The learning algorithm for choosing the weights, w_i, to best fit the set of training examples {(s, E_{train} (s))}. We mean to say the best fit to the training data. One general approach is to define the best hypothesis or a set of weights, which in turn minimizes the squared error, E, between the training values and the values predicted by the hypothesis E'. That is,

$$E' = \Sigma (s, E_{train\ (S)\ \varepsilon\ training\ examples} (E_{train}(s) - E'(s))^2$$

Thus, our objective is to seek the weights or equivalently E' that minimize E for the observed training examples.

1.4 Training data

We cannot wait for fresh data every time as it is very costly and time consuming task. We need to either **randomly split** that data or **tune our learning algorithm.** In first case of a **random split** of data we split the data as follows:

- From 25% to 30% for testing.
- And left over 70% to 75% for training.

Please note that this split is done by keeping in mind the correspondence between each response and its features.

The 2nd type of split is not a good one as it causes a type of overfitting called as **snooping.**

To overcome this problem, we go for the 3rd method of splitting the input data also called as **validation set.** This is done as follows:

- 70% for training.
- 20% for validation.
- 10% for testing.

Deep learning and **Big Data** are inter-related. Big data refers to the pools of data being generated every second while deep learning is an active research area in the field of **machine learning.** But as the data is growing at an exponential rate, deep learning is providing big data predictive analytical solutions. According to the report of National Security Agency (NSA), the Internet itself is processing 1,826 Petabytes of data per day [1]. By 2020, the figure will reach 35 Trillion gigabytes. This leads to a remarkable paradigm shift in computer science literature towards data-driven discovery. Today machine learning techniques and newer software, hardware and computational power help in big data as well as web data analytics.

We need to define **Deep Learning** now.

Deep learning is defined as the machine learning techniques that use supervised and/or unsupervised methods to automatically learn hierarchical representations in deep architectures for classification.

Or

Deep learning is a subset of artificial intelligence field called as machine learning which is predicted on the basic idea of learning from example.

In machine learning, instead of teaching a computer a massive list of rules to solve the complex problem, we give it a **model** with which it can evaluate examples. Also we give it a small set of instructions to modify the model when it makes a mistake. **Please note that we have an expectation that as the time passes, a well suited model would be able to solve the problem more accurately.**

For example, say our model is defined by a function **f(x, θ).** The input x is an example expressed in vector form. Like, if x were a gray-scale image, the vector's components would be pixel intensities at each position. The input θ is a vector of the parameters that our model uses. **Also note now that it is our machine learning program that tries to perfect the values of these parameters as it is exposed to more and more examples.**

The conventional learning methods are shallow-structured learning architectures while deep learning is deep-hierarchical levelled learning architecture. It is based on the biological neurons within human brain. Several research domains like speech recognition, Natural Language Processing (NLP), collaborative filtering and computer vision. Big industry giants like Google, Apple and Facebook who work on sheer size of data lakes today have also given vital importance to Big Data Analytics and Web Data Analytics. Google applies deep learning algorithms to massive chunks of messy data obtained from Internet. IBMs brain-like computer uses techniques like deep learning to make available big data available for its betterment. **Please note that as the data keeps increasing at an exponential rate, deep learning is playing a key role in providing** in big data and web analytics. **Also note that deep learning refers to a set of machine learning techniques that learn multiple levels of representations in deep architectures.** Deep learning is the new name for neural networks. Deep learning algorithms have best performance as far as complex problems like pattern recognition are concerned. Practically speaking, even machine learning approaches find it difficult to solve. Theoretically, they can solve any problem. Several libraries also exist for neural nets or deep learning. But deep learning algorithms as an auto-encoder *i.e.* unsupervised deep learning, several parameters and slower deep learning grids are some of the issues to be discussed in this text. These programs may need more hardware.

The information that is available on the web and is searched using search engines like Google is called as the **surface web.** On the other hand, **deep web** consists of data hidden behind web-based services. Theoretically, the deluge (flood) of data within deep web is exponentially high. Practically, useful information hidden in deep web is finite but very large. Each web form can give rise to thousands of results, each of which qualifies as a **deep web page.** Similarly, every Facebook or Twitter post might be considered as a **deep web page.** Finally, if one considers all possible pages of search results then the size of deep web is infinite. As per the survey report (2007), roughly 2.5% of random sample of web pages were forms that should be considered as part of deep web. **Please understand that deep web is huge, far larger than the indexed web of 50 billion pages.** Searching the deep web is current research area today.
Let us explore this and many more interesting topics in this book.
We define a neuron first of all. **A neuron is a function that takes multiple numeric inputs and gives out one numeric output. These neurons are organized in layers and the outputs from all of the neurons in one layer become the inputs for each neuron in the next layer.**

The functional unit of a normal human brain is the **neuron.** The **objective** book is to use this natural brain concept to build machine learning models that solve problems in a similar fashion. A neuron is optimized to receive information from other neurons, process this information and send its results to other cells. This is shown in figure-1.6 as follows:

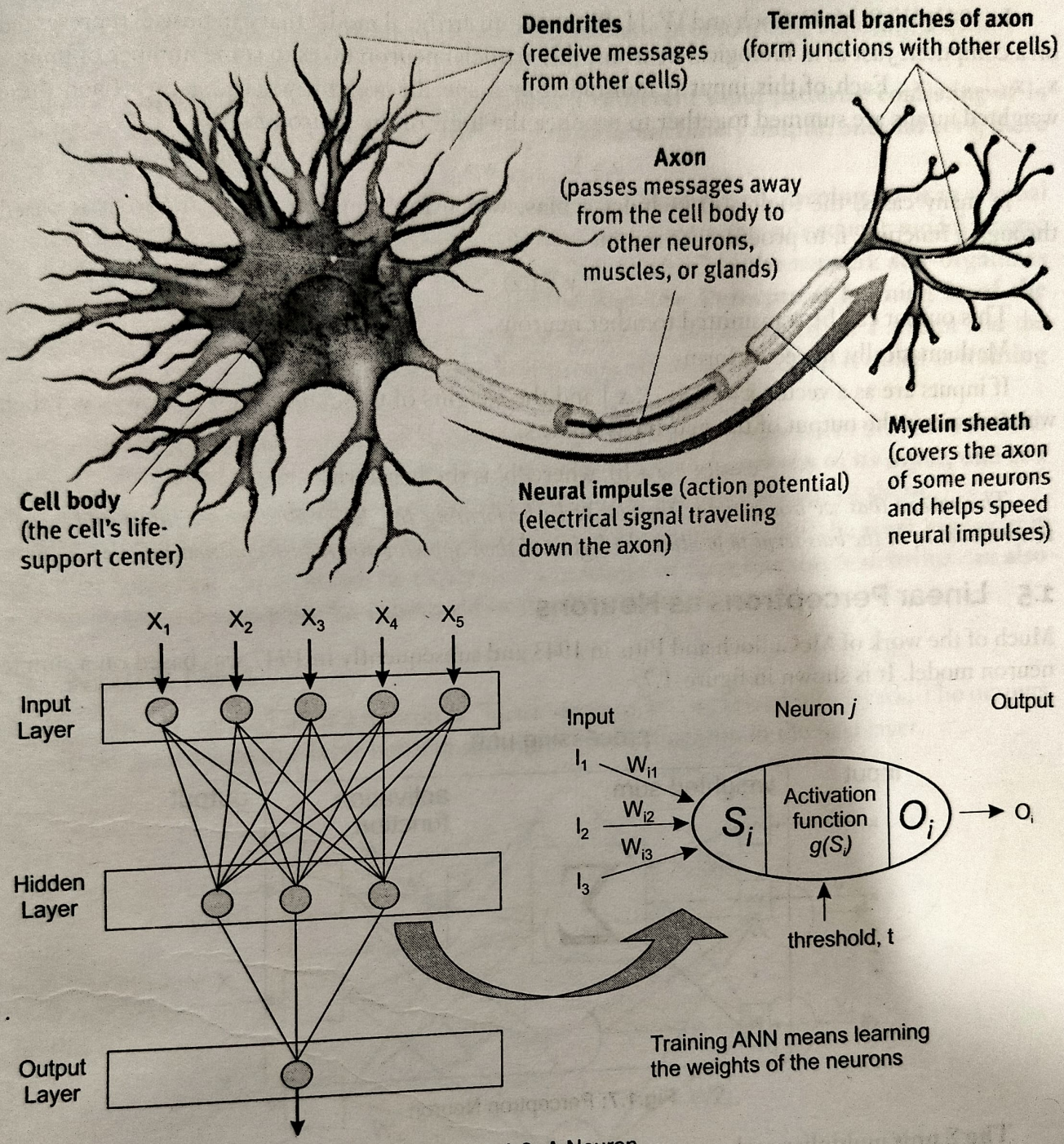


Fig.1.6: A Neuron

The neuron receives its inputs along antennae-like structures called **dendrites.** Each of these incoming connections is dynamically strengthened or weakened based on how often it is used. **Please understand that it is the strength of each connection that determines the contribution of the input to the neuron's output.** After being weighted by the strength of their respective connections, the inputs are summed together in the **cell body. Also understand that this sum is then transformed into a new signal that is propagated along the cell's axon and sent to other neurons.**

In 1943, W. S. McCulloch and W. H. Pitts gave an artificial model that can be well represented on a computer. Just as in biological neurons, this artificial neuron takes in some number of inputs, $x_1, x_2, \ldots\ldots, x_n$. Each of this input is multiplied by a specific weight, $w_1, w_2, \ldots\ldots, w_n$. Then these weighted inputs are summed together to produce the **logit** of the neuron *i.e.*

$$z = \Sigma_{(i=0 \text{ to } n)} w_i x_i$$

in many cases, the logit also includes a **bias,** which is a constant. Then the logit is paaed through a function, f, to produce the output-

$$y = f(z)$$

This output can be transmitted to other neurons.

Mathematically, in vector form-

If inputs are as a vector $x = [x_1 x_2 \ldots\ldots x_n]$ and the weights of the neuron as $w = [w_1 w_2 \ldots\ldots w_n]$ then we can express the output of the neuron as follows:

$$y = f(x \,.\, w + b), \text{ where 'b' is the bias term.}$$

This means that we can compute the output by performing the dot product of the input and weight vectors, adding in the bias term to produce the logit and then applying the transformation function.

1.5 Linear Perceptrons as Neurons

Much of the work of McCulloch and Pitts in 1943 and subsequently in 1947 was based on a simple neuron model. It is shown in figure-1.7.

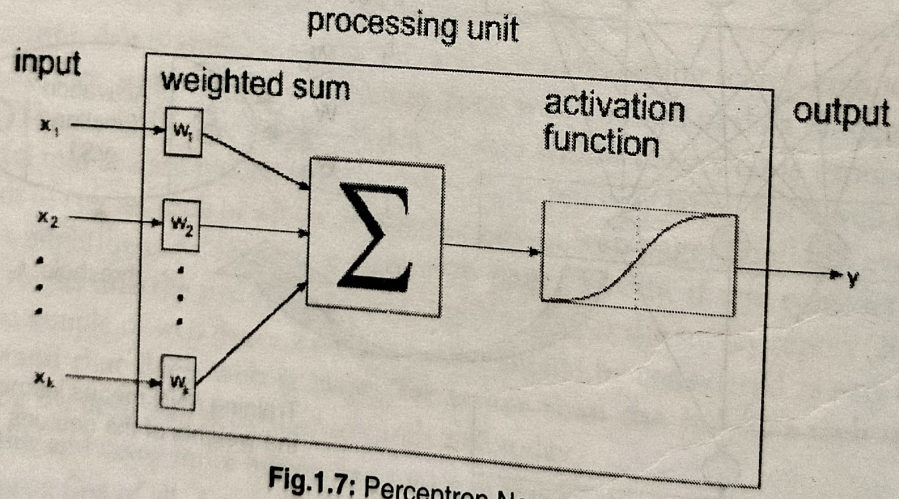


Fig.1.7: Perceptron Neuron

The Σ unit multiplies each input, x, by a weight ,w, and sums the weighted inputs. **If this sum is greater than a predetermined threshold, the output =1 else it is zero.** These systems and their variants collectively have been called as **perceptrons.** In general, they consist of a single layer of artificial neurons connected by weights to a set of inputs although more complicated networks bear the same.

The **PERCEPTRN LEARNING THEOREM states that-"A perceptron could learn anything it could represent." Rosenblatt, in 1962, proved that a perceptron can be trained to any function it can represent.** The difference between **representation** and **learning** lies in the point of reference. **Representation means** the ability of a perceptron to simulate a specified

function while **learning** requires the existence of a systematic procedure for adjusting the network weights to produce that function.

Note: A neuron with n binary inputs can have 2^n different input patterns, consisting of 1s and 0s. Because each input pattern can produce two different binary output, one and zero, there are 2^{2n} different functions of n variables.

Learning is of two types- Supervised and Unsupervised. **Supervised learning** needs an external "teacher" that evaluates the behaviour of the system and directs the subsequent modifications. On the other hand, **unsupervised learning** needs "no teacher" as the network **self-organizes** to produce the desired changes. **Please note here that the Perceptron learning is of the supervised type.** The perceptron training algorithm can be implemented on a system and the network becomes **self adjusting.** This is reason as to why adjusting of weights is called as **training** and the network is said to **learn.**

Rosenblatt's proof was a major milestone and it provided a great impetus to the research field. **Also note that a perceptron is trained by representing a set of patterns to its input, one at a time and adjusting the weights until the desired output occurs for each of them.**

Linear perceptron and the neuron model are somewhat equivalent. But the point to ponder is that **every linear perceptron can be expressed as a single neuron but single neurons can also express models that cannot be expressed by any linear perceptron.**

1.6 Neural Nets

We have already defined what a neuron is. These neurons are organized into layers. The outputs from all the neurons in one layer become the inputs for each neuron in the next layer.

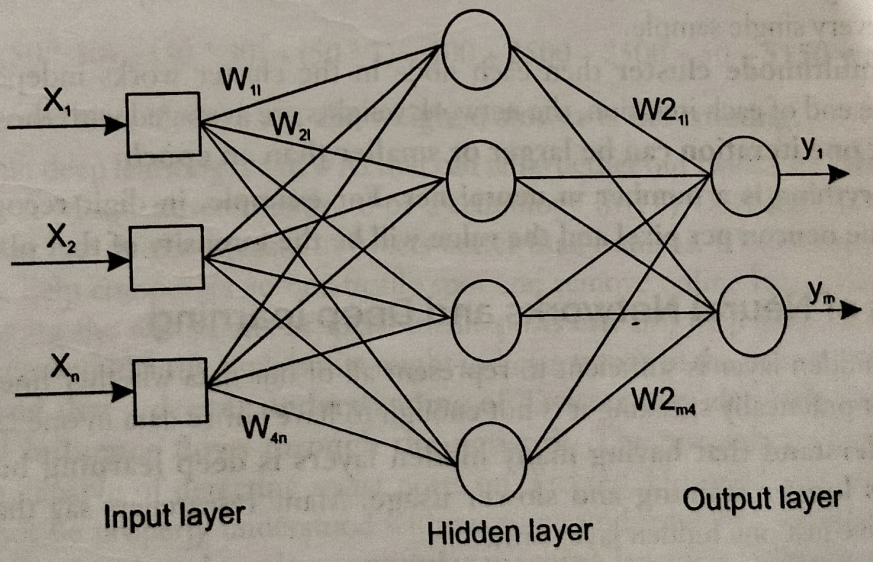


Fig.1.8: Neural Nets

Architecture/ Design

As shown in figure-1.8, the first layer is your data (inputs) that may be a training sample or a test sample or real inputs. The very last layer is your output (answer). Two cases arise:

Case-1: If you are doing a **regression** *i.e.* learning a single value then the output layer will have **one neuron.**

Case-2: If you are doing **classification,** then the output layer will have one neuron for each possible answer. Here, the output (answer) with the highest probability for the set of inputs is chosen.

The layers between the input layers and the output layers are called as the hidden layers.

Rule: "Each neuron in each hidden layer has a weight for each of its inputs and modifying those weights is how the network learns. There is a bias input to each neuron, (a weight only) connected to a constant input and is also tuned during the training process."

1.7 Working of Neural Nets

As shown in figure 1.6, we start with random weights. Then we give it the first training sample and the correct answer (also called as **supervised learning),** calculate the error and then use that to go back and tweak each of the weights so that there is a bit less error. **This process is called as backpropagation.** Then we take the second training sample and repeat the process. **This process of taking each and every training data is called as an epoch. Also note that we specify the number of epochs in the algorithm that we develop. Also note that even fractional epochs are allowed. For example,** requesting 20.5 epochs means all training data samples are processed 20 times while the first half of the training data is also processed an 11^{th} time.

Many deep learning libraries use the concept of **mini batches.** A mini-batch of say, size 20 means that it will process 20 training samples then go back and update the weights in one batch. But deep learning libraries like H2O does not use mini-batch concept in H2O the weights get updated after every single sample.

If it is a **multimode cluster** then each node in the cluster works independently for one iteration. At the end of each iteration, the network weights are averaged with those on every node. **Also note that one iteration can be larger or smaller than an epoch.**

Note: Everything is a number in neural net. For example, in digit recognition problem there will be one neuron per pixel and the value will be the intensity of that pixel.

1.8 Layers of Neural Networks and Deep learning

In theory, one hidden layer is sufficient to represent all of our data whether linear or non-linear relationships but practically speaking it is not enough to have entire data in one layer.

Please understand that having many hidden layers is deep learning but more layers/ neurons means longer training and slower usage. Many researchers say that it is not deep learning if we have just one hidden layer only.

How to choose the number of layers and neurons in a neural net?

