Introduction to artificial Intelligence and machine learning

What Is Artificial Intelligence?

Artificial Intelligence is a method of making a computer, a computer-controlled robot, or a software think intelligently like the human mind.

A Brief History of Artificial Intelligence

- Here’s a brief timeline of the past six decades of how AI evolved from its inception.
- 1956 - John McCarthy coined the term ‘Artificial Intelligence’ and had the first AI conference.
- 1969 - Shakey was the first general-purpose mobile robot built. It is now able to do things with a purpose vs. just a list of instructions.
- 1997 - Supercomputer ‘Deep Blue’ was designed, and it defeated the world champion chess player in a match. It was a massive milestone by IBM to create this large computer.

A Brief History of Artificial Intelligence

- 2002 - The first commercially successful robotic vacuum cleaner was created.
- 2005 - 2019 - Today, we have speech recognition, robotic process automation (RPA), a dancing robot, smart homes, and other innovations make their debut.
- 2020 - Baidu releases the LinearFold AI algorithm to medical and scientific and medical teams developing a vaccine during the early stages of the SARS-CoV-2 (COVID-19) pandemic. The algorithm can predict the RNA sequence of the virus in only 27 seconds, which is 120 times faster than other methods.

Types of Artificial Intelligence

- Below are the various types of AI:

1. Purely Reactive

- These machines do not have any memory or data to work with, specializing in just one field of work. For example, in a chess game, the machine observes the moves and makes the best possible decision to win.

2. Limited Memory

- These machines collect previous data and continue adding it to their memory. They have enough memory or experience to make proper decisions, but memory is minimal. For example, this machine can suggest a restaurant based on the location data that has been gathered.

Types of Artificial Intelligence

3. Theory of Mind

- This kind of AI can understand thoughts and emotions, as well as interact socially. However, a machine based on this type is yet to be built.

4. Self-Aware

- Self-aware machines are the future generation of these new technologies. They will be intelligent, sentient, and conscious.

Goals of Artificial Intelligence

- It helps you reduce the amount of time needed to perform specific tasks.
- Making it easier for humans to interact with machines.
- Facilitating human-computer interaction in a way that is more natural and efficient.
- Improving the accuracy and speed of medical diagnoses.
- Helping people learn new information more quickly.
- Enhancing communication between humans and machines.

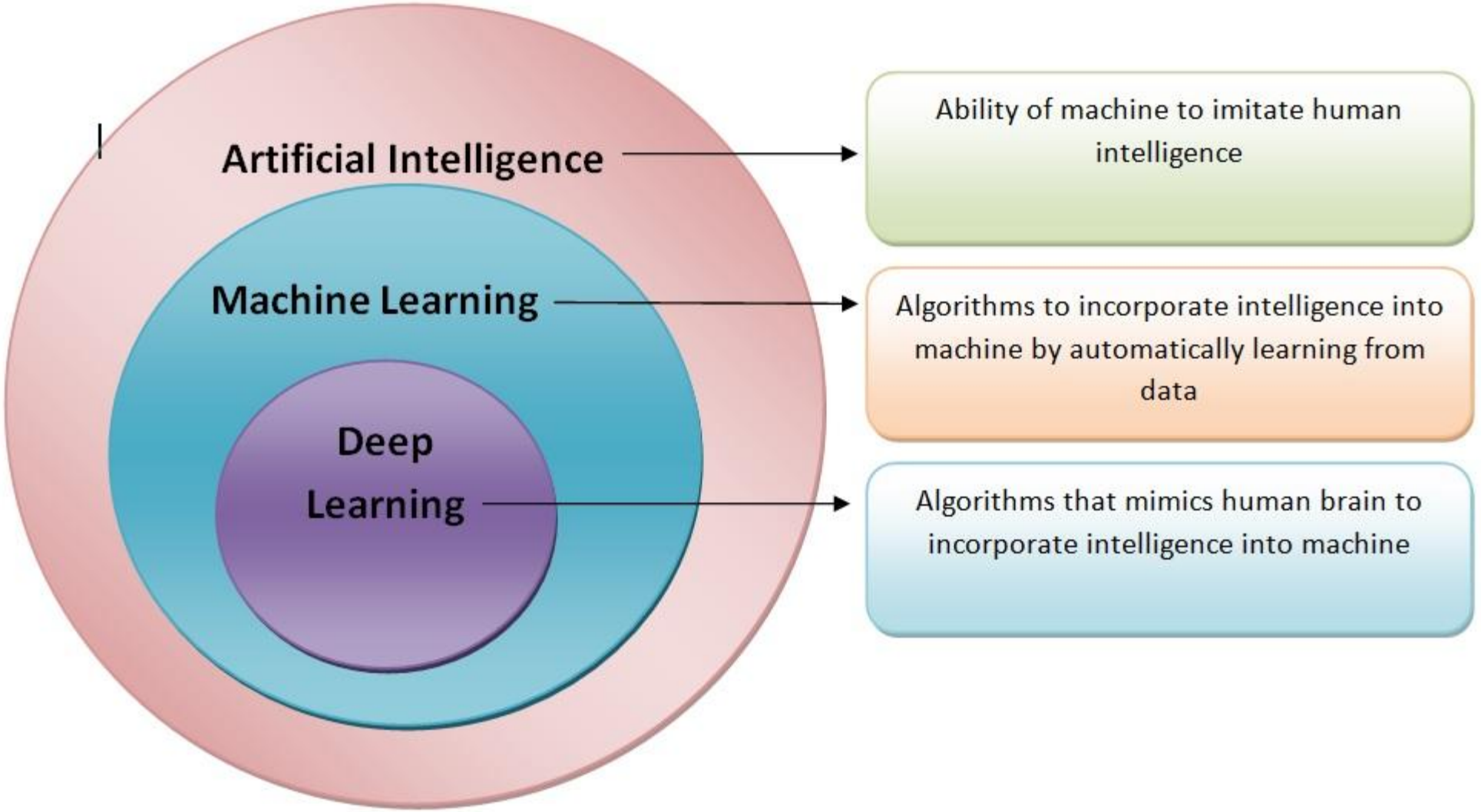
Subfields of Artificial Intelligence

Here, are some important subfields of Artificial Intelligence:

Machine Learning: Machine learning is the art of studying algorithms that learn from examples and experiences. Machine learning is based on the idea that some patterns in the data were identified and used for future predictions. The difference from hard coding rules is that the machine learns to find such rules.

Deep Learning: Deep learning is a sub-field of machine learning. Deep learning does not mean the machine learns more in-depth knowledge; it uses different layers to learn from the data. The depth of the model is represented by the number of layers in the model. For instance, the Google LeNet model for image recognition counts 22 layers.

AI VS ML VS Dl

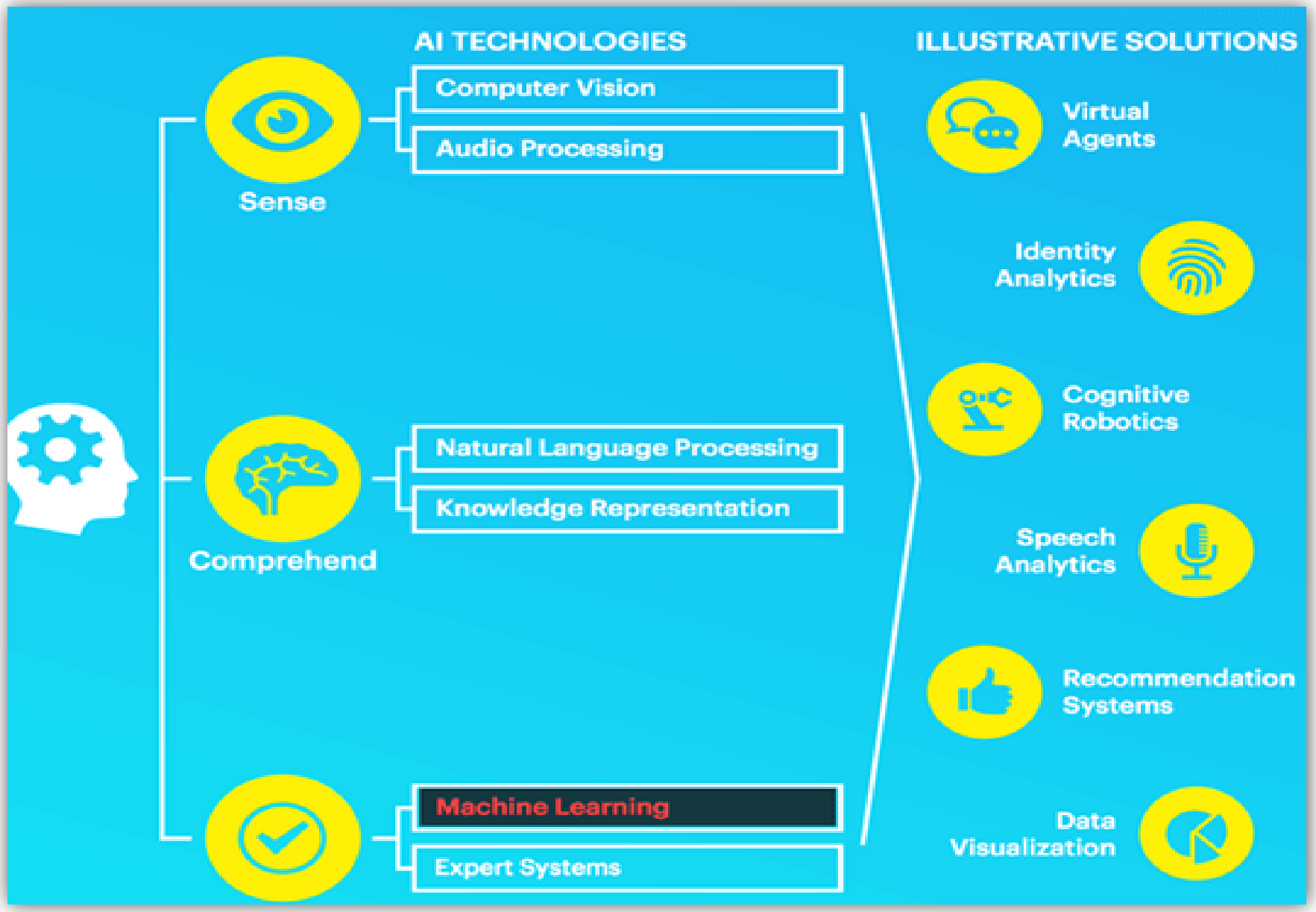

Artificial Intelligence
Machine Learning
Deep Learning
Ability of machine to imitate human intelligence
Algorithms to incorporate intelligence into machine by automatically learning from data
Algorithms that mimics human brain to incorporate intelligence into machine

AI Vs Machine Learning

Most of our smartphone, daily device or even the internet uses Artificial Intelligence. Very often, AI and machine learning are used interchangeably by big companies that want to announce their latest innovation. However, Machine learning and AI are different in some ways.

Artificial Intelligence is a computer that is given human-like properties. Take our brain; it works effortlessly and seamlessly to calculate the world around us.

Machine learning is a distinct subset of AI that trains a machine to learn. Machine learning models look for patterns in data and try to conclude. In a nutshell, the machine does not need to be explicitly programmed by people. The programmers give some examples, and the computer is going to learn what to do from those samples.


AI TECHNOLOGIES
ILLUSTRATIVE SOLUTIONS
Sense
Computer Vision
Audio Processing
Comprehend
Natural Language Processing
Knowledge Representation
Machine Learning
Expert Systems
Virtual Agents
Identity Analytics
Cognitive Robotics
Speech Analytics
Recommendation Systems
Data Visualization

Applications of Artificial intelligence

Advantages of AI

- Reduction in Human Error
- Zero Risk
- 24x7 Availability
- Digital Assistance
- New Inventions
- Unbiased Decisions
- Perform Repetitive Jobs
- Daily Applications

Disadvantages of AI

- High Cost
- No Creativity
- Unemployment
- Make Human Lazy

“Thank You”

Overview of Machine Learning Techniques

What is Machine Learning?

- ML is a branch of artificial intelligence
  - Uses computing-based systems to make sense out of data
    - Extracting patterns, fitting data to functions, classifying data, etc.
  - ML systems can learn and improve
    - With historical data, time, and experience
  - Bridges theoretical computer science and real noise data.

What is Machine Learning?

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

- Herbert Simon

Definition by Tom Mitchell (1998):

Machine Learning is the study of algorithms that

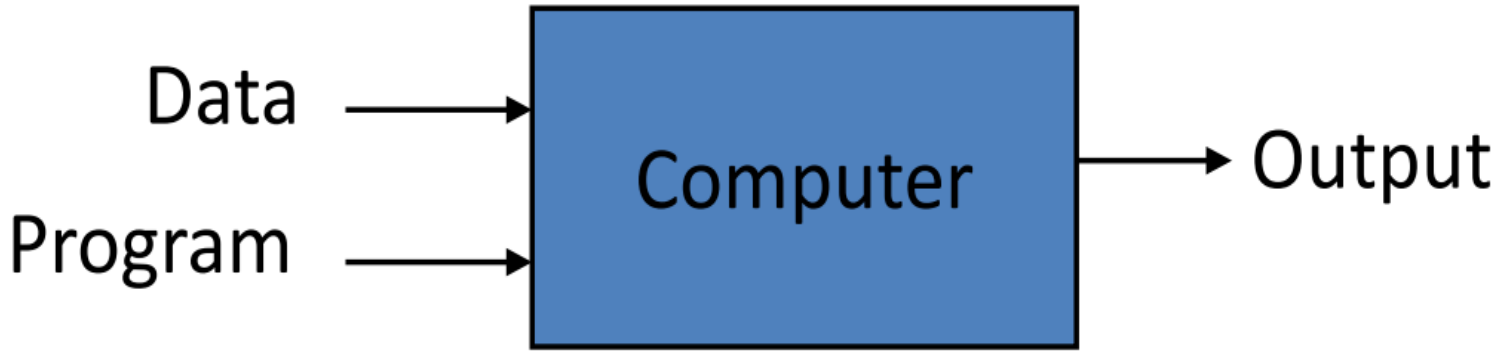
- improve their performance P
- at some task T
- with experience E.

