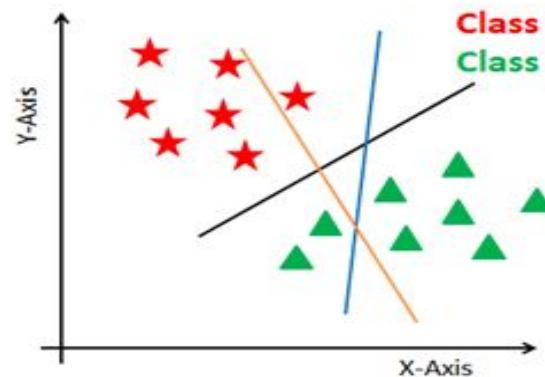


# How does SVM work?



The main objective is to segregate the given dataset in the best possible way. The distance between the either nearest points is known as the margin. The objective is to select a hyperplane with the maximum possible margin between support vectors in the given dataset. SVM searches for the maximum marginal hyperplane in the following steps:

1. Generate hyperplanes which segregates the classes in the best way. Left-hand side figure showing three hyperplanes black, blue and orange. Here, the blue and orange have higher classification error, but the black is separating the two classes correctly.
2. Select the right hyperplane with the maximum segregation from the either nearest data points as shown in the right-hand side figure.

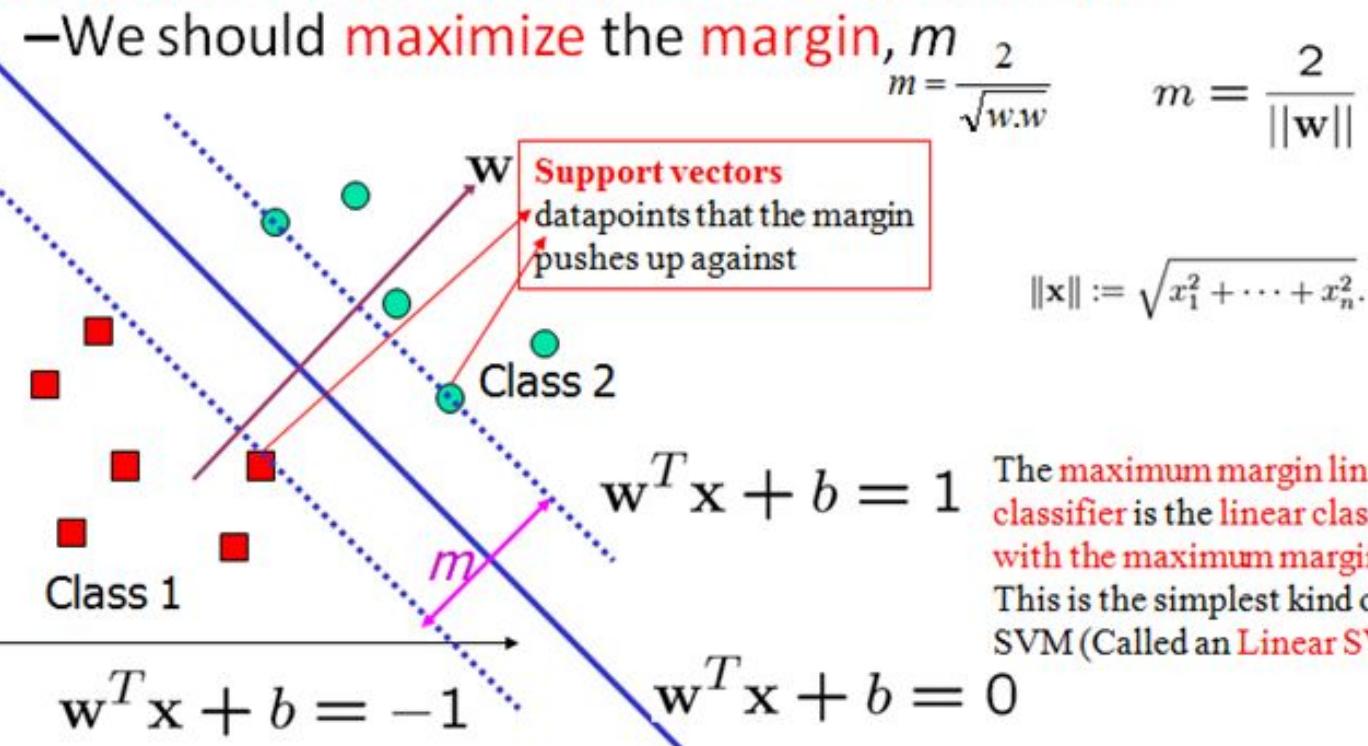




# How does SVM work?

## Good Decision Boundary: Margin Should Be Large

- The decision boundary should be **as far away from the data of both classes as possible**



The **maximum margin linear classifier** is the **linear classifier with the maximum margin**. This is the simplest kind of SVM (Called an **Linear SVM**)

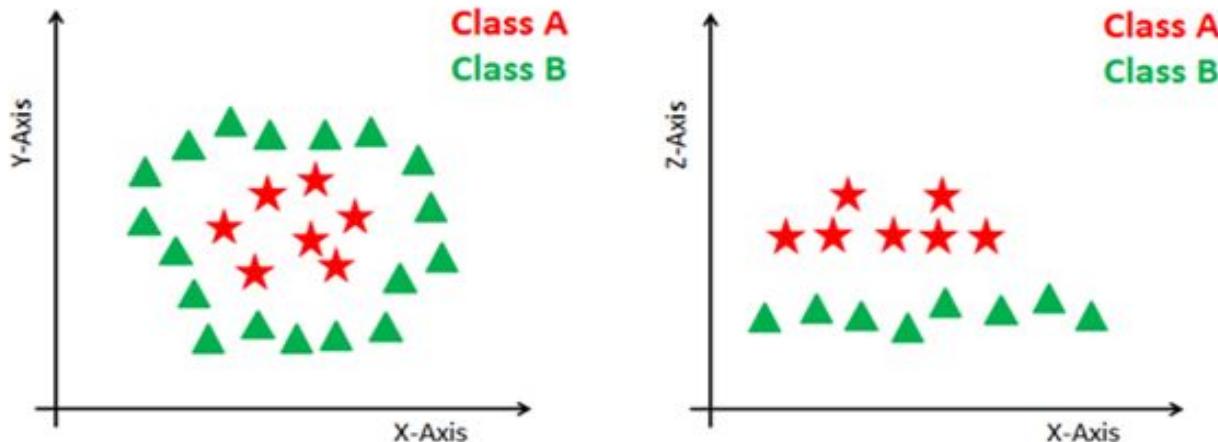


# How does SVM work?

## Dealing with non-linear and inseparable planes

Some problems can't be solved using linear hyperplane, as shown in the figure below (left-hand side).

In such situation, SVM uses a kernel trick to transform the input space to a higher dimensional space as shown on the right. The data points are plotted on the x-axis and z-axis ( $Z$  is the squared sum of both  $x$  and  $y$ :  $z=x^2+y^2$ ). Now you can easily segregate these points using linear separation.





# SVM Kernels

- The SVM algorithm is implemented in practice using a kernel. A kernel transforms an input data space into the required form. SVM uses a technique called the **kernel trick**. Here, the kernel takes a low-dimensional input space and transforms it into a higher dimensional space.(it converts non separable problem to separable problems by adding more dimension to it) It is most useful in non-linear separation problem. Kernel trick helps us to build a more accurate classifier.

## TYPES:

- Linear Kernel :** A linear kernel can be used as normal dot product any two given observations. The product between two vectors is the sum of the multiplication of each pair of input values.

$$K(x, x_i) = \text{sum}(x * x_i)$$

- Polynomial Kernel** A polynomial kernel is a more generalized form of the linear kernel. The polynomial kernel can distinguish curved or nonlinear input space.

$$K(x, x_i) = 1 + \text{sum}(x * x_i)^d$$

Where d is the degree of the polynomial. d=1 is similar to the linear transformation. The degree needs to be manually specified in the learning algorithm.



# SVM Kernels

- **Radial Basis Function Kernel** The Radial basis function kernel is a popular kernel function commonly used in support vector machine classification. RBF can map an input space in infinite dimensional space.

$$K(x, x_i) = \exp(-\gamma * \sum((x - x_i)^2))$$



# SVM -Linear Example

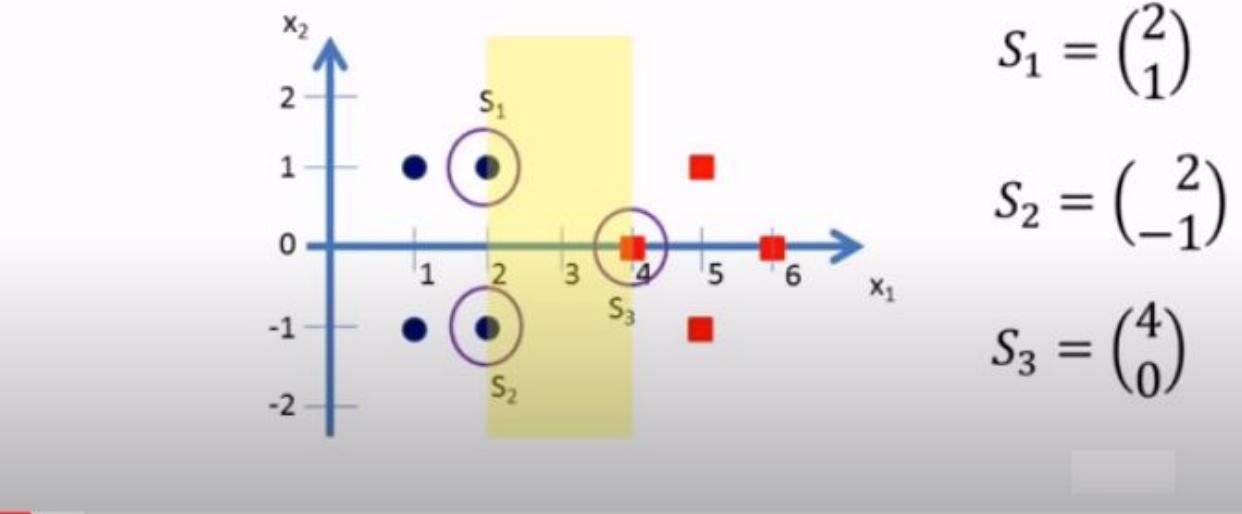
Suppose we are given the following positively labeled data points

$$\left\{ \begin{pmatrix} 4 \\ 0 \end{pmatrix}, \begin{pmatrix} 5 \\ 1 \end{pmatrix}, \begin{pmatrix} 5 \\ -1 \end{pmatrix}, \begin{pmatrix} 6 \\ 0 \end{pmatrix} \right\}$$

and the following negatively labeled data points

$$\left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \begin{pmatrix} 2 \\ -1 \end{pmatrix} \right\}$$

- Here we select 3 Support Vectors to start with.
- They are  $S_1, S_2$  and  $S_3$ .





# SVM -Linear Example

- Here we will use vectors augmented with a 1 as a bias input, and for clarity we will differentiate these with an over-tilde.  
That is:

$$S_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

$$S_2 = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$$

$$S_3 = \begin{pmatrix} 4 \\ 0 \end{pmatrix}$$

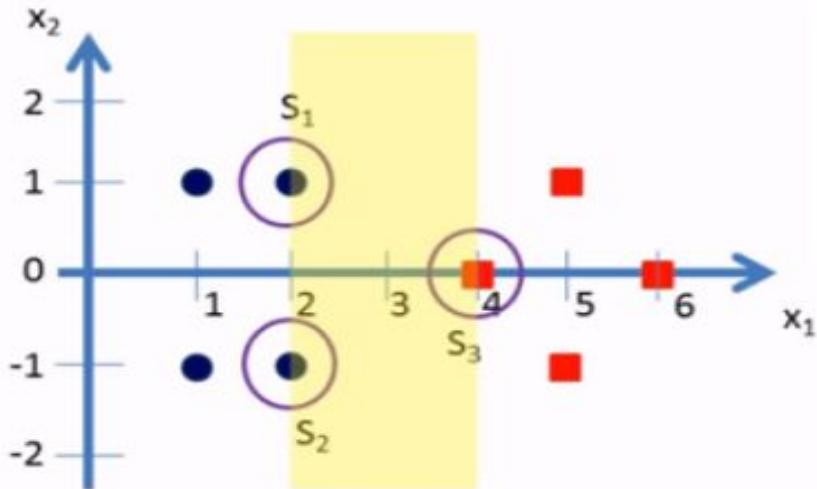
$$\widetilde{S}_1 = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$$

$$\widetilde{S}_2 = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}$$

$$\widetilde{S}_3 = \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$$



# SVM -Linear Example



- Now we need to find 3 parameters  $\alpha_1, \alpha_2$ , and  $\alpha_3$  based on the following 3 linear equations:

$$\alpha_1 \tilde{S_1} \cdot \tilde{S_1} + \alpha_2 \tilde{S_2} \cdot \tilde{S_1} + \alpha_3 \tilde{S_3} \cdot \tilde{S_1} = -1 \quad (-ve\ class)$$

$$\alpha_1 \tilde{S_1} \cdot \tilde{S_2} + \alpha_2 \tilde{S_2} \cdot \tilde{S_2} + \alpha_3 \tilde{S_3} \cdot \tilde{S_2} = -1 \quad (-ve\ class)$$

$$\alpha_1 \tilde{S_1} \cdot \tilde{S_3} + \alpha_2 \tilde{S_2} \cdot \tilde{S_3} + \alpha_3 \tilde{S_3} \cdot \tilde{S_3} = +1 \quad (+ve\ class)$$



# SVM -Linear Example

- Let's substitute the values for  $\tilde{S}_1$ ,  $\tilde{S}_2$  and  $\tilde{S}_3$  in the above equations.

$$\tilde{S}_1 = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \quad \tilde{S}_2 = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \quad \tilde{S}_3 = \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$$

$$\alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} = -1$$

$$\alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} = -1$$

$$\alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} = +1$$

# SVM -Linear Example



- After simplification we get:

$$6\alpha_1 + 4\alpha_2 + 9\alpha_3 = -1$$

$$4\alpha_1 + 6\alpha_2 + 9\alpha_3 = -1$$

$$9\alpha_1 + 9\alpha_2 + 17\alpha_3 = +1$$

- Simplifying the above 3 simultaneous equations we get:  $\alpha_1=\alpha_2=-3.25$  and  $\alpha_3=3.5$ .



# SVM -Linear Example

- The hyper plane that discriminates the positive class from the negative class is given by:

$$\tilde{w} = \sum_i \alpha_i \tilde{s}_i$$

- Substituting the values we get:

$$\tilde{w} = \alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$$

$$\tilde{w} = (-3.25) \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + (-3.25) \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + (3.5) \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -3 \end{pmatrix}$$

# SVM -Linear Example



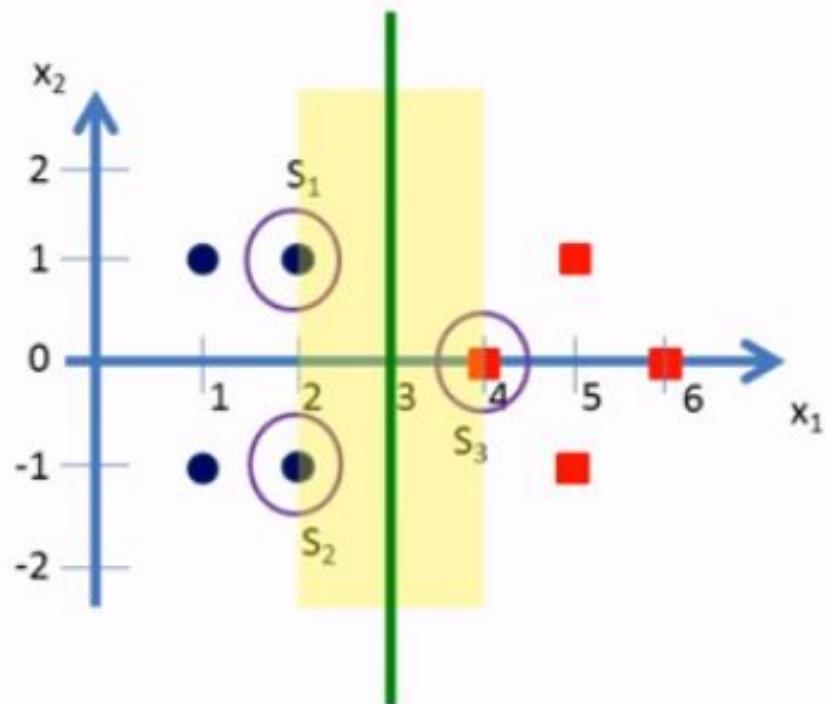
- Our vectors are augmented with a bias.
- Hence we can equate the entry in  $\tilde{w}$  as the hyper plane with an offset  $b$ .
- Therefore the separating hyper plane equation

$$y = \mathbf{w}x + b \text{ with } \mathbf{w} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and offset } b = -3.$$



# SVM -Linear Example

- $y = wx + b$  with  $w = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$  and offset  $b = -3$ .



Instead if  $w = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$  the hyper plane can be drawn horizontally

# Planning and Learning

## Table of Contents



- Planning- Planning problems, Simple planning agent, Planning languages
- Blocks world ,Goal stack planning, Mean Ends Analysis
- Non-linear Planning, Conditional planning, Reactive planning
- Learning- Machine learning, Goals and Challenges of machine learning
- Learning concepts, models, Artificial neural network based learning- Back propagation, Support vector machines,
- **Reinforcement learning**, Adaptive learning, Multi\_agent based learning, Ensemble learning, Learning for decision making, Distributed learning, Speedup learning

# Reinforcement Learning



- Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each **good action**, the **agent gets positive feedback**, and for each **bad action**, the agent gets **negative feedback** or penalty.
- In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, **unlike supervised learning**.
- Since there is no labeled data, so the agent is bound to learn by its experience only.
- RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as **game-playing, robotics**, etc.
- The agent interacts with the environment and explores it by itself. The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards.

# Approaches to implement Reinforcement Learning



There are mainly three ways to implement reinforcement-learning in ML, which are:

- **Value-based:**

The value-based approach is about to find the optimal value function, which is the maximum value at a state under any policy. Therefore, the agent expects the long-term return at any state(s) under policy  $\pi$ .

- **Policy-based:**

Policy-based approach is to find the optimal policy for the maximum future rewards without using the value function. In this approach, the agent tries to apply such a policy that the action performed in each step helps to maximize the future reward.

The policy-based approach has mainly two types of policy:

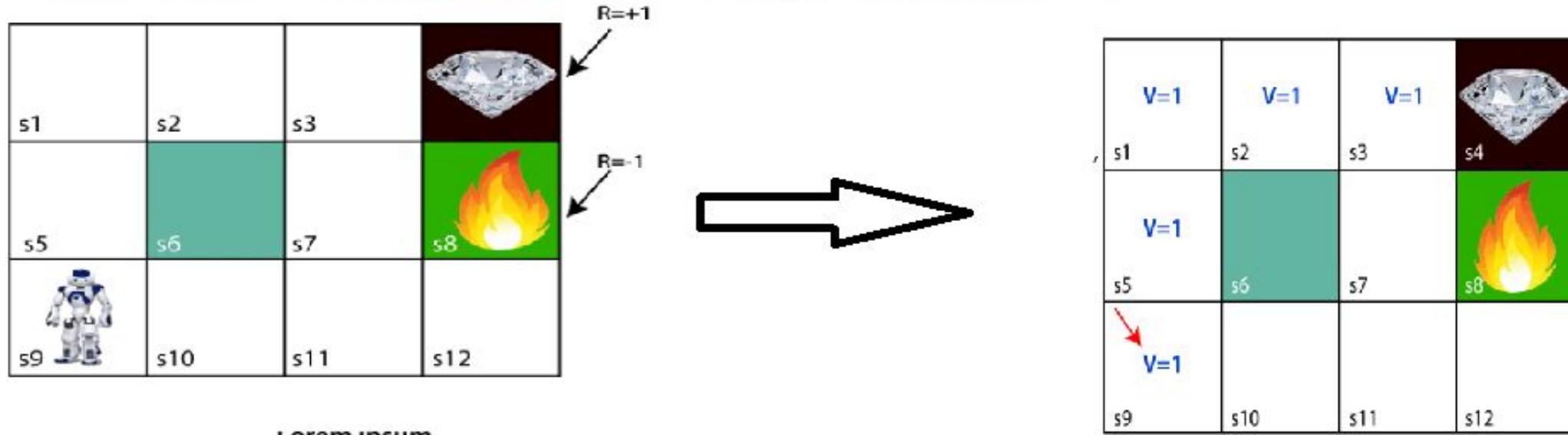
- **Deterministic:** The same action is produced by the policy  $\pi(\pi)$  at any state.
- **Stochastic:** In this policy, probability determines the produced action.

- **Model-based:** In the model-based approach, a virtual model is created for the environment, and the agent explores that environment to learn it. There is no particular solution or algorithm for this approach because the model representation is different for each environment.



# Reinforcement Learning Example

Let's take an example of a maze environment that the agent needs to explore. Consider the below image:



**LOREM IPSUM**

In the above image, the agent is at the very first block of the maze. The maze is consisting of an  $S_6$  block, which is a wall,  $S_8$  a fire pit, and  $S_4$  a diamond block.

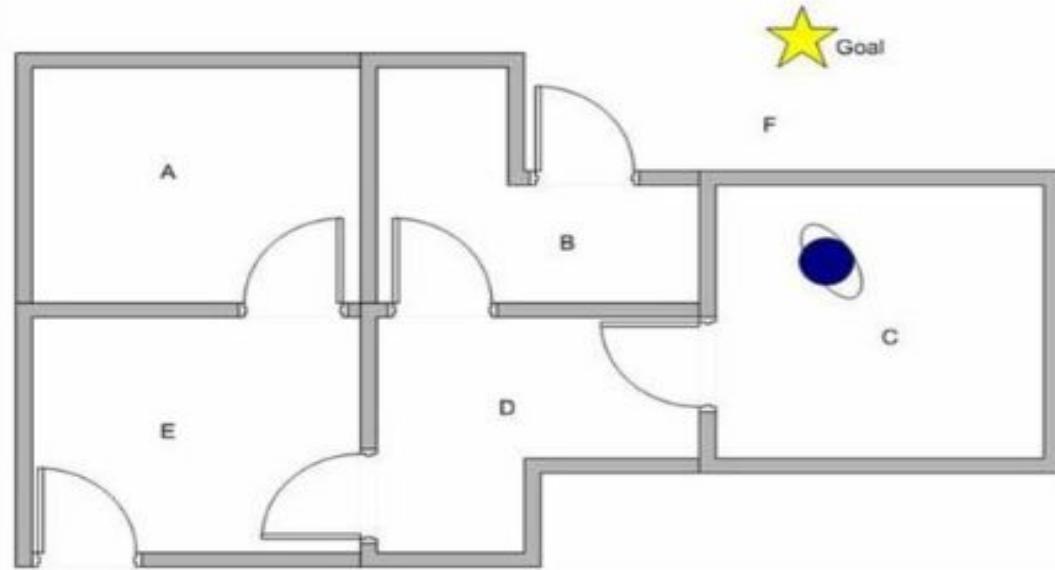
The agent cannot cross the  $S_6$  block, as it is a solid wall. If the agent reaches the  $S_4$  block, then get the +1 reward; if it reaches the fire pit, then gets -1 reward point. It can take four actions: move up, move down, move left, and move right.

The agent can take any path to reach to the final point, but he needs to make it in possible fewer steps. Suppose the agent considers the path  $S_9-S_5-S_1-S_2-S_3$ , so he will get the +1-reward point.



# Reinforcement Learning Example

- Environment



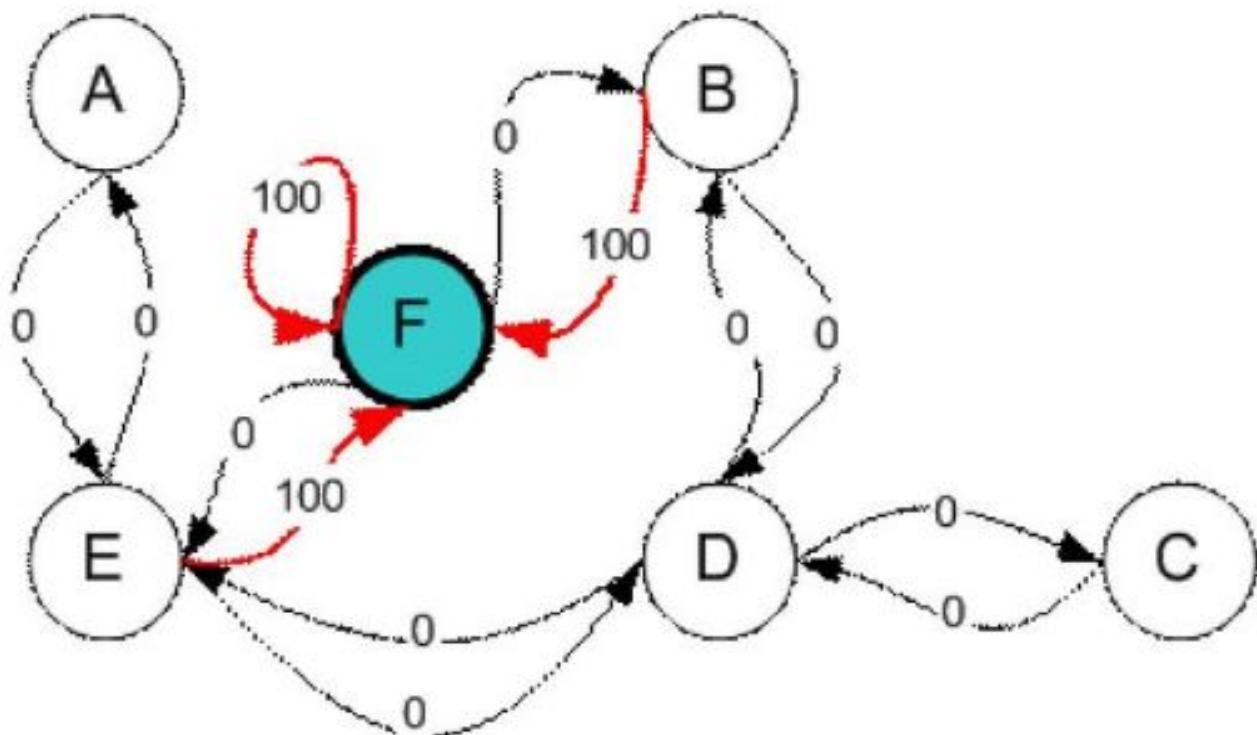
A to E: rooms, F: outside building (target).

The aim is that an agent learn to get out of building from any of rooms in an optimal way.



# Reinforcement Learning Example

- Modeling of the environment





# Reinforcement Learning Example

## State, Action, Reward and Q-value

- Reward matrix

state \ action		A	B	C	D	E	F
<b>R =</b>	A	-	-	-	-	0	-
	B	-	-	-	0	-	100
	C	-	-	-	0	-	-
	D	-	0	0	-	0	-
	E	0	-	-	0	-	100
	F	-	0	-	-	0	100



# Reinforcement Learning Example

- Q-table and the update rule

$$\mathbf{Q} = \begin{matrix} & \begin{matrix} A & B & C & D & E & F \end{matrix} \\ \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \end{matrix} & \left[ \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \right] \end{matrix}$$

Q table update rule:

$$\mathbf{Q}(\text{state}, \text{action}) = \mathbf{R}(\text{state}, \text{action}) + \gamma \cdot \text{Max}[\mathbf{Q}(\text{next state}, \text{all actions})]$$

$0 \leq \gamma < 1$  : learning parameter

## Introducing the Q-Table

Q-Table is just a fancy name for a simple lookup table where we calculate the maximum expected future rewards for action at each state. Basically, this table will guide us to the best action at each state.