There are six (rules) rules of thumb to select the number of layers and neurons:

Rule-1: If the problem is non-linear then we start with two (2) layers and observe how it performs on it.

Rule-2: More non-linear is the problem, the more layers you need. If you observe that the problem is not getting solved, then add another layer to it and try. You can also try with more

epochs, more training data. It has been observed that if you have moved till 5-layers then the 6th layer probably will not help.

Rule-3: More are the number of neurons in a layer, the more clearly it will be able to understand the data. You can add more neurons or more epochs or more training data.

Rule-4: More data inputs you have, the more neurons are needed in the first hidden layer.

Rule-5: More output neurons you have, the more neurons you are likely to need in the final hidden layer.

Rule-6: More are the layers, more will be the benefit from activation functions (to be discussed).

The time spent in training a deep learning model is given by the formula:

Training Time (TT) = [Number of training samples] * [Number of Epochs]

How much training is required depends on the weights in our model. **Please note that more are the weights, more can be learned from the model.**

The number of weights between two layers is given by the formula:

Weights (W) = [Number of Neurons in layer-1] * [Number of Neurons in layer-2]

Also note that you will also have to count two more layers: the input layer and the output layer.

For example, in a 50*50 network, with 2 numeric inputs and 1 output, there are:

(2*50) + (50 * 50) + (50 * 1) = 100 + 2500 + 50 = **2650 weights.**

Now, say, you add another 3rd layer with 50 neurons, then we can compute weights required as follows:

(2 * 50) + (50 * 50) + (50 * 50) + (50 * 1) = 100 + 2500 + 2500 + 50 = **5150 weights.**

It has been observed that if more are the weights, slower is the training.

Researchers say that deep learning is a new AI that can tell whether our tweets are sarcastic (abusive) or not. Scientists at Massachusetts Institute of Technology (MIT) USA have developed a new artificial system that can detect sarcasm in tweets better than humans. It's developers say that this advancement may help computers automatically spot and remove online hate speech and abusive comments. Detecting the sentiments of social media posts can also help gauge attitudes towards brands and products and identify signals that might indicate trends in the financial markets. **Please note that they said that a deeper understanding of Twitter may also help understand how information and influence flows through the network.** The researchers originally aimed to develop a system capable of detecting racist posts on Twitter. However, the meaning of many messages could not be properly understood without some understanding of sarcasm. **Also note that the algorithm uses deep learning, a popular machine learning technique that relies on training a very large simulated neural network to recognise subtle patterns using a large amount of data.** Researchers took advantage of *emoji* to help the algorithm identify and label emotional content.

According to the report of "MIT Technology Review", 2017, once the system read tweets for emotions, the researchers taught it to recognise sarcasm. According to MIT, the neural network learned the connection between a certain kind of language and an emoji.

2 Types of Learning

2.0 Introduction

We need to define a learning system first of all. **It is defined as an interaction of a learning agent / intelligent agent with its environment by perceiving an input and doing actions accordingly.** An agent is one who can perceive its environment with the help of sensors and acts upon its environment through actuators.

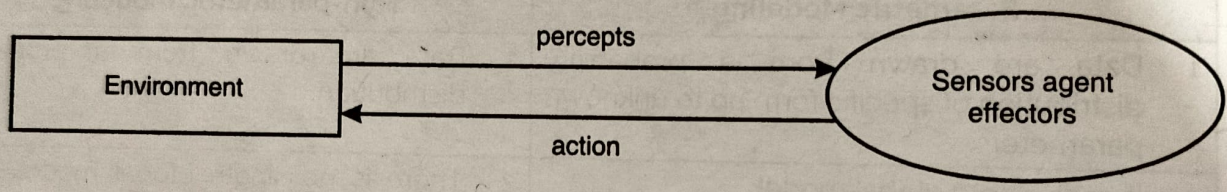


Fig.2.1: A Learning System

An intelligent agent is a software entity which senses its environment and then carries out some set of operations on behalf of a user with some amount of autonomy and to do so it employs some knowledge or representation of the end-user's goal. Note that an agent is different from a program. An agent need not be a program at all. It may be a robot or a college professor. Software agents are by definition programs but a program must measure up to several marks to be an agent. **For example,** humans, animals, autonomous mobile robots or softbots calculators are **real-world agents. Software agents live in the systems OS.**

As shown in figure 2.1, an **agent** perceives its environment through sensors. It then acts upon that environment through **actuators. For example,**

- Human agent has eyes, ears, nose etc. as sensors while hands, legs, mouth are actuators.
- Software agent receives keystrokes, files, packets are sensory inputs and acts on the environment by displaying on the screen, writing files and sending packets over a network.

Capacity, Overfitting and Underfitting

In deep learning, our neural nets have large number of layers containing many neurons. The number of connections in these models is in millions (very high). This results in **overfitting.**

The problem arises in both the cases *i.e.* when the model is simple or when the model is complex. If the model is simple then it may not be powerful to capture all of the useful information required to solve a problem. On the other hand, if the model is complex then it may result in **overfitting problem.** Hence, there is a need of a trade-off. To resolve this problem, deep learning

solves complex problems with complex models but at the same time take additional steps to avoid overfitting.

Generalization is defined as the ability of machine to learn on being introduced with sets of data during training so that when it is introduced to new and unseen examples then it can perform accurately. To ensure that a network generalizes well, we tend to use a neural network in order to generalize from the training examples to all possible inputs. Whether it is training or overtraining of the network, troubles do arise. We can change the weight in a neural network. Thus, the degree of variability is quite high.

In machine learning, overfitting occurs when a statistical model defines random error or noise rather than underlying relationship. When a model is extremely complex, overfitting is observed because of having too many parameters relating to the number of training data types. The model exhibits poor performance which has been overfit. The risk of overfittting exists as the standards used for training the model is not the same as the standards used to judge the effectiveness of a model.

By means of loads of data, overfitting can be avoided. Overfitting occurs when you have a smaller data set and you make effort to learn from it. **Note that if you have a small database and it is desired to build a model based on that then in such a situation a technique known as cross validation can be used.** In this method, the dataset is is divided into two segments-testing and training datasets. The testing dataset will only test the model while in training dataset, the data points will come up with the model. In this method, a model is generally provided with a dataset of a known data on which training data-set is run and a dataset of unknown data against which the model is tested. The purpose of cross validation is to define a dataset to test the model in the training phase.

2.1 Supervised Learning

Machine learning algorithms, based on the basis of desired outcome is put under two categories:

- Supervised learning.
- Unsupervised learning.

Let us discuss these types first.

Supervised learning is the machine learning task of applying inference function on supervised training data. Training data comprises of a set of training examples. We try to infer a function from the training data. A supervised learning algorithm takes a known set of input data and known responses to the data/output and further training a model to generate reasonable predictions for the response to new data. The main **objective is to learn an unknown function f(x) = y where x is an input example (a vector) and y is the desired output (also known as supervisory signal).** The model used in this type of learning describes the effect of one set of observations on the other set of observations. So, we can say that the user tries to find the connection between the sets of inputs and outputs. In nutshell, this algorithm analyzes the training data and produces an inferred function which is called as a **classifer** (in case of discrete output) or a **regression function** (in case of continuous output). **Also note that for a valid input object, the inferred function should predict correct output value.**

Unsupervised learning involves all of the input data, x, and no corresponding output data, y, is available. The main objective is to model the distribution in the data in order to learn

more about data. **We define unsupervised learning as a type of learning with no correct answers and no teacher available for supervision.** Algorithms are supposed to find the interesting structure present in the data thus figuring out the underlying structure. It enables users to learn larger and more complex problems as compared to supervised learning.

The following are the points of differences:

Table-2.1: Supervised versus Unsupervised Learning

Supervised Learning	Unsupervised Learning
It incorporates **an external teacher** so that each output unit is told what should be the desired response to that input signal.	Uses no **external teacher.**
Global error signals govern learning.	**Local information** is used for learning.
Uses pattern class information of each training pattern.	Adaptively clusters patterns to generate decision codes.
Paradigms are error correction learning, reinforcement learning and stochastic learning.	Paradigms are Hebbian learning and competitive learning.
Learning is usually **off line.** It means that learning and operation phases are different. In off-line learning, all of the given patterns are used together.	Learning is **online**. It means that the agent learns and operates at the same time. Here, the information in each new pattern is incorporated into the agent.
All learning data are stored and can be accessed repeatedly. So, we can see whether we are making progress in training or not.	Locality allows learning in **real time.** So, the progress cannot be monitored.
Inputs and outputs are defined.	Only inputs are defined.
Uses pattern class information of each training pattern.	Adaptively clusters patterns to generate decision class without any prior pattern class information.
Global error signal governs learning.	Local information/ rules are used for learning.
Learning is offline usually.	Locality allows **synapses** to learn in real-time.

Unsupervised learning is put under two categories:

1. **Clustering:** In this method, the inherent groupings in the data are discovered. The organization of unlabeled data into similar groups known as **clusters. And the process of partitioning of a set of observations into subsets (or clusters) so that observation in the same cluster is similar is called as clustering.** So, a cluster consists of data items which are similar in comparison to other clusters which contain dissimilar items. **Please understand that the aim of unsupervised learning is to find clusters of similar inputs in the data without knowing that these data points belong to one class and those to different class.** Rather the algorithm has to uncover similarities for itself. **For example,** there is a collection of 30,000 essays on voting process. These essays can be automatically grouped into small number on the basis of word frequency, page count, word limit etc.
2. **Association:** In this method, we discover a set of rules that describes large portions of data. **For example,** a doctor observes same set of symptoms in several patients suffering

from some disease. Then based on his experience the doctor can conclude that the patient is suffering from the same disease.

k-means Clustering

k-means is a simple unsupervised learning algorithm that solves the well known clustering problems. The working of k-means is similar to that of a real life situation in which we have observed that in a group of tourists with a few tourists guides who hold the umbrella up so that everybody can see them and follow them. On the other hand, in static k-means algorithm the data (*i.e.* the tourists) does not move rather only the tour guide (s) move. Analogous is the situation in k-means clustering.

Basic Principle of k-means

"Divide your data into k groups (you have to specify what 'k' is) such that each data item is closer to the center of its cluster than to the centre of any other cluster."

Explanation of k-means: We classify a given data-set into k number of clusters which is fixed in advance. Now the task is to define 'k' centres (middle point) one for each cluster. But we do not know where their centres are. So, we need an algorithm that will find the middle points. The centre of a set of points is based on two parameters:

(a) **Distance Measure:** It measures the distance between a set of points. We may use Euclidean distance but other options are also available.

(b) **Mean Average:** after we compute the distance measure (as stated above) we finally compute the **central point of the set of data points** which is the mean average.

Please note that this is valid in Euclidean space where everything is flat and easily measured. Now the need is to place cluster centres properly. We compute the mean of each cluster, μi (i) and position the cluster centre right there. **Also note that this is similar to minimizing the Euclidean distance** (*i.e.* sum-of-squares again) **from each data point to the centre of cluster.**

Now we would like to find out that which point belongs to which cluster. This is mandatory as it lets us know the position of cluster centres. It is crystal clear to associate closest point with the cluster centre. This might change as the algorithm iterates on. In the input space we start positioning the cluster centres randomly as we do not know where to put them exactly and further update their positions in accordance with the data. Now we start picking up the data points and find out which cluster, particular data point belongs to depending on the distance between each data point and all of the cluster centres and assigning it to cluster that is closest.

The purpose of the hidden layer is to summarize the data and compress it. The multidimensional data is to be reduced to either 2 or 3 dimensions only as then we can plot the graphs too. If we use the PYTHON LANGUAGE then note that layers and column indices are counted from zero while in R they are counted from one.

With auto-encoding neural networks we do learning one layer at a time.

For example, a model, m1, say, has the following-

- 5 hidden layers
- 11 neurons (in middle layer)

It is trained on the raw data. Then that third hidden layer is extracted into a transformation, f1 with still 100 rows but only 11 columns. Model-2 (m2) uses this f1 as its input and builds up a much simpler model, reducing 11 input nodes to 2 hidden nodes then back out to 11 output nodes. At the end, results are put in f2. f2 now has 2 columns but still 100 rows.

Algorithm

k-means clustering involves the following steps:

Step-1: Randomly select 'c' cluster centres.

Step-2: Compute the **distance** between each data point and cluster centres.

Step-3: Assign the data point to the cluster whose distance from the cluster centre is **minimum** in the value of cluster centres.

Step-4: Calculate the **new cluster centre** using the following formula:

$$v = 1/c_i \Sigma (j=1 \text{ to } c_i) x_i$$

where c_i represents the number of data points in the i^{th} cluster.

Step-5: New **cluster centres** are computed.

Step-6: If no data point is reassigned then stop else repeat from step-3.

As we know that H2O is software for machine learning and data analysis. H2O algorithm has been developed keeping in mind clustering only. But clusters in H2O have some limitations:

(a) Each H2O node on the cluster must be of the same size. It implies that if the smallest machine in our cluster has 2GB free, then every node must be given 2GB even if some of them have 32GB.

(b) You cannot add the machines once the cluster has started.

(c) In case a machine dies then the whole of the cluster must be rebuilt. **Please note that the whole cluster becomes unusable if a single node gets removed. In such a case, you cannot even export data or models from the left over clusters.**

(d) Each node must run the same API version of h2o.jar.

(e) To reduce the network latency, the nodes should be physically close.

You can also give your cluster a name and specify this name when starting on each node. In H2O, the cluster can be created in two ways:

1. **Flat-file:** A simple text file giving the IP address and port number of each node in the cluster. **Note that you must prepare this file identically on each machine.**
2. **Auto-discovery:** When you specify the network address then it searches through that subnet to find all nodes and join them together in a cluster. This is slower than the flat file.

n the new space. **Thus, if the classification techniques reforms data features to new** ore applying classifier then we call them as non-linear methods.

3.3 Decision Trees

Some definitions of decision trees are as follows:

1. A decision tree is a **simple representation for classifying examples. Decision tree learning** is one of the most successful techniques for **supervised classification learning.**
2. A decision tree or a **classification tree** is a tree in which each internal (non-leaf) node is labeled with an **input feature.** The **arcs** coning from a node labeled with a feature are labeled with each of the possible values of the feature. Each **leaf** of the tree is labeled with a class or a probability distribution over the classes.
3. In this method, a set of training examples is broken down into smaller and smaller subsets while at the same time an associated decision tree gets incrementally developed. At the end, of the learning process, a decision tree covering the training set is returned.

Principle: "Use a decision tree to partition the data space into the cluster (or dense) regions, and empty (or sparse) regions". In decision tree learning a **new example** is classified by submitting, it to a **series of tests** that determine the class label to the example. these tests are organized in a **hierarchical structure** called as a **decision tree.**

Another term, **classification tree analysis** is used when the predicted outcome is the class to which the data belongs. While **regression tree analysis** term is used when the predicted outcome can be considered a real number e.g. the price of a house.

Internal rode represents a test on an attribute. **Branch** represents an outcome of the test. Leaf **nodes** represent class labels or class distribution. Note that decision trees can be easily converted to classification rules. (see figure 3.6)

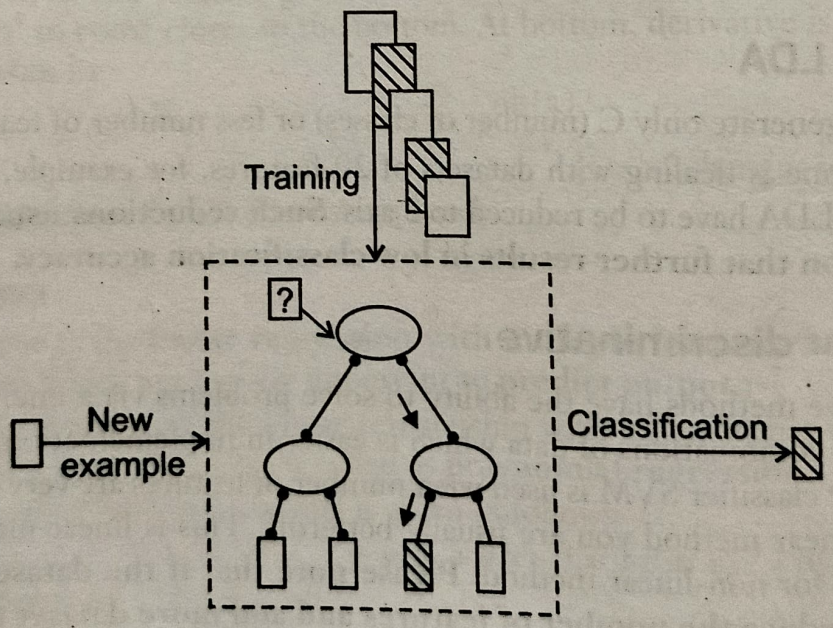


Fig.3.6: A Decision Tree

Decision Tree's basic Divide-and-Conquer Algorithm

Objective : To generate a decision tree from the training tuples of data partition, D.

Input:

1. Data partition (D).
2. Attribute List.
3. Attribute Selection method.

ALGORITHM:

S1: Select a **test** for root node. Create **branch** for each possible outcome of the test.

S2: **Split instances** into **subsets,** one for each branch extending from the node.

S3: **Repeat recursively** for each branch, using only instances that reach the branch.

S4: **Stop recursion** for a **branch** if all its instances have the **same class.**

Goal: "Build a decision tree for classifying examples as positive or negative instances of a concept."

Decision tree generation consists of two phases as follows:

Phase-1: Tree construction phase.

In this phase, all the training examples are at the root. We partition examples recursively based on selected attributes.