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

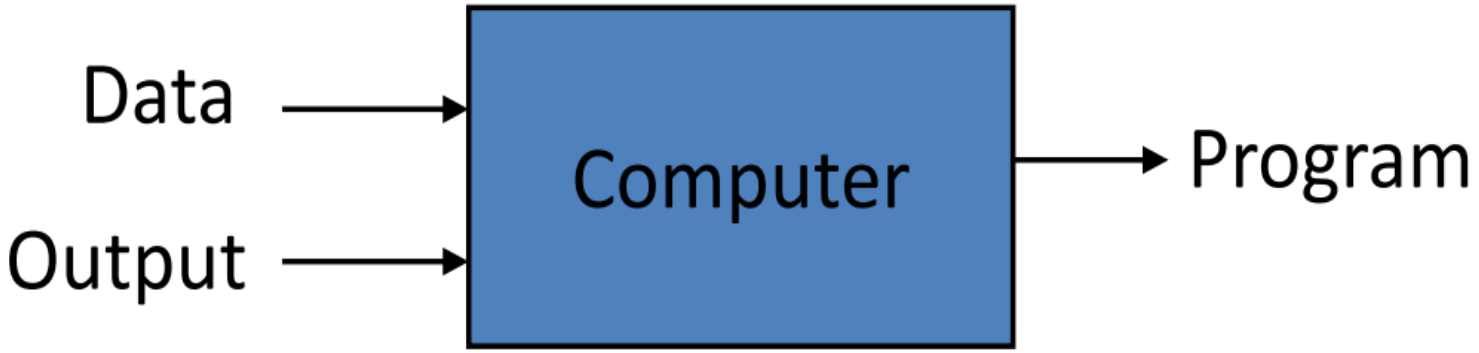
"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. **By Tom Mitchell**

Machine learning: A computer program that learns from experience is called a machine learning program or simply called a learning program

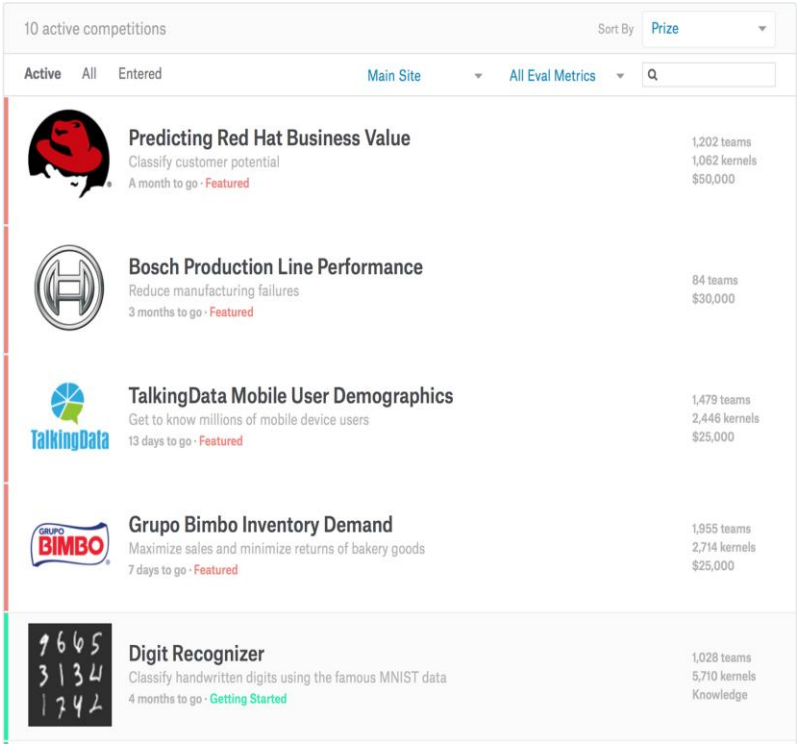
Traditional Programming



Machine Learning



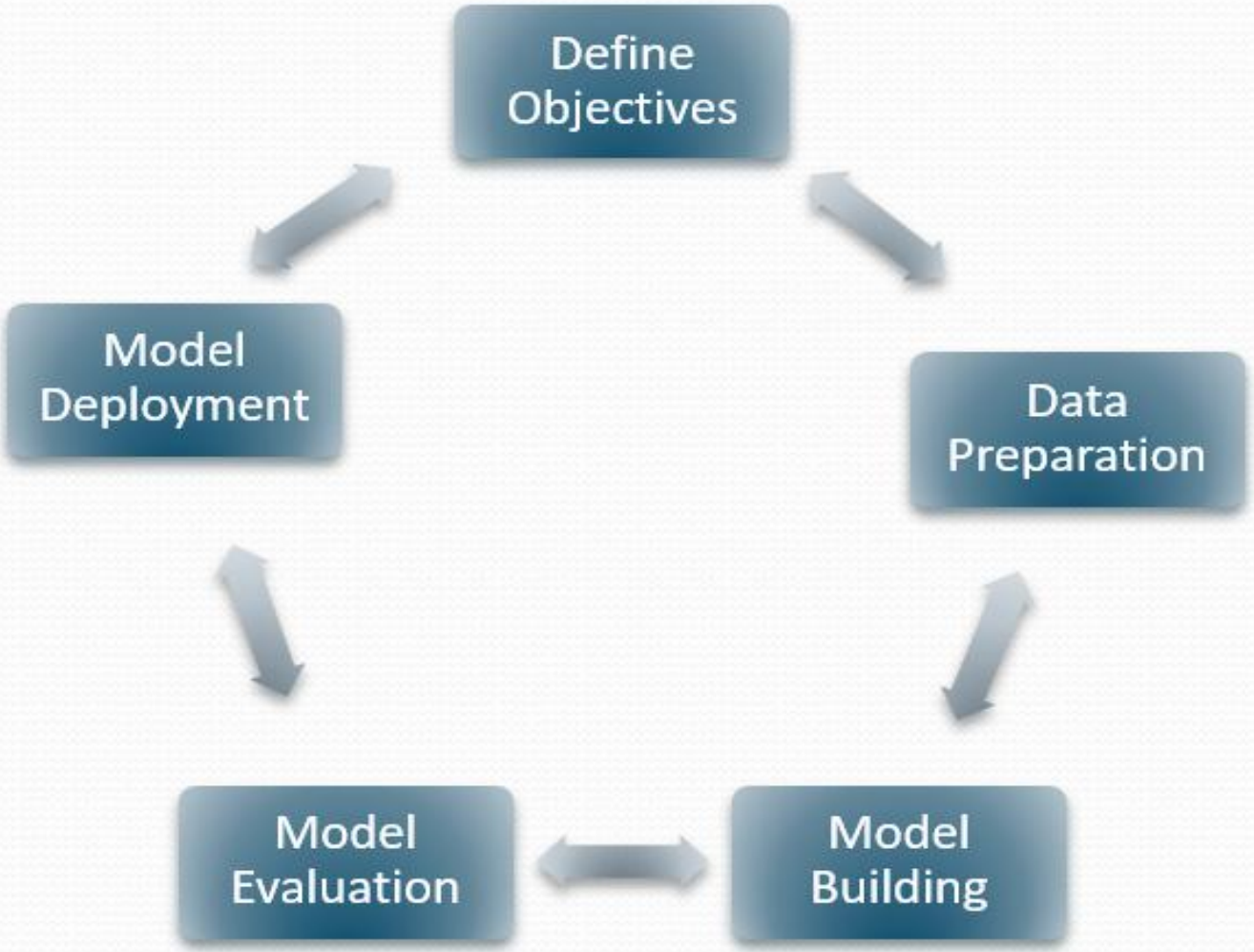
ML in real-life



Growth of Machine Learning

- Machine learning is the preferred approach to
  - Speech recognition, Natural language processing
  - Computer vision
  - Medical outcomes analysis
  - Robot control
  - Computational biology
- This trend is accelerating
  - Improved machine learning algorithms
  - Improved data capture, networking, faster computers
  - Software too complex to write by hand
  - New sensors / IO devices
  - Demand for self-customization to user, environment

Machine Learning as a Process



Applications

- Association Analysis
- Supervised Learning
  - Classification
  - Regression/Prediction
- Unsupervised Learning
- Reinforcement Learning

Learning Associations

- Basket Analysis

$P(Y \mid X)$ probability that somebody who buys X also buys Y where X and Y are products/services.

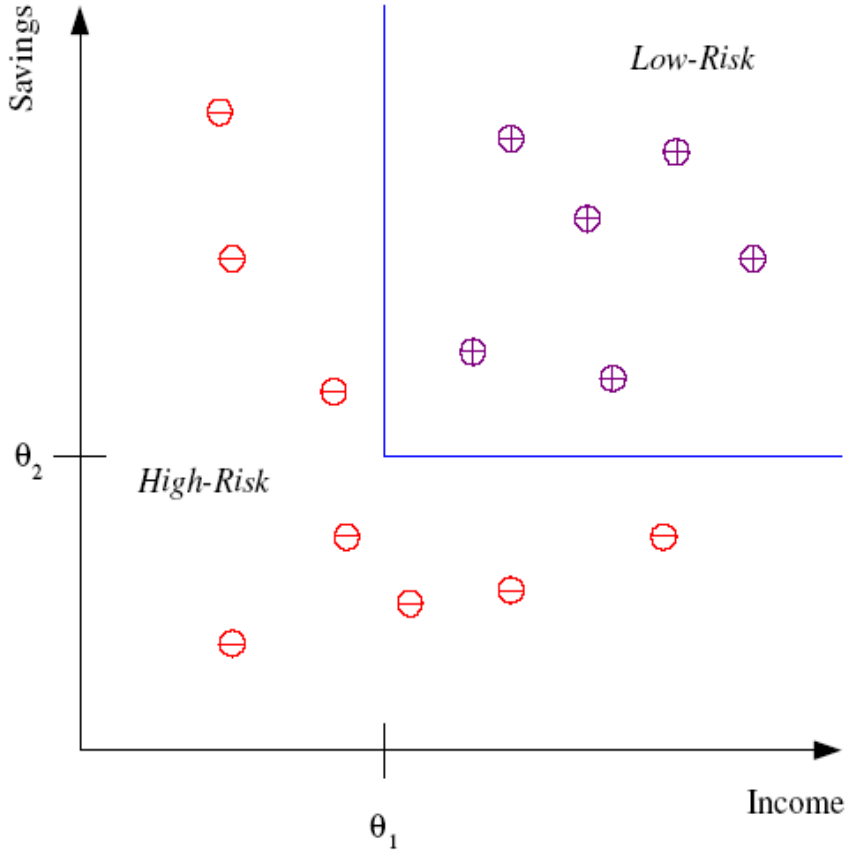
Example: P (chips | beer) = 0.7

Market-Basket transactions

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Classification

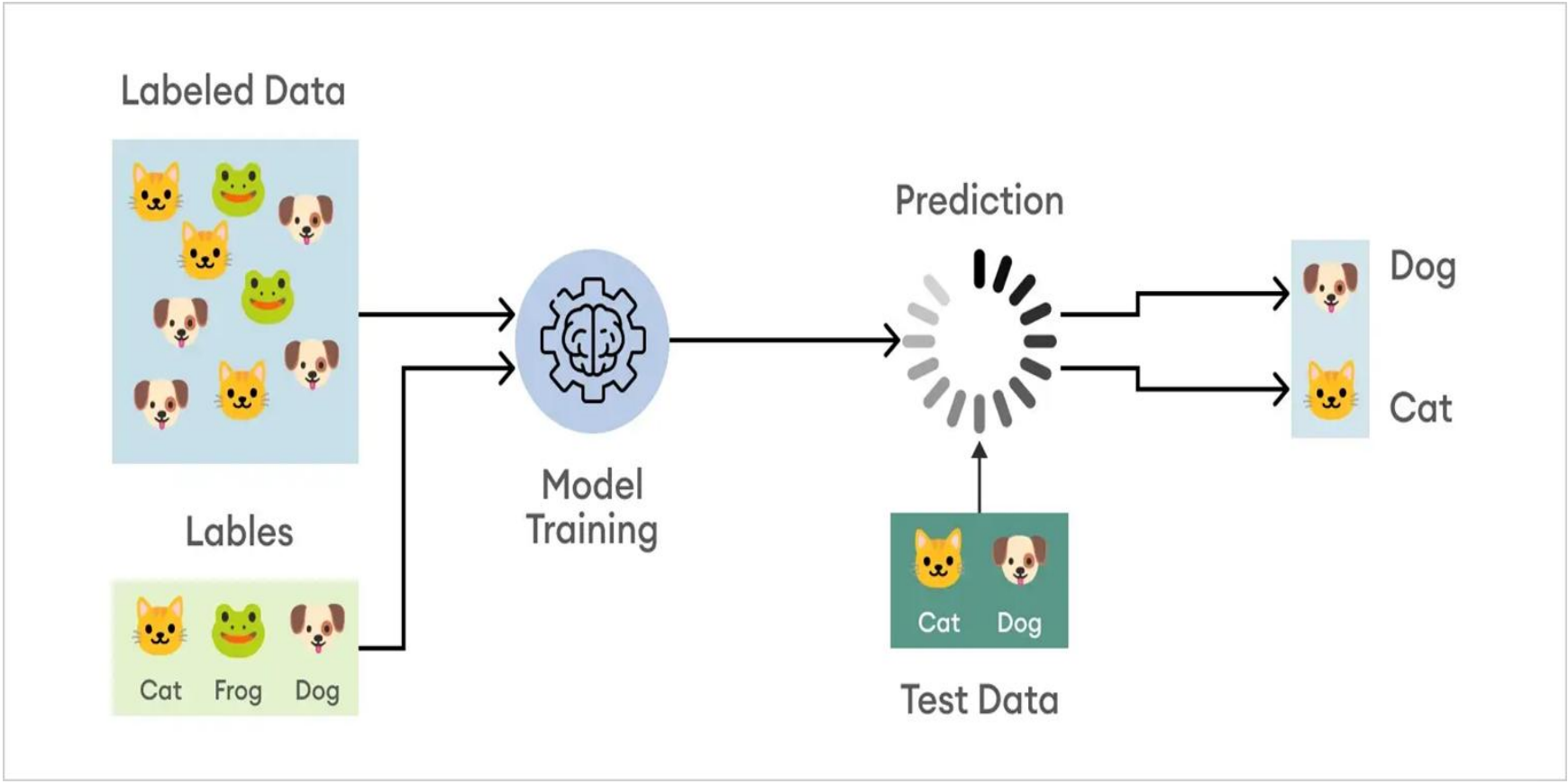
- Example: Credit scoring
- Differentiating between low-risk and high-risk customers from their *income* and *savings*



Discriminant: IF *income* $> \theta_1$ AND *savings* $> \theta_2$
THEN low-risk ELSE high-risk

Model

Classification Example



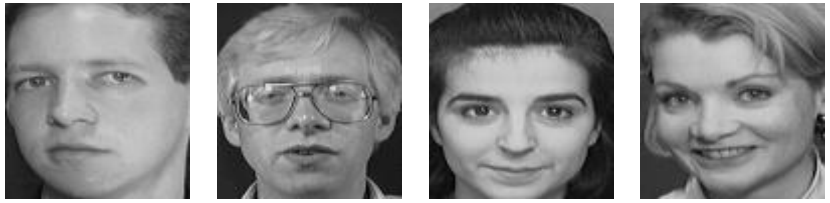
Classification: Applications

- Aka Pattern recognition
- Face recognition: Pose, lighting, occlusion (glasses, beard), make-up, hair style
- Character recognition: Different handwriting styles.
- Speech recognition: Temporal dependency.
  - Use of a dictionary or the syntax of the language.
  - Sensor fusion: Combine multiple modalities; eg, visual (lip image) and acoustic for speech
- Medical diagnosis: From symptoms to illnesses
- Web Advertising: Predict if a user clicks on an ad on the Internet.

