Sri Chandrasekharendra Saraswathi Viswa MahaVidyalaya

Enathur, Kanchipuram – 631 561.

Department of Computer Science and Engineering

Machine Learning

Prepared

Dr. C Sunitha Ram

Dr N Kumaran

OBJECTIVES:

1. To introduce students to the basic concepts and techniques of Machine Learning.
2. To have a thorough understanding of the Supervised and Unsupervised learning techniques
3. To study the various probabilities-based learning techniques
4. To understand graphical models of machine learning algorithms

PROGRAMME OUTCOME:

1. Apply basic principles and practices of computing grounded in mathematics and science to successfully complete software related projects to meet customer business Objective(s) and/or productively engage in research.
2. Apply their knowledge and skills to succeed in a computer science career and/or obtain an advanced degree.
3. Demonstrate an ability to use techniques, skills, and modern computing tools to implement and organize computing works under given constraints.
4. Demonstrate problem solving and design skills including the ability to formulate problems and their solutions, think creatively and communicate effectively.
5. Develop software as per the appropriate software life cycle model.
6. Organize and maintain the information of an organization.
7. Exhibit teamwork, communication, and interpersonal skills which enable them to work effectively with interdisciplinary teams.
8. Provide an excellent education experience through the incorporation of current pedagogical techniques, understanding of contemporary trends in research and technology, and hands-on laboratory experiences that enhance the educational experience.
9. Demonstrate an ability to engage in life-long learning.
10. Function ethically and responsibly, and to remain informed and involved as full participants in our profession and our society.

COURSE OUTCOMES:

Upon completion of the course, the students will be able to: Distinguish between, supervised, unsupervised and semi-supervised learning

1. Apply the apt machine learning strategy for any given problem
2. Suggest supervised, unsupervised or semi-supervised learning algorithms for any given problem
3. Design systems that uses the appropriate Trees in Probabilities Models of machine learning
4. Modify existing machine learning algorithms to improve classification efficiency
5. Design systems that uses the appropriate graph models of machine learning

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10
CO1	L	M		M						
CO2		M	M					H		
CO3	L	M	M						H	H
CO4		L	M						H	
CO5	L		M	M					H	H

UNIT – I INTRODUCTION

Learning – Types of Machine Learning – Supervised Learning – The Brain and the Neuron – Design a Learning System – Perspectives and Issues in Machine Learning – Concept Learning Task – Concept Learning as Search – Finding a Maximally Specific Hypothesis – Version Spaces and the Candidate Elimination Algorithm – Linear Discriminates – Perceptron – Linear Separability – Linear Regression.

UNIT – I LINEAR MODELS

Multi-layer Perceptron – Going Forwards – Going Backwards: Back Propagation Error – Multi-layer Perceptron in Practice – Examples of using the MLP – Overview – Deriving Back-Propagation – Radial Basis Functions and Splines – Concepts – RBF Network – Curse of Dimensionality – Interpolations and Basis Functions – Support Vector Machines

UNIT – III TREE AND PROBABILISTIC MODELS

Learning with Trees – Decision Trees – Constructing Decision Trees – Classification and Regression Trees – Ensemble Learning – Boosting – Bagging – Different ways to Combine Classifiers – Probability and Learning – Data into Probabilities – Basic Statistics – Gaussian Mixture Models – Nearest Neighbor Methods – Unsupervised Learning – K means Algorithms – Vector Quantization – Self Organizing Feature Map.

UNIT – IV DIMENSIONALITY REDUCTION AND EVOLUTIONARY MODELS

Dimensionality Reduction – Linear Discriminant Analysis – Principal Component Analysis – Factor Analysis – Independent Component Analysis – Locally Linear Embedding – Isomap – Least Squares Optimization – Evolutionary Learning – Genetic algorithms – Genetic Offspring: - Genetic Operators – Using Genetic Algorithms – Reinforcement Learning – Overview – Getting Lost Example – Markov Decision Process

UNIT - V GRAPHICAL MODELS

Markov Chain Monte Carlo Methods – Sampling – Proposal Distribution – Markov Chain Monte Carlo – Graphical Models – Bayesian Networks – Markov Random Fields – Hidden Markov Models – Tracking Methods.

TEXT BOOKS:

1. Stephen Marsland, —Machine Learning – An Algorithmic Perspective||, Second Edition, Chapman and Hall/CRC Machine Learning and Pattern Recognition Series, 2014.
2. Tom M Mitchell, —Machine Learning||, First Edition, McGraw Hill Education, 2013.

REFERENCES:

1. Peter Flach, —Machine Learning: The Art and Science of Algorithms that Make Sense of Data||, First Edition, Cambridge University Press, 2012.
2. Jason Bell, —Machine learning – Hands on for Developers and Technical Professionals||, First Edition, Wiley, 2014
3. Ethem Alpaydin, —Introduction to Machine Learning 3e (Adaptive Computation and Machine Learning Series)||, Third Edition, MIT Press, 2014

Introduction:-Data science is an inter-disciplinary is an inter-disciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge is an inter-disciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from many structural and unstructured data is an inter-disciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from many structural and unstructured data. Data science is related to data mining(Data mining is a process of discovering patterns in large data sets is a process of discovering patterns in large data sets), machine learning is a process of discovering patterns in large data sets), machine learning and big data.

Unstructured data (or unstructured information) is information thateither does not have a pre-defined data model) is information that either does not have a pre-defined data model or is not organized in a pre-defined manner. Unstructured information is typically text) is information that either does not have a pre-defined data model or is not organized in a 3

What is Data?

- ➢ Data is often viewed as the lowest level of abstraction from which information and knowledge are derived.
- ➢ Data can be numbers, words,measurements, observations or even justdescriptions of things. Also, data is a representation of a fact, figure and idea.
- ➢ Data on its own carries no meaning. If data to be an information, it must beinterpreted and take on a meaning.

An example of raw data table. It is just a collection of random info and data.

Machine learning (ML) is the study of computer algorithms that improveautomatically through experience. It is seen as a subset of artificialintelligence)is the study of computer algorithms that improve automatically through experience .It is seen as a subset of artificialintelligence. Machine learning algorithms build a mathematical model) isthe study of computer algorithms that improve automatically throughexperience. It is seen as a subset of artificial intelligence. Machine learningalgorithms build a mathematical model based on sample data, known as training data)is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data known as "training data" in order to make [illegible] or decisions without being explicitly programmed to do so.Machine learning algorithms are used in a wide variety of applications,such as email filtering) is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial6

Types of Data / Variables

Categorical Data is the data that is non numeric.

e.g.. Favorite color, Place of Birth, Types of Car

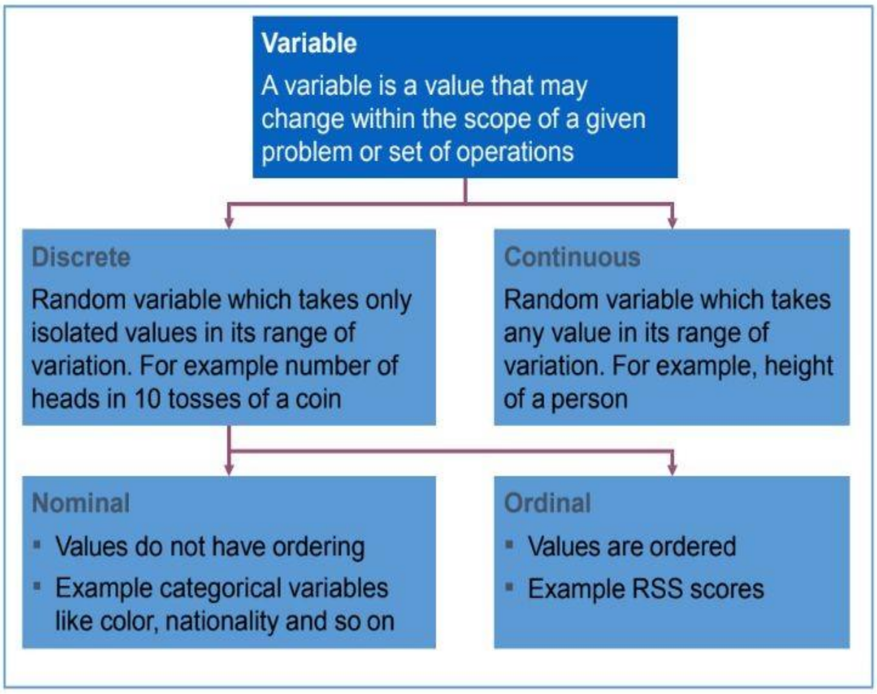
Quantitative Data is numerical. There are 2 types of quantitative data.

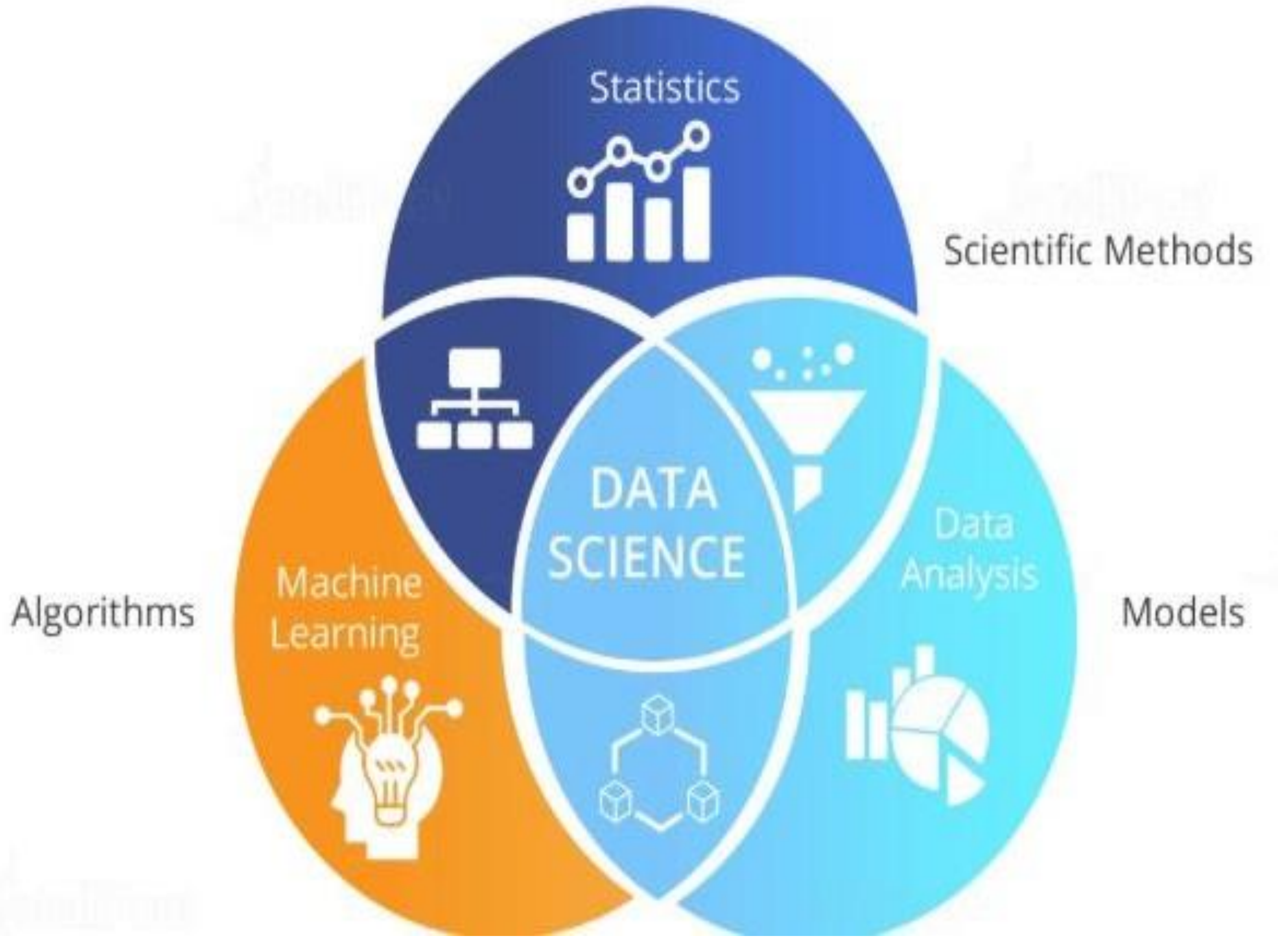
1. Discrete data can only take specific values;

e.g. shoe size, number of brothers, number of cars in a car park.

2. Continuous data can take any numerical value;

e.g. height, mass, length.


Variable
A variable is a value that may change within the scope of a given problem or set of operations
Discrete
Random variable which takes only isolated values in its range of variation. For example number of heads in 10 tosses of a coin
Continuous
Random variable which takes any value in its range of variation. For example, height of a person
Nominal
Values do not have ordering
Example categorical variables like color, nationality and so on
Ordinal
Values are ordered
Example RSS scores

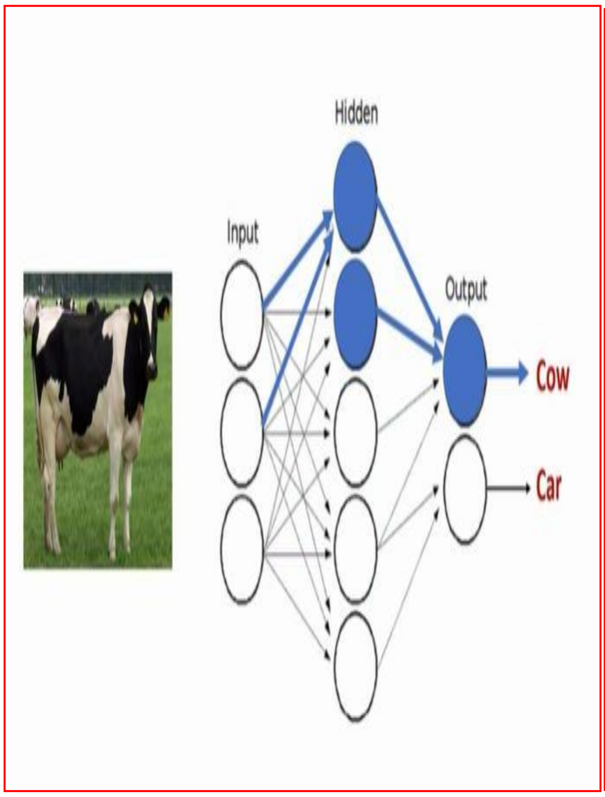


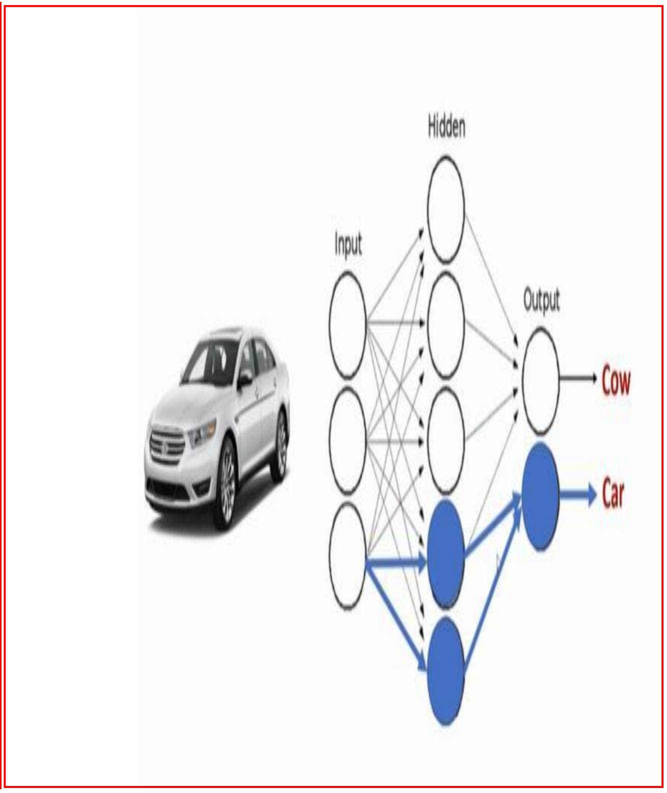
What is Machine Learning?

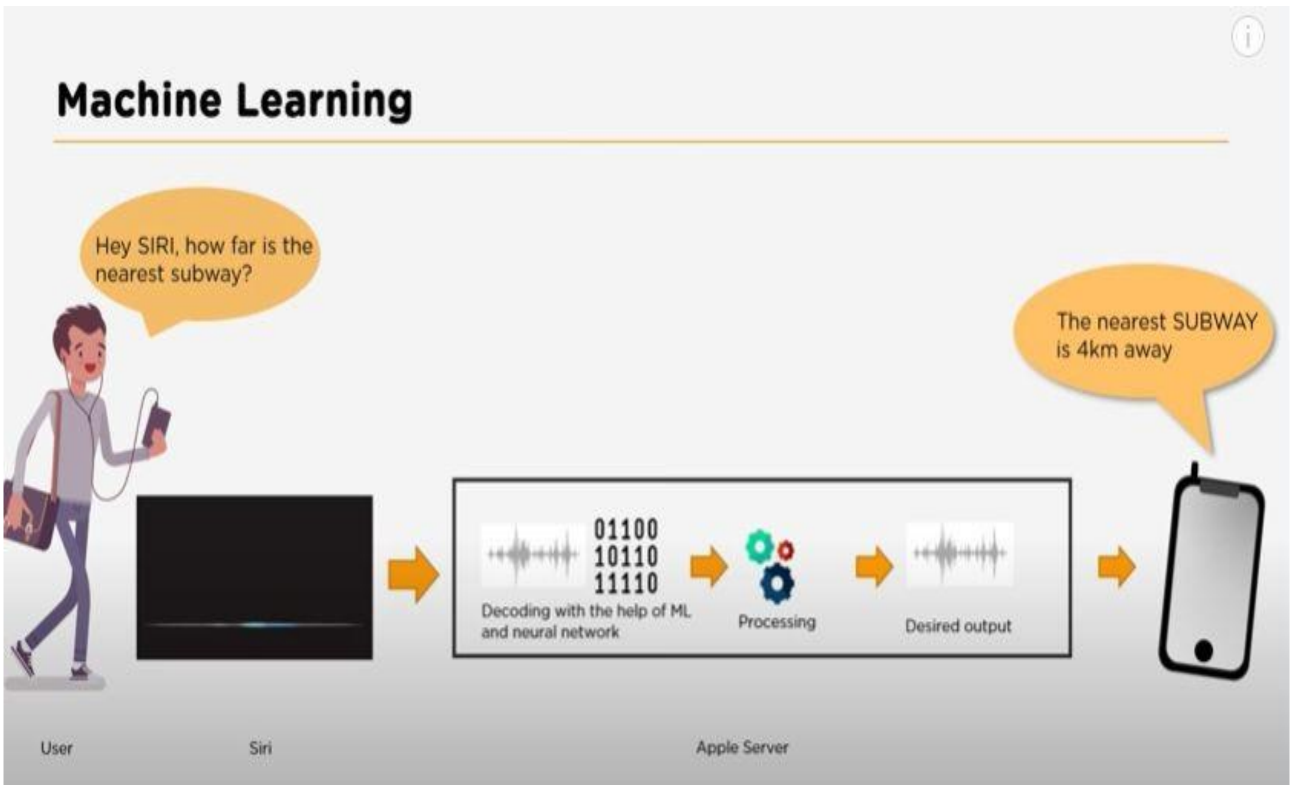
The subfield of computer science that "gives computers the ability to learn without being explicitly programmed".
(Arthur Samuel, 1959)

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."
(Tom Mitchell, 1997)

$\underbrace{\text{Using data}}_{\text{Training}}$ for $\underbrace{\text{answering questions}}_{\text{Predicting}}$

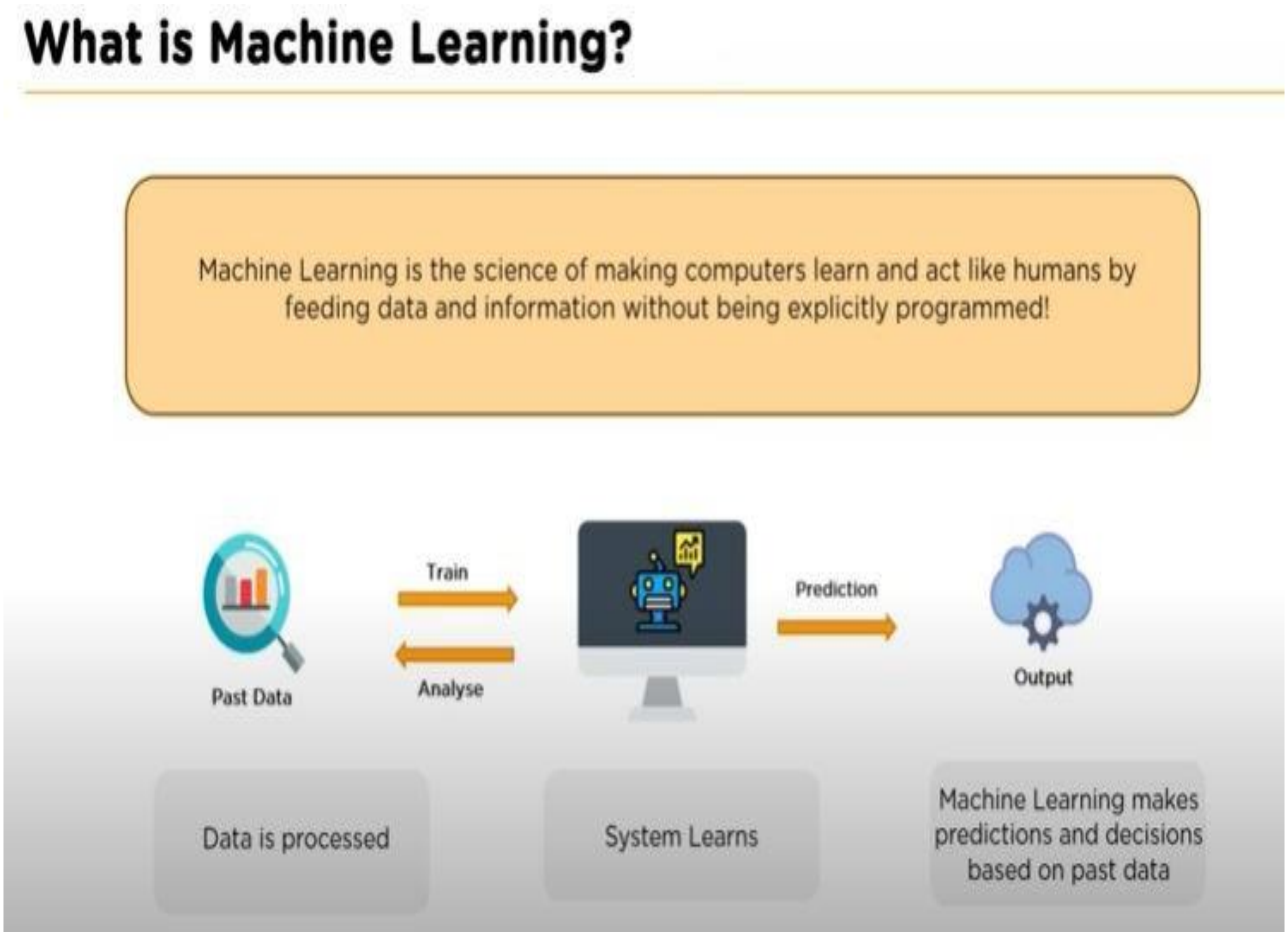






Machine Learning

Hey SIRI, how far is the nearest subway?
01100
10110
11110
Decoding with the help of ML and neural network
Processing
Desired output
The nearest SUBWAY is 4km away
User
Siri
Apple Server



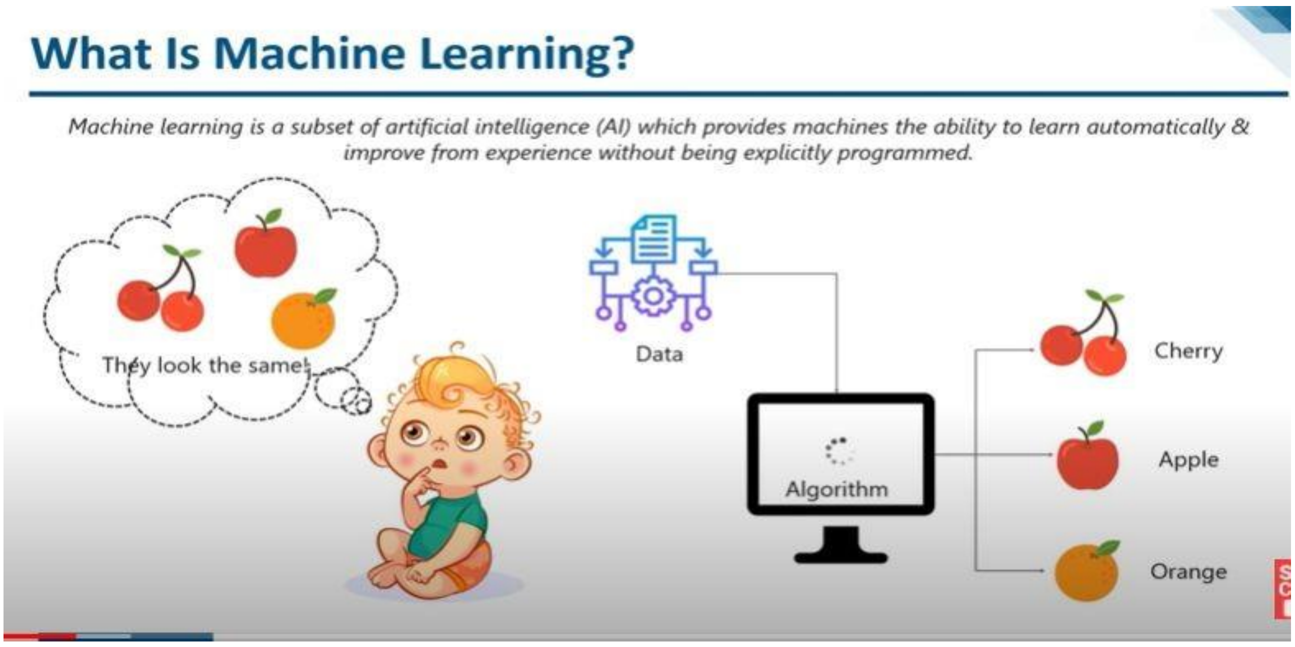
What is Machine Learning?

Machine Learning is the science of making computers learn and act like humans by feeding data and information without being explicitly programmed!

Data is processed

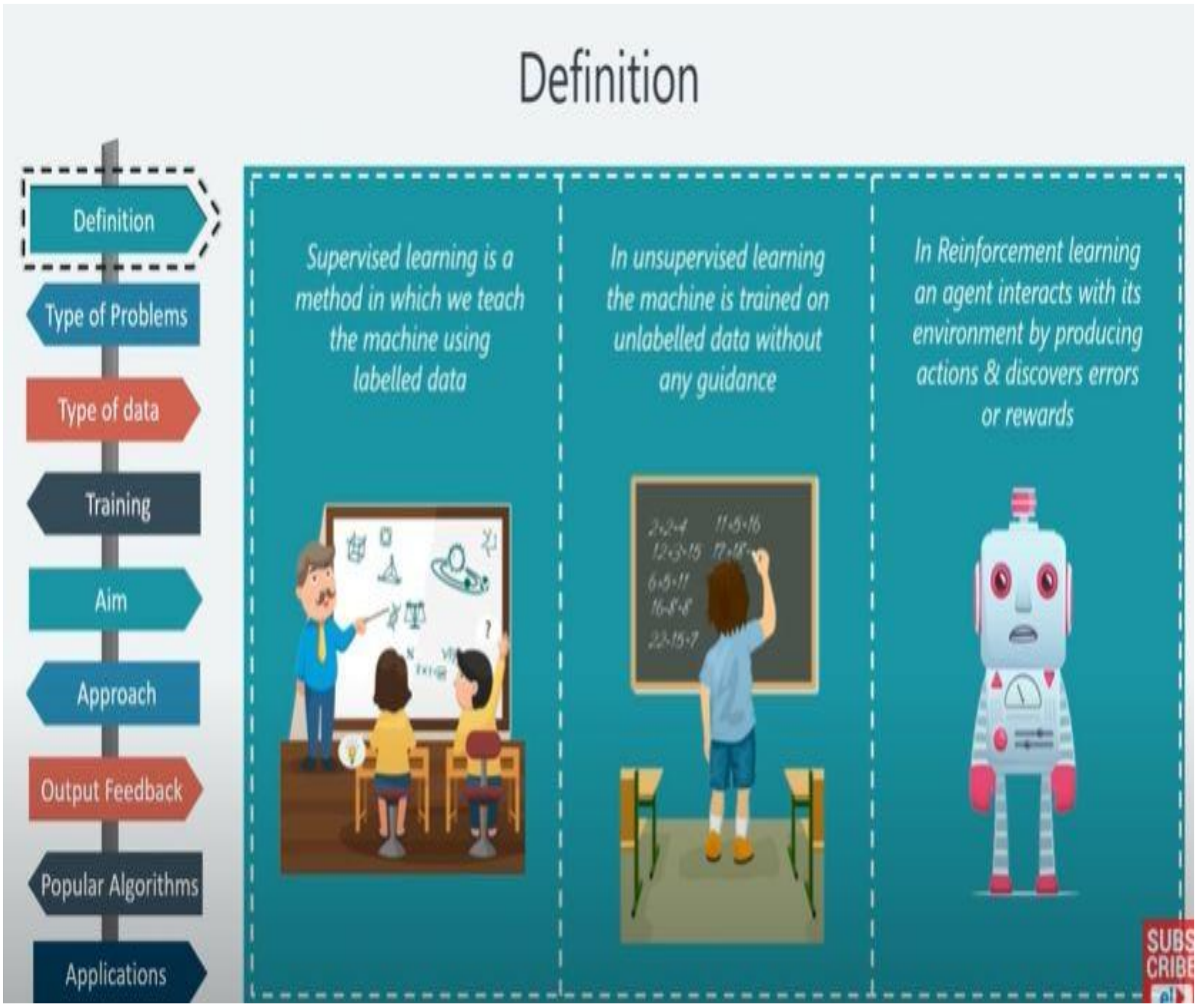
System Learns

Machine Learning makes predictions and decisions based on past data



What Is Machine Learning?

Machine learning is a subset of artificial intelligence (AI) which provides machines the ability to learn automatically & improve from experience without being explicitly programmed.

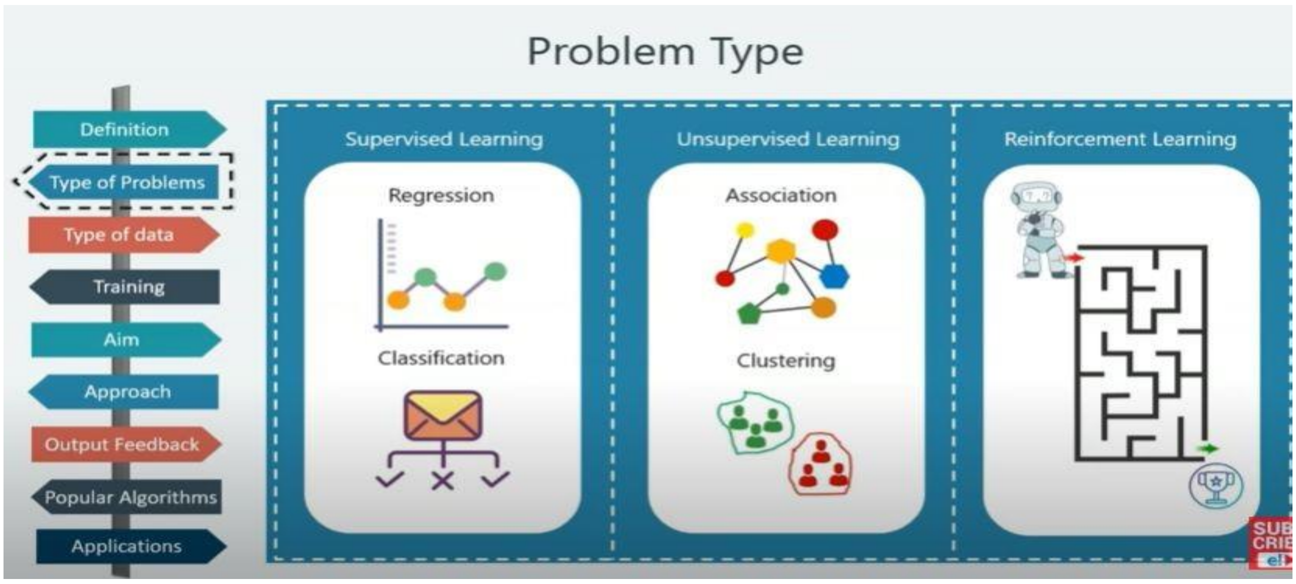


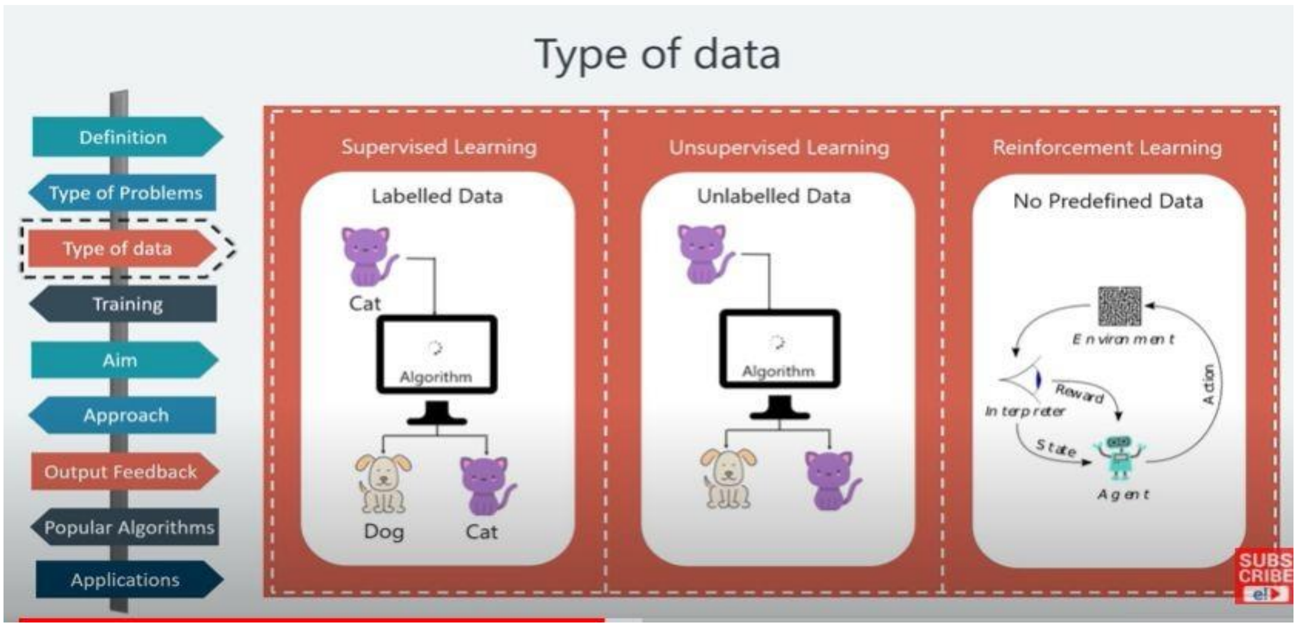
Definition

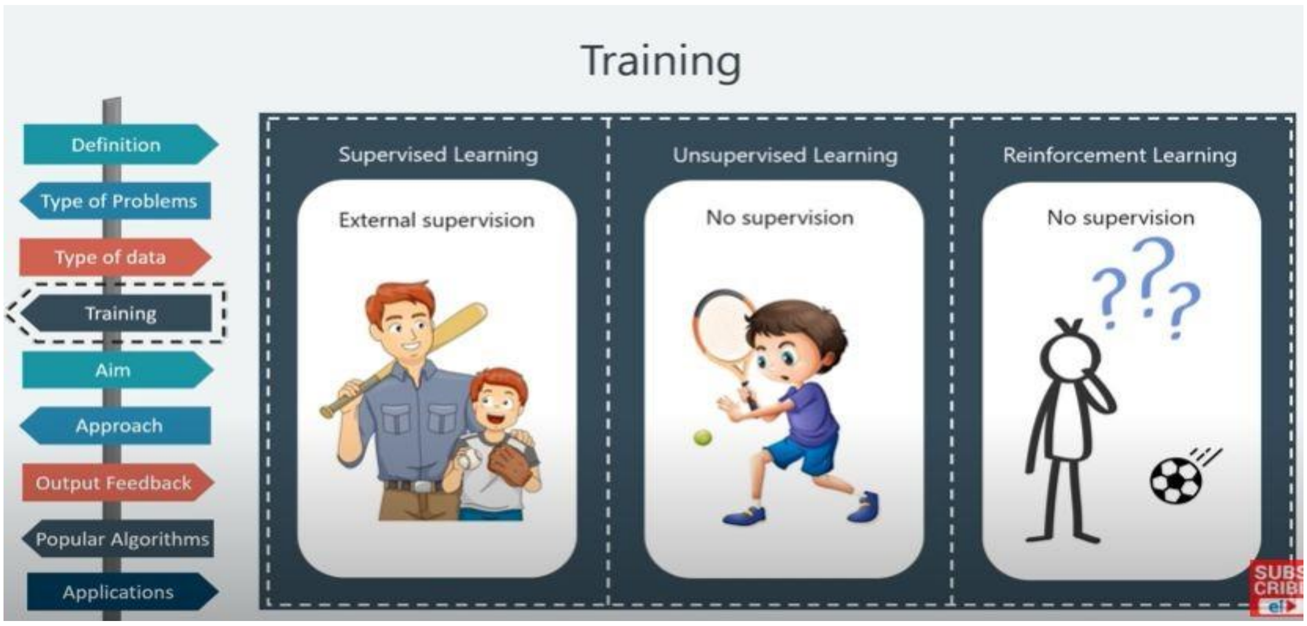
Supervised learning is a method in which we teach the machine using labelled data

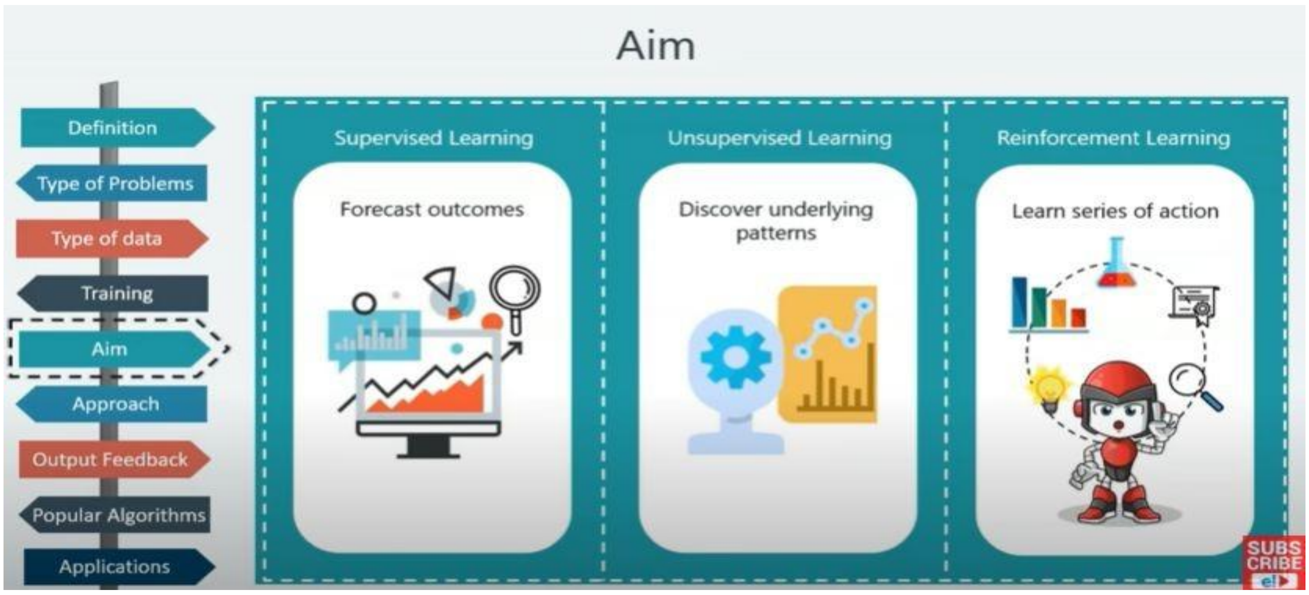
In unsupervised learning the machine is trained on unlabelled data without any guidance

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


Problem Type
Definition
Type of Problems
Type of data
Training
Aim
Approach
Output Feedback
Popular Algorithms
Applications
Supervised Learning
Regression
Classification
Unsupervised Learning
Association
Clustering
Reinforcement Learning


Type of data
Definition
Type of Problems
Type of data
Training
Aim
Approach
Output Feedback
Popular Algorithms
Applications
Supervised Learning
Labelled Data
Cat
Algorithm
Dog
Cat
Unsupervised Learning
Unlabelled Data
Algorithm
Reinforcement Learning
No Predefined Data
Environment
Reward
Interpreter
Action
State
Agent

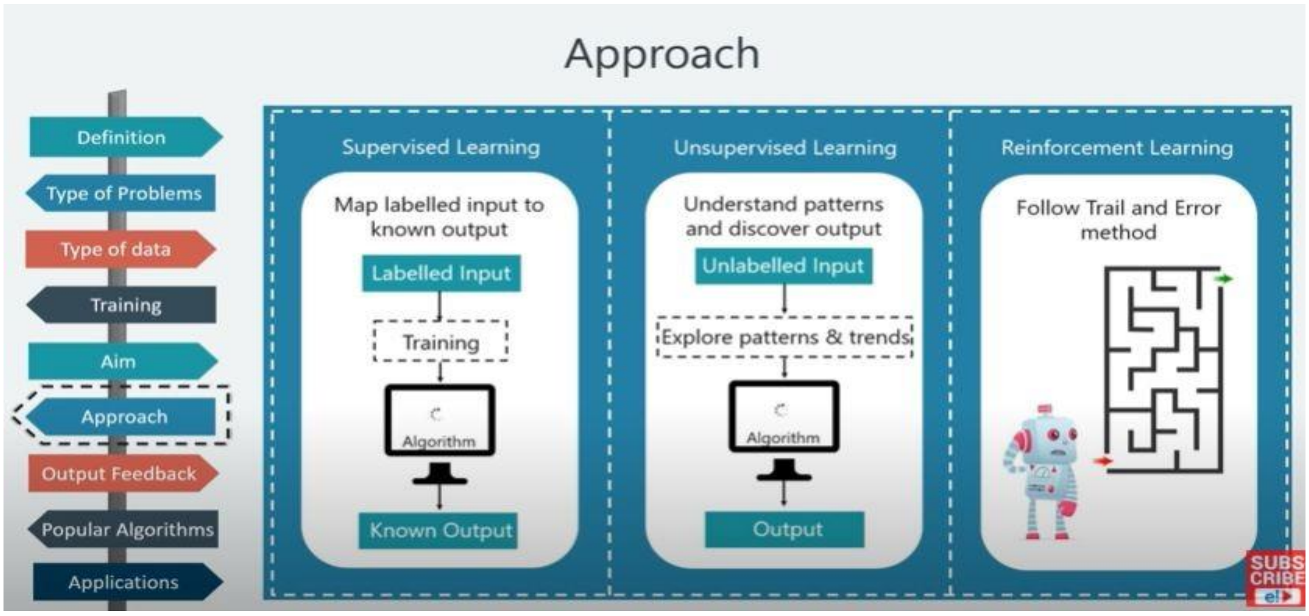

Training
Definition
Type of Problems
Type of data
Training
Aim
Approach
Output Feedback
Popular Algorithms
Applications
Supervised Learning
External supervision
Unsupervised Learning
No supervision
Reinforcement Learning
No supervision



Aim

Definition
Type of Problems
Type of data
Training
Aim
Approach
Output Feedback
Popular Algorithms
Applications

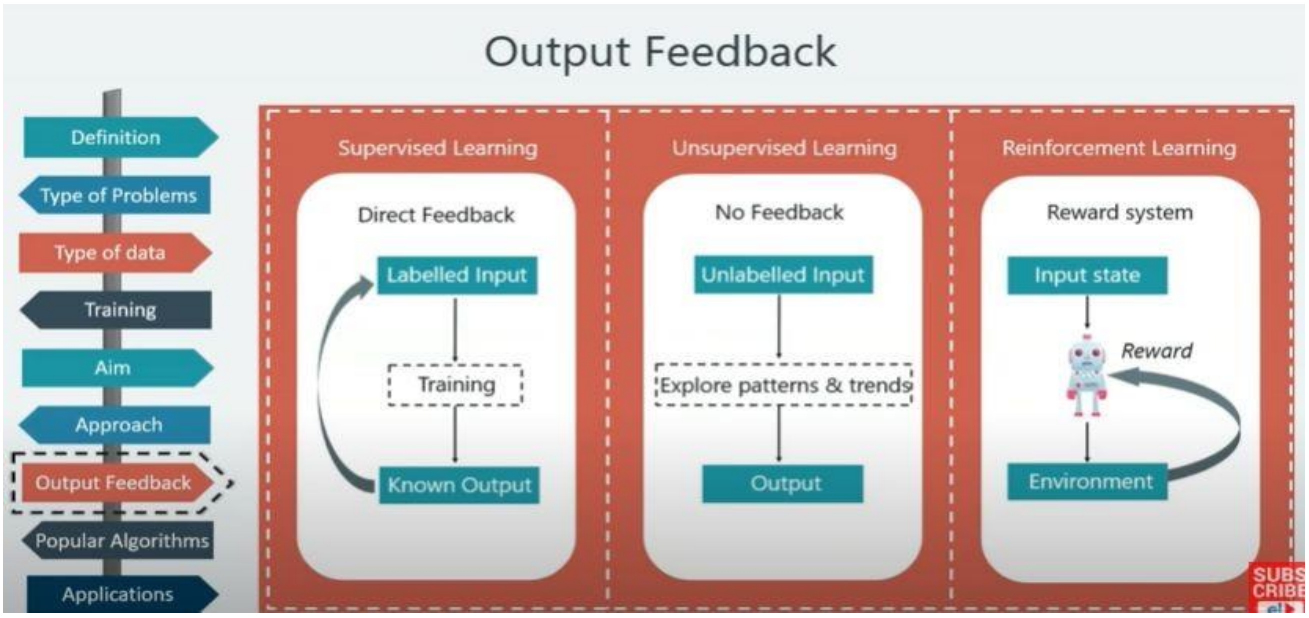
Supervised Learning	Unsupervised Learning	Reinforcement Learning
Forecast outcomes	Discover underlying patterns	Learn series of action



Approach

Definition
Type of Problems
Type of data
Training
Aim
Approach
Output Feedback
Popular Algorithms
Applications

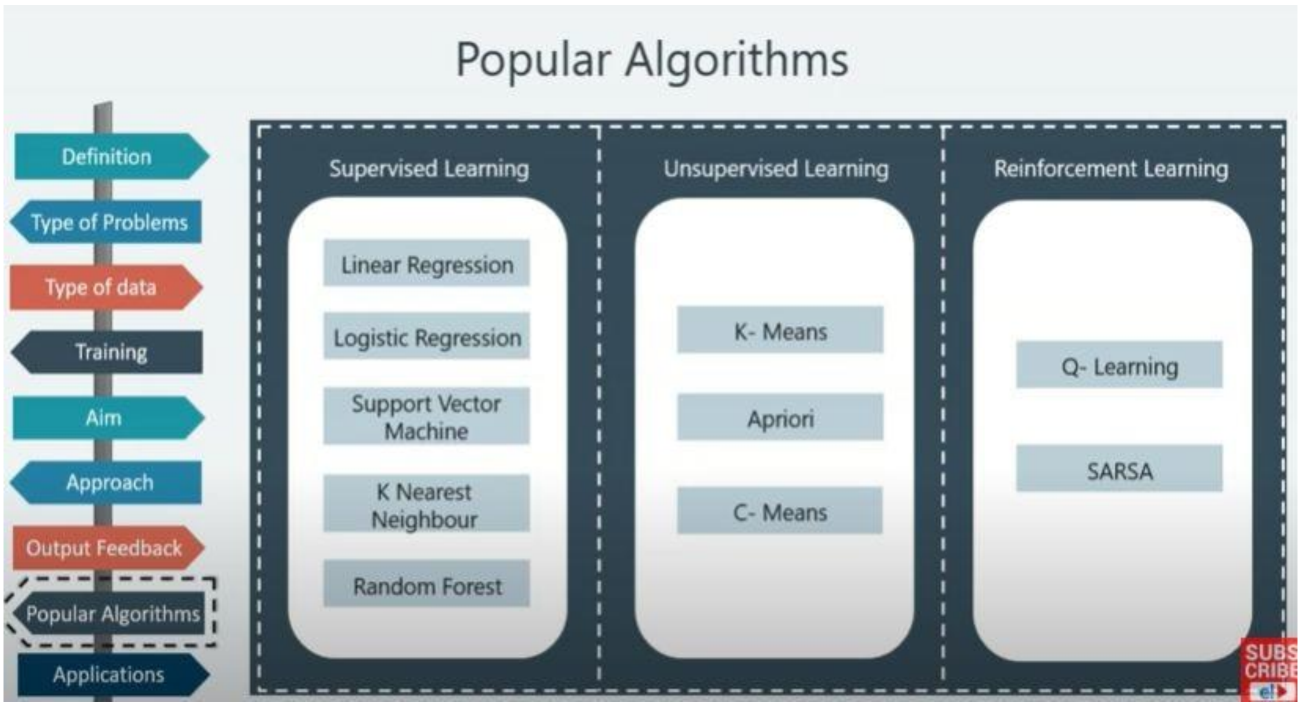
Supervised Learning	Unsupervised Learning	Reinforcement Learning
Map labelled input to known output	Understand patterns and discover output	Follow Trail and Error method



Output Feedback

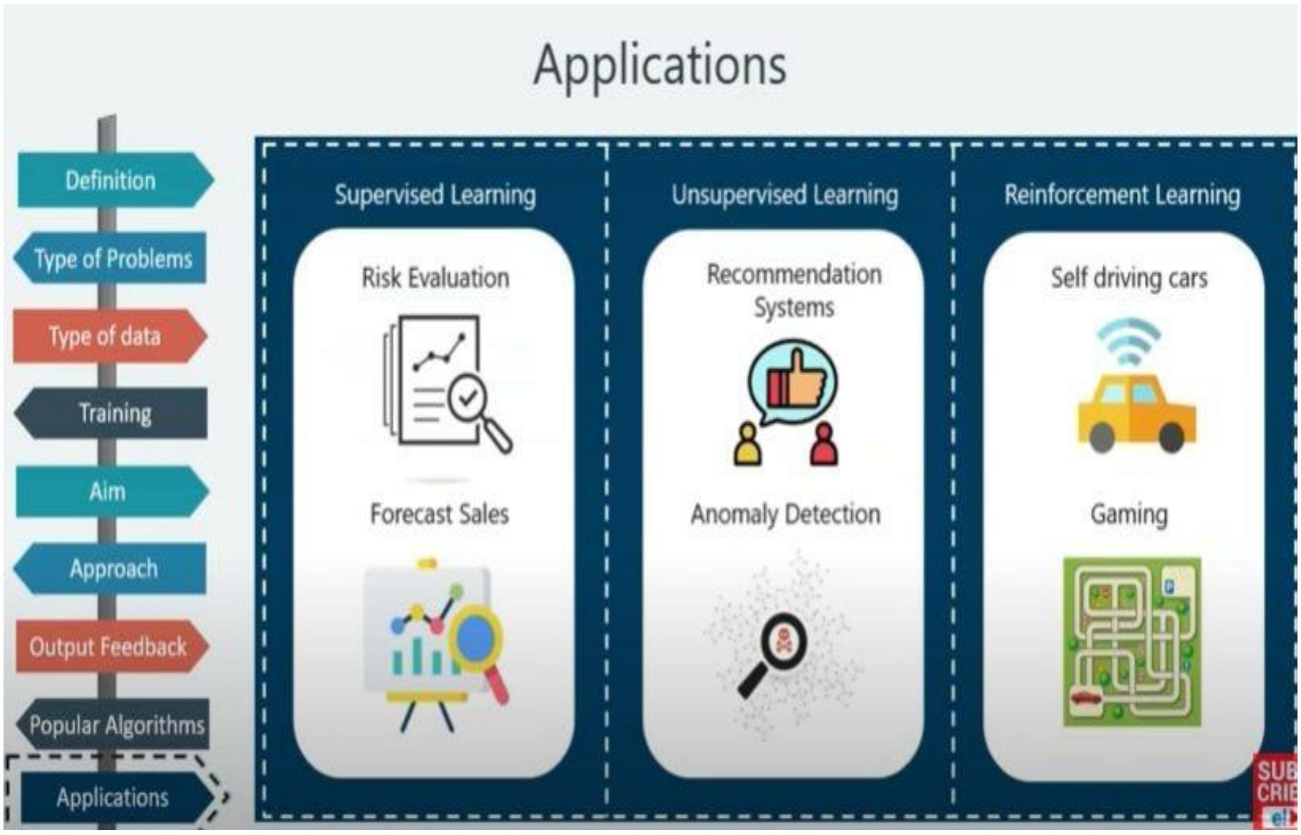
Definition
Type of Problems
Type of data
Training
Aim
Approach
Output Feedback
Popular Algorithms
Applications

Supervised Learning	Unsupervised Learning	Reinforcement Learning
Direct Feedback	No Feedback	Reward system



Popular Algorithms

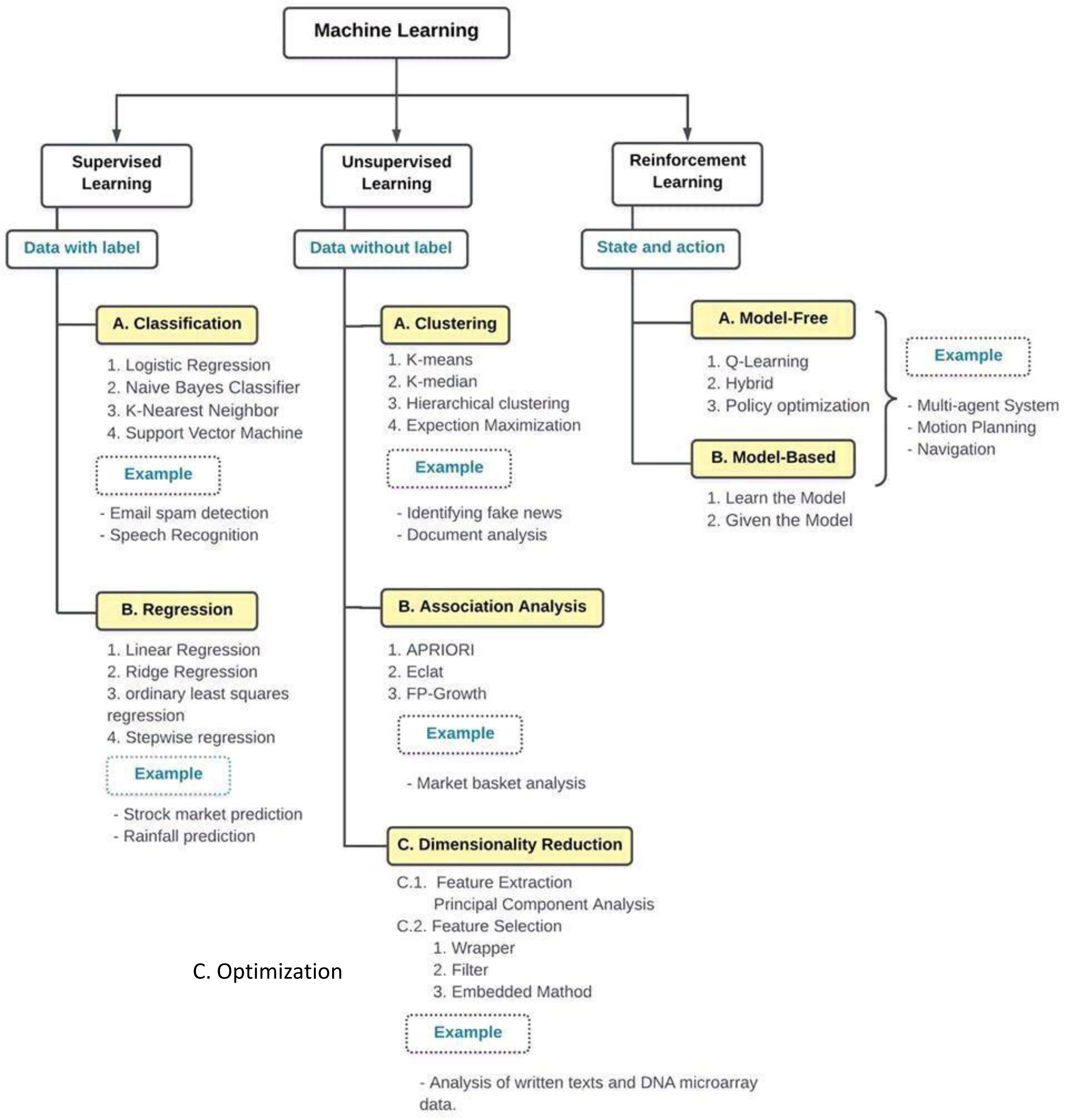
Supervised Learning	Unsupervised Learning	Reinforcement Learning
Linear Regression	K- Means	Q- Learning
Logistic Regression	Apriori	SARSA
Support Vector Machine	C- Means	
K Nearest Neighbour		
Random Forest		



Applications

Supervised Learning	Unsupervised Learning	Reinforcement Learning
Risk Evaluation	Recommendation Systems	Self driving cars
Forecast Sales	Anomaly Detection	Gaming

Types of Machine Learning

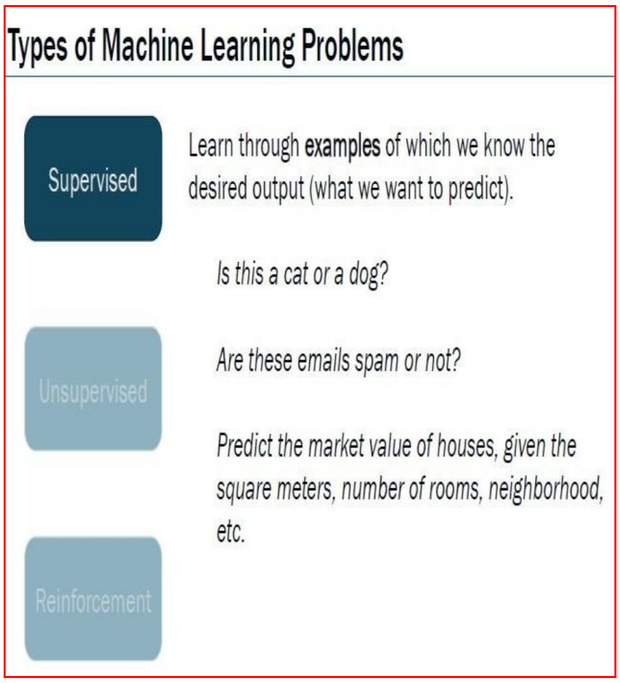


C. Optimization

D. Recommender System

E. Feature Analysis

F. Sentiment Analysis



Types of Machine Learning Problems

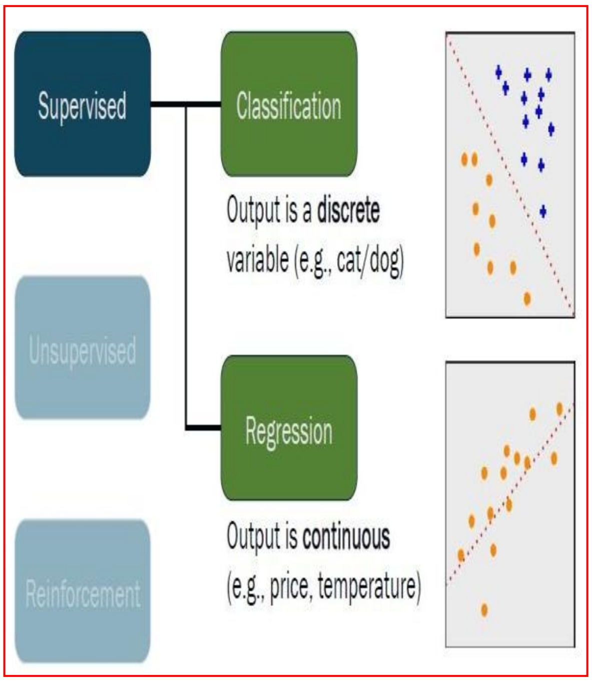
Supervised — Learn through **examples** of which we know the desired output (what we want to predict).

Is this a cat or a dog?

Are these emails spam or not?

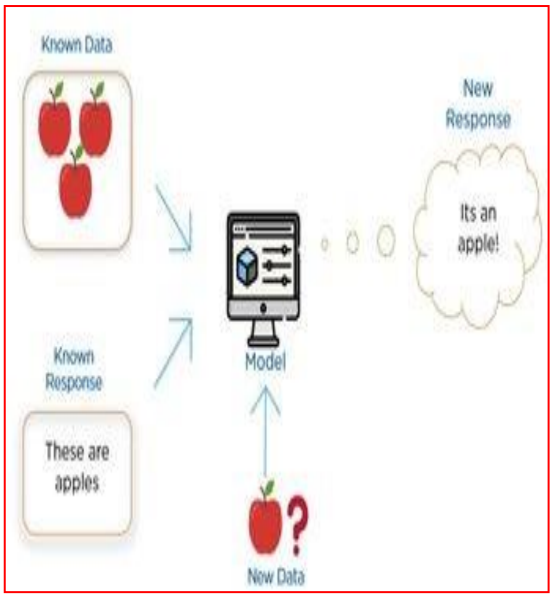
Predict the market value of houses, given the square meters, number of rooms, neighborhood, etc.

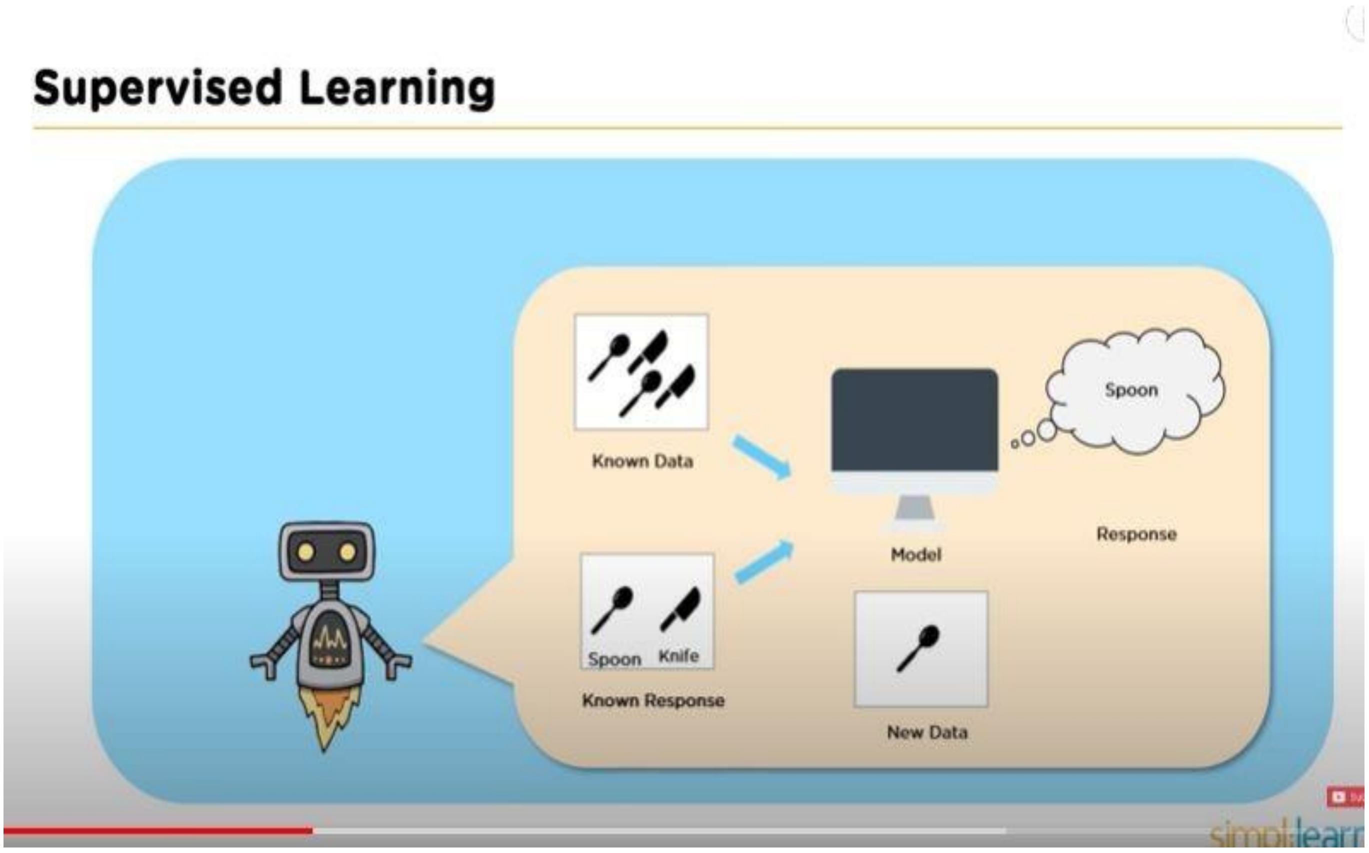
Unsupervised
Reinforcement


Supervised
Unsupervised
Reinforcement
Classification
Output is a discrete variable (e.g., cat/dog)
Regression
Output is continuous (e.g., price, temperature)

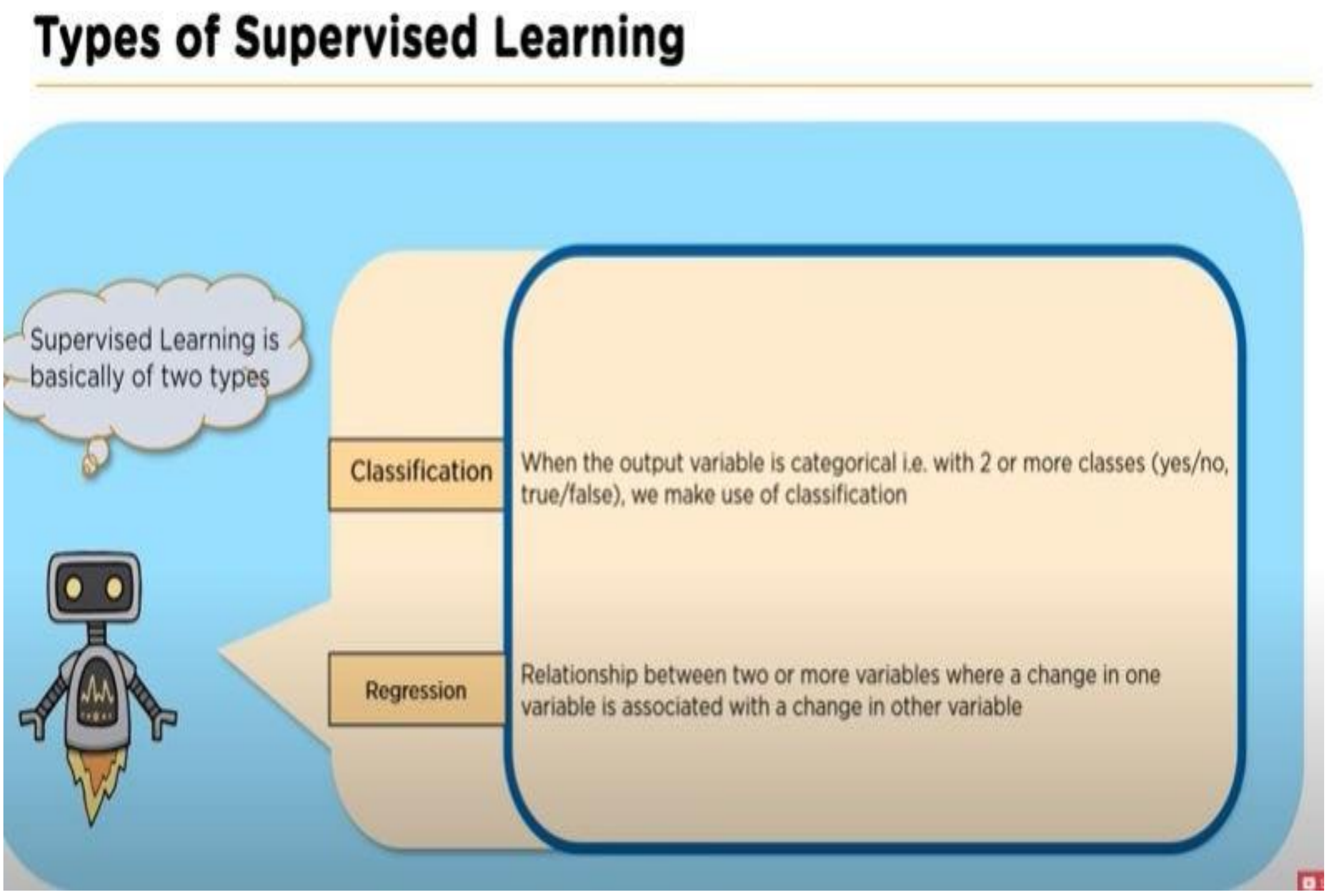
Supervised learning

It is very similar to teaching a child with the use of flash cards





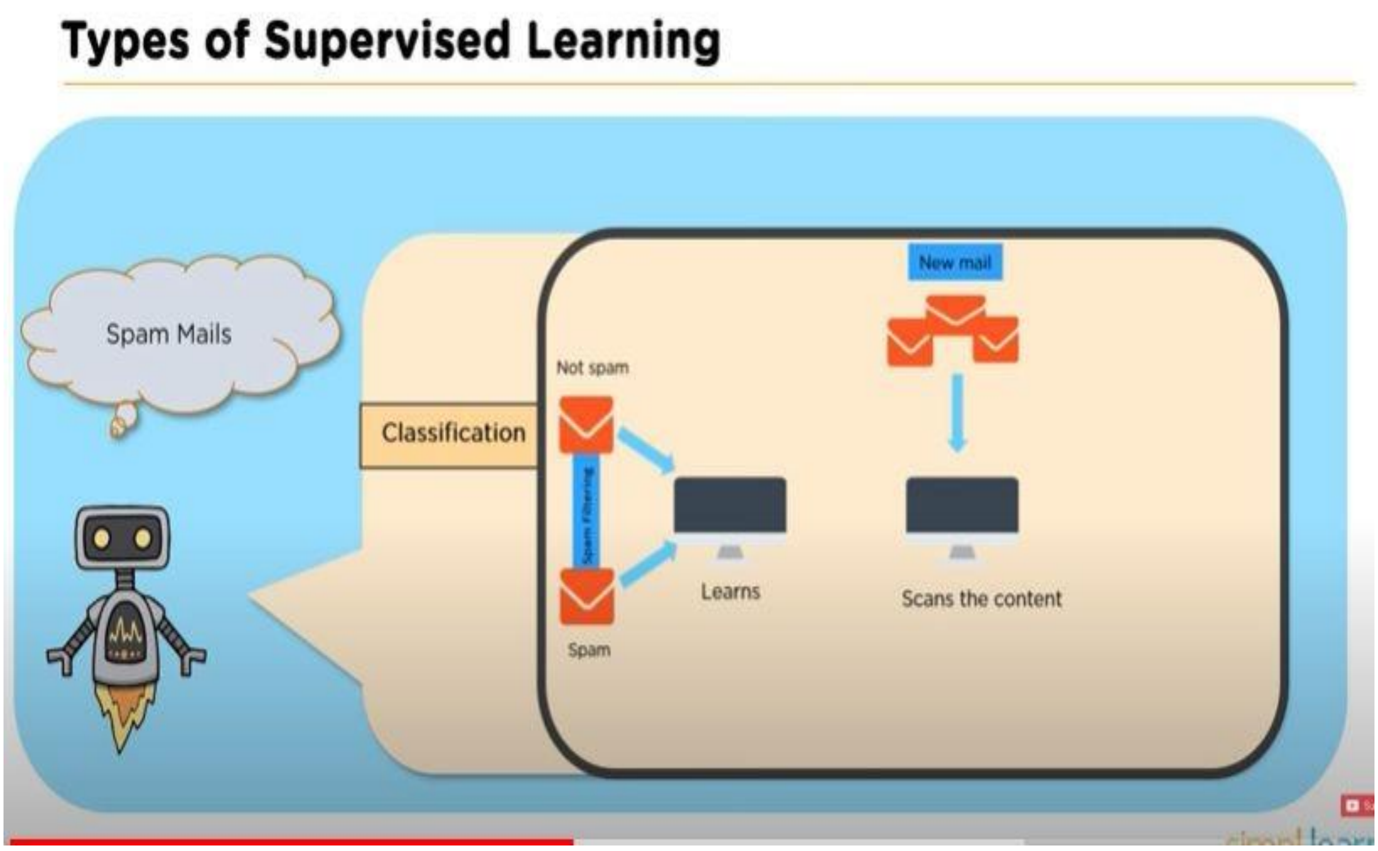
Supervised Learning



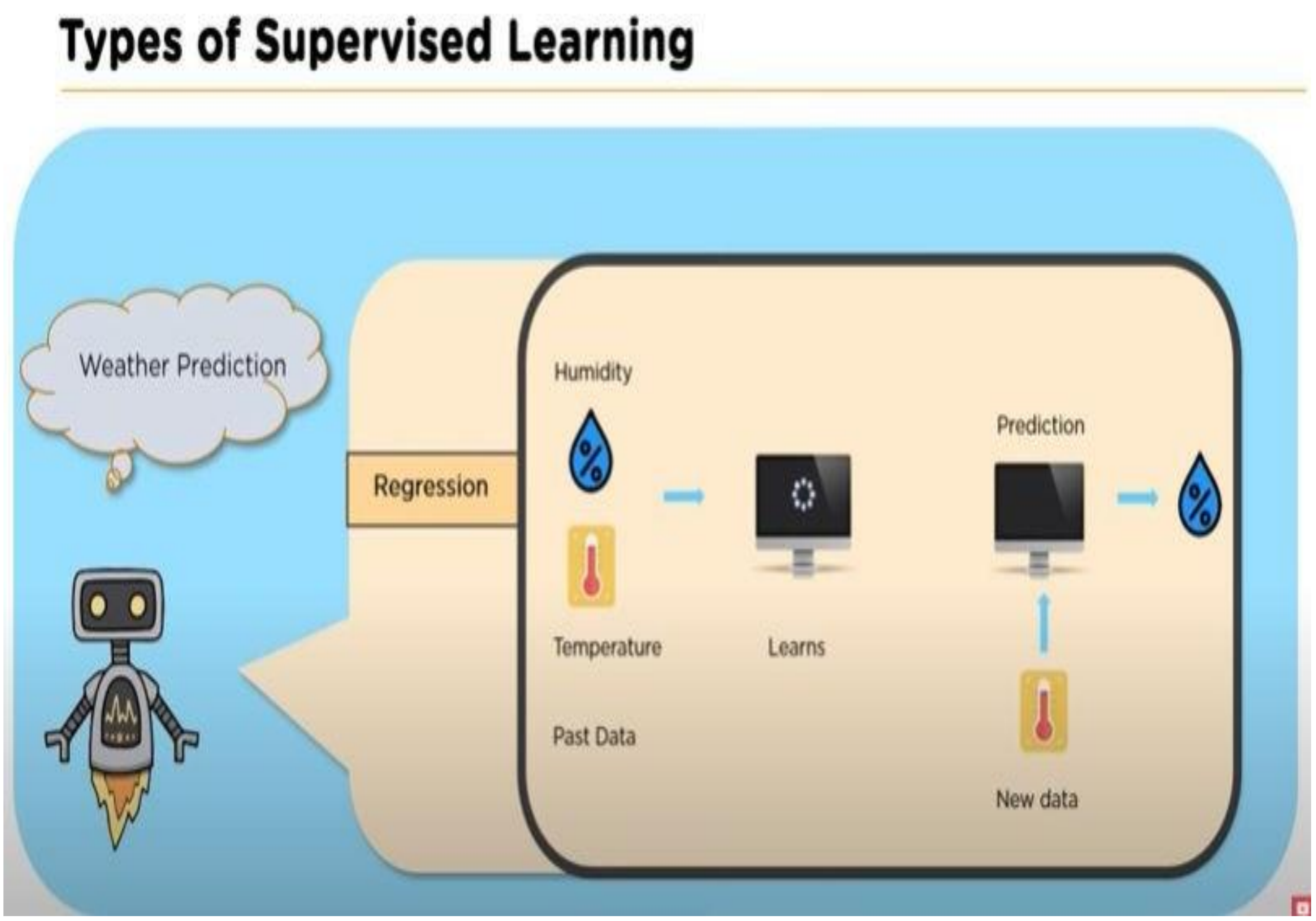
Types of Supervised Learning

Supervised Learning is basically of two types

Classification	When the output variable is categorical i.e. with 2 or more classes (yes/no, true/false), we make use of classification
Regression	Relationship between two or more variables where a change in one variable is associated with a change in other variable



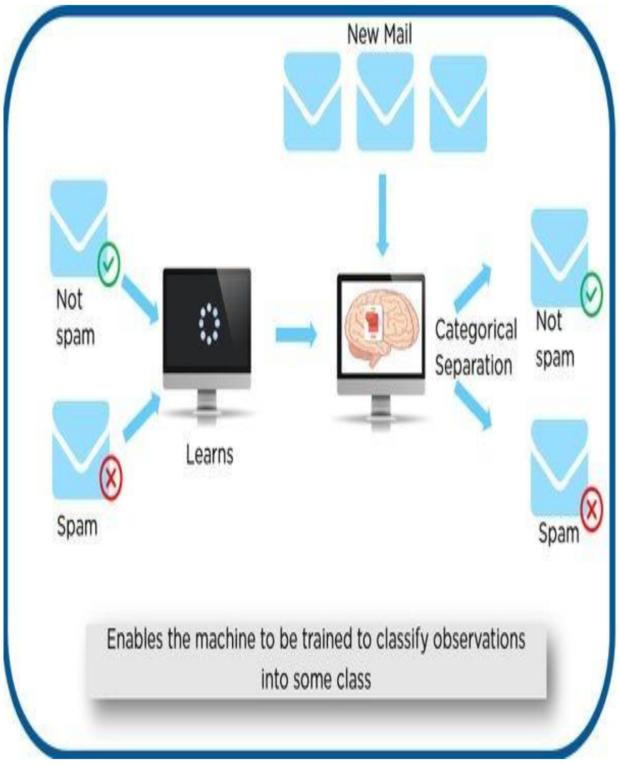
Types of Supervised Learning



Types of Supervised Learning

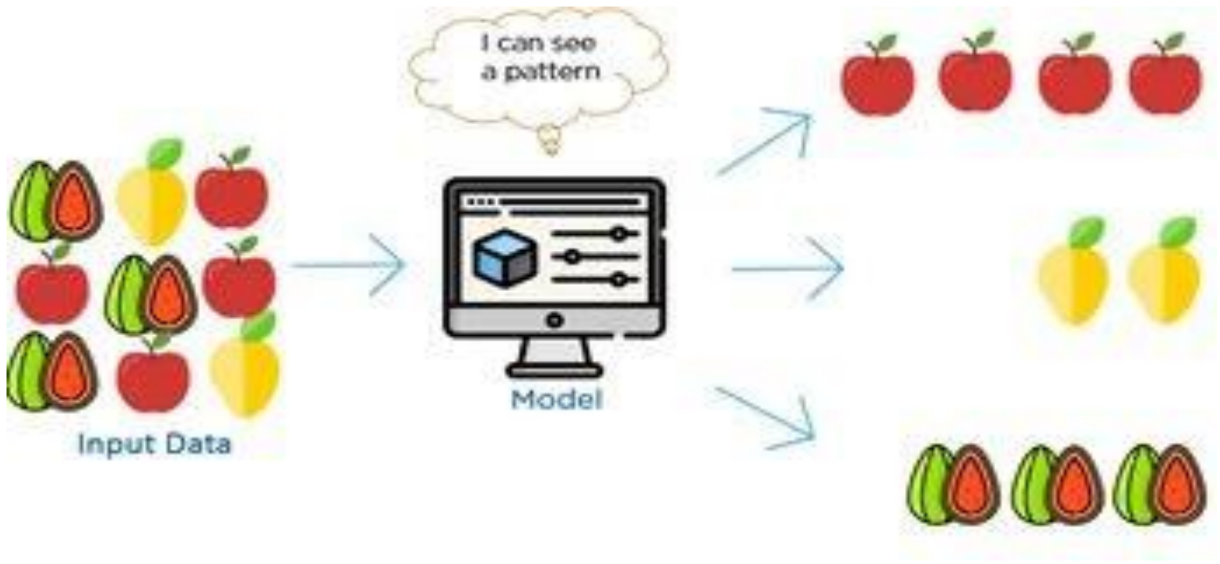
Application of supervised learning

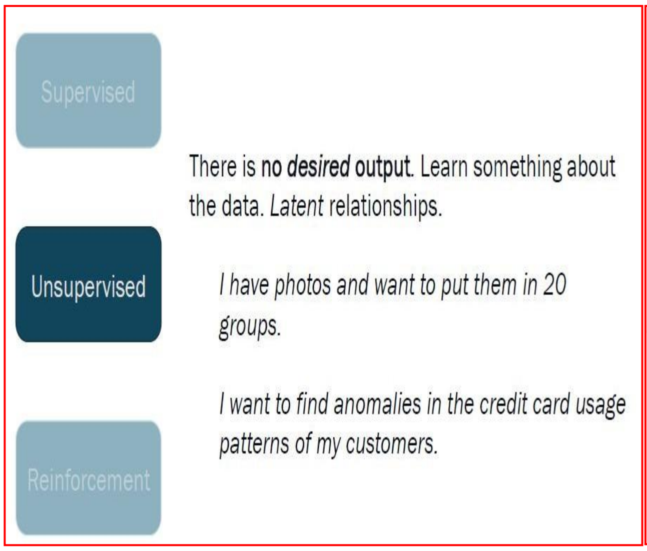
- Advertisement Popularity: Selecting advertisements - Many of the ads - internet are placed because a learning algorithm.
- Spam Classification: If you use a modern email system, chances are you've encountered a spam filter. That spam filter is a supervised learning system. Fed email examples and labels (spam/not spam), these systems learn how to preemptively filter out malicious emails so that their user is not harassed by them. Many of these also behave in such a way that a user can provide new labels to the system and it can learn user preference.
- Face Recognition: Do you use Facebook? Most likely your face has been used in a supervised learning algorithm that is trained to recognize your face. Having a system that takes a photo, findsfaces, and guesses who that is in the photo (suggesting a tag) is a supervised process. It has multiple layers to it, finding faces and then identifying them, but is still supervised nonetheless.


New Mail
Not spam
Spam
Learns
Categorical Separation
Not spam
Spam
Enables the machine to be trained to classify observations into some class

unsupervised learning

- Unsupervised learning is very much the opposite of supervised learning. It featuresno labels. Instead, our algorithm would be fed a lot of data and given the tools to understand the properties of the data. Fromthere, it can learn to group, cluster, and/or organize the data in a way such that a human (or other intelligent algorithm) can come in and make sense of the newly organized data.





Supervised

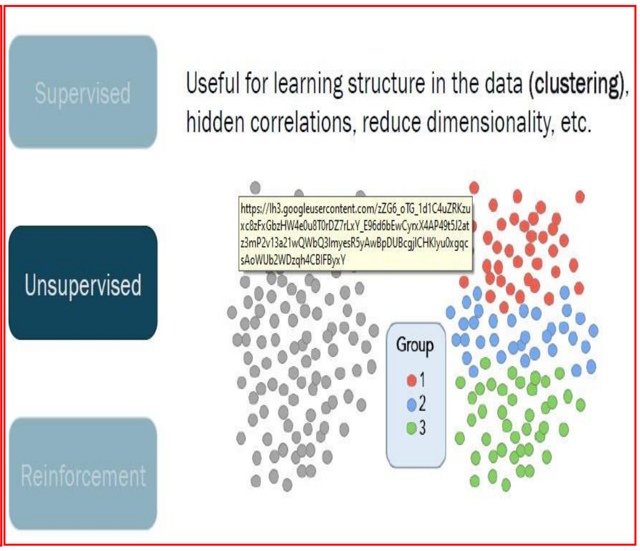
Unsupervised

Reinforcement

There is **no *desired* output**. Learn something about the data. *Latent* relationships.

I have photos and want to put them in 20 groups.

I want to find anomalies in the credit card usage patterns of my customers.



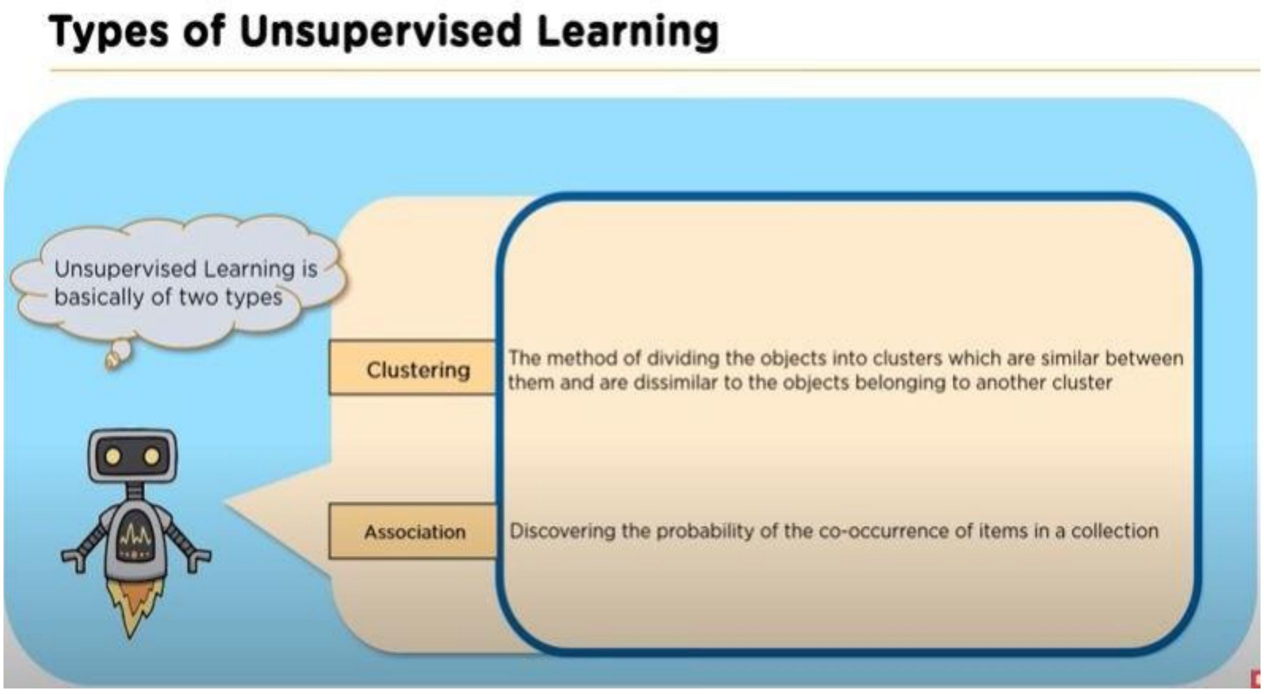
Supervised

Unsupervised

Reinforcement

Useful for learning structure in the data (**clustering**), hidden correlations, reduce dimensionality, etc.

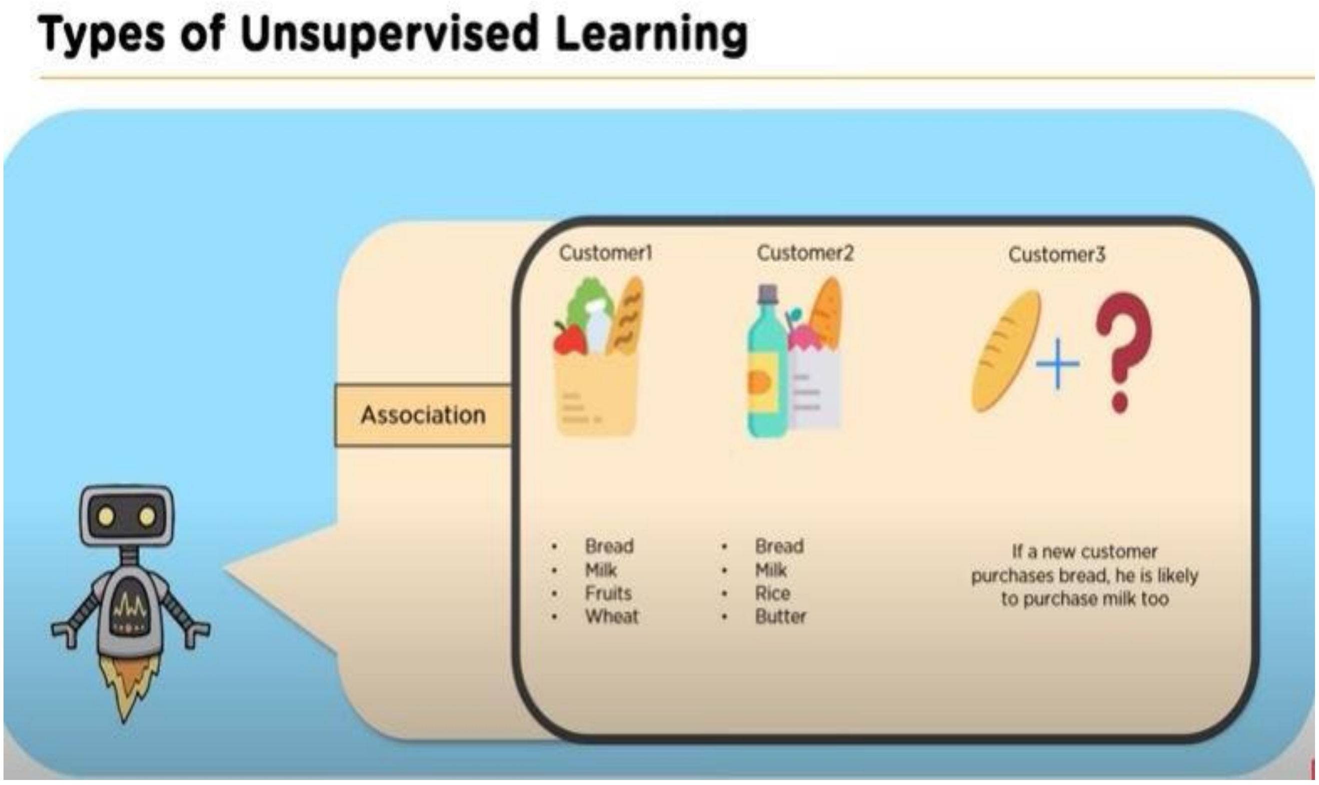
https://lh3.googleusercontent.com/zZG6_oTG_1d1C4uZRKzuxc8zFxGbzHW4e0u8T0rDZ7rLxY_E96d6bEwCyrxX4AP49t5J2atz3mP2v13a21wQWbQ3ImyesR5yAwBpDUBcgjICHKIyu0xgqcsAoWUb2WDzqh4CBIFByxY



Types of Unsupervised Learning

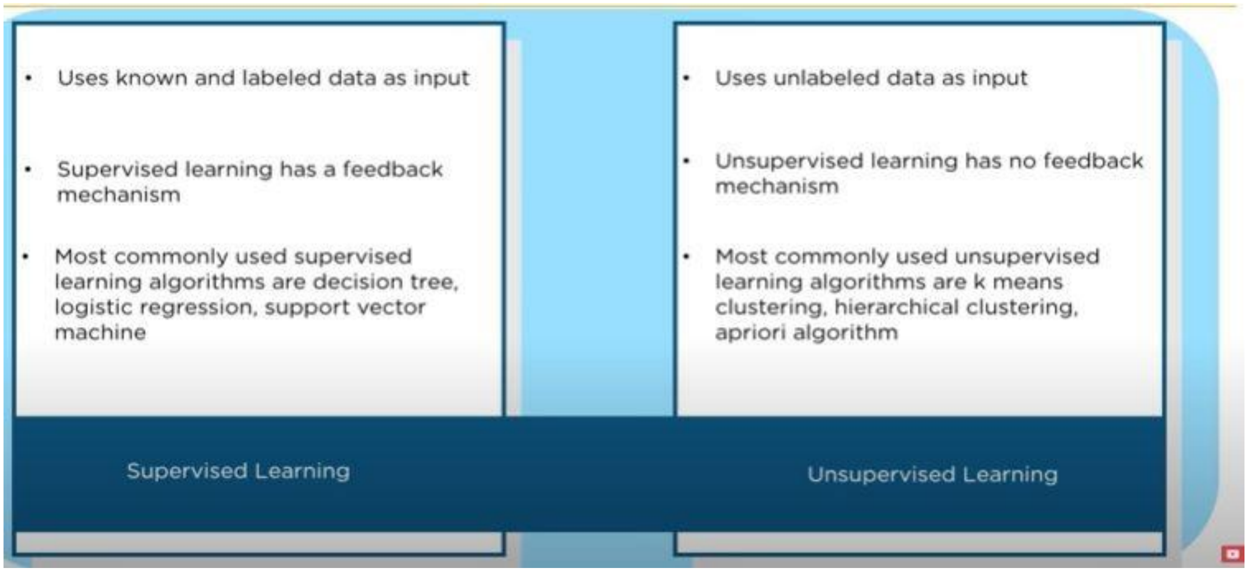
Unsupervised Learning is basically of two types

Clustering	The method of dividing the objects into clusters which are similar between them and are dissimilar to the objects belonging to another cluster
Association	Discovering the probability of the co-occurrence of items in a collection


Types of Unsupervised Learning
Association
Customer1
Customer2
Customer3
• Bread
• Milk
• Fruits
• Wheat
• Bread
• Milk
• Rice
• Butter
If a new customer purchases bread, he is likely to purchase milk too

Application of unsupervised learning

- Recommender Systems: If you've ever used YouTube or Netflix, you've most likely encountered a video recommendation system. These systems are often times placed in the unsupervised domain. We know things about videos, maybe their length, their genre, etc. We also know the watch history of many users. Considering usersthat have watched similar videos as you and then enjoyed other videos that you have yet to see, a recommender system can see this relationship in the data and prompt you with such a suggestion.
- Buying Habits-Market based analysis: Buying habits -contained in a database somewhere and that data is being bought and sold actively at this time. These buying habits can be used in unsupervised learning algorithms to group customers into similar purchasing segments. This helps companies' market to these grouped segments and can even resemble recommender systems.
- Grouping User Logs-semantic clustering: Less user facing, but still very relevant, we can use unsupervised learningto group user logs and issues. This can help companies identify central themes to issues their customers face and rectify these issues, through improving a product or designing an FAQ to handle common issues.



- Uses known and labeled data as input
- Supervised learning has a feedback mechanism
- Most commonly used supervised learning algorithms are decision tree, logistic regression, support vector machine

Supervised Learning

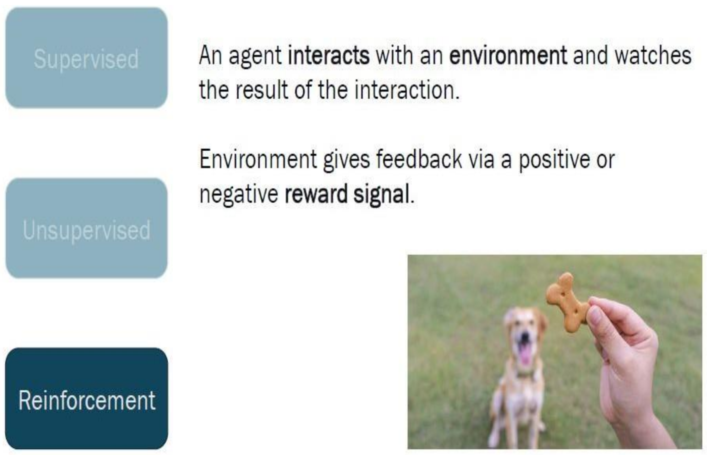
- Uses unlabeled data as input
- Unsupervised learning has no feedback mechanism
- Most commonly used unsupervised learning algorithms are k means clustering, hierarchical clustering, apriori algorithm

Unsupervised Learning

Reinforcement of learning

• Reinforcement learning as learning from mistakes. Placea reinforcement learning algorithm into any environmentand it will make a lotof mistakes in the beginning.

Machine Learning for Humans



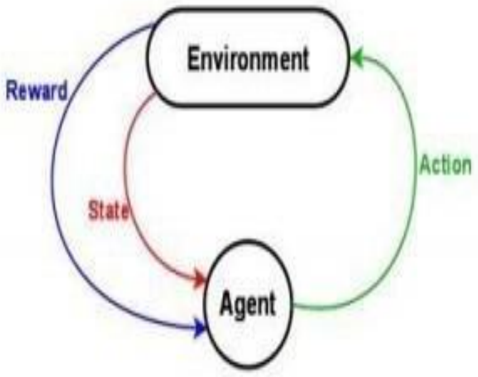
Supervised

Unsupervised

Reinforcement

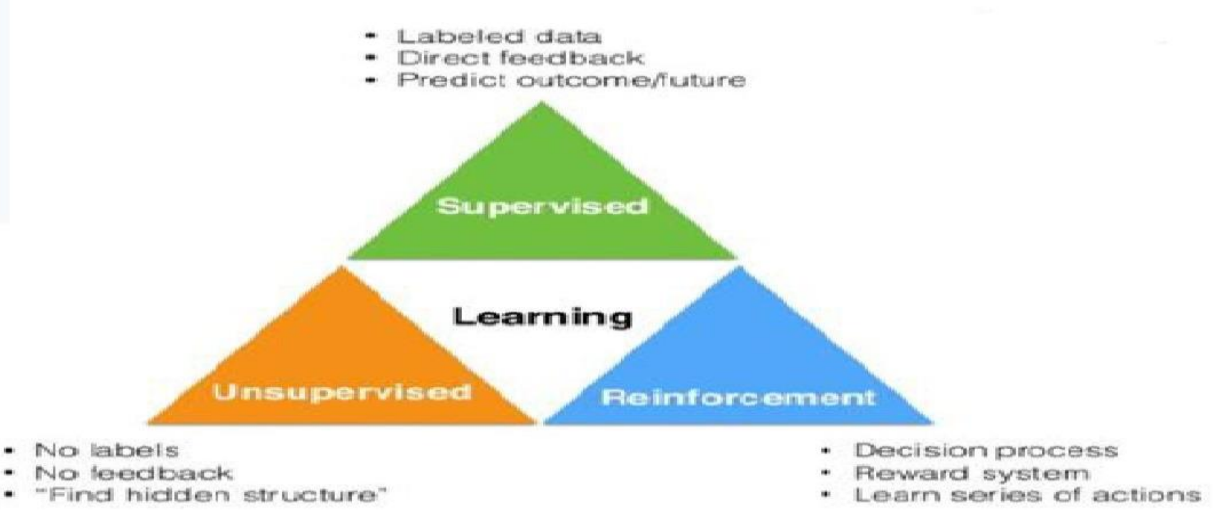
An agent **interacts** with an **environment** and watches the result of the interaction.

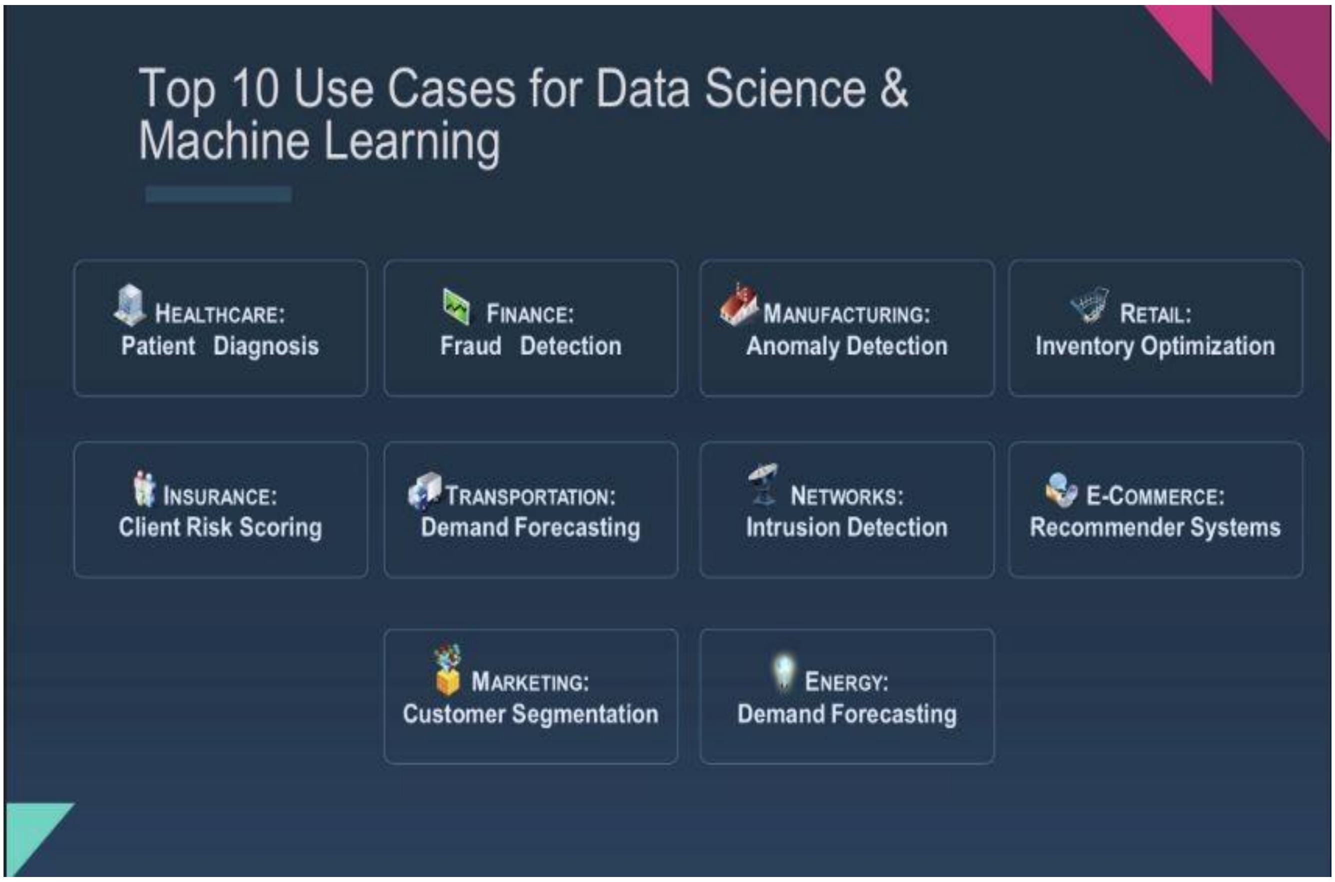
Environment gives feedback via a positive or negative **reward signal**.



Application of Reinforcement learning

- **Video Games**: Google's reinforcement learning application, AlphaZero and AlphaGo which learned to play the game Go. Our Mario example is also a common example
- **Industrial Simulation**: For many robotic applications (think assemblylines), it is useful to have our machines learn to complete their tasks without having to hardcode their processes.
- **Resource Management**: Reinforcement learning is good for navigating complex environments. It can handle the need to balance certain requirements. For example, Google's data centers.
-

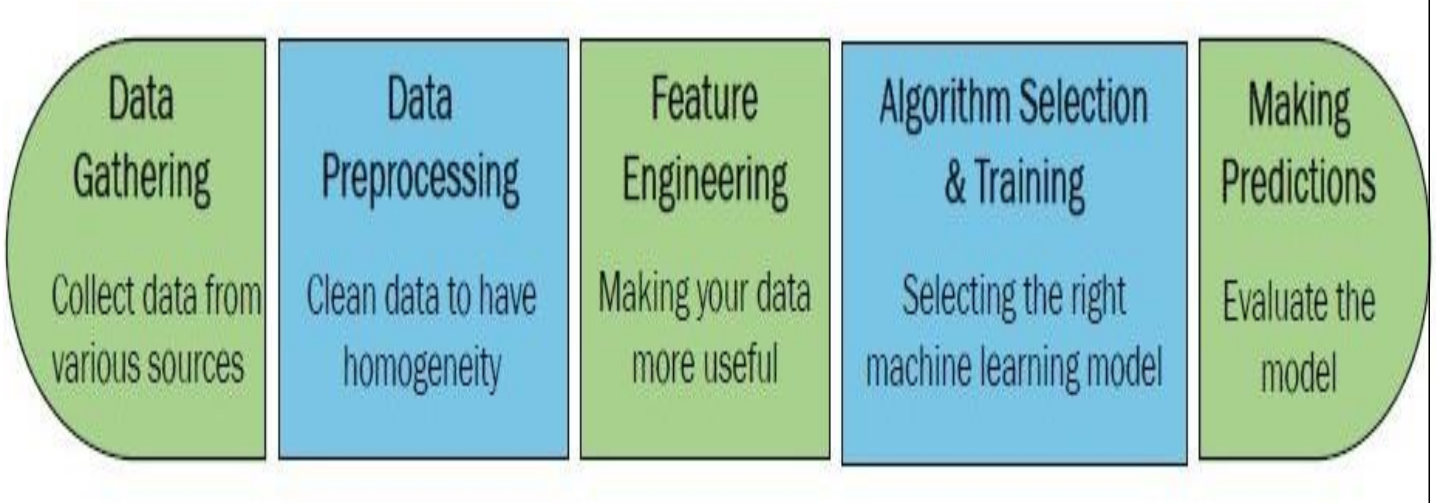


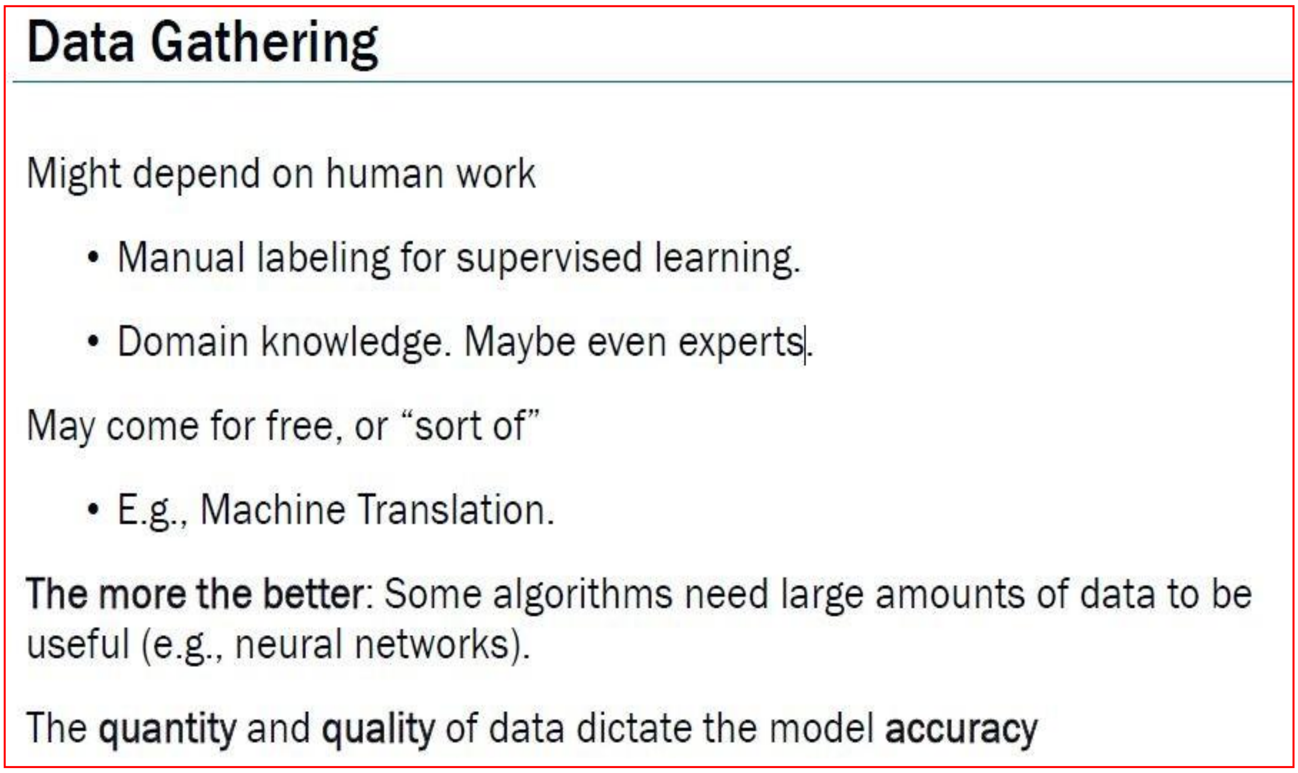


Top 10 Use Cases for Data Science & Machine Learning

HEALTHCARE: Patient Diagnosis	FINANCE: Fraud Detection	MANUFACTURING: Anomaly Detection	RETAIL: Inventory Optimization
INSURANCE: Client Risk Scoring	TRANSPORTATION: Demand Forecasting	NETWORKS: Intrusion Detection	E-COMMERCE: Recommender Systems
	MARKETING: Customer Segmentation	ENERGY: Demand Forecasting	

Steps to solve a Machine Learning Problem


Data Gathering
Collect data from various sources
Data Preprocessing
Clean data to have homogeneity
Feature Engineering
Making your data more useful
Algorithm Selection & Training
Selecting the right machine learning model
Making Predictions
Evaluate the model



Data Gathering

Might depend on human work

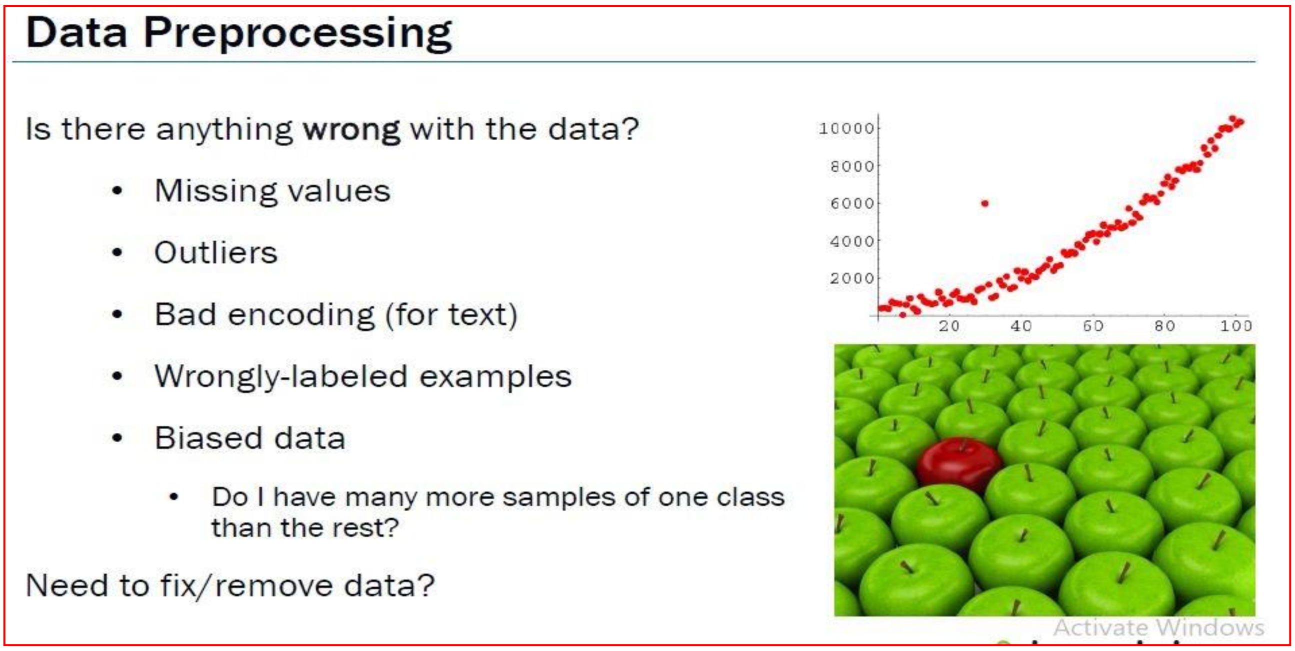
- Manual labeling for supervised learning.
- Domain knowledge. Maybe even experts.

May come for free, or “sort of”

- E.g., Machine Translation.

The more the better: Some algorithms need large amounts of data to be useful (e.g., neural networks).

The **quantity** and **quality** of data dictate the model **accuracy**

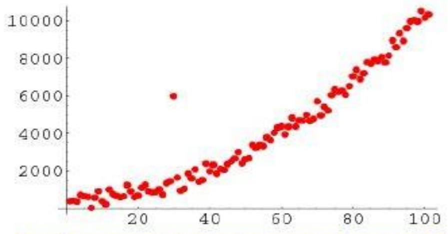


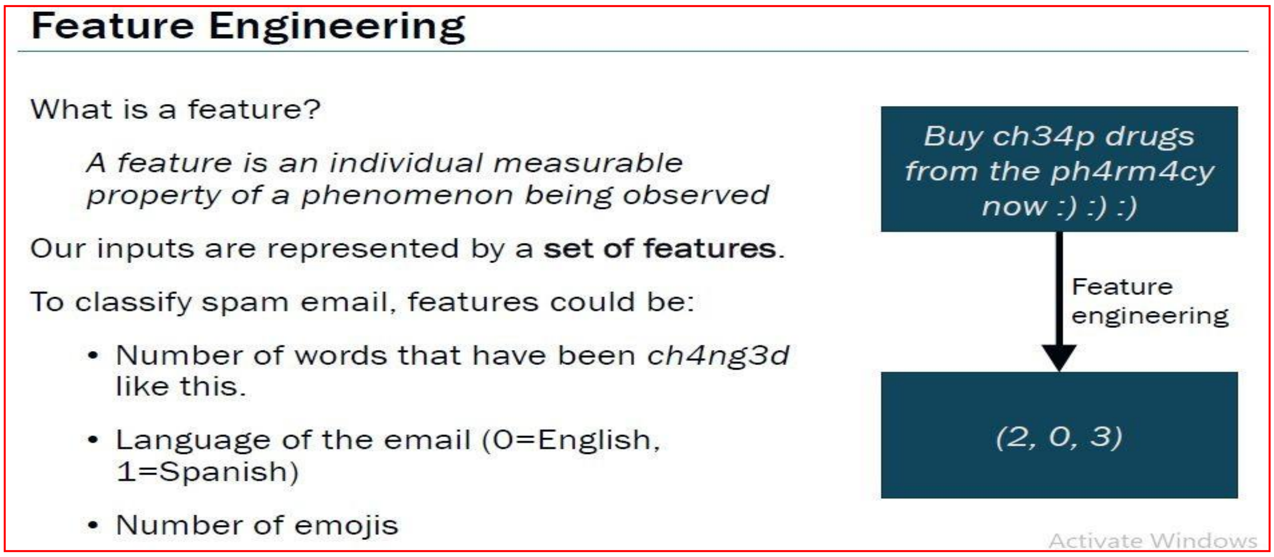
Data Preprocessing

Is there anything **wrong** with the data?

- Missing values
- Outliers
- Bad encoding (for text)
- Wrongly-labeled examples
- Biased data
    - Do I have many more samples of one class than the rest?

Need to fix/remove data?





Feature Engineering

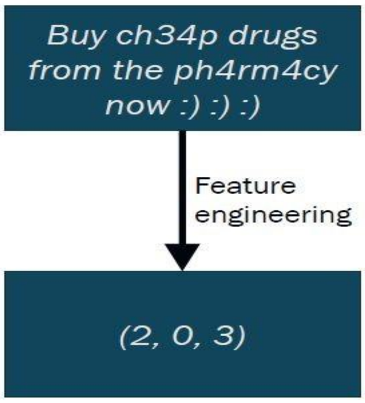
What is a feature?

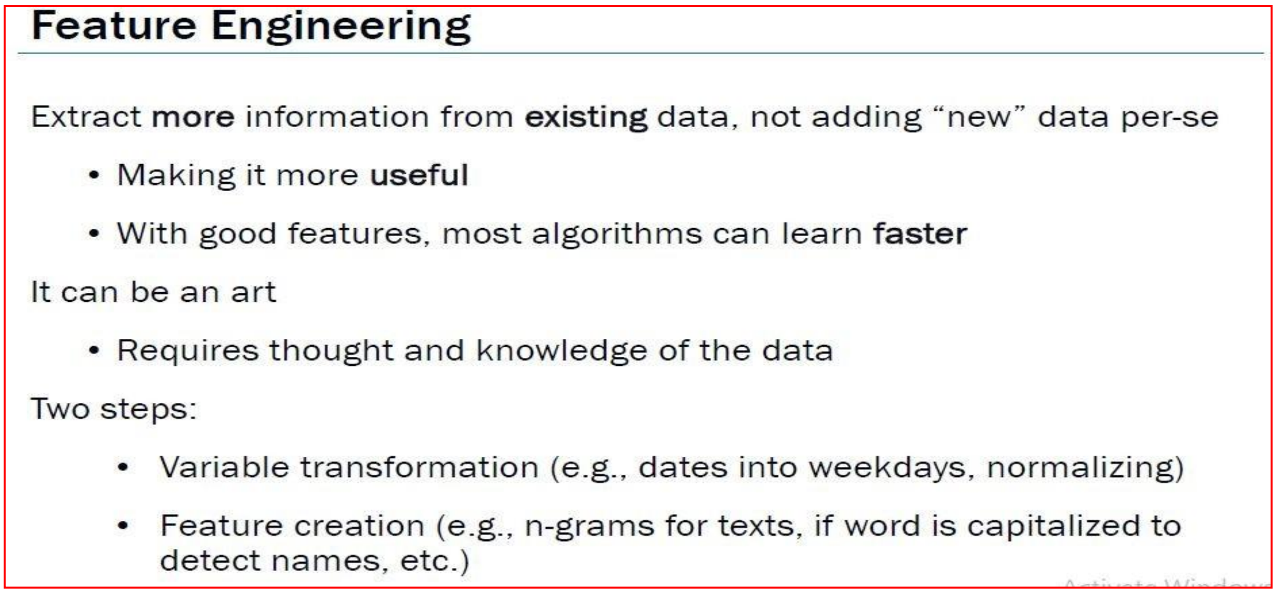
A feature is an individual measurable property of a phenomenon being observed

Our inputs are represented by a **set of features**.

To classify spam email, features could be:

- Number of words that have been *ch4ng3d* like this.
- Language of the email (0=English, 1=Spanish)
- Number of emojis





Feature Engineering

Extract **more** information from **existing** data, not adding "new" data per-se

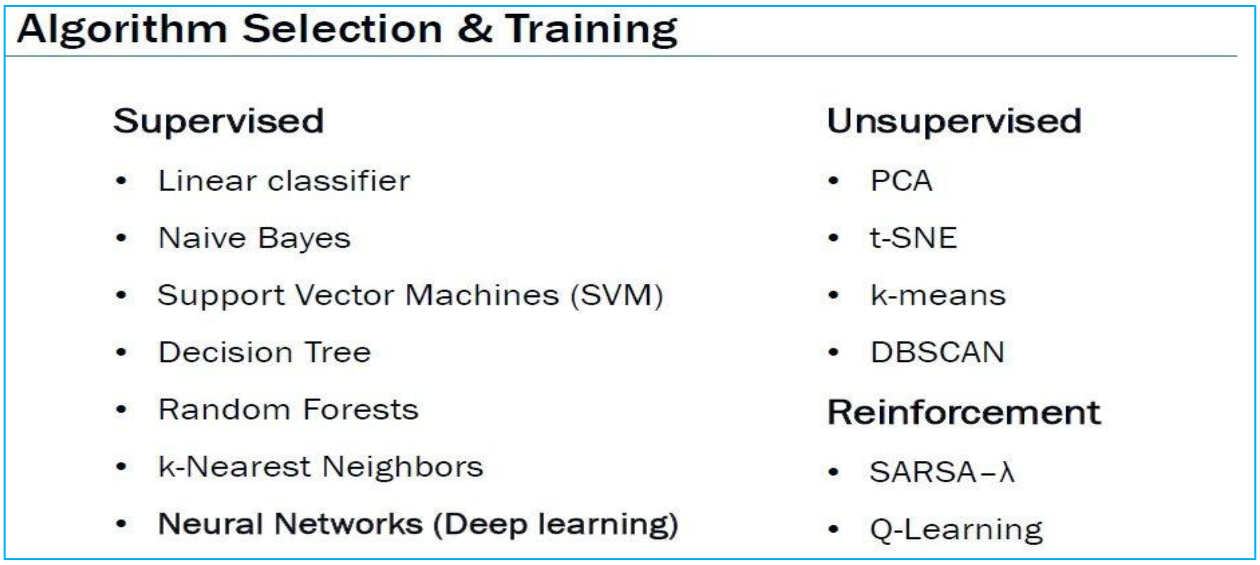
- Making it more **useful**
- With good features, most algorithms can learn **faster**

It can be an art

- Requires thought and knowledge of the data

Two steps:

- Variable transformation (e.g., dates into weekdays, normalizing)
- Feature creation (e.g., n-grams for texts, if word is capitalized to detect names, etc.)



Algorithm Selection & Training

Supervised

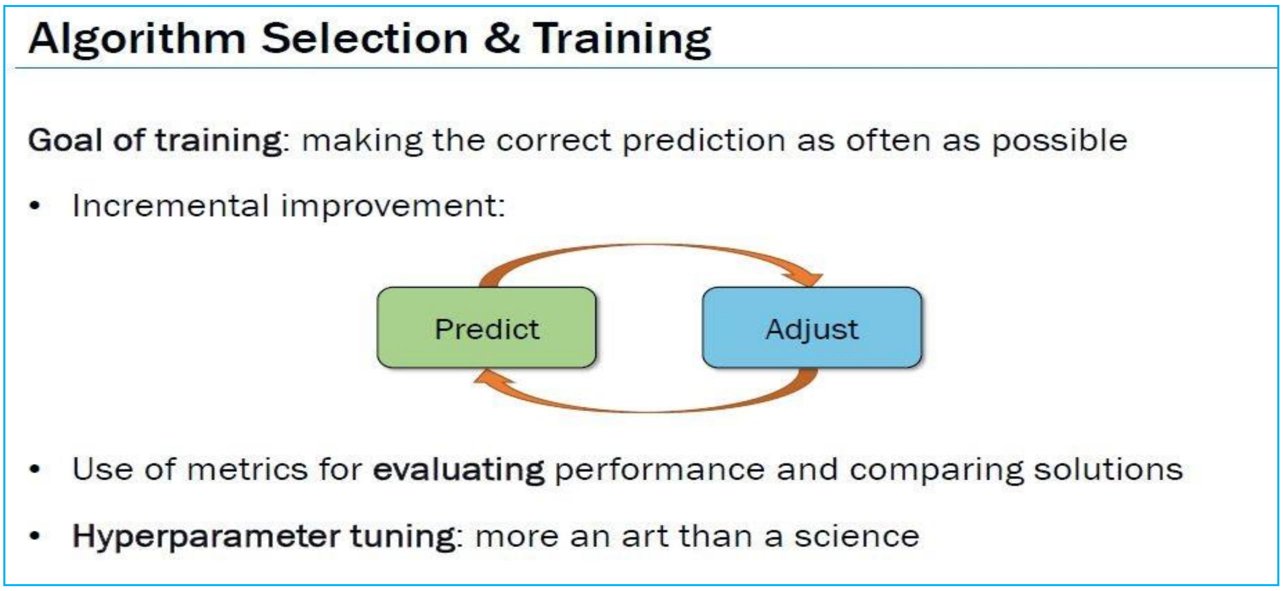
- Linear classifier
- Naive Bayes
- Support Vector Machines (SVM)
- Decision Tree
- Random Forests
- k-Nearest Neighbors
- **Neural Networks (Deep learning)**

Unsupervised

- PCA
- t-SNE
- k-means
- DBSCAN

Reinforcement

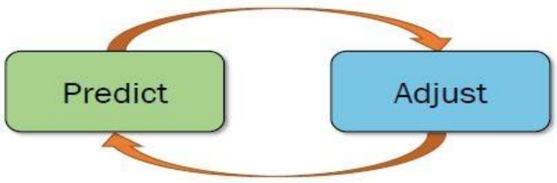
- SARSA-λ
- Q-Learning



Algorithm Selection & Training

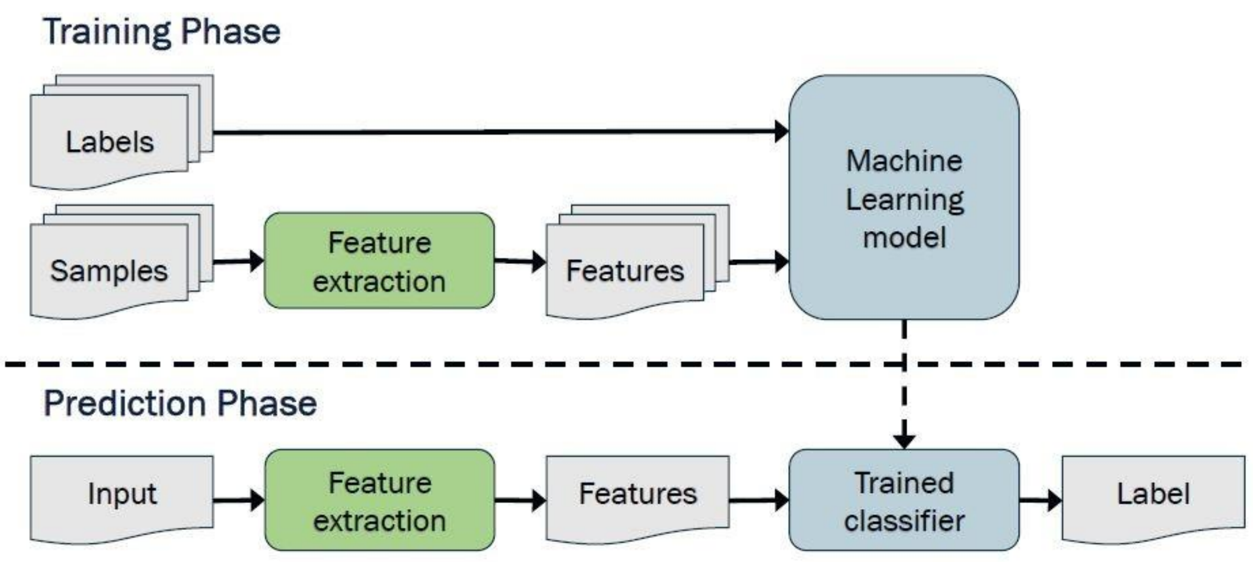
Goal of training: making the correct prediction as often as possible

- Incremental improvement:



- Use of metrics for **evaluating** performance and comparing solutions
- **Hyperparameter tuning**: more an art than a science

Making Predictions



DESIGNING A LEARNING SYSTEM

Basic design issues and approaches to machine learning, let us consider designing a program to learn to playcheckers, with the goal of entering it in the world checkers tournament.

Performance measure: the percent of games it wins in this world tournament.

1.2.1 Choosing the Training Experience

The first design choice is to choose the type of training experience from which our system will learn. The type oftraining experience available can have a significant impact on success or failure of the learner. (driving class to drive a car)

Three are three attributes which impact on success or failure of the learner

Whether the training experience provides direct or indirect feedback regarding the choices made by theperformance system.

The degree to which the learner controls the sequence of training examples

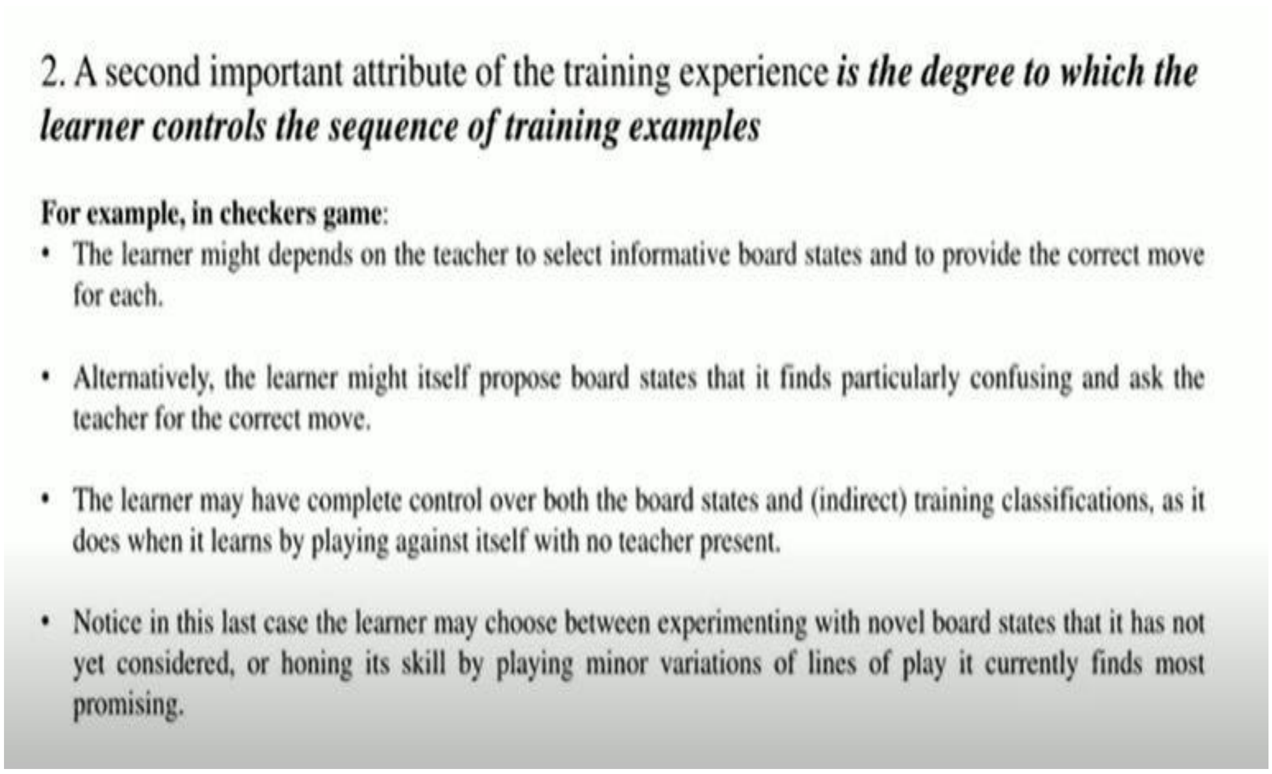
How well it represents the distributing of examples over which the final system performance P must be measured.

Example : Chess problem

1. Whether the training experience provides direct or indirect feedback regarding the choices made by theperformance system.

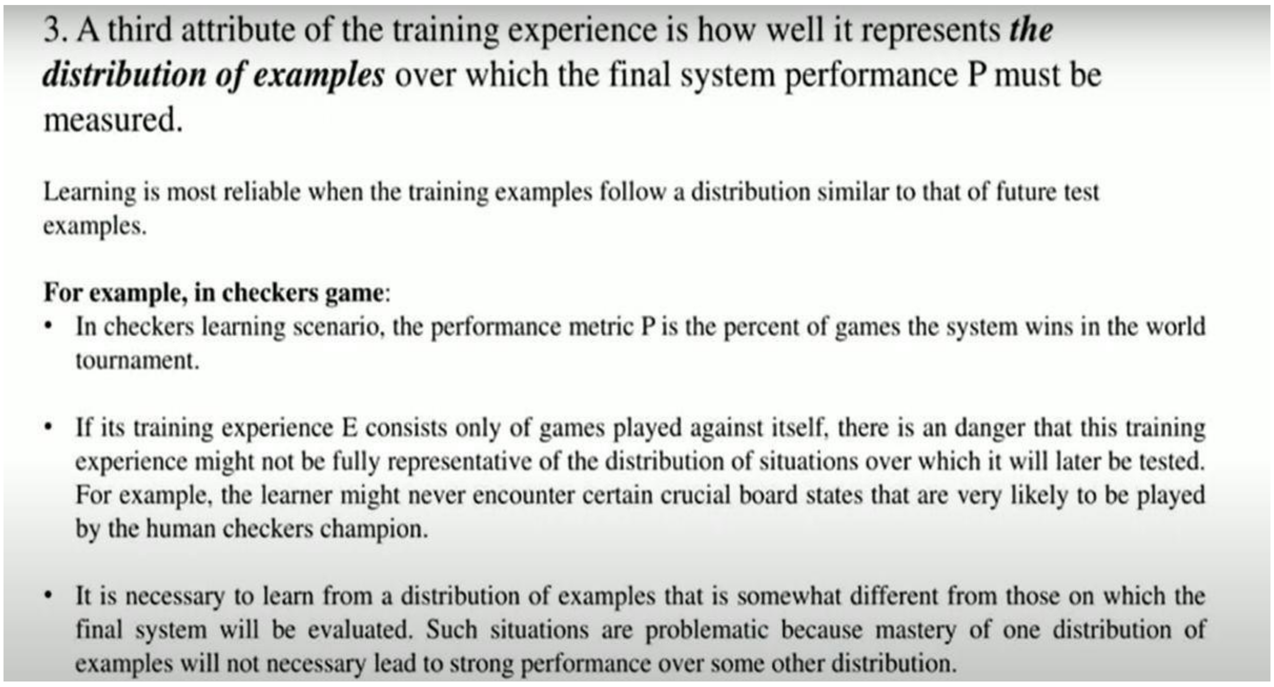
For example in checkers game

- In learning to play checkers, the system might learn from *direct* training examples consisting of individualcheckers board states and the correct move for each.
- *Indirect* training examples consisting of the move sequences and final outcomes of various games played.
- The information about the correctness of specific moves early in the game must be inferred indirectly from thefact that the game was eventually won or lost.
- Here the learner faces an additional problem of *credit assignment*, or determining the degree to which eachmove in the sequence deserves credit or blame for the final outcome.
- Credit assignment can be a particularly difficult problem because the game can be lost even when early movesare optimal, if these are followed later by poor moves
- Hence learning from direct training feedback is typically easier than learning from indirect feedback


2. A second important attribute of the training experience ***is the degree to which the learner controls the sequence of training examples***

For example, in checkers game:

- The learner might depends on the teacher to select informative board states and to provide the correct move for each.
- Alternatively, the learner might itself propose board states that it finds particularly confusing and ask the teacher for the correct move.
- The learner may have complete control over both the board states and (indirect) training classifications, as it does when it learns by playing against itself with no teacher present.
- Notice in this last case the learner may choose between experimenting with novel board states that it has not yet considered, or honing its skill by playing minor variations of lines of play it currently finds most promising.


3. A third attribute of the training experience is how well it represents ***the distribution of examples*** over which the final system performance P must be measured.

Learning is most reliable when the training examples follow a distribution similar to that of future test examples.

For example, in checkers game:

- In checkers learning scenario, the performance metric P is the percent of games the system wins in the world tournament.
- If its training experience E consists only of games played against itself, there is an danger that this training experience might not be fully representative of the distribution of situations over which it will later be tested. For example, the learner might never encounter certain crucial board states that are very likely to be played by the human checkers champion.
- It is necessary to learn from a distribution of examples that is somewhat different from those on which the final system will be evaluated. Such situations are problematic because mastery of one distribution of examples will not necessary lead to strong performance over some other distribution.

A checkers learning problem:

Task T: playing checkers

Performance measure ***P:*** percent of games won in the world tournament

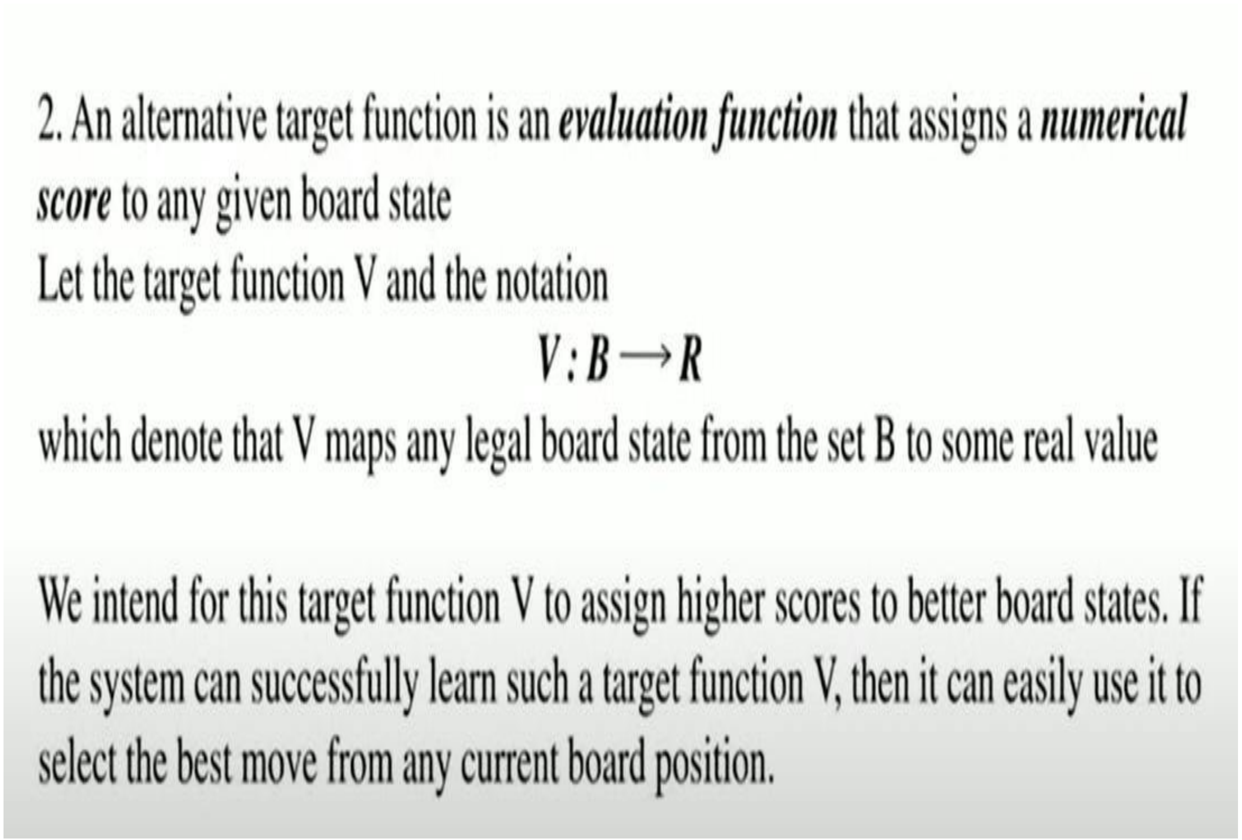
Training experience E: games played against itself

In order to complete the design of the learning system, we must now choose

1. the exact type of knowledge to be, learned
2. a representation for this target knowledge
3. a learning mechanism

Choosing the Target Function

- The next design choice is to determine exactly what type of knowledge will belearned and how this will be used by the performance program. Let us begin with a checkers-playing program that can generate the *legal* moves fromany board state.
- The program needs only to learn how to choose the *best* move from among theselegal moves. This learning task is representative of a large class of tasks for which the legal moves that define some large search space are known a priori, but for which the best search strategy is not known.
- Many optimization problems fall into this class, such as the problems of scheduling and controlling manufacturing processes where the available manufacturing steps are well understood, but the best strategy for sequencing them isnot.



2. An alternative target function is an ***evaluation function*** that assigns a ***numerical score*** to any given board state

Let the target function V and the notation

$$V : B \longrightarrow R$$

which denote that V maps any legal board state from the set B to some real value

We intend for this target function V to assign higher scores to better board states. If the system can successfully learn such a target function V, then it can easily use it to select the best move from any current board position.

Let us therefore define the target value ***V(b)*** for an arbitrary board state ***b*** in

B, ***as*** follows:

1. if ***b*** is a final board state that is won, then ***V(b)*** = 100
2. if b is a final board state that is lost, then ***V(b)*** = -100
3. if b is a final board state that is drawn, then ***V(b) = 0***
4. if b is a not a final state in the game, then V(b) = V(b'), where b' is the bestfinal board state that can be achieved starting from b and playing **optimally** until the end of the game

1.2.2 Choosing a Representation for the Target Function

let us choose a simple representation: for any given board state, the function c will be calculated as **alinear combination** of the following board features:

- ***xl:*** the number of black pieces on the board
- ***x2:*** the number of red pieces on the board
- ***x3:*** the number of black kings on the board
- **x4:** the number of red kings on the board
- ***x5:*** the number of black pieces threatened by red (i.e., which can be captured on red's next turn)
- ***X6:*** the number of red pieces threatened by black

Thus, our learning program will represent v(b) as a linear function of the form

where ***wo*** through w6 are numerical coefficients, or weights, to be chosen by the learning algorithm. Learned values for the weights ***wl*** through ***W6*** will determine the relative importance of the various boardfeatures in determining the value of the board, whereas the weight ***wo*** will provide an additive constant tothe board value

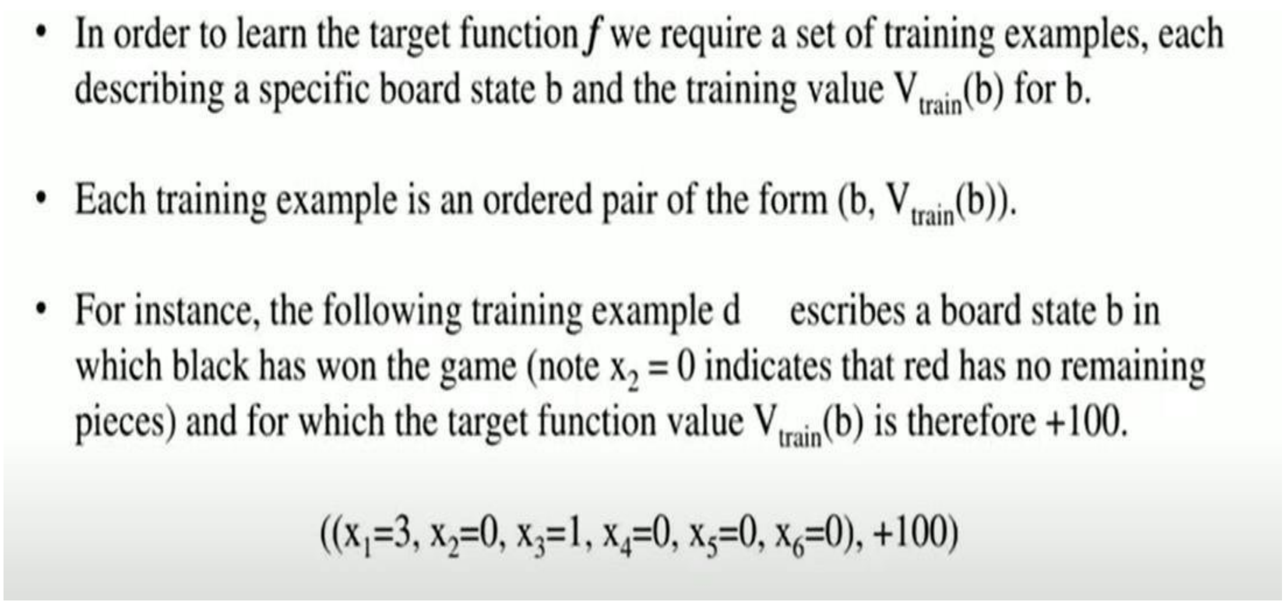
$$\hat{V}(b) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6$$

1.2.3 Choosing a Function Approximation Algorithm

- In order to learn the target function **v** we require a set of training examples, eachdescribing a specific board state b and the training value Vtrain(b) for b.
- In other words, each training example is an ordered pair of the form (b, V',,,i,(b)).
- For instance, the following training example describes a board state b in which black
- Function Approximation Procedure

1 Derive training examples from the indirect training experience available to thelearner

2 Adjust the weights w_i to best fit these training examples- reduce error



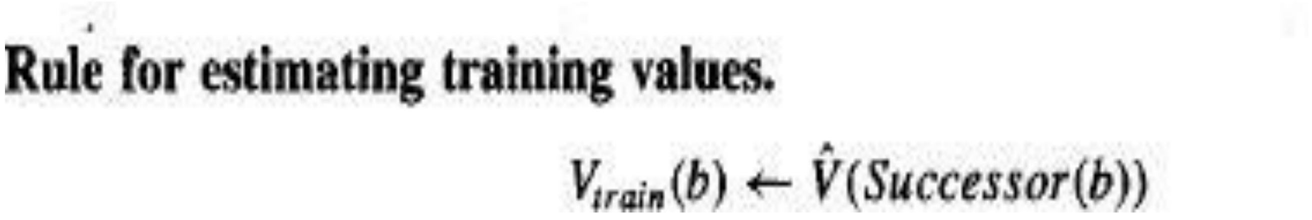
- In order to learn the target function *f* we require a set of training examples, each describing a specific board state b and the training value $V_{train}(b)$ for b.
- Each training example is an ordered pair of the form (b, $V_{train}(b)$).
- For instance, the following training example d escribes a board state b in which black has won the game (note $x_2 = 0$ indicates that red has no remaining pieces) and for which the target function value $V_{train}(b)$ is therefore +100.

$$((x_1=3, x_2=0, x_3=1, x_4=0, x_5=0, x_6=0), +100)$$

1.2.4.1 ESTIMATING TRAINING VALUES

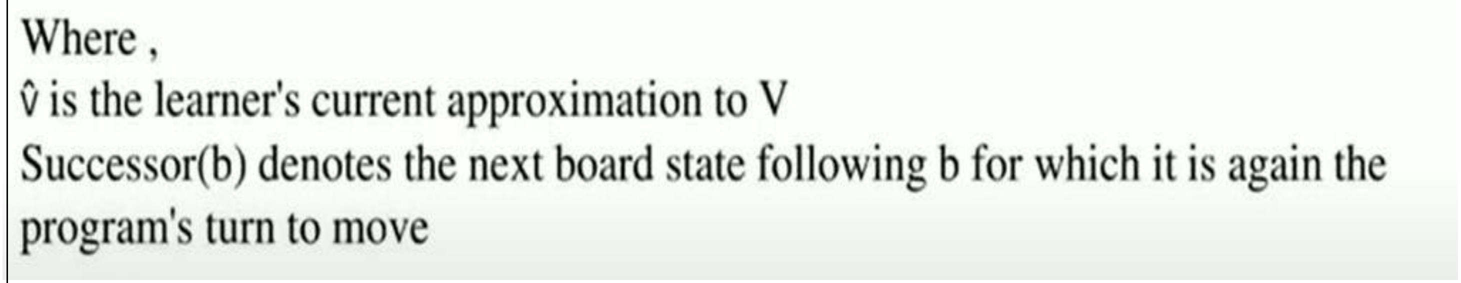
□ assign the training value of Vtrain(b) for any intermediate board state b to be V(successor(b), where v is the learner's current approximation to Vand where Successor(b) denotes the next board state following b

(what will be opponent move)



Rule for estimating training values.

$$V_{train}(b) \leftarrow \hat{V}(Successor(b))$$



Where ,
$\hat{v}$ is the learner's current approximation to V
Successor(b) denotes the next board state following b for which it is again the program's turn to move

1.2.4.2 ADJUSTING THE WEIGHTS

- ❖ All that remains is to specify the learning algorithm for choosing theweights wi to best fit the set of training examples {(b, Vtrain(b))}.
- ❖ As a first step we must define what we mean by the best fit to the trainingdata.
- ❖ One common approach is to define the best hypothesis, or set of weights, asthat which minimizes the squared error E between the training values and the values predicted by the hypothesis V. If error =0.2 then squared 0.04.it reduces error

$$E \equiv \sum_{\langle b, V_{train}(b)\rangle \in \text{ training examples}} (V_{train}(b) - \hat{V}(b))^2$$

- One such algorithm is called the least mean squares, or LMS training rule. For each observed training example, it adjusts the weights a small amount inthe direction that reduces the error on this training example.

LMS weight update rule.

For each training example $\langle b, V_{train}(b)\rangle$

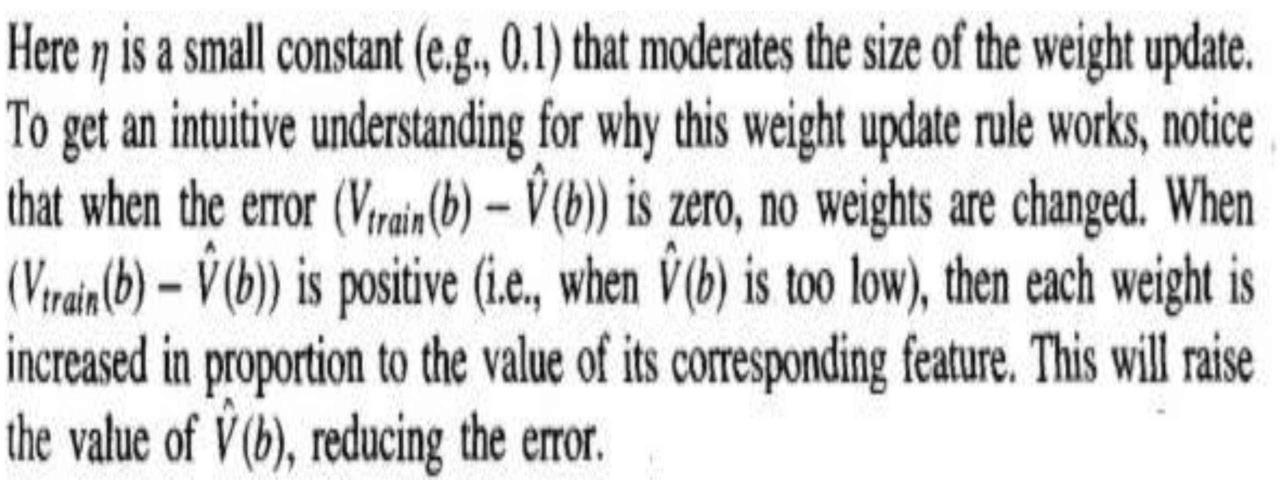
- Use the current weights to calculate $\hat{V}(b)$
- For each weight w_i, update it as

$$w_i \leftarrow w_i + \eta \; (V_{train}(b) - \hat{V}(b)) \; x_i$$

Here η is a small constant (e.g., 0.1) that moderates the size of the weight update.