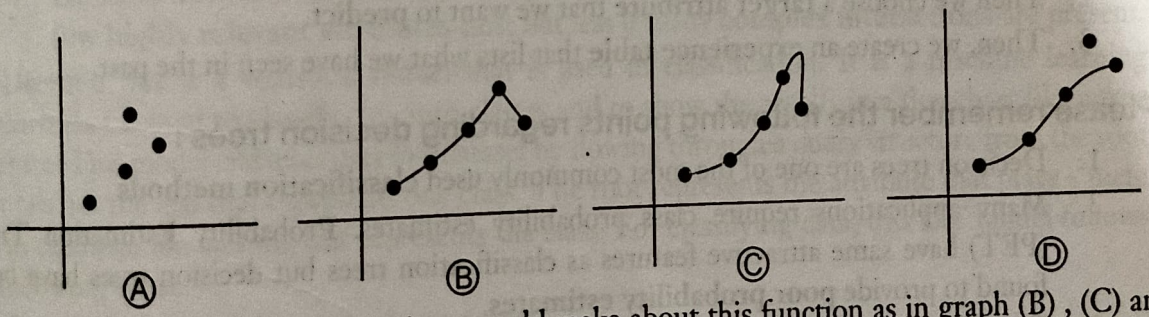
Phase-2: Tree - pruning phase.

In this phase, identify and remove the branches that reflect noise or outliers.

Several different paradigms are used for learning binary classifiers-

1. Decision Trees.
2. Neural Networks.
3. Bayesian Classification
4. Support Vector Machines (SVM).

Suppose we want to learn a function, f(x) as y and we are given some sample (x,y) pairs as shown :



i.e., there are several hypothesis we could make about this function as in graph (B) , (C) and (D).

Now a preference of one over the other reveals the bias of our learning technique.

For instance,

Prefer piece-wise functions.

Prefer a smooth functions.

Prefer a simple functions and treat outliers as noise.

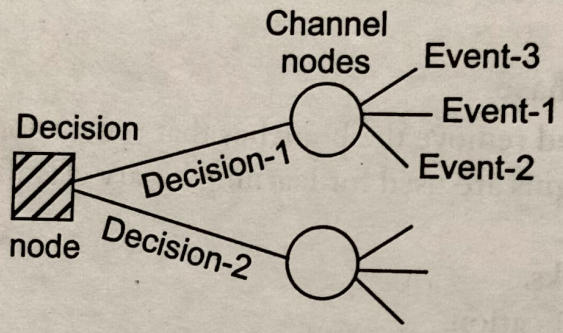
1. **Preference Bias** : The **smallest decision tree** that correctly classifies all of the training examples is best.

Please note that finding the provably smallest decision tree is an NP-hard problem. So, instead of constructing the absolute smallest tree that is consistent with all of the training examples, construct one that is pretty small.

2. **Training Set Error** : For each record, **follow** the **decision tree** to see what it would predict. For what number of records does the decision tree's prediction disagree with the true value in the database? This quantity is called as the **training set error. Also note that the smaller it is, better it is.**
3. **Test Set Error** : We **hide** some data away when we learn the decision tree. But once learned, we see how well the tree predict **future data** . It is known as the **Test Set Error.**

Decision Tree involves 3-types of nodes as follows:

1. **Decision nodes:** represented by **squares.**
2. **Chance nodes:** represented by **circles.**
3. **Terminal nodes:** represented by **triangles** (optional).



To create a decision tree , we follow these steps:

1. We make a **list of attributes** that we can measure. These attributes must be **discrete.**
2. Then we choose a **target attribute** that we want to **predict.**
3. Then, we create an **experience table** that lists what we have seen in the past.

Please remember the following points regarding decision trees :

1. Decision trees are one of the most commonly used **classification methods** .
2. Many applications require class probability estimates. **Probability Estimation Trees** (PET) have same attractive features as classification trees but decision trees have been found to provide **poor probability estimates.**
3. For different **classification models** like decision trees, a variety of methods for obtaining **good probability estimates** have been proposed in literature.
4. A **leaf node** represents the subset of instances corresponding to the conjunction of conditions along its branch or path back to the root. So, estimate the probability of an instance's membership in a class and **assign** that **probability** as the **instance's rank.** A decision tree can easily be used to estimate these probabilities.
5. **Rule learning** is good to yield better class predictions.
6. In **classification rule mining,** one will search for a set of rules that describes the data as accurately as possible.

7. A **probabilistic rule** is an extension of a classification rule (above) which does not only predict a **single class** value but a **set of class probabilities**. This forms a probability distribution over the classes. We get **one class probability per class**.
8. **Error rate** does not consider the probability of the prediction.
9. A **small tree** has a small number of leaves . So, more examples will have the same class probability. This prevents the learning algorithm from building an accurate PET . On the other hand , **if tree is large** , not only may the **tree over fit** the training data but the number of examples in each leaf is also **small**.

   Thus , the probability estimates would **not be accurate** and **reliable**. Such a contradiction does exist in **traditional decision trees**.
10. Applying a **learned PET** involves **minimal** computational effort, which makes this **tree-based approach** particularly suited for a fast re-ranking of large sets of candidates.

Advantages of Decision Tree

1. Decision trees can handle both nominal and **numeric** input attributes.
2. They are good enough to represent any **discrete value classifier.**
3. They are capable of **handling data sets** that may have **errors.**
4. They are capable of **handling data sets** that may have **missing values.**
5. It is **self-explanatory** and when compacted they are easy to follow.

Disadvantages of Decision Trees

1. Most of the **algorithms** require that the **target attribute** will have only **discrete values.**
2. Most **decision-tree algorithms** only examines a **single field** at a time.
3. They are **prone to errors** in classification problems with many classes.
4. Decision trees use **"Divide-and-conquer strategy"** and hence tend to perform well if a **few highly relevant attributes** exist but less if many **complex interactions** are present.

Decision tree is a statistical model that is used in classification. It is a machine learning approach that is used to classify data into classes and to show the results in a flowchart like a tree structure. This model classifies data in a dataset by flowing through a query structure from the root till it reaches the leaf that represents one class. The root represents the attribute that plays a main role in classification and the leaf represents the class. For classifying data, this DT model follows the following steps:

Step-1: Puts all training examples to a **root.**

Step-2: It **divides** training examples based on the selected attributes.

Step-3: It selects attributes by using some **statistical methods.**

Step-4: **Recursive partitioning** continues till no training example is left or until no attribute is left or the remaining training examples belong to the same class.

Decision trees were first of all used in 1970s. In this technique, a sample of observations is used as a starting point. The algorithm retraces the rules that generate the output classes by dividing the input matrix into smaller and smaller partitions till the process triggers a rule for stopping. This is inverse reasoning. But in terms of machine learning, this is achieved by applying a search among all possible ways to split the training in a greedy way. **An algorithm is said to be greedy**

if it always chooses its move to maximize the result in each step during the optimization process regardless of what could happen in the following steps. So, an algorithm tries to maximize the current step without looking forward to obtain a global optimization.

Principle used: "Each partition of the initial data must make it easier to predict the target outcome, which is characterized by a different and more favourable distribution of classes than the original sample. The algorithm creates partitions by splitting the data. It finds out the data splits by first evaluating the features and then the values in the features that could bring the maximum improvement of a special statistical measure that plays the role of the cost function in a decision tree".

A number of statistical measurements determine how to make splits in a decision tree. But they follow one rule that a split must improve on the original sample or on another possible split. Most common measurement is the **information gain. It tells how a decision tree can detect an increased predictive ability or a reduced risk in a simpler way for some split.** in 1970s, Ross Quinlan, developed a decision tree algorithm based on information gain only. Information gain relies on the following formula of **information entropy:**

$$\text{Entropy} = \Sigma - p_i \log_2 p_i$$

Where 'p' is the probability for a class (range of 0 to 1).

For example, consider an example wherein you want to classify two classes having the same probability of 50/50, then the maximum possible entropy is given by:

$$\text{Entropy} = -0.5 * \log_2 0.5 - 0.5 * \log_2 0.5 = 1.0$$

On the other hand, if the decision tree detects a feature that can split the dataset into two partitions where the distribution of the two classes is 40/60 then the average entropy reduces as follows:

$$\text{Entropy} = -0.4 * \log_2 0.4 - 0.6 * \log_2 0.6 = 0.97$$

Please note the entropy sum for all classes. It is observed that using the split of 40/60, the sum is less than the theoretical maximum of 1. **This is known as diminishing of the entropy. We can think of entropy as a measure of the mess in the data. Also note that the lesser is the mess the more is the order and easier it is to guess the right class.**

Explanation: after the 1st split, the algorithm tries to split the obtained partitions further using the same logic of reducing entropy. It progressively splits any successive data partition until no more splits are possible as the sub-sample is a single example or because it has met a stopping rule. Stopping rules are defined as the limits to the expansion of a tree. These rules work by considering 3 aspects of a partition:

(a) Initial partition size.

(b) Resulting partition size.

(c) Information gain; as obtained after the split.

Please understand that these stopping rules are vital because decision tree algorithms approximate a large number of functions. But noise and data errors do affect this algorithm.

Decision trees have several nuts and bolts. They have more variance than bias in their estimations. To overfit the data in less, it also does **tree pruning. Pruning is done when the tree is fully grown.** Starting from the leaves, prune the branches of the tree in order to show some

improvement in the **reduction of information gain. Also understand that retracting from the leaves to the root and keeping only the branches that have some predictive value mitigates the variance of the model and thus making the resulting rules parsimonious.**
Pruning is just like brainstorming due to two reasons:

1. The code generates all possible ramifications of the tree.
2. When the brainstorming concludes the code keeps only what really works.

In nutshell, we induce Decision Tree model using Greedy method from top-to-down. **Please note that this top-down method of induction of decision trees is the most common method used to learn decision trees from the data. Also note that this type of DT model based learning is supervised learning as it constructs decision tree from class-labelled training tuples.**

In decision trees, we break the classification down into a set of choices about each feature starting at the root of the tree and progressing down to leaves where we receive the final classification decision. The data structure, tree, is very easy to understand and implement.

In this approach, we have handy, if-then-else rules which are useful for use in a production system. This type of algorithm has been applied successfully to a broad range of tasks from learning to diagnose medical cases to learning to assess credit risk of loan applicants.

Decision trees classify instances by sorting them down the tree from the root to some leaf nodes which provides the classification of the instance. Each **node** in the tree specifies a test of some attribute of the instance and each **branch** descending from that node corresponds to one of the possible values for this attribute. **An instance is classified by starting at the root node of the tree, testing the attribute specified by this node then moving down the tree branch corresponding to the value of the attribute in the given example.** This process is then **repeated** for the subtree rooted at the new node.

This method is very popular because we can turn into a set of logical disjunctions (if-then-else rules) that fit into program code easily.

Limitations of Decision Trees

1. **Instances** are represented by **attribute-value pairs.**
2. Target function has discrete output values.
3. **Disjunctive descriptions** may be required.
4. **Training data** may contain **errors.**
5. Training data may contain **missing attribute values.**

This method of learning has been applied to problems like learning to classify medical patients on the basis of their disease, equipment breakdowns by their cause and loan applicants by their possibility of non-payment. Such problems, in which the task is to classify examples into one of a discrete set of possible categories, are often stated as **classification problems.**

For constructing a decision tree, the features that are required for the algorithm are as follows:

1. The state of energy level.
2. The date of your nearest deadline.
3. Whether or not there is a party tonight.

Now the challenge is to construct a tree based on these features. Literature provides different types of decision tree algorithms that are based on the same principle but follow different approaches.

Principle: "The algorithm states that the algorithm built the tree in a greedy manner starting at the root and choosing the most informative feature (how the tree works) at each step".

ALGORITHM ID3 (by Quinlan)

This algorithm learns decision trees by constructing them in top-down fashion *i.e.* starts with the question "at the root of the tree, which attribute is to be tested?" In order to answer this question, each **instance attribute** is assessed using a statistical test to find out how well it classifies the training examples. *The best attribute is selected to be used as the test, as the root node of the tree. Then a successor of the root node is produced for each possible value of this attribute and the training examples are sorted to the suitable successor node i.e. down the branch conforming to the example's value for this attribute. The complete process is then repeated using the training examples related with each descendant node to select the best attribute at test at that point in the tree.* This practices a greedy search for an acceptable decision tree, in which the algorithm never backtracks to review earlier options.

Now the question is which attribute is the best classifier?

Selecting which attribute that is to be tested at each node in the tree is the basic factor that is to be considered in ID3 algorithm. We would like to make choice of the attribute that is most beneficial for the classifying examples.

What is a good quantitative measure of the value of an attribute?

A statistical property called as an **information gain will compute how well a given attribute separates the training examples** in accordance to their target classification. ID3 uses up this information gain measure to make a choice amongst the candidate attributes at each step while the tree grows.

ID3 (Examples, Target-attribute, Attributes)

(where, **Examples** are the training examples, **Target-attribute** is the attribute whose value is to be predicted by the tree and **Attributes** is a list of other attributes that may be tested by the learned decision tree) (this algo returns a decision tree that correctly classifies the given Examples)

Step-1: Create a root node for the tree.

Step-2: If all Examples are positive, return the single-node tree Root, with label = +

Step-3: If all Examples are negative, return the single-node tree Root, with label = -

Step-4: If Attributes is empty, return the single-node tree Root with label = most common value of Target-attribute in Examples.

Step-5: Otherwise Begin

1.1 A ← the attribute from Attributes that best classifies Examples.

1.2 The decision attribute for Root ← A.

1.3 For each possible value, v_i, of A,

1.3.1 Add a new tree branch below Root, corresponding to the test $A = v_i$.

1.3.2 Let Example, sv_i be the subset of Examples that have value v_i for A.

1.3.3 If Example sv_i is empty

1.3.3.1 then below this new branch, add a leaf node with label = most common value of Target-attribute in Examples.

1.3.3.2 Else below this new branch add the subtree ID3 (Examples v_i, Target-attribute, Attributes – (A))

Step-6: End

Step-7: Return Root

In nutshell, ID3 algorithm is dedicated to learning Boolean-valued functions. ID3 is a greedy algorithm that develops the tree in top-down fashion, at each node choosing the attribute that best classifies the local training examples. This process carries until the tree perfectly classifies the training examples or until all attributes have been taken into consideration.

ID3 algorithm is also **called as C5.0** It was developed by Ross Quinlan in 1987. ID3 uses **information gain** as a **splitting criteria**. The **growing stops** when all instances belong to a single value of target feature or when **best information gain** is NOT greater than **zero**. ID3 does not apply any **pruning procedures** nor does it handle numeric attributes or missing values.

Top-down construction of the decision tree is done by **recursively** selecting the **"best attribute"** to use at the current node in the tree. Once the **attribute is selected** for the current node ,**generate children** nodes, **one for each possible value** of the selected attribute. We **partition** the examples using the possible values of this attribute and assign these subsets of the examples to the **appropriate child node. Repeat** for each child node until all examples associated with a node are **either all positive** or **all negative.**

The problem lies in choosing which attribute to split a given set of examples. Some possibilities are :

(a) **Random** select any attribute at random.

(b) **Least-Values** : Choose the attribute with the **smallest number** of possible values.

(c) **Most-Values** : Choose the attribute with the **largest number** of possible values.

(d) **Max-Gain** : choose the attribute that has the **largest expected information gain** *i.e.*, the attribute that will result in the **smallest expected size of the sub-trees** rooted at its children.

This ID3 algorithm uses the **Max-Gain Method** of selecting the best attribute. A **branch** is created for each value of the **node-attribute** and the samples are partitioned accordingly. The algorithm uses the **same process recursively** to form a decision tree at each partition. Once an attribute has occurred at a node, it need not be considered in any other of the **node's descendants** **The recursive partitioning stops** only when any one of the following conditions is true. All **records** (samples) for the given node belong to the **same class** or there are **no remaining attributes** on which the records may be further partitioned. In this case , we convert the given node into a **leaf** and label it with the class in majority among samples. There is **no record left.**

ID3 Limitations:

1. It requires **lot of computation** at every stage of **construction of decision tree.**
2. It needs all of the **training data** to be **in memory.**
3. It does **not** suggest any **standard splitting index** for range attributes.

3.4 Probabilistic (conditional and generative)

We discuss two types of models here:

1. Generative Model.
2. Conditional Model.

i. **A generative model is a model that generates observable data values randomly, usually provided with some hidden parameters.** It states *'a joint probability distribution over observation and label sequences'*. As far as machine learning is concerned, these models are are used for either modelling data directly or as an intermediary phase to form a conditional probability density function. A conditional distribution can be derived from a generative model with the help of Bayes Rule. **For example,** Gaussian mixture model, Hidden Markov model, Naive Bayes, Restricted Boltzmann machine, generative adversial networks, probabilistic context-free Grammer etc.

ii. **Conditional models or discriminative models are used in machine learning for demonstrating the dependence of an unobserved variable 'y' on an observed variable 'x'.** In terms of probability, this is prepared by modelling the conditional probability distribution P(y| x), which can be used for predicting from discriminative models, in contrast to generative models, that do not permit to generate samples from the joint distribution of x and y. **For example,** Logistic regression (a type of generalized linear regression used for predicting binary or categorical outputs—called as **maximum entropy classifiers,** Support Vector Machines, Boosting-a meta-algorithm, conditional random fields, linear regression, neural networks etc.

Tasks like classification and regression do not require joint distribution. In fact, conditional models can return excellent performance. **Please understand that these generative models are more flexible than discriminative models in stating dependencies in complex learning tasks. Also note that most discriminative models are fundamentally supervised and cannot easily be stretched to unsupervised learning.** Application decides whether to go for generative or conditional model.