Face Recognition

Training examples of a person

Test Images



Speech Recognition

Prediction: Regression

- Example: Price of a used car

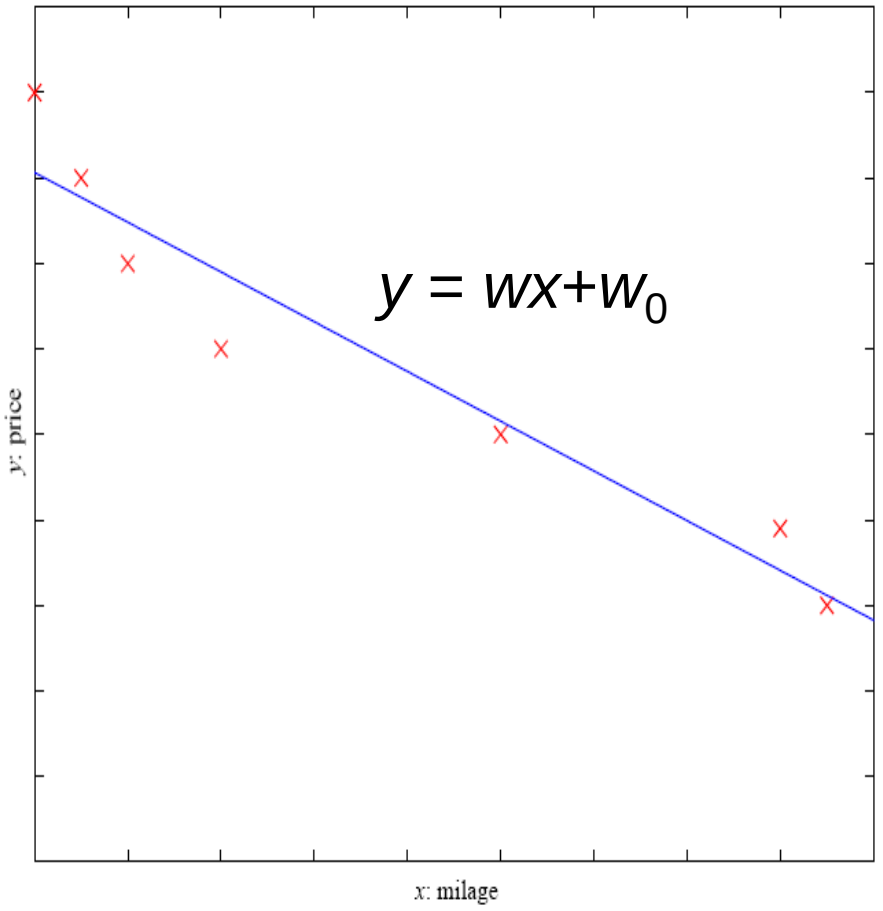
- x : car attributes

  y : price

  $$y = g\,(x \mid \theta\,)$$

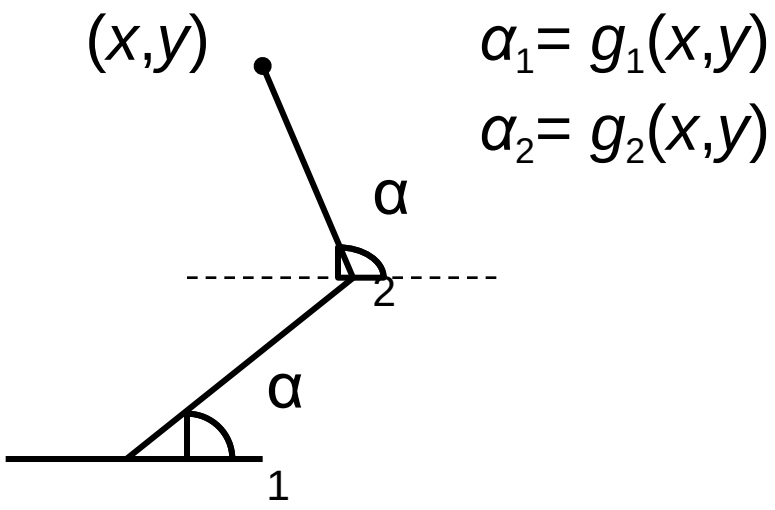
  $g\,(\,)$ model,

  θ parameters



Regression Applications

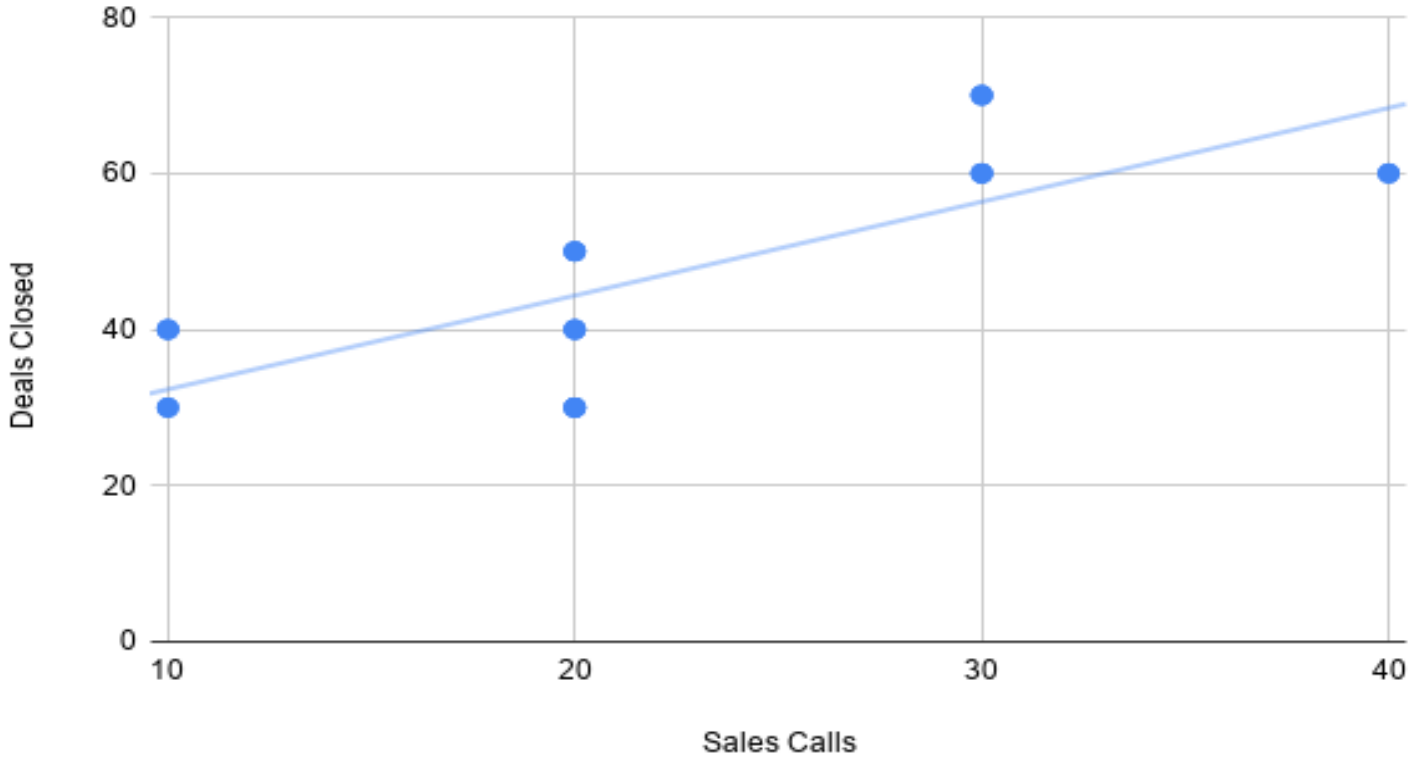
- Navigating a car: Angle of the steering wheel (CMU NavLab)
- Kinematics of a robot arm



$\alpha_1 = g_1(x,y)$

$\alpha_2 = g_2(x,y)$

Sales Forecasting Using Regression Analysis



Supervised Learning: Uses

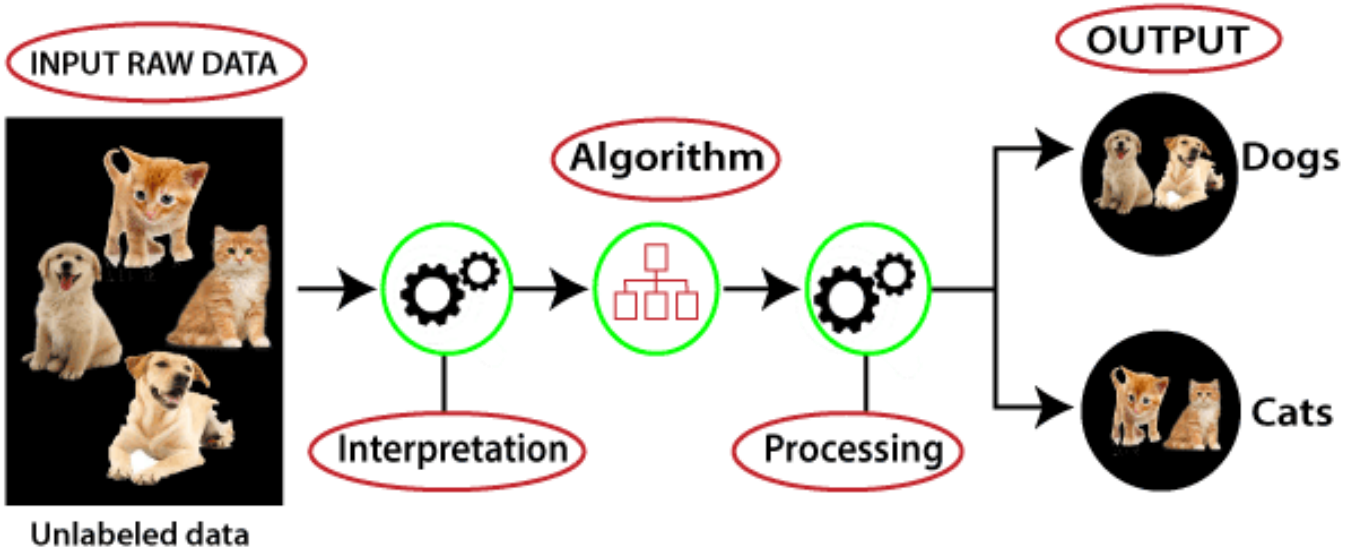
Example: Decision trees tools that create rules

- Prediction of future cases: Use the rule to predict the output for future inputs
- Knowledge extraction: The rule is easy to understand
- Compression: The rule is simpler than the data it explains
- Outlier detection: Exceptions that are not covered by the rule, e.g., fraud

Unsupervised Learning

- Learning “what normally happens”
- No output
- Clustering: Grouping similar instances
- Other applications: Summarization, Association Analysis
- Example applications
  - Customer segmentation in CRM
  - Image compression: Color quantization
  - Bioinformatics: Learning motifs