<u>Working of weight update rule</u>

- When the error (Vtrain(b)- V̂(b)) is zero, no weights are changed.
- When (Vtrain(b) - V̂(b)) is positive (i.e., when V̂(b) is too low), then each weight is increased in proportion to the value of its corresponding feature. This will raise the value of V̂(b), reducing the error.
- If the value of some feature x_i is zero, then its weight is not altered regardless of the error, so that the only weights updated are those whose features actually occur on the training example board.


Here η is a small constant (e.g., 0.1) that moderates the size of the weight update. To get an intuitive understanding for why this weight update rule works, notice that when the error $(V_{train}(b) - \hat{V}(b))$ is zero, no weights are changed. When $(V_{train}(b) - \hat{V}(b))$ is positive (i.e., when $\hat{V}(b)$ is too low), then each weight is increased in proportion to the value of its corresponding feature. This will raise the value of $\hat{V}(b)$, reducing the error.

1.2.4 The Final Design

- The final design of our checkers learning system can be naturally described by four distinct program modules that represent the central components in many learning systems.
- These four modules, summarized in Figure 1.1, are as follows:
- The **Performance System** is the module that must solve the given performance task, in this case playing checkers, by using the learned targetfunction(s). It takes an instance of a new problem (new game) as input andproduces a trace of its solution (game history) as output
- The Critic takes as input the history or trace of the game and produces asoutput a set of training examples of the target function
- The Generalizer takes as input the training examples and produces an output hypothesis that is its estimate of the target function. It generalizes from the specific training examples, hypothesizing a general function that covers these examples and other cases beyond the training examples. In our example, the Generalizer corresponds to the LMS algorithm, and the output hypothesis is the function f described by the learned weights wo, . . . , W6.
- The Experiment Generator takes as input the current hypothesis (currently learnedfunction) and outputs a new problem (i.e., initial board state) for the Performance System to explore. Its role is to pick new practice problems that will maximize thelearning rate of the overall system. In our example, the Experiment Generator follows a very simple strategy: It always proposes the same initial game board to begin a new game

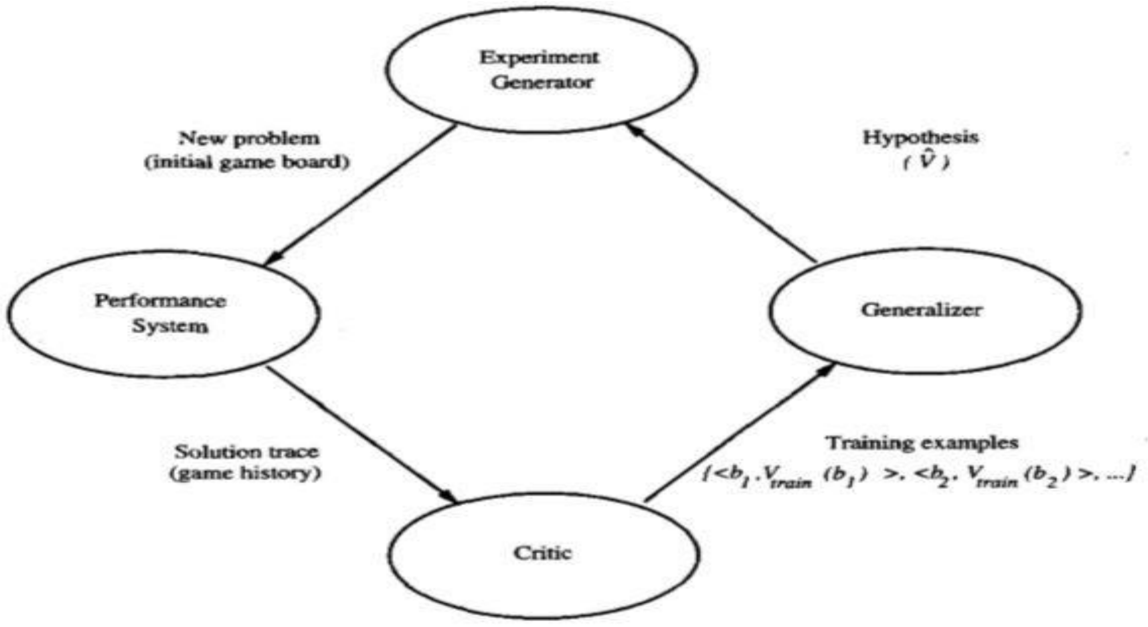

Experiment Generator
New problem (initial game board)
Hypothesis ($\hat{V}$)
Performance System
Generalizer
Solution trace (game history)
Training examples
$\{<b_1, V_{train}(b_1)>, <b_2, V_{train}(b_2)>, ...\}$
Critic

FIGURE 1.1
Final design of the checkers learning program.

Perspective &Issues in Machine Learning Perspective:

It involves searching a very large space of possible hypothesis to determine the onethat best fits the observed data.

Issues:

- Which algorithm performs best for which types of problems & representation?
- How much training data is sufficient?
- Can prior knowledge be helpful even when it is only approximately correct?
- The best strategy for choosing a useful next training experience.
- What specific function should the system attempt to learn?
- How can learner automatically alter it's representation to improve it's ability torepresent and learn the target function?

Concept Learning

- Inducing general functions from specific training examples is a main issue of machine learning.
- Concept Learning: Acquiring the definition of a general category from given sample positive and negative training examples of the category.
- Concept Learning can see as a problem of searching through a predefined space of potential hypotheses for the hypothesis that best fits the training examples.
- The hypothesis space has a general-to-specific ordering of hypotheses, and the search can be efficiently organized by taking advantage of a naturally occurring structure over the hypothesis space.

- A Formal Definition for Concept Learning:

Inferring a Boolean-valued function from training examples of its input and output.

- An example for concept-learning is the learning of bird-concept from the given examples of birds (positive examples) and non-birds (negative examples).
- We are trying to learn the definition of a concept from given examples.

A Concept Learning Task – Enjoy SportTraining Examples

Example	**Sky**	**AirTemp**	**Humidity**	**Wind**	**Water**	**Forecast**	**EnjoySport**
1	Sunny	Warm	Normal	Strong	Warm	Same	**YES**
2	Sunny	Warm	High	Strong	Warm	Same	**YES**
3	Rainy	Cold	High	Strong	Warm	Change	**NO**
4	Sunny	Warm	High	Strong	Warm	Change	**YES**

ATTRIBUTES **CONCEPT**

- A set of example days, and each is described by six attributes.
- The task is to learn to predict the value of Enjoy-Sport for arbitrary day, based on the values of its attribute values.

Enjoy-Sport – Hypothesis Representation

- Each hypothesis consists of a conjunction of constraints on the instance attributes.
- Each hypothesis will be a vector of six constraints, specifying the values of the six attributes
(Sky, Air-Temp, Humidity, Wind, Water, and Forecast).
- Each attribute will be:
? - indicating any value is acceptable for the attribute (don't care) single value – specifying a single required value (ex. Warm) (specific)
0 - indicating no value is acceptable for the attribute (no value)

Hypothesis Representation

- A hypothesis:
Sky AirTemp Humidity Wind Water Forecast
< Sunny, ? , ? , Strong , ? , Same >
- The most general hypothesis – that every day is a positive example
<?, ?, ?, ?, ?, ?>
- The most specific hypothesis – that no day is a positive example
<0, 0, 0, 0, 0, 0>
- EnjoySport concept learning task requires learning the sets of days for which EnjoySport=yes, describing this set by a conjunction of constraints over the instance attributes.

EnjoySport Concept Learning Task

Given

Instances X : set of all possible days, each described by the attributes

- Sky – (values: Sunny, Cloudy, Rainy)

- Air-Temp – (values: Warm, Cold)
- Humidity – (values: Normal, High)
- Wind – (values: Strong, Weak)
- Water – (values: Warm, Cold)
- Forecast – (values: Same, Change)

Target Concept (Function) c : Enjoy-Sport : X □ {0,1}

Hypotheses H : Each hypothesis is described by a conjunction of constraints on the attributes.

Training Examples D : positive and negative examples of the target function

Determine

A hypothesis h in H such that h(x) = c(x) for all x in D.

The Inductive Learning Hypothesis

•Although the learning task is to determine a hypothesis h identical to the target concept cover the entire set of instances X, the only information available about c is its value over the training examples.

Inductive learning algorithms can at best guarantee that the output hypothesis fits the target concept over the training data.

Lacking any further information, our assumption is that the best hypothesis regarding unseen instances is the hypothesis that best fits the observed training data. This is the fundamental assumption of inductive learning.

•The Inductive Learning Hypothesis - Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples.

•Concept learning can be viewed as the task of searching through a large space of hypotheses implicitly defined by the hypothesis representation.

•The goal of this search is to find the hypothesis that best fits the training examples.

•By selecting a hypothesis representation, the designer of the learning algorithm implicitly defines the space of all hypotheses that the program can ever represent and therefore can ever learn.

•Sky has 3 possible values, and other 5 attributes have 2 possible values.

•There are 96 (= 3.2.2.2.2.2) distinct instances in X.

•There are 5120 (=5.4.4.4.4.4) syntactically distinct hypotheses in H.

Two more values for attributes: ? and 0

•Every hypothesis containing one or more 0 symbols represents the empty set of instances; that is, it classifies every instance as negative.

•There are 973 (= 1 + 4.3.3.3.3.3) semantically distinct hypotheses in H.

Only one more value for attributes: ?, and one hypothesis representing empty set of instances.

•Although Enjoy-Sport has small, finite hypothesis space, most learning tasks have much larger (even infinite) hypothesis spaces.

We need efficient search algorithms on the hypothesis spaces.

General-to-Specific Ordering of Hypotheses

Many algorithms for concept learning organize the search through the hypothesis space by relying on a general-to-specific ordering of hypotheses.

By taking advantage of this naturally occurring structure over the hypothesis space, we can design learning algorithms that exhaustively search even infinite hypothesis spaces without explicitly enumerating every hypothesis. Consider two hypotheses

h1 = (Sunny, ?, ?, Strong, ?, ?)

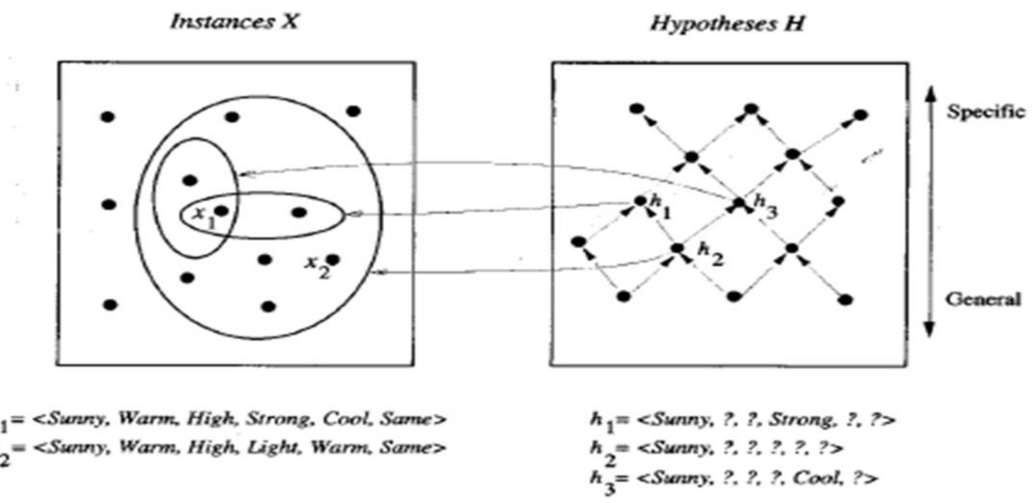
h2 = (Sunny, ?, ?, ?, ?, ?)

Now consider the sets of instances that are classified positive by hl and by h2. Because h2 imposes fewer constraints on the instance, it classifies more instances as positive. In fact, any instance classified positive by hl will also be classified positive by h2. Therefore, we say that h2 is more general than hl.

More-General-Than Relation

For any instance x in X and hypothesis h in H, we say that x satisfies h if and only if h(x) = 1.

More-General-Than-Or-Equal Relation: Let h1 and h2 be two Boolean-valued functions defined over X. Then h1 is more-general-than-or-equal-to h2 (written h1 ≥ h2) if and only if any instance that satisfies h2 also satisfies h1. h1 is more-general-than h2 (h1 > h2) if and only if h1≥h2 is true and h2≥h1 is false. We also say h2 is more-specific-than h1.



•x_1= <Sunny, Warm, High, Strong, Cool, Same>
x_2= <Sunny, Warm, High, Light, Warm, Same>

h_1= <Sunny, ?, ?, Strong, ?, ?>
h_2= <Sunny, ?, ?, ?, ?, ?>
h_3= <Sunny, ?, ?, ?, Cool, ?>

h2 > h1 and   h2 > h3

But there is no more-general relation between h1 and h3

FIND-S Algorithm

FIND-S Algorithm starts from the most specific hypothesis and generalize it by considering only positive examples. FIND-S algorithm ignores negative examples. As long as the hypothesis space contains a hypothesis that describes the true target concept, and the training data contains no errors, ignoring negative examples does not cause to any problem. FIND-S algorithm finds the most specific hypothesis within H that is consistent with the positive training examples. The final hypothesis will also be consistent with negative examples if the correct target concept is in H, and the training examples are correct.

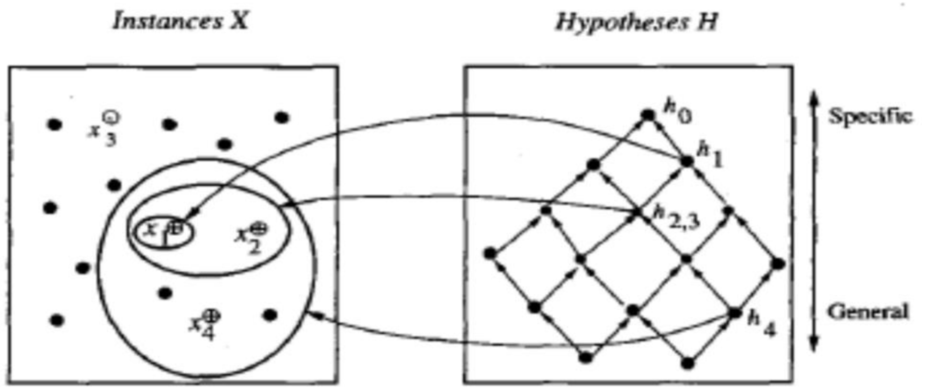
1.Initialize h to the most specific hypothesis in H

2.For each positive training instance x for each attribute constraint a, in h

3.If the constraint a, is satisfied by x Then do nothing

Else replace a, in h by the next more general constraint that is satisfied by x

4. Output hypothesis h



	h_0 = <∅, ∅, ∅, ∅, ∅, ∅>
x_1 = <*Sunny Warm Normal Strong Warm Same*>, +	h_1 = <*Sunny Warm Normal Strong Warm Same*>
x_2 = <*Sunny Warm High Strong Warm Same*>, +	h_2 = <*Sunny Warm ? Strong Warm Same*>
x_3 = <*Rainy Cold High Strong Warm Change*>, -	h_3 = <*Sunny Warm ? Strong Warm Same*>
x_4 = <*Sunny Warm High Strong Cool Change*>, +	h_4 = <*Sunny Warm ? Strong ? ?* >

Unanswered Questions by FIND-S Algorithm

Has FIND-S converged to the correct target concept?

Although FIND-S will find a hypothesis consistent with the training data, it has no way to determine whether it has found the only hypothesis in H consistent with the data (i.e., the correct target concept), or whether there are many other consistent hypotheses as well.

We would prefer a learning algorithm that could determine whether it had converged and, if not, at least characterize its uncertainty regarding the true identity of the target concept.

Why prefer the most specific hypothesis?

In case there are multiple hypotheses consistent with the training examples, FIND-S will find the most specific.

It is unclear whether we should prefer this hypothesis over, say, the most general, or some other hypothesis of intermediate generality.

•Are the training examples consistent?

In most practical learning problems there is some chance that the training examples will contain at least some errors or noise.

Such inconsistent sets of training examples can severely mislead FIND-S, given the fact that it ignores negative examples.

We would prefer an algorithm that could at least detect when the training data is inconsistent and, preferably, accommodate such errors.

•What if there are several maximally specific consistent hypotheses?

In the hypothesis language H for the Enjoy-Sport task, there is always a unique, most specific hypothesis consistent with any set of positive examples.

However, for other hypothesis spaces there can be several maximally specific hypotheses consistent with the data.

In this case, FIND-S must be extended to allow it to backtrack on its choices of how to generalize the hypothesis, to accommodate the possibility that the target concept lies along a different branch of the partial ordering than the branch it has selected.

Candidate-Elimination Algorithm

•FIND-S outputs a hypothesis from H, that is consistent with the training examples, this is just one of many hypotheses from H that might fit the training data equally well.

•The key idea in the Candidate-Elimination algorithm is to output a description of the set of all hypotheses consistent with the training examples.

Candidate-Elimination algorithm computes the description of this set without explicitly enumerating all of its members.

This is accomplished by using the more-general-than partial ordering and maintaining a compact representation of the set of consistent hypotheses.

Consistent hypothesis:

•The key difference between this definition of consistent and satisfies.

•An example x is said to satisfy hypothesis h when h(x) = 1, regardless of whether x is a positive or negative example of the target concept.

•However, whether such an example is consistent with h depends on the target concept, and in particular, whether h(x) = c(x).

A hypothesis h is **consistent** with a set of training examples D of target concept c if and only if $h(x) = c(x)$ for each training example $\langle x, c(x) \rangle$ in D.

$$Consistent(h, D) \equiv (\forall \langle x, c(x) \rangle \in D)\ h(x) = c(x)$$

Version Spaces:

• The Candidate-Elimination algorithm represents the set of

all hypotheses consistent with the observed training examples.

• This subset of all hypotheses is called the version space with respect to the hypothesis space H and the training examples D, because it contains all plausible versions of the target concept.

List-Then-Eliminate Algorithm

The **version space**, $VS_{H,D}$, with respect to hypothesis space H and training examples D, is the subset of hypotheses from H consistent with all training examples in D.

$$VS_{H,D} \equiv \{h \in H | Consistent(h, D)\}$$

•List-Then-Eliminate algorithm initializes the version space to contain all hypotheses in H, then eliminates any hypothesis found inconsistent with any training example.

•The version space of candidate hypotheses thus shrinks as more examples are observed, until ideally just one hypothesis remains that is consistent with all the observed examples.

Presumably, this is the desired target concept.

If insufficient data is available to narrow the version space to a single hypothesis, then the algorithm can output the entire set of hypotheses consistent with the observed data.

•List-Then-Eliminate algorithm can be applied whenever the hypothesis space H is finite.

It has many advantages, including the fact that it is guaranteed to output all hypotheses consistent with the training data.

Unfortunately, it requires exhaustively enumerating all hypotheses in H - an unrealistic requirement for all but the most trivial hypothesis spaces.

```
1. $VersionSpace \leftarrow$ a list containing every
   hypothesis in $H$
2. For each training example, $\langle x, c(x) \rangle$
   remove from $VersionSpace$ any hypothesis $h$ for
   which $h(x) \neq c(x)$
3. Output the list of hypotheses in $VersionSpace$
```

Compact Representation of Version Spaces

•A version space can be represented with its general and specific boundary sets.

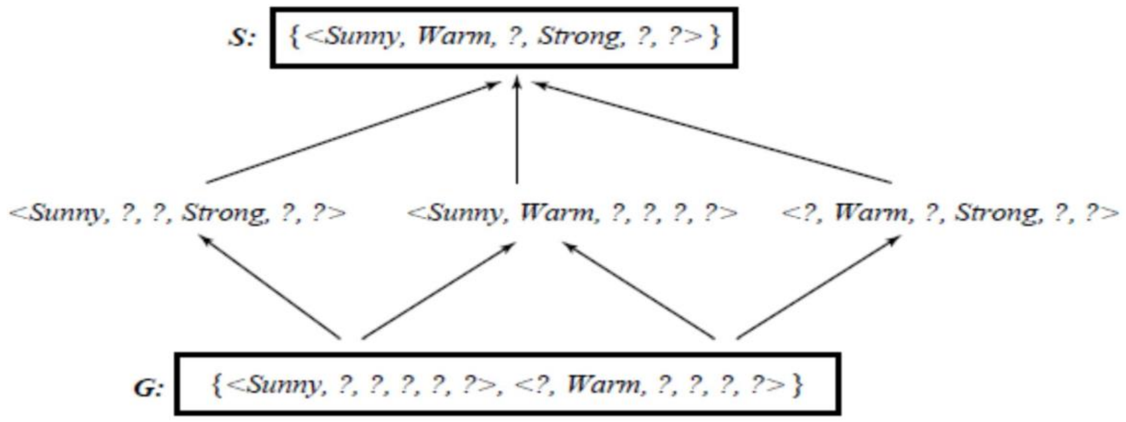
•The Candidate-Elimination algorithm represents the version space by storing only its most general members G and its most specific members S.

•Given only these two sets S and G, it is possible to enumerate all members of a version space by generating hypotheses that lie between these two sets in general-to-specific partial ordering over hypotheses.

•Every member of the version space lies between these boundaries

$$VS_{H,D} = \{h \in H | (\exists s \in S)(\exists g \in G)(g \geq h \geq s)\}$$

where x ≥y means x is more general or equal to y.



• A version space with its general and specific boundary sets.

• The version space includes all six hypotheses shown here, but can be represented more simply by S and G.

Candidate-Elimination Algorithm

• The Candidate-Elimination algorithm computes the version space containing all hypotheses from H that are consistent with an observed sequence of training examples.

• It begins by initializing the version space to the set of all hypotheses in H; that is, by initializing the G boundary set to contain the most general hypothesis in H

G0 ⍰ { <?, ?, ?, ?, ?, ?> }

and initializing the S boundary set to contain the most specific hypothesis S0 ⍰ { <0, 0, 0, 0, 0, 0> }

• These two boundary sets delimit the entire hypothesis space, because every other hypothesis in H is both more general than S0 and more specific than G0.

• As each training example is considered, the S and G boundary sets are generalized and specialized, respectively, to eliminate from the version space any hypotheses found inconsistent with the new training example.

• After all examples have been processed, the computed version space contains all the hypotheses consistent with these examples and only these hypotheses.

• Initialize G to the set of maximally general hypotheses in H

• Initialize S to the set of maximally specific hypotheses in H

• For each training example d, do

– If d is a positive example

• Remove from G any hypothesis inconsistent with d ,

• For each hypothesis s in S that is not consistent with d ,-

Remove s from S

Add to S all minimal generalizations h of s such that h is consistent with d, and some member of G is more general than h

Remove from S any hypothesis that is more general than another hypothesis in S

If d is a negative example

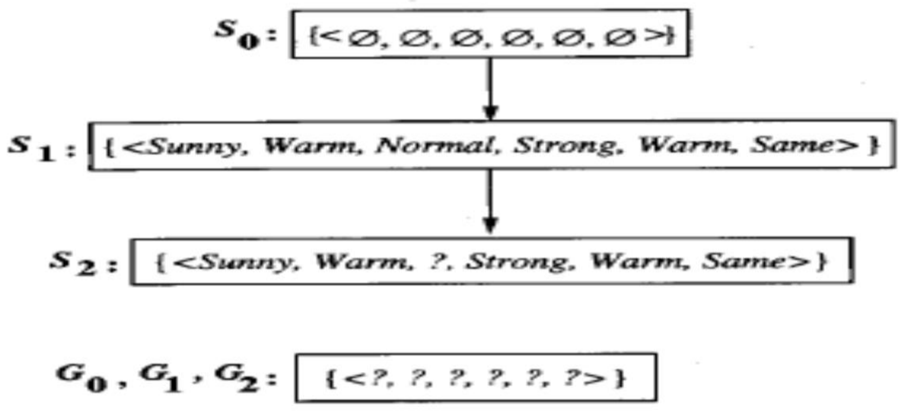
- Remove from S any hypothesis inconsistent with d
- For each hypothesis g in G that is not consistent with d

Remove g from G

Add to G all minimal specializations h of g such that

h is consistent with d, and some member of S is more specific than h

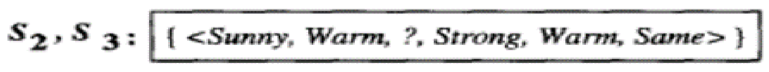
Remove from G any hypothesis that is less general than another hypothesis in G

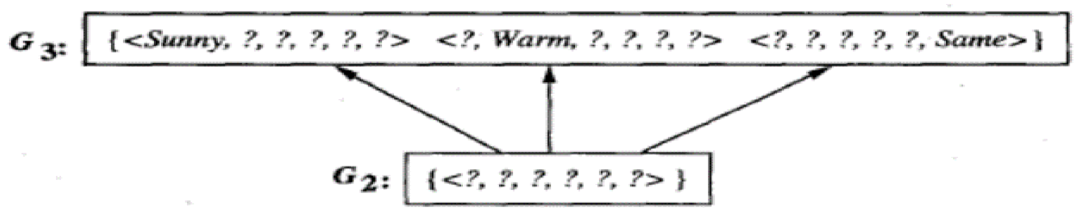


S_0: {<∅, ∅, ∅, ∅, ∅, ∅>}
S_1: {<Sunny, Warm, Normal, Strong, Warm, Same>}
S_2: {<Sunny, Warm, ?, Strong, Warm, Same>}
G_0, G_1, G_2: {<?, ?, ?, ?, ?, ?>}

Training examples:

1. *<Sunny, Warm, Normal, Strong, Warm, Same>, Enjoy Sport = Yes*
2. *<Sunny, Warm, High, Strong, Warm, Same>, Enjoy Sport = Yes*



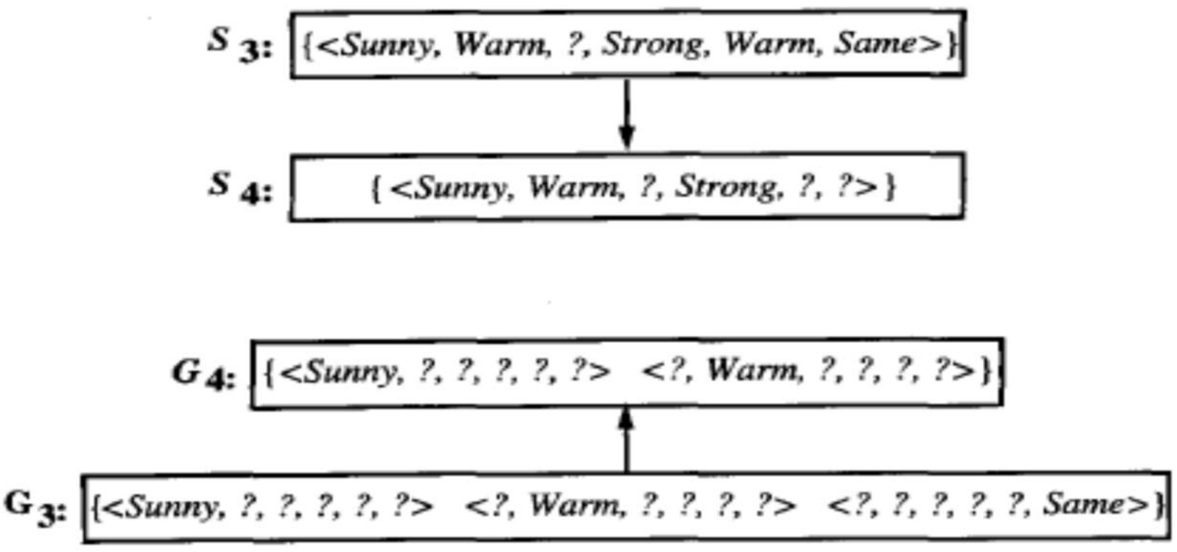


Training Example:

3. *<Rainy, Cold, High, Strong, Warm, Change>, EnjoySport=No*

•Given that there are six attributes that could be specified to specialize G2, why are there only three new hypotheses in G3?

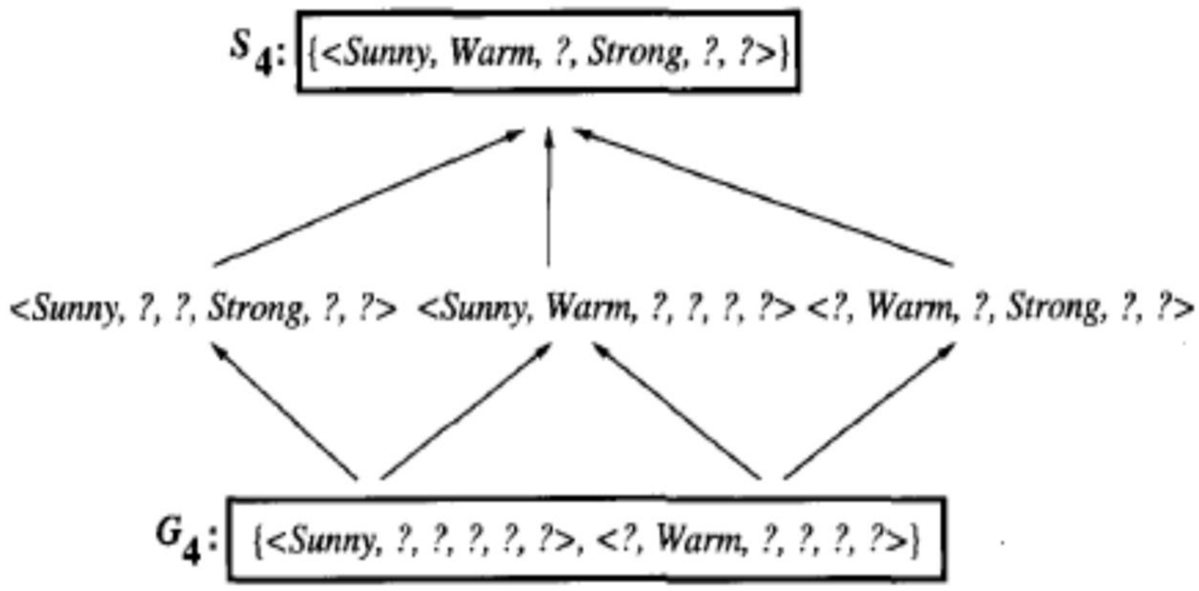
•For example, the hypothesis h = <?, ?, Normal, ?, ?, ?> is a minimal specialization of G2 that correctly labels the new example as a negative example, but it is not included in G3. The reason this hypothesis is excluded is that it is inconsistent with S2. The algorithm determines this simply by noting that h is not more general than the current specific boundary, S2. In fact, the S boundary of the version space forms a summary of the previously encountered positive examples that can be used to determine whether any given hypothesis is consistent with these examples. The G boundary summarizes the information from previously encountered negative examples. Any hypothesis more specific than G is assured to be consistent with past negative examples



Training Example:

4. *<Sunny, Warm, High, Strong, Cool, Change>, EnjoySport = Yes*

•The fourth training example further generalizes the S boundary of the version space. It also results in removing one member of the G boundary, because this member fails to cover the new positive example. To understand the rationale for this step, it is useful to consider why the offending hypothesis must be removed from G. Notice it cannot be specialized, because specializing it would not make it cover the new example. It also cannot be generalized, because by the definition of G, any more general hypothesis will cover at least one negative training example. Therefore, the hypothesis must be dropped from the G boundary, thereby removing an entire branch of the partial ordering from the version space of hypotheses remaining under consideration



• After processing these four examples, the boundary sets S4 and G4 delimit the version space of all hypotheses consistent with the set of incrementally observed training examples.

• This learned version space is independent of the sequence in which the training examples are presented (because in the end it contains all hypotheses consistent with the set of examples).

• As further training data is encountered, the S and G boundaries will move monotonically closer to each other, delimiting a smaller and smaller version space of candidate hypotheses.

Will Candidate-Elimination Algorithm Converge to Correct Hypothesis?

•The version space learned by the Candidate-Elimination Algorithm will converge toward the hypothesis that correctly describes the target concept, provided There are no errors in the training examples, and there is some hypothesis in H that correctly describes the target concept.

•What will happen if the training data contains errors?

The algorithm removes the correct target concept from the version space. S and G boundary sets eventually converge to an empty version space if sufficient additional training data is available. Such an empty version space indicates that there is no hypothesis in H consistent with all observed training examples. A similar symptom will appear when the training examples are correct, but the target concept cannot be described in the hypothesis representation. e.g., if the target concept is a disjunction of feature attributes and the hypothesis space supports only conjunctive descriptions

•We have assumed that training examples are provided to the learner by some external teacher.

•Suppose instead that the learner is allowed to conduct experiments in which it chooses the next instance, then obtains the correct classification for this instance from an external oracle (e.g., nature or a teacher).

This scenario covers situations in which the learner may conduct experiments in nature or in which a teacher is available to provide the correct classification. We use the term query to refer to such instances constructed by the learner, which are then classified by an external oracle. Considering the version space learned from the four training examples of the Enjoy-Sport concept.

What would be a good query for the learner to pose at this point?

What is a good query strategy in general?

The learner should attempt to discriminate among the alternative competing hypotheses in its current version space.

Therefore, it should choose an instance that would be classified positive by some of these hypotheses, but negative by others.

One such instance is <Sunny, Warm, Normal, Light, Warm, Same>

This instance satisfies three of the six hypotheses in the current version space.

If the trainer classifies this instance as a positive example, the S boundary of the version space can then be generalized.

Alternatively, if the trainer indicates that this is a negative example, the G boundary can then be specialized.

In general, the optimal query strategy for a concept learner is to generate instances that satisfy exactly half the hypotheses in the current version space.

When this is possible, the size of the version space is reduced by half with each new example, and the correct target concept can therefore be found with only log2 |VS| experiments.

How Can Partially Learned Concepts Be Used?

•The version space learned by the Candidate-Elimination Algorithm will converge toward the hypothesis that correctly describes the target concept provided. There are no errors in the training examples, and there is some hypothesis in H that correctly describes the target concept.

•What will happen if the training data contains errors?

The algorithm removes the correct target concept from the version space. S and G boundary sets eventually converge to an empty version space if sufficient additional training data is available. Such an empty version space indicates that there is no hypothesis in H consistent with all observed training examples.

•A similar symptom will appear when the training examples are correct, but the target concept cannot be described in the hypothesis representation. e.g., if the target concept is a disjunction of feature attributes and the hypothesis space supports only conjunctive descriptions

We have assumed that training examples are provided to the learner by some external teacher.

Suppose instead that the learner is allowed to conduct experiments in which it chooses the next instance, then obtains the correct classification for this instance from an external oracle (e.g., nature or a teacher).

This scenario covers situations in which the learner may conduct experiments in nature or in which a teacher is available to provide the correct classification.

We use the term query to refer to such instances constructed by the learner, which are then classified by an external oracle.

•Considering the version space learned from the four training examples of the Enjoy-Sport concept.

What would be a good query for the learner to pose at this point?

What is a good query strategy in general?

•The learner should attempt to discriminate among the alternative competing hypotheses in its current version space. Therefore, it should choose an instance that would be classified positive by some of these hypotheses, but negative by others.

One such instance is <Sunny, Warm, Normal, Light, Warm, Same>

This instance satisfies three of the six hypotheses in the current version space. If the trainer classifies this instance as a positive example, the S boundary of the version space can then be generalized.

Alternatively, if the trainer indicates that this is a negative example, the G boundary can then be specialized. In general, the optimal query strategy for a concept learner is to generate instances that satisfy exactly half the hypotheses in the current version space. When this is possible, the size of the version space is reduced by half with each new example, and the correct target concept can therefore be found with only ⌈log2 |VS| ⌉ experiments.

How Can Partially Learned Concepts Be Used?

•Even though the learned version space still contains multiple hypotheses, indicating that the target concept has not yet been fully learned, it is possible to classify certain examples with the same degree of confidence as if the target concept had been uniquely identified.

•Let us assume that the followings are new instances to be classified:

Instance	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
A	Sunny	Warm	Normal	Strong	Cool	Change	?
B	Rainy	Cold	Normal	Light	Warm	Same	?
C	Sunny	Warm	Normal	Light	Warm	Same	?
D	Sunny	Cold	Normal	Strong	Warm	Same	?

Instance A was is classified as a positive instance by every hypothesis in the current version space.

Because the hypotheses in the version space unanimously agree that this is a positive instance, the learner can classify instance A as positive with the same confidence it would have if it had already converged to the single, correct target concept. Regardless of which hypothesis in the version space is eventually found to be the correct target concept, it is already clear that it will classify instance A as a positive example. Notice furthermore that we need not enumerate every hypothesis in the version space in order to test whether each classifies the instance as positive.

This condition will be met if and only if the instance satisfies every member of S. The reason is that every other hypothesis in the version space is at least as general as some member of S. By our definition of more-general-than, if the new instance satisfies all members of S it must also satisfy each of these more general hypotheses.

Instance B is classified as a negative instance by every hypothesis in the version space. This instance can therefore be safely classified as negative, given the partially learned concept. An efficient test for this condition is that the instance satisfies none of the members of G. Half of the version space hypotheses classify instance C as positive and half classify it as negative. Thus, the learner cannot classify this example with confidence until further training examples are available. Instance D is classified as positive by two of the version space hypotheses and negative by the other four hypotheses. In this case we have less confidence in the classification than in the unambiguous cases of instances A and B.

Still, the vote is in favour of a negative classification, and one approach we could take would be to output the majority vote, perhaps with a confidence rating indicating how close the vote was.

•The Candidate-Elimination Algorithm will converge toward the true target concept provided it is given accurate training examples and provided its initial hypothesis space contains the target concept.

•What if the target concept is not contained in the hypothesis space?

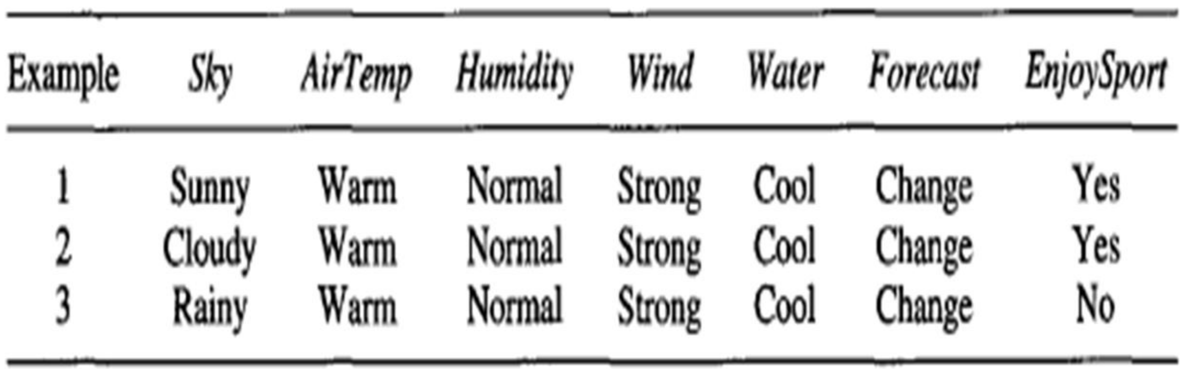
•Can we avoid this difficulty by using a hypothesis space that includes every possible hypothesis?

•How does the size of this hypothesis space influence the ability of the algorithm to generalize to unobserved instances?

•How does the size of the hypothesis space influence the number of training examples that must be observed?

Inductive Bias - A Biased Hypothesis Space

In Enjoy-Sport example, we restricted the hypothesis space to include only conjunctions of attribute values. Because of this restriction, the hypothesis space is unable to represent even simple disjunctive target concepts such as "Sky = Sunny or Sky = Cloudy."



Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Cool	Change	Yes
2	Cloudy	Warm	Normal	Strong	Cool	Change	Yes
3	Rainy	Warm	Normal	Strong	Cool	Change	No

• From first two examples ⮚ S2 : <?, Warm, Normal, Strong, Cool, Change>

• This is inconsistent with third examples, and there are no hypotheses consistent with these three examples

PROBLEM: We have biased the learner to consider only conjunctive hypotheses.

 We require a more expressive hypothesis space.

Inductive Bias - An Unbiased Learner

The obvious solution to the problem of assuring that the target concept is in the hypothesis space H is to provide a hypothesis space capable of representing every teachable concept.

Every possible subset of the instances X the power set of X.

What is the size of the hypothesis space H (the power set of X) ?

In EnjoySport, the size of the instance space X is 96.

The size of the power set of X is 2|X| The size of H is 296

Our conjunctive hypothesis space is able to represent only 973of these hypotheses. A very biased hypothesis space

Inductive Bias - An Unbiased Learner : Problem

•Let the hypothesis space H to be the power set of X. A hypothesis can be represented with disjunctions, conjunctions, and negations of our earlier hypotheses.

The target concept "Sky = Sunny or Sky = Cloudy" could then be described as

<Sunny, ?, ?, ?, ?, ?> <Cloudy, ?, ?, ?, ?, ?>

NEW PROBLEM: our concept learning algorithm is now completely unable to generalize beyond the observed examples.

three positive examples (xl,x2,x3) and two negative examples (x4,x5) to the learner.

S : { x1 x2 x3 } and G : { (x4 x5) } NO GENERALIZATION

Therefore, the only examples that will be unambiguously classified by S and G are the observed training examples themselves.

Inductive Bias –

Fundamental Property of Inductive Inference

•A learner that makes no a priori assumptions regarding the identity of the target concept has no rational basis for classifying any unseen instances.

•Inductive Leap: A learner should be able to generalize training data using prior assumptions in order to classify unseen instances.

•The generalization is known as inductive leap and our prior assumptions are the inductive bias of the learner.

•Inductive Bias (prior assumptions) of Candidate-Elimination Algorithm is that the target concept can be represented by a conjunction of attribute values, the target concept is contained in the hypothesis space and training examples are correct.

Inductive Bias – Formal Definition

Inductive Bias:

Consider a concept learning algorithm L for the set of instances X.

Let c be an arbitrary concept defined over X, and

let Dc = {<x , c(x)>} be an arbitrary set of training examples of c.

Let L(xi, Dc) denote the classification assigned to the instance xi by L

after training on the data Dc.

The inductive bias of L is any minimal set of assertions B such that for any target concept c and corresponding training examples Dc the following formula holds.

$$(\forall x_i \in X)[(B \wedge D_c \wedge x_i) \vdash L(x_i, D_c)]$$

Inductive Bias – Three Learning Algorithms

ROTE-LEARNER: Learning corresponds simply to storing each observed training example in memory. Subsequent instances are classified by looking them up in memory. If the instance is found in memory, the stored classification is returned. Otherwise, the system refuses to classify the new instance.

Inductive Bias: No inductive bias

CANDIDATE-ELIMINATION: New instances are classified only in the case where all members of the current version space agree on the classification. Otherwise, the system refuses to classify the new instance.

Inductive Bias: the target concept can be represented in its hypothesis space.

FIND-S: This algorithm, described earlier, finds the most specific hypothesis consistent with the training examples. It then uses this hypothesis to classify all subsequent instances.

Inductive Bias: the target concept can be represented in its hypothesis space, and all instances are negative instances unless the opposite is entailed by its other know1edge.

Concept Learning – Summary

•Concept learning can be seen as a problem of searching through a large predefined space of potential hypotheses.

•The general-to-specific partial ordering of hypotheses provides a useful structure for organizing the search through the hypothesis space.

•The FIND-S algorithm utilizes this general-to-specific ordering, performing a specific-to-general search through the hypothesis space along one branch of the partial ordering, to find the most specific hypothesis consistent with the training examples.

•The CANDIDATE-ELIMINATION algorithm utilizes this general-to- specific ordering to compute the version space (the set of all hypotheses consistent with the training data) by incrementally computing the sets of maximally specific (S) and maximally general (G) hypotheses.

• Because the S and G sets delimit the entire set of hypotheses consistent with the data, they provide the learner with a description of its uncertainty regarding the exact identity of the target concept. This version space of alternative hypotheses can be examined

To determine whether the learner has converged to the target concept,

To determine when the training data are inconsistent,

To generate informative queries to further refine the version space, and

To determine which unseen instances can be unambiguously classified based on the partially learned concept.

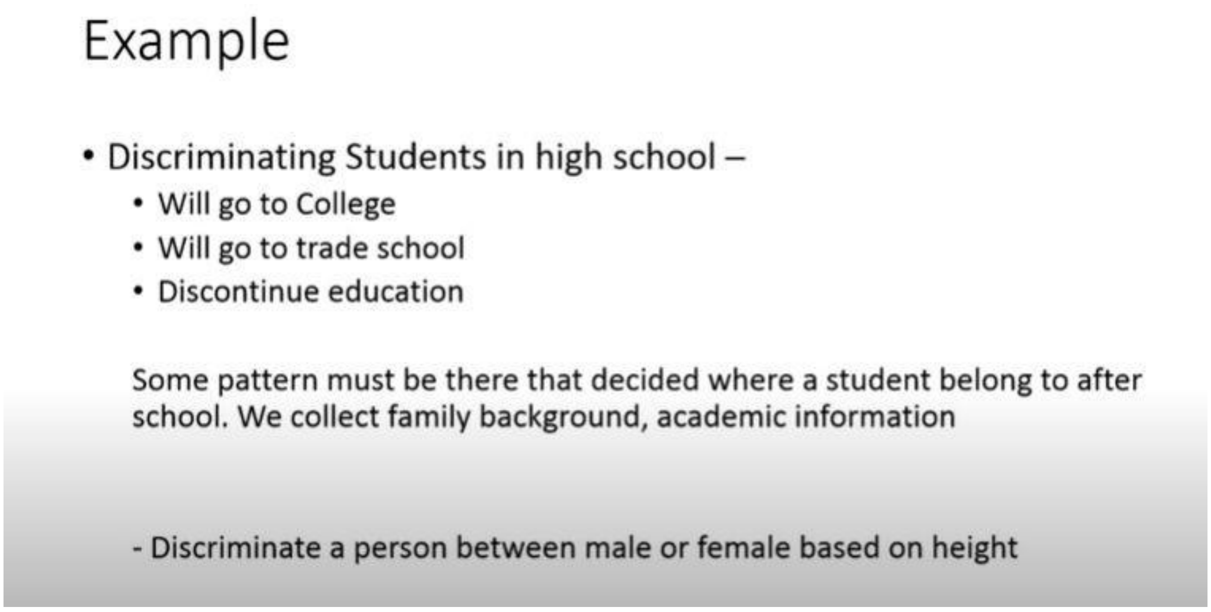
The CANDIDATE-ELIMINATION algorithm is not robust to noisy data or to situations in which the unknown target concept is not expressible in the provided hypothesis space.

Inductive learning algorithms are able to classify unseen examples only because of their implicit inductive bias for selecting one consistent hypothesis over another.

If the hypothesis space is enriched to the point where there is a hypothesis corresponding to every possible subset of instances (the power set of the instances), this will remove any inductive bias from the CANDIDATE-ELIMINATION algorithm .

Unfortunately, this also removes the ability to classify any instance beyond the observed training examples. An unbiased learner cannot make inductive leaps to classify unseen examples.

Linear Discriminant analysis

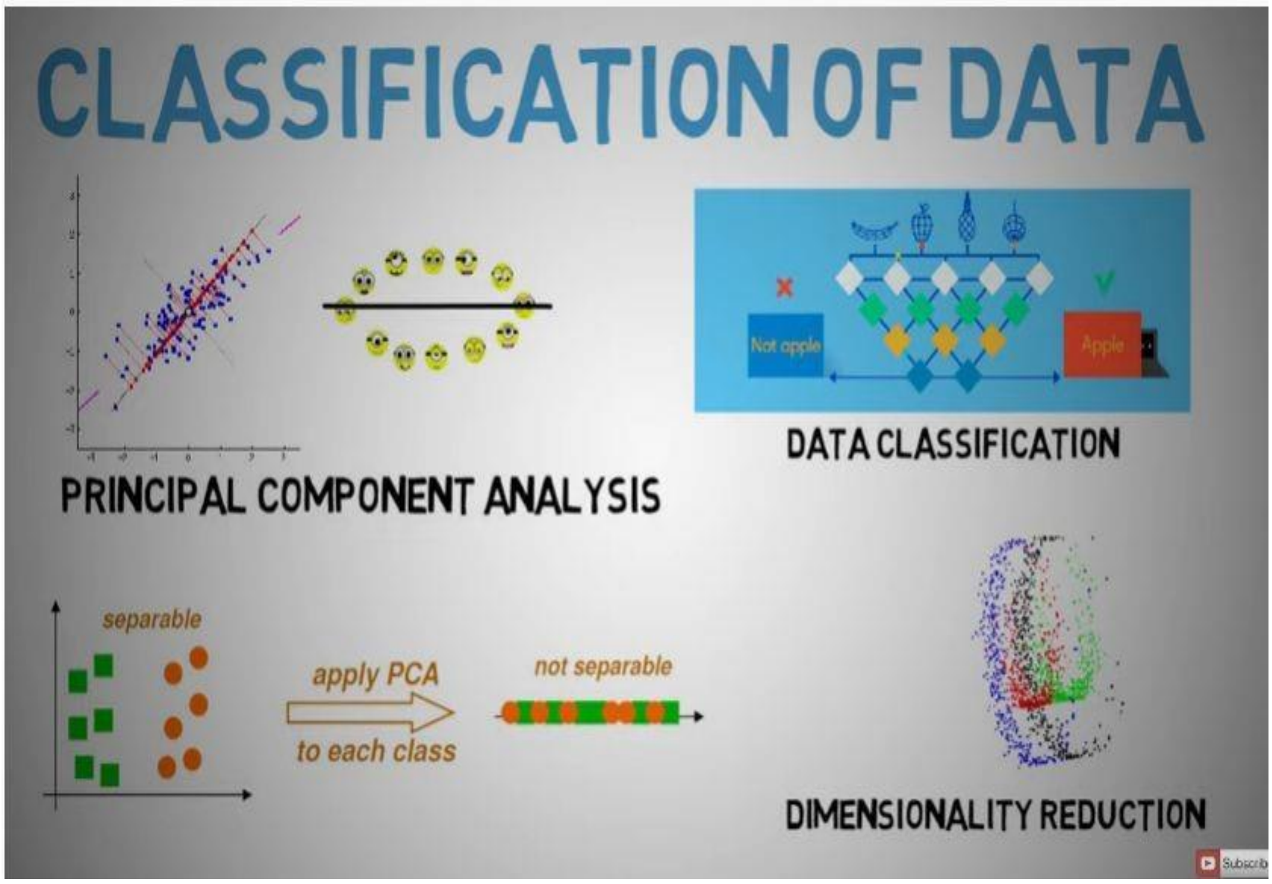


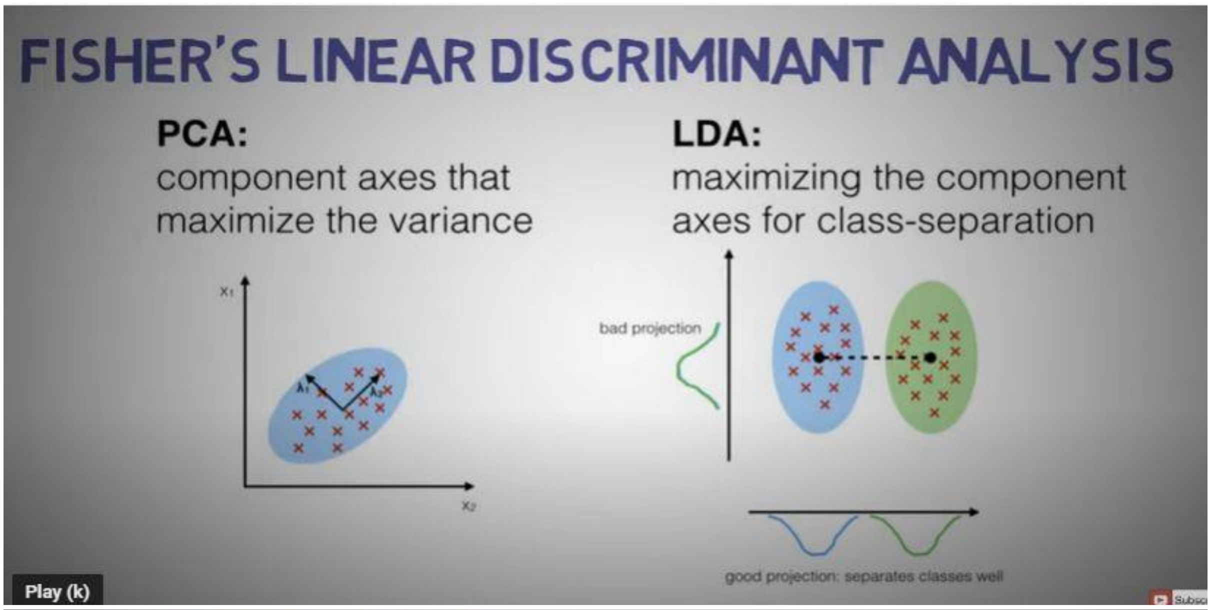
Example

- Discriminating Students in high school –
  - Will go to College
  - Will go to trade school
  - Discontinue education

Some pattern must be there that decided where a student belong to after school. We collect family background, academic information

- Discriminate a person between male or female based on height

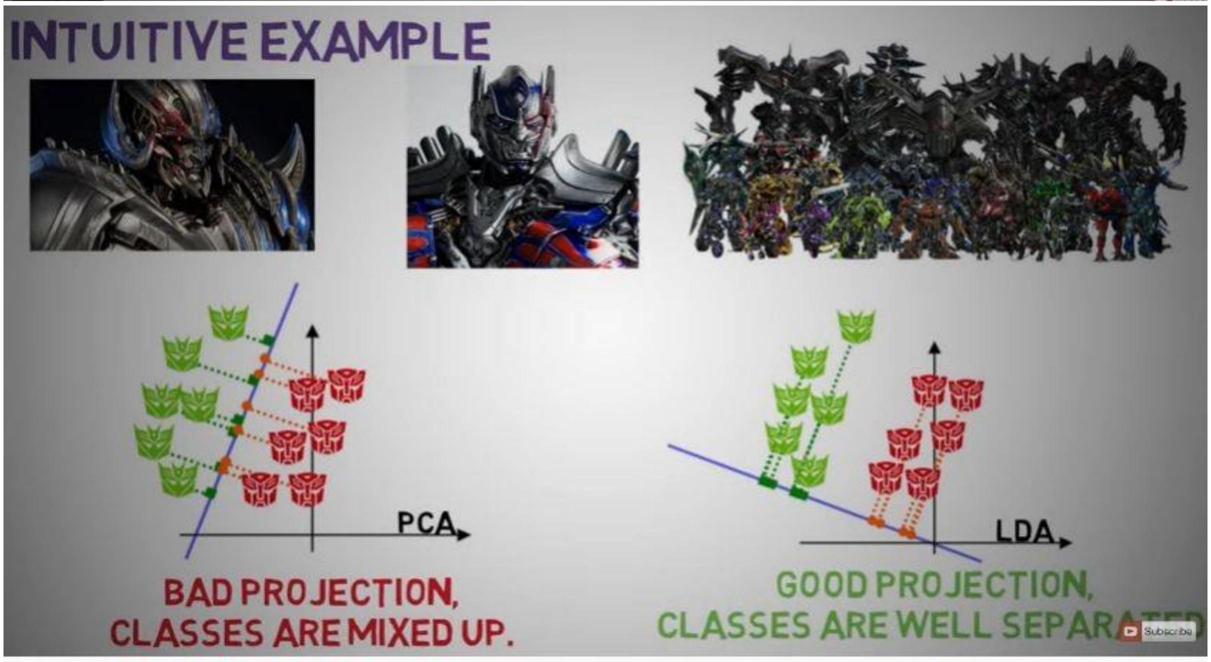




FISHER'S LINEAR DISCRIMINANT ANALYSIS

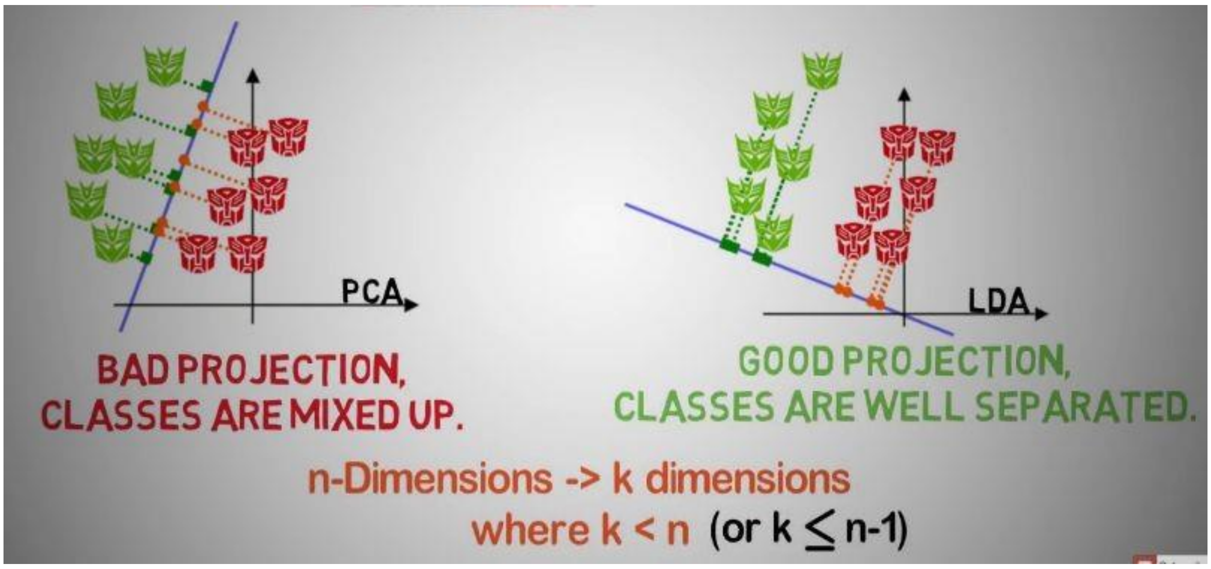
PCA:
component axes that maximize the variance

LDA:
maximizing the component axes for class-separation



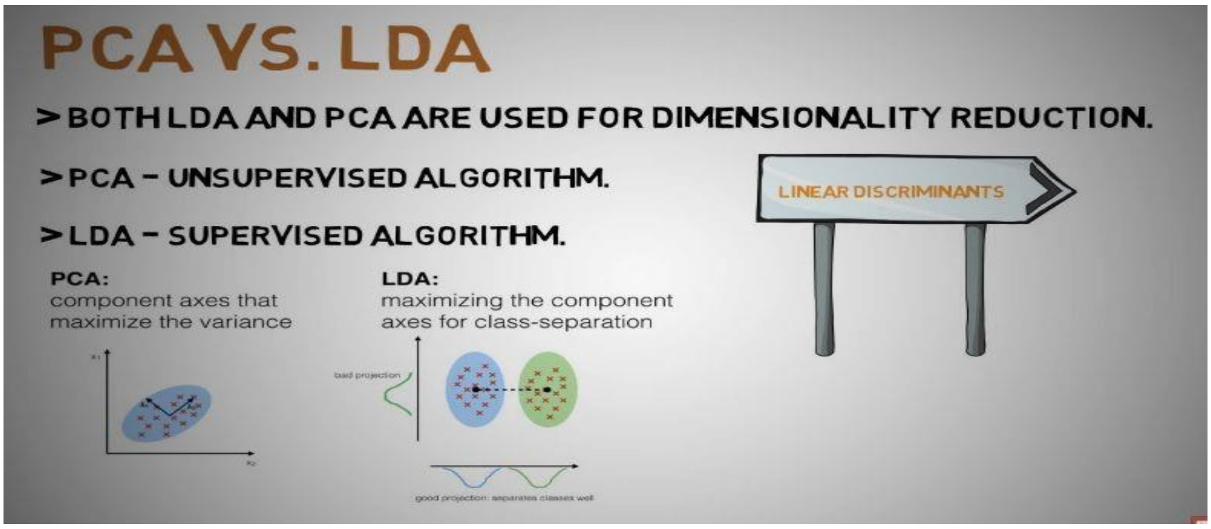
INTUITIVE EXAMPLE

PCA
LDA
BAD PROJECTION, CLASSES ARE MIXED UP.
GOOD PROJECTION, CLASSES ARE WELL SEPARA



PCA
LDA
BAD PROJECTION, CLASSES ARE MIXED UP.
GOOD PROJECTION, CLASSES ARE WELL SEPARATED.

n-Dimensions -> k dimensions
where $k < n$ (or $k \leq n-1$)



PCA VS. LDA

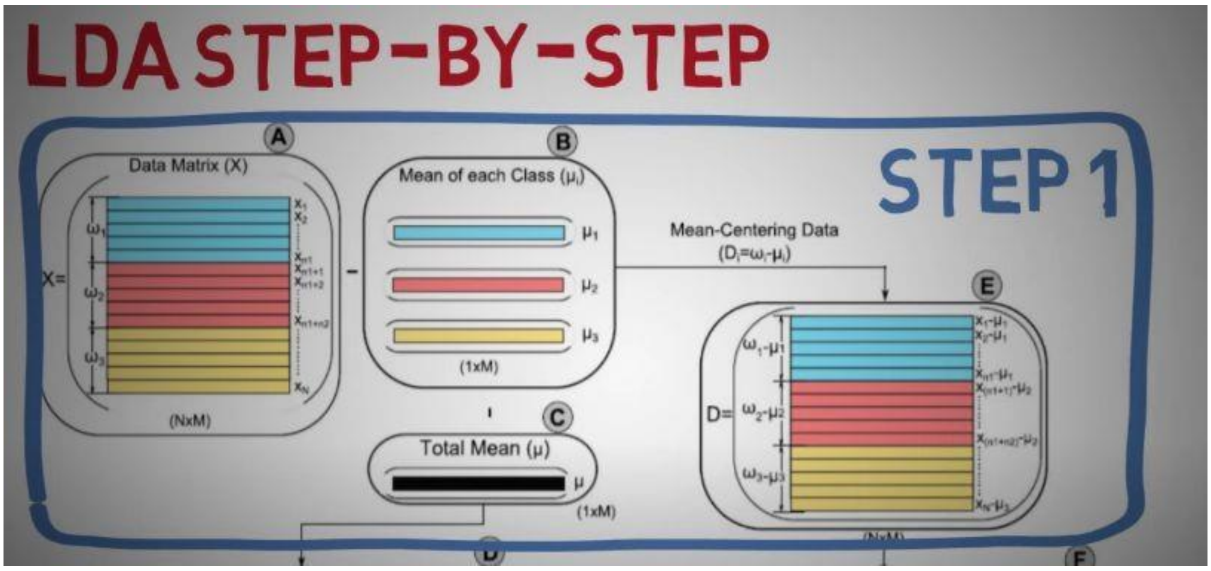
> BOTH LDA AND PCA ARE USED FOR DIMENSIONALITY REDUCTION.

> PCA – UNSUPERVISED ALGORITHM.

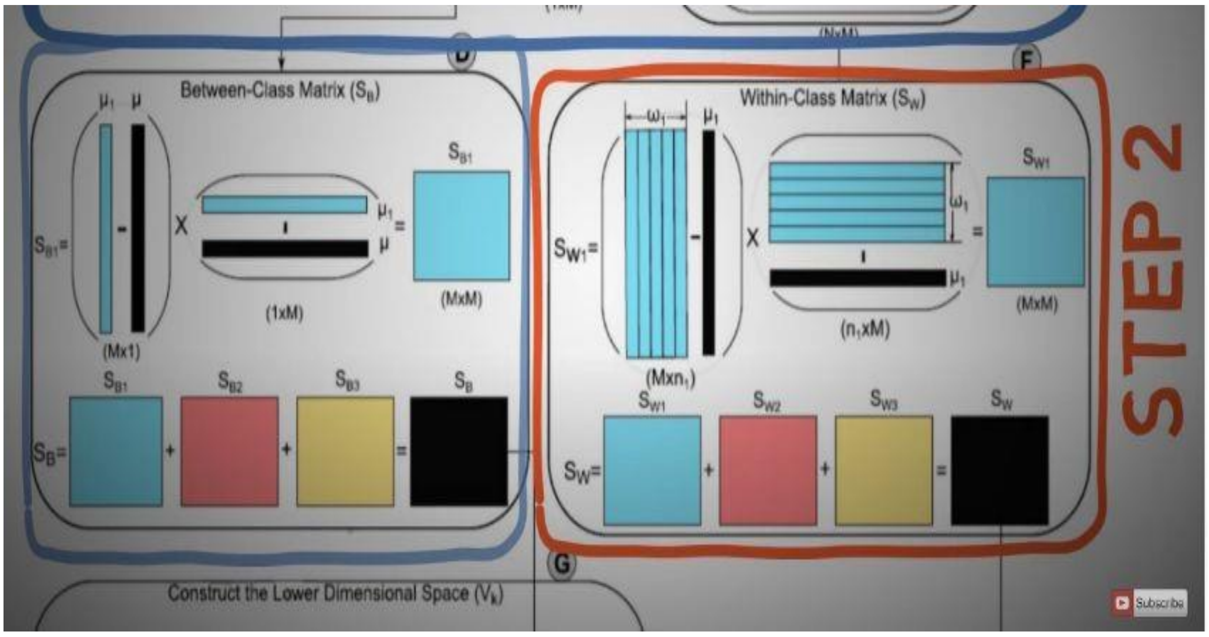
> LDA – SUPERVISED ALGORITHM.

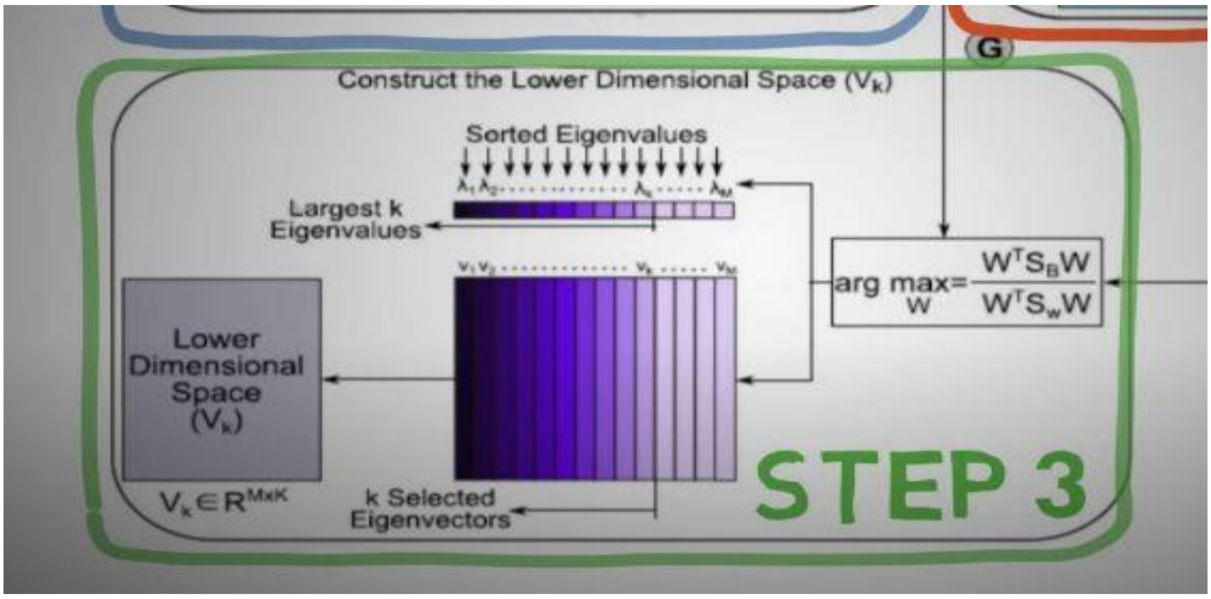
PCA:
component axes that maximize the variance

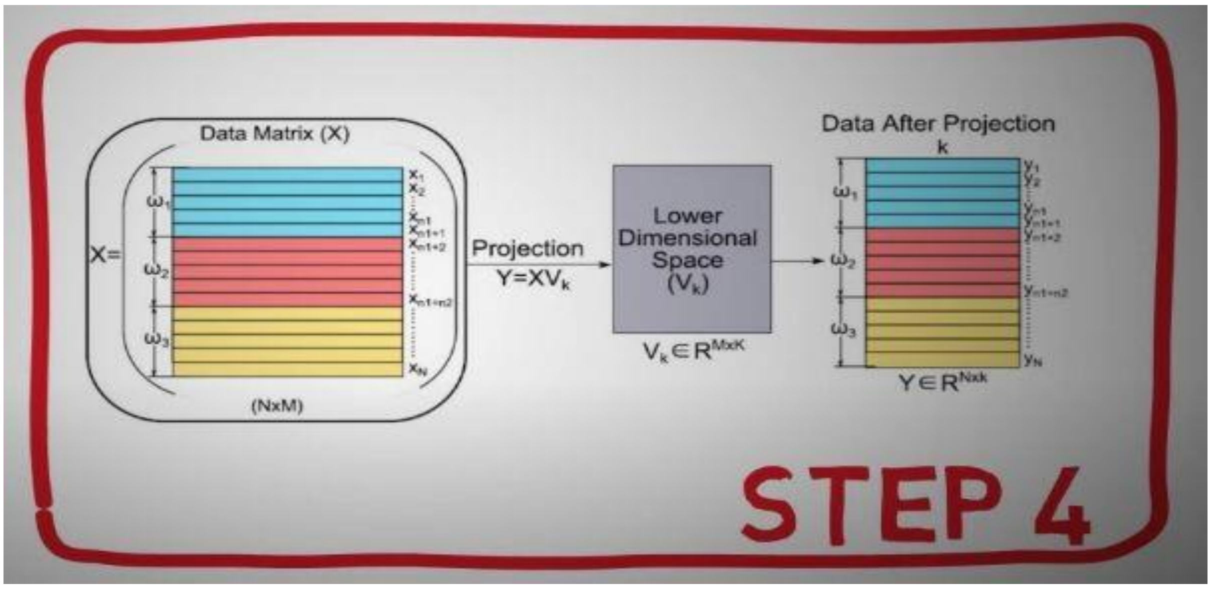
LDA:
maximizing the component axes for class-separation

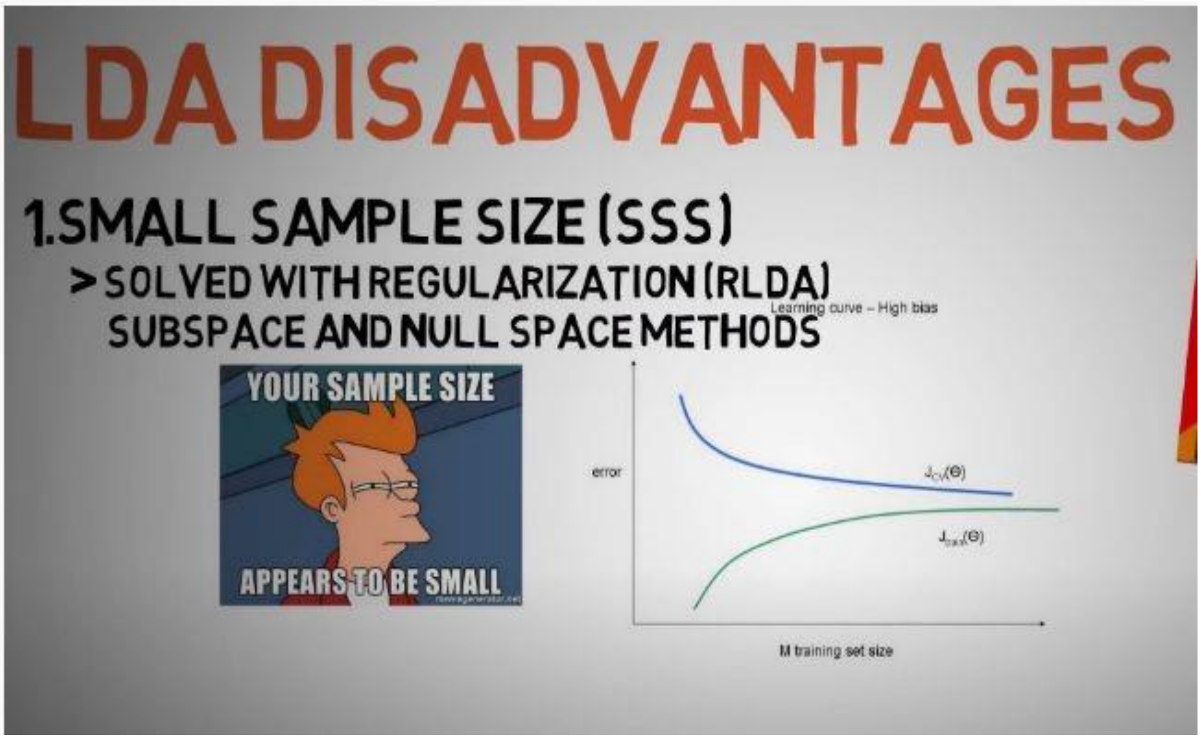


LDA STEP-BY-STEP




Construct the Lower Dimensional Space (V_k)
Sorted Eigenvalues
$\lambda_1 \lambda_2 \cdots \lambda_k \cdots \lambda_M$
Largest k Eigenvalues
$v_1 v_2 \cdots v_k \cdots v_M$
$\arg\max_W = \frac{W^T S_B W}{W^T S_W W}$
Lower Dimensional Space (V_k)
$V_k \in R^{MxK}$
k Selected Eigenvectors
STEP 3
G


Data Matrix (X)
X=
ω_1
ω_2
ω_3
x_1
x_2
x_{n1}
x_{n1+1}
x_{n1+2}
x_{n1+n2}
x_N
(NxM)
Projection
$Y=XV_k$
Lower Dimensional Space (V_k)
$V_k \in R^{MxK}$
Data After Projection
k
y_1
y_2
y_{n1}
y_{n1+1}
y_{n1+2}
y_{n1+n2}
y_N
$Y \in R^{Nxk}$
STEP 4

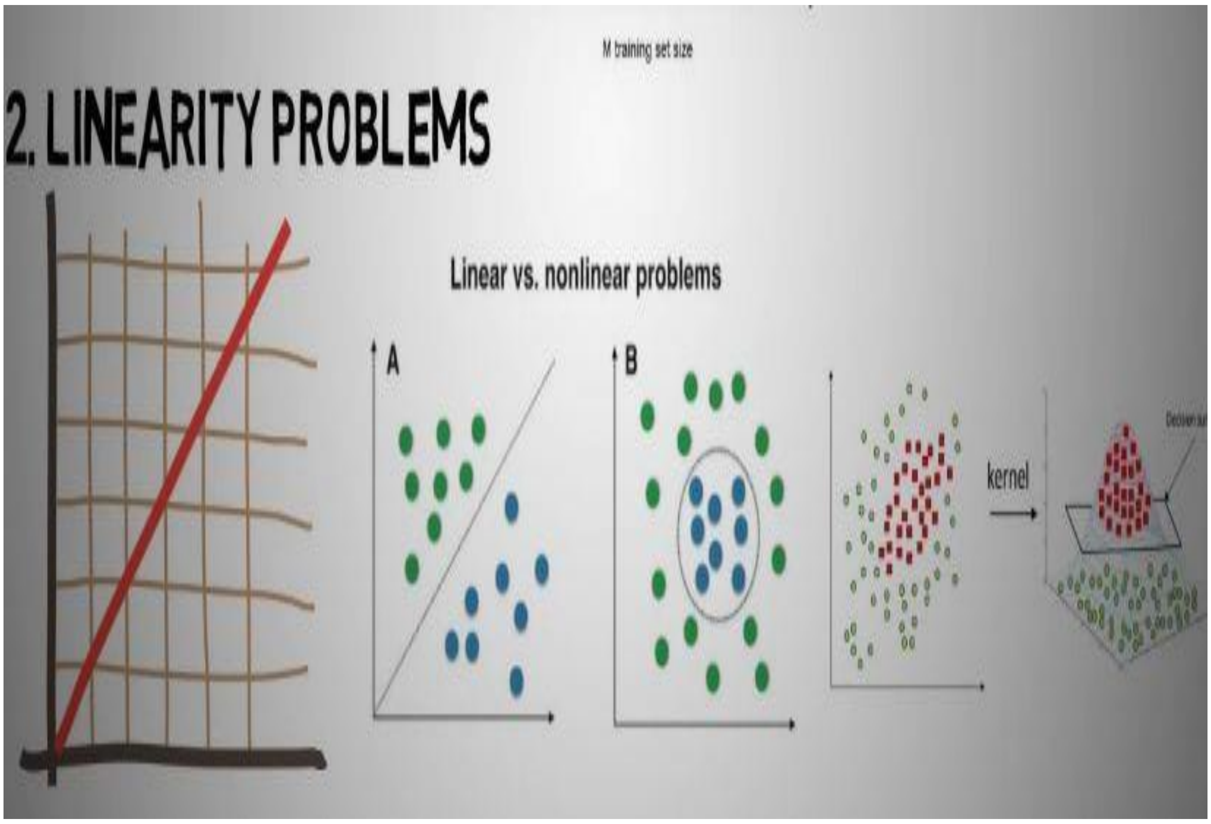


LDA DISADVANTAGES

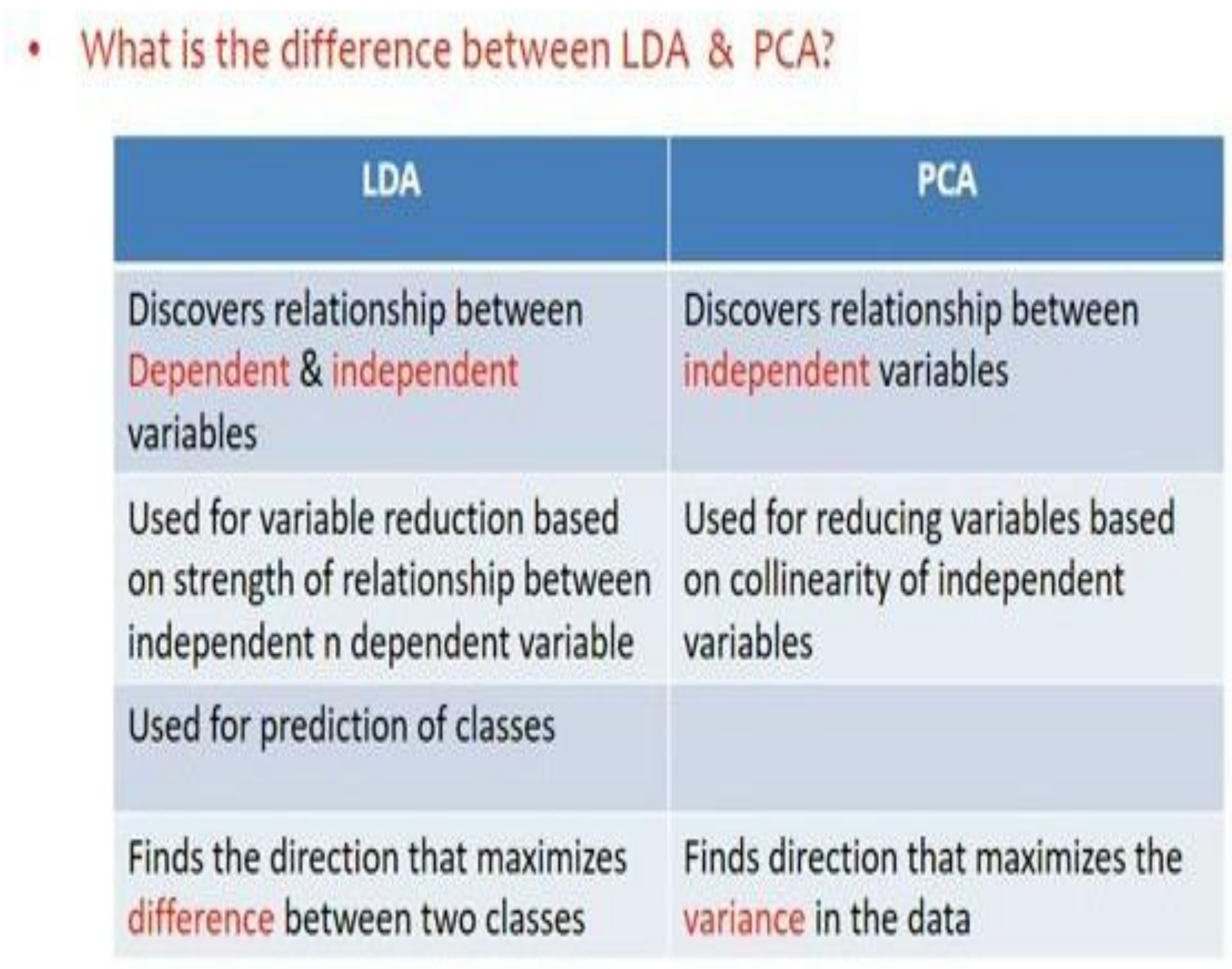
1.SMALL SAMPLE SIZE (SSS)

> SOLVED WITH REGULARIZATION (RLDA)
SUBSPACE AND NULL SPACE METHODS

YOUR SAMPLE SIZE
APPEARS TO BE SMALL
Learning curve – High bias
error
$J_{CV}(\Theta)$
$J_{train}(\Theta)$
M training set size



2. LINEARITY PROBLEMS

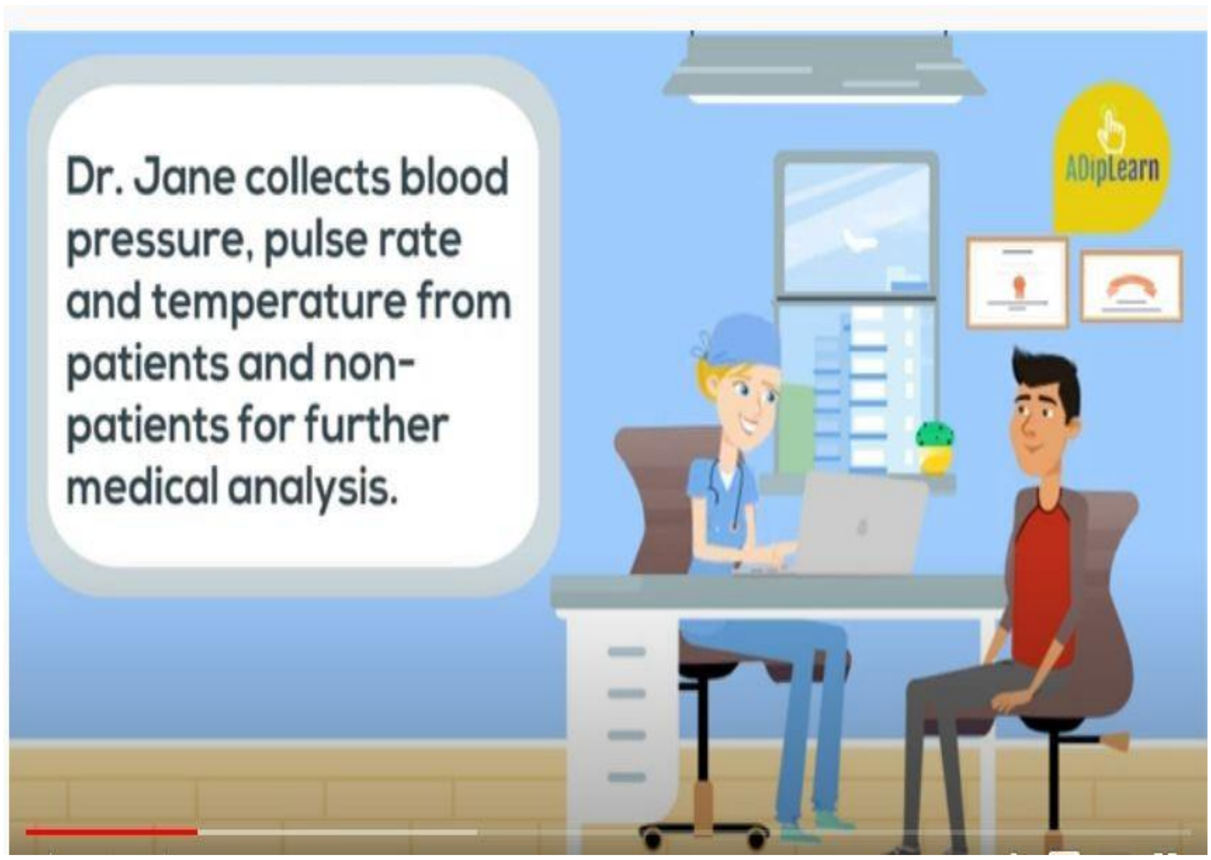


- What is the difference between LDA & PCA?

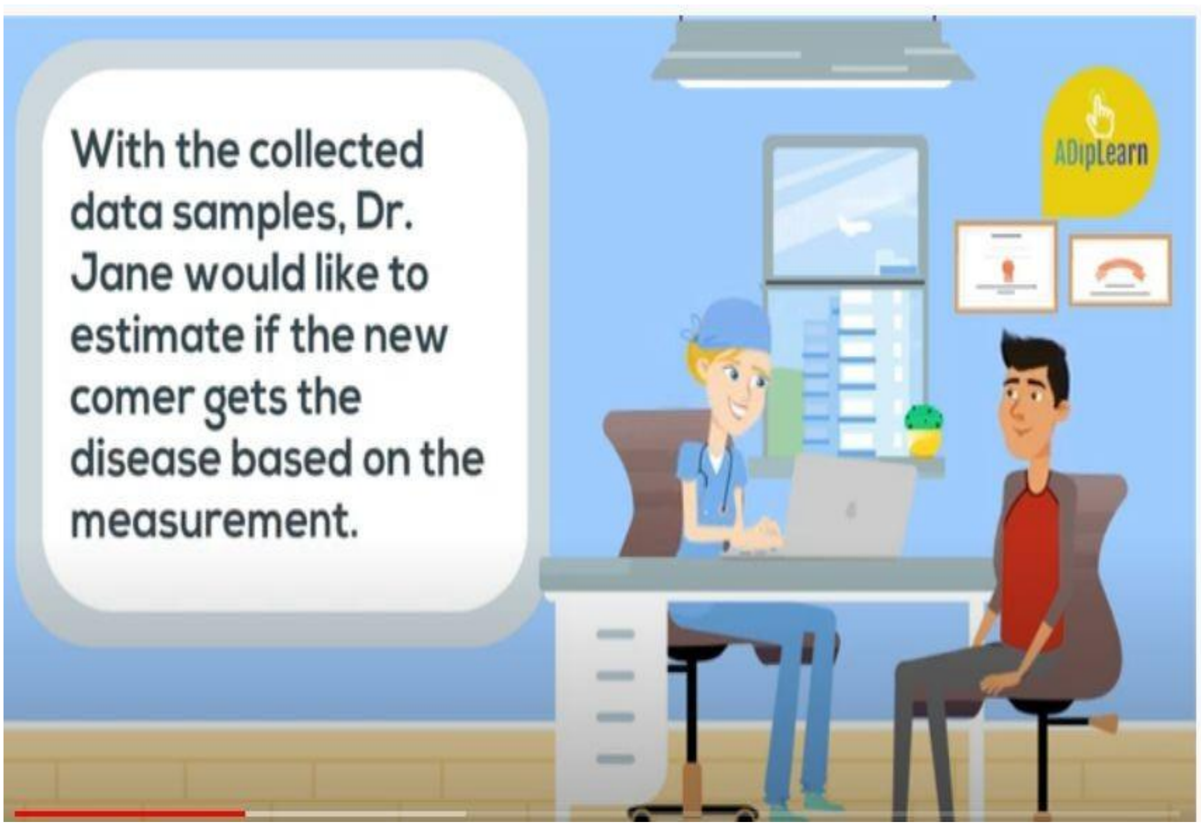
LDA	PCA
Discovers relationship between Dependent & independent variables	Discovers relationship between independent variables
Used for variable reduction based on strength of relationship between independent n dependent variable	Used for reducing variables based on collinearity of independent variables
Used for prediction of classes	
Finds the direction that maximizes difference between two classes	Finds direction that maximizes the variance in the data

HOSPITAL
The symptoms are high blood pressure, pulse rate and temperature.

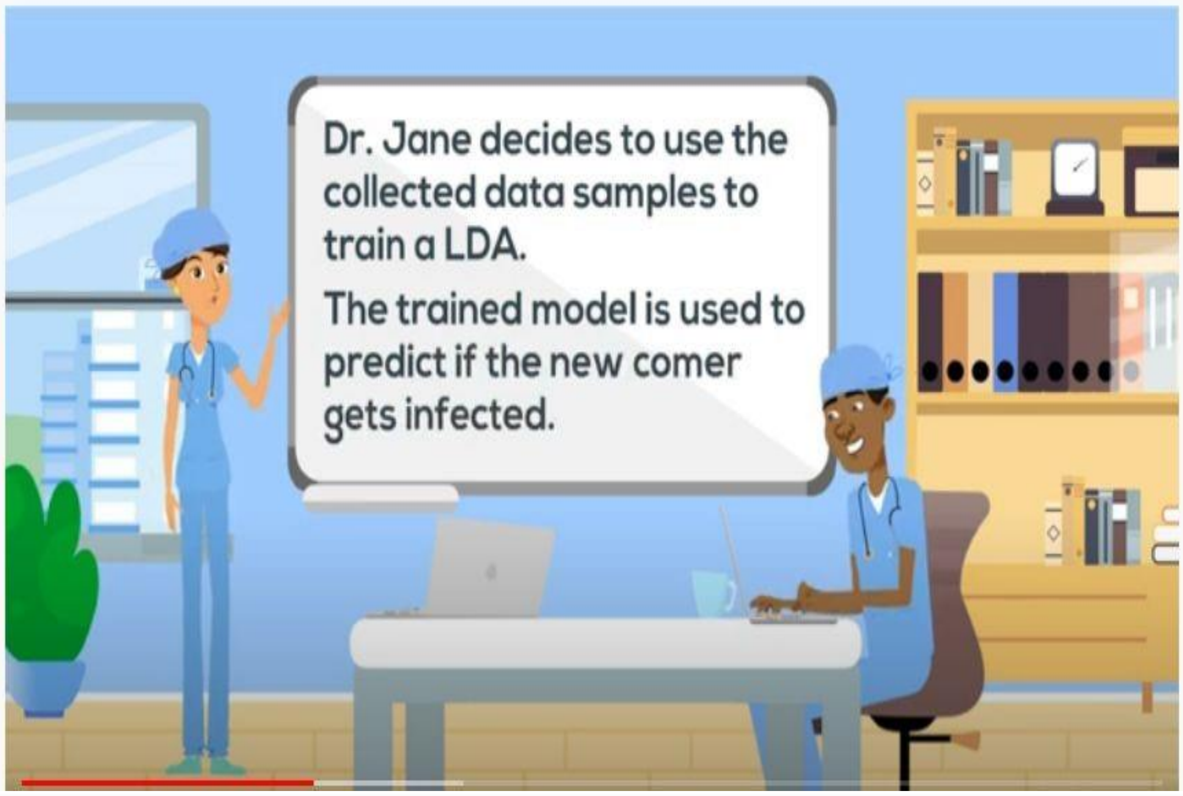
EMERGENCY
More than 100 patients were sent to the emergency room in the last 2 weeks.



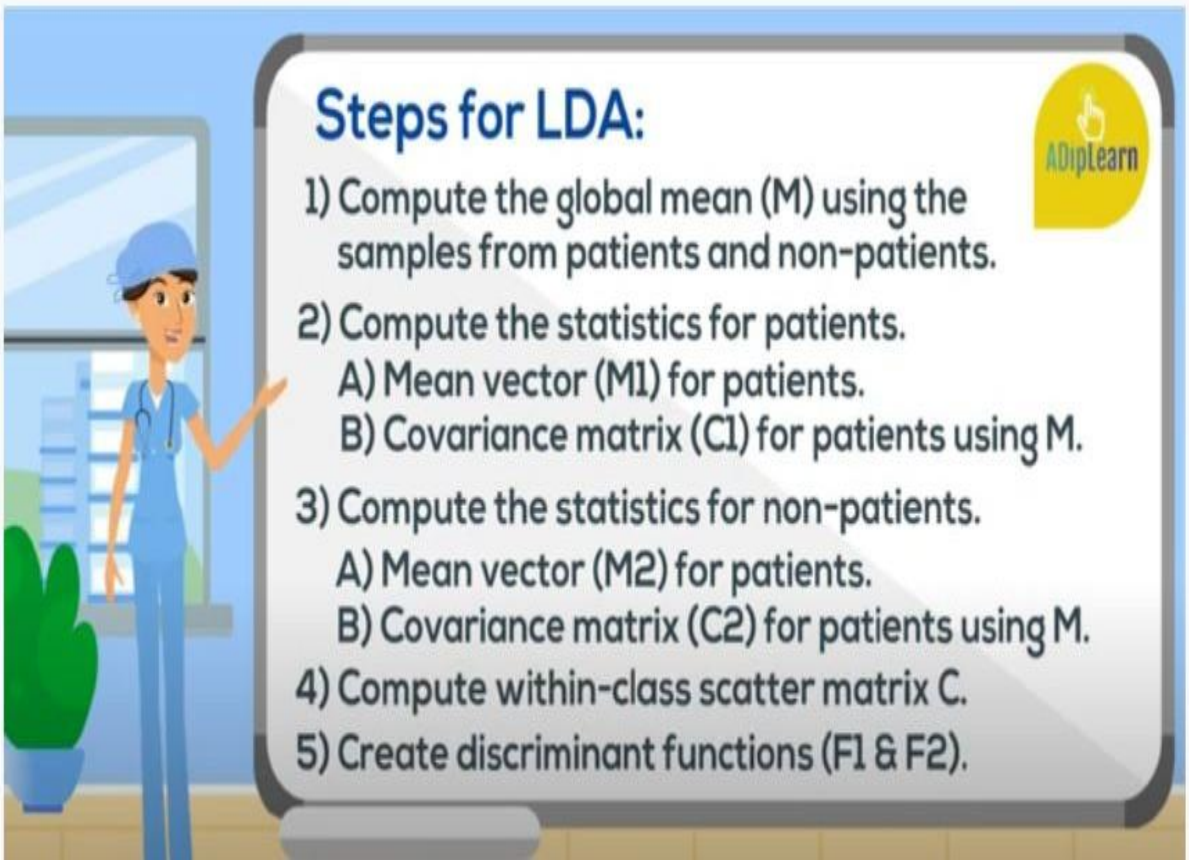
Dr. Jane collects blood pressure, pulse rate and temperature from patients and non-patients for further medical analysis.



With the collected data samples, Dr. Jane would like to estimate if the new comer gets the disease based on the measurement.

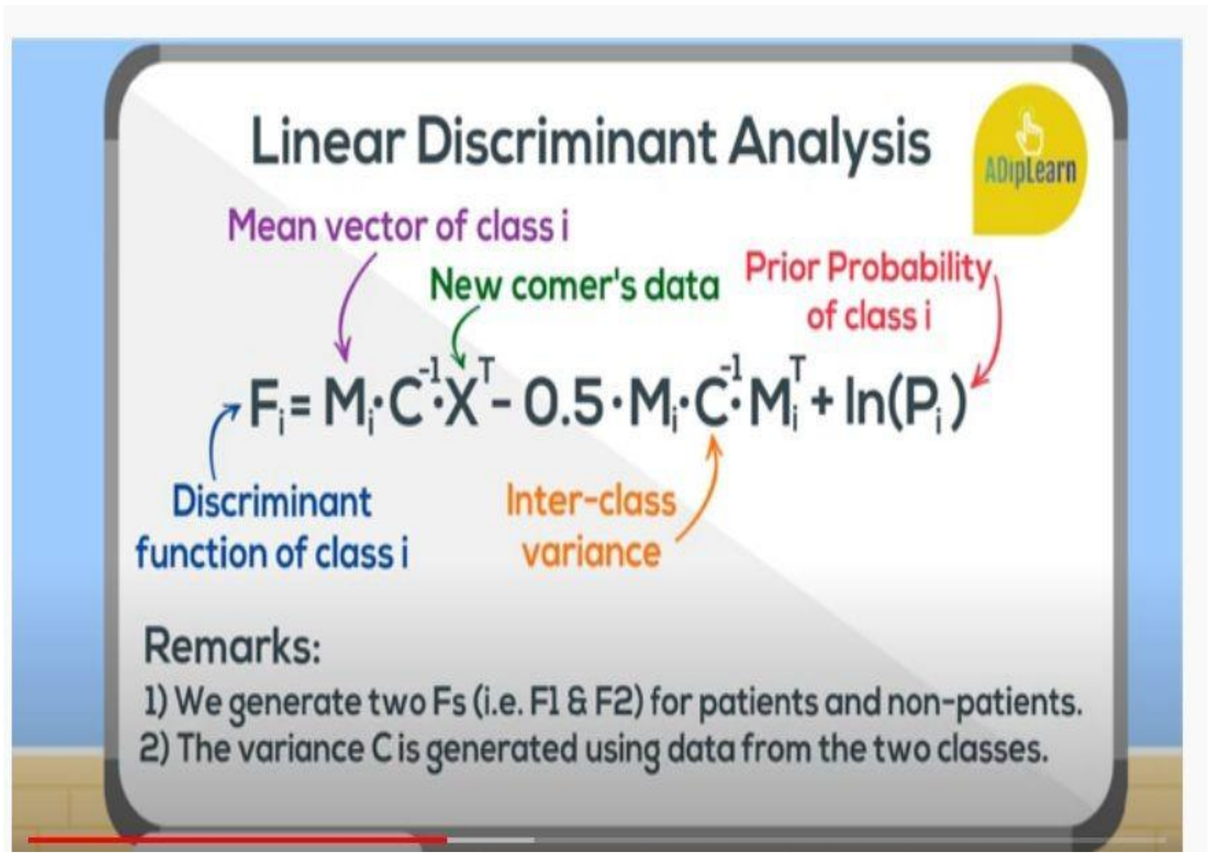


Dr. Jane decides to use the collected data samples to train a LDA.
The trained model is used to predict if the new comer gets infected.



Steps for LDA:

1) Compute the global mean (M) using the samples from patients and non-patients.
2) Compute the statistics for patients.
   A) Mean vector (M1) for patients.
   B) Covariance matrix (C1) for patients using M.
3) Compute the statistics for non-patients.
   A) Mean vector (M2) for patients.
   B) Covariance matrix (C2) for patients using M.
4) Compute within-class scatter matrix C.
5) Create discriminant functions (F1 & F2).

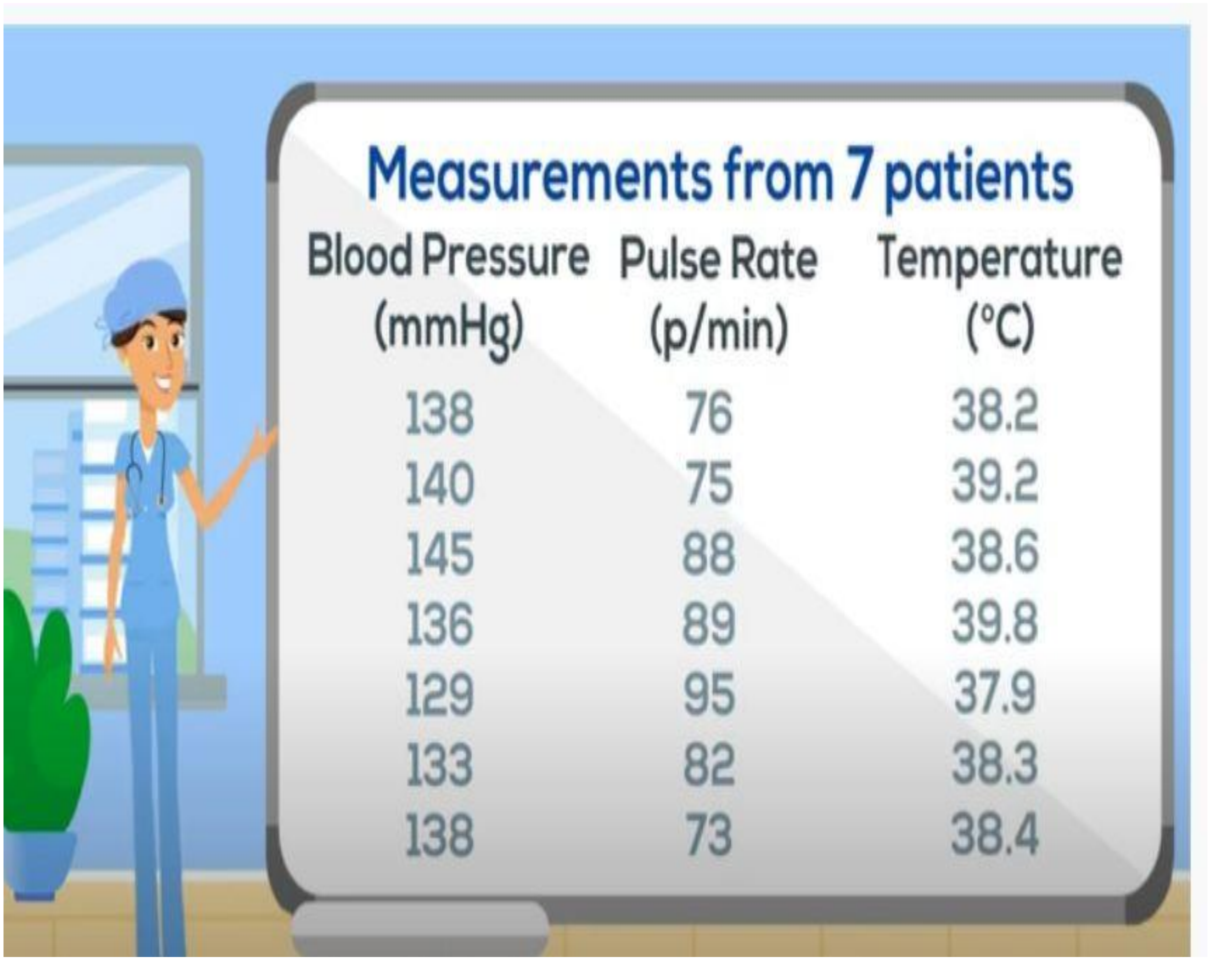


Linear Discriminant Analysis

$$F_i = M_i \cdot C^{-1} \cdot X^T - 0.5 \cdot M_i \cdot C^{-1} M_i^T + \ln(P_i)$$

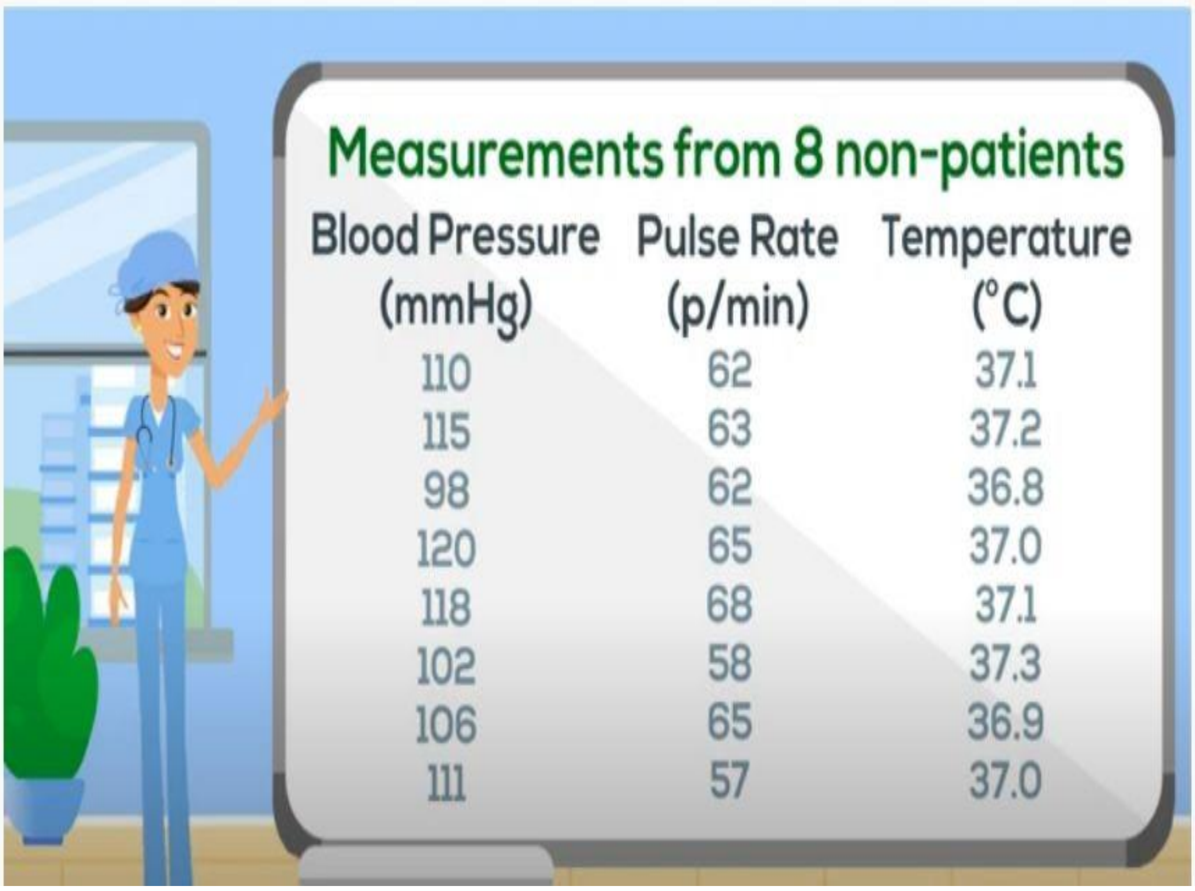
Remarks:

1) We generate two Fs (i.e. F1 & F2) for patients and non-patients.
2) The variance C is generated using data from the two classes.



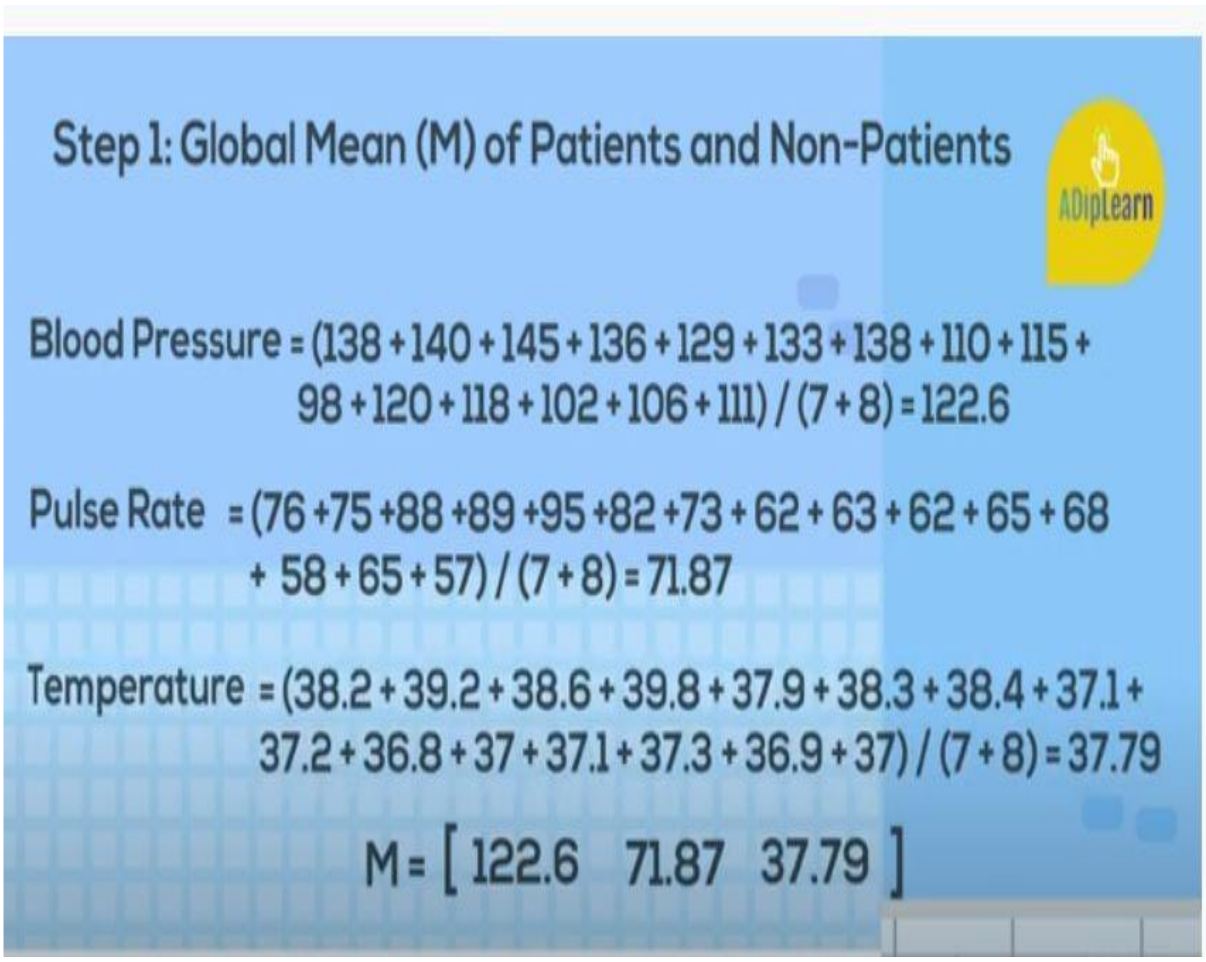
Measurements from 7 patients

Blood Pressure (mmHg)	Pulse Rate (p/min)	Temperature (°C)
138	76	38.2
140	75	39.2
145	88	38.6
136	89	39.8
129	95	37.9
133	82	38.3
138	73	38.4



Measurements from 8 non-patients

Blood Pressure (mmHg)	Pulse Rate (p/min)	Temperature (°C)
110	62	37.1
115	63	37.2
98	62	36.8
120	65	37.0
118	68	37.1
102	58	37.3
106	65	36.9
111	57	37.0



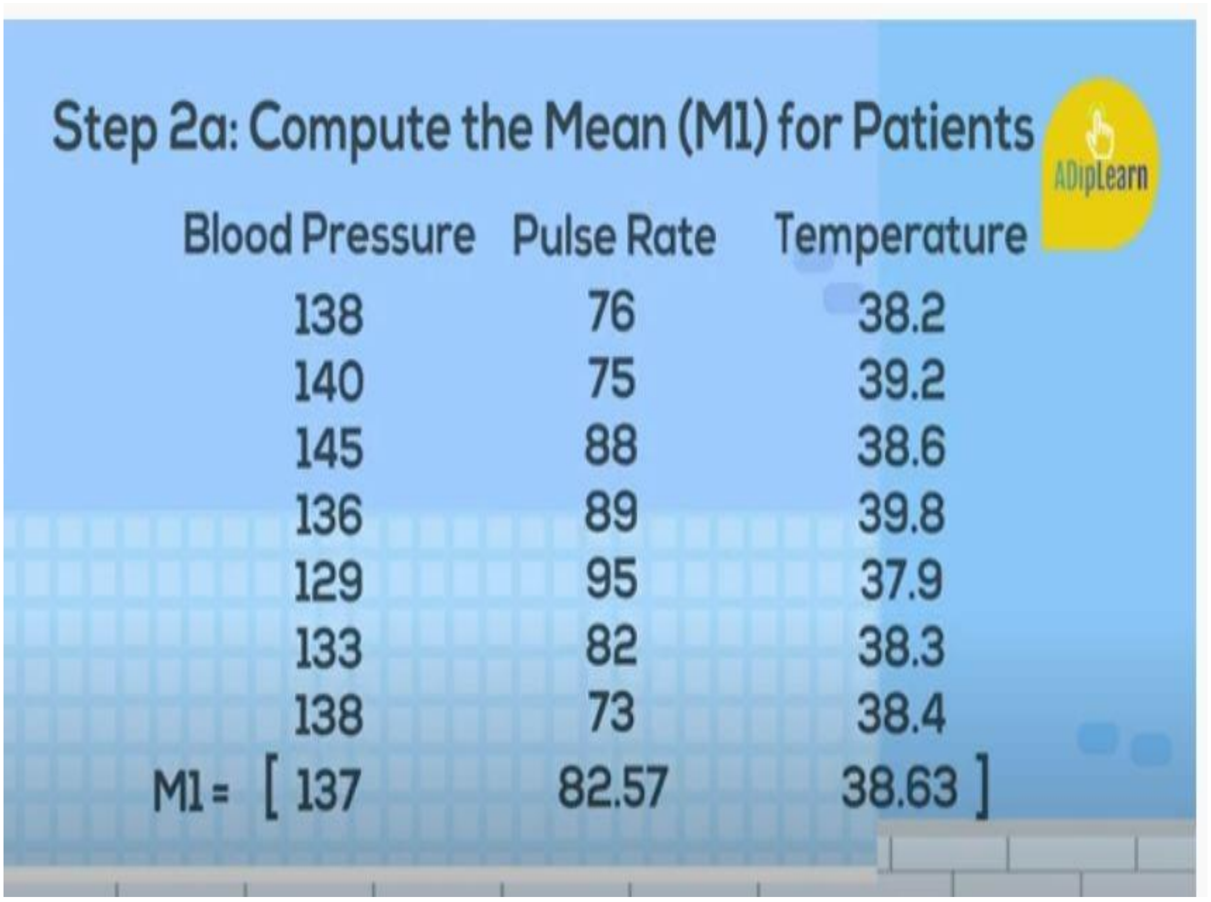
Step 1: Global Mean (M) of Patients and Non-Patients

Blood Pressure = (138 + 140 + 145 + 136 + 129 + 133 + 138 + 110 + 115 + 98 + 120 + 118 + 102 + 106 + 111) / (7 + 8) = 122.6

Pulse Rate = (76 +75 +88 +89 +95 +82 +73 + 62 + 63 + 62 + 65 + 68 + 58 + 65 + 57) / (7 + 8) = 71.87

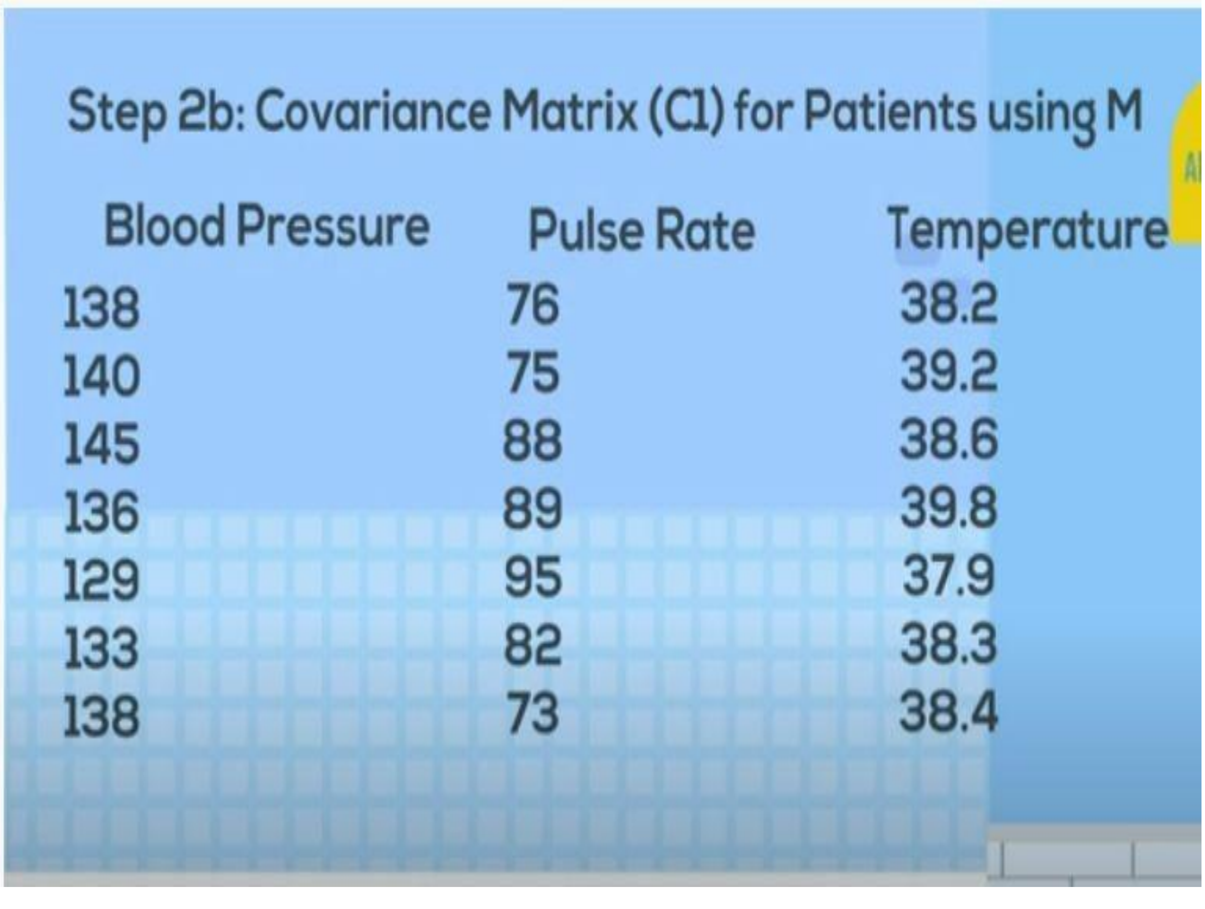
Temperature = (38.2 + 39.2 + 38.6 + 39.8 + 37.9 + 38.3 + 38.4 + 37.1 + 37.2 + 36.8 + 37 + 37.1 + 37.3 + 36.9 + 37) / (7 + 8) = 37.79

$$M = \begin{bmatrix} 122.6 & 71.87 & 37.79 \end{bmatrix}$$



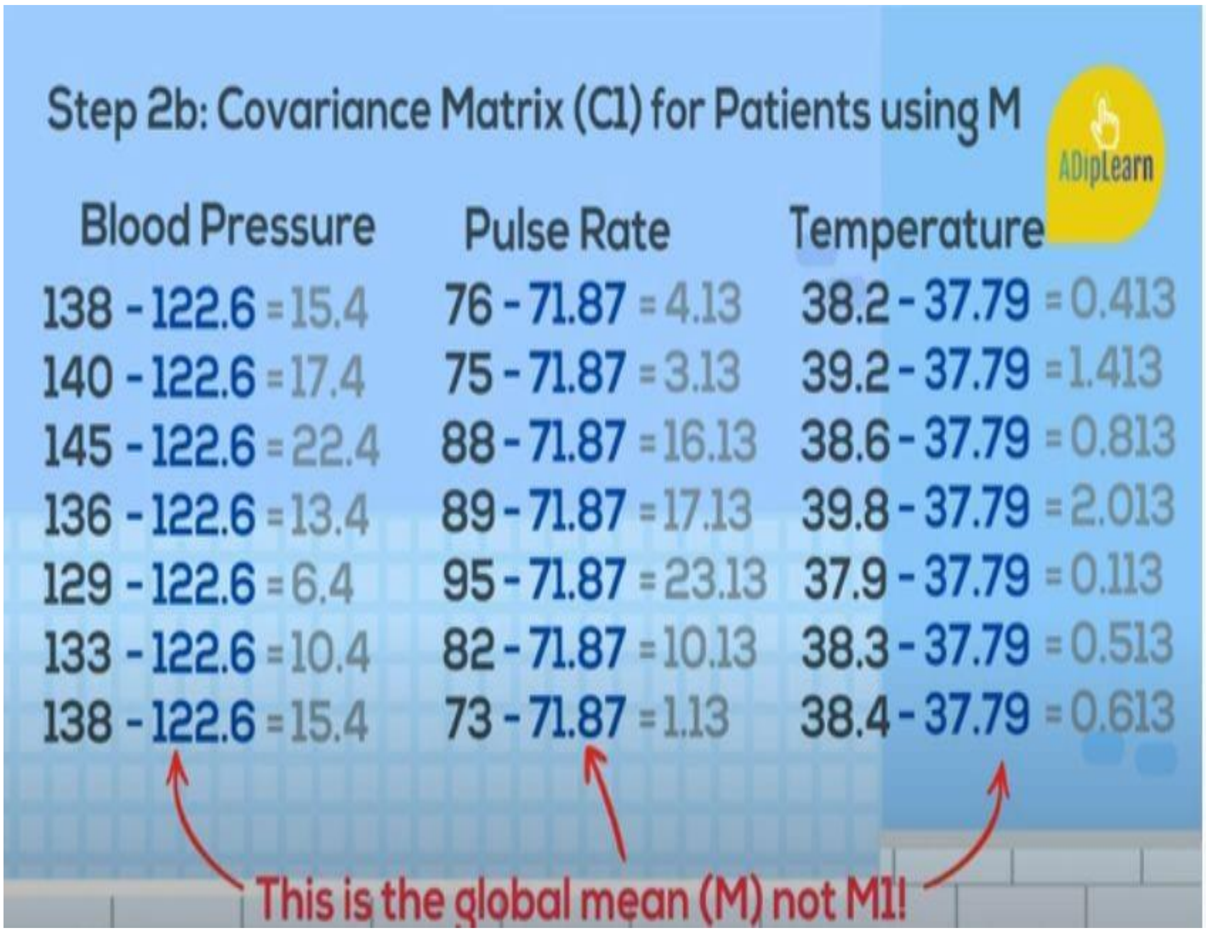
Step 2a: Compute the Mean (M1) for Patients

	Blood Pressure	Pulse Rate	Temperature
	138	76	38.2
	140	75	39.2
	145	88	38.6
	136	89	39.8
	129	95	37.9
	133	82	38.3
	138	73	38.4
M1 =	[137	82.57	38.63]



Step 2b: Covariance Matrix (C1) for Patients using M

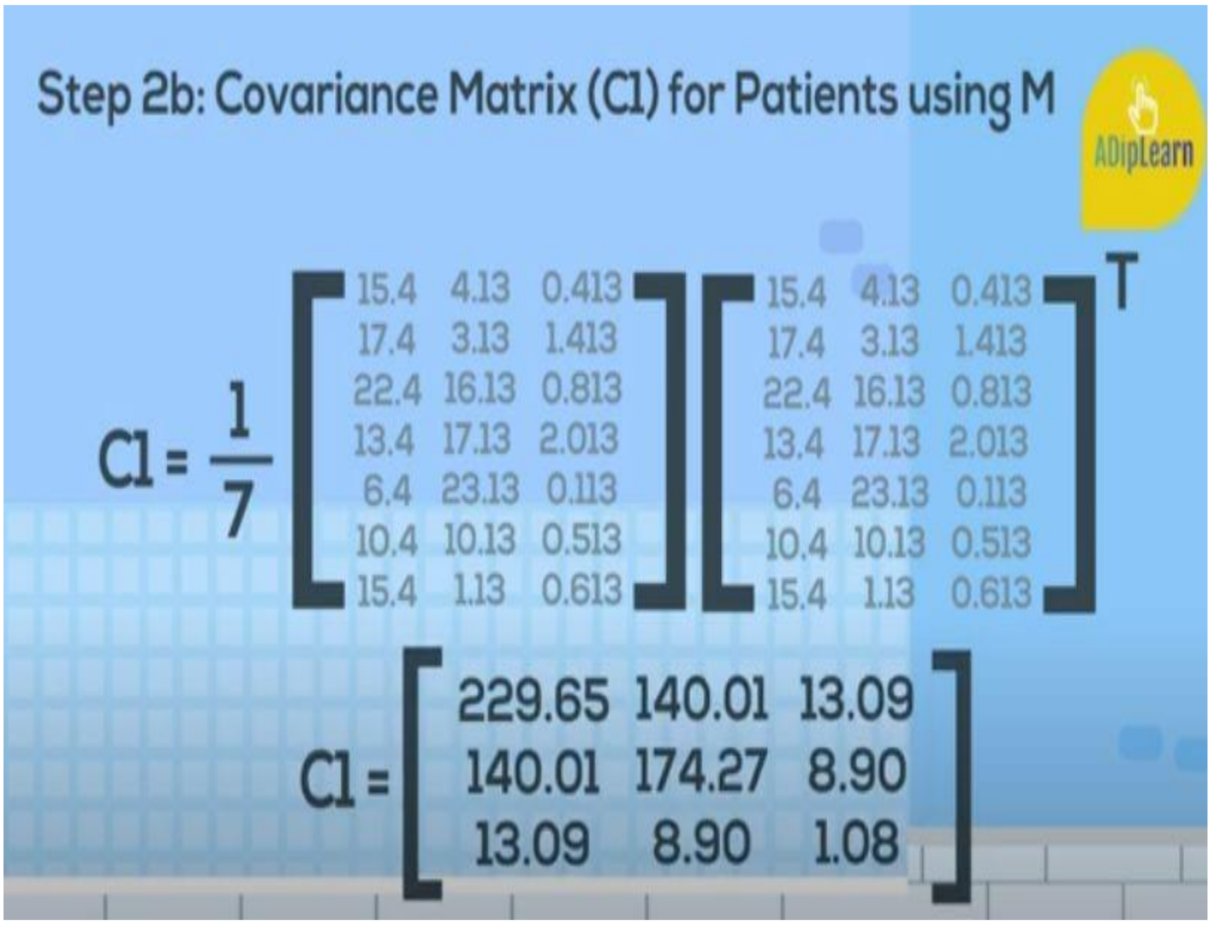
Blood Pressure	Pulse Rate	Temperature
138	76	38.2
140	75	39.2
145	88	38.6
136	89	39.8
129	95	37.9
133	82	38.3
138	73	38.4



Step 2b: Covariance Matrix (C1) for Patients using M

Blood Pressure	Pulse Rate	Temperature
138 - 122.6 = 15.4	76 - 71.87 = 4.13	38.2 - 37.79 = 0.413
140 - 122.6 = 17.4	75 - 71.87 = 3.13	39.2 - 37.79 = 1.413
145 - 122.6 = 22.4	88 - 71.87 = 16.13	38.6 - 37.79 = 0.813
136 - 122.6 = 13.4	89 - 71.87 = 17.13	39.8 - 37.79 = 2.013
129 - 122.6 = 6.4	95 - 71.87 = 23.13	37.9 - 37.79 = 0.113
133 - 122.6 = 10.4	82 - 71.87 = 10.13	38.3 - 37.79 = 0.513
138 - 122.6 = 15.4	73 - 71.87 = 1.13	38.4 - 37.79 = 0.613

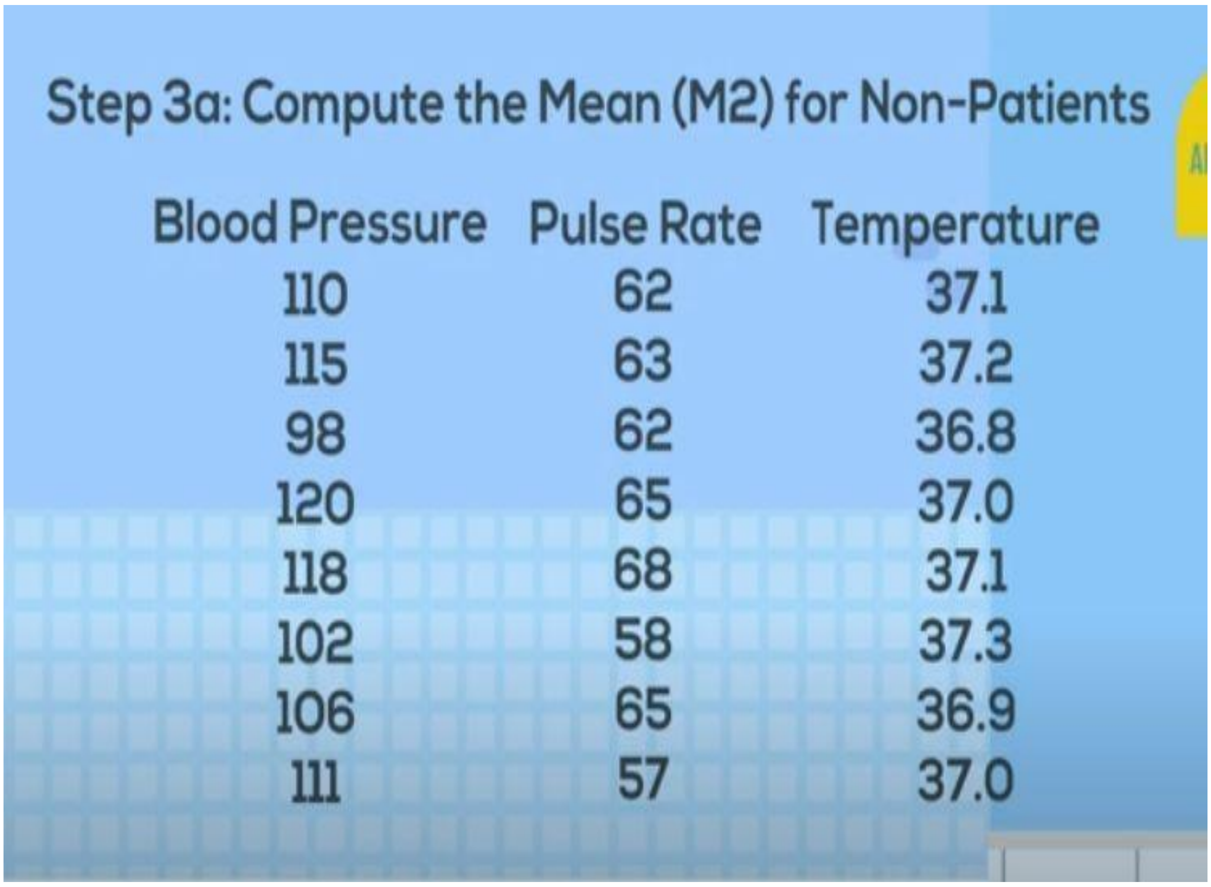
This is the global mean (M) not M1!



Step 2b: Covariance Matrix (C1) for Patients using M

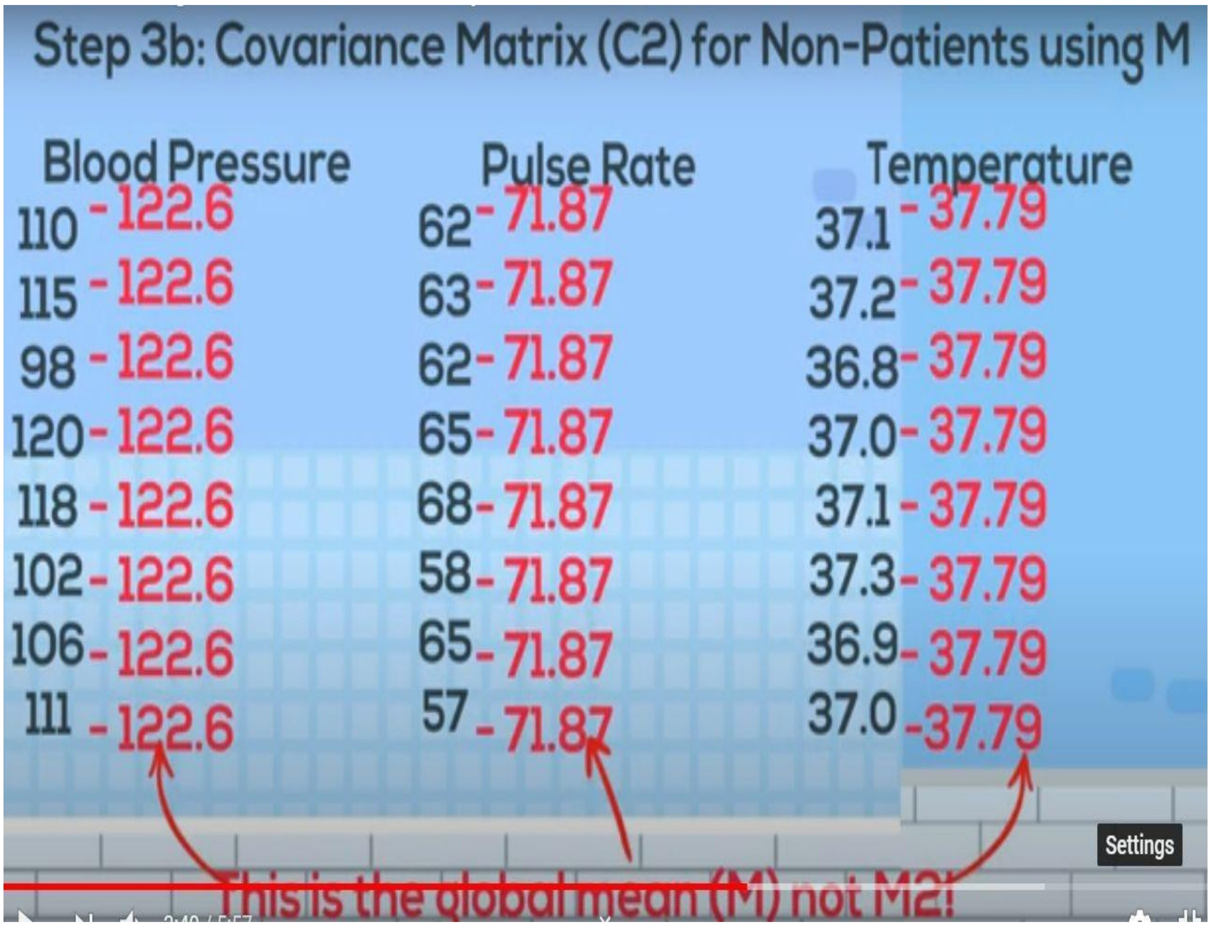
$$C1 = \frac{1}{7}\begin{bmatrix} 15.4 & 4.13 & 0.413 \\ 17.4 & 3.13 & 1.413 \\ 22.4 & 16.13 & 0.813 \\ 13.4 & 17.13 & 2.013 \\ 6.4 & 23.13 & 0.113 \\ 10.4 & 10.13 & 0.513 \\ 15.4 & 1.13 & 0.613 \end{bmatrix}\begin{bmatrix} 15.4 & 4.13 & 0.413 \\ 17.4 & 3.13 & 1.413 \\ 22.4 & 16.13 & 0.813 \\ 13.4 & 17.13 & 2.013 \\ 6.4 & 23.13 & 0.113 \\ 10.4 & 10.13 & 0.513 \\ 15.4 & 1.13 & 0.613 \end{bmatrix}^T$$

$$C1 = \begin{bmatrix} 229.65 & 140.01 & 13.09 \\ 140.01 & 174.27 & 8.90 \\ 13.09 & 8.90 & 1.08 \end{bmatrix}$$



Step 3a: Compute the Mean (M2) for Non-Patients

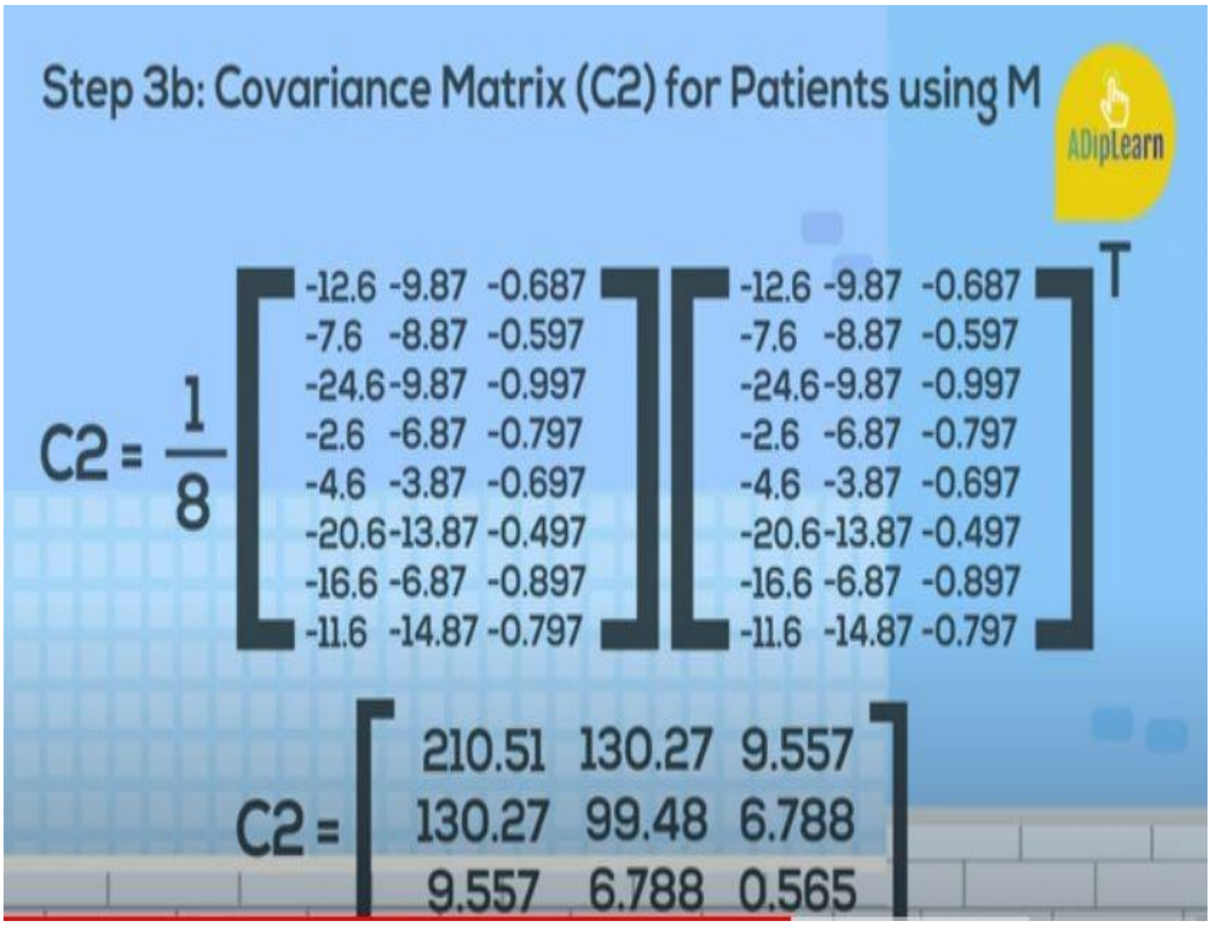
Blood Pressure	Pulse Rate	Temperature
110	62	37.1
115	63	37.2
98	62	36.8
120	65	37.0
118	68	37.1
102	58	37.3
106	65	36.9
111	57	37.0



Step 3b: Covariance Matrix (C2) for Non-Patients using M

Blood Pressure	Pulse Rate	Temperature
110 - 122.6	62 - 71.87	37.1 - 37.79
115 - 122.6	63 - 71.87	37.2 - 37.79
98 - 122.6	62 - 71.87	36.8 - 37.79
120 - 122.6	65 - 71.87	37.0 - 37.79
118 - 122.6	68 - 71.87	37.1 - 37.79
102 - 122.6	58 - 71.87	37.3 - 37.79
106 - 122.6	65 - 71.87	36.9 - 37.79
111 - 122.6	57 - 71.87	37.0 - 37.79

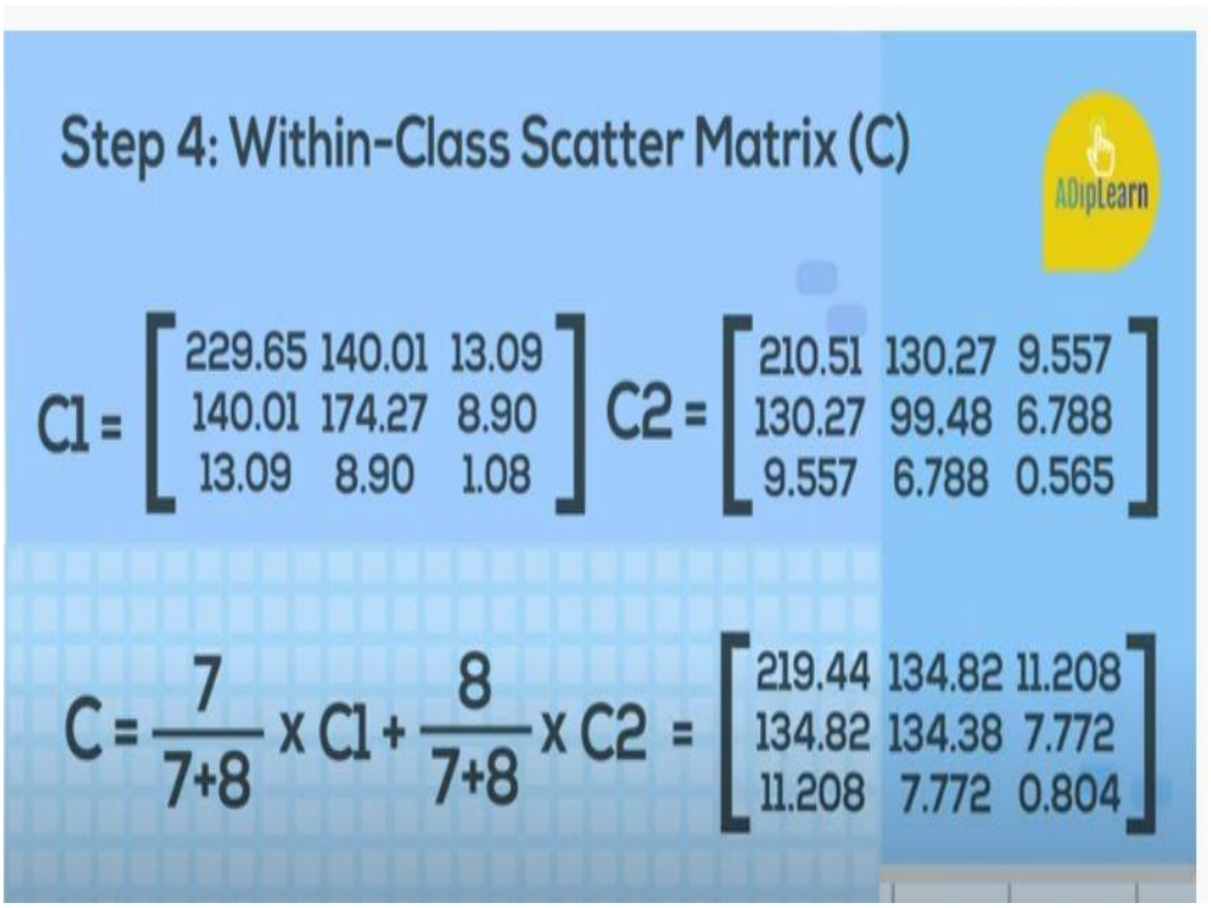
This is the global mean (M) not M2!



Step 3b: Covariance Matrix (C2) for Patients using M

$$C2 = \frac{1}{8}\begin{bmatrix} -12.6 & -9.87 & -0.687 \\ -7.6 & -8.87 & -0.597 \\ -24.6 & -9.87 & -0.997 \\ -2.6 & -6.87 & -0.797 \\ -4.6 & -3.87 & -0.697 \\ -20.6 & -13.87 & -0.497 \\ -16.6 & -6.87 & -0.897 \\ -11.6 & -14.87 & -0.797 \end{bmatrix}\begin{bmatrix} -12.6 & -9.87 & -0.687 \\ -7.6 & -8.87 & -0.597 \\ -24.6 & -9.87 & -0.997 \\ -2.6 & -6.87 & -0.797 \\ -4.6 & -3.87 & -0.697 \\ -20.6 & -13.87 & -0.497 \\ -16.6 & -6.87 & -0.897 \\ -11.6 & -14.87 & -0.797 \end{bmatrix}^T$$

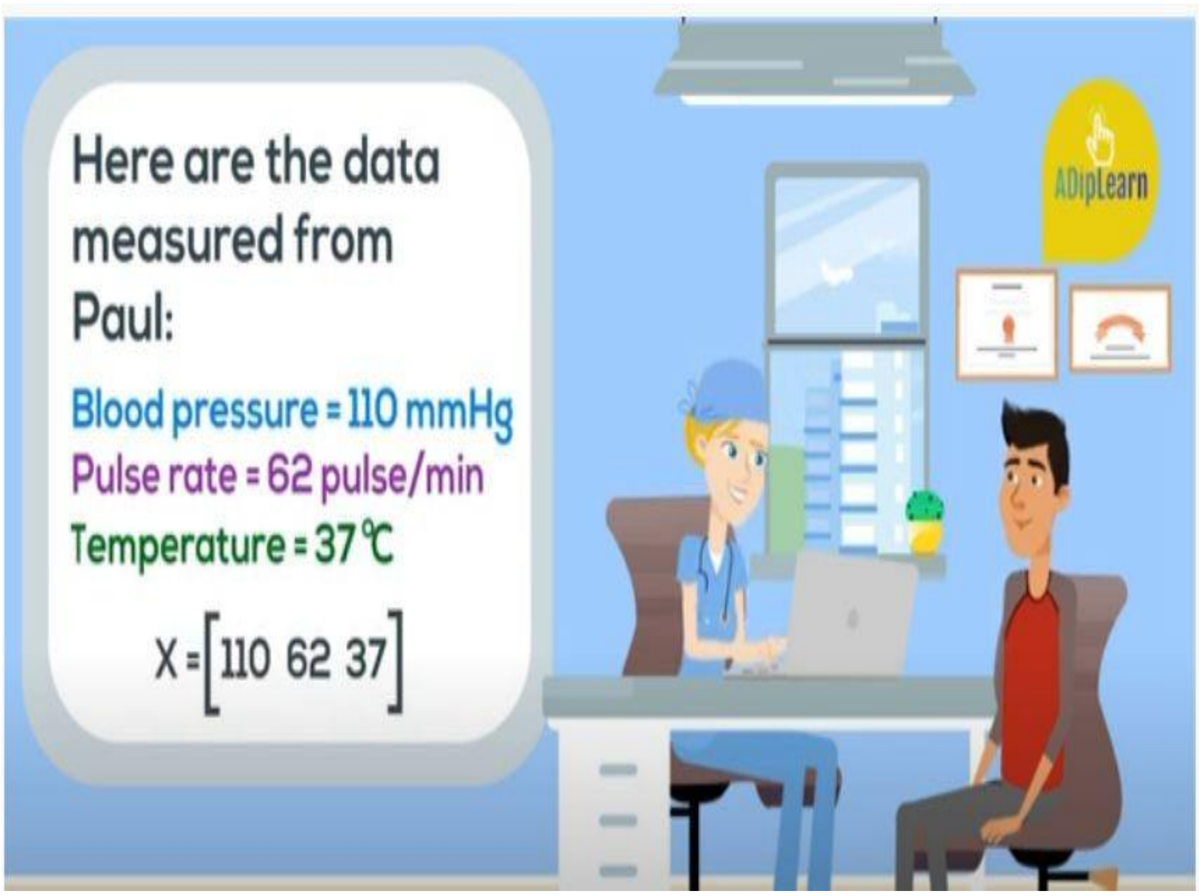
$$C2 = \begin{bmatrix} 210.51 & 130.27 & 9.557 \\ 130.27 & 99.48 & 6.788 \\ 9.557 & 6.788 & 0.565 \end{bmatrix}$$



Step 4: Within-Class Scatter Matrix (C)

$$C1 = \begin{bmatrix} 229.65 & 140.01 & 13.09 \\ 140.01 & 174.27 & 8.90 \\ 13.09 & 8.90 & 1.08 \end{bmatrix} \quad C2 = \begin{bmatrix} 210.51 & 130.27 & 9.557 \\ 130.27 & 99.48 & 6.788 \\ 9.557 & 6.788 & 0.565 \end{bmatrix}$$

$$C = \frac{7}{7+8} \times C1 + \frac{8}{7+8} \times C2 = \begin{bmatrix} 219.44 & 134.82 & 11.208 \\ 134.82 & 134.38 & 7.772 \\ 11.208 & 7.772 & 0.804 \end{bmatrix}$$



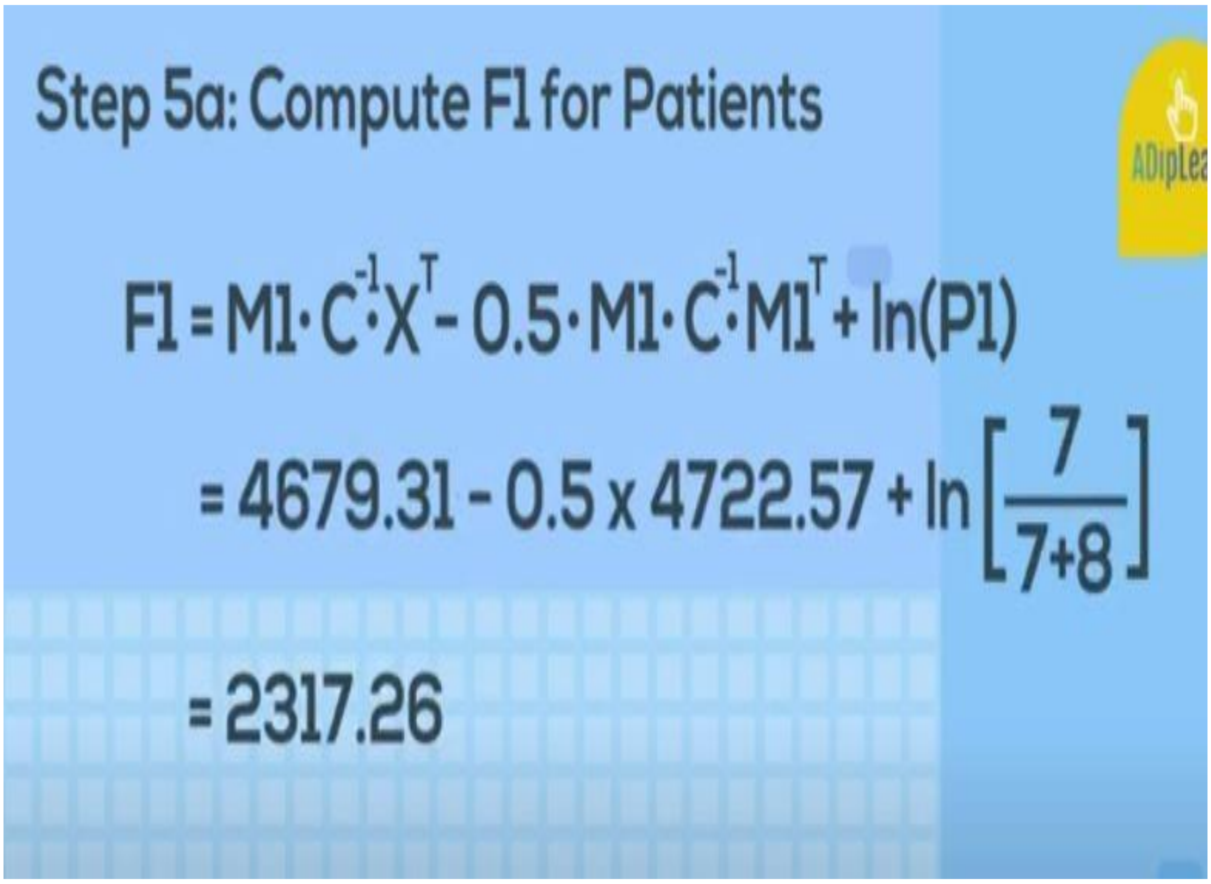
Here are the data measured from Paul:

Blood pressure = 110 mmHg

Pulse rate = 62 pulse/min

Temperature = 37℃

$$X = \begin{bmatrix} 110 & 62 & 37 \end{bmatrix}$$

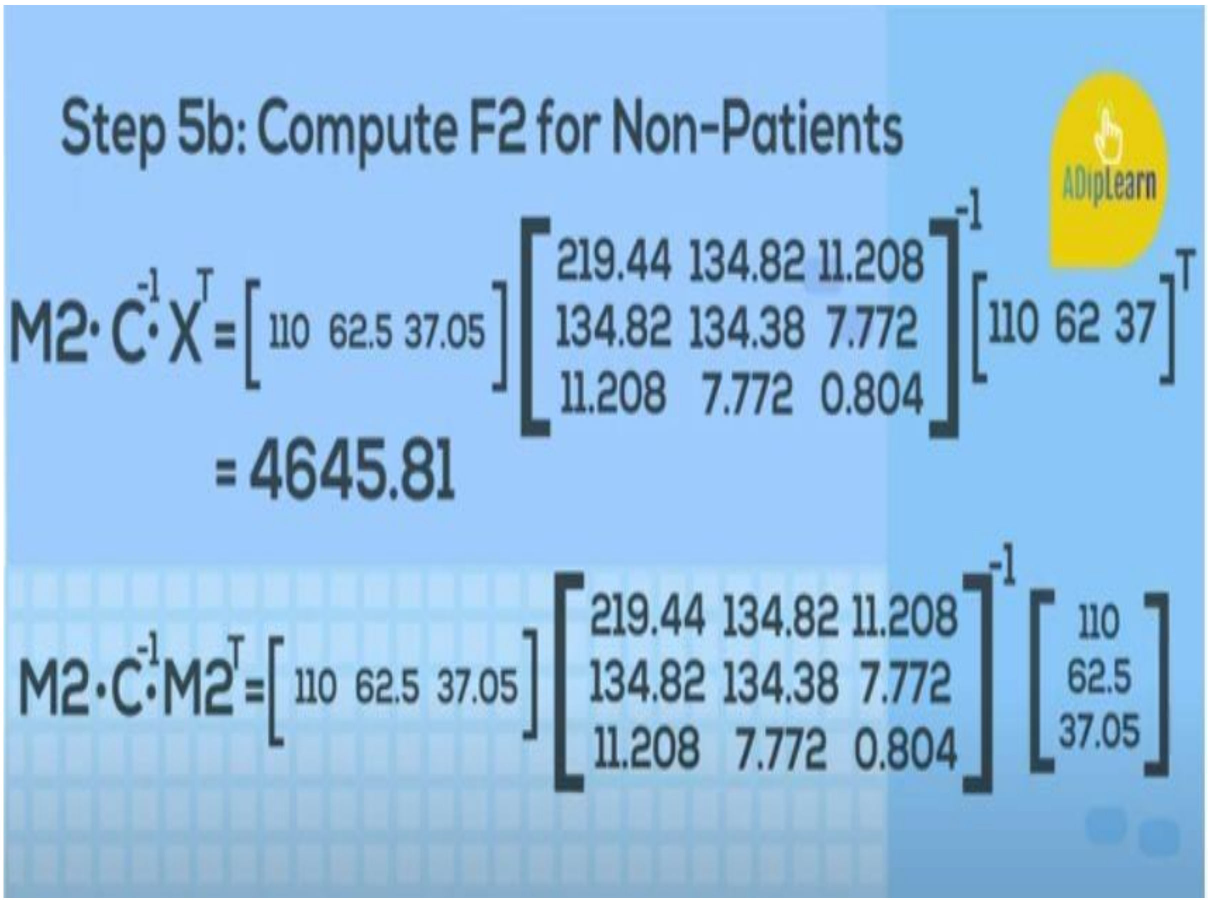


Step 5a: Compute F1 for Patients

$$F1 = M1 \cdot C^{-1} \cdot X^{T} - 0.5 \cdot M1 \cdot C^{-1} \cdot M1^{T} + \ln(P1)$$

$$= 4679.31 - 0.5 \times 4722.57 + \ln\left[\frac{7}{7+8}\right]$$

$$= 2317.26$$



Step 5b: Compute F2 for Non-Patients

$$M2 \cdot C^{-1} \cdot X^{T} = \begin{bmatrix} 110 & 62.5 & 37.05 \end{bmatrix} \begin{bmatrix} 219.44 & 134.82 & 11.208 \\ 134.82 & 134.38 & 7.772 \\ 11.208 & 7.772 & 0.804 \end{bmatrix}^{-1} \begin{bmatrix} 110 & 62 & 37 \end{bmatrix}^{T}$$

$$= 4645.81$$

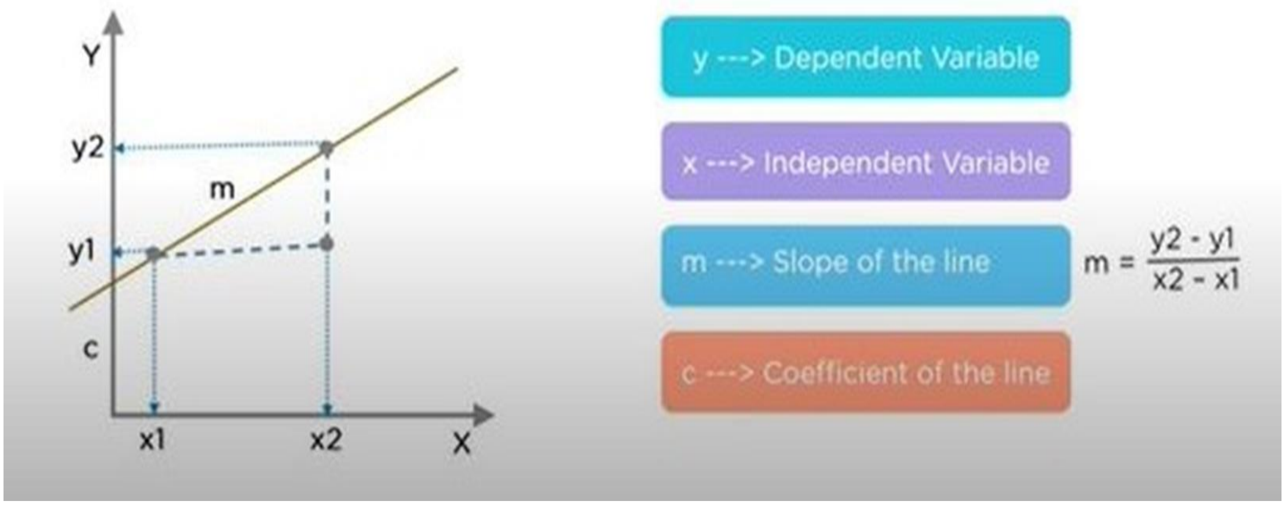
$$M2 \cdot C^{-1} \cdot M2^{T} = \begin{bmatrix} 110 & 62.5 & 37.05 \end{bmatrix} \begin{bmatrix} 219.44 & 134.82 & 11.208 \\ 134.82 & 134.38 & 7.772 \\ 11.208 & 7.772 & 0.804 \end{bmatrix}^{-1} \begin{bmatrix} 110 \\ 62.5 \\ 37.05 \end{bmatrix}$$

Linear Regression Analysis

• Linear regression that attempts to show the relationship between two variables with the linear equation.