Say, the input data is 'x' and the set of labels for x is 'y'. Then,

1. **Generative model learns the joint probability distribution p(x, y).**
2. **Discriminative model learns the conditional probability distribution p(x | y)** *i.e.* **the 'probability of y given x'.**

3.5 K-nearest Neighbor Learning-distance weighted Nearest Neighbour Algorithm

It is an instance-based method.

Principle: "This algorithm assumes that all instances relate to points in the n-dimensional space, R^n. The nearest neighbours of an instance are defined in terms of the standard Euclidean distance."

Let feature vector describe an arbitrary instance, x. The feature vector is-

$$<a_1(x), a_2(x), \ldots\ldots a_n(x)>$$

Where $a_r(x)$ denotes the values of the rth attribute of instance, x. Then the distance between two instances, x_i and x_j is defined to be $d(x_i, x_j)$ and is calculated as follows:

$$d(x_i, x_j) = \sqrt{\Sigma(r = 1 \text{ to } n)\left(a_r(x_i) - a_r(x_j)\right)^2}$$

In nearest-neighbor learning, the target function may be either for discrete-valued or real-valued. Let us first of all study learning discrete valued target functions of the form-

$f: R^n \rightarrow V$ where V is the finite set $\{v_1, \ldots\ldots v_s\}$.

ALGORITHM k-nearest Neighbor { a discrete-valued target function }

k-Nearest Neighbor Algorithm (1970s)

Let 'm' be the number of training data samples. Let 'p' an unknown point. KNN algorithm is as follows :

S1: **Store the training samples** in an array of data points, a [ ]. **This means that each element of this array represents a tuple (x , y).**

S2 :For i=0 to m, calculate **Euclidean distance** *i.e.*, d (a [i] , p).

S3: Make set S of K **smallest distances** obtained. Each of these distances corresponds to an already classified data point.

S4: Return the majority label among S.

K-nearest neighbor is an algorithm which **stores** all of the dataset and classifies new cases based on a similarity measures such as **Euclidean distance.** It does not require any parameters to be estimated.

Applications: KNN is used in statistical estimation and pattern recognition.

Characteristics of KNN

1. It stores the **entire training dataset** that is used to classify every single data instance. This affects **computational time** of the system.
2. It does not learn any model. The prediction of class is instantaneous as there is **no model.** But classification can take **more time** for larger datasets.
3. It makes **just-in-time** predictions by calculating the **similarity** between an input sample and each training instance.

Advantages (of KNN)

1. **Insensitive to outliers.** As the distance from the outliers will be more so the probability of outliers selection in the neighborhood is very less.
2. **Pattern of data distribution** has no effect on classification as we consider a neighborhood for classification.
3. KNN is **simple** to implement.
4. Better **accuracy.**
5. KNN allows us to use the **same algorithm** for the purpose of classification and regression.

Disadvantages (of KNN)

1. KNN requires the evaluation of similarity measure of all training instances, so it is **computationally expensive.**
2. **More memory** is required to store data .
3. It is **sensitive** to irrelevant features.
4. **More time** needed to compute similarity measures for all instances.

We are in a position to solve an example based on KNN now.

Example 1: Consider the data from a survey to classify whether the given piece of cloth is good or NOT. Four training samples are given in table :

X1 (Durability)	X2 (Strength)	Y (Classification)
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

Now, a company produces a new cloth that pass lab. test with X1=3 and X2=7. Classify this now cloth without any further survey.

Solution 1 It is possible to classify this new cloth by following these five steps as follows:

S1: **Determine parameter, K** *i.e.*, no. of nearest neighbors. Say, K=3.

S2: **Compute the distance between the query-instance (3,7)and training samples** as follows :

X1	X2	Square distance to (3,7)
7	7	$(7-3)^2+(7-7)^2=16$
7	4	$(7-3)^2+(4-7)^2=25$
3	4	$(3-3)^2+(4-7)^2=9$
1	4	$(1-3)^2+(4-7)^2=13$

S3 : **Sort the distance** and find the **nearest neighbors** based on the **K^{th} minimum distance** as follows :

X1	X2	Squire distance to (3,7)	Rank based on distance	Included in Neighborhood
7	7	16	3	Yes
7	4	25	4	No
3	4	9	1	Yes
1	4	13	2	Yes

S4 : **Gather** the category Y of the nearest neighbors.

The categories are given (in Question). So, the result is :

X1	X2	Squire distance to (3,7)	Rank based on distance	Included in Neighborhood	Category
7	7	16	3	Yes	Bad
7	4	25	4	No	–
3	4	9	1	Yes	Good
1	4	13	2	Yes	Good

S5 : Observe the category column in the above table :

No. of bad categories =1

No. of good categories is = 2

Now, **majority wins**. As 2>1 , by voting, the cloth can be classified a **"good"** category.

Training algorithm

For each training example (x, f (x)), add the example to the list of training examples.

Classification algorithm

Given instance x_q is to be classified. Let x_1.....x_k denote the k-instances from training examples that are nearest to x_q. Return

$$f'(x_q) \leftarrow \text{argmax}\,(v \varepsilon V) \sum_{(i=1 \text{ to } k)} \delta(v, f(x_i))$$

where δ (a, b) = 1 if a=b and where δ (a, b) = 0 otherwise.

Explanation: The value f' (x), given back by the algorithm as its approximation of $f(x_i)$ is just the most common value of f amongst the k training examples which are nearest to x.

If we choose k=1, then the 1-nearest neighbour algorithm assigns the value $f'(x_i)$ to $f(x_i)$ where x_i is the training instance which is nearest to x_i.

For larger values of k, the algorithm allots the most common value among the k-nearest training examples.

Figure 3.7 explains the working of the k-nearest neighbour algorithm for the case where the instances are basically points in a 2D space. Also the target function is in form of Boolean. The positive and negative training examples are depicted by "+" and "-" respectively. **Please note that the 1-nearest neighbour algorithm classifies x, as a positive example in figure 3.7 whereas the 5-nearest neighbour algorithm classifies it as a negative example. Also note that k-nearest neighbour algorithm never forms a clear hypothesis regarding the target function, f.** It just works out for the classification of each new query instance encountered as needed.

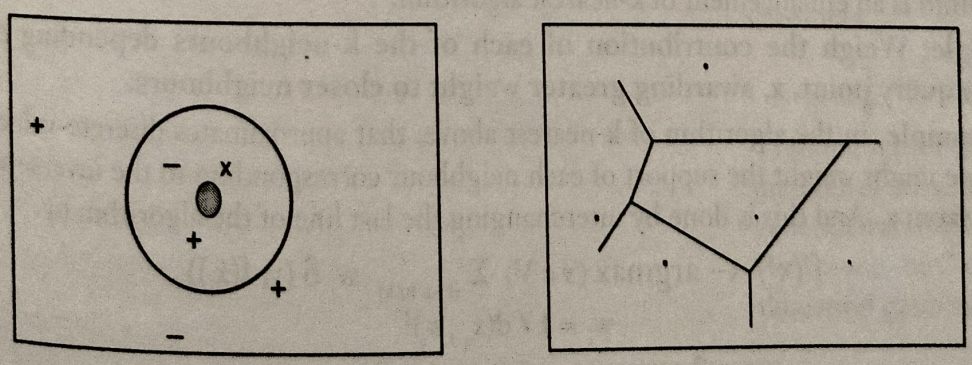


Fig.3.7: k-Nearest Neighbour

Explanation: A set of positive and negative training examples is shown on the left diagram of figure 3.7. The 1-nearest neighbour algorithm classifies x as positive whereas 5-nearest neighbour classifies it as negative. On the right hand side of the figure 3.7, is the decision surface induced by the 1-nearest neighbour algorithm for a typical set of training examples. **Please note that the convex polygon surrounding each training example indicates the region of instance space closest to that point.** This means that the instances for which the 1-nearest neighbour algorithm will assign the classification belonging to that training example. On the right side of the figure 3.8, is shown a shape of the **decision surface** made by 1-nearest neighbour over the complete instance space. The decision surface is basically a grouping of convex polyhedral surrounding each of the training examples. For every training example, the polyhedron shows the set of query points whose classification will be completely decided by that training example. **Also note that the query points that lie outside the polyhedron are closer to some other training example. And such a type of diagram is often called as the Voronoi diagram of the set of training examples.**
The k-nearest neighbour algorithm is easily adapted to approximating continuous-valued target functions. To accomplish this, we need the algorithm to calculate the mean value of k-nearest training examples rather than calculate their most common value. Practically speaking, to estimate a real-valued target function f: Rn → R, we replace the final line of the above algorithm by:

$$f'(x_q) = \Sigma \text{ (i=1 to k) } f(x_i) / k$$

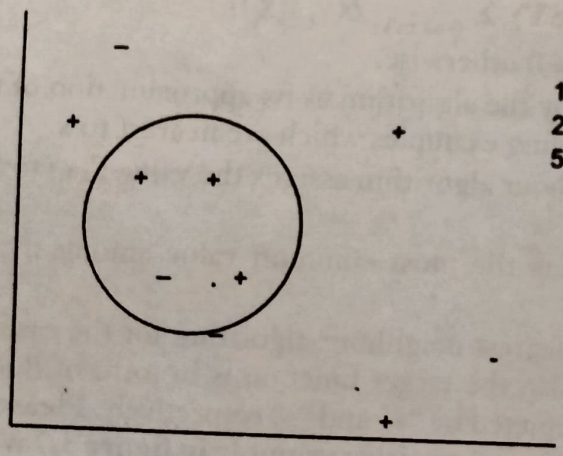


Fig.3.8: Accurate Estimate of Real-valued target Function

Distance weighted nearest-Neighbour Algorithm

This algorithm is an enhancement of k-nearest algorithm.

Principle: Weigh the contribution of each of the k-neighbours depending on their distance to query point, x, awarding greater weight to closer neighbours.

For example, in the algorithm of k-nearest above, that approximates discrete-valued target functions, we might weight the support of each neighbour corresponding to the inverse square of its distance from x_i. And this is done by interchanging the last line of the algorithm by:

$$f'(x_q) \leftarrow \text{argmax } (v \varepsilon V) \ \Sigma_{(i=1 \text{ to } k)} \ w_i \ \delta (v, f(x_i))$$

where

$$w_i = 1 / d(x_q, x_i)^2$$

To consider the case where the query point, x, exactly matches one of the training instances, x_i and the denominator $d(x_q, x_i)^2$ is therefore zero, we assign $f'(x_q)$ to be $f'(x_i)$ in this particular case. We can distance-weight the instances for real-valued target functions in alike fashion, interchanging the final line in the algorithm in this case by-

$$f'(x_q) \leftarrow \Sigma \text{ (i=1 to k) } w_i f(x_i) / k / \Sigma \text{ (i=1 to k) } w_i$$

where w_i is defined as above.

3.6 Curse of Dimensionality

Distance usually relates to all the attributes and assumes all of them have the same effects on the distance. The similarity metrics do not consider the relation of attributes which result in inaccurate distance and then impact on classification precision. **This wrong classification due to presence of many irrelevant attributes is often termed as the curse of dimensionality.**

For example, say each instance is described by 20 attributes out of which only 2 are relevant in determining the classification of the target function. In this case, instances that have identical values for the 2 relevant attributes may nevertheless be distant from one another in the 20 dimensional instance space.

3.7 Ensemble Learning

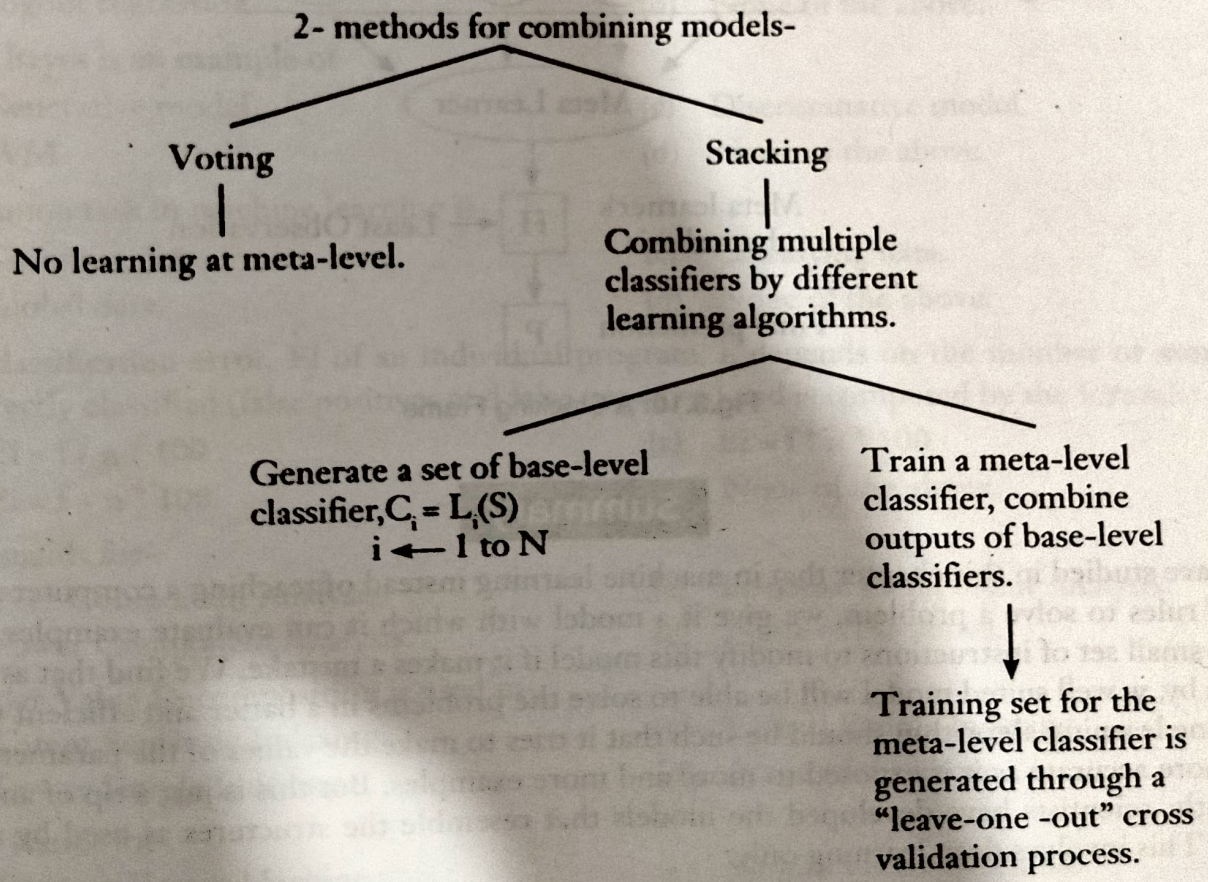


Fig.3.9: Overall Voting and Stacking Process

Conceptual Short Questions with Answers

Q1. What are the differences between data mining and machine learning?

Ans. **Data mining** is the process in which the unstructured data puts efforts to extract knowledge or unfamiliar interesting patterns. On the other hand, **machine learning** relates to the study, design and development of the algorithms that give systems the ability to learn without being clearly programmed.

Q2. Give functions supervised and unsupervised learning.

Ans. **Supervised learning** involves-

(a) Classifications.
(b) Speech recognition.
(c) Regression.
(d) Predict time series.
(e) Annotate strings.

On the other hand, unsupervised learning involves-

(a) Finding the clusters of data.
(b) Finding low dimensional representations of data.
(c) Finding interesting coordinates and correlations.

Finding novel observations or cleaning databases.

Q3. What is inductive machine learning?

Ans. **Inductive machine learning covers the process of learning by examples** where a system, from a set of perceived instances, tries to encourage a general rule.

Q4. List at least five common machine learning algorithms.

Ans. Some of the popular ML algorithms are as follows:

(a) *Decision Trees.*
(b) *Neural Networks.*

(c) Probabilistic Networks.
(d) Nearest Neighbour.
Support Vector Machines (SVM).

Q5. Name different algorithm techniques in ML.

Ans. Different types of techniques in ML are as follows:
(a) Supervised learning.
(b) Unsupervised learning.
(c) Semi-supervised learning.
(d) Reinforcement learning.
(e) Transduction.
Learning to learn.

Q6. Name three stages to build the models (or hypotheses) in machine learning.

Ans. Three stages to build the models in machine learning (ML) are as follows:
(a) Model building.
(b) Model testing.
Applying the model.

Q7. What is the standard approach to supervised learning?

Ans. The standard approach to supervised learning is to break the data set of example into training set and the test set.

Q8. Differentiate between 'training set' and 'test set'.

Ans. In different fields of data science like machine learning, a set of data is used to determine the potentially predictive relationship identified as **training set.** Training set is the example provided to the learner while test set is used to test the exactness of the hypotheses produced by the learner and it is the set of illustration weighed down from the learner. So, training sets are different from test sets.

Q9. What are the different approaches of machine learning?

Ans. Different approaches of machine learning are as follows:
(a) Concept versus Classification Learning.
(b) Symbolic versus Statistical Learning.
Inductive versus Analytical Learning.

Q10. Define Overfitting during machine learning. Why it happens?

Ans. In ML, overfitting occurs when a statistical model defines a random error or noise rather than underlying relationship. When a model is extremely complex, overfitting is usually observed because of having too many parameters relating to the number of training data types. The model exihibits poor performance which has to be overfit. The risk of overfitting exists as the standards used for training the model is not the same as the standards used to judge the effectiveness of a model.

Q11. How can overfitting be avoided?