# Reinforcement Learning Example

## Q Learning

**Given :** State diagram with a goal state (represented by matrix  $\mathbf{R}$ )

**Find :** Minimum path from any initial state to the goal state (represented by matrix  $\mathbf{Q}$ )

**Q Learning Algorithm** goes as follow

1. Set parameter  $\gamma$ , and environment reward matrix  $\mathbf{R}$
2. Initialize matrix  $\mathbf{Q}$  as zero matrix
3. For each episode:
  - Select random initial state
  - Do while not reach goal state
    - Select one among all possible actions for the current state
    - Using this possible action, consider to go to the next state
    - Get maximum Q value of this next state based on all possible actions
    - Compute  $\mathbf{Q}(\text{state}, \text{action}) = \mathbf{R}(\text{state}, \text{action}) + \gamma \cdot \text{Max}[\mathbf{Q}(\text{next state}, \text{all actions})]$
    - Set the next state as the current state

End Do

End For

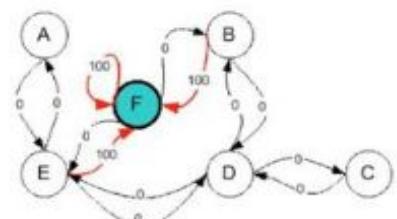


# Reinforcement Learning Example

## Numerical Example

Let us set the value of learning parameter **0.8**  
and initial state as **room B**.

$$\mathbf{Q} = \begin{matrix} A & B & C & D & E & F \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$
$$\mathbf{R} = \begin{matrix} state \backslash action & A & B & C & D & E & F \\ \begin{bmatrix} A & \begin{bmatrix} - & - & - & - & 0 & - \\ - & - & - & 0 & - & 100 \\ - & - & - & 0 & - & - \\ - & 0 & 0 & - & 0 & - \\ 0 & - & - & 0 & - & 100 \\ - & 0 & - & - & 0 & 100 \end{bmatrix} \\ B \\ C \\ D \\ E \\ F \end{bmatrix} \end{matrix}$$





# Reinforcement Learning Example

## Episode 1: start from B

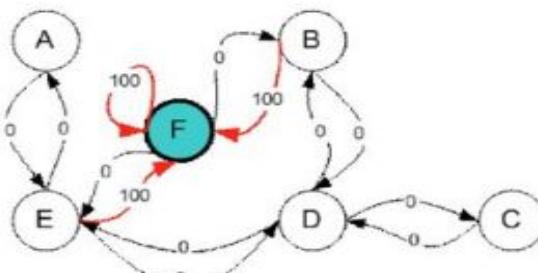
Look at the second row (state B) of matrix  $\mathbf{R}$ . There are two possible actions for the current state B, that is to go to state D, or go to state F. By random selection, we select to go to F as our action.

$$Q(state, action) = R(state, action) + \gamma \cdot \text{Max}[Q(next\ state, all\ actions)]$$

$$Q(B, F) = R(B, F) + 0.8 \cdot \text{Max}\{Q(F, B), Q(F, E), Q(F, F)\} = 100 + 0.8 \cdot 0 = 100$$



	A	B	C	D	E	F
A	0	0	0	0	0	0
B	0	0	0	0	0	100
C	0	0	0	0	0	0
D	0	0	0	0	0	0
E	0	0	0	0	0	0
F	0	0	0	0	0	0





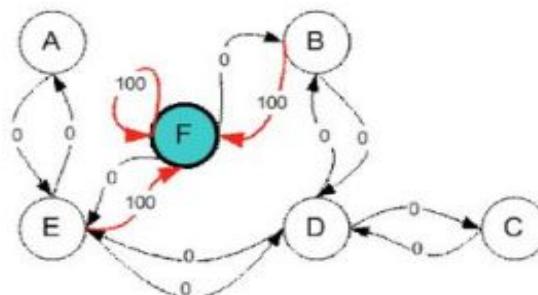
# Reinforcement Learning Example

## Episode 2: start from D

This time for instance we randomly have state D as our initial state. From R; it has 3 possible actions, B, C and E. We randomly select to go to state B as our action.

$$Q(state, action) = R(state, action) + \gamma \cdot \text{Max}[Q(next state, all actions)]$$

$$Q(D, B) = R(D, B) + 0.8 \cdot \text{Max}\{Q(B, D), Q(B, F)\} = 0 + 0.8 \cdot \text{Max}\{0, 100\} = 80$$


$$Q = \begin{bmatrix} A & B & C & D & E & F \\ A & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ B & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 100 \\ C & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ D & \begin{bmatrix} 0 & 80 & 0 & 0 & 0 & 0 \\ E & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ F & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$




# Reinforcement Learning Example

## Episode 2 (cont'd)

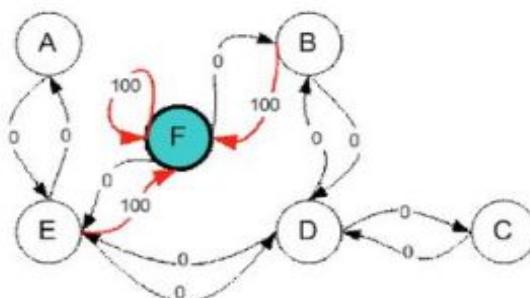
The next state is B, now become the current state. We repeat the inner loop in Q learning algorithm because state B is not the goal state. There are two possible actions from the current state B, that is to go to state D, or go to state F. By lucky draw, our action selected is state F.

$$Q(state, action) = R(state, action) + \gamma \cdot \text{Max}[Q(next\ state, all\ actions)]$$

$$\begin{aligned} Q(B, F) &= R(B, F) + 0.8 \cdot \text{Max}\{Q(F, B), Q(F, E), Q(F, F)\} \\ &= 100 + 0.8 \cdot \text{Max}\{0, 0, 0\} = 100 \end{aligned}$$

	A	B	C	D	E	F
A	0	0	0	0	0	0
B	0	0	0	0	0	100
C	0	0	0	0	0	0
D	0	80	0	0	0	0
E	0	0	0	0	0	0
F	0	0	0	0	0	0

No change





# Reinforcement Learning Example

## After Many Episodes

If our agent learns more and more experience through many episodes, it will finally reach convergence values of Q matrix as

state \ action	A	B	C	D	E	F
A	-	-	-	-	400	-
B	-	-	-	320	-	500
C	-	-	-	320	-	-
D	-	400	256	-	400	-
E	320	-	-	320	-	500
F	-	400	-	-	400	500

Normalized to  
percentage



state \ action	A	B	C	D	E	F
A	-	-	-	-	80	-
B	-	-	-	64	-	100
C	-	-	-	64	-	-
D	-	80	51	-	80	-
E	64	-	-	64	-	100
F	-	80	-	-	80	100



# Reinforcement Learning Example

To use the Q matrix, the agent traces the sequence of states, from the initial state until goal state. The algorithm is as simple as finding action that makes maximum Q for current state:

## Algorithm to utilize the Q matrix

Input: Q matrix, initial state

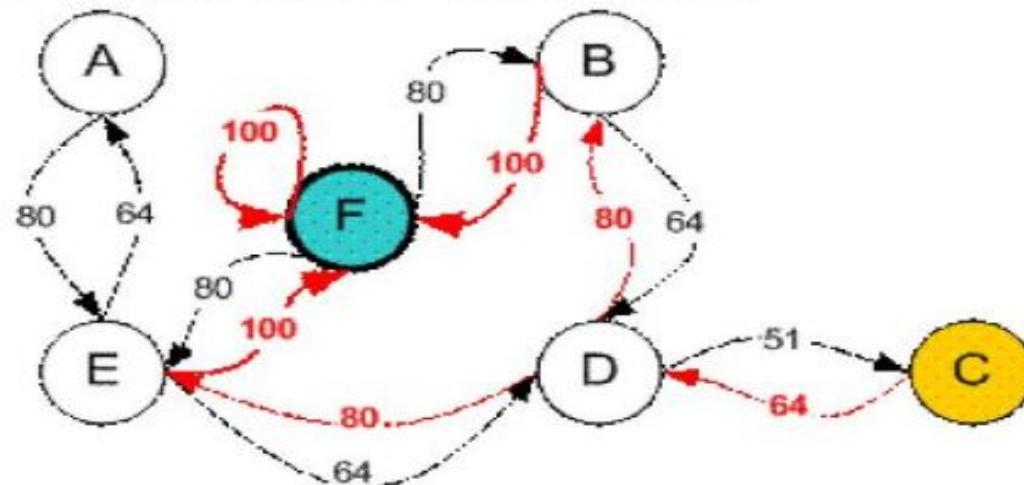
1. Set current state = initial state
2. From current state, find action that produce maximum Q value
3. Set current state = next state
4. Go to 2 until current state = goal state

The algorithm above will return sequence of current state from initial state until goal state.



# Reinforcement Learning Example

Once the Q matrix reaches almost the convergence value, our agent can reach the goal in an optimum way. To trace the sequence of states, it can easily compute by finding action that makes maximum Q for this state.



For example from initial State *C*, using the Q matrix, we can have the sequences *C – D – B – F* or *C-D-E-F*



# Elements of Reinforcement Learning

**1) Policy:** A policy can be defined as a way how an agent behaves at a given time. It maps the perceived states of the environment to the actions taken on those states. A policy is the core element of the RL as it alone can define the behavior of the agent. In some cases, it may be a simple function or a lookup table, whereas, for other cases, it may involve general computation as a search process. It could be deterministic or a stochastic policy:

For deterministic policy:  $a = \pi(s)$

For stochastic policy:  $\pi(a | s) = P[A_t = a | S_t = s]$

**2) Reward Signal:** The goal of reinforcement learning is defined by the reward signal. At each state, the environment sends an immediate signal to the learning agent, and this signal is known as a reward signal. These rewards are given according to the good and bad actions taken by the agent. The agent's main objective is to maximize the total number of rewards for good actions. The reward signal can change the policy, such as if an action selected by the agent leads to low reward, then the policy may change to select other actions in the future.

**3) Value Function:** The value function gives information about how good the situation and action are and how much reward an agent can expect. A reward indicates the immediate signal for each good and bad action, whereas a value function specifies the good state and action for the future. The value function depends on the reward as, without reward, there could be no value. The goal of estimating values is to achieve more rewards.

**4) Model:** The last element of reinforcement learning is the model, which mimics the behavior of the environment. With the help of the model, one can make inferences about how the environment will behave. Such as, if a state and an action are given, then a model can predict the next state and reward.

The model is used for planning, which means it provides a way to take a course of action by considering all future situations before actually experiencing those situations. The approaches for solving the RL problems with the help of the model are termed as the model-based approach. Comparatively, an approach without using a model is called a model-free approach.



# Types of Reinforcement learning

There are mainly two types of reinforcement learning, which are:

- **Positive Reinforcement**
- **Negative Reinforcement**

# Positive Reinforcement



- The positive reinforcement learning means adding something to increase the tendency that expected behavior would occur again. It **impacts positively on the behavior of the agent and increases the strength of the behavior.**
- This type of reinforcement can sustain the changes for a long time, but too much positive reinforcement **may lead to an overload of states that can reduce the consequences.**

# Negative Reinforcement



- The negative reinforcement learning is opposite to the positive reinforcement as it **increases the tendency that the specific behavior will occur again by avoiding the negative condition.**
- It can be more effective than the positive reinforcement depending on situation and behavior, but it provides reinforcement only to meet minimum behavior.



# Reinforcement Learning Vs Supervised Learning

## Reinforcement Learning

RL works by interacting with the environment.

The RL algorithm works like the human brain works when making some decisions.

There is no labeled dataset present.

No previous training is provided to the learning agent.

RL helps to take decisions sequentially.

## Supervised Learning

Supervised learning works on the existing dataset.

Supervised Learning works as when a human learns things in the supervision of a guide.

The labeled dataset is present.

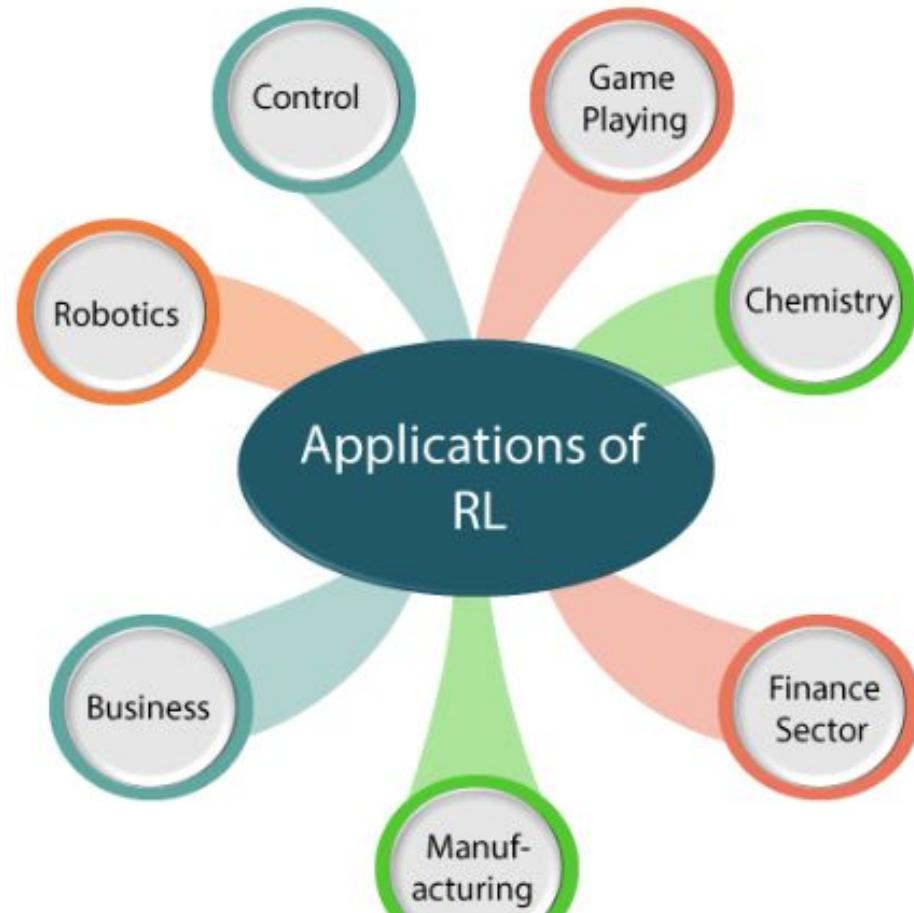
Training is provided to the algorithm so that it can predict the output.

In Supervised learning, decisions are made when input is given.

# Reinforcement Learning Applications



Applications:



# Reinforcement Learning Challenges



- Feature/reward design which should be very involved
- Parameters may affect the speed of learning.
- Realistic environments can have partial observability.
- Too much Reinforcement may lead to an overload of states which can diminish the results.
- Realistic environments can be non-stationary.

# Planning and Learning

## Table of Contents



- Planning- Planning problems, Simple planning agent, Planning languages
- Blocks world ,Goal stack planning, Mean Ends Analysis
- Non-linear Planning, Conditional planning, Reactive planning
- Learning- Machine learning, Goals and Challenges of machine learning
- Learning concepts, models, Artificial neural network based learning- Back propagation, Support vector machines,
- Reinforcement learning, **Adaptive learning**, Multi\_agent based learning, Ensemble learning, Learning for decision making, Distributed learning, Speedup learning

# Reinforcement Learning

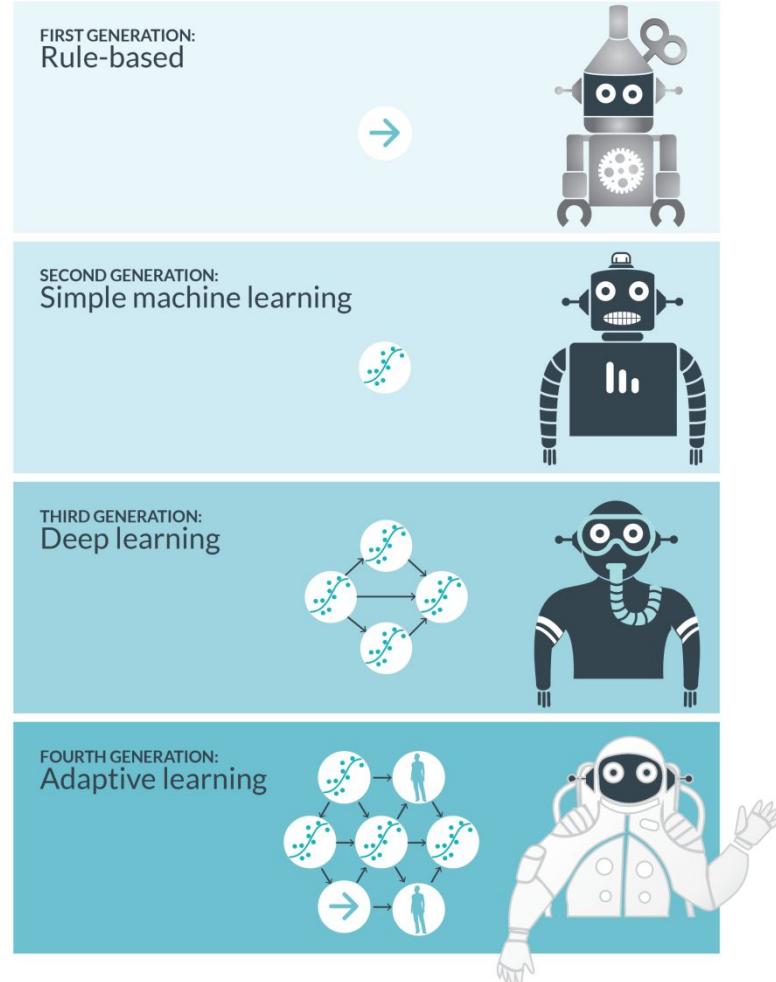


- Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each **good action**, the **agent gets positive feedback**, and for each **bad action**, the agent gets **negative feedback** or penalty.
- In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, **unlike supervised learning**.
- Since there is no labeled data, so the agent is bound to learn by its experience only.
- RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as **game-playing, robotics**, etc.
- The agent interacts with the environment and explores it by itself. The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards.



# Adaptive Learning

- Next Generation of machine Intelligence



# Adaptive Learning



- Adaptive learning brings human analysts into the process at every step. This is in contrast to rule-based, simple machine learning and deep learning approaches, where the humans only create rules and label data at the start of the process.

For example, if you had the sentence “We Will Help Tom Ford Escape from New York”, and your system hadn’t seen any examples of “Tom Ford” or “Ford Escape”, you will need human input to build the knowledge.

- Adaptive learning systems require the least human effort because they only require human input when it matters most and continually expand their knowledge when new information is encountered. They are also the most accurate. They combine the three other types of machine intelligence, adding new types of ‘unsupervised machine learning’ and methods for optimizing the input from multiple, possibly disagreeing, humans.

# Adaptive Learning Requirements



Adaptive real-time machine learning requires

- Efficient reinforcement learning (how an algorithm should continuously interact with its environment to maximise its reward)
- Online learning (dealing with continuous sequences of real-time data), and
- Adaptive learning from a small sample size.
  - developing high level 'meta-learning' algorithms that can rapidly notice data changes based on a limited amount of data samples, and continuously adjust the rest of the learning model accordingly.

# Planning and Learning

## Table of Contents



- Planning- Planning problems, Simple planning agent, Planning languages
- Blocks world ,Goal stack planning, Mean Ends Analysis
- Non-linear Planning, Conditional planning, Reactive planning
- Learning- Machine learning, Goals and Challenges of machine learning
- Learning concepts, models, Artificial neural network based learning- Back propagation, Support vector machines,
- Reinforcement learning, Adaptive learning, **Multi\_agent based learning**, Ensemble learning, Learning for decision making, Distributed learning, Speedup learning

# Multi\_agent based learning



## ADVANTAGES OF MULTI-AGENT LEARNING

- We live in a multi-agent world
- Robustness
- Scalability
- Reusability of constituents



# Multi\_agent based learning

## WHY MAS + ML

- A Goal of AI and Robotics:

**Robust and fully autonomous agent in the real world**

- Sense, decide and action
- Improve from experience (Machine learning)
- Interact with other agents (Multiagent system)
- MAS+ML in RoboCup





# Multi\_agent based learning

## LEARNING IN PRACTICE AND MAS

### ▪ Layered Learning

- Machine learning: exploit data to train
- Learning in one layer feeds into next layer
- First applied in simulated robot soccer [Stone & Veloso, 1997]

Layered Learning in Multi-Agent Systems Peter Stone

	Strategic Level	Example
$L_1$	individual	ball interception
$L_2$	multiagent	pass evaluation
$L_3$	<b>team</b>	pass selection

### ▪ Ad hoc teamwork

- PLASTIC-Policy Algorithm
- Samuel Barret, Peter Stone

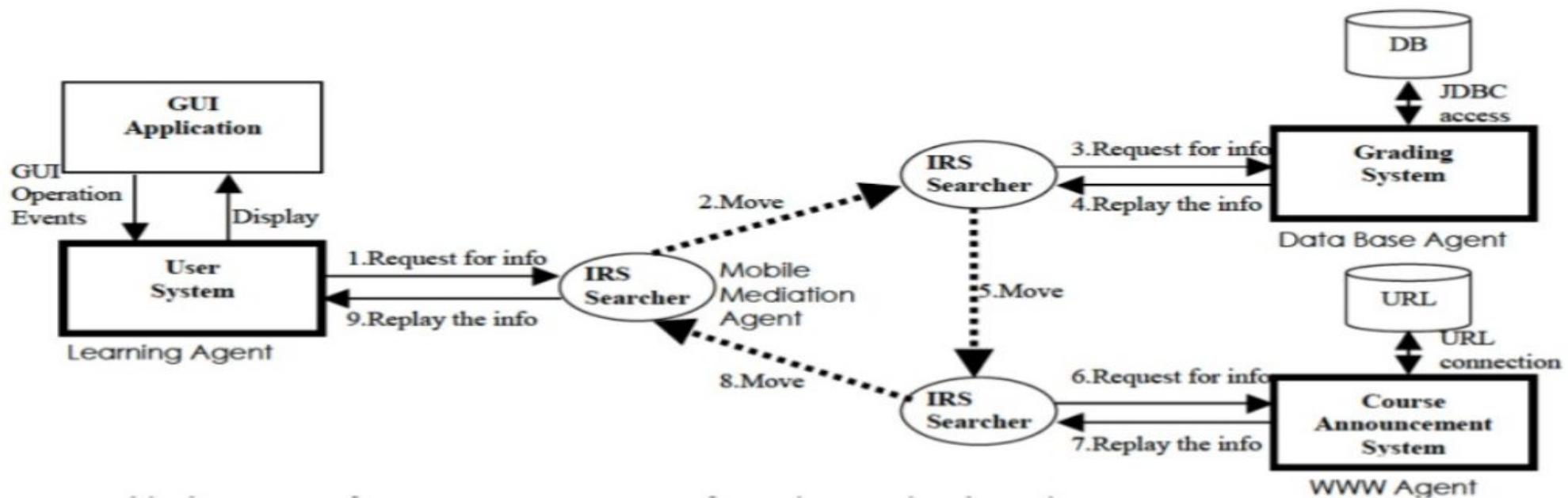


# Multi\_agent based learning Applications



## EDUCATION

- A learning Multi-Agent system that mines the Web to advise students



# Other Applications



- A Multiagent Approach to Autonomous Intersection Management [\[Traffic Control\]](#)
- Influence in Classification via Cooperative Game Theory [\[Education\]](#)
- Multi-issue automated negotiation with different strategies for a car dealer business scenario [\[Business\]](#)

# Planning and Learning

## Table of Contents



- Planning- Planning problems, Simple planning agent, Planning languages
- Blocks world ,Goal stack planning, Mean Ends Analysis
- Non-linear Planning, Conditional planning, Reactive planning
- Learning- Machine learning, Goals and Challenges of machine learning
- Learning concepts, models, Artificial neural network based learning- Back propagation, Support vector machines,
- Reinforcement learning, Adaptive learning, Multi\_agent based learning, **Ensemble learning**, Learning for decision making, Distributed learning, Speedup learning

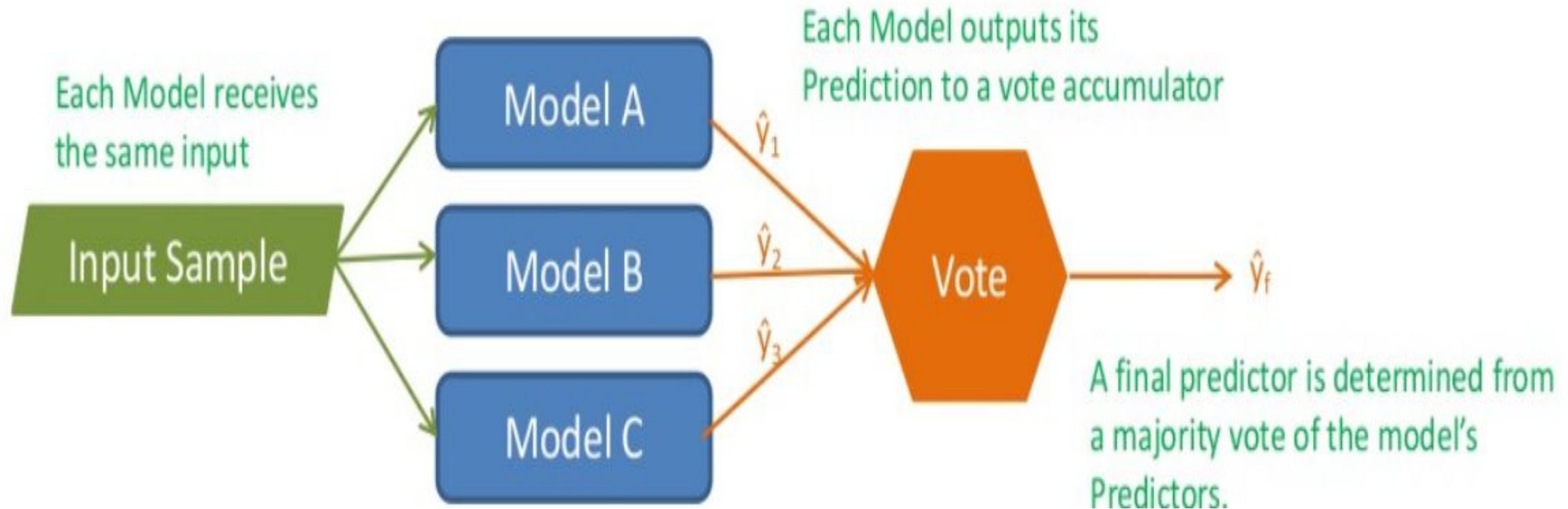
# Ensemble Learning



- Ensemble learning is an ML paradigm where numerous base models (which are often referred to as “weak learners”) are combined and trained to solve the same problem.
- This method is based on the theory that by correctly combining several base models together
- Together these ensemble models (aptly called “strong learners”) achieve better, more accurate results.
- There are three main ensemble techniques: bagging, boosting and stacking.



# Ensemble Learning – Base Idea



# Ensemble Learning Techniques - Combining Weak Learners



- **Bagging:** Bagging attempts to incorporate similar learners on small-sample populations and calculates the average of all the predictions. Generally, bagging allows you to use different learners in different populations. By doing so, this method helps to reduce the variance error.
- **Boosting:** Boosting is an iterative method that fine-tunes the weight of an observation according to the most recent classification. If an observation was incorrectly classified, this method will increase the weight of that observation in the next round (in which the next model will be trained) and will be less prone to misclassification. Similarly, if an observation was classified correctly, then it will reduce its weight for the next classifier. The weight represents how important the correct classification of the specific data point should be, as this enables the sequential classifiers to focus on examples previously misclassified. Generally, boosting reduces the bias error and forms strong predictive models, but at times they may overfit on the training data.
- **Stacking:** Stacking is a clever way of combining the information provided by different models. With this method, a learner of any sort can be used to combine different learners' outputs. The result can be a decrease in bias or variance determined by which combined

# Planning and Learning

## Table of Contents



- Planning- Planning problems, Simple planning agent, Planning languages
- Blocks world ,Goal stack planning, Mean Ends Analysis
- Non-linear Planning, Conditional planning, Reactive planning
- Learning- Machine learning, Goals and Challenges of machine learning
- Learning concepts, models, Artificial neural network based learning- Back propagation, Support vector machines,
- Reinforcement learning, Adaptive learning, Multi\_agent based learning, Ensemble learning, **Learning for decision making**, Distributed learning, Speedup learning



# Learning for decision making

AI systems for decision-making can be understood as lying along a spectrum according to their levels of autonomy.

- In some cases, human experts use AI techniques to support them in reasoning about a single, high-stakes decision.
- In other settings, it makes sense for an AI system to make autonomous decisions and
- In between there are ‘mixed-initiative’ systems that share varying degrees of responsibility for action between humans and an AI systems



Single high-stakes decisions

E.g., reallocation of radio spectrum;  
environmental policy



Mixed initiative Systems

E.g., medical diagnosis with clinician override;  
intelligent tutoring



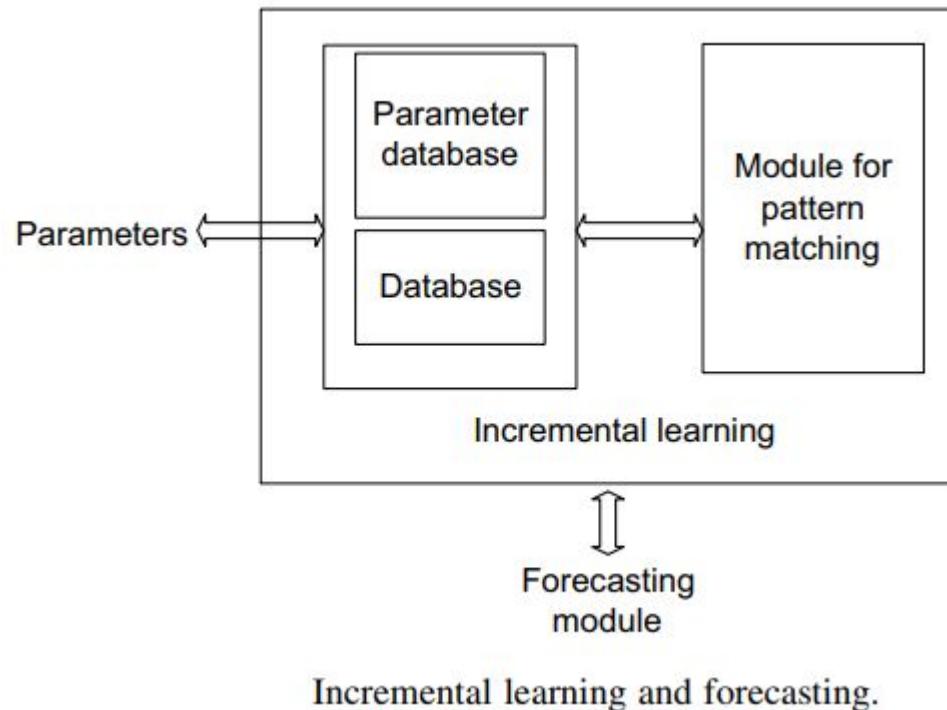
Autonomous Systems

E.g., advanced autopilots; closed loop control  
systems

# Learning and Decision-Making Model



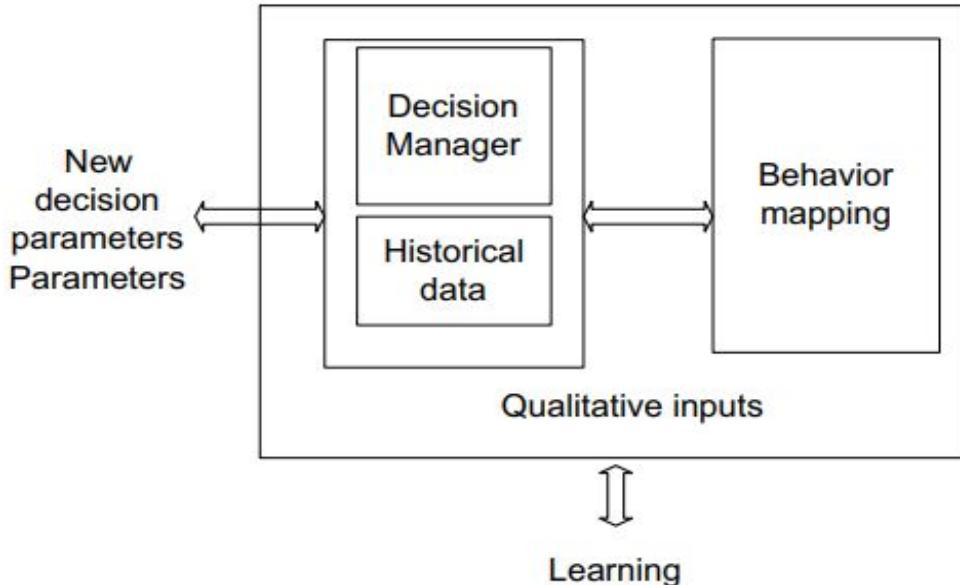
- The following figure gives the architecture of the new forecasting module.
- Its applications vary from health-care decision making to hospitality industry and revenue management. This forecasting tool will lie beneath the decision system.



# Learning and Decision-Making Model



- Complete decision-system architecture for decision making based on incremental learning is depicted in figure below.
- The decision manager is responsible for decision making and works on historical data and behavior mapping. Qualitative inputs and incremental quantitative inputs facilitate decision making.



Incremental learning and decision making.