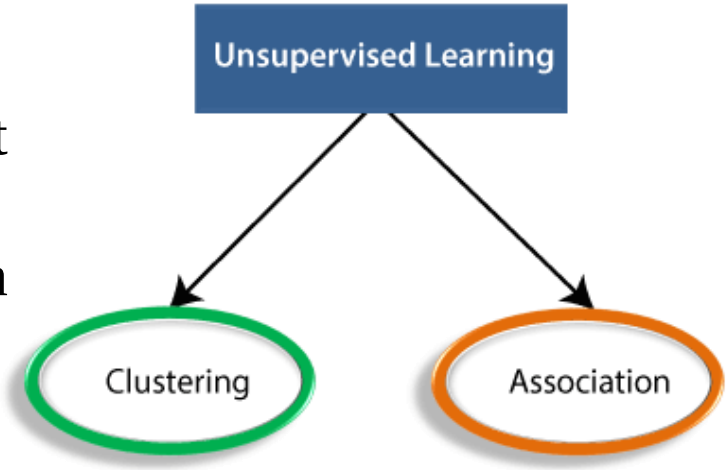
Working of Unsupervised Learning



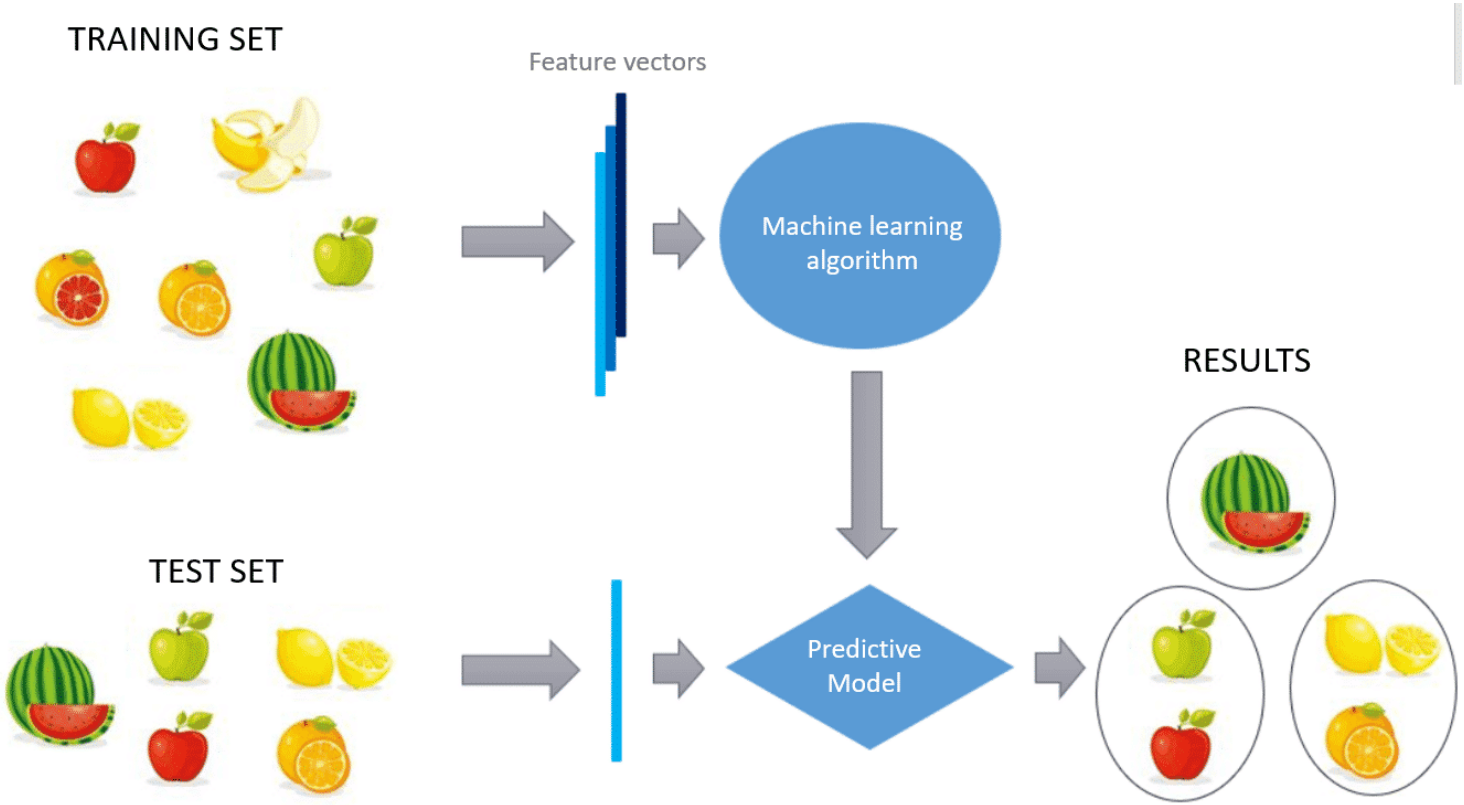
Types of Unsupervised Learning Algorithm

Clustering: Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group.

Association: An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset.



Clustering Example



Unsupervised Learning algorithms

Below is the list of some popular unsupervised learning algorithms:

- K-means clustering
- KNN (k-nearest neighbors)
- Hierarchal clustering
- Anomaly detection
- Neural Networks
- Principle Component Analysis

Reinforcement Learning

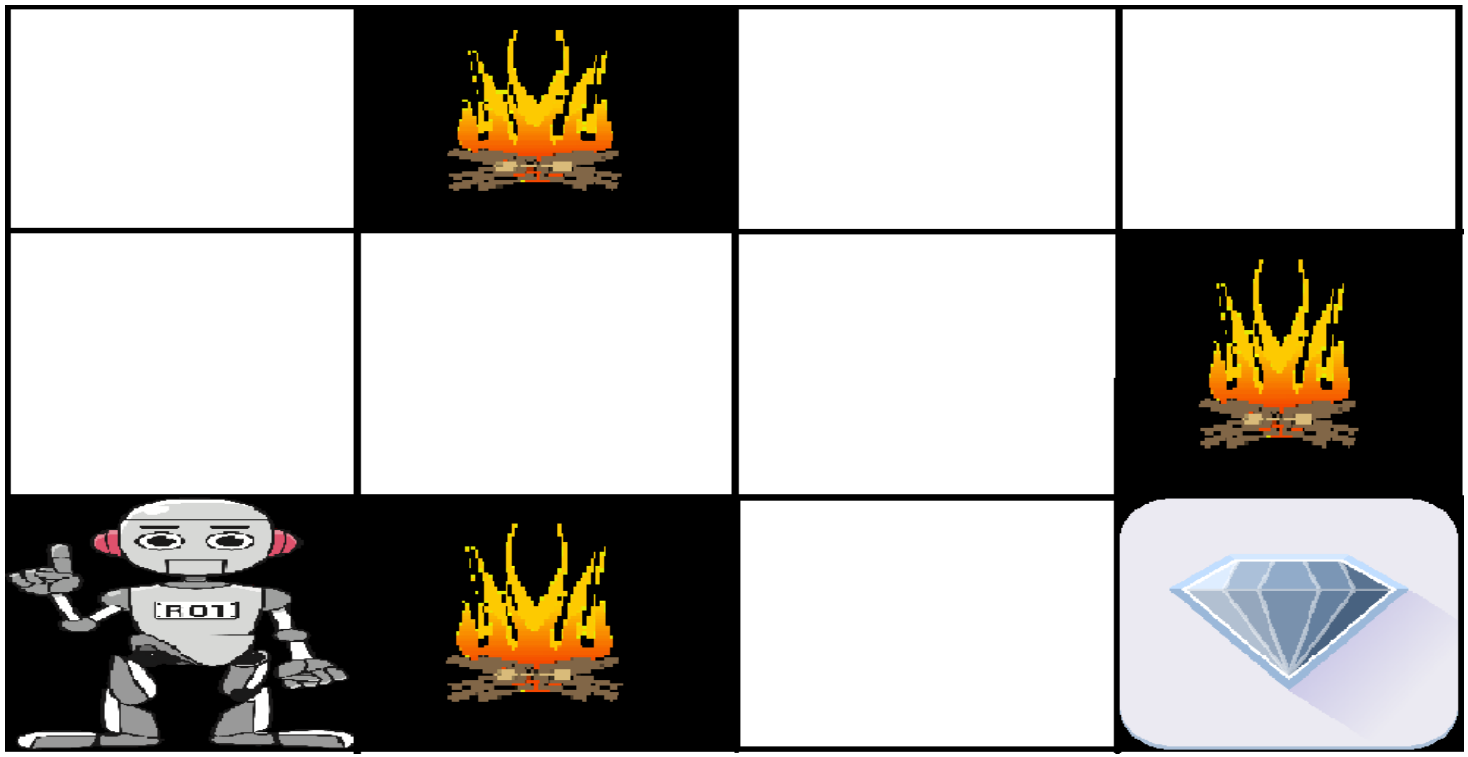
Topics

- Policies: what actions should an agent take in a particular situation
- Utility estimation: how good is a state (used by policy)

- No supervised output but delayed reward
- Credit assignment problem (what was responsible for the outcome)
- Applications:
  - Game playing
  - Robot in a maze
  - Multiple agents, partial observability, ...

Example

- The problem is as follows: We have an agent and a reward, with many hurdles in between. The agent is supposed to find the best possible path to reach the reward. The following problem explains the problem more easily.



- The above image shows the robot, diamond, and fire.
- The goal of the robot is to get the reward that is the diamond and avoid the hurdles that are fired.
- The robot learns by trying all the possible paths and then choosing the path which gives him the reward with the least hurdles.
- Each right step will give the robot a reward and each wrong step will subtract the reward of the robot.
- The total reward will be calculated when it reaches the final reward that is the diamond.

Main points in Reinforcement learning

- Input: The input should be an initial state from which the model will start
- Output: There are many possible outputs as there are a variety of solutions to a particular problem
- Training: The training is based upon the input, The model will return a state and the user will decide to reward or punish the model based on its output.
- The model keeps continues to learn.
- The best solution is decided based on the maximum reward.

“Thank You”

Probability Distributions

What Is Probability?

- Probability denotes the possibility of something happening.
- It is a mathematical concept that predicts how likely events are to occur. The probability values are expressed between 0 and 1.
- The definition of probability is the degree to which something is likely to occur.

What Are Probability Distributions?

- A probability distribution is a statistical function that describes all the possible values and probabilities for a random variable within a given range.
- This range will be bound by the minimum and maximum possible values, but where the possible value would be plotted on the probability distribution will be determined by a number of factors.
- The mean (average), standard deviation, skewness, and kurtosis of the distribution are among these factors.

Types of Probability Distribution

The probability distribution is divided into two parts:

- Discrete Probability Distributions
- Continuous Probability Distributions

Discrete Probability Distribution

- A discrete distribution describes the probability of occurrence of each value of a discrete random variable.
- The number of spoiled apples out of 6 in your refrigerator can be an example of a discrete probability distribution.
- Each possible value of the discrete random variable can be associated with a non-zero probability in a discrete probability distribution.
- Let's discuss some significant probability distribution functions.

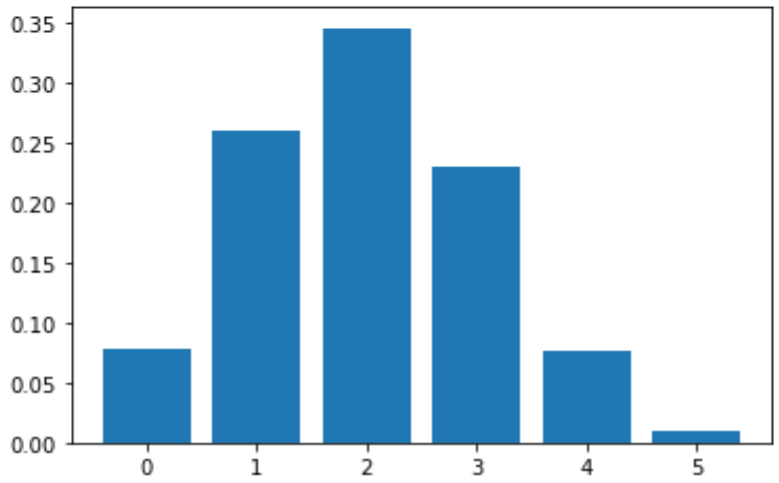
Binomial Distribution

- The binomial distribution is a discrete distribution with a finite number of possibilities. When observing a series of what are known as Bernoulli trials, the binomial distribution emerges. A Bernoulli trial is a scientific experiment with only two outcomes: success or failure.
- Consider a random experiment in which you toss a biased coin six times with a 0.4 chance of getting head. If 'getting a head' is considered a ‘success’, the binomial distribution will show the probability of r successes for each value of r.
- The binomial random variable represents the number of successes (r) in n consecutive independent Bernoulli trials.

Binomial Distribution

- Here's a Python Code to show Binomial distribution:

```
from scipy.stats import binom
import matplotlib.pyplot as plt
# setting the values
# of n and p
n = 5
p = 0.4
# defining list of r values
r_values = list(range(n + 1))
# list of pmf values
dist = [binom.pmf(r, n, p) for r in r_values ]
# plotting the graph
plt.bar(r_values, dist)
plt.show()
```



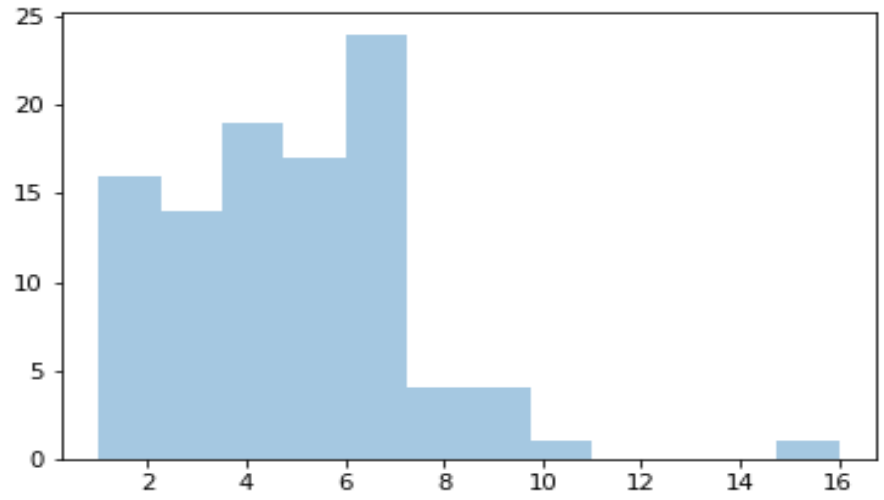
Poisson Distribution

- A Poisson distribution is a probability distribution used in statistics to show how many times an event is likely to happen over a given period of time. To put it another way, it's a count distribution. Poisson distributions are frequently used to comprehend independent events at a constant rate over a given time interval. Siméon Denis Poisson, a French mathematician, was the inspiration for the name.
- The Python code below shows a simple example of Poisson distribution.
- It has two parameters:
- Lam: Known number of occurrences
- Size: The shape of the returned array

Poisson Distribution

- The below-given Python code generates the 1x100 distribution for occurrence 5.

```
from numpy import random
import matplotlib.pyplot as plt
import seaborn as sns

sns.distplot(random.poisson(lam=5, size=100), kde=False)

plt.show()
```



Continuous Probability Distributions

- A continuous distribution describes the probabilities of a continuous random variable's possible values.
- A continuous random variable has an infinite and uncountable set of possible values (known as the range).
- The mapping of time can be considered as an example of the continuous probability distribution. It can be from 1 second to 1 billion seconds, and so on.
- The area under the curve of a continuous random variable's PDF is used to calculate its probability. As a result, only value ranges can have a non-zero probability. A continuous random variable's probability of equaling some value is always zero.
- Now, look at some varieties of the continuous probability distribution.

Normal Distribution

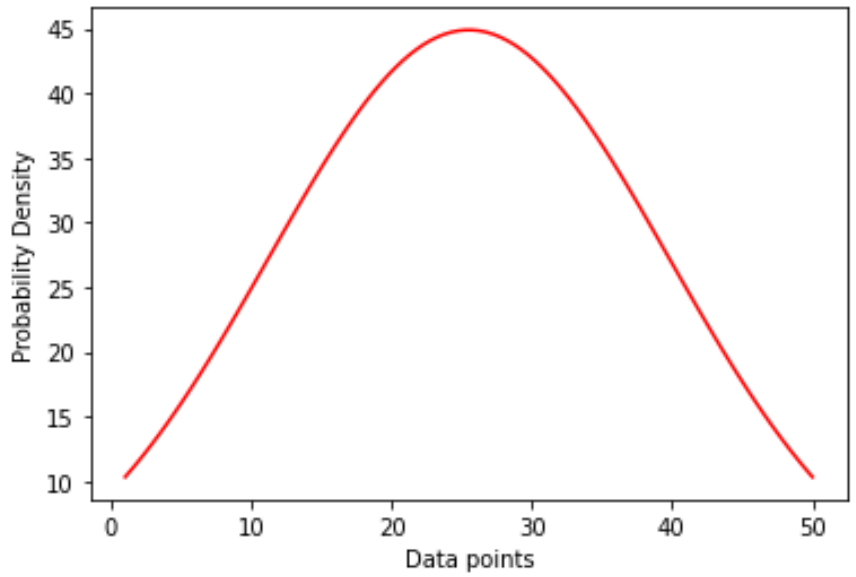
- Normal Distribution is one of the most basic continuous distribution types.
- Gaussian distribution is another name for it.
- Around its mean value, this probability distribution is symmetrical.
- It also demonstrates that data close to the mean occurs more frequently than data far from it. Here, the mean is 0, and the variance is a finite value.