Ans. By means of loads of data, overfitting can be avoided. **Overfitting** happens when comparatively you have a small dataset and you make effort to learn from it. But if you have a small database and it is necessary to come with a model based on that then in such a scenario, **cross validation** method may be used. In this method, the dataset is divided into two segments, testing and training datasets. The testing dataset will only test the model while in training dataset, the data-points will come up with a model. In this method, a model is generally provided with a dataset of known data on which training data set is run and a dataset of unknown data against which the model is tested. **The basic idea of cross validation is to define a dataset to test the model in the training phase.**

Q12. What is algorithm independent machine learning?

Ans. A type of machine learning in which mathematical foundations is independent of any particular classifier or learning algorithm is known as **algorithm independent machine learning.**

Q13. Distinguish between AI and machine learning.

Ans. Designing and developing algorithms according to the behaviours based on empirical data is known as machine learning. On the other hand, artificial intelligence, in addition to machine learning, also covers other aspects like knowledge representation, natural language processing (NLP), planning, robotics etc.

Q14. Define classifier as used in machine learning.

Ans. A classifier is a system that inputs a vector of discrete or continuous feature vales and outputs a single discrete value, the class.

Q15. Where is pattern recognition used?

Ans. It is used in computer vision, speech recognition, data mining, statistics, information retrieval and bio-informatics.

Q16. What is model selection in ML?

Ans. The process of selecting the models from different mathematical models that refer to the same data set is called as model selection. It is applied to the fields of statistics, machine learning and data mining.

Q17. Name the method that is frequently used to prevent overfitting.

Ans. 'Isotonic Regression' is used to prevent overfiting when the data is sufficient.

Q18. Why instance-based learning algorithm is sometimes also referred to as lazy-learning algorithm?

Ans. Instance-based learning algorithm is sometimes also referred to as lazy-learning algorithm because they delay the induction or generalization process until classification is performed.

Q19. What are the 2 classification methods that SVM handle?

Ans. The two classification methods that SVM can handle are as follows:

(a) Combining binary classifiers.

Modifying binary to incorporate multiclass learning.

Q20. For what purpose PCA and ICA are used?

Ans. PCA and ICA are used to extract features of data to reduce dimensions of data. They are important feature extraction techniques used for dimensionality reduction.

Q21. What do you mean by dimension reduction?

Ans. **It is the process of reducing the number of random variables under considerations and can be divided into feature selection and feature extraction.**

Q22. What are SVMs?

Ans. **SVMs or Support Vector Machines are supervised learning algorithms that are used for classification and regression analysis.**

Q23. What are the different methods to solve sequential supervised learning problems?

Ans. The different methods to solve sequential supervised learning problems are:

(a) Sliding window methods.
(b) Recurrent sliding windows.
(c) Hidden Makov Models.
(d) Maximum entropy Markov Models.
(e) Conditional random fields.

Graph transformer networks.

Q24. Name two techniques of ML. Give a very popular application of ML that you see everyday.

Ans. Genetic programming and inductive learning are the two methods of ML.

The recommendation engine implemented by major ecommerce websites uses machine learning (ML).

Q25. How is KNN different from k-means clustering?

Ans. k-nearest neighbours is a supervised classification algorithm. On the other hand, k-means clustering is an unsupervised clustering algorithm. In order for k-nearest neighbours to work, labelled data is needed that we want to classify an unlabeled point. In case of k-means clustering only a set of unlabelled points and a threshold is required. The algorithm will take unlabelled points in consideration and slowly learn how to cluster them into groups by computing the mean of the distance between different points. In sutshell, KNN needs labelled points and is thus supervised learning while k-means doesn't need labelled points and is thus unsupervised learning.

Q26. Why is "Naive" Bayes naive?

Ans. Even though it has practical applications, specially in text mining, Naive Bayes is still considered as **Naive** because the assumption that it makes becomes virtually impossible to be seen in real-life data. The conditional probability is computed as purely the product of the individual probabilities of different components. This denotes the absolute independence of features *i.e.* a condition that is probably never met in real life.

Q27. Differentiate between a generative and discriminative model.

Ans. A generative model learns categories of data while a discriminative model merely learns the distinction between different categories of data. Discriminative models will leave behind generative models on the basis of classification tasks.

Q28. Name at least three methods of reducing dimensionality.

Ans. The three methods of reducing dimensionality are as follows:

(a) Removing collinear features.

(b) Performing PCA, ICA or other forms of algorithmic dimensionality reduction.

Combining features with feature engineering.

Q29. What is the EM technique? How this algorithm works?

Ans. EM is an algorithm for maximizing a likelihood function when some of the variables in the model are unobserved *i.e.* when there are latent variables with you. To solve for our model parameters you need to know **the distribution of your unobserved data** but the **distribution of your unobserved data is a function of your model parameters.** EEM tries to get around this by guessing repeatedly about the distribution for the unobserved data then further estimating the model parameters by maximizing something that is lower bound on the actual likelihood function and repeating until convergence.

The EM algorithm starts with a guess for values of your model parameters as follows:

The E-step: For each data point that has some missing values to solve for the distribution of the missing data, use your model equation, provided with your current guess of the model parameters and given the observed data. **Please note that you are solving for a distribution for each missing value and not for the expected value. Also note that now we have available with us a distribution for each missing value, we can calculate the expectation of the likelihood function with respect to the unobserved variables.** If our guess for the model parameter was correct, this expected likelihood will be the actual likelihood of our observed data. If the parameters were not correct, it will just be a lower bound.

The M-step: Now that we have got an expected likelihood function with no unobserved variables in it, maximize the function as you would in the fully observed case, to get a new estimate of your model parameters.

Repeat till convergence.

Q30. What do you mean by curse of dimensionality?

Ans. The curse of dimensionality refers to the tendency of a state space to grow exponentially in its dimension *i.e.* in the number of state variables.

Q31. What is H2O? What is prcomp?

Ans. H2O is software for machine learning and data analysis. PCA is named as **prcomp** in H2O and R system.

Q32. What is the principle of operation of the back-propagation algorithm?

Ans. Firstly, the inputs from the training data are propagated through the network to the output. Then the model outputs are compared with the actual outputs. The difference is the error. This error is then fed into the network backwards and connections weights are adjusted.

Q33. Define the following:

(a) **Learning rate.**
(b) **Momentum.**

Ans. (a) **Learning rate:** A higher value of learning rate will make the algorithm converge more rapidly towards the desired values but it will reduce the stability of the algorithm.

(b) **Momentum:** It is used to control oscillations in weights which could be caused by alternately signed error signals.

Q34. WEKA tool may be used to construct neural network. Can it do the parameter value selection also for neural networks?

Ans. A neural network trains by repeated passes of the data. Each pass is called an **epoch. With each epoch, the connecting weights are adjusted until a satisfactory level of performance is reached.** We can ponder that if 50 epochs are good then 80 epochs should be better. But this argument does not hold with neural networks. With too many passes the network may over-train also. **Please understand that a well trained neural network model would show similar performance levels with both test and training data. Also understand that keeping this in mind we can change the parameter values of the algorithm.** We should not assume that default parameters are the best. We can change the parameters too. As the number of epochs increases we need more time for training. If we compare the outputs from the two different epochs, we can see the accuracy remains at the same level of 97.2%. that is why we say that higher the number of epochs has not improved the model performance. **Just keep in mind that we should restrict ourselves to the lower number of epochs as it not only involves less computing time but also lowers the danger of over-fitting.**

Q35. Define the following terms:

(a) **Activation function.**
(b) **Back-propagation learning.**
(c) **Delta rule.**
(d) **Epoch.**
(e) **Feedforward neural networks.**
(f) **Genetic algorithm.**
(g) **Hyperplane.**
(h) **Kernel.**

Ans.

(a) **Activation function:** A function that describes the output behaviours of a neuron in neural network.

(b) **Back-propagation learning:** It is a form of supervised learning used to train mainly, feedforward neural networks, that works by making modifications in weights values, starting at the output layer and then moving backwards through the hidden layer.

(c) **Delta Rule:** The delta rule changes the weight vector for a neural network in such a way that the error is reduced.

(d) **Epoch:** It is one complete pass of the entire training set through the neural network. The learning process is repeated epoch-by-epoch until the averaged square error converges to a minimum value.

(e) **Feedforward neural network**: It is a sort of ANN where connections between the units do not form a directed cycle.

(f) **Genetic algorithm:** It is an adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection theory.

(g) **Hyperplane:** A plane (as in geometry) that is used to do object separation during SVM implementation.

(h) **Kernel:** Linear, polynomial and rbf are some of the kernels of SVM that allows SVM to fit the maximum margin hyperplane in the transformed feature space.

Q36. Tabulate the differences between supervised and unsupervised learning.

Ans. The following are the points of differences:

Supervised Learning	Unsupervised Learning
1. It incorporates **an external teacher** so that each output unit is told what should be the desired response to that input signal.	1. Uses no **external teacher.**
2. **Global error signals** govern learning.	2. **Local information** is used for learning.
3. Uses pattern class information of each training pattern.	3. Adaptively clusters patterns to generate decision codes.
4. Paradigms are error correction learning, reinforcement learning and stochastic learning.	4. Paradigms are Hebbian learning and competitive learning.
5. Learning is usually **off line.** It means that learning and operation phases are different. In off-line learning, all of the given patterns are used together.	5. Learning is **online**. It means that the agent learns and operates at the same time. Here, the information in each new pattern is incorporated into the agent.
6. All learning data are stored and can be accessed repeatedly. So, we can see whether we are making progress in training or not.	6. Locality allows learning in **real time.** So, the progress cannot be monitored.
7. Inputs and outputs are defined.	7. Only inputs are defined.
8. Uses pattern class information of each training pattern.	8. Adaptively clusters patterns to generate decision class without any prior pattern class information.
9. Global error signal governs learning.	9. Local information/ rules are used for learning.
10. Learning is offline usually.	10. Locality allows **synapses** to learn in real-time.

Q37. Explain different nets classes.

ns. It is put under 2 main categories as follows:

I. Unsupervised learning (*i.e.* without a teacher):

4. Feedback Nets:
   (a) Binary Adaptive Resonance Theory (ART1).
   (b) Analog adaptive Resonance Theory (ART2).

(c) Discrete Hopfield (DH).
(d) Continuous Hopfield (CH).
(e) Discrete Bi-directional Associative Memory (BAM).
(f) Temporal Associative Memory (TAM).
(g) Adaptive Bi-directional Associative Memory (ABAM).
(h) Kohonen Self-Organizing Map/ Topology Preserving Map (SOM/TPM).
(i) Competitive learning.

5. Feed-forward only Nets:
   (a) Learning Matrix (LM).
   (b) Driver-reinforcement Learning (DR).
   (c) Counter Propagation (CPN).

II. Supervised learning (*i.e.* with a teacher):

1. Feedback Nets:
   (a) Boltzmann Machine. (BM).
   (b) Mean Field Annealing (MFT).
   (c) Recurrent Cascade Correlation (RCC).
   (d) Learning Vector Quantization (LVQ).
   (e) Back Propagation through Time (BPTT).
   (f) Real-time Recurrent Learning (RTRL).
2. Feed-forward only Nets:
   (a) Perceptron.
   (b) Adaline, Madaline.
   (c) Backpropagation (BP).
   (d) Cauchy Machine (CM).
   (e) Artmap.
   (f) Cascade Correlation (CasCor).

Q38. Differentiate between clustering and classification.

Ans.

Clustering	Classification
(1) This function maps the data into one of the several clusters which is the **grouping of data** items based on the similarities between them.	(1) This model function **classifies the data** into one of the several predefined categorical classes.
(2) Involved in **unsupervised learning**.	(2) Involved in **Supervised learning**.
(3) Training sample is not provided.	(3) Training sample is provided.
(4) Number of clusters is **not** known before clustering. These are identified after the completion of clustering.	(4) Number of classes is **known** before classification as there is predefined output based on input data.
(5) Data is not labeled.	(5) Labeled data points.
(6) Unknown number of classes.	(6) Known number of classes.
(7) Used to understand data.	(7) Used to classify future observations.

4.1 Perceptron

According to Minsky et al. a perceptron is defined as **"A machine which learns from examples, to assign input samples (vectors) to different classes using a linear function of inputs".** Simply stated, it is a neural network used for pattern classification. It consists of a single neuron with adjustable synaptic weights. It is also possible to expand the output layer of perceptron to include more than one neuron. So, they are also called as **weighted networks.** There are two types of perceptions as follows:

(a) Single layer perception.

(b) Multi layer perception.

Single layer perception: A single layer perceptron is shown in figure 2.8. It consists up of an input layer and an output layer with **adjustable weights and a bias.** It is used for classification of patterns that are linearly separable. The perception algorithm converges and positions the **decision surface** in the form of a hyper plane between the two classes. The limitation of this network is that it performs pattern classification with only two classes which are linearly separable. **Please understand that the linearity and integrity learning makes the single layer perception network very simple. Also understand that the training in the perception continues till no error occurs.**

In this network, only the weights between the **associator** and the **response unit** are adjusted. The input layer consists of input neurons from $x_1, x_2, ..., x_n$. **There always exists a common bias** "1". The output is compared with this target, if any difference occurs, update weights else stop training process.

In machine learning, the perceptron is an algorithm for supervised learning of binary classifiers *i.e.* functions that can decide whether an input represented by a vector of numbers, belongs to some specified class or not.

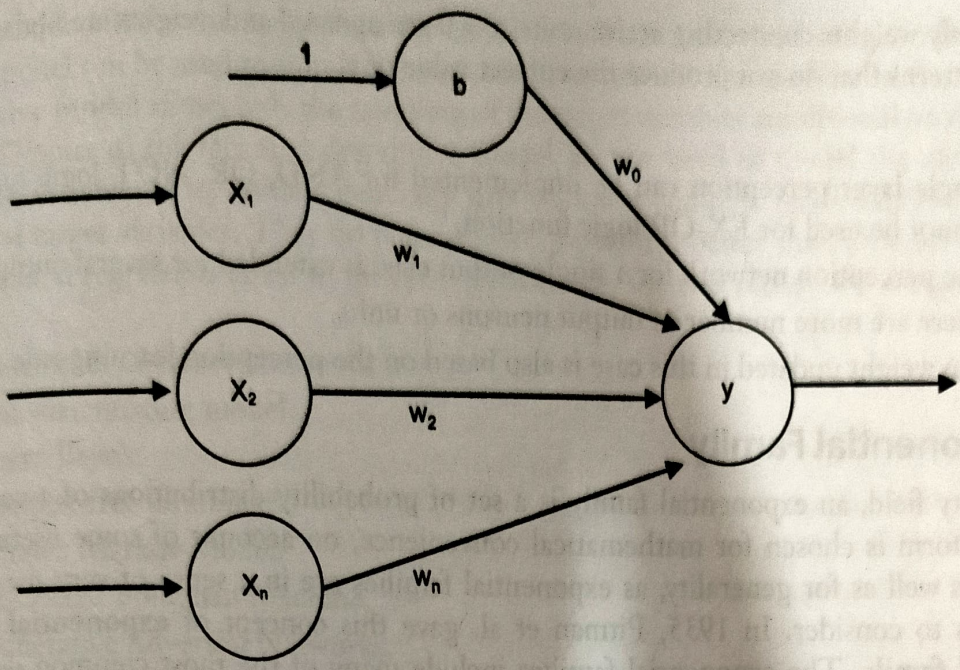


Fig.4.2: Single Layer Perceptron

The Training Algorithm

Step-1: Initialize weights and bias to zero and set learning rate, $\alpha = 0$ to 1.

Step-2: While stopping condition is false do step-3 to step-7.

Step-3: For each training pair s: t do step-4 to step-6.

Step-4: Set activation of input units-

$$x_i = s_j \text{ for } i = 1 \text{ to } n.$$

Step-5: Determine the output unit response-

$$y_{in} = b + \Sigma\,(i = 1 \text{ to } n)\, x_i\, w_i$$

Where the activation function is-

$$y = f(y_{in}) = \begin{cases} = 1, \text{if } y_{in} \geq \theta \\ = 0, \text{if } y_{in} = -\theta \text{ to } \theta \\ = -1, \text{if } y_{in} < -\theta \end{cases}$$

Step-6: Observe the match between the target and the output response. If they do not match, update the weights and bias. So,

if $t \neq y$ and $x_i \neq 0$

$$w_{i\,(new)} = w_{i\,(old)} + \alpha\, t x_i$$
$$b_{(new)} = b_{(old)} + \alpha_t$$

else

$$w_{i\,(new)} = w_{i\,(old)}$$
$$b_{(new)} = b_{(old)}$$

Step-7: Test for stopping condition based on weight changes.

Step-8: Only weights connecting active units, $x_i \neq 0$ are updated and weights are updated only for patterns that do not produce the correct value of y.

Notes:

1. Single layer perception can be implemented for AND, OR, NOT logic functions but cannot be used for EX-OR logic function.
2. The perception network for a single output class is extended for several output classes.
3. There are more number of output neurons or units.
4. The weight updated in this case is also based on the perception learning rule.

4.5 Naive Bayes

It is a supervised machine learning algorithm. It is often associated with NLP (Natural Language Processing) applications like spam recognition or sentiment analysis.

By the term **'naive'** we mean **"immature".** A naive Bayes classifier is a simple probability-based algorithm. It uses Bayes theorem. It assumes that instances are independent of each other. They work very well in complex real world scenarios also. The naive Bayes algorithm offers a fast model building and scoring both for binary and multiclass situations for relatively low volumes of data. According to Hand et al., 2001, this algorithm makes predictions using Bayes theorem which includes evidence or prior knowledge in its prediction.