• The simples form of a simple linear regression equation with one dependent and one independent variable is represented by

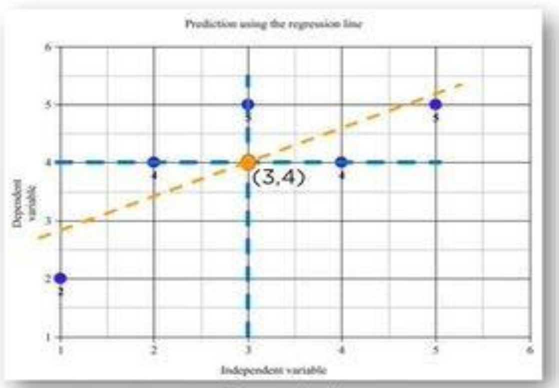
$$y = m * x + c$$


Y
y2
m
y1
c
x1
x2
X
y ---> Dependent Variable
x ---> Independent Variable
m ---> Slope of the line
$m = \frac{y2 - y1}{x2 - x1}$
c ---> Coefficient of the line

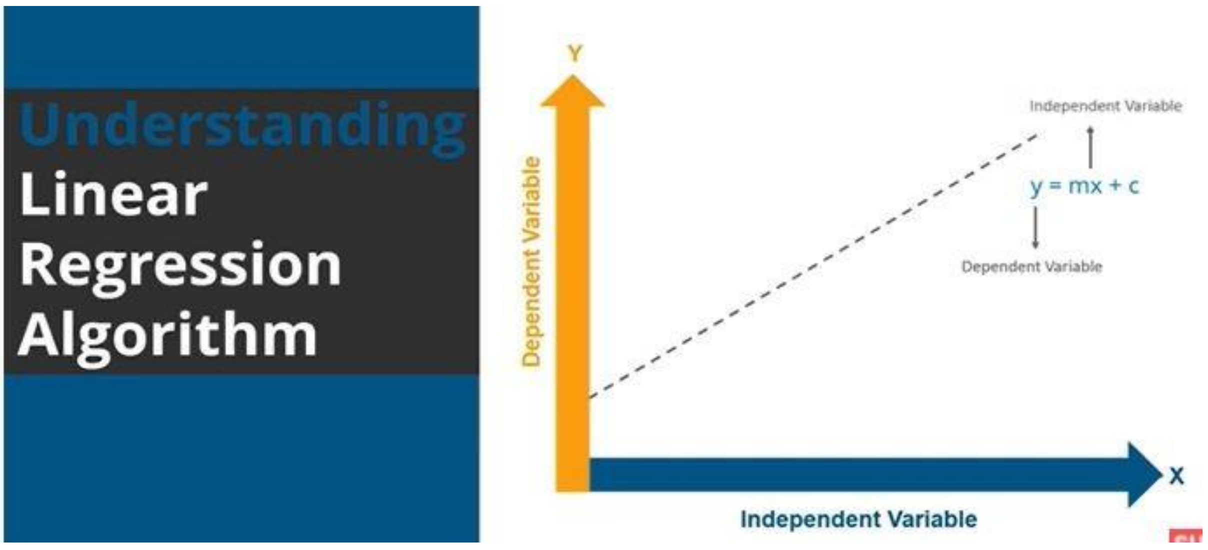
Intuition behind the Regression line

Regression line should ideally pass through the mean of X and Y

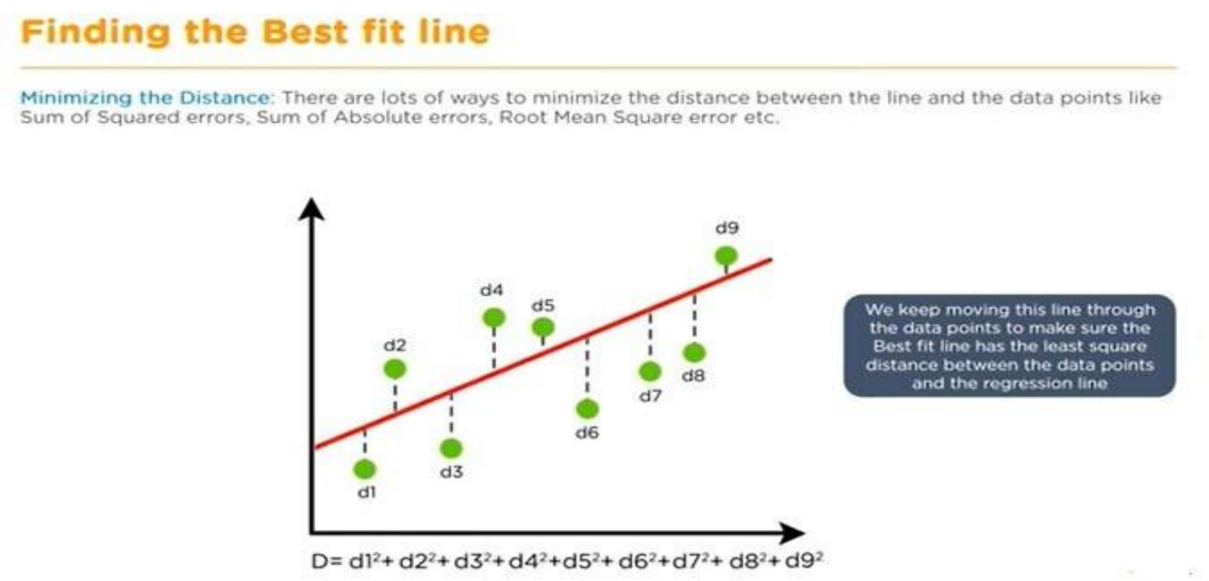
Independent variable	Dependent variable
X	**Y**
1	2
2	4
3	5
4	4
5	5
Mean 3	4



Regression line



Understanding Linear Regression Algorithm



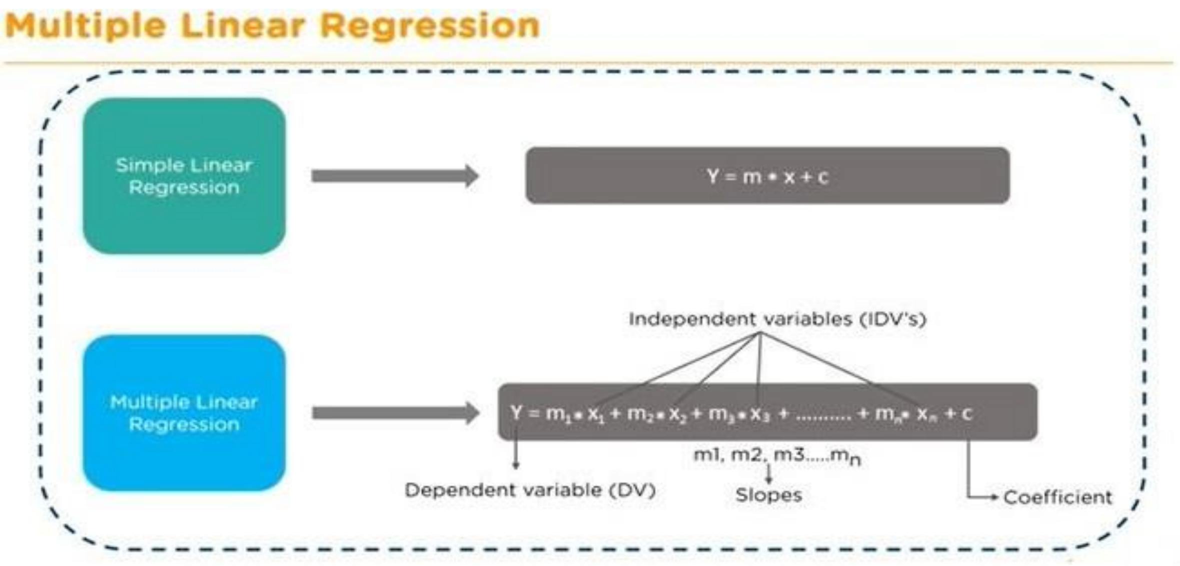
Finding the Best fit line

Minimizing the Distance: There are lots of ways to minimize the distance between the line and the data points like Sum of Squared errors, Sum of Absolute errors, Root Mean Square error etc.

We keep moving this line through the data points to make sure the Best fit line has the least square distance between the data points and the regression line

$$D = d1^2 + d2^2 + d3^2 + d4^2 + d5^2 + d6^2 + d7^2 + d8^2 + d9^2$$

d1
d2
d3
d4
d5
d6
d7
d8
d9

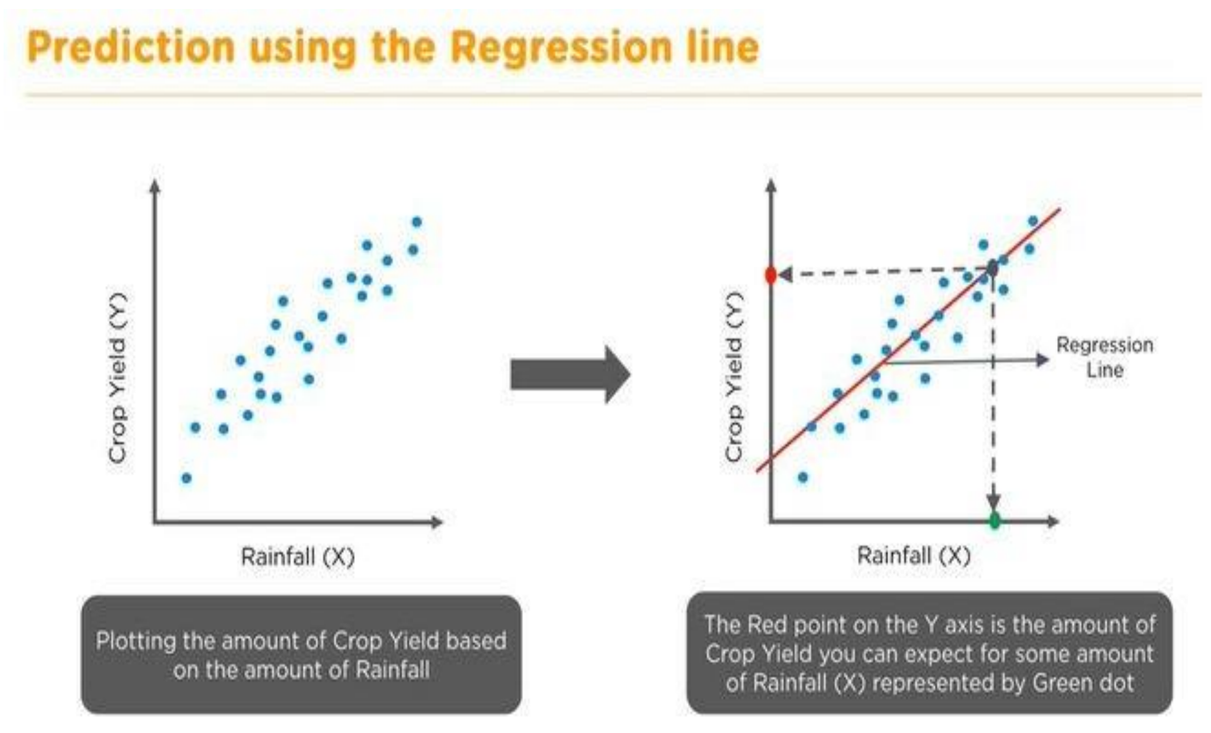


Multiple Linear Regression

Simple Linear Regression
$Y = m * x + c$
Independent variables (IDV's)
Multiple Linear Regression
$Y = m_1 * x_1 + m_2 * x_2 + m_3 * x_3 + \dots\dots\dots + m_n * x_n + c$
m1, m2, m3.....m_n
Dependent variable (DV)
Slopes
Coefficient

- Product price- Sale
- Height-Weight
- Temperature-Ice Cream sales
- Process- RAM

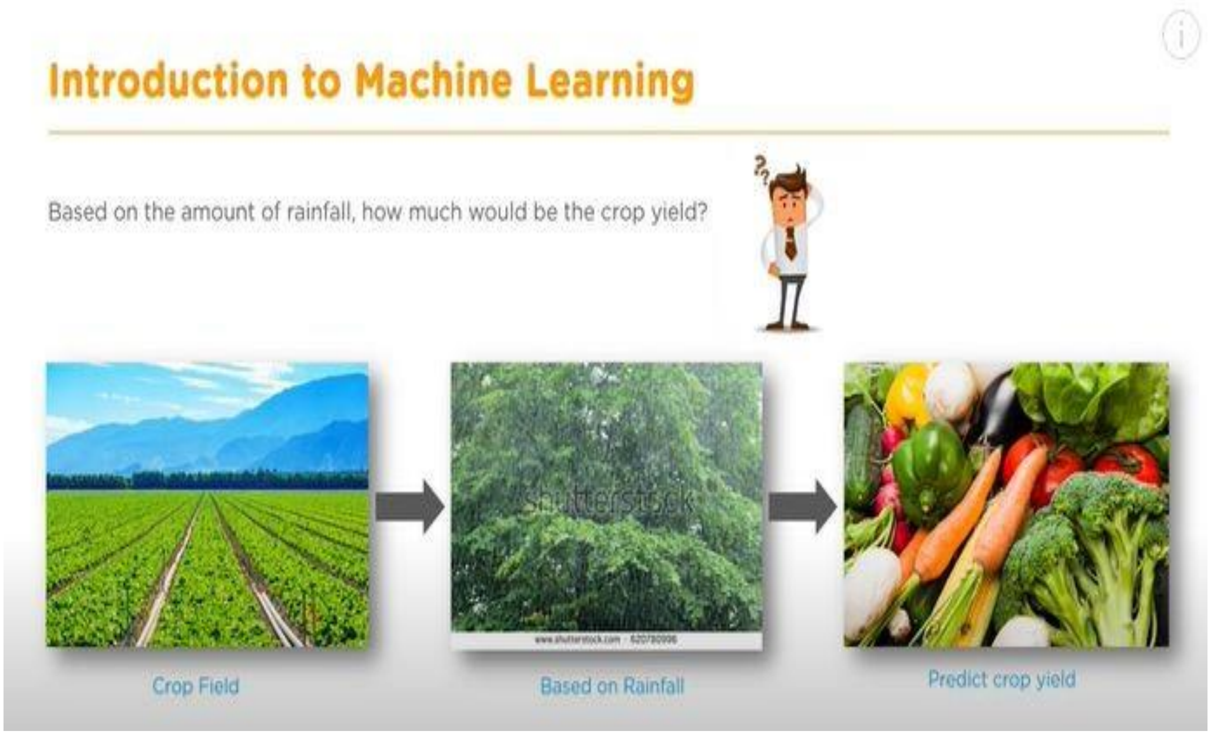
Linear Regression formula: y=mx+b



Prediction using the Regression line

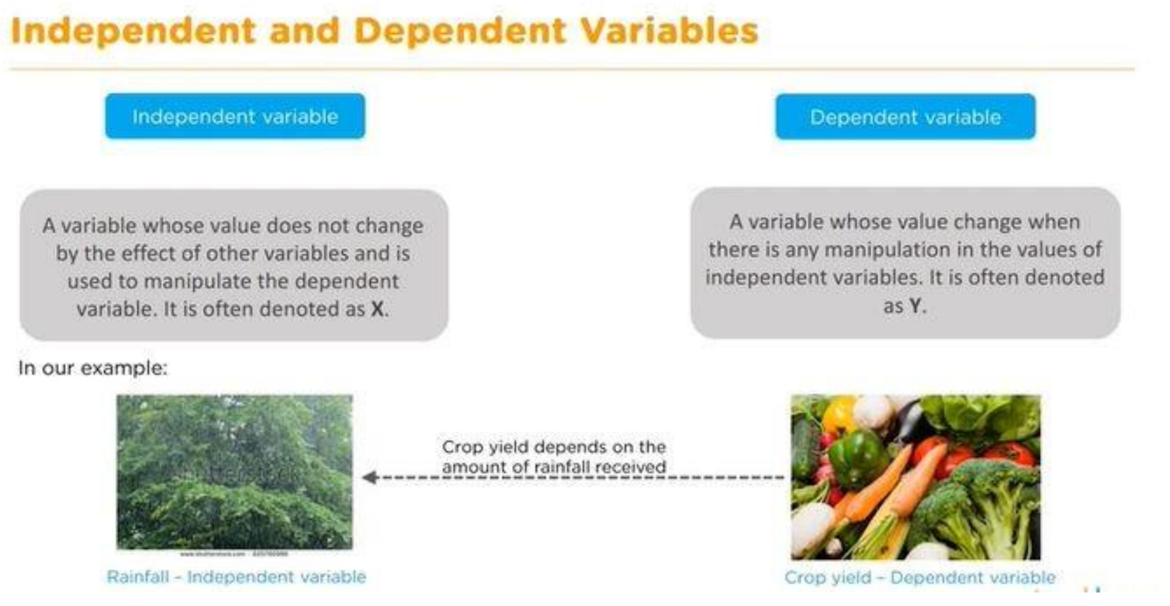
Plotting the amount of Crop Yield based on the amount of Rainfall

The Red point on the Y axis is the amount of Crop Yield you can expect for some amount of Rainfall (X) represented by Green dot



Introduction to Machine Learning

Based on the amount of rainfall, how much would be the crop yield?



Independent and Dependent Variables

Independent variable

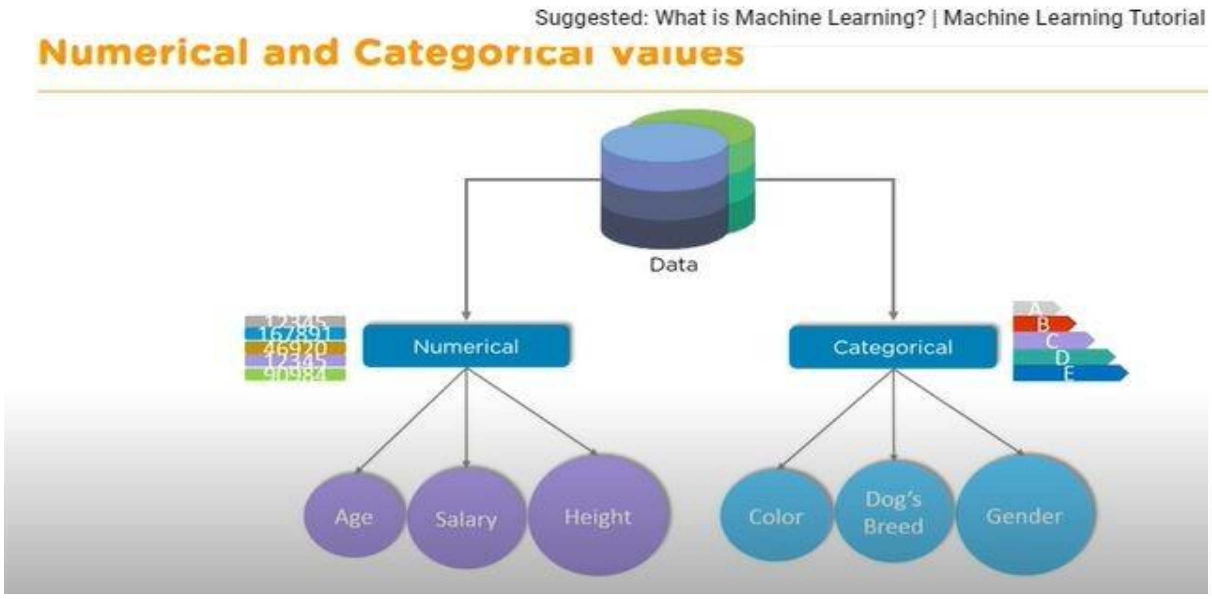
A variable whose value does not change by the effect of other variables and is used to manipulate the dependent variable. It is often denoted as **X**.

Dependent variable

A variable whose value change when there is any manipulation in the values of independent variables. It is often denoted as **Y**.

In our example:

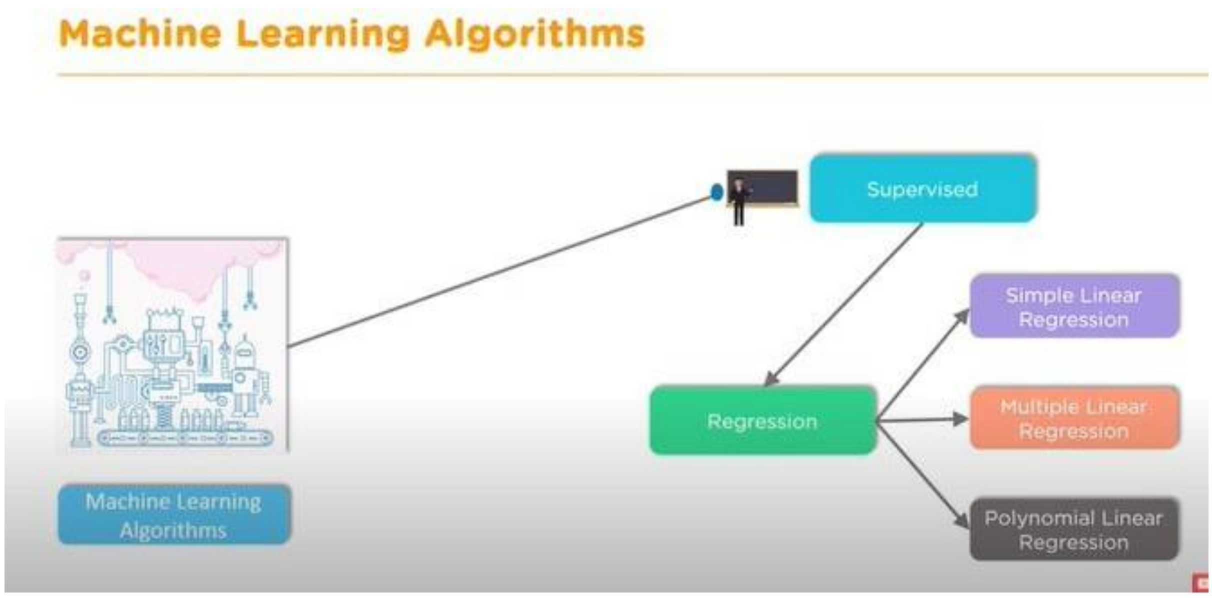
Crop yield depends on the amount of rainfall received
Rainfall - Independent variable
Crop yield - Dependent variable



Suggested: What is Machine Learning? | Machine Learning Tutorial

Numerical and Categorical values

Data
Numerical
Categorical
Age
Salary
Height
Color
Dog's Breed
Gender
A
B
C
D
E



Machine Learning Algorithms

Machine Learning Algorithms
Supervised
Regression
Simple Linear Regression
Multiple Linear Regression
Polynomial Linear Regression

Applications of Linear Regression

Economic Growth

Used to determine the Economic Growth of a country or a state in the coming quarter, can also be used to predict the GDP of a country

Applications of Linear Regression

Product price

Can be used to predict what would be the price of a product in the future

Applications of Linear Regression

Housing sales

To estimate the number of houses a builder would sell and at what price in the coming months

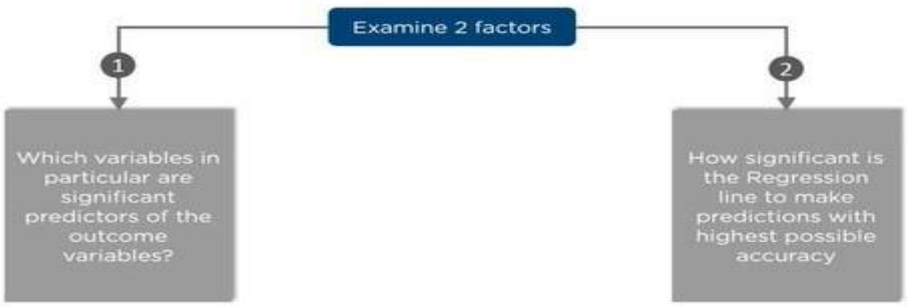
Applications of Linear Regression

Score Prediction

To predict the number of runs a player would score in the coming matches based on previous performance

Understanding Linear Regression

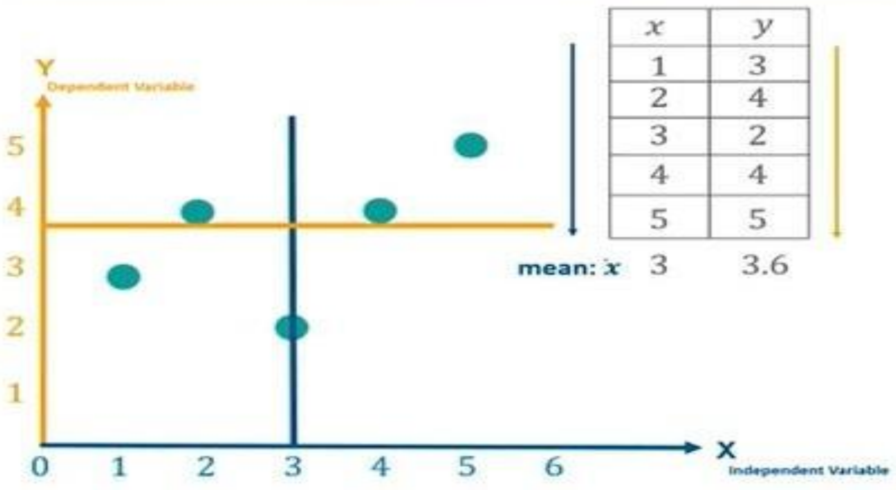
Linear Regression is a statistical model used to predict the relationship between independent and dependent variables.



Which variables in particular are significant predictors of the outcome variables?

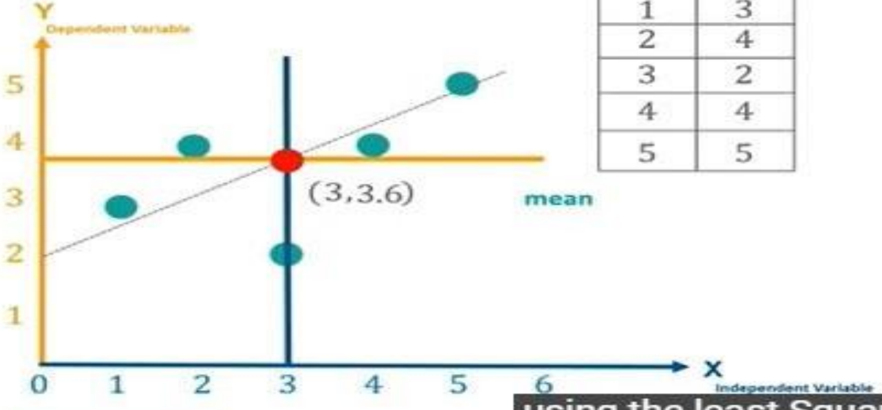
How significant is the Regression line to make predictions with highest possible accuracy

Understanding Linear Regression Algorithm



x	y
1	3
2	4
3	2
4	4
5	5
mean: $\bar{x}$ 3	3.6

Understanding Linear Regression Algorithm

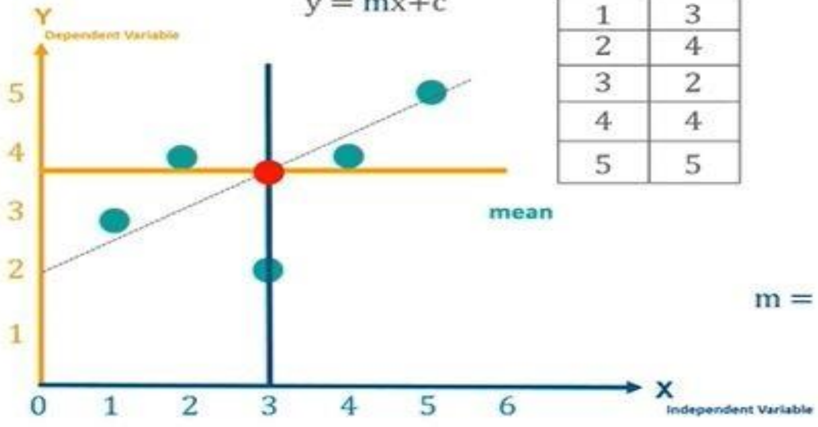


x	y
1	3
2	4
3	2
4	4
5	5

using the least Square

Understanding Linear Regression Algorithm

$y = mx + c$

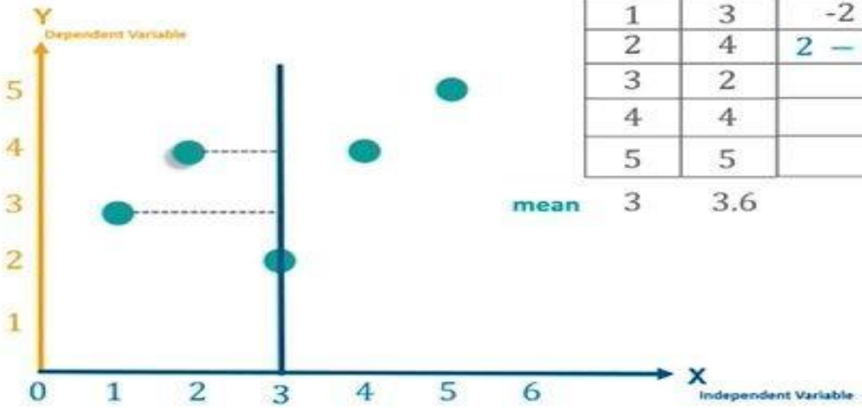


x	y
1	3
2	4
3	2
4	4
5	5

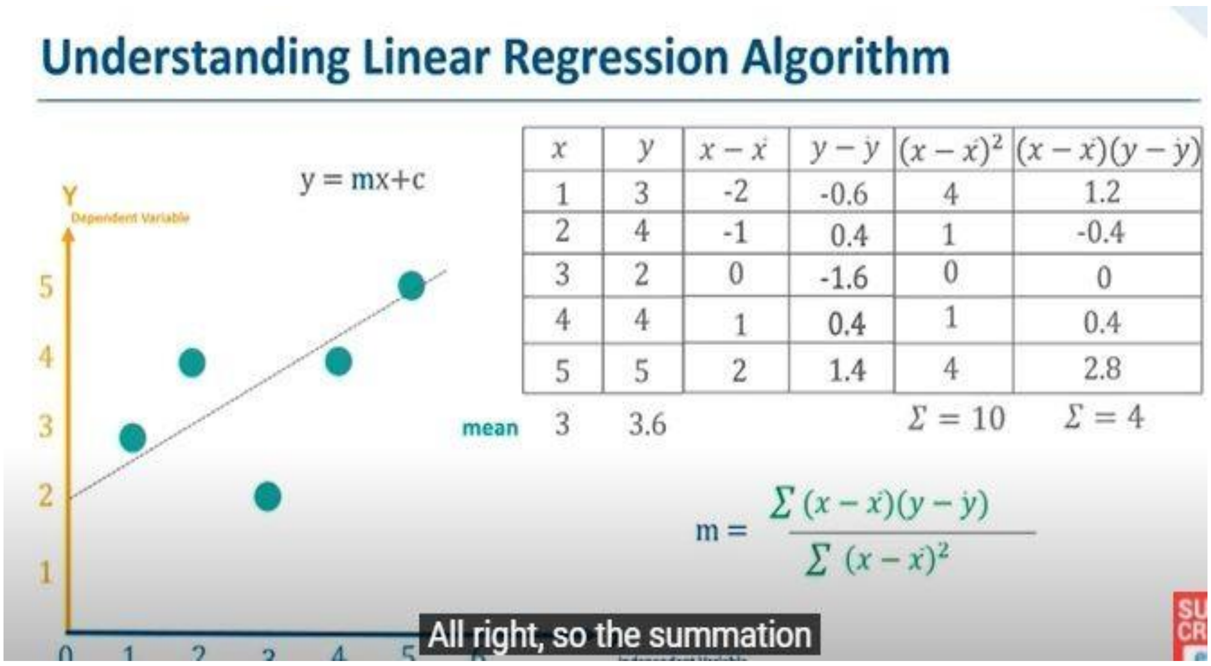
$$m = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sum (x - \bar{x})^2}$$

where m equals summation of x

Understanding Linear Regression Algorithm



	x	y	$x - \bar{x}$
	1	3	-2
	2	4	$2 - 3$
	3	2	
	4	4	
	5	5	
mean	3	3.6	



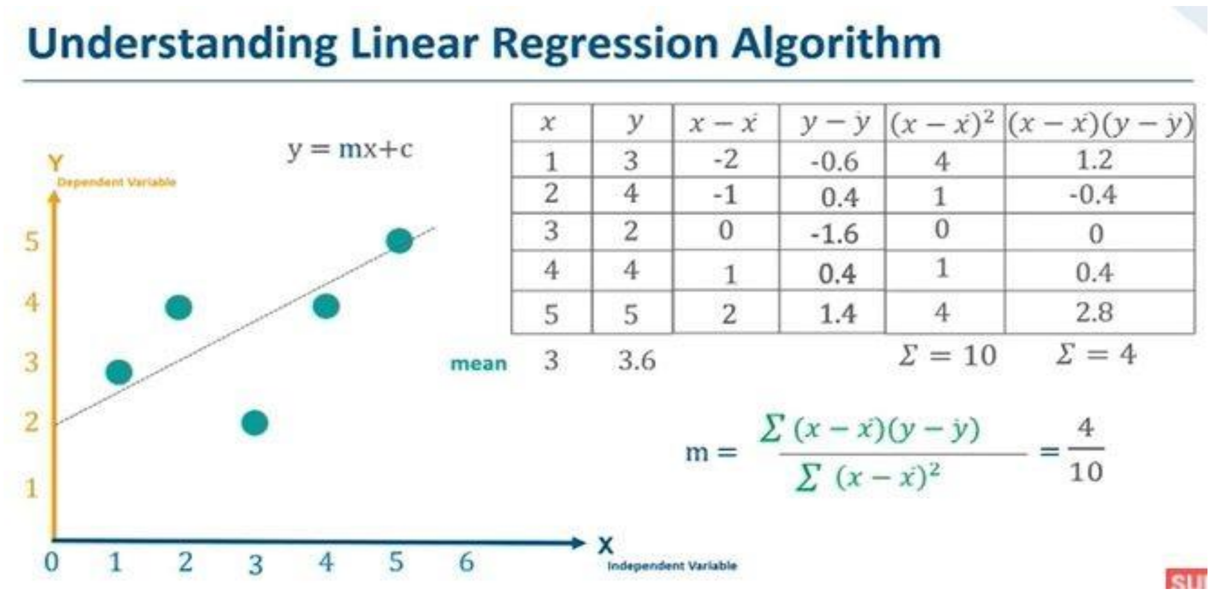
Understanding Linear Regression Algorithm

$y = mx + c$

x	y	$x - \bar{x}$	$y - \bar{y}$	$(x - \bar{x})^2$	$(x - \bar{x})(y - \bar{y})$
1	3	-2	-0.6	4	1.2
2	4	-1	0.4	1	-0.4
3	2	0	-1.6	0	0
4	4	1	0.4	1	0.4
5	5	2	1.4	4	2.8
mean 3	3.6			$\Sigma = 10$	$\Sigma = 4$

$$m = \frac{\Sigma (x - \bar{x})(y - \bar{y})}{\Sigma (x - \bar{x})^2}$$

All right, so the summation

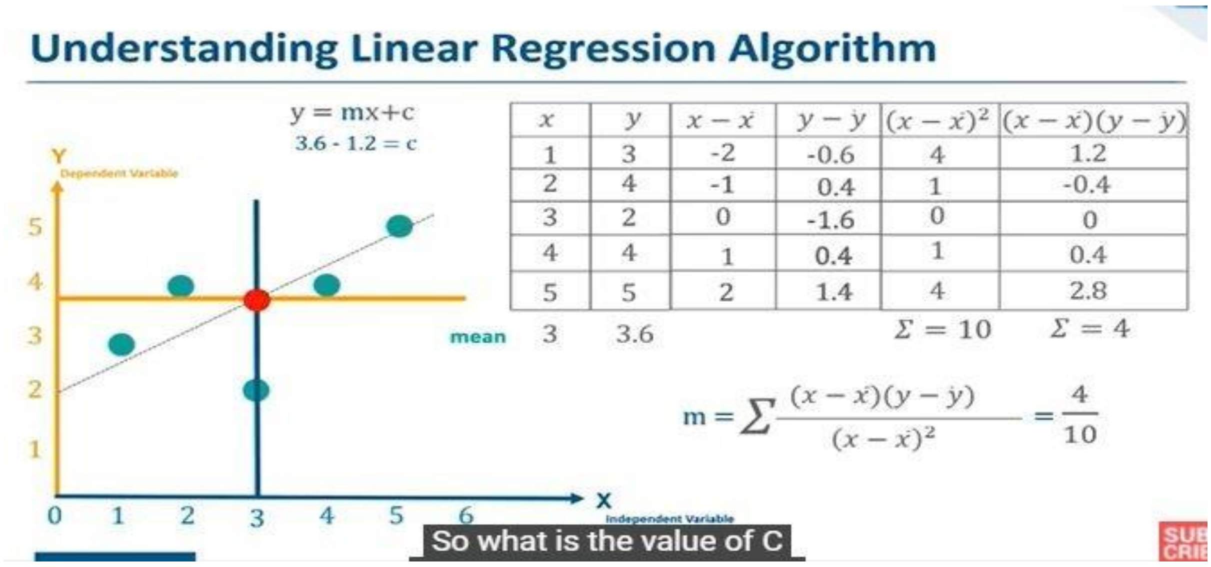


Understanding Linear Regression Algorithm

$y = mx + c$

x	y	$x - \bar{x}$	$y - \bar{y}$	$(x - \bar{x})^2$	$(x - \bar{x})(y - \bar{y})$
1	3	-2	-0.6	4	1.2
2	4	-1	0.4	1	-0.4
3	2	0	-1.6	0	0
4	4	1	0.4	1	0.4
5	5	2	1.4	4	2.8
mean 3	3.6			$\Sigma = 10$	$\Sigma = 4$

$$m = \frac{\Sigma (x - \bar{x})(y - \bar{y})}{\Sigma (x - \bar{x})^2} = \frac{4}{10}$$



Understanding Linear Regression Algorithm

$y = mx + c$

$3.6 - 1.2 = c$

x	y	$x - \bar{x}$	$y - \bar{y}$	$(x - \bar{x})^2$	$(x - \bar{x})(y - \bar{y})$
1	3	-2	-0.6	4	1.2
2	4	-1	0.4	1	-0.4
3	2	0	-1.6	0	0
4	4	1	0.4	1	0.4
5	5	2	1.4	4	2.8
mean 3	3.6			$\Sigma = 10$	$\Sigma = 4$

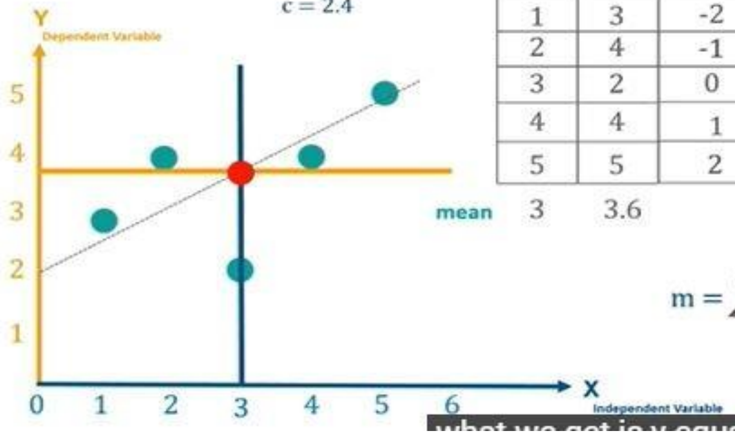
$$m = \Sigma \frac{(x - \bar{x})(y - \bar{y})}{(x - \bar{x})^2} = \frac{4}{10}$$

So what is the value of C

Understanding Linear Regression Algorithm

$y = mx + c$

$c = 2.4$



x	y	$x - \bar{x}$	$y - \bar{y}$	$(x - \bar{x})^2$	$(x - \bar{x})(y - \bar{y})$
1	3	-2	-0.6	4	1.2
2	4	-1	0.4	1	-0.4
3	2	0	-1.6	0	0
4	4	1	0.4	1	0.4
5	5	2	1.4	4	2.8
mean 3	3.6			$\Sigma = 10$	$\Sigma = 4$

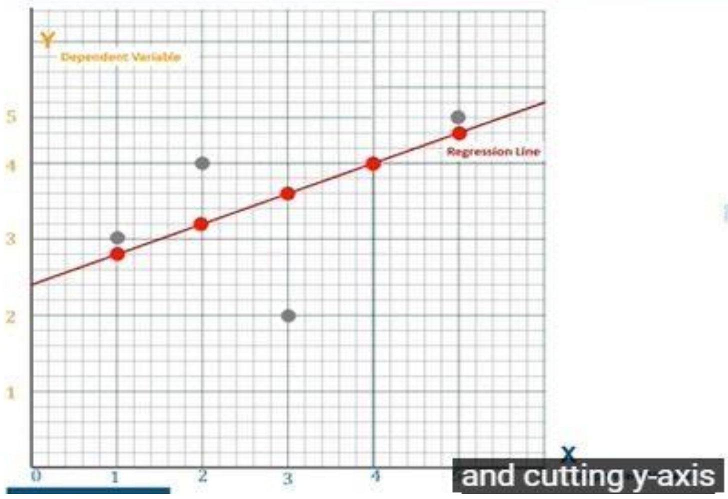
$$m = \sum \frac{(x - \bar{x})(y - \bar{y})}{(x - \bar{x})^2} = \frac{4}{10}$$

$m = 0.4$

$c = 2.4$

$y = 0.4x + 2.4$

Mean Square Error



$m = 0.4$

$c = 2.4$

$y = 0.4x + 2.4$

For given m = 0.4 & c = 2.4, lets predict values for y for x = {1,2,3,4,5}

$y = 0.4 \times 1 + 2.4 = 2.8$

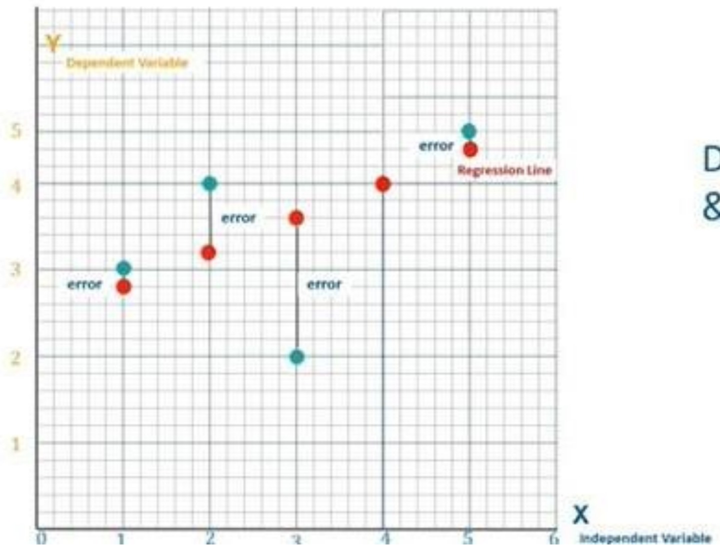
$y = 0.4 \times 2 + 2.4 = 3.2$

$y = 0.4 \times 3 + 2.4 = 3.6$

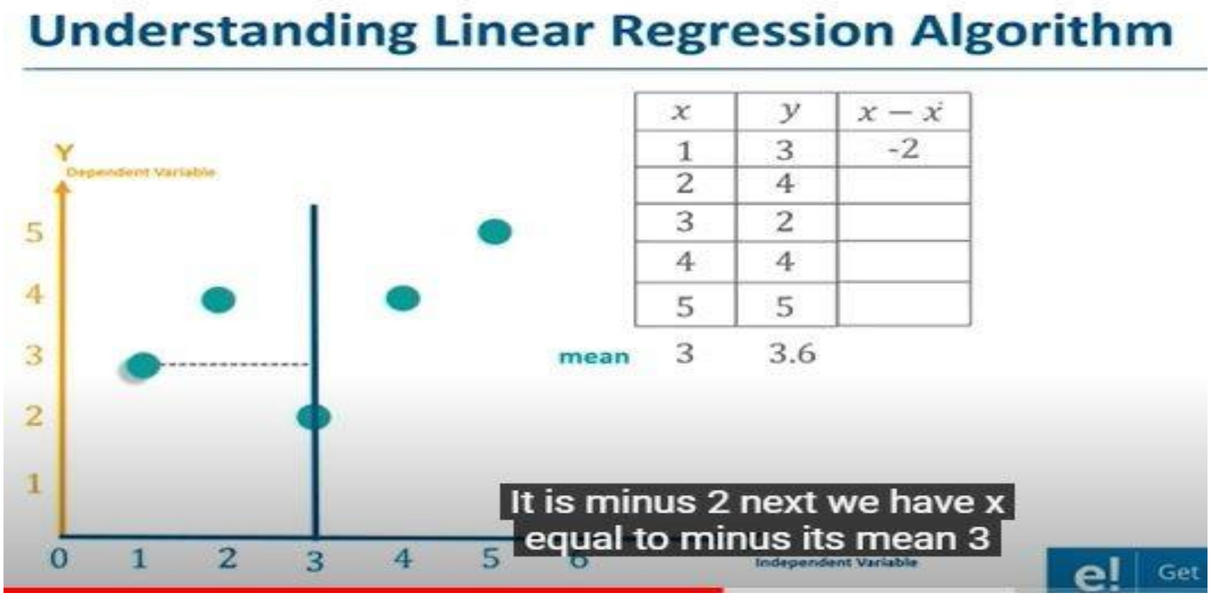
$y = 0.4 \times 4 + 2.4 = 4.0$

$y = 0.4 \times 5 + 2.4 = 4.4$

Mean Square Error



Distance between actual & predicted value

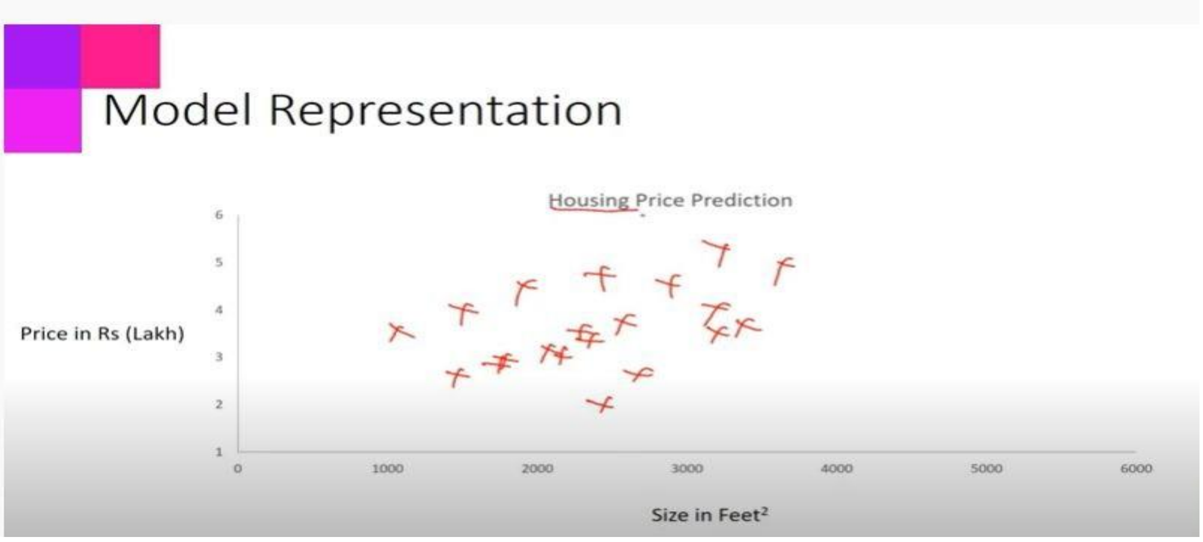


Understanding Linear Regression Algorithm

x	y	$x - \bar{x}$
1	3	-2
2	4	
3	2	
4	4	
5	5	
mean 3	3.6	

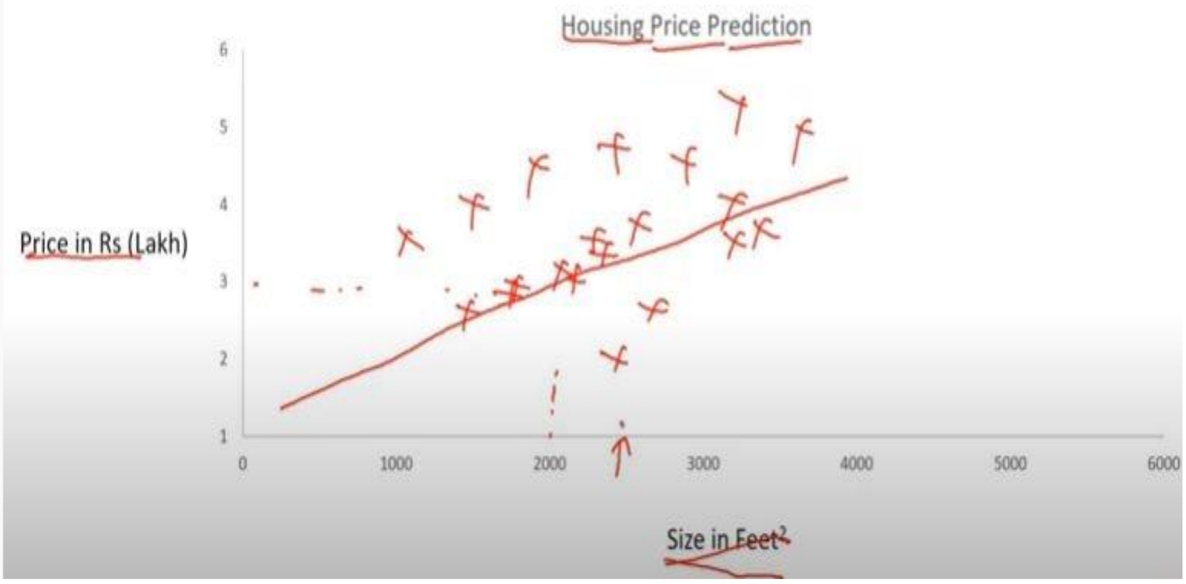
It is minus 2 next we have x equal to minus its mean 3

Model Representation

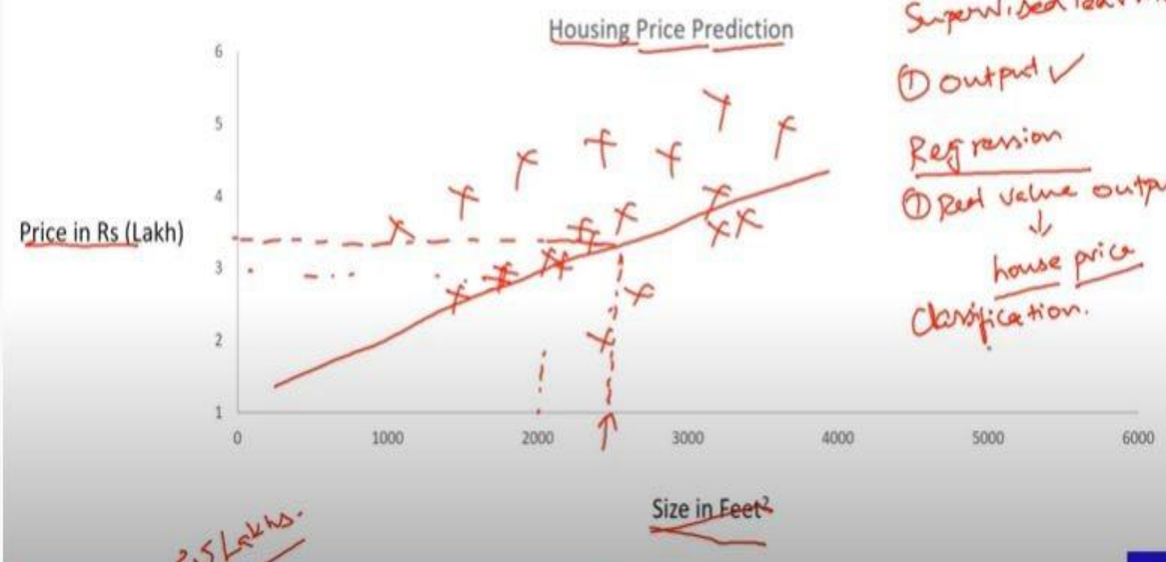


Model Representation

Model Representation



Model Representation



Supervised learning

① output ✓

Regression

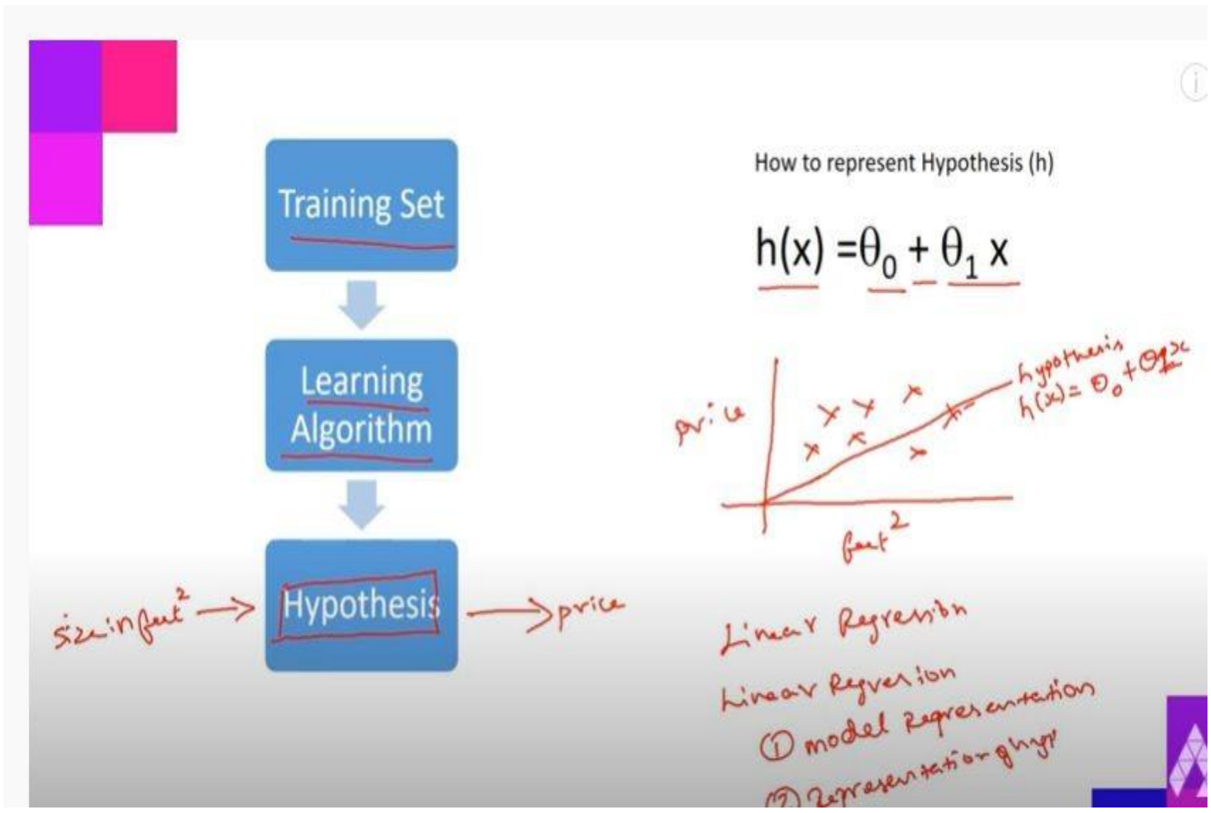
① Real value output

↓

house price

Classification.

3.5 Lakhs.



1. Why was Machine Learning Introduced? ...
2. What are Different Types of Machine Learning algorithms? ...
3. What is Supervised Learning? ...
4. What is Unsupervised Learning? ...
5. How is neuroscience related to machine learning?
6. How does machine learning work similar to a brain?
7. What is the main difference between human brain and a computer?
8. What is concept learning task in machine learning?
9. What is hypothesis concept learning?
10. What are the objectives of machine learning?
11. What is the goal of concept learning Search task?
12. Solved Numerical Example (Candidate Elimination Algorithm):

Example	**Citations**	**Size**	**InLibrary**	**Price**	**Editions**	**Buy**
1	Some	Small	No	Affordable	One	No
2	Many	Big	No	Expensive	Many	Yes
3	Many	Medium	No	Expensive	Few	Yes
4	Many	Small	No	Affordable	Many	Yes

1. What is linear discriminant analysis in machine learning?
2. What is the purpose of linear discriminant analysis?
3. How do you do linear discriminant analysis? What is a Linear Regression?
4. Can you list out the critical assumptions of linear regression?
5. What is Heteroscedasticity?
6. What is the primary difference between R square and adjusted R square?
7. Can you list out the formulas to find RMSE and MSE?

UNIT – II LINEAR MODELS

Multi – Layer Perceptron in Practice

To solve real problems, MLP find solutions to four different types of problem: regression, classification, time-series prediction, and data compression.

2.1 Amount of Training Data

2. 2Number of Hidden Layers

2.3 When to Stop Learning

2.1 Amount of Training Data

- For the MLP - (L+1)×M +(M +1)×N weights, where L,M,N are the number of nodes in the input, hidden, and output layers, respectively.
- +1s -> bias nod es, which is **adjustable weights**. Huge number of adjustable parameters that we need to set during the training phase.
- Setting the values of adjustable weights is the job of the back-propagation algorithm, which is driven by the errors coming from the training data.
- More training data is the better for learning, algorithm takes to learn increases.
- Minimum amount of data required is, it depends on the problem.
- Use a number of training examples is 10 times the number of weights.
- If MLP has large number of examples, so neural network training has expensive operation
- Two hidden layers is the need for normal MLP learning. This result can be strengthened by showing mathematically that one hidden layer with lots of hidden nodes is sufficient. This is known as the Universal Approximation Theorem
- Training networks with different numbers of hidden nodes and then choosing the one that gives the best results.
- Using back-propagation algorithm for a network with as many layers, it harder to keep track of which weights are being update.
- The basic idea is that by combining sigmoid functions we can generate ridge-like functions, and by combining ridge-like functions, generate functions with a unique maximum.
- By combining these and transforming them using another layer of neurons, we obtain localized

response (a ‘bump’ function), and any functional mapping can be approximated to arbitrary accuracy using a linear combination of such bumps.

- Two hidden layers are sufficient to compute these bump functions for different inputs, and so if the function learns (approximate) is continuous, the network can compute it.

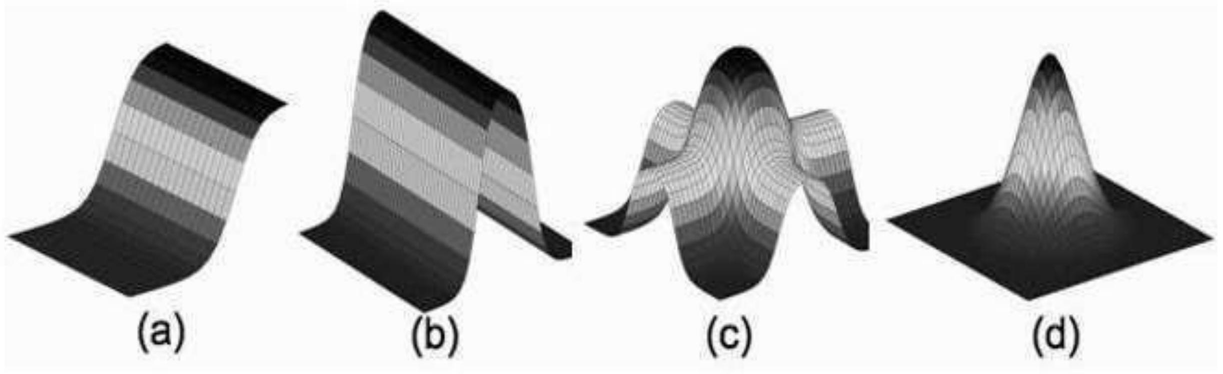
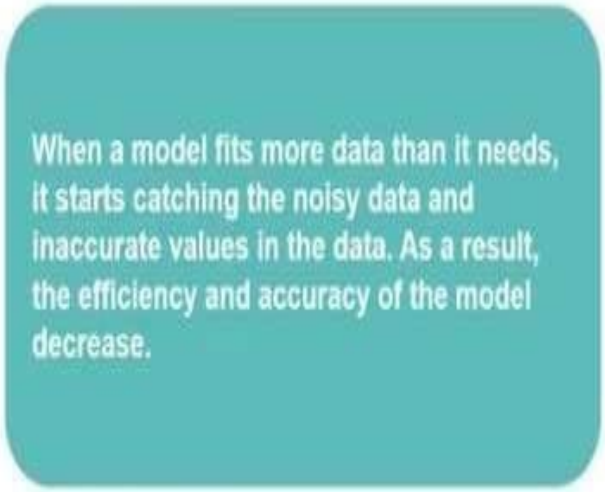


FIGURE 4.9 The learning of the MLP can be shown as the output of a single sigmoidal neuron (a), which can be added to others, including reversed ones, to get a hill shape (b). Adding another hill at 90° produces a bump (c) , which can be sharpened to any extent we want (d), with the bumps added together in the output layer. Thus the MLP learns a local representation of individual inputs.

2.3 When to Stop Learning

- The training of the MLP requires that the algorithm runs over the entire data set many times, with the weights changing as the network makes errors in each iteration.
- Set predefined N number of iteration, the network has overfitted (Overfitting (overtraining): when the NN learns too many I/O examples it may end up memorizingthe training data) or learn sufficiently and when it stops, some predefined minimum error is reached that means algorithm never terminates. Sum of squares errors during training
- At some stage the error on the validation set will start increasing again, because the network has stopped learning about the function that generated the data, andstarted to learn about the noise. At this stage we stop the training. This technique is called early stopping


When a model fits more data than it needs, it starts catching the noisy data and inaccurate values in the data. As a result, the efficiency and accuracy of the model decrease.

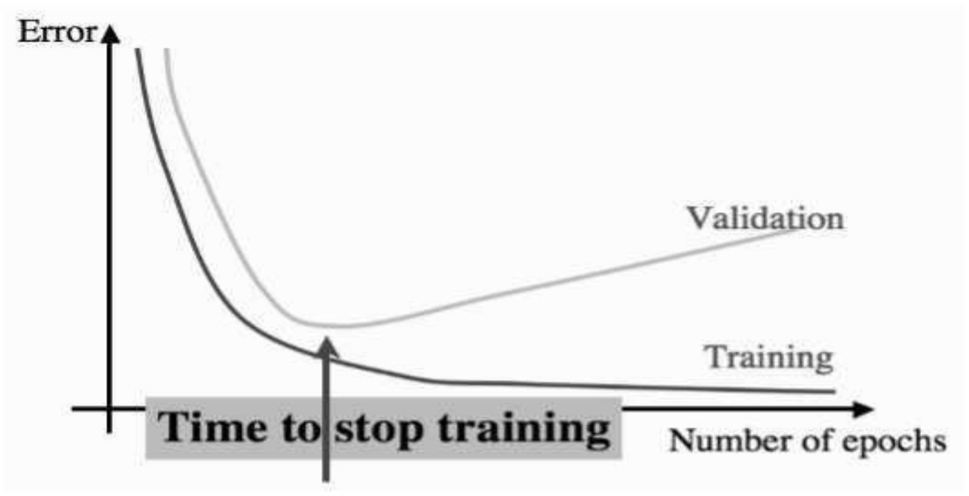
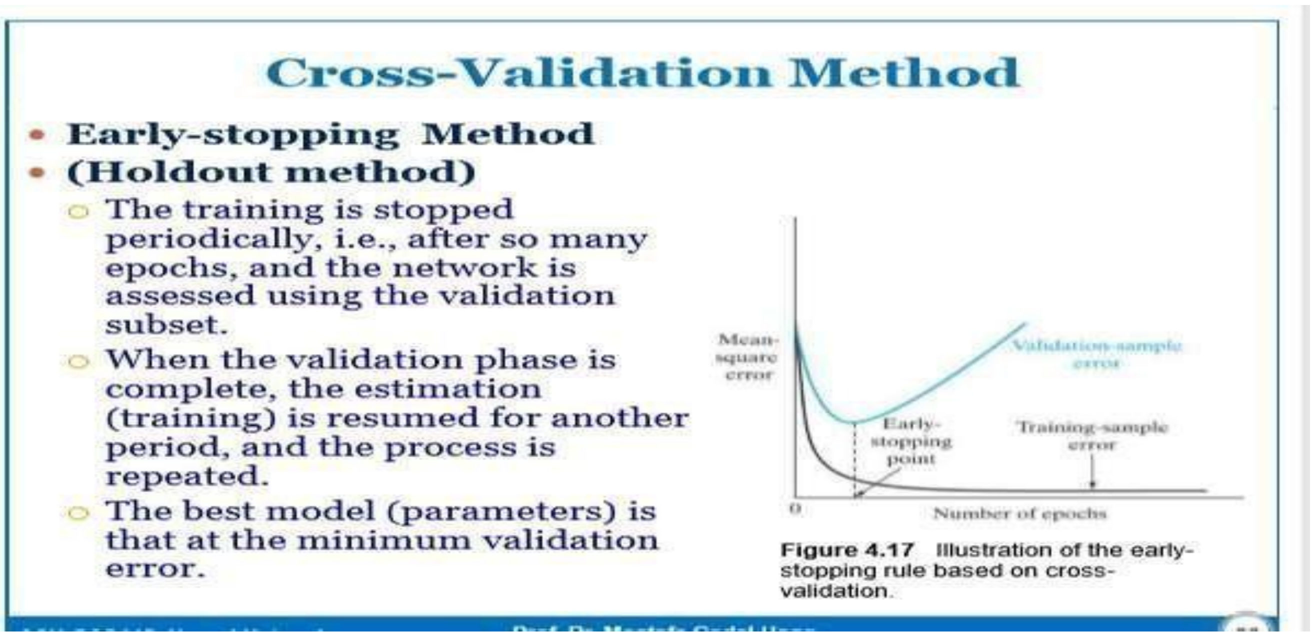


FIGURE 4.11 The effect of overfitting on the training and validation error curves, with the point at which early stopping will stop the learning marked.



Cross-Validation Method

- **Early-stopping Method**
- **(Holdout method)**
  - The training is stopped periodically, i.e., after so many epochs, and the network is assessed using the validation subset.
  - When the validation phase is complete, the estimation (training) is resumed for another period, and the process is repeated.
  - The best model (parameters) is that at the minimum validation error.

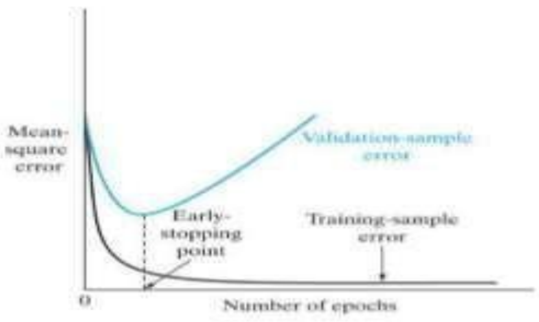
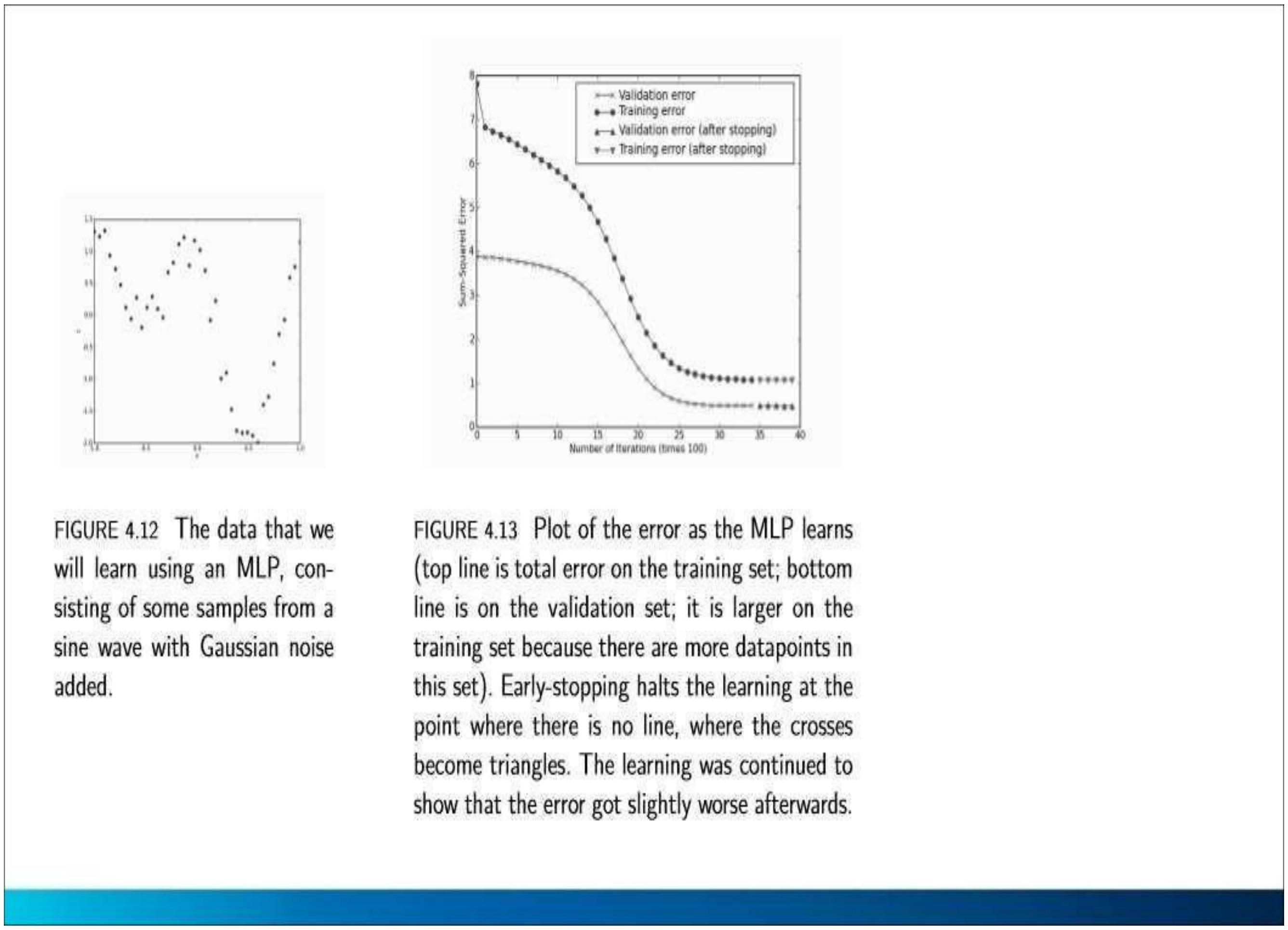


Figure 4.17 Illustration of the early-stopping rule based on cross-validation.



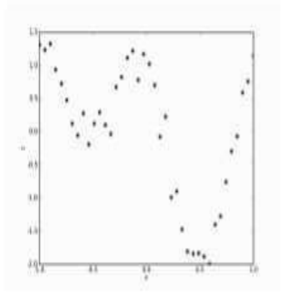


FIGURE 4.12 The data that we will learn using an MLP, consisting of some samples from a sine wave with Gaussian noise added.

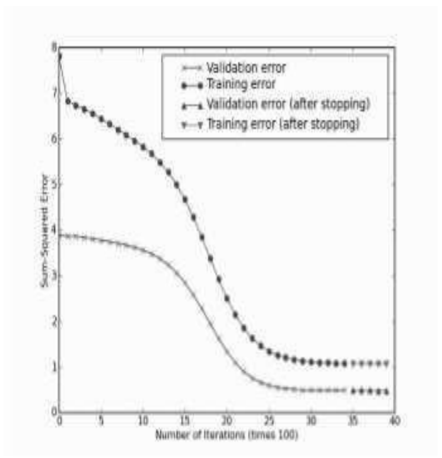


FIGURE 4.13 Plot of the error as the MLP learns (top line is total error on the training set; bottom line is on the validation set; it is larger on the training set because there are more datapoints in this set). Early-stopping halts the learning at the point where there is no line, where the crosses become triangles. The learning was continued to show that the error got slightly worse afterwards.

A Regression Problem: Find the values of any inputs and train the data Function –sin wave

Train an MLP on the data. There is one input value, x and one output value t, so the neural network will have one input and one output. Before getting started, we need to normalise the data, and then separate the data into training, testing, and validation sets. In given example there are only 40 datapoints and use half of them as the training set Split the data in the ratio 50:25:25 by using the odd-numbered elements as training data, the even-numbered ones that do not divide by 4 for testing, and the rest for validation Construct a network with three nodes in the hidden layer, and run it for 101 iterations with a learning rate of 0.25. The output: Iteration: 0 Error: 12.3704163654

Iteration: 100 Error: 8.2075961385 so that the network is learning, since the error is decreasing.

To do two things: how many hidden nodes we need, and decide how long to train the network for.

In order to solve the first problem, need to test out different networks and see which get lower errors, but to do that properly need to know when to stop training. Solve the second problem first, which is to implement early stopping. keep track of the validation error and stop when it starts to increase.

2.3.2 Classification with the MLP

$$\text{Class is: } \begin{cases} C_1 & \text{if } y \leq -0.5 \\ C_2 & \text{if } -0.5 < y \leq 0 \\ C_3 & \text{if } 0 < y \leq 0.5 \\ C_4 & \text{if } y > 0.5 \end{cases}$$

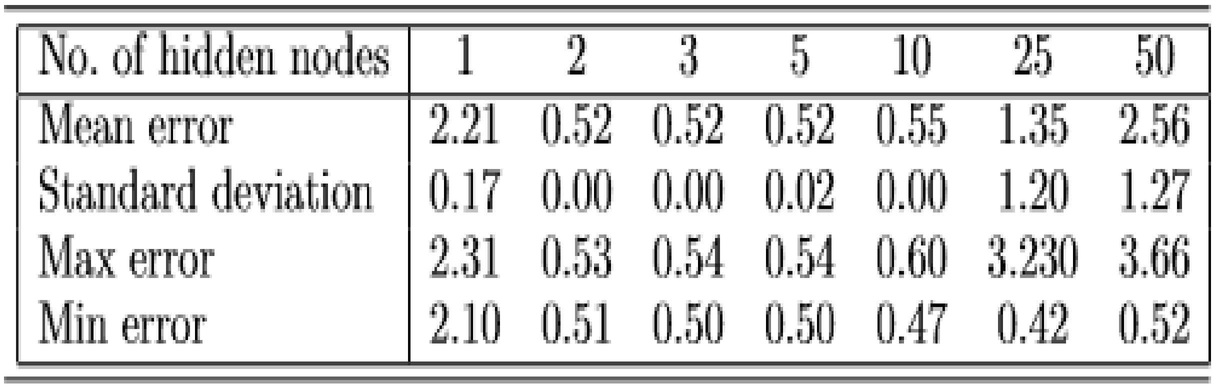
Figure gives an example of the output of running the function.

It plots the training and validation errors.

- The point at which early stopping makes the learning finish is the point where there is a missing validation datapoint.

- The validation error did not improve after that, and so early stopping found the correct point .

- Problem of finding the right size of network.
- Each network size is run 10 times, and the average is monitored.
- The following table shows the results of doing this, reporting the sum-of-squares validation error, for a few different sizes of network:



No. of hidden nodes	1	2	3	5	10	25	50
Mean error	2.21	0.52	0.52	0.52	0.55	1.35	2.56
Standard deviation	0.17	0.00	0.00	0.02	0.00	1.20	1.27
Max error	2.31	0.53	0.54	0.54	0.60	3.230	3.66
Min error	2.10	0.51	0.50	0.50	0.47	0.42	0.52

Couple of choices for the outputs.

- First-use a single linear node for the output, y, and put some thresholds on theactivation value of that node. For example, for a four-class problem,

- Close to a boundary, say y = 0.5? It belongs to class C3, close to the boundary in theoutput.

- A more suitable output encoding is called1-of-N encoding. e.g., (0,0,1,0) means that the correct resultis the 3rd class out of 4 Element yk of the output vector that is the largest element of y(in mathematical notation, pick the yk for which yk > yj Aj ≠ k;A means for all.

  Two output neurons will have identical largest output values.

  - This is known as the hard-max activation function (since the neuron with the highest activation is chosen to fire and the rest are ignored).
  - Two-class classification, 90% of our data belongs to class 1. (This scan happens: for example, in medical data, most tests are negativein general.)
  - There is an alternative solution, known as novelty detection, which is to train the data on the data in the negative class only, and to assume that anything that looks different to that is a positive example.

2.3.3 A Classification Example: The Iris Dataset

- Three types of iris (flower) by the length and width of the sepals and petals and is called iris.
- **stext1 = 'Iris-setosa' ,stext2 = 'Iris-versicolor' ,stext3 = 'Iris-virginica' .**
- Need to separate the data into training, testing, and validationsets.
- There are 150 examples in the dataset, and they are split evenly amongst the three classes, so the three classes are the same size.
- Split them into ½ training, and ¼ each testing and validation.50 are class1, 50class 2, etc.,

2.3.4 Time – Series Prediction

- Stock market, disease pattern, seasonal variation

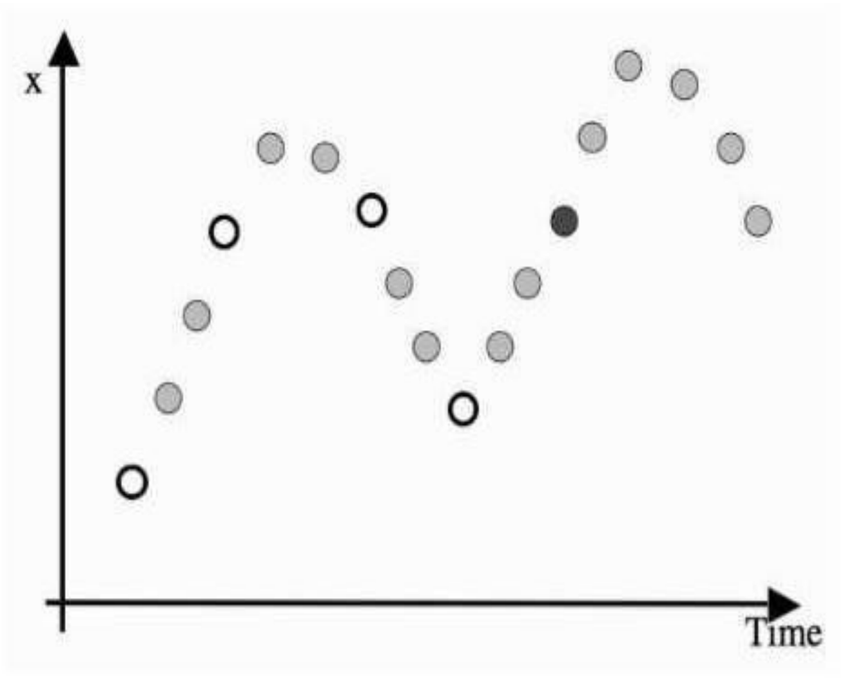


FIGURE 4.14 Part of a time-series plot, showing the datapoints and the meanings of τ and k.

Decision boundary

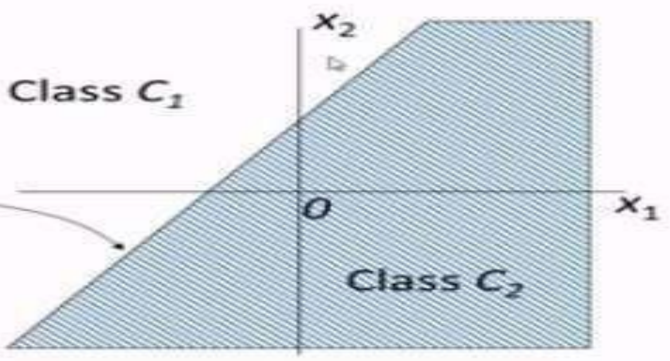
- The hyper-plane

$$\sum_{i=1}^{m} w_i x_i + b = 0$$

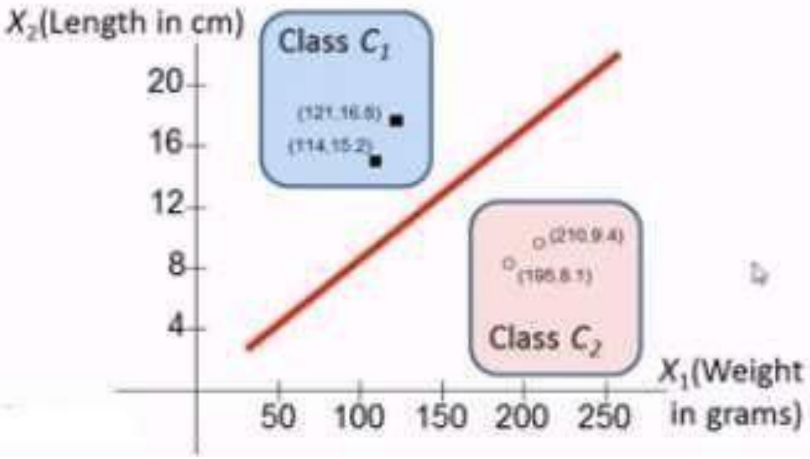
or

$$w_1 x_1 + w_2 x_2 + b = 0$$

is the decision boundary for a two class classification problem.



	Weight (grams)	Length (cm)
Fruit 1 (Class C1)	121	16.8
	114	15.2
Fruit 2 (Class C2)	210	9.4
	195	8.1



simplest form of a neural network used for the classification of patterns said to be linearly separable. Basically, it consists of a single neuron with adjustable synaptic weights and bias.

$$\left.\begin{array}{l} w_1(0) = -30,\ w_2(0) = 300, \\ b(0) = 50,\ \eta = 0.01 \end{array}\right\}\textit{given}$$

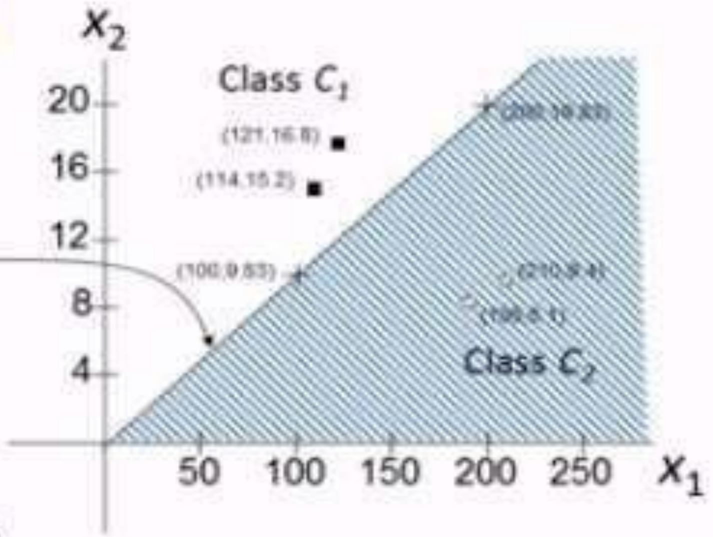
Therefore the Initial Decision Boundary for this example is:

$$w_1x_1 + w_2x_2 + b = 0$$

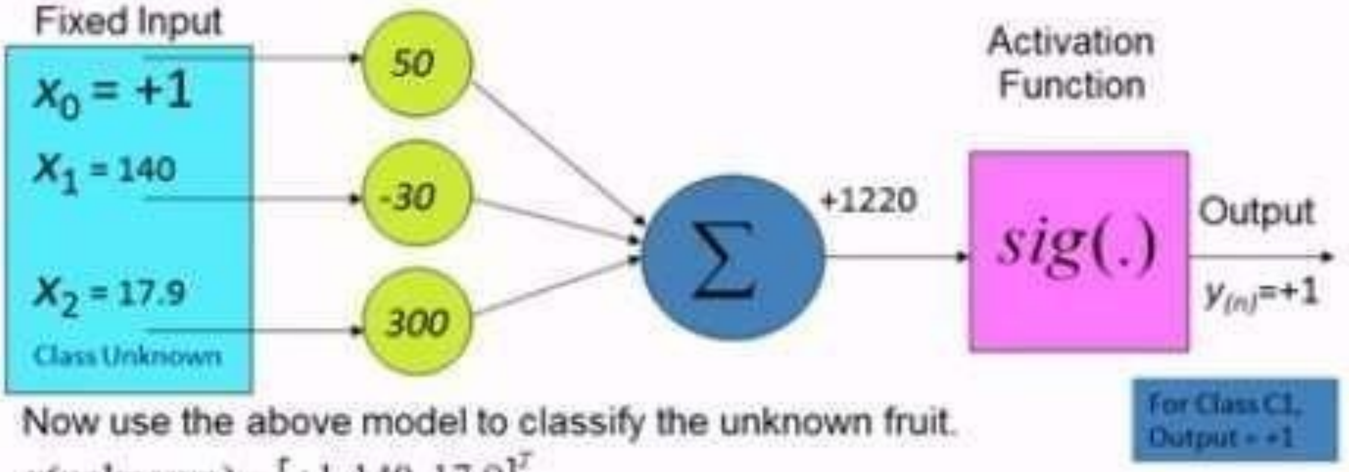
$$-30x_1 + 300x_2 + 50 = 0$$

$$x_1 = 100,\ x_2 = \frac{30\times 100 - 50}{300} = 9.83$$

$$x_1 = 200,\ x_2 = \frac{30\times 200 - 50}{300} = 19.83$$



Initial hyper-plane does separate the two classes.



Now use the above model to classify the unknown fruit.

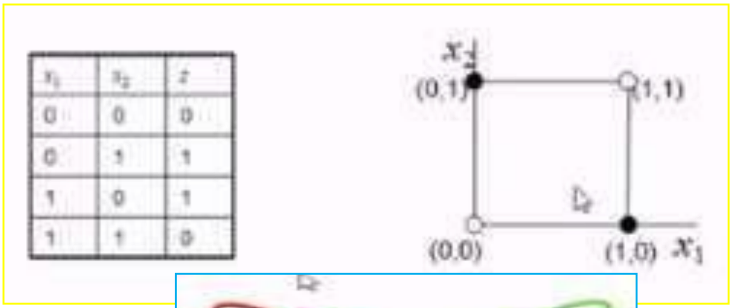
$$\mathbf{x}(\text{unknown}) = [+1, 140, 17.9]^T$$

$$\mathbf{w}(3) = [50, -30, 300]^T$$

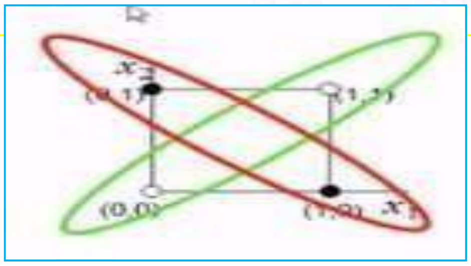
$$y(\text{unknown}) = \operatorname{sgn}\left(\mathbf{w}^T(3)\mathbf{x}(\text{unknown})\right) = \operatorname{sgn}(50\times 1 - 30\times 140 + 300\times 17.9)$$

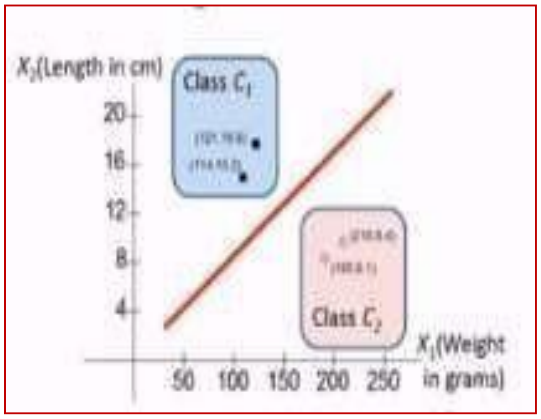
$$= \operatorname{sgn}(1220) = +1$$

$\therefore$ this unknown fruit belongs to the class C_1.



x_1	x_2	z
0	0	0
0	1	1
1	0	1
1	1	0





If there is a solution to be found then the single layer perceptron learning algorithm will find it.

- It can separate classes that lie either side of a straight line easily.
- But in reality, division between classes are much more complex.
- Take for example the classical exclusive-or (XOR) problem.
- XOR logic function has two inputs and one output.
- It has limited set of functions Decision boundaries must be hyperplanes
- It can only perfectly separate linearly separable data

- We consider this as a problem in which we want the perceptron to learn to solve:
- Output 1 if x1 is on and x2 is off, or is x2 is on and x1 is off, otherwise output a 0.
- This appears a simple problem but there is no linear solution and this problem is linearly inseparable.

The XOR Problem

- **A two-layer Network to solve the XOR Problem**

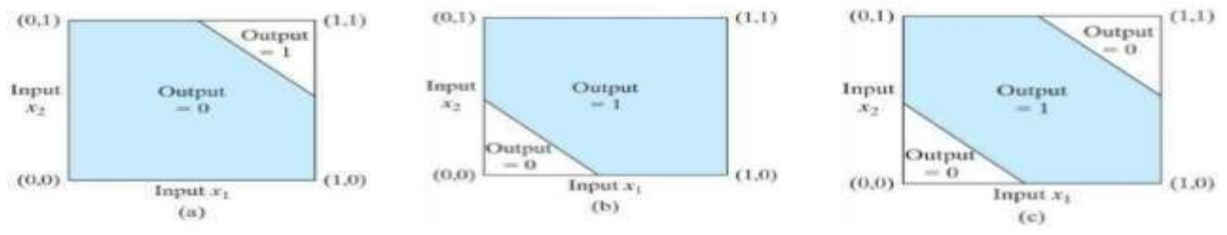
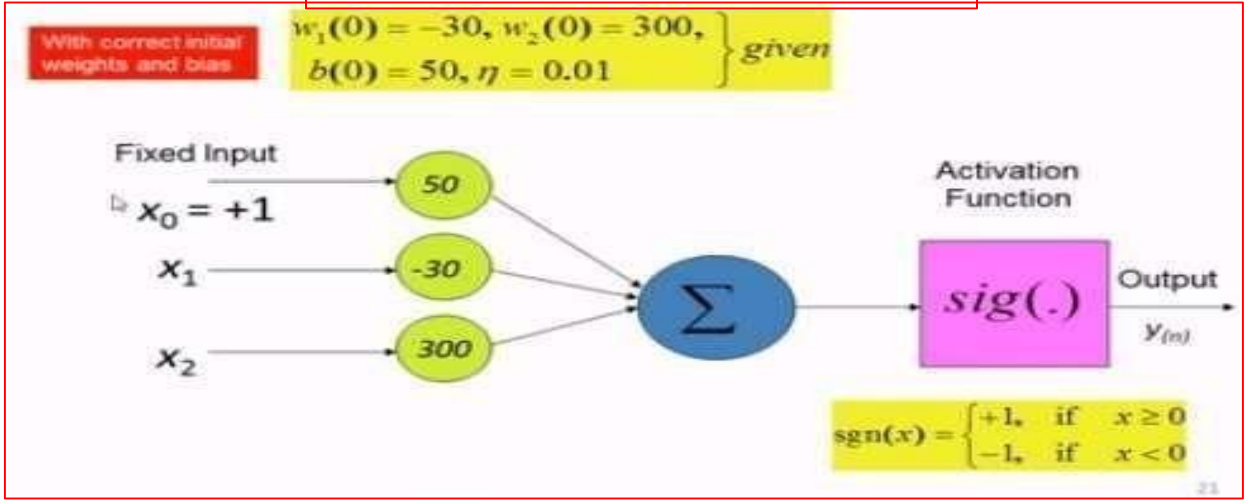
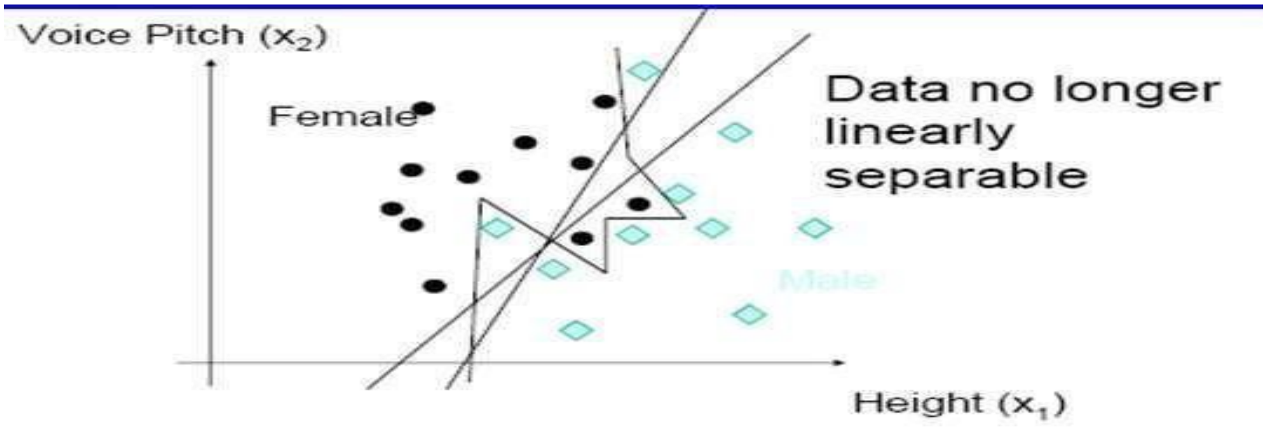


Figure 4.9 (a) Decision boundary constructed by hidden neuron 1 of the network in Fig. 4.8. (b) Decision boundary constructed by hidden neuron 2 of the network. (c) Decision boundaries constructed by the complete network.

Example single layer perceptron


With correct initial weights and bias
$w_1(0) = -30, w_2(0) = 300, b(0) = 50, \eta = 0.01$ } given
Fixed Input
$x_0 = +1$
x_1
x_2
50
-30
300
Σ
Activation Function
$sig(.)$
Output
$y_{(n)}$
$\mathrm{sgn}(x) = \begin{cases} +1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases}$

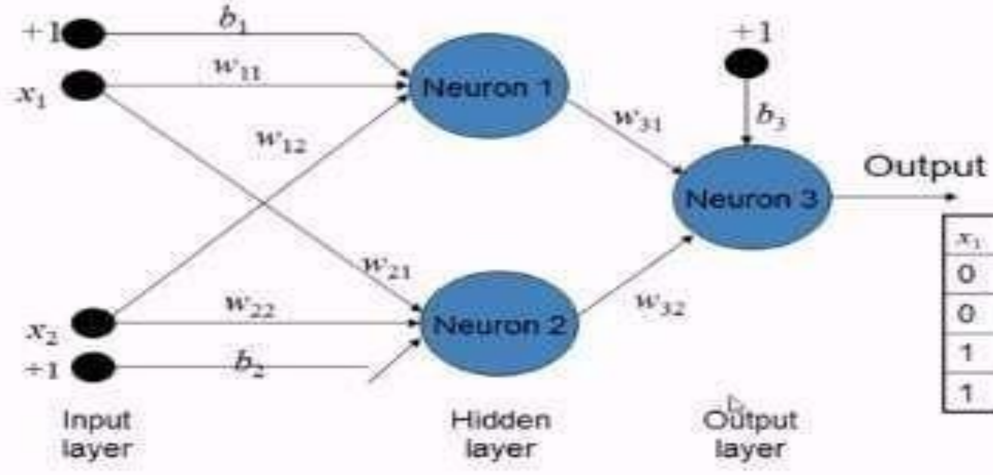


Data no longer linearly separable

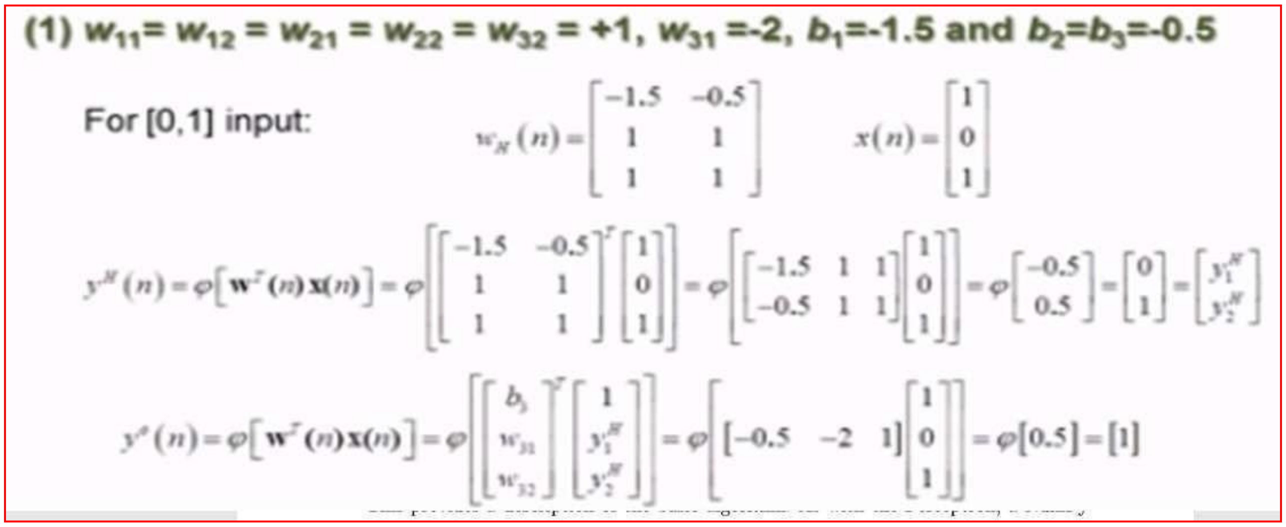
What is a good decision boundary ?

XOR Problem – with two layers

Now consider the following network with two inputs, 1 hidden layer and 1 output layer.



x_1	x_2	z
0	0	0
0	1	1
1	0	1
1	1	0



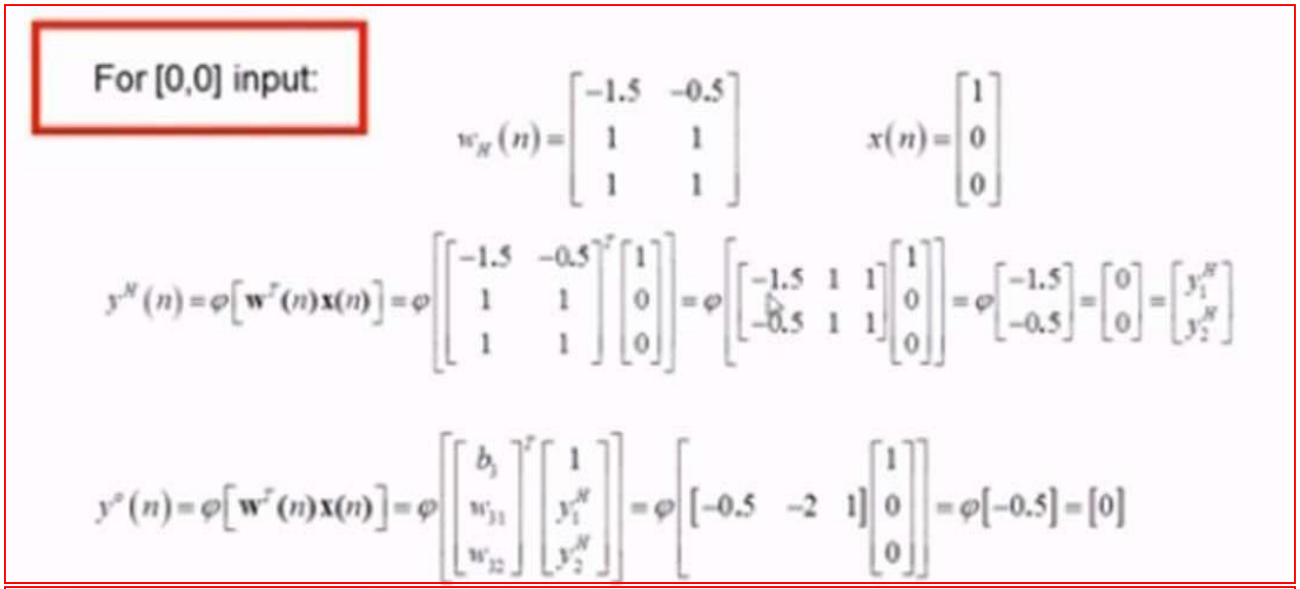
(1) $w_{11} = w_{12} = w_{21} = w_{22} = w_{32} = +1$, $w_{31} = -2$, $b_1 = -1.5$ and $b_2 = b_3 = -0.5$

For [0,1] input:

$$w_H(n) = \begin{bmatrix} -1.5 & -0.5 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \qquad x(n) = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$y^H(n) = \varphi\left[\mathbf{w}^T(n)\mathbf{x}(n)\right] = \varphi\left[\begin{bmatrix} -1.5 & -0.5 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}^T \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}\right] = \varphi\left[\begin{bmatrix} -1.5 & 1 & 1 \\ -0.5 & 1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}\right] = \varphi\begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} y_1^H \\ y_2^H \end{bmatrix}$$

$$y^o(n) = \varphi\left[\mathbf{w}^T(n)\mathbf{x}(n)\right] = \varphi\left[\begin{bmatrix} b_3 \\ w_{31} \\ w_{32} \end{bmatrix}^T \begin{bmatrix} 1 \\ y_1^H \\ y_2^H \end{bmatrix}\right] = \varphi\left[\begin{bmatrix} -0.5 & -2 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}\right] = \varphi[0.5] = [1]$$

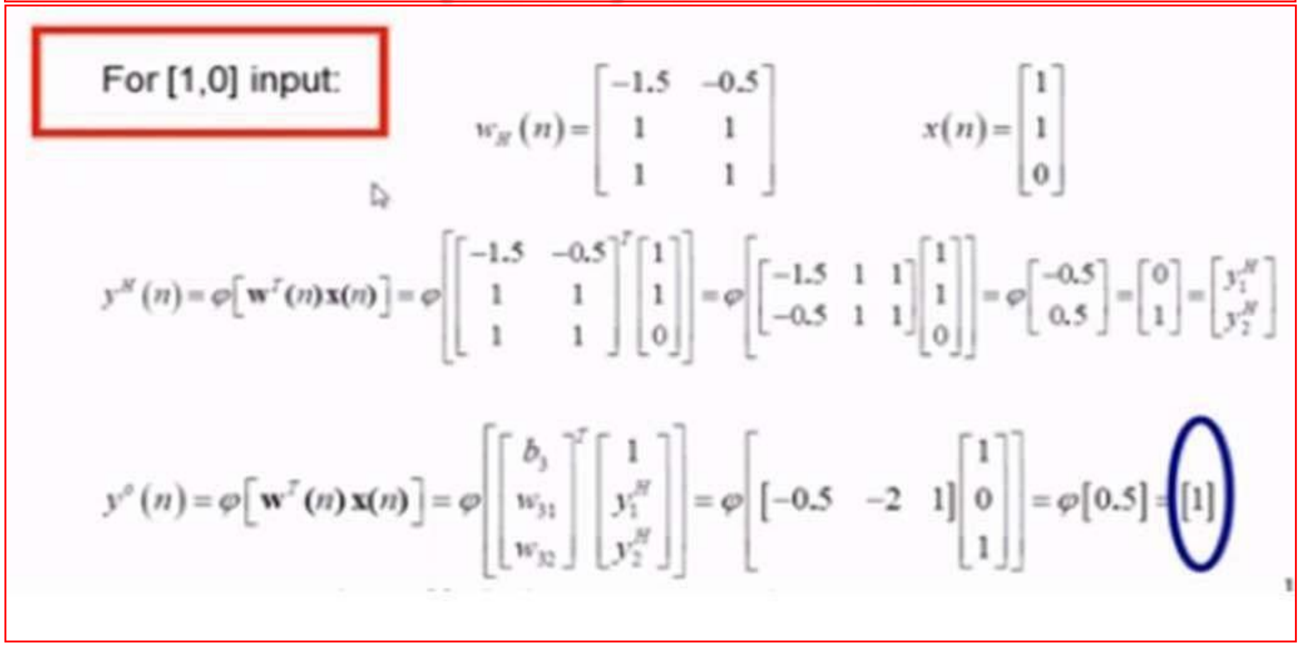


For [0,0] input:

$$w_H(n) = \begin{bmatrix} -1.5 & -0.5 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \qquad x(n) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$y^H(n) = \varphi\left[\mathbf{w}^T(n)\mathbf{x}(n)\right] = \varphi\left[\begin{bmatrix} -1.5 & -0.5 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}^T \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\right] = \varphi\left[\begin{bmatrix} -1.5 & 1 & 1 \\ -0.5 & 1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\right] = \varphi\begin{bmatrix} -1.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} y_1^H \\ y_2^H \end{bmatrix}$$

$$y^o(n) = \varphi\left[\mathbf{w}^T(n)\mathbf{x}(n)\right] = \varphi\left[\begin{bmatrix} b_3 \\ w_{31} \\ w_{32} \end{bmatrix}^T \begin{bmatrix} 1 \\ y_1^H \\ y_2^H \end{bmatrix}\right] = \varphi\left[\begin{bmatrix} -0.5 & -2 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\right] = \varphi[-0.5] = [0]$$

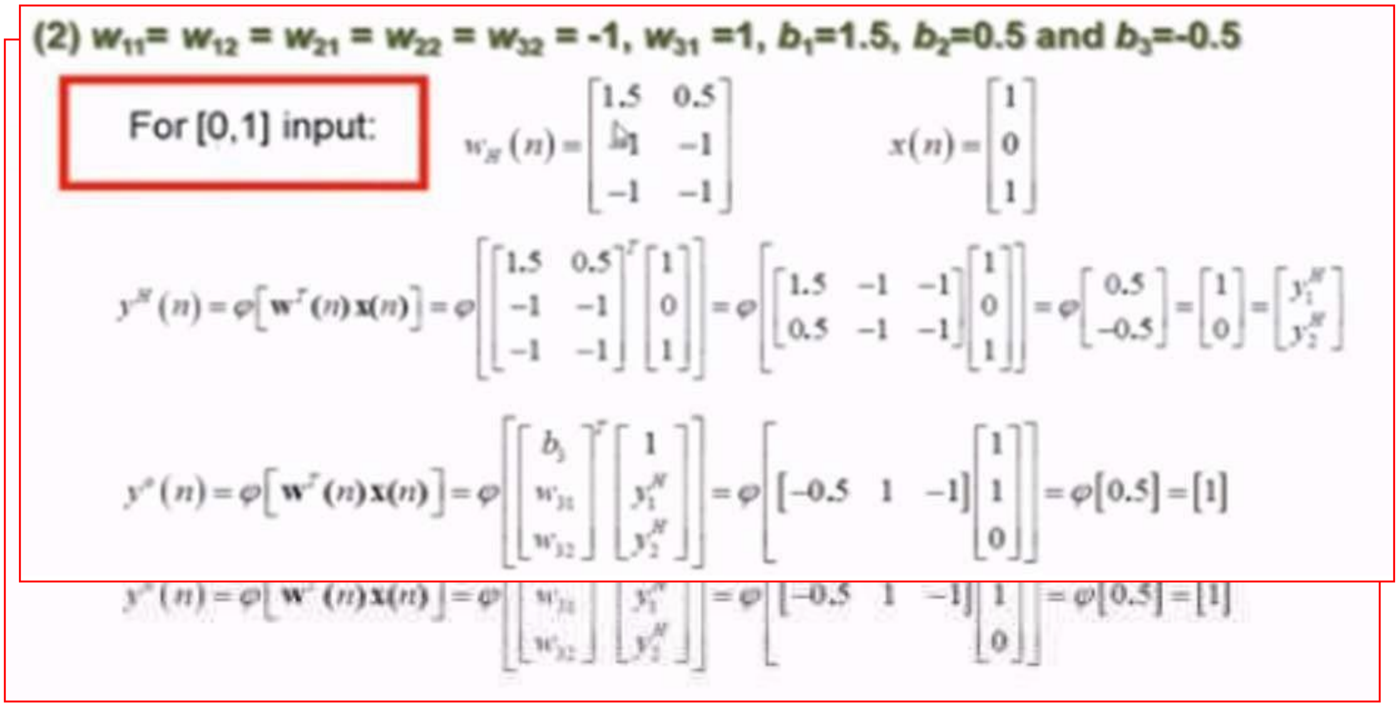


For [1,0] input:

$$w_H(n) = \begin{bmatrix} -1.5 & -0.5 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \qquad x(n) = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

$$y^H(n) = \varphi\left[\mathbf{w}^T(n)\mathbf{x}(n)\right] = \varphi\left[\begin{bmatrix} -1.5 & -0.5 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}^T \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}\right] = \varphi\left[\begin{bmatrix} -1.5 & 1 & 1 \\ -0.5 & 1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}\right] = \varphi\begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} y_1^H \\ y_2^H \end{bmatrix}$$

$$y^o(n) = \varphi\left[\mathbf{w}^T(n)\mathbf{x}(n)\right] = \varphi\left[\begin{bmatrix} b_3 \\ w_{31} \\ w_{32} \end{bmatrix}^T \begin{bmatrix} 1 \\ y_1^H \\ y_2^H \end{bmatrix}\right] = \varphi\left[\begin{bmatrix} -0.5 & -2 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}\right] = \varphi[0.5] = [1]$$



(2) $w_{11} = w_{12} = w_{21} = w_{22} = w_{32} = -1$, $w_{31} = 1$, $b_1 = 1.5$, $b_2 = 0.5$ and $b_3 = -0.5$

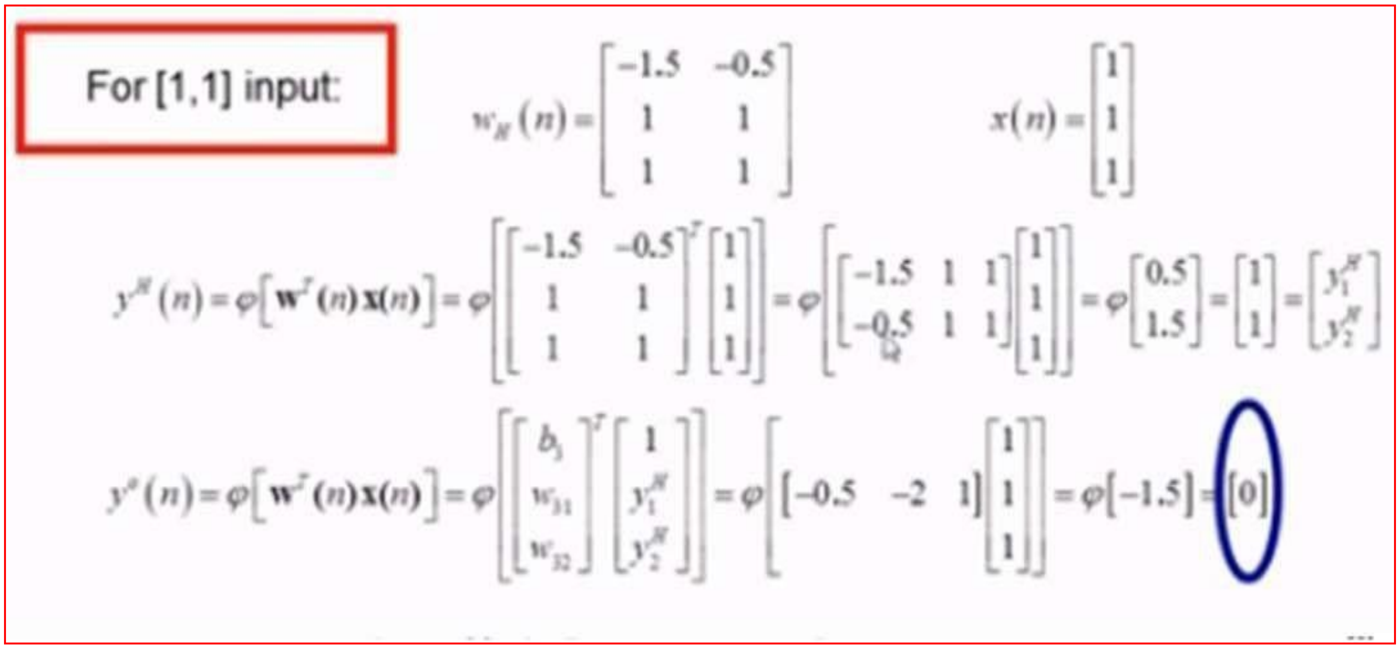
For [0,1] input:

$$w_H(n) = \begin{bmatrix} 1.5 & 0.5 \\ -1 & -1 \\ -1 & -1 \end{bmatrix} \qquad x(n) = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$y^H(n) = \varphi\left[\mathbf{w}^T(n)\mathbf{x}(n)\right] = \varphi\left[\begin{bmatrix} 1.5 & 0.5 \\ -1 & -1 \\ -1 & -1 \end{bmatrix}^T \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}\right] = \varphi\left[\begin{bmatrix} 1.5 & -1 & -1 \\ 0.5 & -1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}\right] = \varphi\begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} y_1^H \\ y_2^H \end{bmatrix}$$

$$y^o(n) = \varphi\left[\mathbf{w}^T(n)\mathbf{x}(n)\right] = \varphi\left[\begin{bmatrix} b_3 \\ w_{31} \\ w_{32} \end{bmatrix}^T \begin{bmatrix} 1 \\ y_1^H \\ y_2^H \end{bmatrix}\right] = \varphi\left[\begin{bmatrix} -0.5 & 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}\right] = \varphi[0.5] = [1]$$

$$y^o(n) = \varphi\left[\mathbf{w}^T(n)\mathbf{x}(n)\right] = \varphi\left[\begin{bmatrix} w_{31} \\ w_{32} \end{bmatrix} \begin{bmatrix} y_1^H \\ y_2^H \end{bmatrix}\right] = \varphi\left[\begin{bmatrix} -0.5 & 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix}\right] = \varphi[0.5] = [1]$$

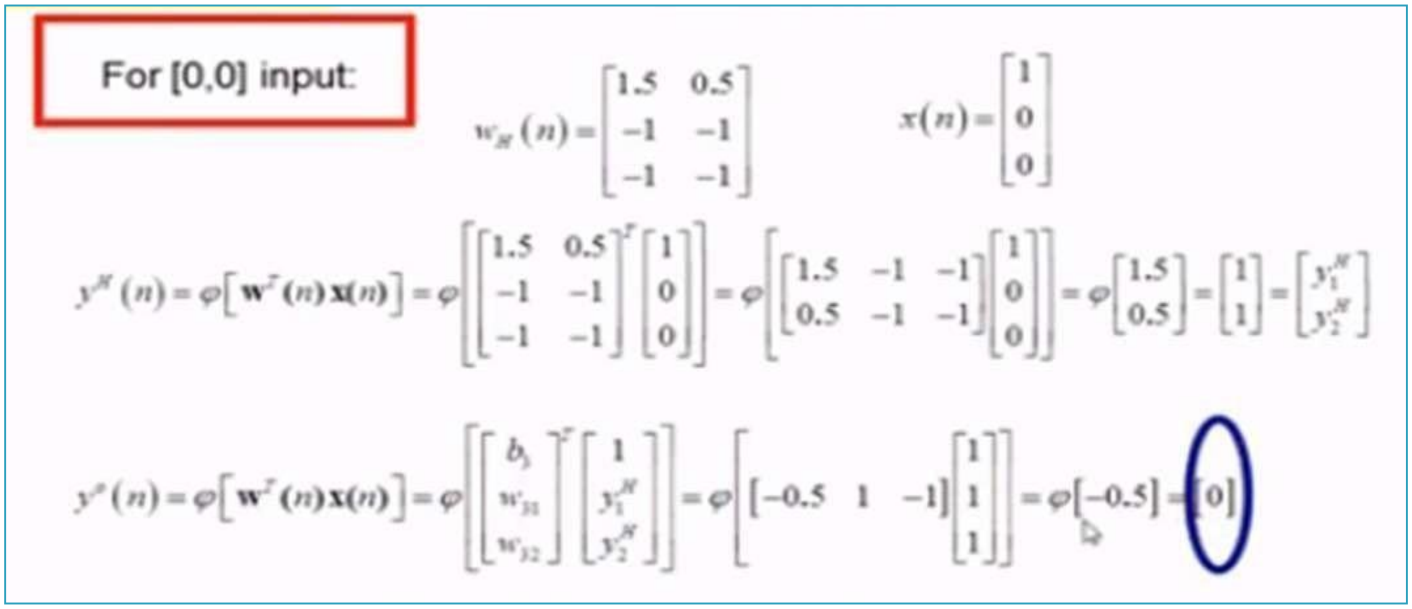


For [1,1] input:

$$w_H(n) = \begin{bmatrix} -1.5 & -0.5 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \qquad x(n) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$y^H(n) = \varphi\left[\mathbf{w}^T(n)\mathbf{x}(n)\right] = \varphi\left[\begin{bmatrix} -1.5 & -0.5 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}^T \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\right] = \varphi\left[\begin{bmatrix} -1.5 & 1 & 1 \\ -0.5 & 1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\right] = \varphi\begin{bmatrix} 0.5 \\ 1.5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} y_1^H \\ y_2^H \end{bmatrix}$$

$$y^o(n) = \varphi\left[\mathbf{w}^T(n)\mathbf{x}(n)\right] = \varphi\left[\begin{bmatrix} b_3 \\ w_{31} \\ w_{32} \end{bmatrix}^T \begin{bmatrix} 1 \\ y_1^H \\ y_2^H \end{bmatrix}\right] = \varphi\left[\begin{bmatrix} -0.5 & -2 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\right] = \varphi[-1.5] = \boxed{[0]}$$

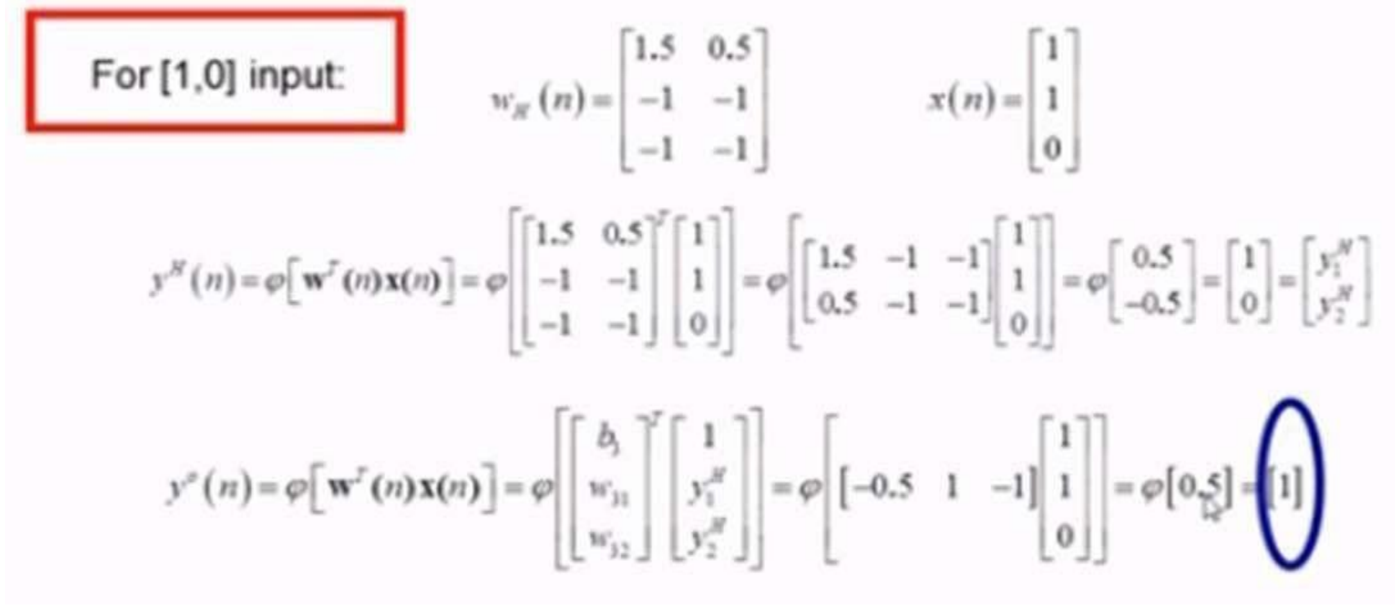


For [0,0] input:

$$w_H(n)=\begin{bmatrix}1.5 & 0.5\\ -1 & -1\\ -1 & -1\end{bmatrix}\qquad x(n)=\begin{bmatrix}1\\0\\0\end{bmatrix}$$

$$y^H(n)=\varphi\left[\mathbf{w}^T(n)\mathbf{x}(n)\right]=\varphi\left[\begin{bmatrix}1.5 & 0.5\\ -1 & -1\\ -1 & -1\end{bmatrix}^T\begin{bmatrix}1\\0\\0\end{bmatrix}\right]=\varphi\left[\begin{bmatrix}1.5 & -1 & -1\\ 0.5 & -1 & -1\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix}\right]=\varphi\begin{bmatrix}1.5\\0.5\end{bmatrix}=\begin{bmatrix}1\\1\end{bmatrix}=\begin{bmatrix}y_1^H\\y_2^H\end{bmatrix}$$

$$y^o(n)=\varphi\left[\mathbf{w}^T(n)\mathbf{x}(n)\right]=\varphi\left[\begin{bmatrix}b_3\\w_{31}\\w_{32}\end{bmatrix}^T\begin{bmatrix}1\\y_1^H\\y_2^H\end{bmatrix}\right]=\varphi\left[\begin{bmatrix}-0.5 & 1 & -1\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix}\right]=\varphi[-0.5]=\boxed{[0]}$$

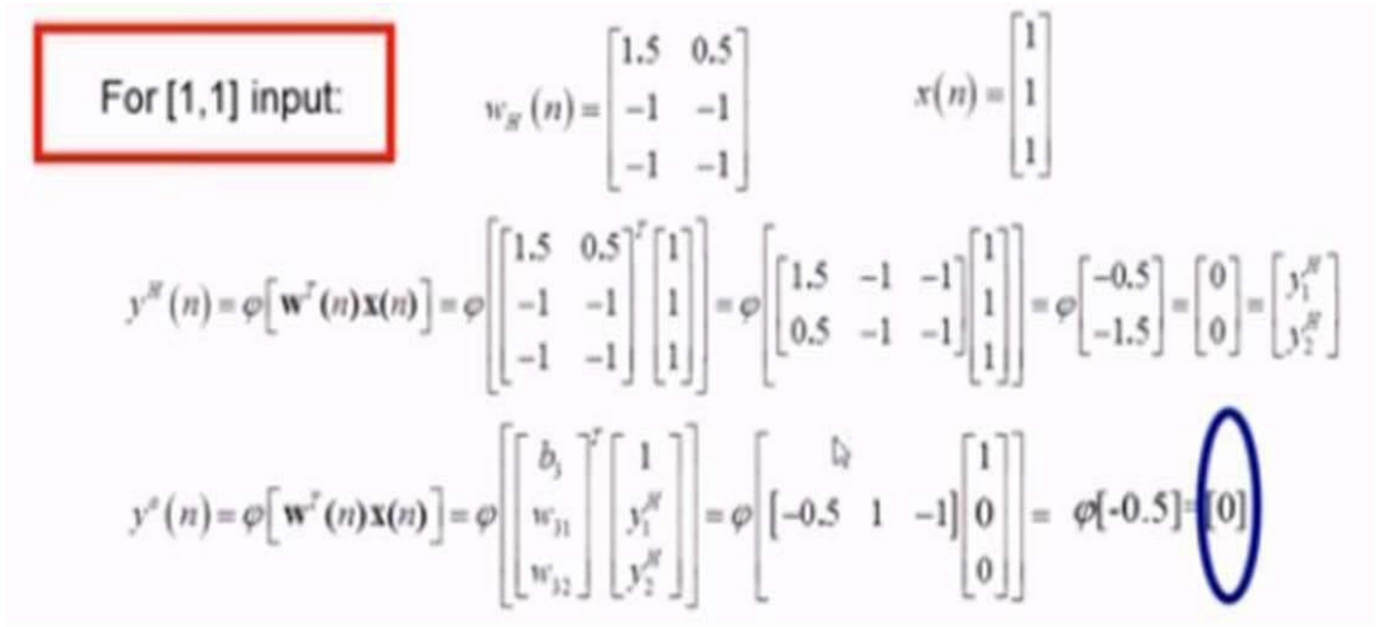


For [1,0] input:

$$w_H(n)=\begin{bmatrix}1.5 & 0.5\\ -1 & -1\\ -1 & -1\end{bmatrix}\qquad x(n)=\begin{bmatrix}1\\1\\0\end{bmatrix}$$

$$y^H(n)=\varphi\left[\mathbf{w}^T(n)\mathbf{x}(n)\right]=\varphi\left[\begin{bmatrix}1.5 & 0.5\\ -1 & -1\\ -1 & -1\end{bmatrix}^T\begin{bmatrix}1\\1\\0\end{bmatrix}\right]=\varphi\left[\begin{bmatrix}1.5 & -1 & -1\\ 0.5 & -1 & -1\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix}\right]=\varphi\begin{bmatrix}0.5\\-0.5\end{bmatrix}=\begin{bmatrix}1\\0\end{bmatrix}=\begin{bmatrix}y_1^H\\y_2^H\end{bmatrix}$$

$$y^o(n)=\varphi\left[\mathbf{w}^T(n)\mathbf{x}(n)\right]=\varphi\left[\begin{bmatrix}b_3\\w_{31}\\w_{32}\end{bmatrix}^T\begin{bmatrix}1\\y_1^H\\y_2^H\end{bmatrix}\right]=\varphi\left[\begin{bmatrix}-0.5 & 1 & -1\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix}\right]=\varphi[0.5]=\boxed{[1]}$$



For [1,1] input:

$$w_H(n)=\begin{bmatrix}1.5 & 0.5\\ -1 & -1\\ -1 & -1\end{bmatrix}\qquad x(n)=\begin{bmatrix}1\\1\\1\end{bmatrix}$$

$$y^H(n)=\varphi\left[\mathbf{w}^T(n)\mathbf{x}(n)\right]=\varphi\left[\begin{bmatrix}1.5 & 0.5\\ -1 & -1\\ -1 & -1\end{bmatrix}^T\begin{bmatrix}1\\1\\1\end{bmatrix}\right]=\varphi\left[\begin{bmatrix}1.5 & -1 & -1\\ 0.5 & -1 & -1\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix}\right]=\varphi\begin{bmatrix}-0.5\\-1.5\end{bmatrix}=\begin{bmatrix}0\\0\end{bmatrix}=\begin{bmatrix}y_1^H\\y_2^H\end{bmatrix}$$

$$y^o(n)=\varphi\left[\mathbf{w}^T(n)\mathbf{x}(n)\right]=\varphi\left[\begin{bmatrix}b_3\\w_{31}\\w_{32}\end{bmatrix}^T\begin{bmatrix}1\\y_1^H\\y_2^H\end{bmatrix}\right]=\varphi\left[\begin{bmatrix}-0.5 & 1 & -1\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix}\right]=\varphi[-0.5]=\boxed{[0]}$$

Multi-Layer Perceptron(MLP)

- Architecture of a multilayer perceptron

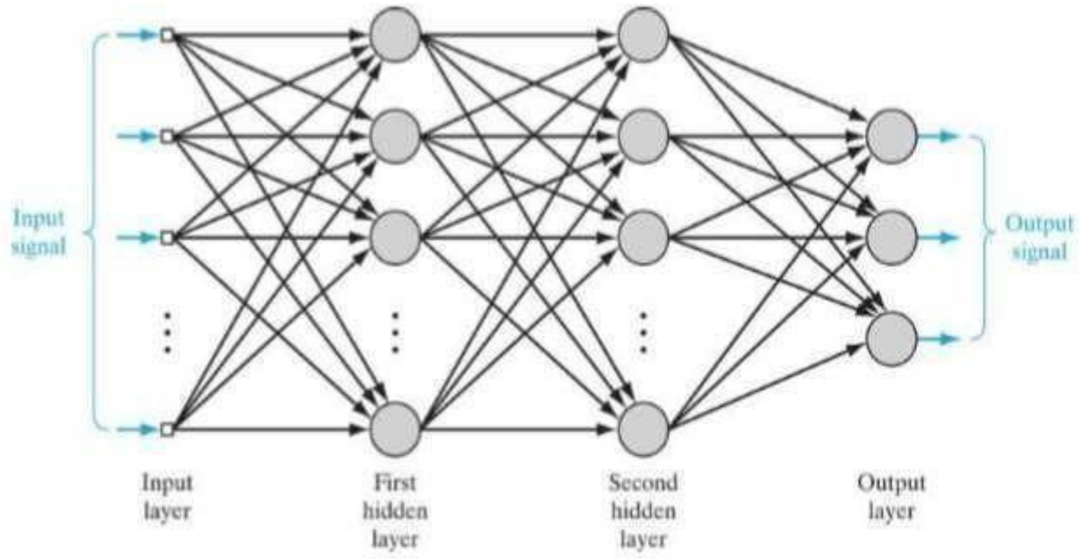
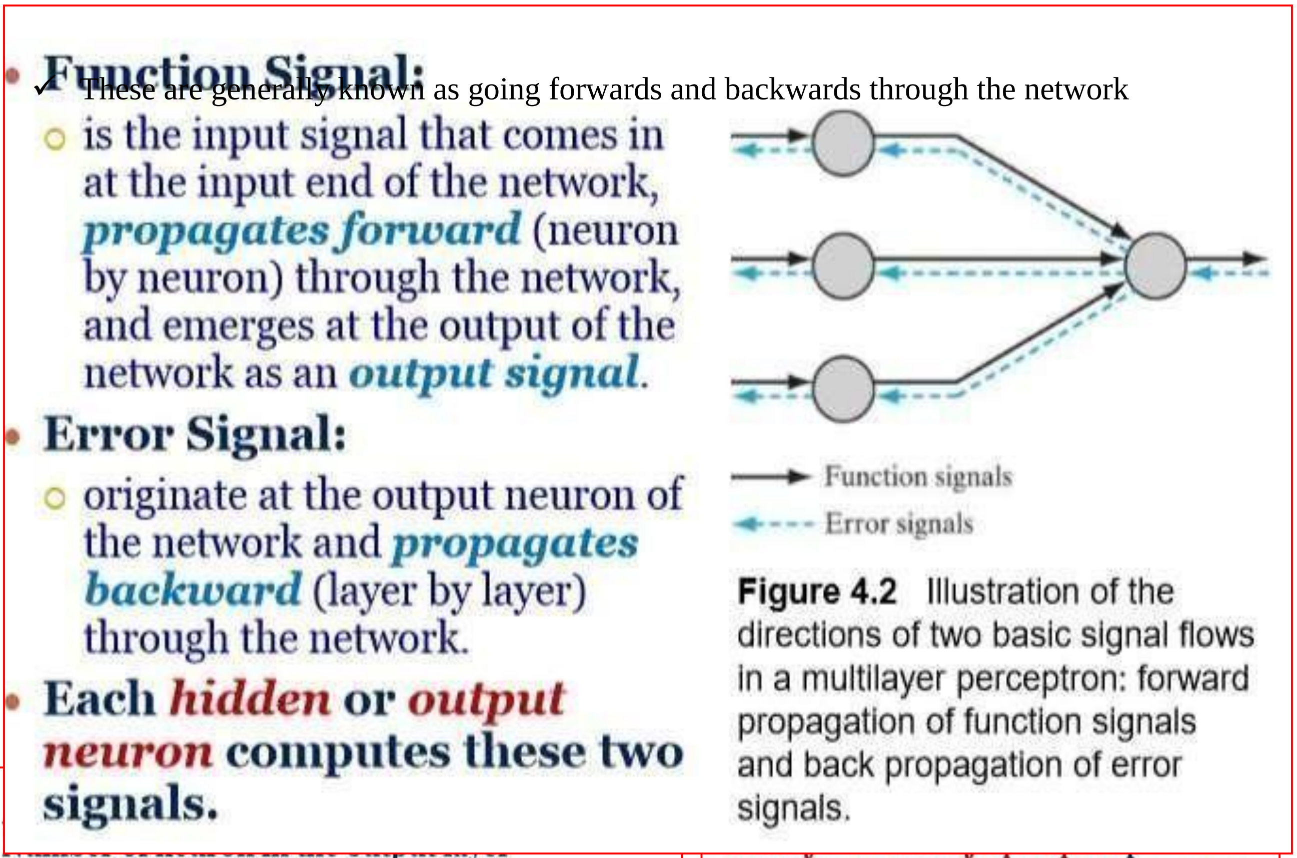


Figure 4.1 Architectural graph of a multilayer perceptron with two hidden layers.

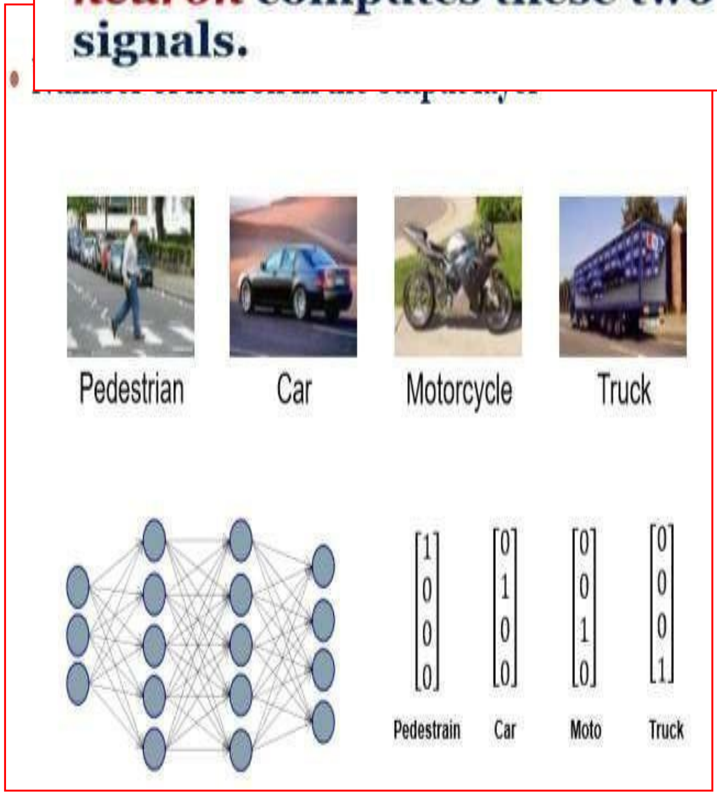
- ✓ MLP have been applied to solve some difficult problems.
- ✓ This consist of input layer, one or more hidden layer and an output layer.
- ✓ The training of the network is done by the highly popular algorithm known as error back propagation algorithm
- ✓ This algorithm is based on the error correcting learning rule. Basically, there are two passes through the different layers of the network: forward pass and backward pass.
- ✓ The Perceptron, training the MLP consists of two parts:
- ✓ working out what the outputs are for the given inputs and current weights
- ✓ Update the weight according to the error, which is a function of the difference between the outputs and the targets.

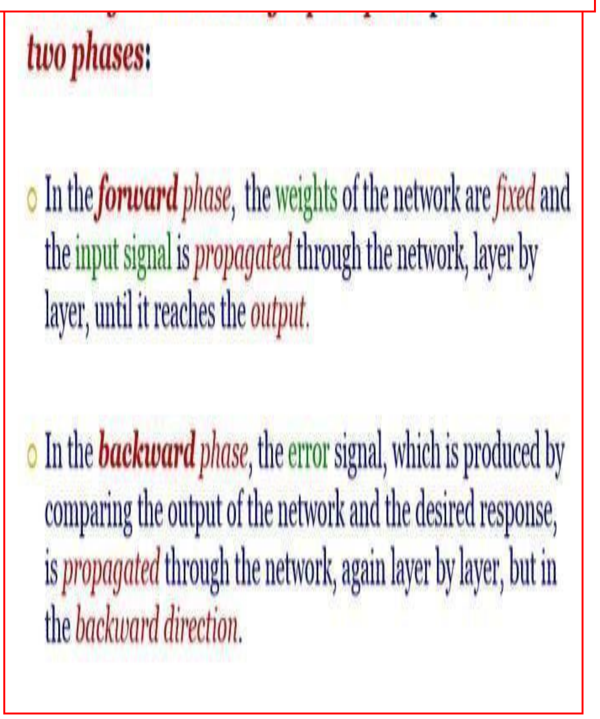
✓ These are generally known as going forwards and backwards through the network



- **Function Signal:**
  - is the input signal that comes in at the input end of the network, ***propagates forward*** (neuron by neuron) through the network, and emerges at the output of the network as an ***output signal***.
- **Error Signal:**
  - originate at the output neuron of the network and ***propagates backward*** (layer by layer) through the network.
- **Each *hidden* or *output neuron* computes these two signals.**

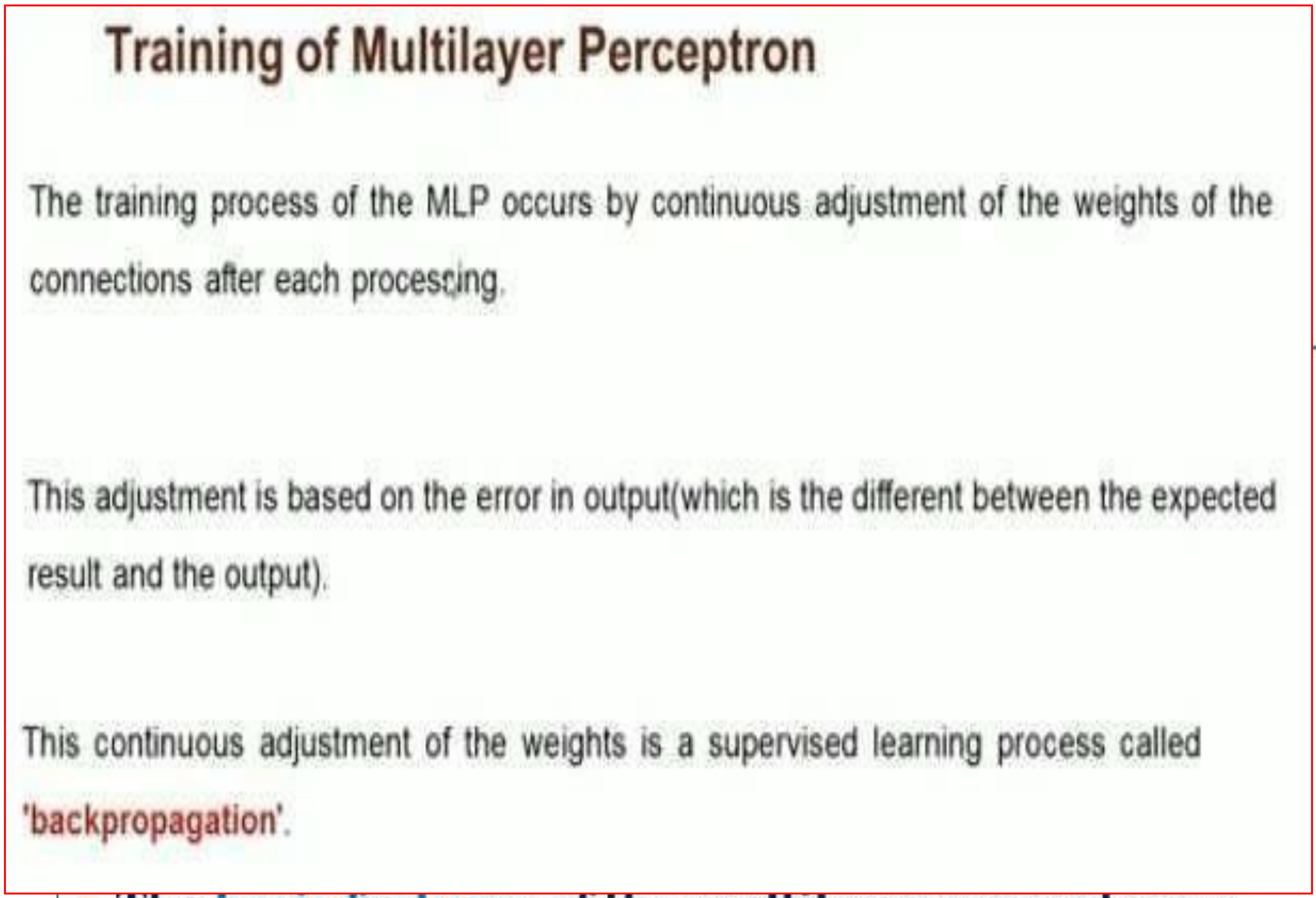
Figure 4.2 Illustration of the directions of two basic signal flows in a multilayer perceptron: forward propagation of function signals and back propagation of error signals.





two phases:

- In the ***forward phase***, the weights of the network are *fixed* and the input signal is *propagated* through the network, layer by layer, until it reaches the *output*.
- In the ***backward phase***, the error signal, which is produced by comparing the output of the network and the desired response, is *propagated* through the network, again layer by layer, but in the *backward direction*.

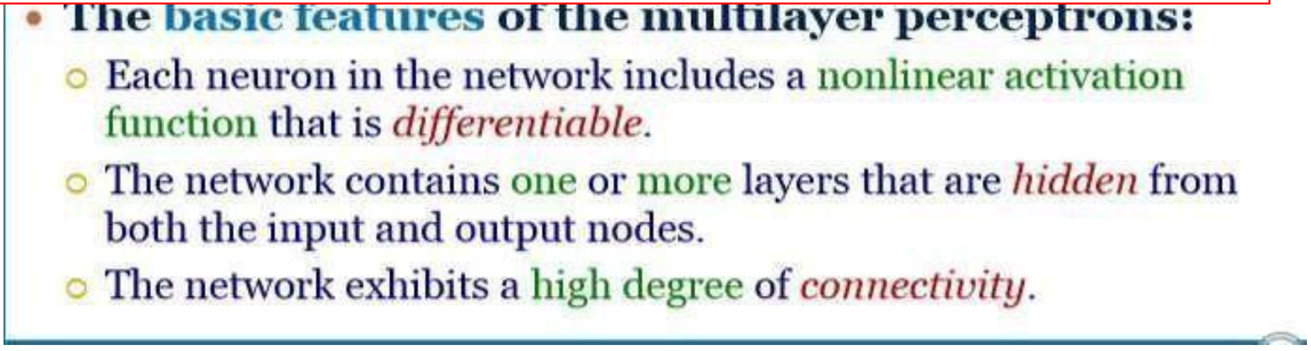


Training of Multilayer Perceptron

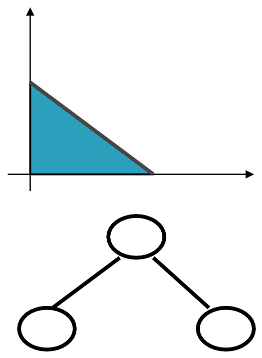
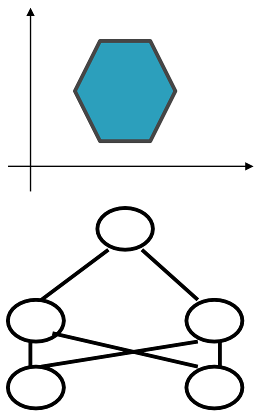
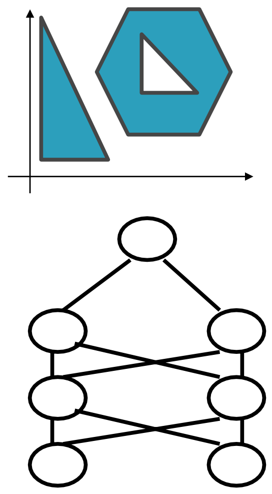
The training process of the MLP occurs by continuous adjustment of the weights of the connections after each processing.