# Planning and Learning

## Table of Contents



- Planning- Planning problems, Simple planning agent, Planning languages
- Blocks world ,Goal stack planning, Mean Ends Analysis
- Non-linear Planning, Conditional planning, Reactive planning
- Learning- Machine learning, Goals and Challenges of machine learning
- Learning concepts, models, Artificial neural network based learning- Back propagation, Support vector machines,
- Reinforcement learning, Adaptive learning, Multi\_agent based learning, Ensemble learning, Learning for decision making, **Distributed learning**, Speedup learning

# Distributed Learning



- Distributed machine learning refers to multinode machine learning algorithms and systems that are designed to improve performance, increase accuracy, and scale to larger input data sizes.
- Increasing the input data size for many algorithms can significantly reduce the learning error and can often be more effective than using more complex methods
- Distributed machine learning allows companies, researchers, and individuals to make informed decisions and draw meaningful conclusions from large amounts of data

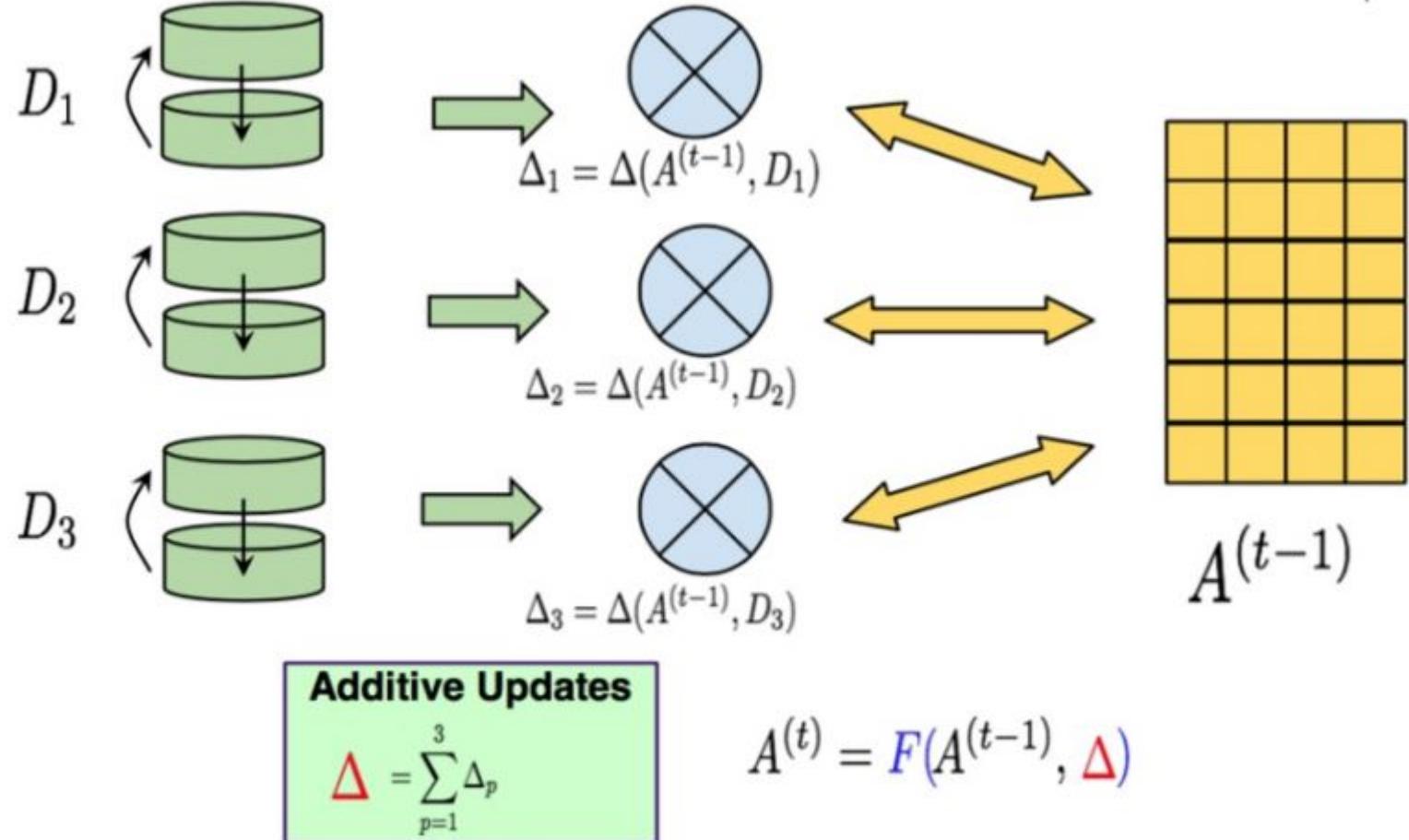


# Distributed ML Framework

- **Data Centric:** Train over large data
  - Data split over multiple machines
  - Model replicas train over different parts of data and communicate model information periodically
- **Model Centric:** Train over large models
  - Models split over multiple machines
  - A single training iteration spans multiple machines
- **Graph Centric:** Train over large graphs
  - Partitions data as graph associated with every vertex/edge;
  - Parallel apply update functions are operations on a vertex and transforming data in the scope of the vertex;

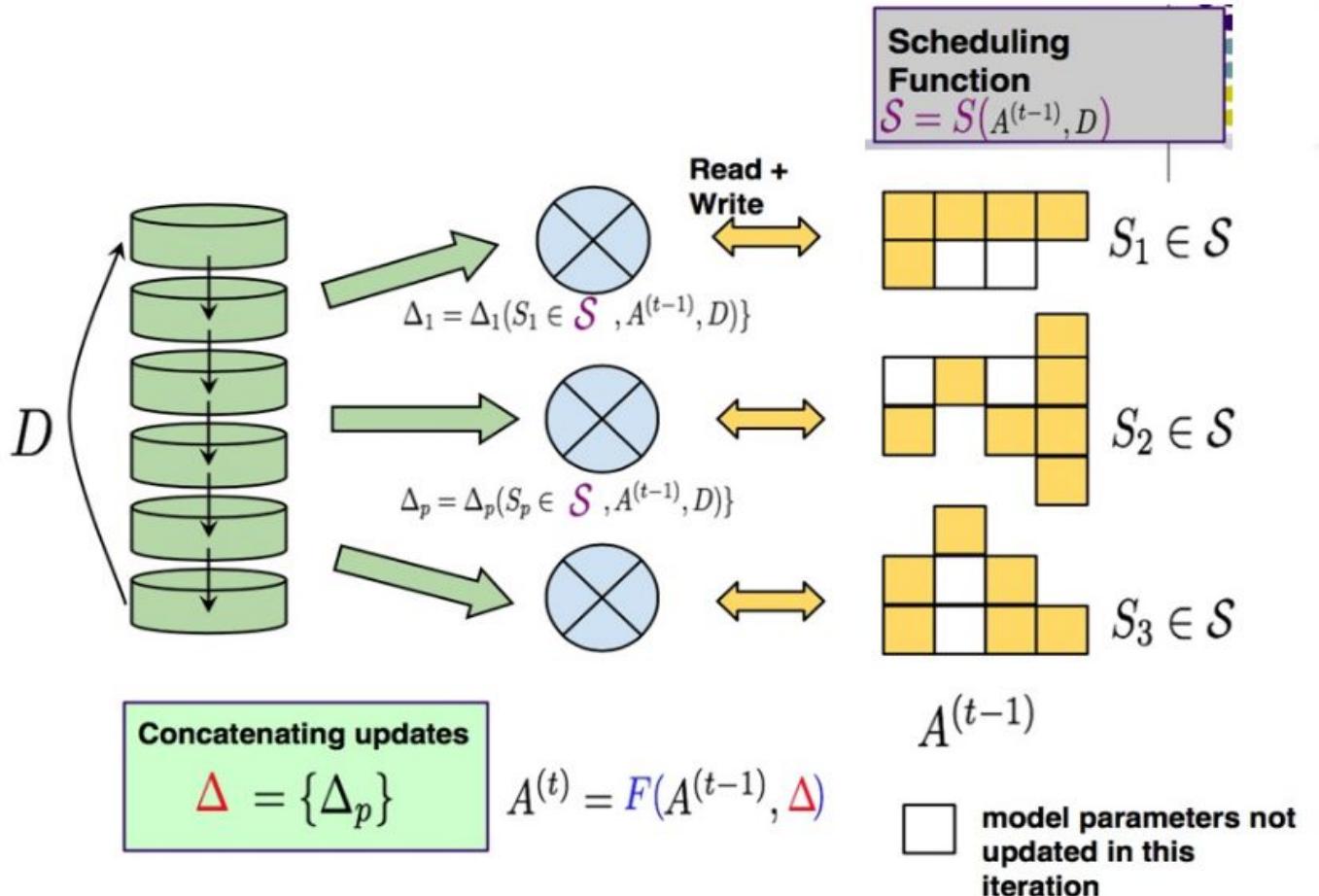


# Data parallelism





# Model parallelism



# Distributed ML algorithms -Challenges



- Even if algorithm is theoretically sound and has attractive properties, still need to pay attention **to system aspects**
  - Bandwidth (communication volume limits)
  - Latency (communication timing limits)
  - Data and Model partitioning (machine memory limitation, also affects comms volume)
  - Data and Model scheduling (affects convergence rate, comms volume & timing)
  - **Non-ideal systems behavior:** uneven machine performance, other cluster users

# Planning and Learning

## Table of Contents



- Planning- Planning problems, Simple planning agent, Planning languages
- Blocks world ,Goal stack planning, Mean Ends Analysis
- Non-linear Planning, Conditional planning, Reactive planning
- Learning- Machine learning, Goals and Challenges of machine learning
- Learning concepts, models, Artificial neural network based learning- Back propagation, Support vector machines,
- Reinforcement learning, Adaptive learning, Multi\_agent based learning, Ensemble learning, Learning for decision making, Distributed learning, **Speedup learning**



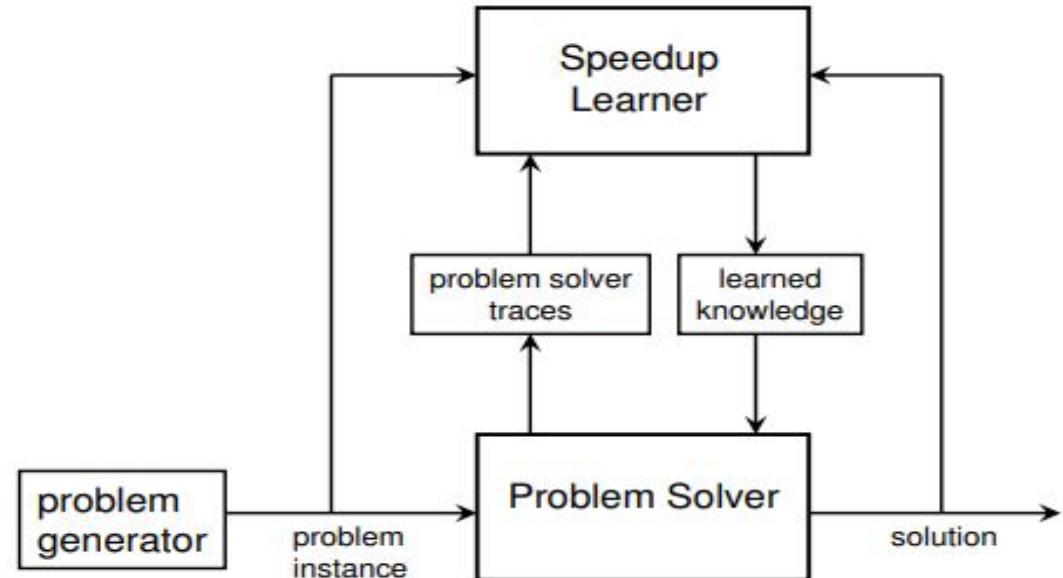
# Speedup learning

- Speedup learning is a branch of machine learning that studies learning mechanisms for speeding up problem solvers based on problem solving experience.
- The input to a speedup learner typically consists of observations of prior problem-solving experience, which may include traces of the problem solver's operations and/or solutions to solved problems.
- The output is knowledge that the problem solver can exploit to find solutions more quickly than before learning without seriously effecting solution quality
- The most distinctive feature of speedup learning, compared to most branches of machine learning, is that the learned knowledge does not provide the problem solver with the ability to solve new problem instances. Rather, the learned knowledge is intended solely to facilitate faster solution times compared to the solver without the knowledge.



# Speedup learning

- GENERAL IDEA



Schematic diagram of a speedup learning system. The problem solver receives problem instances from a problem generator and produces solutions. The speedup learner can observe the input problem instances, traces of the problem solver while solving the problem instances, and sometimes also the solutions to previously solved problem instances. The speedup learner outputs knowledge that can be used by the problem solver to speedup its solution time either on the current problem instance (intra-problem speedup) and/or future related instances (inter-problem speedup).

# Speedup learning -Types



## Intra-Problem versus Inter-Problem Speedup

- Intra-problem speedup learning is when knowledge is learned during the solution of the current problem instance and is only applicable to speeding up the solution of the current instance. After a solution is found, the knowledge is discarded as it is not applicable to future instances.
- Inter-problem speedup learning is when the learned knowledge is applicable not only to the problem(s) it was learned on but also to new problems encountered in the future



# Speedup learning Examples

- Much of the speedup learning work arising from research in AI search and constraint satisfaction falls into the intra-problem paradigm.
- The most common forms of learning are deductive and are based on computing explanations of “search failures” that occur during the solution of a particular problem.
- Here a search failure typically corresponds to a point where the problem solver must backtrack.
- By computing and forming such failure explanations the problem solver is typically able to avoid similar types of failures in the future by detecting that a search path will lead to failure without fully exploring that path.



# Thank you

# **18CSC305J-Artificial Intelligence**

## **Unit- V**

- Expert System Architecture
  - Pros and cons of Expert system
- Rule based systems
  - Frame based expert system
- Case study
- NLP – levels of NLP
- Syntactic and Semantic Analysis
  - Information Retrieval
- Information Extraction
  - Machine Translation
- NLP Applications
  - Advance topics in Artificial Intelligence- Cloud Computing and Intelligent agent
  - Business Intelligence and Analytics
  - Sentiment Analysis
  - Deep Learning Algorithms
  - Planning and Logic in intelligent Agents

# Expert Systems

# Expert Systems - Objectives

- Learn the meaning of an expert system
- Understand the problem domain and knowledge domain
- Learn the advantages of an expert system
- Understand the stages in the development of an expert system
- Examine the general characteristics of an expert system

# Objectives

- Examine earlier expert systems which have given rise to today's knowledge-based systems
- Explore the applications of expert systems in use today
- Examine the structure of a rule-based expert system
- Learn the difference between procedural and nonprocedural paradigms
- What are the characteristics of artificial neural systems

# What is an expert system?

“An expert system is a computer system that emulates, or acts in all respects, with the decision-making capabilities of a human expert.”

Professor Edward Feigenbaum  
Stanford University

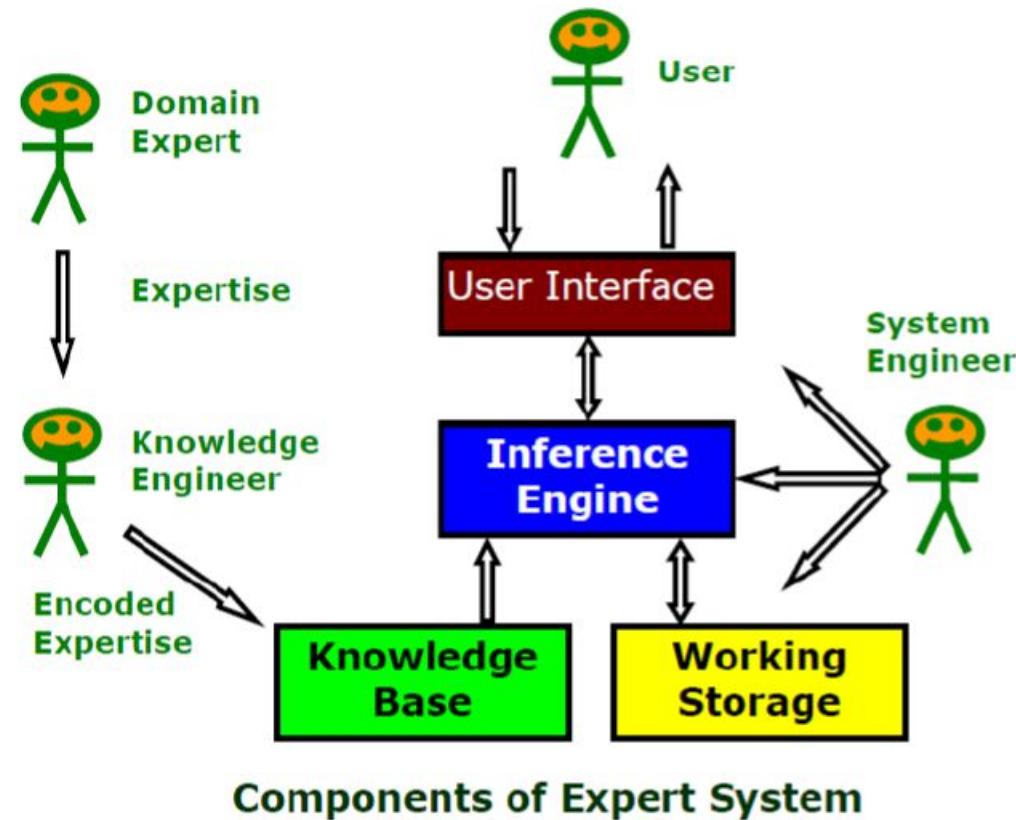
- An expert system compared with traditional computer :  
Inference engine + Knowledge = Expert system  
( Algorithm + Data structures = Program in traditional computer )
- First expert system, called DENDRAL, was developed in the early 70's at Stanford University.

# Architecture of Expert Systems

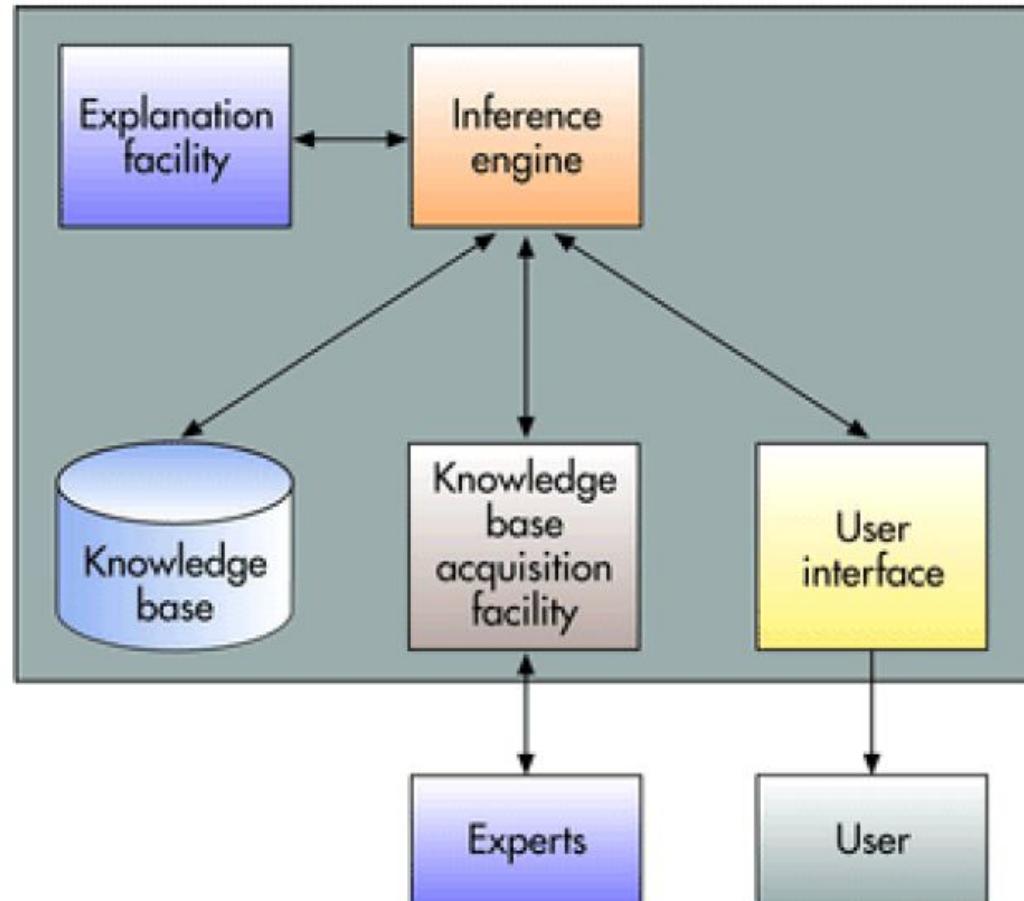
## Components of Expert Systems

The components of ES include –

- Knowledge Base
- Interface Engine
- User Interface



# Architecture of Expert Systems



## Knowledge Base

Stores all relevant information, data, rules, cases, and relationships used by the expert system.

### Uses

- Rules
- If-then Statements
- Fuzzy Logic

## Inference Engine

Seeks information and relationships from the knowledge base and provides answers, predictions, and suggestions the way a human expert would.

### Uses

- Backward Chaining
- Forward Chaining

# Architecture of Expert Systems

## **Explanation Facility**

Allows a user to understand how the expert system arrived at certain conclusions or results.

For example: it allows a doctor to find out the logic or rationale of the diagnosis made by a medical expert system

## **Knowledge acquisition facility**

Provide convenient and efficient means of capturing and storing all the components of the knowledge base.

Acts as an interface between experts and the knowledge base.

## **User Interface**

Specialized user interface software employed for designing, creating, updating, and using expert systems.

The main purpose of the user interface is to make the development and use of an expert system easier for users and decision makers

# General Methods of Inferencing

- Forward chaining (data-driven)– reasoning from facts to the conclusions resulting from those facts – best for prognosis, monitoring, and control.
  - Examples: CLIPS, OPS5
- Backward chaining (query/Goal driven)– reasoning in reverse from a hypothesis, a potential conclusion to be proved to the facts that support the hypothesis – best for diagnosis problems.
  - Examples: MYCIN

# Expert Systems Development

## Steps in the Expert System Development Process



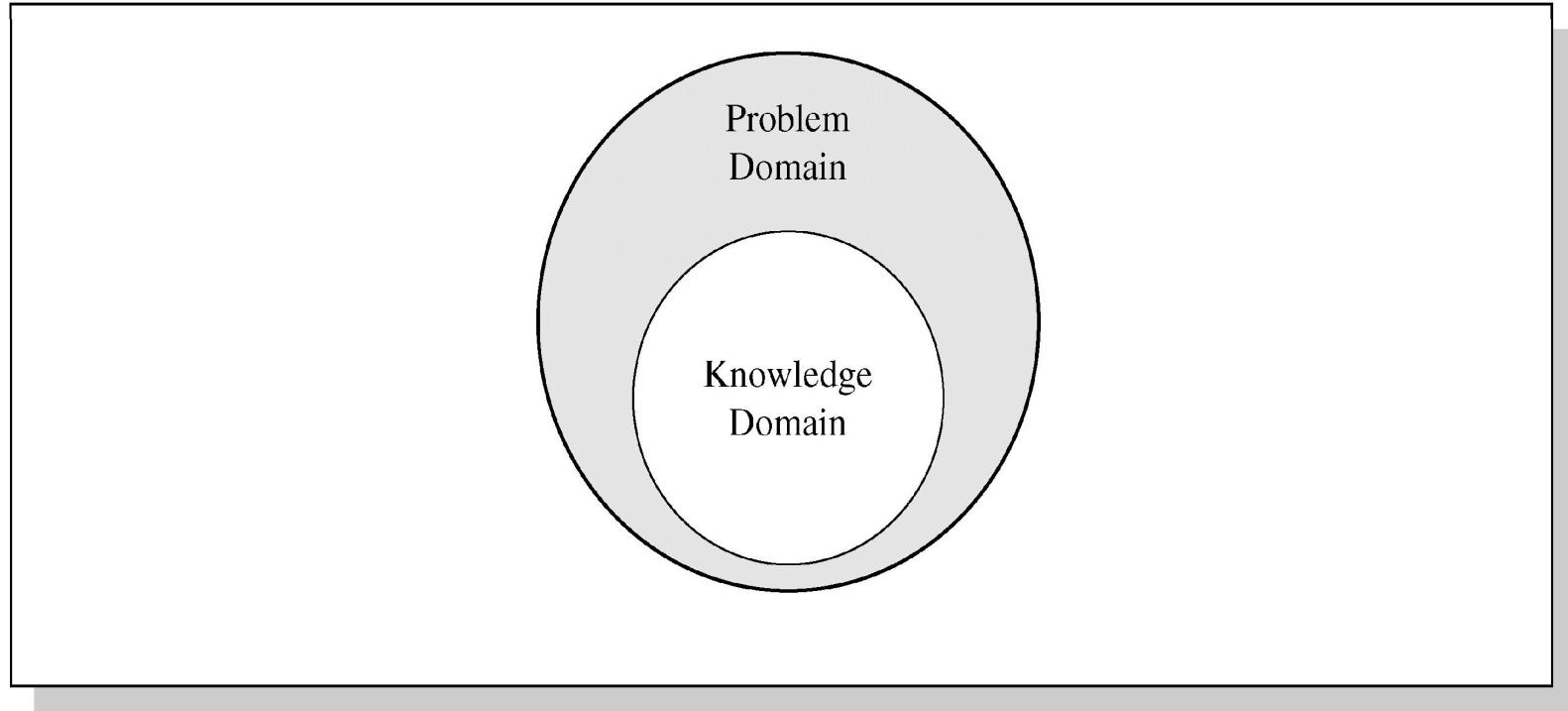
# Expert system technology may include:

- Special expert system languages – CLIPS
- Programs
- Hardware designed to facilitate the implementation of those systems

## Problem Domain vs. Knowledge Domain

- An expert's knowledge is specific to one problem domain – medicine, finance, science, engineering, etc.
- The expert's knowledge about solving specific problems is called the knowledge domain.
- The problem domain is always a superset of the knowledge domain.

# Problem and Knowledge Domain Relationship



# Advantages of Expert Systems

1. A consistent output
2. Quick and fast response
3. Location/date/day/time independent
4. Can be made generalised. (A change in application would result in looking out for a human expert related to that field if we are not relying on expert systems. But with expert systems, having generalisation, the process gets less complicated.)
5. Efficient utilisation of the knowledge (Human experts may forget some aspects, whereas expert systems consider all the rules and scenarios.)
6. Simple future enhancements with additional information of the knowledge and the rules (but with humans, it is difficult)
7. Easy maintenance of system.

# Disadvantages of Expert Systems

The disadvantages of expert systems are mentioned below:

1. They cannot handle new dynamic situation.
2. The systems cannot be adaptive based on the decisions taken earlier.
3. Limited set of knowledge will leave them with limited set of decision outcomes.
4. Development cost could be high depending on the purpose they are used for.

# Representing the Knowledge

The knowledge of an expert system can be represented in a number of ways, including IF- THEN rules:

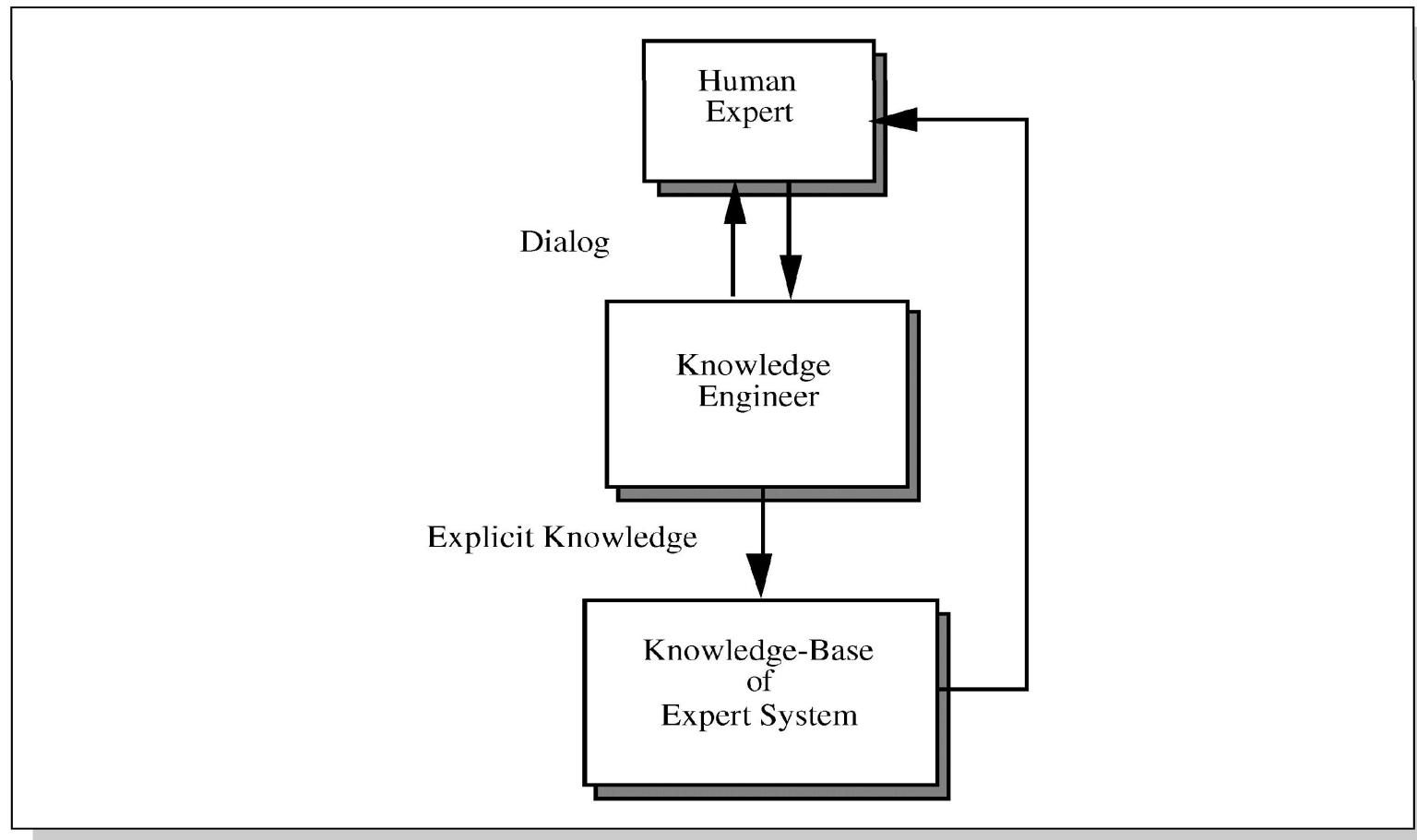
IF you are hungry THEN eat

# Knowledge Engineering

The process of building an expert system:

1. The knowledge engineer establishes a dialog with the human expert to elicit knowledge.
2. The knowledge engineer codes the knowledge explicitly in the knowledge base.
3. The expert evaluates the expert system and gives a critique to the knowledge engineer.