Tool like WEKA also uses this naive Bayes algorithm on the dataset provided to it. In WEKA, there is an option of naive Bayes classifier under **'bayes' subdirectory.** Then in next step we keep the **default settings** of the Weka environment and the classifiers parameters. Next, we click on the Start button and the program executes and we will get the needed analysis. Thus, this model is used for analysis of the input dataset. Naive Bayes is a simple technique for constructing classifiers-models that assign class labels to problem instances that are represented as vectors of feature values. The class labels are drawn from some finite set.

Principle: "All Naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable."

For example, a rectangle may be considered to be a whiteboard if it has a length, breadt and some thickness. A **Naive Bayes classifier** considers each of these features to contribute independently to the probability that this rectangle is a whiteboard, regardless of any possible correlations between the color, its shape and its thickness.

Note that for some types of probability models, Naive Bayes classifiers can be trained very efficiently in a supervised learning setting. Practically, parameter estimation for **Naive Bayes models** uses the method of maximum likelihood. In other words, one can work with **Naive Bayes model** without accepting Bayesian methods.the advantage of this is that it requires only a small number of training data to estimate the parameters necessary for classification. As far as machine learning is concerned, Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Baye's theorem with strong independence assumptions between the features.

Naive Bayes classifiers have been studied extensively since 1950s. It forms the baseline for text catagorization. With suitable pre-processing, it is competitive in this domain with more advanced methods including support vector machines. Also it has some applications in automatic medical diagnosis. **Naive Bayes classifiers are highly scalable,** requiring a number of parameters linear in the number of variables (features/ predictors) in a learning problem. **Also note that maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time rather than by expensive iterative approximation as used for many other types of classifiers.**

Advantages of Naive Bayes Algorithm

1. The algorithm as well as its implementation process is simple. Calculations do not involve any numerical optimization, no matrix algebra and no calculus.
2. The algorithm is efficient to train and use. It can be easily updated with new data. This algorithm is similar to linear classifier and hence is fast too.

3. From one dataset we can estimate content features from messages without headers.
4. Unlabelled data can be used in parameter estimation for other applications. This algorithm performs well with smaller training sets.

Disadvantages of Naive Bayes Algorithm

1. The independence assumption is a very strong inference and untrue for most real world problems.
2. The boundaries between classes have to be fine tuned and not to be set analytically.
3. It is not very effective algorithm as it is simple.

Applications of Naive Bayes Algorithm

1. The classifiers used in text classification have higher success rate as compared to other algorithms. As a result, it is widely used in spam filtering *i.e.* identifying spam e-mail and sentiment analysis *i.e.* in social media analysis to identify positive and negative customer sentiments.
2. Naive Bayes classifier and collaborative filtering together builds a recommendation system that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not.
3. It is an eager learning classifier and is thus fast. It could be used for making predictions in real-time too.
4. It can be used to do parameter estimation for any model application.
5. It can be used for document classification also.
6. In medical diagnosis field, it has been used for localization of primary tumour, thyroid diseases etc.

4.6 Support Vector Machines (SVM)

SVM belongs to the type of **supervised learning** methods used for **classification and regression. SVM were developed by Vladimir Vapnik et al. in 1998 at UK.** Statistical learning became important after its introduction. They are different from Neural Networks.

Table-4.1: differences between SVM & NN.

	Support Vector Machine (SVM)		Neural Networks (NN)
1	Kernel maps to a very **high dimensional space.**	1	Hidden layers map to **lower dimensional spaces.**
2	Search space has a **unique minimum.**	2	Search space has **multiple local minima.**
3	**Kernel** and **cost** are the two parameters to select.	3	Requires a number of **hidden units** and **layers.**
4	**Training** is extremely efficient.	4	**Training** is expensive.
5	**High accuracy** in typical domains.	5	**Good accuracy** in typical domains.

Basic idea: Construct a hyper plane that best separates classes. If this is not feasible in the input space then the goal is achieved in a higher dimensional space called as a **feature space,** reached by mapping the input vector through a non-linear function.

This mapping has a property that will linearly separate the data sets in the feature space. The hyper plane in the feature space will maximally separate the data set classes. The method is based on **statistical learning theory** and is successfully used in a number of applications like particle identification, face detection, text categorization. Technically speaking, it is mathematically intuitive, reproducible and does not have problems of **local minima** as the optimization function is **convex** and is robust to new classification data tasks.

In machine learning, SVMs or Support Vector Networks are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic **binary linear classifier.** An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. Then new examples are mapped into that same space and predicted to belong to a category based on which side of the gap they fall. SVMs can also perform a non-linear classification using what is called the kernel trick *i.e.* implicitly mapping their inputs into high dimensional feature spaces. When data are not labelled, supervised learning is not possible and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups and then map new data to these formed groups. The clustering algorithm which provides an improvement to SVM is called **support vector clustering.** It is often used in industrial applications either when data are not labelled or when only some data are labelled as a pre-processing for a classification pass.

This is shown in figure 4.3.

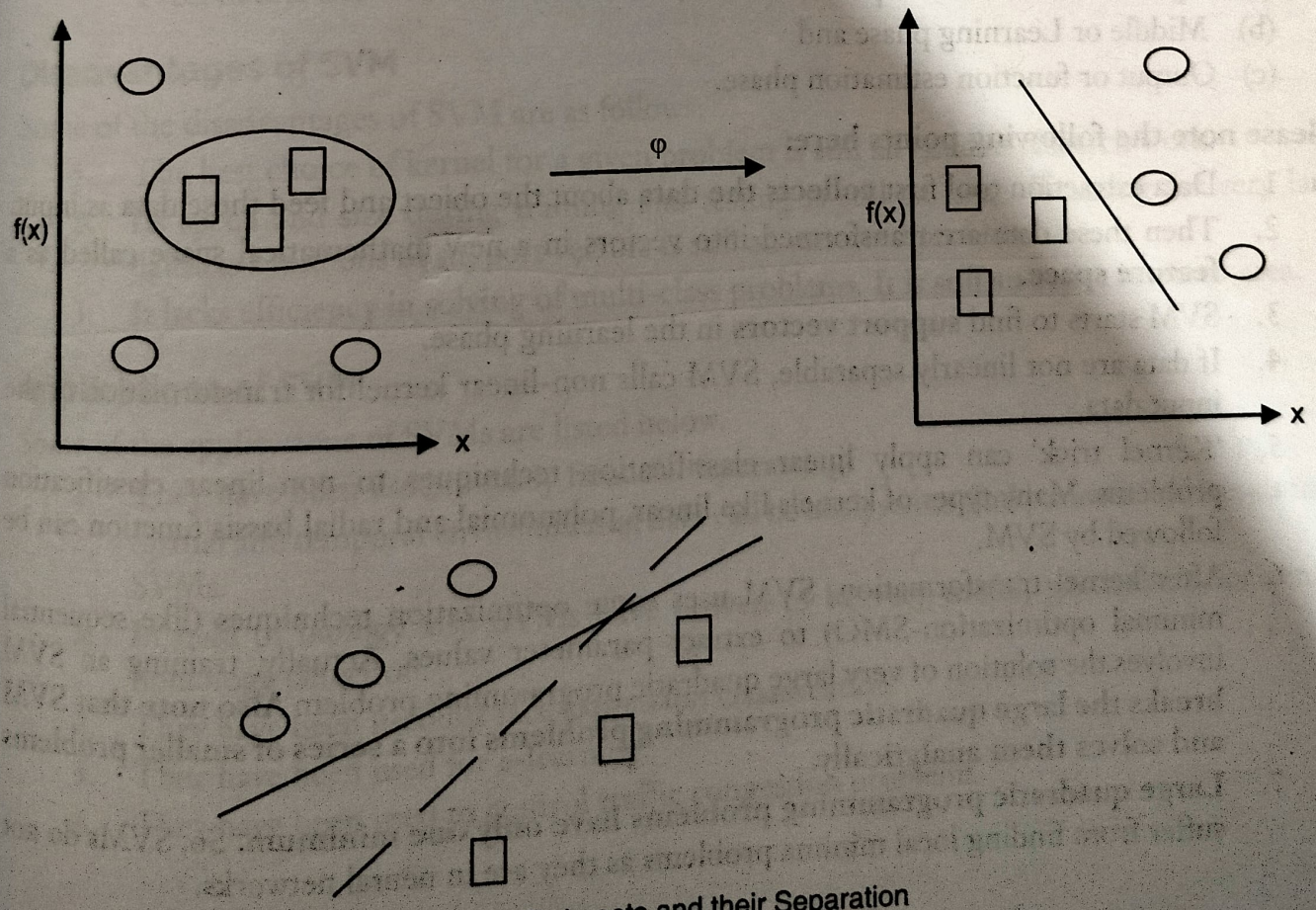


Fig.4.3: Data sets and their Separation

The upper part of the figure 4.3 shows the mapping of the training data non-linearly into a higher dimensional feature space and constructs a hyperplane with maximum margins. The lower part of the figure 4.3 shows two potential hyperplanes that separate data sets. A hyperplane is a concept in geometry. **Please note that hyperplane is a generalization of the concept of a plane which is used in SVM implementation for object separation. Also note that a hyperplane that separates the data points in the feature space with maximumdistances among the classes in SVM is called as a maximum-margin hyperplane or optimal hyperplane.** The data points on the outer margins/ outside ovals and the inner margins/ inside ovals that are on the boundary lines are called as **support vectors.**

Support vectors or SVs are transformed training patterns that are always placed at the optimal hyperplane in the feature space during SVM LEARNING. SVM is a statistical based learning algorithm that has been widely used by researchers in various fields like business, text categorization, pattern recognition abd protein function prediction. Recently, researchers added a new dimension to SVM in cancer classification through its ability to deal with high dimensional data. SVM can handle any classification, clustering, regression and even novelty detection problems. The attractiveness of SVM algorithm lies in the fact that we do not have to consider all instances *i.e.* we can focus on the support vectors to do the analysis.

Algorithm: SVM

SVM architecture is analogous to neural network architecture. Like we have three layers of neural networks, similarly we have three phases of SVM-

(a) Input or transformation phase,
(b) Middle or Learning phase and
(c) Output or function estimation phase.

Please note the following points here:

1. Data extraction tool first collects the data about the object and feed these data as input.
2. Then these data are transformed into vectors in a new mathematical space called as a feature space.
3. SVM starts to find support vectors in the learning phase.
4. If data are not linearly separable, SVM calls non-linear kernel for transformation of the input data.
5. 'Kernel trick' can apply linear classification techniques to non-linear classification problems. Many types of kernels like linear, polynomial and radial bassis function can be followed by SVM.
6. After kernel transformation, SVM uses some optimization techniques (like sequential minimal optimization-SMO) to extract parameter values. Actually, training an SVM involves the solution of very large quadratic programming problem Also note that SVM breaks the large quadratic programming problems into a series of smaller problems and solves them analytically.
7. Large quadratic programming problems have only one minimum. So, SVMs do not suffer from finding local minima problems as they are in neural networks.

8. **The amount of memory required for SMO is linear with the training set size, which allows it to handle very larger training sets also.**
9. The time taken by SMO depends on the SVM's evaluation only. So, SMO is faster for linear SVMs and sparse datasets.
10. During optimization, SVM will try to find the support vectors for prediction.
11. SVM model can be constructed through WEKA tool. **Please note that SVM's default kernel is polynomial in WEKA.**

Advantages of SVM

Some of the advantages of SVM are as follows:

1. They can not only solve binary class pattern recognition problems but also for multiclass classification, regression estimation, feature selection etc.
2. Lesser chances of information loss as SVM kernel considers the dot product of the feature vectors in high dimensional space to construct the optimal hyperplane rather than clustering or interpolating in input space. SVM can handle a large number of attributes of a data set unlike neural nets **as these traditional algorithms need reduction of dimensionality** *i.e.* **the number of attributes, before they can be applied to a high dimensional problem.**
3. Even in the field of bioinformatics, it has proven to be useful as there we have more than 20,000 attributes.
4. They are very versatile as they have many extensions too like Support Vector Decision Tree, neural network based SVM, fuzzy SVM etc.

Disadvantages of SVM

Some of the disadvantages of SVM are as follows:

1. The best choice of kernel for a given problem is still an issue in research work.
2. Its speed and size during training and testing is still an issue. Training for very large datasets / millions of support vectors is still an unsolved problem.
3. It lacks efficiency in solving of multi-class problems. It is still an active research area.

Applications of SVM

Some of the applications of SVMs are listed below.

1. They have been used to study financial forecasting.
2. Spatial and temporal environmental data can be analysed to find patterns and trends using SVMs.
3. In field of biology also SVMs have proven to be very useful, particularly for remote homology detection.
4. They have been used for face expression classification.
5. They have been used for e-learning.
6. They have been used to control traffic congestion problems.

4.7 Kernels

A kernel is a similarity function. It is a function that a domain expert provides to a machine learning algorithm. It takes two inputs and splits out how similar they are.

For instance, the task is to learn to classify images. We have (image, label) pairs as training data.

Consider a typical machine learning pipeline. Take your images, compute features, string these features for each image into a vector and then feed these **features-vectors and labels into a learning algorithm.** That is,

$$\text{Data} \rightarrow \text{Features} \rightarrow \text{Learning Algorithm}$$

Kernels offer an alternative. Instead of defining a slew of features, you define a single kernel function to calculate similarity between images. You provide this kernel, together with the images and labels to the learning algorithm and outcome is a classifier. **Please note that the standard SVM or logistic regression or perceptron formulation does not work with kernels.** But it works with feature vectors:

1. Under some conditions, every kernel function can be expressed as a dot product in a possibly infinite dimensional feature space (Mercer's theorem).
2. Many machine learning algorithms can be expressed entirely in terms of dot products.

Also note that this means that we can now take my best machine learning algorithm, express it in terms of dot product and then since my kernel is also a dot product in some space, replace the dot product by your suitable and best kernel.

Importance of Kernels

Kernels are opposed to feature vectors. It has been found that computing the kernel is easy but computing the feature vector corresponding to the kernel is quite difficult. The feature vector for even simple kernels can blow up in size and for kernels like RBF kernel the corresponding feature vector is infinite dimensional. Yet computing the kernel is almost trivial. Many machine learning algorithms can be written to only use dot products and then we can replace the dot products with kernels. By doing this, we do not have to use the feature vector at all. **Please understand that this means that we can work with highly complex, efficient-to-compute and yet high performing kernels without ever having to write down the huge and potentially infinite dimensional feature vector.** Thus, if not for the ability to use kernel functions directly, we would be stuck with relatively low dimensional, low performance feature vectors. **This trick is also called as the kernel trick.**

Outlier's Role

During data pre-processing step, we need to check whether the given attributes contain any outliers (or insignificant values) or NOT.

Suppose you have the **weights of 5 children** as 15, 12, 13, 10 and 35 kg. Now, if you try to find out the **average of these values**, the **answer is 17kg**. If you look at the given weight range carefully, then you will observe that the **last observation** is out of the normal range as compared to the other observations. Now, let us **remove the last observations**. *i.e.*, weight of 35kg and **recompute the average of other** observations. Then we get **average (new) of 12.5kg. Please note that this new average is more meaningful as compared to the last average value.** Here, 35kg was an out lier value to detect that impact accuracy. In similar fashion, we replace the value of data points in more meaningful way.

(a) Factor Analysis.
(b) ICA.
(c) PCA.
(d) Singular Value Composition.

Answers

1. a 2. b 3. b 4. b 5. b
6. a 7. b 8. a 9. b 10. a
11. a 12. a 13. a

Conceptual Short Questions with Answers

Q1. What are the differences between data mining and machine learning?

Ans. **Data mining** is the process in which the unstructured data puts efforts to extract knowledge or unfamiliar interesting patterns. On the other hand, **machine learning** relates to the study, design and development of the algorithms that give systems the ability to learn without being clearly programmed.

Q2. Give functions supervised and unsupervised learning.

Ans. **Supervised learning** involves-

(a) Classifications.
(b) Speech recognition.
(c) Regression.
(d) Predict time series.
(e) Annotate strings.

On the other hand, unsupervised learning involves-

(a) Finding the clusters of data.
(b) Finding low dimensional representations of data.
(c) Finding interesting coordinates and correlations.

Finding novel observations or cleaning databases.

Q3. What is inductive machine learning?

Ans. **Inductive machine learning covers the process of learning by examples** where a system, from a set of perceived instances, tries to encourage a general rule.

Q4. List at least five common machine learning algorithms.

Ans. Some of the popular ML algorithms are as follows:

(a) Decision Trees.
(b) Neural Networks.
(c) Probabilistic Networks.
(d) Nearest Neighbour.

Support Vector Machines (SVM).

Q5. Name different algorithm techniques in ML.

Ans. Different types of techniques in ML are as follows:

(a) Supervised learning.
(b) Unsupervised learning.
(c) Semi-supervised learning.
(d) Reinforcement learning.
(e) Transduction.

Learning to learn.

Q6. Name three stages to build the models (or hypotheses) in machine learning.

Ans. Three stages to build the models in machine learning (ML) are as follows:

(a) Model building.
(b) Model testing.

Applying the model.

Q7. What is the standard approach to supervised learning?

Ans. The standard approach to supervised learning is to break the data set of example into training set and the test set.

Q8. Differentiate between 'training set' and 'test set'.

Ans. In different fields of data science like machine learning, a set of data is used to determine the potentially predictive relationship identified as **training set.** Training set is the example provided to the learner while test set is used to test the exactness of the hypotheses produced by the learner and it is the set of illustration weighed down from the learner. So, training sets are different from test sets.

Q9. What are the different approaches of machine learning?

Ans. Different approaches of machine learning are as follows:

(a) Concept versus Classification Learning.
(b) Symbolic versus Statistical Learning.

Inductive versus Analytical Learning.

Q10. Define Overfitting during machine learning. Why it happens?