This adjustment is based on the error in output(which is the different between the expected result and the output).

This continuous adjustment of the weights is a supervised learning process called 'backpropagation'.



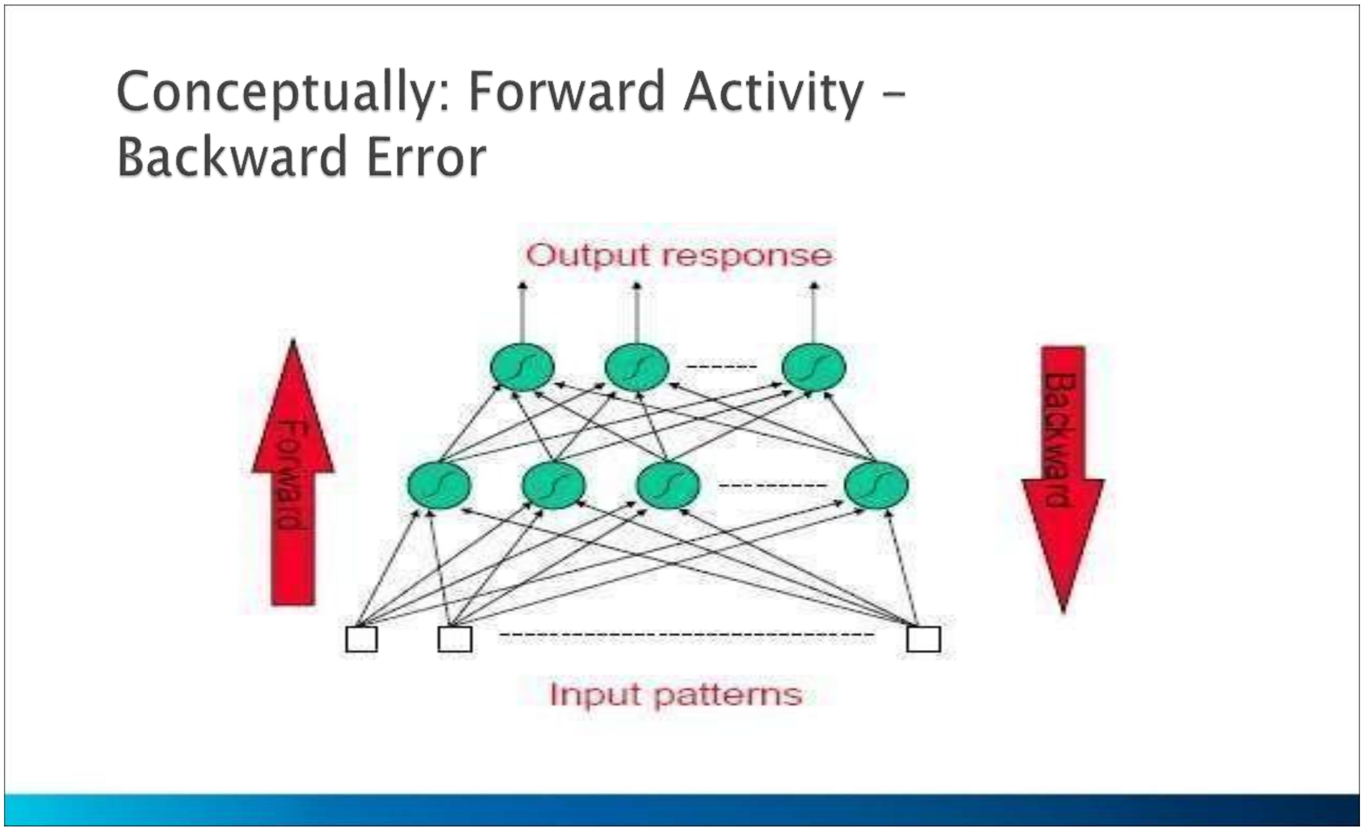
- The basic features of the multilayer perceptrons:
  - Each neuron in the network includes a nonlinear activation function that is *differentiable*.
  - The network contains one or more layers that are *hidden* from both the input and output nodes.
  - The network exhibits a high degree of *connectivity*.

Properties of architecture

- No connections within a layer
- No direct connections between input and output layers
- Fully connected between layers
- Often more than 3 layers
- Number of output units need not equal number of input units
- Number of hidden units per layer can be more or less than input or output units

1st layer draws linearboundaries   2nd layer combines the boundaries   3rd layer can generate arbitrarily complex boundaries



Conceptually: Forward Activity – Backward Error

2.1 Going Forward

- Start at the left by filling in the values for the inputs.
- Use these inputs and the first level of weights to calculate the activations of the hidden layer
- Use those activations and the next set of weights to calculate the activations of the output layer.
- Then got the outputs of the network, we can compare them to the targets and compute the error. Biases
- Include a bias input to each neuron as such in Perceptron. by having an extra input that is
- permanently set to -1, and adjusting the weights to each neuron as part of the training.
- Thus, each neuron in the network (whether it is a hidden layer or the output) has 1 extra input, with
- fixed value.

2 .2 GOINGBACKWARDS: BACK-PROPAGATION OF ERROR

- Computing the errors at the output is no more difficult in Perceptron, but working out what to do with those errors is more difficult.
- The method is called back-propagation of error, that the errors are sent backwards through the network.
- The best way to describe back-propagation properly is mathematically, by choose an error function
- k: Ek = yk −tk for each neuron and tried to make it as small as possible.
- If it has only one set of weights in the network, it was sufficient to train the network.
- But, with the addition of extra layers of weights, this is harder to arrange.
- The problem is that try to adapt the weights of the Multi-layer Perceptron, it has to work out which weights caused the error. This could be the weights connecting the inputs to the hidden layer, or the weights connecting the hidden layer to the output layer.
- The error function that used for the Perceptron was where N is the number of output nodes.

If MLP has two errors,

1.The target is bigger than the output

2.The output is bigger than the target.

If these two errors are the same size, then add them up to get 0, which means there was no error.

• To get no errors make all errors have the same sign.

• It will be done in a few different ways, but the one that will turn out to be best is the sum-of-squares error function, which calculates the difference between y and t for each node, squares them, and adds them all together: ½ makes it easier when differentiate the function.

If differentiate a function, then it is called gradient of function, which is the direction along which it increases and decreases the most.

If differentiate an error function, it gets the gradient of the error. Since the purpose of learning is to minimize the error, following the error function downhill (in other words, in the direction of the negative gradient) .

Imagine a ball rolling around on a surface that looks like the line in Figure 4.3. Gravity will make the ball roll downhill (follow the downhill gradient) until it ends up in the bottom of one of the hollows.

The places where the error is small, that algorithm is called gradient descent.

Differentiate with respect to three things in the network that change:

The inputs

The activation function that decides whether or not the node fires

The weights.

- It saturates(reaches its constant values) at ± 1 instead of 0 and1.
- It also has a relatively simple derivative: d/dx tanh x = (1−tanh2(x)).
- It can convert between the two easily, because if the saturation points are (±1), then it can convert to (0,1) by using 0.5×(x+1).

New form of error computation and new activation function decided whether or not a neuron should fire.

If change the weights means improving the error function of the network.

Fed inputs forward through the network and worked out which nodes are firing.

At the output, computed the errors as the sum squared difference between the outputs and the target.

When the output is computing the gradient of these errors and to decide how much update each weight in the network. Inputs connected to the output layer and after updated means, it will work backwards through the network until get back to the inputs again.

It raises two problems

For the output neurons, don’t know which input.

For the hidden neurons, don’t know the target.

The Multi-layer Perceptron Algorithm

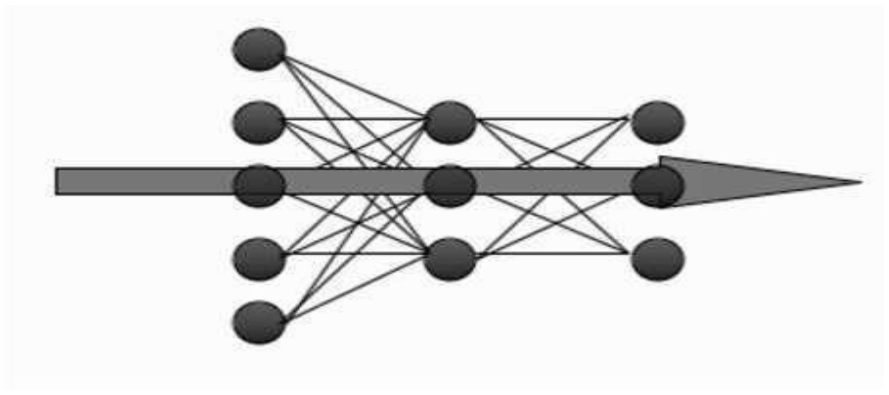


FIGURE 4.6 The forward direction in a Multi-layer Perceptron.

Assume

L input nodes, plus the bias

M hidden nodes, plus a bias

N output nodes

(L+1)×M weights between the input and the hidden layer

(M+1)×N between the hidden layer and the output.

x0 =−1 is the bias input

a0 = −1 is the bias hidden node.

i,j,k to index the nodes in each layer in the sums, and the corresponding Greek letters (ι,ζ,κ) for fixed indices

.

The Multi-layer Perceptron Algorithm

MLP training algorithm using back-propagation of error is described.

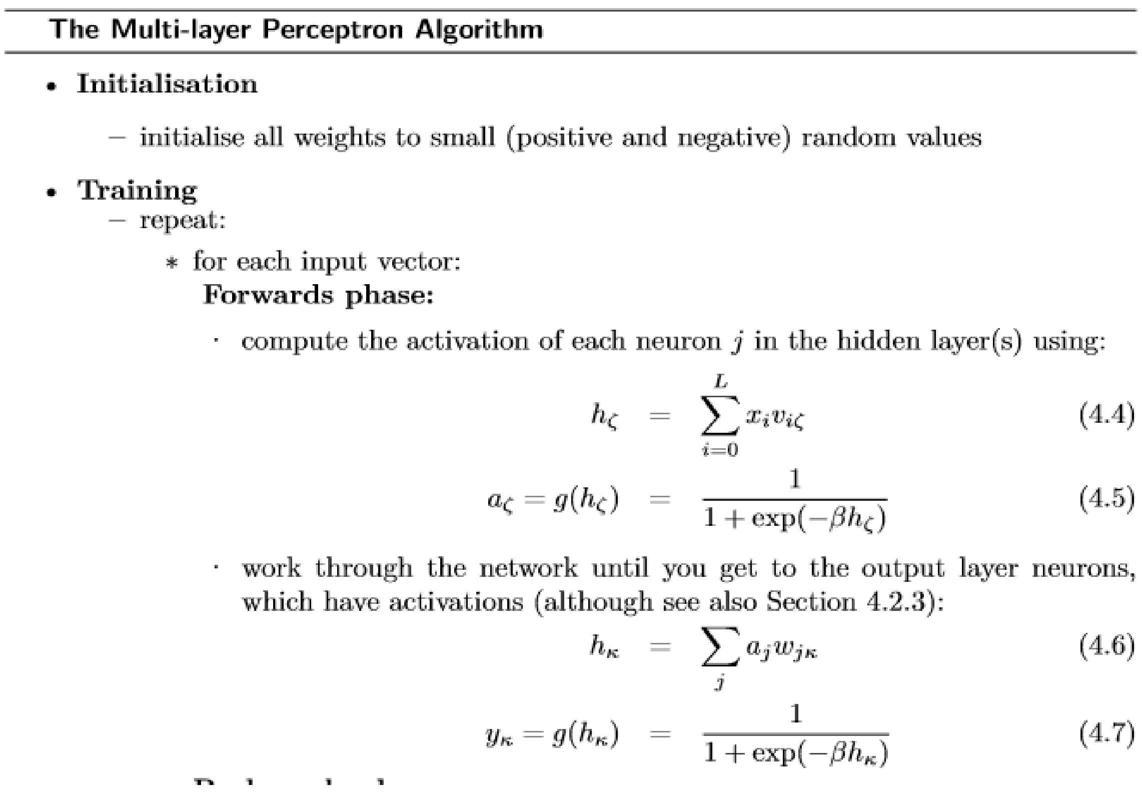
1.An input vector is put into the input nodes 2. the inputs are fed forward through the network (Figure 4.6)

2.The inputs and the first-layer weights (here labelled as v) are used to decide whether the hidden nodes fire or not. The activation function g(·) is the sigmoid function given in Equation (4.2)above. The outputs of these neurons and the second-layer weights (labelled as w)are used to decide if the output neurons fire or not

3.Error is computed as the sum-of-squares difference between the network outputs and the targets

4.This error is fed backwards through the network in order to

- first update the second-layer weights
- and then afterwards, the first-layer weights



The Multi-layer Perceptron Algorithm

- **Initialisation**
  - initialise all weights to small (positive and negative) random values
- **Training**
  - repeat:
    - for each input vector:

      Forwards phase:
      - compute the activation of each neuron j in the hidden layer(s) using:

$$h_\zeta = \sum_{i=0}^{L} x_i v_{i\zeta} \tag{4.4}$$

$$a_\zeta = g(h_\zeta) = \frac{1}{1+\exp(-\beta h_\zeta)} \tag{4.5}$$

      - work through the network until you get to the output layer neurons, which have activations (although see also Section 4.2.3):

$$h_\kappa = \sum_j a_j w_{j\kappa} \tag{4.6}$$

$$y_\kappa = g(h_\kappa) = \frac{1}{1+\exp(-\beta h_\kappa)} \tag{4.7}$$

Backwards phase:

- compute the error at the output using:

$$\delta_o(\kappa) = (y_\kappa - t_\kappa)\, y_\kappa(1 - y_\kappa) \tag{4.8}$$

- compute the error in the hidden layer(s) using:

$$\delta_h(\zeta) = a_\zeta(1 - a_\zeta) \sum_{k=1}^{N} w_\zeta \delta_o(k) \tag{4.9}$$

- update the output layer weights using:

$$w_{\zeta\kappa} \leftarrow w_{\zeta\kappa} - \eta\delta_o(\kappa) a_\zeta^{\text{hidden}} \tag{4.10}$$

- update the hidden layer weights using:

$$v_\iota \leftarrow v_\iota - \eta\delta_h(\kappa) x_\iota \tag{4.11}$$

* (if using sequential updating) randomise the order of the input vectors so that you don't train in exactly the same order each iteration

– until learning stops (see Section 4.3.3)

- **Recall**
  – use the Forwards phase in the training section above

Initialising method:

weights are initialized to small random numbers, both positive and negative.

If the initial weights values are close to 1 or -1 then the inputs to the sigmoid are also likely to be close to ±1 and so the output of the neuron is either 0 or 1 (the sigmoid has saturated, reached its maximum or minimum value).

If the weights are very small (close to zero) then the input is still close to 0 and so the output of the neuron is just linear, so gets a linear model.

Input to the neuron will be $w\sqrt{n}$, where w is the initialization value of the weights.

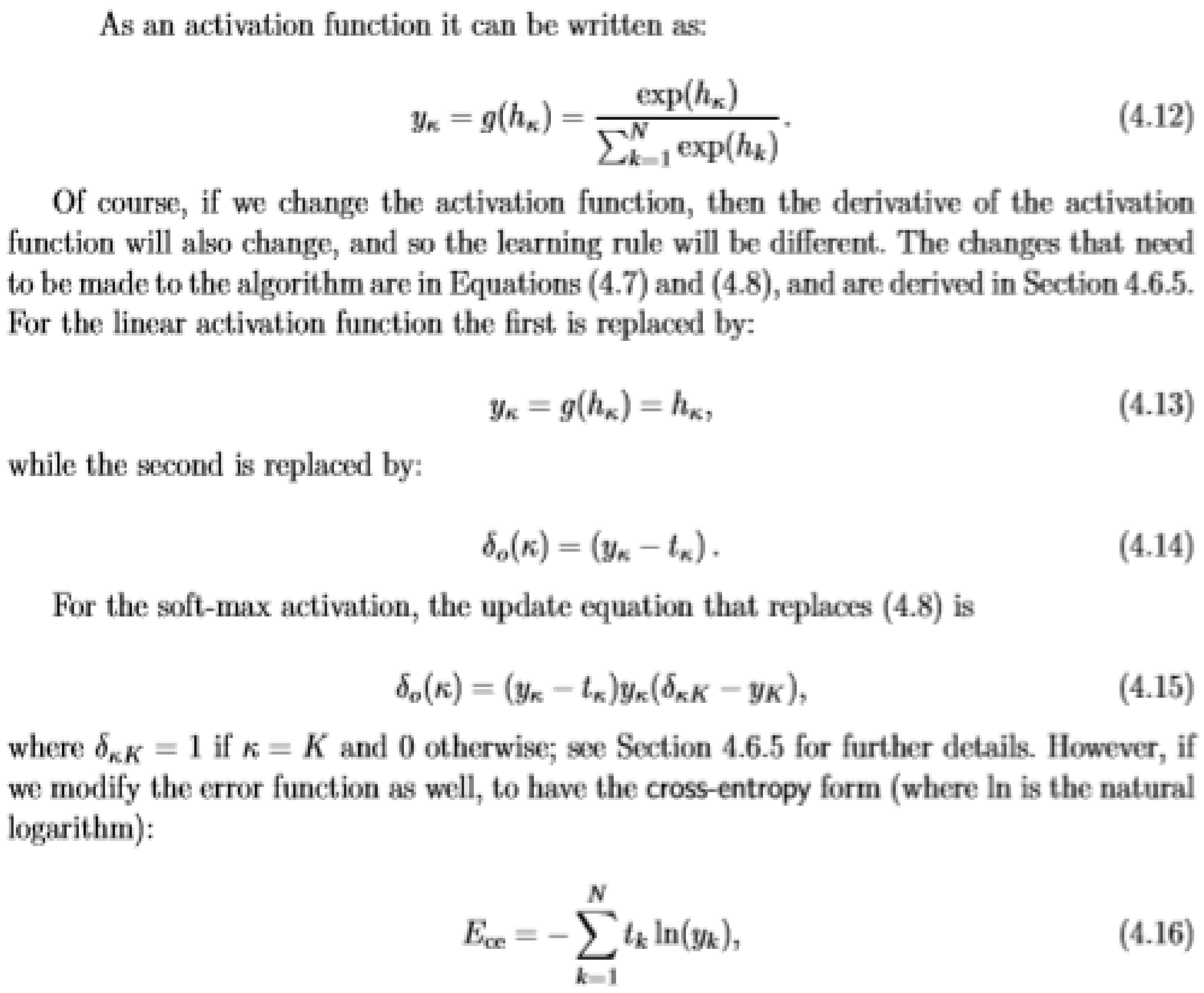
Set the weights in the range $-1/\sqrt{n} < w < 1/\sqrt{n}$, where n is the number of nodes in the input layer.

β in the logistic function (say $\beta = 3.0$ or less) are more effective.

3 Different Output Activation Functions

Sigmoid neurons in hidden and output layer- 0 and 1 ,Regression problem- continuous range

Soft-max activation – 1 of N output encoding. The soft-max function rescales the outputs by calculating the exponential of the inputs to that neuron and dividing by the total sum of the inputs to all of the neurons, so that the activations sum to 1 and lie between 0 and 1.



As an activation function it can be written as:

$$y_\kappa = g(h_\kappa) = \frac{\exp(h_\kappa)}{\sum_{k=1}^{N}\exp(h_k)}. \tag{4.12}$$

Of course, if we change the activation function, then the derivative of the activation function will also change, and so the learning rule will be different. The changes that need to be made to the algorithm are in Equations (4.7) and (4.8), and are derived in Section 4.6.5. For the linear activation function the first is replaced by:

$$y_\kappa = g(h_\kappa) = h_\kappa, \tag{4.13}$$

while the second is replaced by:

$$\delta_o(\kappa) = (y_\kappa - t_\kappa)\,. \tag{4.14}$$

For the soft-max activation, the update equation that replaces (4.8) is

$$\delta_o(\kappa) = (y_\kappa - t_\kappa)y_\kappa(\delta_{\kappa K} - y_K), \tag{4.15}$$

where $\delta_{\kappa K} = 1$ if $\kappa = K$ and 0 otherwise; see Section 4.6.5 for further details. However, if we modify the error function as well, to have the cross-entropy form (where ln is the natural logarithm):

$$E_{ce} = -\sum_{k=1}^{N} t_k \ln(y_k), \tag{4.16}$$

2.2.4 Sequential and Batch Training

The MLP is designed to be a batch algorithm.

All of the training examples are presented to the neural network, the average sum-of-squares error is then computed, and this is used to update the weights.

Thus there is only one set of weight updates for each epoch (pass through all the training examples).

This means that only update the weights once for each iteration of the algorithm, which means that the weights are moved in the direction that most of the inputs want them to move, rather than being pulled around by each input individually.

The batch method performs a more accurate estimate of the error gradient, and will thus converge to the local minimum more quickly

2.2.5 Local minima

The learning rule is the minimisation of the network error by gradient descent (using the derivative of the error function to make the error smaller).

Perform an optimisation-adapting the values of the weights in order to minimise the error function.

2.2.6 Picking Up Momentum

A local minimum of a function is a point where the function value is smaller than at nearby points, but possibly

• Neural network learning by adding in some contribution from the previous weight change that made to the current one

•Benefit to momentum: Use a smaller learning rate, which means that the learning is more stable.

• Weight decay- reduces the size of the weights as the number of iterations increases. This gives better result to lead a network

ie., small weights are closer to linear greater than at a distant point.

A global minimum is a point where the function value is smaller than at all other feasible points.

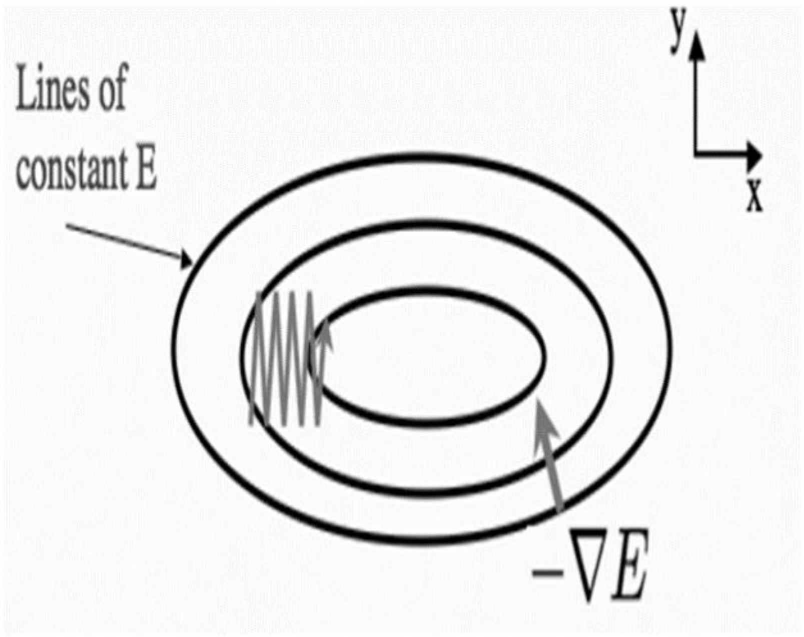
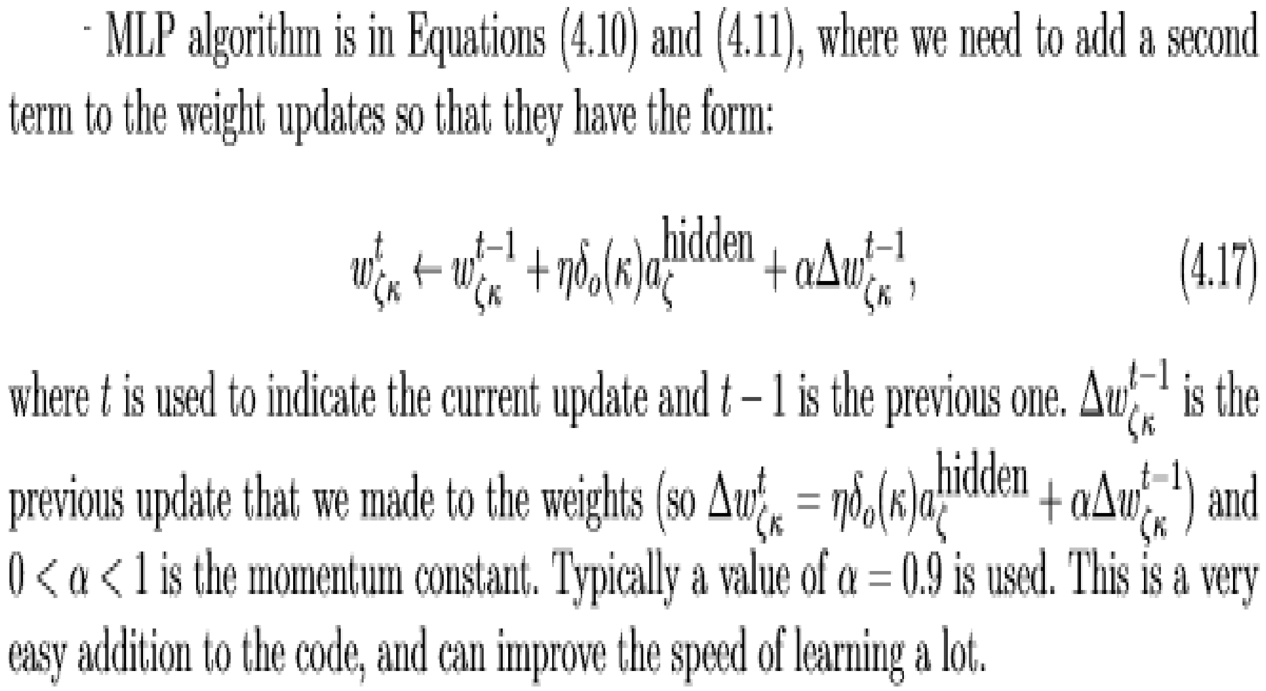


FIGURE 4.7 In 2D, downhill means at right angles to the lines of constant contour. Imagine walking down a hill with your eyes closed. If you find a direction that stays flat, then it is quite likely that perpendicular to that the ground goes uphill or downhill. However, this is not the direction that takes you directly towards the local minimum.



- MLP algorithm is in Equations (4.10) and (4.11), where we need to add a second term to the weight updates so that they have the form:

$$w_{\zeta\kappa}^{t} \leftarrow w_{\zeta\kappa}^{t-1} + \eta\delta_o(\kappa)a_{\zeta}^{\text{hidden}} + \alpha\Delta w_{\zeta\kappa}^{t-1}, \tag{4.17}$$

where t is used to indicate the current update and $t-1$ is the previous one. $\Delta w_{\zeta\kappa}^{t-1}$ is the previous update that we made to the weights (so $\Delta w_{\zeta\kappa}^{t} = \eta\delta_o(\kappa)a_{\zeta}^{\text{hidden}} + \alpha\Delta w_{\zeta\kappa}^{t-1}$) and $0 < \alpha < 1$ is the momentum constant. Typically a value of $\alpha = 0.9$ is used. This is a very easy addition to the code, and can improve the speed of learning a lot.

Minibatches and Stochastic Gradient Descent

Batch algorithm converges to a local minimum faster than the sequential algorithm, which computes the error for each input individually and then does a weight update, but latter stuck in local minima. The idea of a minibatch method is by splitting the training set into random batches, estimating the gradient based on one of the subsets of the training set, performing a weight update, and then using the next subset to estimate a new gradient and using that for the weight update, until all of the training set has been used. The training set are then randomly shuffled into new batches and the next iteration takes place. A more extreme version of the minibatch idea is to use just one piece of data to estimate the gradient at each iteration of the algorithm, and to pick that piece of data uniformly at random from the training set. So a single input vector is chosen from the training set, and the output and hence the error for that one vector computed, and this is used to estimate the gradient and so update the weights. A new random input vector (which could be the same as the previous one) is then chosen and the process repeated. This is known as stochastic gradient descent. t It is often used if the training set is very large, since it would be very expensive to use the whole dataset to estimate the gradient in that case.

Other Improvements

One is to reduce the learning rate as the algorithm progresses. The network making large-scale changes to the weights at the beginning, when the weights are random. Results gives larger performance gains the second derivatives of the error with respect to the weights. In the back-propagation algorithm - use the first derivatives to drive the learning. Knowledge of the second derivatives , it helps to improve the network

A Step by Step Backpropagation Example

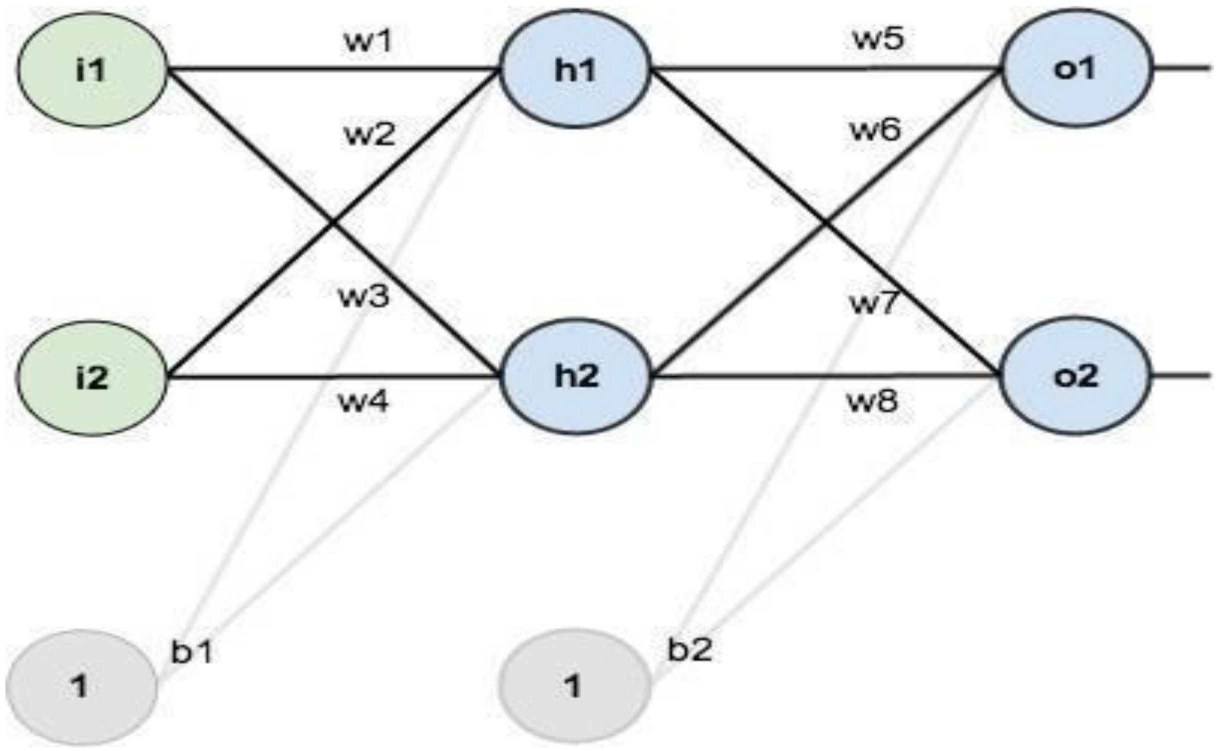
Background

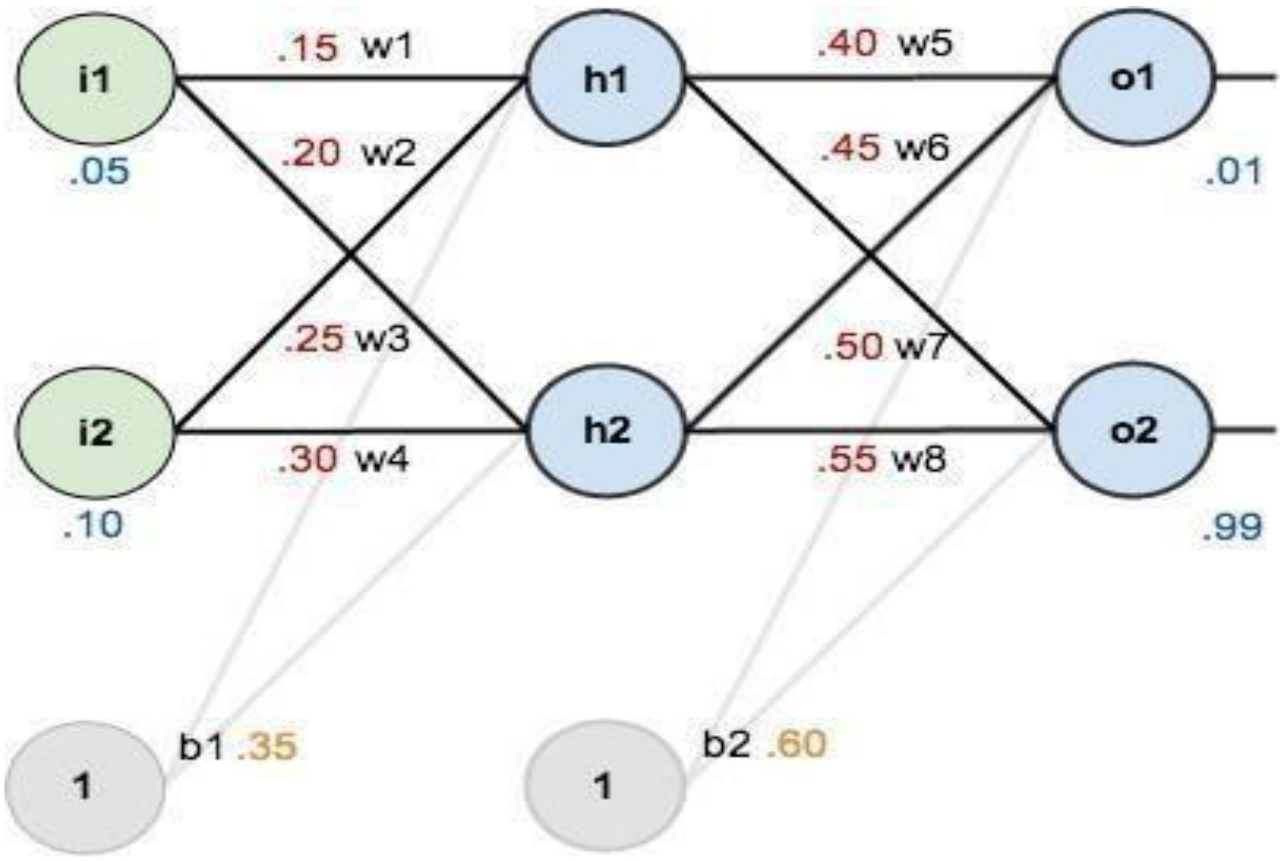
Backpropagation is a common method for training a neural network.

Overview

For this tutorial, we're going to use a neural network with two inputs, two hidden neurons, two output neurons. Additionally, the hidden and output neurons will include a bias.

Here's the basic structure:





The goal of backpropagation is to optimize the weights so that the neural network can learn howto correctly map arbitrary inputs to outputs.

Single training set:

- Inputs 0.05 and 0.10

- Expected output 0.01 and 0.99.

The Forward Pass

- To begin, let’s see what the neural network currently predicts given the weights andbiases above and inputs of 0.05 and 0.10.
- To do this we’ll feed those inputs forward though 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:

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

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

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

$$out_{h2} = 0.596884378$$

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:

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

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

And carrying out the same process for we get:

$$out_{o2} = 0.772928465$$

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

Calculating the Total Error

Calculate the error for each output neuron using the squared error function and sum them to get the total error:

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

Some sources refer to the target as the *ideal* and the output as the *actual*.

The $\frac{1}{2}$ is included so that exponent is cancelled when we differentiate later on. The result is eventually multiplied by a learning rate anyway so it doesn't matter that we introduce a constant here [1].

For example, 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 Backwards Pass

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

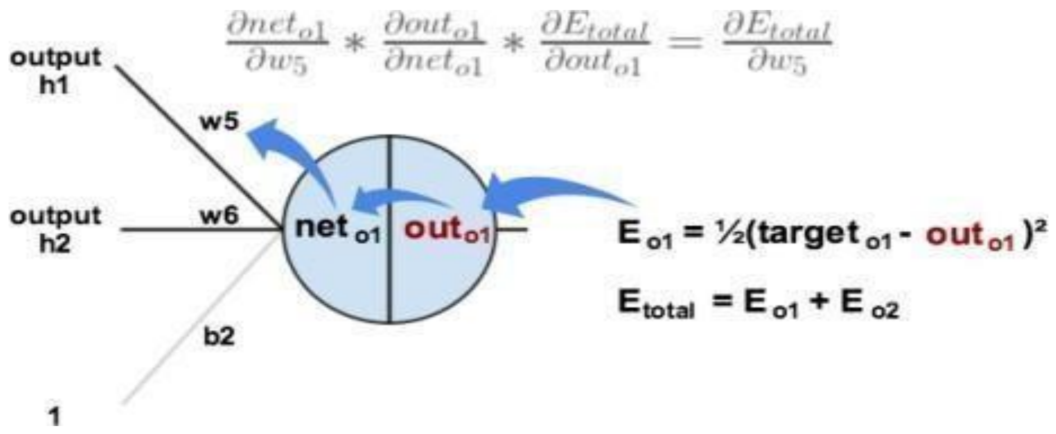
Output Layer

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

$\frac{\partial E_{total}}{\partial w_5}$ is read as "the partial derivative of E_{total} with respect to w_5". You can also say "the gradient with respect to w_5".

By applying the chain rule we know that:

Visually, here's what we're doing:



$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$
output h1
output h2
1
w5
w6
b2
net_{o1}
out_{o1}
$E_{o1} = ½(target_{o1} - out_{o1})^2$
$E_{total} = E_{o1} + E_{o2}$

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

The Backwards Pass

Our goal with backpropagation is to update each of the weights in the network so that they causethe 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,

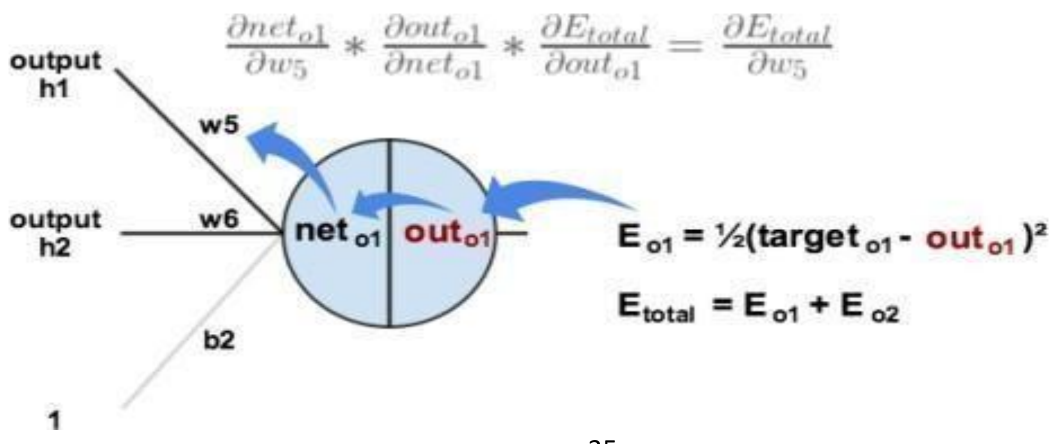
$$\frac{\partial E_{total}}{\partial w_5}$$

.

$\frac{\partial E_{total}}{\partial w_5}$ is read as "the partial derivative of E_{total} with respect to w_5". You can also say "the gradient with respect to w_5".

By applying the chain rule we know that:

Visually, here's what we're doing:



$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$
output h1
output h2
1
w5
w6
b2
net o1
out o1
E o1 = ½(target o1 - out o1)²
E total = E o1 + E o2

Putting it all together:

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

You'll often see this calculation combined in the form of the delta rule:

$$\frac{\partial E_{total}}{\partial w_5} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1}) * out_{h1}$$

Alternatively, we have $\frac{\partial E_{total}}{\partial out_{o1}}$ and $\frac{\partial out_{o1}}{\partial net_{o1}}$ which can be written as $\frac{\partial E_{total}}{\partial net_{o1}}$, aka δ_{o1} (the Greek letter delta) aka the *node delta*. We can use this to rewrite the calculation above:

$$\delta_{o1} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1})$$

Therefore:

$$\frac{\partial E_{total}}{\partial w_5} = \delta_{o1} out_{h1}$$

Some sources extract the negative sign from δ so it would be written as:

$$\frac{\partial E_{total}}{\partial w_5} = -\delta_{o1} out_{h1}$$

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):

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

Some sources use α (alpha) to represent the learning rate, others use η (eta), and others even use ϵ (epsilon).

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).

Hidden Layer

Next, we'll continue the backwards pass by calculating new values for , $w_2$$_)$$w_1$, and w_{4_3}

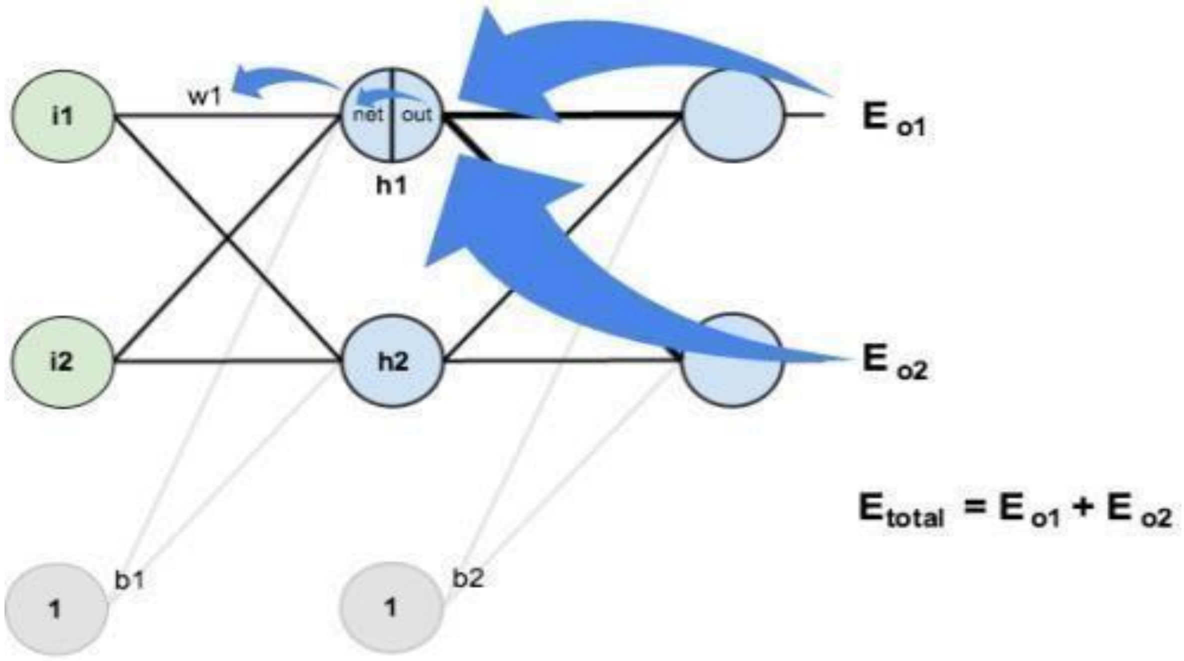
Big picture, here's what we need to figure out:

Visually:

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\downarrow$$

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



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:

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

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

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:

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

$$\frac{\partial E_{o2}}{\partial out_{h1}} = -0.019049119$$

Therefore:

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 tothe same as we w_1 did for the output neuron:

$$net_{h1} = w_1 * i_1 + w_3 * i_2 + b_1 * 1$$

$$\frac{\partial net_{h1}}{\partial w_1} = i_1 = 0.05$$

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

You might also see this written as:

$$\frac{\partial E_{total}}{\partial w_1} = (\sum_o \delta_o * w_{ho}) * out_{h1}(1 - out_{h1}) * i_1$$

$$\frac{\partial E_{total}}{} = \delta_{h1} i_1$$

$$w_1^+ = w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 * 0.000438568 = 0.149780716$$

We can now update w_1:

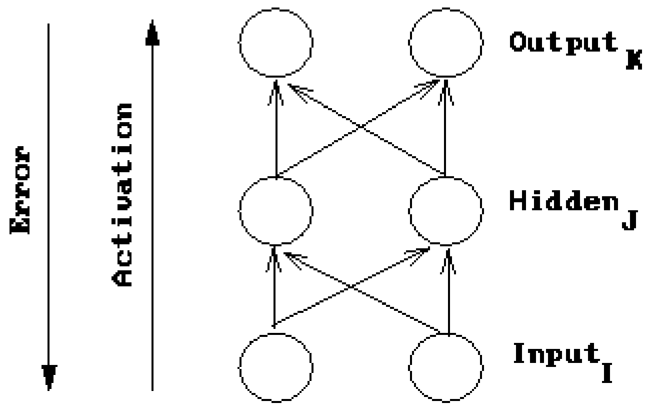
$$w_2^+ = 0.19956143$$

$$w_3^+ = 0.24975114$$

$$w_4^+ = 0.29950229$$

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 thisprocess 10,000 times, for example, the error plummets to 0.0000351085. At this point, when wefeed forward 0.05 and 0.1, the two outputs neurons generate 0.015912196 (vs 0.01 target) and 0.984065734 (vs 0.99 target).

1 Introduction



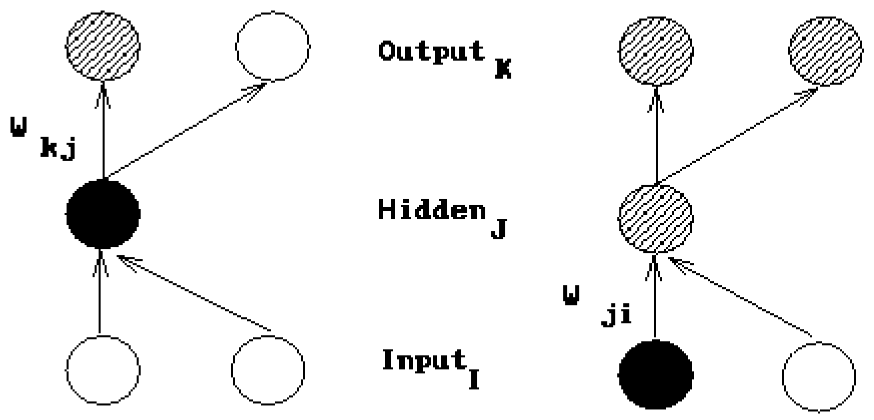
Conceptually, a network forward propagates activation to produce an output and it backward propagates error to determine weight changes (as shown in Figure 1). The weightson the connections between neurons mediate the passed values in both directions.

The Backpropagation algorithm is used to learn the weights of a multilayer neural network with a fixed architecture. It performs gradient descent to try to minimize the sum squared error between the network's output values and the given target values. Figure 2 depicts the network components which affect a particular weight change. Noticethat all the necessary components are locally related to the weight being updated. This is one feature of backpropagation that seems biologically plausible. However, brain connections appear to be unidirectional and not bidirectional as would be required to implement backpropagation.

1 Notation

For the purpose of this derivation, we will use the following notation:

- The subscript *k* denotes the output layer.
- The subscript *j* denotes the hidden layer.
- The subscript *i* denotes the input layer.



Linear models

- Radial Basis Functions and Splines – Concepts
- RBF Network
- Curse of Dimensionality
- Interpolations and Basis

Radial Basis Functions and Splines – Concepts

An RBFN performs classification by measuring the input's similarity to examples from the training set. Each RBFN neuron stores a prototype, which is just one of the examples from the training set.

When we want to classify a new input, each neuron computes the Euclidean distance between the input and its prototype. Ie., If the input more closely resembles the Class A prototypes than the Class B prototypes, it is classified as Class A.

Classification:

Purpose: assign previously unseen patterns to their respective classes.

Training: Previous examples of each class. Output: A class out of a discrete set of classes. Classification problems can be made to look like nonparametric regression.

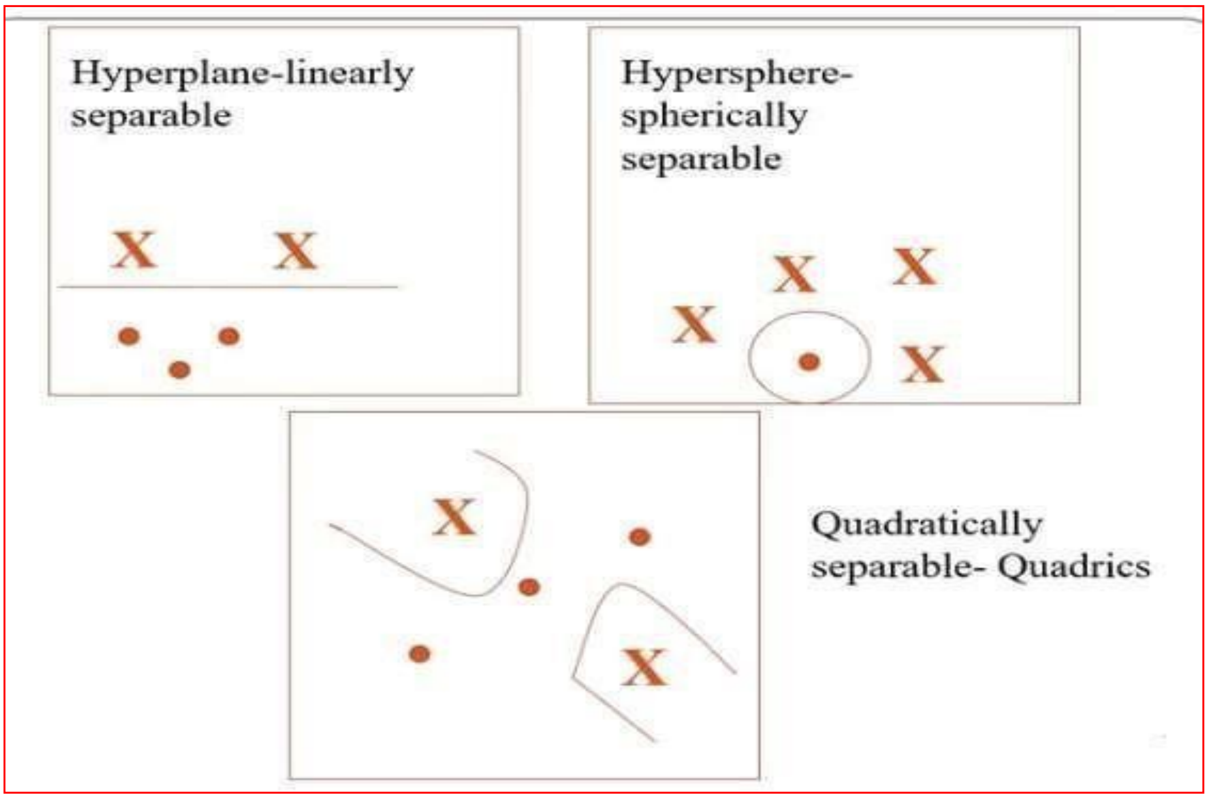
RBF would be separate class distributions by localizing radial basis functions.

Types of separating surfaces are

Hyperplane- linearly separable

Spherically separable-Hypersphere

Quadratically separable- Quadrics


Hyperplane-linearly separable
X
X
Hypersphere-spherically separable
X
X
X
X
X
X
Quadratically separable- Quadrics

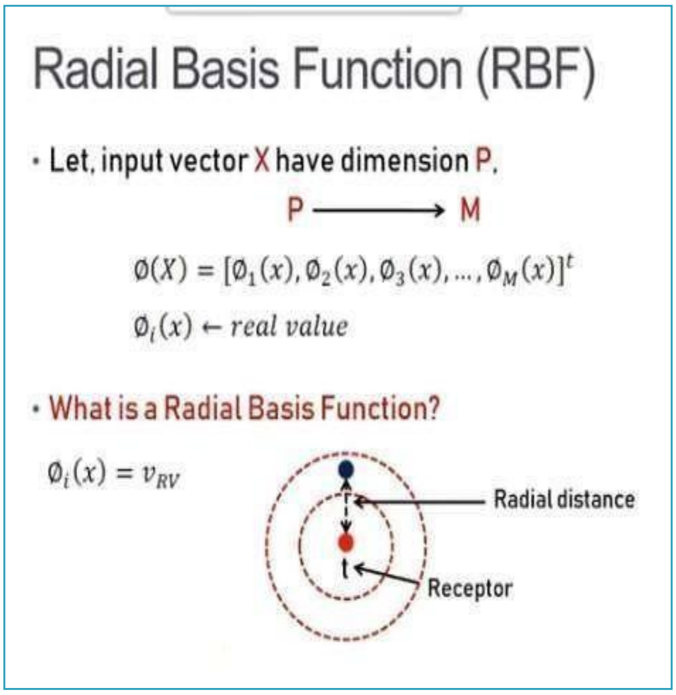
■ Input layer should mapping with hidden layer , there they provide a set of functions which forms a based for mapping into the hidden layer space.

■ To do mapping from input space to the hidden layer , it is need some basis functions providing the neurons in the hidden layer. Hidden neurons in hidden layer providing the basis functions and this kind of architecture is called Radial Basis function networks.

- Approximate function with linear combination of Radial basis functions
    - $F(x) = \sum w_i \, h(x)$
    - h(x) is mostly Gaussian function
- Three layers

- Input layer – Source nodes that connect to the network to its environment / hidden layer. (Non linear mapping)
- Hidden layer – Hidden units provide a set of basis function– Nonlinear transformation
- Ie Input space -> Hidden space(High dimensionality)
- Output layer – Linear combination of hidden functions

■ Cover's Theorem : A pattern classification problem cast in high dimensional space is more likely to be linearly separable than in a low dimension space.



Radial Basis Function (RBF)

- Let, input vector X have dimension P.

$$P \longrightarrow M$$

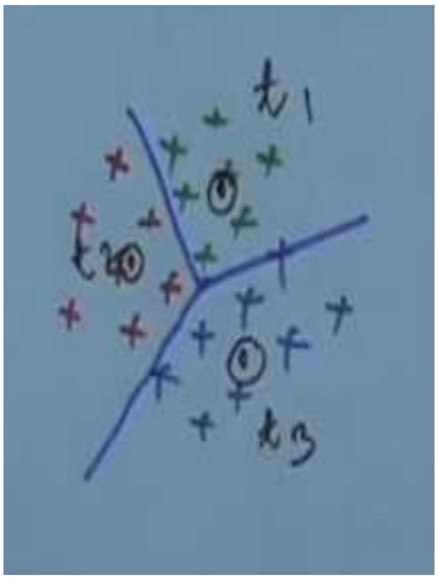
$$\emptyset(X) = [\emptyset_1(x), \emptyset_2(x), \emptyset_3(x), \dots, \emptyset_M(x)]^t$$

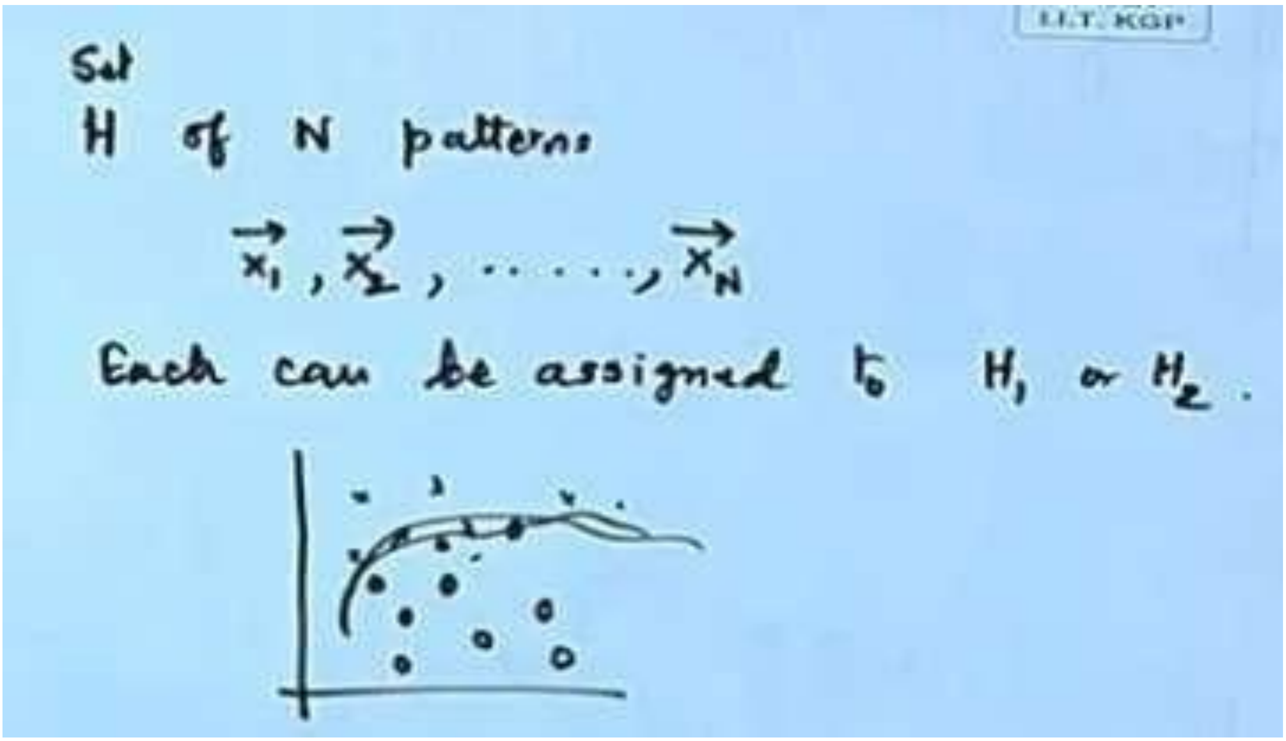
$$\emptyset_i(x) \leftarrow real\ value$$

- What is a Radial Basis Function?

$$\emptyset_i(x) = v_{RV}$$

Radial distance
Receptor
t



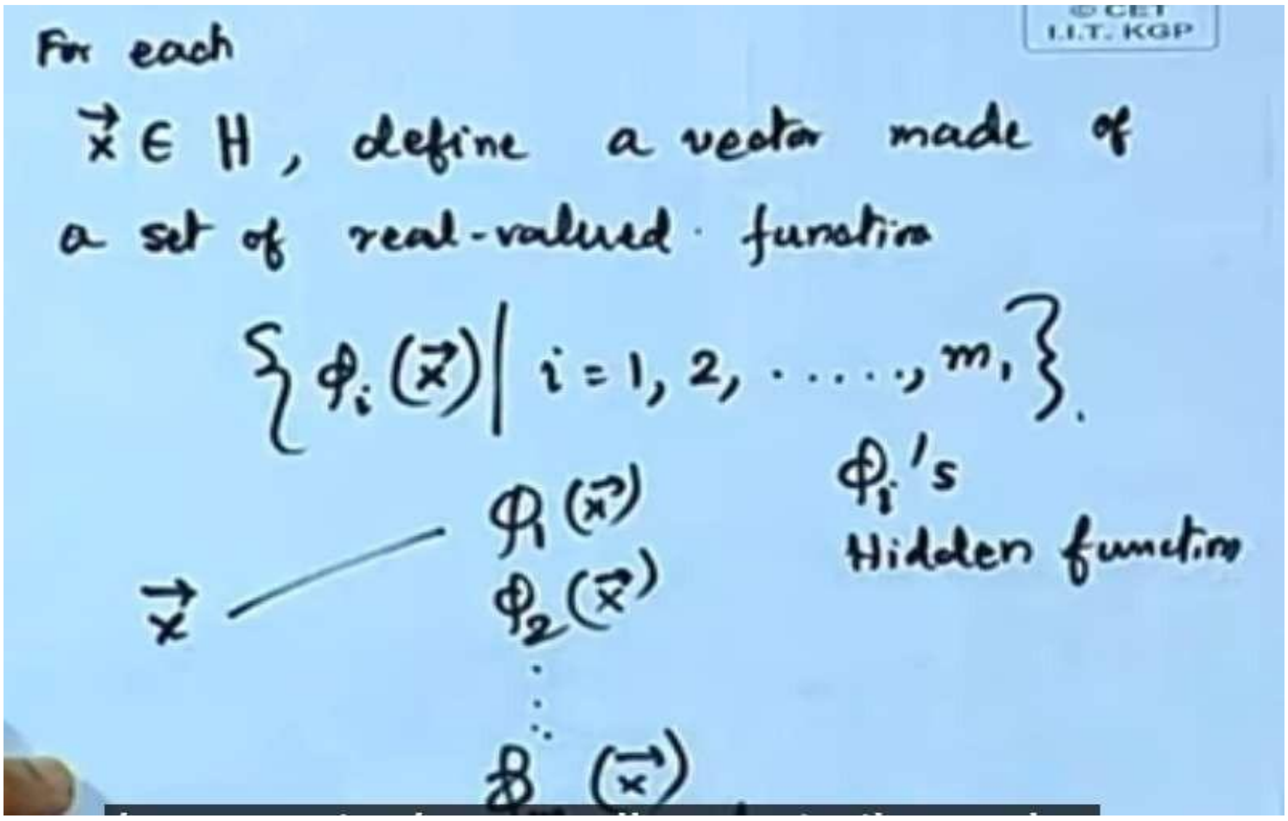


Set

H of N patterns

$$\vec{x}_1, \vec{x}_2, \ldots\ldots, \vec{x}_N$$

Each can be assigned to H_1 or H_2.



For each

$\vec{x} \in H$, define a vector made of a set of real-valued function

$$\{\phi_i(\vec{x}) \mid i = 1, 2, \ldots\ldots, m_1\}.$$

ϕ_i's

Hidden function

$\vec{x}$ — $\phi_1(\vec{x})$

$\phi_2(\vec{x})$

$\vdots$

$\phi(\vec{x})$

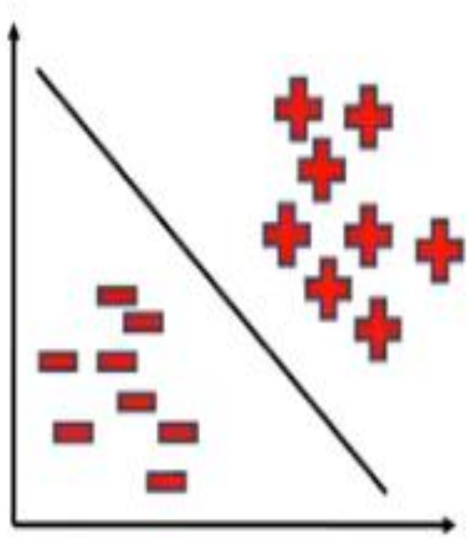
Linear separability

- Data distribution as per its feature space in 2D space.

- Data

  + −

- Separate data linearly



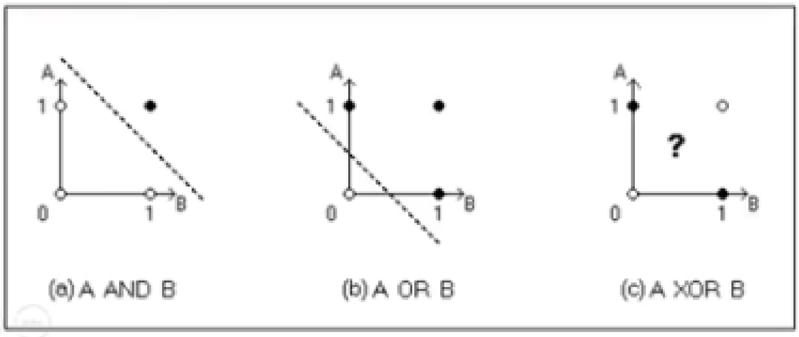
Need of RBFN

- In Single Perceptron, we only have linear separability because they are composed of input and output layers.

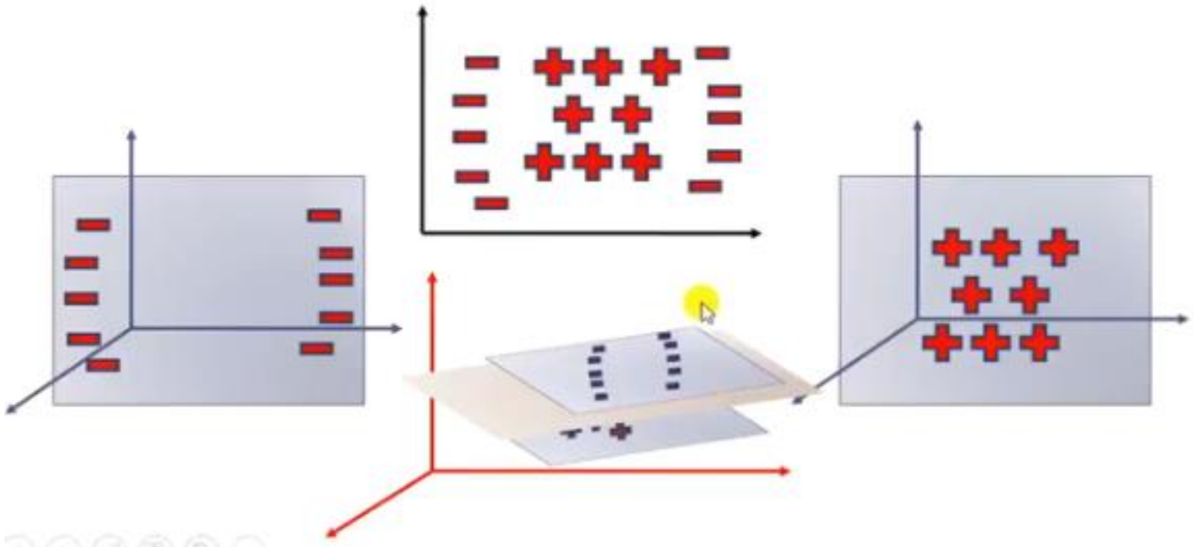
A	B	AND
1	1	1
1	0	0
0	1	0
0	0	0

A	B	OR
1	1	1
1	0	1
0	1	1
0	0	0

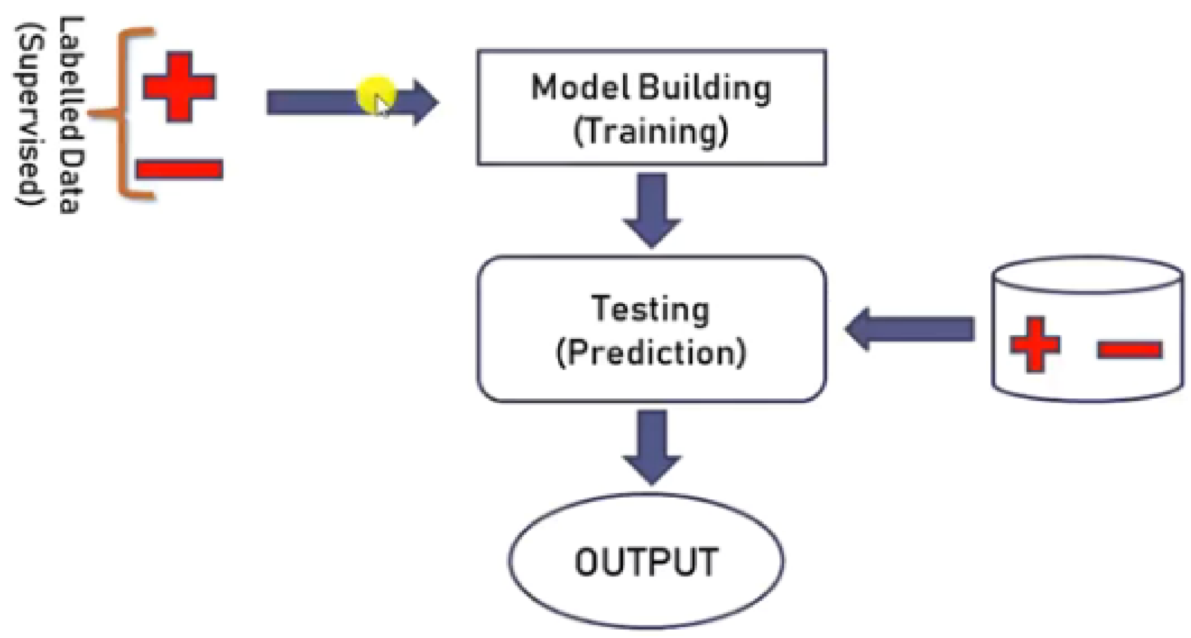
A	B	XOR
1	1	0
1	0	1
0	1	1
0	0	0

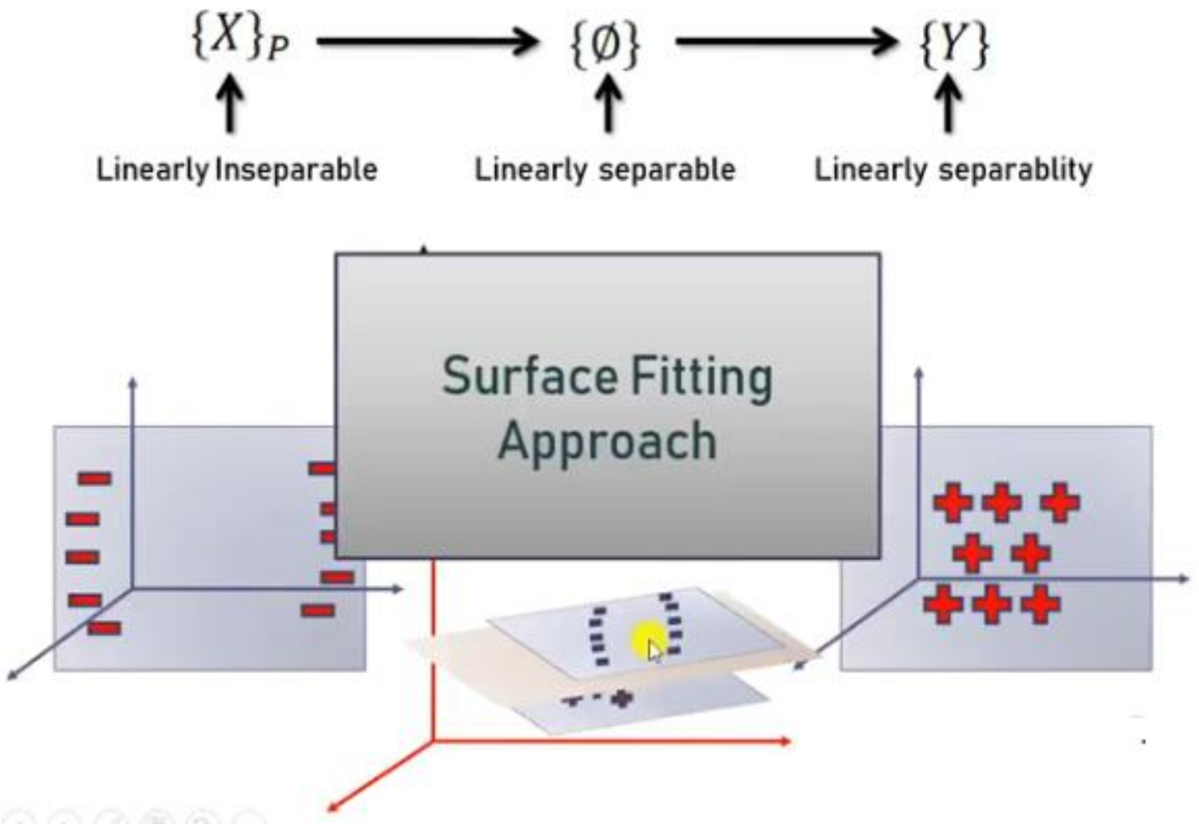


$$\begin{tikzcd}
\{X\}_P \arrow[r] & \{\emptyset\} \arrow[r] & \{Y\} \\
\text{Linearly Inseparable} \arrow[u] & \text{Linearly separable} \arrow[u] & \text{Linearly separablity} \arrow[u]
\end{tikzcd}$$



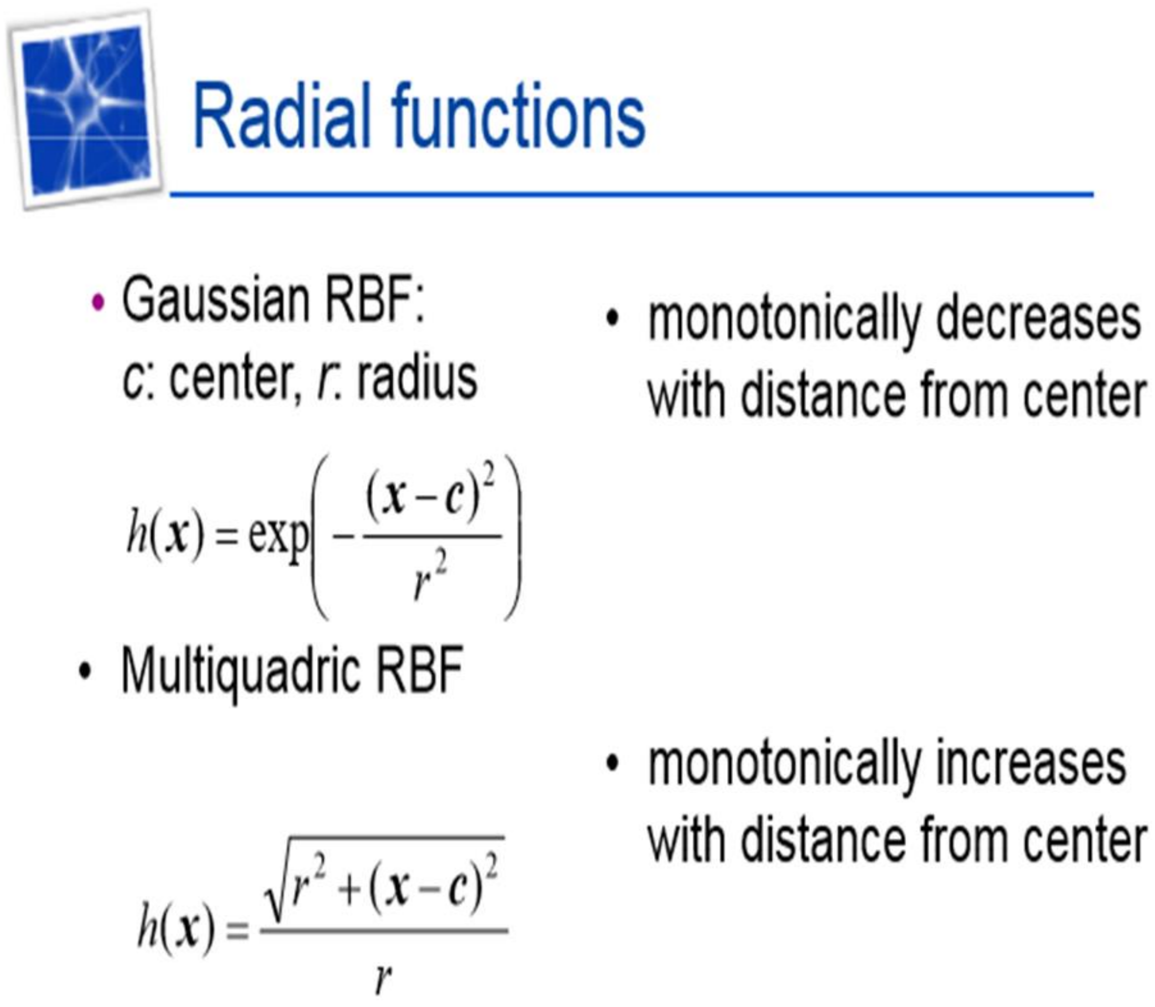
Model Building





What happens in Hidden layer?

The patterns in the input space form clusters. If the centres of these clusters are known then the distance from the cluster centre can be measured. The most commonly used radial basis function is a **Gaussian function.** The other function are **multi quadrics function and inverse multi quadric function.** In an RBF network **r** is the distance from the cluster centre.



Radial functions

- Gaussian RBF:
  c: center, r: radius

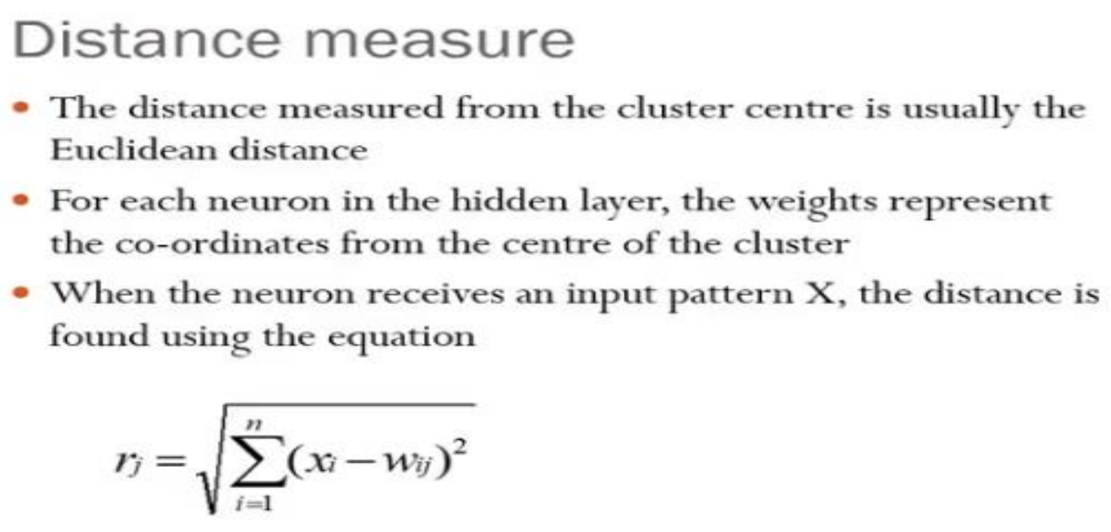
$$h(x) = \exp\left(-\frac{(x-c)^2}{r^2} \right)$$

- monotonically decreases with distance from center

- Multiquadric RBF

$$h(x) = \frac{\sqrt{r^2 + (x-c)^2}}{r}$$

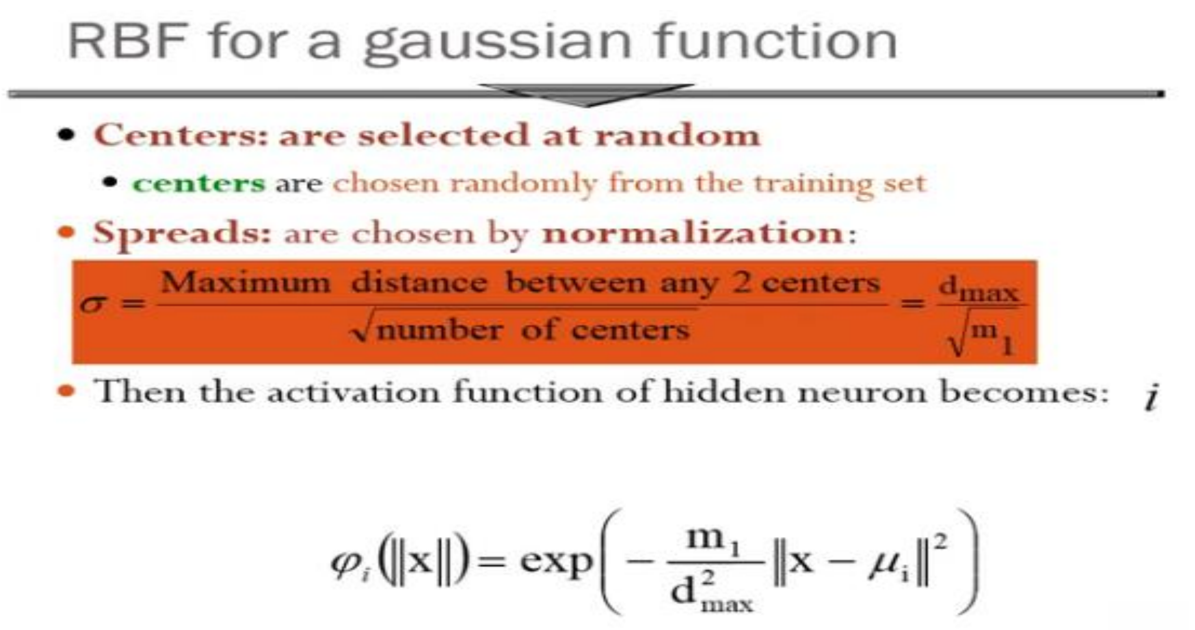
- monotonically increases with distance from center



Distance measure

- The distance measured from the cluster centre is usually the Euclidean distance
- For each neuron in the hidden layer, the weights represent the co-ordinates from the centre of the cluster
- When the neuron receives an input pattern X, the distance is found using the equation

$$r_j = \sqrt{\sum_{i=1}^{n} (x_i - w_{ij})^2}$$



RBF for a gaussian function

- **Centers: are selected at random**
  - **centers** are chosen randomly from the training set
- **Spreads:** are chosen by **normalization**:

$$\sigma = \frac{\text{Maximum distance between any 2 centers}}{\sqrt{\text{number of centers}}} = \frac{d_{max}}{\sqrt{m_1}}$$

- Then the activation function of hidden neuron becomes: i

$$\varphi_i(\|x\|) = \exp\left(-\frac{m_1}{d_{max}^2}\|x - \mu_i\|^2\right)$$

Width of hidden unit

$$\phi j = \exp\left(-\frac{\sum_{i=1}^{n}(xi - \mu j)^2}{2\sigma^2}\right) \text{ ——— } 1$$

where $\sigma = \frac{d_{\max}}{\sqrt{2M}}$ ——— 2

σ Is the width or radius of the bell shape and has to be determined empirically

M=no. of basis function μ_j =basis function centre

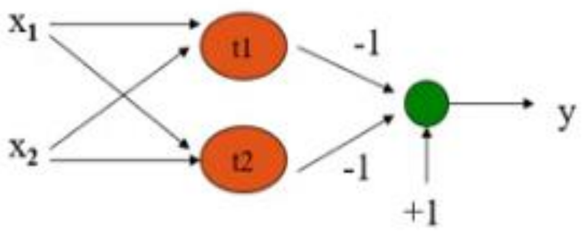
D_{max}=distance between them

$$\phi j = \exp\left(-\frac{M}{d^2_{\max}}\sum_{i=1}^{n}(xi - \mu j)^2\right) \text{ ——— } 3$$

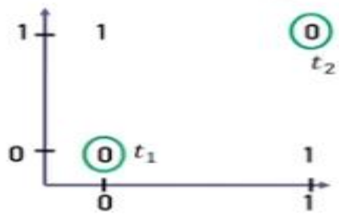
RBF NN for the XOR problem

$$\varphi_1(x) = e^{-\|x-\mu_1\|^2} \quad \text{with } \mu_1 = (0,0) \text{ and } \mu_2 = (1,1)$$

$$\varphi_2(x) = e^{-\|x-\mu_2\|^2}$$



Pattern	X1	X2	φ1	φ2
1	0	0	1	0.135
2	0	1	.36	.36
3	1	0	.36	.36
4	1	1	.135	1

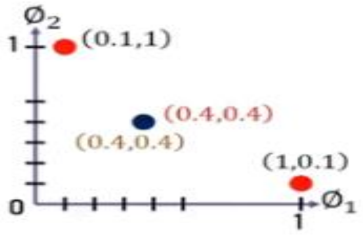


$$\emptyset_i(r) = \frac{exp[-r^2]}{2\sigma^2}, \quad \sigma > 0$$

$t_1 = (0,0)$ $\quad \boxed{\emptyset(X) = e^{-\|x-t\|^2}}$ $\quad$ Let, $2\sigma^2 = 1$

$t_2 = (1,1)$ $\quad \emptyset_1 = e^{-\|x-t_1\|^2}$

$\emptyset_2 = e^{-\|x-t_2\|^2}$



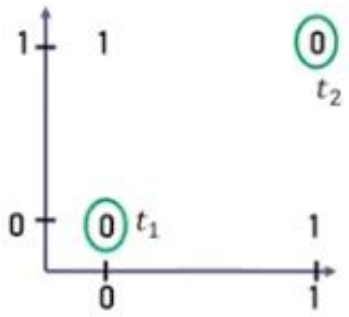
		$\emptyset_1$	$\emptyset_2$
0	0	1	0.1
0	1	0.4	0.4
1	0	0.4	0.4
1	1	0.1	1

$\emptyset_1(X) = e^{-((0-0)^2+(0-0)^2)} = 1$

where$(x_1, x_2) = (0{,}0)$and $t_1 = (0{,}0)$

$\emptyset_2(X) = e^{-((0-1)^2+(0-1)^2)} = 0.1$

where$(x_1, x_2) = (0{,}0)$and $t_2 = (1{,}1)$

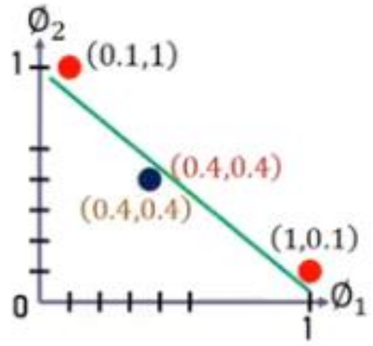


$$\emptyset_i(r) = \frac{exp[-r^2]}{2\sigma^2}, \quad \sigma > 0$$

$t_1 = (0,0)$ $\quad \boxed{\emptyset(X) = e^{-\|x-t\|^2}}$ $\quad$ Let, $2\sigma^2 = 1$

$t_2 = (1,1)$ $\quad \emptyset_1 = e^{-\|x-t_1\|^2}$

$\emptyset_2 = e^{-\|x-t_2\|^2}$



		$\emptyset_1$	$\emptyset_2$
0	0	1	0.1
0	1	0.4	0.4
1	0	0.4	0.4
1	1	0.1	1

$\emptyset_1(X) = e^{-((0-0)^2+(0-0)^2)} = 1$

where$(x_1, x_2) = (0{,}0)$and $t_1 = (0{,}0)$

$\emptyset_2(X) = e^{-((0-1)^2+(0-1)^2)} = 0.1$

where$(x_1, x_2) = (0{,}0)$and $t_2 = (1{,}1)$

Training

- Self organized selection of centres
- K-means algorithm
  - Initialization : Create random cluster centre points
  - Sampling: Draw a sample vector x from a input space
  - Similarity Matching: Index of the winning cluster centre

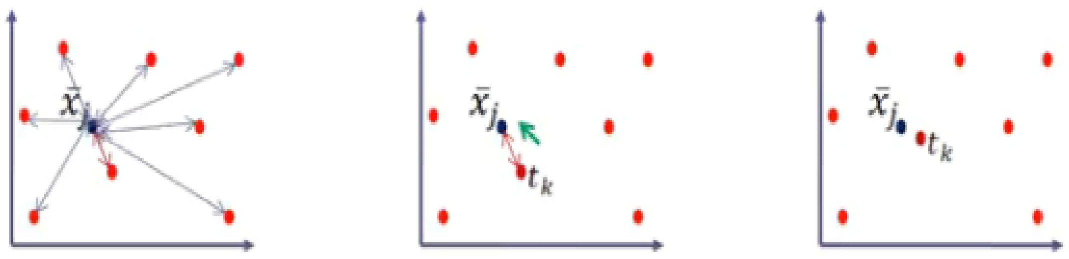
$$K\left(\underset{x}{\rightarrow}\right) = arg\ \min_k \|X - t_k\|$$

  - Updation:

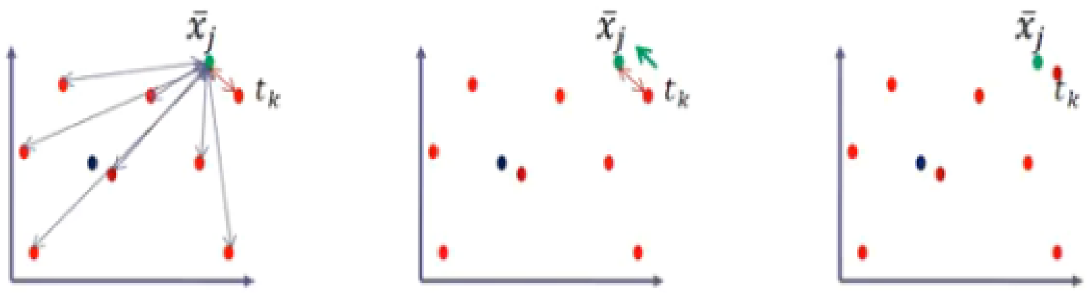
$$t_k(n+1) = \begin{cases} t_k(n) + \eta[\bar{x}(n) - t_{k(n)}], & if\ K = K(x) \\ t_{k(n)}, & if\ K \neq K(x) \end{cases}$$

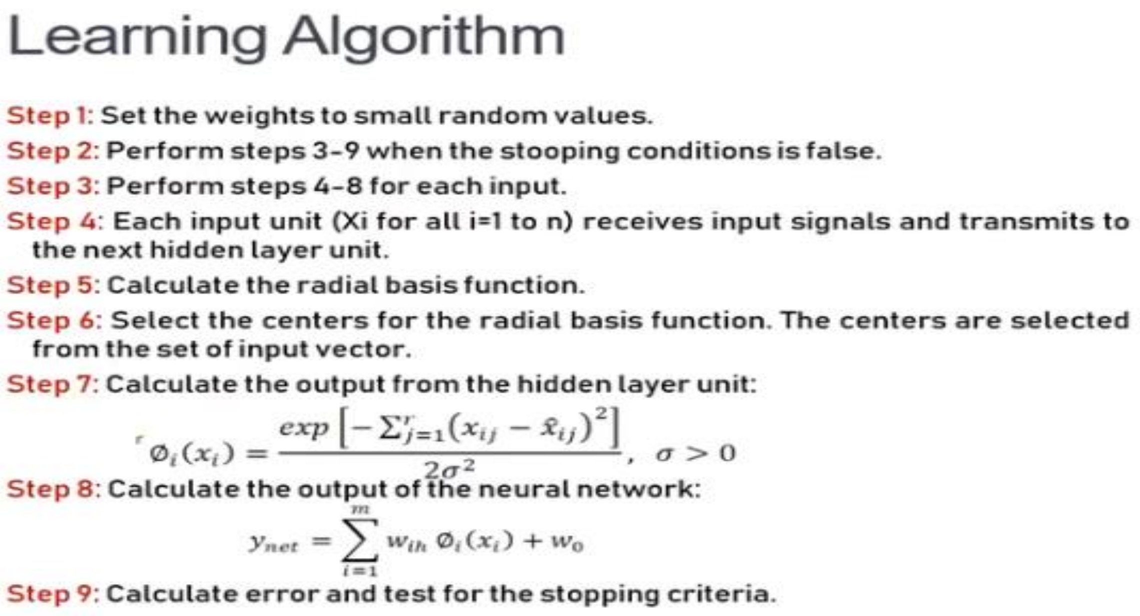
$S = \{\bar{x}_1, \bar{x}_2, \dots \bar{x}_n\}$

I/P 01



I/P 02





Learning Algorithm

Step 1: Set the weights to small random values.

Step 2: Perform steps 3-9 when the stooping conditions is false.

Step 3: Perform steps 4-8 for each input.

Step 4: Each input unit (Xi for all i=1 to n) receives input signals and transmits to the next hidden layer unit.

Step 5: Calculate the radial basis function.

Step 6: Select the centers for the radial basis function. The centers are selected from the set of input vector.

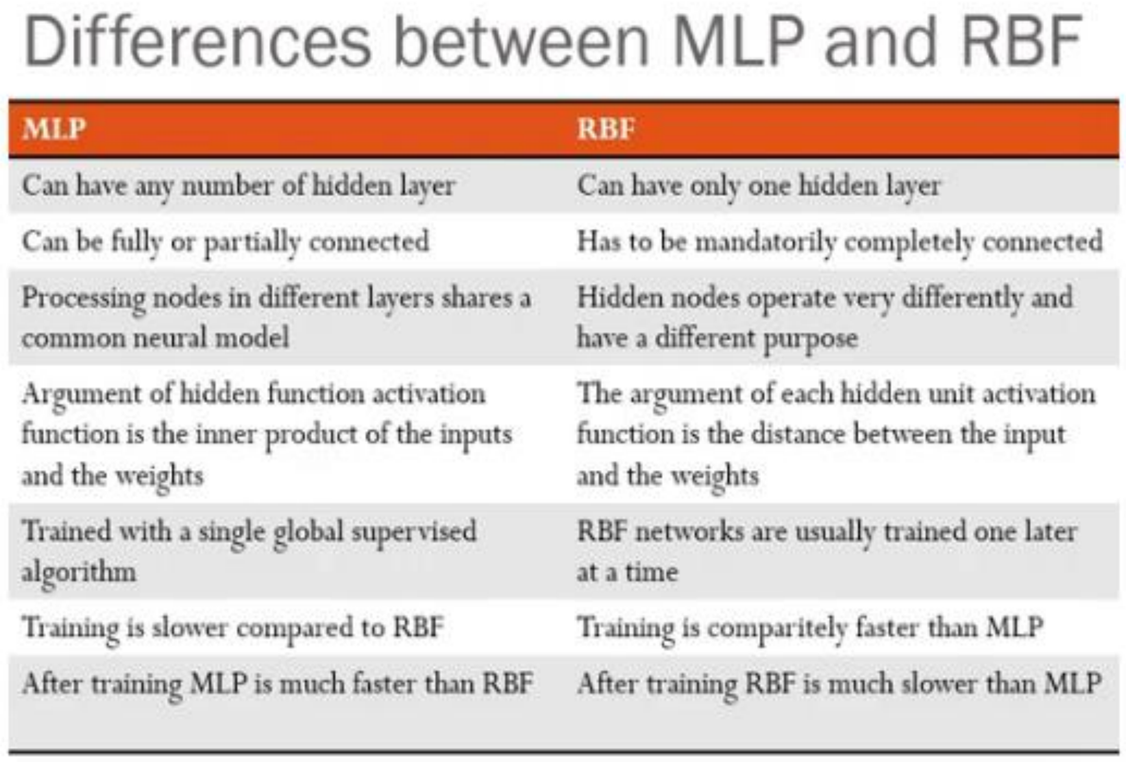
Step 7: Calculate the output from the hidden layer unit:

$$\emptyset_i(x_i) = \frac{exp\left[-\sum_{j=1}^{r}(x_{ij} - \hat{x}_{ij})^2\right]}{2\sigma^2}, \quad \sigma > 0$$

Step 8: Calculate the output of the neural network:

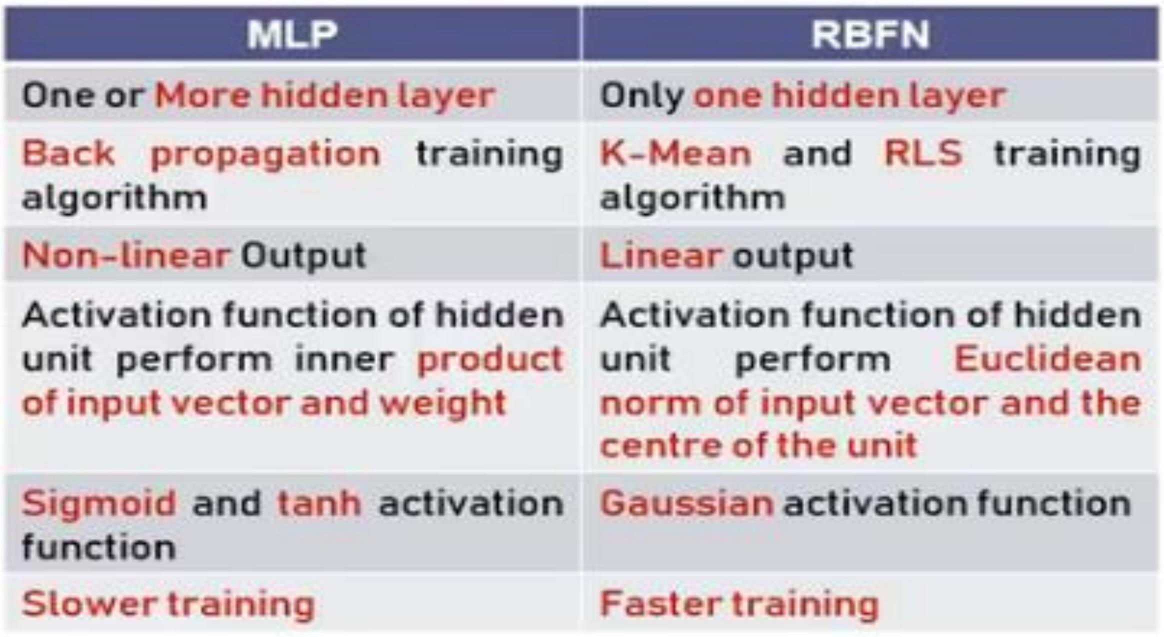
$$y_{net} = \sum_{i=1}^{m} w_{ih}\, \emptyset_i(x_i) + w_0$$

Step 9: Calculate error and test for the stopping criteria.



Differences between MLP and RBF

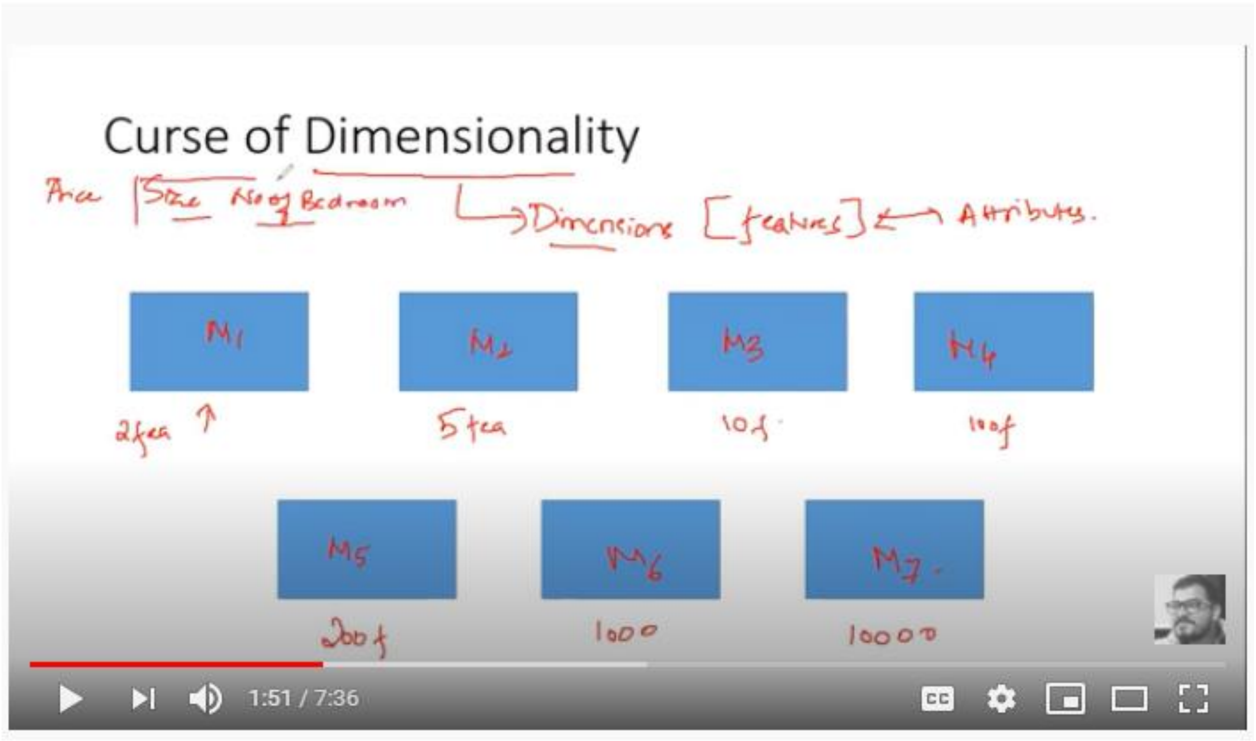
MLP	RBF
Can have any number of hidden layer	Can have only one hidden layer
Can be fully or partially connected	Has to be mandatorily completely connected
Processing nodes in different layers shares a common neural model	Hidden nodes operate very differently and have a different purpose
Argument of hidden function activation function is the inner product of the inputs and the weights	The argument of each hidden unit activation function is the distance between the input and the weights
Trained with a single global supervised algorithm	RBF networks are usually trained one later at a time
Training is slower compared to RBF	Training is comparitely faster than MLP
After training MLP is much faster than RBF	After training RBF is much slower than MLP

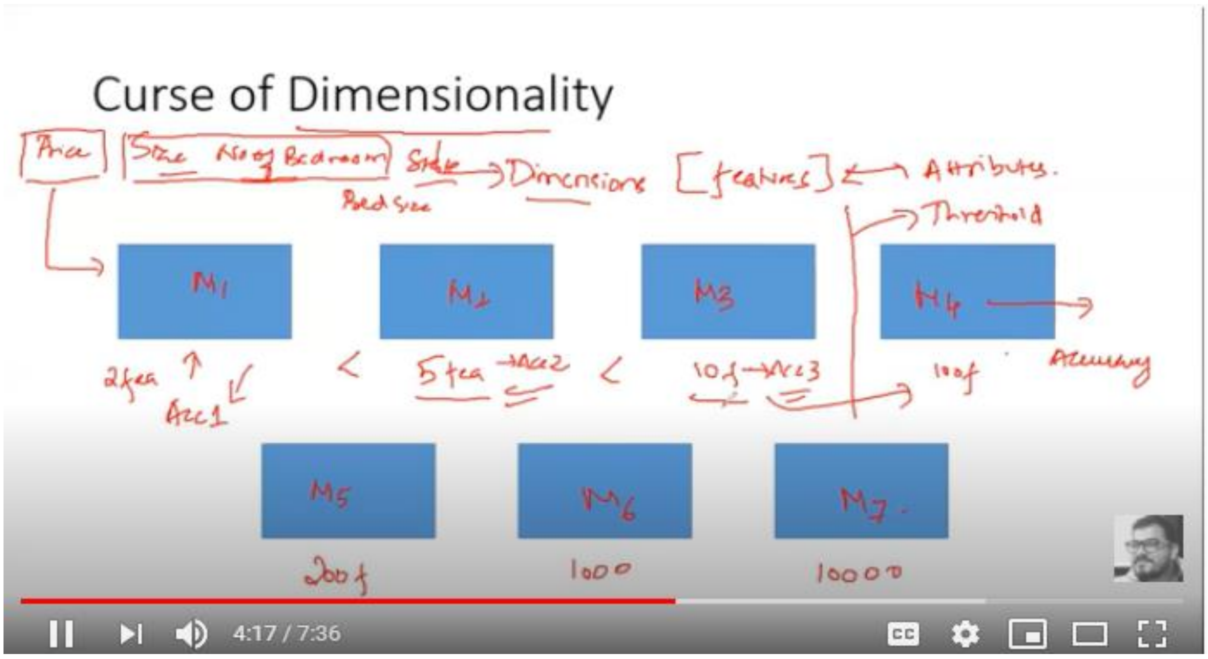


MLP	RBFN
One or More hidden layer	Only one hidden layer
Back propagation training algorithm	K-Mean and RLS training algorithm
Non-linear Output	Linear output
Activation function of hidden unit perform inner product of input vector and weight	Activation function of hidden unit perform Euclidean norm of input vector and the centre of the unit
Sigmoid and tanh activation function	Gaussian activation function
Slower training	Faster training

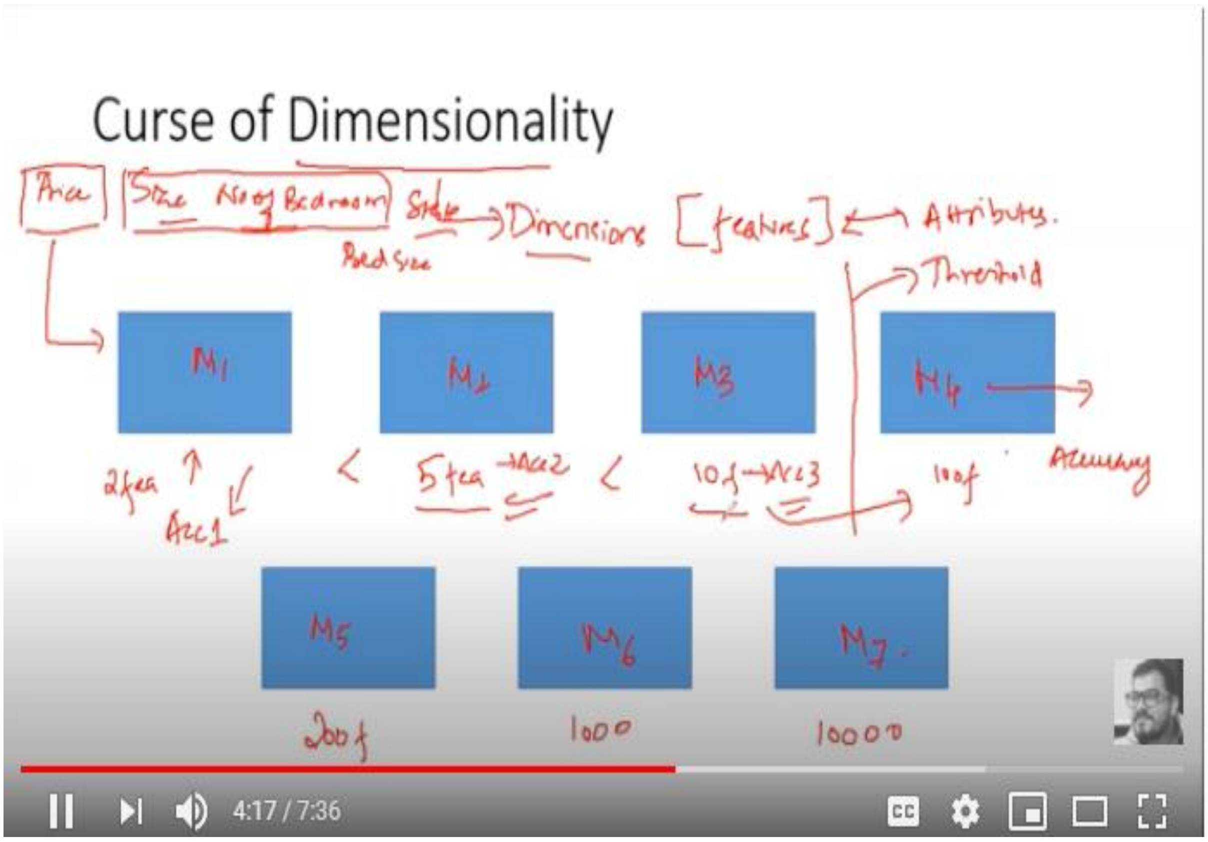
Curse of Dimensionality

As the Number of features or dimension grows. The amount of data we need to generate accurately grows exponentially. Feature selection and feature engineering. The dimension also called features. The features independent features or target output features. Features basically attributes





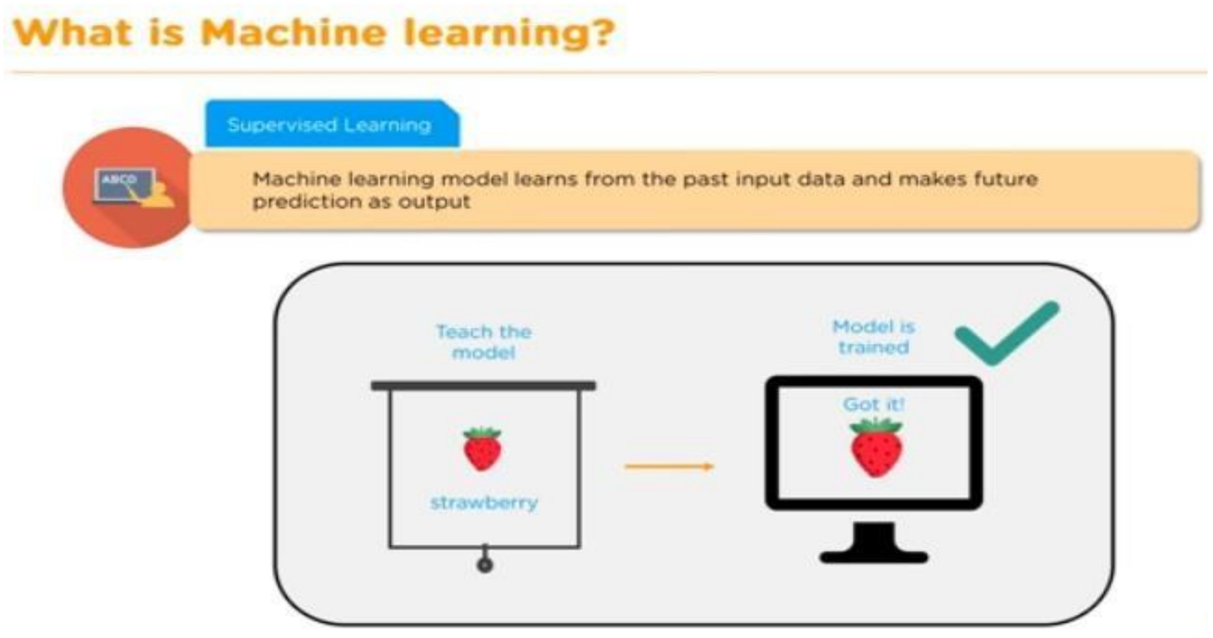
Curse of Dimensionality

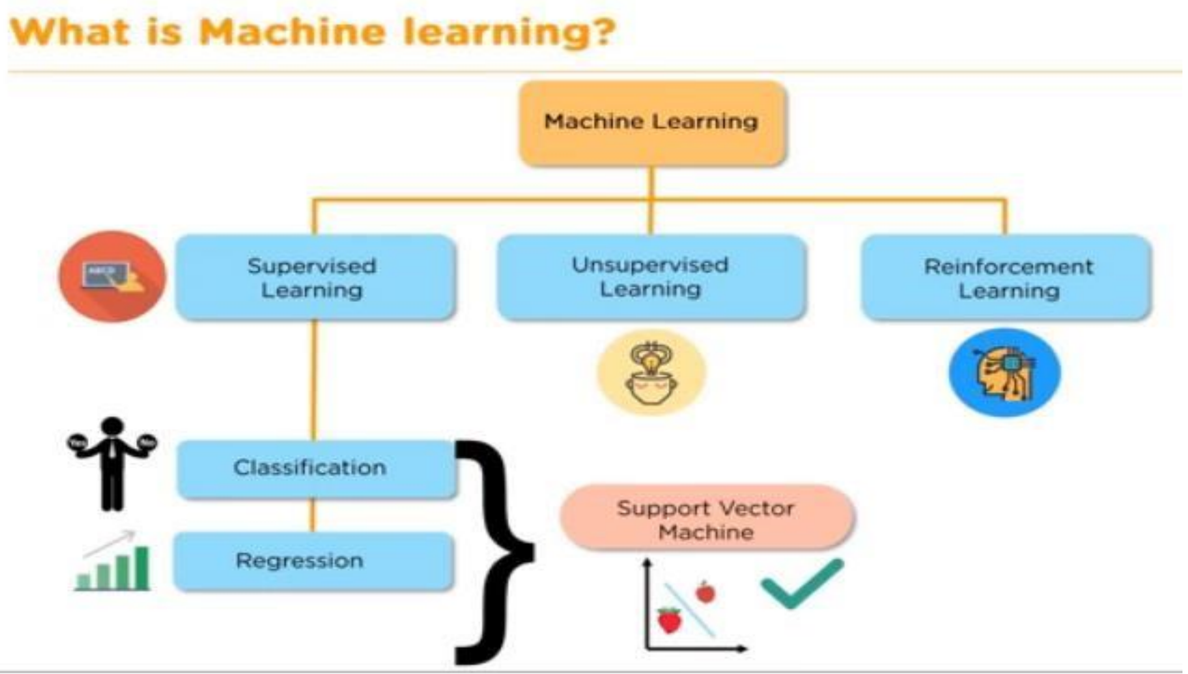


Curse of Dimensionality

Model learn more features exponentially get confusion. Once it reach threshold value, The accuracy not changed. If feature is increasing exponentially from 100 to 200 or 1000. The accuracy is decreased is called curse of dimensionality,

Support Vector Machine


What is Machine learning?
Supervised Learning
ABCD
Machine learning model learns from the past input data and makes future prediction as output
Teach the model
strawberry
Model is trained
Got it!



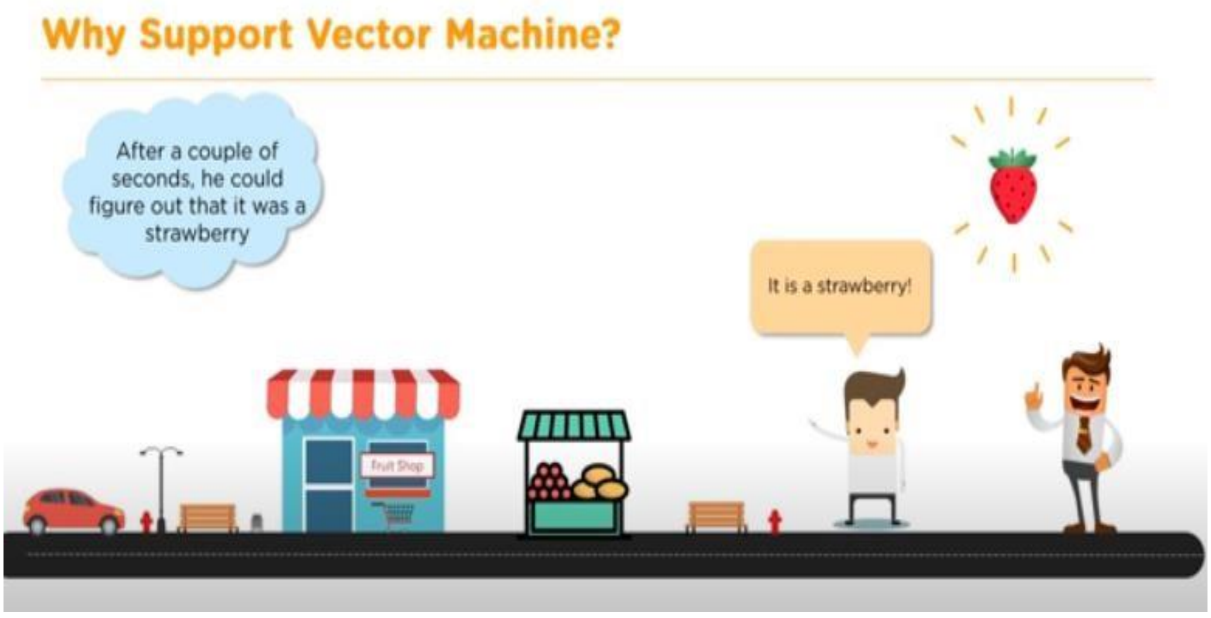
Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression.

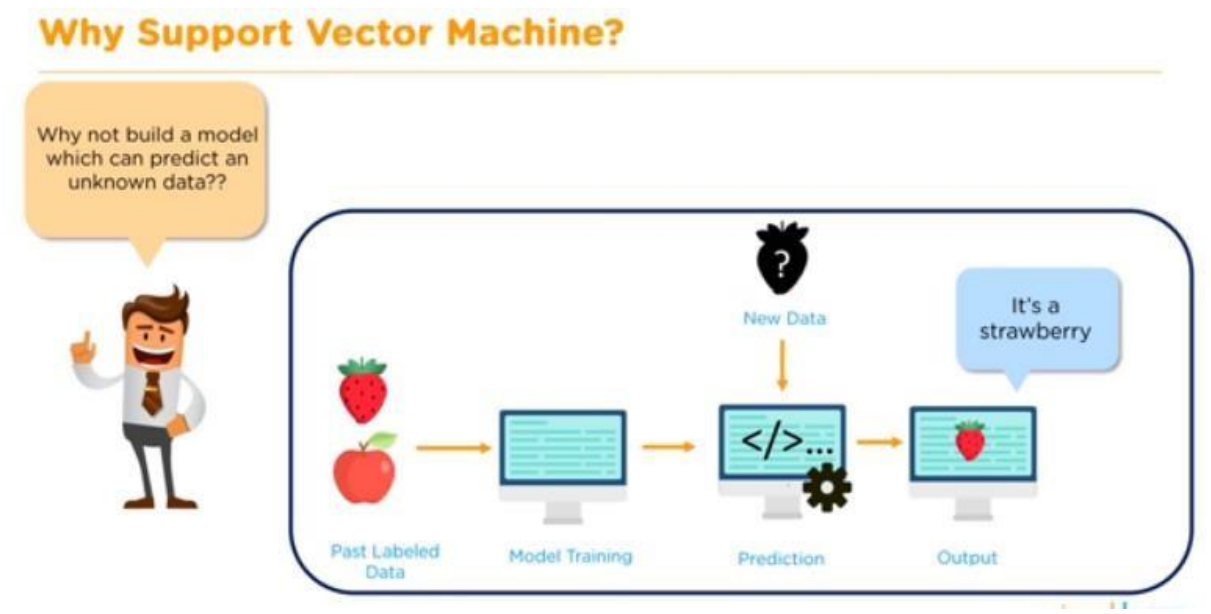
But generally, they are used in classification problems. In 1960s, SVMs were first introduced but later they got refined in 1990. SVMs have their unique way of implementation as compared to other machine learning algorithms. Lately, they are extremely popular because of their ability to handle multiple continuous and categorical variables.

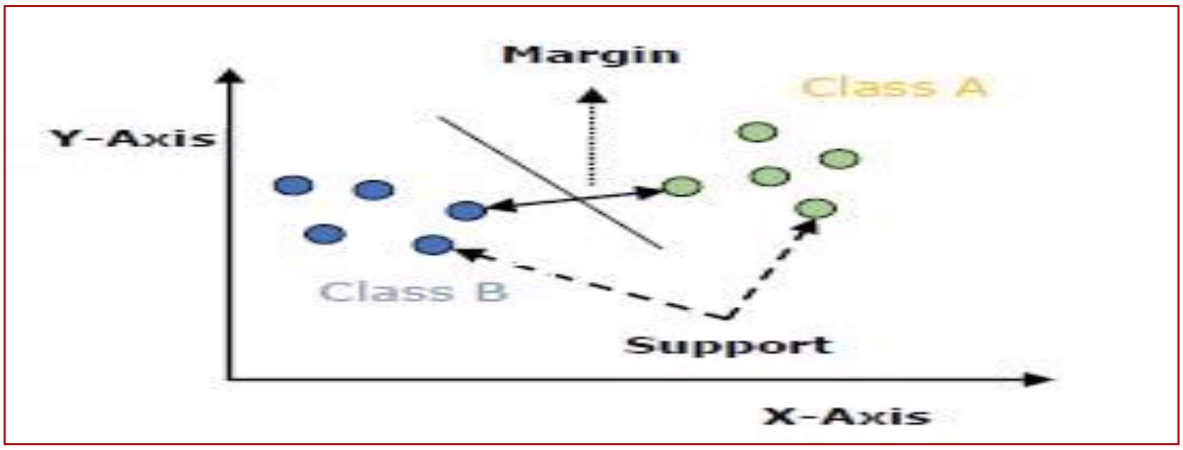
Working of SVM

An SVM model is basically a representation of different classes in a hyperplane in multidimensional space. The hyperplane will be generated in an iterative manner by SVM so that the error can be minimized.

The goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH).


Why Support Vector Machine?
After a couple of seconds, he could figure out that it was a strawberry
It is a strawberry!
Fruit Shop


Why Support Vector Machine?
Why not build a model which can predict an unknown data??
New Data
It's a strawberry
Past Labeled Data
Model Training
Prediction
Output

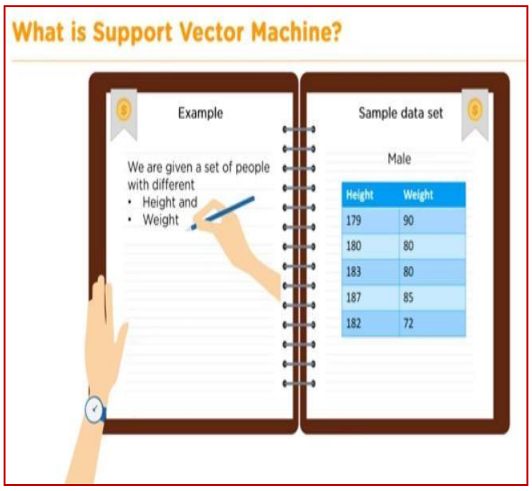


•Support Vectors − Datapoints that are closest to the hyperplane is called support vectors. Separating line will be defined with the help of these data points.

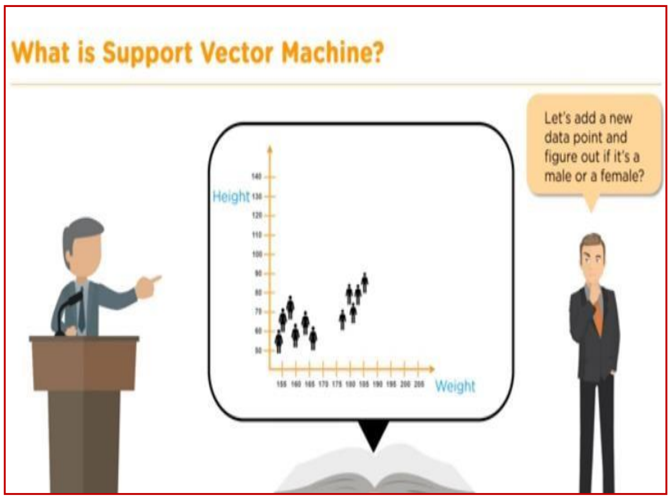
•Hyperplane − As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.

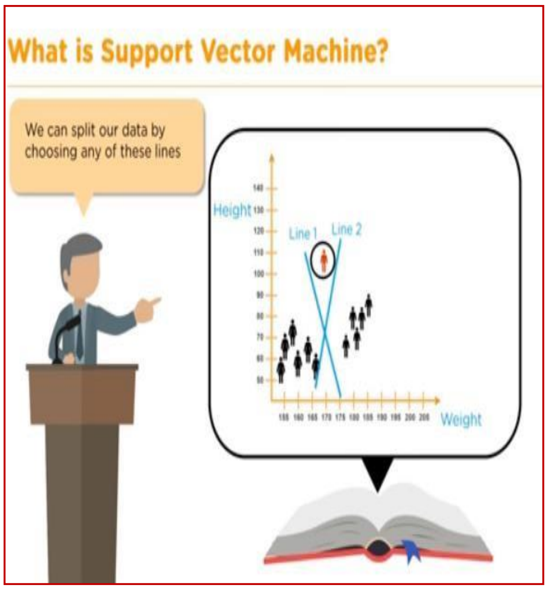
•Margin − It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin. Support Vector Machine is a supervised learning method , it is a discriminative classifier that is formally designed by a separative hyperplane.

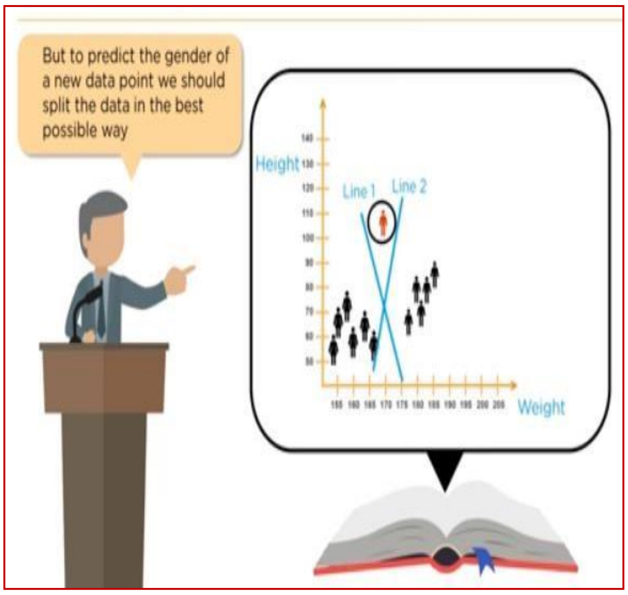
It is a representation of examples as points in space that are mapped so that the points of different categories are separated by a gap as SVM is the extreme points in the dataset Hyperplane is the maximum distance to the support vectors of any class.

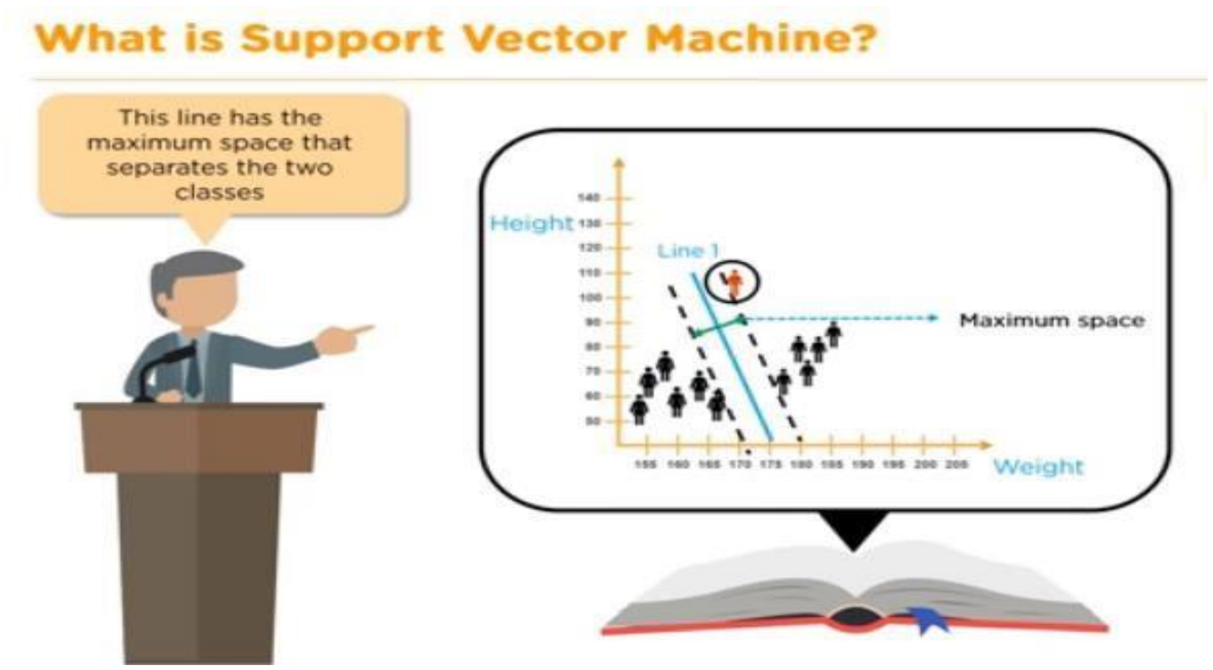

What is Support Vector Machine?
Example
We are given a set of people with different
• Height and
• Weight
Sample data set
Male

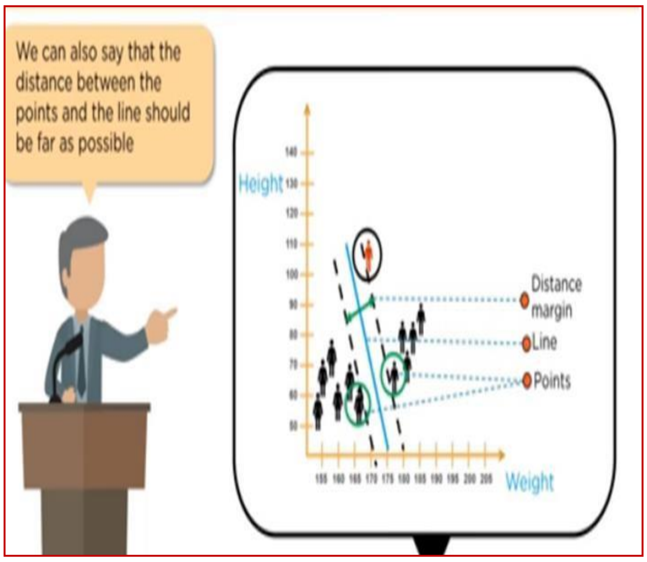
Height	Weight
179	90
180	80
183	80
187	85
182	72

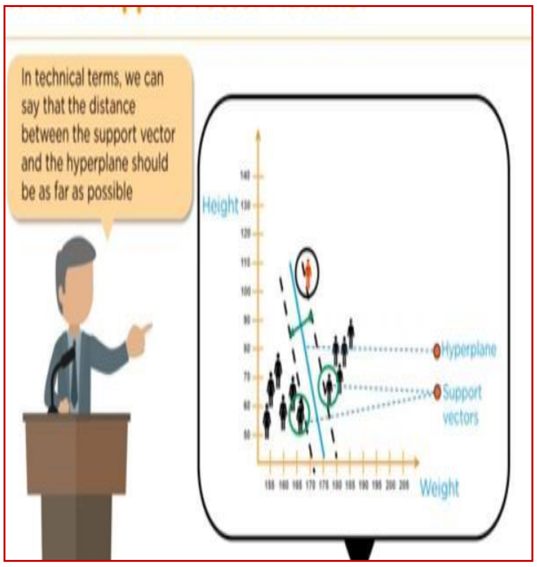

What is Support Vector Machine?
Let's add a new data point and figure out if it's a male or a female?
Height
Weight

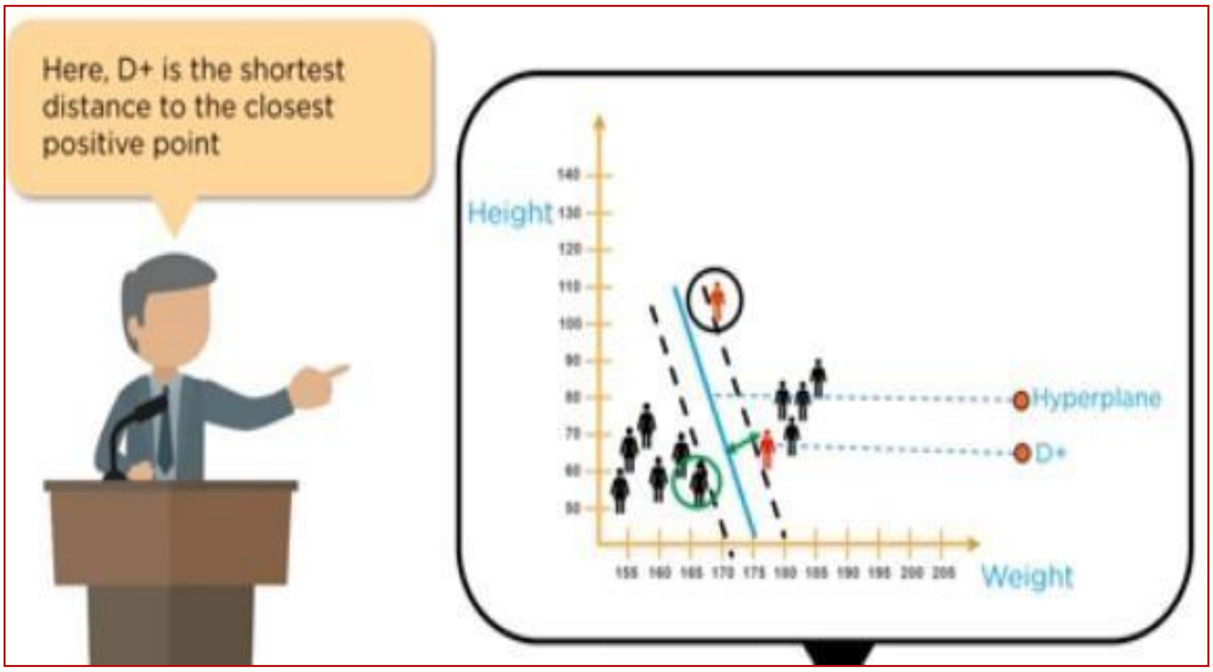

What is Support Vector Machine?
We can split our data by choosing any of these lines
Height
Line 1
Line 2
Weight

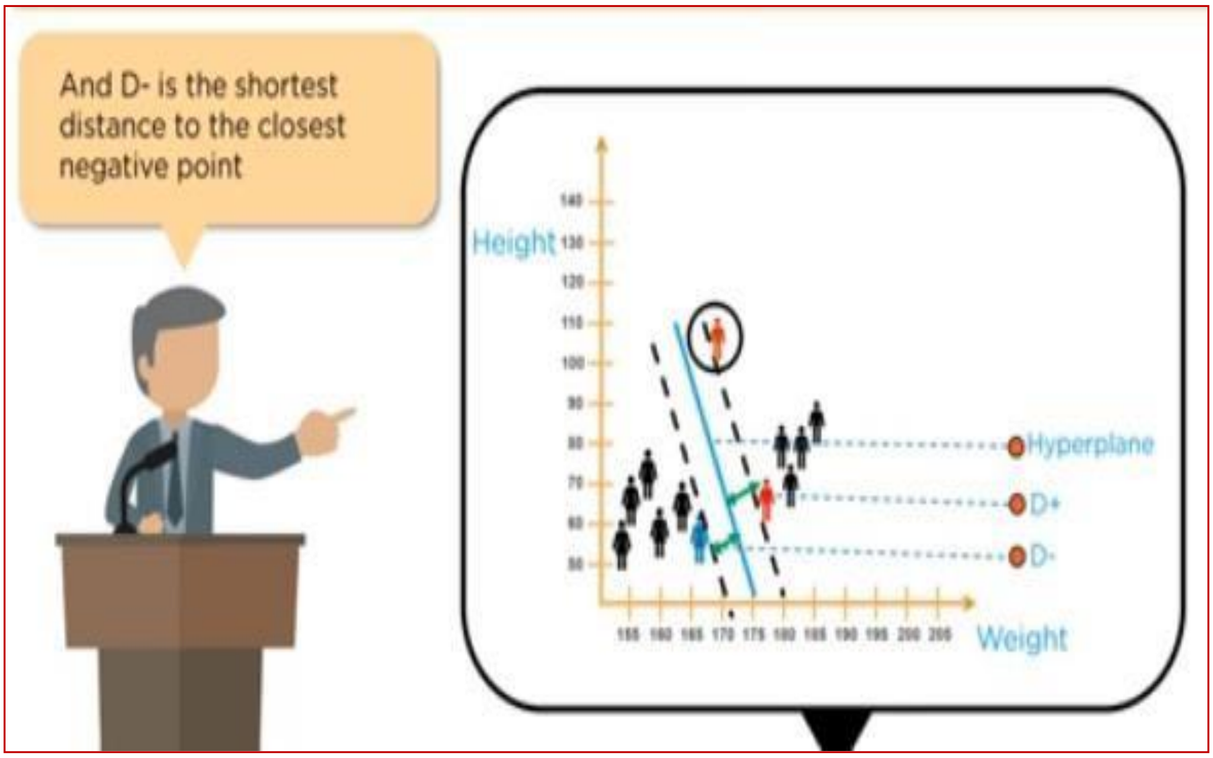

But to predict the gender of a new data point we should split the data in the best possible way
Height
Line 1
Line 2
Weight

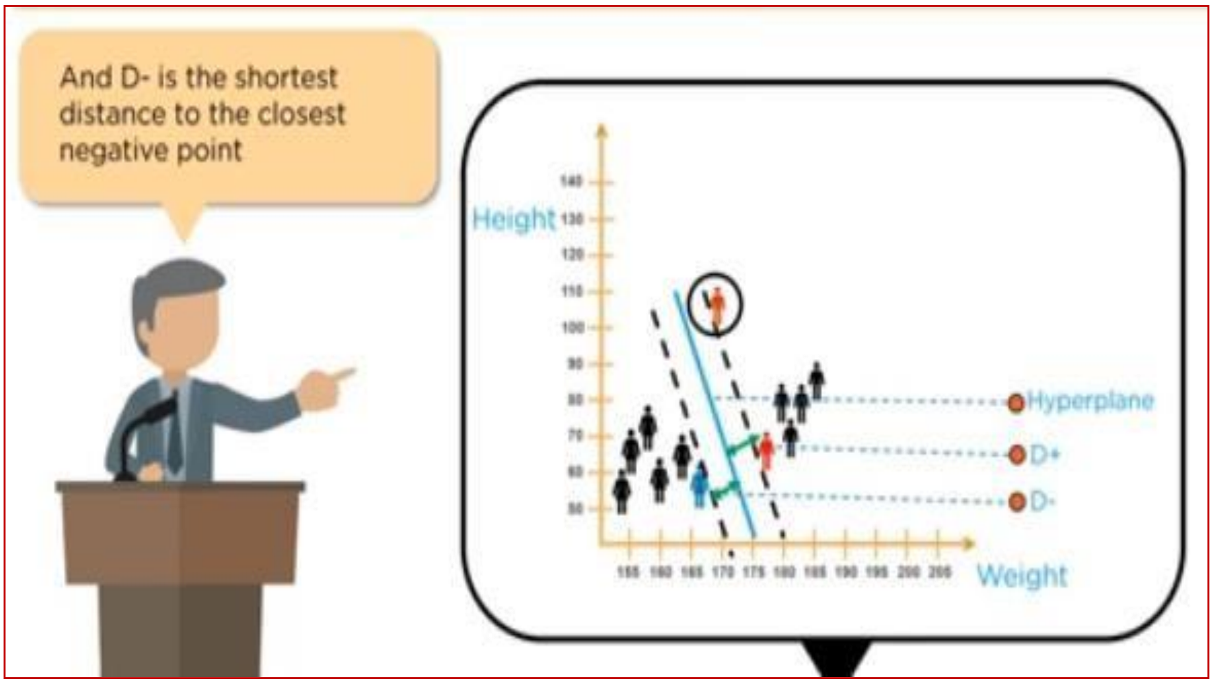

What is Support Vector Machine?
This line has the maximum space that separates the two classes
Height
Line 1
Maximum space
Weight


We can also say that the distance between the points and the line should be far as possible
Height
Weight
Distance margin
Line
Points


In technical terms, we can say that the distance between the support vector and the hyperplane should be as far as possible
Height
Weight
Hyperplane
Support vectors

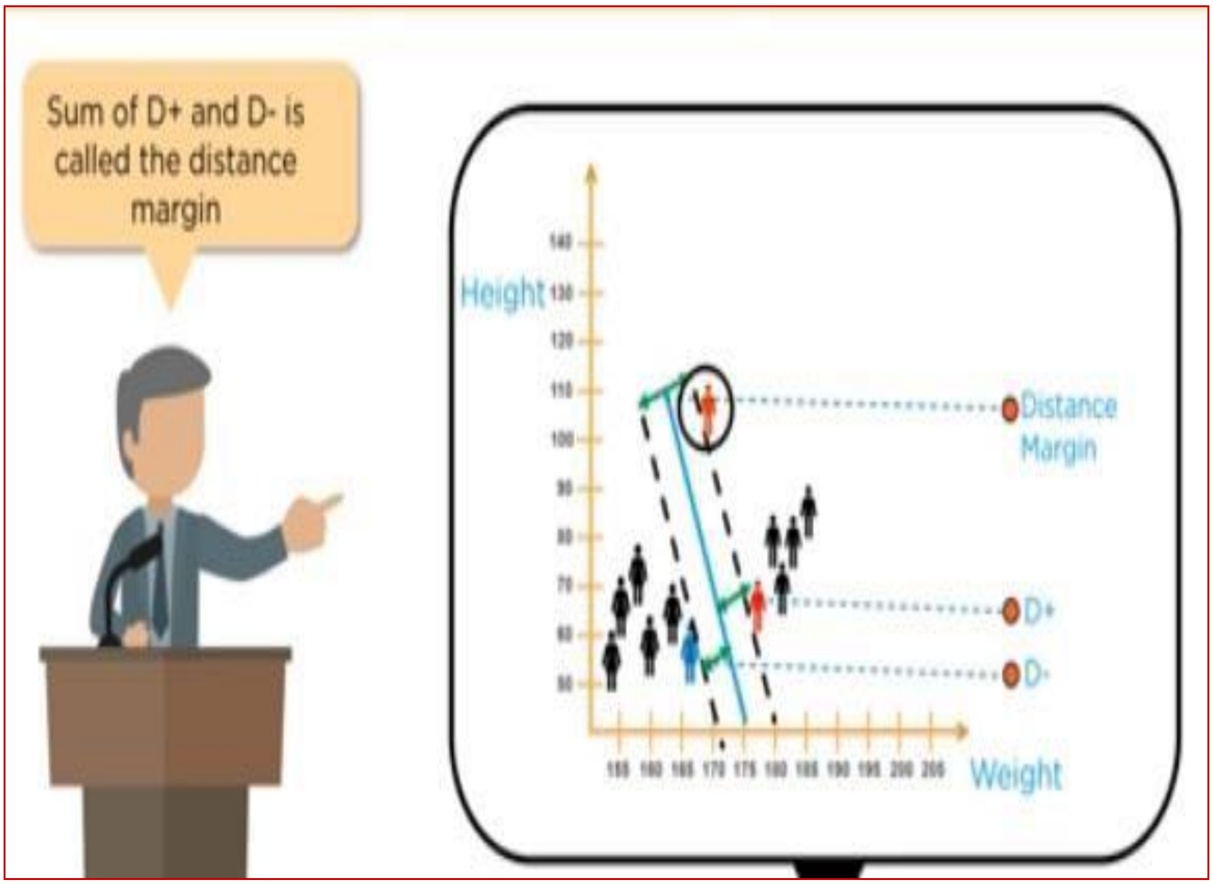

Here, D+ is the shortest distance to the closest positive point
Height
Weight
Hyperplane
D+

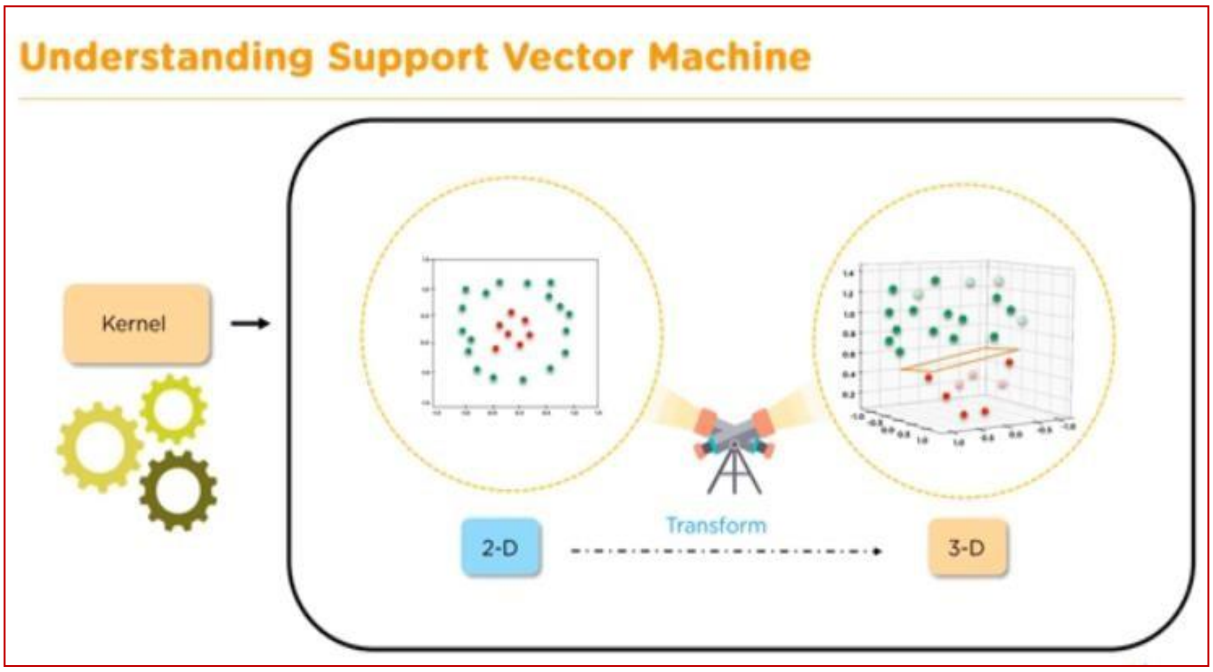

And D- is the shortest distance to the closest negative point
Height
Weight
Hyperplane
D+
D-

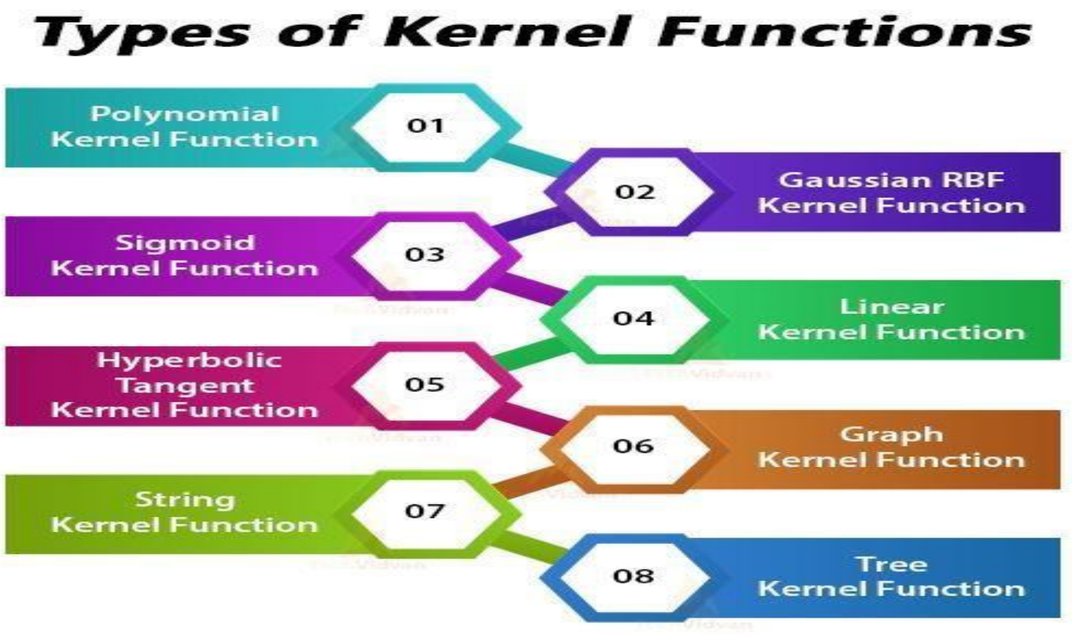

And D- is the shortest distance to the closest negative point
Height
Weight
Hyperplane
D+
D-

- From the distance margin – it get the optimal hyperplane.
- Based on the hyperplane , it can say the new data point belongs to male gender.