Normal Distribution

- In the example, you generated 100 random variables ranging from 1 to 50. After that, you created a function to define the normal distribution formula to calculate the probability density function. Then, you have plotted the data points and probability density function against X-axis and Y-axis, respectively.

```
import numpy as np
import matplotlib.pyplot as plt

# Creating a series of data of in range of 1-50.
x = np.linspace(1,50,100)

#Creating a Function.
def normal_dist(x , mean , sd):
    prob_density = (np.pi*sd) * np.exp(-0.5*((x-mean)/sd)**2)
    return prob_density

#Calculate mean and Standard deviation.
mean = np.mean(x)
sd = np.std(x)

#Apply function to the data.
pdf = normal_dist(x,mean,sd)

#Plotting the Results
plt.plot(x,pdf , color = 'red')
plt.xlabel('Data points')
plt.ylabel('Probability Density')
```

```
Text(0, 0.5, 'Probability Density')
```



Uniform Distribution

- In continuous uniform distribution, all outcomes are equally possible.
- Each variable has the same chance of being hit as a result.
- Random variables are spaced evenly in this symmetric probabilistic distribution, with a 1/(b-a) probability.

Uniform Distribution

- The below Python code is a simple example of continuous distribution taking 1000 samples of random variables.

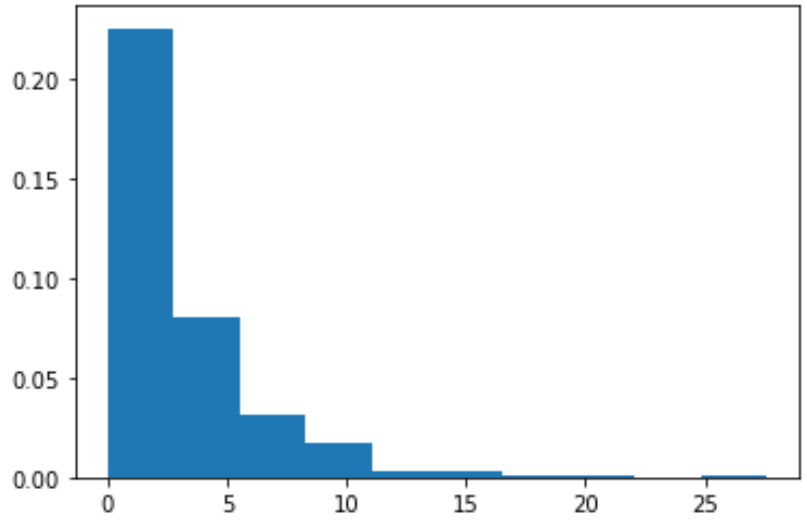
```
from numpy import random
import matplotlib.pyplot as mpl
import seaborn as sb
def uniformDist():
    sb.distplot(random.uniform(size = 1000), hist = True)
    mpl.show()

uniformDist()
```


```
from numpy import random
import matplotlib.pyplot as mpl
import seaborn as sb
def uniformDist():
    sb.distplot(random.uniform(size = 1000), hist = True)
    mpl.show()

uniformDist()
```

Exponential Distribution

- In a Poisson process, an exponential distribution is a continuous probability distribution that describes the time between events (success, failure, arrival, etc.).
- You can see in the below example how to get random samples of exponential distribution and return NumPy array samples by using the numpy.random.exponential() method.

```
# import exponential
import numpy as np
import matplotlib.pyplot as plt

# Using exponential() method
gfg = np.random.exponential(3, 1000)

count, bins, ignored = plt.hist(gfg, 10, density = True)
plt.show()
```



Introduction to Bayesian Learning

- Bayesian reasoning provides a probabilistic approach to inference.
- Bayesian reasoning provides the basis for learning algorithms that directly manipulate probabilities, as well as a framework for analyzing the operation of other algorithms that do not explicitly manipulate probabilities.
- Bayesian learning uses Bayes' theorem to determine the conditional probability of a hypotheses given some evidence or observations.

What is Bayes' Theorem?

- Bayes theorem is also known as the Bayes Rule or Bayes Law. It is used to determine the conditional probability of event A when event B has already happened. The general statement of Bayes' theorem is "The conditional probability of an event A, given the occurrence of another event B, is equal to the product of the event of B, given A and the probability of A divided by the probability of event B." i.e.

$$P(A|B) = P(B|A)P(A) / P(B)$$

where,

P(A) and P(B) are the probabilities of events A and B

P(A|B) is the probability of event A when event B happens

P(B|A) is the probability of event B when A happens

Bayesian learning

- In machine learning we are often interested in determining the best hypothesis from some space H, given the observed training data D. best hypothesis is to say that we demand the most probable hypothesis, given the data D plus any initial knowledge about the prior probabilities of the various hypotheses in H.
- Bayes theorem provides a direct method for calculating such probabilities. More precisely,
- Bayes theorem provides a way to calculate the probability of a hypothesis based on its prior probability, the probability of observing various data given the hypothesis, and the observed data itself.

Notations in Bayesian learning

Prior Probability

- **P(h)** to denote the initial probability that hypothesis **h** holds, before we have observed the training data.
- **P(h)** is called the prior probability of **h** and may reflect any background knowledge we have about the chance that **h** is a correct hypothesis.
- **P(D/h)** : the probability of observing data **D** given some world in which hypothesis **h** holds.

Posterior probability

- P (h/ D) is called the posterior probability of **h**, because it reflects our confidence that **h** holds after we have seen the training data **D**.

Bayesian learning

- Bayes theorem is the corner stone of Bayesian learning methods because it provides a way to calculate the posterior probability P(hlD), from the priorprobability P(h), together with P(D) and P(D(h).

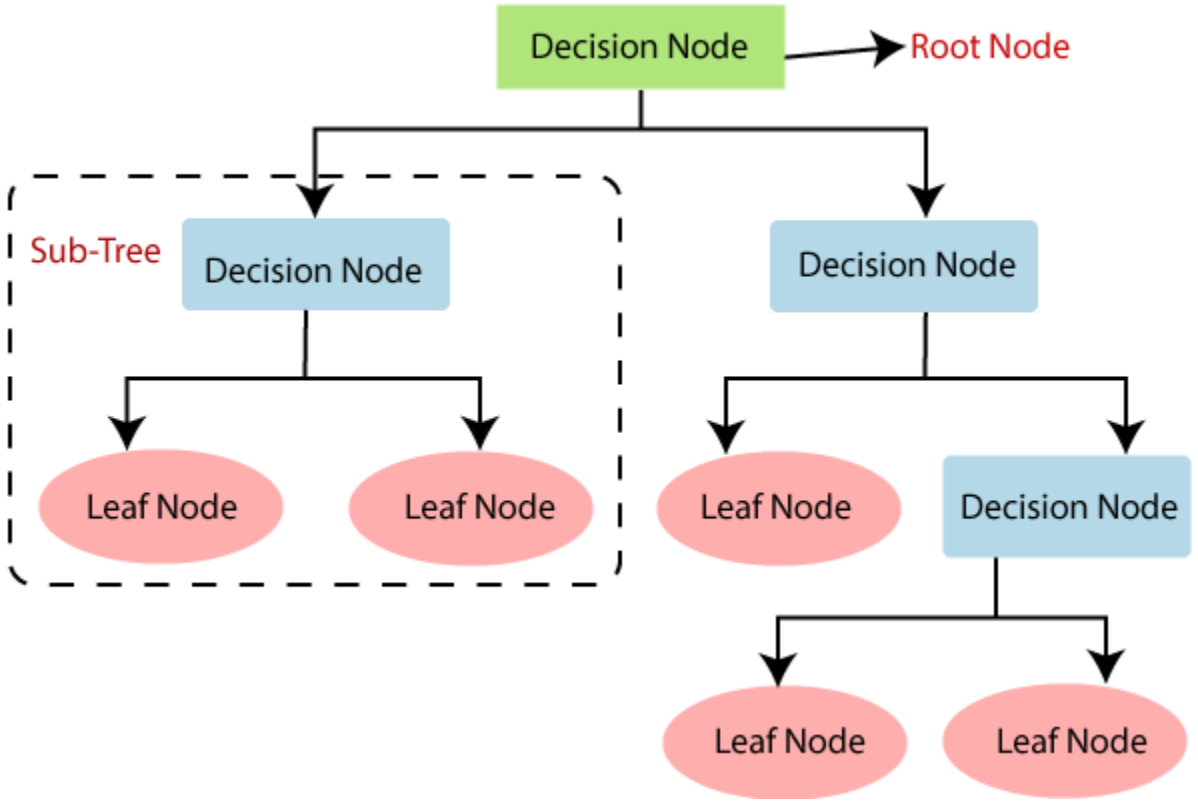
$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Introduction to Decision Tree

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.
- It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

Introduction to Decision Tree

- Below diagram explains the general structure of a decision tree:



Decision Tree Terminologies

- **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.

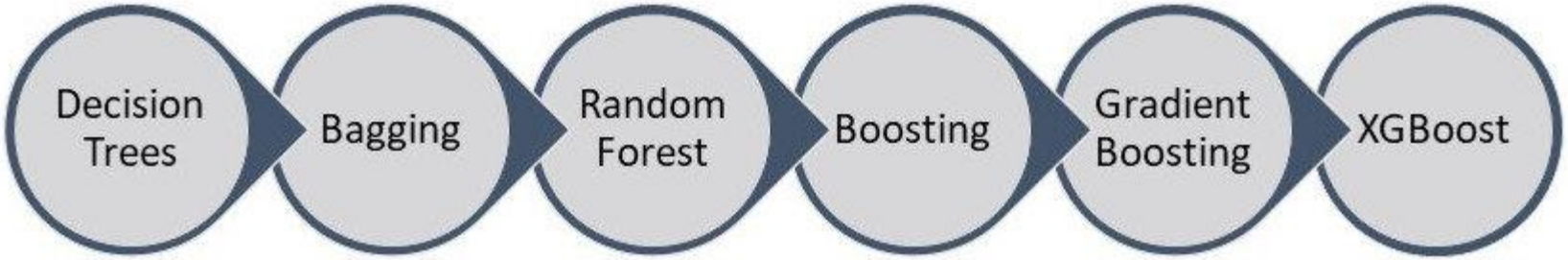
Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

The evolution of Decision tree-based methods

Decision tree-based methods are a class of machine learning algorithms that use a hierarchical structure of decision nodes and leaf nodes to make predictions or decisions based on input features. Here are some commonly used decision tree-based methods:



Decision tree-based methods

1.Decision Tree: The basic decision tree algorithm constructs a tree-like structure where each internal node represents a feature, and the edges emanating from the node correspond to different feature values. At the leaf nodes, predictions or decisions are made based on the majority class or a regression value. Decision trees are interpretable and can handle both categorical and numerical features.

2.Random Forest: Random Forest is an ensemble method that combines multiple decision trees. It creates an ensemble of decision trees by training each tree on a random subset of the training data and using a random subset of features at each split. Random Forest improves the generalization and robustness of the model by averaging the predictions of individual trees.

Decision tree-based methods

3.Gradient Boosting Trees: Gradient Boosting Trees, such as Gradient Boosting Machines (GBM) or XGBoost, build an ensemble of decision trees in a sequential manner. Each tree is trained to correct the mistakes made by the previous trees. The algorithm iteratively fits new trees to the negative gradient of the loss function, gradually improving the overall model performance.

4.CART (Classification and Regression Trees): CART is a decision tree algorithm that can be used for both classification and regression tasks. It constructs binary trees, where each internal node represents a splitting criterion on a feature, and the leaf nodes provide the final predictions or regression values. CART uses techniques such as Gini impurity or mean squared error to determine the optimal splits during the tree construction.

How does the Decision Tree algorithm Work?

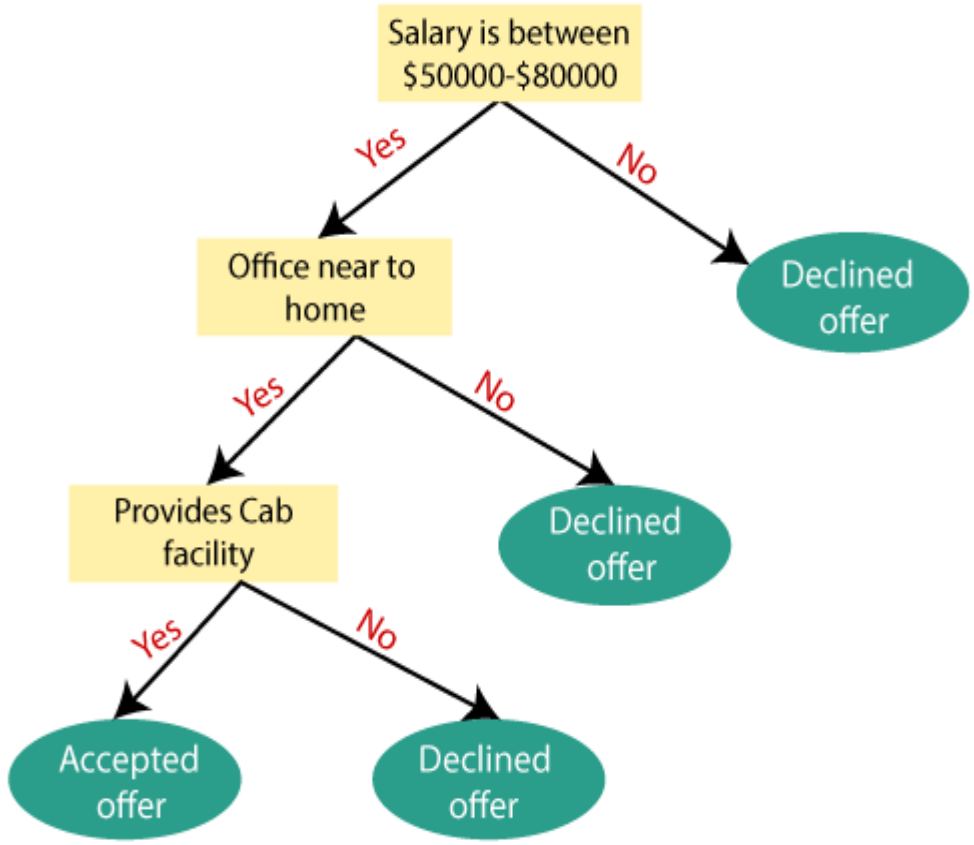
- In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree.
- This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.
- For 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.