# Development of an Expert System



# The Role of AI

- An algorithm is an ideal solution guaranteed to yield a solution in a finite amount of time.
- When an algorithm is not available or is insufficient, we rely on artificial intelligence (AI).
- Expert system relies on inference – we accept a “reasonable solution.”

# Uncertainty

- Both human experts and expert systems must be able to deal with uncertainty.
- It is easier to program expert systems with shallow knowledge than with deep knowledge.
- Shallow knowledge – based on empirical and heuristic knowledge.
- Deep knowledge – based on basic structure, function, and behavior of objects.

# Early Expert Systems

- DENDRAL – used in chemical mass spectroscopy to identify chemical constituents
- MYCIN – medical diagnosis of illness
- DIPMETER – geological data analysis for oil
- PROSPECTOR – geological data analysis for minerals
- XCON/R1 – configuring computer systems

# Broad Classes of Expert Systems

<b>Class</b>	<b>General Area</b>
Configuration	Assemble proper components of a system in the proper way.
Diagnosis	Infer underlying problems based on observed evidence.
Instruction	Intelligent teaching so that a student can ask <i>why</i> , <i>how</i> , and <i>what if</i> questions just as if a human were teaching.
Interpretation	Explain observed data.
Monitoring	Compares observed data to expected data to judge performance.
Planning	Devise actions to yield a desired outcome.
Prognosis	Predict the outcome of a given situation.
Remedy	Prescribe treatment for a problem.
Control	Regulate a process. May require interpretation, diagnosis, monitoring, planning, prognosis, and remedies.

# Problems with Algorithmic Solutions

- Conventional computer programs generally solve problems having algorithmic solutions.
- Algorithmic languages include C, Java, and C#.
- Classical AI languages include LISP and PROLOG.

# Considerations for Building Expert Systems

- Can the problem be solved effectively by conventional programming?
- Is there a need and a desire for an expert system?
- Is there at least one human expert who is willing to cooperate?
- Can the expert explain the knowledge to the knowledge engineer can understand it.
- Is the problem-solving knowledge mainly heuristic and uncertain?

# Languages, Shells, and Tools

- Expert system languages are post-third generation.
- Procedural languages (e.g., C) focus on techniques to represent data.
- More modern languages (e.g., Java) focus on data abstraction.
- Expert system languages (e.g. CLIPS) focus on ways to represent knowledge.

# Production Rules

- Knowledge base is also called production memory.
- Production rules can be expressed in IF-THEN pseudocode format.
- In rule-based systems, the inference engine determines which rule antecedents are satisfied by the facts.

# Rule-Based Expert System

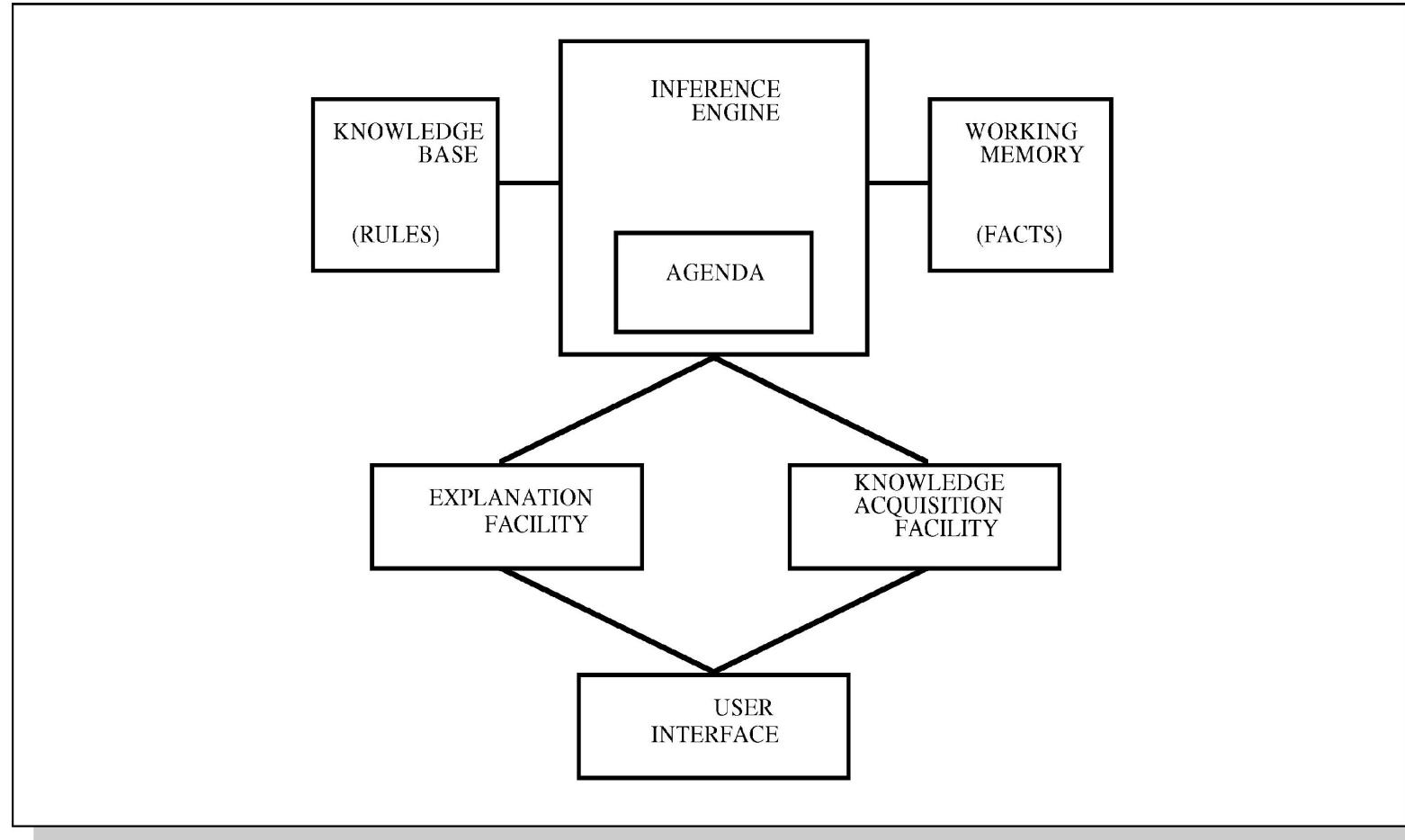
- A rule based expert system is the simplest form of artificial intelligence and uses prescribed knowledge based rules to solve a problem
- The aim of the expert system is to take knowledge from a human expert and convert this into a number of hardcoded rules to apply to the input data
- In their most basic form, the rules are commonly conditional statements (if a, then do x, else if b, then do y)
- These systems should be applied to smaller problems, as the more complex a system is, the more rules that are required to describe it, and thus increased difficulty to model for all possible outcomes

Example:

A very basic example of rule based expert system would be a program to direct the management of abdominal aneurysms. The system would input the diameter of an aneurysm. Using conditional arguments, the input diameter would be stratified to recommend whether immediate intervention was required, and if not what appropriate follow up is recommended.

Note: with problems related to radiological images, often preprocessing of the images is required prior to the expert system being applied.

# Structure of a Rule-Based Expert System



# Rule-Based ES

- knowledge is encoded as **IF ... THEN** rules
  - these rules can also be written as *production rules*
- the inference engine determines which rule antecedents are satisfied
  - the left-hand side must “match” a fact in the working memory
- satisfied rules are placed on the agenda
- rules on the agenda can be activated (“fired”)
  - an activated rule may generate new facts through its right-hand side
  - the activation of one rule may subsequently cause the activation of other rules

8.2

# Example Rules

## IF ... THEN Rules

Rule: Red\_Light

IF      the light is red ←  
THEN     stop

antecedent  
(left-hand-side)

Rule: Green\_Light

IF      the light is green ←  
THEN     go

consequent  
(right-hand-side)

## Production Rules

antecedent (left-hand-side)

the light is red ==> stop

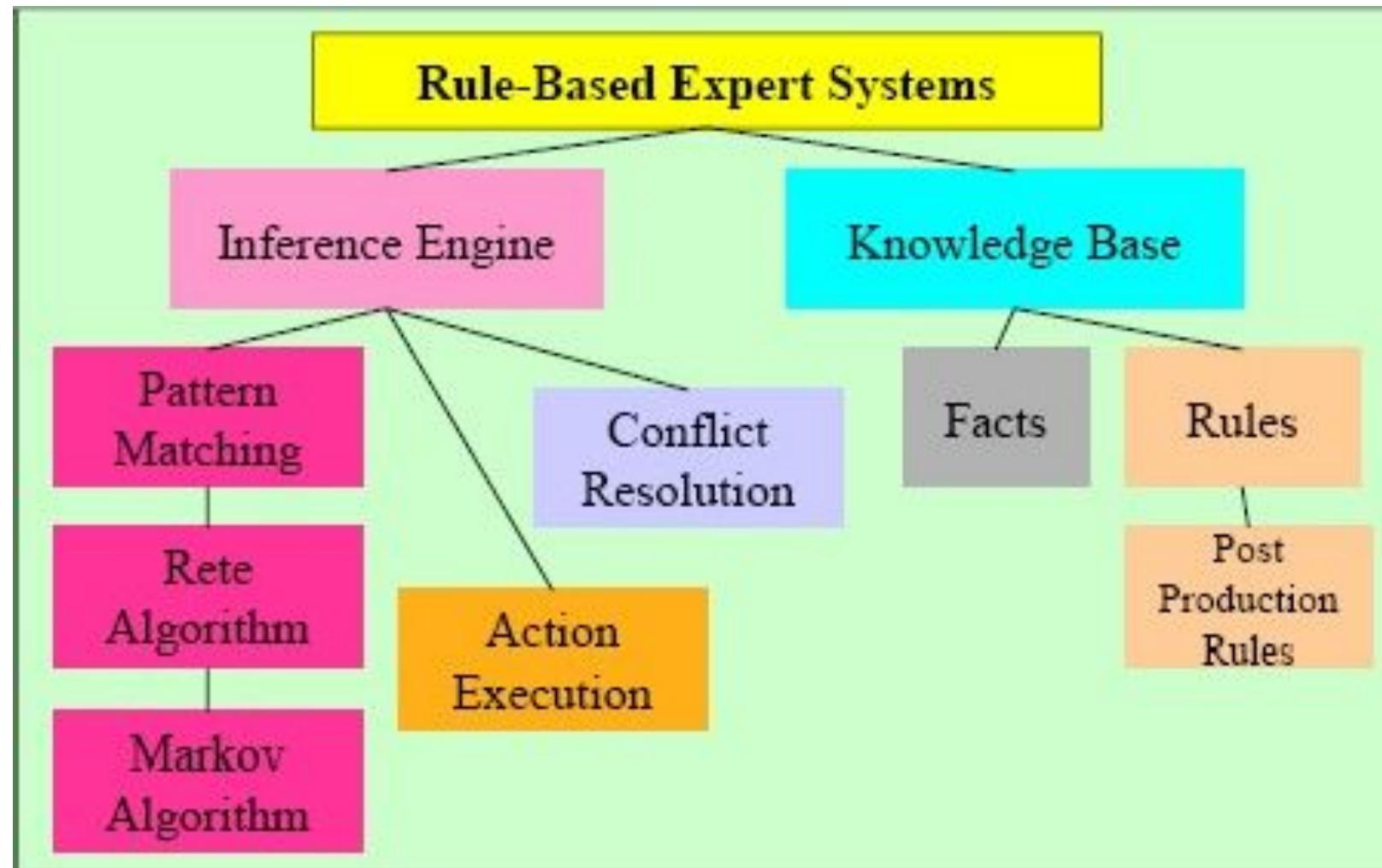
consequent  
(right-hand-side)

the light is green ==> go

# Inference Engine Cycle

- The inference engine determines the execution of the rules by the following cycle:
  - conflict resolution
    - select the rule with the highest priority from the agenda
  - execution (Act)
    - perform the actions on the consequent of the selected rule
    - remove the rule from the agenda
  - match
    - update the agenda
      - add rules whose antecedents are satisfied to the agenda
      - remove rules with non-satisfied agendas
- the cycle ends when no more rules are on the agenda, or when an explicit stop command is encountered

# Foundation of Expert Systems



# Markov Algorithm

- An ordered group of productions applied in order or priority to an input string.
- If the highest priority rule is not applicable, we apply the next, and so on.
- inefficient algorithm for systems with many rules.
- Termination on (1) last production not applicable to a string, or (2) production ending with period applied
- Can be applied to substrings, beginning at left

# Markov Algorithm

(1)  $\alpha xy \rightarrow yax$

(2)  $\alpha \rightarrow ^\wedge$

(3)  $^\wedge \rightarrow \alpha$

Rule	Success or Failure	String
1	F	ABC
2	F	ABC
3	S	$\alpha ABC$
1	S	B $\alpha AC$
1	S	BC $\alpha A$
1	F	BC $\alpha A$
2	S	BCA

**Table 1.11 Execution Trace of a Markov Algorithm**

# Rete Algorithm

- Markov: too inefficient to be used with many rules
- Functions like a net – holding a lot of information.
- Much faster response times and rule firings can occur compared to a large group of IF-THEN rules which would have to be checked one-by-one in conventional program.
- Takes advantage of temporal redundancy and structural similarity.
- Looks only for changes in matches (ignores static data)
- Drawback is high memory space requirements.

# Frame-Based Expert System

- The expert systems which make use of frames for the knowledge are called frame-based expert systems.
- What is a frame? – A frame is a data structure with typical knowledge about the object or concept.
- Frame has its name and set of attributes
- Example : A car frame can have make, type, color and so on as slots/attributes in the frame
- Each slot/ attribute has unique value associated to it

# Frame-Based Expert System

We can have the following included in the slot

1. Frame Name
2. Relationship with other frames
3. Values or Ranges
4. Procedural information

Class: Car
Make:
Type:
Colour:
Airbags:
Auto-transmission:
Windows:

Class: Car
Instance: i10
Make: Hyundai
Type: Small car
Colour : Red
Airbags: 2
Auto-transmission: No
Windows: Power

Class: Car
Instance: Verna
Make: Hyundai
Type: Sedan
Colour: Blue
Airbags: 4
Auto-transmission: No
Windows: Power

Represents the frame structure for Class and Instance

# Frame-Based Expert System

- Relationship – hierarchy, or to be specific, the inheritance is depicted
- Values & Ranges – Represent the actual/default values or specified ranges
- Procedural information – the slot is attached to a procedure that is executed when any event is triggered such as change in value for the slot
- Instance Frame refers to an object and Class frame refers to group

Class: Car
Make:
Type:
Colour:
Airbags:
Auto-transmission:
Windows:

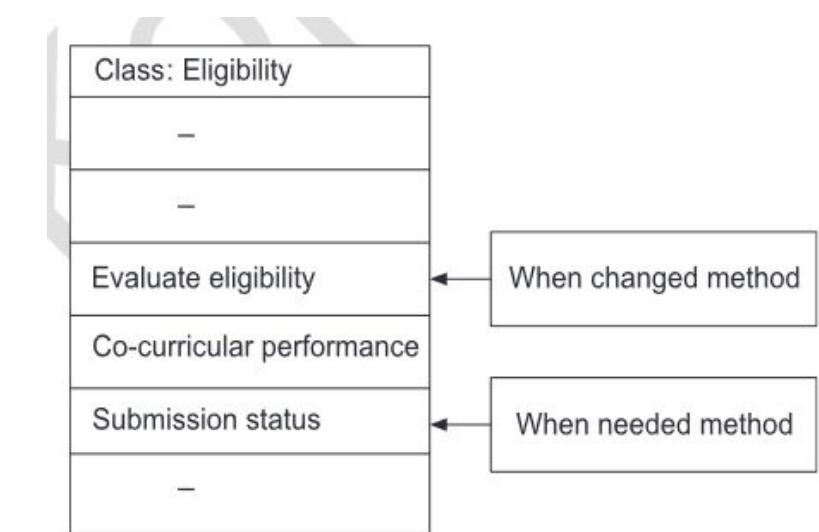
Class: Car
Instance: i10
Make: Hyundai
Type: Small car
Colour : Red
Airbags: 2
Auto-transmission: No
Windows: Power

Class: Car
Instance: Verna
Make: Hyundai
Type: Sedan
Colour: Blue
Airbags: 4
Auto-transmission: No
Windows: Power

Frames : Class and Instance

# Working of Frame-Based Expert System

- Method – A method is a procedure that is executed when requested
- Demon – makes use of if-then structure
- Frames that have the knowledge representation, the methods or the demons essentially add actions to them.
- As an example, how the process of an expert system works to check the eligibility of a student appearing for an exam is explained in the picture



**When needed and when required methods : Snapshot**

# Guidelines to build a Frame-Based Expert System

## *Guidelines to Build a Frame-based Expert System*

1. The first step that is involved in designing any expert system is the scope and the specificity of the problem at hand. There has to be utmost clarity about these two factors.
2. The next step is the identification of classes, instances and the attributes.
  
3. Since the events or to be specific, the methods are the key role players in evaluation, the display of the system has to be presented in the most simple and transparent manner.
4. The next task is to have the methods of when changed and when needed. This is dependent to the design in step 3, where the events are decided.
5. Rules are to be also defined along with the methods.
6. Finally, a full-proof evaluation of the built system, and if required, expansion should be handled.

A detailed example is handled in the case study for the frame-based expert system, provided further in the chapter.

# MYCIN

# MYCIN

- MYCIN was an early expert system that used artificial intelligence to identify bacteria causing severe infections.
- recommend antibiotics, with the dosage adjusted for patient's body weight
- The MYCIN system was also used for the diagnosis of blood clotting diseases.
- MYCIN was developed over five or six years in the early 1970s at Stanford University.
- It was written in Lisp

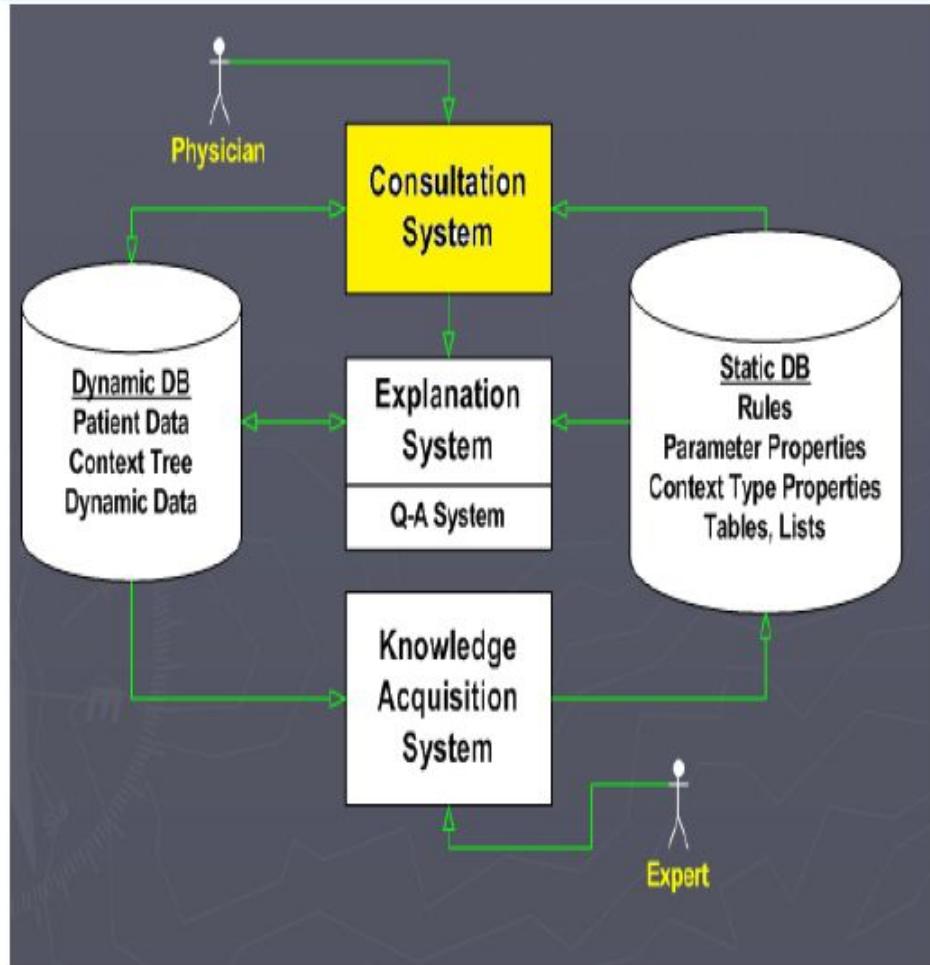
## MYCIN

- MYCIN was a stand alone system that required a user to enter all relevant information about a patient by typing in responses to questions MYCIN posed.
- MYCIN operated using a fairly simple inference engine, and a knowledge base of ~600 rules.
- It would query the physician running the program via along series of simple yes/no or textual questions.

# Tasks and Domain

- Disease DIAGNOSIS and Therapy SELECTION
- Advice for non-expert physicians with time considerations and incomplete evidence on:
  - Bacterial infections of the blood
  - Expanded to meningitis and other ailments
  - Meet time constraints of the medical field

# Consultation System

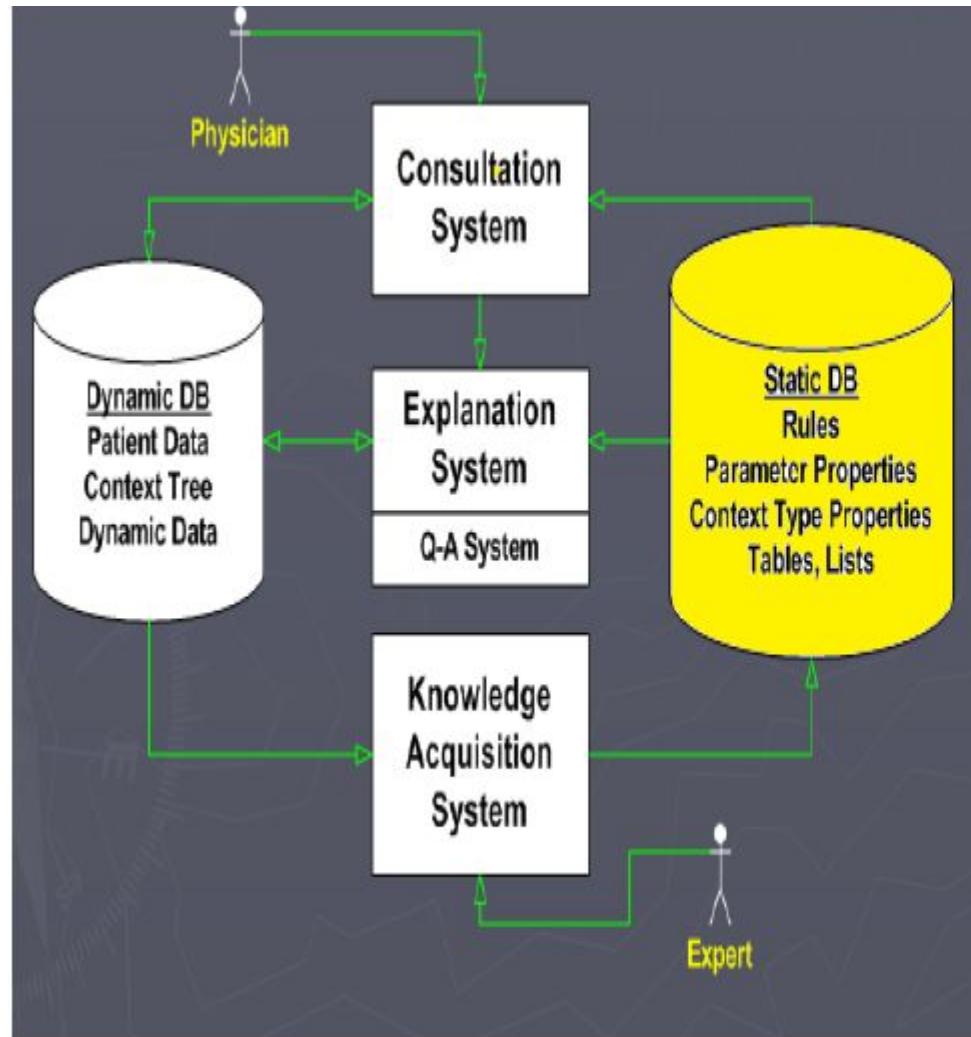


- Performs Diagnosis and Therapy Selection
- Control Structure reads Static DB (rules) and read/writes to Dynamic DB (patient, context)
- Linked to Explanations
- Terminal interface to Physician

## Consultation “Control Structure”

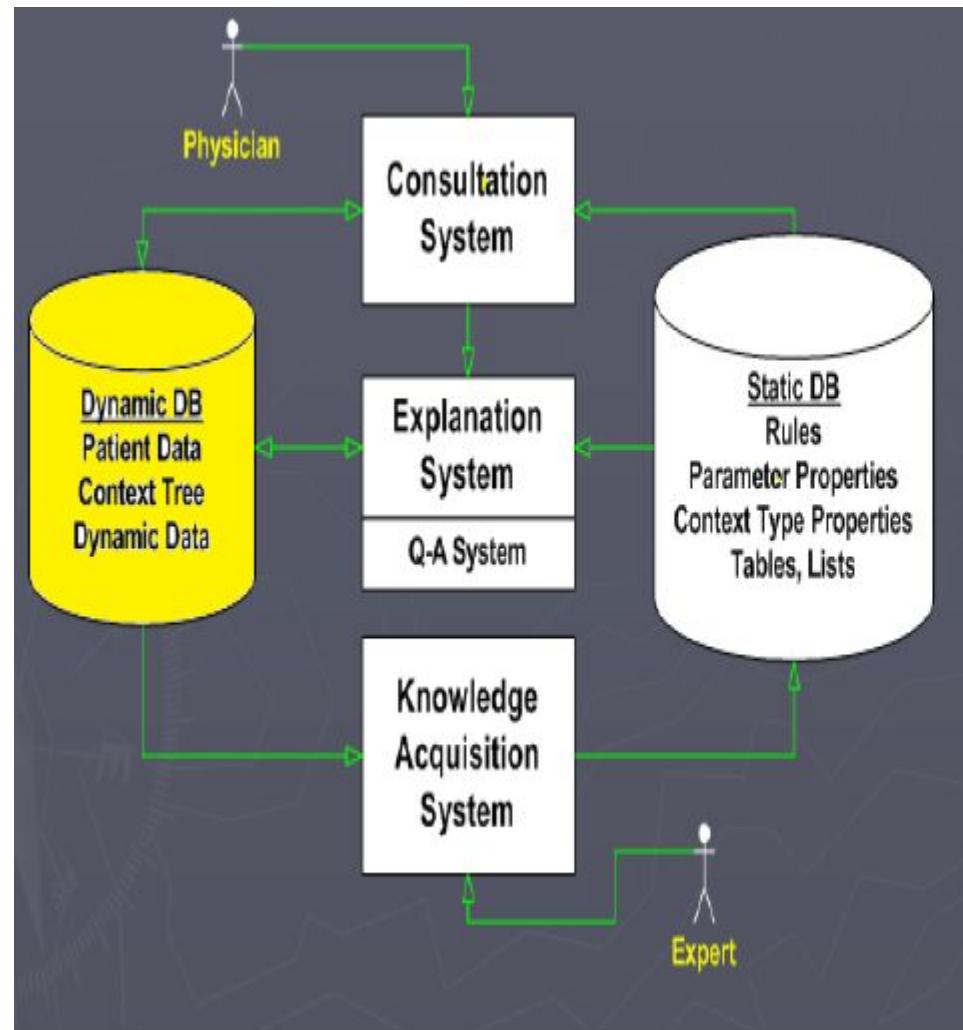
- Goal-directed Backward-chaining Depth-first Tree Search
- High-level Algorithm:
  1. Determine if Patient has significant infection
  2. Determine likely identity of significant organisms
  3. Decide which drugs are potentially useful
  4. Select best drug or coverage of drugs

# Static Database



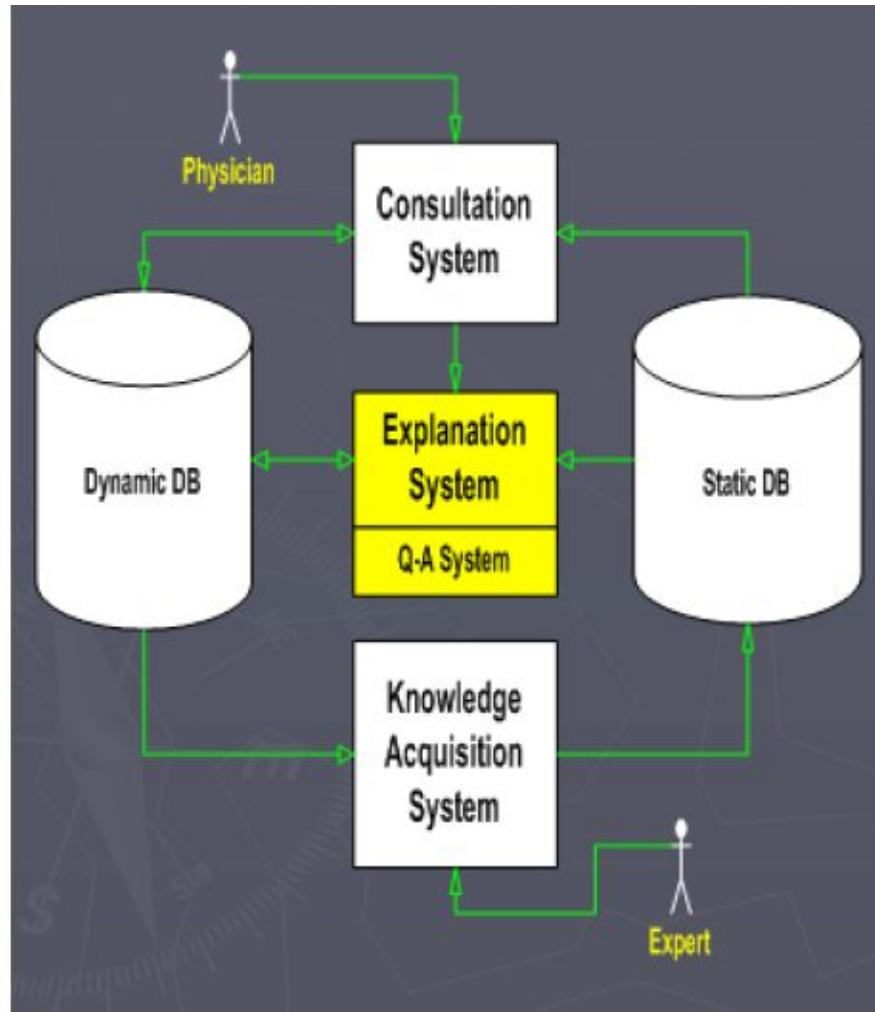
- Rules
- Meta-Rules
- Templates
- Rule Properties
- Context Properties
- Fed from Knowledge Acquisition System

# Dynamic Database



- Patient Data
- Laboratory Data
- Context Tree
- Built by Consultation System
- Used by Explanation System

# Explanation System



- Provides reasoning why a conclusion has been made, or why a question is being asked
- Q-A Module
- Reasoning Status Checker

# Xcon

- The R1 (internally called XCON, for eXpertCONfigurer) program was a production rule based system written in OPS5 by John P. McDermott of CMU in 1978.
  - configuration of DEC VAX computer systems
- ordering of DEC's VAX computer systems by automatically selecting the computer system components based on the customer's requirements.
- XCON first went into use in 1980 in DEC's plant in Salem, New Hampshire. It eventually had about 2500 rules.
- By 1986, it had processed 80,000 orders, and achieved 95.98% accuracy.
- It was estimated to be saving DEC \$25M a year by reducing the need to give customers free components when technicians made errors, by speeding the assembly process, and by increasing customer satisfaction.
- XCON interacted with the sales person, asking critical questions before printing out a coherent and workable system specification/order slip.
- XCON's success led DEC to rewrite XCON as XSEL a version of XCON intended for use by DEC's salesforce to aid a customer in properly configuring their VAX.