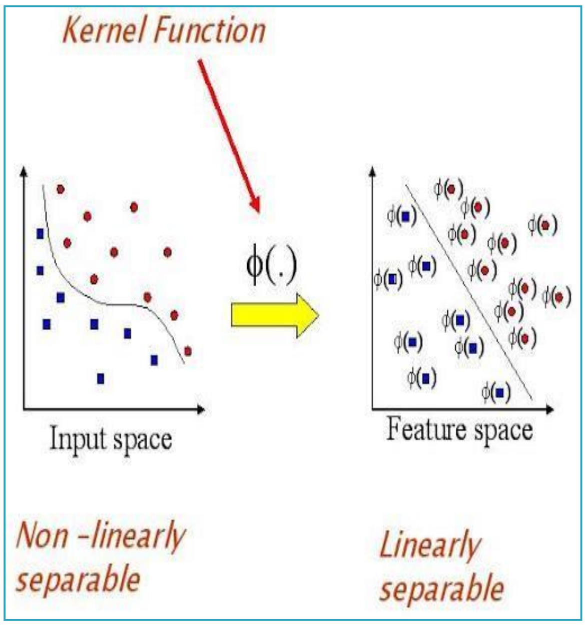
- If select a hyperplane having low margin then there is high chance of misclassification



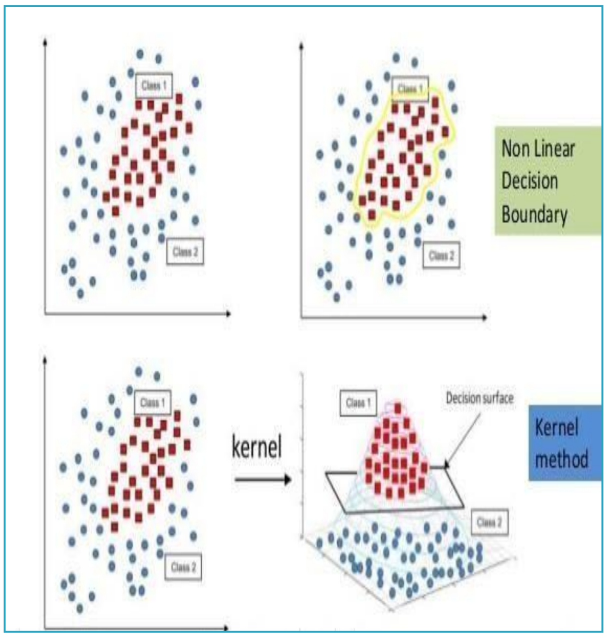




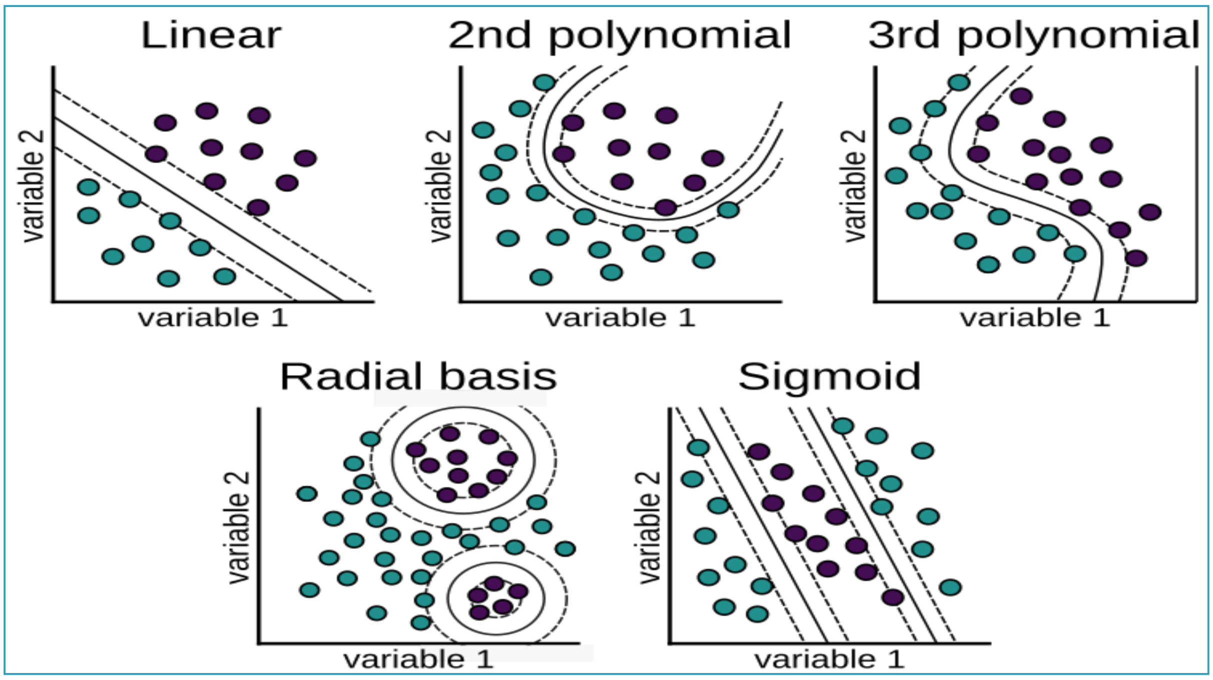
- Common kernel functions for SVM
  - linear $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 \cdot \mathbf{x}_2$
  - polynomial $k(\mathbf{x}_1, \mathbf{x}_2) = (\gamma\, \mathbf{x}_1 \cdot \mathbf{x}_2 + c)^d$
  - Gaussian or radial basis $k(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\gamma \|\mathbf{x}_1 - \mathbf{x}_2\|^2\right)$
  - sigmoid $k(\mathbf{x}_1, \mathbf{x}_2) = \tanh(\gamma\, \mathbf{x}_1 \cdot \mathbf{x}_2 + c)$



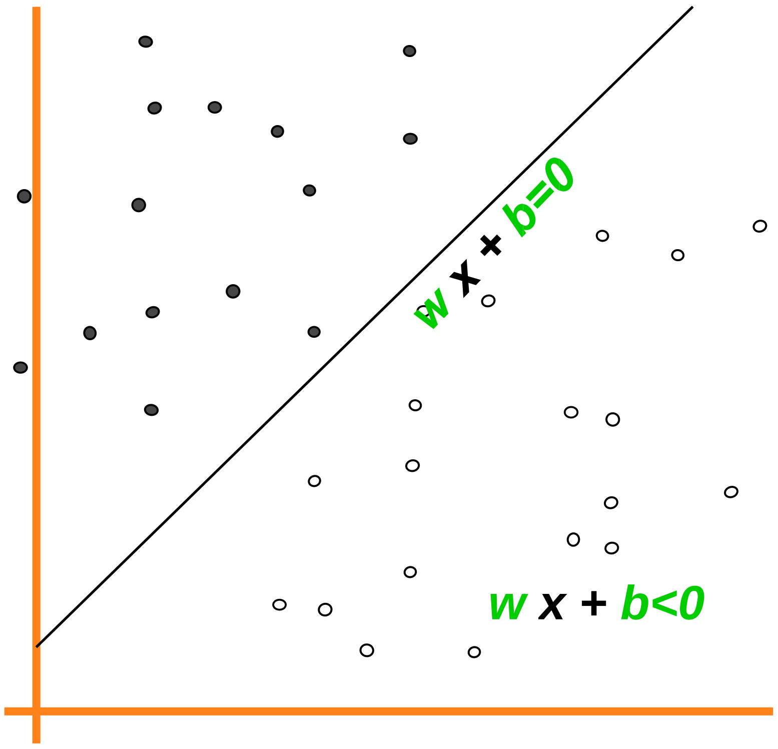
Kernel Function
ϕ(.)
Input space
Feature space
Non -linearly separable
Linearly separable



Class 1
Class 2
Non Linear Decision Boundary
kernel
Decision surface
Kernel method



Linear
2nd polynomial
3rd polynomial
Radial basis
Sigmoid
variable 1
variable 2



w x + b=0
w x + b<0

Support Vectors with the maximum are those margin. datapoints that the margin   This is the pushes up simplest kind of against SVM (Called an LSVM)Linear SVM

Linear SVM Mathematically

■ Goal: 1) Correctly classify all training data

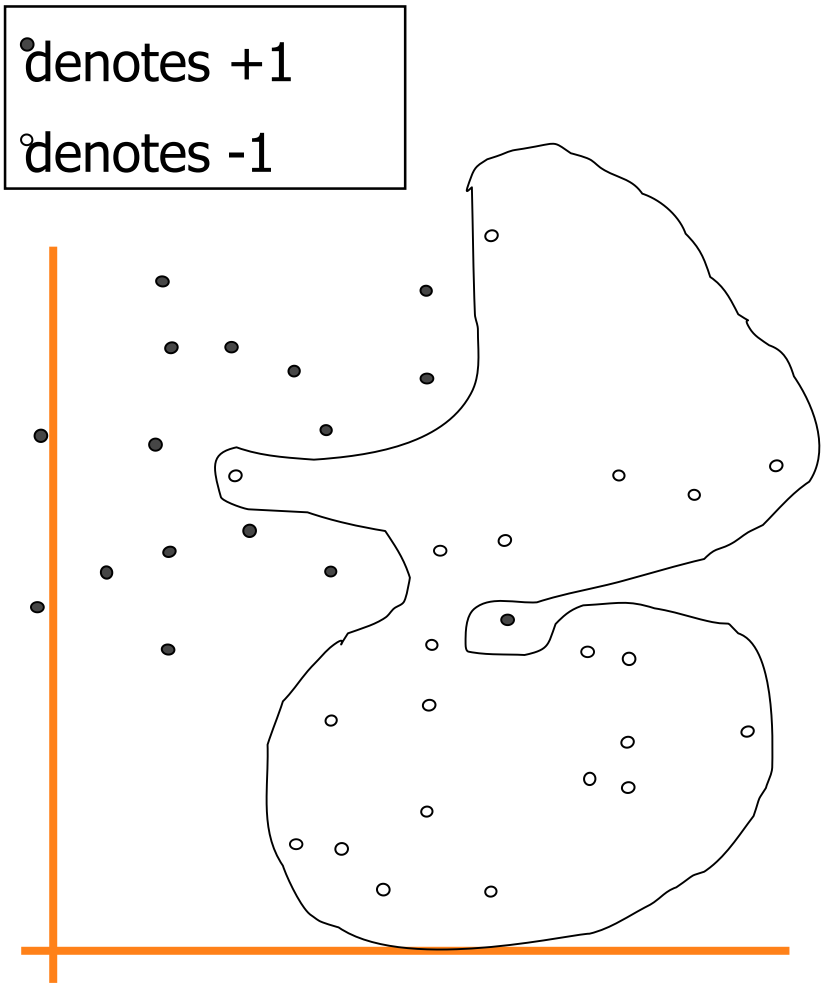
2) Maximize the Margin same as minimize

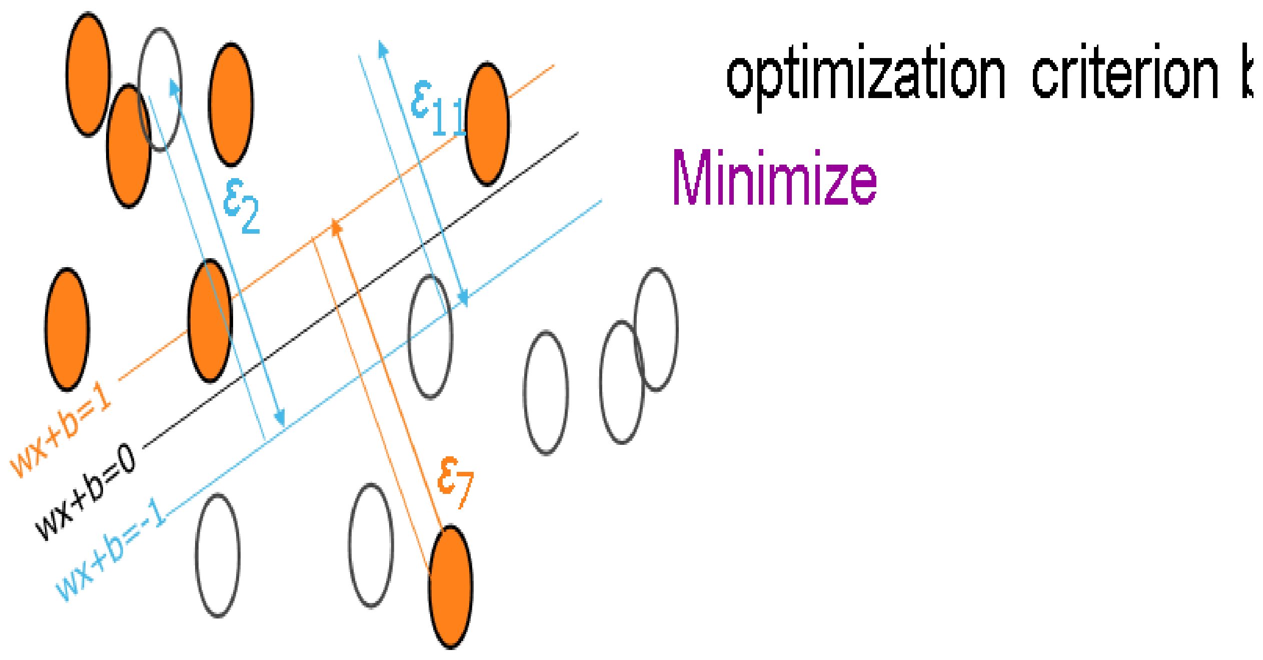
■ We can formulate a Quadratic Optimization Problem and solve for w and b What if the training set is noisy? - Solution 1: use very powerful kernels

OVERFITTI NG!

Soft Margin Classification

Slack variables ξi can be added to allow misclassification of difficult or noisy examples





Hard Margin v.s. Soft Margin

■ The old formulation: Find w and b such that

Φ(w) =½ wTw is minimized and for all {(xi ,yi)} yi (wTxi + b) ≥ 1

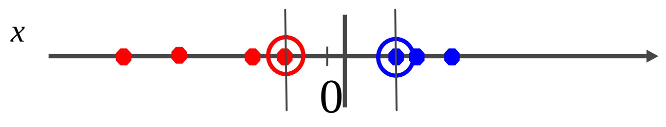
■ The new formulation incorporating slack variables: Find w and b such that

Φ(w) =½ wTw + CΣξi is minimized and for all {(xi ,yi)} yi (wTxi + b) ≥ 1- ξi and ξi ≥ 0 for all i

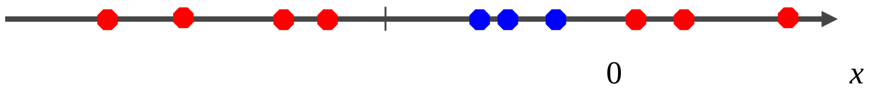
■ **Parameter *C* can be viewed as a way to control overfitting.**

Non-linear SVMs

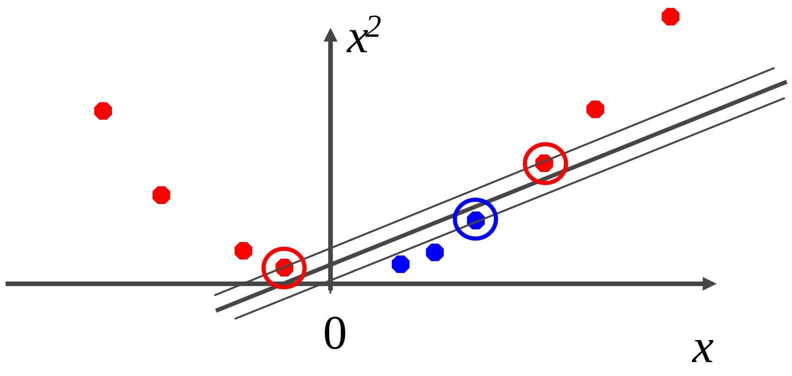
■ Datasets that are linearly separable with some noise work out great:



■ But what are we going to do if the dataset is just too hard?



■ How about… mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

■ General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable: Nonlinear SVM - Overview

■ SVM locates a separating hyperplane in the feature space and classify points in that space

■ It does not need to represent the space explicitly, simply by defining a kernel function

■ The kernel function plays the role of the dot product in the feature space.

SVM Applications

SVM has been used successfully in many real-world problems

Weakness of SVM

It is sensitive to noise

A relatively small number of mis labelled examples can dramatically decrease the performance. It only considers two classes, how to do multi-class classification with SVM?

Answer:

1) with output parity m, learn m SVM's

◦ SVM 1 learns "Output==1" vs "Output != 1"

◦ SVM 2 learns "Output==2" vs "Output != 2" ◦ :

◦ SVM m learns "Output==m" vs "Output != m"

2)To predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

Application 2: Text Categorization

Task: The classification of natural text (or hypertext) documents into a fixed number of predefined categories based on their content. email filtering, web searching, sorting documents by topic, etc. A document can be assigned to more than one category, so this can be viewed as a series of binary classification problems, one for each category

Representation of Text

IR's vector space model (aka bag-of-words representation) ■ A doc is represented by a vector indexed by a pre-fixed set or dictionary of terms

- ■ Values of an entry can be binary or weights
- ■ Normalization, stop words, word stems
- ■ Doc x => φ(x)

Text Categorization using SVM

The distance between two documents is φ(x)·φ(z) K(x,z) = 〈φ(x)·φ(z) is a valid kernel, SVM can be used with K(x,z) for discrimination.

Why SVM?

High dimensional input space

Few irrelevant features (dense concept)

Sparse document vectors (sparse instances)

Text categorization problems are linearly separable, Some Issues

Choice of kernel

Gaussian or polynomial kernel is default if ineffective, more elaborate kernels are needed domain experts can give assistance in formulating appropriate similarity measures Choice of kernel parameters. e.g. σ in Gaussian kernel, σ is the distance between closest points with different classifications . In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.

Pros and Cons of SVM Classifiers

Pros of SVM classifiers

SVM classifiers offers great accuracy and work well with high dimensional space. SVM classifiers basically use a subset of training points hence in result uses very less memory.

Cons of SVM classifiers

They have high training time hence in practice not suitable for large datasets. Another disadvantage is that SVM classifiers do not work well with overlapping classes.

Key terms include:

- **Epoch** — an arbitrary cut off, generally defined as "one pass over the entire dataset", used to separate training into distinct phases, which is useful for logging and periodic evaluation. In layman's term, a number of epochs means how many times you go through your training set.
- **Learning Rate** — "a scalar used to train a model via gradient descent. During each iteration, the **gradientdescent** algorithm multiplies the learning rate by the gradient. The resulting product is called the **gradient step**. Learning rate is a key **hyperparameter**." Specifying the learning rate is equivalent to determining **how fast** weights change for each iteration. In Tensorflow playground, the learning rate ranges from 0.00001 to 10.
- **Activation Function** — the output of that node, or "neuron," given an input or set of inputs. This output is then used as input for the next node and so on until a desired solution to the original problem is found. Available activation functions in Tensorflow playground are ReLU, Tanh, Sigmoid, and Linear.
- **Regularization** — a hyperparameter to prevent overfitting. Available values are L1 and L2. **L1 computes the sum of the weights, whereas L2 takes the sum of the square of the weights.**

- **Regularization Rate** — a scalar used to specify the rate at which the model applies the regularization, ranging from 0 to 10.
- **Problem Type** — classification (categorical output) vs. regression (numerical output)
- **Ratio of the Training and Testing Sets** — the proportion of a subset to train a model and a subset to test a model. *I usually set it to 80/20*
- **Noise** — a distortion in data that is construed to be extraneous to the original data.
- **Batch Size** — "a small, randomly selected subset of the entire batch of **examples** run together in a single iteration of training or inference. The **batch size** of a mini-batch is usually between 10 and 1,000."
- **Features** — represents *an input layer* to feed in.
- **Hidden Layer** — a layer in between input layers and output layers, where artificial neurons take in a set of weighted inputs and produce an output through an activation function. In this context, you can specify as many as you want, but bear in mind that the more hidden layer you add, the more complex the model becomes.
- **Output** — an output layer in the neural network, often involving the loss evaluation. *Loss function (or a cost function) is a method of evaluating how well the neural network performs in the given data*. If predictions deviates too much from actual results, loss function will be high. We often evaluate the losses both on training and testing sets.

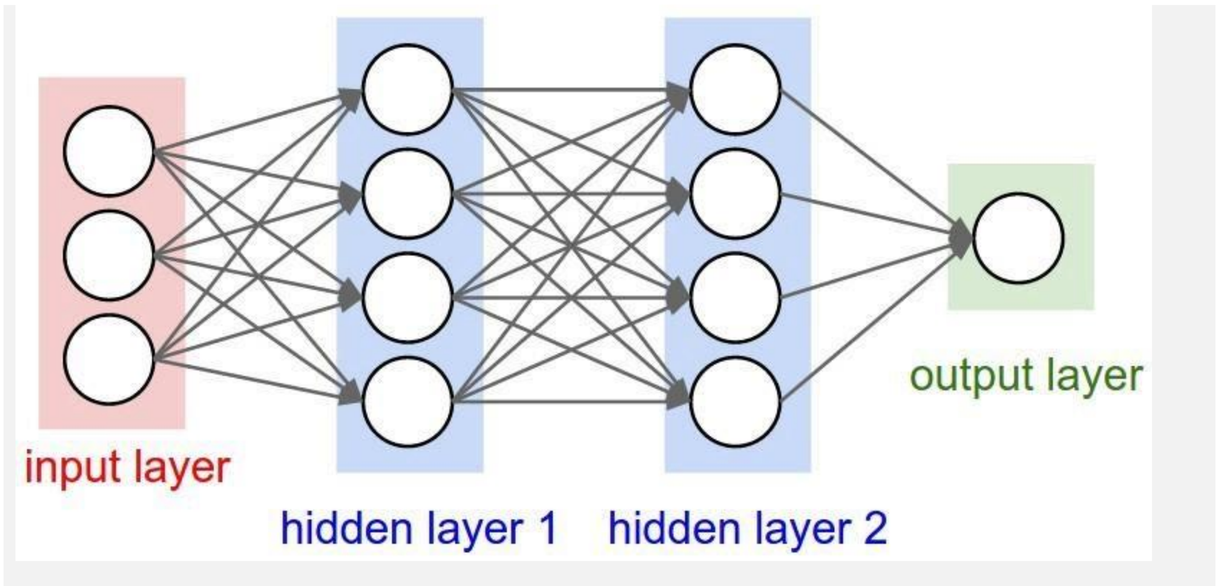


Figure 2: represents an artificial **neural network** (ANN) with multiple layers between the input and output layers. For example, given input data of image pixels from MNIST dataset, we can specify 2 hidden layers, each of which has 4 hidden neurons. Ultimately, we have the predicted probabilities of the possible number for the given image. Image Source: Deep learning — Convolutional neural networks and feature extraction with Python, Perone (2015)

UNIT – III TREE AND PROBABILISTIC MODELS

Learning with Trees – Decision Trees – Constructing Decision Trees – Classification and Regression Trees – Ensemble Learning – Boosting – Bagging – Different ways to Combine Classifiers – Probability and Learning – Data into Probabilities – Basic Statistics – Gaussian Mixture Models – Nearest Neighbor Methods – Unsupervised Learning – K means Algorithms – Vector Quantization – Self Organizing Feature Map.

A tree has many analogies in real life, and turns out that it has influenced a wide area of **machine learning**, covering both **classification and regression**. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal, its also widely used in machine learning.

Problems in Machine learning

Classification:

Problems with categorical solutions like 'yes' or 'No' ,'True' or 'False','1' or '0'.

Regression:

Problems wherein continuous value needs to be predicted like 'Product Prices', 'Profit'.

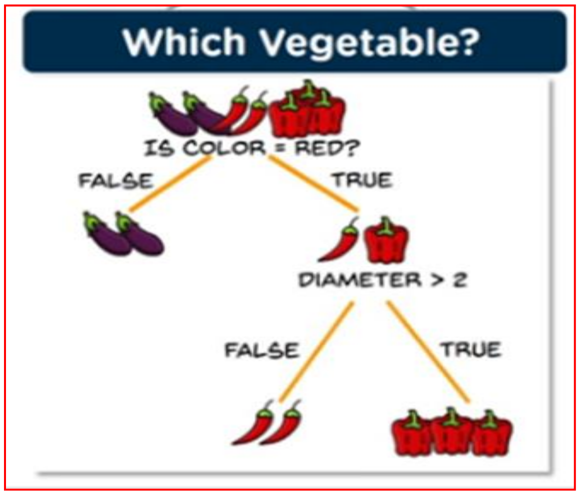
Clustering:

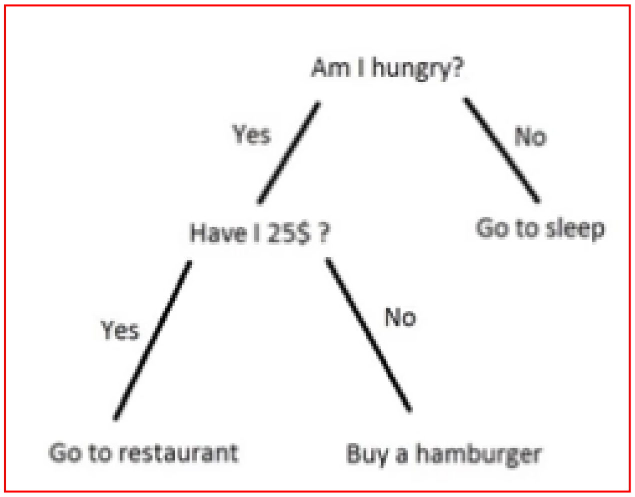
Problems wherein the data needs to be organized to find specific patterns like in the case of 'Product' Recommendation

- Classification is the process of dividing the datasets into different categories or groups by adding label. Ie., It adds the data point to a particular labelled group on the basis of some condition.
- Types of Classification
    - **Decision Tree**
    - Random Forest
    - Naïve Bayes
    - K Nearest Neighbour
    - Logistic Regression

A classification tree will determine a set of logical if then conditions to classify problems. For example, discriminating between three types of flowers based on certain features.

- A decision tree is a graphical representation of all possible solutions to a decision based on certain conditions.
- It is a tree shaped diagram used to determine a course of action. Each branch of the tree represents a possible decision, occurrence or relation.





Advantages of Decision Tree

- Simple to understand, interpret and visualize
- Little effort for data preparation and less requirement of data cleaning
- Can handle both numerical and categorical data
- Useful for solving decision related problems.
- Non linear parameters don’t effect its performance

Disadvantages

- Over fitting occurs when the algorithm captures noise in the data
- High variance- The model can get unstable due to small variation in data
- Low biased tree- A highly complication DT tends to have a low bias which makes it difficult for the model to work with new data

Motivation for tree based models

  - Handling of categorical variables

    - Handling of missing values and unknown levels
    - Detection of nonlinear relationships
    - Visualization and interpretation in decision trees.

3.1 Decision Tree- Terminology

- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

Building a Decision tree

There are several algorithms to build a decision tree.

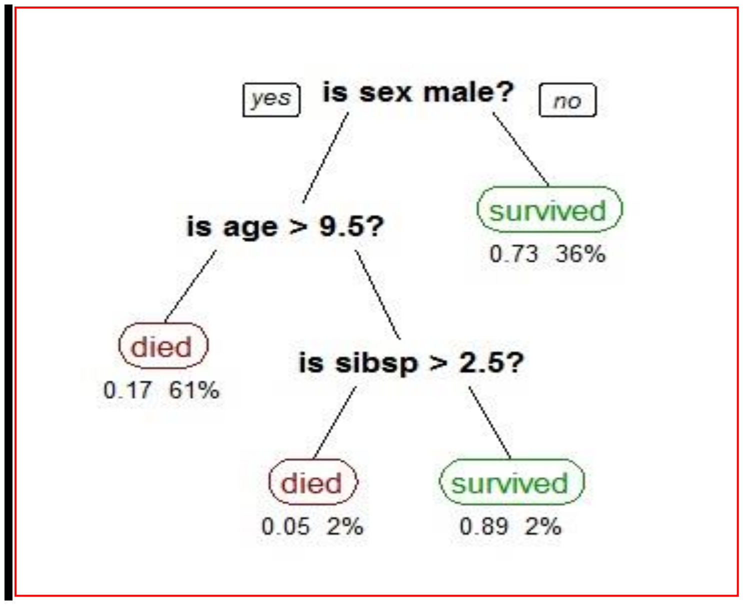
- CART-Classification And Regression Trees –Gini Index
- ID3-Iterative Dichotomiser 3
    - Entropy function
    - Information Gain
- C4.5
- CHAID-Chi-squared Automatic Interaction Detection

Only CART and ID3 algorithms as they are the ones majorly used.

A Decision tree is tree which each node represents a Feature(Attribute) , each link (branch) represents a Decision (Rule) and each leaf represents an outcome.

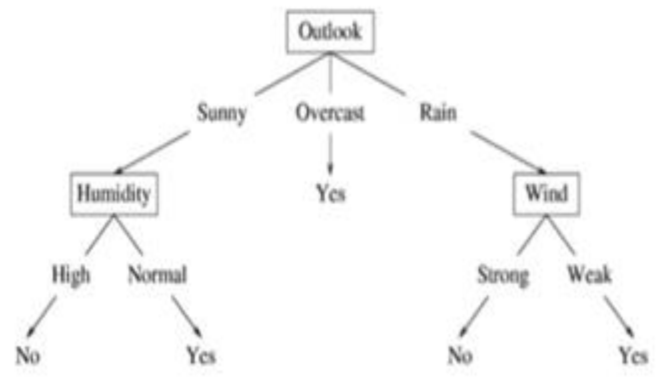
How can an algorithm be represented as a tree?

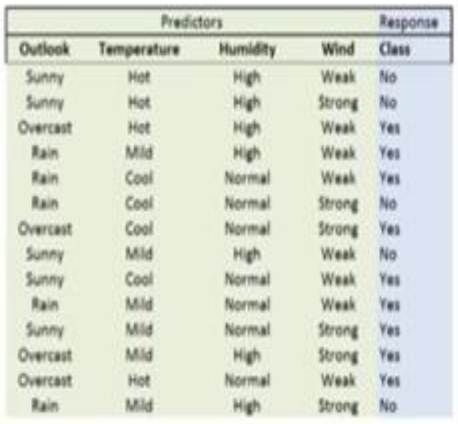
For this let’s consider a very basic example that uses titanic data set for predicting whether a passenger will survive or not. Below model uses 3 features/attributes/columns from the data set, namely sex, age and sibsp (number of spouses or children along).



A possible decision tree for the data:

- Each internal node: test on attribute X_i
- Each branch from a node: selects one value for X_i
- Each leaf node: Predict Y

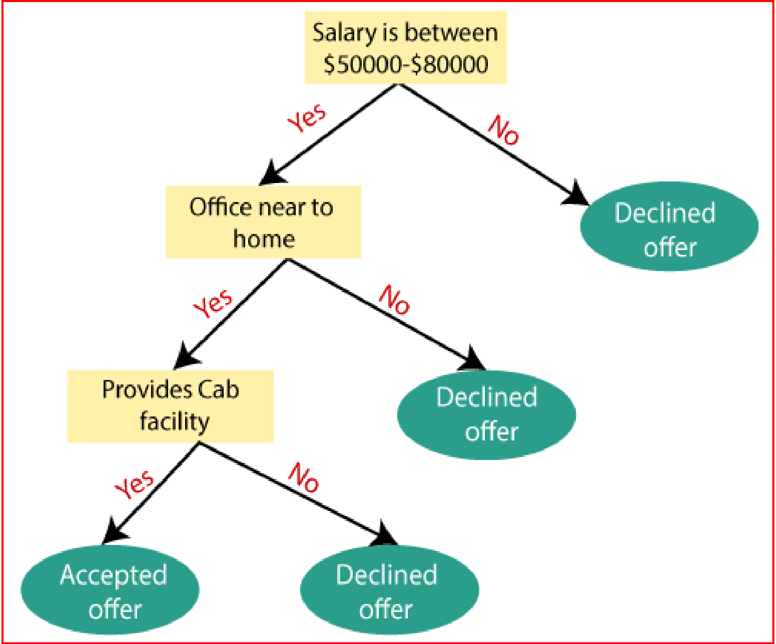




	Predictors			Response
Outlook	Temperature	Humidity	Wind	Class
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

How does the Decision Tree algorithm Work?

- In a decision tree, for predicting the class of the given dataset, the algorithm starts from the **root node of the tree.** This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.
- For the next node, the algorithm again compares the **attribute value** with the other **sub-nodes** and move further. It continues the process until it reaches the **leaf node** of the tree.



The complete process can be better understood using the below algorithm:

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.

- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Attribute Selection Measures

- While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.**
- By this measurement, it can easily select the best attribute for the nodes of the tree.

These are popular techniques for ASM:

- **Information Gain-ID3**
- **Gini Index-CART**
- **Entropy-ID3**
- Gain Ratio-C4.5
- Reduction in Variance-C4.5
- Chi-Square-CHAID

Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, split the node and build the decision tree.

ie., decide which attribute should be selected as the decision node

- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

Information Gain= Entropy(S)-[{Weighted avg)* Entropy(Each feature)]

For example

IG(T,X)=Entropy(T)-Entropy(T,X)

IG(PlayGolf,Outlook)=E(PlayGolf)-E(PlayGolf,Outlook)

= 0.940-0.693=0.247

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data.

Entropy can be calculated as:

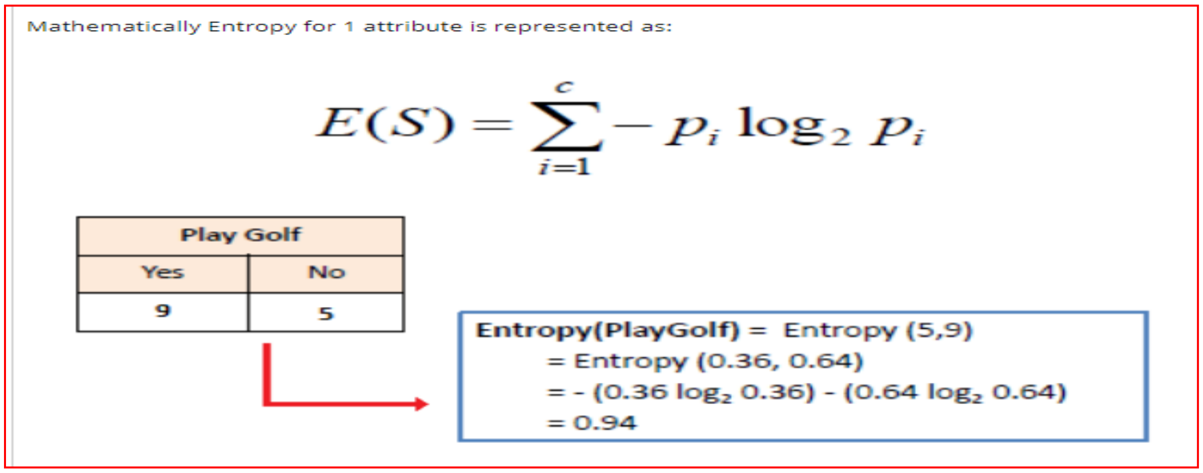
$$\text{Entropy}(s)= P(\text{yes})\log_2 P(\text{yes})- P(\text{no}) \log_2 P(\text{no})$$

Where,

S= Total number of samples

P(yes)= probability of yes

P(no)= probability of no

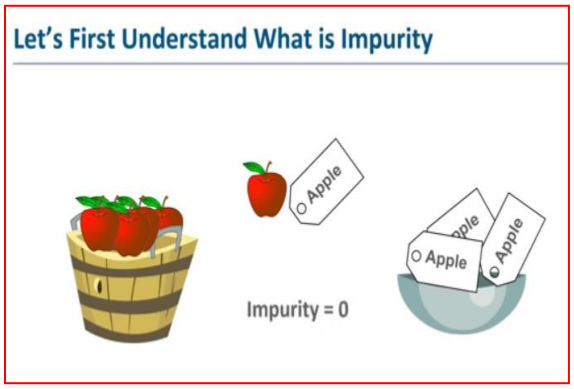


Mathematically Entropy for 1 attribute is represented as:

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

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)
= 0.94



Gini Index:

Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.

An attribute with the low Gini index should be preferred as compared to the high Gini index.

It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.

Gini index can be calculated using the below formula:

$$\text{Gini Index} = 1 - \sum_j P_j^2$$

Steps to Calculate Gini index for a split

- Calculate Gini for sub-nodes, using the above formula for success(p) and failure(q) (p^2+q^2).
- Calculate the Gini index for split using the weighted Gini score of each node of that split.

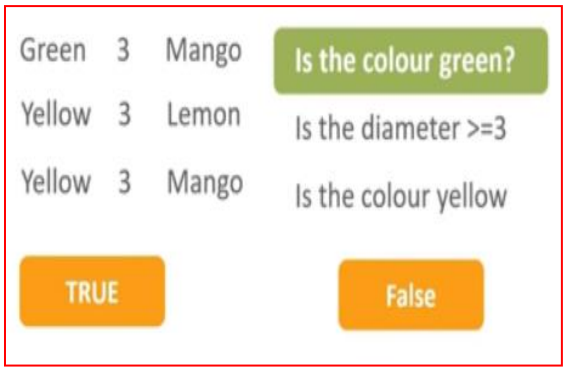
CART (Classification and Regression Tree) uses the Gini index method to create split points.

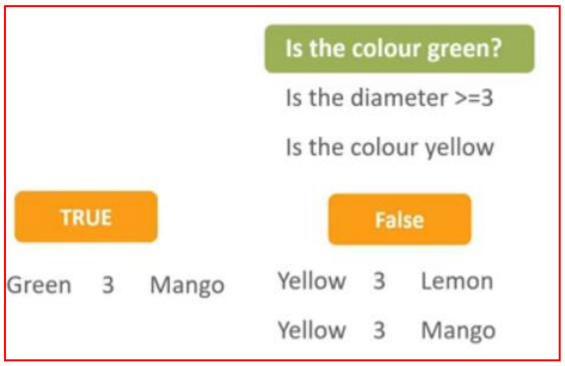
Pruning: Getting an Optimal Decision tree

- *Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.*
- A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning.

There are mainly two types of tree **pruning** technology used:

- **Cost Complexity Pruning**
- **Reduced Error Pruning.**

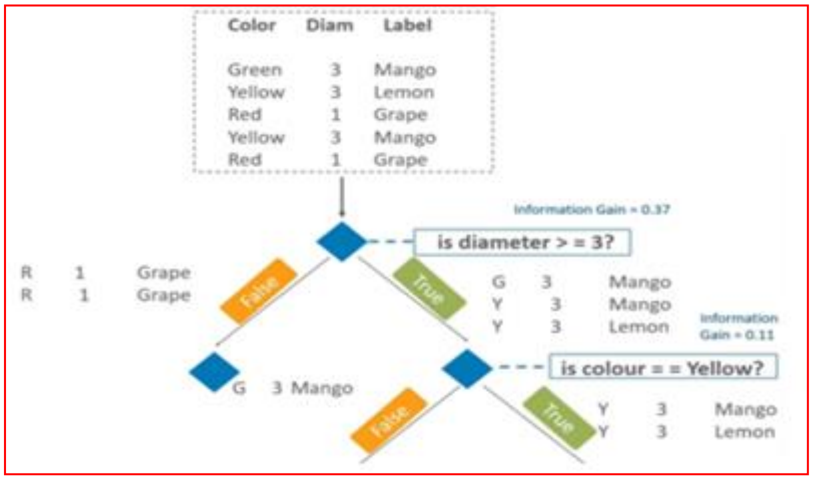




What is a CART in Machine Learning?

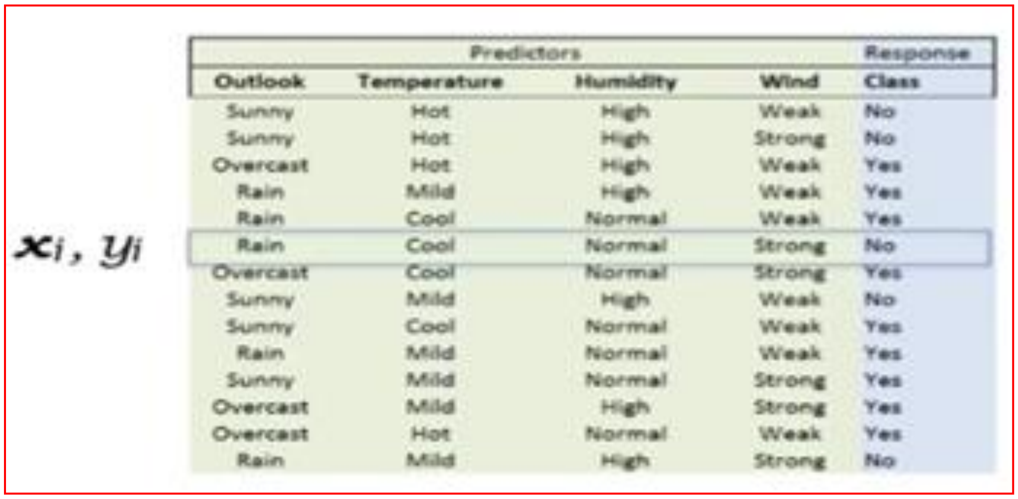
- A Classification and Regression Tree(CART) is a predictive algorithm used in machine learning. It explains how a target variable's values can be predicted based on other values.
- It is a decision tree where each fork is a split in a predictor variable and each node at the end has a prediction for the target variable.
- The CART algorithm is an important decision tree algorithm that lies at the foundation of machine learning.
- Moreover, it is also the basis for other powerful machine learning algorithms like bagged decision trees, random forest and boosted decision trees.

The Classification and regression tree(CART) methodology is one of the oldest and most fundamental algorithms. It is used to predict outcomes based on certain predictor variables.



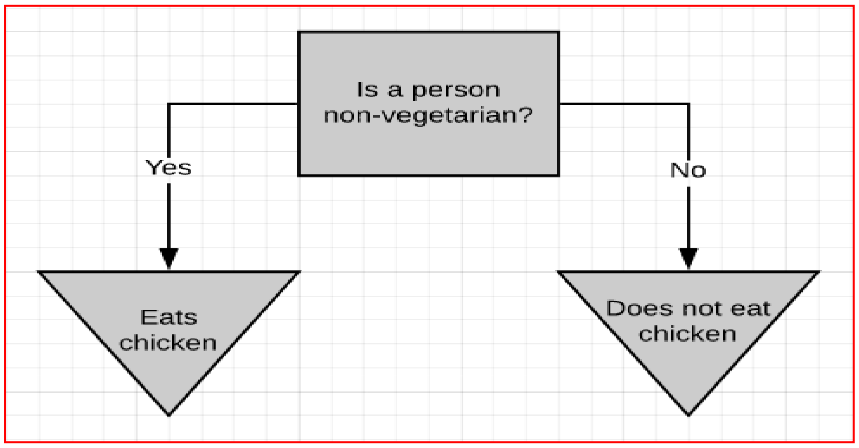
Sample Dataset(was Tennis Played?)

- Columns denote features Xi
- Rows denote labeled instances xi,yi
- Class label denotes whether a tennis game was played

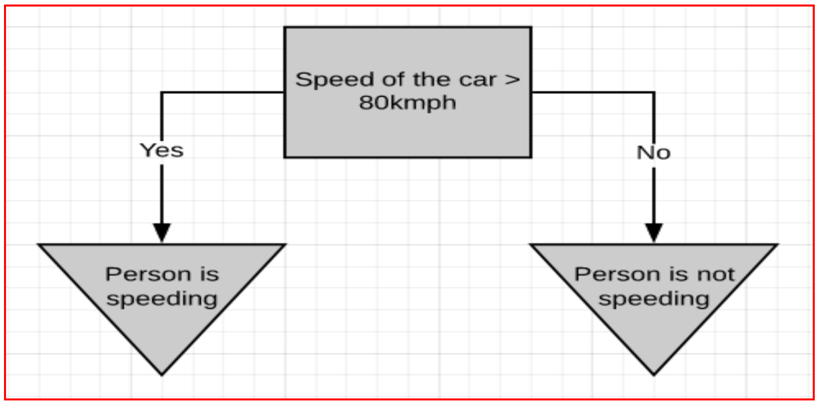


Predictors				Response
Outlook	Temperature	Humidity	Wind	Class
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

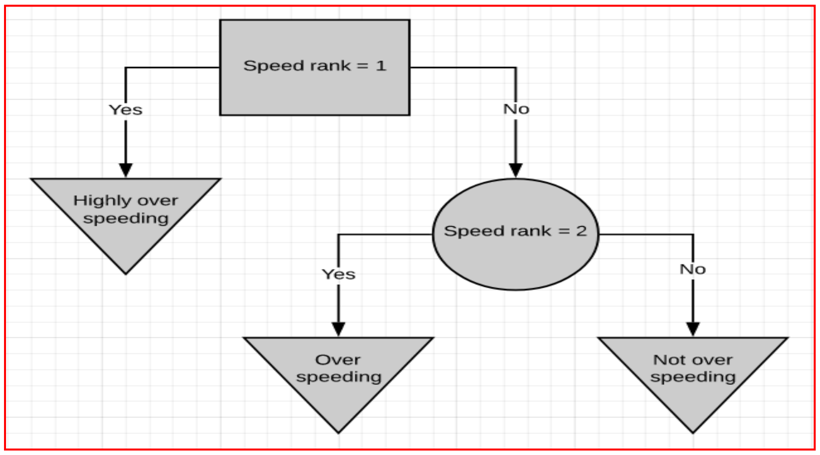
- **Decision Trees (DTs)** are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in machine learning.



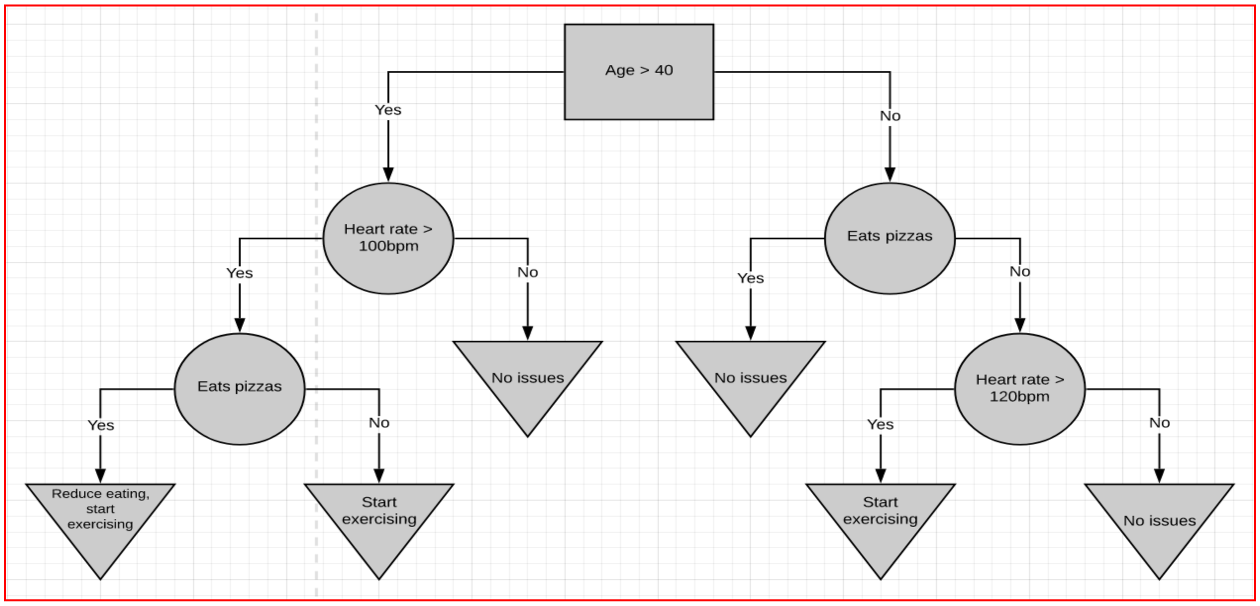
Example 2: Decision tree is based on numeric data. If a person is driving above 80kmph, we can consider it as over-speeding, else not.



Example 3: If a person is driving above 80kmph, we can consider it as over-speeding, else not.Here is one more simple decision tree. This decision tree is based on ranked data, where 1 means the speed is too high, 2 corresponds to a much lesser speed. If a person is speeding above rank 1 then he/she is highly over-speeding. If the person is above speed rank 2 but below speed rank 1 then he/she is over-speeding but not that much. If the person is below speed rank 2 then he/she is driving well within speed limits.



The classification in a decision tree can be both categorical or numeric which is used in Bioinformatics



- The tree is called the ***root node*** . The nodes in between are called ***internal nodes***. Internal nodes have arrows pointing to them and arrows pointing away from them. The end nodes are called the ***leaf nodes*** or just ***leaves***. Leaf nodes have arrows pointing to them but no arrows pointing away from them.

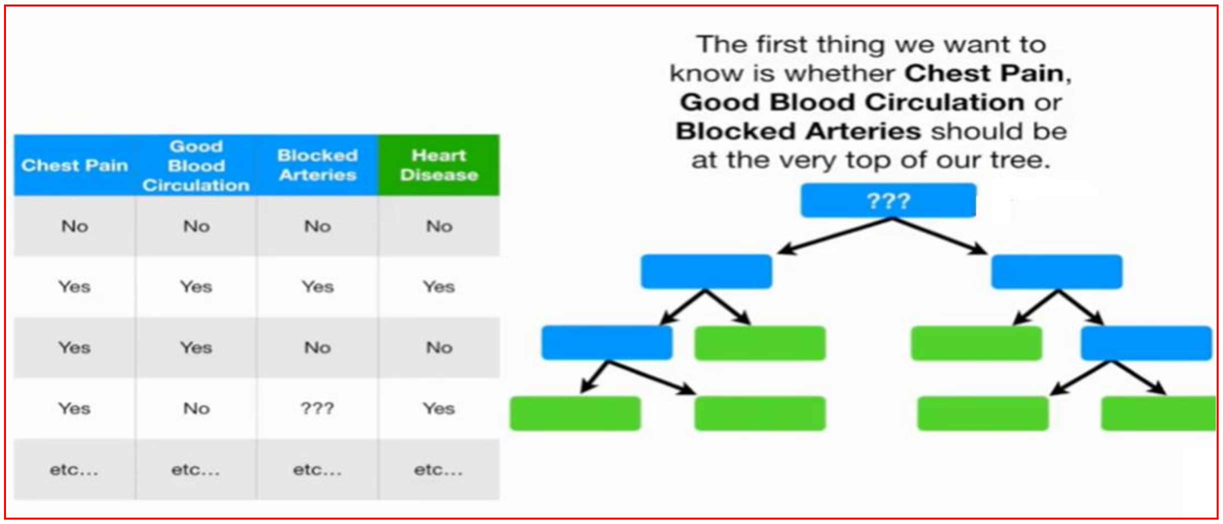
- In the above diagrams, root nodes are represented by rectangles, internal nodes by circles, and leaf nodes by inverted-triangles.

Classification and Regression Trees

CART is a DT algorithm that produces binary Classification or Regression Trees, depending on whether the dependent (or target) variable is categorical or numeric, respectively. It handles data in its raw form (no preprocessing needed) and can use the same variables more than once in different parts of the same DT, which may uncover complex interdependencies between sets of variables.

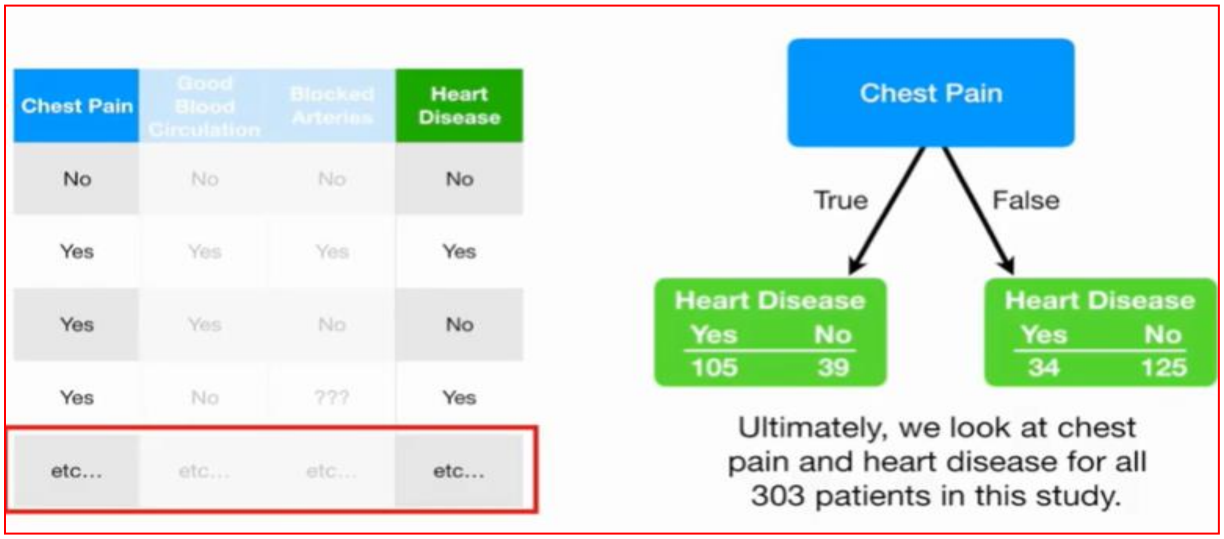
- In the example given, build a decision tree that uses chest pain, good blood circulation, and the status of blocked arteries to predict if a person has heart disease or not.

Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
NO	NO	NO	NO
YES	YES	YES	YES
YES	YES	NO	NO
YES	NO	YES	YES
etc.	etc.	etc.	etc.



Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc...	etc...	etc...	etc...

The first thing we want to know is whether **Chest Pain**, **Good Blood Circulation** or **Blocked Arteries** should be at the very top of our tree.

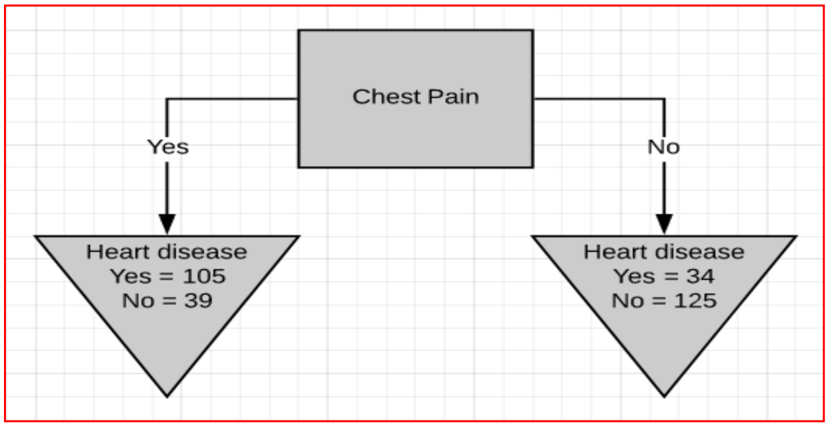


Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc...	etc...	etc...	etc...

Ultimately, we look at chest pain and heart disease for all 303 patients in this study.

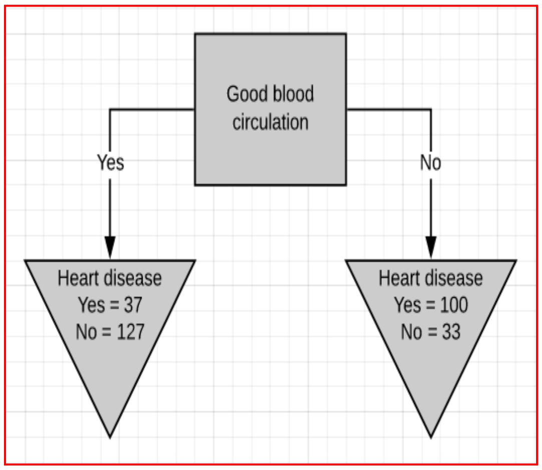
There are two leaf nodes, one each for the two outcomes of chest pain. Each of the leaves contains the no. of patients having heart disease and not having heart disease for the corresponding entry of chest pain

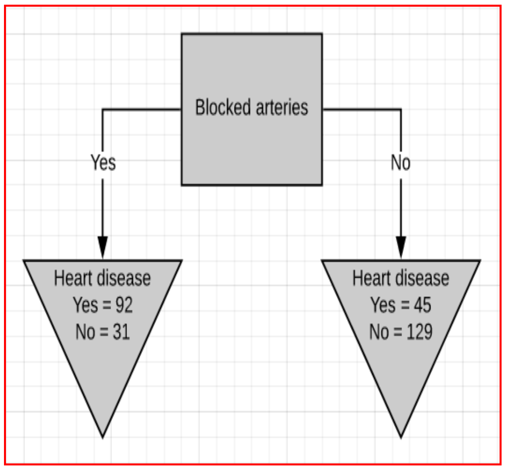
Chest pain as the root node



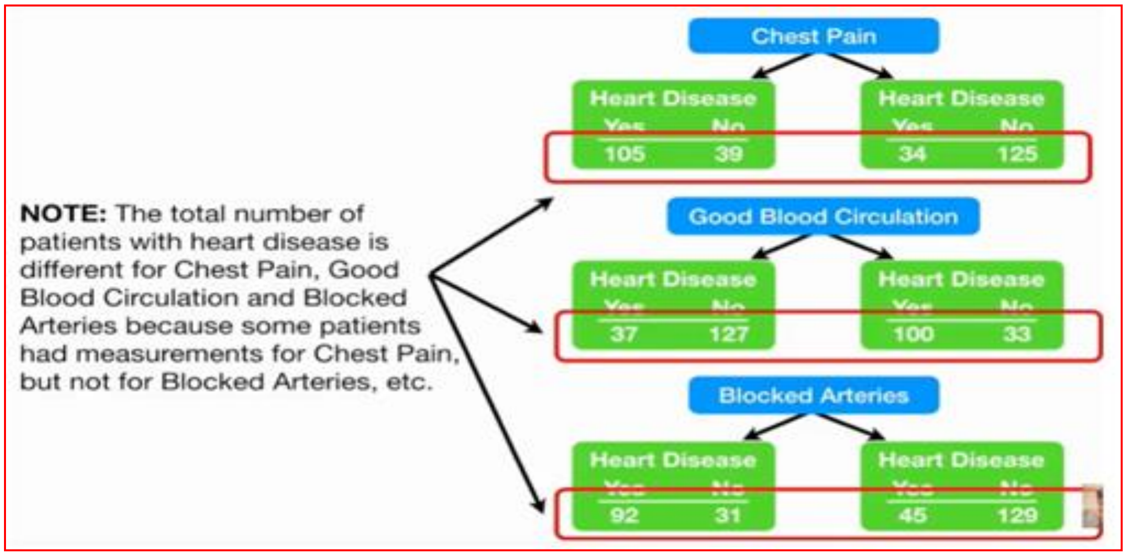
There are two leaf nodes, one each for the two outcomes of chest pain. Each of the leaves contains the no. of patients having heart disease and not having heart disease for the corresponding entry of chest pain.

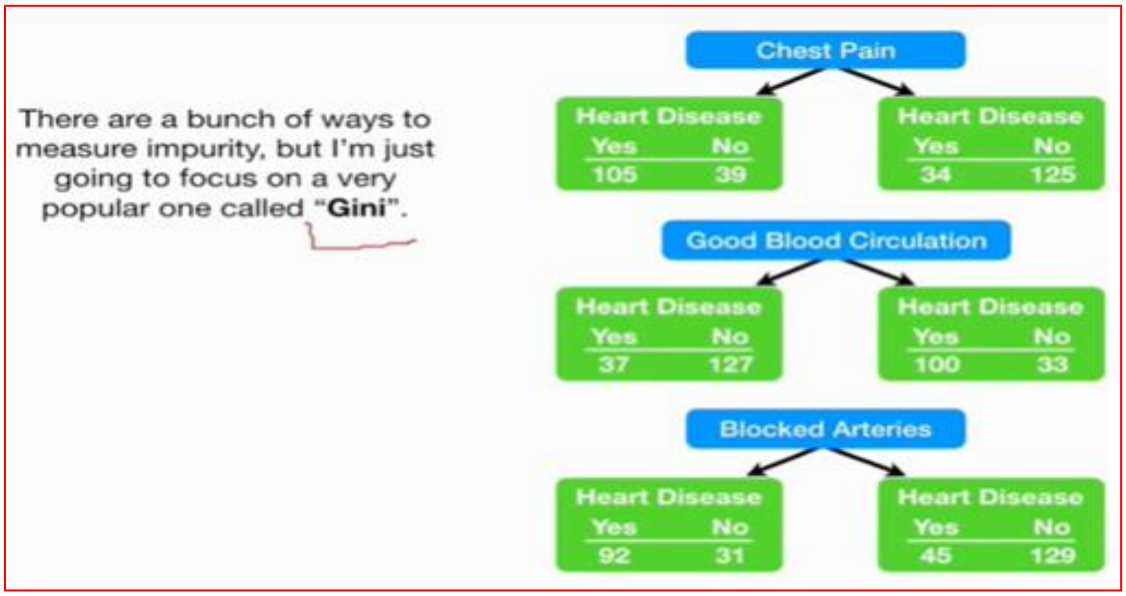
- **The same thing for good blood circulation and blocked arteries.**





Good blood circulation as the root node **Blocked arteries as the root node**


NOTE: The total number of patients with heart disease is different for Chest Pain, Good Blood Circulation and Blocked Arteries because some patients had measurements for Chest Pain, but not for Blocked Arteries, etc.


There are a bunch of ways to measure impurity, but I'm just going to focus on a very popular one called **"Gini"**.

- The 3 features separates the patients having heart disease from the patients not having heart disease perfectly. It is to be noted that the total no. of patients having heart disease is different in all three cases. This is done to simulate the missing values present in real-world datasets.

- Because none of the leaf nodes is either 100% 'yes heart disease' or 100% 'no heart disease', they are all considered *impure*. To decide on which separation is the best, we need a method to measure and compare *impurity*.

- The metric used in the CART algorithm to measure impurity is the *Gini impurity score*.

  Gini impurity = 1 - (probability of 'yes')^2 - (probability of 'no')^2

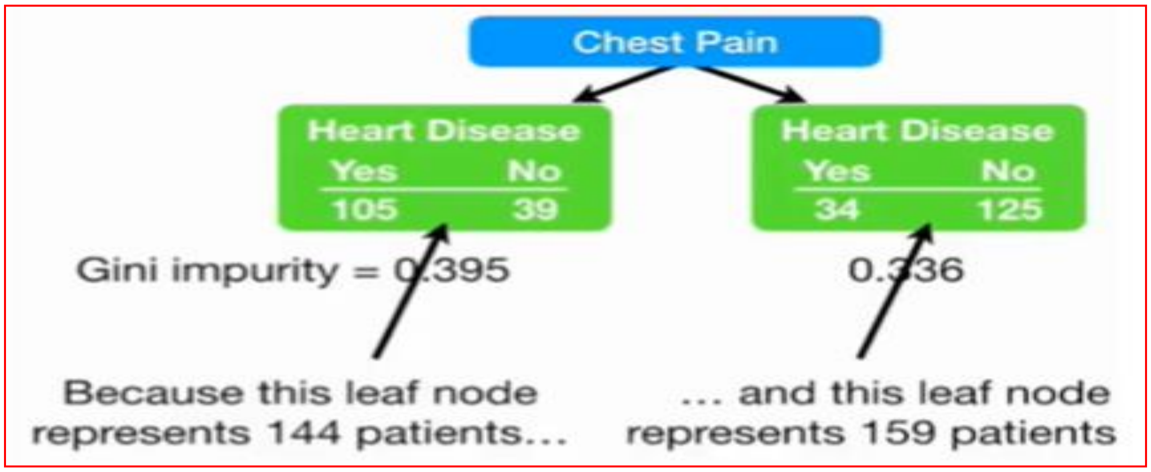
Chest Pain

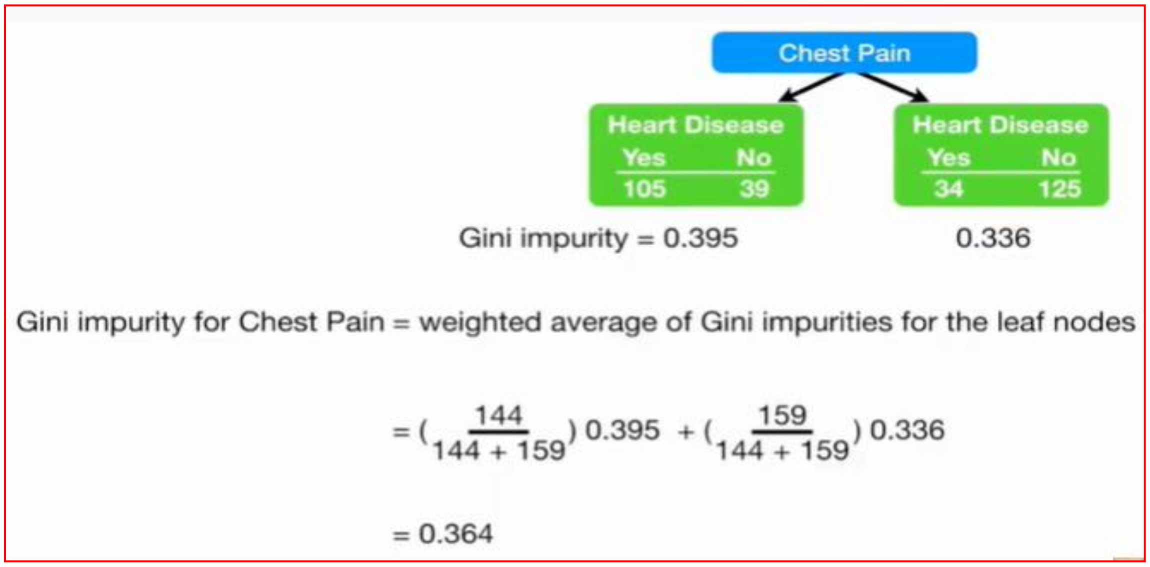
Calculating the Gini impurity for chest pain for the left leaf,

- Gini impurity = 1 - (probability of ‘yes’)^2 - (probability of ‘no’)^2
  = 1 - (105/105+39)^2 - (39/105+39)^2
  Gini impurity = 0.395

Similarly, calculate the Gini impurity for the right leaf node.

- Gini impurity = 1 - (probability of ‘yes’)^2 - (probability of ‘no’)^2
  = 1 - (34/34+125)^2 - (125/34+125)^2
  Gini impurity = 0.336

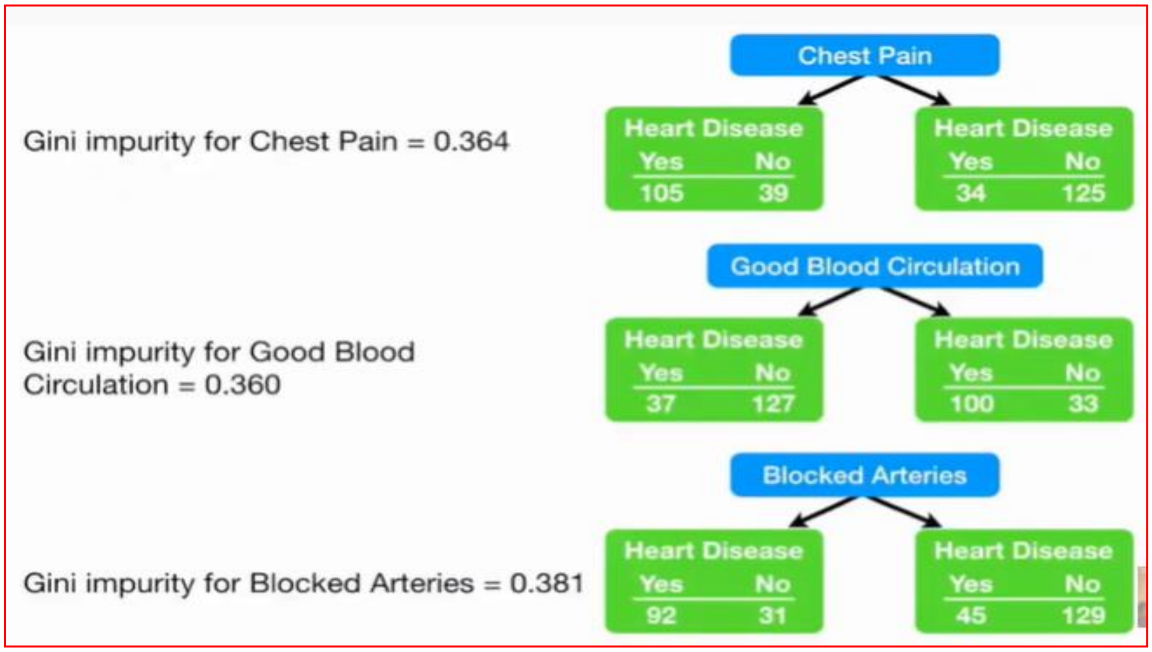

Chest Pain
Heart Disease
Yes No
105 39
Heart Disease
Yes No
34 125
Gini impurity = 0.395
0.336
Because this leaf node represents 144 patients...
... and this leaf node represents 159 patients



Gini impurity for Chest Pain = weighted average of Gini impurities for the leaf nodes

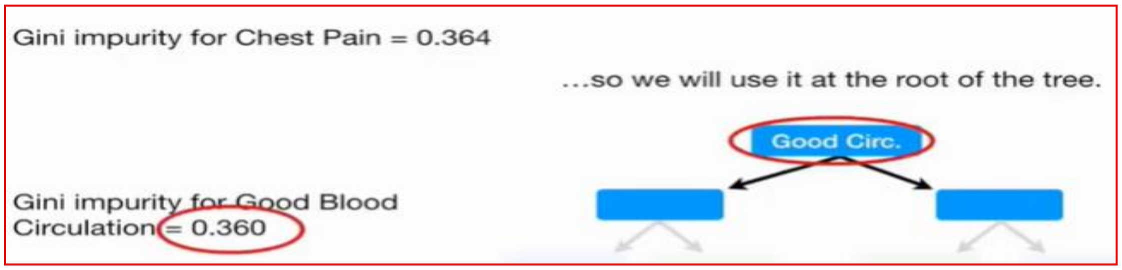
$$= \left(\frac{144}{144 + 159}\right) 0.395 + \left(\frac{159}{144 + 159}\right) 0.336$$

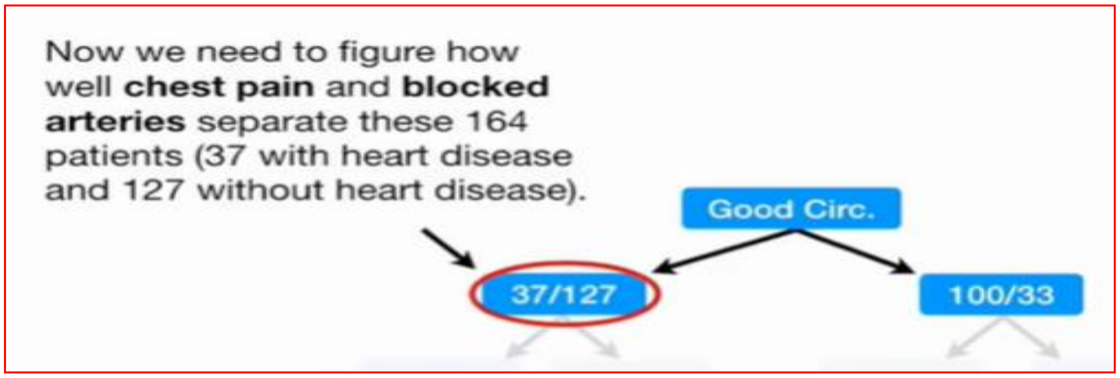
$$= 0.364$$


Gini impurity for Chest Pain = 0.364
Chest Pain
Heart Disease Yes No 105 39
Heart Disease Yes No 34 125
Gini impurity for Good Blood Circulation = 0.360
Good Blood Circulation
Heart Disease Yes No 37 127
Heart Disease Yes No 100 33
Gini impurity for Blocked Arteries = 0.381
Blocked Arteries
Heart Disease Yes No 92 31
Heart Disease Yes No 45 129

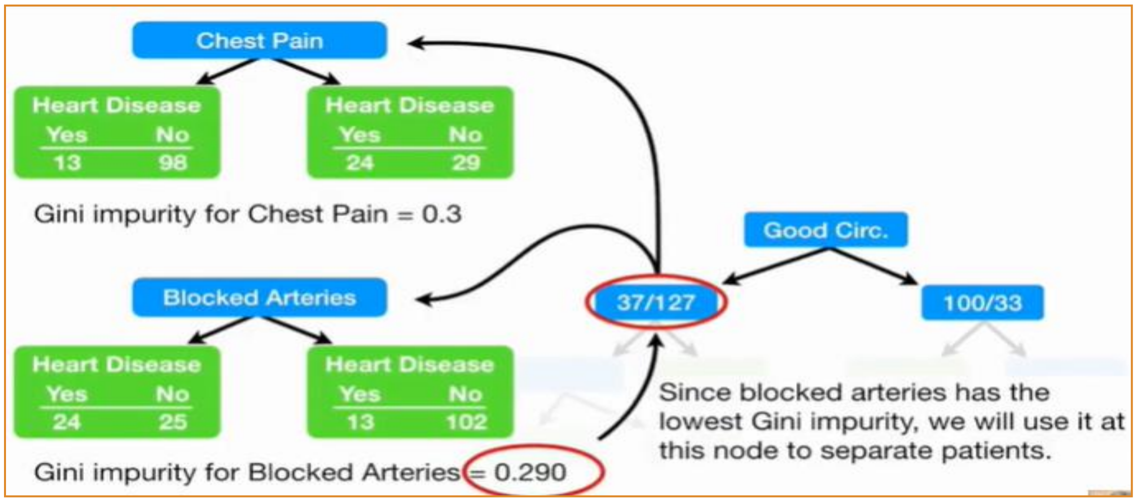
- Calculate the total Gini impurity for using chest pain to separate patients with and without heart disease.
- The leaf nodes do not represent the same no. of patients as the left leaf represents 144 patients and the right leaf represents 159 patients. Thus the total Gini impurity will be the weighted average of the leaf node Gini impurities.
- Gini impurity = (144/144+159)*0.395 + (159/144+159)*0.336 = 0.364
- Similarly the total Gini impurity for 'good blood circulation' and 'blocked arteries' is calculated as
- Gini impurity for 'good blood circulation' = 0.360 Gini impurity for 'blocked arteries' = 0.381

'Good blood circulation' has the lowest impurity score among the tree which symbolizes that it best separates the patients having and not having heart disease, so we will use it at the root node.


Gini impurity for Chest Pain = 0.364
...so we will use it at the root of the tree.
Good Circ.
Gini impurity for Good Blood Circulation = 0.360



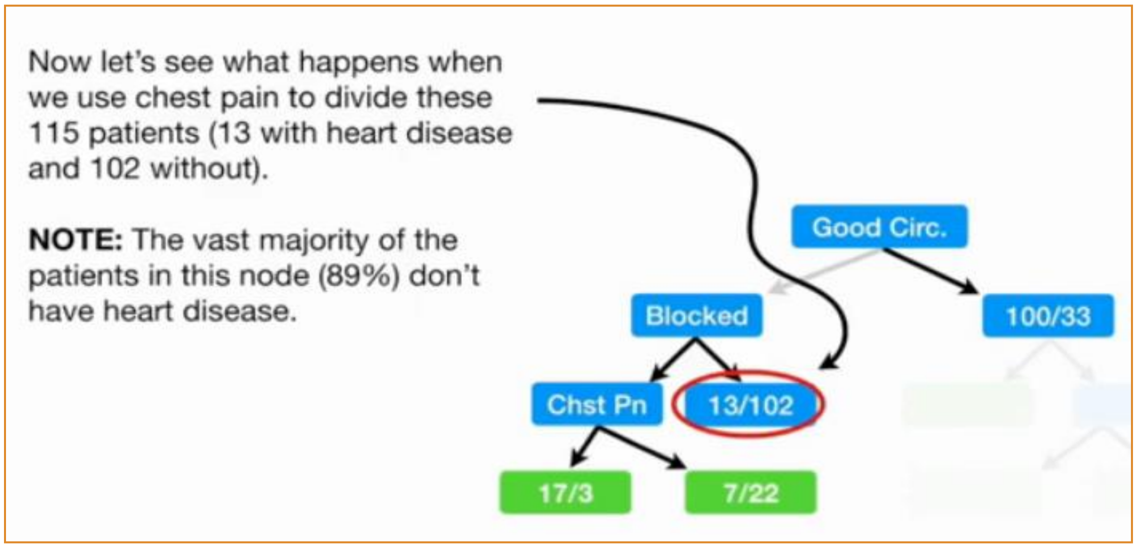
Now we need to figure how well **chest pain** and **blocked arteries** separate these 164 patients (37 with heart disease and 127 without heart disease).



Gini impurity for Chest Pain = 0.3

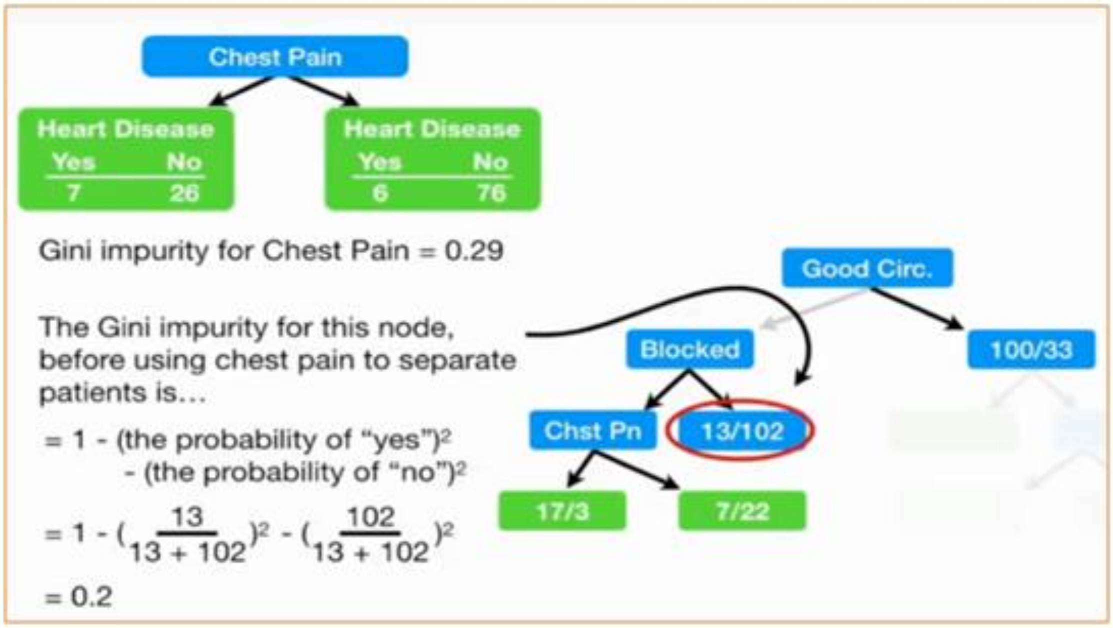
Gini impurity for Blocked Arteries = 0.290

Since blocked arteries has the lowest Gini impurity, we will use it at this node to separate patients.



Now let's see what happens when we use chest pain to divide these 115 patients (13 with heart disease and 102 without).

NOTE: The vast majority of the patients in this node (89%) don't have heart disease.



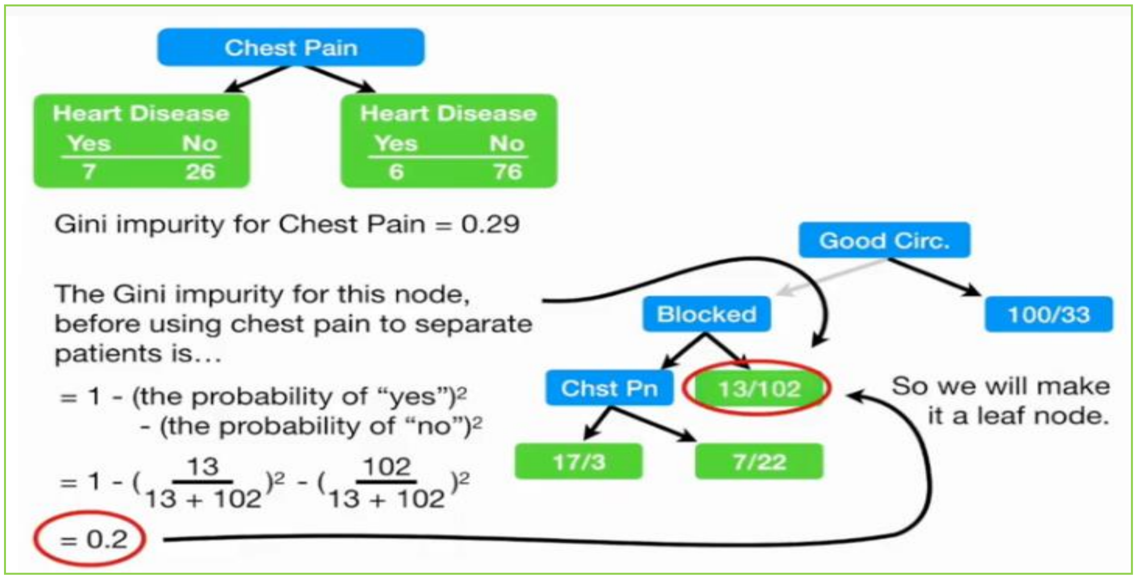
Gini impurity for Chest Pain = 0.29

The Gini impurity for this node, before using chest pain to separate patients is…

$$= 1 - (\text{the probability of "yes"})^2 - (\text{the probability of "no"})^2$$

$$= 1 - \left(\frac{13}{13 + 102}\right)^2 - \left(\frac{102}{13 + 102}\right)^2$$

$$= 0.2$$



Gini impurity for Chest Pain = 0.29

The Gini impurity for this node, before using chest pain to separate patients is…

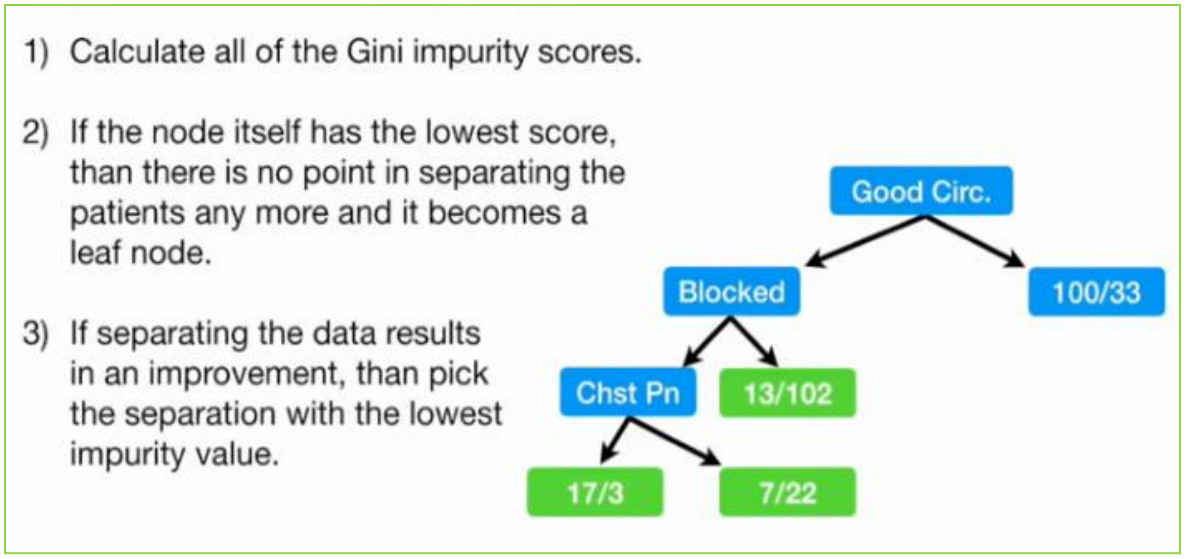
$$= 1 - (\text{the probability of "yes"})^2 - (\text{the probability of "no"})^2$$

$$= 1 - \left(\frac{13}{13 + 102}\right)^2 - \left(\frac{102}{13 + 102}\right)^2$$

$$= 0.2$$

So we will make it a leaf node.

Steps to be repeated on the left side

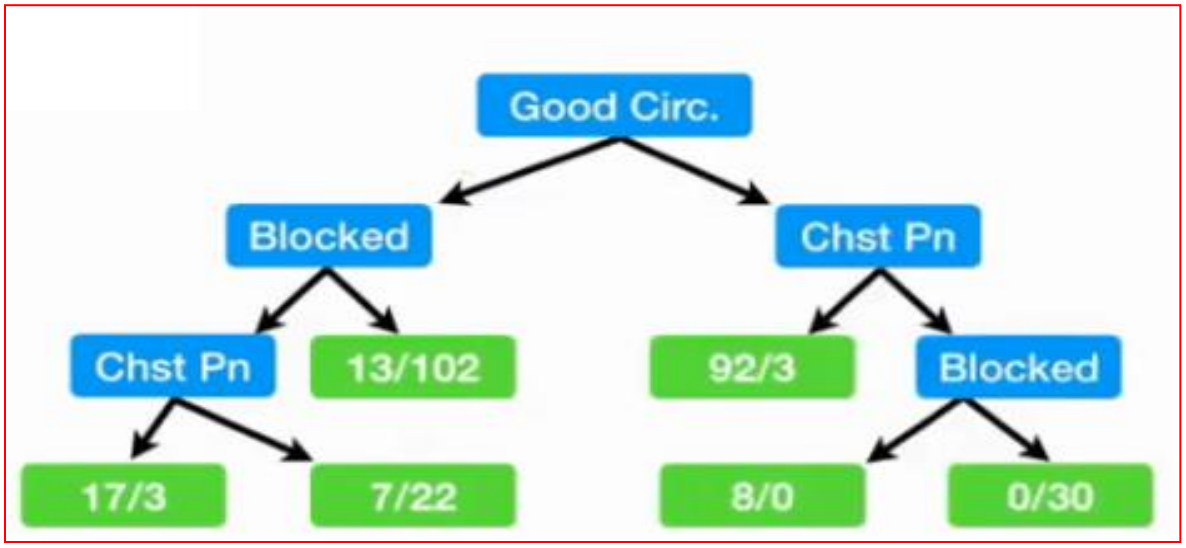


1) Calculate all of the Gini impurity scores.

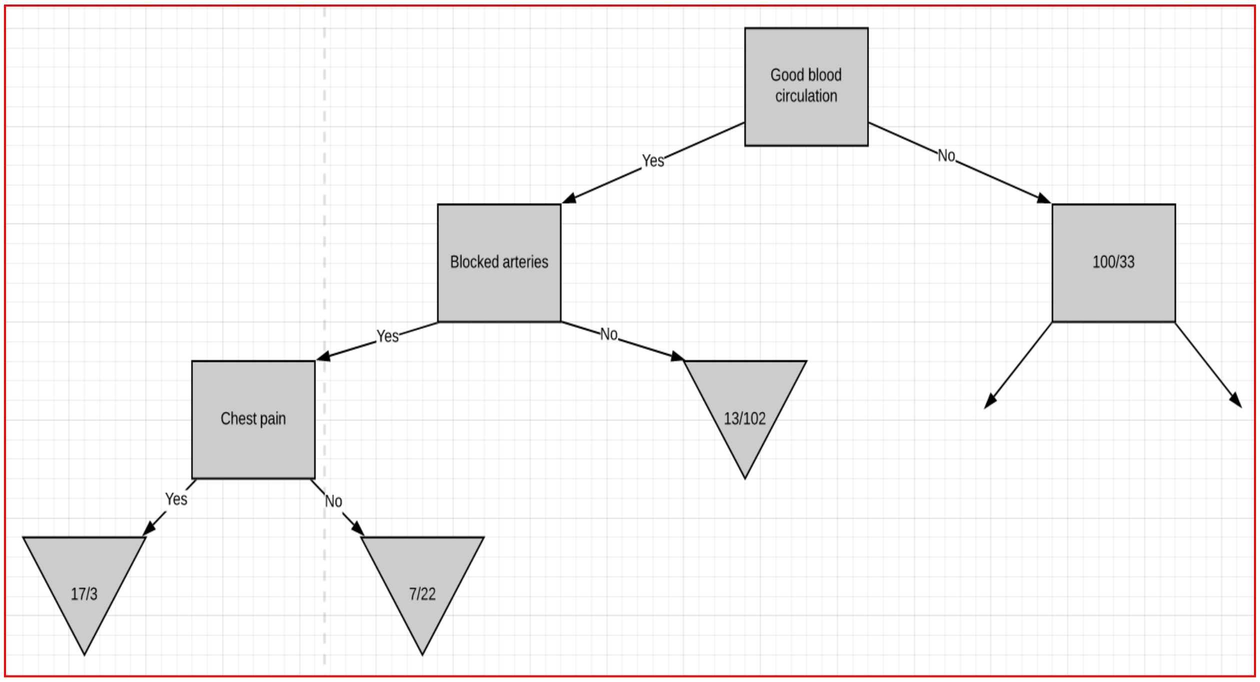
2) If the node itself has the lowest score, than there is no point in separating the patients any more and it becomes a leaf node.

3) If separating the data results in an improvement, than pick the separation with the lowest impurity value.

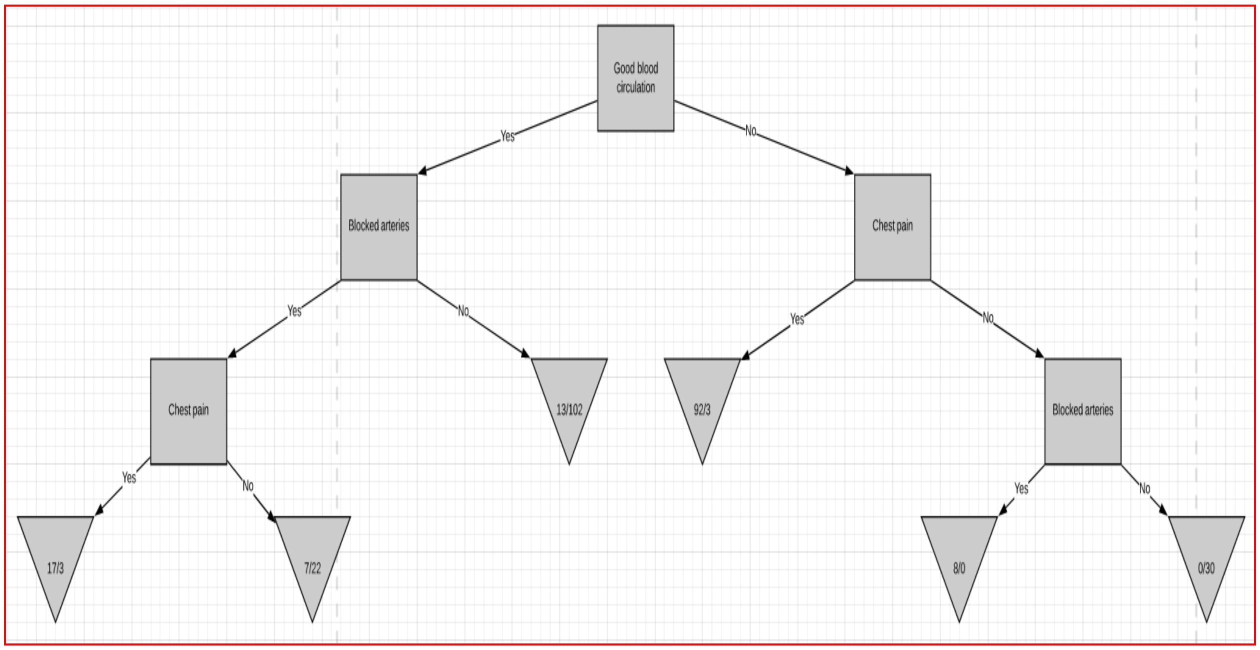
Good Circ.
Blocked
100/33
Chst Pn
13/102
17/3
7/22



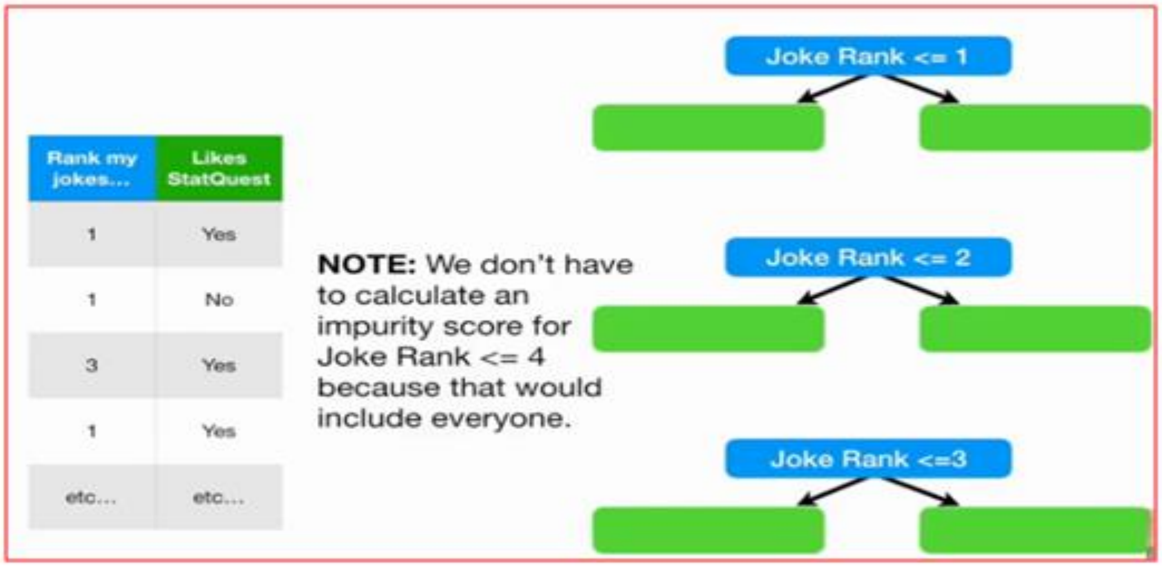
- Calculate the Gini impurity scores.
- If the node itself has the lowest score, then there is no point in separating the patients anymore and it becomes a leaf node.
- If separating the data results in improvement then pick the separation with the lowest impurity value.



Complete Decision tree

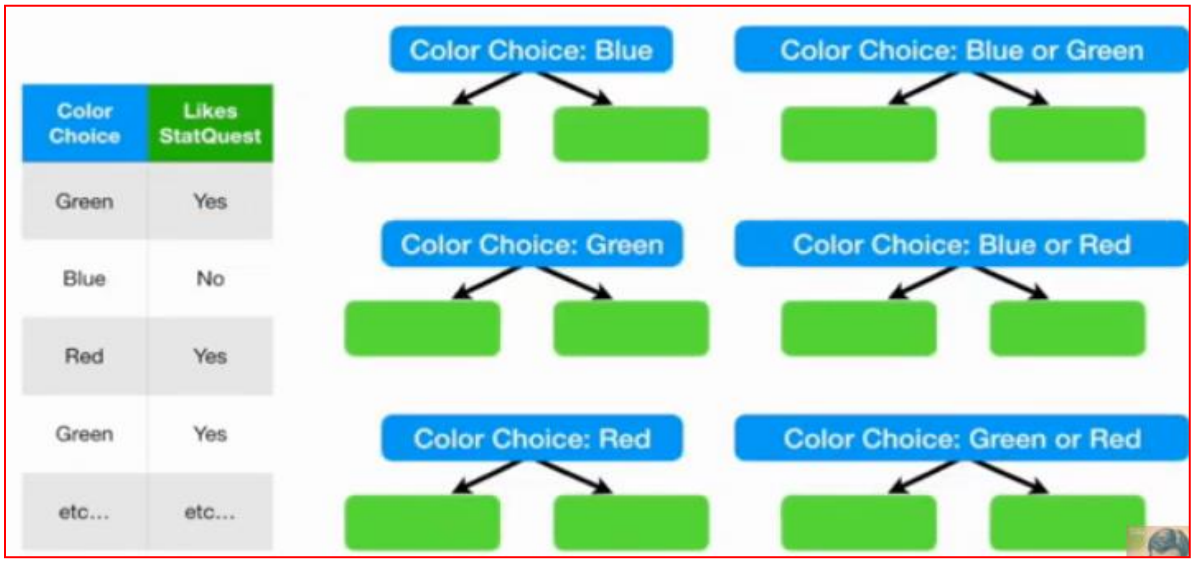


Some examples



Rank my jokes...	Likes StatQuest
1	Yes
1	No
3	Yes
1	Yes
etc...	etc...

NOTE: We don't have to calculate an impurity score for Joke Rank <= 4 because that would include everyone.


Color Choice: Blue
Color Choice: Blue or Green
Color Choice: Green
Color Choice: Blue or Red
Color Choice: Red
Color Choice: Green or Red

Color Choice	Likes StatQuest
Green	Yes
Blue	No
Red	Yes
Green	Yes
etc...	etc...

Classification And Regression Trees (CART) for Machine Learning

Classification Trees	**Regression Tress**
It is used when dependent variables is categorical	It is used when dependent variable is continuous

Use Mode/Class(Mode- happens most often)	Use Mean/ Average
A classification tree is an algorithm where the target variable is fixed or categorical.	A regression tree refers to an algorithm where the target variable is and the algorithm is used to predict it's value.
Classification trees are used when the dataset needs to be split into classes which belong to the response variable. In many cases, the classes Yes or No.	Regression trees, on the other hand, are used when the response variable is continuous. For instance, if the response variable is something like the price of a property or the temperature of the day, a regression tree is used.
classification trees are used for classification-type problems.	Regression trees are used for prediction-type problems
A classification tree splits the dataset based on the homogeneity of data.	In a regression tree, a regression model is fit to the target variable using each of the independent variables.
Measures of impurity like entropy or Gini index are used to quantify the homogeneity of the data when it comes to classification trees.	The error between the predicted values and actual values is squared to get "A Sum of Squared Errors"(SSE). The SSE is compared across the variables and the variable or point which has the lowest SSE is chosen as the split point. This process is continued recursively.

3.2 Ensemble Learning –Boosting-Bagging -Different ways to Combine Classifiers

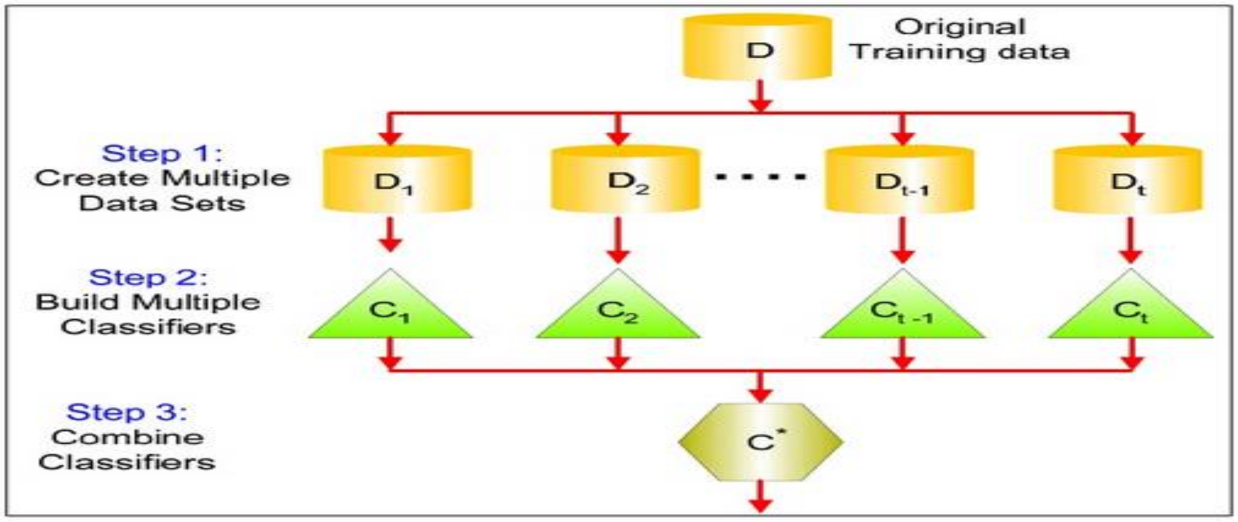
- **First level of Machine Learning**
  - Linear Regression
  - Logistics Regression
  - Support Vector Machine
  - Naive Bayes
  - Decision Tree
- **Ensemble Learning(2nd level)**
  - Weak Learner to Strong Learners
  - Group/ collection of Models/Predictors

- Wisdom of Crowd
- Voting
- Combining different models

What is Classification

“Classification is the process of grouping things according to similar features they share”

- Ensemble Learning is a technique combines individual models together to improve the stability and predictive power of the model.
- This technique permits higher predictive performance
- It combines multiple machine learning models into one predictive model.


D
Original Training data
Step 1: Create Multiple Data Sets
D_1
D_2
D_{t-1}
D_t
Step 2: Build Multiple Classifiers
C_1
C_2
C_{t-1}
C_t
Step 3: Combine Classifiers
C^*

- Certain models do well in modelling one aspect of the data, while others do well in modelling another.
- Learn several simple models and combine their output to produce the final decision.
- The combined strength of the models offsets individual model variances and biases.
- This provides a composite prediction where the final accuracy is better than the accuracy of individual models.

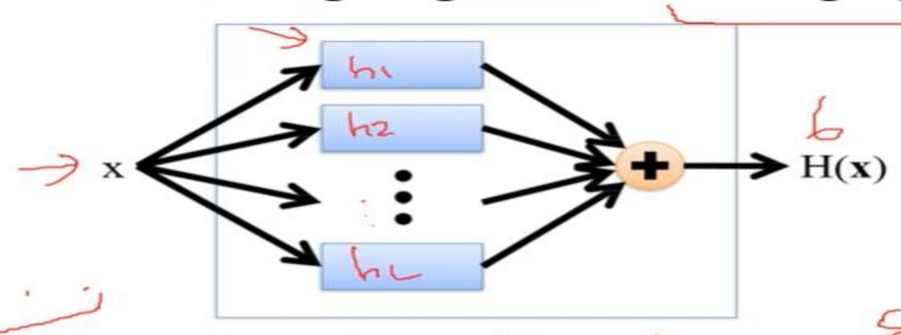
Significance

- This model is the application of multiple models to obtain better performance than from a single model
- Robustness- Ensemble models incorporate the predictions from all the base learners
- Accuracy-Ensemble models deliver accurate predictions and have improved performances
- Consider a set of classifiers h_1 , …., h_L

Construct a classifier H(x) that combines the individual decisions of h_1 , …., h_L

- Successful ensembles requires diversity
  - Classifiers should make different mistakes
  - Can have different types of base learners

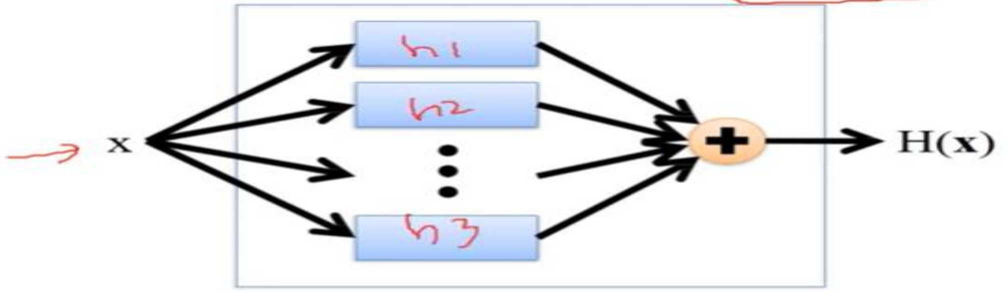
Combining Regressors: Averaging



h1	h2	h3	h4
LR	PR	SVR	DT
5	6	6	7

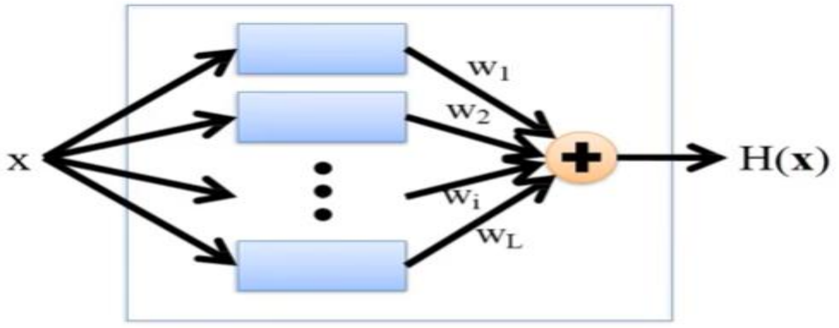
$$\frac{5+6+6+7}{4} = 6$$

Combining Classifiers: Voting

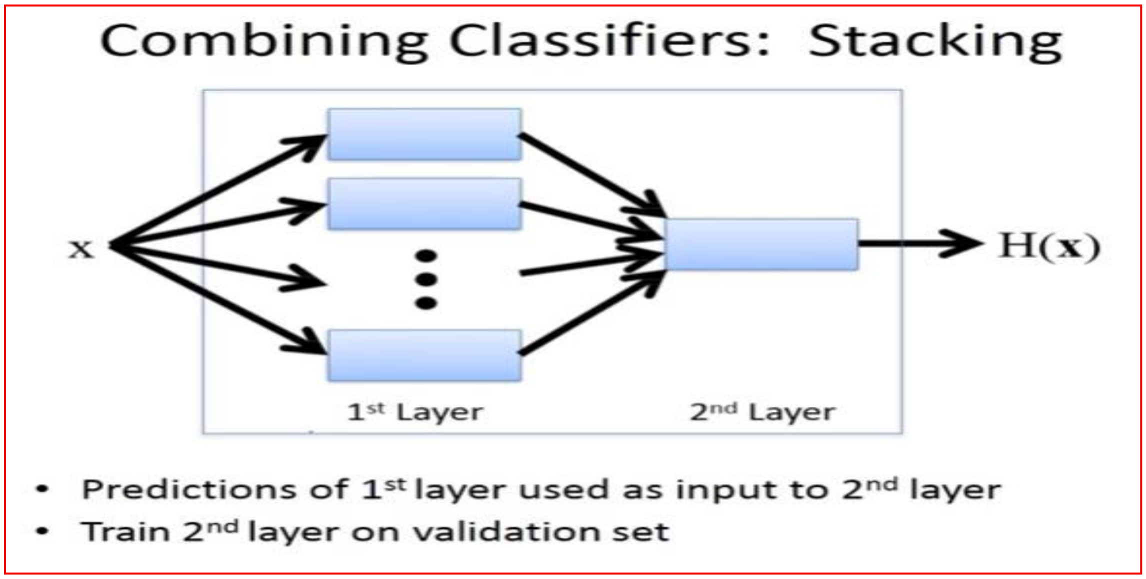


- Final hypothesis is a simple vote of the members

Combining Classifiers: Weighted Average



- Coefficients of individual members are trained using a validation set

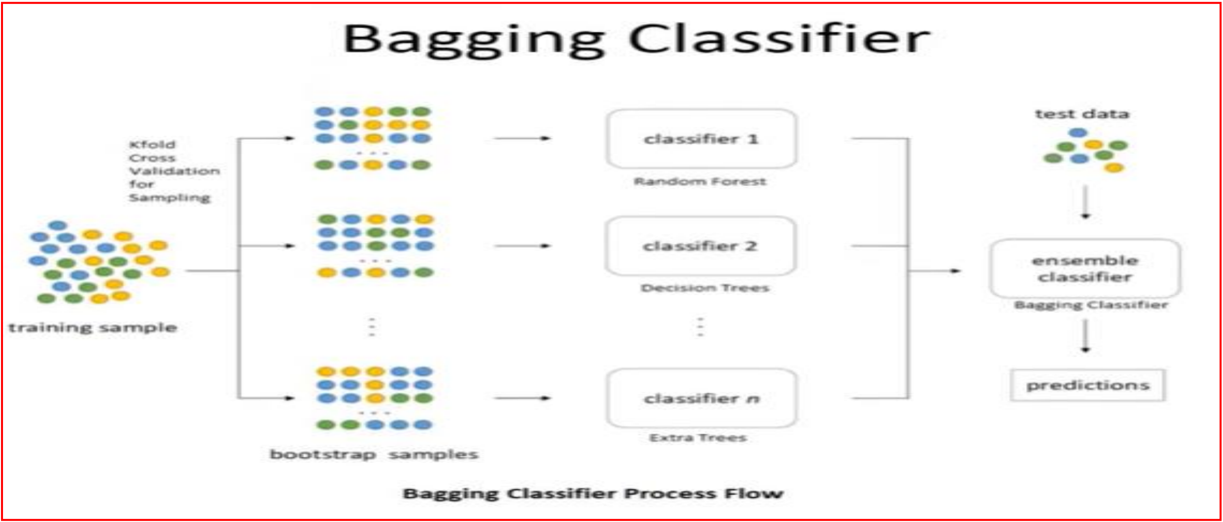


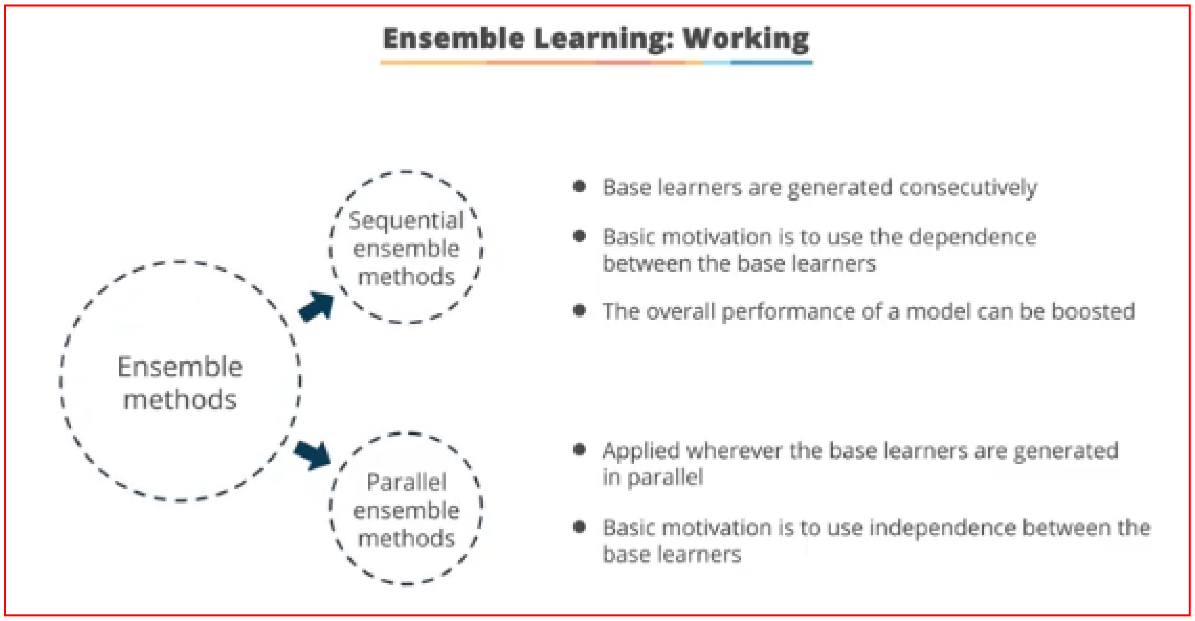
Combining Classifiers: Stacking

- Predictions of 1st layer used as input to 2nd layer
- Train 2nd layer on validation set

Manipulating the training data

- **Bootstrap replication:**
  - Given n training examples, construct a new training set by sampling n instances with replacement
  - Excludes 30% of the training instances
- **Bagging:**
  - Create bootstrap replicates of training set
  - Train a classifier(e.g., a decision tree) for each replicate
  - Estimate classifier performance using out-of-bootstrap data
  - Average output of all classifiers




Ensemble Learning: Working
Ensemble methods
Sequential ensemble methods
Parallel ensemble methods
Base learners are generated consecutively
Basic motivation is to use the dependence between the base learners
The overall performance of a model can be boosted
Applied wherever the base learners are generated in parallel
Basic motivation is to use independence between the base learners

Labelled images of Hot DOGS and other objects

- Images of hot dogs labelled 1
- Images of other objects labelled 0

Why use Ensemble models?

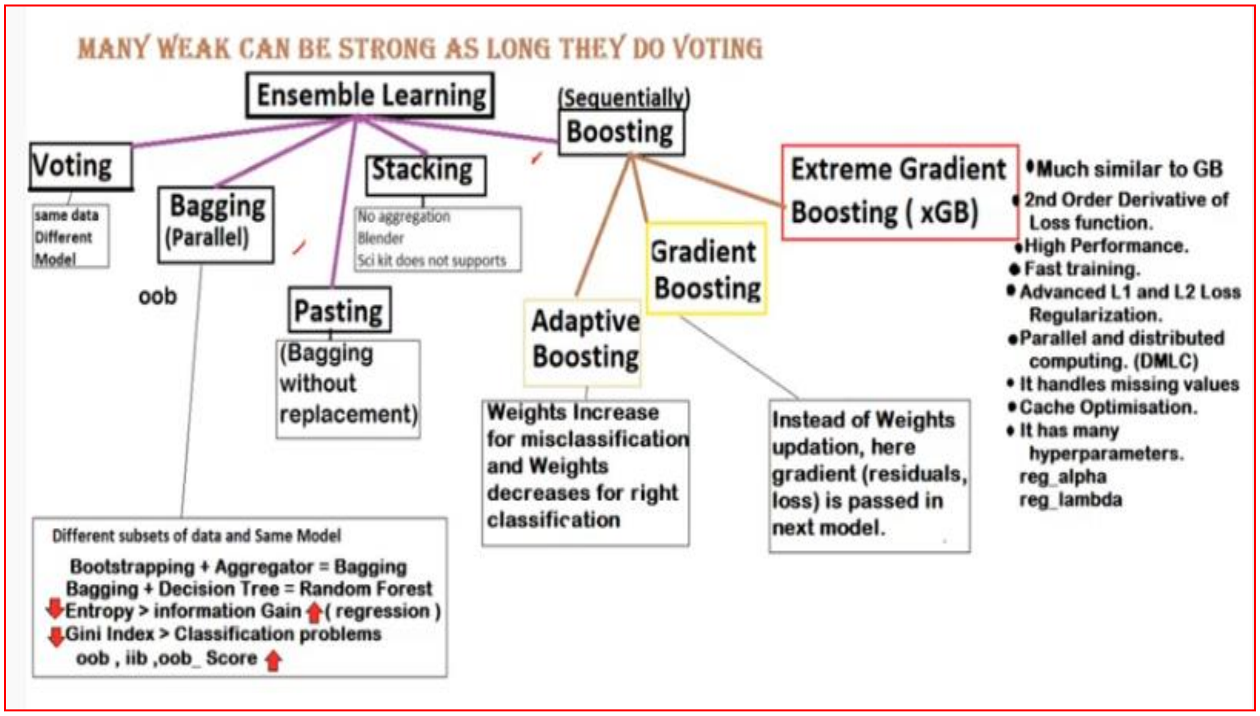
- Better Accuracy(Low Error)
- Higher Consistency(Avoids Over fitting)
- Reduces bias and Variance errors

When and Where to use Ensemble models

- Single model overfits
- Results worth the extra trining.
- Can be used for Classification as well as Regression

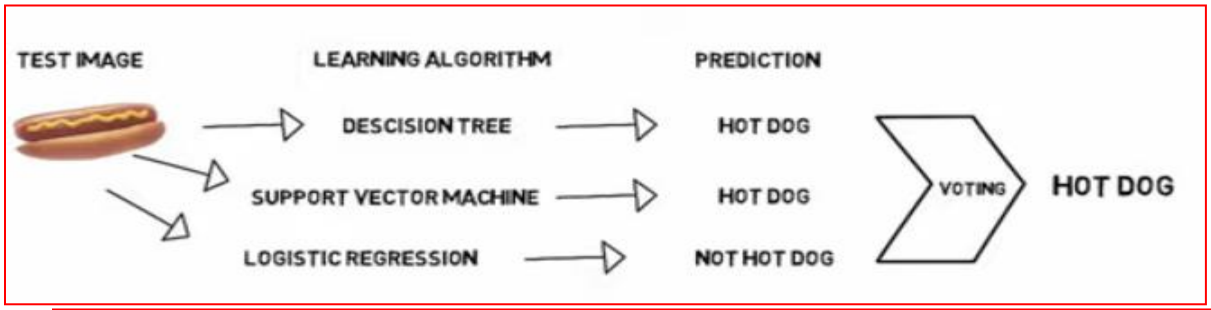
Ensemble Learning Methods

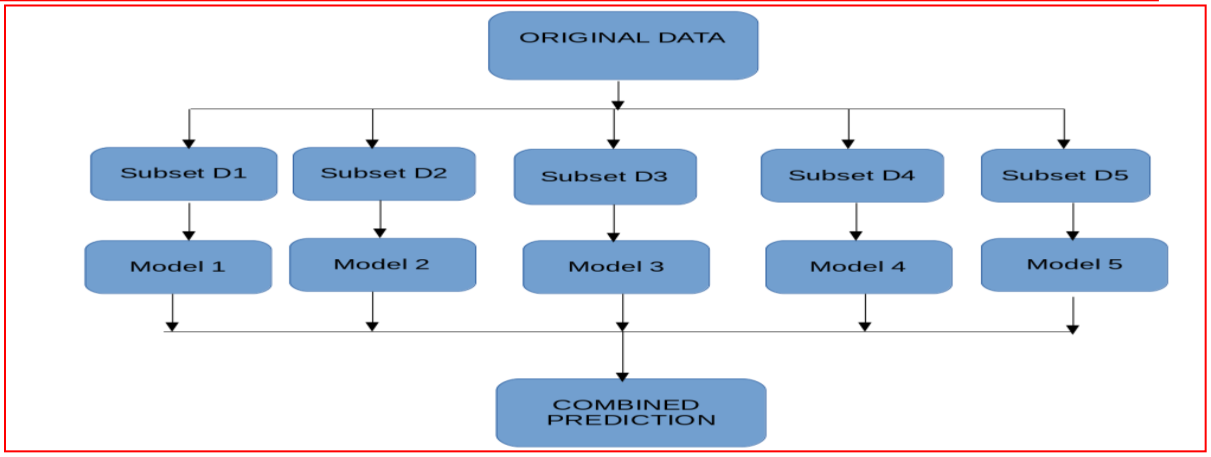
- Combine all weak learners to form an ensemble (or) Create an ensemble of well chosen strong and diverse models
- This models gain more accuracy and robustness by combining data from numerous modelling approaches


MANY WEAK CAN BE STRONG AS LONG THEY DO VOTING
Ensemble Learning
(Sequentially)
Boosting
Voting
same data Different Model
Bagging (Parallel)
Stacking
No aggregation Blender Sci kit does not supports
oob
Pasting
(Bagging without replacement)
Extreme Gradient Boosting (xGB)
Gradient Boosting
Adaptive Boosting
Weights Increase for misclassification and Weights decreases for right classification
Instead of Weights updation, here gradient (residuals, loss) is passed in next model.
•Much similar to GB
•2nd Order Derivative of Loss function.
•High Performance.
•Fast training.
•Advanced L1 and L2 Loss Regularization.
•Parallel and distributed computing. (DMLC)
•It handles missing values
•Cache Optimisation.
•It has many hyperparameters. reg_alpha reg_lambda
Different subsets of data and Same Model
Bootstrapping + Aggregator = Bagging
Bagging + Decision Tree = Random Forest
Entropy > information Gain (regression)
Gini Index > Classification problems
oob , iib ,oob_ Score

Bagging

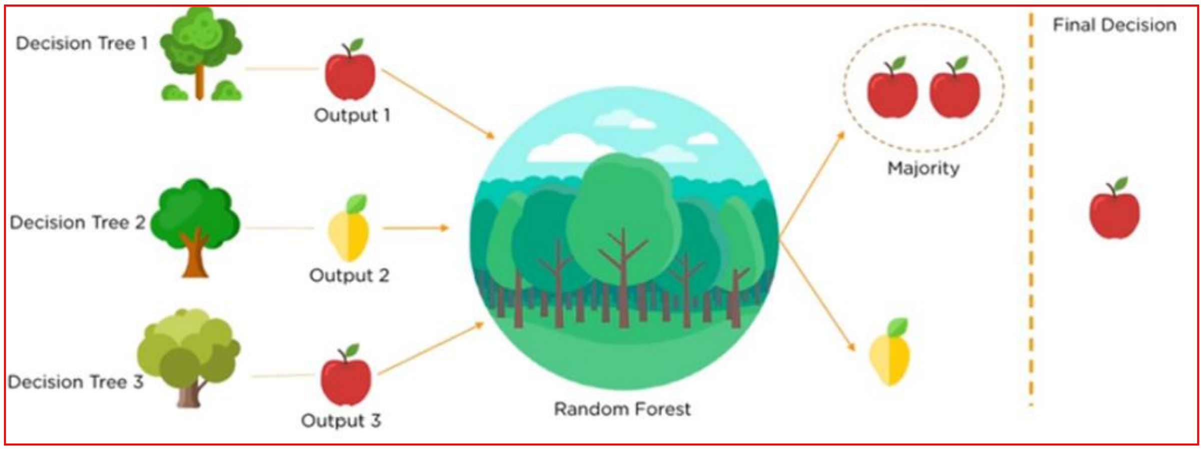
- The idea behind bagging is combining the results of multiple models (for instance, all decision trees) to get a generalized result.
- If create all the models on the same set of data and combine it, Whether these models will give the same result ,since they are getting the same input. One of the techniques is **bootstrapping.**
- Bootstrapping is a sampling technique in which create subsets of observations from the original dataset, **with replacement**. The size of the subsets is the same as the size of the original set.
- Bagging (or Bootstrap Aggregating) technique uses these subsets (bags) to get a fair idea of the distribution (complete set). The size of subsets created for bagging may be less than the original set.
- Multiple subsets are created from the original dataset, selecting observations with replacement.
- A base model (weak model) is created on each of these subsets.
- The models run in parallel and are independent of each other.
- The final predictions are determined by combining the predictions from all the models.





What is Random forest?

Random forest or Random decision forest is a method that operates by constructing multiple decision trees during training phase. The decision of the majority of the tree is chosen by the random forest as the final decision.



Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

- The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

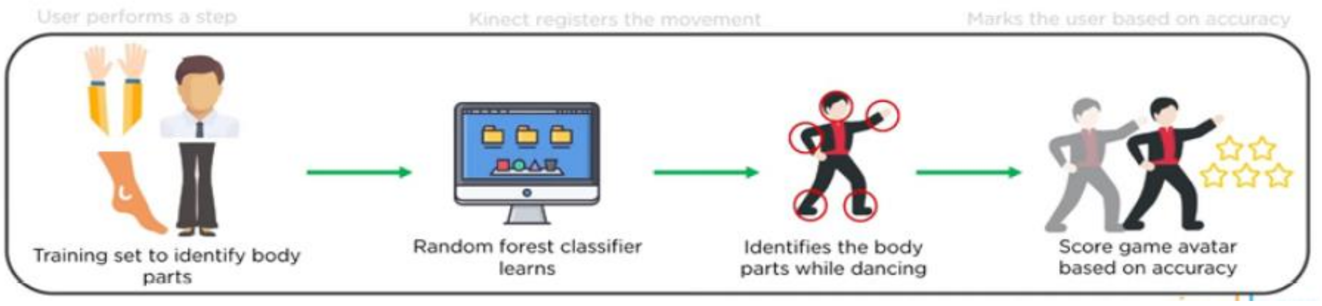
Note: To better understand the Random Forest Algorithm, you should have knowledge of the Decision Tree Algorithm.

Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

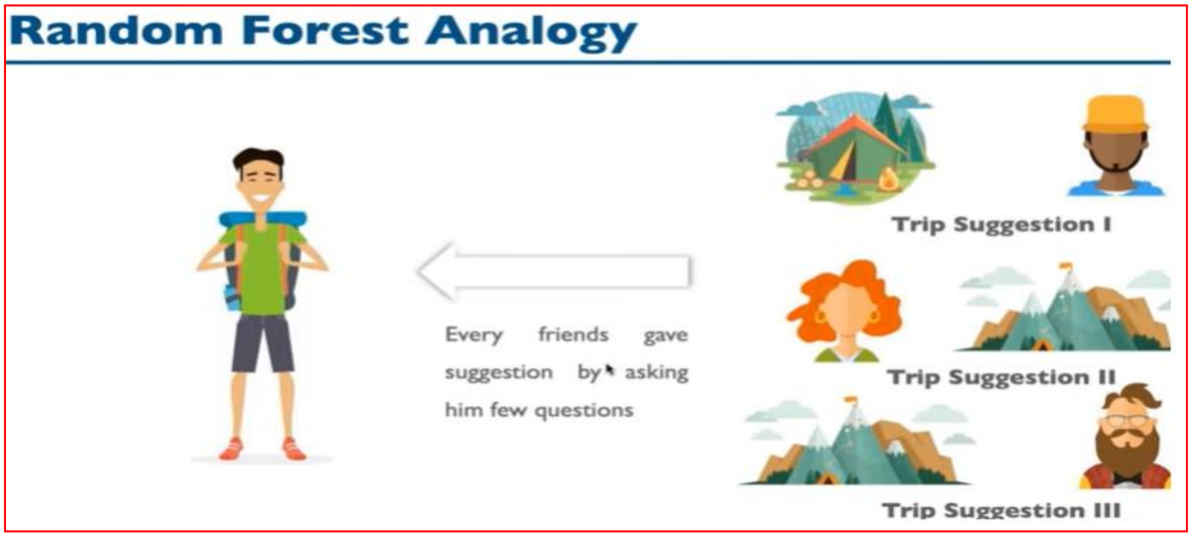
- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations**.**

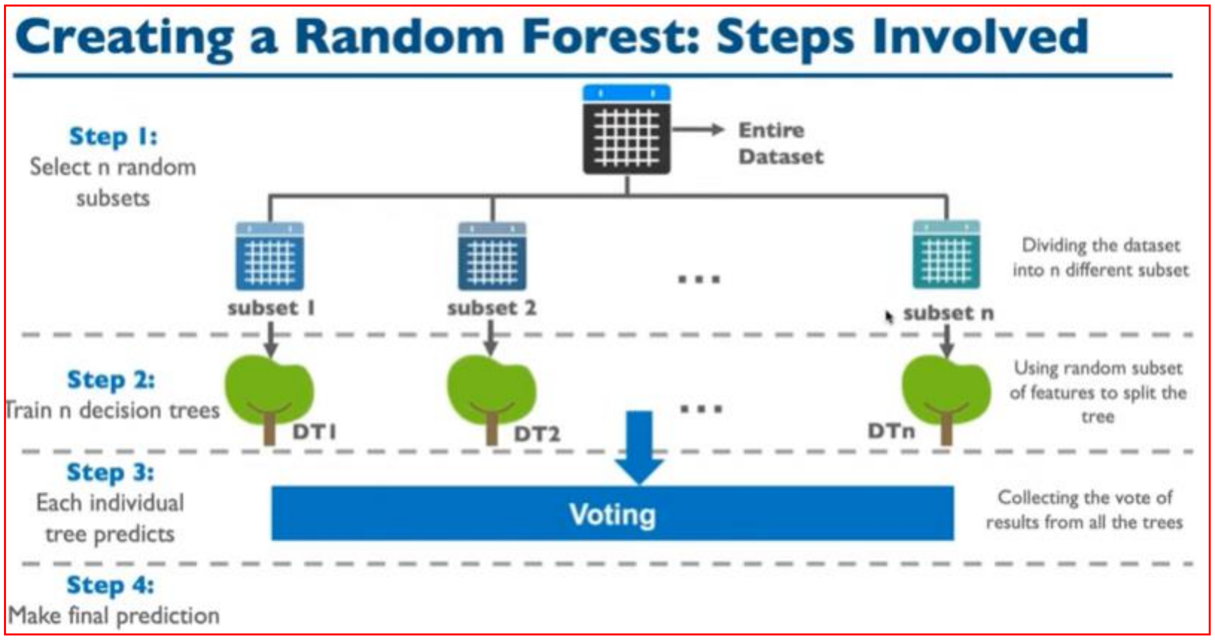
Why Random Forest


User performs a step
Kinect registers the movement
Marks the user based on accuracy
Training set to identify body parts
Random forest classifier learns
Identifies the body parts while dancing
Score game avatar based on accuracy

- Random forest is the most used supervised machine learning algorithm for classification and regression
- RF uses ensemble learning method in which the predictions are based on the combined results of various individual models
- No Overfitting
    - Use of multiple trees reduce the risk of overfitting

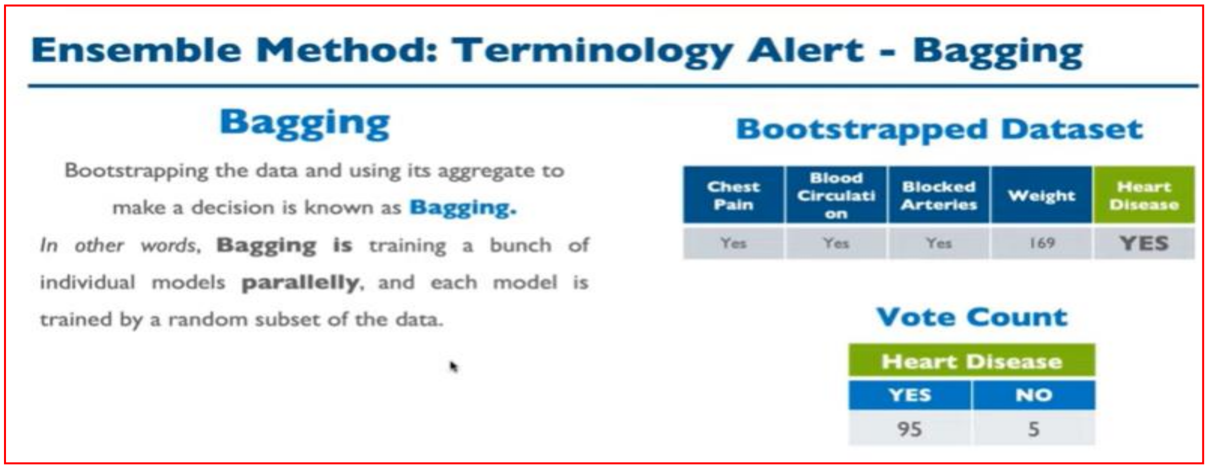
    - Training time is less
- High Accuracy
    - Runs efficiently on large database
    - For large data. It produces highly accurate predictions
- Estimates missing data
    - Random forest can maintain accuracy when a large proportion of data is missing




Creating a Random Forest: Steps Involved
Step 1: Select n random subsets
Entire Dataset
subset 1
subset 2
subset n
Dividing the dataset into n different subset
Step 2: Train n decision trees
DT1
DT2
DTn
Using random subset of features to split the tree
Step 3: Each individual tree predicts
Voting
Collecting the vote of results from all the trees
Step 4: Make final prediction

How does Random Forest algorithm work?

- Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.
- The Working process can be explained in the below steps and diagram:
- **Step-1:** Select random K data points from the training set.
- **Step-2:** Build the decision trees associated with the selected data points (Subsets).
- **Step-3:** Choose the number N for decision trees that you want to build.
- **Step-4:** Repeat Step 1 & 2.
- **Step-5:** For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.



Ensemble Method: Terminology Alert - Bagging

Bagging

Bootstrapping the data and using its aggregate to make a decision is known as **Bagging.**

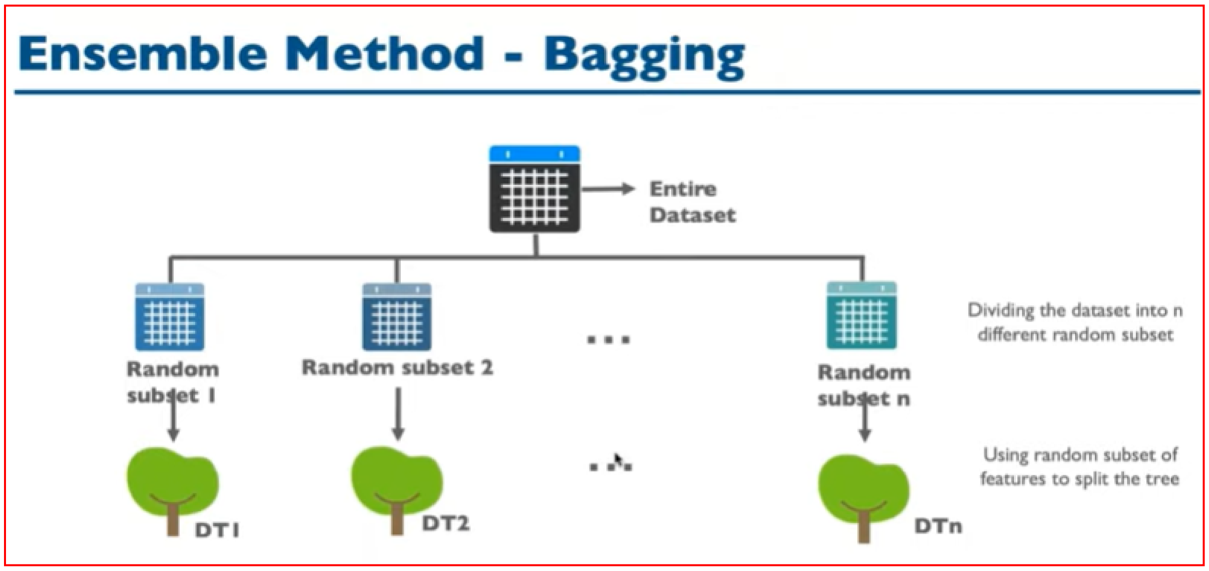
In other words, **Bagging is** training a bunch of individual models **parallelly**, and each model is trained by a random subset of the data.

Bootstrapped Dataset

Chest Pain	Blood Circulation	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	169	YES

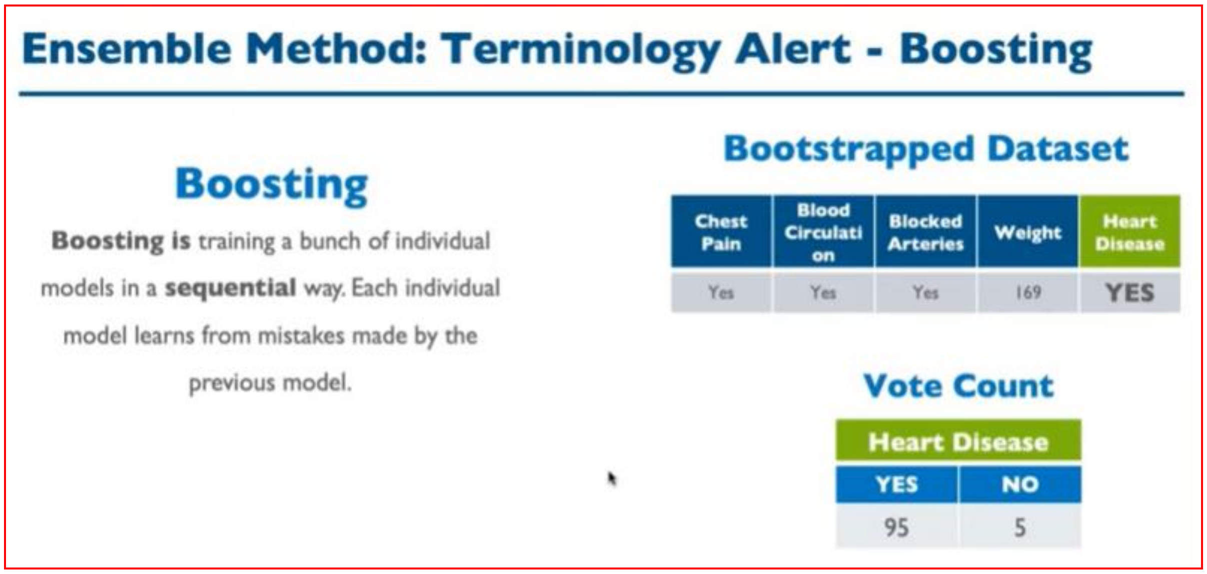
Vote Count

Heart Disease	
YES	NO
95	5



Ensemble Method - Bagging

Entire Dataset
Random subset 1
Random subset 2
Random subset n
Dividing the dataset into n different random subset
Using random subset of features to split the tree
DT1
DT2
DTn



Ensemble Method: Terminology Alert - Boosting

Boosting

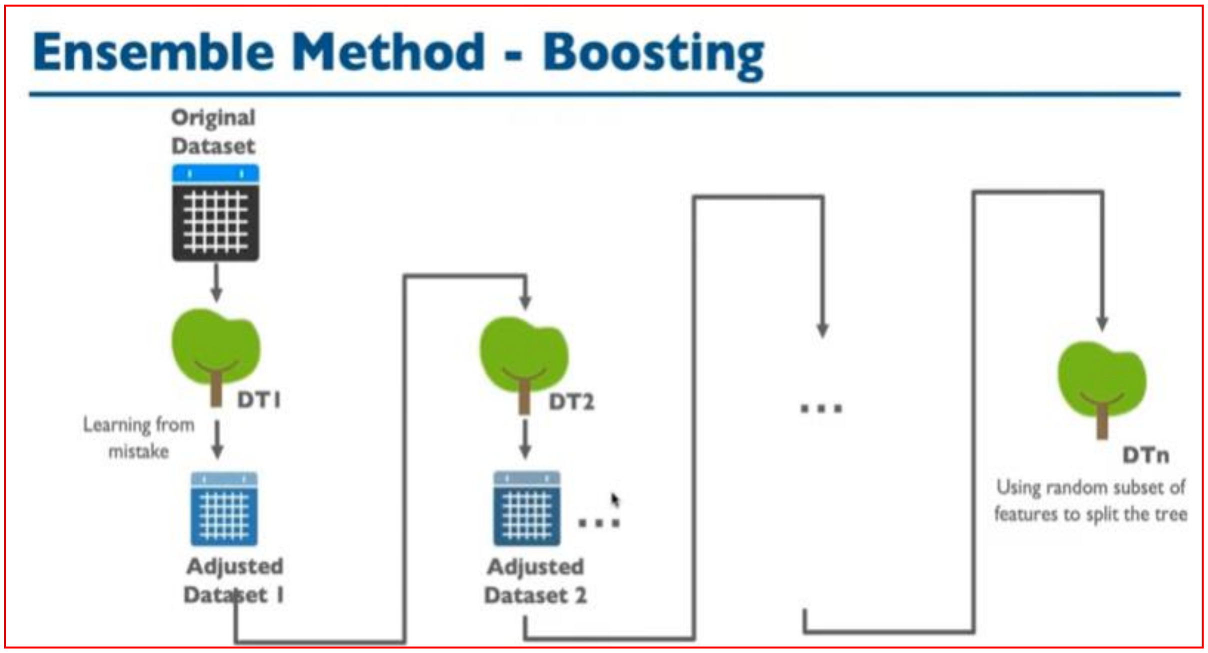
Boosting is training a bunch of individual models in a **sequential** way. Each individual model learns from mistakes made by the previous model.

Bootstrapped Dataset

Chest Pain	Blood Circulation	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	169	YES

Vote Count

Heart Disease	
YES	NO
95	5



How Boosting Algorithm Works?

- The basic principle behind the working of the boosting algorithm is to generate multiple weak learners and combine their predictions to form one strong rule.
- These weak rules are generated by applying base Machine Learning algorithms on different distributions of the data set. These algorithms generate weak rules for each iteration.
- After multiple iterations, the weak learners are combined to form a strong learner that will predict a more accurate outcome.

The algorithm :

- **Step 1:** The base algorithm reads the data and assigns equal weight to each sample observation.
- **Step 2:** False predictions made by the base learner are identified. In the next iteration, these false predictions are assigned to the next base learner with a higher weightage on these incorrect predictions.
- **Step 3:** Repeat step 2 until the algorithm can correctly classify the output.

Therefore, the main aim of Boosting is to focus more on miss-classified predictions

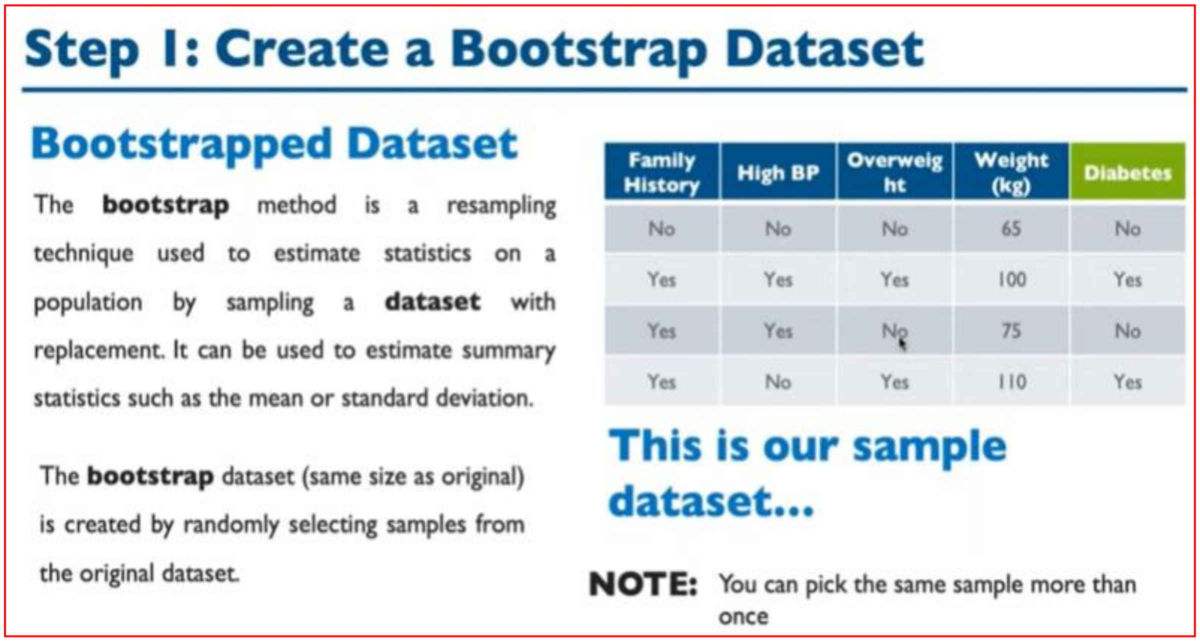
Algorithms based on Bagging and Boosting

- Bagging and Boosting are two of the most commonly used techniques in machine learning. Following are the algorithms will be focusing on:
- Bagging algorithms:
    - Bagging meta-estimator
    - **Random forest**
- Boosting algorithms:
    - **AdaBoost(Adaptive Boosting)**
    - GBM(Gradient Boosting)
    - XGBM
    - Light GBM
    - CatBoost

Similarities Between Bagging and Boosting –

- Both are ensemble methods to get N learners from 1 learner.
- Both generate several training data sets by random sampling.
- Both make the final decision by averaging the N learners (or taking the majority of them i.e Majority Voting).
- Both are good at reducing variance and provide higher stability.

S.No	Boosting	Bagging
1.	Simplest way of combining predictions that belong to the same type	A way of combining predictions that belong to the different types
2.	Aim to decrease variance, not bias	Aim to decrease bias, not variance
3.	Each model receives equal weight	Models are weighted according to their performance
4.	Each model is built independently	New model are influenced by performance of previously built models
5.	Different training data subsets are randomly drawn with replacement from the entire training dataset.	Every new subsets contains the elements that were misclassified by previous models.
6.	Bagging tries to solve overfitting problem	Boosting tries to reduce bias
7.	If the classifier is unstable (high variance) then apply bagging	If the classifier is stable and simple(high bias) the apply boosting
8.	Random forest	Gradient boosting



Step 1: Create a Bootstrap Dataset

Bootstrapped Dataset

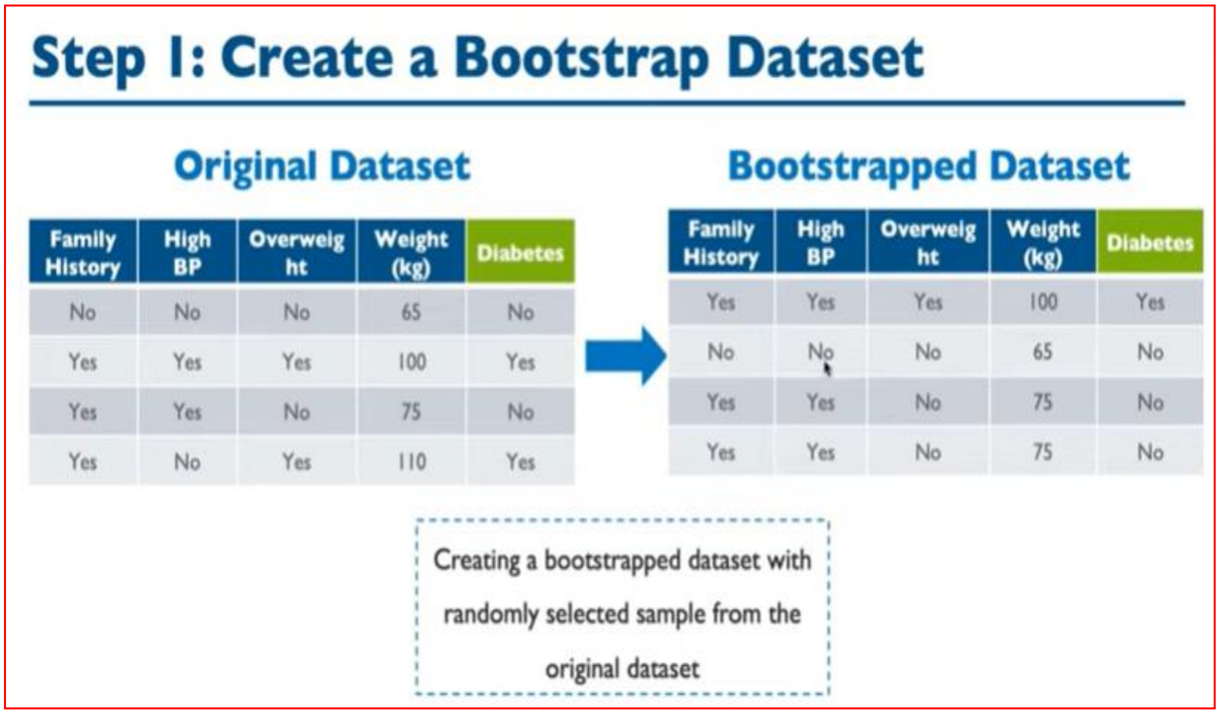
The **bootstrap** method is a resampling technique used to estimate statistics on a population by sampling a **dataset** with replacement. It can be used to estimate summary statistics such as the mean or standard deviation.

The **bootstrap** dataset (same size as original) is created by randomly selecting samples from the original dataset.

Family History	High BP	Overweight	Weight (kg)	Diabetes
No	No	No	65	No
Yes	Yes	Yes	100	Yes
Yes	Yes	No	75	No
Yes	No	Yes	110	Yes

This is our sample dataset...

NOTE: You can pick the same sample more than once



Step 1: Create a Bootstrap Dataset

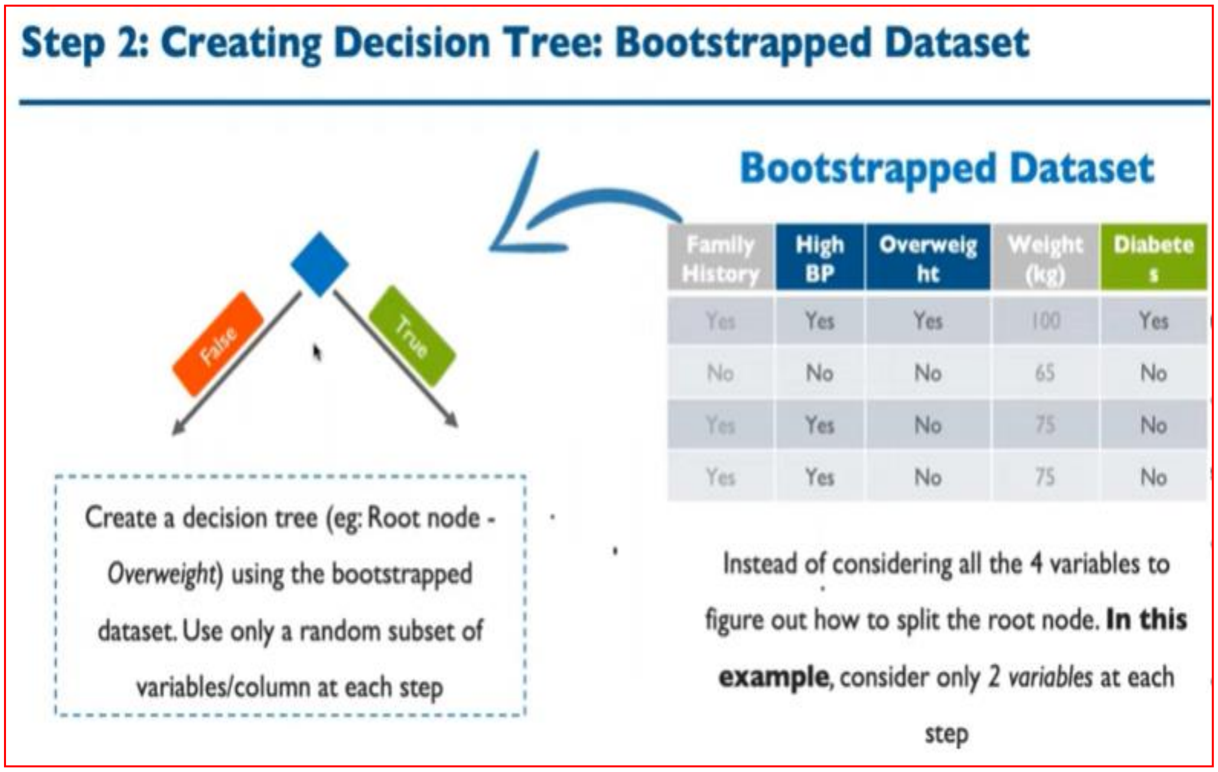
Original Dataset

Family History	High BP	Overweight	Weight (kg)	Diabetes
No	No	No	65	No
Yes	Yes	Yes	100	Yes
Yes	Yes	No	75	No
Yes	No	Yes	110	Yes

Bootstrapped Dataset

Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	Yes	Yes	100	Yes
No	No	No	65	No
Yes	Yes	No	75	No
Yes	Yes	No	75	No

Creating a bootstrapped dataset with randomly selected sample from the original dataset



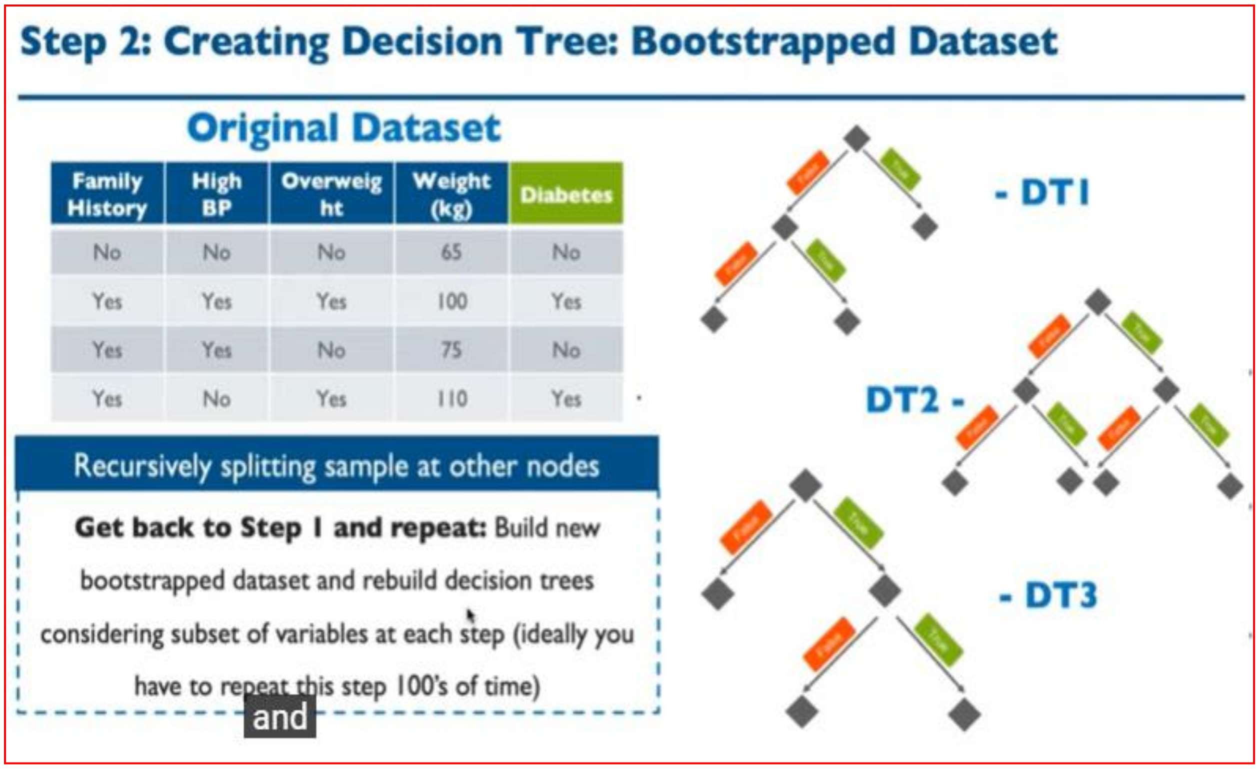
Step 2: Creating Decision Tree: Bootstrapped Dataset

Bootstrapped Dataset

Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	Yes	Yes	100	Yes
No	No	No	65	No
Yes	Yes	No	75	No
Yes	Yes	No	75	No

Create a decision tree (eg: Root node - *Overweight*) using the bootstrapped dataset. Use only a random subset of variables/column at each step

Instead of considering all the 4 variables to figure out how to split the root node. **In this example**, consider only *2 variables* at each step



Step 2: Creating Decision Tree: Bootstrapped Dataset

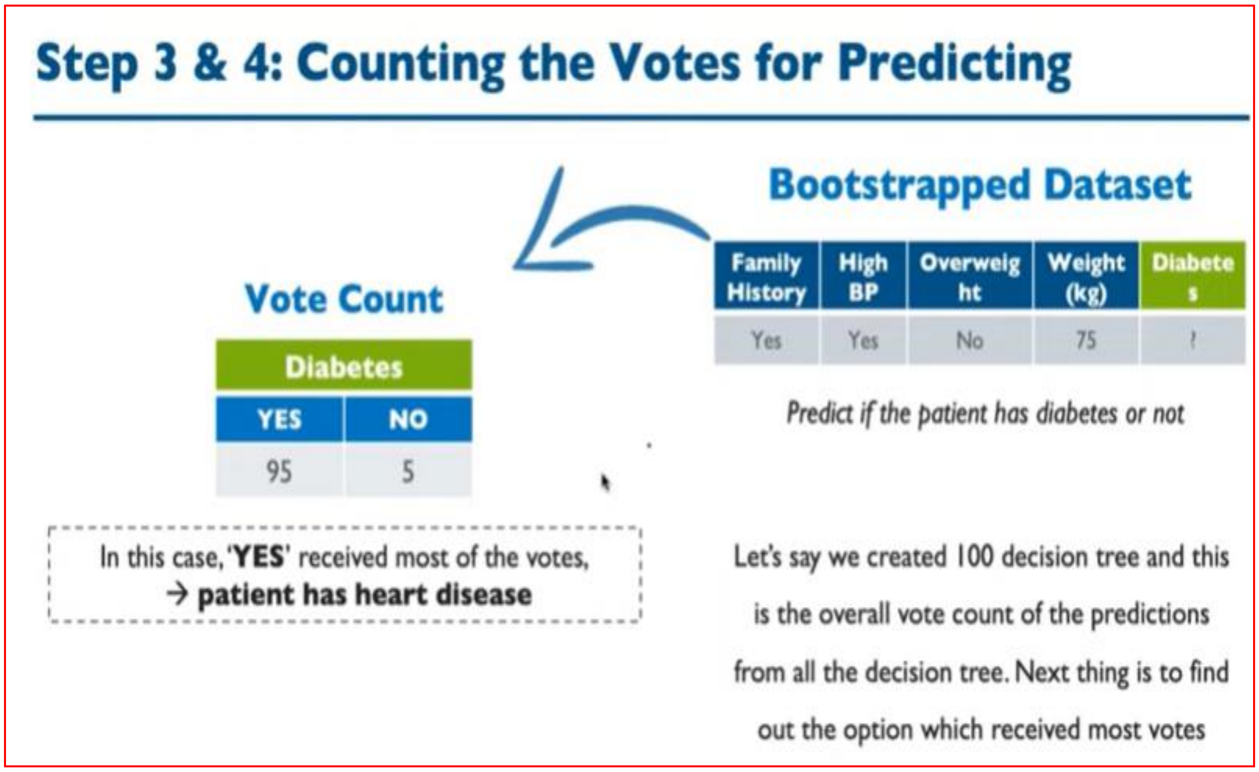
Original Dataset

Family History	High BP	Overweight	Weight (kg)	Diabetes
No	No	No	65	No
Yes	Yes	Yes	100	Yes
Yes	Yes	No	75	No
Yes	No	Yes	110	Yes

Recursively splitting sample at other nodes

Get back to Step 1 and repeat: Build new bootstrapped dataset and rebuild decision trees considering subset of variables at each step (ideally you have to repeat this step 100's of time)

and



Step 3 & 4: Counting the Votes for Predicting

Bootstrapped Dataset

Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	Yes	No	75	?