How does the Decision Tree algorithm Work?

- 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.

Example:

- Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer).
- Consider the below diagram:



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, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- Information Gain
- Gini Index

1. 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, we split the node and build the decision tree.
- 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)**
- Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no) Where,

S= Total number of samples

P(yes)= probability of yes

P(no)= probability of no

2. 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:

Gini Index= 1- ∑jPj2

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.

Advantages and Disadvantages of the Decision Tree

Advantages

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages

The decision tree contains lots of layers, which makes it complex.

It may have an overfitting issue, which can be resolved using the Random Forest algorithm.

For more class labels, the computational complexity of the decision tree may increase.

“Thank You”

Dimensionality Reduction Techniques

Dimensionality of input

- Number of Observables (e.g. age and income)
- If number of observables is increased
  - More time to compute
  - More memory to store inputs and intermediate results
  - More complicated explanations (knowledge from learning)
    - Regression from 100 vs. 2 parameters
  - No simple visualization
    - 2D vs. 10D graph
  - **Need much more data (curse of dimensionality)**
    - 1M of 1-d inputs is not equal to 1 input of dimension 1M

Dimensionality reduction

- Some features (dimensions) bear little or nor useful information (e.g. color of hair for a car selection)
  - Can drop some features
  - Have to estimate which features can be dropped from data

- Several features can be combined together without loss or even with gain of information (e.g. income of all family members for loan application)
  - Some features can be combined together
  - Have to estimate which features to combine from data

Feature Selection vs Extraction

- Feature selection: Choosing $k<d$ important features, ignoring the remaining $d - k$
  - Subset selection algorithms
- Feature extraction: Project the original x_i , i =1,...,d dimensions to new $k<d$ dimensions, z_j , j =1,...,k
  - Principal Components Analysis (PCA)
  - Linear Discriminant Analysis (LDA)
  - Factor Analysis (FA)

Usage

- Have data of dimension d
- Reduce dimensionality to k<d
  - Discard unimportant features
  - Combine several features in one
- Use resulting k-dimensional data set for
  - Learning for classification problem (e.g. parameters of probabilities P(x|C)
  - Learning for regression problem (e.g. parameters for model y=g(x|Thetha)

Subset selection

- Have initial set of features of size d
- There are 2^d possible subsets
- Need a criteria to decide which subset is the best
- A way to search over the possible subsets
- Can’t go over all 2^d possibilities
- Need some heuristics

“Goodness” of feature set

- Supervised
  - Train using selected subset
  - Estimate error on validation data set

- Unsupervised
  - Look at input only(e.g. age, income and savings)
  - Select subset of 2 that bear most of the information about the person

Mutual Information

- Have a 3 random variables(features) X,Y,Z and have to select 2 which gives most information
- If X and Y are “correlated” then much of the information about of Y is already in X
- Make sense to select features which are “uncorrelated”
- Mutual Information (Kullback–Leibler Divergence) is more general measure of “mutual information”
- Can be extended to n variables (information variables x_1,.. x_n have about variable x_{n+1})

Subset-selection

- Forward search
  - Start from empty set of features
  - Try each of remaining features
  - Estimate classification/regression error for adding specific feature
  - Select feature that gives maximum improvement in validation error
  - Stop when no significant improvement

- Backward search
  - Start with original set of size d
  - Drop features with smallest impact on error

Floating Search

- Forward and backward search are “greedy” algorithms
  - Select best options at single step
  - Do not always achieve optimum value

- Floating search
  - Two types of steps: Add *k*, remove *l*
  - More computations

Feature Extraction

- Face recognition problem
  - Training data input: pairs of Image + Label(name)
  - Classifier input: Image
  - Classifier output: Label(Name)
- Image: Matrix of 256X256=65536 values in range 0..256
- Each pixels bear little information so can’t select 100 best ones
- Average of pixels around specific positions may give an indication about an eye color.

Projection

- Find a projection matrix w from d-dimensional to k-dimensional vectors that keeps error low

$$z = \boldsymbol{w}^T \boldsymbol{x}$$

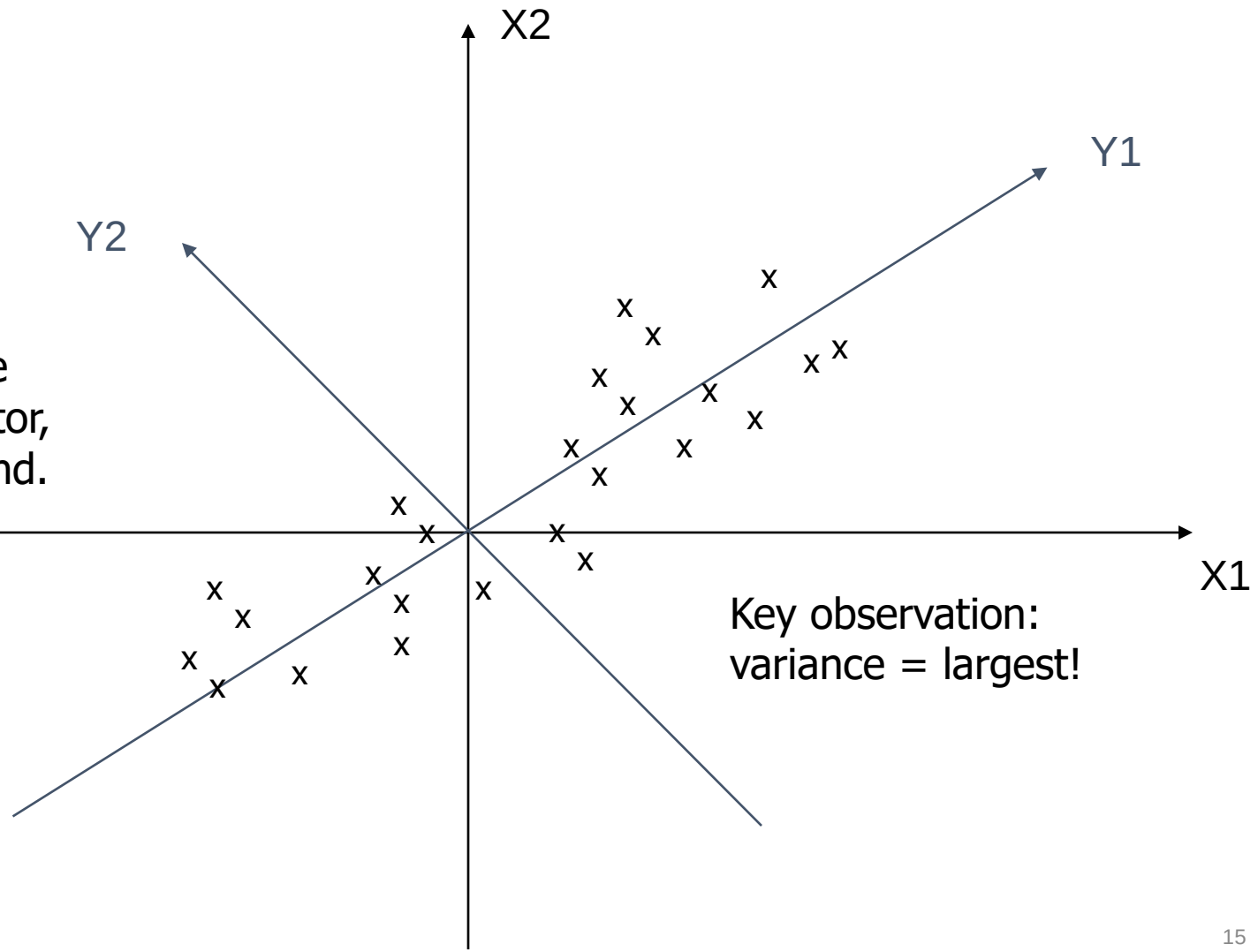
Principal Components Analysis (PCA)

- An exploratory technique used to reduce the dimensionality of the data set to 2D or 3D
- Can be used to:
  - Reduce number of dimensions in data
  - Find patterns in high-dimensional data
  - Visualize data of high dimensionality
- Example applications:
  - Face recognition
  - Image compression
  - Gene expression analysis

Principal Components Analysis Ideas (PCA)

- Does the data set ‘span’ the whole of d dimensional space?
- For a matrix of *m* samples x *n* genes, create a new covariance matrix of size *n* x *n*.
- Transform some large number of variables into a smaller number of uncorrelated variables called principal components (PCs).
- developed to capture as much of the variation in data as possible

Principal Component Analysis



Note: Y1 is the first eigen vector, Y2 is the second. Y2 ignorable.

Key observation: variance = largest!

Principal Component Analysis: one attribute first

- Question: how much spread is in the data along the axis? (distance to the mean)
- Variance=Standard deviation^2

$$s^2 = \frac{\sum_{i=1}^{n} (X_i - \bar{X})^2}{(n-1)}$$

Temperature
42
40
24
30
15
18
15
30
15
30
35
30
40
30

Now consider two dimensions

Covariance: measures the correlation between X and Y

- cov(X,Y)=0: independent
- Cov(X,Y)>0: move same dir
- Cov(X,Y)<0: move oppo dir

$$\mathrm{cov}(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

X=Temperature	Y=Humidity
40	90
40	90
40	90
30	90
15	70
15	70
15	70
30	90
15	70
30	70
30	70
30	90
40	70
30	90

More than two attributes: covariance matrix

- Contains covariance values between all possible dimensions (=attributes):

$$C^{nxn} = (c_{ij} \mid c_{ij} = \text{cov}(Dim_i, Dim_j))$$

- Example for three attributes (x,y,z):

$$C = \begin{pmatrix} \text{cov}(x,x) & \text{cov}(x,y) & \text{cov}(x,z) \\ \text{cov}(y,x) & \text{cov}(y,y) & \text{cov}(y,z) \\ \text{cov}(z,x) & \text{cov}(z,y) & \text{cov}(z,z) \end{pmatrix}$$

Eigenvalues & eigenvectors

- Vectors **x** having same direction as *A***x** are called *eigenvectors* of *A* (*A* is an n by n matrix).
- In the equation *A***x**=λ**x**, λ is called an *eigenvalue* of *A*.

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} x \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4x \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

Eigenvalues & eigenvectors

- $A\mathbf{x}=\lambda\mathbf{x} \Leftrightarrow (A-\lambda I)\mathbf{x}=0$
- How to calculate $\mathbf{x}$ and λ:
  - Calculate $det(A-\lambda I)$, yields a polynomial (degree n)
  - Determine roots to $det(A-\lambda I)=0$, roots are eigenvalues λ
  - Solve $(A-\lambda I)\,\mathbf{x}=0$ for each λ to obtain eigenvectors $\mathbf{x}$

Principal components

- 1. principal component (PC1)
  - The eigenvalue with the largest absolute value will indicate that the data have the largest variance along its eigenvector, the direction along which there is greatest variation
- 2. principal component (PC2)
  - the direction with maximum variation left in data, orthogonal to the 1. PC
- In general, only few directions manage to capture most of the variability in the data.

Steps of PCA

- Let $\bar{X}$ be the mean vector (taking the mean of all rows)
- Adjust the original data by the mean
  $X' = X - \bar{X}$
- Compute the covariance matrix C of adjusted X
- Find the eigenvectors and eigenvalues of C.

- For matrix *C*, *v*ectors **e** (=column vector) having same direction as *C***e** :
  - *eigenvectors* of *C* is **e** such that $C\mathbf{e}=\lambda\mathbf{e}$,
  - λ is called an *eigenvalue* of *C*.
- $C\mathbf{e}=\lambda\mathbf{e} \Leftrightarrow (C-\lambda I)\mathbf{e}=0$
  - **Most data mining packages do this for you.**

Eigenvalues

- Calculate eigenvalues λ and eigenvectors **x** for covariance matrix:
  - Eigenvalues λ_j are used for calculation of [% of total variance] (V_j) for each component j:

$$V_j = 100 \cdot \frac{\lambda_j}{\sum_{x=1}^{n} \lambda_x} \qquad \sum_{x=1}^{n} \lambda_x = n$$

Transformed Data

- Eigenvalues λ_j corresponds to variance on each component *j*
- *Thus, sort by* λ_j
- Take the first *p* eigenvectors $\mathbf{e_i}$, where p is the number of top eigenvalues
- These are the directions with the largest variances

$$\begin{pmatrix} y_{i1} \\ y_{i2} \\ \dots \\ y_{ip} \end{pmatrix} = \begin{pmatrix} e_1 \\ e_2 \\ \dots \\ e_p \end{pmatrix} \begin{pmatrix} x_{i1} - \overline{x_1} \\ x_{i2} - \overline{x_2} \\ \dots \\ x_{in} - \overline{x_n} \end{pmatrix}$$

Factor Analysis

- Assume set of unobservable (“latent”) variables
- Goal: Characterize dependency among observables using latent variables
- Suppose group of variables having large correlation among themselves and small correlation with other variables
- Single factor?