# XCON: Expert Configurer

Stages of Expert System building

- **Identification:**  
Problems, data, goals, company, people...
- **Conceptualization:**  
Characterize different kinds of concepts and relations
- **Formalization:**  
Express character of search
- **Implementation:**  
Build the system in executable form
- **Testing and Evaluation:**  
Does it do what we wanted?
- **Maintenance**  
Adapt to changing environment or requirements

# Natural Language Processing (NLP)

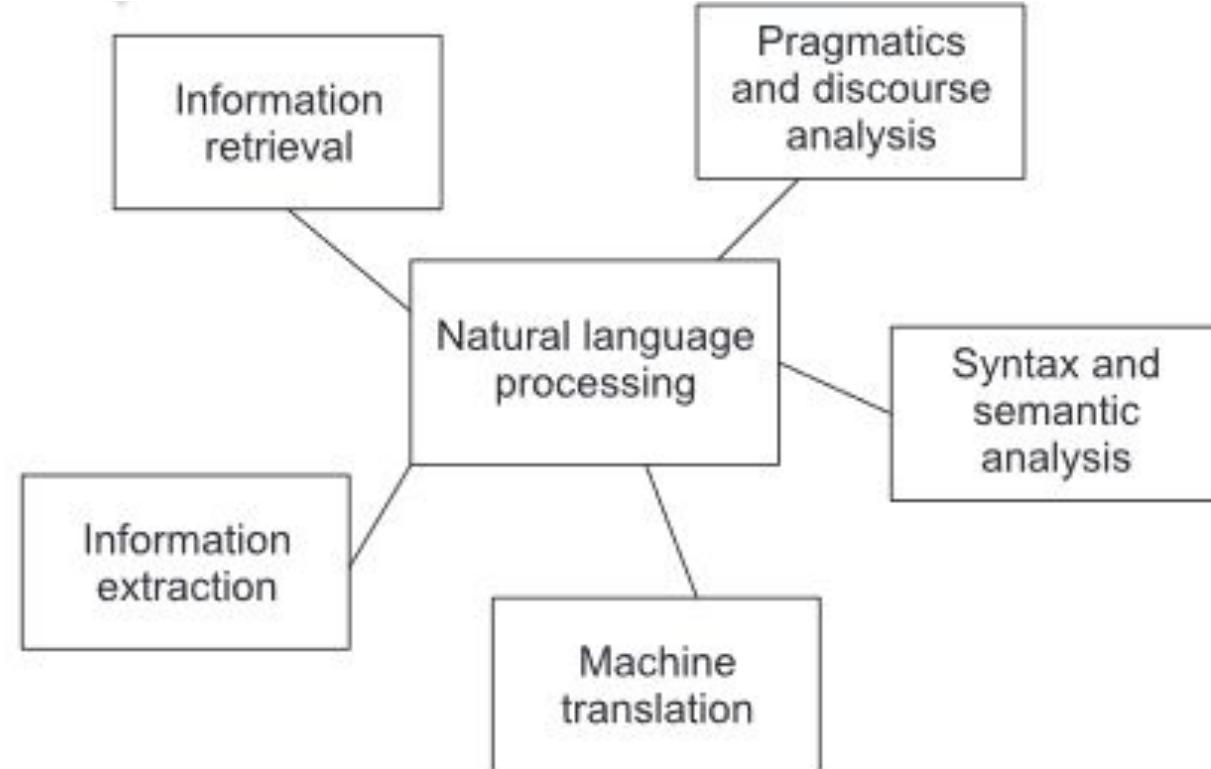
# Natural language

- **Natural languages** are languages that living creatures use for communication,
- A machine is considered to be really intelligent only when it can understand and interpret or speak the matter of natural language.
- The capability to understand, interpret and communicate through natural language is a very important criteria of intelligent behavior.

# Why Natural language processing?

- Huge amount of data?
  - Internet=at least 2.5 billion pages
- Applications for processing large amounts of texts.
  - Classify text into categories
  - Index and search large texts
  - Automatic translation
  - Speech understanding: Understanding phone conversation
  - Information extraction: Extract useful information from resumes
  - Automatic summarization
  - Question answering
  - Knowledge acquisition: knowledge from expert
  - Text generations/dialogs
- All these requires natural language expertise.

# NLP Tasks



# Levels of NLP

- 1. Morphology:** It is the analysis of individual words that consist of morphemes—the smallest grammatical unit. Generally, words with 'ing', 'ed' change the meaning of the word. This analysis becomes necessary in the determination of tense as well.
- 2. Syntax:** Syntax is concerned with the rules. It includes legal formulation of the sentences to check the structures. (Some aspects are covered in compiler's phase of syntax analysis that you must have studied). For example, 'Hari is good not to.' The sentence structure is totally invalid here.
- 3. Semantic:** During this phase, meaning check is carried out. The way in which the meaning is conveyed is analysed. The previous example is syntactically as well as semantically wrong. Now, consider one more example, i.e., 'The table is on the ceiling.' This is syntactically correct, but semantically wrong.
- 4. Discourse integration:** In communication or even in text formats, often the meaning of the current sentence is dependent on the one that is prior to it. Discourse analysis deals with the identification of discourse structure.
- 5. Pragmatics:** In this phase, analysis of the response from the user with reference to what actually the language meant to convey is handled. So, it deals with the mapping for what the user has interpreted from the conveyed part and what was actually expected. For a question like 'Do you know how long it will take to complete the job?', the expected answer is the number of hours rather than a yes or no.
- 6. Prosody:** It is an analysis phase that handles rhythm. This is the most difficult analysis that plays an important role in the poetry or *shlokas* (chants involving the name of God) that follow a rhythm.
- 7. Phonology:** This involves analysis of the different kinds of sounds that are combined. It is concerned with speech recognition.

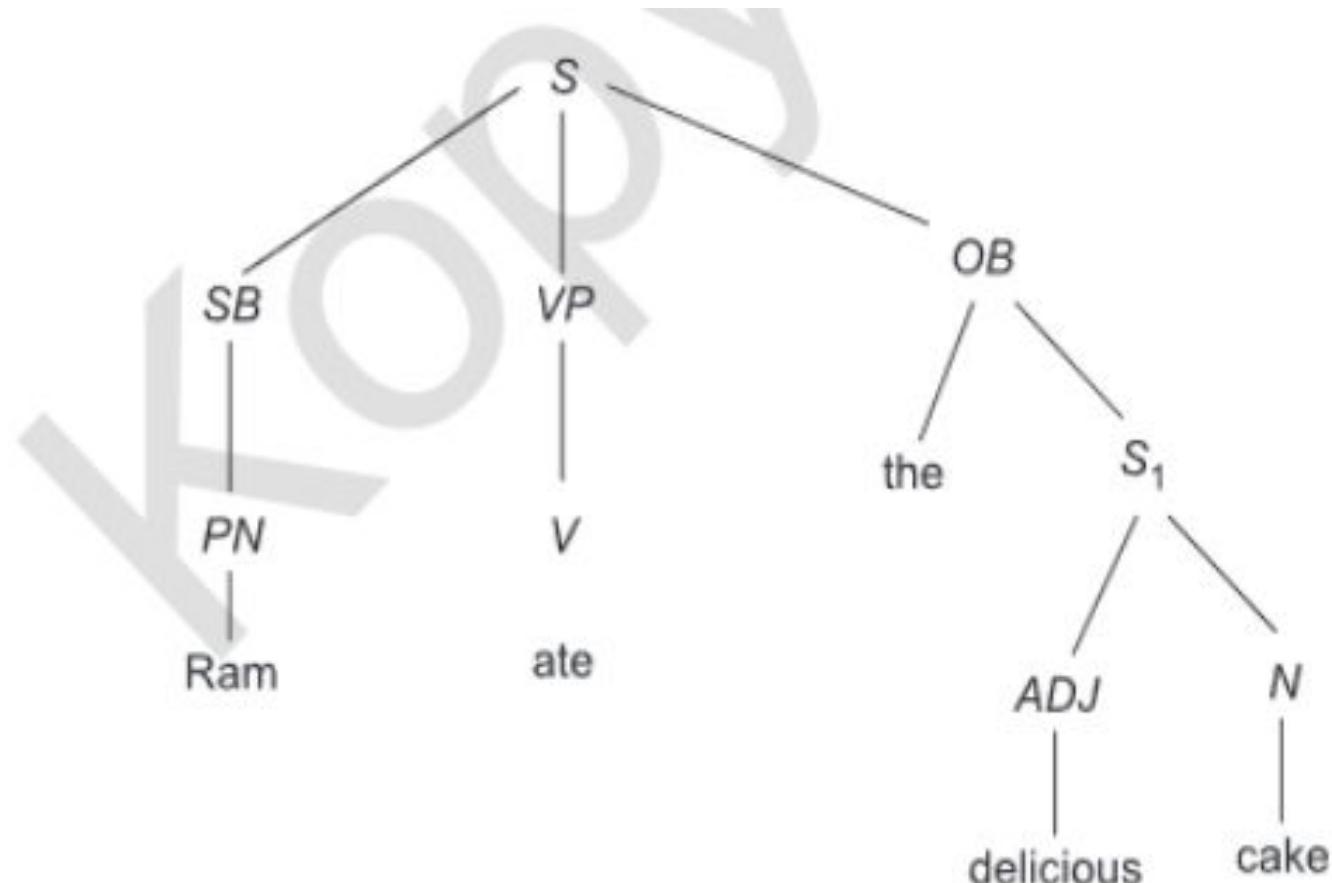
# Syntactic Analysis

- Rules of syntax (grammar) specify the possible organization of words in sentences and allows us to determine sentence's structure(s)
  - “John saw Mary with a telescope”
    - John saw (Mary with a telescope)
    - John (saw Mary with a telescope)
- Parsing: given a sentence and a grammar
  - Checks that the sentence is correct according with the grammar and if so returns a **parse tree** representing the structure of the sentence

# Syntactic Analysis - Grammar

- **sentence** → **noun\_phrase**, **verb\_phrase**
- **noun\_phrase** → **proper\_noun**
- **noun\_phrase** → **determiner**, **noun**
- **verb\_phrase** → **verb**, **noun\_phrase**
- **proper\_noun** → [mary]
- **noun** → [apple]
- **verb** → [ate]
- **determiner** → [the]

# Syntactic Analysis - Parsing



# Syntactic Analysis – Complications (1)

- Number (singular vs. plural) and gender
  - sentence-> noun\_phrase(**n**),verb\_phrase(**n**)
  - proper\_noun(**s**) -> [mary]
  - noun(**p**) -> [apples]
- Adjective
  - noun\_phrase-> determiner,adjectives,noun
  - adjectives-> adjective, adjectives
  - adjective->[ferocious]
- Adverbs, ...

## Syntactic Analysis – Complications (2)

- Handling ambiguity
  - Syntactic ambiguity: “fruit flies like a banana”
- Having to parse syntactically incorrect sentences

# Semantic Analysis

- Syntax analysis is doing the parsing activity
- But we need to understand the meaning of the words and it is done by semantic analysis
- For example,
  - ‘Keep the book on the table’ – Here table refers physical object
  - ‘Learn the table of number 23’ – here table refers mathematics concept of table

# Lexical Processing

- In lexical processing, the meaning of the tokens is found out
- Word sense disambiguation: Understanding the meaning of a particular word in the context
- It is concerned with the sense where it would be operational
- It would be done with the help of semantic marker
- Semantic marker: ‘Keep’ in sentence 1
- Semantic marker: ‘Learn’ in sentence 2

# Semantic grammars

- Example,
- ‘The pen is on the ceiling’
- Solution is,
- **S -> Action the Food**
- Action -> eat|drink|shallow|chew – Set of words
- Food -> burger|sandwich|coke|pizza – Set of words

# Case Grammar

- Case grammar is also called as Fillmore grammar
- Elements of the sentence are:
  - Object (thing on which it is acted)
  - Agent/actor (Someone who carries out the action or the event)
  - Dative (Someone who is affected by the event)
  - Location (place where the event/action occurs)
  - Time (date or time at which the action/event takes place)
- Example,
  - Rohit will meet Kunal at Mall

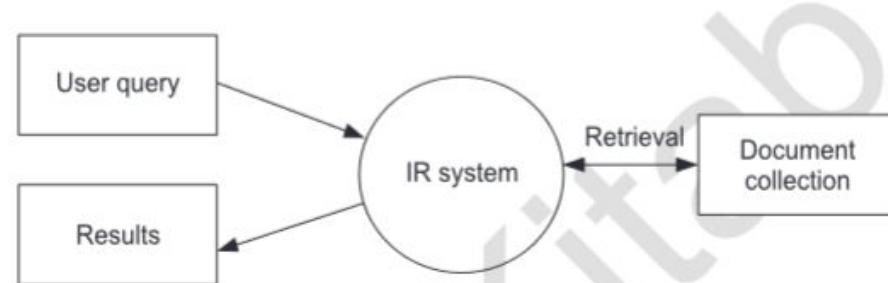
# Conceptual Dependency

- ATRANS: Transfer of some relationship for verb (e.g.) 'give'
- PTRANS: Transfer of location of an object, say the verb 'go'
- GRASP: This primitive occurs with verb like 'throw'
- SPEAK: This primitive has some sound production, say the verb 'speak' or 'say'

# Information retrieval

Let us first understand the problem of information retrieval. So, what is it? Suppose you want to retrieve information about first players who scored 200+ runs in a single inning in ODI. So, you are in discussion with your friend who knows about cricket. First, you will get information about all players who scored 200+ runs, and then you will get years in which they scored. This simple retrieval will, conclude that Sachin Tendulkar is the first player to score 200+ runs. Similarly, we come across various scenarios in everyday life which demand information retrieval.

A very common example that we have discussed in the introduction is of the search engines. The search engine performs information retrieval (IR). We type in a query and the information required by us is retrieved. Figure 12.8 depicts the basic IR system.



**Figure 12.8** An IR system—Getting relevant documents.

We have talked about the search, but the question here is—is it possible to have an optimal way that can be exploited by the search engines for retrieval? What could it be and how would the things work here?

# Information retrieval - Models

## *Boolean Model*

In Boolean model, the tokens are treated in the form of 1's and 0's. The presence or absence of a term is marked accordingly. So, the document is represented in the form of  $t_1 \wedge t_2 \dots t_n$ , where  $t_i$  forms a term. The query can be  $t_1 \wedge \sim t_2$ . The Boolean mapping is found to be the simplest one that was used in earlier systems. With the Boolean model, the requirement is to find the probability of relevance  $R$  for a document  $d$  and a query  $q$ . It is represented as follows:

$$P(R | d, q), \text{ where } R \text{ can be true or false.}$$

But it has a lot of drawbacks. Since it is dependent on the bit presence, the ordering becomes an issue. (In what way the relevant ones are to be presented?). Further, matching similar words, for example, if the terms are cardigan and sweater, then it becomes another issue. Most importantly, a question raises up here—is it possible as an end user to formulate the Boolean query? Though it is pretty complex, still, many probabilistic approaches have been developed with Boolean models.

# Information retrieval - Models

## *Bag of Words*

The frequency of a word is taken into account instead of ordering. Consider the following texts:

Rohan drives Audi. (text 1)

Sameer drives BMW. He drives Mercedes also. (text 2)

(At a very basic level, to understand the concept, the words identified here are Rohan, drives, Audi, Sameer, he, Mercedes, also). The dictionary or the bag of words has these tokens. So, a vector for the text 1 and 2 would be

[1, 1, 1, 0, 0, 0, 0] for text 1

[0, 2, 0, 1, 1, 1, 1] for text 2

The count in the vectors is the frequency of occurrence of the identified terms in the text.

# Information retrieval - Models

## Vector Space Model

In vector space model, the documents and the query are represented as *vectors*. A vector comprises dimensions that are the terms used to index it. The model is given below:

Vector comprising the terms  $\langle t_1, t_2, \dots, t_n \rangle$

The document and query are formed as  $\langle p_1, p_2, p_3, \dots, p_n \rangle$ , where  $p_x$  represents the weight of the term  $x$  in the document. The query is represented as  $\langle q_1, q_2, q_3, \dots, q_n \rangle$ .

Relevance of a document  $d$  to the query  $q$  is computed based on the similarity between them. There are various functions to do so. The different ways to compute the similarity measures are cosine, dot product, dice, Jaccard and so on.

Let us proceed with the matrix representation in vector space model. Figure 12.9 shows the matrix.

		Terms			
		$p_{11}$	$p_{12}$	$\dots$	$p_{1n}$
		$p_{21}$	$p_{22}$	$\dots$	$p_{2n}$
Documents		$\vdots$			
		$p_{m1}$	$p_{m2}$	$\dots$	$p_{mn}$
Query		$q_1$	$q_2$	$\dots$	$q_n$

# Information retrieval - Models

**Term-frequency, document frequency and inverse document frequency:** The concept of term frequency and inverse document frequency is highlighted now. Most often what is observed is when we refer to the term, the weights are the occurrence of the terms in the matrix. So, the *term frequency* is defined as the frequency of occurrence of the term in the document. It is represented as *tf*. The document frequency tells about the number of documents that have a specific term. It is represented as *df*. Inverse document frequency determines the unevenness of the term in the distribution throughout the documents. This is represented as *idf*. The weights are now computed as follows:

$P_{11}$  = Weight of 1st term in document 1

So, to make it more generalised, we have (Remember the term frequencies are normalised by maximum frequency count of that term.)

Weight of a term in  $d$  =  $wt(t, d) = tf(t, d) * idf(t)$

where  $tf(t, d)$  is the occurrence of the term  $t$  in  $d$ .

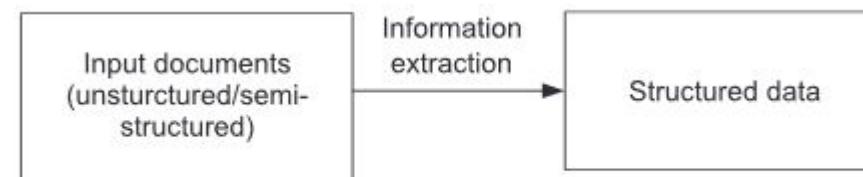
$idf(t) = \log (\text{Total number of documents/Documents containing } t \text{ term})$

## Probabilistic Model

Bayesian probability is the most common mechanism that is used in probabilistic inferring. The major drawback of using the Bayesian approach is that it requires prior knowledge. But it is able to address the uncertainty factor that could occur while submitting the query.

# Information Extraction

In information retrieval, we have studied about getting the information—the search engines being the common example. Now, what does information extraction (IE) do? Is it not the case that information extraction and retrieval are the same? Do remember that in information retrieval, the data is not at all modified or changed. Simply, the relevant data is presented to the user. In IE, template matching is carried out. The IE module could make entries in the databases. So, there is a pre-defined fixed format in which the text entries are carried out. In short, IE makes things structured from unstructured inputs. With IE, the information from the documents is extracted into the templates. Are you feeling like there is some similarity between retrieval and extraction? IR gets relevant documents, whereas IE gets relevant information from the documents. Though there could be an overlap between them. IE could lie between IR and text understanding. Figure 12.12 depicts the basic IE system.



**Figure 12.12** Information extraction.

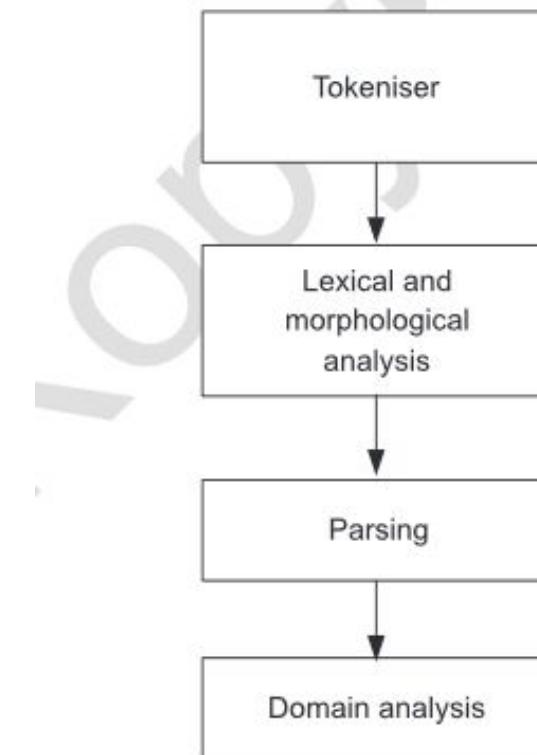
# Information Extraction

Let us take an example to explain IE. An employment agency has to send mails to the clients whose profile suit some ABC company's requirements. From the CVs of these clients, only the mail-id needs to be extracted to send the date for interview. In another example, a company wants to categorise candidates based on their skill set. In this case, the information about the skills is retrieved from their CVs. Here, terms like Java, .Net etc. are used for template matching. This is also a task of IE.

There are various aspects of IE such as what is to be extracted?, what is the input data? what methods are to be used?, How is the output expected? Let us try to address them. To extract, it could be any particular entity, or any specific table or relations. Whereas, the input data could be a simple set of documents, web pages and so on, which is unstructured. The methods applied can be handwritten patterns, learning-based methods, rule-based methods or statistical methods. Figure 12.13 shows the basic modules of IE.

What is the job of each of them? Tokeniser, as we are already familiar, does the word separation. In *lexical and morphological analysis*, part of speech tagging is done. Identifying the parts of speech or learning the word sense is carried out here. In *parsing*, syntax analysis is done, whereas in *domain analysis*, merging the partial results or co-reference is carried out.

FRUMP, LSP were the early IE systems. Later on, systems for web-based information extraction were Crunch, Content Seeker, etc. At present, wrappers are used on a large scale. They are called *information extraction procedures* that extract some content or bits from the text. Meta-crawlers are the type of wrappers. Currently, the focus of the research is on investigating and inventing different techniques for text summarisation as well as on cross-media and language-based IE systems.



**Figure 12.13** Modules of IE.

- Advance topics in Artificial Intelligence- Cloud Computing and Intelligent agent
- Business Intelligence and Analytics
- Sentiment Analysis
- Deep Learning Algorithms
- Planning and Logic in intelligent Agents

# Advance topics in Artificial Intelligence- Cloud Computing and Intelligent agent

## Cloud computing

The practice of using a network of remote servers hosted on the Internet

to:

- store,
- manage,
- and process data,

rather than a local server or a personal computer.



# Cloud computing and AI(contd.)

## Cloud computing and AI

While artificial intelligence (A.I.) has struggled to gain footholds in other niches, it is finding its place in the world of cloud computing, a sort of revolution within the revolution that could rapidly change the face of businesses using cloud computing solutions over the next few years.



#190988452

# Cloud computing and AI(contd.)

In three areas of cloud computing, A.I. is taking long strides.

Those areas are

- Parallel processing
- Machine Learning-ML Algorithms
- Big Data

# What's parallel processing and how it work in cloud

- Parallel processing means more than one microprocessor handling parts of the same overall task. Parallel processing essentially means that multiple processors shoulder the load. To have multiple processors working on the same problem at the same time, there are two big things you need:

- Latency
- Bandwidth

# What's parallel processing and how it work in cloud(contd.)

## Latency

it refers to the amount of time it takes for a processor to send results back to the system. The longer the wait, the longer it will take the entire system to process the problem.

## Bandwidth

Bandwidth is a more common term, referring to how much data a processor can send in a given length of time.

# ML algorithms for cloud applications

**Machine learning (ML)** is a type of **artificial intelligence (AI)** that allows software applications to become more accurate in predicting outcomes without being explicitly programmed

For cloud applications Machine Learning algorithms are built

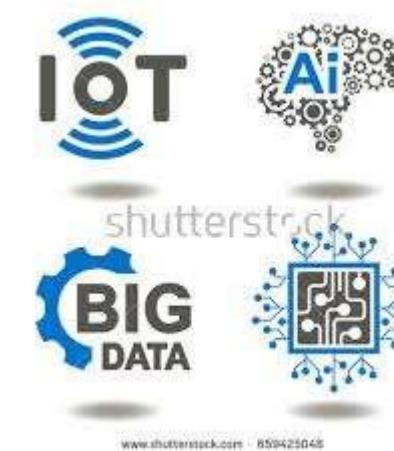
# ML algorithms for cloud applications(contd.)

ML algorithms for cloud applications involve:

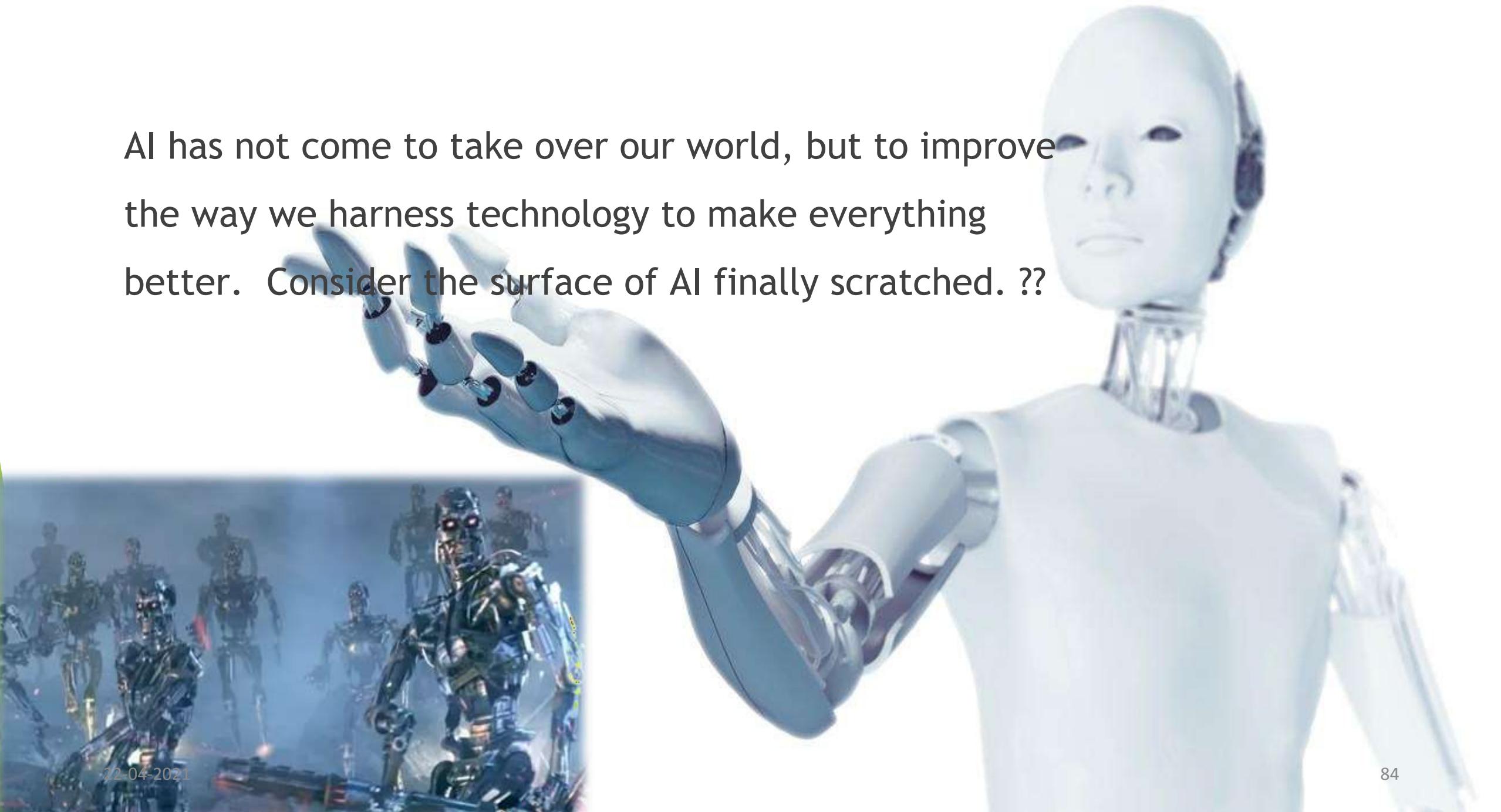
- **Cognitive computing**(to combine different patterns together; i.e. voice, imagery or other such data; for mimicking human behavior)
- **Chatbots and virtual assistants** (they are getting smarter every time they have a conversation)
- **Internet of things-IoT** (It connects every potentially “smart” machine in the world to the cloud and add that massive amount of data to the conversation)

# How AI uses big data

As business enterprises increasingly need a massive data-crunching champion, cloud computing companies have begun to deploy Artificial Intelligence as a service (AlaaS). Once AlaaS is deployed, it can begin crunching data at a faster rate than any single microprocessor or human mind could ever hope to compete with.



AI has not come to take over our world, but to improve the way we harness technology to make everything better. Consider the surface of AI finally scratched. ??



# Business Intelligence and Analytics

So, how does AI actually work in the business world? let's try to understand what artificial intelligence is and why it is so important for today's business corporations.