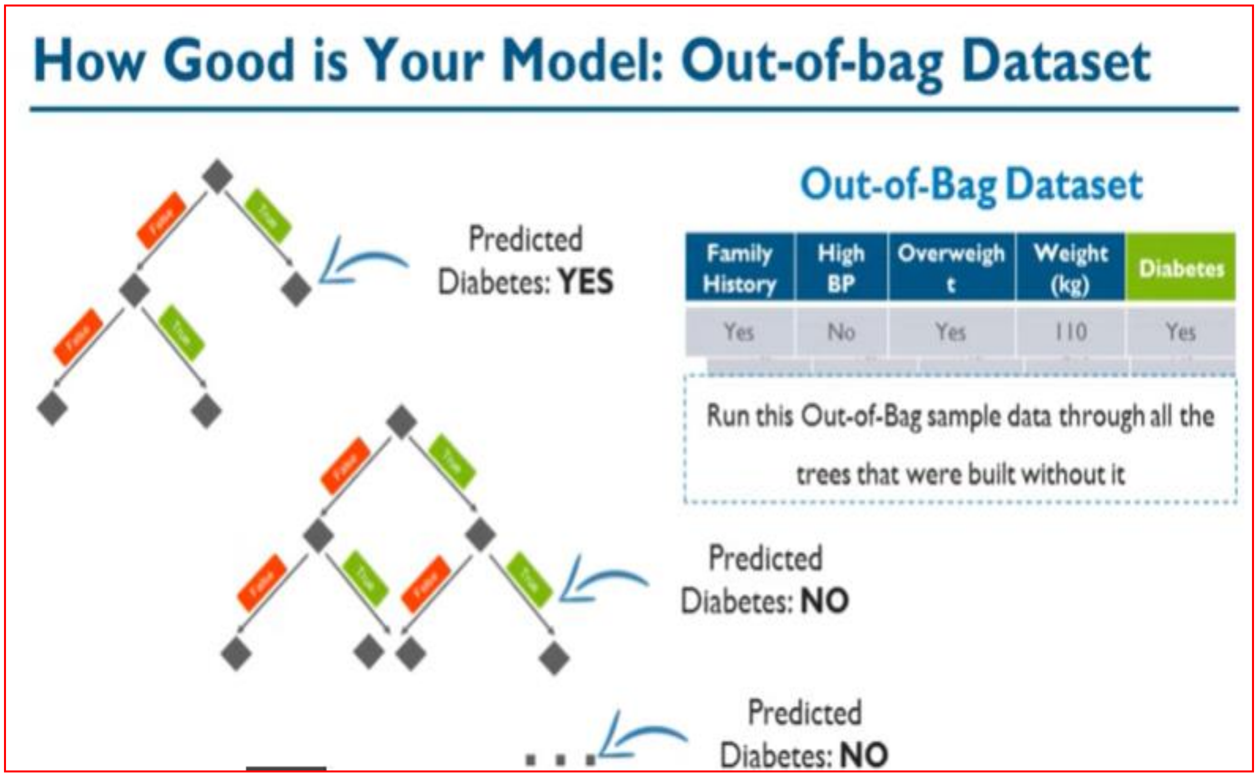
Predict if the patient has diabetes or not

Vote Count

Diabetes	
YES	NO
95	5

In this case, '**YES**' received most of the votes, → **patient has heart disease**

Let's say we created 100 decision tree and this is the overall vote count of the predictions from all the decision tree. Next thing is to find out the option which received most votes



How Good is Your Model: Out-of-bag Dataset

Predicted Diabetes: **YES**

Out-of-Bag Dataset

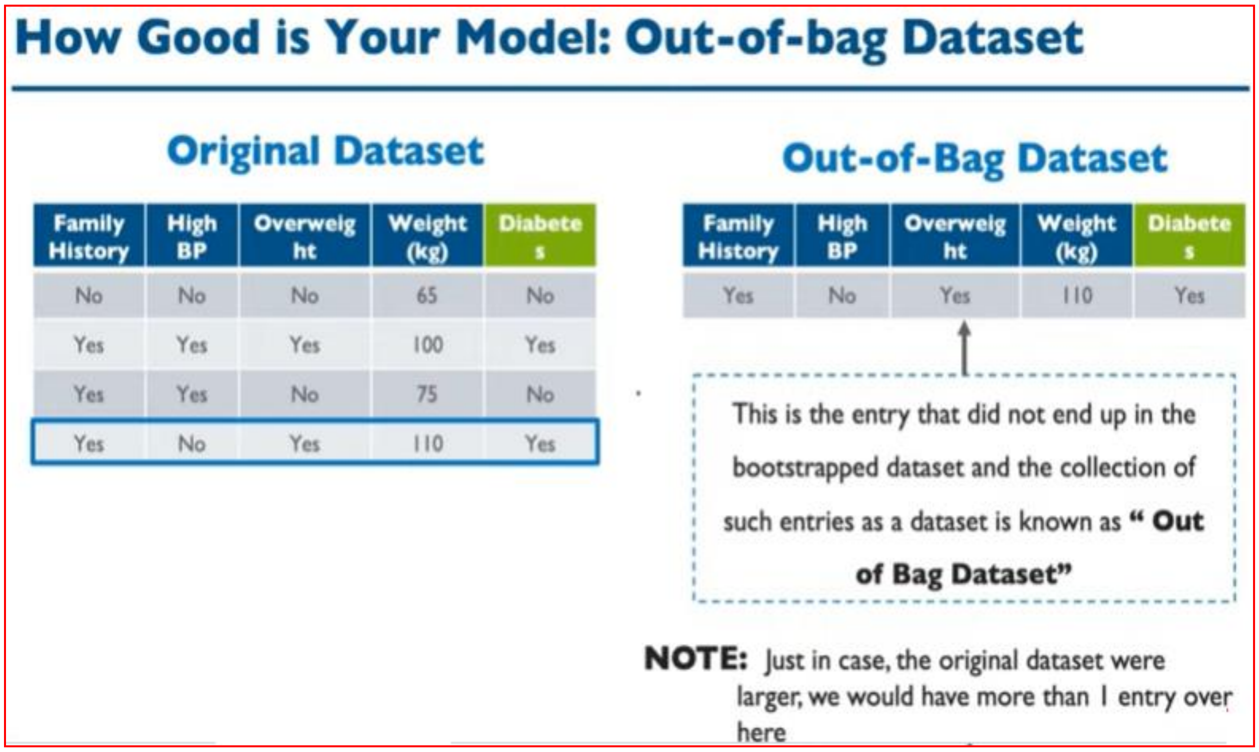
Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	No	Yes	110	Yes

Run this Out-of-Bag sample data through all the trees that were built without it

Predicted Diabetes: **NO**

. . .

Predicted Diabetes: **NO**



How Good is Your Model: Out-of-bag Dataset

Original Dataset

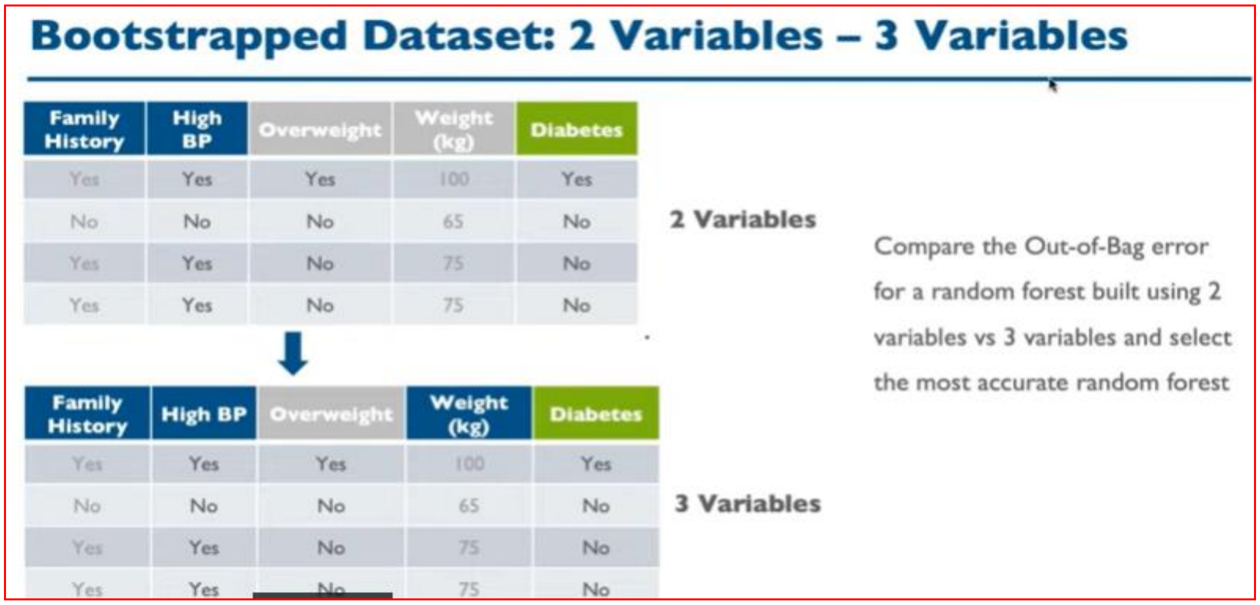
Family History	High BP	Overweight	Weight (kg)	Diabetes
No	No	No	65	No
Yes	Yes	Yes	100	Yes
Yes	Yes	No	75	No
Yes	No	Yes	110	Yes

Out-of-Bag Dataset

Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	No	Yes	110	Yes

This is the entry that did not end up in the bootstrapped dataset and the collection of such entries as a dataset is known as **" Out of Bag Dataset"**

NOTE: Just in case, the original dataset were larger, we would have more than 1 entry over here



Bootstrapped Dataset: 2 Variables – 3 Variables

Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	Yes	Yes	100	Yes
No	No	No	65	No
Yes	Yes	No	75	No
Yes	Yes	No	75	No

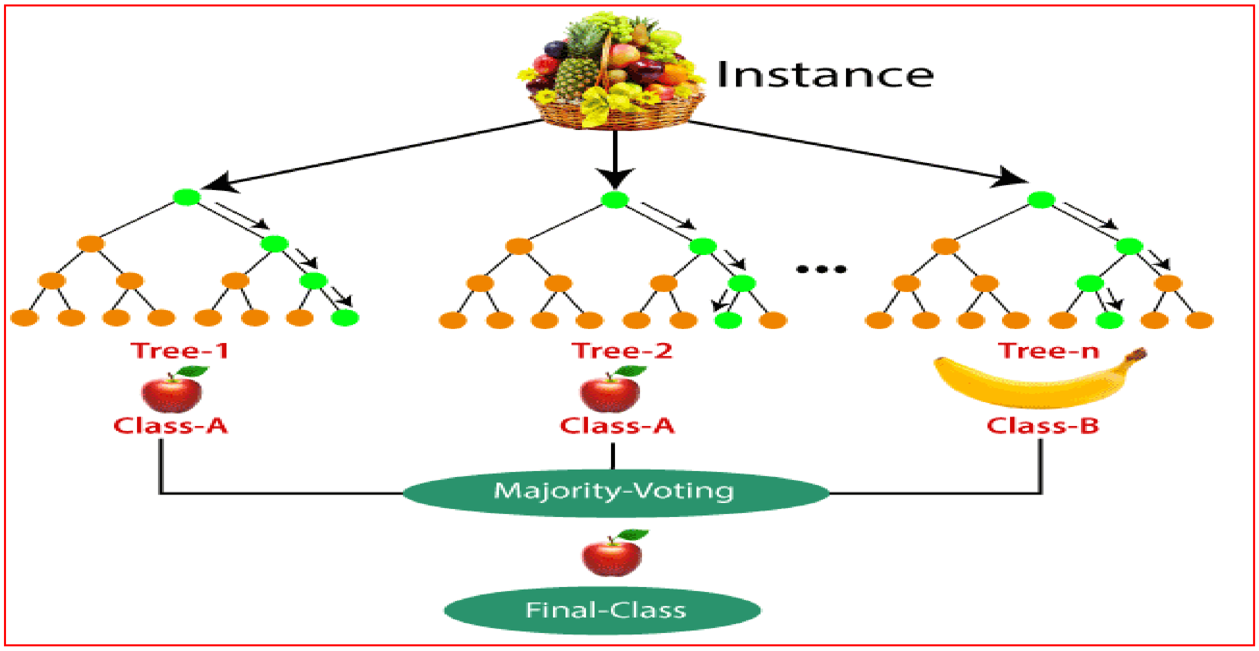
2 Variables

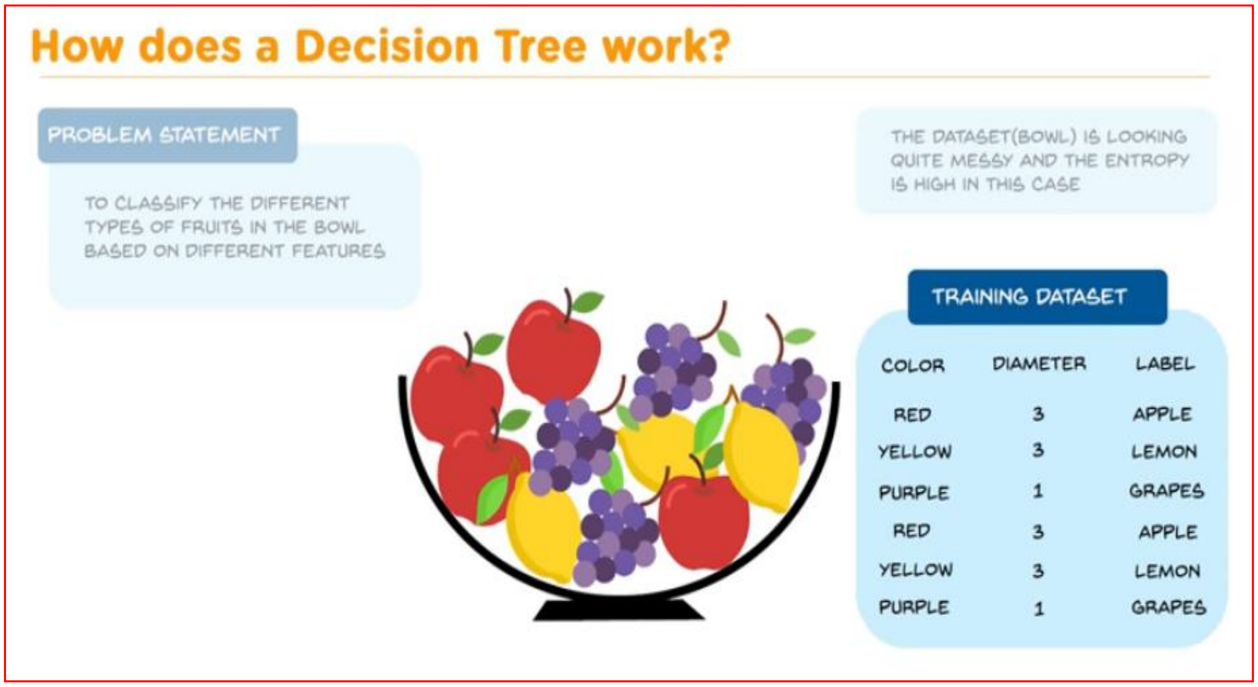
Compare the Out-of-Bag error for a random forest built using 2 variables vs 3 variables and select the most accurate random forest

Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	Yes	Yes	100	Yes
No	No	No	65	No
Yes	Yes	No	75	No
Yes	Yes	No	75	No

3 Variables

Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:





How does a Decision Tree work?

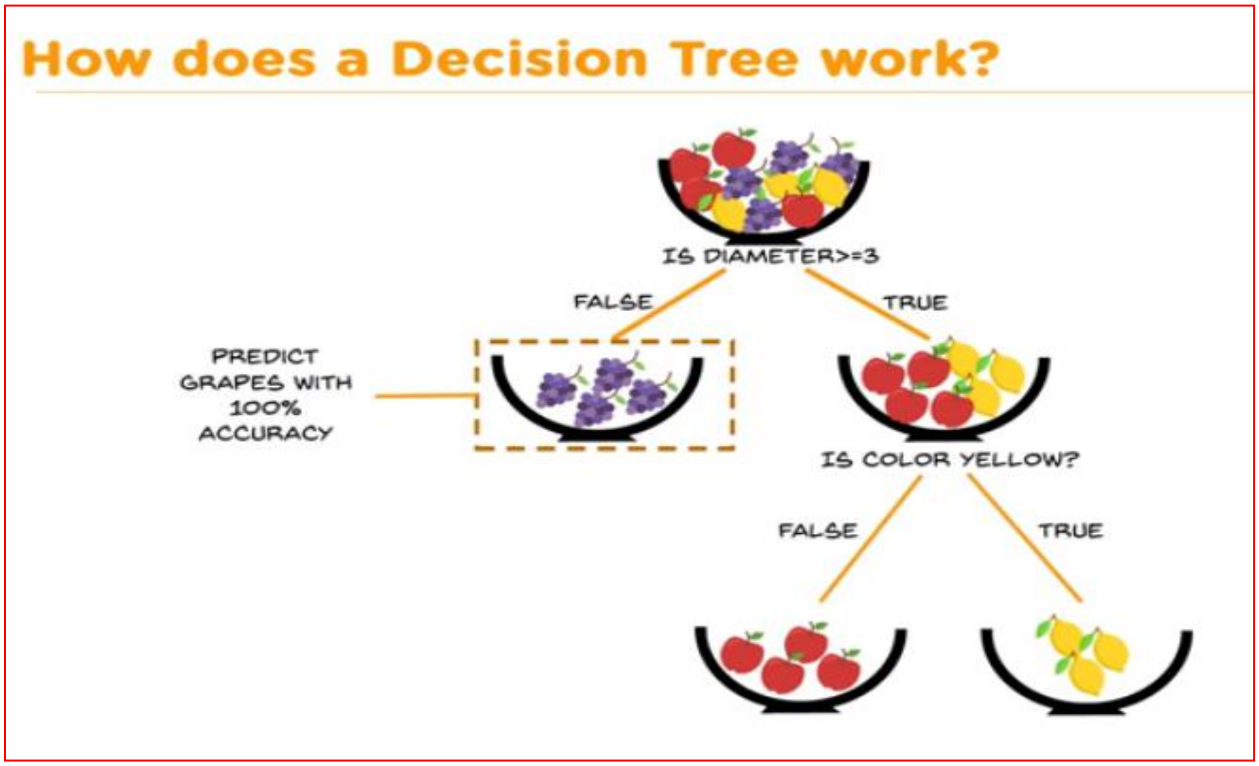
PROBLEM STATEMENT

TO CLASSIFY THE DIFFERENT TYPES OF FRUITS IN THE BOWL BASED ON DIFFERENT FEATURES

THE DATASET(BOWL) IS LOOKING QUITE MESSY AND THE ENTROPY IS HIGH IN THIS CASE

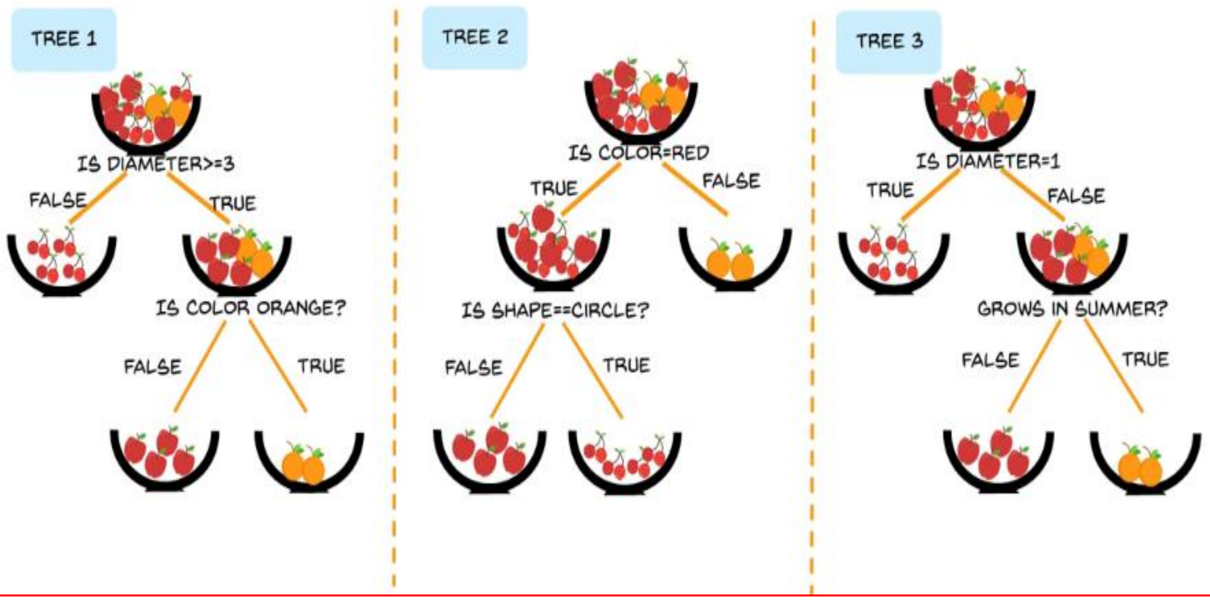
TRAINING DATASET

COLOR	DIAMETER	LABEL
RED	3	APPLE
YELLOW	3	LEMON
PURPLE	1	GRAPES
RED	3	APPLE
YELLOW	3	LEMON
PURPLE	1	GRAPES

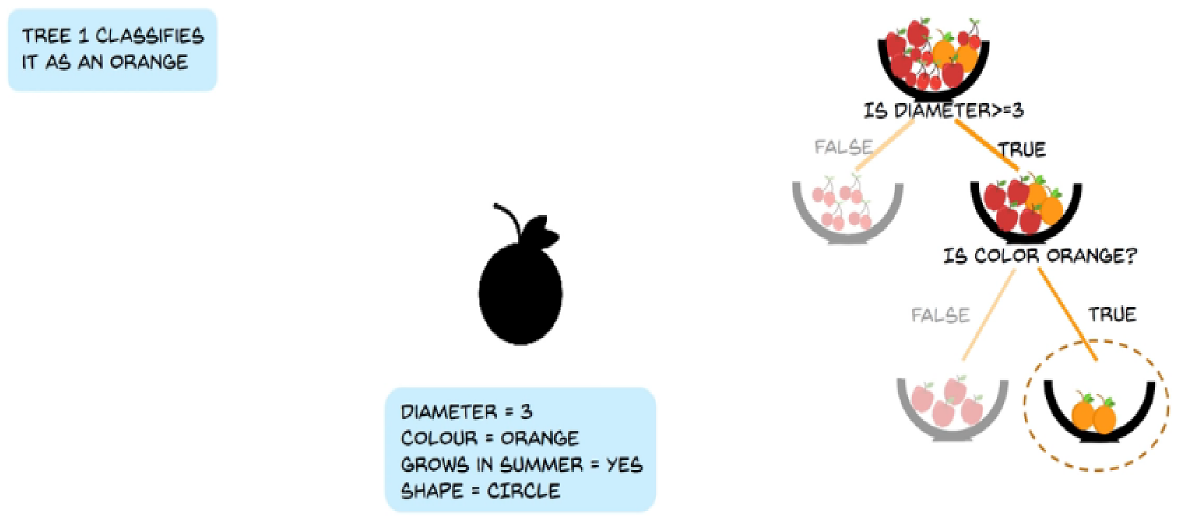


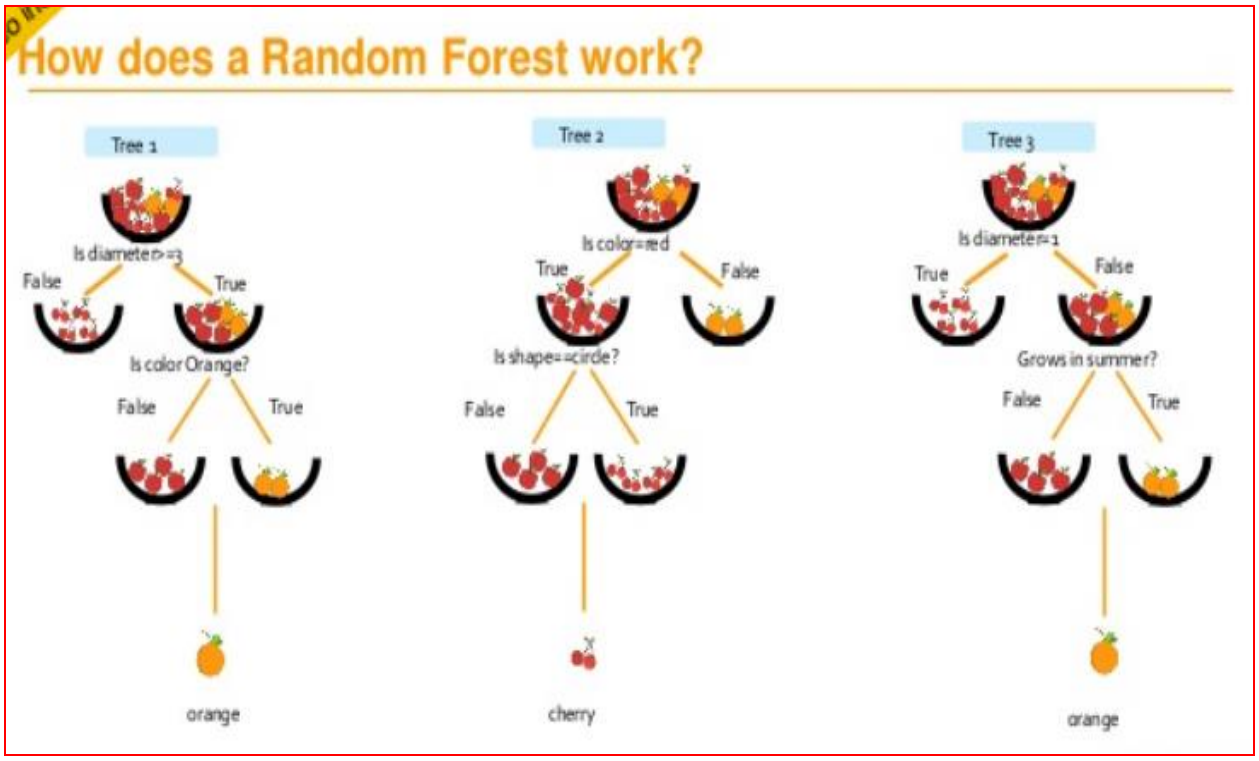
How does a Decision Tree work?

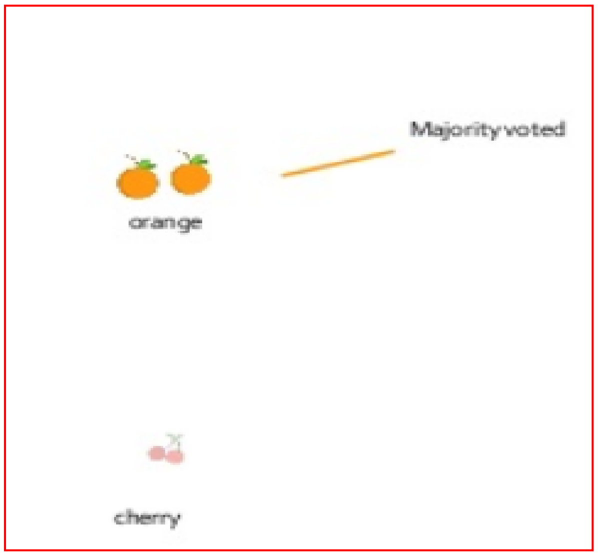
How does a Random Forest work?



How does a Random Forest work?







Applications of Random Forest

There are mainly four sectors where Random forest mostly used:

- **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
- **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
- **Land Use:** We can identify the areas of similar land use by this algorithm.

- **Marketing:** Marketing trends can be identified using this algorithm.

Advantages of Random Forest

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages of Random Forest

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

3.3 Probability & Statistics

- Machine Learning is an interdisciplinary field that uses statistics, probability, algorithms to learn from data and provide insights which can be used to build intelligent applications.

Key concepts widely used in machine learning.

- Probability and statistics are related areas of mathematics which concern themselves with analyzing the relative frequency of events.
- *Probability* deals with predicting the likelihood of future events, while *statistics* involves the analysis of the frequency of past events.

Probability

- Most people have an intuitive understanding of degrees of probability, which is why we use words like “probably” and “unlikely” in our daily conversation, but we will talk about how to make quantitative claims about those degrees .
- In probability theory, an **event** is a set of outcomes of an experiment to which a probability is assigned. If **E** represents an event, then **P(E)** represents the probability that **E** will occur. A situation where **E** might happen (*success*) or might not happen (*failure*) is called a ***trial***.
- This event can be anything like *tossing a coin, rolling a die* or *pulling a colored ball out of a bag*. In these examples the outcome of the event is random, so the variable that represents the outcome of these events is called a **random variable.**
- Let us consider a basic example of tossing a coin. If the coin is fair, then it is just as likely to come up heads as it is to come up tails. In other words, if we were to repeatedly toss the coin many times, we would expect about about half of the tosses to be heads and and half to be tails. In this case, we say that the probability of getting a head is 1/2 or 0.5 .

The **empirical probability** of an event is given by number of times the event occurs divided by the total number of incidents observed. If for trials and we observe **s**uccesses, the probability of success is s/n. In the above example. any sequence of coin tosses may have more or less than exactly 50% heads.

Theoretical probability on the other hand is given by the number of ways the particular event can occur divided by the total number of possible outcomes. So a head can occur once and possible outcomes are two (head, tail). The true (theoretical) probability of a head is 1/2.

Joint Probability

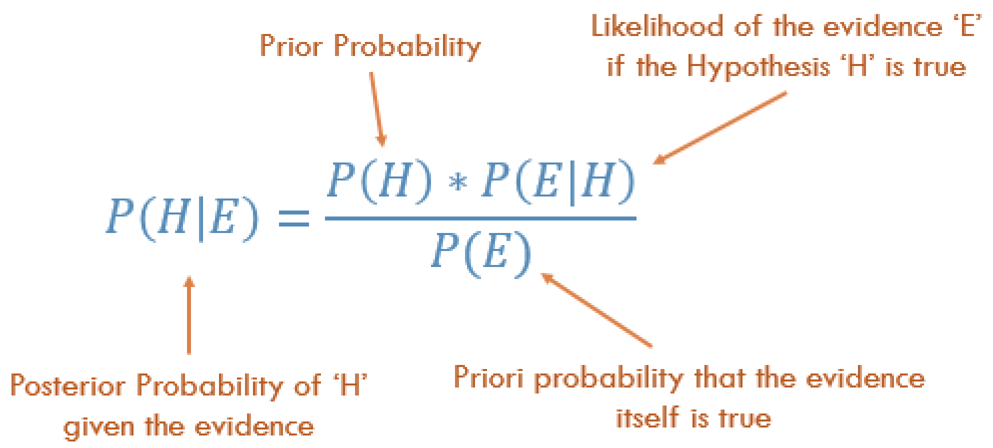
- Probability of events A and B denoted by**P(A and B) or P(A ∩ B)**is the probability that events A and B both occur. **P(A ∩ B) = P(A). P(B) .** This only applies if **A** and **B**are independent, which means that if **O**ccurred, that doesn't change the probability of **B**, and vice versa.

Conditional Probability

- Let us consider A and B are not independent, because if A occurred, the probability of B is higher. When A and B are not independent, it is often useful to compute the conditional probability, P (A|B), which is the probability of A given that B occurred: **P(A|B) = P(A ∩ B)/ P(B).**
- The probability of an event A conditioned on an event B is denoted and defined P(A|B) = P(A∩B)/P(B)
- Similarly, **P(B|A) = P(A ∩ B)/ P(A) .** We can write the joint probability of as A and B as **P(A ∩ B)= p(A).P(B|A)**, which means : *"The chance of both things happening is the chance that the first one happens, and then the second one given the first happened."*

Bayes' Theorem

- Bayes's theorem is a relationship between the conditional probabilities of two events. For example, if we want to find the probability of selling ice cream on a hot and sunny day, Bayes' theorem gives us the tools to use prior knowledge about the likelihood of selling ice cream on any other type of day (rainy, windy, snowy etc.).


Prior Probability
Likelihood of the evidence 'E' if the Hypothesis 'H' is true
$P(H|E) = \frac{P(H) * P(E|H)}{P(E)}$
Posterior Probability of 'H' given the evidence
Priori probability that the evidence itself is true

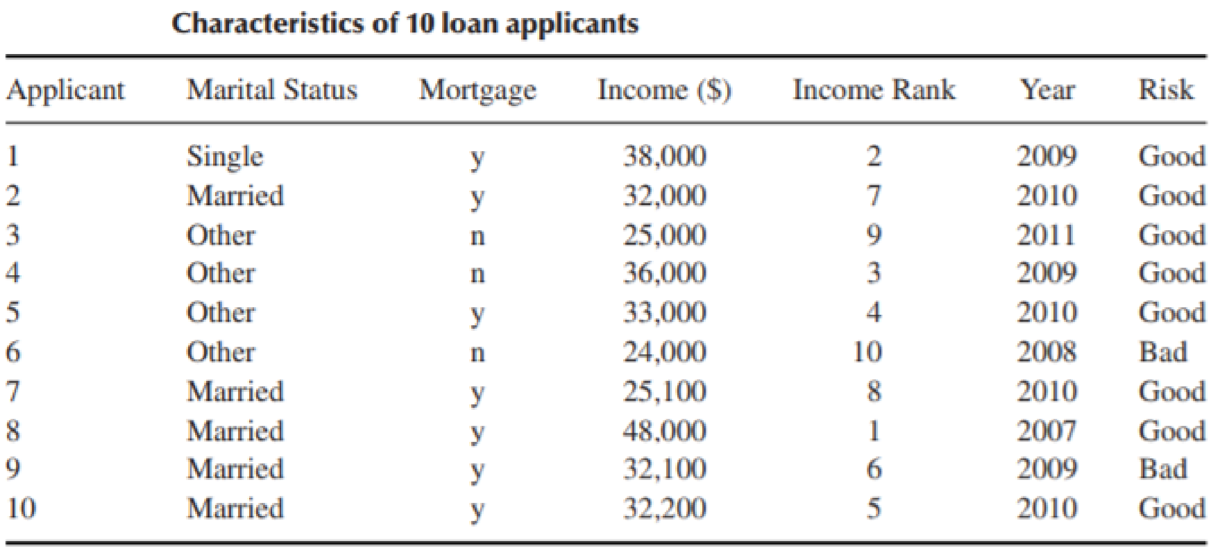
- where *H*and *E* are events, *P(H|E)* is the conditional probability that event *H* occurs given that event *E* has already occurred. The probability *P(H)* in the equation is basically frequency analysis; given our ***prior*** *data* what is the probability of the event occurring. The *P(E|H)* in the equation is called the ***likelihood*** and is essentially the probability that the evidence is correct, given the information from the frequency analysis. *P(E)* is the probability that the actual **evidence** is true.
- Let *H* represent the event that we sell ice cream and *E* be the event of the weather. Then we might ask *what is the probability of selling ice cream on any given day given the type of weather?* Mathematically this is written as P(H=ice cream sale | E= type of weather) which is equivalent to the left-hand side of the equation. *P(H)* on the right-hand side is the expression that is known as the **prior** because we might already know the marginal probability of the sale of ice cream**.** In our example this is *P(H = ice cream sale)*, i.e. the probability of selling ice cream regardless of the type of weather outside. For example, I could look at data that said 30 people out of a potential 100 actually bought ice cream at some shop somewhere. So my *P(H = ice cream sale) = 30/100 = 0.3, prior to me knowing anything about the weather*. This is how Bayes' Theorem allows us to incorporate prior information .
- A classic use of Bayes's theorem is in the interpretation of clinical tests. Suppose that during a routine medical examination, your doctor informs you that you have tested positive for a rare disease. You are also aware that there is some uncertainty in the results of these tests. Assuming we have a **Sensitivity** (also called the **true positive rate)** result for 95% of the

patients with the disease, and a **Specificity** (also called the **true negative rate**) result for 95% of the healthy patients.

- If we let “+” and “–” denote a positive and negative test result, respectively, then the test accuracies are the conditional probabilities : *P*(+|disease) = *0.95*, *P*(-|healthy) = *0.95*,
- In Bayesian terms, we want to compute the probability of disease given a positive test, *P*(disease|+).
- *P*(disease|+) = *P*(+|disease)* *P*(disease)/*P*(+)
- **How to evaluate** *P(+)*, **all positive cases ?** We have to consider two possibilities, *P*(+|disease) and *P*(+|healthy). The probability of a false positive, *P*(+|healthy), is the complement of the *P*(-|healthy). Thus *P*(+|healthy) = 0.05.
- Importantly, Bayes’ theorem reveals that in order to compute the conditional probability that you have the disease given the test was positive, you need to know the “prior” probability you have the disease*P*(disease), given no information at all. That is, you need to know the overall incidence of the disease in the population to which you belong. Assuming these tests are applied to a population where the actual disease is found to be 0.5%, *P*(disease)= *0.005* which means *P*(healthy) = *0.995*.
- So, *P(disease|+) = 0.95 * 0.005 /(0.95 * 0.005 + 0.05 * 0.995) = 0.088*
- In other words, despite the apparent reliability of the test, the probability that you actually have the disease is still less than 9%. Getting a positive result increases the probability you have the disease. But it is incorrect to interpret the 95 % test accuracy as the probability you have the disease.

Descriptive Statistics

- Descriptive statistics refers to methods for summarizing and organizing the information in a data set. We will use below table to describe some of the statistical concepts [4].
- **Elements**: The entities for which information is collected are called the elements. In the above table, the elements are the 10 applicants. Elements are also called cases or subjects.
- **Variables**: The characteristic of an element is called a variable. It can take different values for different elements.e.g., marital status, mortgage, income, rank, year, and risk. Variables are also called attributes.



Characteristics of 10 loan applicants

Applicant	Marital Status	Mortgage	Income ($)	Income Rank	Year	Risk
1	Single	y	38,000	2	2009	Good
2	Married	y	32,000	7	2010	Good
3	Other	n	25,000	9	2011	Good
4	Other	n	36,000	3	2009	Good
5	Other	y	33,000	4	2010	Good
6	Other	n	24,000	10	2008	Bad
7	Married	y	25,100	8	2010	Good
8	Married	y	48,000	1	2007	Good
9	Married	y	32,100	6	2009	Bad
10	Married	y	32,200	5	2010	Good

Variables can be either **qualitative** or **quantitative**.

- **Qualitative:** A qualitative variable enables the elements to be classified or categorized according to some characteristic. The qualitative variables are marital status, mortgage, rank, and risk. Qualitative variables are also called **categorical** variables.
- **Quantitative:** A quantitative variable takes numeric values and allows arithmetic to be meaningfully performed on it. The quantitative variables are income and year. Quantitative variables are also called **numerical** variables.
- **Discrete Variable**: A numerical variable that can take either a finite or a countable number of values is a discrete variable, for which each value can be graphed as a separate point, with space between each point. 'year' is an example of a discrete variable..
- **Continuous Variable**: A numerical variable that can take infinitely many values is a continuous variable, whose possible values form an interval on the number line, with no space between the points. 'income' is an example of a continuous variable.
- **Population**: A population is the set of all elements of interest for a particular problem. A parameter is a characteristic of a population.
- **Sample**: A sample consists of a subset of the population. A characteristic of a sample is called a statistic.

Random sample: When we take a sample for which each element has an equal chance of being selected.

- Measures of Center: Mean, Median, Mode, Mid-range

- *Indicate where on the number line the central part of the data is located.*

Mean

- The mean is the arithmetic average of a data set. To calculate the mean, add up the values and divide by the number of values.The sample mean is the arithmetic average of a sample, and is denoted x̄ (“x-bar”). The population mean is the arithmetic average of a population, and is denoted μ (“myu”, the Greek letter for m).

Median

- The median is the middle data value, when there is an odd number of data values and the data have been sorted into ascending order. If there is an even number, the median is the mean of the two middle data values. When the income data are sorted into ascending order, the two middle values are $32,100 and $32,200, the mean of which is the median income, $32,150.

Mode

- The mode is the data value that occurs with the greatest frequency. Both quantitative and categorical variables can have modes, but only quantitative variables can have means or medians. Each income value occurs only once, so there is no mode. The mode for year is 2010, with a frequency of 4.

Mid-range

- The mid-range is the average of the maximum and minimum values in a data set. The mid-range income is:
- mid-range(income) = (max(income) + min(income))/2 = (48000 + 24000)/2 = $36000
- Measures of Variability: Range, Variance, Standard Deviation
- *Quantify the amount of variation, spread or dispersion present in the data.*

Range

- The range of a variable equals the difference between the maximum and minimum values. The range of income is:
- range(income) = max (income) − min (income) = 48,000 − 24,000 =$24000
- Range only reflects the difference between largest and smallest observation, but it fails to reflect how data is centralized.

Variance

- Population variance is defined as the average of the squared differences from the Mean, denoted as σ^2 (“sigma-squared”):

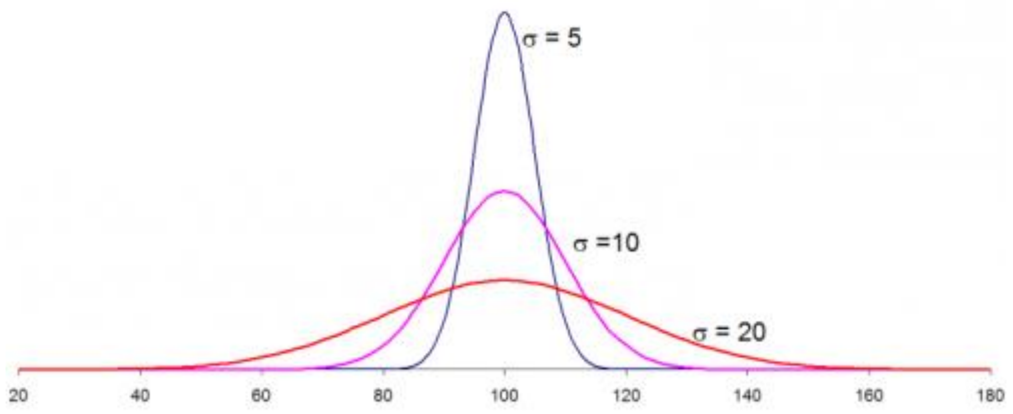
$$\sigma^2 = \frac{\sum (x - \mu)^2}{N}$$

- Larger Variance means the data are more spread out.
- The sample variance s^2 is approximately the mean of the squared deviations, with N replaced by n-1. This difference occurs because the sample mean is used as an approximation of the true population mean.

$$s^2 = \frac{\sum(x - \bar{x})^2}{n - 1}$$

Standard Deviation

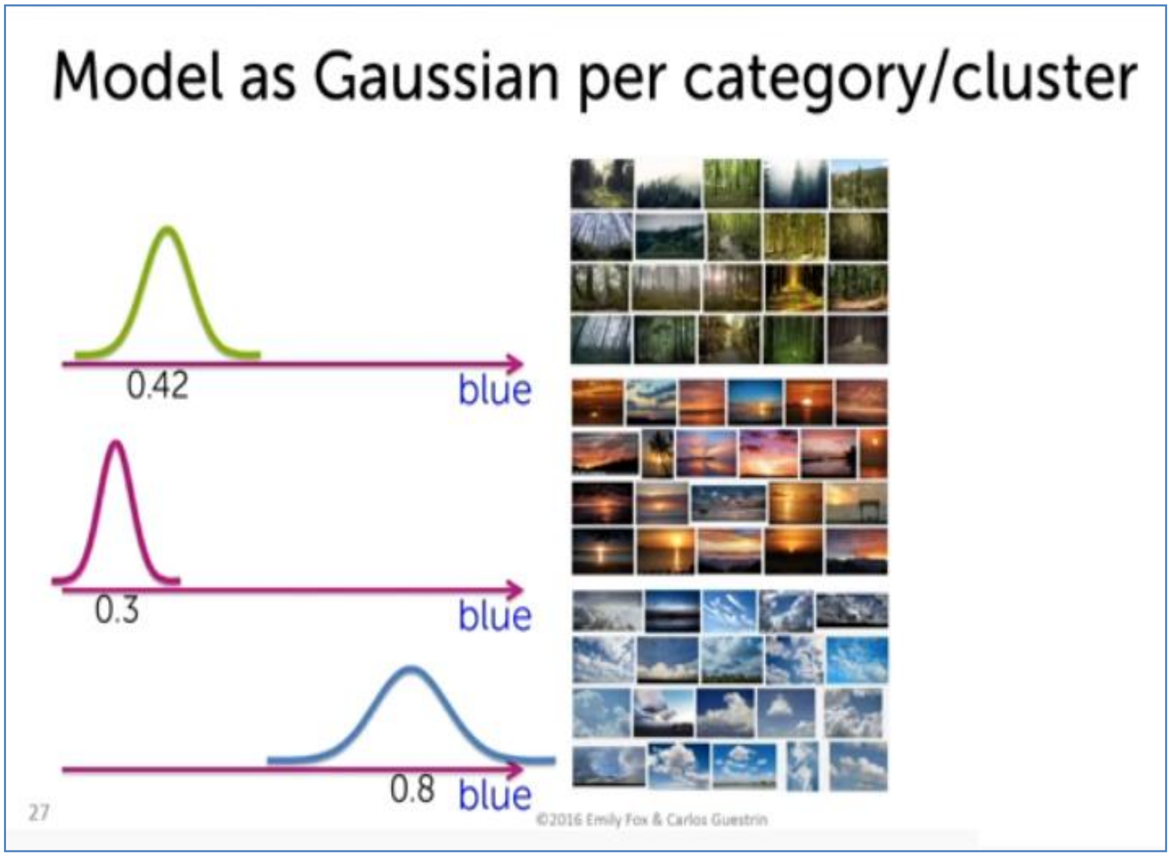
- The *standard deviation* or sd of a bunch of numbers tells you how much the individual numbers tend to differ from the mean.
- The sample standard deviation is the square root of the sample variance: sd = $\sqrt{s^2}$. For example, incomes deviate from their mean by $7201.
- The population standard deviation is the square root of the population variance: sd= $\sqrt{\sigma^2}$.

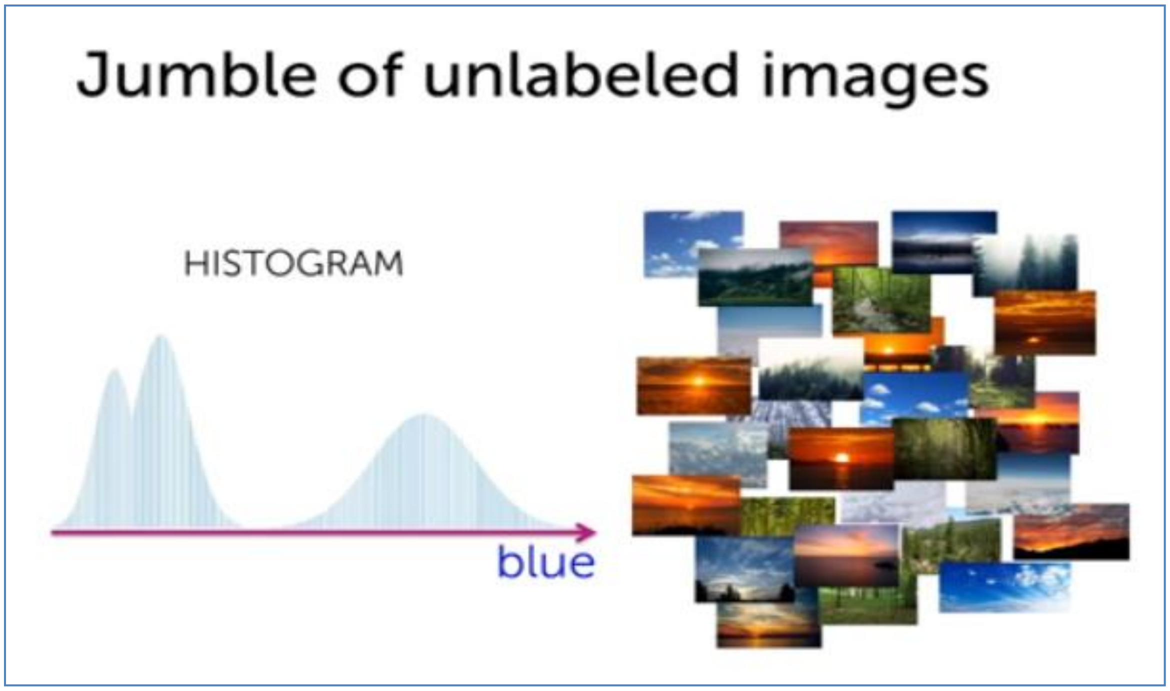


- Three different data distributions with same mean (100) and different standard deviation (5,10,20)
- The smaller the standard deviation, narrower the peak, the data points are closer to the mean. The further the data points are from the mean, the greater the standard deviation.

3.4 Gaussian Mixture Model

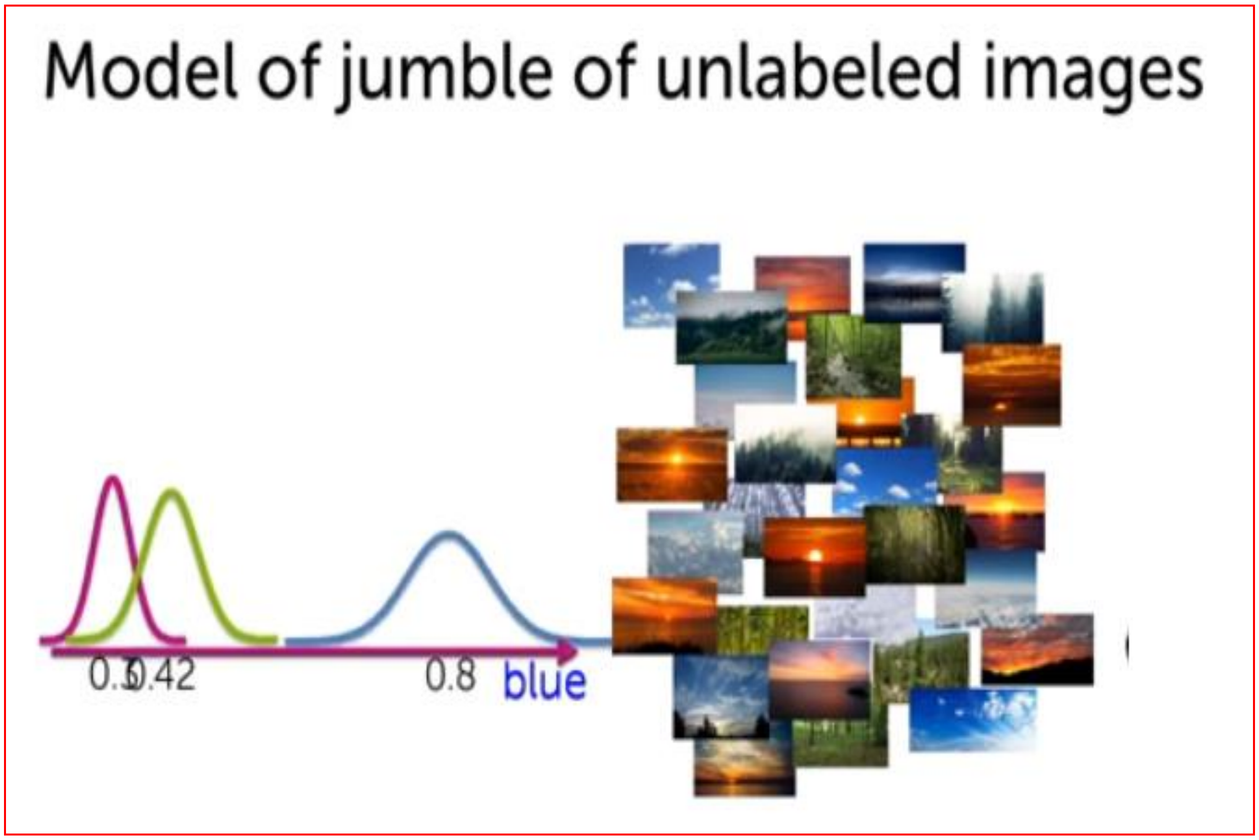
- A mixture model is a model comprised of an unspecified combination of multiple probability distribution functions.
- A statistical procedure or learning algorithm is used to estimate the parameters of the probability distributions to best fit the density of a given training dataset.
- The Gaussian Mixture Model, or GMM for short, is a mixture model that uses a combination of Gaussian (Normal) probability distributions and requires the estimation of the mean and standard deviation parameters for each.


Model as Gaussian per category/cluster
0.42
blue
0.3
blue
0.8
blue
27
©2016 Emily Fox & Carlos Guestrin

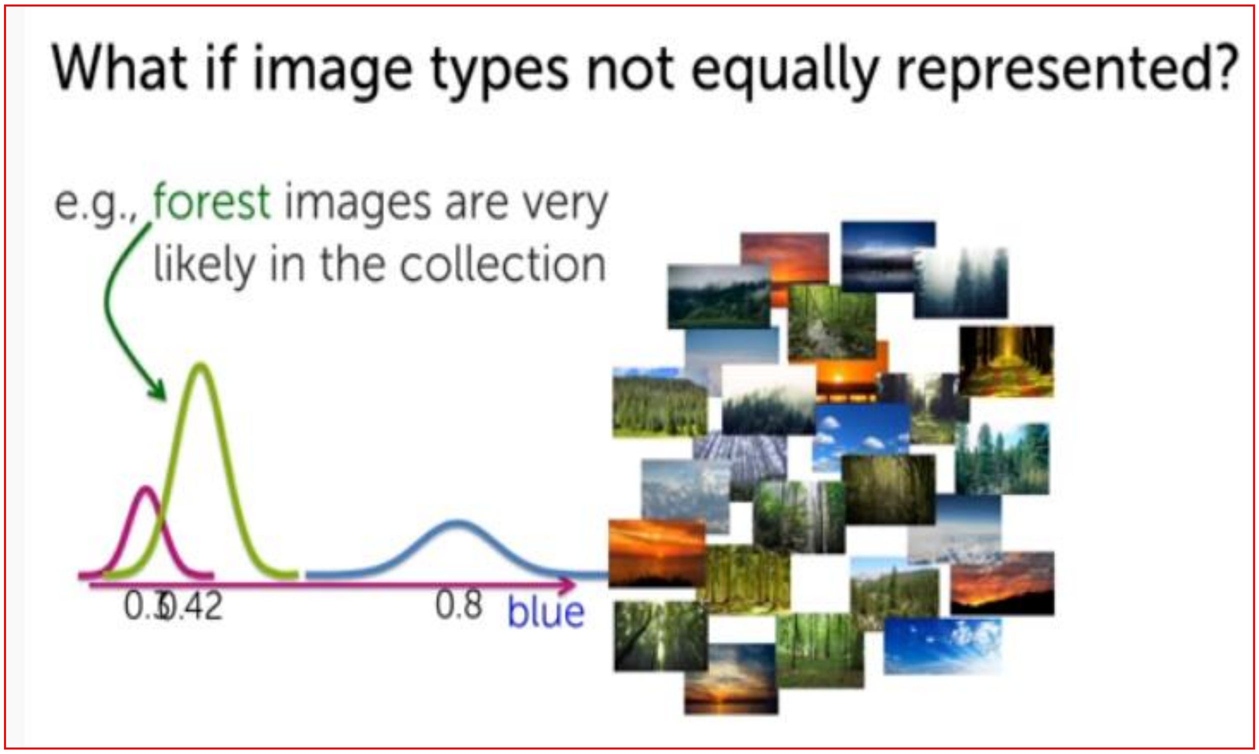


Jumble of unlabeled images

How do we model this distribution?



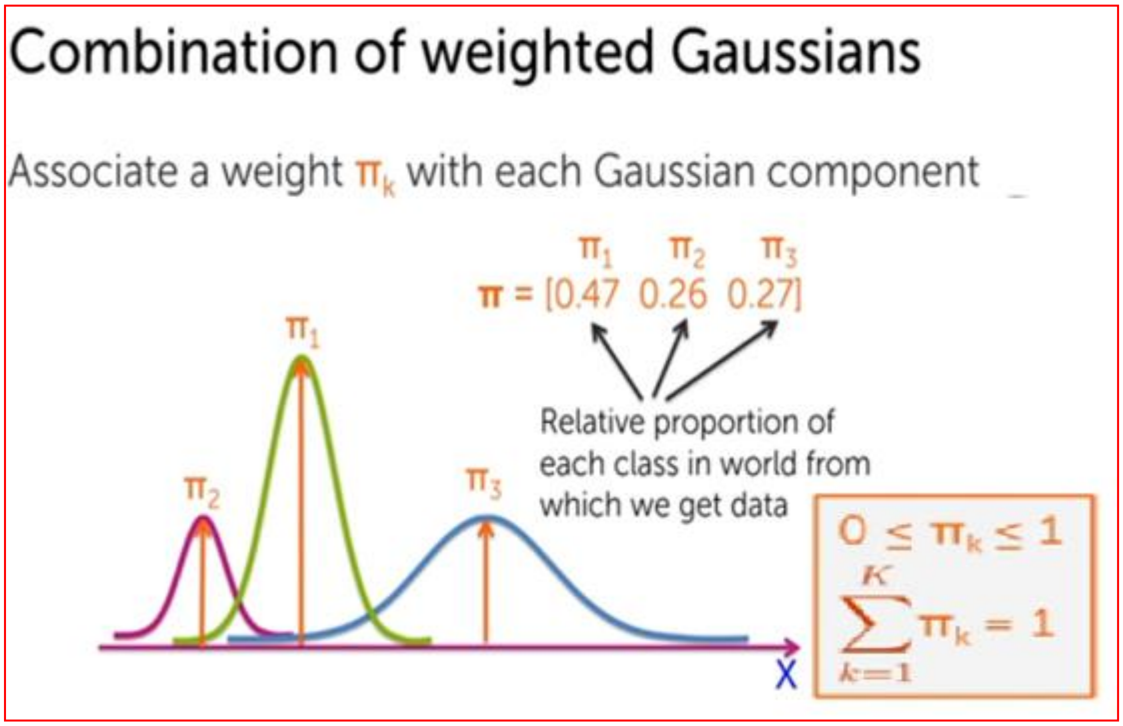
Model of jumble of unlabeled images



What if image types not equally represented?

e.g., forest images are very likely in the collection

Forest images have highest value



Combination of weighted Gaussians

Associate a weight π_k with each Gaussian component

π_1 π_2 π_3

$\pi = [0.47\ \ 0.26\ \ 0.27]$

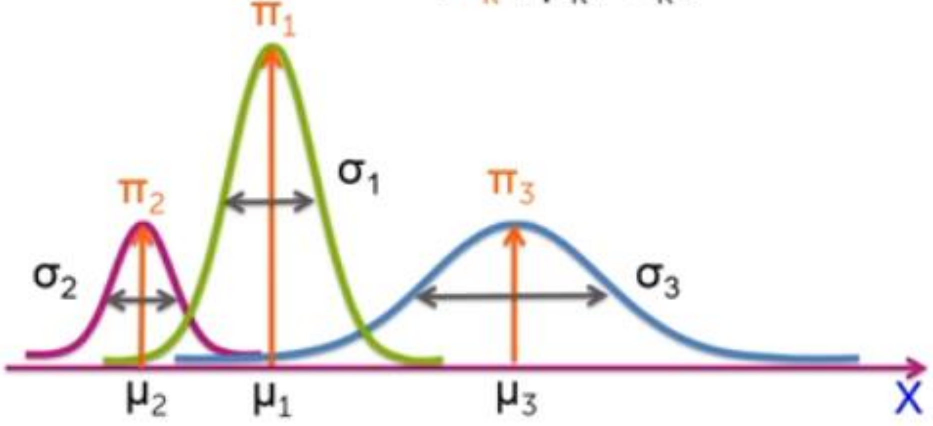
Relative proportion of each class in world from which we get data

$$\boxed{0 \leq \pi_k \leq 1 \qquad \sum_{k=1}^{K} \pi_k = 1}$$

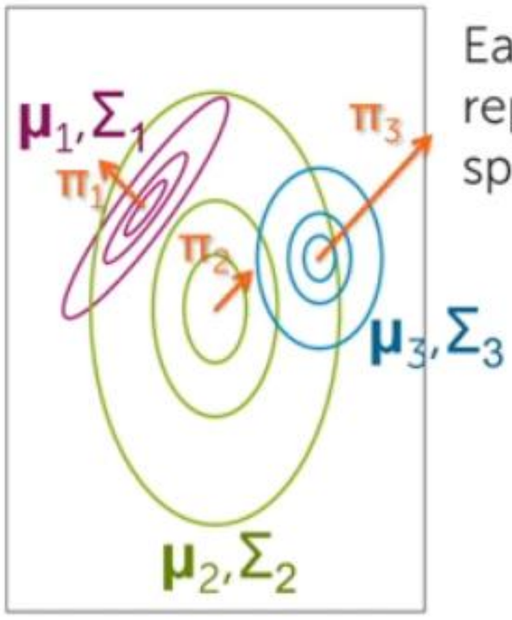
Mixture of Gaussians (1D)

Each mixture component represents a unique cluster specified by:

$$\{\pi_k, \mu_k, \sigma_k^2\}$$

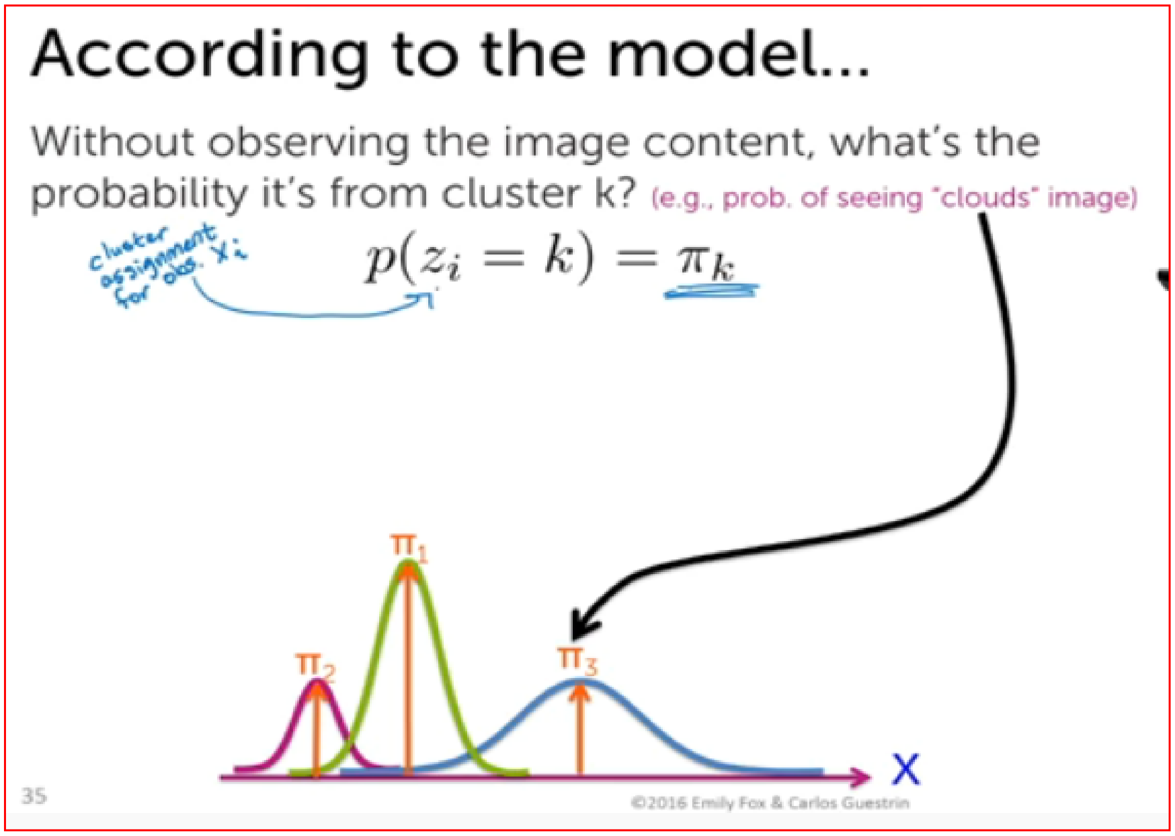


Mixture of Gaussians (general)



Each mixture component represents a unique cluster specified by:

$$\{\pi_k, \mu_k, \Sigma_k\}$$

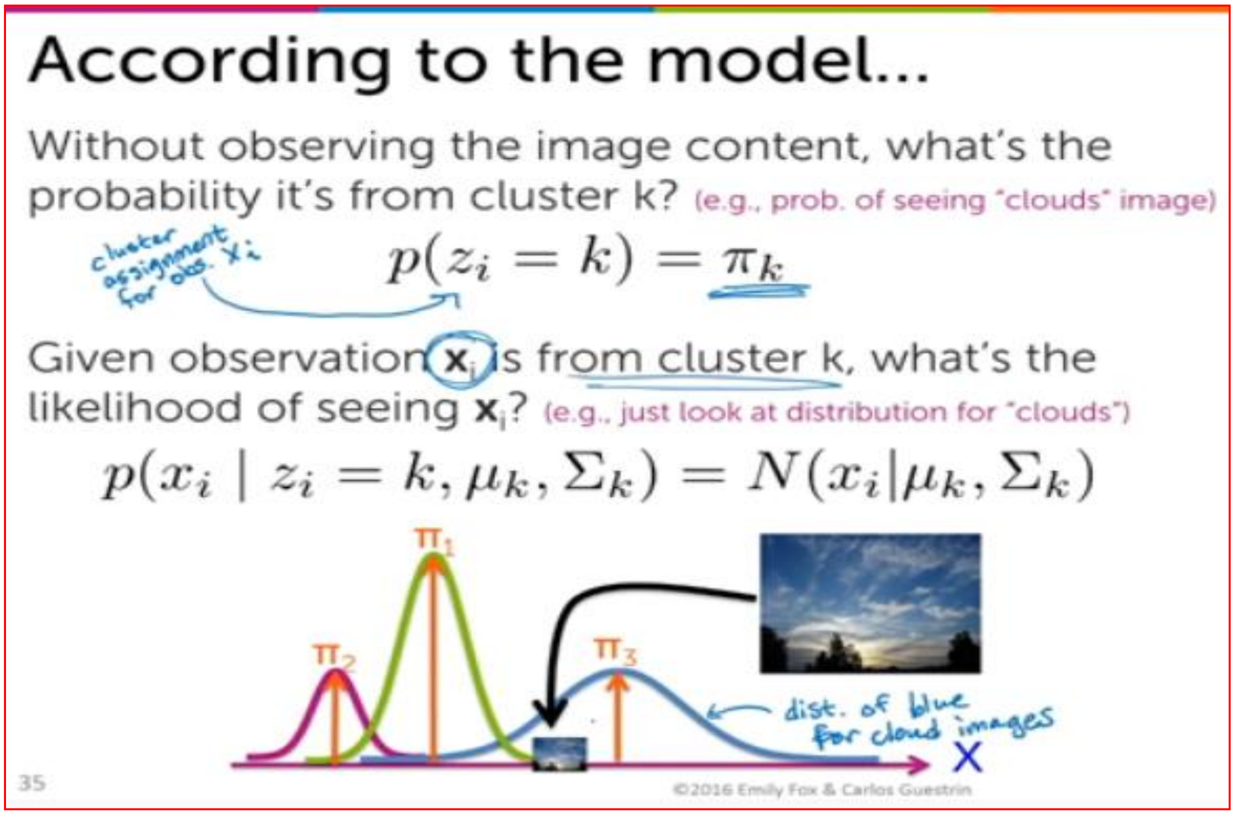


According to the model...

Without observing the image content, what's the probability it's from cluster k? (e.g., prob. of seeing "clouds" image)

cluster assignment for obs. x_i

$$p(z_i = k) = \underline{\pi_k}$$



According to the model...

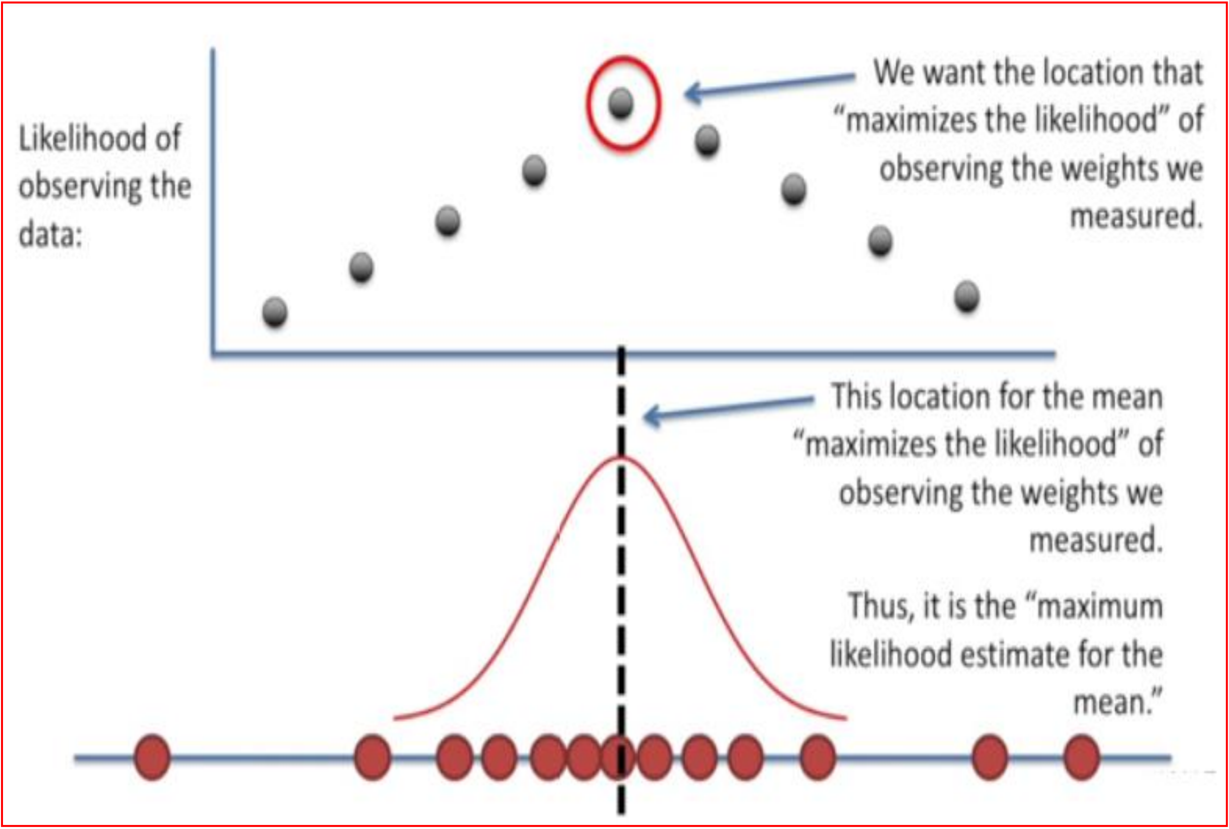
Without observing the image content, what's the probability it's from cluster k? (e.g., prob. of seeing "clouds" image)

cluster assignment for obs. x_i

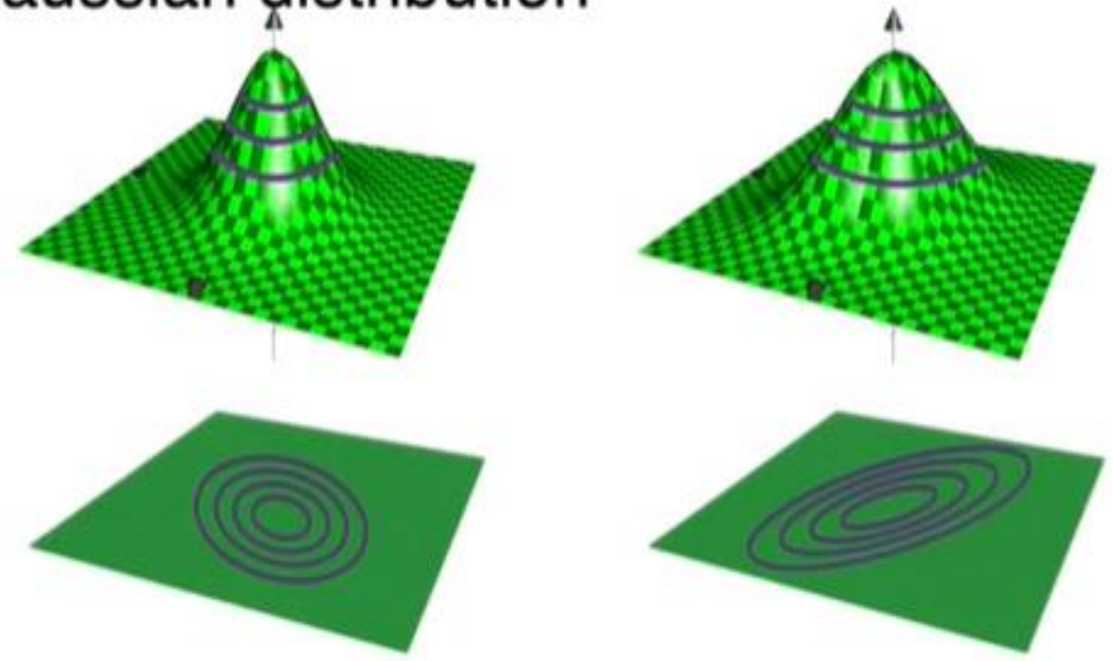
$$p(z_i = k) = \underline{\pi_k}$$

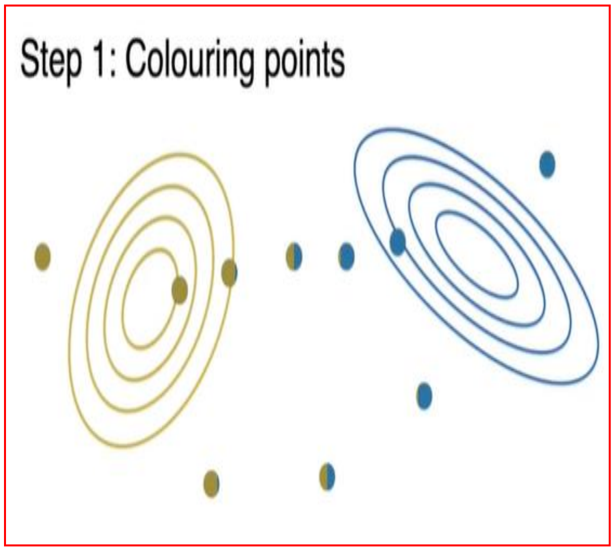
Given observation $\mathbf{x}_i$ is from cluster k, what's the likelihood of seeing $\mathbf{x}_i$? (e.g., just look at distribution for "clouds")

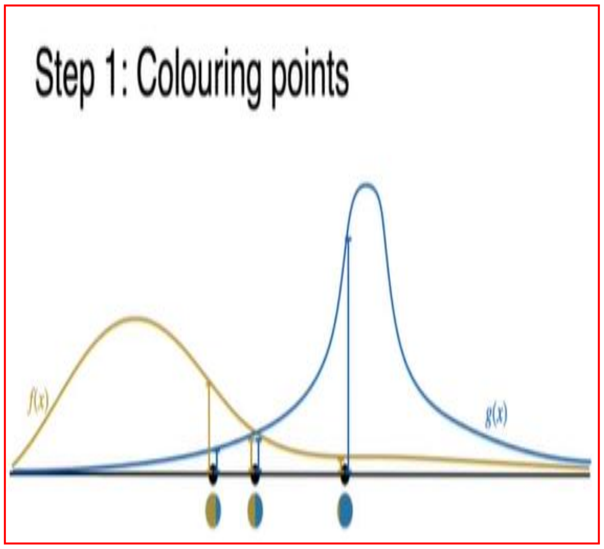
$$p(x_i \mid z_i = k, \mu_k, \Sigma_k) = N(x_i \mid \mu_k, \Sigma_k)$$

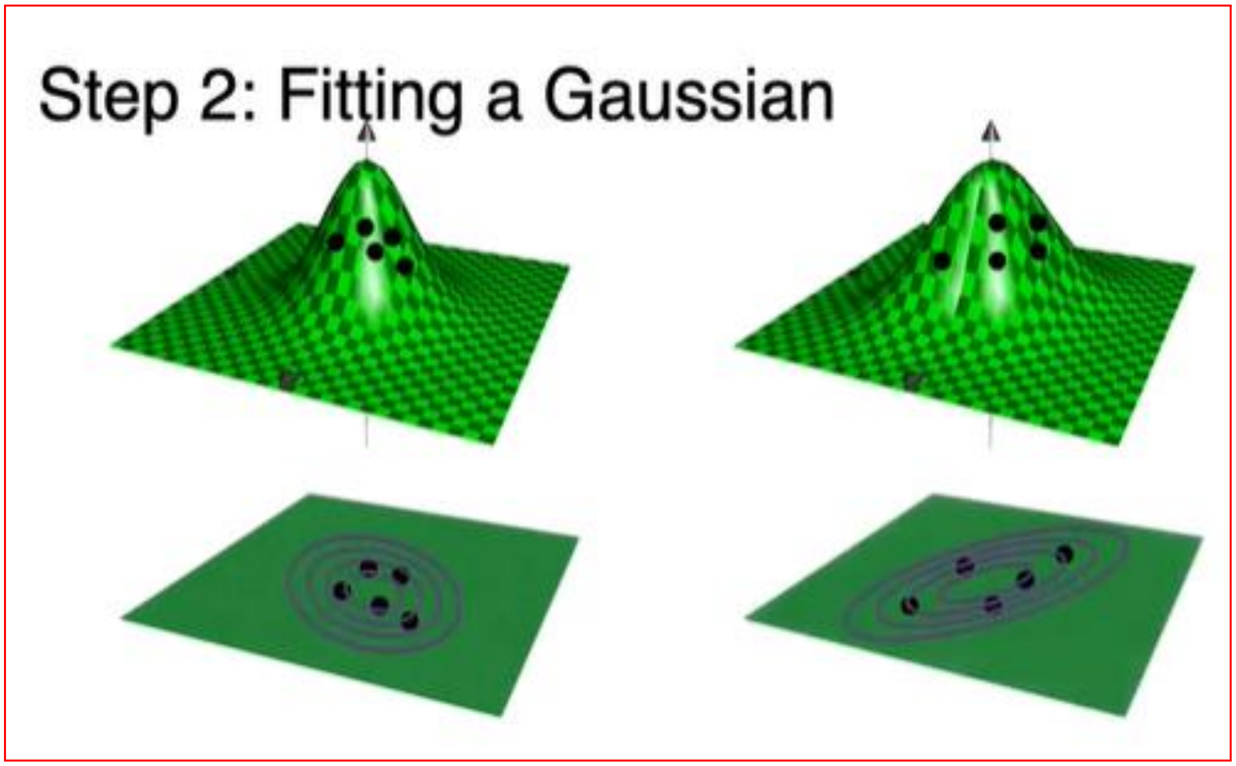

Likelihood of observing the data:
We want the location that "maximizes the likelihood" of observing the weights we measured.
This location for the mean "maximizes the likelihood" of observing the weights we measured.
Thus, it is the "maximum likelihood estimate for the mean."

Gaussian distribution

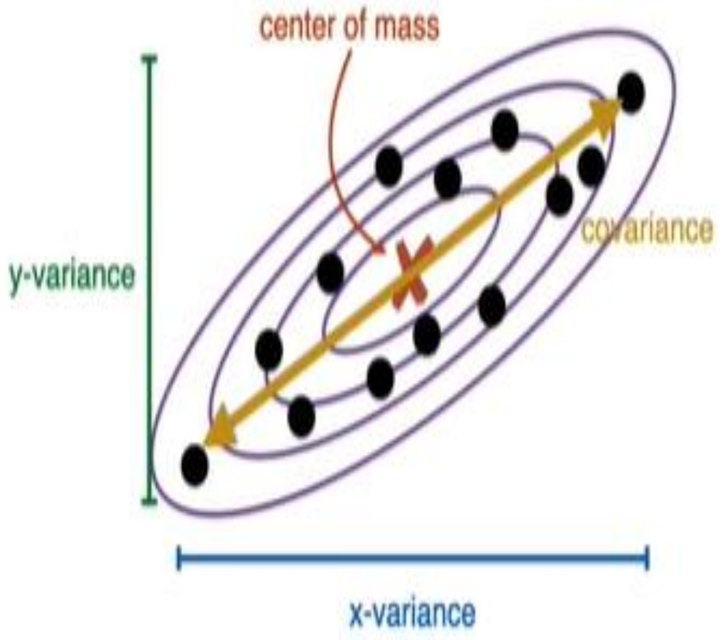



Step 1: Colouring points


Step 1: Colouring points
f(x)
g(x)


Step 2: Fitting a Gaussian

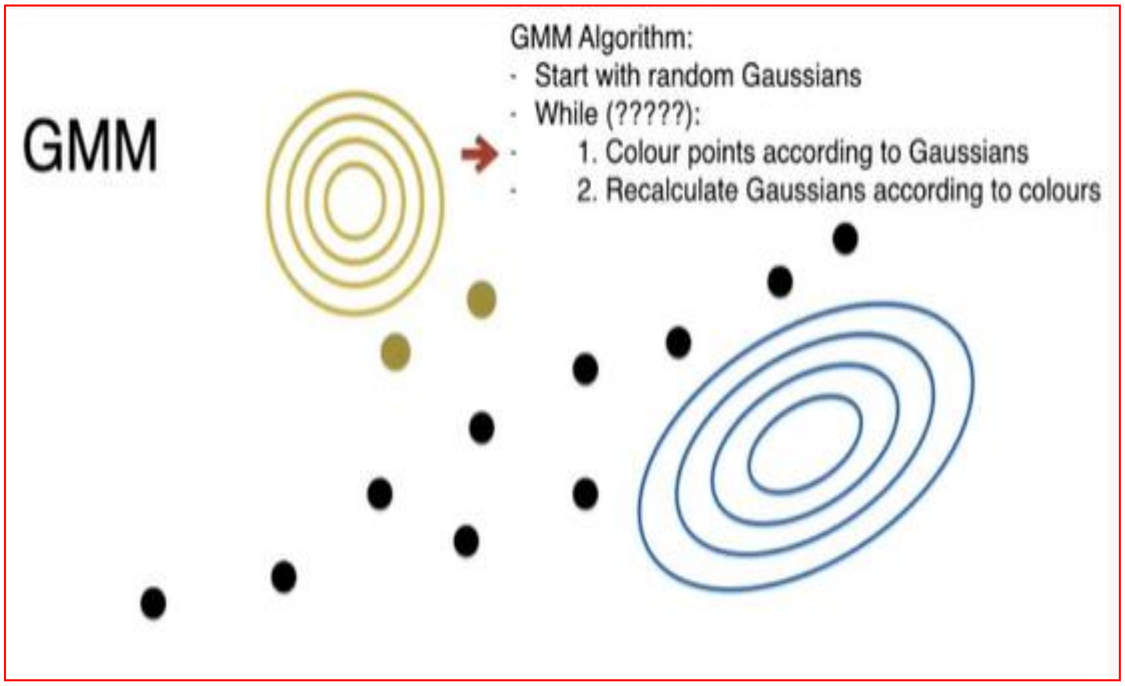
Step 2: Fitting a Gaussian



$$\mu = Average$$

$$\Sigma = \begin{pmatrix} Var(x) & Cov(x,y) \\ Cov(x,y) & Var(y) \end{pmatrix}$$

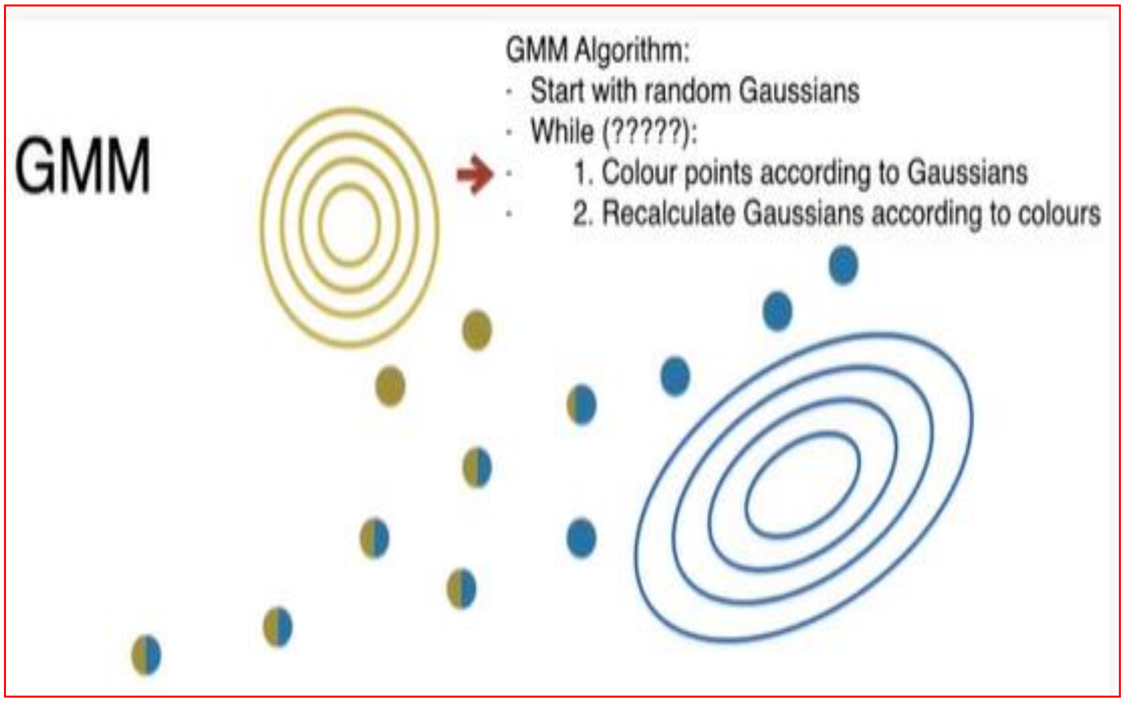
$$f(x) = \frac{\exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))}{2\pi\sqrt{|\Sigma|}}$$



GMM

GMM Algorithm:

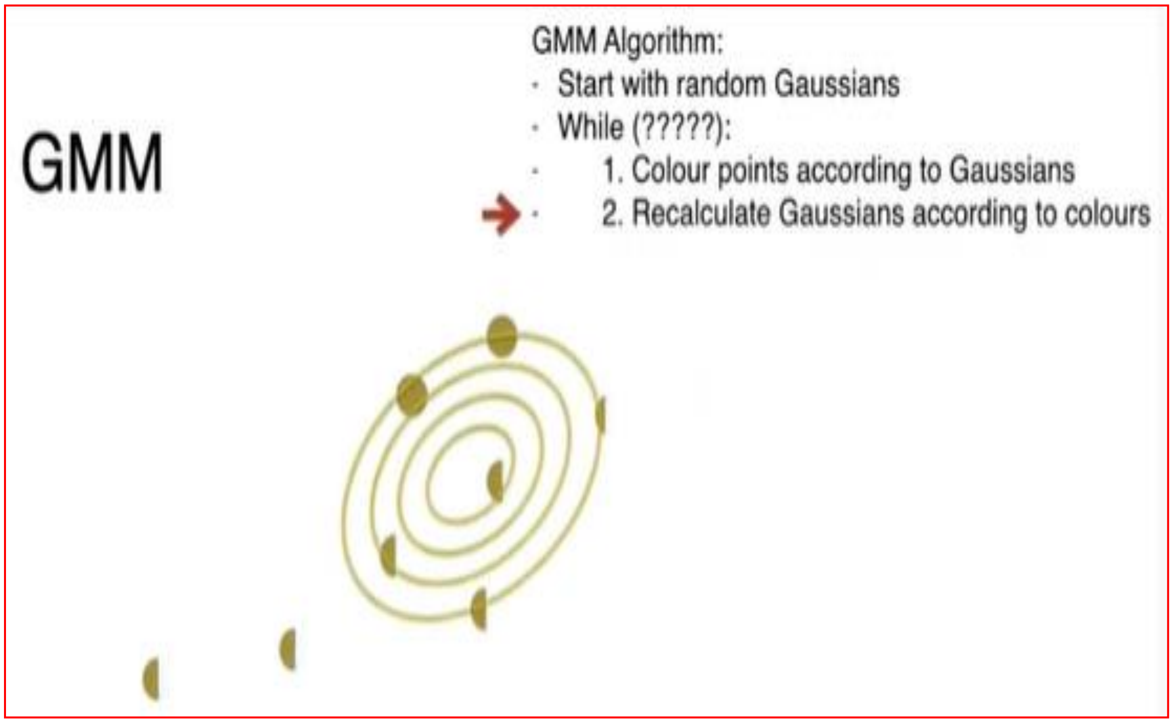
- Start with random Gaussians
- While (?????):
- 1. Colour points according to Gaussians
- 2. Recalculate Gaussians according to colours



GMM

GMM Algorithm:

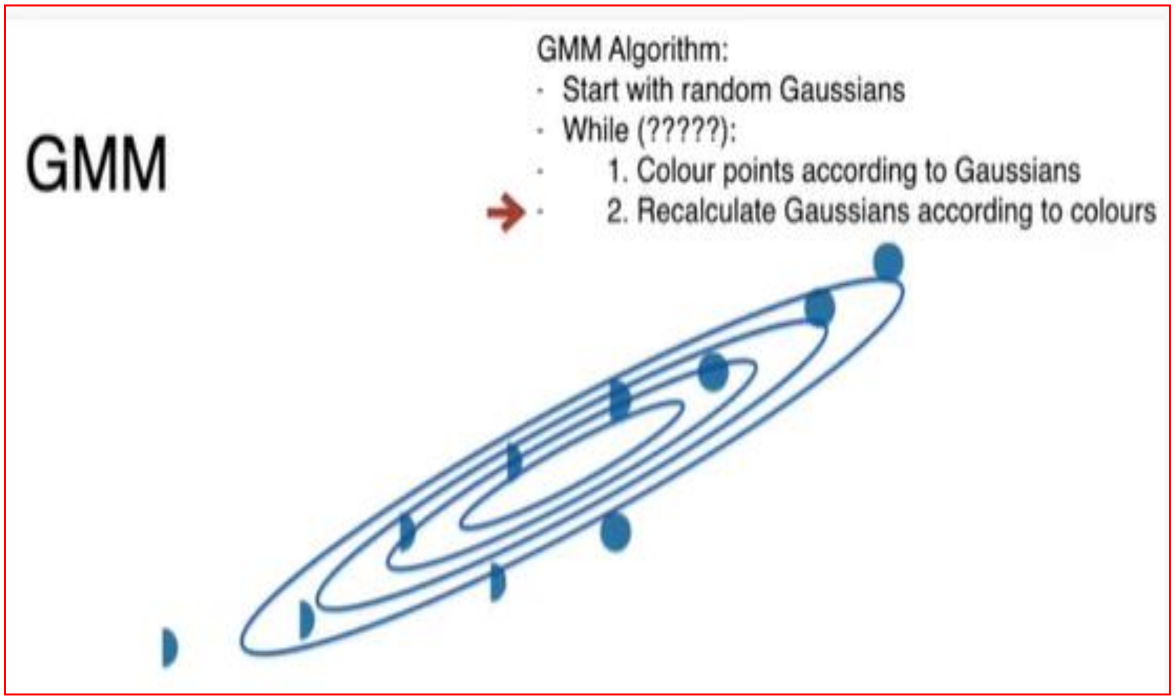
- Start with random Gaussians
- While (?????):
- 1. Colour points according to Gaussians
- 2. Recalculate Gaussians according to colours



GMM

GMM Algorithm:

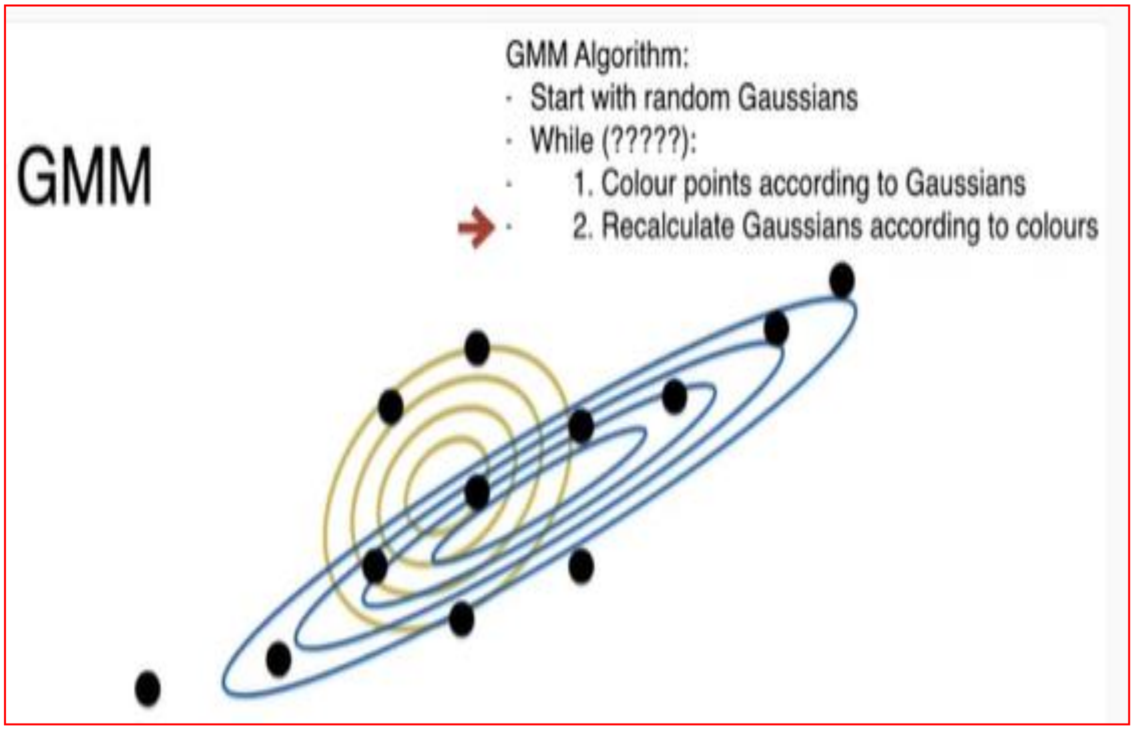
- Start with random Gaussians
- While (?????):
- 1. Colour points according to Gaussians
- 2. Recalculate Gaussians according to colours



GMM

GMM Algorithm:

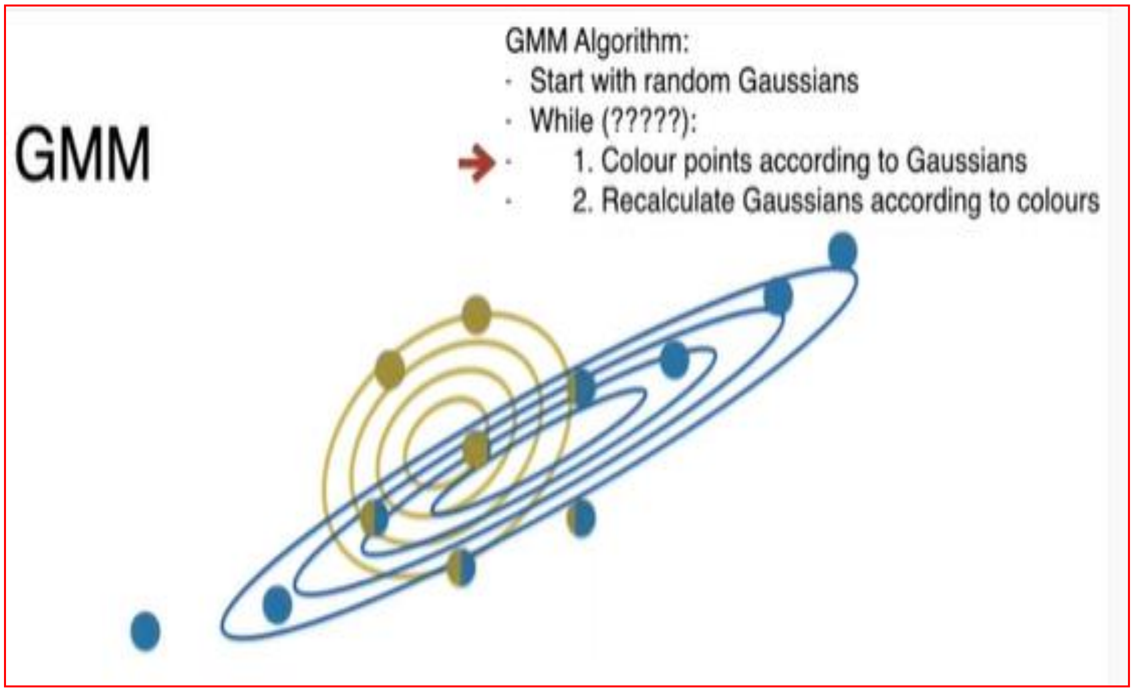
- Start with random Gaussians
- While (?????):
- 1. Colour points according to Gaussians
- 2. Recalculate Gaussians according to colours



GMM

GMM Algorithm:

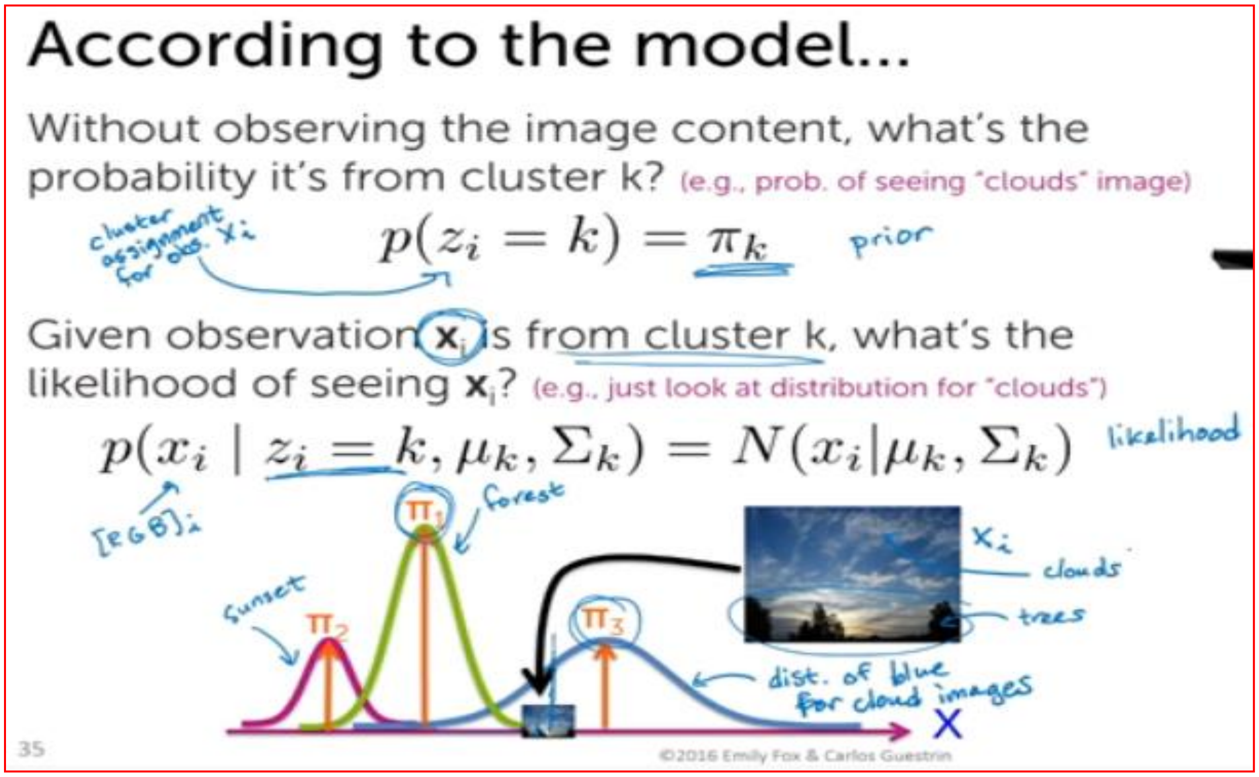
- Start with random Gaussians
- While (?????):
- 1. Colour points according to Gaussians
- ➜ 2. Recalculate Gaussians according to colours



GMM

GMM Algorithm:

- Start with random Gaussians
- While (?????):
- ➜ 1. Colour points according to Gaussians
- 2. Recalculate Gaussians according to colours



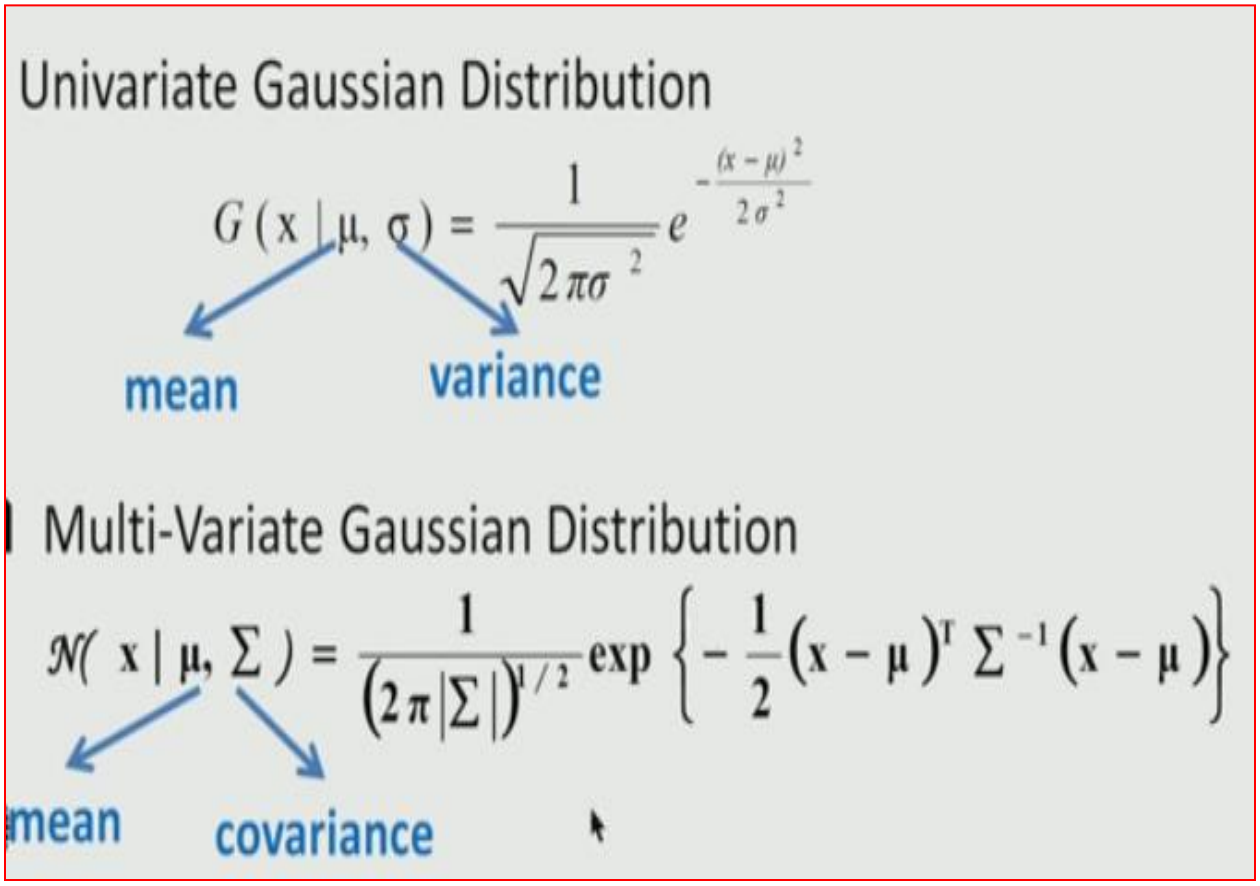
According to the model...

Without observing the image content, what's the probability it's from cluster k? (e.g., prob. of seeing "clouds" image)

$$p(z_i = k) = \underline{\pi_k}$$

Given observation $\mathbf{x}_i$ is from cluster k, what's the likelihood of seeing $\mathbf{x}_i$? (e.g., just look at distribution for "clouds")

$$p(x_i \mid \underline{z_i = k}, \mu_k, \Sigma_k) = N(x_i|\mu_k, \Sigma_k)$$



Univariate Gaussian Distribution

$$G(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

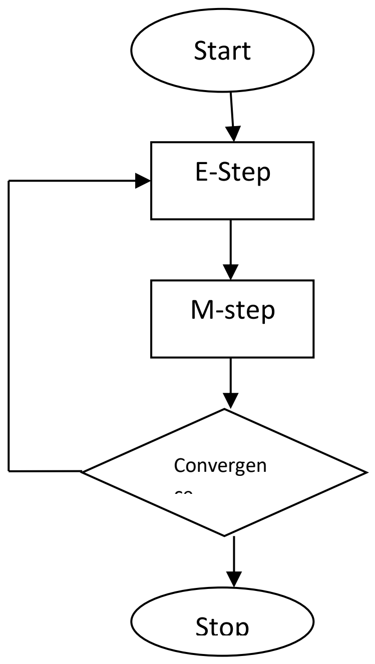
Multi-Variate Gaussian Distribution

$$\mathcal{N}(\mathbf{x} \mid \mu, \Sigma) = \frac{1}{(2\pi|\Sigma|)^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)\right\}$$

3.4.1 EM algorithm

- EM algorithm is an appropriate approach to use to estimate the parameters of the distributions. It can be used for the latent variables (Variables that are not directly observable and are actually inferred from the values of the other observed variables).
- In the EM algorithm, the estimation-step would estimate a value for the process latent variable for each data point, and the maximization step would optimize the parameters of the probability distributions in an attempt to best capture the density of the data.
- In the real world applications of machine learning, it is very common that there are many relevant features available for learning but only a small subset of them are observable.
- The process is repeated until a good set of latent values and a maximum likelihood is achieved that fits the data.
- **E-Step**. Estimate the expected value for each latent variable.
- **M-Step**. Optimize the parameters of the distribution using maximum likelihood.
- Initially**, a set of initial values of the parameters are considered**. A set of incomplete observed data is given to the system with the assumption that the observed data comes from a specific model.
- The next step is known as “Expectation”- step or E-step. In this step**, the observed data in order to estimate or guess the values of the missing or incomplete data.** It is basically used to update the variables.
- The next step if known as “Maximization”- step or M-step. In this step, the **complete data generated in the preceding “Expectation”- step in order to update the values of the parameters.** It is basically used to update the hypothesis.
- In the next step, it is checked whether the values are converging or not, if yes then stop other **wise repeat step 2 and step3** ie., Expectation- step and Maximization step until the convergence occurs.

How does it work?



Advantages

- It is guaranteed that the likelihood will increase with each iteration
- During implementation, the E-step and M-step are very easy for many problems
- The solution for M-step often exists in closed form

Disadvantages

- EM algorithm has a very slow convergence
- It makes the convergence to the local optima only
- EM requires both forward and backward possibilities

Uses

- It can be used to fill the missing data in a sample
- It can be used as the basis of unsupervised learning of clusters
- It can be used for the purpose of estimating the parameters of Hidden Markov Model(HMM)

- It can be used for discovering the values of latent variables

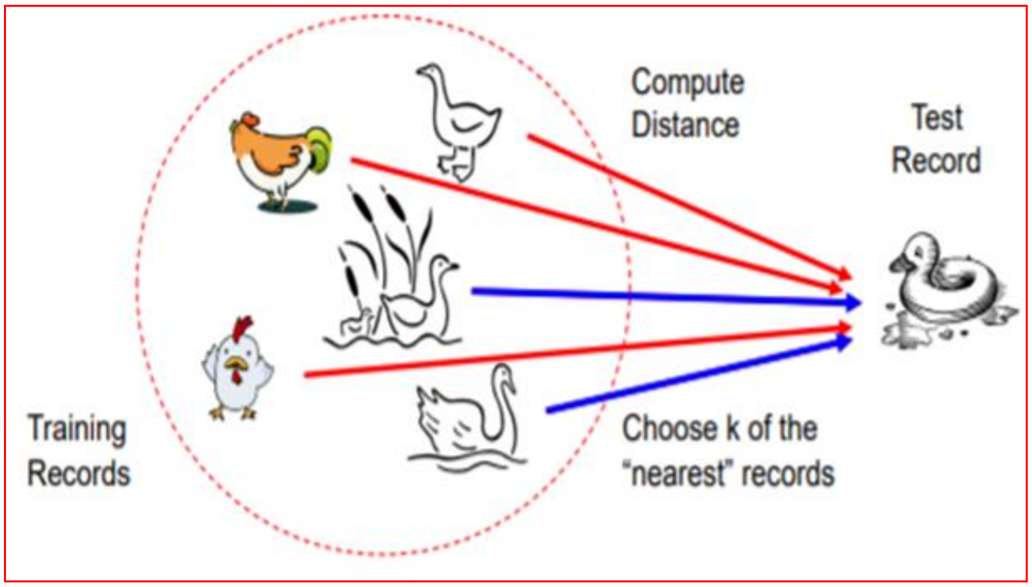
3.5 Nearest Neighbor Methods-K Nearest Neighbor

Simple Analogy..

- Tell me about your friends (*who your neighbors are*) and *I will tell you who you are*.

Nearest Neighbor Classifiers

-Basic idea- IF it walk like a duck, quacks like a duck, then it’s probably duck



KNN – Different names

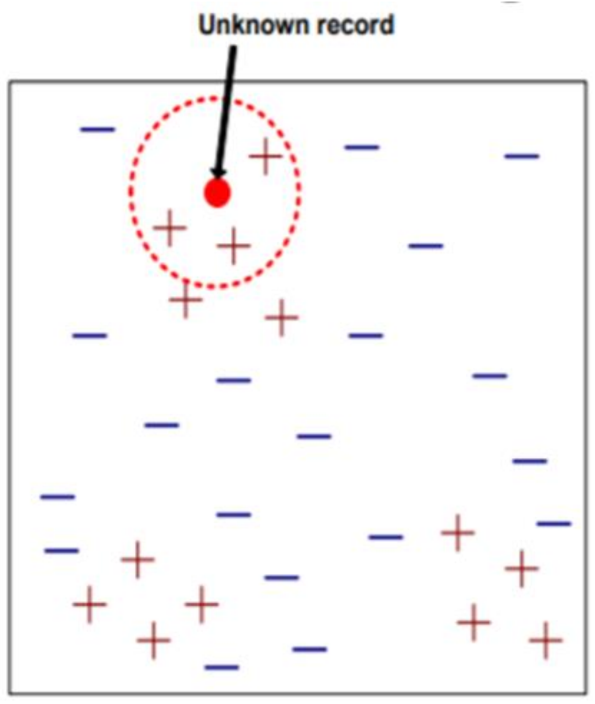
- K-Nearest Neighbors
- Memory-Based Reasoning
- Example-Based Reasoning
- Instance-Based Learning
- Lazy Learning

Requires three things

- The set of stored records
- Distance Metric to compute distance between records
- The value of k, the number of nearest neighbors to retrieve

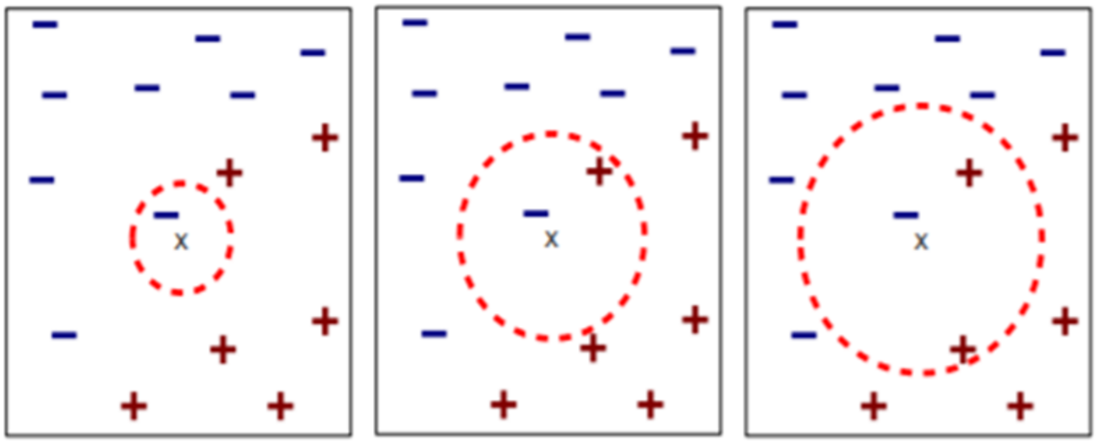
To classify an unknown record

- Compute distance to other training records
- Identify k nearest neighbors
- Use class labels of nearest neighbors to determine the class label of unknown record(e.g., by taking majority vote)

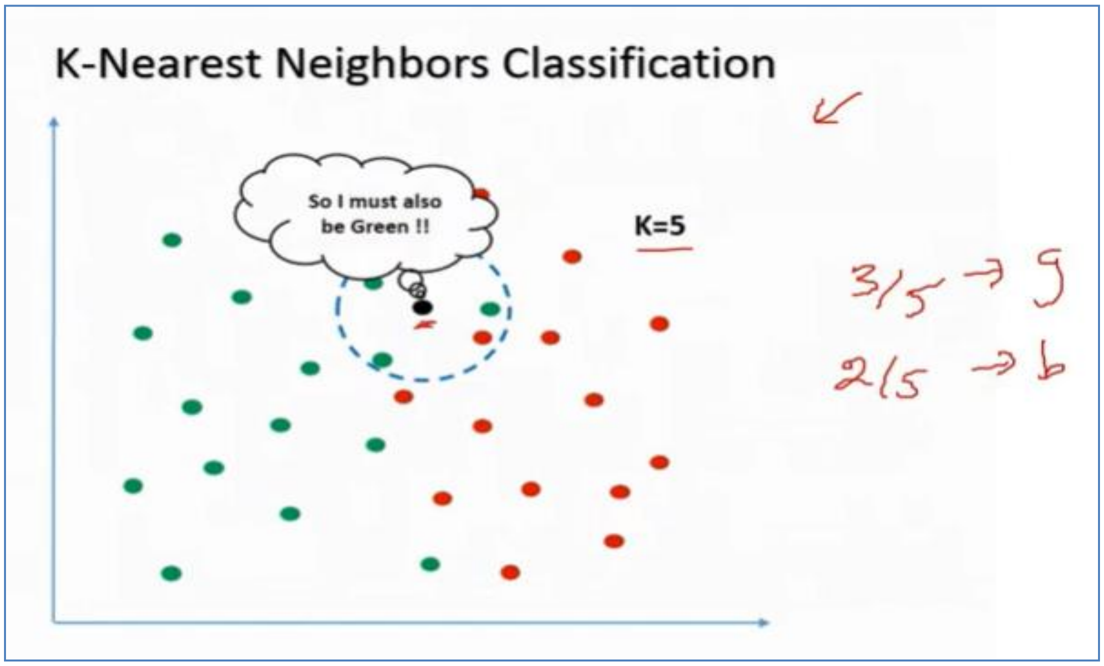


Definition of Nearest Neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

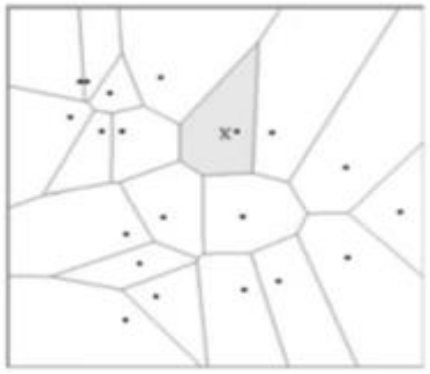


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

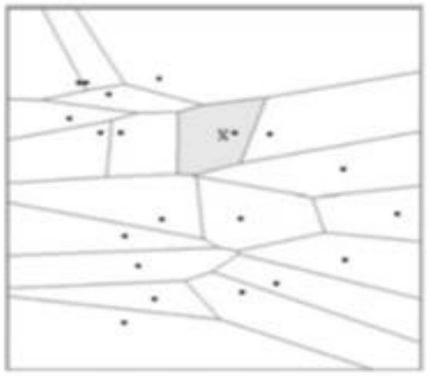

K-Nearest Neighbors Classification
So I must also be Green !!
K=5
3/5 → g
2/5 → b

Distance Metrics

- Different metrics can change the decision surface



Dist(**a**,**b**) =$(a_1 - b_1)^2 + (a_2 - b_2)^2$



Dist(**a**,**b**) =$(a_1 - b_1)^2 + (3a_2 - 3b_2)^2$

- Standard Euclidean distance metric:
  - Two-dimensional: Dist(a,b) = sqrt($(a_1 - b_1)^2 + (a_2 - b_2)^2$)
  - Multivariate: Dist(a,b) = sqrt($\sum (a_i - b_i)^2$)

Nearest Neighbor Classification

- Compute distance between two points:
  - Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

  - Manhatten distance

$$d(p,q) = \sum_i |p_i - q_i|$$

  - q norm distance

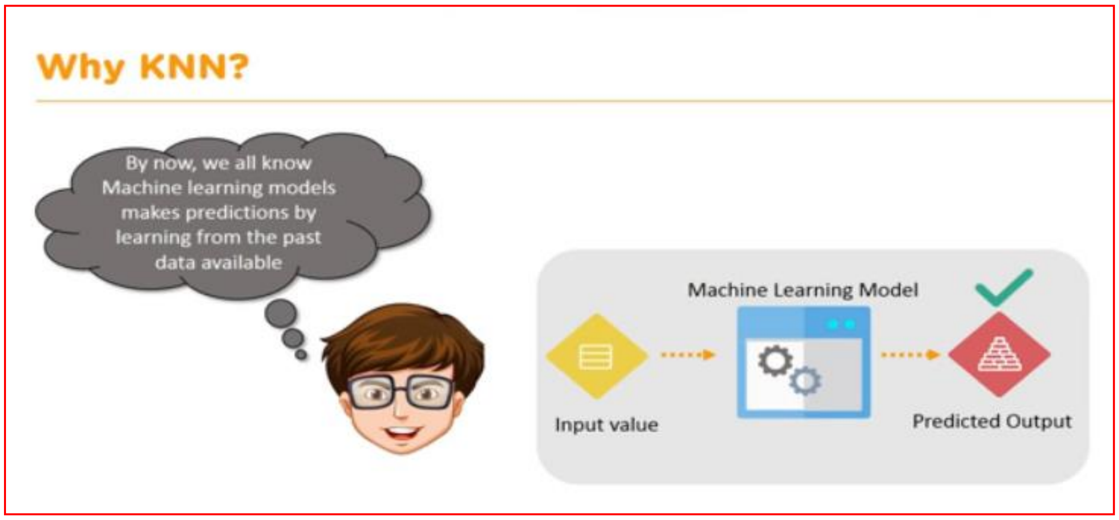
$$d(p,q) = (\sum_i |p_i - q_i|^q)^{1/q}$$

Choosing k – increases k reduce variance, increases bias

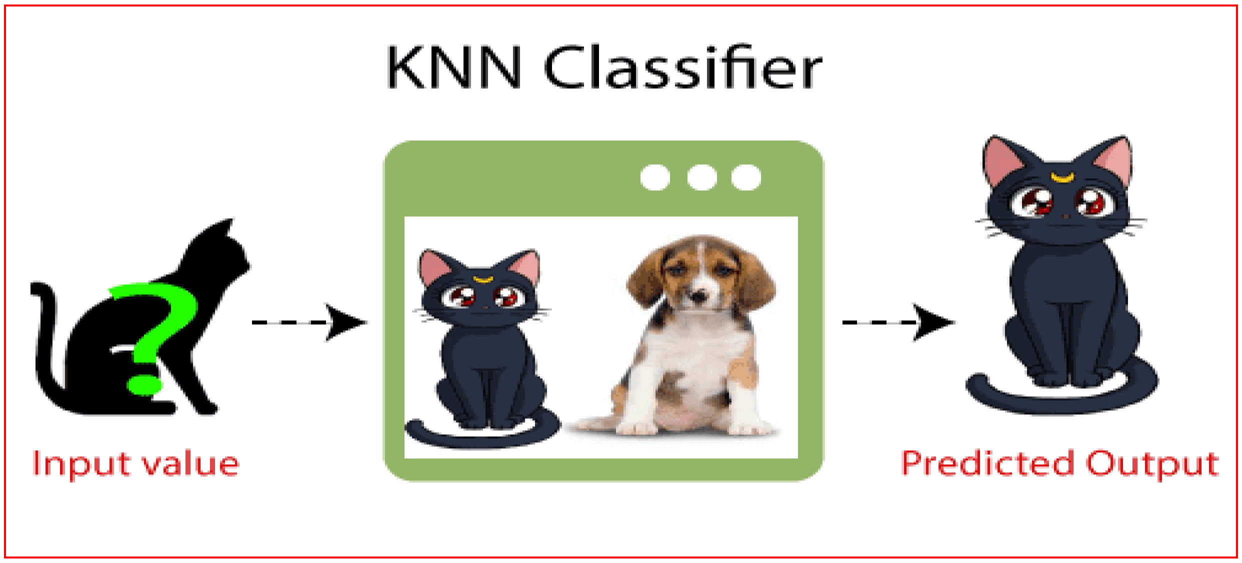
NN smoothing

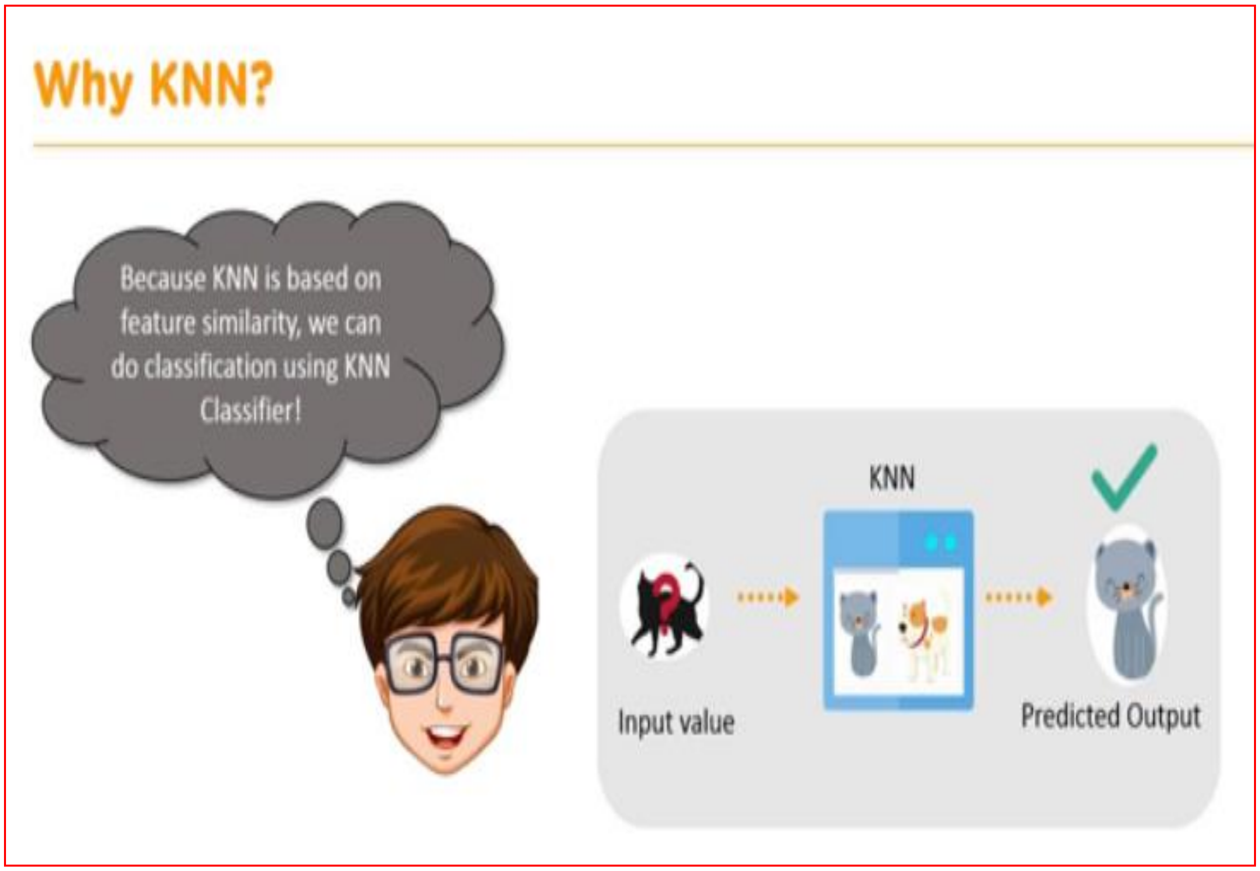
- Nearest neighbour methods can also be used for **regression by returning the average value of the neighbours to a poin**t, or a spline or similar fit as the new value.
- The most common methods are known as kernel smoothers, and they use a kernel (a **weighting function** between pairs of points) that decides how much emphasis(weight) to put on to the contribution from each data point according to its distance from the input.

Both of these kernels are designed to give more weight to points that are closer to the current input, with the weights decreasing smoothly to zero as they pass out of the range of the current input, with the range specified by a parameter λ.


Why KNN?
By now, we all know Machine learning models makes predictions by learning from the past data available
Machine Learning Model
Input value
Predicted Output

CATS
DOGS
Sharp Claws, uses to climb
Smaller length of ears
Meows and purrs
Doesn't love to play around
Dull Claws
Bigger length of ears
Barks
Loves to run around

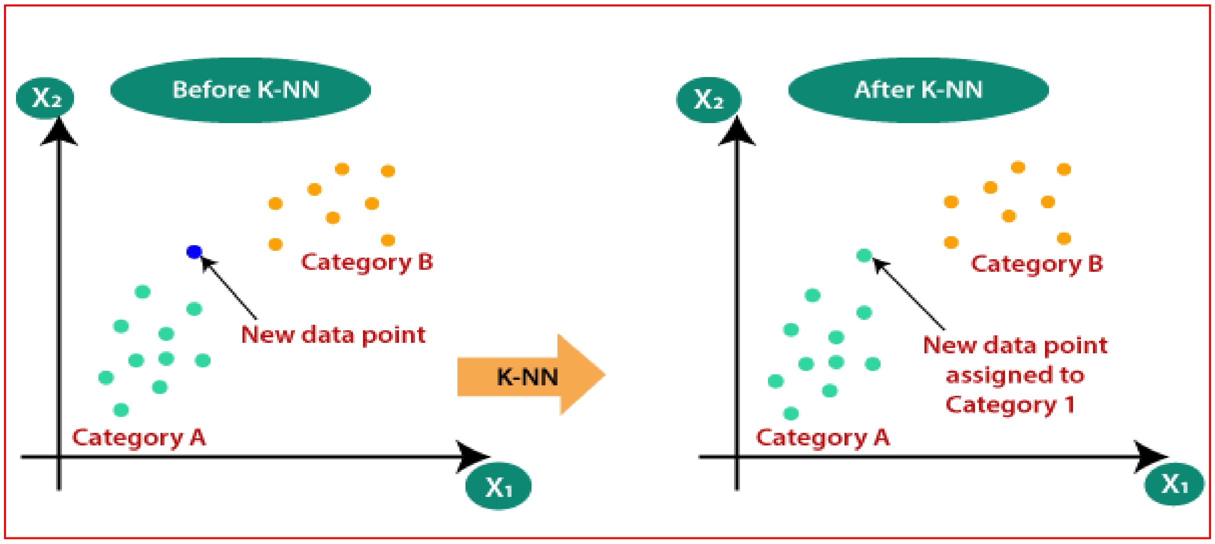



Why KNN?
Because KNN is based on feature similarity, we can do classification using KNN Classifier!
KNN
Input value
Predicted Output

Why do we need a K-NN Algorithm?

- Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories.

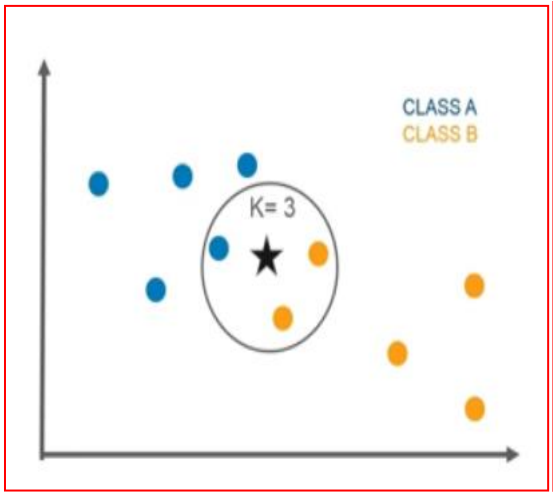
- To solve this type of problem, need a K-NN algorithm. With the help of K-NN, can easily identify the category or class of a particular dataset. Consider the below diagram:

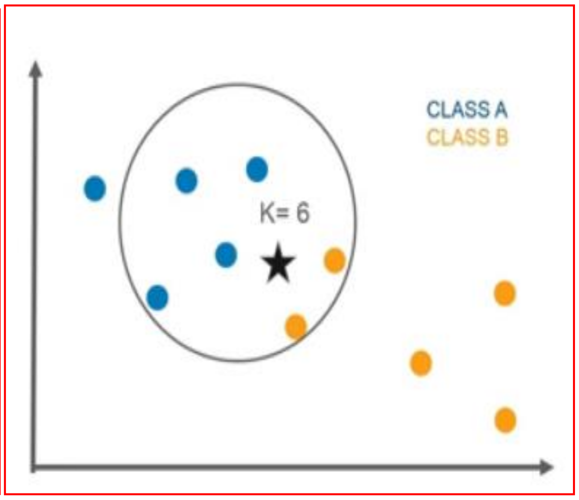


kNN algorithm is one of the simplest supervised machine learning algorithm mostly used for Classification- It classifies a data point based on how its neighbors are classified.

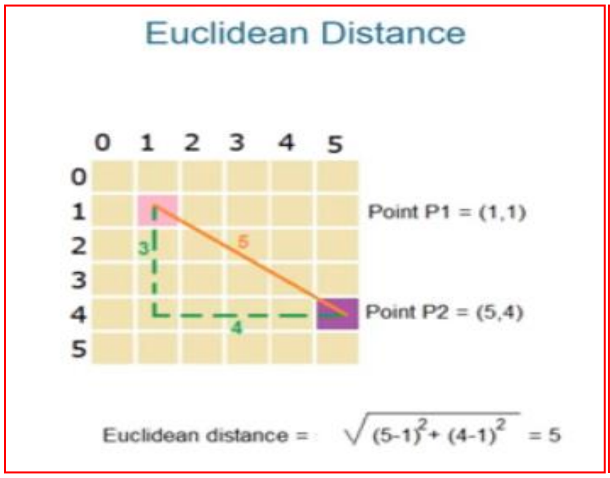
- K nearest neighbour is a simple algorithm that stores all the available cases and classifies the new data or case based on a similarity measure.
- Ie. Similar things are near to each other
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

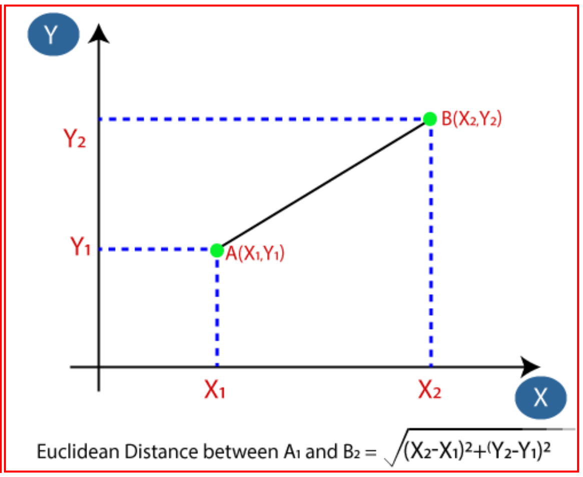
Examples:



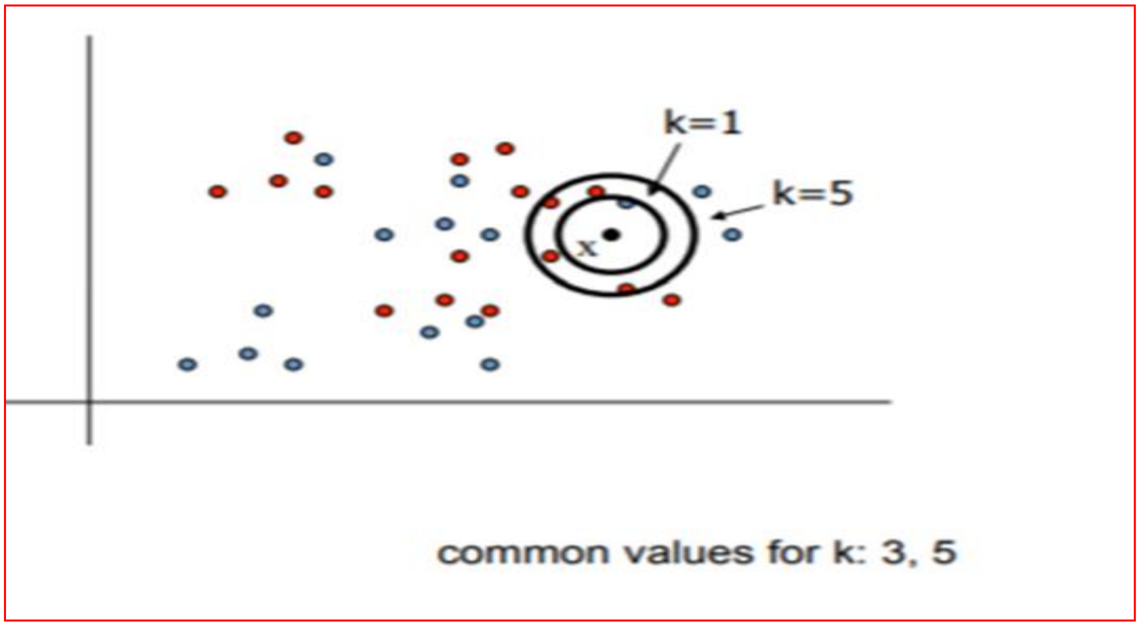


How to choose the value of k in kNN algorithm? By using Euclidean distance




Y
X
Y_2
Y_1
X_1
X_2
$B(X_2,Y_2)$
$A(X_1,Y_1)$
Euclidean Distance between A_1 and B_2 = $\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

To classify a new input vector x, examine the k-closest training data points to x and assign the object to the most frequently occurring class

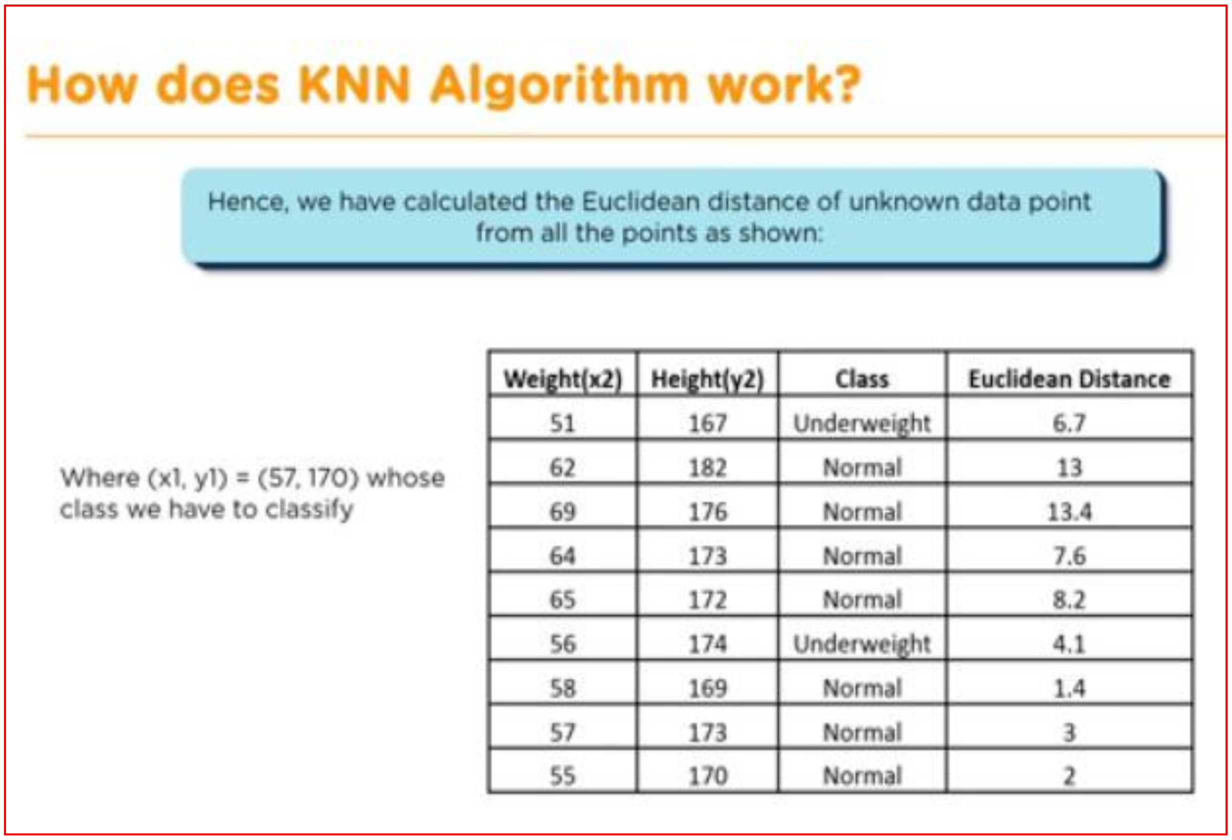


kNN steps

- Handle Data: Open the dataset and split into test/train datasets
- Similarity: Calculate the distance two data instances
- Neighbors: Locate k most similar data instance
- Response: Generate a response from a set of data instance
- Accuracy: Summarizing the accuracy of predictions
- Tie it all together

How does K-NN work?

- The K-NN working can be explained on the basis of the below algorithm:
- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

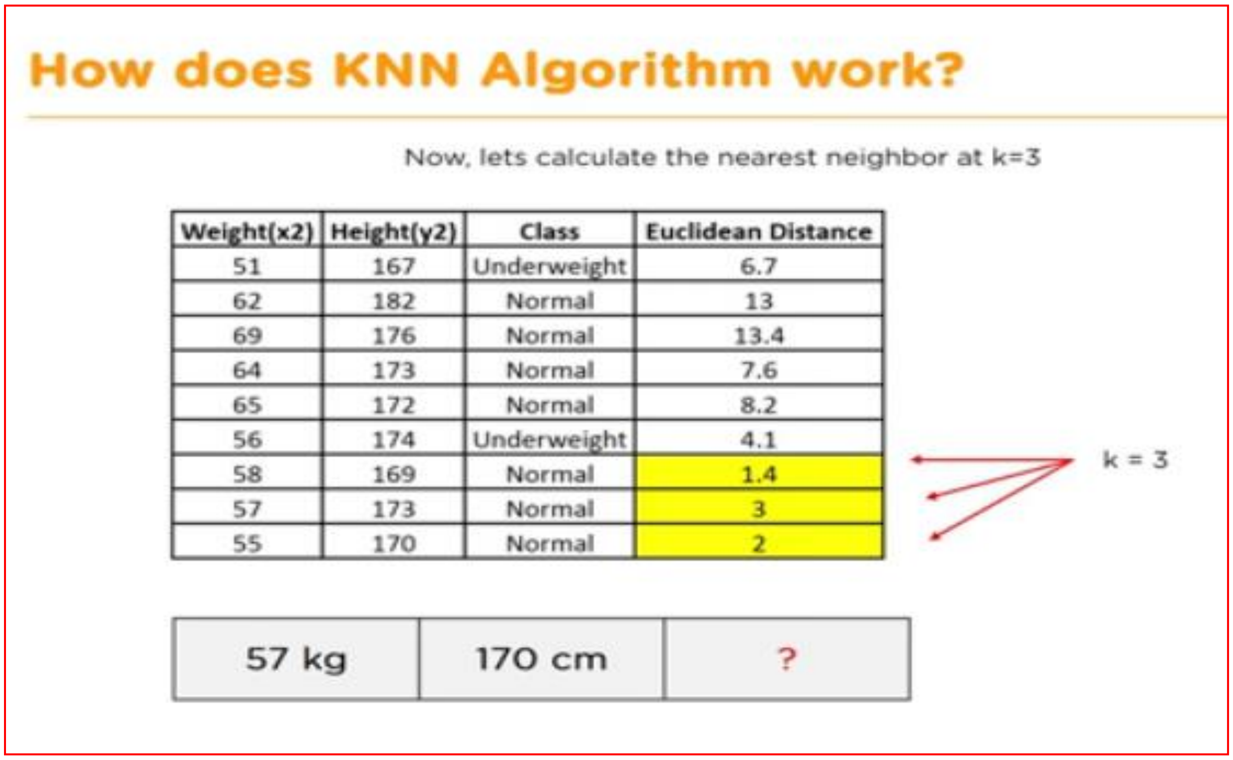


How does KNN Algorithm work?

Hence, we have calculated the Euclidean distance of unknown data point from all the points as shown:

Where (x1, y1) = (57, 170) whose class we have to classify

Weight(x2)	Height(y2)	Class	Euclidean Distance
51	167	Underweight	6.7
62	182	Normal	13
69	176	Normal	13.4
64	173	Normal	7.6
65	172	Normal	8.2
56	174	Underweight	4.1
58	169	Normal	1.4
57	173	Normal	3
55	170	Normal	2



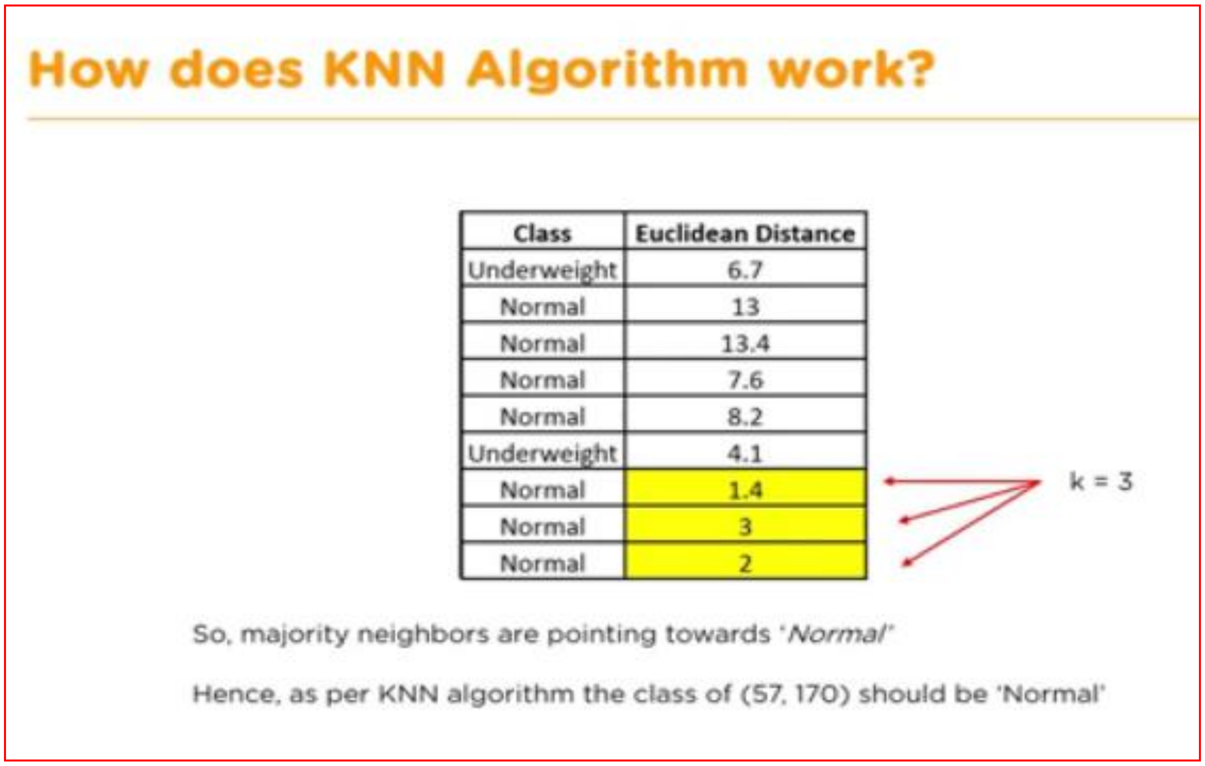
How does KNN Algorithm work?

Now, lets calculate the nearest neighbor at k=3

Weight(x2)	Height(y2)	Class	Euclidean Distance
51	167	Underweight	6.7
62	182	Normal	13
69	176	Normal	13.4
64	173	Normal	7.6
65	172	Normal	8.2
56	174	Underweight	4.1
58	169	Normal	1.4
57	173	Normal	3
55	170	Normal	2

k = 3

57 kg	170 cm	?



How does KNN Algorithm work?