Ans. In ML, overfitting occurs when a statistical model defines a random error or noise rather than underlying relationship. When a model is extremely complex, overfitting is usually observed because of having too many parameters relating to the number of training data types. The model exihibits poor performance which has to be overfit. The risk of overfitting exists as the standards used for training the model is not the same as the standards used to judge the effectiveness of a model.

Q11. How can overfitting be avoided?

Ans. By means of loads of data, overfitting can be avoided. **Overfitting happens when** comparatively you have a small dataset and you make effort to learn from it. But if you

have a small database and it is necessary to come with a model based on that then in such a scenario, **cross validation** method may be used. In this method, the dataset is divided into two segments, testing and training datasets. The testing dataset will only test the model while in training dataset, the data-points will come up with a model. In this method, a model is generally provided with a dataset of known data on which training data set is run and a dataset of unknown data against which the model is tested. **The basic idea of cross validation is to define a dataset to test the model in the training phase.**

Q12. What is algorithm independent machine learning?

Ans. A type of machine learning in which mathematical foundations is independent of any particular classifier or learning algorithm is known as **algorithm independent machine learning.**

Q13. Distinguish between AI and machine learning.

Ans. Designing and developing algorithms according to the behaviours based on empirical data is known as machine learning. On the other hand, artificial intelligence, in addition to machine learning, also covers other aspects like knowledge representation, natural language processing (NLP), planning, robotics etc.

Q14. Define classifier as used in machine learning.

Ans. A classifier is a system that inputs a vector of discrete or continuous feature vales and outputs a single discrete value, the class.

Q15. Where is pattern recognition used?

Ans. It is used in computer vision, speech recognition, data mining, statistics, information retrieval and bio-informatics.

Q16. What is model selection in ML?

Ans. The process of selecting the models from different mathematical models that refer to the same data set is called as model selection. It is applied to the fields of statistics, machine learning and data mining.

Q17. Name the method that is frequently used to prevent overfitting.

Ans. **'Isotonic Regression'** is used to prevent overfiting when the data is sufficient.

Q18. Why instance-based learning algorithm is sometimes also referred to as lazy-learning algorithm?

Ans. Instance-based learning algorithm is sometimes also referred to as lazy-learning algorithm because they delay the induction or generalization process until classification is performed.

Q19. What are the 2 classification methods that SVM handle?

Ans. The two classification methods that SVM can handle are as follows:

(a) Combining binary classifiers.

Modifying binary to incorporate multiclass learning.

Q20. For what purpose PCA and ICA are used?

Ans. PCA and ICA are used to extract features of data to reduce dimensions of data. They are important feature extraction techniques used for dimensionality reduction.

Q21. What do you mean by dimension reduction?

Ans. **It is the process of reducing the number of random variables under considerations and can be divided into feature selection and feature extraction.**

Q22. What are SVMs?

Ans. **SVMs or Support Vector Machines are supervised learning algorithms that are used for classification and regression analysis.**

Q23. What are the different methods to solve sequential supervised learning problems?

Ans. The different methods to solve sequential supervised learning problems are:

(a) Sliding window methods.
(b) Recurrent sliding windows.
(c) Hidden Makov Models.
(d) Maximum entropy Markov Models.
(e) Conditional random fields.

Graph transformer networks.

Q24. Name two techniques of ML. Give a very popular application of ML that you see everyday.

Ans. Genetic programming and inductive learning are the two methods of ML.

The recommendation engine implemented by major ecommerce websites uses machine learning (ML).

Q25. How is KNN different from k-means clustering?

Ans. k-nearest neighbours is a supervised classification algorithm. On the other hand, k-means clustering is an unsupervised clustering algorithm. In order for k-nearest neighbours to work, labelled data is needed that we want to classify an unlabeled point. In case of k-means clustering only a set of unlabelled points and a threshold is required. The algorithm will take unlabelled points in consideration and slowly learn how to cluster them into groups by computing the mean of the distance between different points. In sutshell, KNN needs labelled points and is thus supervised learning while k-means doesn't need labelled points and is thus unsupervised learning.

Q26. Why is "Naive" Bayes naive?

Ans. Even though it has practical applications, specially in text mining, Naive Bayes is still considered as Naive because the assumption that it makes becomes virtually impossible to be seen in real-life data. The conditional probability is computed as purely the product of the individual probabilities of different components. This denotes the absolute independence of features *i.e.* a condition that is probably never met in real life.

Q27. Differentiate between a generative and discriminative model.

Ans. A generative model learns categories of data while a discriminative model merely learns the distinction between different categories of data. Discriminative models will leave behind generative models on the basis of classification tasks.

Q28. Name at least three methods of reducing dimensionality.

Ans. The three methods of reducing dimensionality are as follows:

(a) Removing collinear features.

(b) Performing PCA, ICA or other forms of algorithmic dimensionality reduction.

Combining features with feature engineering.

Q29. What is the EM technique? How this algorithm works?

Ans. EM is an algorithm for maximizing a likelihood function when some of the variables in the model are unobserved *i.e.* when there are latent variables with you. To solve for our model parameters you need to know **the distribution of your unobserved data** but the **distribution of your unobserved data is a function of your model parameters.** EEM tries to get around this by guessing repeatedly about the distribution for the unobserved data then further estimating the model parameters by maximizing something that is lower bound on the actual likelihood function and repeating until convergence.

The EM algorithm starts with a guess for values of your model parameters as follows:

The E-step: For each data point that has some missing values to solve for the distribution of the missing data, use your model equation, provided with your current guess of the model parameters and given the observed data. **Please note that you are solving for a distribution for each missing value and not for the expected value. Also note that now we have available with us a distribution for each missing value, we can calculate the expectation of the likelihood function with respect to the unobserved variables.** If our guess for the model parameter was correct, this expected likelihood will be the actual likelihood of our observed data. If the parameters were not correct, it will just be a lower bound.

The M-step: Now that we have got an expected likelihood function with no unobserved variables in it, maximize the function as you would in the fully observed case, to get a new estimate of your model parameters.

Repeat till convergence.

Q30. What do you mean by curse of dimensionality?

Ans. The curse of dimensionality refers to the tendency of a state space to grow exponentially in its dimension *i.e.* in the number of state variables.

Q31. What is H2O? What is prcomp?

Ans. H2O is software for machine learning and data analysis. PCA is named as **prcomp** in H2O and R system.

Q32. What is the principle of operation of the back-propagation algorithm?

Ans. Firstly, the inputs from the training data are propagated through the network to the output. Then the model outputs are compared with the actual outputs. The difference is the error. This error is then fed into the network backwards and connections weights are adjusted.

Q33. Define the following:

(a) **Learning rate.**
(b) **Momentum.**

Ans. (a) **Learning rate:** A higher value of learning rate will make the algorithm converge more rapidly towards the desired values but it will reduce the stability of the algorithm.

(b) **Momentum:** It is used to control oscillations in weights which could be caused by alternately signed error signals.

Q34. WEKA tool may be used to construct neural network. Can it do the parameter value selection also for neural networks?

Ans. A neural network trains by repeated passes of the data. Each pass is called an **epoch. With each epoch, the connecting weights are adjusted until a satisfactory level of performance is reached.** We can ponder that if 50 epochs are good then 80 epochs should be better. But this argument does not hold with neural networks. With too many passes the network may over-train also. **Please understand that a well trained neural network model would show similar performance levels with both test and training data. Also understand that keeping this in mind we can change the parameter values of the algorithm.** We should not assume that default parameters are the best. We can change the parameters too. As the number of epochs increases we need more time for training. If we compare the outputs from the two different epochs, we can see the accuracy remains at the same level of 97.2%. that is why we say that higher the number of epochs has not improved the model performance. **Just keep in mind that we should restrict ourselves to the lower number of epochs as it not only involves less computing time but also lowers the danger of over-fitting.**

Q35. Define the following terms:

(a) **Activation function.**
(b) **Back-propagation learning.**
(c) **Delta rule.**
(d) **Epoch.**
(e) **Feedforward neural networks.**
(f) **Genetic algorithm.**
(g) **Hyperplane.**
(h) **Kernel.**

Ans.

(a) **Activation function:** A function that describes the output behaviours of a neuron in neural network.

(b) **Back-propagation learning:** It is a form of supervised learning used to train mainly, feedforward neural networks, that works by making modifications in weights values, starting at the output layer and then moving backwards through the hidden layer.

(c) **Delta Rule:** The delta rule changes the weight vector for a neural network in such a way that the error is reduced.

(d) **Epoch:** It is one complete pass of the entire training set through the neural network. The learning process is repeated epoch-by-epoch until the averaged square error converges to a minimum value.

(e) **Feedforward neural network:** It is a sort of ANN where connections between the units do not form a directed cycle.

(f) **Genetic algorithm:** It is an adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection theory.

(g) **Hyperplane:** A plane (as in geometry) that is used to do object separation during SVM implementation.

(h) **Kernel:** Linear, polynomial and rbf are some of the kernels of SVM that allows SVM to fit the maximum margin hyperplane in the transformed feature space.

Q36. Tabulate the differences between supervised and unsupervised learning.

Ans. The following are the points of differences:

Supervised Learning	Unsupervised Learning
1. It incorporates **an external teacher** so that each output unit is told what should be the desired response to that input signal.	1. Uses no **external teacher.**
2. **Global error signals** govern learning.	2. **Local information** is used for learning.
3. Uses pattern class information of each training pattern.	3. Adaptively clusters patterns to generate decision codes.
4. Paradigms are error correction learning, reinforcement learning and stochastic learning.	4. Paradigms are Hebbian learning and competitive learning.
5. Learning is usually **off line.** It means that learning and operation phases are different. In off-line learning, all of the given patterns are used together.	5. Learning is **online**. It means that the agent learns and operates at the same time. Here, the information in each new pattern is incorporated into the agent.
6. All learning data are stored and can be accessed repeatedly. So, we can see whether we are making progress in training or not.	6. Locality allows learning in **real time.** So, the progress cannot be monitored.
7. Inputs and outputs are defined.	7. Only inputs are defined.
8. Uses pattern class information of each training pattern.	8. Adaptively clusters patterns to generate decision class without any prior pattern class information.
9. Global error signal governs learning.	9. Local information/ rules are used for learning.
10. Learning is offline usually.	10. Locality allows **synapses** to learn in real-time.

Q37. Explain different nets classes.

Ans. It is put under 2 main categories as follows:

I. Unsupervised learning (*i.e.* without a teacher):

1. Feedback Nets:
   (a) Binary Adaptive Resonance Theory (ART1).

(b) Analog adaptive Resonance Theory (ART2).
(c) Discrete Hopfield (DH).
(d) Continuous Hopfield (CH).
(e) Discrete Bi-directional Associative Memory (BAM).
(f) Temporal Associative Memory (TAM).
(g) Adaptive Bi-directional Associative Memory (ABAM).
(h) Kohonen Self-Organizing Map/ Topology Preserving Map (SOM/TPM).
(i) Competitive learning.

2. Feed-forward only Nets:
   (a) Learning Matrix (LM).
   (b) Driver-reinforcement Learning (DR).
   (c) Counter Propagation (CPN).

II. Supervised learning (*i.e.* with a teacher):

1. Feedback Nets:
   (a) Boltzmann Machine. (BM).
   (b) Mean Field Annealing (MFT).
   (c) Recurrent Cascade Correlation (RCC).
   (d) Learning Vector Quantization (LVQ).
   (e) Back Propagation through Time (BPTT).
   (f) Real-time Recurrent Learning (RTRL).
2. Feed-forward only Nets:
   (a) Perceptron.
   (b) Adaline, Madaline.
   (c) Backpropagation (BP).
   (d) Cauchy Machine (CM).
   (e) Artmap.
   (f) Cascade Correlation (CasCor).

Q38. Compare Bagging, Boosting and Stacking.

Ans. Comparison between Bagging, Boosting and Stacking:

S.No.	Features	Bagging	Booting	Stacking
1.	Partitioning of data into subsets.	Random	Giving misclassified samples higher priority.	Various.
2.	Goals	Minimize variance.	Increase predictive force.	Both.
3.	Methods where used.	Random subspace.	Gradient descent.	Blending.
4.	Function to combine single model.	Weighted average.	Weighted majority vote.	Logistic regression
5.	Misclassification rate.	Solve the over-fitting problem.	Reduce bias but increase over fitting.	Both.
6.	Training of data.	Parallel.	Sequential.	Both.

Q39. For a given data having 100 samples , if squared errors SE_1, SE_2 and SE_3 are 13.33, 3.33 and 4.00 respectively. Find out the Mean Squared Error (MSE). State the formula for MSE.

[Hint:

$$\because \text{Mean Squared Error (MSE)} \equiv \frac{\text{Squared Error}_1 + \text{Squared Error}_2 + \ldots + \text{Squared Error}_N}{\text{Number of data samples}}$$

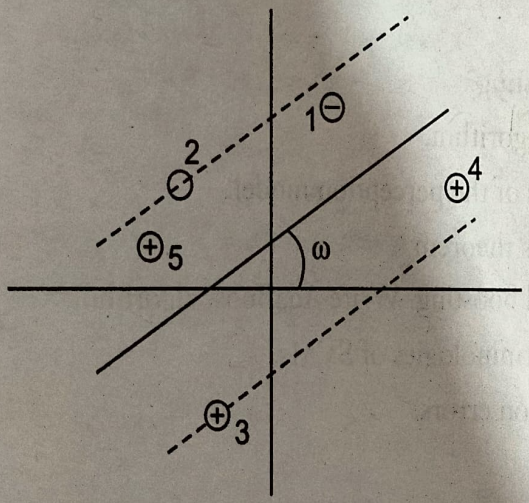
$$\text{MSE} = \frac{13.33 + 3.33 + 4.00}{100} = 0.2066\ ]$$

Q40. Compare SVM and NN.

Hint:

Support Vector Machine	Neural Network.
(1) Kernel maps to a very **high dimensional spaces.**	(1) Hidden layers map to **lower dimensional spaces.**
(2) Search space has a **unique minimum.**	(2) Search space has **multiple local minima.**
(3) Classification is difficult.	(3) Classification is extremely efficient.
(4) Very good accuracy in typical domains.	(4) Very good accuracy in typical domains only.

Q41. From the following diagram, identify which data points (1, 2, 3, 4, 5) are support vectors (if any), slack variables on correct side of classifier (if any) and slack variables on wrong side of classifier (if any). Mention which point will have maximum penalty any why?



Ans. Data points 1 and 5 will have **maximum penalty.** Margin (m) is the **gap** between data points and the classifier boundary. The **margin is the minimum distance of any sample to the decision boundary.** If this hyperplane is in the canonical form, the margin can be measured by the length of the weight vector.

Maximal margin classifier is a classifier in the family, F, that maximizes the margin. It is good as per intuition and PCA theory. It implies that only support vectors matter, as other training examples are ignorable.

If the **training set is not linearly separable** then slack variables can be added to allow misclassification of difficult or noisy examples, resulting margin called as **soft margin.** It allows

a few variables to cross into the margin or over the hyperplane, allowing misclassifications. **This is a trade-off between the hyperplane violations and the margin size.** The slack variables are bounded by some set cost. **Note that farther they are from the soft margin, the less influence they have on prediction.** All observations have an associated slack variable:

- Slack variable = 0 then all points on the margin.
- Slack variable > 0 then a point in the margin or on the wrong side of the hyperplane.
- C is the trade off between the slack variable penalty and the margin.

Exercise Questions

Q1. Explain in brief logistic regression.

Q2. What is ICA? Discuss.

Q3. Explain generative algorithm in detail.

Q4. Describe the limitations of the perceptron model.

Q5. Explain the decision tree algorithm with example.

Q6. Write a short note on Naive Baye's model of statistical learning.

Q7. What are SVMs? Discuss their characteristics.

Q8. What is
(a) Overfitting.
(b) Early Stopping.
(c) Curse of Dimensionality.
(d) Regularization.

Q9. Explain what is boosting?

Q10. Explain AdaBoost algorithm.

Q11. Describe limitations of the perception model.

Q12. Discuss Naive Baye's theorem.

Q13. Discuss bagging and boosting. Write AdaBoost algorithm.

Q14. What are the key terminologies of SVMs.

Q15. What are classification errors.

Q16. What is SVM.

Q17. Discuss the differences between bagging and boosting.

Q18. Write short note on "Naive Baye's" technique.

Q19. What are support vectors and margins? Also explain soft margin SVM.

Q20. (a) Explain Perceptron Training algorithm for Linear Classification.
(b) Explain prior probability, likelihood and marginal likelihood in context of Naive Baye's algorithm.

Q21. Explain ADA boost algorithm.

Q22. What are classification errors.

5 Unsupervised Learning and their Algorithms

5.0 Clustering

Unsupervised learning is put under two categories:

1. **Clustering:** In this method, the inherent groupings in the data are discovered. The organization of unlabeled data into similar groups known as **clusters. And the process of partitioning of a set of observations into subsets (or clusters) so that observation in the same cluster is similar is called as clustering.** So, a cluster consists of data items which are similar in comparison to other clusters which contain dissimilar items. **Please understand that the aim of unsupervised learning is to find clusters of similar inputs in the data without knowing that these data points belong to one class and those to different class.** Rather the algorithm has to uncover similarities for itself. **For example,** there is a collection of 30,000 essays on voting process. These essays can be automatically grouped into small number on the basis of word frequency, page count, word limit etc.
2. **Association:** In this method, we discover a set of rules that describes large portions of data. **For example,** a doctor observes same set of symptoms in several patients suffering from some disease. Then based on his experience the doctor can conclude that the patient is suffering from the same disease.