# What is Business Intelligence (BI)?

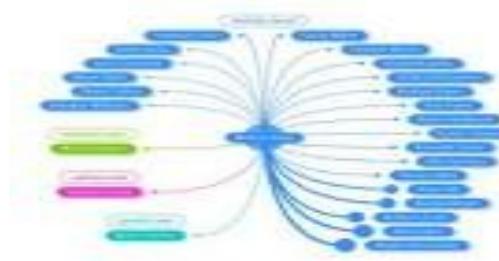
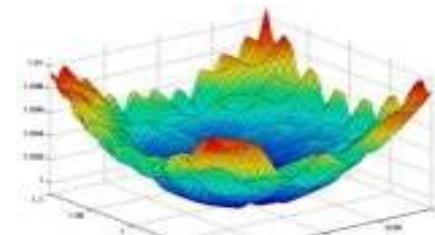
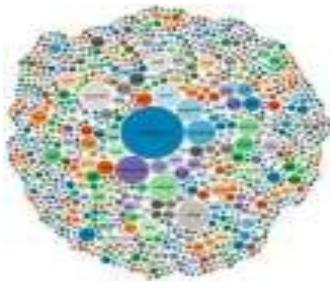


Business intelligence systems are used to maintain, optimize and streamline current operations. BI improves and maintains operational efficiency and helps businesses **increase organizational productivity**. Business intelligence software confers many benefits, notably **powerful reporting and data analysis** capabilities. Using BI's rich visualization mechanisms, managers are able to generate intuitive, readable reports that contain relevant, actionable data.

*Popular business intelligence solutions include; SAP BusinessObjects, QlikView, IBM Cognos, Microstrategy, etc.*

<https://selecthub.com/business-intelligence/business-intelligence-vs-business-analytics/>

# What is Business Analytics (BA)?



Like business intelligence, BA collects and analyzes data, employs **predictive analytics** and generates **richly visualized reports**, helping identify and address an organization's weak points. That's where similarities end. Business analytics software is used to explore and analyze historical and current data. It utilizes **statistical analysis, data mining** and quantitative analysis to identify past business trends.

*Popular business analytics solutions include; SAP Business Analytics Suite, Pentaho BA, Birst BI and Tableau Blg Data Analytics.*

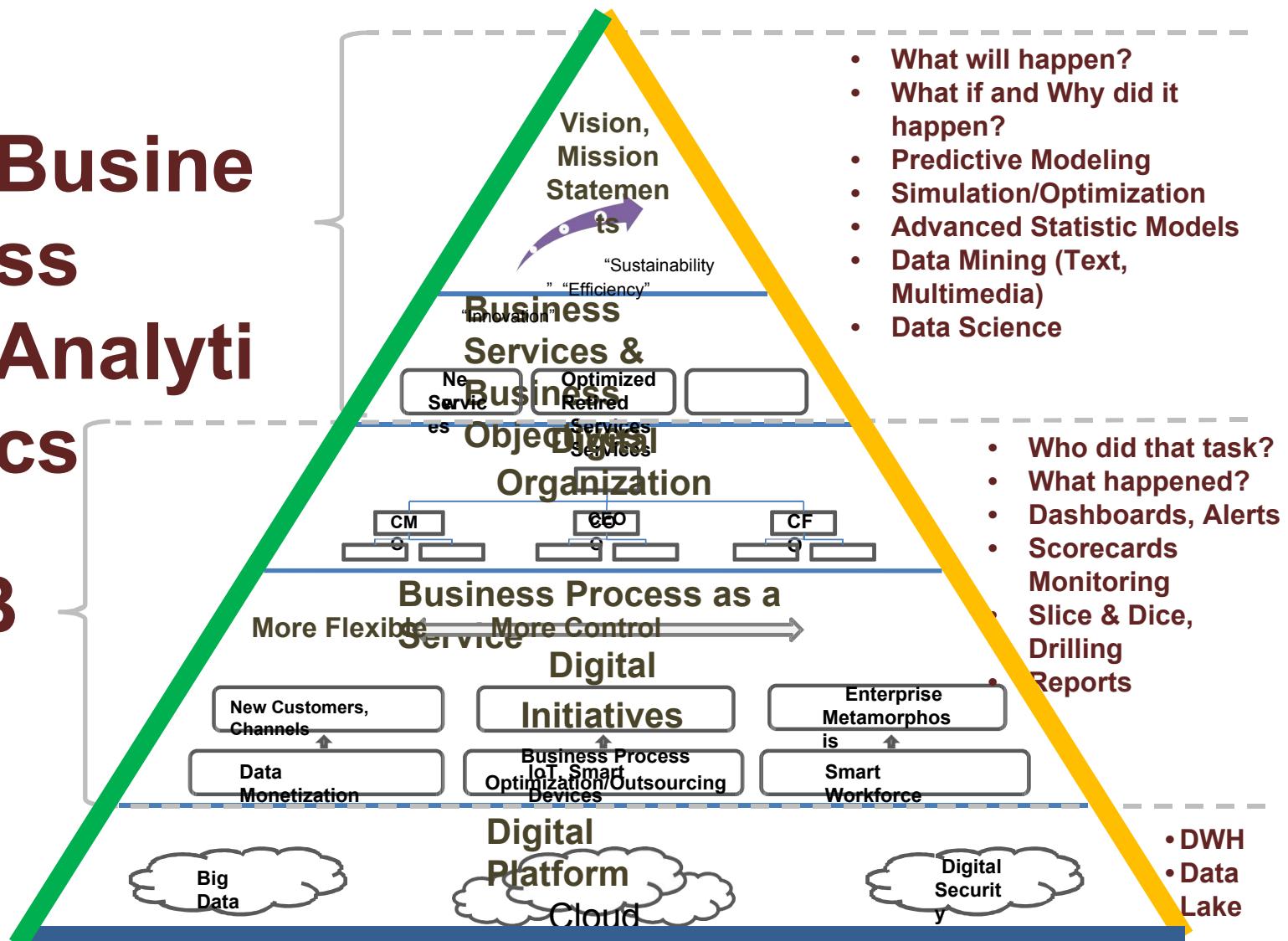
<https://selecthub.com/business-intelligence/business-intelligence-vs-business-analytics/>

# Business Analytics vs. Business Intelligence

**Business Analytics**

**CS**

**BI**



# Choosing between Business Intelligence (BI) and Business Analytics (BA)

While superficially similar, the difference between business intelligence vs business analytics is clear:

- BI uses past and **current data to optimize the present for current success.**
- BA uses the past and analyzes the present to **prepare businesses for the future.**

Choosing the solution for your business depends on your aims.

- If you are satisfied with your business model as a whole and mainly wish to improve operations, **increase efficiency and meet organizational goals, business intelligence may be an optimal solution.**
- If you intend to **change your business model and need to know where to start, business analytics might be the best option.**

<https://selecthub.com/business-intelligence/business-intelligence-vs-business-analytics/>

# Choosing between Business Intelligence (BI) and Business Analytics (BA)

## **Business Intelligence (BI)**

BI has the added advantages of targeting a business's weak areas and providing actionable solutions to those problems. Business Intelligence software is an excellent solution for managers who want to improve decision making and understand their **organization's productivity, work processes and employees. And, with that understanding, improve their business from the ground up.**

## **Business Analytics (BA)**

If your organization is a new entity, or in the midst of significant changes, business analytics software is a serious contender. BA uses historical data, current information, and projected trends to ensure your business makes the right changes. Business analytics is the solution if you want to **analyze your company, your market, and your industry with the dual goals of optimizing current performance and predicting business trends to help you remain competitive in the future.**

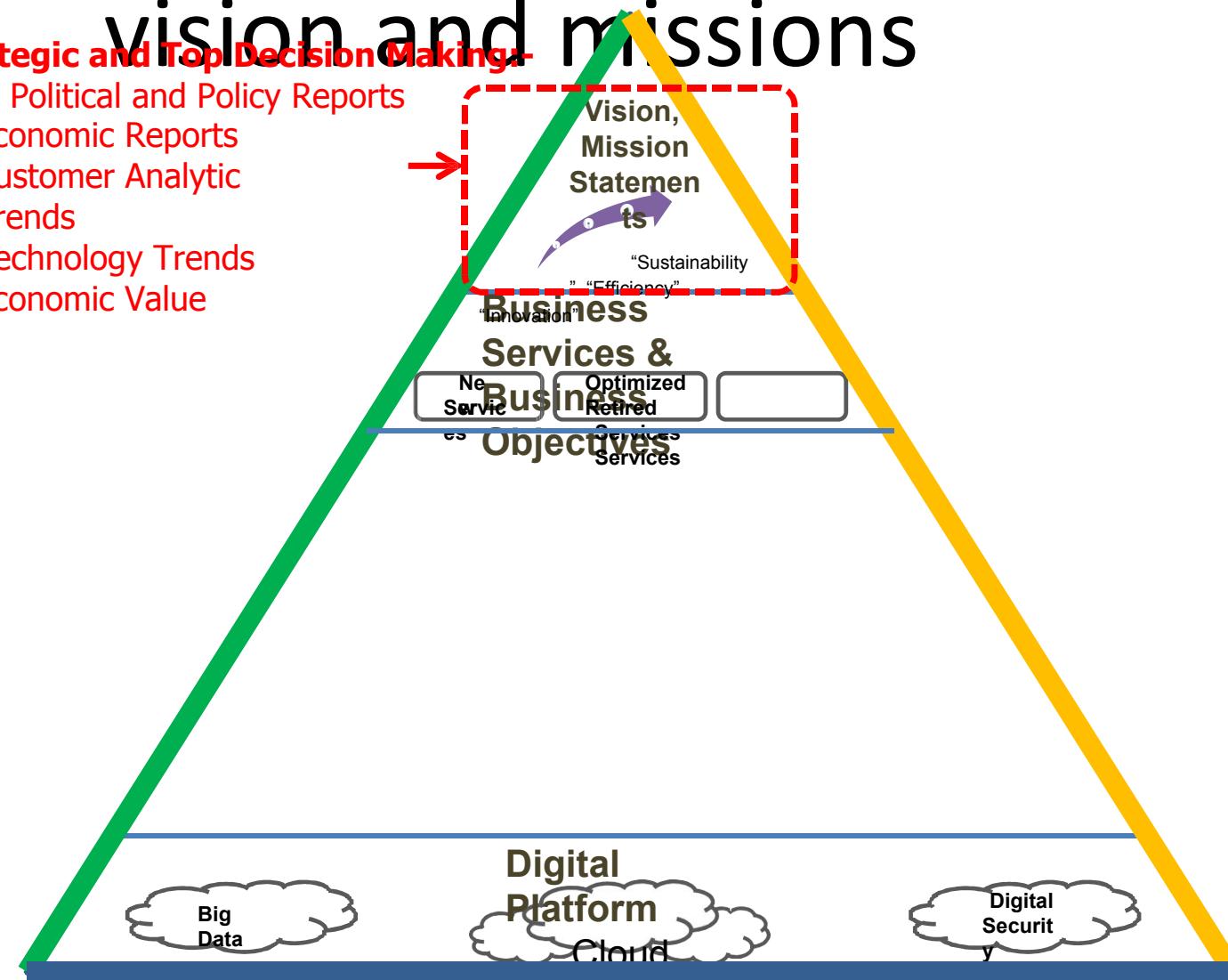
*Most businesses want a combination of current success and future preparation. Alone or together, business analytics and business intelligence can help you take your business where you want it to go.*

<https://selecthub.com/business-intelligence/business-intelligence-vs-business-analytics/>

# 1. (Re)Identifying your vision and missions

## Strategic and Top Decision Making:-

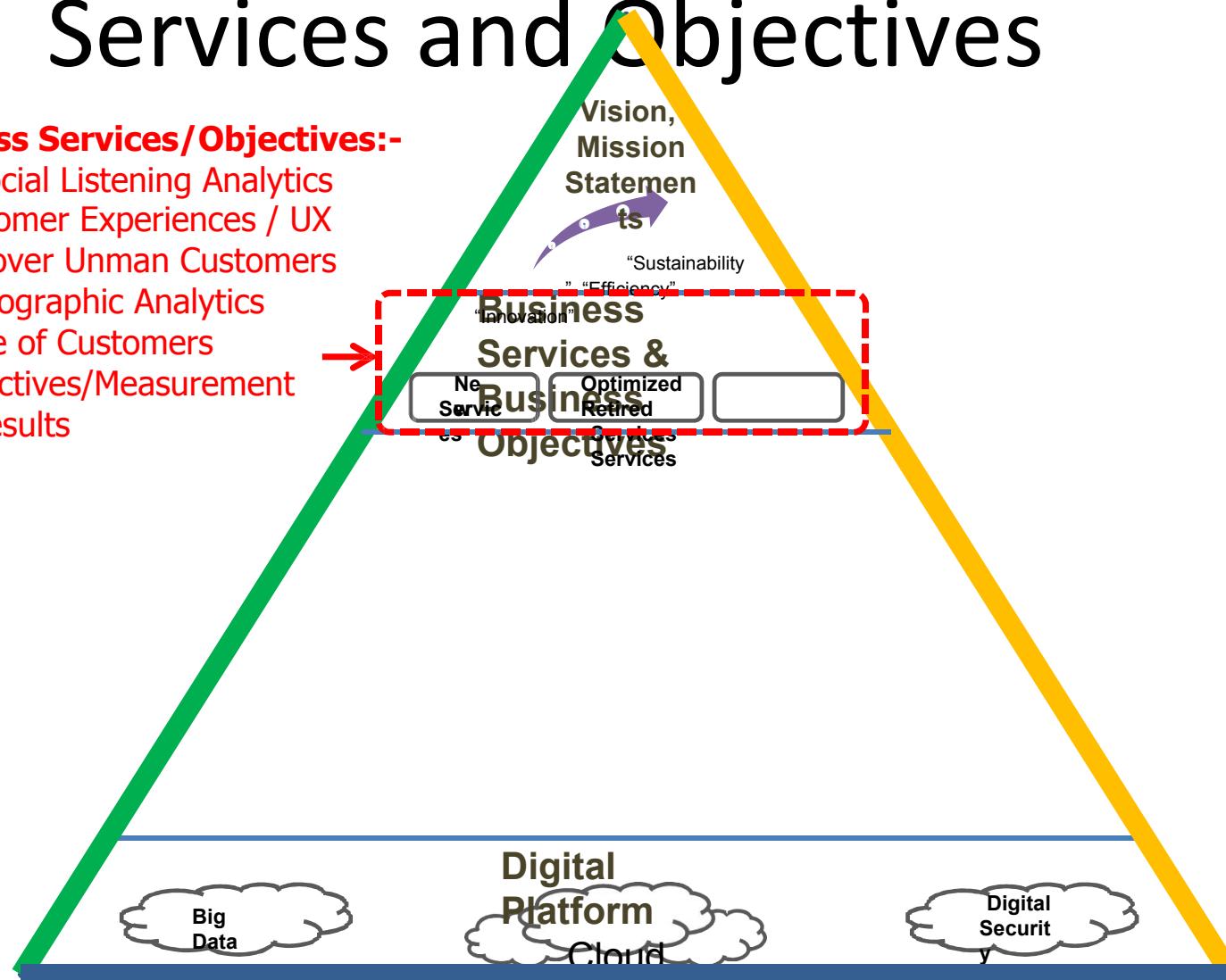
- Political and Policy Reports
- Economic Reports
- Customer Analytic Trends
- Technology Trends
- Economic Value



# 2. Identifying Business Services and Objectives

## Business Services/Objectives:-

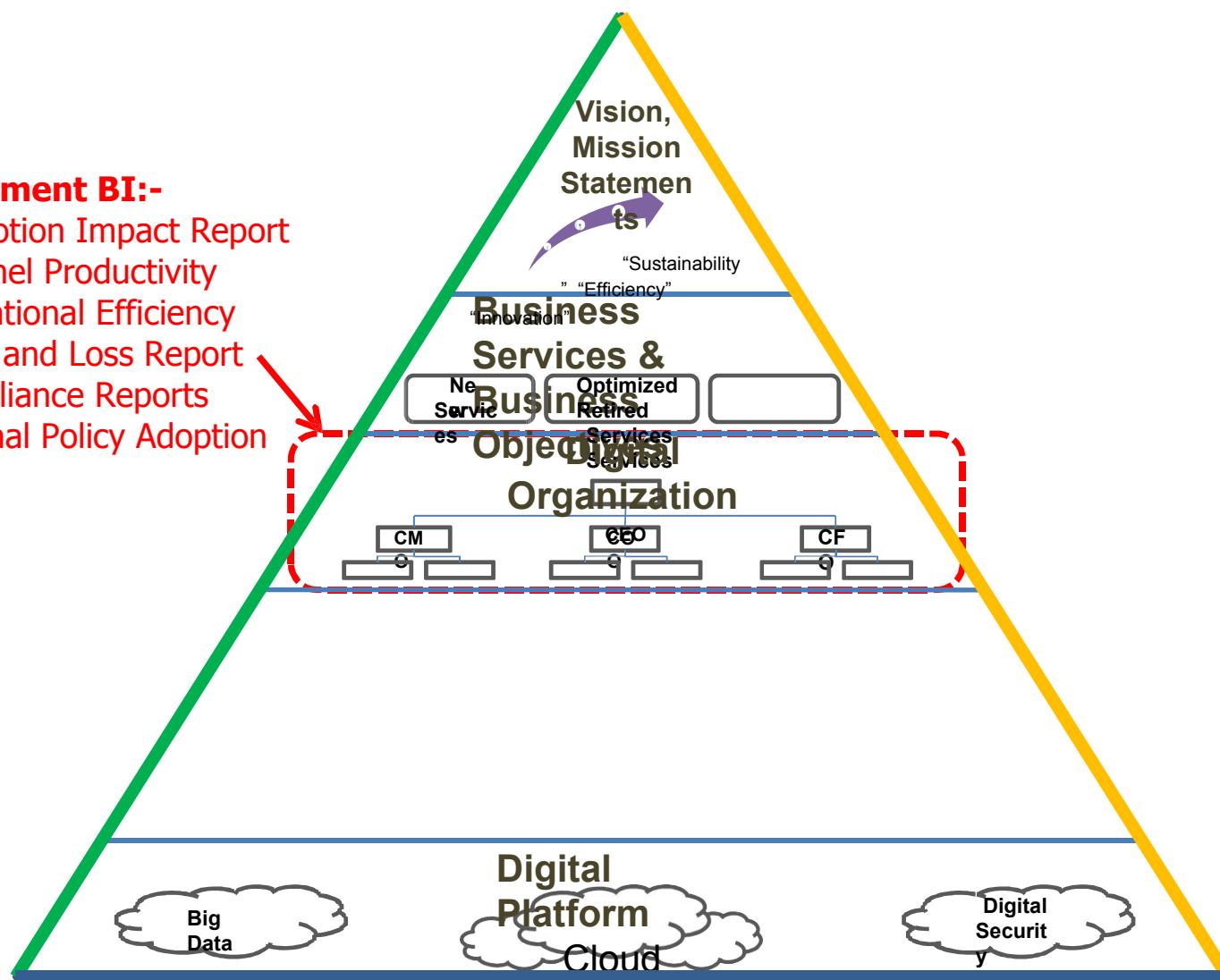
- Social Listening Analytics
- Customer Experiences / UX
- Discover Unman Customers
- Demographic Analytics
- Voice of Customers
- Objectives/Measurement s Results



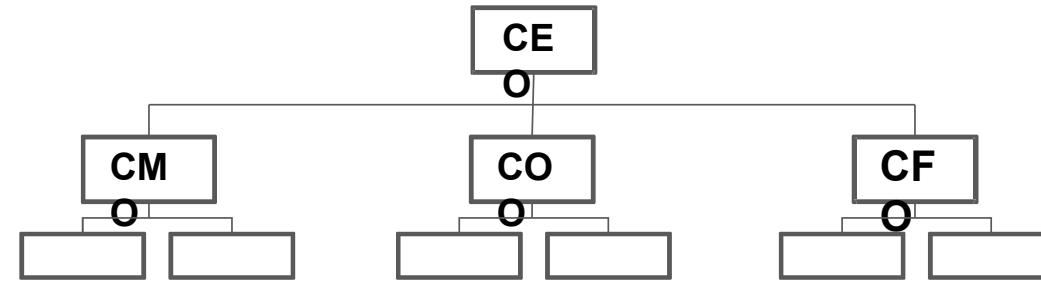
### 3. Identifying BI for Management Level

#### Management BI:-

- Promotion Impact Report
- Channel Productivity
- Operational Efficiency
- Profit and Loss Report
- Compliance Reports
- Internal Policy Adoption

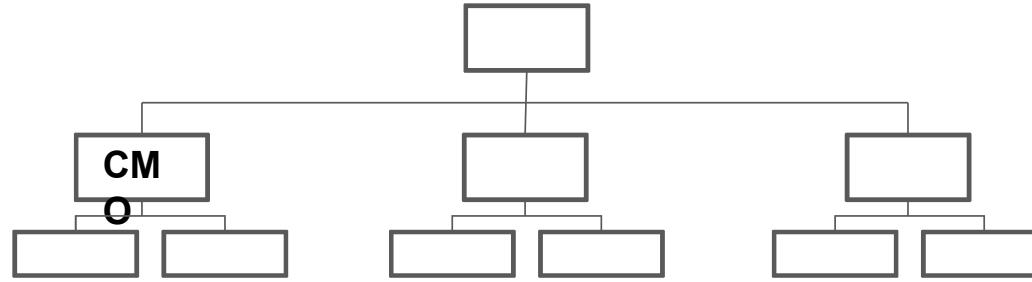


# Digital Organization



- CEO : combine all successes from all C-Level
- CMO: innovation for new products offering
- COO : operation and automation
- CFO : finance, budgeting, HR, Audit, QA and IT
- Put the right skill on the right role
- **Promote paperless policy organization**

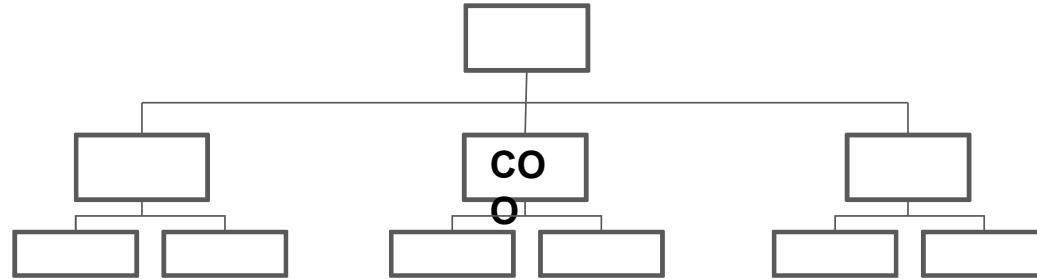
# Top Business Questions from CMO



## **Chief Marketing Officer (CMO), Innovation, Sales and Promotion:-**

- Which customers should we target?
- What has caused the change in my pipeline?
- Which are my most profitable campaigns/region?
- Did store sales spike when we advertised in the local paper or launched the campaign?
- What is the most profitable sales channel and how has that changed over time?

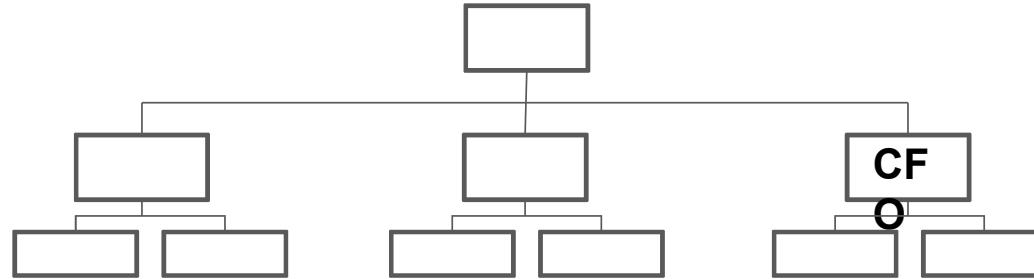
# Top Business Questions from COO



## **Chief Operation Officer (COO):-**

- Lead time and cost of production for each products
- Which order processing processes are most inefficient?
- Which vendors are best at delivering on time and on budget?—
- How many additional personnel do we need to add per branch?
- Percent of error or defect trend for each product

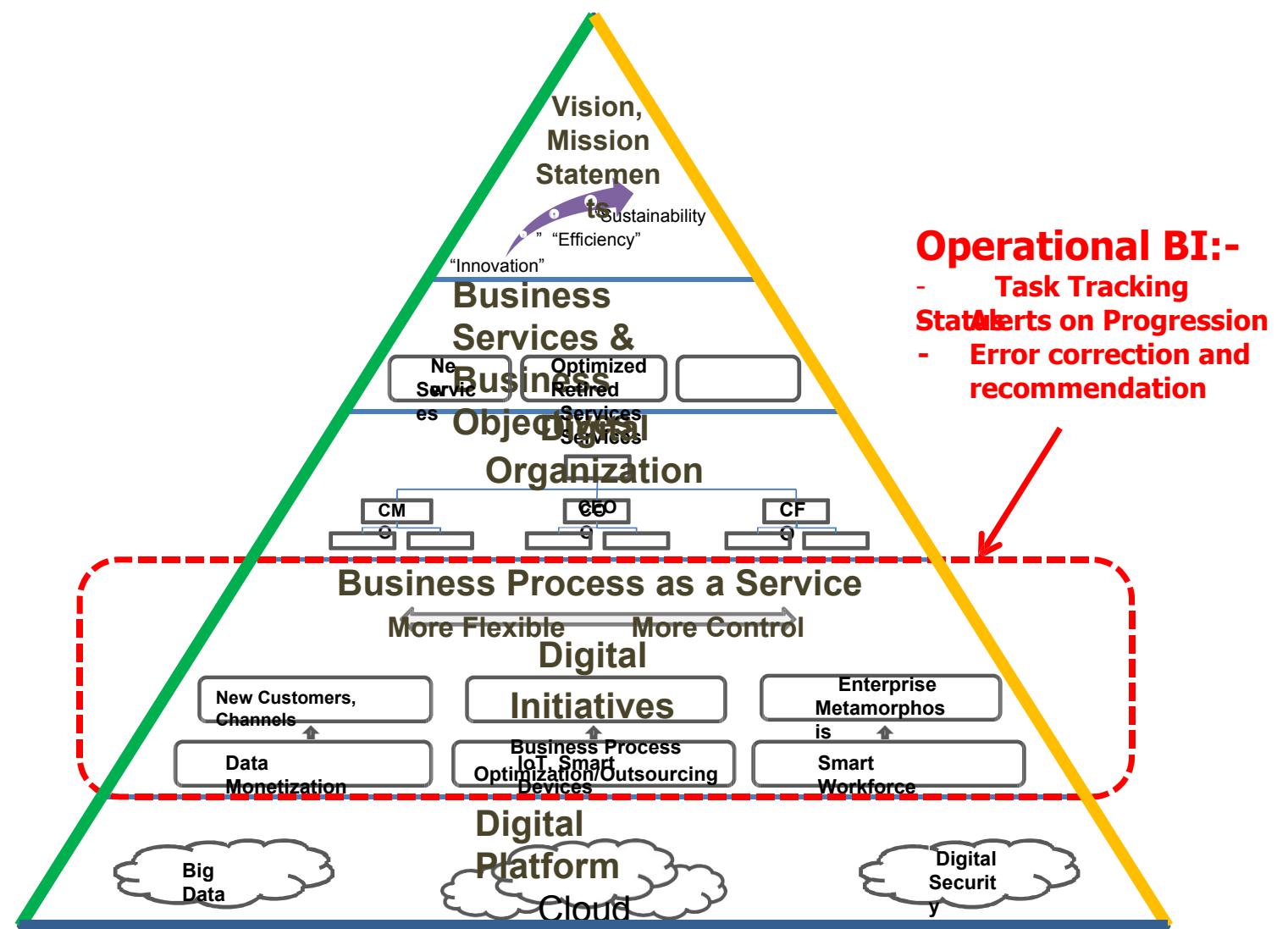
# Top Business Questions from CFO



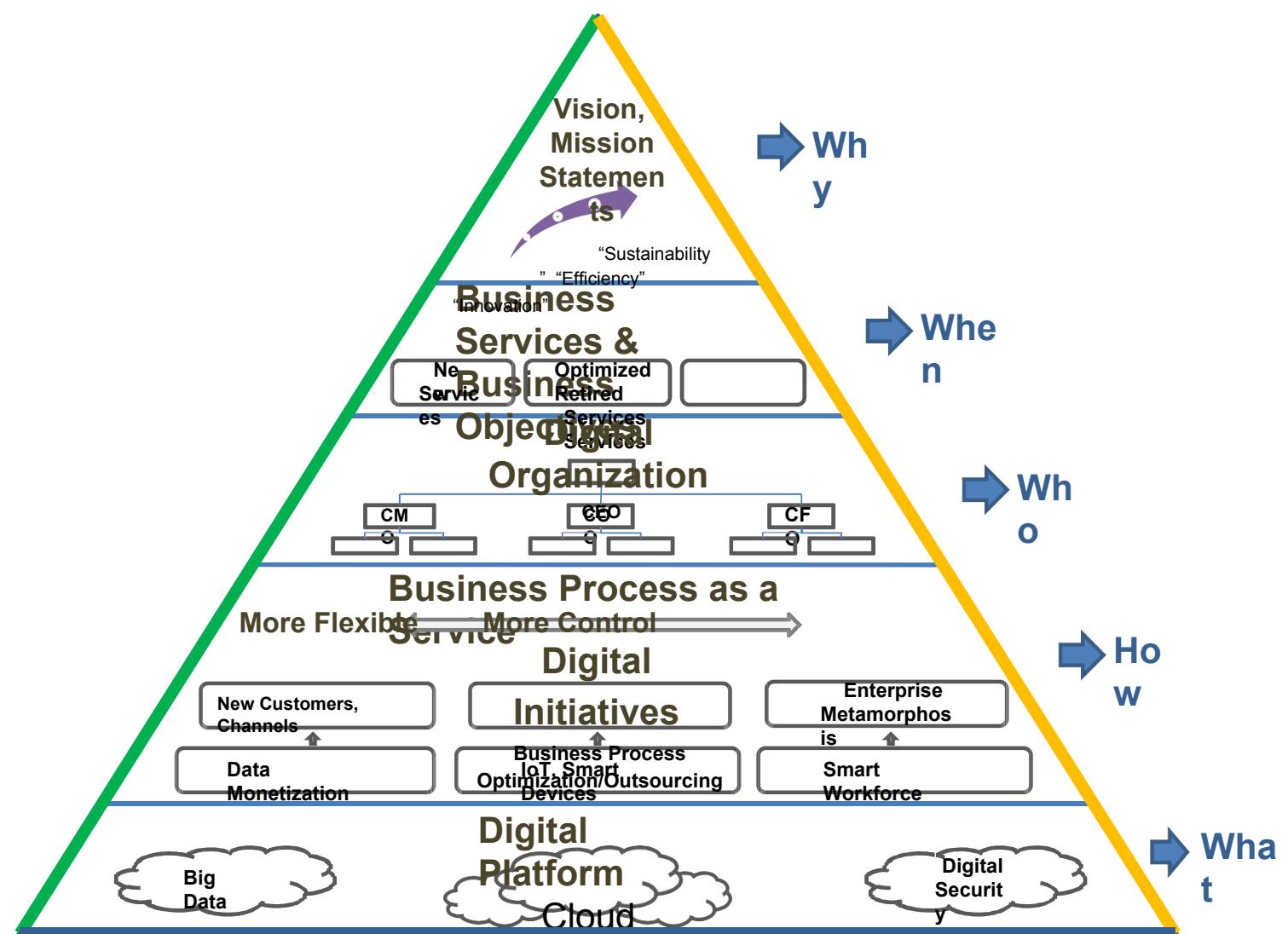
## **Chief Financial Officer (CFO):-**

- What is the fully loaded cost of new products deployment?
- What are the current trends in cash flow, accounts payable and accounts receivable and how do they compare with plan?
- What is the expected annual profit/loss based on current marketing and sales forecasts?
- How are forecasts trending against the annual plan?