Class	Euclidean Distance
Underweight	6.7
Normal	13
Normal	13.4
Normal	7.6
Normal	8.2
Underweight	4.1
Normal	1.4
Normal	3
Normal	2

So, majority neighbors are pointing towards '*Normal*'

Hence, as per KNN algorithm the class of (57, 170) should be 'Normal'

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm:

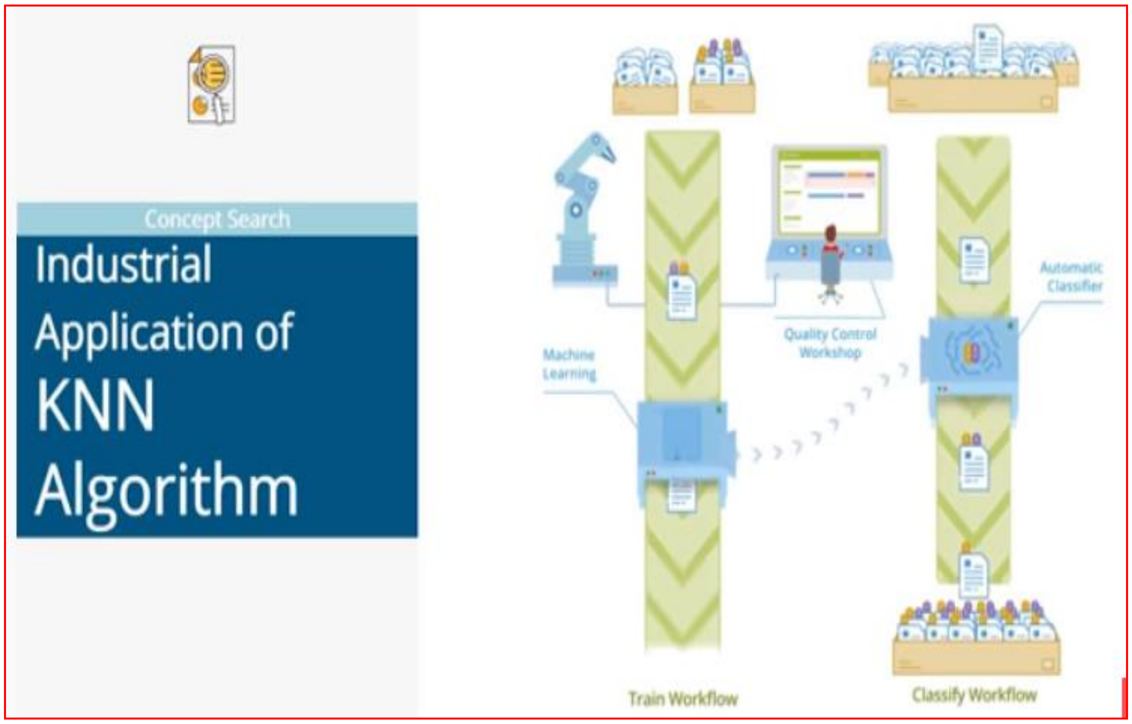
- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

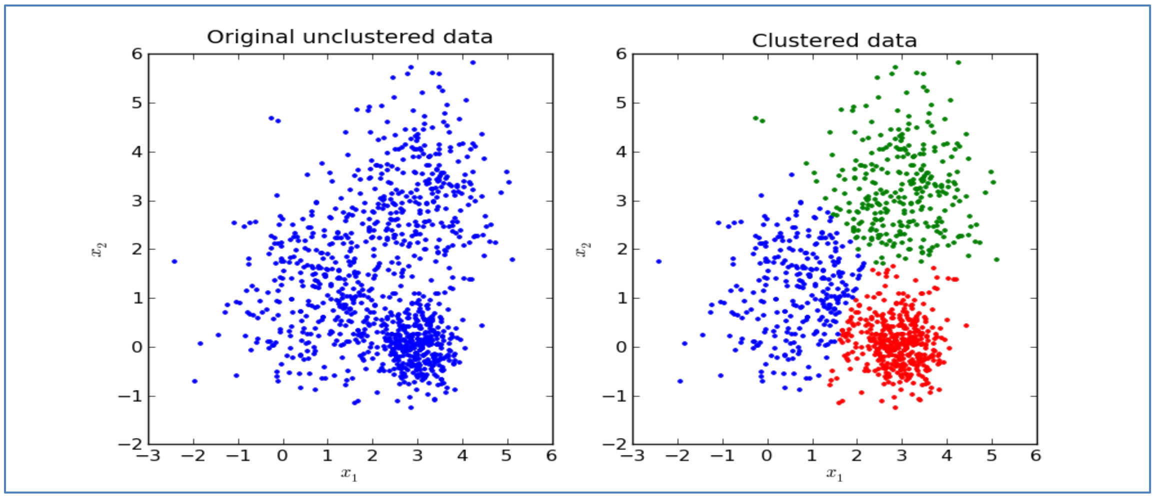
Real time example

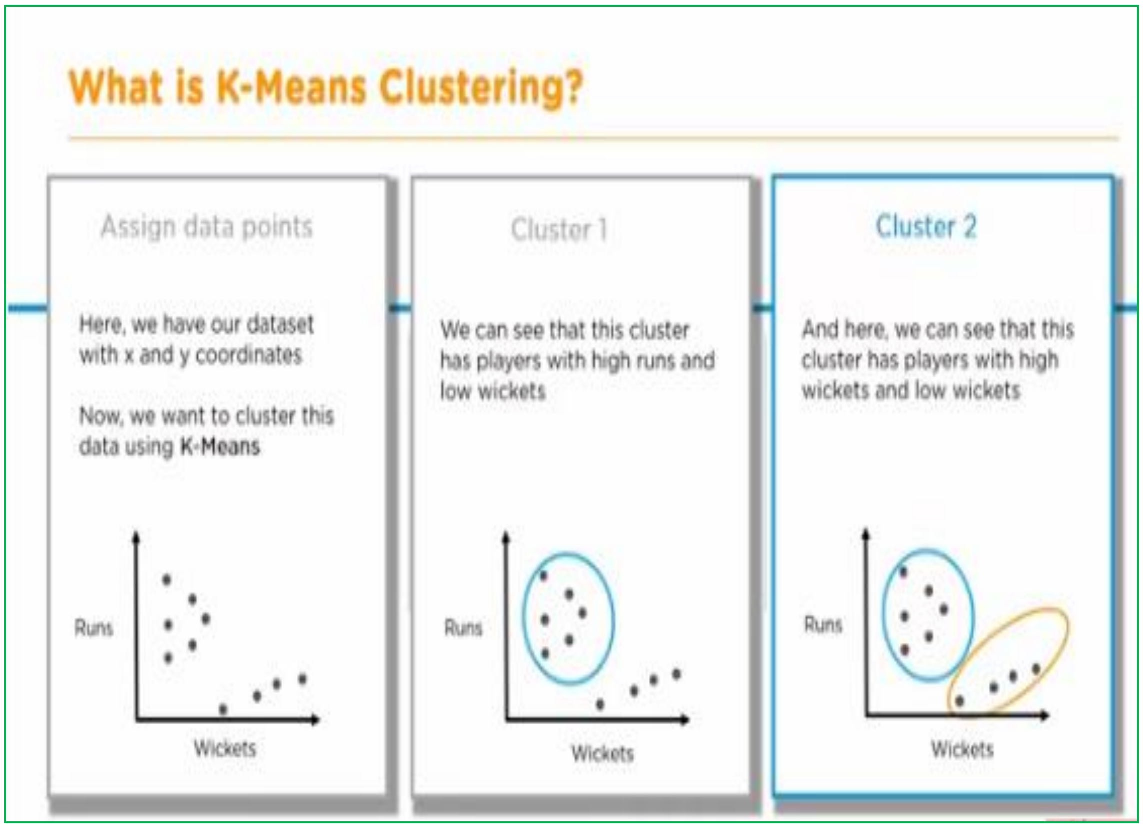
amazon
Recommender System
Industrial Application of KNN Algorithm
Apple iPhone 8 (Space Grey, 64GB)
Customers who bought this item also bought



3.6 Unsupervised Learning -K means Algorithms

k-means clustering aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest means. It works for n-dimensional spaces





What is K-Means Clustering?

Assign data points

Here, we have our dataset with x and y coordinates

Now, we want to cluster this data using **K-Means**

Cluster 1

We can see that this cluster has players with high runs and low wickets

Cluster 2

And here, we can see that this cluster has players with high wickets and low wickets

Input: k-number of clusters, D list of data points

1. Choose k number of random data points as initial centroids(cluster centers)

2. Repeat till cluster centers stabilize:

    Allocate each point in D to the nearest of Kth centroids.

    Compute centroid for the cluster using all points in the cluster.

K-Means algorithm for clustering

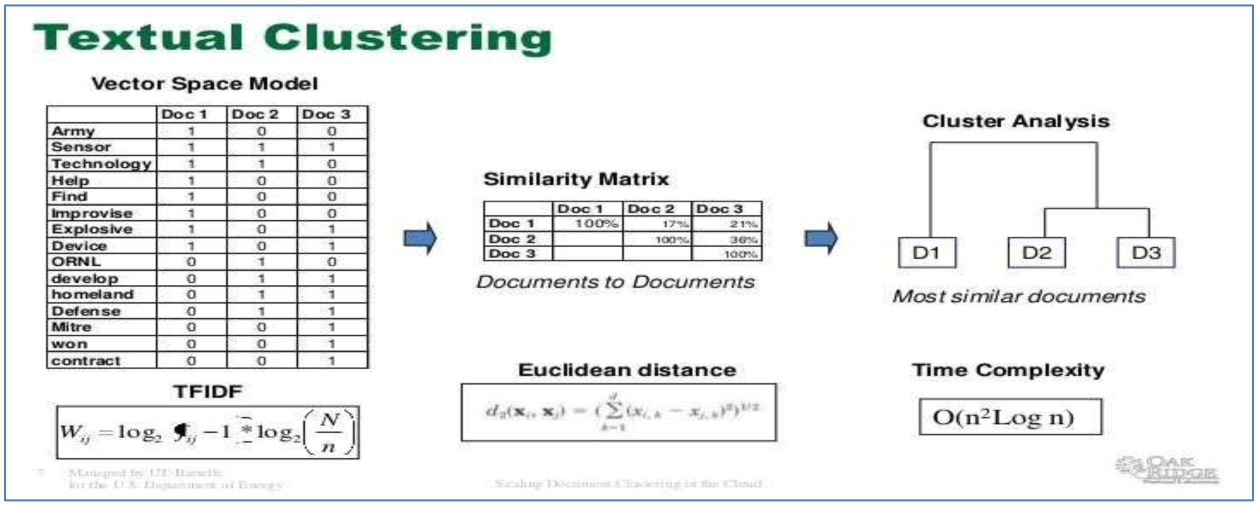
- K means algorithm is an unsupervised learning algorithm
- Given a data set of items, with certain features and values for these features ,the algorithm will categorize the items into k groups or clusters of similarity
- To calculate the similarity, we can use the Euclidean distance, Manhattan distance, Hamming distance, Cosine distance as measurement.
- K-means is a clustering algorithm whose goal is to group similar elements or data points into a cluster.
- It performs division of objects into clusters which are similar between them and are dissimilar to the objects belonging to another cluster
- K- means number of clusters

Eg., Identify bowlers (wickets)and batsman(higher runs)

- Clustering is the process of diving the datasets into groups, consisting of similar data points
- Points in the same group are as similar as possible.
- Points in different group are as dissimilar as possible

Example: Web document search

- A web search engine often returns thousands of pages in response to a broad query, making it difficult for users to browse or to identify relevant information.
- Clustering methods can be used to automatically group the retrieved documents into a list of meaningful categories.



Textual Clustering

Vector Space Model

	Doc 1	Doc 2	Doc 3
Army	1	0	0
Sensor	1	1	1
Technology	1	1	0
Help	1	0	0
Find	1	0	0
Improvise	1	0	0
Explosive	1	0	1
Device	1	0	1
ORNL	0	1	0
develop	0	1	1
homeland	0	1	1
Defense	0	1	1
Mitre	0	0	1
won	0	0	1
contract	0	0	1

TFIDF

$$\boxed{W_{ij} = \log_2 (f_{ij} - 1) * \log_2\left(\frac{N}{n}\right)}$$

Similarity Matrix

	Doc 1	Doc 2	Doc 3
Doc 1	100%	17%	21%
Doc 2		100%	36%
Doc 3			100%

Documents to Documents

Euclidean distance

$$\boxed{d_2(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2\right)^{1/2}}$$

Cluster Analysis

D1 D2 D3

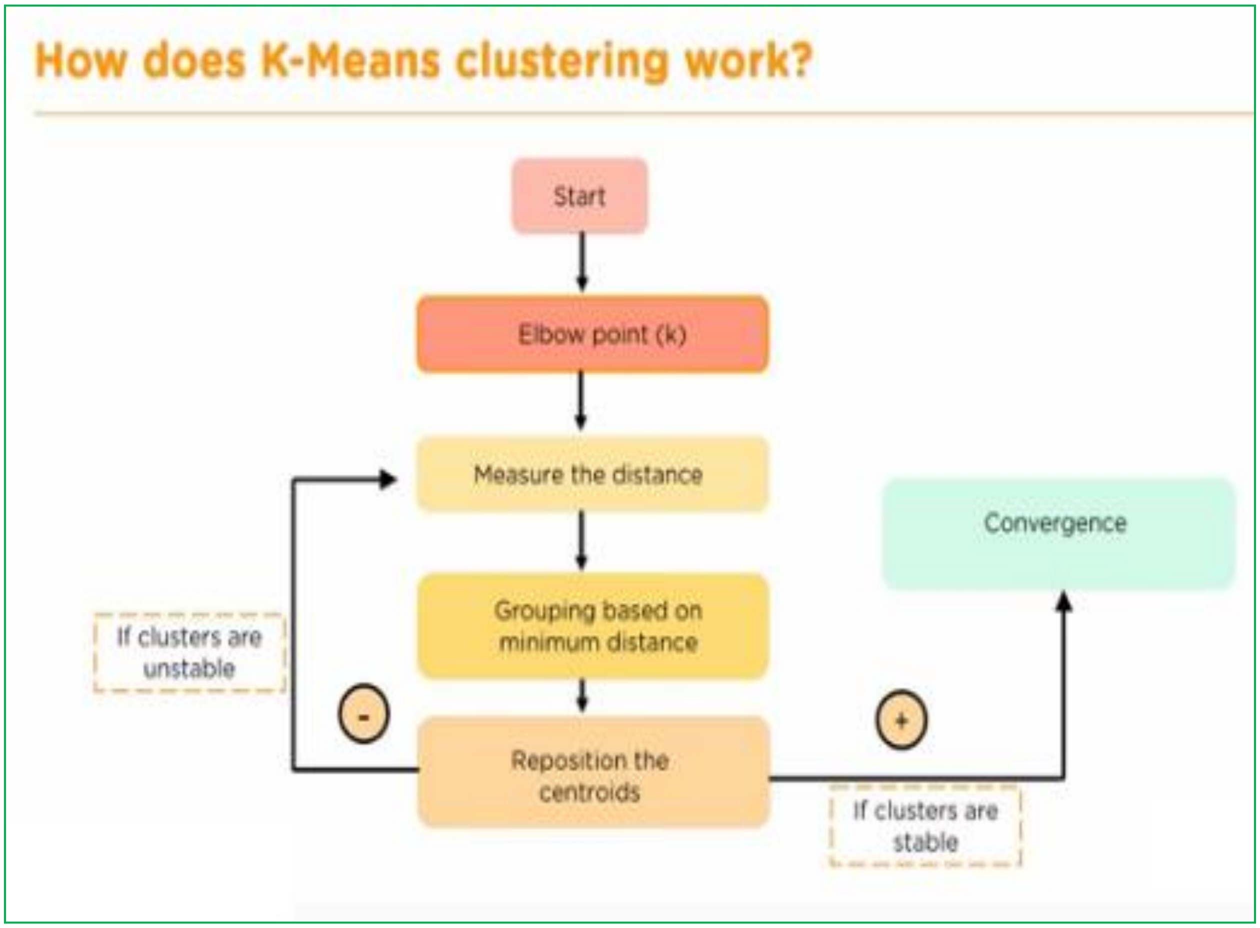
Most similar documents

Time Complexity

$$\boxed{O(n^2 \text{Log } n)}$$

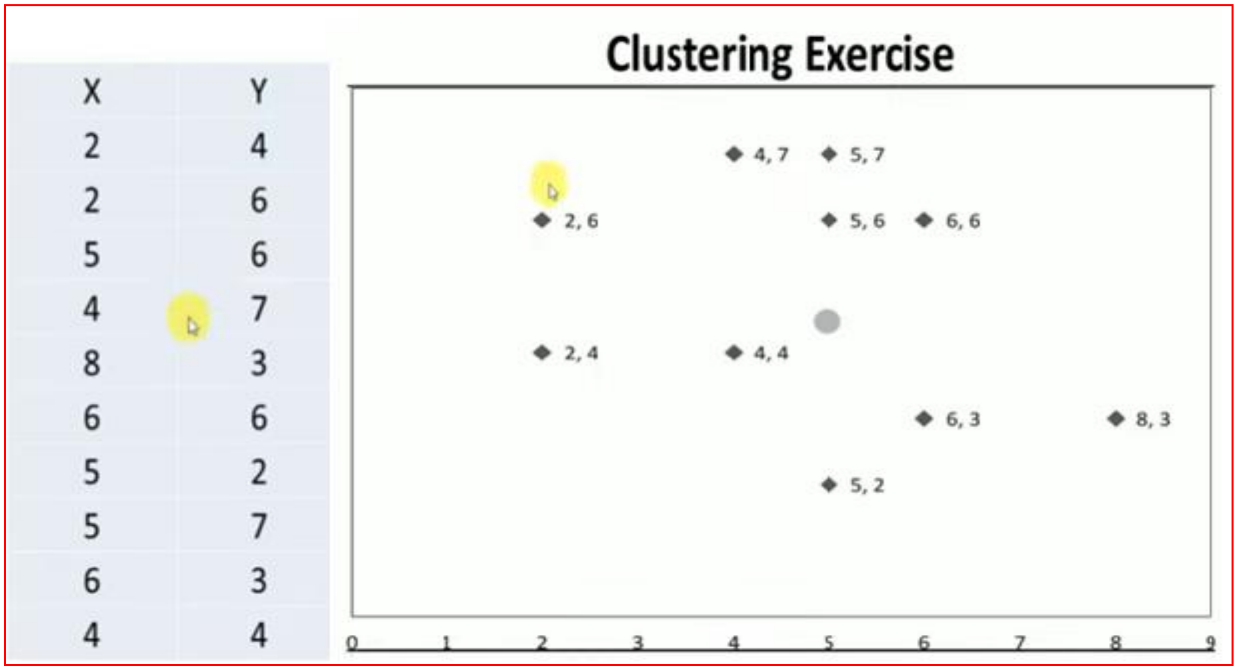
Managed by UT-Battelle for the U.S. Department of Energy

Scaling Document Clustering in the Cloud

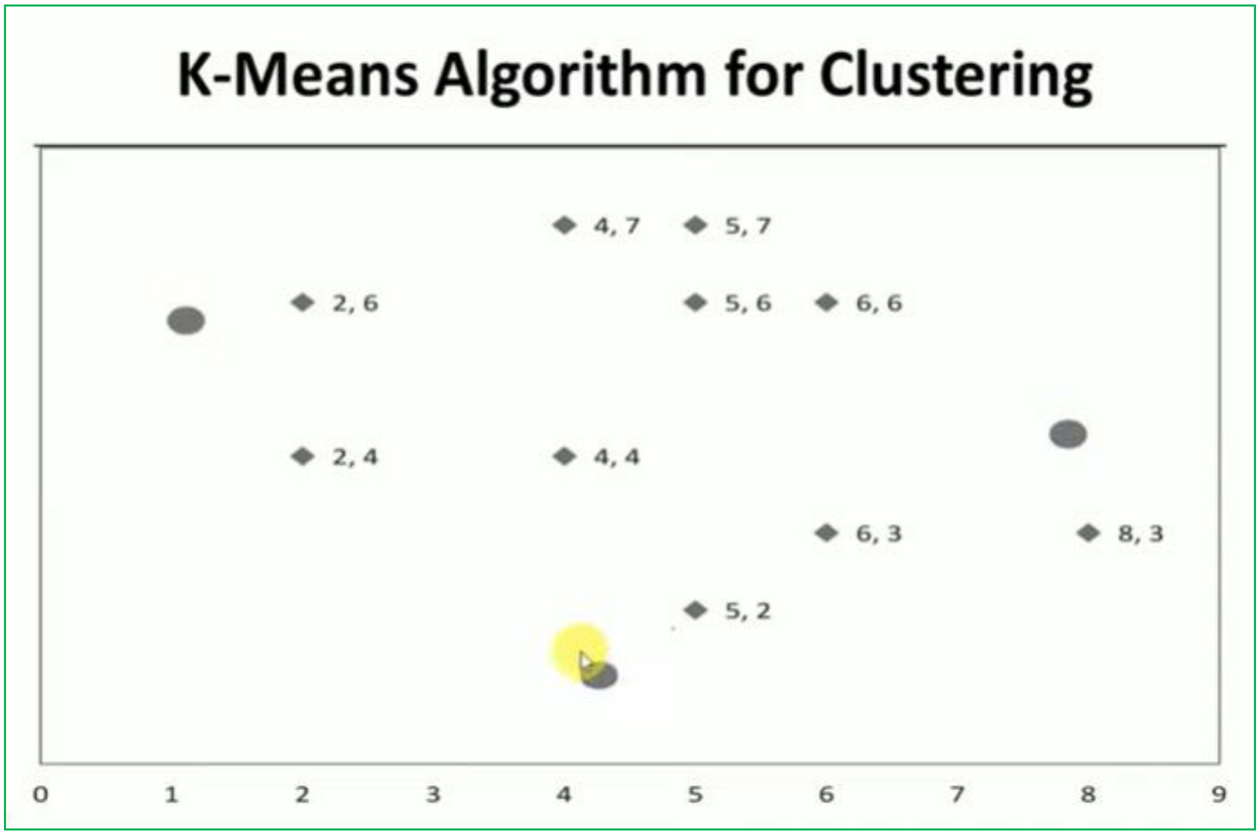


How does K-Means clustering work?

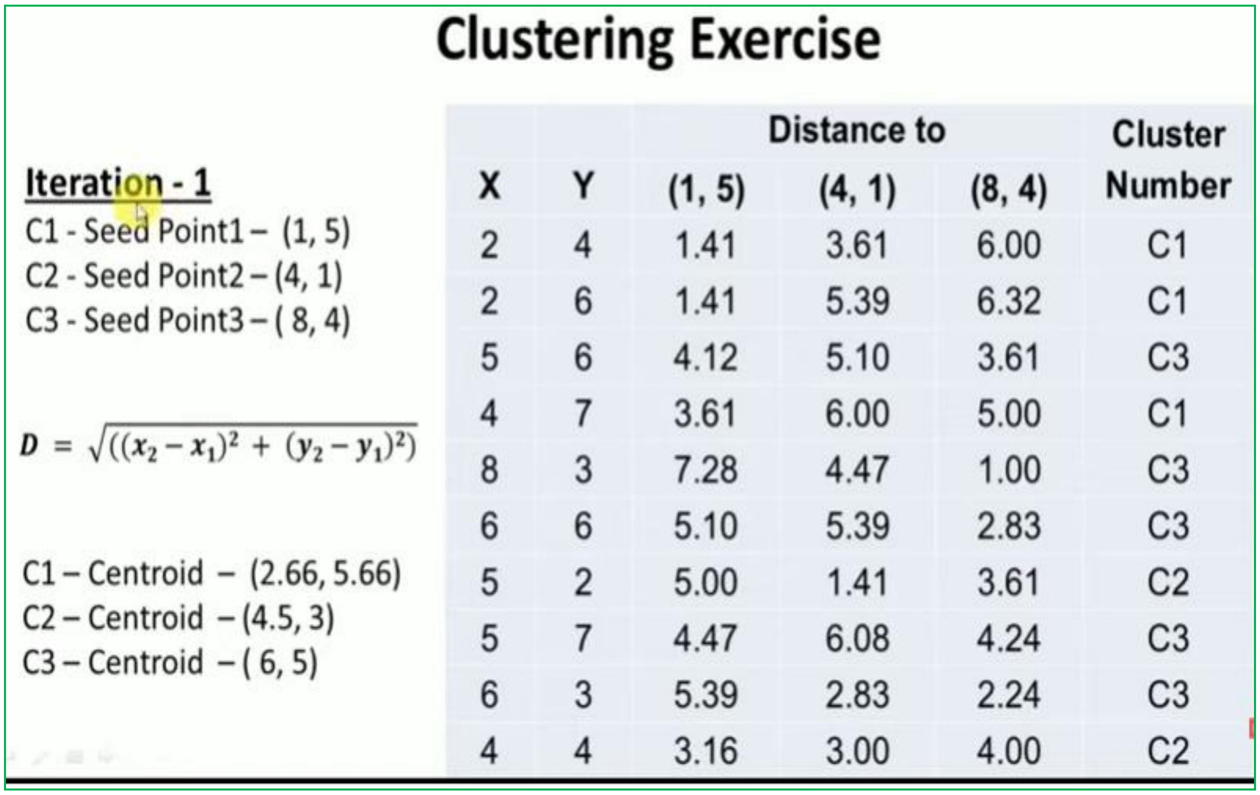
- **Applications of K-means clustering**
  - Academic performance
  - Diagnostic systems
  - Search engines
  - Wireless Sensor Networks
- **Strengths**
  - Simple iterative method
  - User provides “K”
  - No other clustering algorithm performs better than k-means
  - It is alos efficient, in which the time taken to cluster k-means rises linearly with the number of data points.
- **Weaknesses**
  - The user needs to specify an initial value of K
  - Difficult to guess the correct “K”
  - The process of findings the clusters may not converge


Clustering Exercise

X	Y
2	4
2	6
5	6
4	7
8	3
6	6
5	2
5	7
6	3
4	4



Randomly pick C1,C2 and C3 cluster centers are (1,5), (4,1) and (8,4)



Clustering Exercise

Iteration - 1

C1 - Seed Point1 – (1, 5)
C2 - Seed Point2 – (4, 1)
C3 - Seed Point3 – (8, 4)

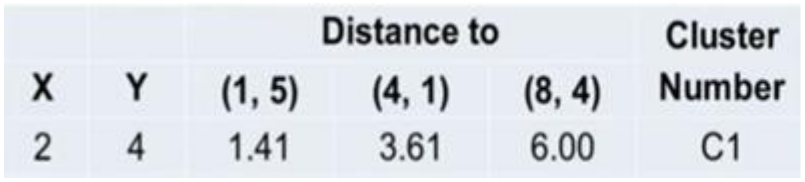
$$D = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)}$$

C1 – Centroid – (2.66, 5.66)
C2 – Centroid – (4.5, 3)
C3 – Centroid – (6, 5)

		Distance to			Cluster
X	Y	(1, 5)	(4, 1)	(8, 4)	Number
2	4	1.41	3.61	6.00	C1
2	6	1.41	5.39	6.32	C1
5	6	4.12	5.10	3.61	C3
4	7	3.61	6.00	5.00	C1
8	3	7.28	4.47	1.00	C3
6	6	5.10	5.39	2.83	C3
5	2	5.00	1.41	3.61	C2
5	7	4.47	6.08	4.24	C3
6	3	5.39	2.83	2.24	C3
4	4	3.16	3.00	4.00	C2

Iteration 1 – In row 1 , assign each data point to closest Cluster.

For example



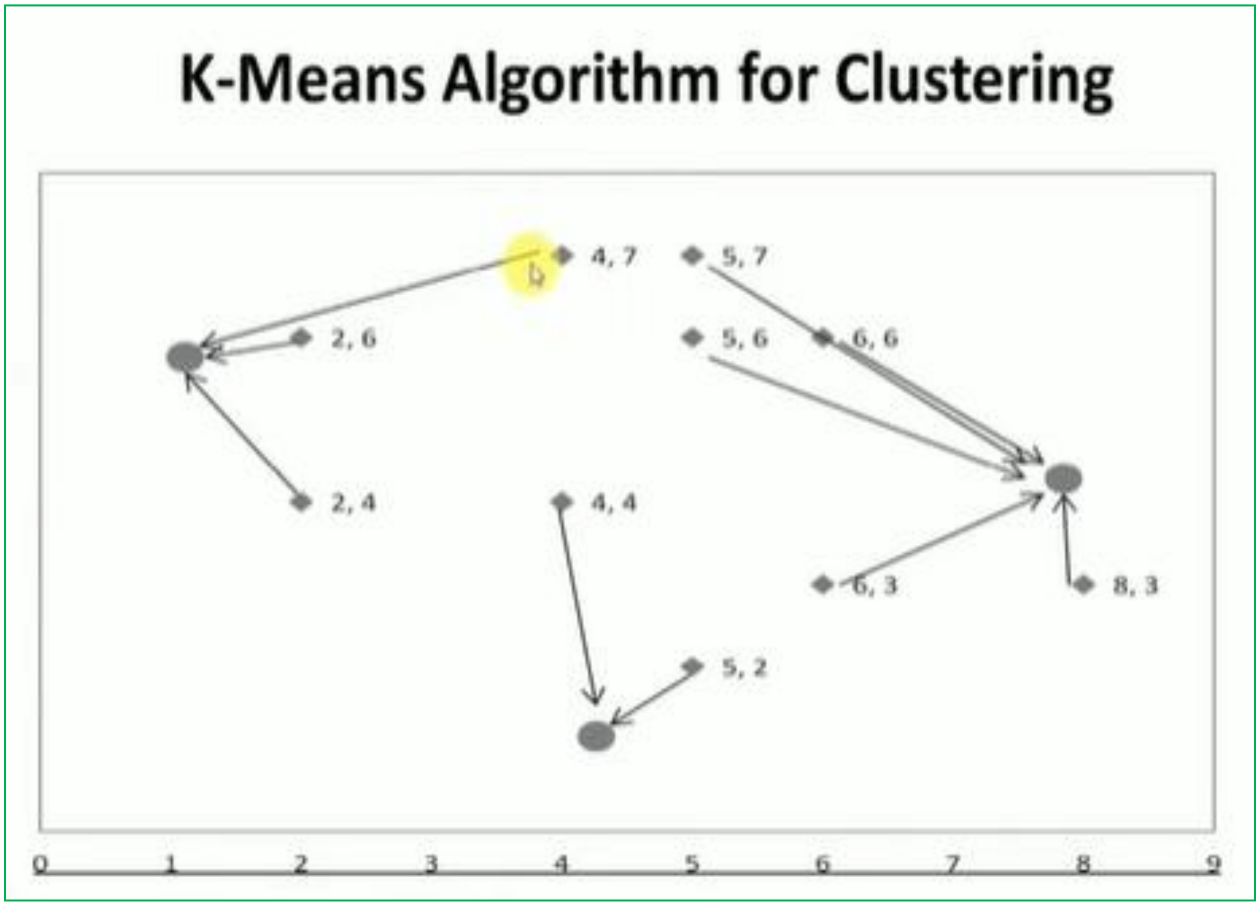
		Distance to			Cluster
X	Y	(1, 5)	(4, 1)	(8, 4)	Number
2	4	1.41	3.61	6.00	C1

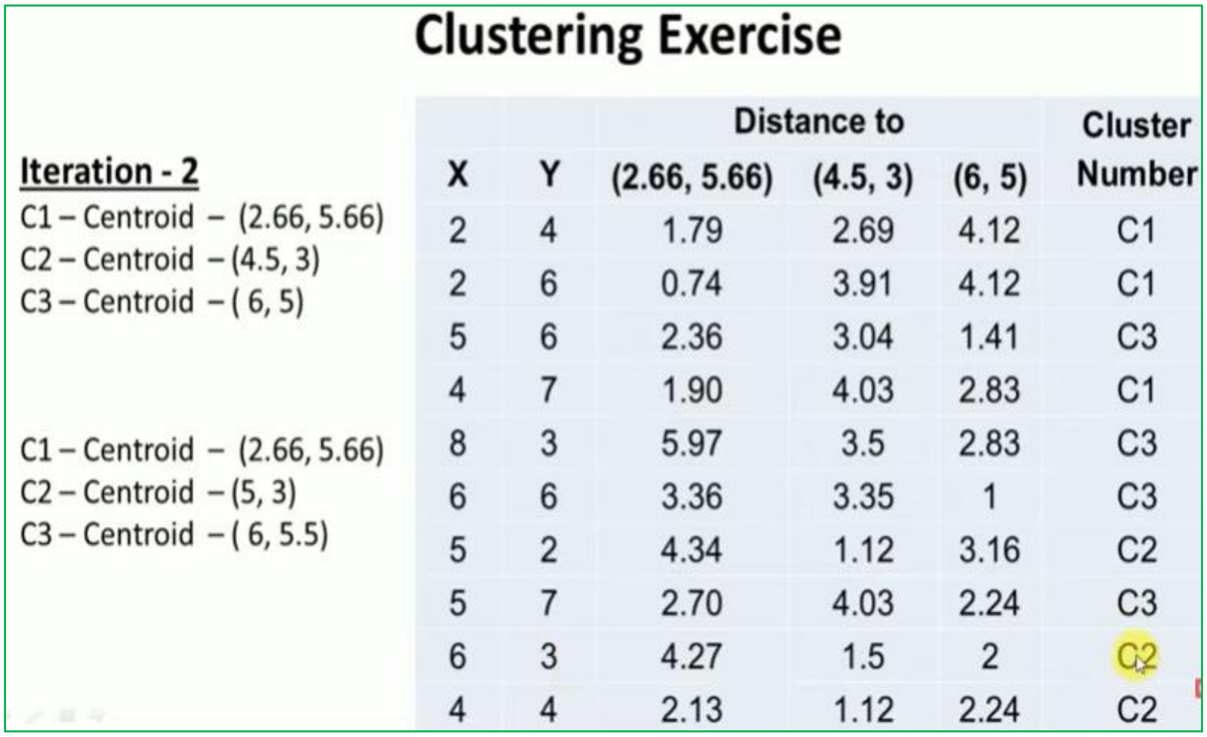
Among these points 1.41,3.61 and 6.00 , assign minimum as Cluster number ie., 1.41 is minimum and name as C1

Add centroid points (c1) and find the average. Similarly C2 and C3

For example c1(x)= (2+2+4)/3=2.66

c1(y)=(4+6+7)/3=5.66





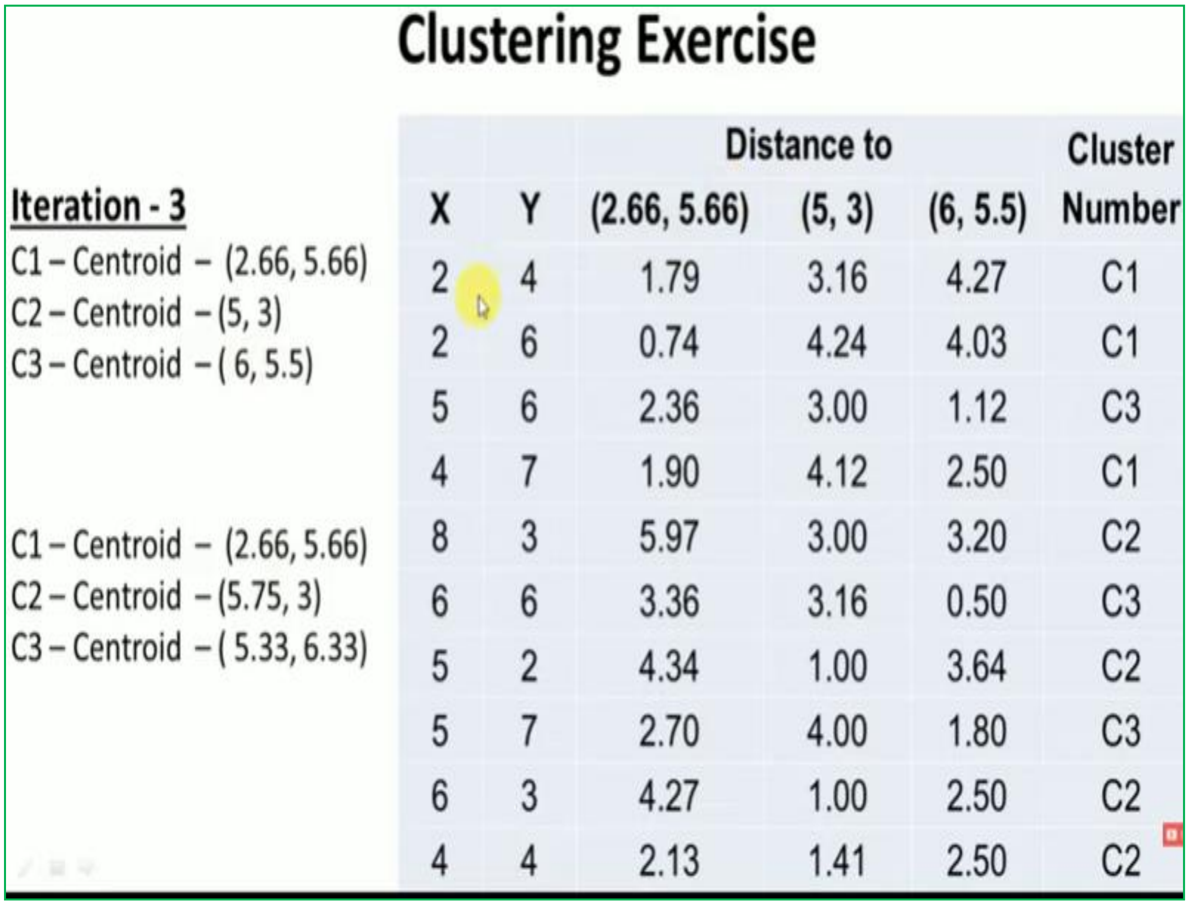
Clustering Exercise

Iteration - 2

C1 – Centroid – (2.66, 5.66)
C2 – Centroid – (4.5, 3)
C3 – Centroid – (6, 5)

C1 – Centroid – (2.66, 5.66)
C2 – Centroid – (5, 3)
C3 – Centroid – (6, 5.5)

		Distance to			Cluster
X	**Y**	**(2.66, 5.66)**	**(4.5, 3)**	**(6, 5)**	**Number**
2	4	1.79	2.69	4.12	C1
2	6	0.74	3.91	4.12	C1
5	6	2.36	3.04	1.41	C3
4	7	1.90	4.03	2.83	C1
8	3	5.97	3.5	2.83	C3
6	6	3.36	3.35	1	C3
5	2	4.34	1.12	3.16	C2
5	7	2.70	4.03	2.24	C3
6	3	4.27	1.5	2	C2
4	4	2.13	1.12	2.24	C2



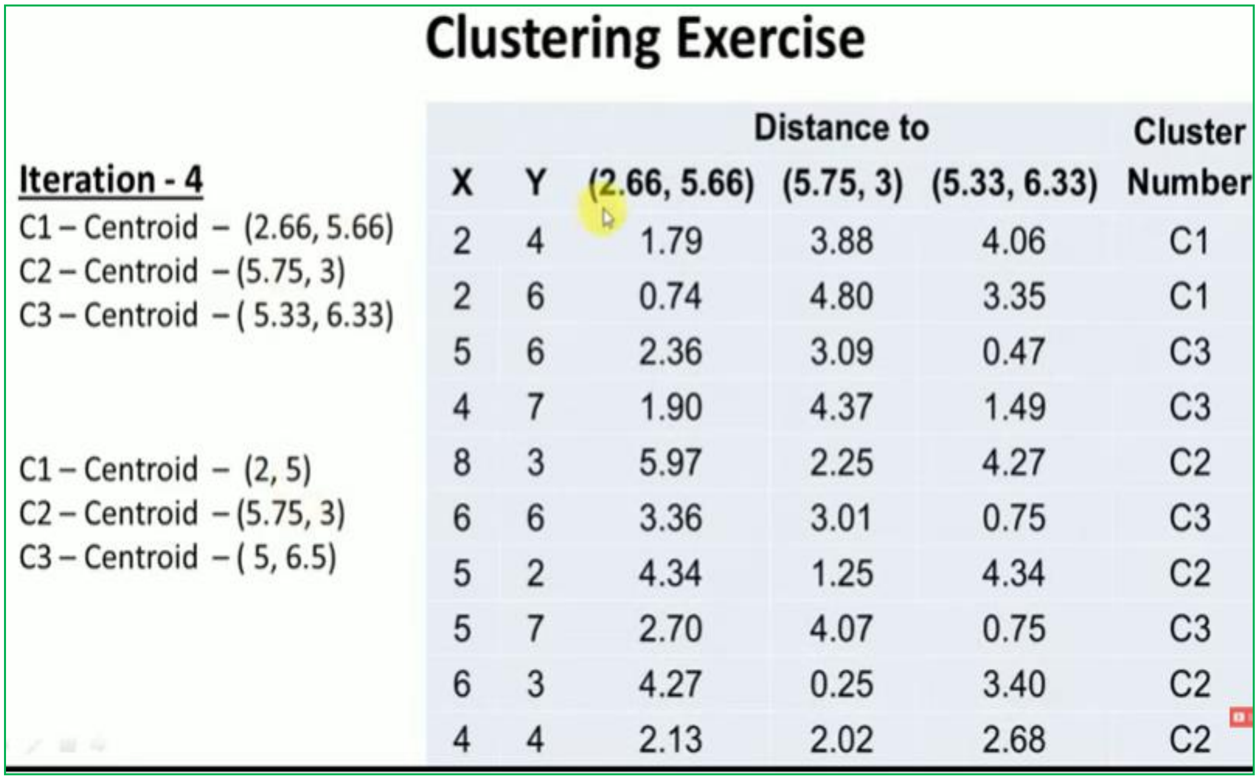
Clustering Exercise

Iteration - 3

C1 – Centroid – (2.66, 5.66)
C2 – Centroid – (5, 3)
C3 – Centroid – (6, 5.5)

C1 – Centroid – (2.66, 5.66)
C2 – Centroid – (5.75, 3)
C3 – Centroid – (5.33, 6.33)

		Distance to			Cluster
X	**Y**	**(2.66, 5.66)**	**(5, 3)**	**(6, 5.5)**	**Number**
2	4	1.79	3.16	4.27	C1
2	6	0.74	4.24	4.03	C1
5	6	2.36	3.00	1.12	C3
4	7	1.90	4.12	2.50	C1
8	3	5.97	3.00	3.20	C2
6	6	3.36	3.16	0.50	C3
5	2	4.34	1.00	3.64	C2
5	7	2.70	4.00	1.80	C3
6	3	4.27	1.00	2.50	C2
4	4	2.13	1.41	2.50	C2



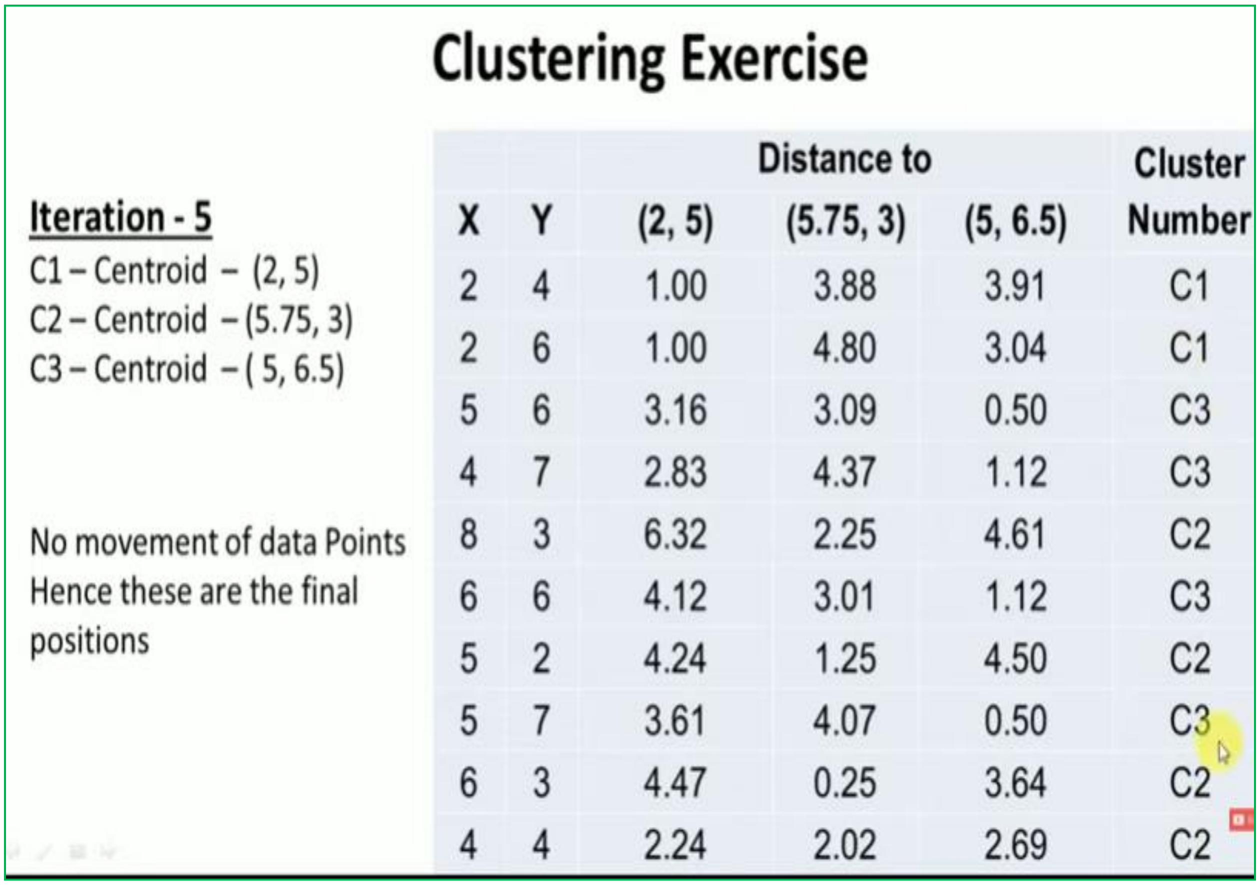
Clustering Exercise

Iteration - 4

C1 – Centroid – (2.66, 5.66)
C2 – Centroid – (5.75, 3)
C3 – Centroid – (5.33, 6.33)

C1 – Centroid – (2, 5)
C2 – Centroid – (5.75, 3)
C3 – Centroid – (5, 6.5)

		Distance to			Cluster
X	Y	(2.66, 5.66)	(5.75, 3)	(5.33, 6.33)	Number
2	4	1.79	3.88	4.06	C1
2	6	0.74	4.80	3.35	C1
5	6	2.36	3.09	0.47	C3
4	7	1.90	4.37	1.49	C3
8	3	5.97	2.25	4.27	C2
6	6	3.36	3.01	0.75	C3
5	2	4.34	1.25	4.34	C2
5	7	2.70	4.07	0.75	C3
6	3	4.27	0.25	3.40	C2
4	4	2.13	2.02	2.68	C2



Clustering Exercise

Iteration - 5

C1 – Centroid – (2, 5)
C2 – Centroid – (5.75, 3)
C3 – Centroid – (5, 6.5)

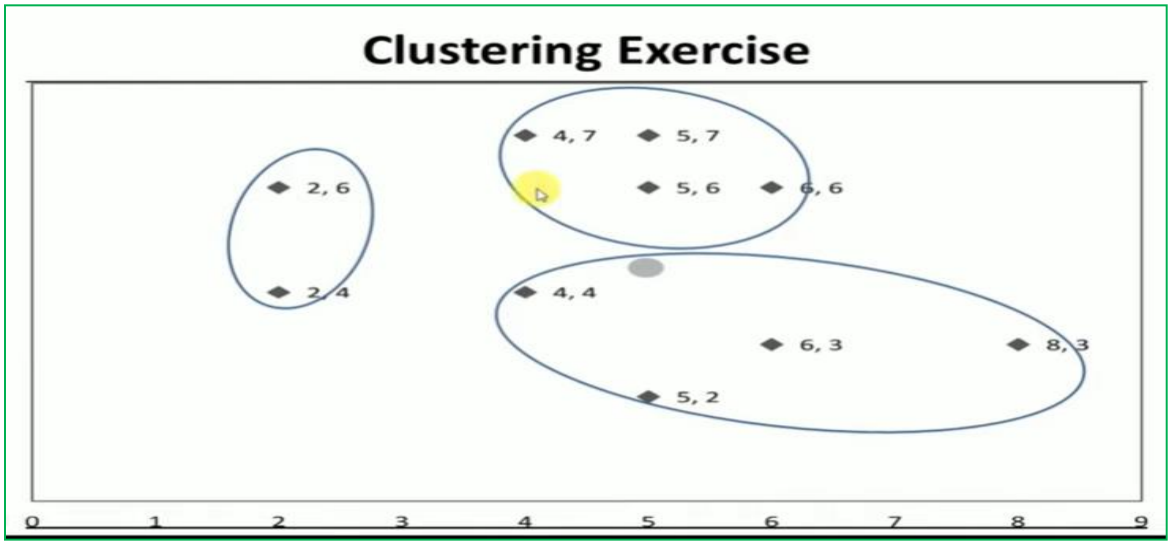
No movement of data Points
Hence these are the final positions

		Distance to			Cluster
X	Y	(2, 5)	(5.75, 3)	(5, 6.5)	Number
2	4	1.00	3.88	3.91	C1
2	6	1.00	4.80	3.04	C1
5	6	3.16	3.09	0.50	C3
4	7	2.83	4.37	1.12	C3
8	3	6.32	2.25	4.61	C2
6	6	4.12	3.01	1.12	C3
5	2	4.24	1.25	4.50	C2
5	7	3.61	4.07	0.50	C3
6	3	4.47	0.25	3.64	C2
4	4	2.24	2.02	2.69	C2

Clustering – solution

Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
C1	C1	C1	C1	C1
C1	C1	C1	C1	C1
C3	C3	C3	C3	C3
C1	C1	C1	**C3**	C3
C3	C3	**C2**	C2	C2
C3	C3	C3	C3	C3
C2	C2	C2	C2	C2
C3	C3	C3	C3	C3
C3	**C2**	C2	C2	C2
C2	C2	C2	C2	C2

At iteration 5 , there is no change in the clusters.



3.7 **Vector Quantization- Self Organizing Map**

- Vector quantization (VQ) is **an efficient coding technique to quantize signal vectors**. It has been widely used in signal and image processing, such as pattern recognition and speech and image coding.
- Vector quantization, also called "block quantization" or "pattern matching quantization" is often used in lossy data compression. It works by encoding values from a multidimensional vector space into a finite set of values from a discrete subspace of lower dimension.
- A lower-space vector requires less storage space, so the data is compressed. Due to the density matching property of vector quantization, the compressed data has errors that are inversely proportional to density.
- A vector quantizer maps *k-dimensional* vectors in the vector space R^k into a finite set of vectors $Y = \{y_i: i = 1, 2, ..., N\}$. Each vector y_i *is* called a code vector or a *codeword*. and the set of all the codewords is called a *codebook*. Associated with each codeword, y_i, is a nearest neighbor region called *Voronoi* region, and it is defined by:
- The set of Voronoi regions partition the entire space R^k.

$$V_i = \left\{ x \in R^k : \|x - y_i\| \le \|x - y_j\|, \text{for all } j \ne i \right\}$$

- Related application, data compression, which is used both for storing data and for the transmission of speech and image data. The reason that the applications are related is that both replace the current input by the cluster centre that it belongs to.
- For noise reduction, do this to replace the noisy input with a cleaner one, while for data compression do it to reduce the number of data points that send.
- instead of transmitting the actual data, I can transmit the index of that data point in the codebook, which is shorter- sound and image compression algorithm has a different method of solving it.
- Problem is that the codebook won't contain every possible datapoint - datapoint isn't in the codebook? In that case ,need to accept that our data will not look exactly the same, and I send you **the index of the prototype vector** that is closest to it (this is known as **vector quantisation**, and is the way that lossy compression works).
- The name for each cell is the Voronoi set of a particular prototype. Together, they produce the Voronoi tesselation of the space. If connect together every pair of points that share an

edge, as is shown by the dotted lines, then get the Delaunay triangulation, which is the optimal way to organize the space to perform function approximation

- Choose prototype vectors that are as close as possible to all of the possible inputs. This application is called learning vector quantisation because learning an efficient vector quantisation. The k-means algorithm can be used to solve the problem

3.7.1 Self Organizing Map

The self-organizing map also known as a Kohonen map.

• SOM is a technique which reduce the dimensions of data through the use of self-organizing neural networks.

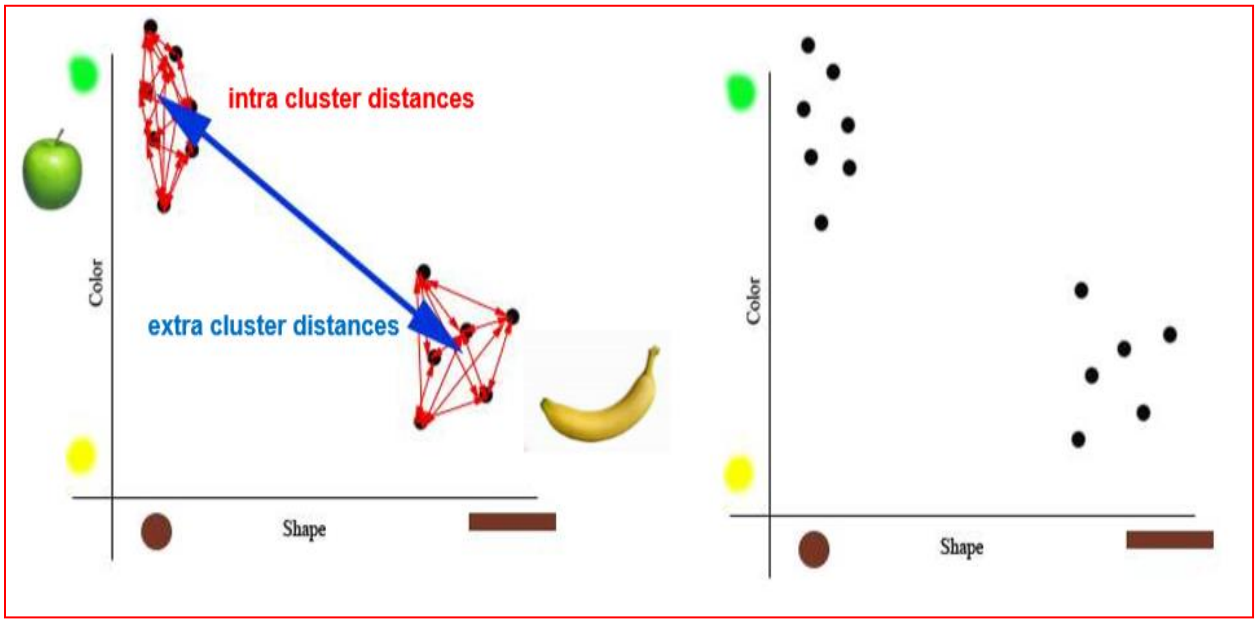
• The model was first described as an artificial neural network by professor Teuvo Kohonen.

Unsupervised learning

• Unsupervised learning is a class of problems in which one seeks to determine how the data are organized.

• One form of unsupervised learning is clustering.

How could we know what constitutes "different" clusters? • Green Apple and Banana Example. – two features: shape and color.



Competitive learning

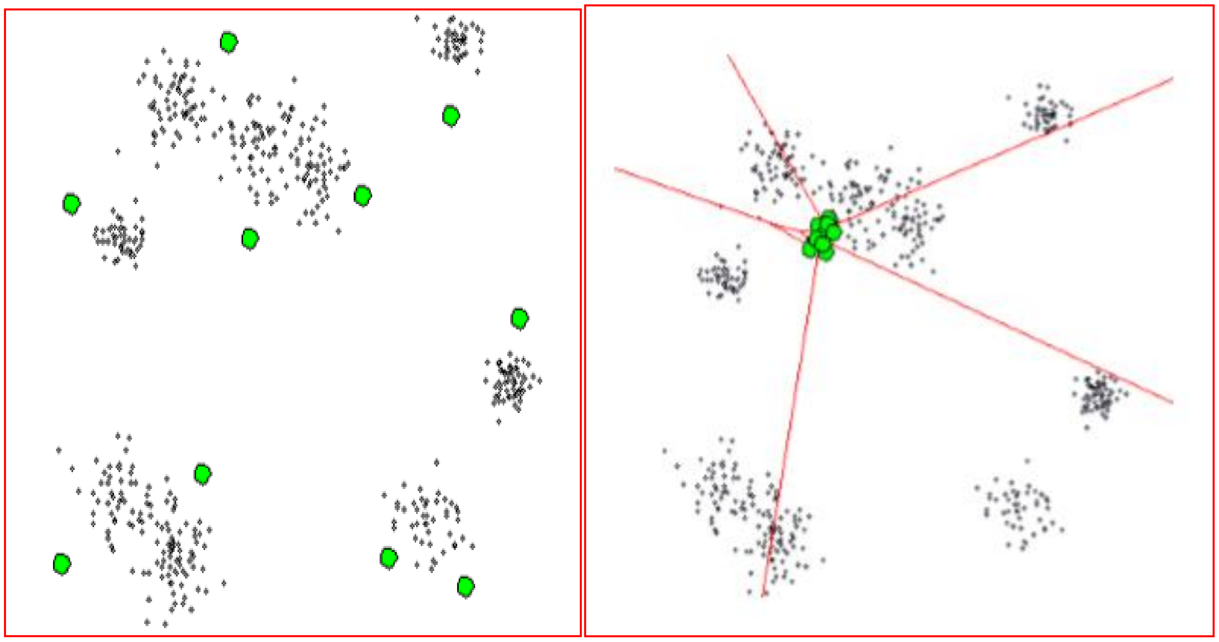
The position of the unit for each data point can be expressed as follows:

p(t+1) = a(p(t)-x) d(p(t),x)

• a is a factor called learning rate.

• d(p,x) is a distance scaling function.

Competitive learning is useful for clustering of input patterns into a discrete set of output clusters.

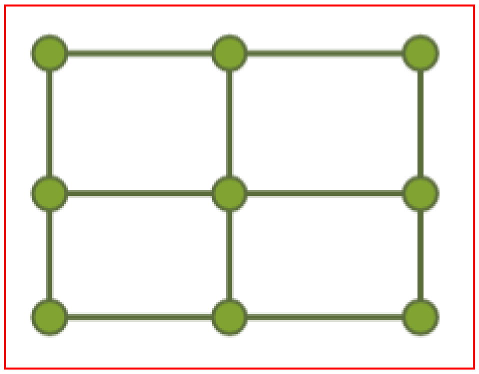


SOM is based on competitive learning.

- The difference is units are all interconnected in a grid.
- The unit closet to the input vector is call Best Matching Unit (BMU).
- The BMU and other units will adjust its position toward the input vector.
- The update formula is Wv(t + 1) = Wv(t) + Θ (v, t) α(t)(D(t) - Wv(t))
- Wv(t + 1) = Wv(t) + Θ (v, t) α(t)(D(t) - Wv(t))
  - Wv(t) = weight vector
  - α(t) = monotonically decreasing learning coefficient
  - D(t) = the input vector
  - Θ (v, t) = neighborhood function
- This process is repeated for each input vector for a (usually large) number of cycles λ.

Algorithm

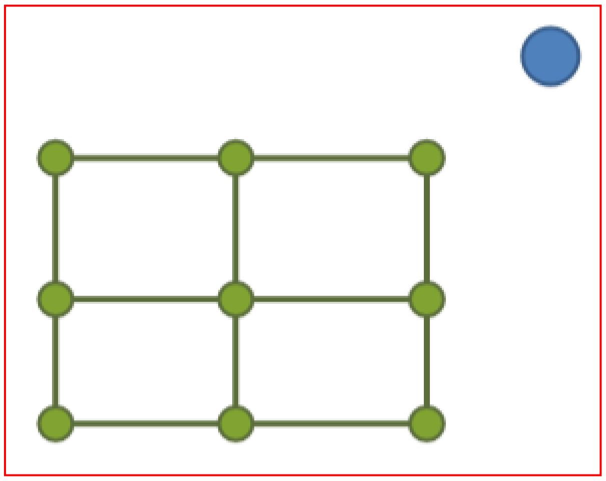
1. Randomize the map’s nodes’s weight vectors



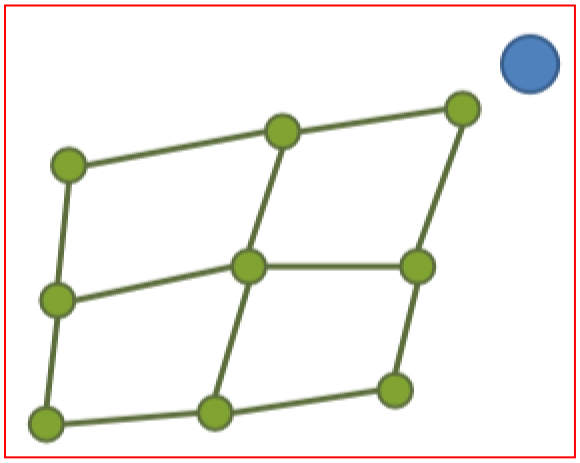
2. Grab an input vector

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix}$$

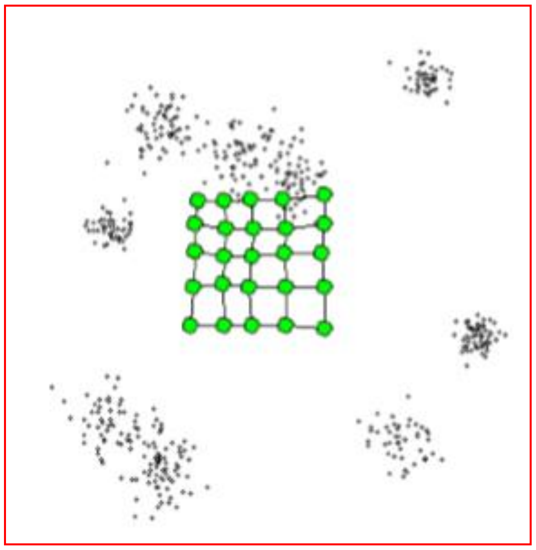
3. Traverse each node in the map

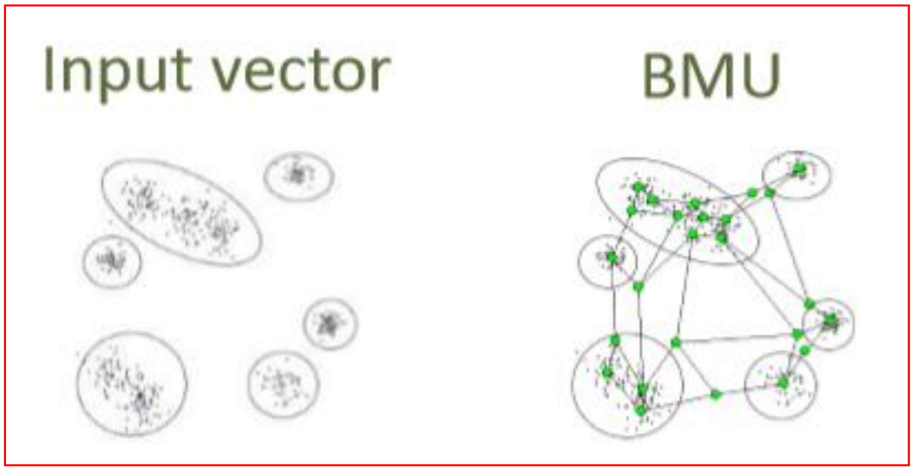


4. Update the nodes in the neighbourhood of BMU by pulling them closer to the input vector
   Wv(t + 1) = Wv(t) + Θ(t)α(t)(D(t) - Wv(t))



5. Increment t and repeat from 2 while t < λ





Application

- Dimensionality Reduction using SOM based Technique for Face Recognition
- A comparative study of PCA, SOM and ICA.
- SOM is better than the other techniques for the given face database and the classifier used.
- The results also show that the performance of the system decreases as the number of classes increase
- Gene functions can be analyzed using an adaptive method – SOM.
- Clustering with the SOM Visualizer - create a standard neural network based classifier based on the results.
- A neural network comprised of a plurality of layers of nodes.
- A system for organization of multi-dimensional pattern data into a two-dimensional representation comprising
- It allows for a reduced-dimension description of a body of pattern data to be representative of the original body of data.

UNIT – IV DIMENSIONALITY REDUCTION AND EVOLUTIONARY MODELS

Dimensionality Reduction – Linear Discriminant Analysis – Principal Component Analysis- Factor Analysis – Independent Component Analysis – Locally Linear Embedding – Isomap – Least Squares Optimization – Evolutionary Learning – Genetic algorithms – Genetic Offspring: - Genetic Operators – Using Genetic Algorithms – Reinforcement Learning – Overview – Getting Lost Example – Markov Decision Process

4.1 Dimensionality Reduction

- In machine learning classification problems, there are often too many factors on the basis of which the final classification is done. These factors are basically variables called features.
- The higher the number of features, the harder it gets to visualize the training set and then work on it. Sometimes, most of these features are correlated, and hence redundant. This is where dimensionality reduction algorithms come into play.
- Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables.
- It can be divided into feature selection and feature extraction.

There are two components of dimensionality reduction:

- **Feature selection:** In this, we try to find a subset of the original set of variables, or features, to get a smaller subset which can be used to model the problem. It usually involves three ways:
  - Filter
  - Wrapper
  - Embedded
- **Feature extraction:** This reduces the data in a high dimensional space to a lower dimension space, i.e. a space with lesser no. of dimensions.

Methods of Dimensionality Reduction

The various methods used for dimensionality reduction include:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)

Principal components analysis is a dimensionality reduction technique that enables you to identify correlations and patterns in a data set so that it can be transformed into a data set of significantly lower dimension without loss of any important information. Principal component analysis (PCA) is a standard tool in modern data analysis - in diverse fields from neuroscience to computer graphics. It is very useful method for extracting relevant information from confusing data sets. Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables.

Goals

- The main goal of a PCA analysis is to identify patterns in data
- PCA aims to detect the correlation between variables.
- It attempts to reduce the dimensionality.

Need for PCA

- -High dimension data is extremely complex to process due to inconsistencies in the features which increase the computation time and make data processing and Exploratory data Analysis (EDA) more convoluted.

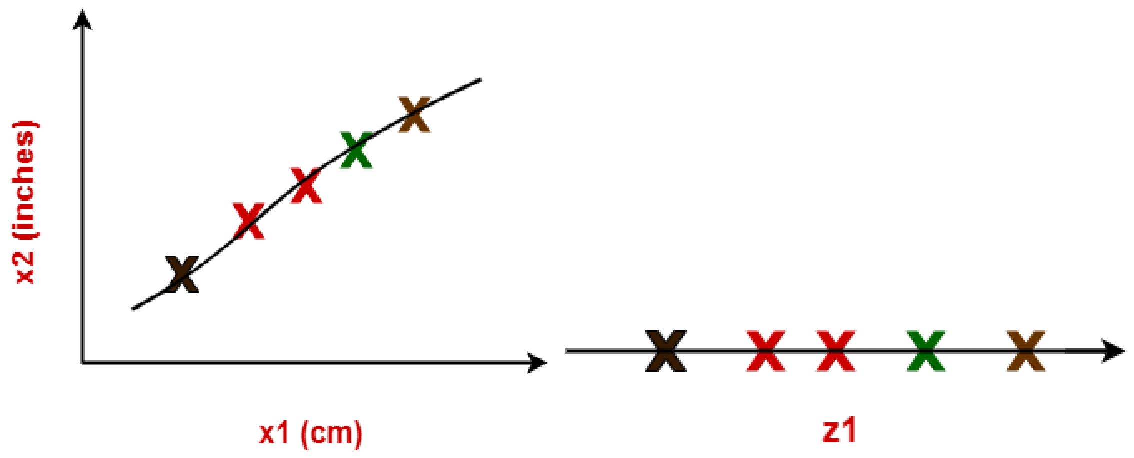
Dimensionality Reduction

- It reduces the dimensions of a d-dimensional dataset by projecting it onto a (k)-dimensional subspace (where k<d) in order to increase the computational efficiency while retaining most of the information.
- In pattern recognition, Dimension Reduction is defined as a process of converting a data set having vast dimensions into a data set with lesser dimensions.
- It ensures that the converted data set conveys similar information concisely.

Example-

Consider the following example-

- The following graph shows two dimensions x1 and x2.
- x1 represents the measurement of several objects in cm.
- x2 represents the measurement of several objects in inches.



In machine learning,

- Using both these dimensions convey similar information.
- Also, they introduce a lot of noise in the system.
- So, it is better to use just one dimension.

Using dimension reduction techniques-

- We convert the dimensions of data from 2 dimensions (x1 and x2) to 1 dimension (z1).
- It makes the data relatively easier to explain.

Benefits-

- Dimension reduction offers several benefits such as-
- It compresses the data and thus reduces the storage space requirements.
- It reduces the time required for computation since less dimensions require less computation.
- It eliminates the redundant features.
- It improves the model performance.

Dimension Reduction Techniques-

- The two popular and well-known dimension reduction techniques are-
  - Principal Component Analysis
  - Linear Discriminant Analysis

Transformation

- This transformation is defined in such a way that the first principal component has the largest possible variance and each succeeding component in turn has the next highest possible variance.

PCA Approach

- Standardize the data.
- Perform Singular Vector Decomposition to get the Eigenvectors and Eigenvalues.
- Sort eigenvalues in descending order and choose the k- eigenvectors
- Construct the projection matrix from the selected k- eigenvectors.
- Transform the original dataset via projection matrix to obtain a k-dimensional feature subspace.
- Principal Component Analysis is a well-known dimension reduction technique.
- It transforms the variables into a new set of variables called as principal components.
- These principal components are linear combination of original variables and are orthogonal.
- The first principal component accounts for most of the possible variation of original data.
- The second principal component does its best to capture the variance in the data.

There can be only two principal components for a two-dimensional data set.

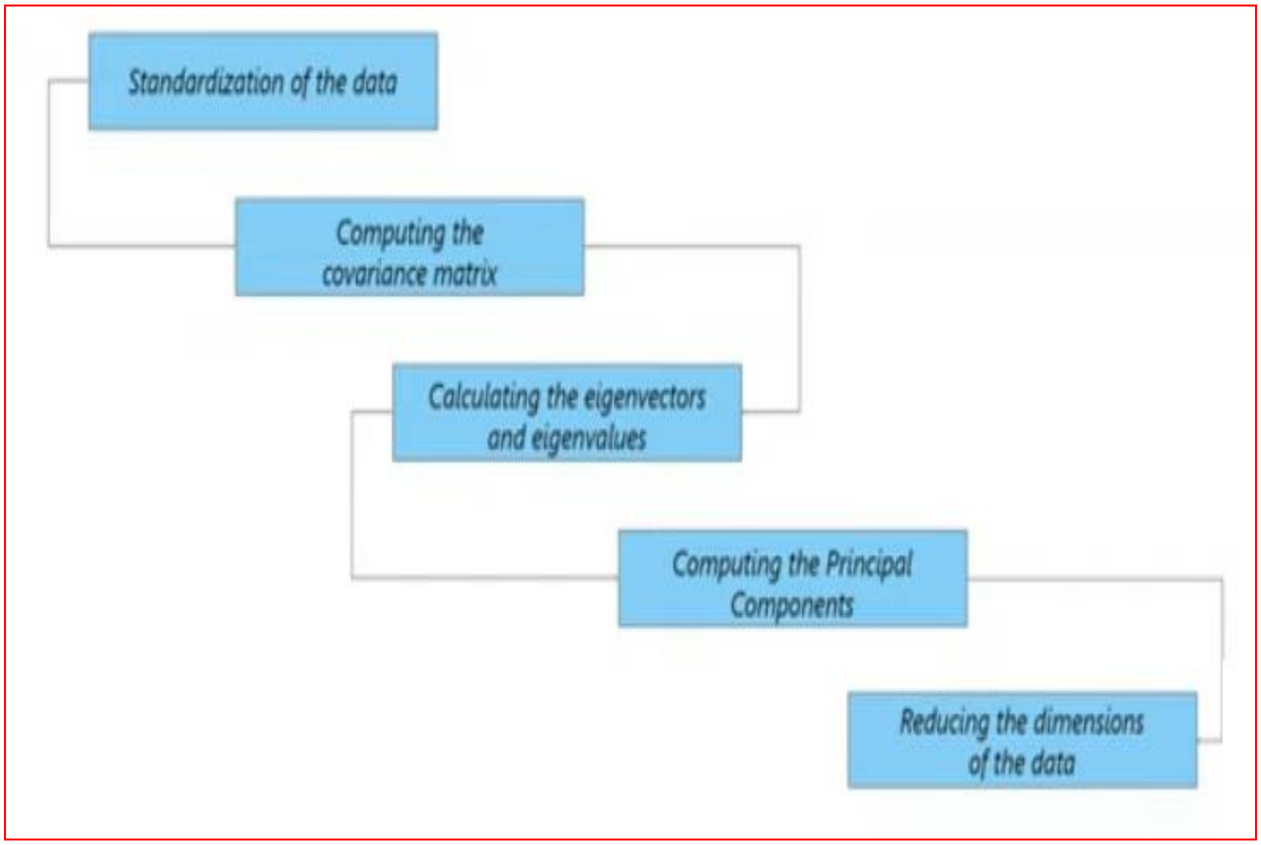
Limitation of PCA

- The results of PCA depend on the scaling of the variables.
- A scale-invariant form of PCA has been developed.

Applications of PCA :

- Interest Rate Derivatives Portfolios
- Neuroscience

PCA algorithm steps:


Standardization of the data
Computing the covariance matrix
Calculating the eigenvectors and eigenvalues
Computing the Principal Components
Reducing the dimensions of the data

Step 1: Standardization of the data- is all about scaling your data in such a way that all the variables and their values lie within a similar range

PCA Algorithm-

The steps involved in PCA Algorithm are as follows-

Step-01: Get data.

Step-02: Compute the mean vector (μ).

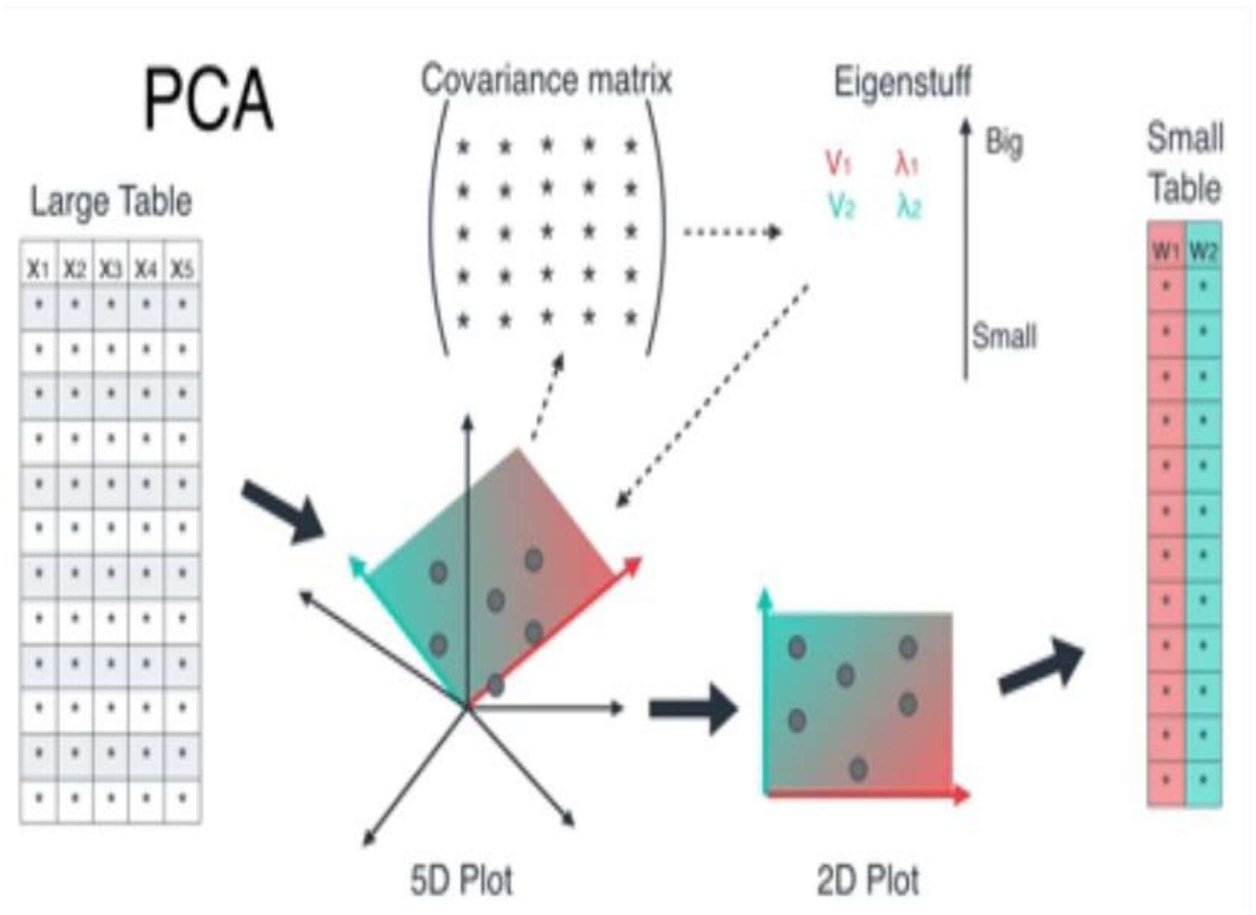
Step-03: Subtract mean from the given data.

Step-04: Calculate the covariance matrix.

Step-05: Calculate the eigen vectors and eigen values of the covariance matrix.

Step-06: Choosing components and forming a feature vector.

Step-07: Deriving the new data set.



<u>Problem-01:</u>

Given data = { 2, 3, 4, 5, 6, 7 ; 1, 5, 3, 6, 7, 8 }.

Compute the principal component using PCA Algorithm.

OR

Consider the two dimensional patterns (2, 1), (3, 5), (4, 3), (5, 6), (6, 7), (7, 8).

Compute the principal component using PCA Algorithm.

OR

Compute the principal component of following data-

CLASS 1

X = 2 , 3 , 4

Y = 1 , 5 , 3

CLASS 2

X = 5 , 6 , 7

Y = 6 , 7 , 8

Solution-

Step-01:

Get data.

The given feature vectors are-

$x_1 = (2, 1)$

$x_2 = (3, 5)$

$x_3 = (4, 3)$

$x_4 = (5, 6)$

$x_5 = (6, 7)$

$x_6 = (7, 8)$

$$\begin{bmatrix} 2 \\ 1 \end{bmatrix} \begin{bmatrix} 3 \\ 5 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \end{bmatrix} \begin{bmatrix} 6 \\ 7 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \end{bmatrix}$$

Step-02:

Calculate the mean vector (μ).

Mean vector (μ)

= ((2 + 3 + 4 + 5 + 6 + 7) / 6, (1 + 5 + 3 + 6 + 7 + 8) / 6)= (4.5, 5)

Thus,

$$\text{Mean vector } (\mu) = \begin{bmatrix} 4.5 \\ 5 \end{bmatrix}$$

Step-03:

Subtract mean vector (μ) from the given feature vectors.

x1 – μ = (2 – 4.5, 1 – 5) = (-2.5, -4)

x2 – μ = (3 – 4.5, 5 – 5) = (-1.5, 0)

x3 – μ = (4 – 4.5, 3 – 5) = (-0.5, -2)

x4 – μ = (5 – 4.5, 6 – 5) = (0.5, 1)

x5 – μ = (6 – 4.5, 7 – 5) = (1.5, 2)

x6 – μ = (7 – 4.5, 8 – 5) = (2.5, 3)

Feature vectors (xi) after subtracting mean vector (μ) are-

$$\begin{bmatrix} -2.5 \\ -4 \end{bmatrix} \begin{bmatrix} -1.5 \\ 0 \end{bmatrix} \begin{bmatrix} -0.5 \\ -2 \end{bmatrix} \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \begin{bmatrix} 1.5 \\ 2 \end{bmatrix} \begin{bmatrix} 2.5 \\ 3 \end{bmatrix}$$

Step-04:

Calculate the covariance matrix.

Covariance matrix is given by-

$$\boxed{\text{Covariance Matrix} = \frac{\Sigma (x_i - \mu)(x_i - \mu)^t}{n}}$$

$$m_1 = (x_1 - \mu)(x_1 - \mu)^t = \begin{bmatrix} -2.5 \\ -4 \end{bmatrix} \begin{bmatrix} -2.5 & -4 \end{bmatrix} = \begin{bmatrix} 6.25 & 10 \\ 10 & 16 \end{bmatrix}$$

$$m_2 = (x_2 - \mu)(x_2 - \mu)^t = \begin{bmatrix} -1.5 \\ 0 \end{bmatrix} \begin{bmatrix} -1.5 & 0 \end{bmatrix} = \begin{bmatrix} 2.25 & 0 \\ 0 & 0 \end{bmatrix}$$

$$m_3 = (x_3 - \mu)(x_3 - \mu)^t = \begin{bmatrix} -0.5 \\ -2 \end{bmatrix} \begin{bmatrix} -0.5 & -2 \end{bmatrix} = \begin{bmatrix} 0.25 & 1 \\ 1 & 4 \end{bmatrix}$$

$$m_4 = (x_4 - \mu)(x_4 - \mu)^t = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \begin{bmatrix} 0.5 & 1 \end{bmatrix} = \begin{bmatrix} 0.25 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

$$m_5 = (x_5 - \mu)(x_5 - \mu)^t = \begin{bmatrix} 1.5 \\ 2 \end{bmatrix} \begin{bmatrix} 1.5 & 2 \end{bmatrix} = \begin{bmatrix} 2.25 & 3 \\ 3 & 4 \end{bmatrix}$$

$$m_6 = (x_6 - \mu)(x_6 - \mu)^t = \begin{bmatrix} 2.5 \\ 3 \end{bmatrix} \begin{bmatrix} 2.5 & 3 \end{bmatrix} = \begin{bmatrix} 6.25 & 7.5 \\ 7.5 & 9 \end{bmatrix}$$

Now,

Covariance matrix= $(m_1 + m_2 + m_3 + m_4 + m_5 + m_6) / 6$

On adding the above matrices and dividing by 6, we get-

$$\text{Covariance Matrix} = \frac{1}{6} \begin{bmatrix} 17.5 & 22 \\ 22 & 34 \end{bmatrix}$$

$$\text{Covariance Matrix} = \begin{bmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{bmatrix}$$

From here,

$(2.92 – \lambda)(5.67 – \lambda) – (3.67 \text{ x } 3.67) = 0$

$16.56 – 2.92\lambda – 5.67\lambda + \lambda^2 – 13.47 = 0$

$\lambda^2 – 8.59\lambda + 3.09 = 0$

Solving this quadratic equation, we get $\lambda = 8.22, 0.38$

Thus, two eigen values are $\lambda_1 = 8.22$ and $\lambda_2 = 0.38$.

Clearly, the second eigen value is very small compared to the first eigen value.

So, the second eigen vector can be left out.

Eigen vector corresponding to the greatest eigen value is the principal component for the given data set. So. we find the eigen vector corresponding to eigen value λ_1.

We use the following equation to find the eigen vector-

$MX = \lambda X$

where- M = Covariance Matrix, X = Eigen vector. λ = Eigen value

Substituting the values in the above equation, we get-

$$\begin{bmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{bmatrix} \begin{bmatrix} X1 \\ X2 \end{bmatrix} = 8.22 \begin{bmatrix} X1 \\ X2 \end{bmatrix}$$

Solving these, we get-

$2.92X_1 + 3.67X_2 = 8.22X_1$

$3.67X_1 + 5.67X_2 = 8.22X_2$

On simplification, we get-

$5.3X_1 = 3.67X_2$ ....(1)

$3.67X_1 = 2.55X_2$ ....(2)

From (1) and (2), **$X_1 = 0.69X_2$**

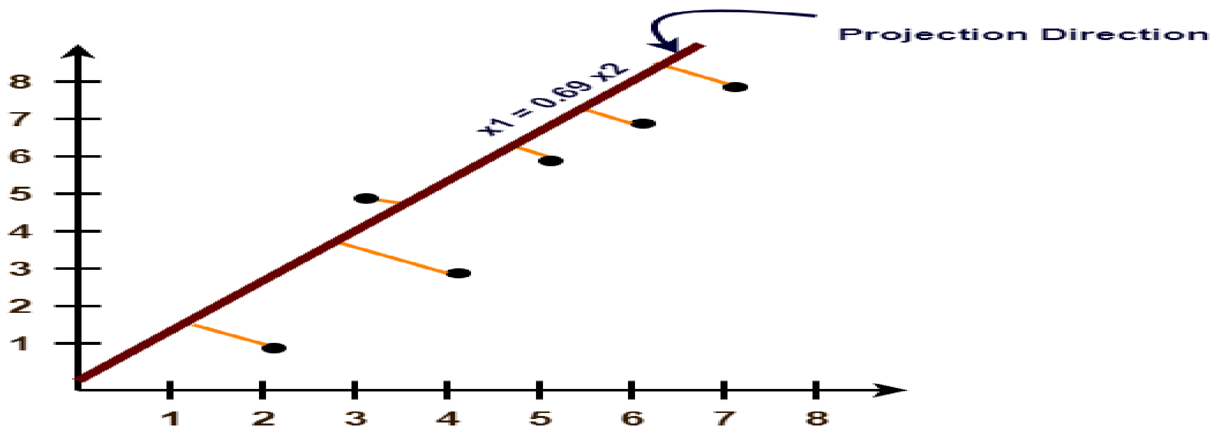
From (2), the eigen vector is-

$$\text{Eigen Vector :} \quad \begin{bmatrix} X1 \\ X2 \end{bmatrix} = \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}$$

Thus , Principal component for the given data set is

$$\text{Principal Component :} \quad \begin{bmatrix} X1 \\ X2 \end{bmatrix} = \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}$$

Lastly, we project the data points onto the new subspace as-

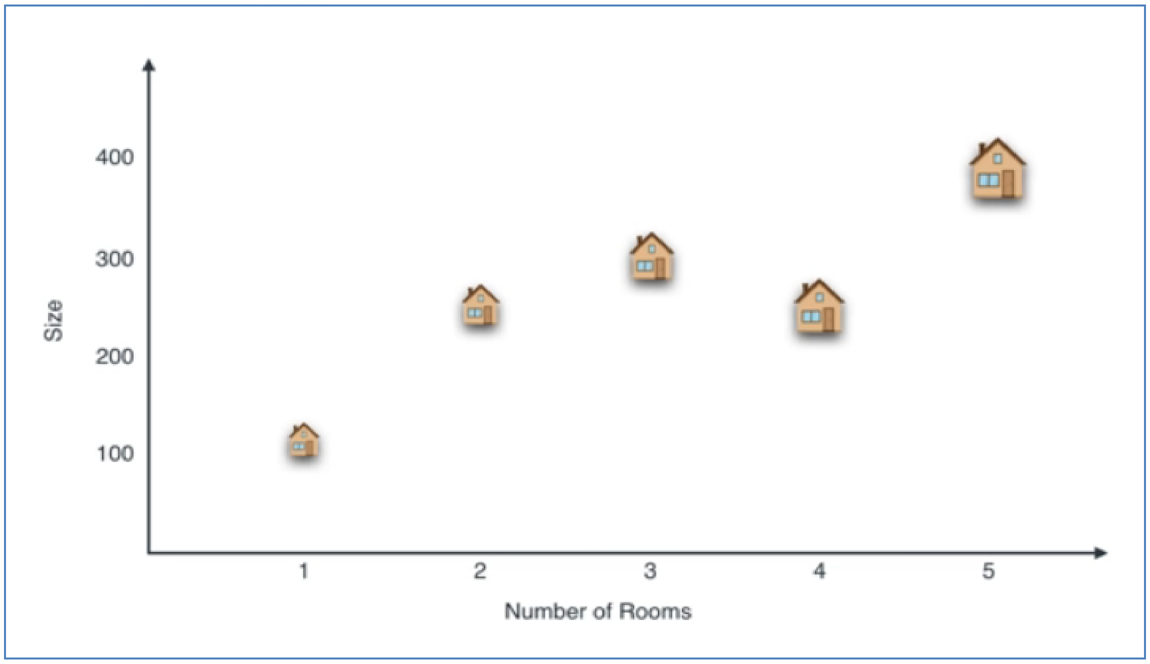


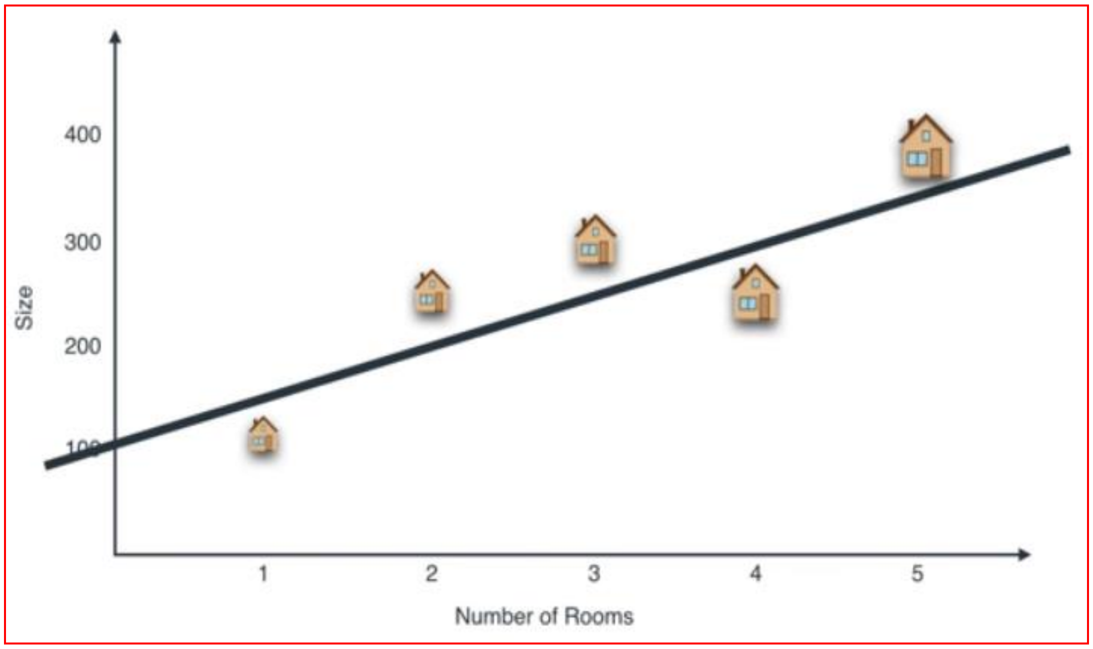
Example

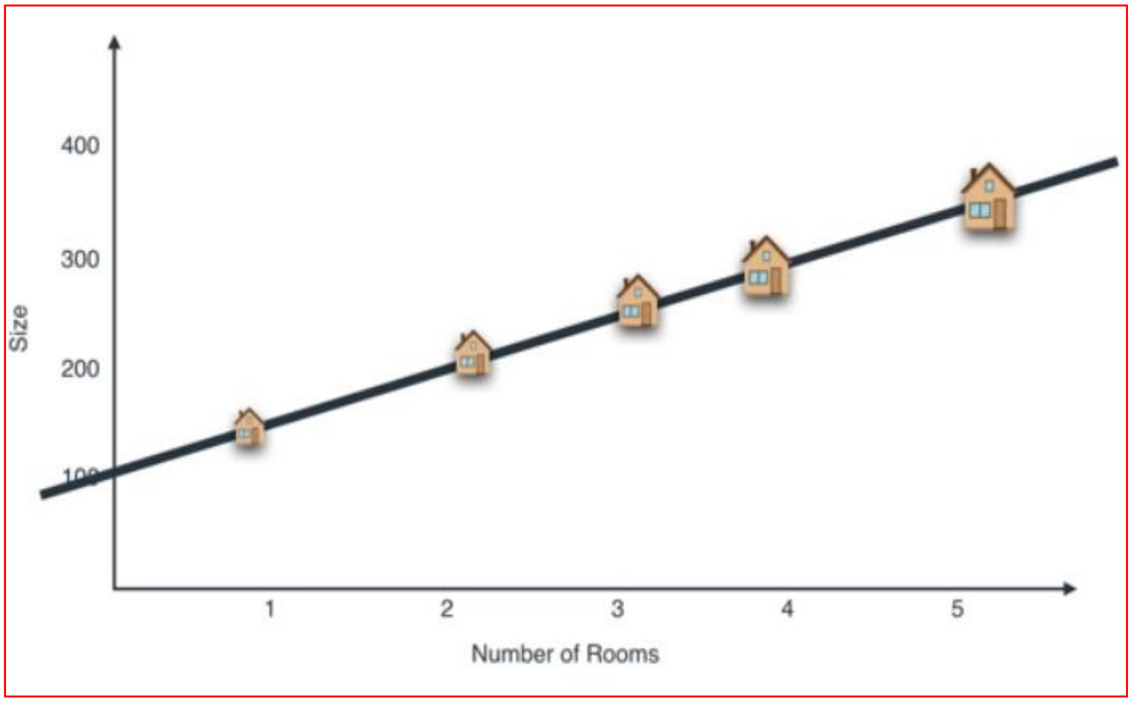
Housing Data

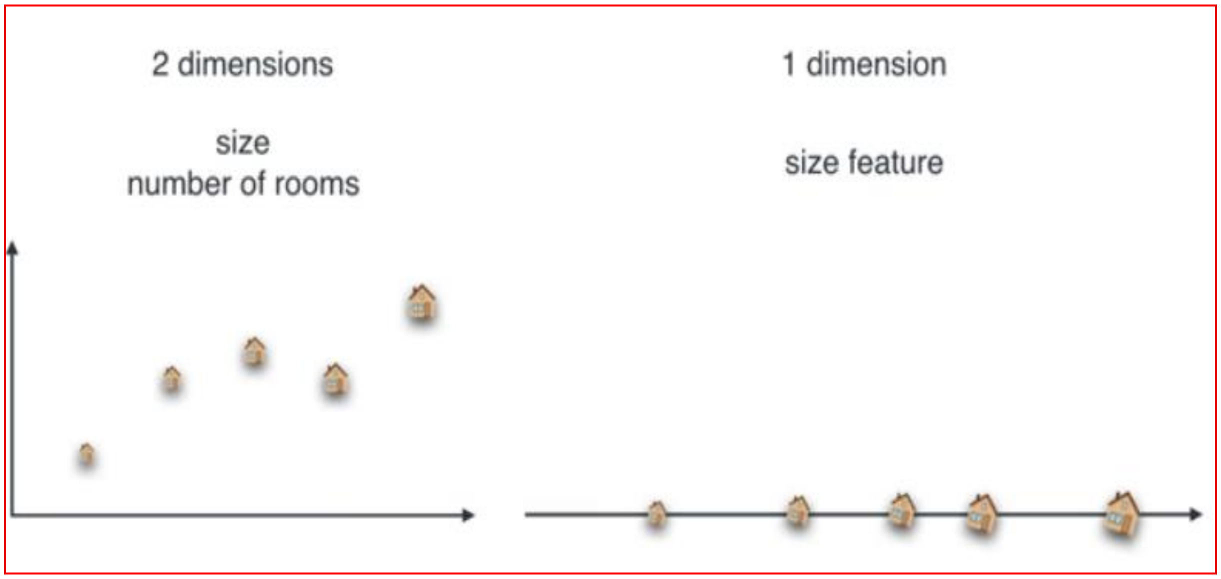
Size
Number of rooms $\longrightarrow$ Size feature
Number of bathrooms

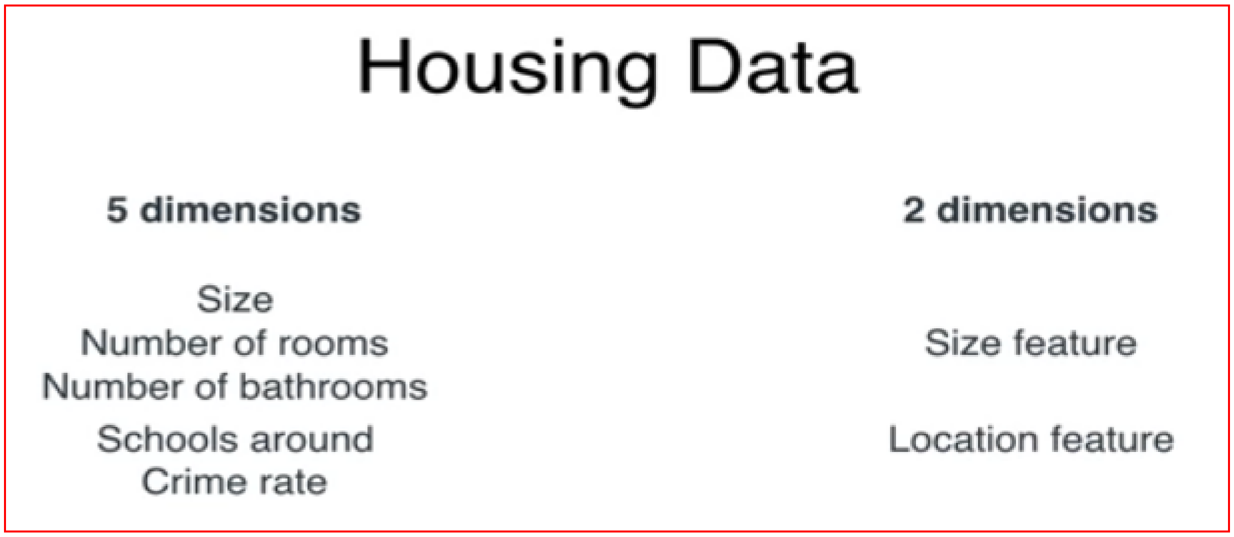
Schools around $\longrightarrow$ Location feature
Crime rate




Size
400
300
200
100
1
2
3
4
5
Number of Rooms


Size
400
300
200
100
1
2
3
4
5
Number of Rooms




Housing Data
5 dimensions
Size
Number of rooms
Number of bathrooms
Schools around
Crime rate
2 dimensions
Size feature
Location feature

4.2 Factor Analysis – Independent Component Analysis – Locally Linear Embedding – Isomap

- It is a statistical method used to describe variability among observed/correlated variables.
  - It is possible that variations in 6 observed variables mainly reflect variations in 2 unobserved(underlying) variables
- Aim to find independent latent variables
- Variable reduction technique
- Reduction of set of variables in terms of latent factors

- Unobserved factors account for correlation among observed variables
- ✓ Exploratory FA-Explore the pattern among the variables
  - ✓ No prior hypothesis to start with
- ✓ Confirmatory FA-Used for confirming model specification
  - ✓ Model is already in place
- ✓ FA is a correlational method used to find and describe the underlying factors driving data values for a large set of variables.
- ✓ FA identifies correlations between and among variables to bind them into one underlying factor driving their values
- ✓ FA is used for data summarization or data reduction
- ✓ FA examines the interrelationships among a large number of variables and then attempts to explain them in terms of their common underlying dimension
  - ✓ Common underlying dimensions are referred to as factors
- ✓ Interdependence techniques
  - ✓ No Interdependent Variable or Dependent Variable
  - ✓ All variables are considered simultaneously

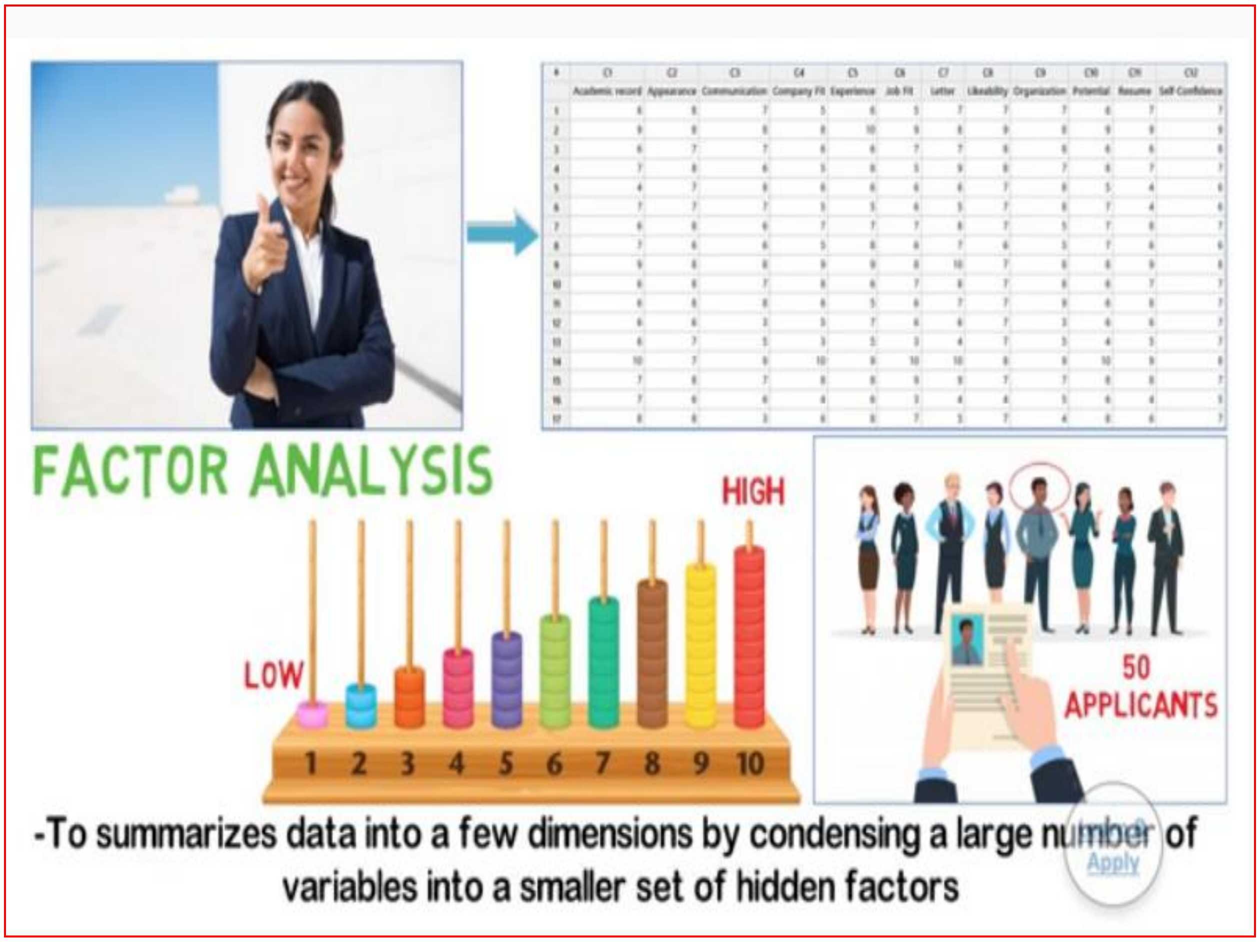
Assumptions of FA

- No outliers in the data set
  - This means that no extreme data are present in the data set. For example, consider the following values:1,5,8,-4,23,18 and 1,247,942
  - The latter value is an outlier that overwhelms the remaining data
- Adequate sample size must be present
  - This means that you must have more variables than you have factors. You cannot have only 3 variables and have 4 factors
  - Each variable must also have more data values than you have factors. Our data sets will be large.
- No perfect multicollinearity

    - This means that each value is unique.
- Homogeneity or Homoscedasticity not required between variables
    - Homoscedasticity means that all variables have the same finite variance. In other words, the curves do not have to possess the same size standard deviations
- Linearity of variables
    - Each of the variables should be linear in nature
    - Factor analysis is a linear function of measured variables.
- The data must be at least Interval
    - Nominal and ordinal data don't work with factor analysis

Understanding Factor Analysis

- Regardless of purpose, factor analysis is used in
    - The determination of a small number of factors based on a particular number of inters related quantitative variables.
- Unlike variables directly measured such as speed, height, weight etc., some variables such as creativity, happiness, religiosity, comfort are not a single measureable entity
- They are constructs that are derived from the measurement of other directly observable variables



- Factor analysis is a correlational method used to find and describe the underlying factors driving data values for a large set of variables
- It identifies correlations between and among variables to bind them into one underlying factor driving their values.
- For example: In a set variables(V1,V2,V3,V4,V5,V6) a correlational relationship may be found between V1,V3 and V4. This means that these 3 variables may in fact be only one value or factor.
- Accordingly, large numbers of variables may be reduced to only several factors
- V1,V3,V4 are a factor
- V2 & V6 are a factor
- V5 is a factor.
- Data set is explained by 3 factors rather than by 6 variables. It is referred to as data reduction.

- Example : Result analysis
- Need to
    - Determine the assumptions for factor analysis
    - Develop a means of identifying factors.
    - Determine if a factor is important or not and
    - Examine the interaction of the variables on the factor

The problem of factor analysis is to find those independent factors, and the noise that is inherent in the measurements of each factor. Factor analysis is commonly used in psychology and other social sciences

Uses of FA

- Variable reduction- large collection of correlated variables
- Patter among variables-Exploratory Data Analysis
- Confirmatory Modelling

INDEPENDENT COMPONENTS ANALYSIS (ICA)

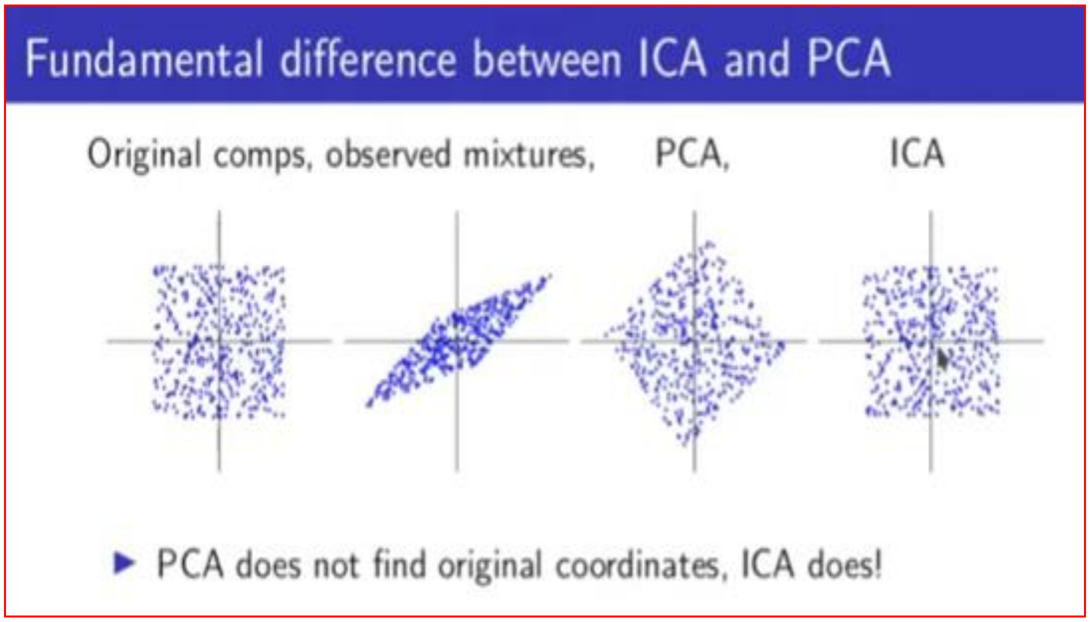
- A technique for revealing the hidden structure of data
- ICA is a computational technique used for extracting the source signals from an observed mixture of multivariate signal.
- ICA model
    - The observed data variables are linear mixtures of latent variables
    - The mixing process is unknown
    - The latent variables are assumed to be non-Gaussian and statistically independent; they are called independent components
- ICA can find independent components
- ICA is used for source signals separation in many applications including medical data, audio signals, or optical imaging. Data can be in the form of images, sounds or other time series data
- ICA can be applied as a dimensionality reduction algorithm because after extracting the source signals, the unnecessary signals can be removed or deleted.

- ICA is considered as an extension of the PCA. But PCA tries to find the axis that maximizes the variance using second order statistics, while ICA tries to maximize the independence between source signals using the higher order statistics
- Factor Analysis related approach known as Independent Components Analysis.
- In PCA, the components were orthogonal and uncorrelated (so that the covariance matrix was diagonal, i.e., so cov(bi,bj)=0.
- Require that the components are statistically independent (so that for E[bi,bj]= E[bi]E[bj] as well as the bi being uncorrelated), then get ICA. The common motivation for ICA is the problem of blind source separation(Blind signal separation (BSS) aims at recovering unknown source signals from the observed sensor signals where the mixing process is also unknown).
- The most popular way to describe blind source separation is known as the cocktail party problem.

The cock tail party problem is the challenge of separating out   hear lots of different sounds coming from lots of different locations (different people talking, the clink of glasses, background music, etc.) sources from the party. The objective is to detect the speech of different people where a group of people are talking simultaneously

Properties of the source signals

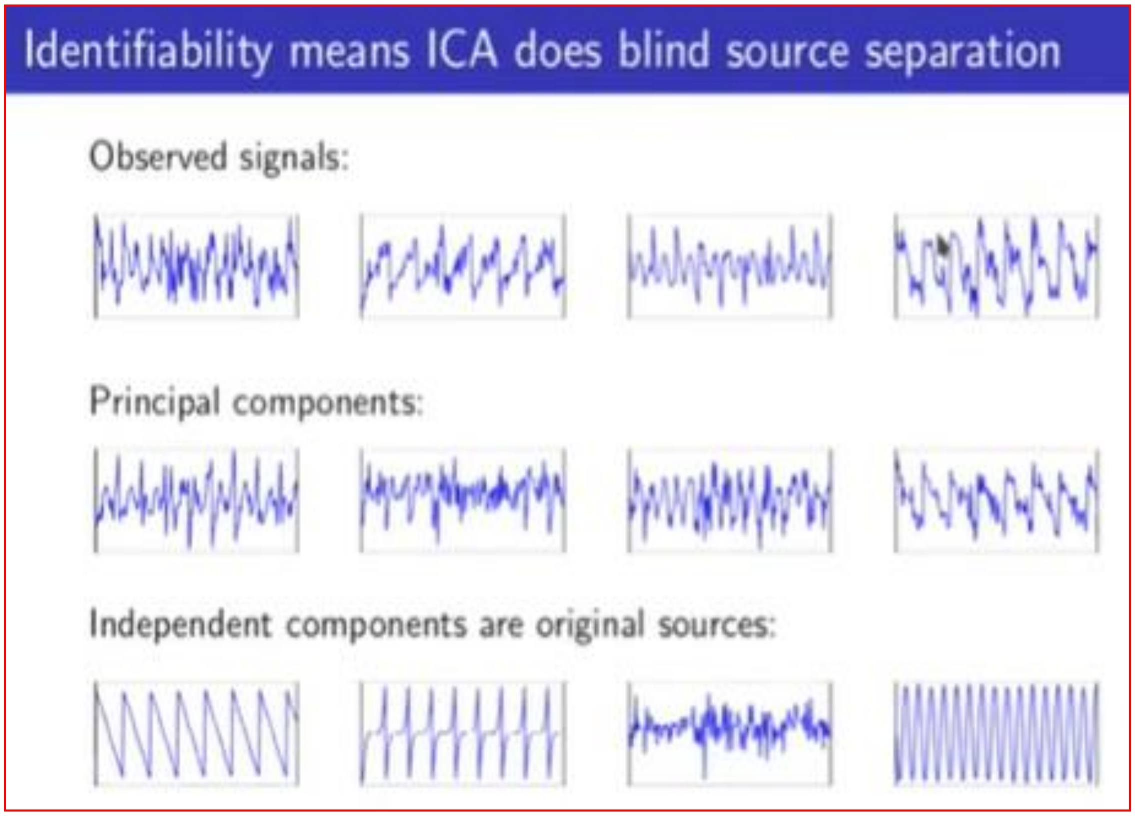
- Three assumptions about the source signals:
  - Independence: the source signal are independent; however, their signal mixtures are not as they share the same source signals
  - Non normality: independent signals should come from non Gaussian distributions. Otherwise, the ICA cannot be applied to extract source signals.
  - Complexity: the complexity of the mixed signals is greater than that of its components



Fundamental difference between ICA and PCA

Original comps, observed mixtures, PCA, ICA

- PCA does not find original coordinates, ICA does!

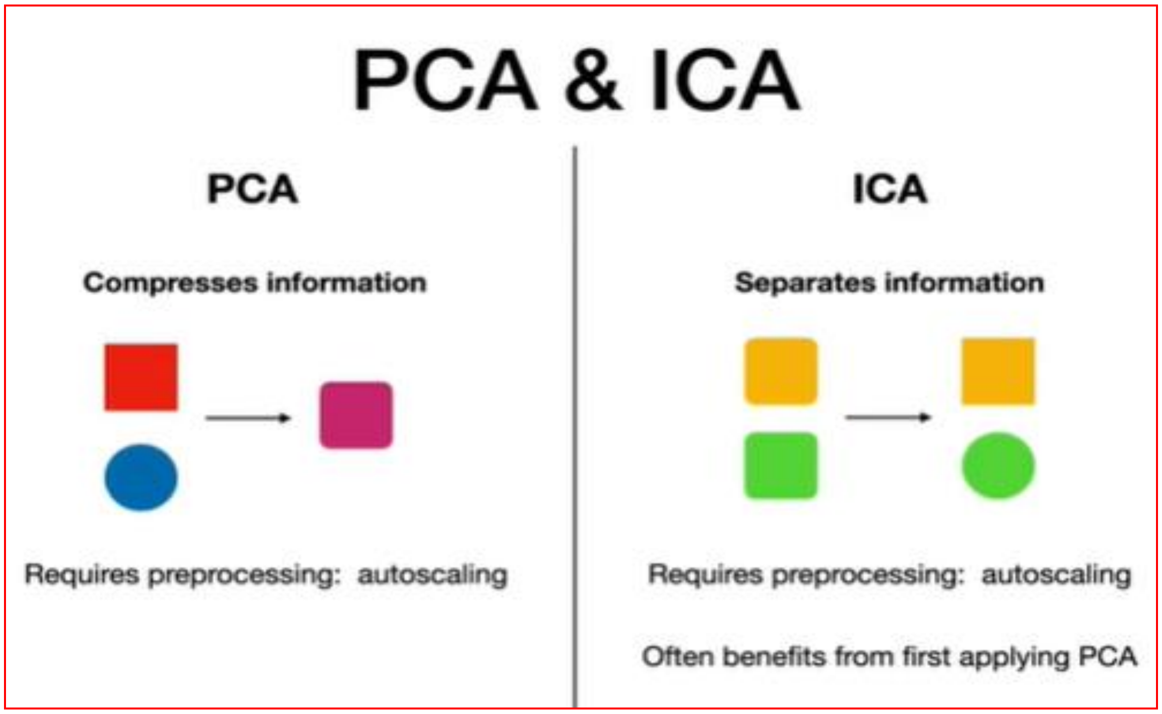


Identifiability means ICA does blind source separation

Observed signals:

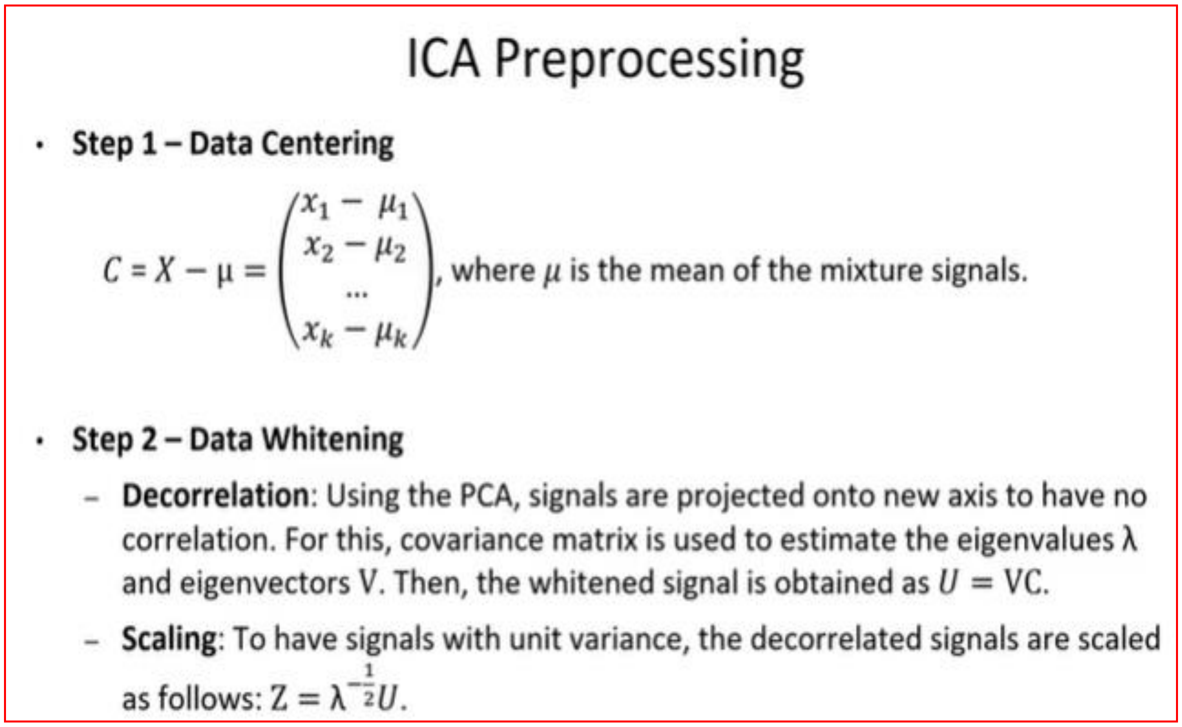
Principal components:

Independent components are original sources:



PCA & ICA

PCA	ICA
Compresses information	Separates information
Requires preprocessing: autoscaling	Requires preprocessing: autoscaling
	Often benefits from first applying PCA

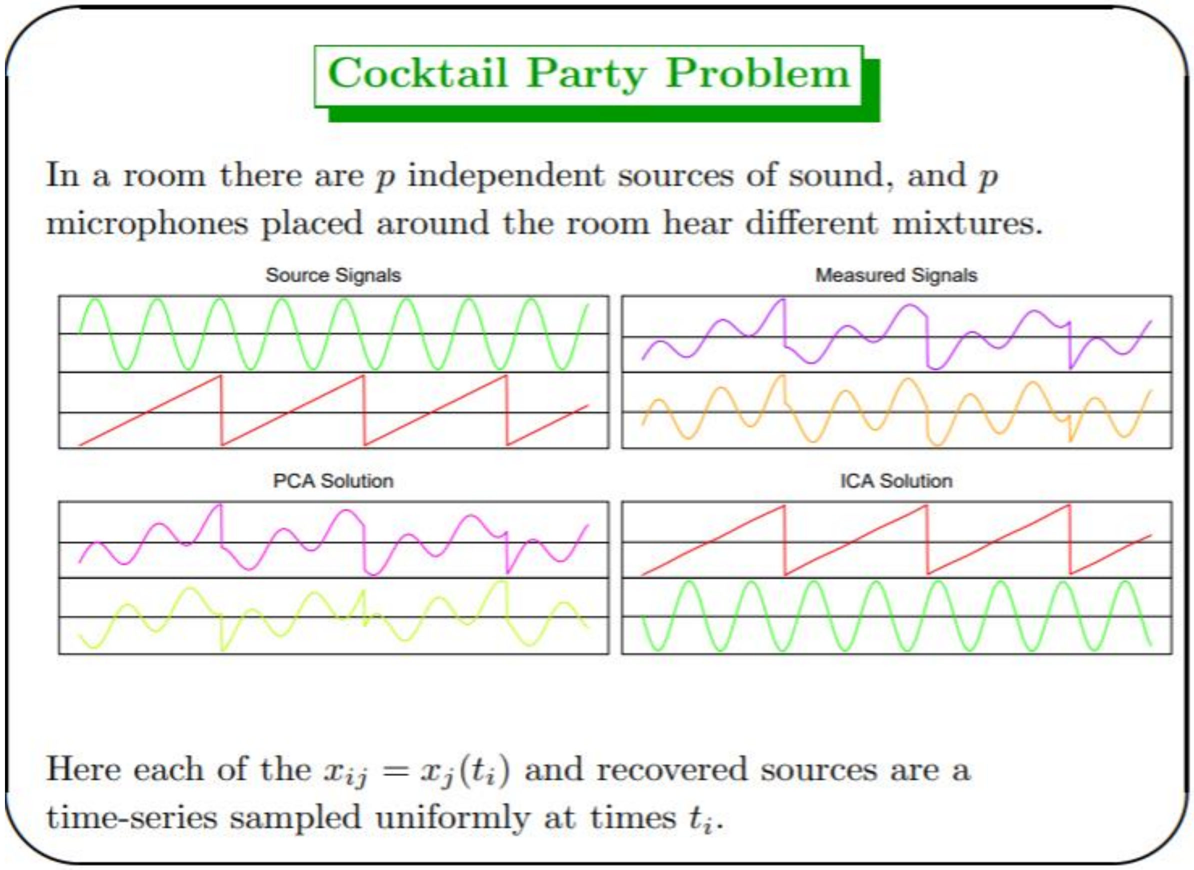


ICA Preprocessing

- **Step 1 – Data Centering**

$$C = X - \mu = \begin{pmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \\ \dots \\ x_k - \mu_k \end{pmatrix}$$, where μ is the mean of the mixture signals.

- **Step 2 – Data Whitening**
  - **Decorrelation**: Using the PCA, signals are projected onto new axis to have no correlation. For this, covariance matrix is used to estimate the eigenvalues λ and eigenvectors V. Then, the whitened signal is obtained as $U = VC$.
  - **Scaling**: To have signals with unit variance, the decorrelated signals are scaled as follows: $Z = \lambda^{-\frac{1}{2}} U$.


Source Signals
Measured Signals
PCA Solution
ICA Solution

Cocktail Party Problem

In a room there are p independent sources of sound, and p microphones placed around the room hear different mixtures.

Here each of the $x_{ij} = x_j(t_i)$ and recovered sources are a time-series sampled uniformly at times t_i.

- At a cocktail party, there are lots of sounds going around you
  - Conversation you're apart of
  - Conversations you're not apart of
  - Background music
  - Noises from outside

Our brain do a really good job of filtering out the background noise and focusing on just one sound, but how would a computer do this?

Assumptions

- Independent sources of sound
- Non-Gaussian sources
- As many signals as there are sources
- Each signal is a linear combinations of the sources

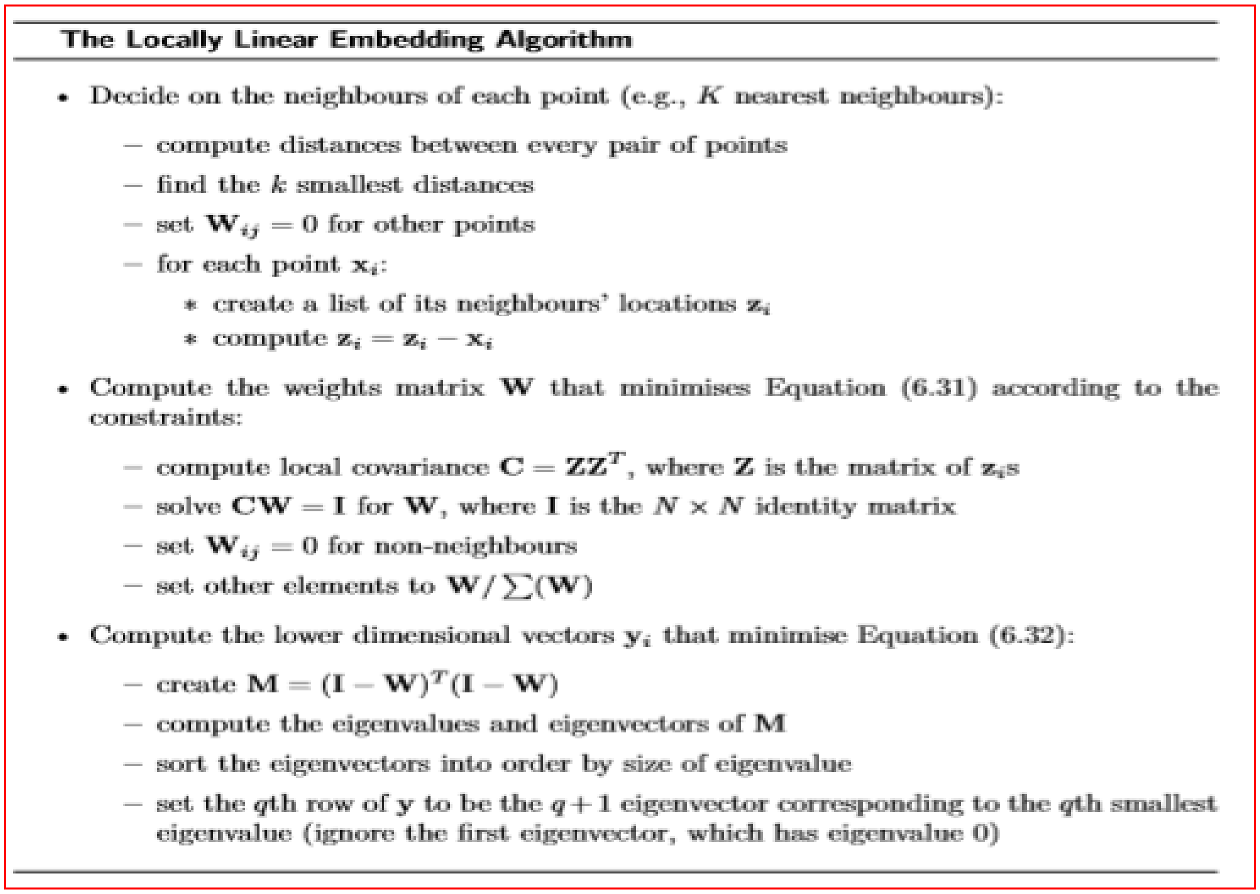
Audio recorded on a microphone=signal

Locally Linear Embedding

- The first tries to approximate the data by sticking together sets of locally flat patches that cover the dataset, while the second uses the shortest distances (geodesics) on the non-linear space to find a globally optimal solution.
- The locally linear algorithm, which is called Locally Linear Embedding (LLE). It was introduced by Roweis and Saul in 2000. The idea is that making linear approximations will make some errors, so should make these errors as small as possible by making the patches small where there is lots of non-linearity in the data. The error is known as the reconstruction error and is simply the sum-of-squares of the distance between the original point and its reconstruction**:**

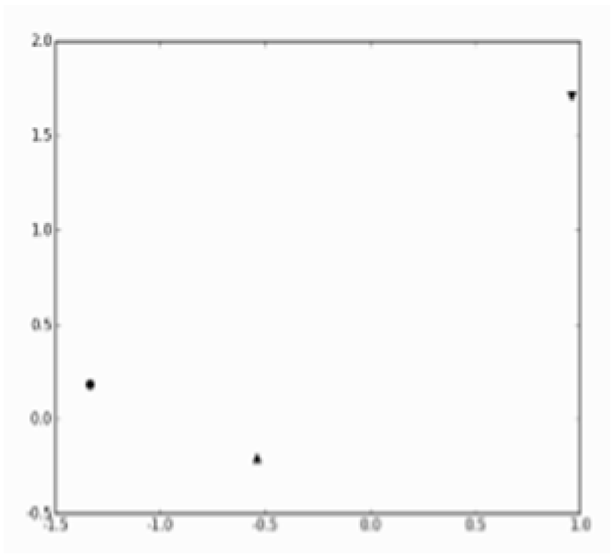
$$\epsilon = \sum_{i=1}^{N} \left(\mathbf{x}_i - \sum_{j=1}^{N} \mathbf{W}_{ij} \mathbf{x}_j \right)^2 .$$

- **There are two common ways to create neighbourhoods:**
  - Points that are less than some predefined distanced to the current point are neighbours (so don’t know how many neighbours there are, but they are all close)
  - The k nearest points are neighbours (so e know how many there are, but some could be far away)



The Locally Linear Embedding Algorithm

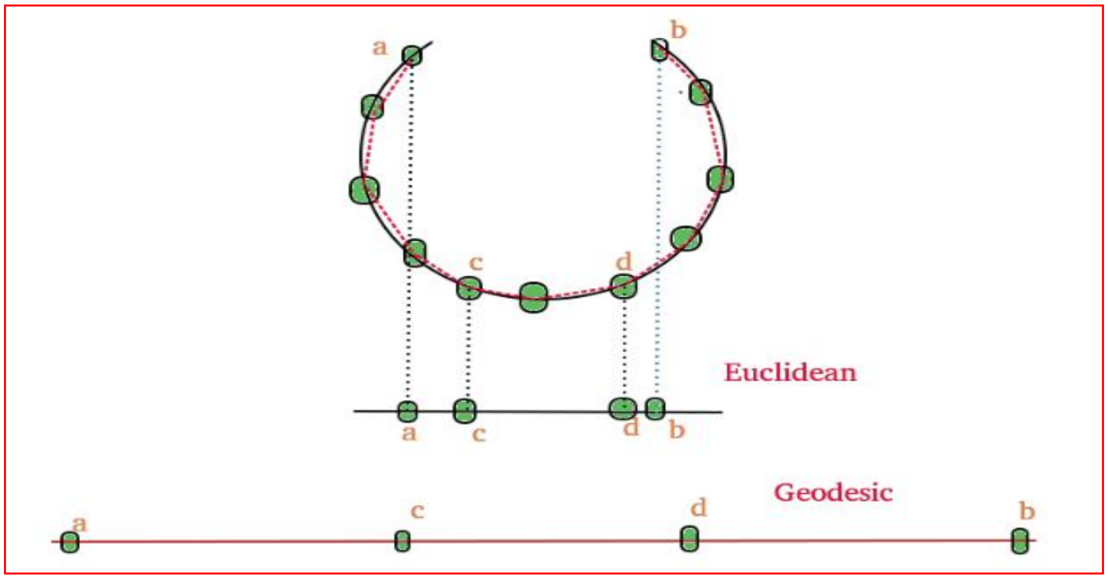
- Decide on the neighbours of each point (e.g., K nearest neighbours):
  - compute distances between every pair of points
  - find the k smallest distances
  - set $\mathbf{W}_{ij} = 0$ for other points
  - for each point $\mathbf{x}_i$:
    - create a list of its neighbours' locations $\mathbf{z}_i$
    - compute $\mathbf{z}_i = \mathbf{z}_i - \mathbf{x}_i$
- Compute the weights matrix $\mathbf{W}$ that minimises Equation (6.31) according to the constraints:
  - compute local covariance $\mathbf{C} = \mathbf{Z}\mathbf{Z}^T$, where $\mathbf{Z}$ is the matrix of $\mathbf{z}_i$s
  - solve $\mathbf{CW} = \mathbf{I}$ for $\mathbf{W}$, where $\mathbf{I}$ is the $N \times N$ identity matrix
  - set $\mathbf{W}_{ij} = 0$ for non-neighbours
  - set other elements to $\mathbf{W}/\sum(\mathbf{W})$
- Compute the lower dimensional vectors $\mathbf{y}_i$ that minimise Equation (6.32):
  - create $\mathbf{M} = (\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W})$
  - compute the eigenvalues and eigenvectors of $\mathbf{M}$
  - sort the eigenvectors into order by size of eigenvalue
  - set the qth row of $\mathbf{y}$ to be the $q+1$ eigenvector corresponding to the qth smallest eigenvalue (ignore the first eigenvector, which has eigenvalue 0)



The above diagram is the Locally linear embedding(LLE) algorithm with k-12 neighbours transforms the iris dataset into three points separating the data perfectly. The LLE produces a very interesting result on the iris dataset. It separates the three groups into three points.

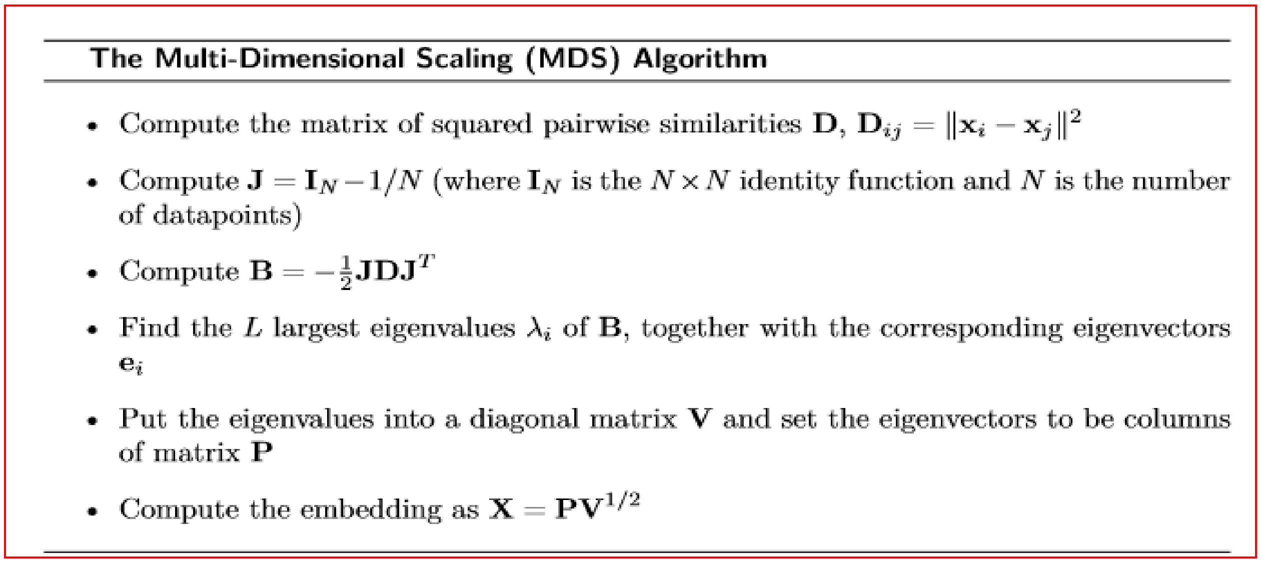
ISOMAP

- Isomap stands for isometric mapping.
- Isomap is a non-linear dimensionality reduction method based on the spectral theory which tries to preserve the geodesic distances in the lower dimension.
- Isomap starts by creating a neighborhood network.
- After that, it uses graph distance to the approximate geodesic distance between all pairs of points.



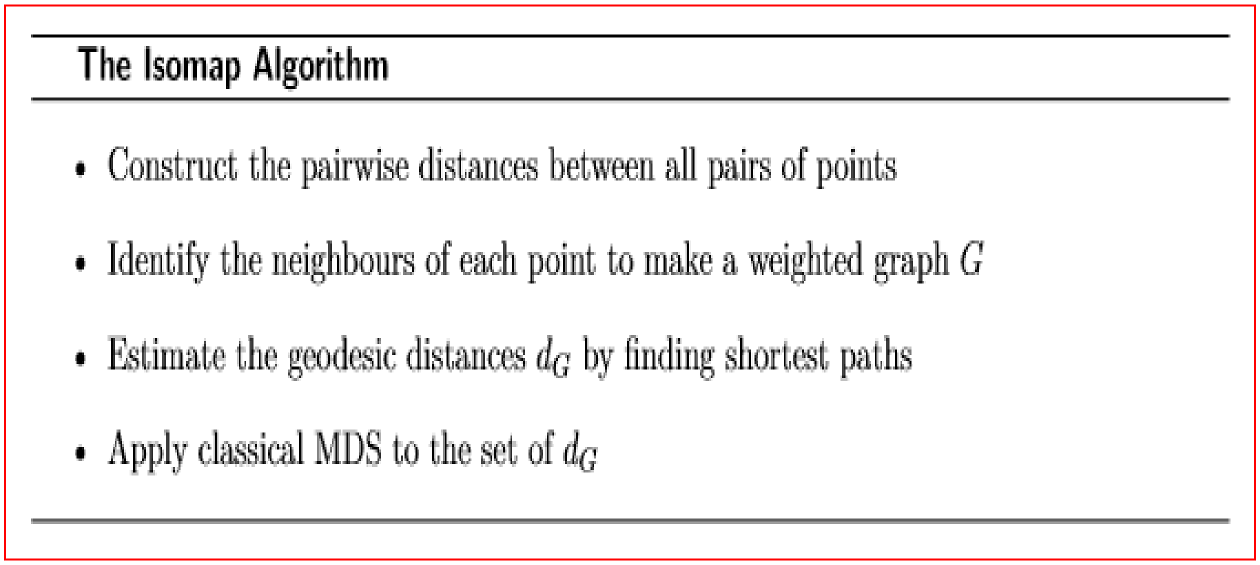
Multi-Dimensional Scaling (MDS)

- Like PCA, MDS tries to find a linear approximation to the full dataspace that embeds the data into a lower dimensionality.
- In the case of MDS the embedding tries to preserve the distances between all pairs of points



The Multi-Dimensional Scaling (MDS) Algorithm

- Compute the matrix of squared pairwise similarities $\mathbf{D}$, $\mathbf{D}_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$
- Compute $\mathbf{J} = \mathbf{I}_N - 1/N$ (where $\mathbf{I}_N$ is the $N \times N$ identity function and N is the number of datapoints)
- Compute $\mathbf{B} = -\frac{1}{2}\mathbf{J}\mathbf{D}\mathbf{J}^T$
- Find the L largest eigenvalues λ_i of $\mathbf{B}$, together with the corresponding eigenvectors $\mathbf{e}_i$
- Put the eigenvalues into a diagonal matrix $\mathbf{V}$ and set the eigenvectors to be columns of matrix $\mathbf{P}$
- Compute the embedding as $\mathbf{X} = \mathbf{P}\mathbf{V}^{1/2}$

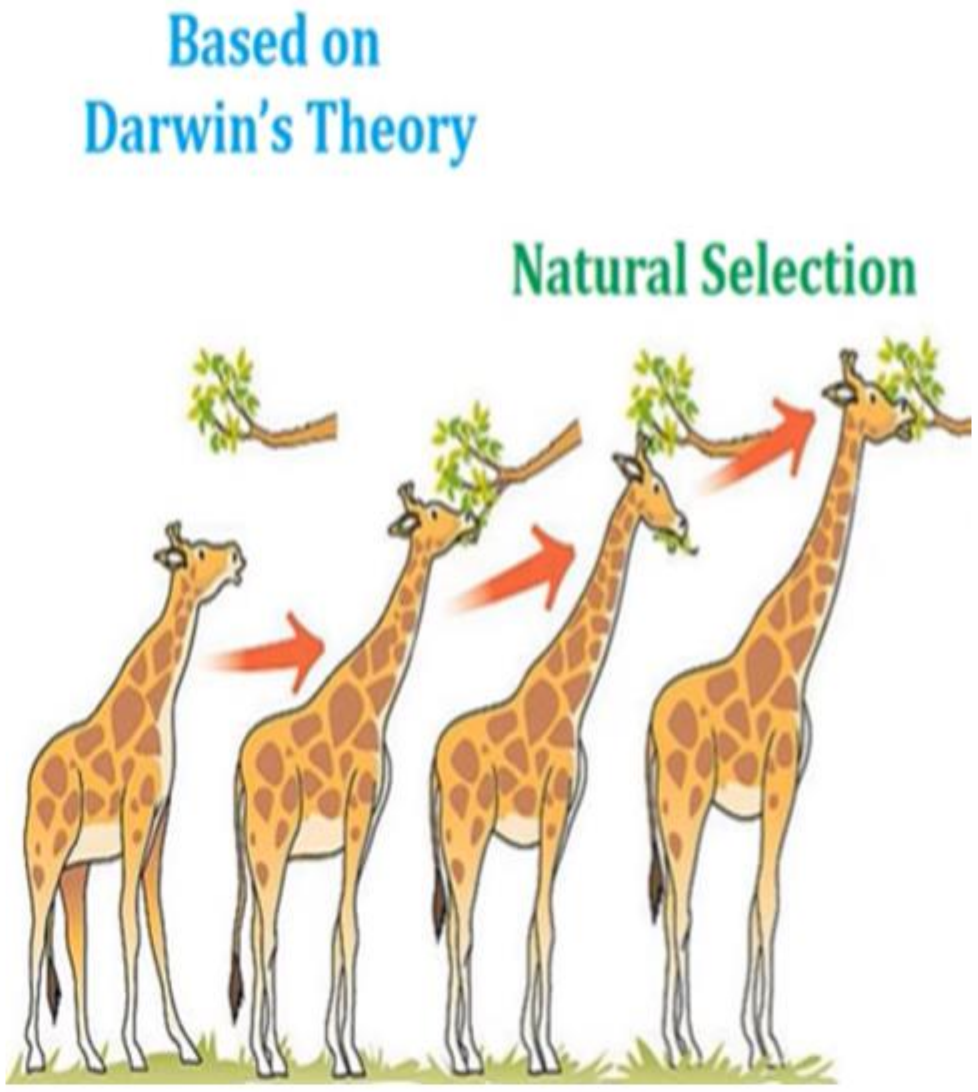


The Isomap Algorithm

- Construct the pairwise distances between all pairs of points
- Identify the neighbours of each point to make a weighted graph G
- Estimate the geodesic distances d_G by finding shortest paths
- Apply classical MDS to the set of d_G

4.3 Evolutionary Learning-Genetic algorithms - Genetic Offspring: - Genetic Operators-Using Genetic Algorithms

- Genetic Algorithm- uses concept from evolutionary Biology(Natural Genetics & Natural selection).
- John Holland introduced in 1975
- Populations of possible solutions to the given problem.
- It is a search based optimization technique



Principle of natural selection "**Select the best, discard the rest**"

- Genetic Algorithms are the heuristic search and optimization techniques that mimic the process of natural evolution.
- Thus genetic algorithms implement the optimization strategies by simulating evolution of species through natural selection

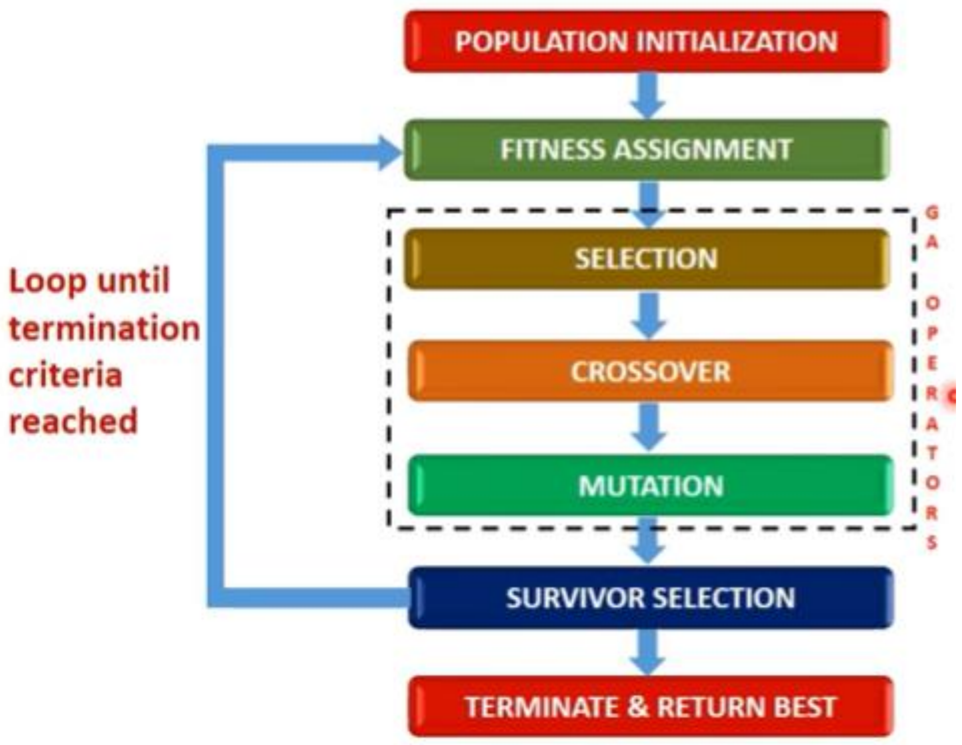
Applications

- DNA analysis
- Robotics
- Game playing

- Business
- Machine learning
- Image processing
- Vehicle Routing
- Neural Network

Genetic Algorithm

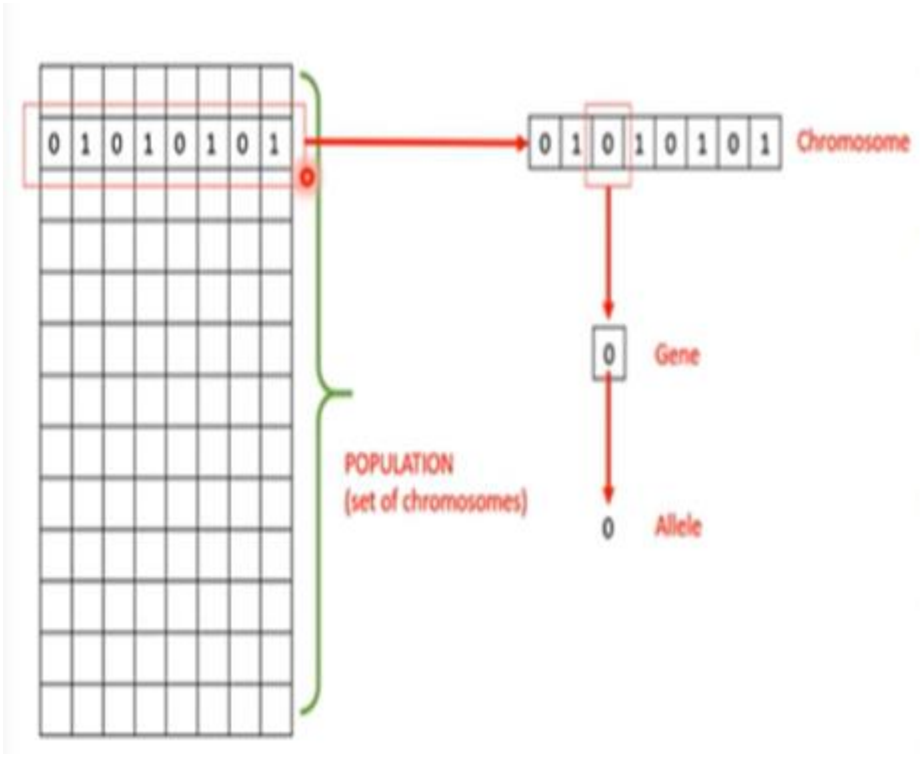
- Each iteration in the cycle produces a new generation of chromosomes
- The entire set of chromosomes is called a run
- Typical GA run is from 50 to 500 or more generations
- At the end of a run often there is at least one highly fit chromosome in the population

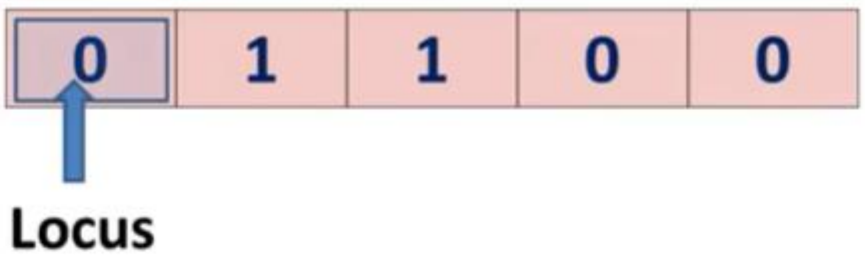


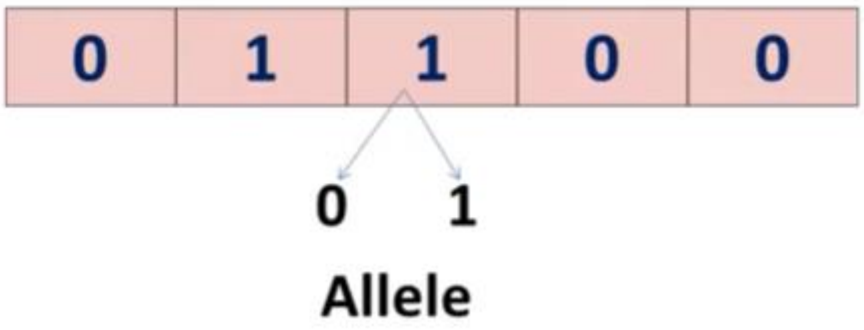
Basic Terminology of GA

- Population- subset of all the possible solutions to the given problem

- Chromosomes-one such solution to given problem
- Gene- one element position of a chromosome
- Allele- value a gene takes for particular chromosome
- Genotype- population in the computation space.
- Phenotype-population in the actual real world solution space
- Decoding- transforming a solution from the genotype to the phenotype space
- Encoding- transforming from the phenotype to genotype space







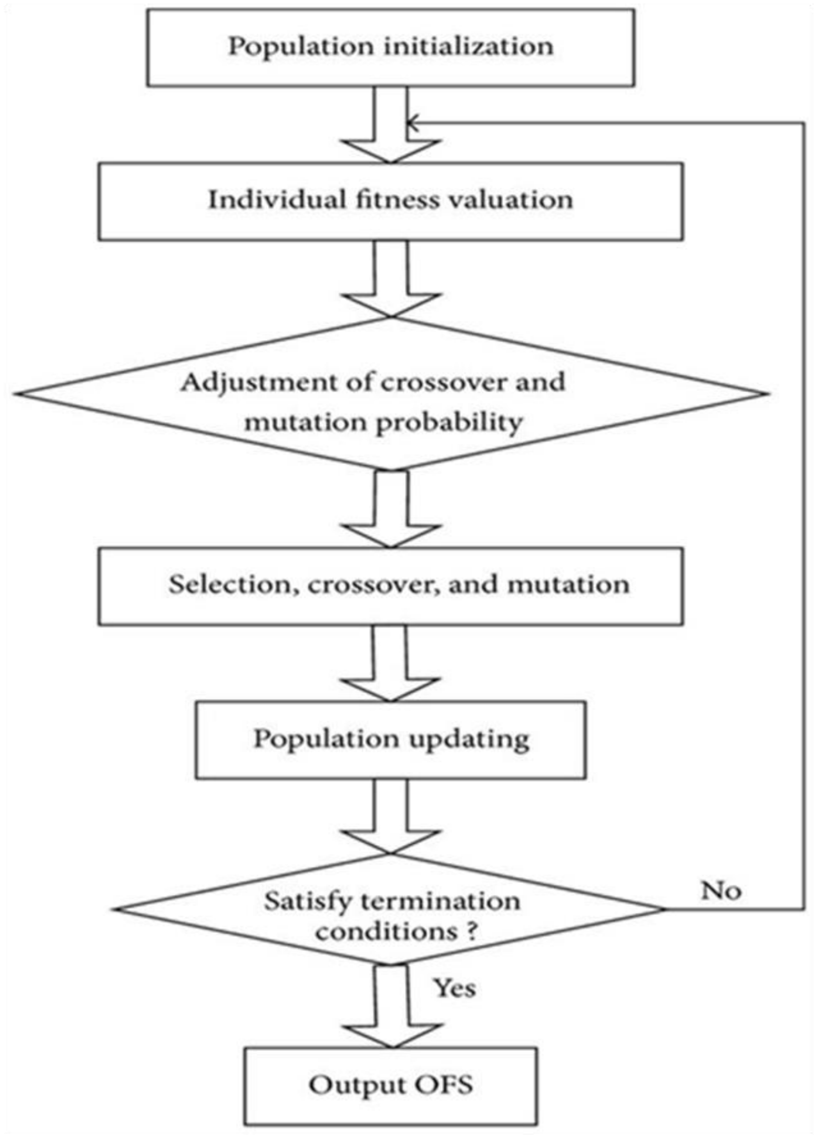
Genotype and Phenotype

Genotype- refers to the different pairing of alleles

- Example:
    - B- black hair
    - b-brown hair
    - Bb,bB,BB,bb- Genotype

Phenotype-refers to the characters of the trait

- Example
    - BB,bB-black hair
    - Bb-brown hair
    - Black hair,Brown hair-Phenotype


Population initialization
Individual fitness valuation
Adjustment of crossover and mutation probability
Selection, crossover, and mutation
Population updating
Satisfy termination conditions ?
No
Yes
Output OFS

Selection

The process that determines which solutions are to be preserved and allowed to reproduce and which ones deserve to die out.

The primary objective of the selection operator is to emphasize the good solutions and eliminate the bad solutions in a population while keeping the population size constant.

“Selects the best, discards the rest”

Functions of Selection operator

- Identify the good solutions in a population

- Make multiple copies of the good solutions Eliminate bad solutions from the population so that
- multiple copies of good solutions can be placed in the
- population
- Now how to identify the good solutions?
- There are different techniques to implement selection in Genetic Algorithms.
- They are:
    - Tournament selection
    - Roulette wheel selection
    - Proportionate selection
    - Rank selection
    - Steady state selection, *etc*

Tournament selection

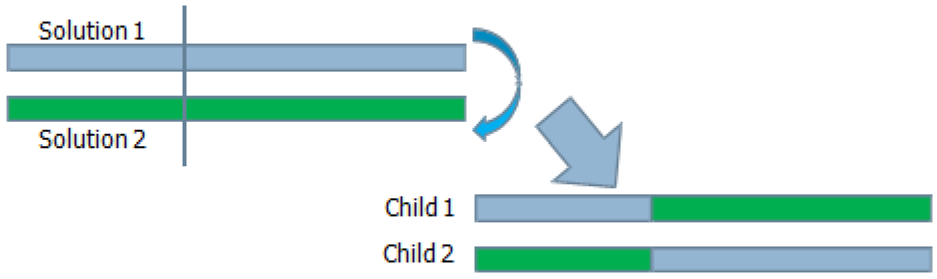
- In tournament selection several tournaments are played among a few individuals. The individuals are chosen at random from the population.
- The winner of each tournament is selected for next generation.
- Selection pressure can be adjusted by changing the tournament size.
- Weak individuals have a smaller chance to be selected if tournament size is large.