Factor Analysis

- Assume k input factors (latent unobservable) variables generating d observables
- Assume all variations in observable variables are due to latent or noise (with unknown variance)
- Find transformation from unobservable to observables which explain the data

Factor Analysis

- Find a small number of factors ***z***, which when combined generate ***x***

$$x_i - \mu_i = v_{i1}z_1 + v_{i2}z_2 + \ldots + v_{ik}z_k + \varepsilon_i$$

where z_j, $j = 1,\ldots,k$ are the latent factors with

$$E[\, z_j \,]=0,\ \mathrm{Var}(z_j)=1,\ \mathrm{Cov}(z_i, z_j)=0,\ i \neq j,$$

ε_i are the noise sources

$$E[\, \varepsilon_i \,]= \psi_i,\ \mathrm{Cov}(\varepsilon_i, \varepsilon_j) =0,\ i \neq j,\ \mathrm{Cov}(\varepsilon_i, z_j) =0,$$

and v_{ij} are the factor loadings

$$\boldsymbol{x} - \boldsymbol{\mu} = \mathbf{V}\boldsymbol{z} + \boldsymbol{\epsilon}$$

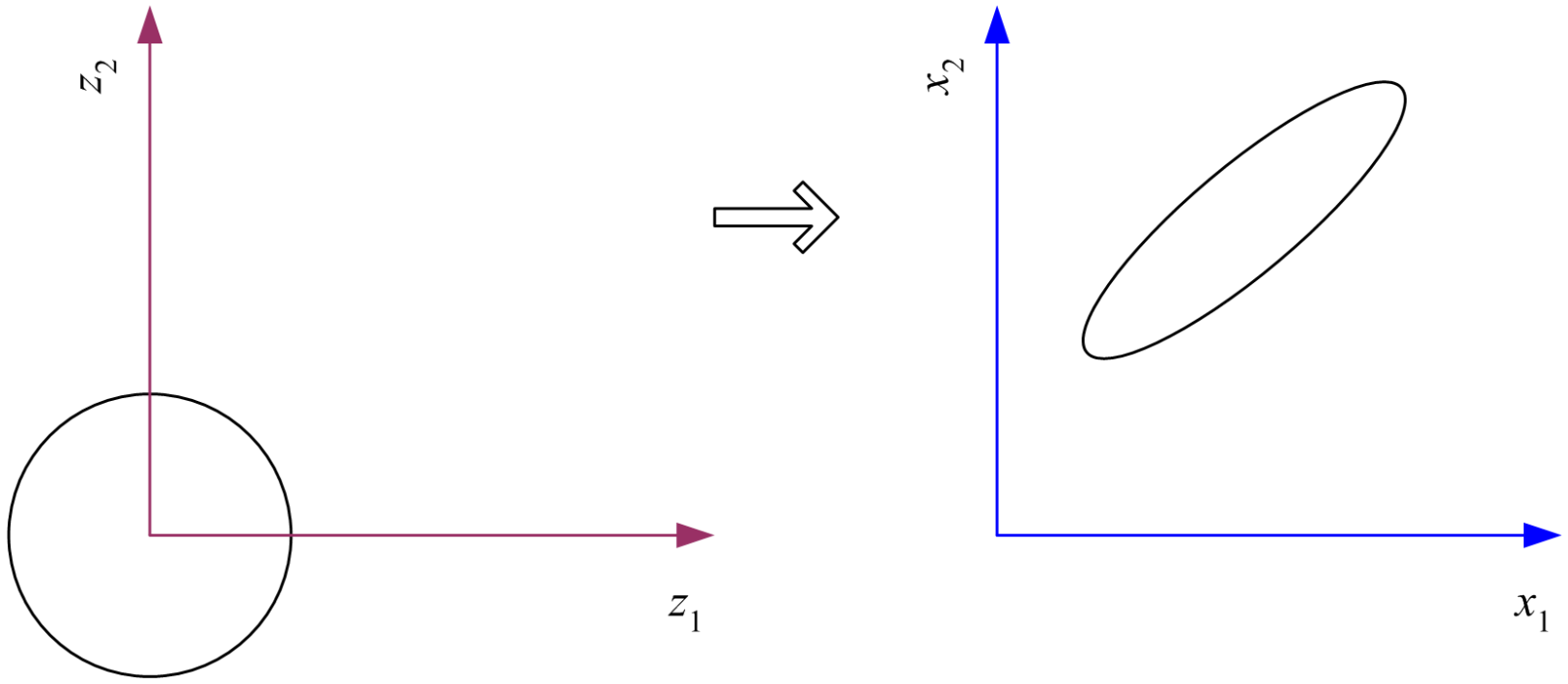
Factor Analysis

- Find V such that $\mathbf{S} = \mathbf{V}\mathbf{V}^T + \mathbf{\Psi}$ where S is estimation of covariance matrix and V loading (explanation by latent variables)
- V is d x k matrix (k<d)
- Solution using eigenvalue and eigenvectors

$$\mathbf{Z} = \mathbf{X}\mathbf{W} = \mathbf{X}\mathbf{S}^{-1}\mathbf{V}$$

Factor Analysis

- In FA, factors z_j are stretched, rotated and translated to generate $\boldsymbol{x}$

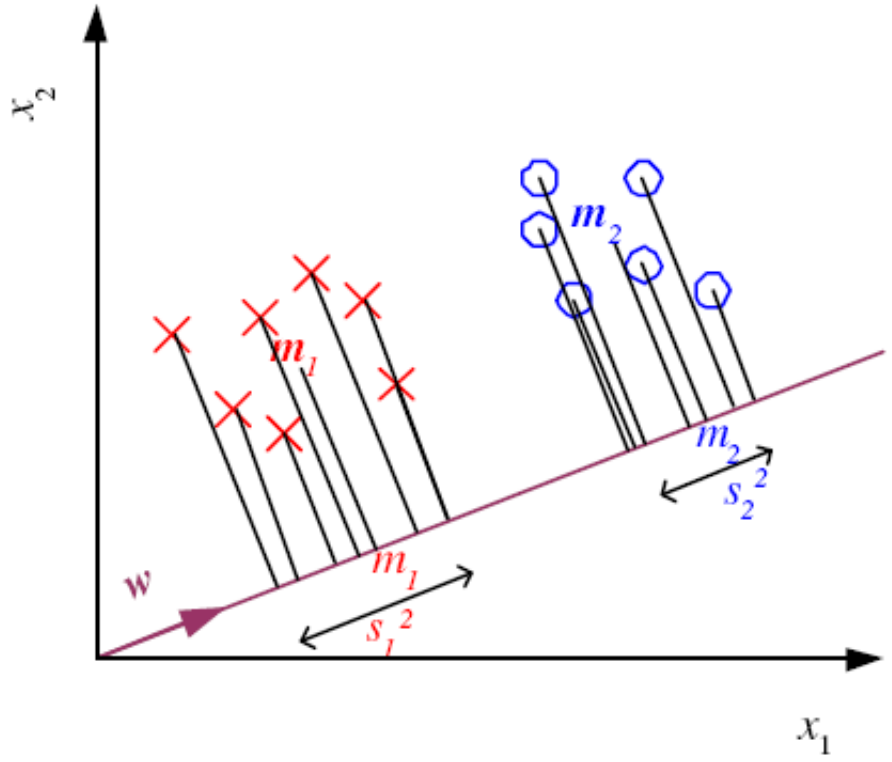


FA Usage

- Speech is a function of position of small number of articulators (lungs, lips, tongue)
- Factor analysis: go from signal space (4000 points for 500ms) to articulation space (20 points)
- Classify speech (assign text label) by 20 points
- Speech Compression: send 20 values

Linear Discriminant Analysis

- Find a low-dimensional space such that when $\boldsymbol{x}$ is projected, classes are well-separated



Means and Scatter after projection

$$m_1 = \frac{\sum_t \mathbf{w}^T \mathbf{x}^t r^t}{\sum_t r^t} = \mathbf{w}^T \mathbf{m}_1$$

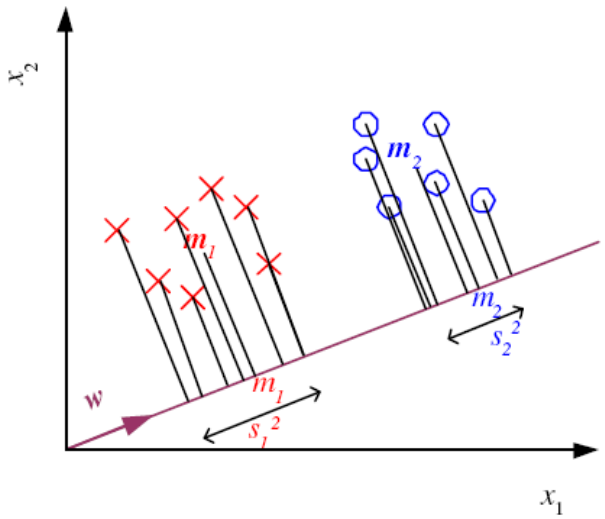
$$m_2 = \frac{\sum_t \mathbf{w}^T \mathbf{x}^t (1 - r^t)}{\sum_t (1 - r^t)} = \mathbf{w}^T \mathbf{m}_2$$

$$s_1^2 = \sum_t (\mathbf{w}^T \mathbf{x}^t - m_1)^2 r^t$$

$$s_2^2 = \sum_t (\mathbf{w}^T \mathbf{x}^t - m_2)^2 (1 - r^t)$$

Good Projection

- Means are far away as possible
- Scatter is small as possible
- Fisher Linear Discriminant



$$J(\mathbf{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$

Summary

- Feature selection
  - Supervised: drop features which don’t introduce large errors (validation set)
  - Unsupervised: keep only uncorrelated features (drop features that don’t add much information)
- Feature extraction
  - Linearly combine feature into smaller set of features
  - Supervised
    - PCA: explain most of the total variability
    - FA: explain most of the common variability
  - Unsupervised
    - LDA: best separate class instances

“Thank You”

Support Vector Machines

What is Support Vector Machines?

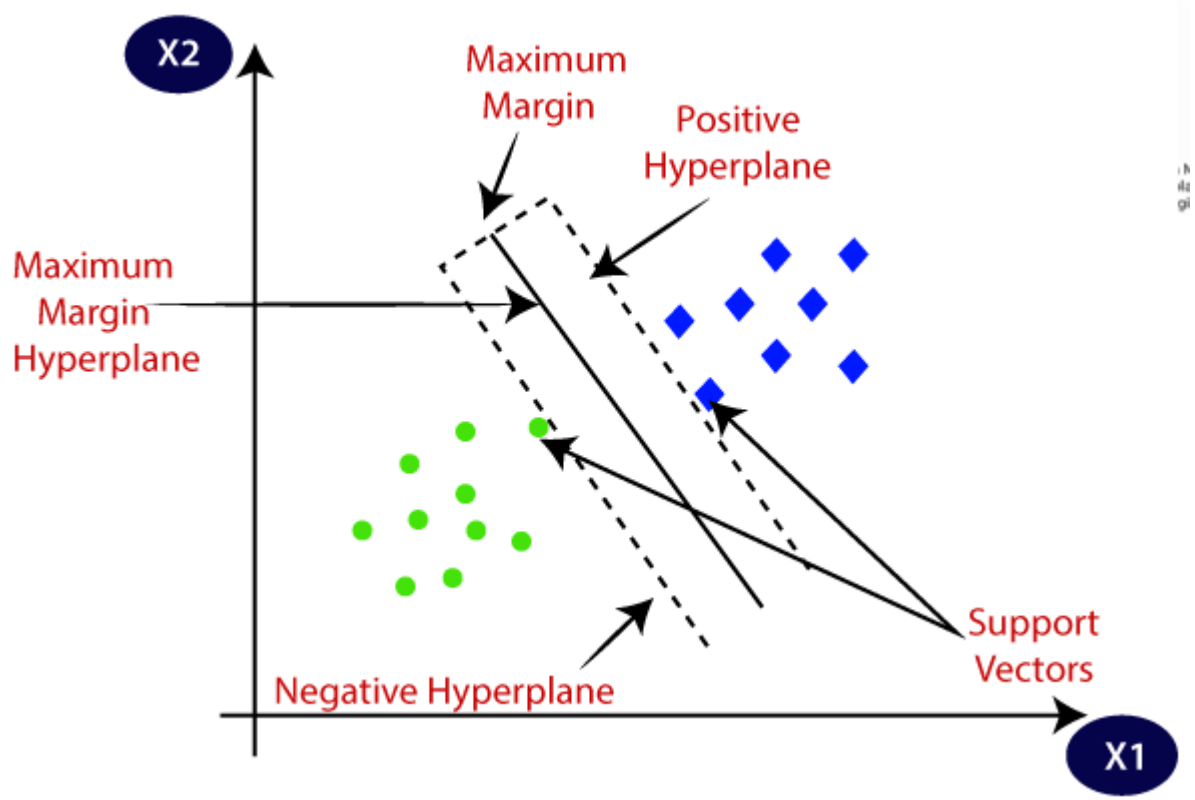
- Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems.
- The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.
- SVM chooses the extreme points/vectors that help in creating the hyperplane.
- These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine

Why SVMs are used in machine learning

- SVMs are used in applications like handwriting recognition, intrusion detection, face detection, email classification, gene classification, and in web pages. This is one of the reasons we use SVMs in machine learning. It can handle both classification and regression on linear and non-linear data.
- Another reason we use SVMs is because they can find complex relationships between your data without you needing to do a lot of transformations on your own. It's a great option when you are working with smaller datasets that have tens to hundreds of thousands of features. They typically find more accurate results when compared to other algorithms because of their ability to handle small, complex datasets.

SVM

Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Types of SVM

- SVM can be of two types:

- Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hyperplane and Support Vectors in the SVM algorithm

Hyperplane:

- There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

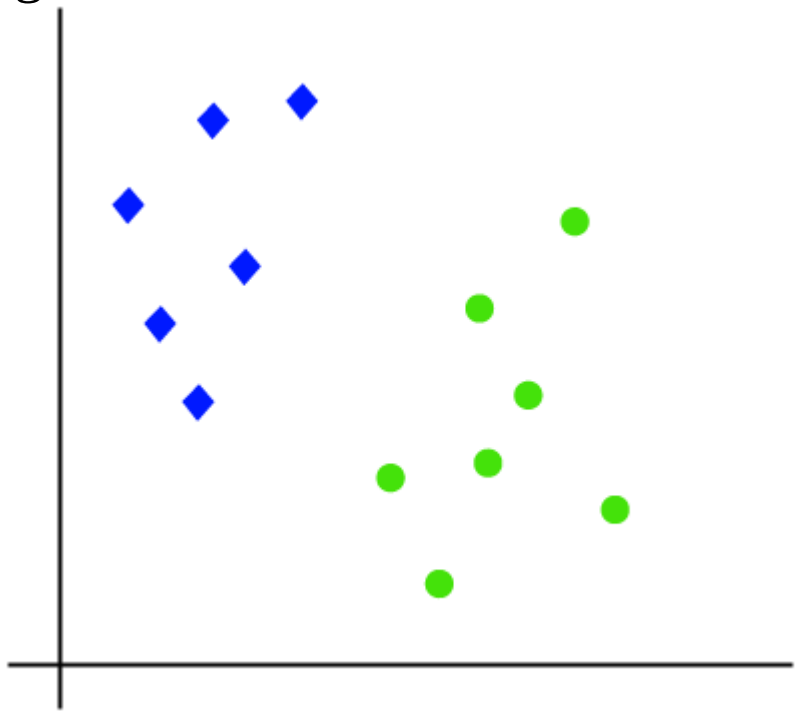
Support Vectors:

- The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

How does SVM works?