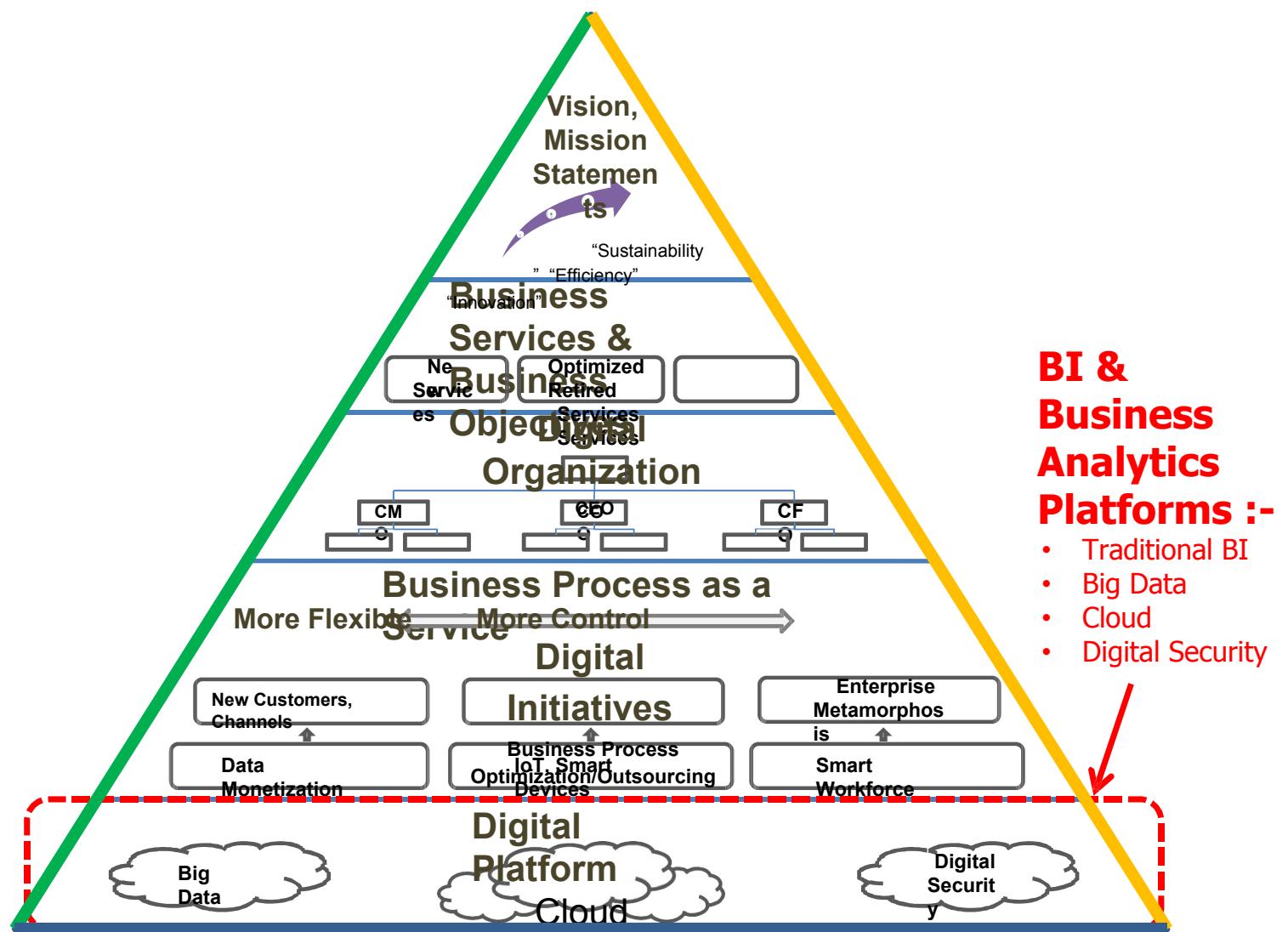
## 4. Identifying Operational BI



# Key Questions Type in each level of enterprise



## 5. Identifying BI and BA Platform

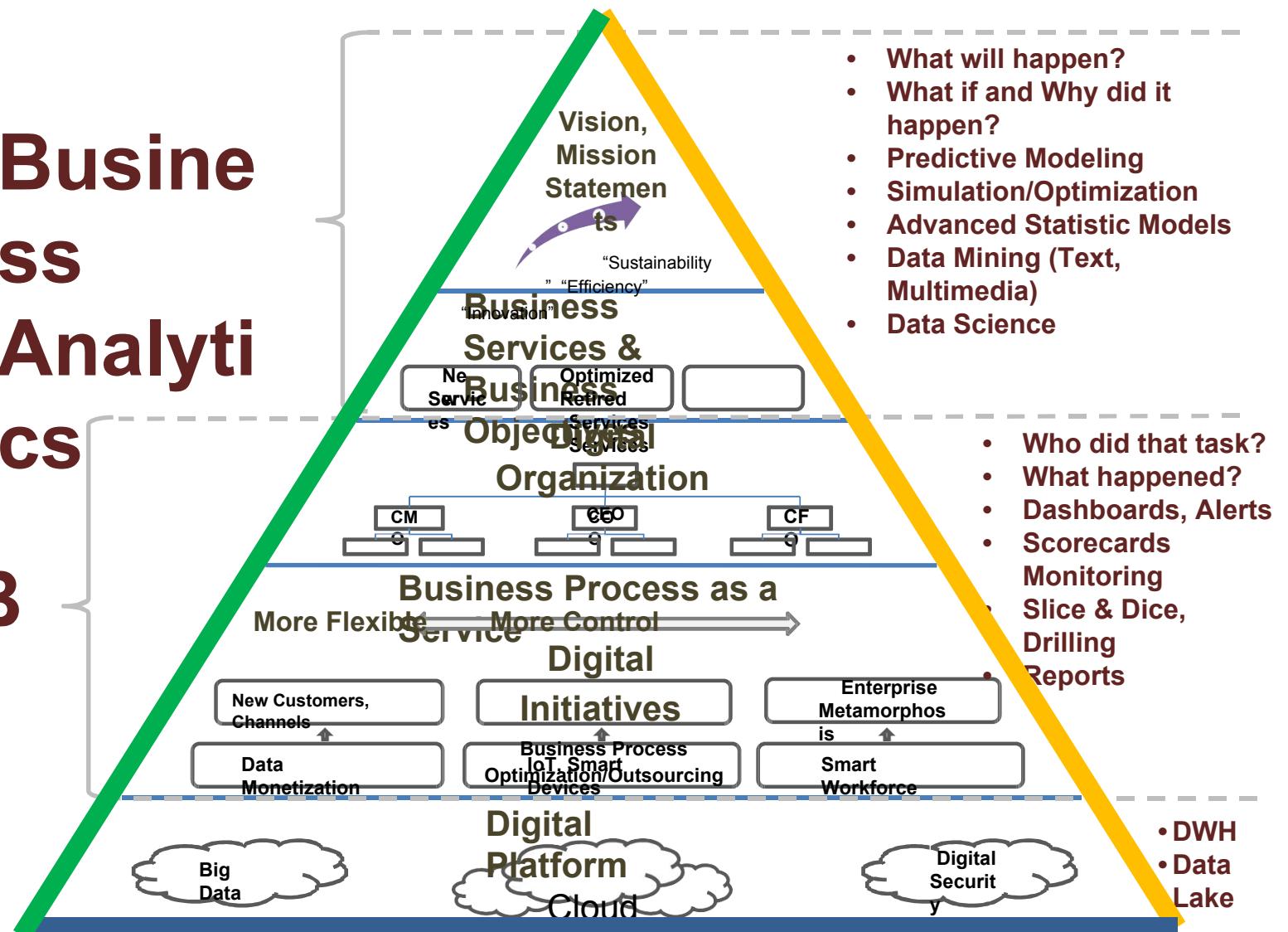


### BI & Business Analytics Platforms :-

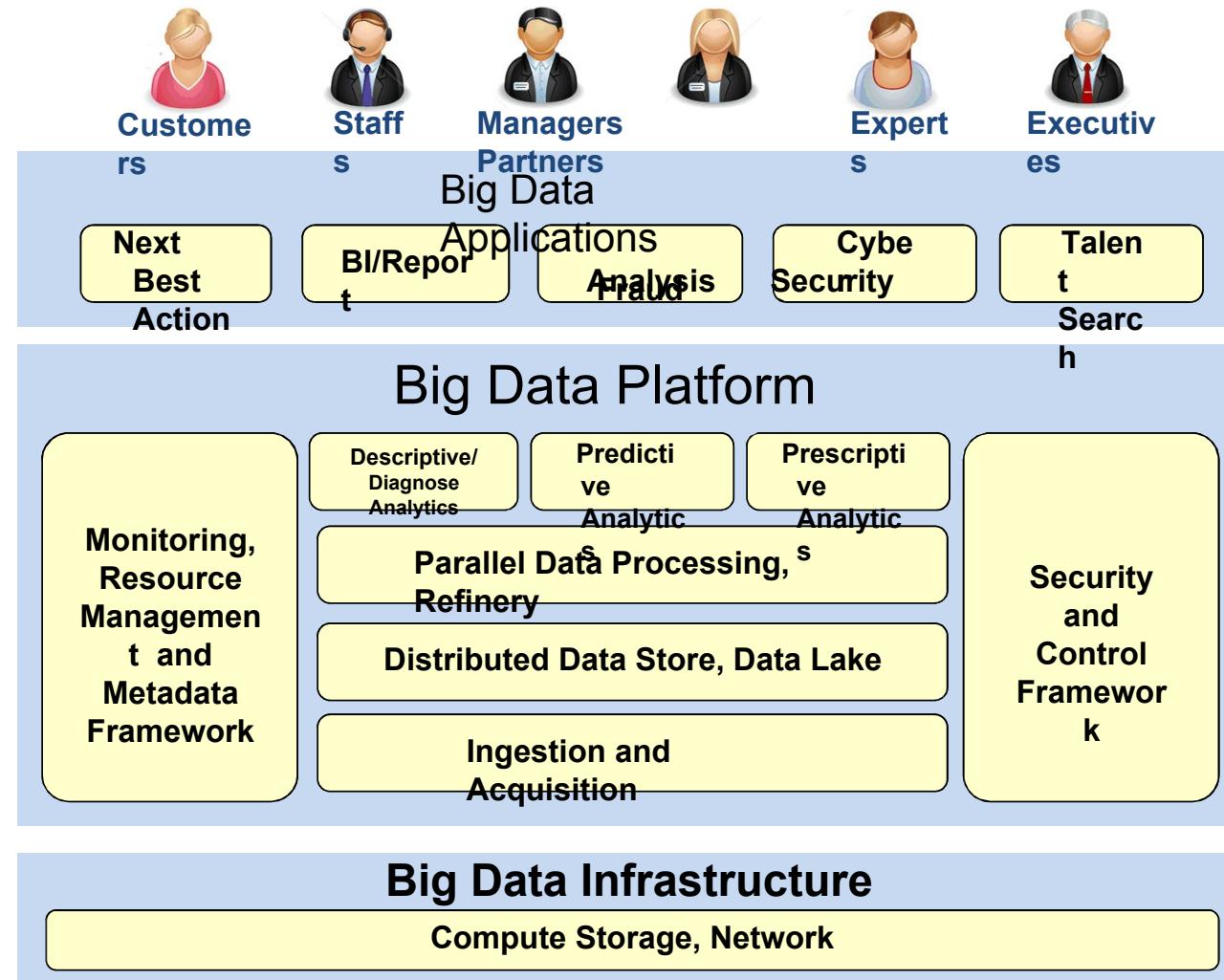
- Traditional BI
- Big Data
- Cloud
- Digital Security

# Business Analytics

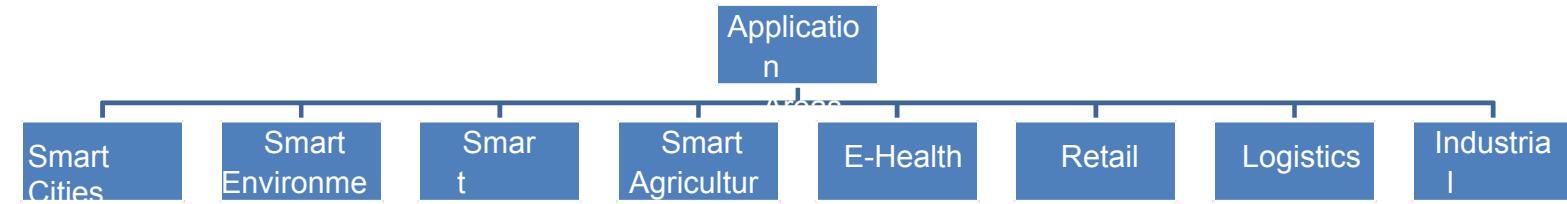
## Summary



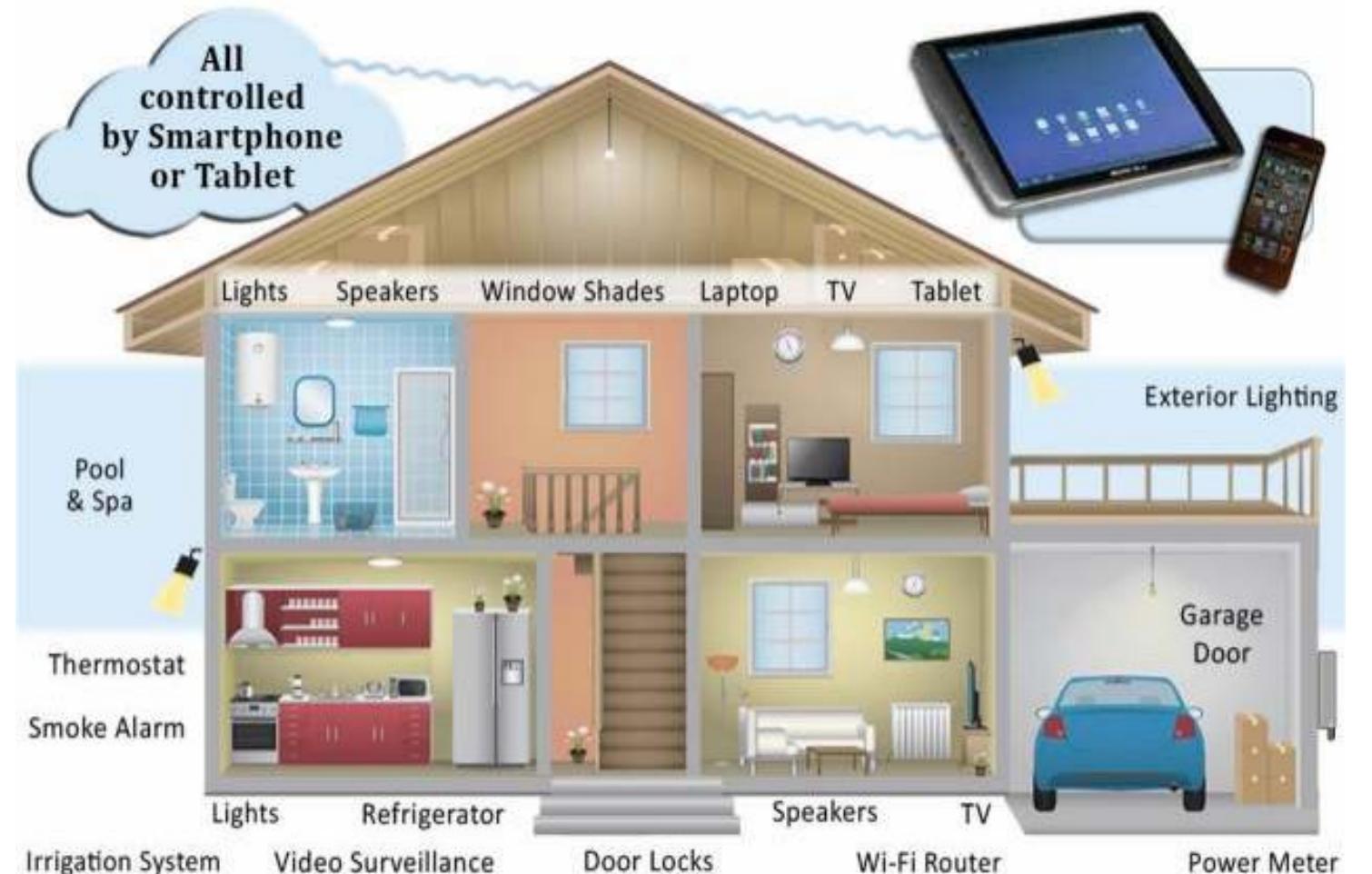
# Big Data for Business Analytics Platform



# Internet of Things and Real-time Data Feeds



# Smart Home



Based on these capabilities, we have seen multiple applications of artificial intelligence in business in the form of:



Chatbots, virtual assistants, and business intelligence bots



Targeted online advertising



Predictive analytics



Voice recognition



Pattern recognition

# Sentiment Analysis

# Sentiment Sentiment Analysis

A thought, view, or attitude, especially one based mainly on emotion instead of reasonSentiment Analysisaka opinion mininguse of natural language processing (NLP) and computational techniques to automate the extraction or classification of sentiment from typically unstructured text



# What is Sentiment Analysis

- ✓ Sentiments are feelings, opinions, emotions, likes/dislikes, good/bad
- ✓ Sentiment Analysis is a *Natural Language Processing and Information Extraction task* that aims to obtain writer's feelings expressed in positive or negative comments, questions and requests, by analyzing a large numbers of documents.
- ✓ Sentiment Analysis is a study of human behavior in which we extract user opinion and emotion from plain text.
- ✓ Sentiment Analysis is also known as Opinion Mining.



# Sentiment Analysis contd....

 Clip slide

- ✓ It is a task of identifying whether the opinion expressed in a text is positive or negative.
- ✓ Automatically extracting opinions, emotions and sentiments in text.
- ✓ Language-independent technology that understand the meaning of the text.
- ✓ It identifies the opinion or attitude that a person has towards a topic or an object.



# Example

Clip slide

## ✓ User's Opinions :

Sameer : It's a great movie (Positive statement)

Neha : Nah!! I didn't like it at all (Negative statement)

Mayur : The new iOS7 is awesome..!!!(Positive statement)

## ✓ Polarity :

- Positive
- Negative
- Complex

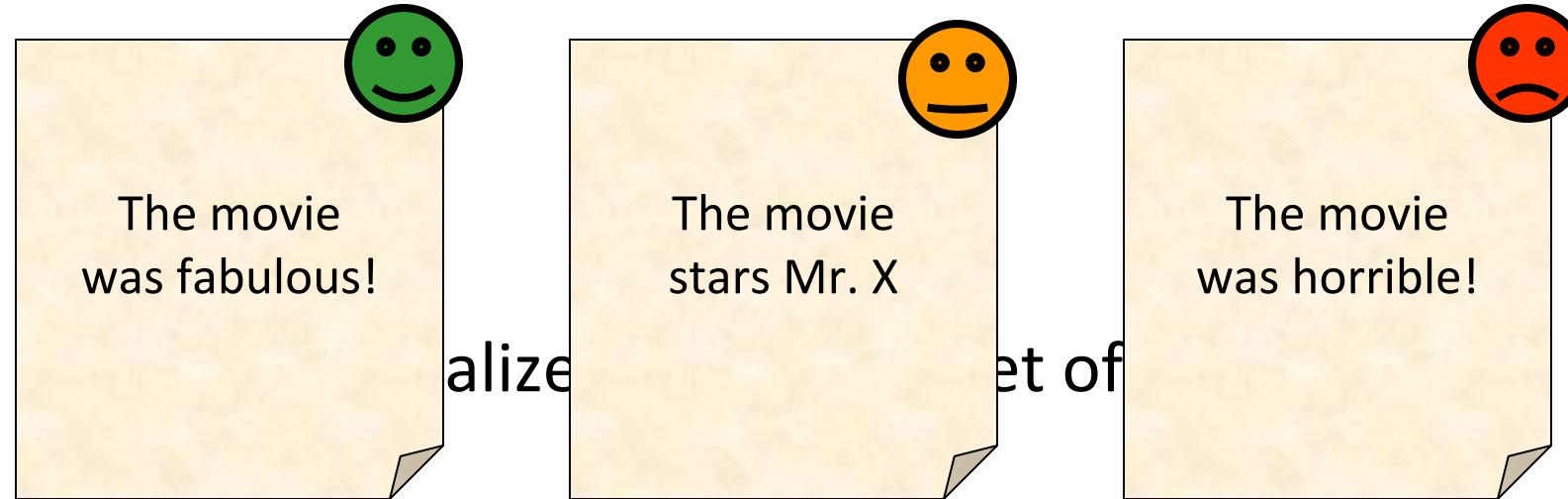


# Sentiment analysis has many other names

- Opinion extraction
- Opinion mining
- Sentiment mining
- Subjectivity analysis

# What is SA & OM?

- Identify the orientation of opinion in a piece of text



# Positive or negative movie review?



- *unbelievably disappointing*
- Full of zany characters and *richly applied satire*, and some great plot twists
- this is the greatest *screwball comedy* ever filmed
- It was *pathetic*. The worst part about it was the boxing scenes.



# Why sentiment analysis?

- *Movie*: is this review positive or negative?
- *Products*: what do people think about the new iPhone?
- *Public sentiment*: how is consumer confidence? Is despair increasing?
- *Politics*: what do people think about this candidate or issue?  
11  
5
- *Prediction*: predict election outcomes or market trends from sentiment

# Why compute affective meaning?

- Detecting:
  - sentiment towards politicians, products, countries, ideas
  - frustration of callers to a help line
  - stress in drivers or pilots
  - depression and other medical conditions
  - confusion in students talking to e---tutors
  - emotions in novels (e.g., for studying groups that are feared over time)
- Could we generate:
  - emotions or moods for literacy tutors in the children's storybook domain
  - emotions or moods for computer games

# Scherer's typology of affective states

**Emotion:** relatively brief episode of synchronized response of all or most organismic subsystems in response to the evaluation of an event as being of major significance

**angry, sad, joyful, fearful, ashamed, proud, desperate**

**Mood:** diffuse affect state ...change in subjective feeling, of low intensity but relatively long duration, often without apparent cause

**cheerful, gloomy, irritable, listless, depressed, buoyant**

**Interpersonal stance:** affective stance taken toward another person in a specific interaction, coloring the interpersonal exchange

**distant, cold, warm, supportive, contemptuous**

**Attitudes:** relatively enduring, affectively colored beliefs, preferences predispositions towards objects or persons

**liking, loving, hating, valuing, desiring**

**Personality traits:** emotionally laden, stable personality dispositions and behavior tendencies typical for a person

# Google Product Search



**HP Officejet 6500A Plus e-All-in-One Color Ink-jet - Fax / copier / printer / scanner**

\$89 online, \$100 nearby ★★★★☆ 377 reviews

September 2010 - Printer - HP - Inkjet - Office - Copier - Color - Scanner - Fax - 250 sh

## Reviews

**Summary** - Based on 377 reviews



### What people are saying

11  
8

<b>ease of use</b>		"This was very easy to setup to four computers."
<b>value</b>		"Appreciate good quality at a fair price."
<b>setup</b>		"Overall pretty easy setup."
<b>customer service</b>		"I DO like honest tech support people."
<b>size</b>		"Pretty Paper weight."
<b>mode</b>		"Photos were fair on the high quality mode."
<b>colors</b>		"Full color prints came out with great quality."

# TwiGer sentiment:

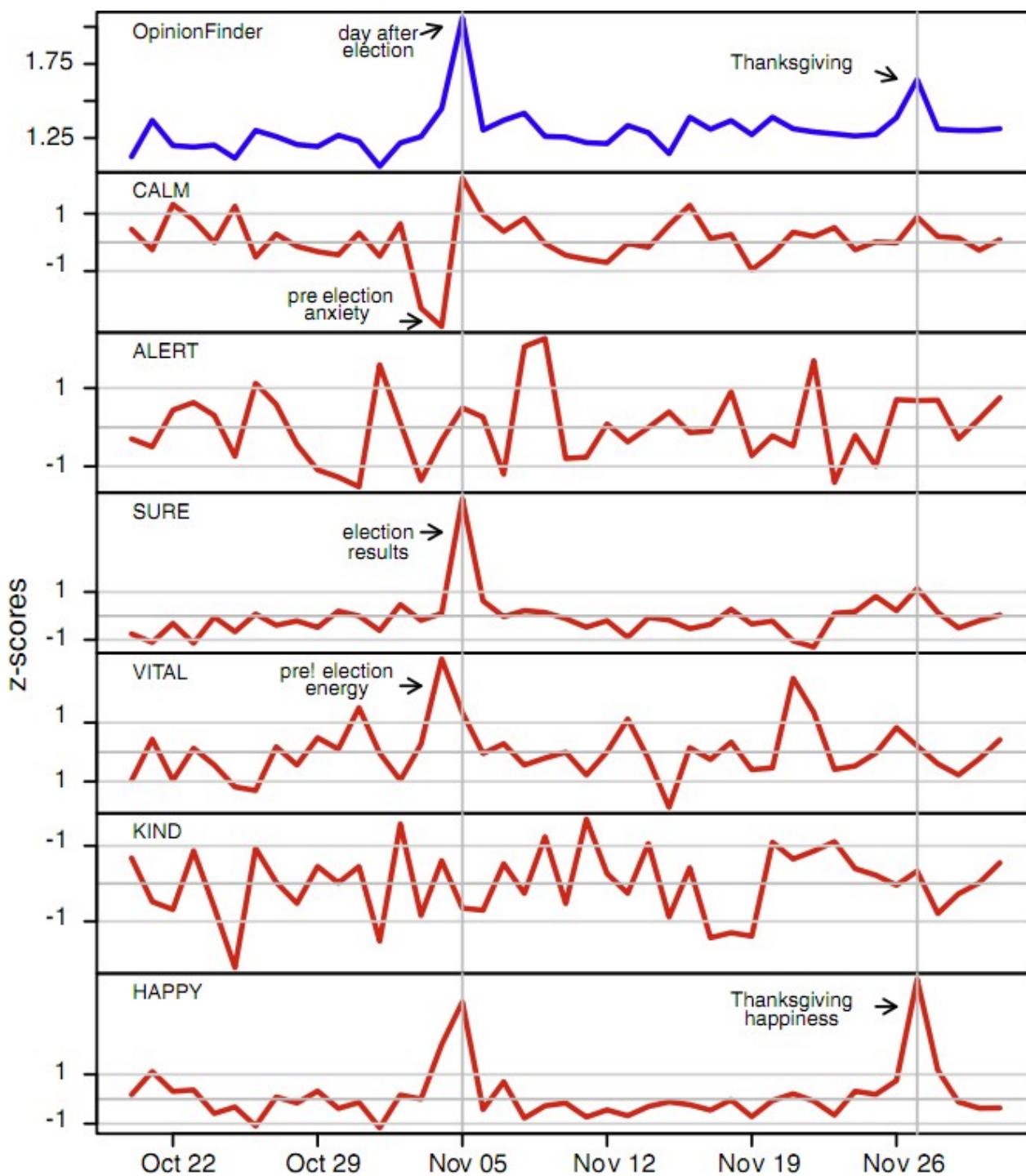
Johan Bollen, Huina Mao, Xiaojun Zeng.  
2011.

Twitter mood predicts the stock market,

Journal of Computational Science 2:1,  
1---8. 10.1016/j.jocs.2010.12.007.

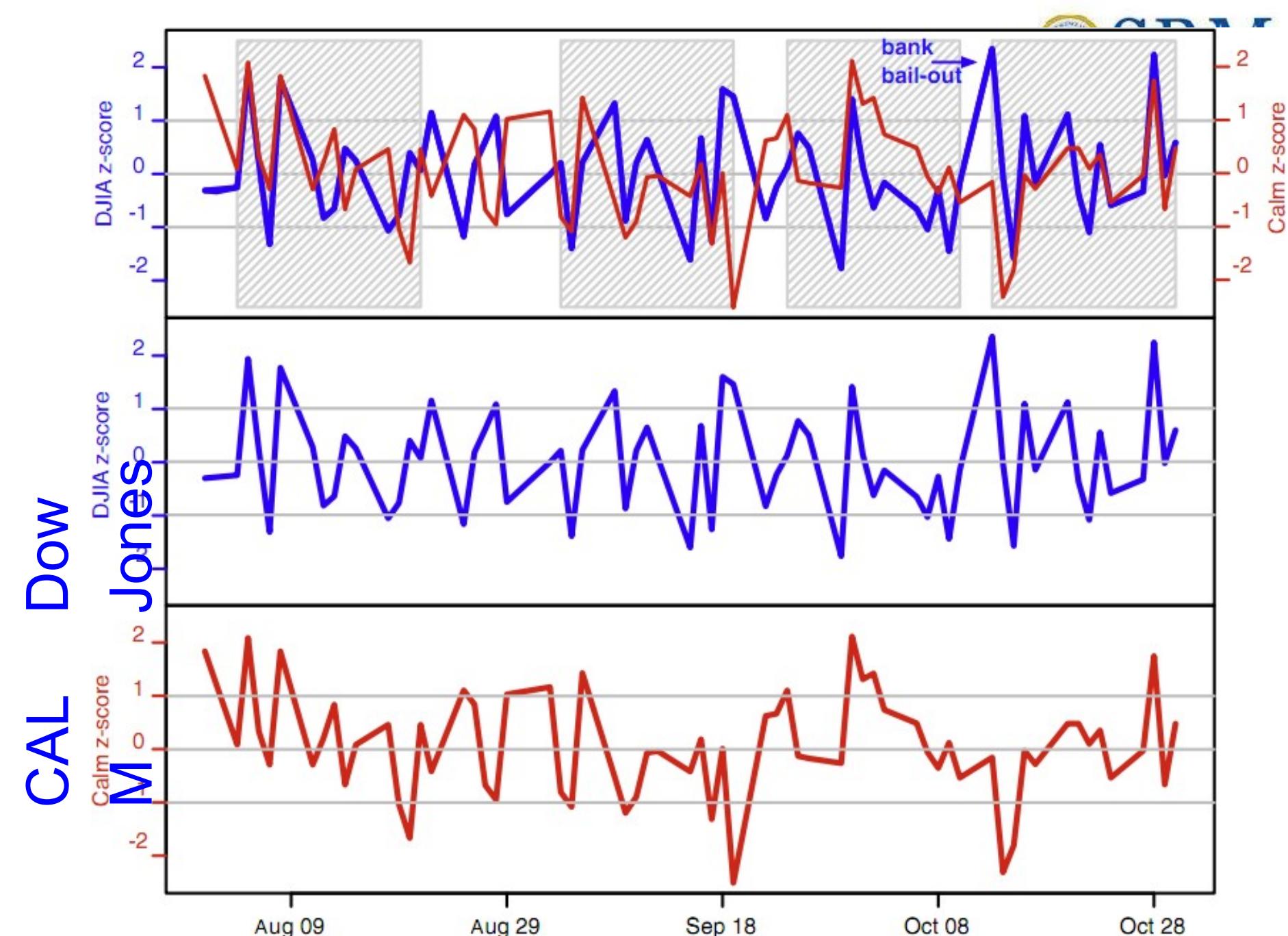
11  
9

22-04-2021



Bollen et al.  
(2011)

- CALM  
predicts  
DJIA 3 days  
later
- At least one  
current  
hedge fund  
uses this  
algorithm





# Applications

 Clip slide

## ✓ **Businesses and Organizations :**

- Brand analysis
- New product perception
- Product and Service benchmarking
- Business spends a huge amount of money to find consumer sentiments and opinions.