Fitness function

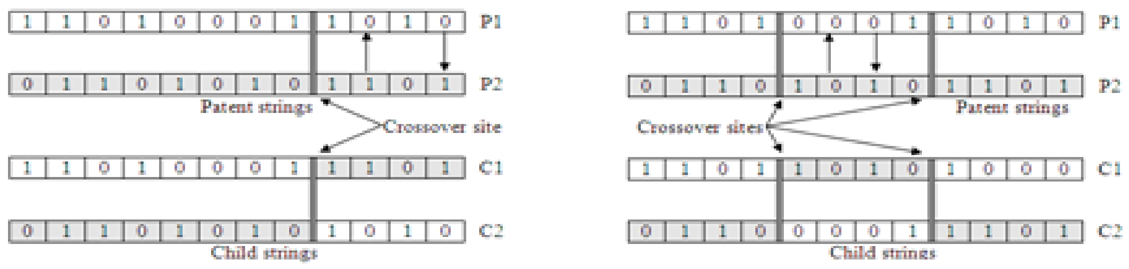
- A fitness value can be assigned to evaluate the solutions
- A fitness function value quantifies the optimality of a solution. The value is used to rank a particular solution against all the other solutions
- A fitness value is assigned to each solution depending on how close it is actually to the optimal solution of the problem

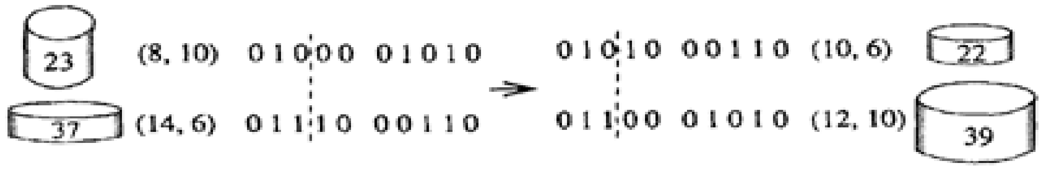
Crossover operator

- The most popular crossover selects any two solutions strings randomly from the mating pool and some portion of the strings is exchanged between the strings.
- The selection point is selected randomly.
- A probability of crossover is also introduced in order to give freedom to an individual solution string to determine whether the solution would go for crossover or not.



Binary Crossover





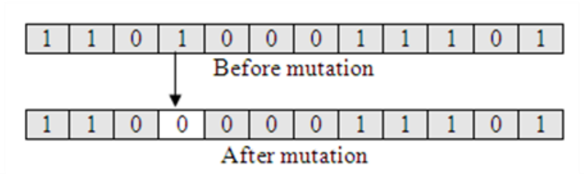
Mutation operator

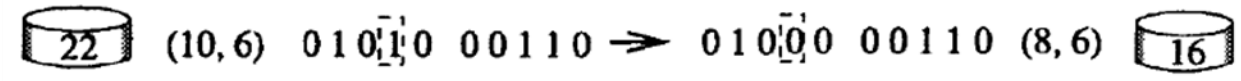
- Mutation is the occasional introduction of new features in to the solution strings of the population pool to maintain diversity in the population.
- Though crossover has the main responsibility to search for the optimal solution, mutation is also used for this purpose.

Binary Mutation

- Mutation operator changes a 1 to 0 or vise versa, with a mutation probability of .

- The mutation probability is generally kept low for steady convergence.
- A high value of mutation probability would search here and there like a random search technique





Example

Maximize the function $f(x)=x^2$ over the range of integers from 0 …31

1. Devise a means to represent a solution to the problem:

   Assume we represent x with five-digit unsigned binary integers.

2. Devise a heuristic for evaluating the fitness of any particular solution:

   The function f(x) is simple, so it is easy to use the f(x) value itself to rate the fitness of a solution; else we might have considered a more simpler heuristic that would more or less serve the same purpose.

3. Coding-Binary and the string length:

   GAs often process binary representations of solutions. This works well, because crossover and mutation can be clearly defined for binary solutions. A binary string of length 5 can represent 32 numbers (0 to 31)

4. Randomly generate a set of solutions:

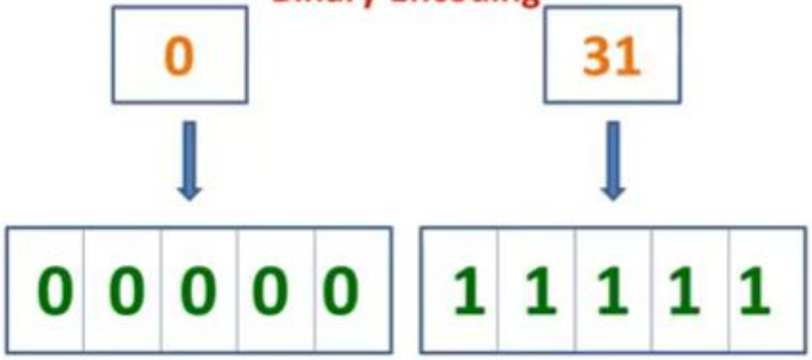
   Here, considered a population of four solutions. However, large populations are used in real applications to explore a larger part of the search. Assume four randomly generated solutions as :01101,11000,01000,10011. These are chromosomes or genotypes.

Genetic Algorithm

Maximize the function $f(x)=x^2$, where x value range from 0-31

- Step 1: Choose a encoding technique

Binary Encoding



- Step 2: Choose the population size

$n = 4$

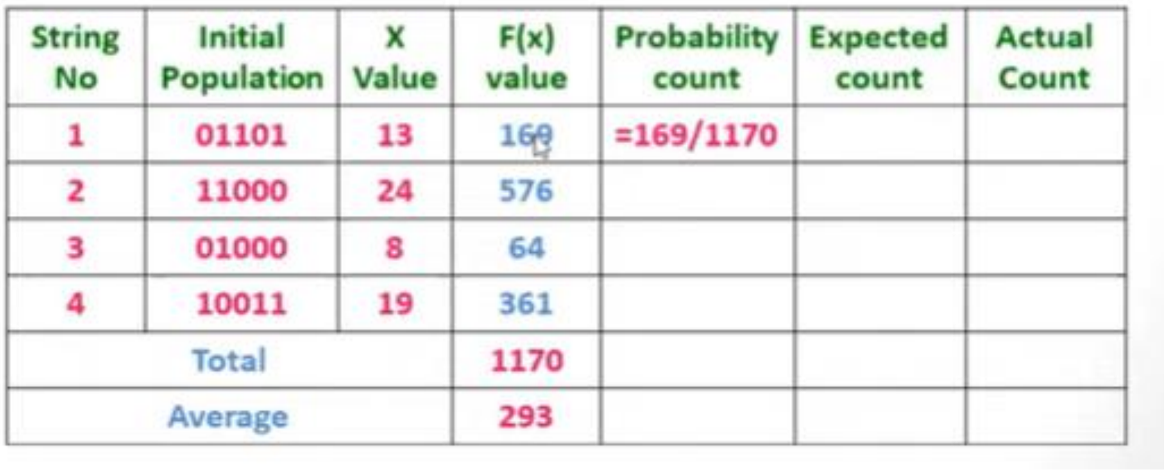
- Step 3: Randomly choose initial population

13, 24 , 8, 19

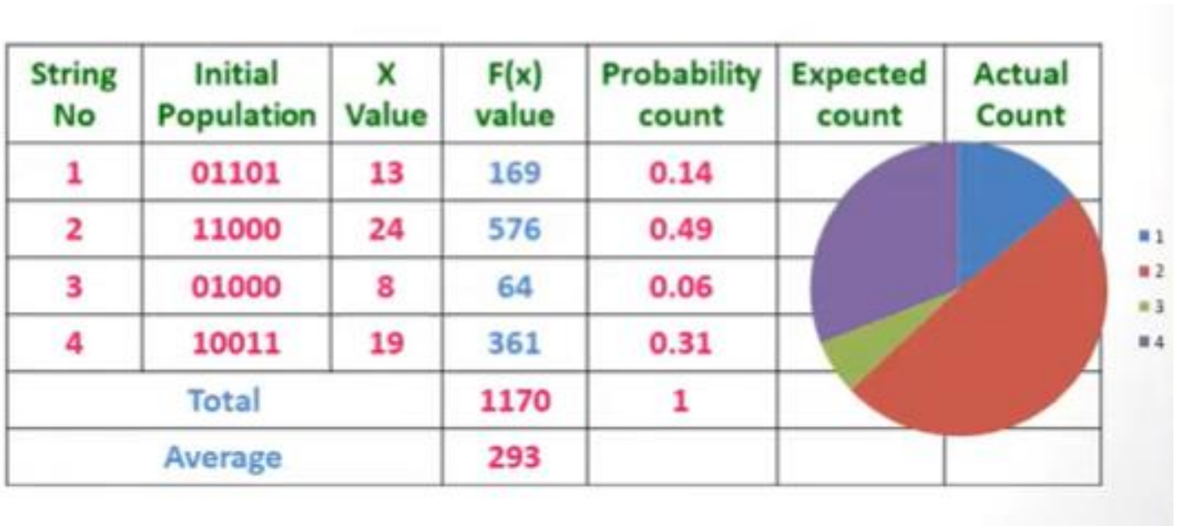
- Step 4: Select parental chromosomes

Roulette wheel selection method

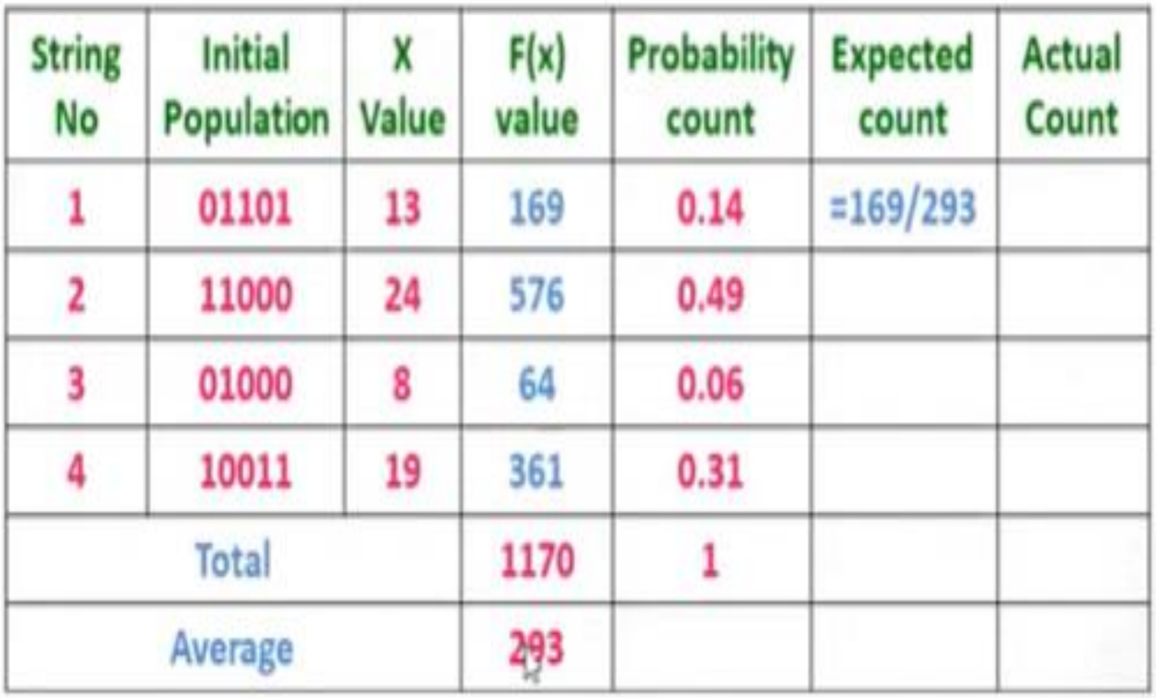
Roulette wheel selection method



String No	Initial Population	X Value	F(x) value	Probability count	Expected count	Actual Count
1	01101	13	169	=169/1170		
2	11000	24	576			
3	01000	8	64			
4	10011	19	361			
Total			1170			
Average			293			

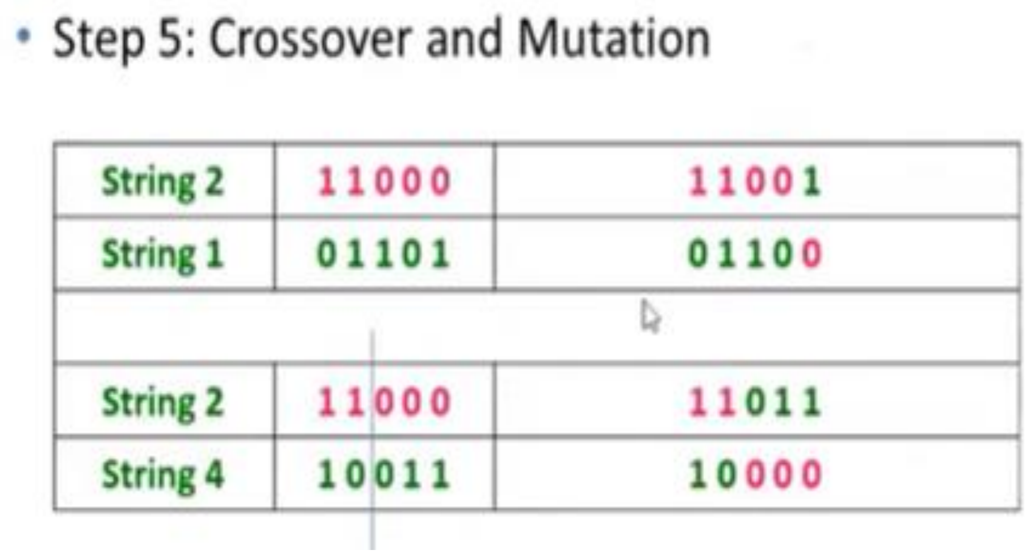


String No	Initial Population	X Value	F(x) value	Probability count	Expected count	Actual Count
1	01101	13	169	0.14		
2	11000	24	576	0.49		
3	01000	8	64	0.06		
4	10011	19	361	0.31		
Total			1170	1		
Average			293			



String No	Initial Population	X Value	F(x) value	Probability count	Expected count	Actual Count
1	01101	13	169	0.14	=169/293	
2	11000	24	576	0.49		
3	01000	8	64	0.06		
4	10011	19	361	0.31		
Total			1170	1		
Average			293			

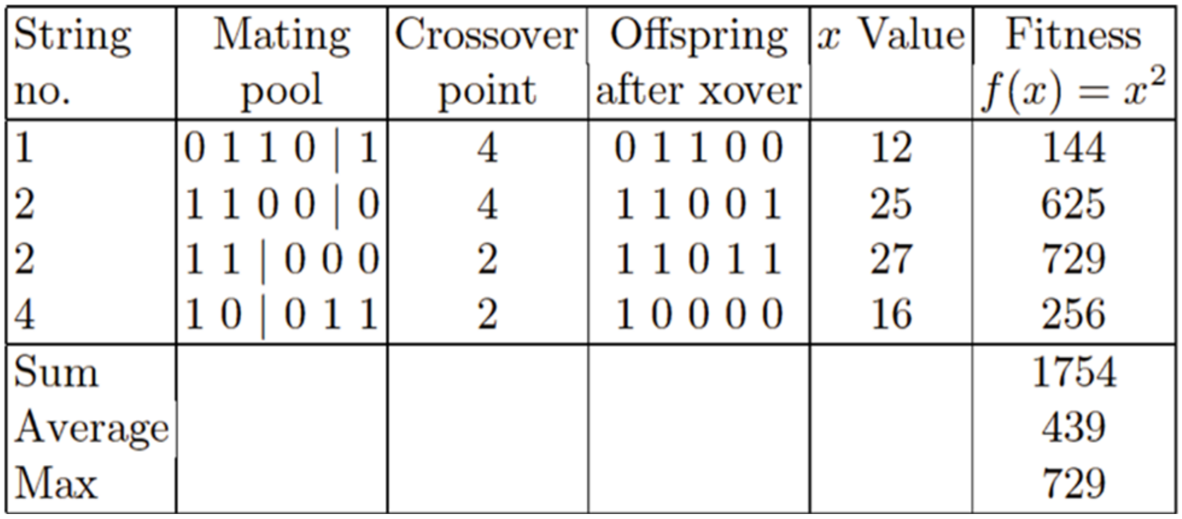
String no.	Initial population	x Value	Fitness $f(x) = x^2$	$Prob_i$	Expected count	Actual count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2



- Step 5: Crossover and Mutation

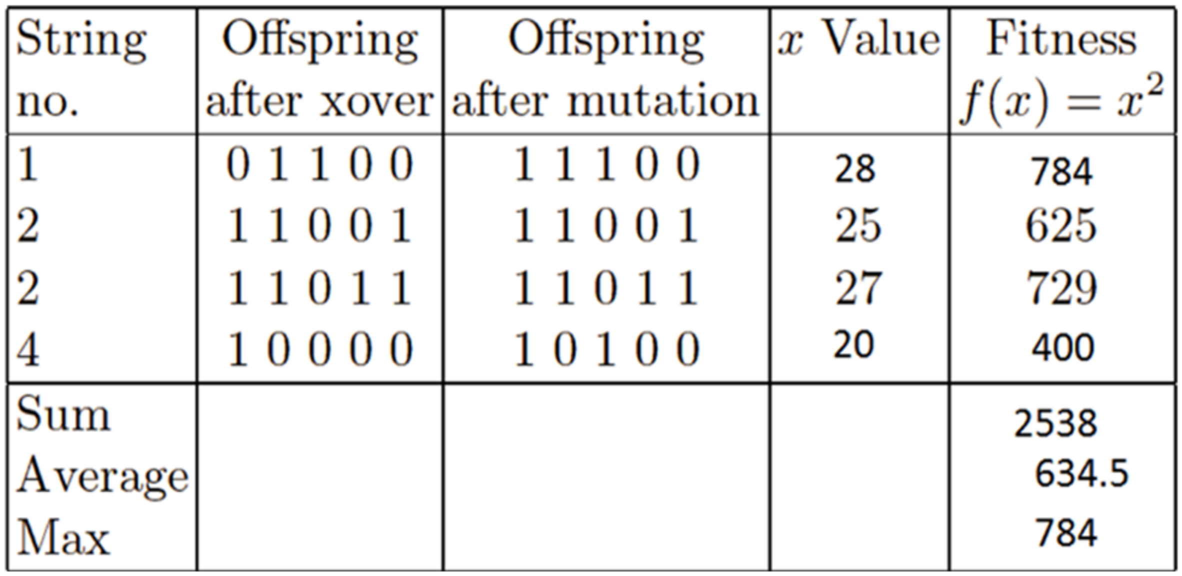
String 2	11000	11001
String 1	01101	01100
String 2	11000	11011
String 4	10011	10000

Cross over operator



String no.	Mating pool	Crossover point	Offspring after xover	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 | 1	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	4	1 1 0 0 1	25	625
2	1 1 | 0 0 0	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

Mutation operator



String no.	Offspring after xover	Offspring after mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	1 1 1 0 0	28	784
2	1 1 0 0 1	1 1 0 0 1	25	625
2	1 1 0 1 1	1 1 0 1 1	27	729
4	1 0 0 0 0	1 0 1 0 0	20	400
Sum				2538
Average				634.5
Max				784

Advantages of Genetic Algorithm

- Does not require any derivative information which may not be available for many real world problems
- Faster and more efficient as compared to the traditional methods
- Optimizes both continuous and discrete functions and also multi objective problems
- Provides a list of 'good' solutions and not just a single solution. Always gets an answer which gets better over the time.
- Useful when the search space is very large and there are a number of parameters involved.

4.4 Reinforcement Learning-Overview -Markov Decision Process

Reinforcement learning

Close to human learning.

Algorithm learns a policy of how to act in a given environment.

Every action has some impact in the environment, and the environment provides rewards that guides the learning algorithm.

Reinforcement Learning

Step: 1

- World: You are in state 9. Choose action A or C.

- Learner: Action A.
- World: Your reward is 100.

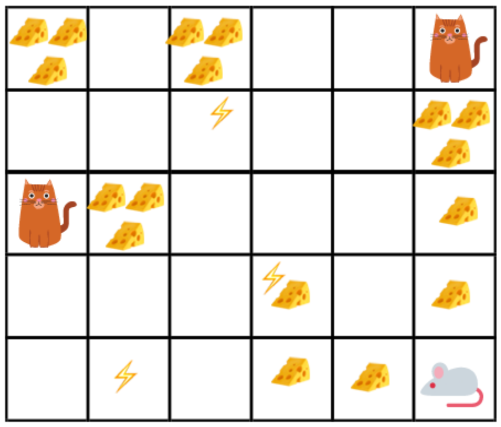
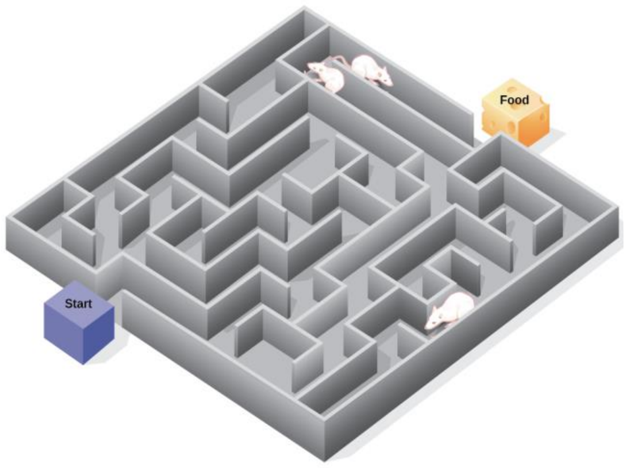
Step: 2

- World: You are in state 32. Choose action B or E.
- Learner: Action B.
- World: Your reward is 50.

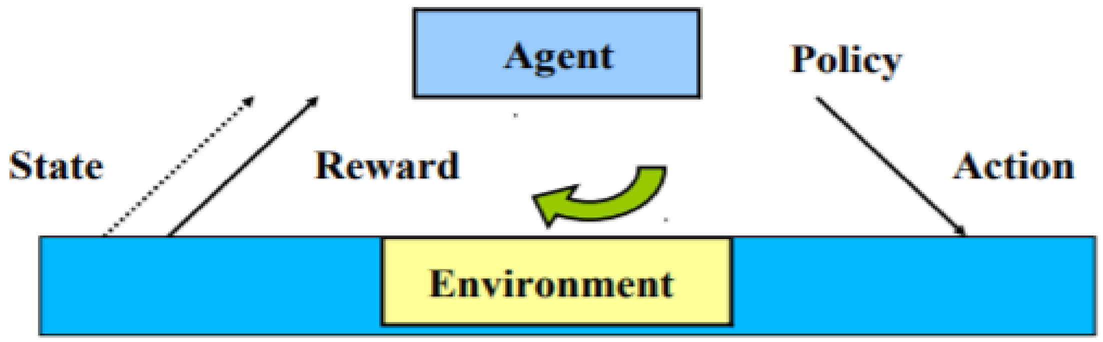
Step: 3 ....

- Meaning of Reinforcement:
  - Occurrence of an event, in the proper relation to a response, that tends to increase the probability that the response will occur again in the same situation.
- Reinforcement learning is the problem faced by an agent that learns behavior through trial-and-error interactions with a dynamic environment.
  - Reinforcement Learning is learning how to act in order to maximize a numerical reward

Examples of RL

Element of reinforcement learning

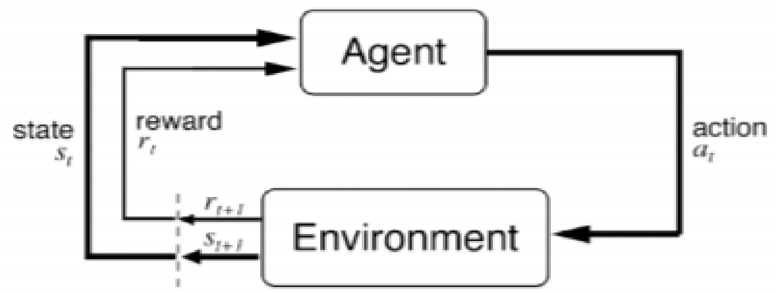


- **Agent**: Intelligent programs
- **Environment**: External condition
- **Policy**:
  - Defines the agent's behavior at a given time
  - A mapping from states to actions
  - Lookup tables or simple function

Element of reinforcement learning

- **Reward function** :
  - Defines the goal in an RL problem
  - Policy is altered to achieve this goal
- **Value function**:
  - Reward function indicates what is good in an immediate sense while a value function specifies what is good in the long run.
  - Value of a state is the total amount of reward an agent can expect to accumulate over the future, starting form that state.
- **Model of the environment** :
  - Predict mimic behavior of environment,
  - Used for planning & if Know current state and action then predict the resultant next state and next reward.

Agent –Environment Interface



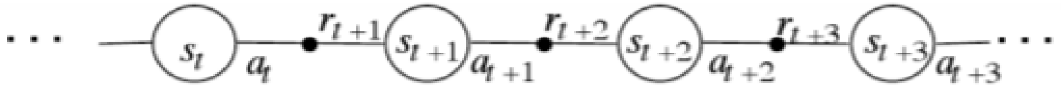
Agent and environment interact at discrete time steps : $t = 0, 1, 2, \ldots$

Agent observes state at step t: $s_t \in S$

produces action at step t: $a_t \in A(s_t)$

gets resulting reward : $r_{t+1} \in \Re$

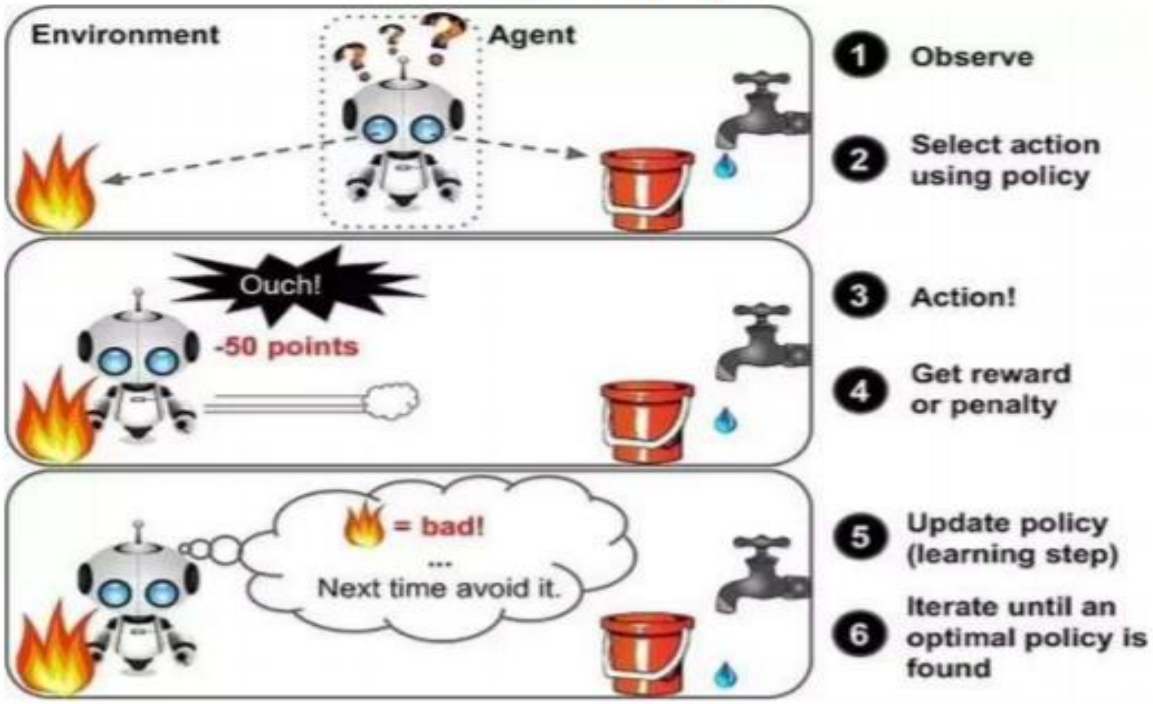
and resulting next state : s_{t+1}



Steps for Reinforcement Learning

1. The agent(The RL algorithm that learns from trial and error) observes an input state.
2. An action is determined by a decision making function (policy)
3. The action is performed. All the possible steps that the agent can take.
4. The agent receives a scalar reward or reinforcement from the environment(The world through which the agent moves).
5. Information about the reward given for that state / action pair is recorded.

Example (2)...

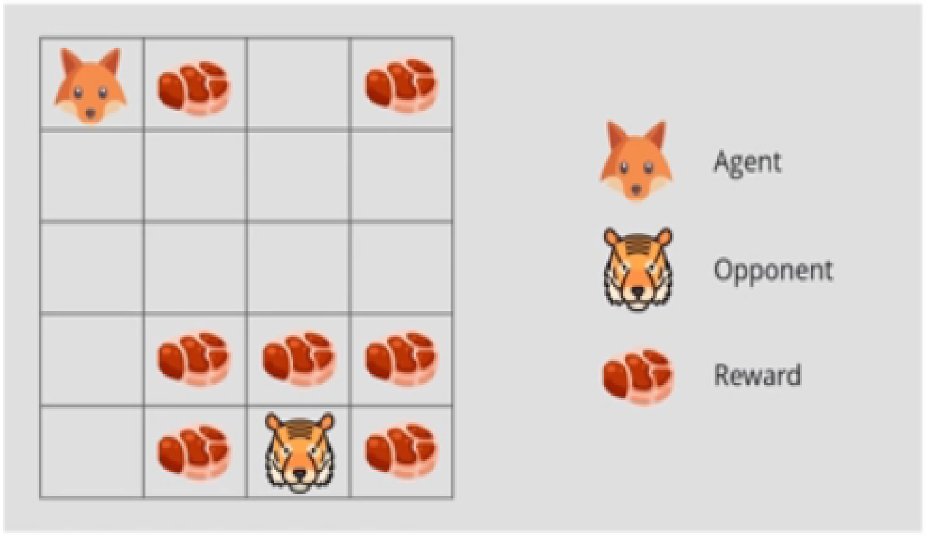

Environment
Agent
1 Observe
2 Select action using policy
Ouch!
-50 points
3 Action!
4 Get reward or penalty
= bad!
...
Next time avoid it.
5 Update policy (learning step)
6 Iterate until an optimal policy is found

Counter Strike Example

1. The RL Agent (Player1) collects state S^0 from the environment
2. Based on the state S^0, the RL agent takes an action A^0, initially the action is random
3. The environment is now in a new state S^1
4. RL agent now gets a reward R^1 from the environment
5. The RL loop goes on until the RL agent is dead or reaches the destination

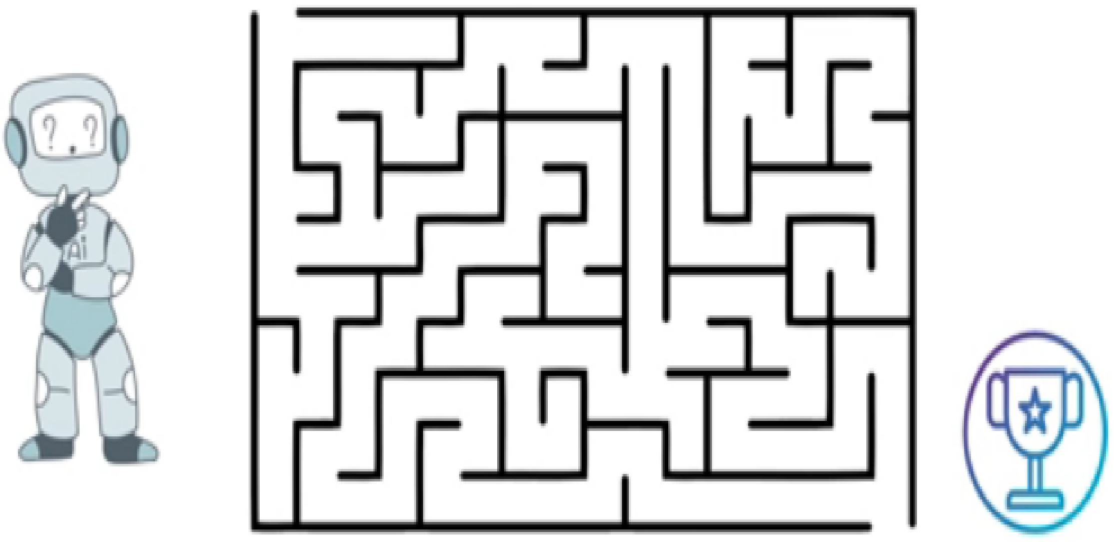
Reward Maximization

Reward maximization theory states that, *a RL agent must be trained in such a way that, he takes the best action so that the reward is maximum*.



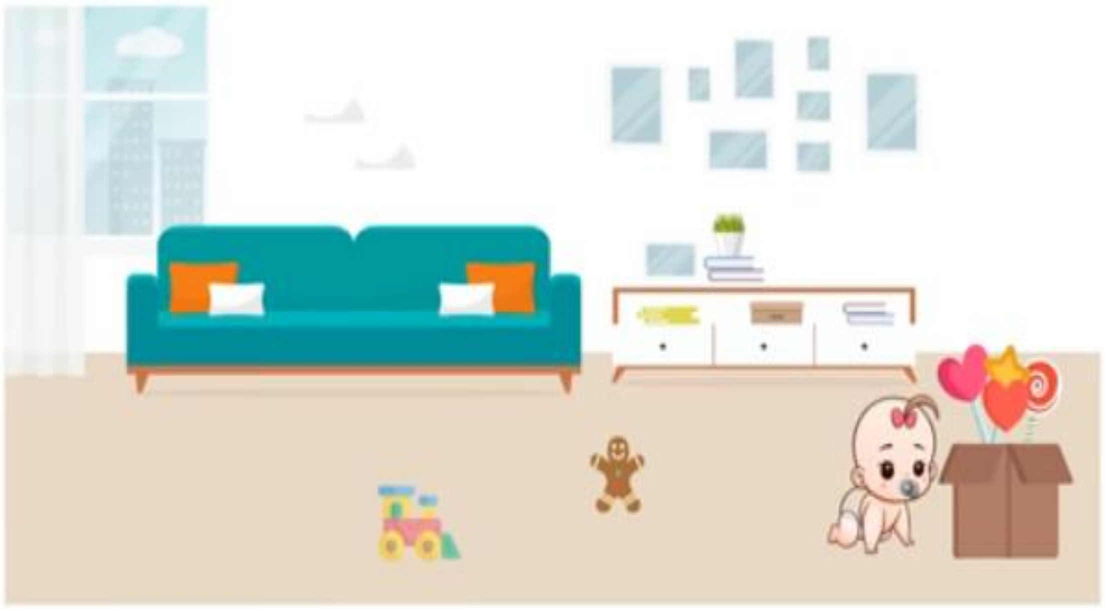
What Is Reinforcement Learning?

Reinforcement learning is a type of Machine Learning where an agent learns to behave in a environment by performing actions and seeing the results



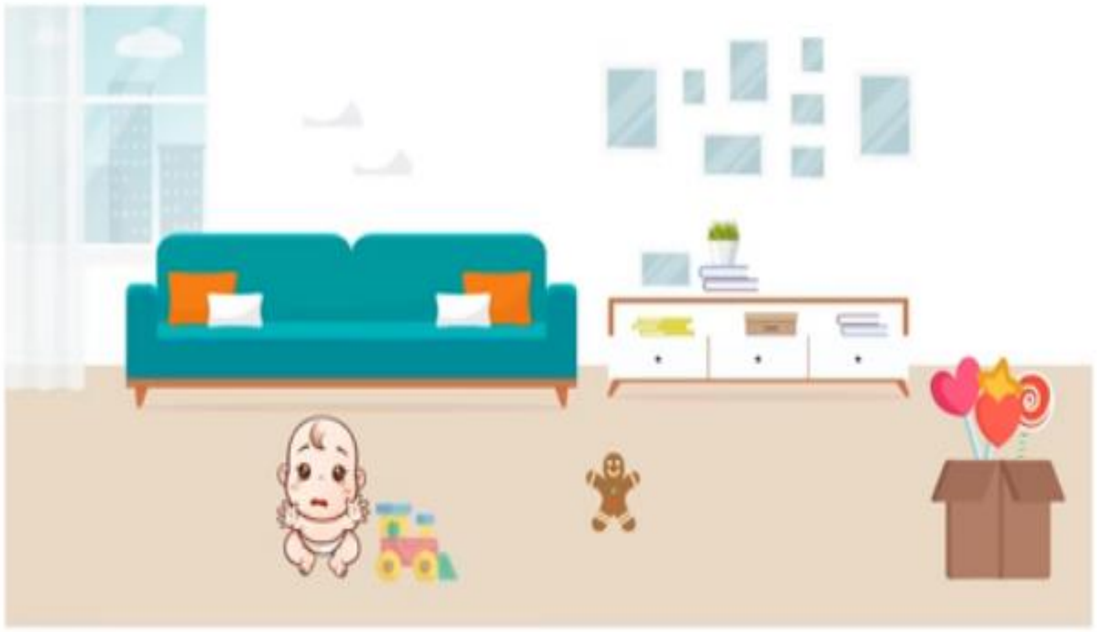
Reinforcement Learning With An Analogy

Scenario 1: Baby starts crawling and makes it to the candy



Reinforcement Learning With An Analogy

Scenario 2: Baby starts crawling but falls due to some hurdle in between



Example Getting lost

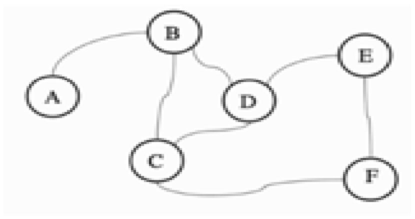
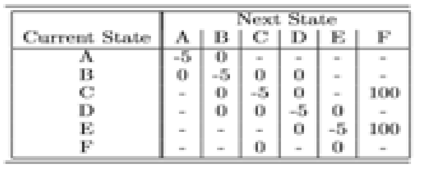


FIGURE 11.3 The old town that you find yourself lost in.



Current State	Next State A	B	C	D	E	F
A	-5	0	-	-	-	-
B	0	-5	0	0	-	-
C	-	0	-5	0	-	100
D	-	0	0	-5	0	-
E	-	-	-	0	-5	100
F	-	-	0	-	0	-

[illegible]

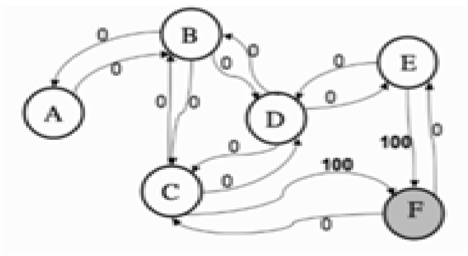
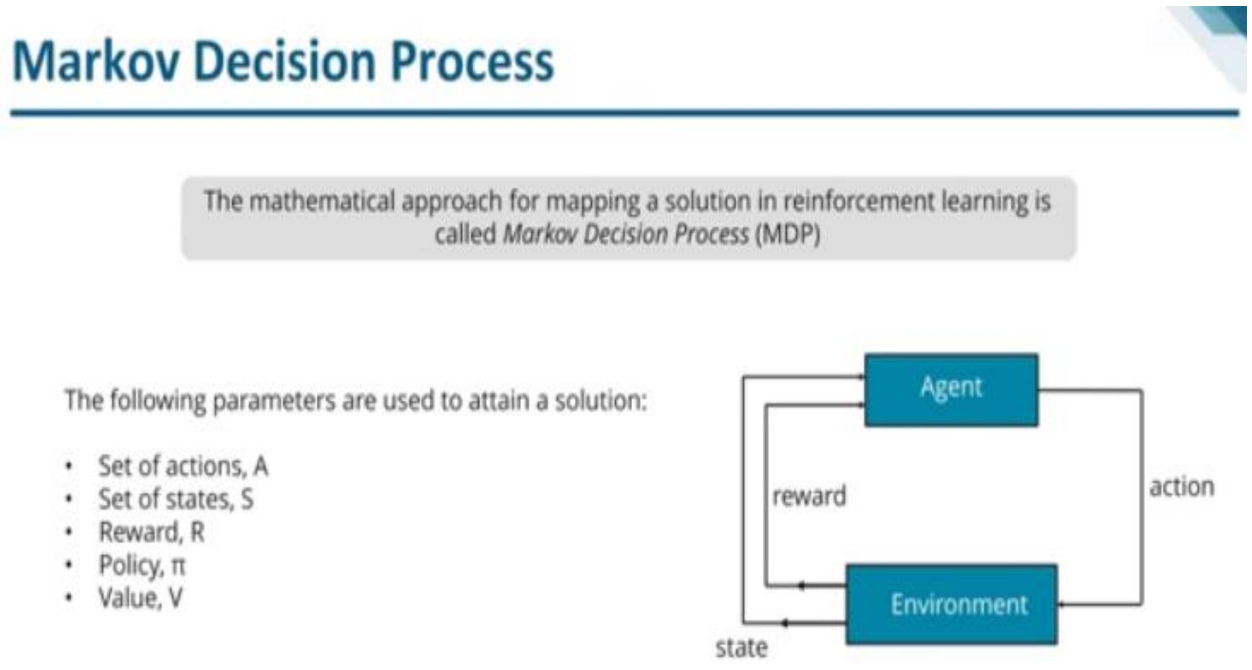


FIGURE 11.4 The *state diagram* if you are correct and the backpacker's is in square (state) F. The connections from each state back into itself (meaning that you don't move) are not shown, to avoid the figure getting too complicated. They are each worth −5 (except for staying in state F, which means that you are in the backpacker's).

Markov Decision Process(MDP) : Reinforcement learning deals with knowledge based on current state and its future prediction of next state with optimal way. The direction of this movement towards optimal solution is proposed by agent under MDP.



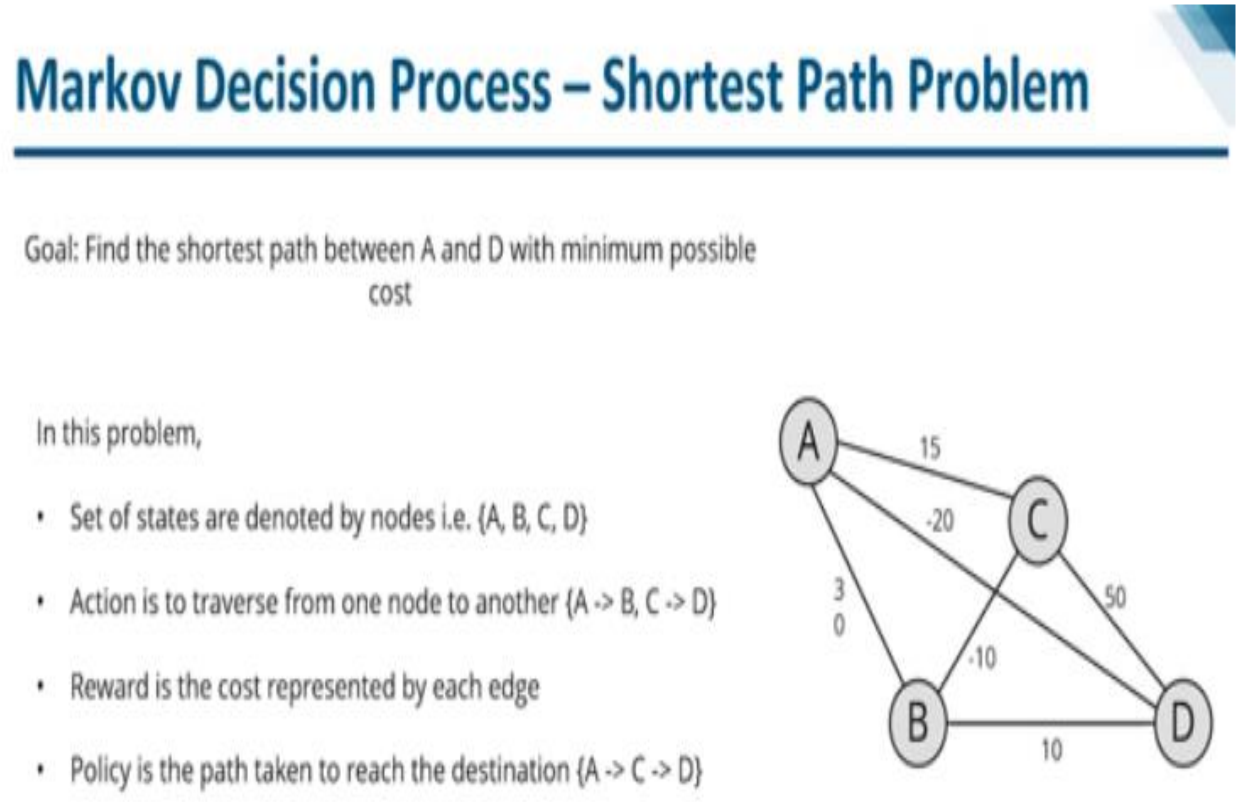
Markov Decision Process

The mathematical approach for mapping a solution in reinforcement learning is called *Markov Decision Process* (MDP)

The following parameters are used to attain a solution:

- Set of actions, A
- Set of states, S
- Reward, R
- Policy, π
- Value, V

Agent
reward
action
Environment
state



Markov Decision Process – Shortest Path Problem

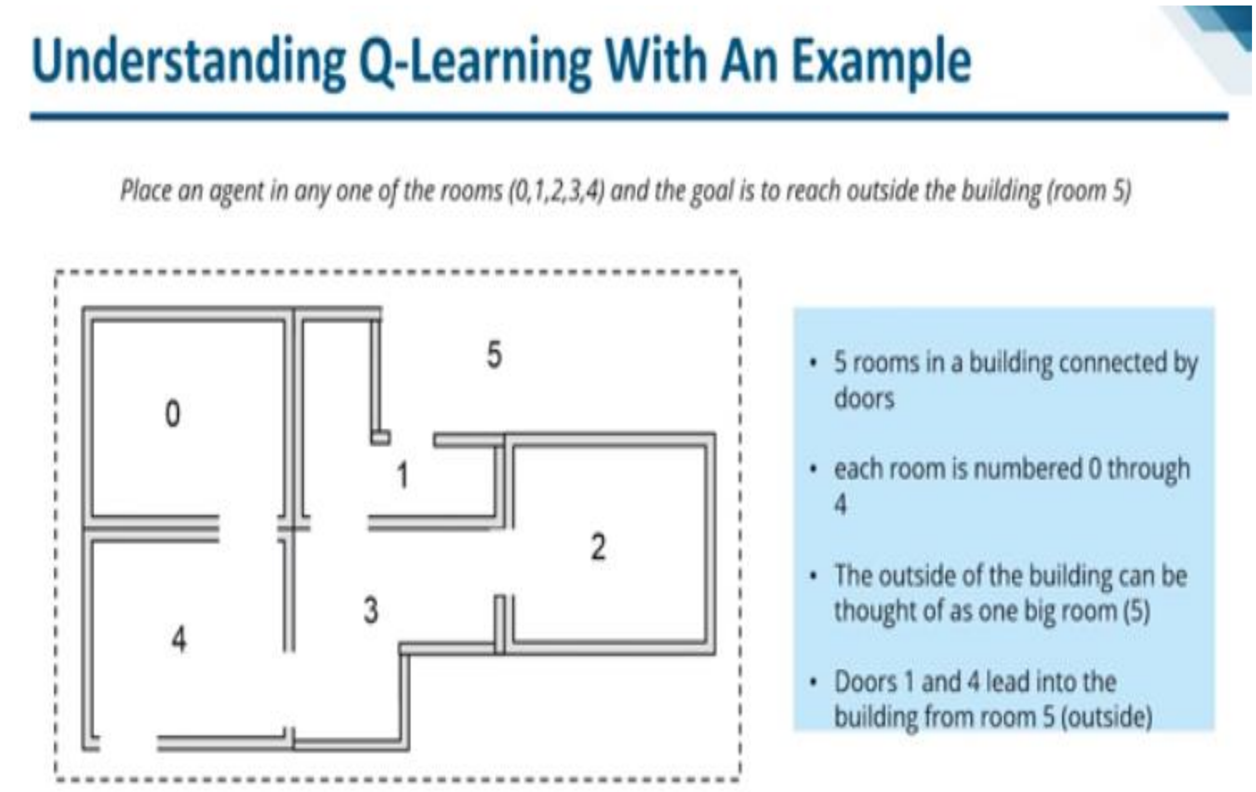
Goal: Find the shortest path between A and D with minimum possible cost

In this problem,

- Set of states are denoted by nodes i.e. {A, B, C, D}
- Action is to traverse from one node to another {A -> B, C -> D}
- Reward is the cost represented by each edge
- Policy is the path taken to reach the destination {A -> C -> D}

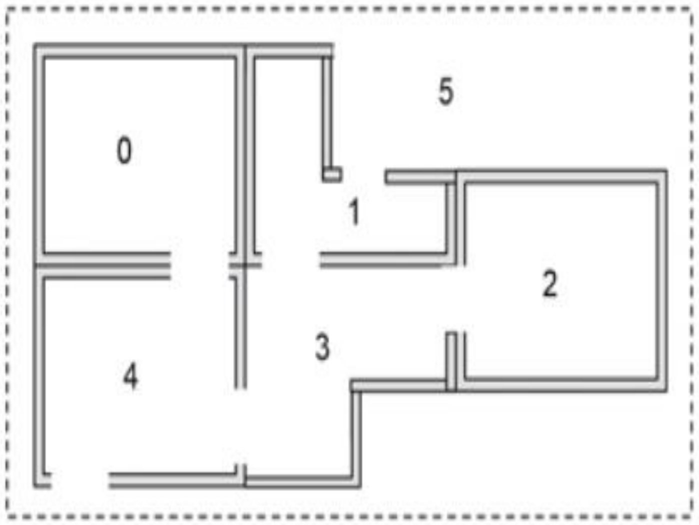
A
15
C
-20
3
0
50
-10
B
10
D

Q-learning is a model-free reinforcement learning algorithm to learn the value of an action in a particular state



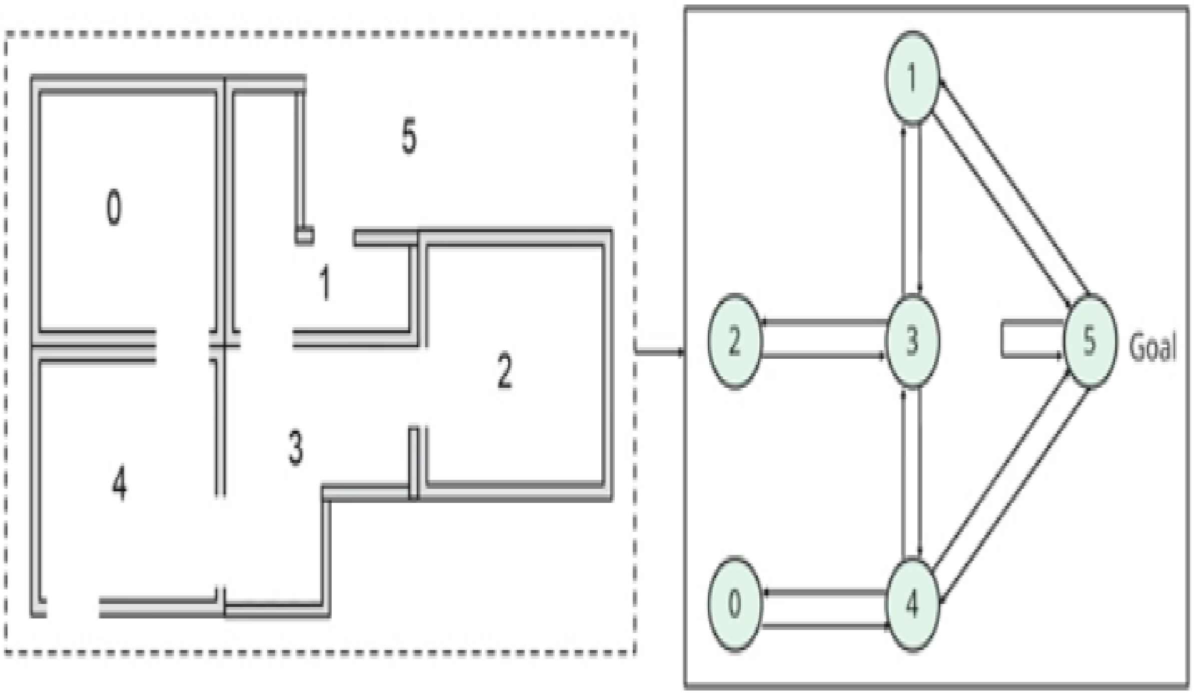
Understanding Q-Learning With An Example

Place an agent in any one of the rooms (0,1,2,3,4) and the goal is to reach outside the building (room 5)



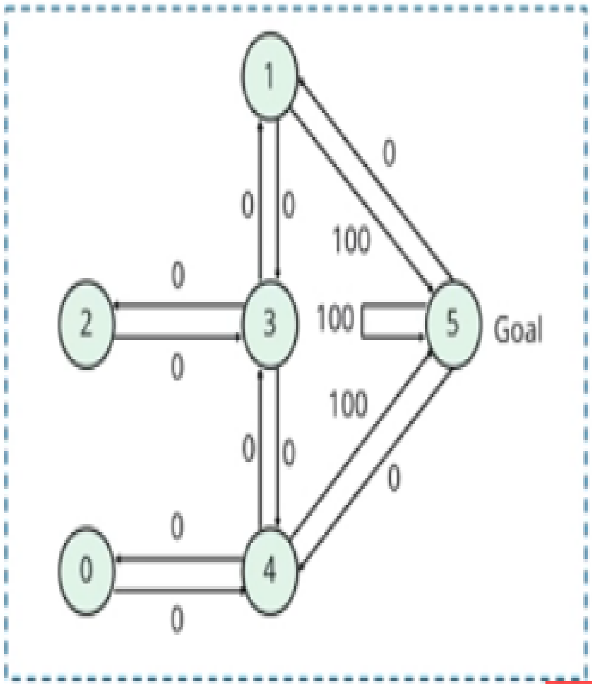
- 5 rooms in a building connected by doors
- each room is numbered 0 through 4
- The outside of the building can be thought of as one big room (5)
- Doors 1 and 4 lead into the building from room 5 (outside)

Let's represent the rooms on a graph, each room as a node, and each door as a link



Next step is to associate a reward value to each door:

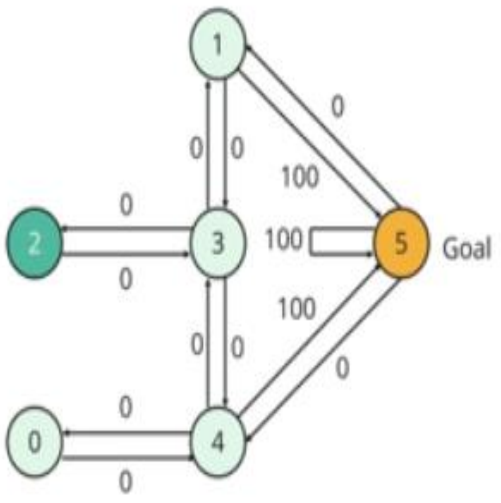
- doors that lead directly to the goal have a reward of 100
- Doors not directly connected to the target room have zero reward
- Because doors are two-way, two arrows are assigned to each room
- Each arrow contains an instant reward value



Understanding Q-Learning With An Example

The terminology in Q-Learning includes the terms state and action:

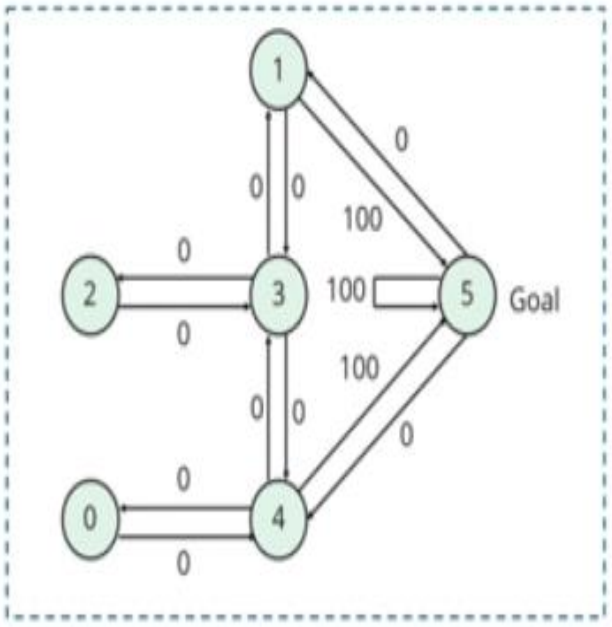
- Room (including room 5) represents a state
- agent's movement from one room to another represents an action
- In the figure, a state is depicted as a node, while "action" is represented by the arrows



Example (Agent traverse from room 2 to room5):

1. Initial state = state 2
2. State 2 -> state 3
3. State 3 -> state (2, 1, 4)
4. State 4 -> state 5

We can put the state diagram and the instant reward values into a reward table, matrix R.



$$R = \begin{array}{c|cccccc} & \multicolumn{6}{c}{\text{Action}} \\ \text{State} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & -1 & -1 & -1 & -1 & 0 & -1 \\ 1 & -1 & -1 & -1 & 0 & -1 & 100 \\ 2 & -1 & -1 & -1 & 0 & -1 & -1 \\ 3 & -1 & 0 & 0 & -1 & 0 & -1 \\ 4 & 0 & -1 & -1 & 0 & -1 & 100 \\ 5 & -1 & 0 & -1 & -1 & 0 & 100 \end{array}$$

The -1's in the table represent null values

Understanding Q-Learning With An Example

Add another matrix Q, representing the memory of what the agent has learned through experience.

- The rows of matrix Q represent the current state of the agent
- columns represent the possible actions leading to the next state
- Formula to calculate the Q matrix:

*Q(state, action) = R(state, action) + Gamma * Max [Q(next state, all actions)]*

Note

The Gamma parameter has a range of 0 to 1 (0 <= Gamma > 1).

- If Gamma is closer to zero, the agent will tend to consider only immediate rewards.
- If Gamma is closer to one, the agent will consider future rewards with greater weight

Q – Learning Algorithm

1. Set the gamma parameter, and environment rewards in matrix R
2. Initialize matrix Q to zero
3. Select a random initial state
4. Set initial state = current state
5. Select one among all possible actions for the current state
6. Using this possible action, consider going to the next state
7. Get maximum Q value for this next state based on all possible actions
8. Compute: Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]
9. Repeat above steps until current state = goal state

Q – Learning Example

First step is to set the value of the learning parameter Gamma = 0.8, and the initial state as Room 1.

Next, initialize matrix Q as a zero matrix:

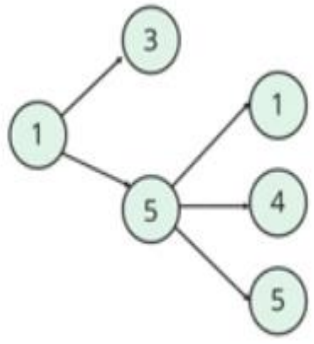
- From room 1 you can either go to room 3 or 5, let's select room 5.
- From room 5, calculate maximum Q value for this next state based on all possible actions:

*Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]*

Q(1,5) = R(1,5) + 0.8 * Max[Q(5,1), Q(5,4), Q(5,5)] = 100 + 0.8 * 0 = 100

$$Q = \begin{array}{c|cccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

$$R = \begin{array}{c|cccccc} & & & & \text{Action} & & \\ \text{State} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & -1 & -1 & -1 & -1 & 0 & -1 \\ 1 & -1 & -1 & -1 & 0 & -1 & 100 \\ 2 & -1 & -1 & -1 & 0 & -1 & -1 \\ 3 & -1 & 0 & 0 & -1 & 0 & -1 \\ 4 & 0 & -1 & -1 & 0 & -1 & 100 \\ 5 & -1 & 0 & -1 & -1 & 0 & 100 \end{array}$$



Q – Learning Example

For the next episode, we start with a randomly chosen initial state, i.e. state 3

- From room 3 you can either go to room 1,2 or 4, let's select room 1.
- From room 1, calculate maximum Q value for this next state based on all possible actions:

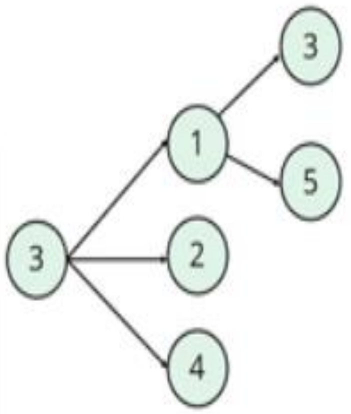
*Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]*

Q(3,1) = R(3,1) + 0.8 * Max[Q(1,3), Q(1,5)]= 0 + 0.8 * [0, 100] = 80

The matrix Q get's updated

$$Q = \begin{array}{c|cccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 100 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 80 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

$$R = \begin{array}{c|cccccc} & & & & \text{Action} & & \\ \text{State} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & -1 & -1 & -1 & -1 & 0 & -1 \\ 1 & -1 & -1 & -1 & 0 & -1 & 100 \\ 2 & -1 & -1 & -1 & 0 & -1 & -1 \\ 3 & -1 & 0 & 0 & -1 & 0 & -1 \\ 4 & 0 & -1 & -1 & 0 & -1 & 100 \\ 5 & -1 & 0 & -1 & -1 & 0 & 100 \end{array}$$



Q – Learning Example

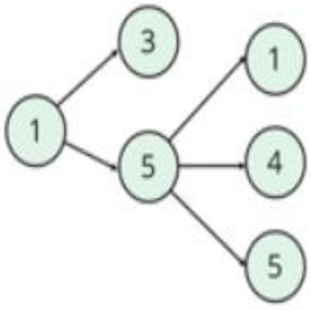
For the next episode, the next state, 1, now becomes the current state. We repeat the inner loop of the Q learning algorithm because state 1 is not the goal state.

- From room 1 you can either go to room 3 or 5, let's select room 5.
- From room 5, calculate maximum Q value for this next state based on all possible actions:

*Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]*

Q(1,5) = R(1,5) + 0.8 * Max[Q(5,1), Q(5,4), Q(5,5)] = 100 + 0.8 * 0 = 100
The matrix Q remains the same since, Q(1,5) is already fed to the agent

$$Q = \begin{array}{c|cccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 100 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 80 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \qquad R = \begin{array}{c|cccccc} & & & & \text{Action} & & \\ \text{State} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & -1 & -1 & -1 & -1 & 0 & -1 \\ 1 & -1 & -1 & -1 & 0 & -1 & 100 \\ 2 & -1 & -1 & -1 & 0 & -1 & -1 \\ 3 & -1 & 0 & 0 & -1 & 0 & -1 \\ 4 & 0 & -1 & -1 & 0 & -1 & 100 \\ 5 & -1 & 0 & -1 & -1 & 0 & 100 \end{array}$$



UNIT - V GRAPHICAL MODELS

Markov Chain Monte Carlo Methods – Sampling – Proposal Distribution – Markov Chain Monte Carlo – Graphical Models – Bayesian Networks – Markov Random Fields – Hidden Markov Models – Tracking Methods.

Introduction to Markov Chain Monte Carlo for Probability

- Markov Chain Monte Carlo sampling provides a class of algorithms for systematic random sampling from high-dimensional probability distributions. Unlike Monte Carlo sampling methods that are able to draw independent samples from the distribution, Markov Chain Monte Carlo methods draw samples where the next sample is dependent on the existing sample, called a Markov Chain. This allows the algorithms to narrow in on the quantity that is being approximated from the distribution, even with a large number of random variables.

- Monte Carlo sampling is not effective and may be intractable for high-dimensional probabilistic models.

- Markov Chain Monte Carlo provides an alternate approach to random sampling a high-dimensional probability distribution where the next sample is dependent upon the current sample.

- Gibbs Sampling and the more general Metropolis-Hastings algorithm are the two most common approaches to Markov Chain Monte Carlo sampling.

- Markov Chain Monte Carlo is a method to sample from a population with a complicated probability distribution.

Let's define some terms:

- **Sample -** A subset of data drawn from a larger population. (Also used as a verb to sample; i.e. the act of selecting that subset. Also, reusing a small piece of one song in another song, which is not so different from the statistical practice, but is more likely to lead to lawsuits.) Sampling permits us to approximate data without exhaustively analyzing all of it, because some datasets are too large or complex to compute. We're often stuck behind a veil of ignorance, unable to gauge reality around us with much precision. So we sample.1

- **Population -** The set of all things we want to know about; e.g. coin flips, whose outcomes we want to predict. Populations are often too large for us to study them in toto, so we sample. For example, humans will never have a record of the outcome of all coin flips since the dawn of time. It's physically impossible to collect, inefficient to compute, and politically unlikely to be allowed. Gathering information is expensive. So in the name of efficiency, we select subsets of the population and pretend they represent the whole. Flipping a coin 100 times would be a sample of the population of all coin tosses and would allow us to reason inductively about all the coin flips we cannot see.

- **Distribution (or probability distribution)** - You can think of a distribution as table that links outcomes with probabilities. A coin toss has two possible outcomes, heads (H) or tails (T). Flipping it twice can result in either HH, TT, HT or TH. So let's construct a table that shows the outcomes of two-coin tosses as measured by the number of H that result.
- Markov Chain Monte Carlo (MCMC) is a mathematical method that draws samples randomly from a black box to approximate the probability distribution of attributes over a range of objects or future states. You could say it's a large-scale statistical method for guess-and-check. MCMC methods help gauge the distribution of an outcome or statistic you're trying to predict, by randomly sampling from a complex probabilistic space. As with all statistical techniques, we sample from a distribution when we don't know the function to succinctly describe the relation to two variables (actions and rewards). MCMC helps us approximate a black-box probability distribution.
- Markov Property says that given a process which is at a state Xn at a particular point of time, the probability of Xn+1=k, where k is any of the M states the process can jump to, will only be dependent on which state it is at the given moment. And not on how it reached the current state. Mathematically speaking

$$P(X_{n+1} = k | X_n = k_n, X_{n-1} = k_{n-1}, \dots, X_1 = k_1) = P(X_{n+1} = k | X_n = k_n)$$

-
- **Markov Chain Monte Carlo** (MCMC) methods are a class of algorithms for **sampling from a probability distribution** based on constructing a Markov chain that has the desired distribution as its stationary distribution. The state of the chain after a number of steps is then used as a sample of the desired distribution. The quality of the sample improves as a function of the number of steps.
- MCMC methods make life easier for us by providing us with algorithms that could create a Markov Chain which has the Beta distribution as its **stationary distribution** given that we can sample from a uniform distribution(which is relatively easy).

MARKOV IDEA

- Design a Markov Chain on finite state space

$$\text{state space}: x^{(i)} \in \{x_1, x_2, \dots, x_s\}$$

- $$\text{Markov property}: \ p(x^{(i)} | x^{(i-1)}, \dots, x^{(1)}) = T(x^{(i)} | x^{(i-1)})$$

- such that when simulating a trajectory of states from it, it will explore the state space spending more time in the most important regions (i.e. where p(x) is large)

GIBBS SAMPLING ALGORITHM

- Like other MCMC methods, the Gibbs sampler constructs a Markov Chain whose values converge towards a target distribution. Gibbs Sampling is in fact a specific case of the Metropolis-Hastings algorithm wherein proposals are always accepted.
- To elaborate, suppose you wanted to sample a multivariate probability distribution.

```
initialize $Y^0, X^0$
for j = 1, 2, 3,... do
    sample $X^j \sim p(X|Y^{j-1})$
    sample $Y^j \sim p(Y|X^j)$
end for
```

Let's take a look at an example. Suppose we had the following posterior and conditional probability distributions.

$$p(x, y) = \frac{1}{C} e^{-\frac{x^2y^2+x^2+y^2-8x-8y}{2}}$$

$$p(x|y) = g(y) e^{-(x-\frac{4}{1+y^2})^2(\frac{1+y^2}{2})}$$

$$p(y|x) = g(x) e^{-(y-\frac{4}{1+x^2})^2(\frac{1+x^2}{2})}$$

where g(y) contains the terms that don't include x, and g(x) contains the those don't depend on y. We don't know the value of C (normalizing constant). However, we do know the conditional probability distributions. Therefore, we can use Gibbs Sampling to approximate the posterior distribution.

METRO POLIS –HASTING ALGORTIHM:-

```
1. Initialise $x^{(0)}$.
2. For $i = 0$ to $N - 1$
    — Sample $u \sim \mathcal{U}_{[0,1]}$.
    — Sample $x^\star \sim q(x^\star | x^{(i)})$.
    — If $u < \mathcal{A}(x^{(i)}, x^\star) = \min\left\{1, \frac{p(x^\star)q(x^{(i)}|x^\star)}{p(x^{(i)})q(x^\star|x^{(i)})}\right\}$
            $x^{(i+1)} = x^\star$
      else
            $x^{(i+1)} = x^{(i)}$
```

The Metropolis algorithm assumes a symmetric random walk proposal $q(x^\star|x^{(i)}) = q(x^{(i)}|x^\star)$ and, hence, the acceptance ratio simplifies to

$$\mathcal{A}(x^{(i)}, x^\star) = \min\left\{1, \frac{p(x^\star)}{p(x^{(i)})}\right\}.$$

SAMPLINGS:-

Statistical sampling is a large field of study, but in applied machine learning, there may be three types of sampling that you are likely to use: simple random sampling, systematic sampling, and stratified sampling.

Simple Random Sampling: Samples are drawn with a uniform probability from the domain.

Systematic Sampling: Samples are drawn using a pre-specified pattern, such as at intervals.

Stratified Sampling: Samples are drawn within pre-specified categories (i.e. strata).

Although these are the more common types of sampling that you may encounter, there are other techniques.

PROPOSTIONAL SAMPLING

A walk through the concept of proportional sampling by an example explanation with python codes to perform the same.

What is proportional sampling?

Example problem

Algorithm

What is proportional sampling?

In most simple words, proportional sampling is a sampling of a population in which the probability of finding an element is proportional to some common shared attribute or property of all the elements in the population. For example, suppose you have a set of numbers, say {2,5,8,15,46,90}, and you want to randomly pick a number but you don't want the probability to be uniform. Instead, you want the probability of finding a number to be proportional to the face values of the number which precisely means that 90 should have the highest probability and 2 should have the lowest probability to be picked up.

EXAMPLE:

In Europe a new football tournament was announced. Many big business men came forward to start their own clubs. There was one owner, Mr. Robert, who had no knowledge about how to choose right player for the team. But he was quite certain that a greater number of goals a player has scored, the better is the player. With this much of knowledge is arranged the player vs number of goals data set.

	Player	Total Goals
1	Giotto	95
2	David	652
3	Arthur	588
4	Tom	379
5	Eric	47
	.	.
	.	.
	.	.
99	Nelson	283
100	Nick	546

Each team has to select 18 players. Now the rule of player choosing was that Mr. Robert could take 4 players of his choice and has to select the other 14 randomly. So basically Mr. Robert has to select 14 random players from table. Now the problem is how to randomly select the player such that the probability of selecting the player is more if the number of goals is more.

Compute the total sum of goals

$$S = \sum_{i=1}^{n} G_i$$

Normalize goals of each player with respect to S.

$$G'_i = \frac{G_i}{S}$$

$$Such\ that, \sum_{i=1}^{n} G'_i = 1$$

Compute Cumulative Normalized Sum of goals for each player. Note: The cumulative normalized sum for the last player will be equal to 1.0

$$\begin{aligned} G''_1 &= G'_1 \\ G''_2 &= G''_1 + G'_2 \\ G''_3 &= G''_2 + G'_3 \\ G''_4 &= G''_3 + G'_4 \\ &\dots \\ G''_{100} &= G''_{99} + G'_{100} \end{aligned}$$

Pick a value randomly, r, from the range (0,1).

For each cumulative normalized G in the list of G” , if r ≤ G’’_i ,then return player corresponding to (G’’_i)*S.

Why Graphical Models?

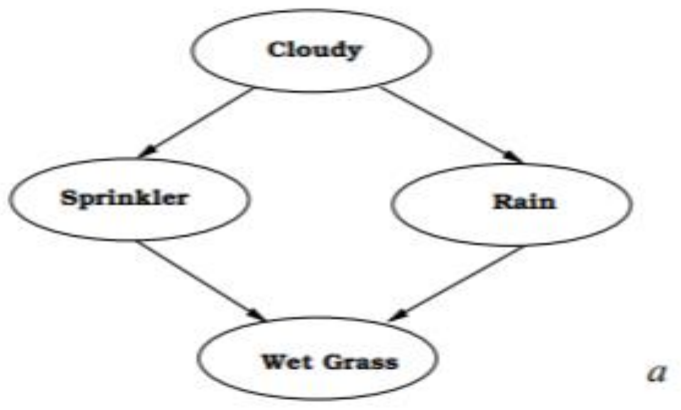
- Framework for modeling and efficiently reasoning about multiple correlated random variables
- Provides insights into the assumptions of existing models
- Allows qualitative specification of independence assumptions

Why Graphical Models? Recent Trends in Data Mining

- Traditional learning algorithms assume – Data available in record format – Instances are i.d samples

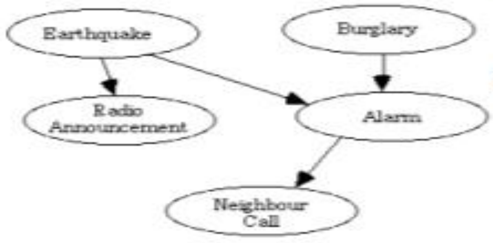
- Recent domains like Web, Biology, Marketing have more richly structured data
- Examples : DNA Sequences, Social Networks, Hyperlink structure of Web, Phylogeny Trees
- Relational Data Mining - Data spread across multiple tables
- Relational Structure helps significantly in enhancing accuracy
- Graphical Models offer a natural formalism to model such data
- Graph G =< V, E > representing a family of probability distributions
- Nodes V - Random Variables
- Edges E - Indicate Stochastic Dependence
- G encodes Conditional Independence assertions in domain
- Mainly two kinds of Models
- – Directed (Bayesian Networks)
- – Undirected (Markov Random Fields (MRFs))

Graphical Models (Contd...)



- Direction of edges based on causal knowledge
  - $A \rightarrow B$: A "causes" B
  - $A - B$: Not sure of causality
- Mixed versions also possible - *Chain Graphs*

Directed Models : Bayesian Networks



[a]

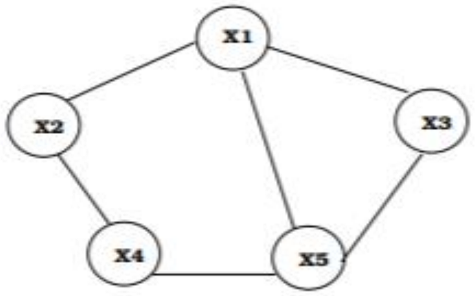
[a]Figure adapted from [RN95]

- *Bayes Net* - DAG encoding the conditional independence assumptions among the variables
- Cycles not allowed - Edges usually have causal interpretations
- Specifies a compact representation of joint distribution over the variables given by

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P_i(X_i \mid Pa(X_i))$$

  where $Pa(X_i)$ = Parents of Node X_i in the network
- $P_i \rightarrow$ *Conditional Probability Distribution (CPD)* of X_i

Undirected Graphical Models
Markov Random Fields



- Have been well studied and applied in Vision
- No underlying causal structure
- Joint distribution can be factorized into

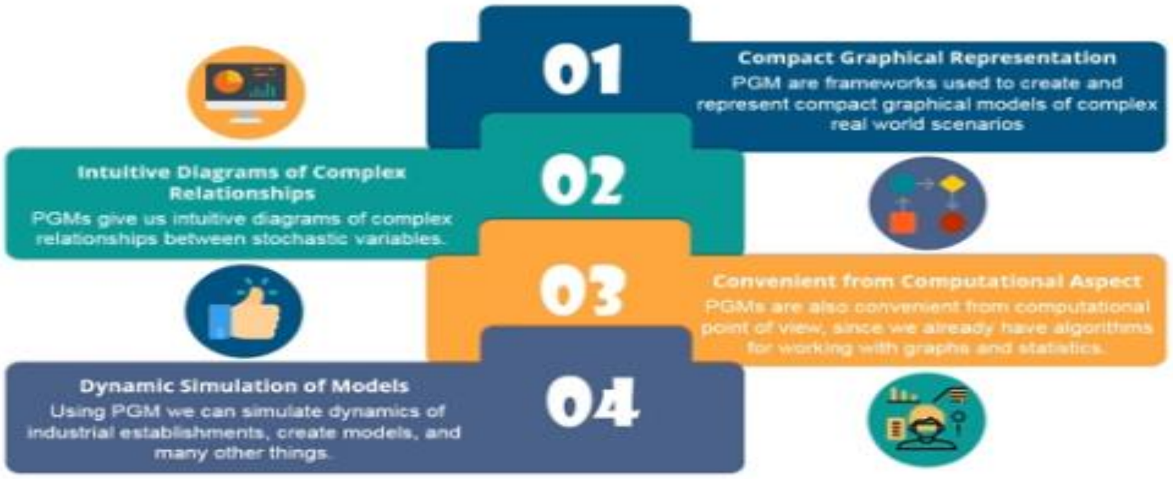
$$P(X_1, \ldots, X_n) = \frac{1}{Z} \prod_{c \in C} \psi_c(X_c)$$

where C - Set of cliques in graph

- ψ_c - Potential function (a positive function) on the clique X_c
- Z - Partition Function given by

$$Z = \sum_{\vec{x}} \prod_{c \in C} \psi_c(X_c)$$

***Probabilistic Graphical Models** are rich frameworks for encoding probability distributions over complex domains.*

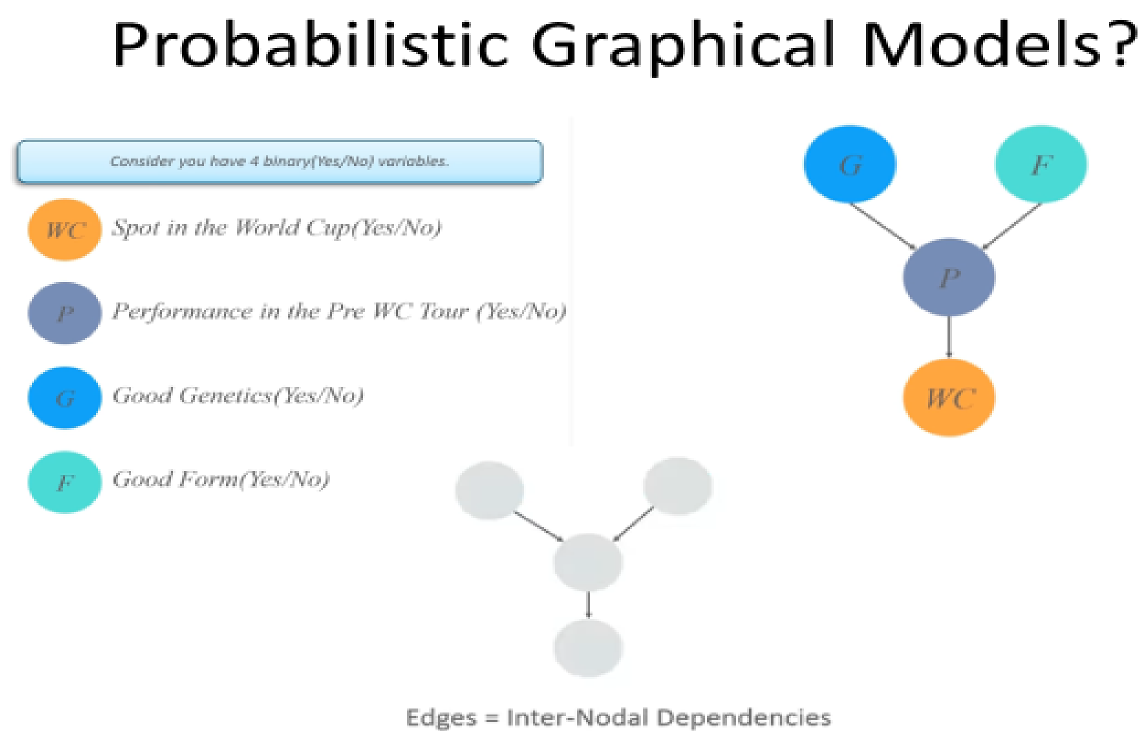


01 **Compact Graphical Representation**
PGM are frameworks used to create and represent compact graphical models of complex real world scenarios

02 **Intuitive Diagrams of Complex Relationships**
PGMs give us intuitive diagrams of complex relationships between stochastic variables.

03 **Convenient from Computational Aspect**
PGMs are also convenient from computational point of view, since we already have algorithms for working with graphs and statistics.

04 **Dynamic Simulation of Models**
Using PGM we can simulate dynamics of industrial establishments, create models, and many other things.


Probabilistic Graphical Models?

Consider you have 4 binary(Yes/No) variables.

WC	Spot in the World Cup(Yes/No)
P	Performance in the Pre WC Tour (Yes/No)
G	Good Genetics(Yes/No)
F	Good Form(Yes/No)

Edges = Inter-Nodal Dependencies

Probabilistic

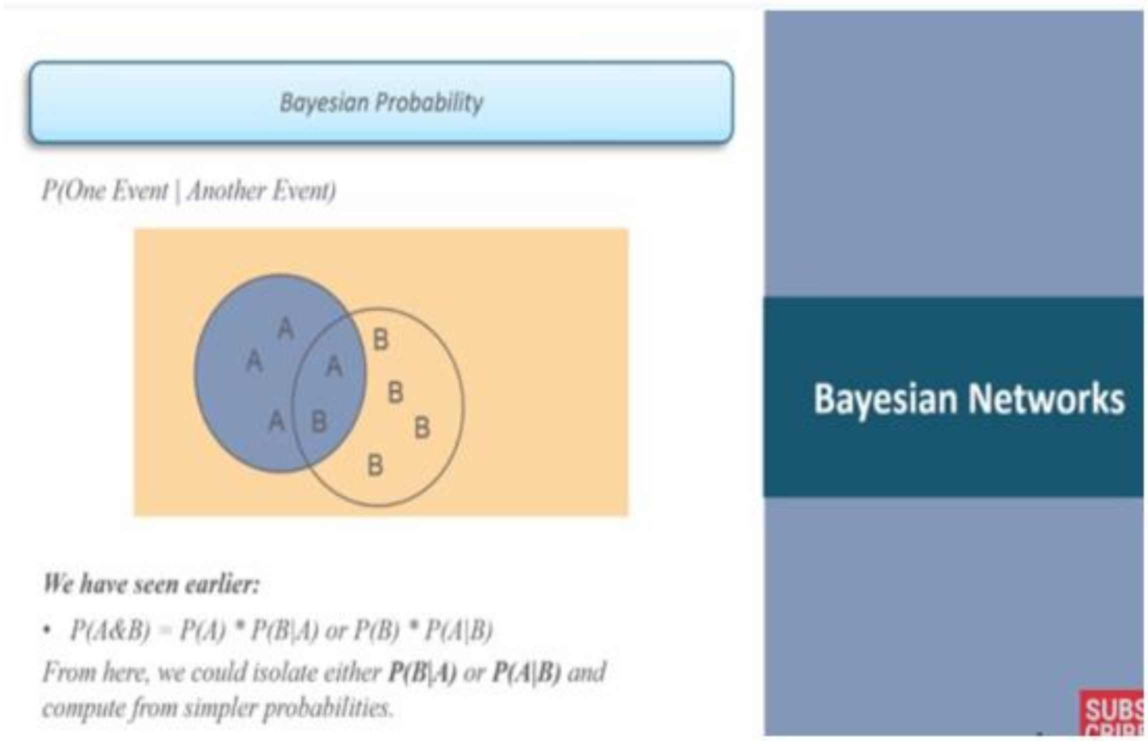
The nature of the problem that we are generally interested to solve or the type of queries we want to make are all probabilistic because of uncertainty. There are many reasons that contributes to it.

Graphical

Graphical representation helps us to visualize better and so, we use Graph Theory to reduce the no of relevant combinations of all the participating variables to represent the high dimensional probability distribution model more company.

Models

A Model is a declarative representation of a real-world scenario or a problem that we want to analysis. It is represented by using any mathematical tools like graph or even simply by an equation.



Bayesian Probability

P(One Event | Another Event)

We have seen earlier:

- *P(A&B) = P(A) * P(B|A) or P(B) * P(A|B)*

From here, we could isolate either ***P(B|A)*** *or* ***P(A|B)*** *and compute from simpler probabilities.*

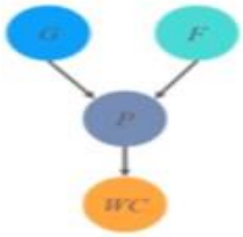
Bayesian Networks

SUBS CRIB

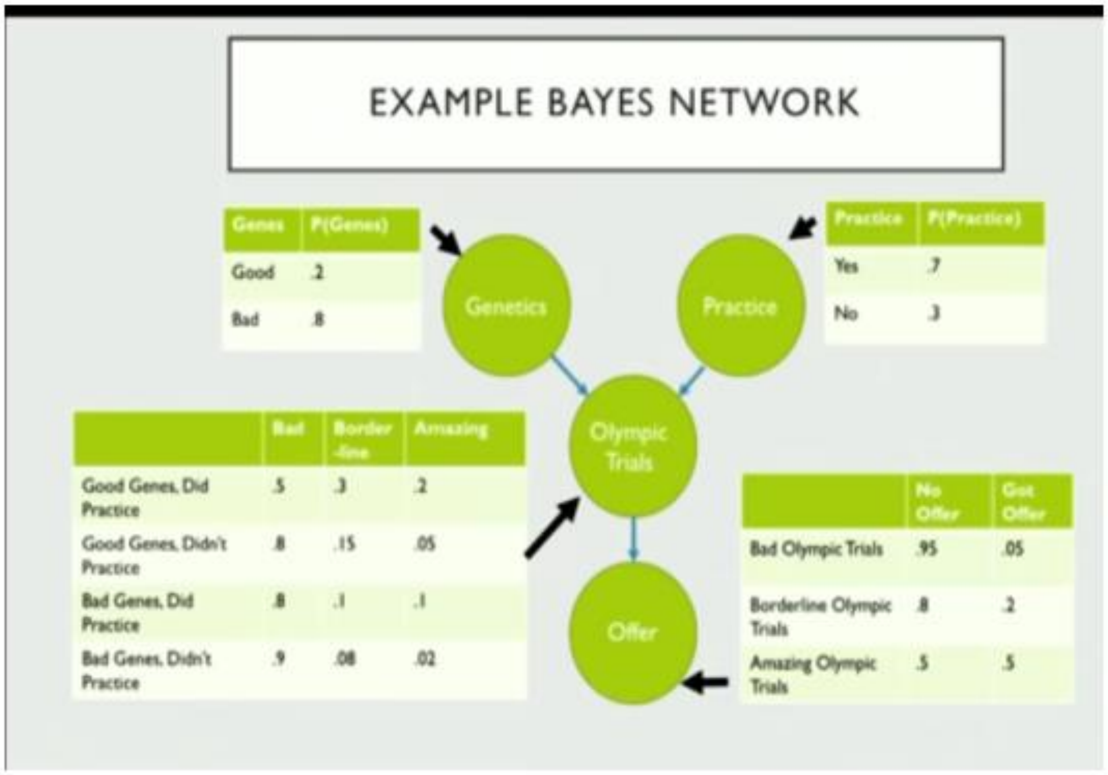
Bayes Network

*A **Bayes Network** is a structure that can be represented as a **Direct Acyclic Graph**.*

1. *It allows **a compact representation** of the distribution from the chain rule of Bayes Networks.*
2. *It observes **conditional independence relationships** between random variables.*



*A **DAG(Direct Acyclic Graph)** is a finite directed graph with no directed cycles.*



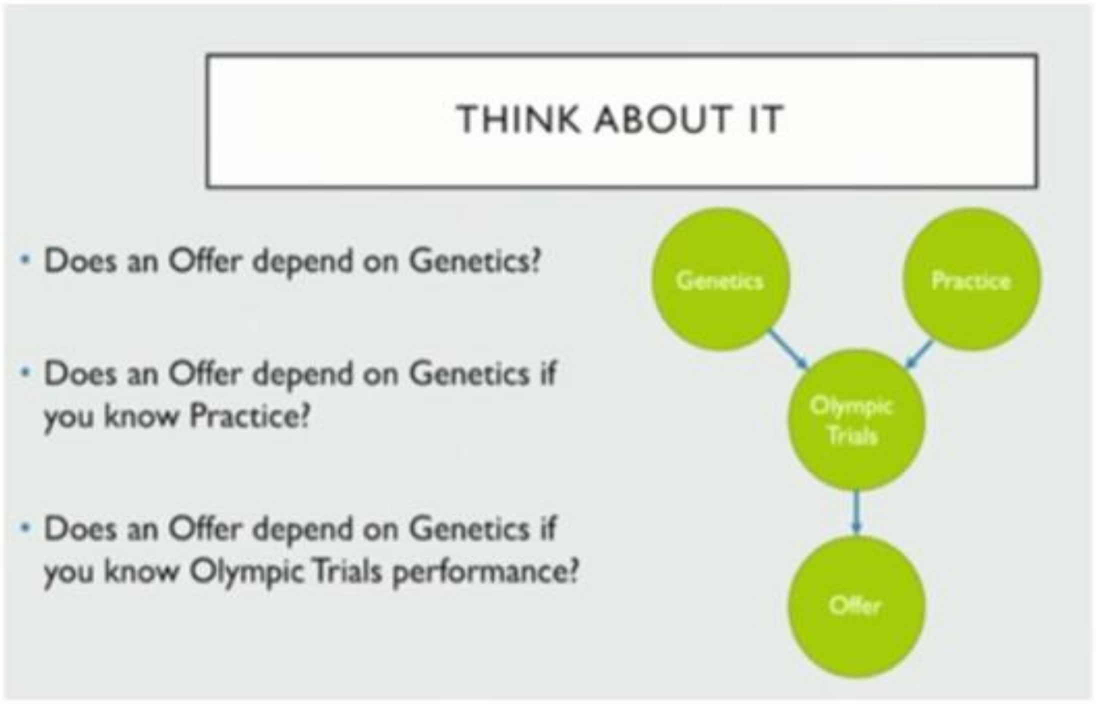
EXAMPLE BAYES NETWORK

Genes	P(Genes)
Good	.2
Bad	.8

Practice	P(Practice)
Yes	.7
No	.3

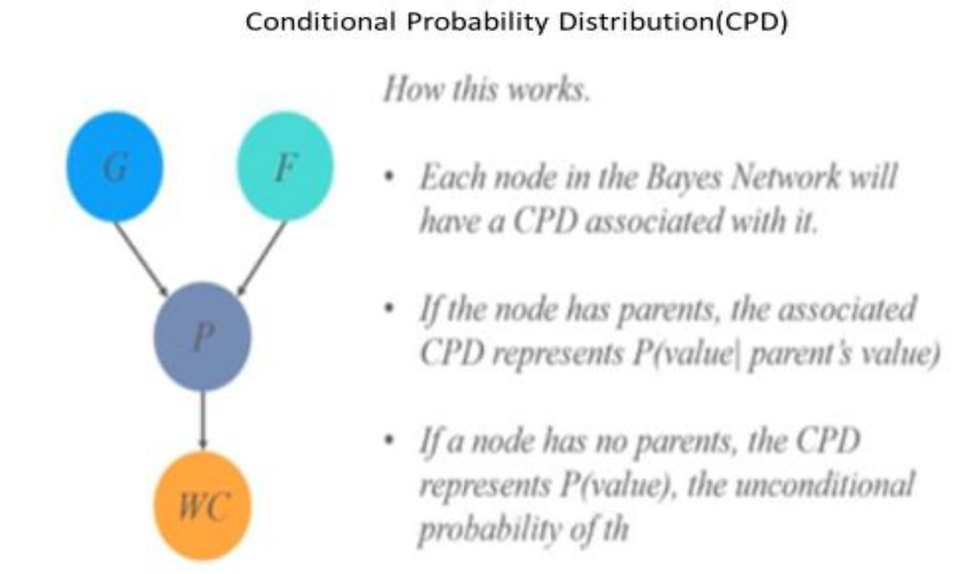
	Bad	Border-line	Amazing
Good Genes, Did Practice	.5	.3	.2
Good Genes, Didn't Practice	.8	.15	.05
Bad Genes, Did Practice	.8	.1	.1
Bad Genes, Didn't Practice	.9	.08	.02

	No Offer	Got Offer
Bad Olympic Trials	.95	.05
Borderline Olympic Trials	.8	.2
Amazing Olympic Trials	.5	.5



THINK ABOUT IT

- Does an Offer depend on Genetics?
- Does an Offer depend on Genetics if you know Practice?
- Does an Offer depend on Genetics if you know Olympic Trials performance?



Conditional Probability Distribution(CPD)

How this works.

- *Each node in the Bayes Network will have a CPD associated with it.*
- *If the node has parents, the associated CPD represents P(value| parent's value)*
- *If a node has no parents, the CPD represents P(value), the unconditional probability of th*

Application

- Medical diagnostics- model how a physician thinks
- Image processing- labelling pixels with information about their neighbours

- Natural language processing
- Directed graphical models specify a factorization of the joint distribution over a set of variables into a product of local conditional distributions

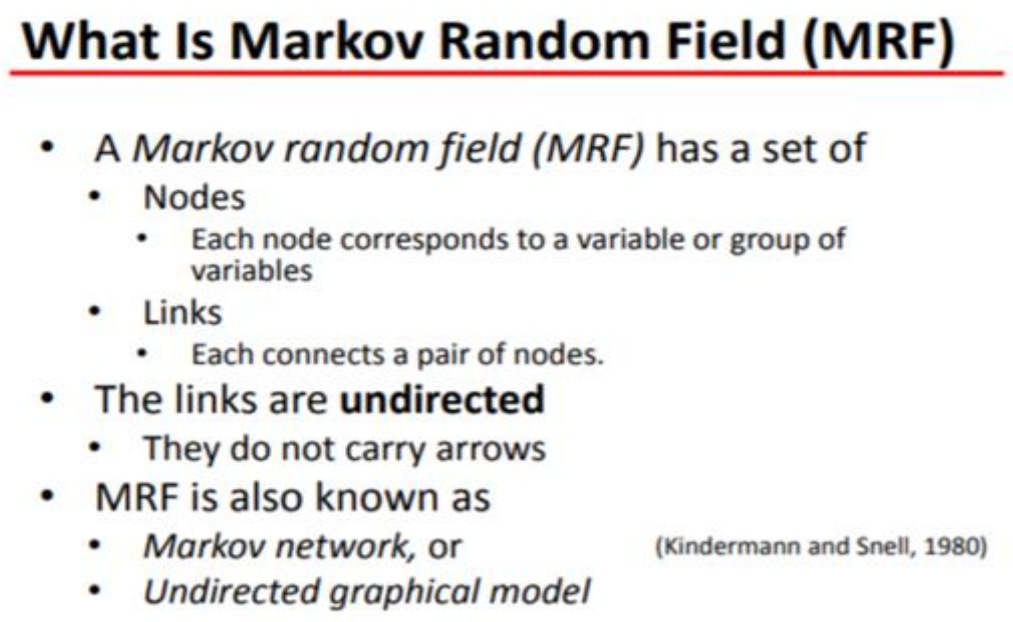
• Second major class of graphical models that are described by undirected graphs and that again

specify both a factorization and a set of conditional independence relations.

• Markov Random Field (MRF)

• No inference algorithms

• But more on modeling and energy function

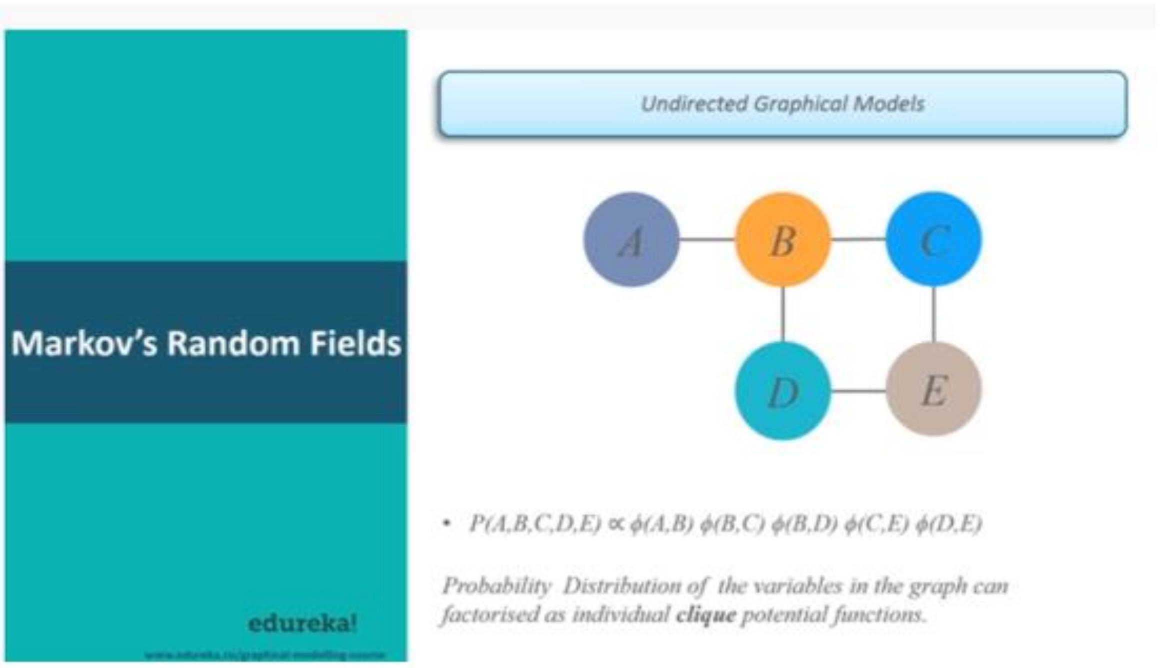


What Is Markov Random Field (MRF)

- A *Markov random field (MRF)* has a set of
  - Nodes
    - Each node corresponds to a variable or group of variables
  - Links
    - Each connects a pair of nodes.
- The links are **undirected**
  - They do not carry arrows
- MRF is also known as
  - *Markov network*, or (Kindermann and Snell, 1980)
  - *Undirected graphical model*

Why use MRF for Computer vision

- **Image de-noising**
- **Image de-blurring**
- **Image segmentation**
- **Image super-resolution**



Markov's Random Fields

Undirected Graphical Models

- $P(A,B,C,D,E) \propto \phi(A,B)\, \phi(B,C)\, \phi(B,D)\, \phi(C,E)\, \phi(D,E)$

Probability Distribution of the variables in the graph can factorised as individual ***clique*** *potential functions.*

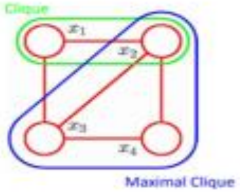
Clique

$$p(\mathbf{x}) = \prod_{C} p_C(\mathbf{x}_C), \qquad \mathbf{x} = \bigcup_{C} \mathbf{x}_C$$

- How to find the set of $\{\mathbf{x}_c\}$?
- We need to consider a graph terminology: ***clique***
  - It is a subset of the nodes in a graph such that there exists a **link between *all* pairs** of nodes in the subset.
  - The set of nodes in a clique is **fully connected**.

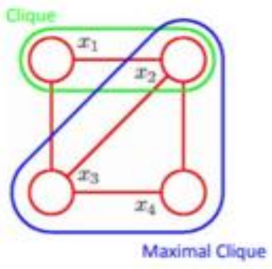
Cliques and Maximal Cliques

- A *maximal clique* is a clique that
  - It is not possible to include any other nodes from the graph to the set without it ceasing to be a clique.

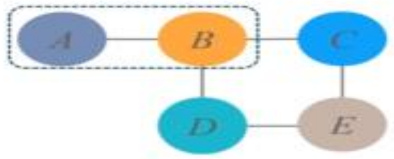


An Example of Clique

- This graph has ***five cliques*** of two nodes
  - $\{x_1, x_2\}, \{x_2, x_3\}, \{x_3, x_4\}, \{x_4, x_2\}, \{x_1, x_3\}$
- It has ***two maximal cliques***
  - $\{x_1, x_2, x_3\}, \{x_2, x_3, x_4\}$
- The set $\{x_1, x_2, x_3, x_4\}$ is **not a clique** because of the missing link from x1 to x4.

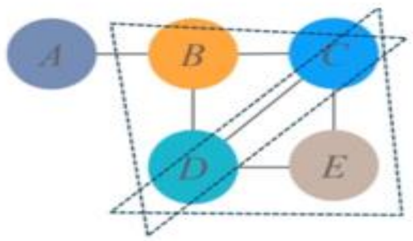


Cliques



- $P(A,B,C,D,E) \propto \phi(A,B)\ \phi(B,C)\ \phi(B,D)\ \phi(C,E)\ \phi(D,E)$

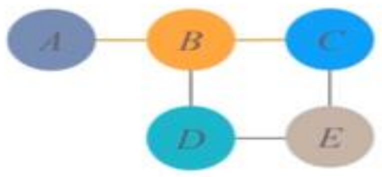
  $P(X) = \frac{1}{z} \prod_{c \in cliques(G)} \Phi_c\ (x_c\ )$ ***potential functions***



- $P(A,B,C,D,E) \propto \phi(A,B)\ \phi(B,C,D)\ \phi(C,D,E)$

  $P(X) = \frac{1}{z} \prod_{c \in cliques(G)} \Phi_c\ (x_c\ )$ ***potential functions***

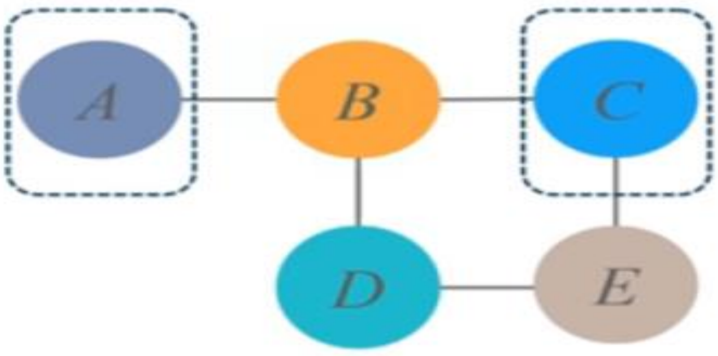
Markov Random Fields



- *Paths between A and C :*

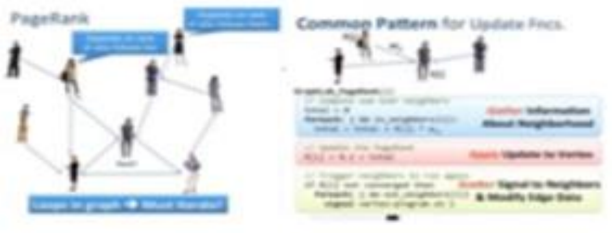
  A-B-C

  A-B-D-E-C



- *Any two subsets of variables are conditionally independent, given* ***a separating subset****.*
- ***{B,D}**,**{B,E}** & **{B,D,E}** are the separating subsets.*

Applications of PGMs

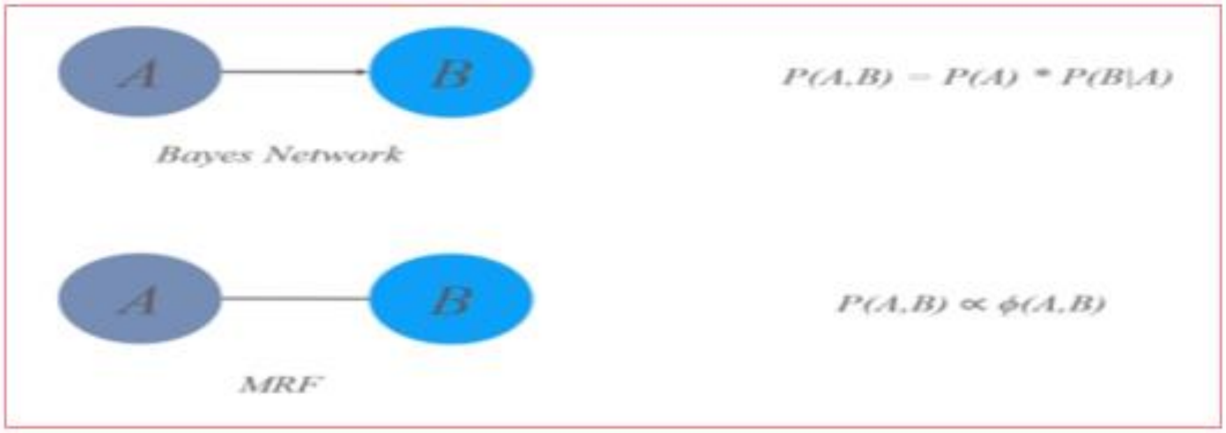


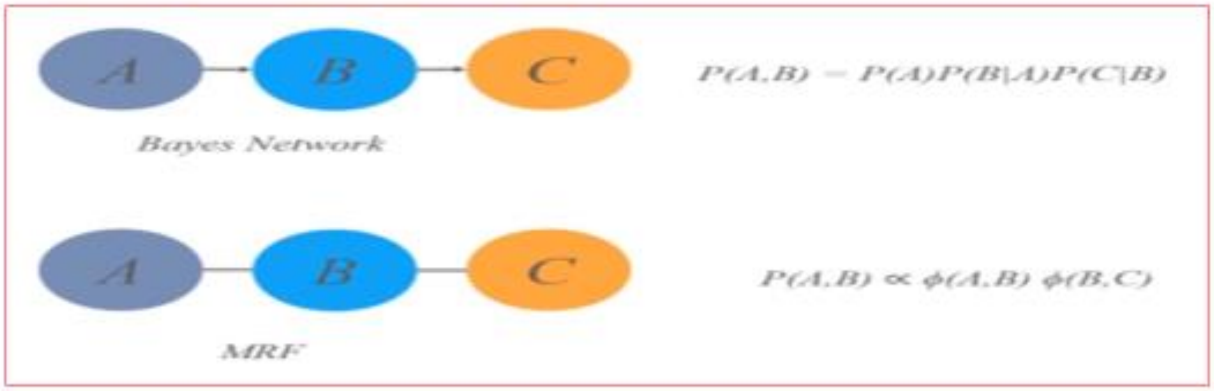
***Google** is based on a very simple graph algorithm called **page rank**.*

Netflix, Amazon, Facebook all use PGMs to recommend what is best for you.

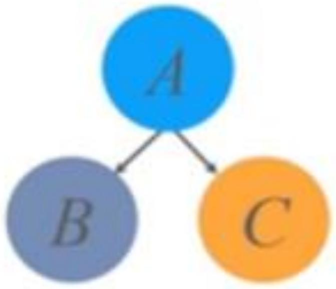
Women on the Verge of a Nervous Breakdown
The Celebration
City of God
Wild Strawberries
La Dolce Vita

Belief Network & Markov Random Fields



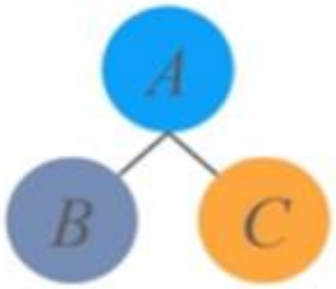

A
B
C
Bayes Network
$P(A,B) = P(A)P(B|A)P(C|B)$
A
B
C
MRF
$P(A,B) \propto \phi(A,B)\ \phi(B,C)$

Bayes Nets as MRFs : Shared Parents



Bayes Network

$$P(A,B,C) = P(A)P(B|A)P(C|B)$$



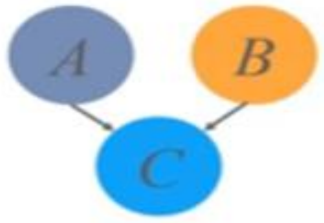
MRF

$$P(A,B,C) \propto \phi(A,B)\ \phi(A,C)$$

$$\phi(A,B) \leftarrow P(A)P(B|A)$$

$$\phi(A,C) \leftarrow P(C|A)$$

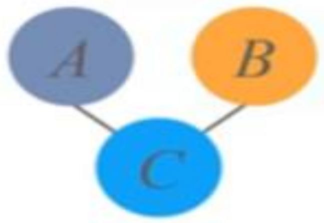
Bayes Nets as MRFs : Shared Child



Bayes Network

$P(A,B,C) = P(A)P(B)P(C|A,B)$

A and B are ***dependent*** *given C*

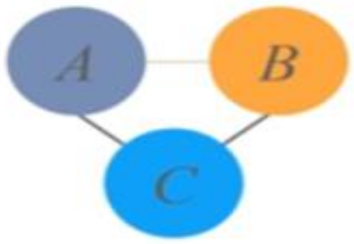


MRF

$P(A,B,C) \propto \phi(A,C)\ \phi(B,C)$

A and B are ***independent*** *given C*

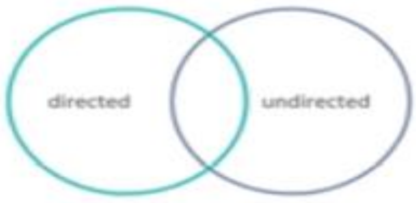
Converting Bayes Nets to MRFs : Moralizing Parents



$P(A,B,C) \propto \phi(A,C)\ \phi(B,C)$

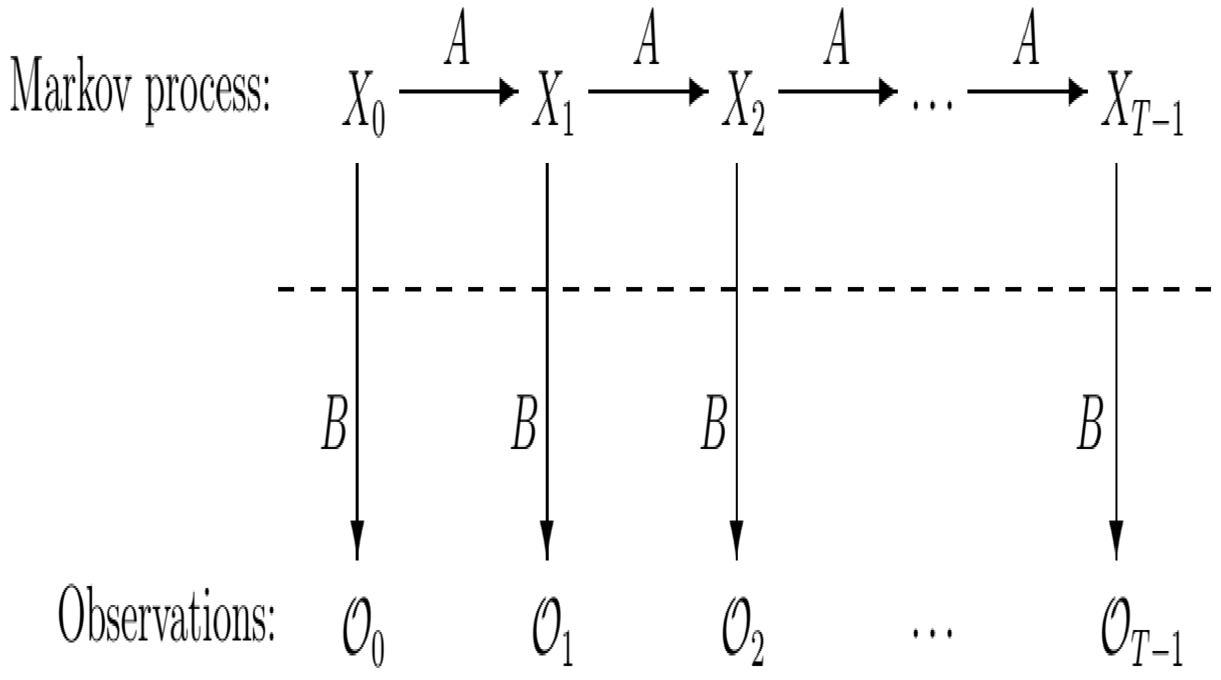
A and B are ***independent*** *given C*

- *Moralize all co-parents.*
- *Lose marginal independence of parents*



Hidden Markov Model

The Hidden Markov Model (HMM) is a relatively simple way to model sequential data. A hidden Markov model implies that the Markov Model underlying the data is hidden or unknown to you. More specifically, you only know observational data and not information about the states. In other words, there's a specific type of model that produces the data (a Markov Model) but you don't know what processes are producing it. You basically use your knowledge of Markov Models to make an educated guess about the model's structure.



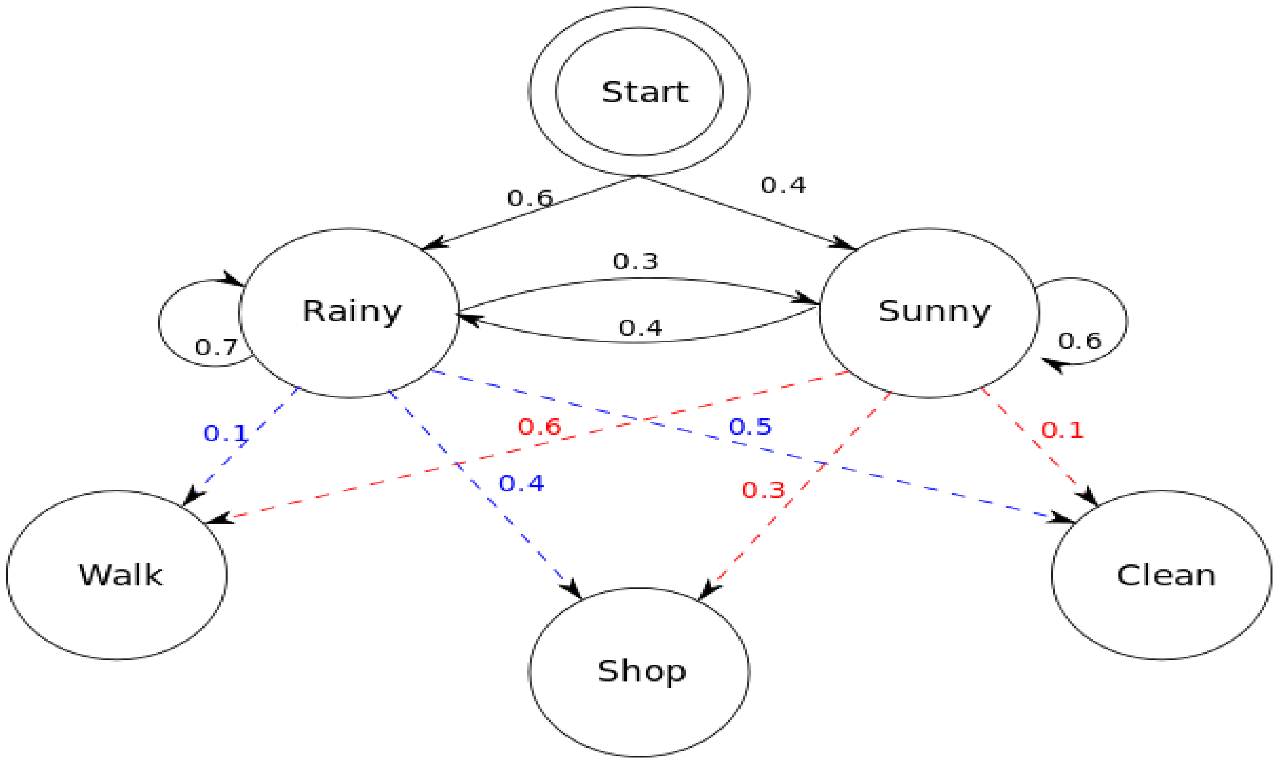
$$\begin{tikzcd}
X_0 \arrow[r, "A"] \arrow[d, "B"'] & X_1 \arrow[r, "A"] \arrow[d, "B"'] & X_2 \arrow[r, "A"] \arrow[d, "B"'] & \cdots \arrow[r, "A"] & X_{T-1} \arrow[d, "B"'] \\
\mathcal{O}_0 & \mathcal{O}_1 & \mathcal{O}_2 & \cdots & \mathcal{O}_{T-1}
\end{tikzcd}$$

HMM states (X), observations (O) and probabilities (A, B). Source: Stamp 2018

Consider weather, stock prices, DNA sequence, human speech or words in a sentence. In all these cases, current state is influenced by one or more previous states. Moreover, often we can observe the effect but not the underlying cause that remains hidden from the observer. Hidden Markov Model (HMM) helps us figure out the most probable hidden state given an observation.

In practice, we use a sequence of observations to estimate the sequence of hidden states. In HMM, the next state depends only on the current state. As such, it's good for modelling time series data.We can classify HMM as a **generative probabilistic model** since a sequence of observed variables is generated by a sequence of hidden states. HMM is also seen as a specific kind of **Bayesian network**.

Discussion



- Could you explain HMM with an example?

An example of Hidden Markov Model. Source: Wikipedia 2019.

Suppose Bob tells his friend Alice what he did earlier today. Based on this information Alice guesses today's weather at Bob's location. In HMM, we model weather as *states* and Bob's activity as *observations*.

To solve this problem, Alice needs to know three things:

- **Transition Probabilities**: Probability of moving from one state to another. For example, "If today was sunny, what's the probability that it will rain tomorrow?" If there are N states, this is an NxN matrix.
- **Emission Probabilities**: Probability of a particular output given a particular state. For example, "What's the chance that Bob is walking if it's raining?" Given a choice of M possible observation symbols, this is an NxM matrix. This is also called output or observation probabilities.

- **Initial Probabilities**: Probability of being in a state at the start, say, yesterday or ten days ago.

  Unlike a typical Markov chain, we can't see the states in HMM. However, we can observe the output and then predict the state. Thus, the states are hidden, giving rise to the term "hidden" in the name HMM.

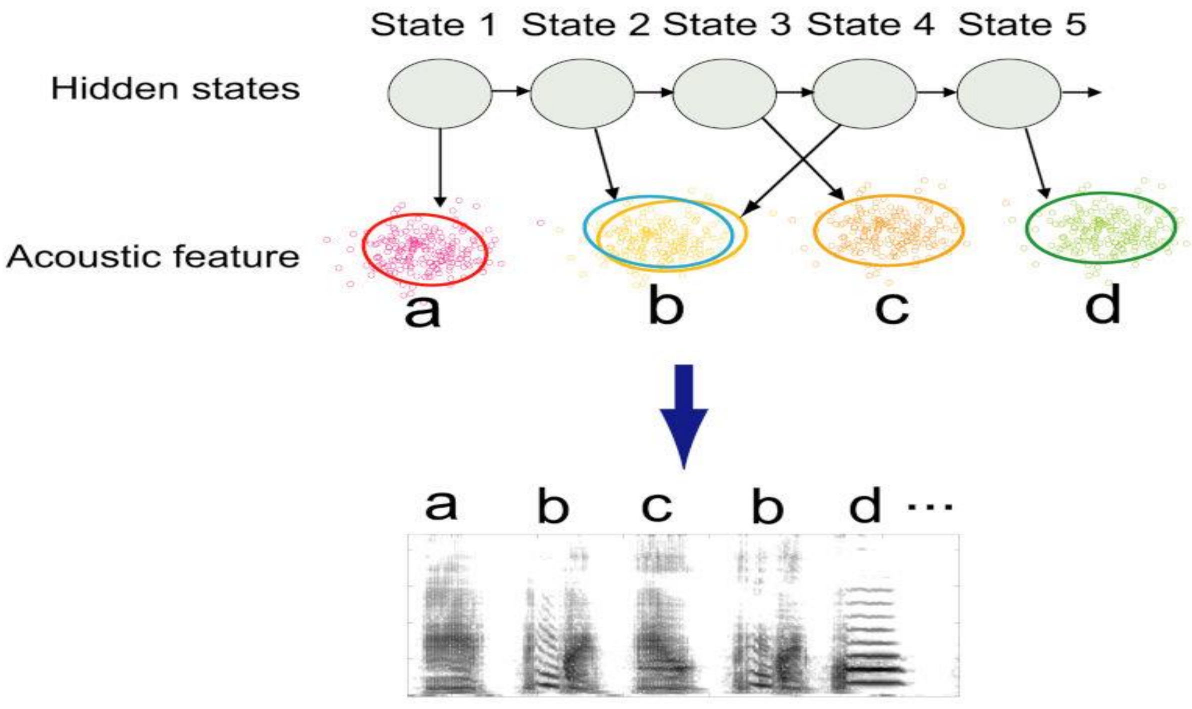
-
$$\begin{aligned}
T &= \text{length of the observation sequence}\\
N &= \text{number of states in the model}\\
M &= \text{number of observation symbols}\\
Q &= \{q_0, q_1, \ldots, q_{N-1}\} = \text{distinct states of the Markov process}\\
V &= \{0, 1, \ldots, M-1\} = \text{set of possible observations}\\
A &= \text{state transition probabilities}\\
B &= \text{observation probability matrix}\\
\pi &= \text{initial state distribution}\\
\mathcal{O} &= (\mathcal{O}_0, \mathcal{O}_1, \ldots, \mathcal{O}_{T-1}) = \text{observation sequence.}
\end{aligned}$$

  Typical notation used in HMM. Source: Kang 2017.

  Let A, B and π denote the transition matrix, observation matrix and initial state distribution respectively. HMM can be represented as $\lambda = (A, B, \pi)$. Let observation sequence be O and state sequence be Q.

  HMM can be used to solve three types of problems:

- **Likelihood Problem**: Given O and λ, find the likelihood P(O|λ). How likely is a particular sequence of observations? *Forward algorithm* solves this problem.
- **Decoding Problem**: Given O and λ, find the best possible Q that explains O. Given the observation sequence, what's the best possible state sequence? *Viterbi algorithm* solves this problem.
- **Learning Problem**: Given O and Q, learn λ, perhaps by maximizing P(O|λ). What model best maps states to observations? *Baum-Welch algorithm*, also called *forward-backward algorithm*, solves this problem. In the language of machine learning, we can say that O is training data and the number of states N is the model's hyperparameter.
- What are some applications where HMM is useful?



Complex birdsong analyzed using HMM. Source: Adapted from Katahira et al. 2011 .

HMM has been applied in many areas including automatic speech recognition, handwriting recognition, gesture recognition, part-of-speech tagging, musical score following, partial discharges and bioinformatics. In speech recognition, a spectral analysis of speech gives us suitable observations for HMM. States are modelled after phonemes or syllables, or after the average number of observations in a spoken word. Each word gets its own model. To tag words with their parts of speech, the tags are modelled as hidden states and the words are the observations. In computer networking, HMMs are used in intrusion detection systems. This has two flavours: anomaly detection in which normal behaviour is modelled; or misuse detection in which a predefined set of attacks is modelled. In computer vision, HMM has been used to label human activities from skeleton output. Each activity is modelled with a HMM. By linking multiple HMMs on common states, a compound HMM is formed. The purpose is to allow robots to be aware of human activity.

What are the different types of Hidden Markov Models?

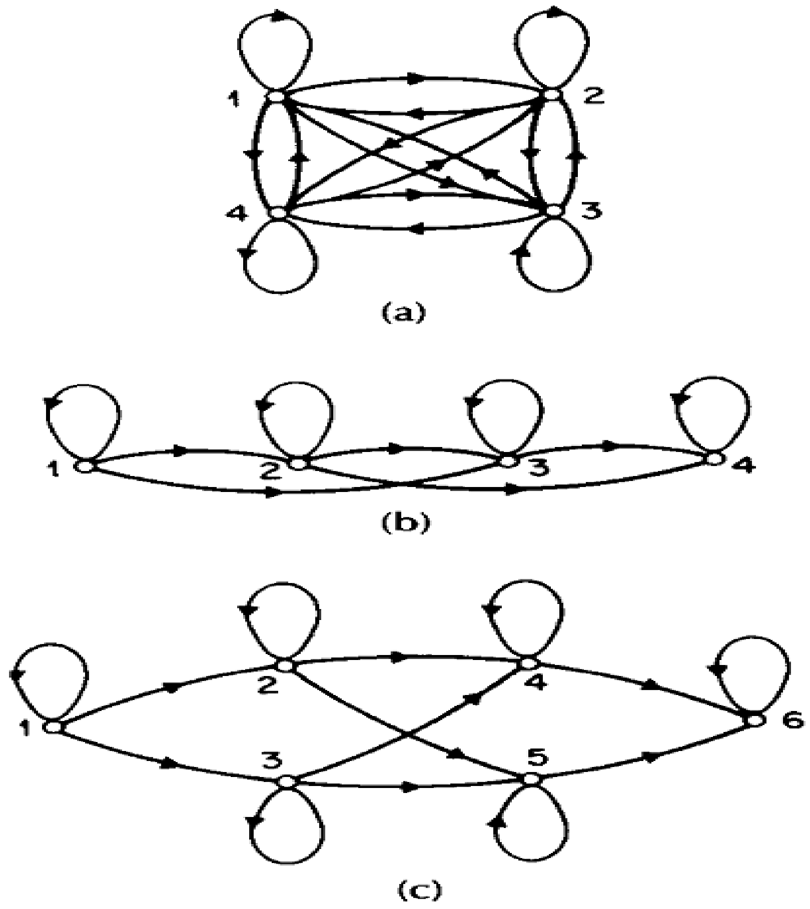


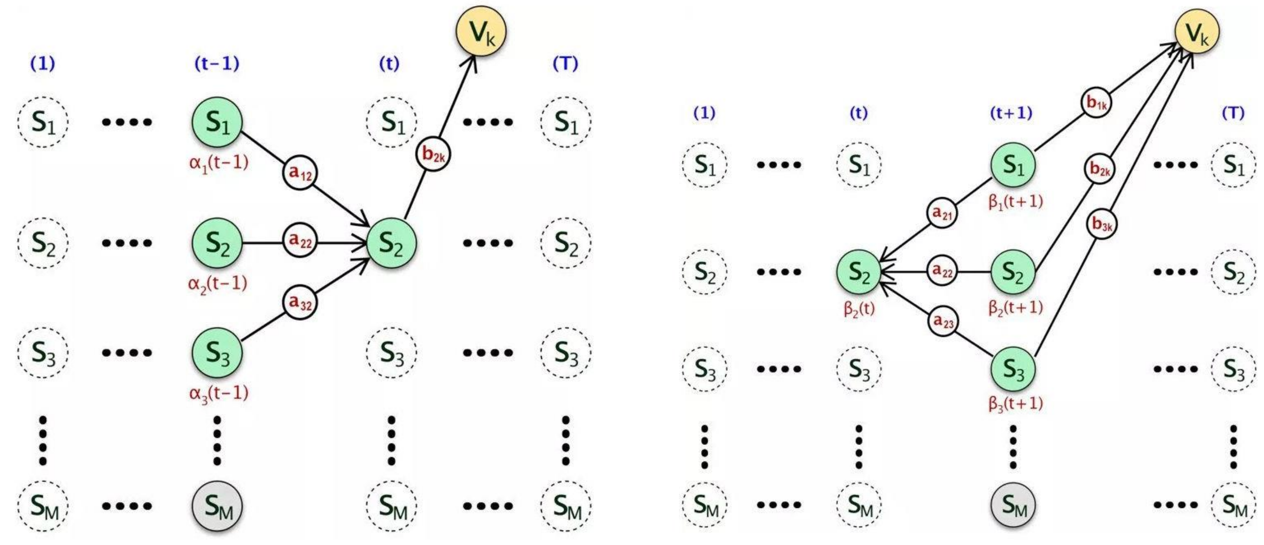
Fig. 7. Illustration of 3 distinct types of HMMs. (a) A 4-state ergodic model. (b) A 4-state left-right model. (c) A 6-state parallel path left-right model.

Some types of HMMs. Source: Rabiner 1989.

In the typical model, called the **ergodic HMM**, the states of the HMM are fully connected so that we can transition to a state from any other state. **Left-right HMM** is a more constrained model in which state transitions are allowed only from lower indexed states to higher indexed ones. Variations and combinations of these two types are possible, such as having two parallel left-to-right state paths. HMM started with observations of discrete symbols governed by **discrete** probabilities. If observations are continuous signals, then we would use **continuous** observation density.

There are also domain-specific variations of HMM. For example, in biological sequence analysis, there are at least three types including profile-HMMs, pair-HMMs, and context-sensitive HMMs.

- Could you explain forward algorithm and backward algorithm?



(a) Forward Algorithm (b) Backward Algorithm

-

Trellis diagrams showing forward and backward algorithms. Source: Adapted from Jana 2019b.

Every state sequence has a probability that it will lead to a given sequence of observations. Given T observations and N states, there are N^TNT possible state sequences. Thus, the complexity of calculating the probability of a given sequence of observations is $O(N^T T)$O(NTT). Both forward and backward algorithms bring down the complexity to $O(N^2 T)$O(N2T) through **dynamic programming**.

In the forward algorithm, we consider the probability of being in a state at the current time step. Then we consider the transition probabilities to calculate the state probabilities for the next step. Thus, at each time step we have considered all state sequences preceding it. The algorithm is more efficient since it reuses calculations from earlier steps. Instead of keeping all path sequences, paths are folded into a forward trellis. Backward algorithm is similar except that we start from the last time step and calculate in reverse. We're finding the probability that from a given state, the model will generate the output sequence that follows.

A combination of both algorithms, called forward-backward algorithm, is used to solve the learning problem.

- What's the algorithm for solving HMM's decoding problem?

  A simple explanation of Viterbi algorithm. Source: Chugg 2017.

  Viterbi algorithm solves HMM's decoding problem. It's similar to the forward algorithm except that instead of summing the probabilities of all paths leading to a state, we retain only one path that gives maximum probability. Thus, at every time step or iteration, given that we have N states, we retain only N paths, the most likely path for each state. For the next iteration, we use the most likely paths of current iteration and repeat the process.

  When we reach the end of the sequence, we'll have N most likely paths, each ending in a unique state. We then select the most likely end state. Once this selection is made, we backtrack to read the state sequence, that is, how we got to the end state. This state sequence is now the most likely sequence given our sequence of observations.

- How can we solve the learning problem of HMM?

  In HMM's learning problem, we are required to learn the transition (A) and observation (B) probabilities when given a sequence of observations and the vocabulary of hidden states. The **forward-backward algorithm** solves this problem. It's an iterative algorithm. It starts with an initial estimate of the probabilities and improves these estimates with each iteration.

  The algorithm consists of two steps:

- **Expectation or E-step**: We compute the expected state occupancy count and the expected state transition count based on current probabilities A and B.
- **Maximization or M-step**: We use the expected counts from the E-step to recompute A and B.

  While this algorithm is unsupervised, in practice, initial conditions are very important. For this reason, often extra information is given to the algorithm. For example, in speech recognition, the HMM structure is set manually and the model is trained to set the initial probabilities.

What is Object Tracking?

Object tracking is an application of deep learning where the program takes an initial set of object detections and develops a unique identification for each of the initial detections and then tracks the detected objects as they move around frames in a video. In other words, object tracking is the task of automatically identifying objects in a video and interpreting them as a set of trajectories with high accuracy. Often, there's an indication around the object being tracked, for example, a surrounding square that follows the object, showing the user where the object is on the screen.

Uses and Types of Object Tracking

Object tracking is used for a variety of use cases involving different types of input footage. Whether or not the anticipated input will be an image or a video, or a real-time video vs. a prerecorded video, impacts the algorithms used for creating object tracking applications. The kind of input also impacts the category, use cases, and applications of object tracking. Here, we will briefly describe a few popular uses and types of object tracking, such as video tracking, visual tracking, and image tracking.

Video Tracking

Video tracking is an application of object tracking where moving objects are located within video information. Hence, video tracking systems are able to process live, real-time footage and also recorded video files. The processes used to execute video tracking tasks differ based on which type of video input is targeted. This will be discussed more in-depth when we compare batch and online tracking methods later in this article. Visual tracking or visual target-tracking is a research topic in computer vision that is applied in a large range of everyday scenarios. The goal of visual tracking is to estimate the future position of a visual target that was initialized without the availability of the rest of the video.

Image Tracking

Image tracking is meant for detecting two-dimensional images of interest in a given input. That image is then continuously tracked as they move in the setting. Image tracking is ideal for datasets with highly contrasting images (ex. black and white), asymmetry, few patterns, and multiple identifiable differences between the image of interest and other images in the image set. Image tracking relies on computer vision to detect and augment images after image targets are predetermined.

Object tracking camera

Modern object tracking methods can be applied to real-time video streams of basically any camera. Therefore, the video feed of a USB camera or an IP camera can be used to perform object tracking, by feeding the individual frames to a tracking algorithm. Frame skipping or parallelized processing are common methods to improve object tracking performance with real-time video feeds of one or multiple cameras.

What makes Object Tracking difficult

What are the common challenges and advantages of Object Tracking?

The main challenges usually stem from issues in the image that make it difficult for object tracking models to effectively perform detections on the images. Here, we will discuss the few most common issues with the task of tracking objects and methods of preventing or dealing with these challenges.

1. Training and Tracking Speed

Algorithms for tracking objects are supposed to not only accurately perform detections and localize objects of interest but also do so in the least amount of time possible. Enhancing tracking speed is especially imperative for real-time object tracking models. To manage the time taken for a model to perform, the algorithm used to create the object tracking model needs to be either customized or chosen carefully. Fast R-CNN and Faster R-CNN can be used to increase the speed of the most common R-CNN approach. Since CNNs (Convolutional Neural Networks) are commonly used for object detection, CNN modifications can be the differentiating factor between a faster object tracking model and a slower one. Design choices besides the detection framework also influence the balance between speed and accuracy of an object detection model.

2. Background Distractions

The backgrounds of inputted images or images used to train object tracking models also impact the accuracy of the model. Busy backgrounds of objects meant to be tracked can make it harder for small objects to be detected. With a blurry or single-color background, it is easier for an AI system to detect and track objects. Backgrounds that are too busy, have the same color as the object, or that are too cluttered can make it hard to track results for a small object or a lightly colored object.

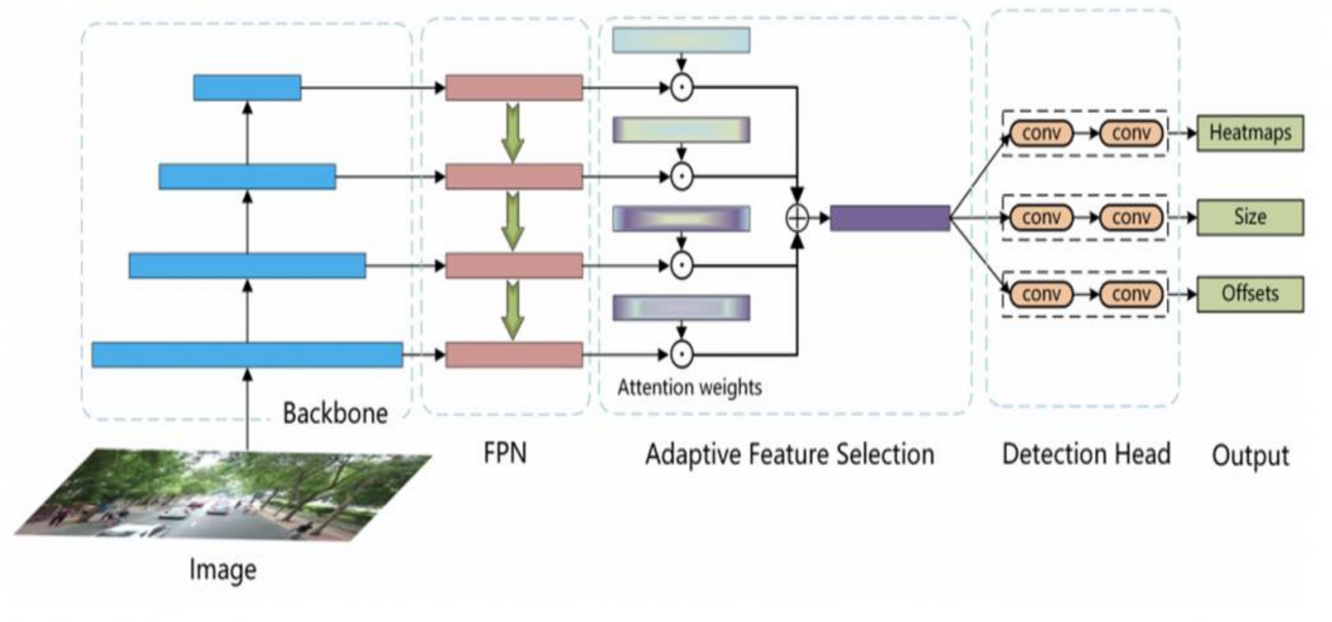
3. Multiple Spatial Scales

Objects meant to be tracked can come in a variety of sizes and aspect ratios. These ratios can confuse the object tracking algorithms into believing objects are scaled larger or smaller than their actual size. The size misconceptions can negatively impact detections or detection speed. To combat the issue of varying spatial scales, programmers can implement techniques such as feature maps, anchor boxes, image pyramids, and feature pyramids.

Anchor Boxes: Anchor boxes are a compilation of bounding boxes that have a specified height and width. The boxes are meant to acquire the scale and aspect ratios of objects of interest. They are chosen based on the average object size of the objects in a given dataset. Anchor boxes allow various types of objects to be detected without having the bounding box coordinates alternated during localization.

Feature Maps: A feature map is the output image of a layer when a Convolutional Neural Network (CNN) is used to capture the result of applying filters to that input image. Feature maps allow a deeper understanding of the features being detected by a CNN. Single-shot detectors have to consider the issue of multiple scales because they detect objects with just one pass through a CNN framework. This will occur in a detection decrease for small images. Small images can lose signal during down sampling in the pooling layers, which is when the CNN was trained on a low subset of those smaller images. Even if the number of objects is the same, down sampling can occur because the CNN wasn't able to detect the small images and count them towards the sample size. To prevent this, multiple feature maps can be used to allow single-shot detectors to look for objects within CNN layers – including earlier layers with higher resolution images. Single-shot detectors are still not an ideal option for small object tracking because of the difficulty they experience when detecting small objects. Tight groupings can prove especially difficult. For instance, overhead drone shots of a group of herd animals will be difficult to track using single-shot detectors.

Image and Feature Pyramid Representations: Feature pyramids, also known as multi-level feature maps because of their pyramidal structure, are a preliminary solution for object scale variation when using object tracking datasets. Hence, feature pyramids model the most useful information regarding objects of different sizes in a top-down representation and therefore make it easier to detect objects of varying sizes. Strategies such as image pyramids and feature pyramids are useful for preventing scaling issues. The feature pyramid is based on multi-scale feature maps, which uses less computational energy than image pyramids. This is because image pyramids consist of a set of resized versions of one input image that are then sent to the detector at testing.



4. Occlusion

Occlusion has a lot of definitions. In medicine, occlusion is the “blockage of a blood vessel” due to the vessel merging to a close; in deep learning, it has a similar meaning. In AI vision tasks using deep learning, occlusion happens when multiple objects come too close together (merge). This causes issues for object tracking systems because the occluded objects are seen as one or simply track the object incorrectly. The system can get confused and identify the initially tracked object as a new object. Occlusion sensitivity prevents this misidentification by allowing the user to understand which parts of an image are the most important for the object tracking system to classify. Occlusion sensitivity refers to a measure of the network’s sensitivity to occlusion in different data regions. It is done using small subsets of the original dataset.

Levels of Object Tracking

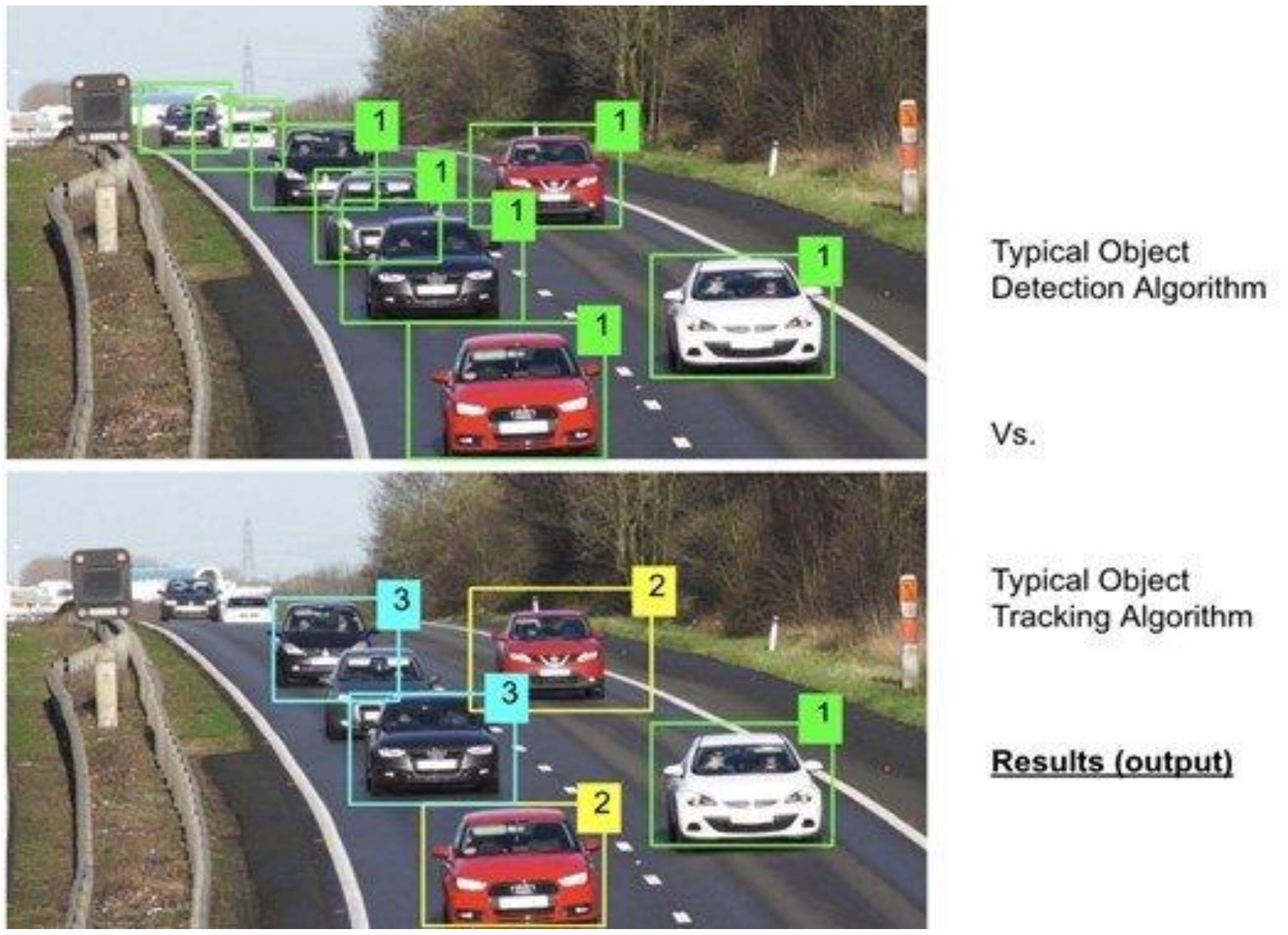
Object Tracking consists of multiple subtypes because it is such a broad application. Levels of object tracking differ depending on the number of objects being tracked.

Multiple Object Tracking (MOT)

Multiple object tracking is defined as the problem of automatically identifying multiple objects in a video and representing them as a set of trajectories with high accuracy. Hence, multi-object tracking aims to track more than one object in digital images. It is also called multi-target tracking, as it attempts to analyze videos to identify objects (“targets”) that belong to more than one predetermined class Multiple object tracking is of great importance in autonomous driving, where it is used to detect and predict the behavior of pedestrians or other vehicles. Hence, the algorithms are often benchmarked on the KITTI tracking test. KITTI is a challenging real-world computer vision benchmark and image dataset, popularly used in autonomous driving. In 2021, the best performing multiple object tracking algorithms are DEFT (88.95 MOTA, Multiple Object Tracking Accuracy), Center Track (89.44 MOTA), and SRK ODESA (90.03 MOTA).

Multiple Object Tracking (MOT) vs. General Object Detection

Object detections typically produce a collection of bounding boxes as outputs. Multiple object tracking often has little to no prior training regarding the appearance and number of targets. Bounding boxes are identified using their height, width, coordinates, and other parameters. Meanwhile, MOT algorithms additionally assign a target ID to each bounding box. This target ID is known as a detection, and it is important because it allows the model to distinguish among objects within a class. For example, instead of identifying all cars in a photo where multiple cars are present as just “car,” MOT algorithms attempt to identify different cars as being different from each other rather than all falling under the “car” label. For a visual representation of this metaphor, refer to the image below.


Typical Object Detection Algorithm
Vs.
Typical Object Tracking Algorithm
Results (output)
1
2
3

Single Object Tracking

Single Object Tracking (SOT) creates bounding boxes that are given to the tracker based on the first frame of the input image. Single Object Tracking is also sometimes known as Visual Object Tracking. SOT implies that one singular object is tracked, even in environments involving other objects. Single Object Trackers are meant to focus on one given object rather than multiple. The object of interest is determined in the first frame, which is where the object to be tracked is initialized for the first time. The tracker is then tasked with locating that unique target in all other given frames. SOT falls under the detection-free tracking category, which means that it requires manual initialization of a fixed number of objects in the first frame. These objects are then localized in consequent frames. A drawback of detection-free tracking is that it cannot deal with scenarios where new objects appear in the middle frames. SOT models should be able to track any given object.

Object Tracking Algorithms -Multiple Object Tracking (MOT) Algorithm Introduction

Most multiple object tracking algorithms incorporate an approach called tracking-by-detection. The tracking-by-detection method involves an independent detector that is applied to all image frames to obtain likely detections, and then a tracker, which is run on the set of detections. Hereby, the tracker attempts to perform data association (for example, linking the detections to obtain complete trajectories). The detections extracted from video inputs are used to guide the tracking process by connecting them and assigning identical IDs to bounding boxes containing the same target.

Batch method: Batch tracking algorithms use information from future video frames when deducing the identity of an object in a certain frame. Batch tracking algorithms use non-local information regarding the object. This methodology results in a better quality of tracking.

Online method: While batch tracking algorithms access future frames, online tracking algorithms only use present and past information to come to conclusions regarding a certain frame.

Online tracking methods for performing MOT generally perform worse than batch methods because of the limitation of online methods staying constrained to the present frame. However, this methodology is sometimes necessary because of the use case. For example, real-time problems requiring the tracking of objects, like navigation or autonomous driving, do not have access to future video frames, which is why online tracking methods are still a viable option.

Multiple Object Tracking Algorithm Stages

Most multiple object tracking algorithms contain a basic set of steps that remain constant as algorithms vary. Most of the so-called multi-target tracking algorithms share the following stages:

Stage #1: Designation or Detection: Targets of interest are noted and highlighted in the designation phase. The algorithm analyzes input frames to identify objects that belong to target classes. Bounding boxes are used to perform detections as part of the algorithm.

Stage #2: Motion: Feature extraction algorithms analyze detections to extract appearance and interaction features. A motion predictor, in most cases, is used to predict subsequent positions of each tracked target.

Stage #3: Recall: Feature predictions are used to calculate similarity scores between detection couplets. Those scores are then used to associate detections that belong to the same target. IDs are assigned to similar detections, and different IDs are applied to detections that are not part of pairs.

Some object tracking models are created using these steps separately from each other, while others combine and use the steps in conjunction. These differences in algorithm processing create unique models where some are more accurate than others.

Popular Object Tracking Algorithms

Convolutional Neural Networks (CNN) remain the most used and reliable network for object tracking. However, multiple architectures and algorithms are being explored as well. Among these algorithms are Recurrent Neural Networks (RNNs), Autoencoders (AEs), Generative Adversarial Networks (GANs), Siamese Neural Networks (SNNs), and custom neural networks.

Although object detectors can be used to track objects if it is applied frame-by-frame, this is a computationally limiting and therefore a rather inefficient method of performing object tracking. Instead, object detection should be applied once, and then the object tracker can handle every frame after the first. This is a more computationally effective and less cumbersome process of performing object tracking.

1. OpenCV Object Tracking

OpenCV object tracking is a popular method because OpenCV has so many algorithms built-in that are specifically optimized for the needs and objectives of object or motion tracking. Specific Open CV object trackers include the BOOSTING, MIL, KCF, CSRT, Median Flow, TLD, MOSSE, and GOTURN trackers. Each of these trackers is best for different goals. For example, CSRT is best when the user requires a higher object tracking accuracy and can tolerate slower FPS throughput. The selection of an OpenCV object tracking algorithm depends on the advantages and disadvantages of that specific tracker and the benefits:

The KCF tracker is not as accurate compared to the CSRT but provides comparably higher FPS.

The MOSSE tracker is very fast, but its accuracy is even lower than tracking with KCF. Still, if you are looking for the fastest object tracking OpenCV method, MOSSE is a good choice.

The GOTURN tracker is the only detector for deep learning-based object tracking with OpenCV. The original implementation of GOTURN is in Caffe, but it has been ported to the OpenCV Tracking API.

2. DeepSORT

DeepSORT is a good object tracking algorithm choice, and it is one of the most widely used object tracking frameworks. Appearance information is integrated within the algorithm, which vastly improves DeepSORT performance. Because of the integration, objects are trackable through longer periods of occlusion – reducing the number of identity switches.

For complete information on the inner workings of DeepSORT and specific algorithmic differences between DeepSORT and other algorithms, we suggest the article “Object Tracking using DeepSORT in TensorFlow 2” by Anushka Dhiman.

3. Object Tracking MATLAB

MATLAB is a numeric computing platform, which makes it different in its implementation compared to DeepSORT and OpenCV, but it is nevertheless a fine choice for visual tracking tasks. The Computer Vision Toolbox in MATLAB provides video tracking algorithms, such as continuously adaptive mean shift (CAMShift) and Kanade-Lucas-Tomasi (KLT) for tracking a single object or for use as building blocks in a more complex tracking system.

4. MDNet

MDNet is a fast and accurate, CNN-based visual tracking algorithm inspired by the R-CNN object detection network. It functions by sampling candidate regions and passing them through a CNN. The CNN is typically pre-trained on a vast dataset and refined at the first frame in an input video. Therefore, MDNet is most useful for real-time object tracking use cases. However, while it suffers from high computational complexity in terms of speed and space, it still is an accurate option. The computation-heavy aspects of MDNet can be minimized by performing RoI (Region of Interest) Pooling, however, which is a relatively effective way of avoiding repetitive observations and accelerating inference.