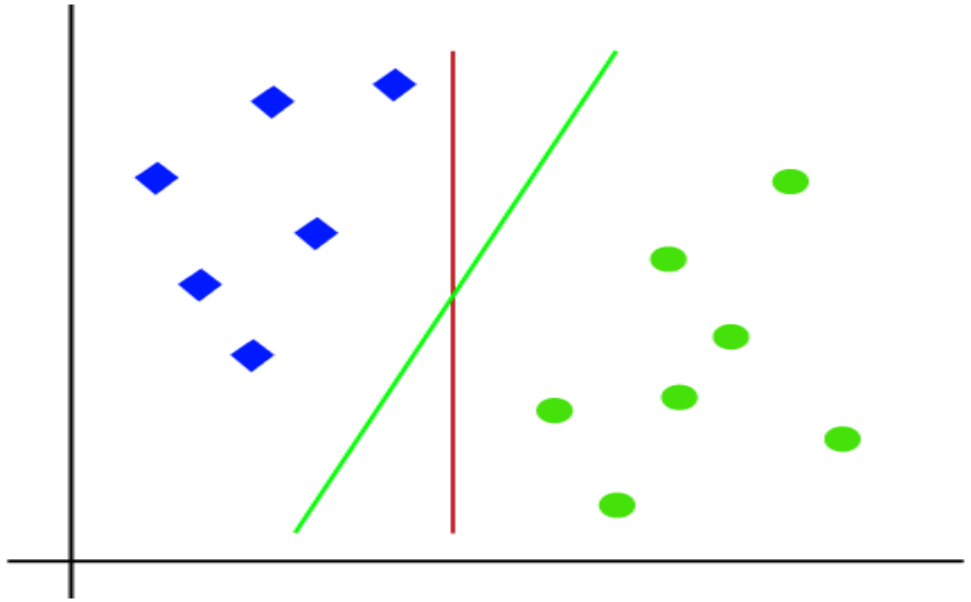
Linear SVM

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



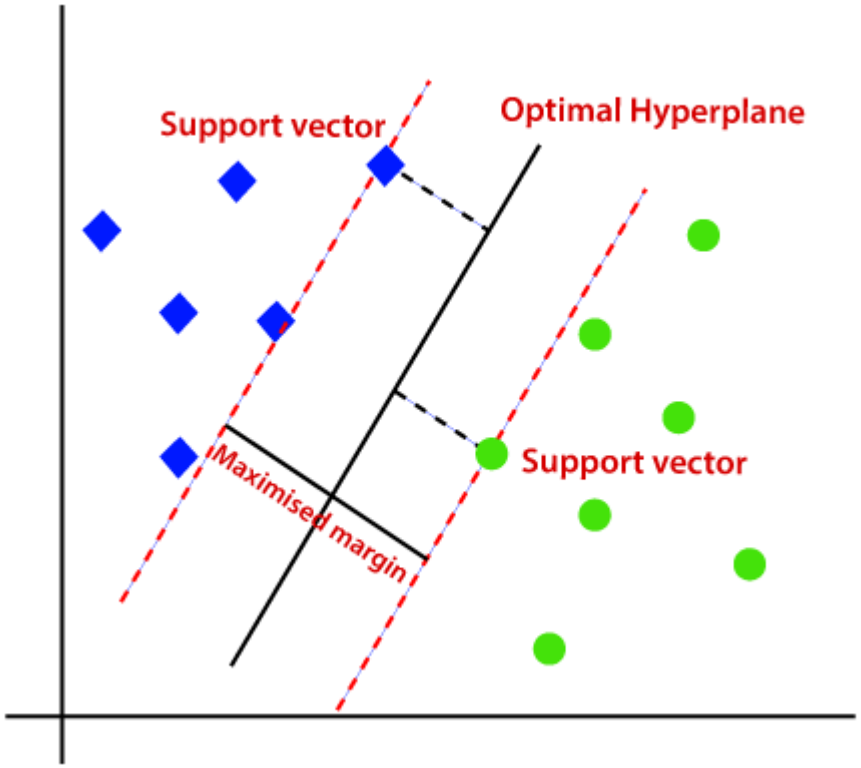
Linear SVM (Continue..)

- So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:



Linear SVM (Continue..)

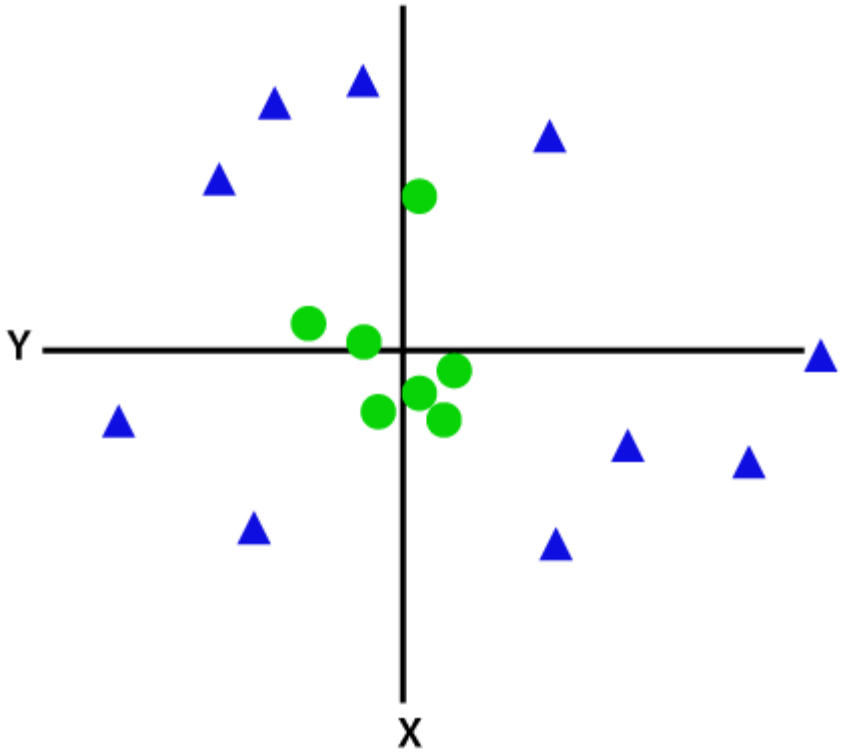
- Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.



Non-Linear SVM

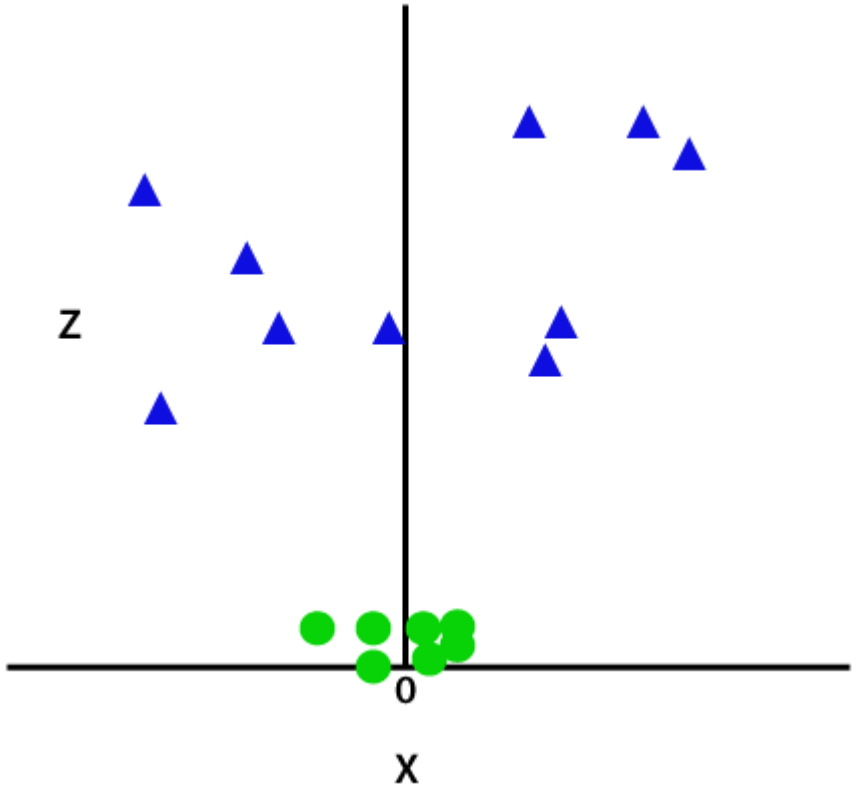
- If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:
- So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$$z=x^2 +y^2$$



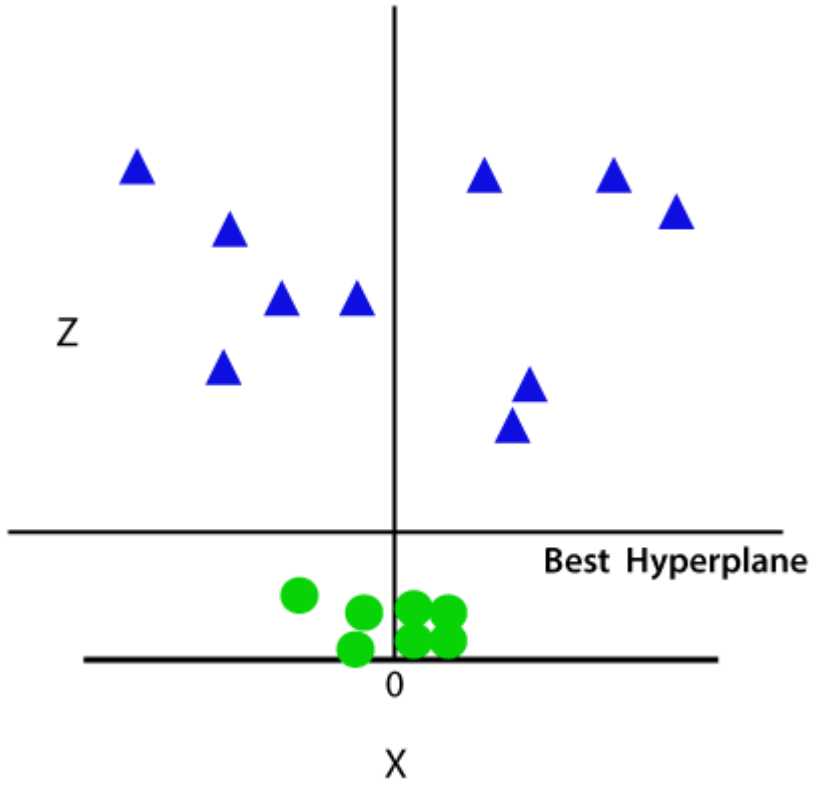
Non-Linear SVM(Continue..)

- By adding the third dimension, the sample space will become as below image:



Non-Linear SVM(Continue..)

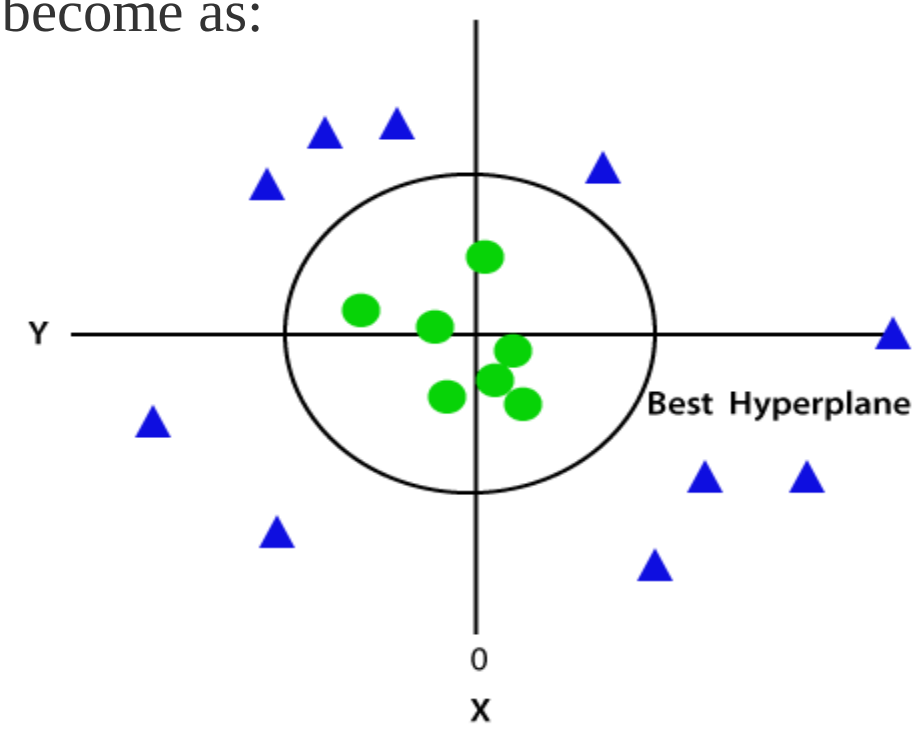
- So now, SVM will divide the datasets into classes in the following way. Consider the below image:



Non-Linear SVM(Continue..)

- Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:

Hence we get a circumference of radius 1 in case of non-linear data.



Advantages of SVM

- Effective on datasets with multiple features, like financial or medical data.
- Effective in cases where number of features is greater than the number of data points.
- Uses a subset of training points in the decision function called support vectors which makes it memory efficient.
- Different kernel functions can be specified for the decision function. You can use common kernels, but it's also possible to specify custom kernels.

Disadvantages of SVM

- If the number of features is a lot bigger than the number of data points, avoiding over-fitting when choosing kernel functions and regularization term is crucial.
- SVMs don't directly provide probability estimates. Those are calculated using an expensive five-fold cross-validation.
- Works best on small sample sets because of its high training time.

Kernel functions

- **Linear Kernel**
- These are commonly recommended for text classification because most of these types of classification problems are linearly separable.

- The linear kernel works really well when there are a lot of features, and text classification problems have a lot of features. Linear kernel functions are faster than most of the others and you have fewer parameters to optimize.

- Here's the function that defines the linear kernel:

- $f(X) = w^T * X + b$
- In this equation, w is the weight vector that you want to minimize, X is the data that you're trying to classify, and b is the linear coefficient estimated from the training data. This equation defines the decision boundary that the SVM returns.

Polynomial Kernel

- The polynomial kernel isn't used in practice very often because it isn't as computationally efficient as other kernels and its predictions aren't as accurate.

- Here's the function for a polynomial kernel:

- $f(X1, X2) = (a + X1^T * X2) ^ b$
- This is one of the more simple polynomial kernel equations you can use. f(X1, X2) represents the polynomial decision boundary that will separate your data. X1 and X2 represent your data.

Applications of SVM in Real World

“Thank You”