## ✓ **Individuals : Interested in other's opinions when...**

- Purchasing a product or using a service
- Finding opinions on political topics ,movies,etc.



# Applications

 Clip slide

## ✓ **Social Media :**

Finding general opinion about recent hot topics in town

## ✓ **Ads Placements :**

Placing ads in the user-generated content

- Place an ad when one praises a product.
- Place an ad from a competitor if one criticizes a product.



# Approach

 Clip slide

## ✓ **NLP**

- Use semantics to understand the language.
- Uses SentiWordNet

## ✓ **Machine Learning**

- Don't have to understand the meaning
- Uses classifiers such as Naïve Bayes, SVM, etc.

# DEEP LEARNING

# History of Deep Learning Ideas and Milestones\*

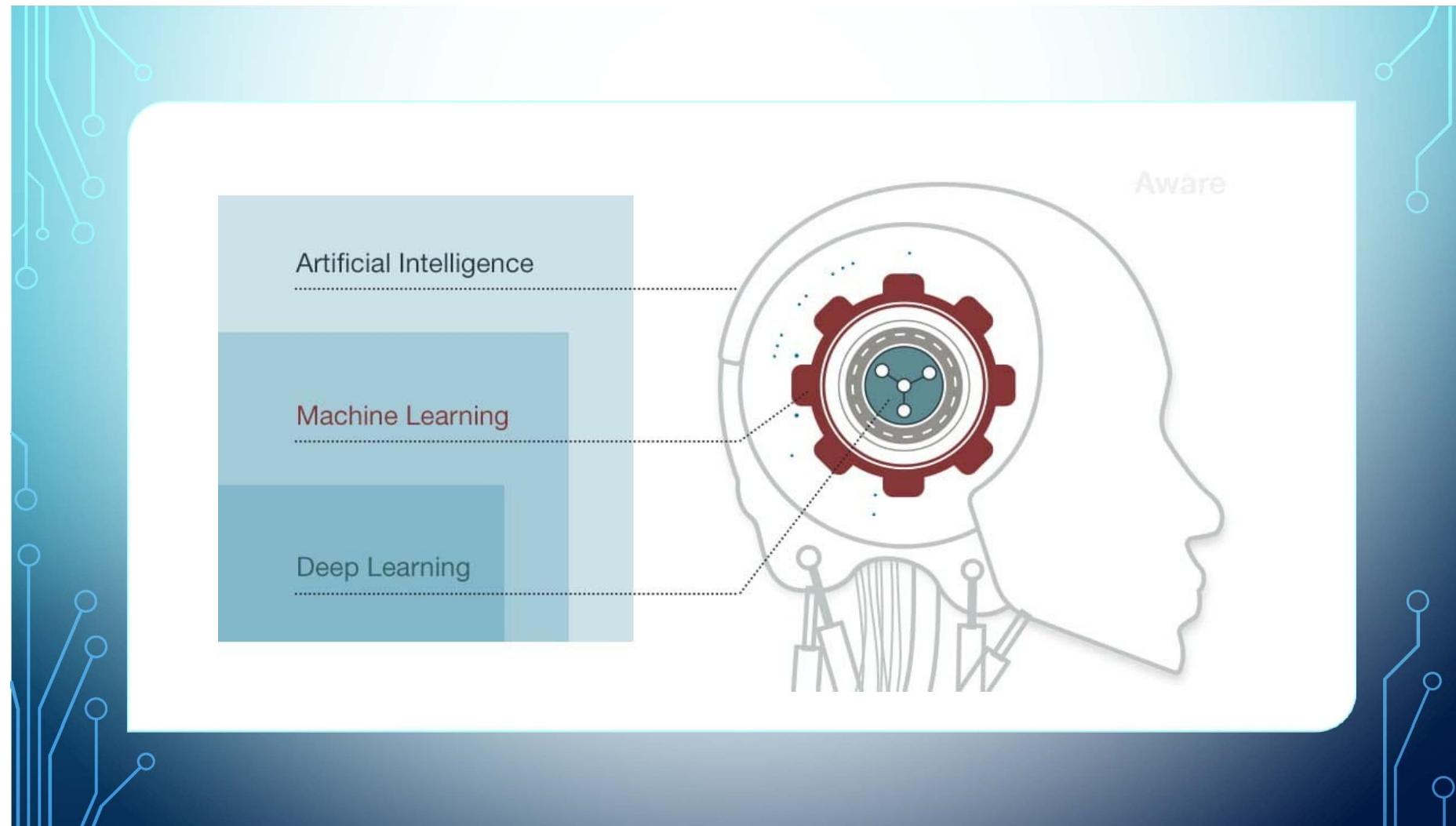


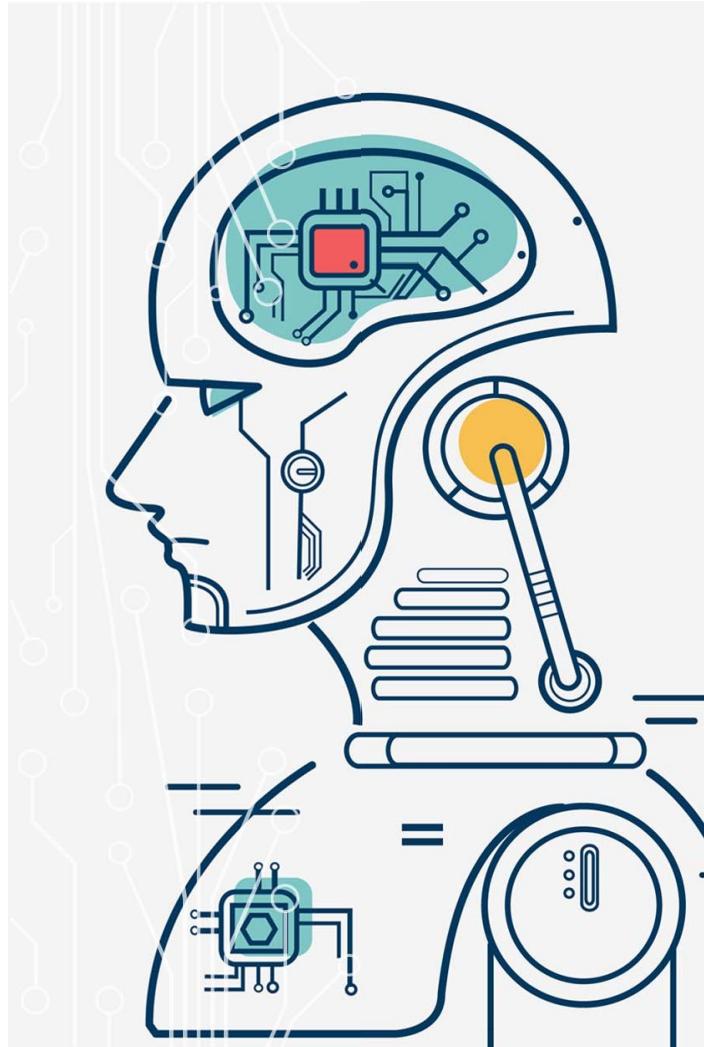
## Perspective:

- **Universe created**  
13.8 billion years ago
- **Earth created**  
4.54 billion years ago
- **Modern humans**  
300,000 years ago
- **Civilization**  
12,000 years ago
- **Written record**  
5,000 years ago

- 1943: Neural networks
- 1957-62: Perceptron
- 1970-86: Backpropagation, RBM, RNN
- 1979-98: CNN, MNIST, LSTM, Bidirectional RNN
- 2006: “Deep Learning”, DBN
- 2009: ImageNet + AlexNet
- 2014: GANs
- 2016-17: AlphaGo, AlphaZero
- 2017-2019: Transformers

\* Dates are for perspective and not as definitive historical record of invention or credit

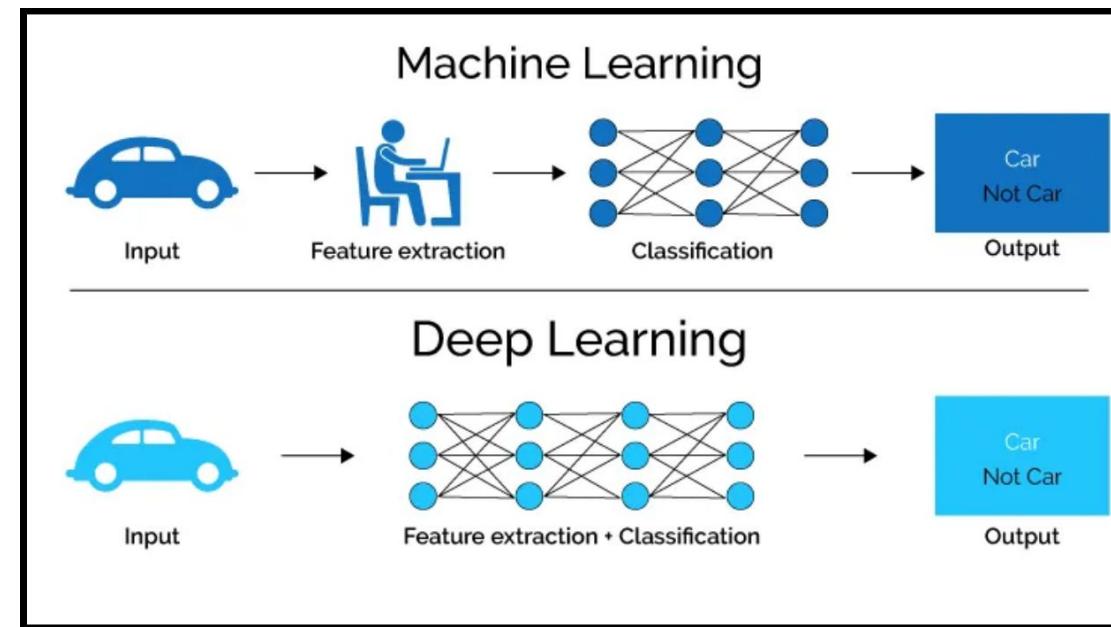




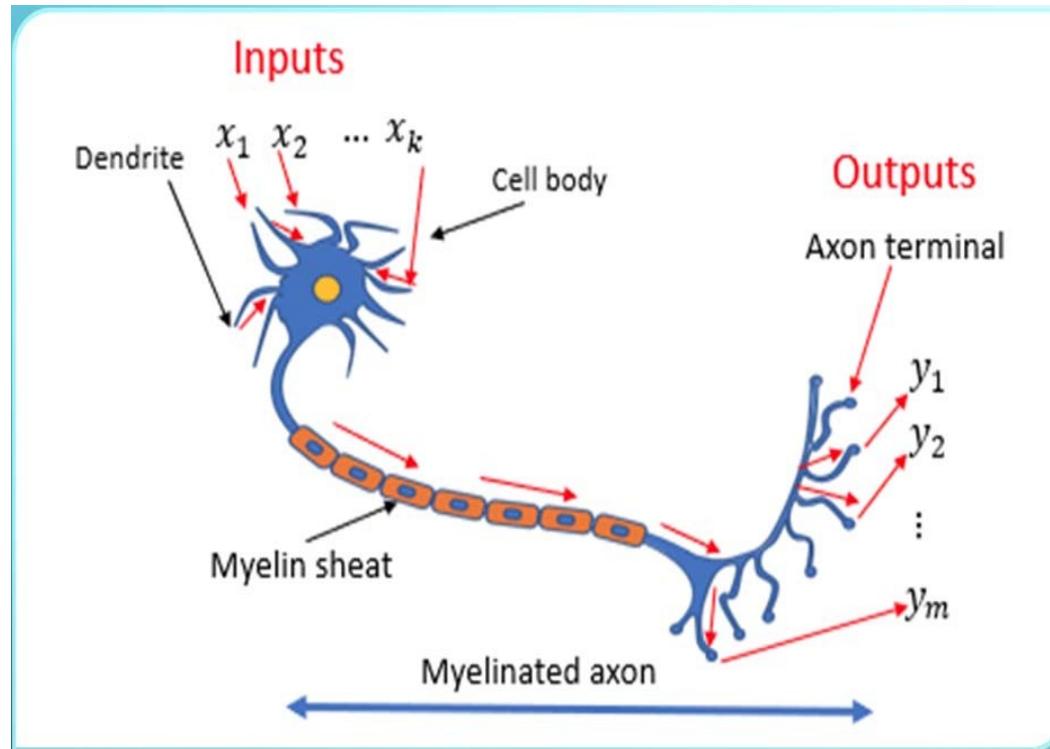
## MACHINE LEARNING

- Train machine by ourselves
- Extract feature and feed to the machine  
then apply algorithm to train it

## ML VS DL

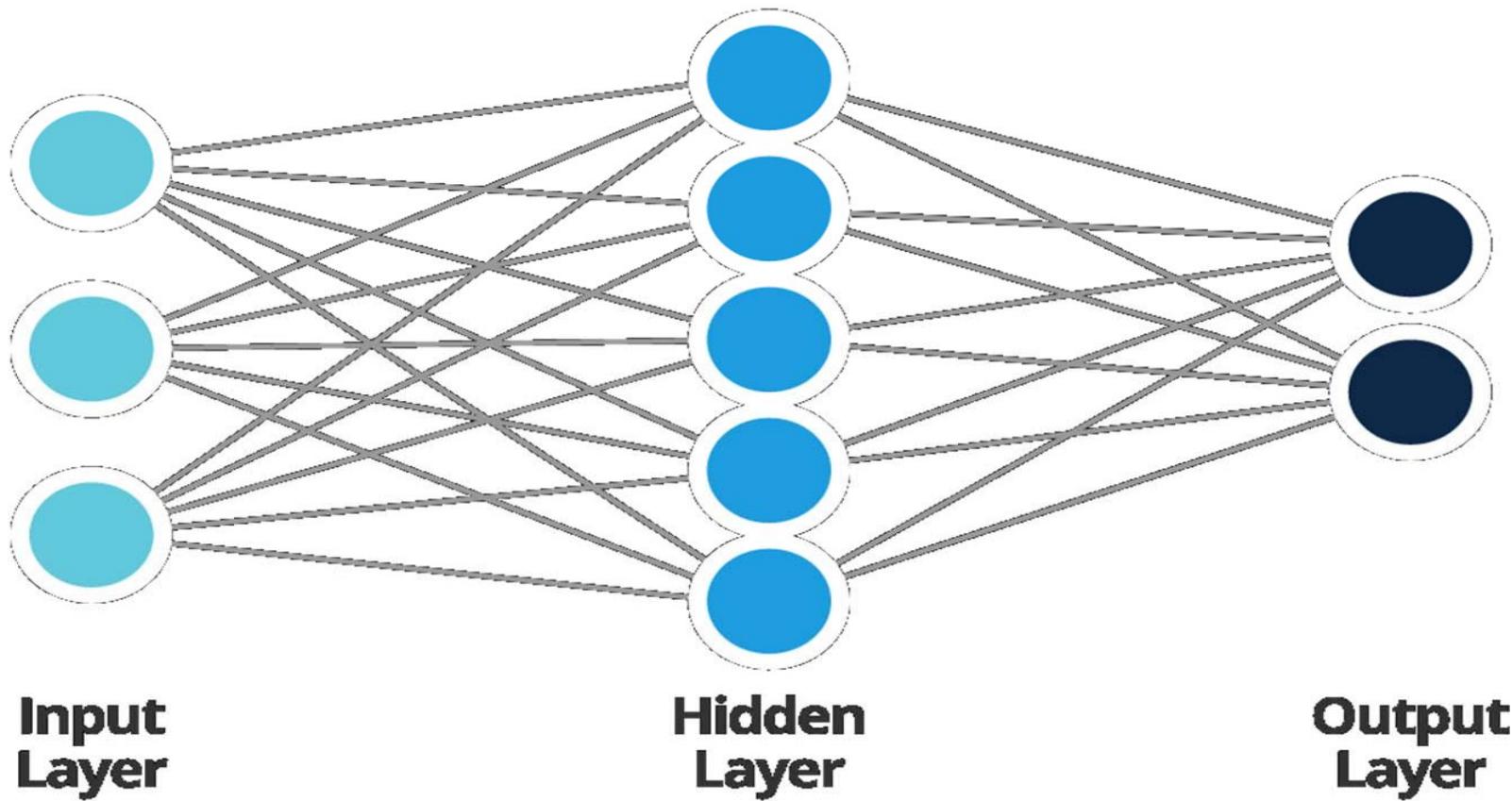


# DEEP LEARNING

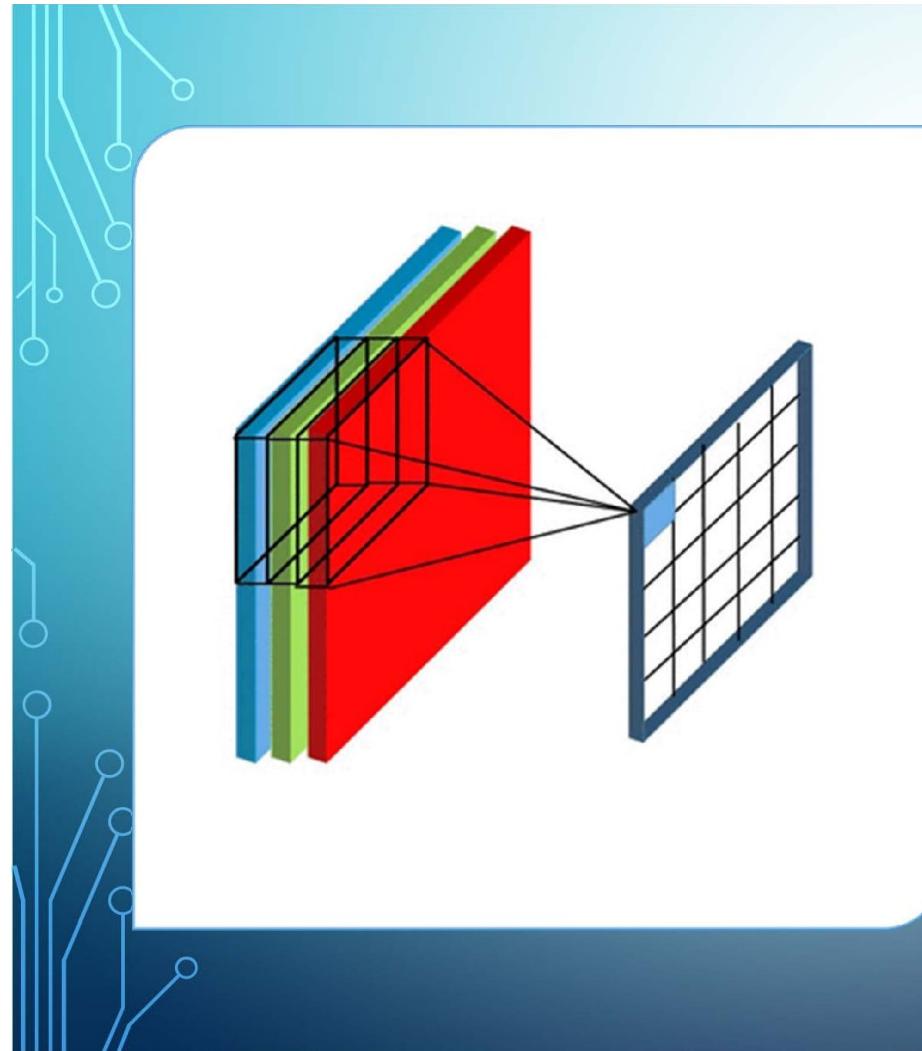


- It's a type of ML inspired by human brain.
- In DL, the structure is called **artificial neural network**.
- In DL, machine learns itself using **artificial neural network** that mimics biological neural network.

# Artificial Neural Network Architecture

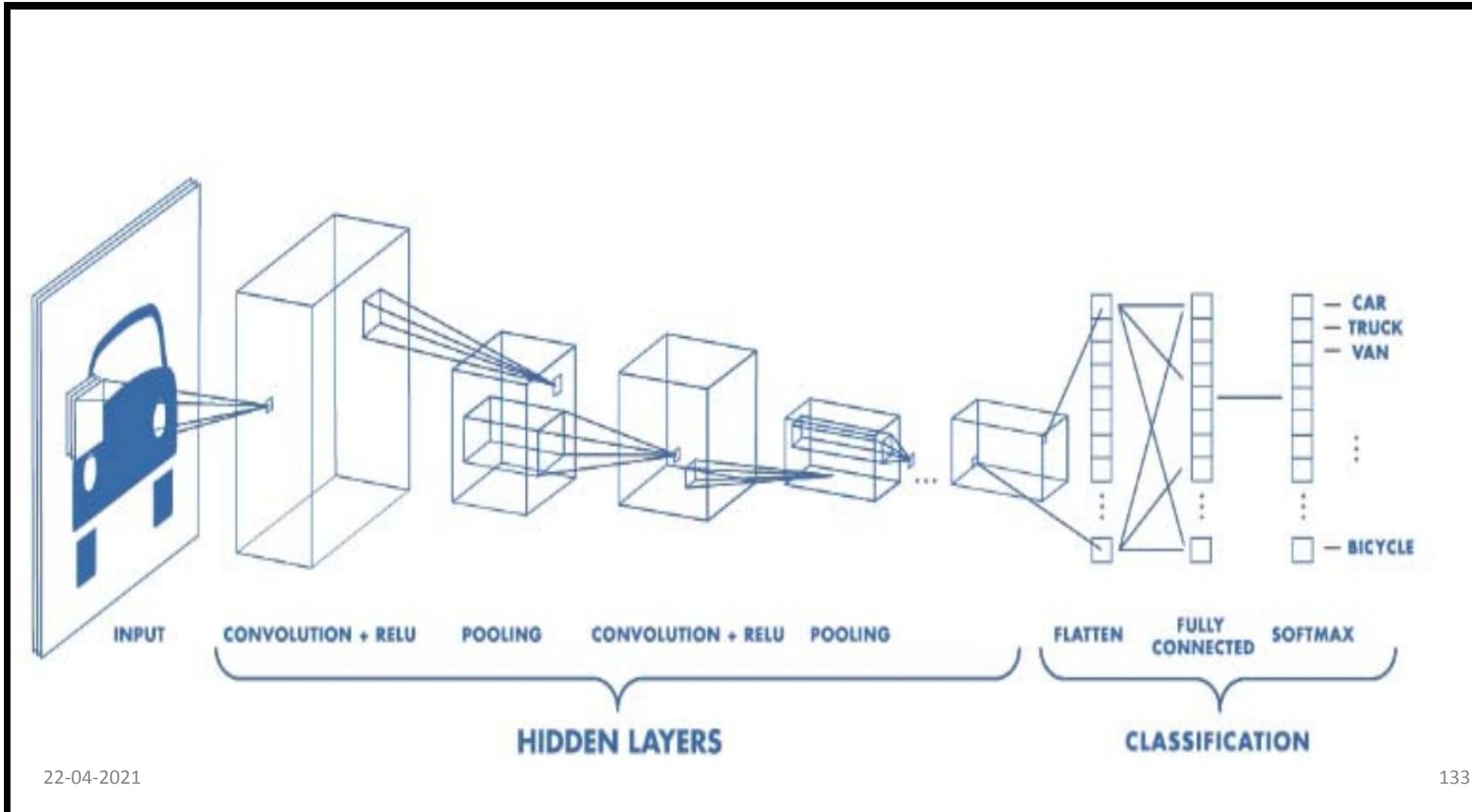


# CONVOLUTIONAL NEURAL NETWORK



## CONVOLUTIONAL NEURAL NETWORK (CNN)

- Image recognition
- Image classification
- Object detection



# CONVOLUTION LAYER

- An image matrix (volume) of dimension ( $h \times w \times d$ )
- A filter ( $f_h \times f_w \times f_d$ )
- Out put a volume dimension

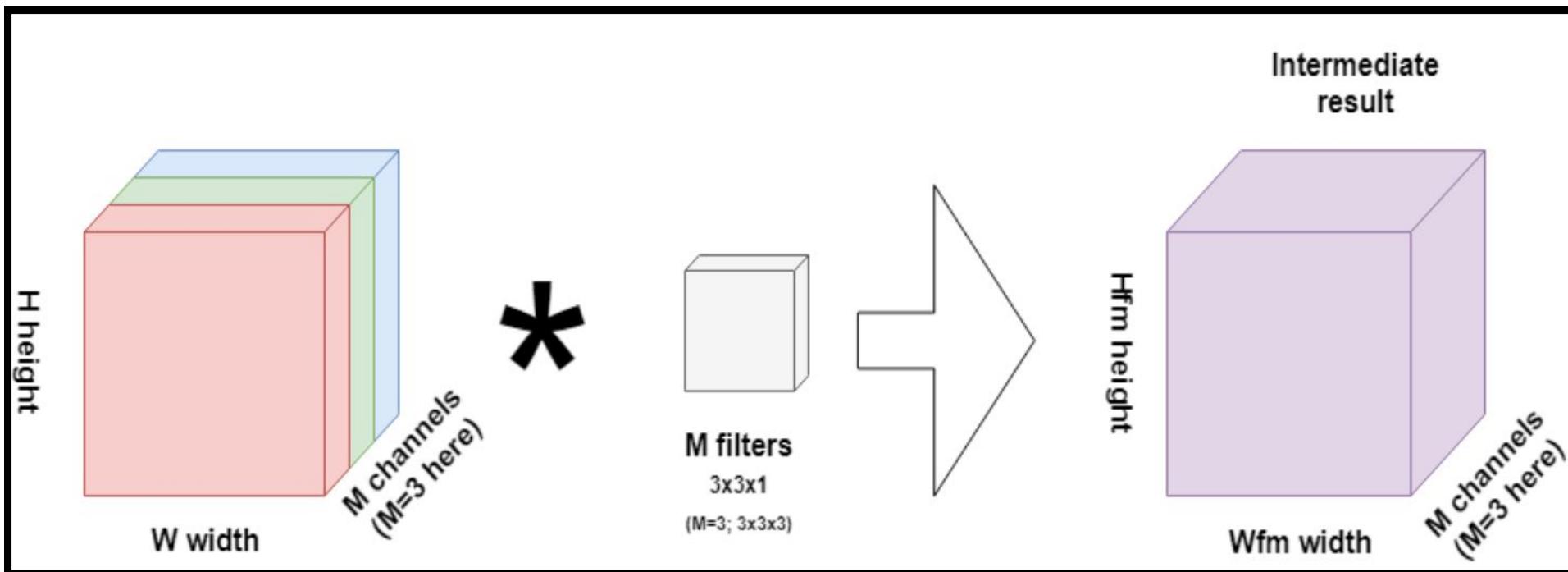


Image Matrix				
7	2	3	3	8
4	5	3	8	4
3	3	2	8	4
2	8	7	2	7
5	4	4	5	4

Filter Matrix

1	0	-1
1	0	-1
1	0	-1

\*

Convolved Feature

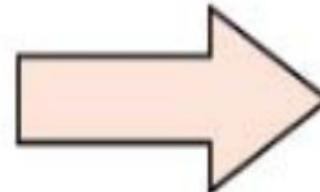
6		

=

$$\begin{aligned}
 & 7x1 + 4x1 + 3x1 + \\
 & 2x0 + 5x0 + 3x0 + \\
 & 3x-1 + 3x-1 + 2x-1 \\
 & = 6
 \end{aligned}$$

1	2	3	4	5	6	7
11	12	13	14	15	16	17
21	22	23	24	25	26	27
31	32	33	34	35	36	37
41	42	43	44	45	46	47
51	52	53	54	55	56	57
61	62	63	64	65	66	67
71	72	73	74	75	76	77

Convolve with 3x3  
filters filled with ones



108	126	
288	306	

# PADDING

0	0	0	0	0	0	0	0
0	60	113	56	139	85	0	0
0	73	121	54	84	128	0	0
0	131	99	70	129	127	0	0
0	80	57	115	69	134	0	0
0	104	126	123	95	130	0	0
0	0	0	0	0	0	0	0

Kernel
0 -1 0
-1 5 -1
0 -1 0



- Pad the picture with zero-padding so that it fits.
- Drop part where image did not fit. This called **valid padding** which keep only valid part of image.

## SEPARABLE CONVOLUTION LAYER

- The spatial separable convolution is so named because it deals primarily with the **spatial dimensions** of an image and kernel: the width and the height. (The other dimension, the “depth” dimension, is the number of channels of each image).

I also use this layer in my project too

**Simple Convolution**

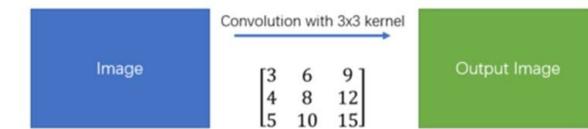


Image  $\xrightarrow{\text{Convolution with } 3 \times 3 \text{ kernel}}$  Output Image

$$\begin{bmatrix} 3 & 6 & 9 \\ 4 & 8 & 12 \\ 5 & 10 & 15 \end{bmatrix}$$

**Spatial Separable Convolution**



Image  $\xrightarrow{\text{Convolution with } 3 \times 1 \text{ kernel}}$  Intermediate Image  $\xrightarrow{\text{Convolution with } 1 \times 3 \text{ kernel}}$  Output Image