Dendrograms

1. The visual representation of the compound correlation data is known as a **dendrogram.**
2. The individual compounds are arranged along the bottom of the dendrogram. These are called as **leaf nodes.**

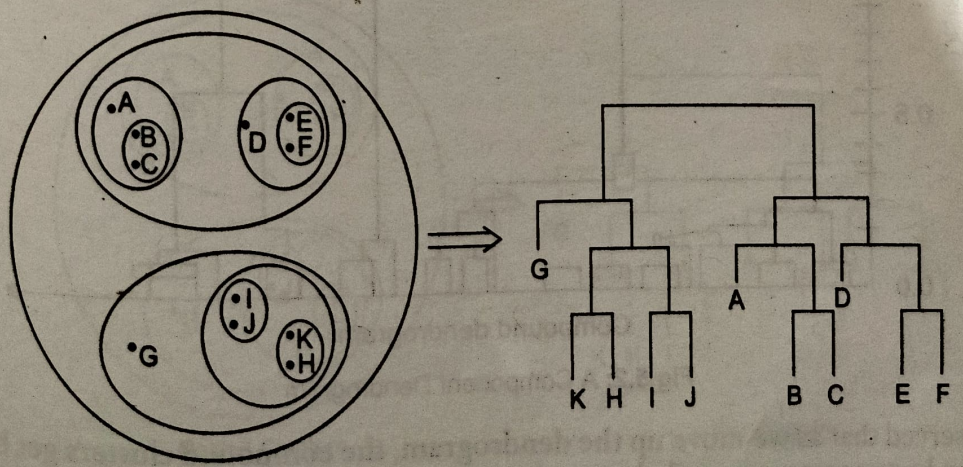


Fig.5.1: Dendogram (on right) shows nested clusters (on left)

3. **Compound clusters** are formed by joining individual compounds or existing compound clusters with the joint point named as a **node.**
   As shown in figure 5.1 above, at each dendogram node we have a right and a left sub-branch of clustered compounds. In the following discussions, compound clusters can refer to a single compound or a group of compounds.
4. **Vertical axis** is labelled distance and it refers to a **distance measure** between compounds or compound clusters.
5. The **height** of the node con be thought of as the distance value between the right and left sub-branch clusters.
6. The distance measure between two clusters is calculated by the formula :

$$D = 1 - C$$

   where D = Distance,
   C = correlation between compound clusters.
7. **Note that if compounds are correlated** , they will have a **correlation value close to 1.** So, D = 1 – C will have a value close to zero.
   ∴ highly correlated clusters are nearer to the bottom of the dendrogram. Compound clusters **that are not correlated** have a correlation value of zero and a corresponding distance value of 1.
8. **Compounds that are negatively correlated** *i.e.*, showing opposite abundance behavior, will have a correlation value of –1 and D = 1 – (–1) = 1 + 1 = 2

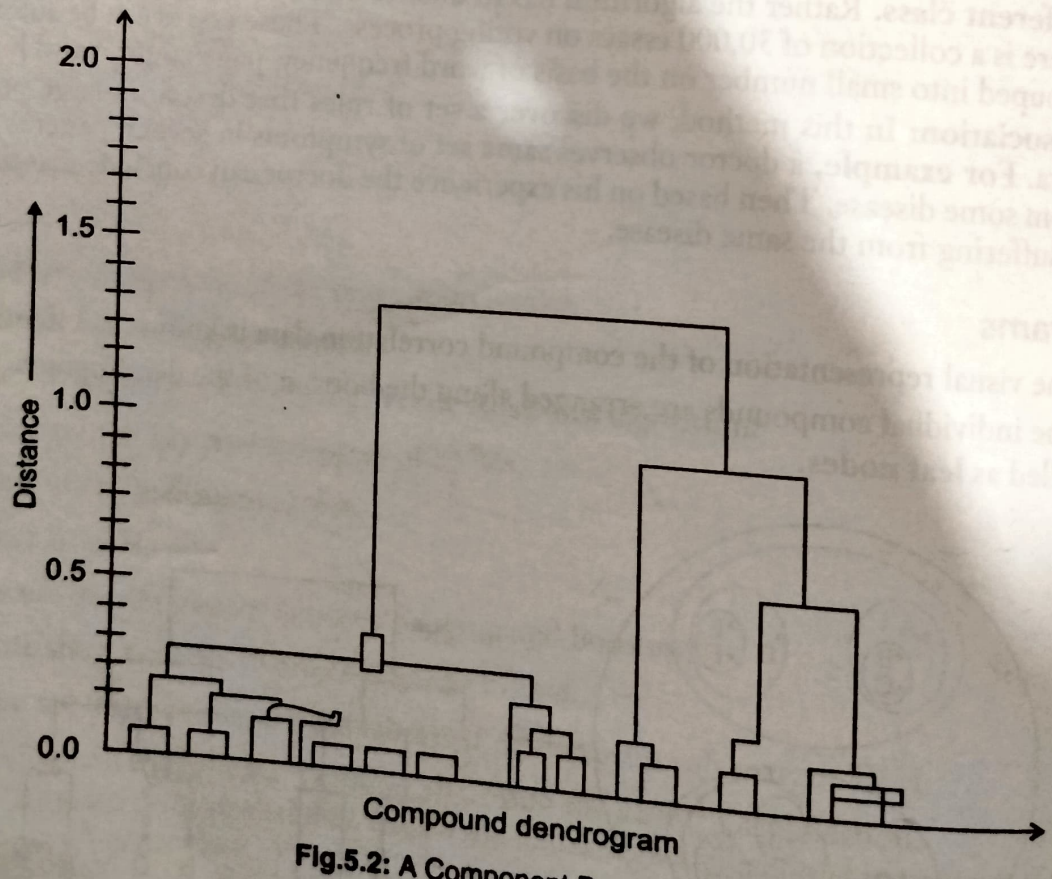


Fig.5.2: A Component Dendrogram

It is observed that as we move up the dendrogram, the compound clusters get bigger and the distance between compound clusters increases in value.

It is difficult to interpret distance between compound clusters when **compound clusters increase in size.**

Let us create a dataset and then compute the dendrogram.

```
#dendrogram
import numpy as np
import matplotlib.pyplot as ptt
from sklearn.datasets import make_blobs
from scipy.spatial.distance import pdist
from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram
#for reproducibility
np.random.seed(1000)
nb_samples = 25
#create the dataset
x,y = make_blobs (n_samples = nb_samples,
n_features = 2, centers =3,
cluster_std=1.5)
```

keep the number of samples very low to avoid complexity in the resulting plot.

Let us now compute the dendrogram. Firstly, we compute a distance matrix as follows :

```
# compute the distance matrix
Xdist = pdist (X , metric = 'euclidean')
# compute the linkage
X1 = linkage (xdist , method = 'ward')
```

Now it is possible to create and visualize a dendrogram -

```
Xd = dendrogram (X1)
```

A dendrogram shows how the clusters are **'merged'** iteratively (agglomerative clustering) or 'split' iteratively (divisive clustering) to the **optimal clustering** solution.

A visual representation of the compound correlation data is known as a **dendrogram.** The individual compounds that are arranged along the bottom of the dendrogram are known as **leaf nodes.** And the compound clusters that are formed by joining individual compounds or existing compound clusters with join point are called as a **node.**

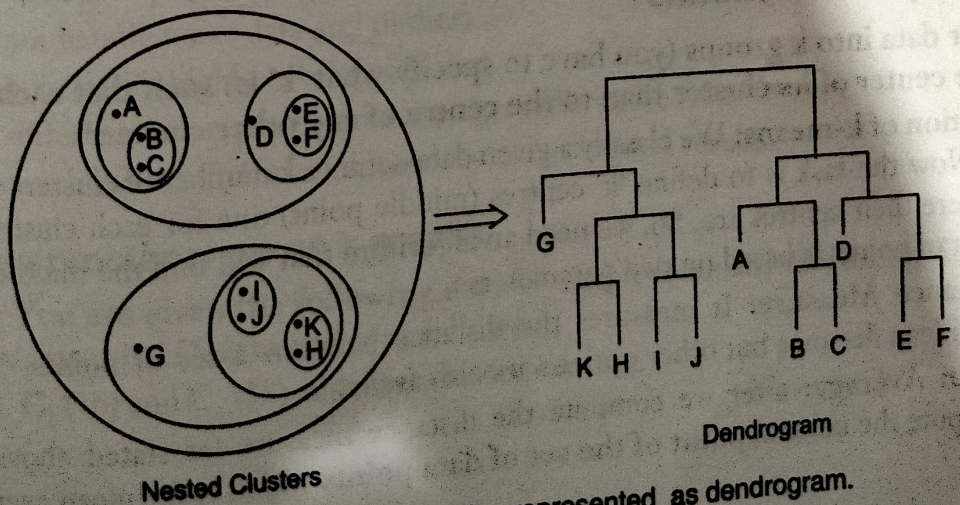


Fig.5.3: Nested Clusters being represented as dendrogram.

Vertical axis is a labeled distance and refers to a distance measure between compounds or compound clusters. The **height** of the node is the distance value between the right and left sub-branch clusters. The **distance measure** between two clusters is given by the formula:

$$D = 1 - C$$

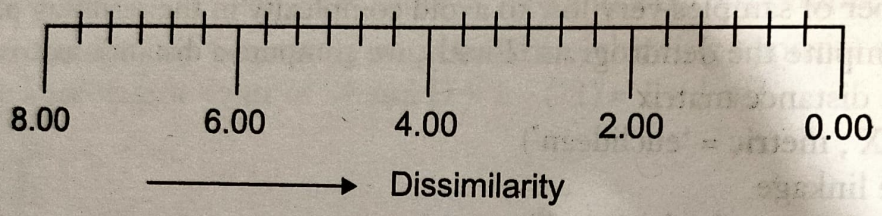
where D = distance

C = correlation between compound clusters.

Please note that if the compounds are highly correlated than they will have a correlation value close to 1 and so D = 1 – C will have a value close to compounds **zero.** That is why, highly correlated clusters are nearer to the bottom of the dendrogram. **Also note that the compound clusters that are not correlated have a correlation value of zero and a corresponding distance value of 1.** Compounds that are negatively correlated *i.e.*, showing opposite abundance behavior, will have a correlation value of –1 and D = 1 – (–1) = 1 + 1 = 2

As we move up the dendrogram, the compound clusters get bigger and the distance between compound clusters increases in value. It becomes difficult to interpret distance between compound clusters when compound clusters increase in size.

The **horizontal axis** on the dendrogram represents the distance or dissimilarity between clusters.



5.1 k-means Clustering

k-means is a simple unsupervised learning algorithm that solves the well known clustering problems. The working of k-means is similar to that of a real life situation in which we have observed that in a group of tourists with a few tourists guides who hold the umbrella up so that everybody can see them and follow them. On the other hand, in static k-means algorithm the data (*i.e.* the tourists) does not move rather only the tour guide (s) move. Analogous is the situation in k-means clustering.

Basic Principle of k-means-

"Divide your data into k groups (you have to specify what 'k' is) such that each data item is closer to the center of its cluster than to the centre of any other cluster."

Explanation of k-means: We classify a given data-set into k number of clusters which is fixed n advance. Now the task is to define 'k' centres (middle point) one for each cluster. But we do ot know where their centres are. So, we need an algorithm that will find the middle points. The entre of a set of points is based on two parameters:

(a) **Distance Measure:** It measures the distance between a set of points. We may use Euclidean distance but other options are also available.

(b) **Mean Average:** after we compute the distance measure (as stated above) we finally compute the **central point of the set of data points** which is the mean average.

Please note that this is valid in Euclidean space where everything is flat and easily measured. Now the need is to place cluster centres properly. We compute the mean of each cluster, μ_i (i) and position the cluster centre right there. **Also note that this is similar to minimizing the Euclidean distance** (*i.e.* sum-of-squares again) **from each data point to the centre of cluster.**

Now we would like to find out that which point belongs to which cluster. This is mandatory as it lets us know the position of cluster centres. It is crystal clear to associate closest point with the cluster centre. This might change as the algorithm iterates on. In the input space we start positioning the cluster centres randomly as we do not know where to put them exactly and further update their positions in accordance with the data. Now we start picking up the data points and find out which cluster, particular data point belongs to depending on the distance between each data point and all of the cluster centres and assigning it to cluster that is closest.

The purpose of the hidden layer is to summarize the data and compress it. The multidimensional data is to be reduced to either 2 or 3 dimensions only as then we can plot the graphs too. If we use the PYTHON LANGUAGE then note that layers and column indices are counted from zero while in R they are counted from one.

With auto-encoding neural networks we do learning one layer at a time.

For example, a model, m1, say, has the following-

- 5 hidden layers
- 11 neurons (in middle layer)

It is trained on the raw data. Then that third hidden layer is extracted into a transformation, f1 with still 100 rows but only 11 columns. Model-2 (m2) uses this f1 as its input and builds up a much simpler model, reducing 11 input nodes to 2 hidden nodes then back out to 11 output nodes. At the end, results are put in f2. f2 now has 2 columns but still 100 rows.

Advantages of k-means clustering

1. It is fast, robust and simple algorithm.
2. It is a an efficient algorithm.
3. It gives best results when dataset are well separated from each other.

Disadvantages of k-means clustering

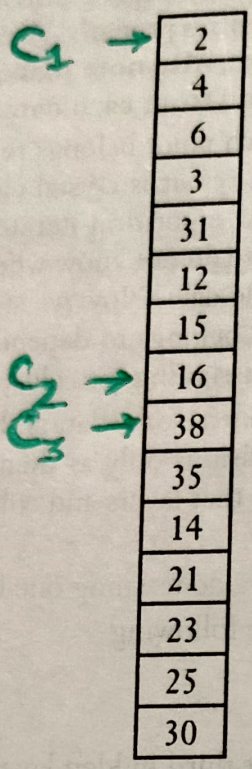
1. It requires pre-specification of number of clusters.
2. It cannot handle noisy data and outliers.
3. It cannot classify highly overlapping data.
4. With different data representations we get different results.
5. Random choosing of cluster center does not give good results.

Let us solve a problem now on k-means.

Example: You are given the following dataset. Apply k-means clustering for k=3 (*i.e.*, 3 clusters). **The dataset is 1D-data and is as follows :**

Use $C_1(2)$, C_2 (16) and C_3 (38) **as initial cluster centers.**

Data:



Solution:

Given : Initial cluster centers as $C_1(2)$, C_2 (16) and C_3 (38).

To compute : Distance between each data point and cluster centers.

Calculations ; The distances are calculated as follows :

Data Points	Distance From C_1 (2)
2	$(2-2)^2 = 0^2 = 0$
4	$(4-2)^2 = 2^2 = 4$
6	$(6-2)^2 = 4^2 = 16$
3	$(3-2)^2 = 1^2 = 1$
31	$(31-2)^2 = (29)^2 = 841$
12	$(12-2)^2 = (10)^2 = 100$
15	$(15-2)^2 = (13)^2 = 169$
16	$(16-2)^2 = (14)^2 = 196$
38	$(38-2)^2 = (36)^2 = 1296$
35	$(35-2)^2 = (33)^2 = 1089$
14	$(14-2)^2 = (12)^2 = 144$
21	$(21-2)^2 = (19)^2 = 361$
23	$(23-2)^2 = (21)^2 = 441$
25	$(25-2)^2 = (23)^2 = 529$
30	$(30-2)^2 = (28)^2 = 784$

Similarly, we compute distances from $C_2(16)$ and $C_3(38)$,we get :
it is minimum of all the cluster - centers, we get :

Distance from C_2 (16)	Distance from C_3 (38)
$(2-16)^2 = (-14)^2 = 196$	$(2-38)^2 = (-36)^2 = 1296$
$(4-16)^2 = (-12)^2 = 144$	$(4-38)^2 = (-34)^2 = 1156$
$(6-16)^2 = (-10)^2 = 100$	$(6-38)^2 = (-32)^2 = 1024$
$(3-16)^2 = (-13)^2 = 169$	$(3-38)^2 = (-35)^2 = 1225$
$(3-16)^2 = (-15)^2 = 225$	$(31-38)^2 = (-7)^2 = 49$
$(12-16)^2 = (-4)^2 = 16$	$(12-38)^2 = (-26)^2 = 676$
$(15-16)^2 = (-1)^2 = 1$	$(15-38)^2 = (-23)^2 = 529$
$(16-16)^2 = (0)^2 = 0$	$(16-38)^2 = (-22)^2 = 484$
$(38-16)^2 = (22)^2 = 484$	$(38-38)^2 = 0$
$(35-16)^2 = (-19)^2 = 361$	$(35-38)^2 = (-3)^2 = 9$
$(14-16)^2 = (-2)^2 = 4$	$(14-38)^2 = (-24)^2 = 576$
$(21-16)^2 = (5)^2 = 25$	$(21-38)^2 = (-17)^2 = 289$
$(23-16)^2 = (7)^2 = 49$	$(23-38)^2 = (-15)^2 = 225$
$(25-16)^2 = (9)^2 = 81$	$(25-38)^2 = (-13)^2 = 169$
$(30-16)^2 = (-14)^2 = 196$	$(30-38)^2 = (-8)^2 = 64$

Now, we assign the data points to the cluster-center whose distance from it is minimum of all the cluster-centers, we get:

$C_1(2)$	$C_2(16)$	$C_3(38)$
$m_1=2$	$m_2=16$	$m_3=38$
[2,3,4,6]	[12,14,15,16,21,23,25]	[31,35,38]

∴ New cluster - centers are:

$m_1=3.75$
$m_2=18$
$m_3=34.67$

Similarly, using the new cluster centers, we can compute the distances from it and allocate clusters based on minimum distance. Please note that it is observed now that there is no difference in the cluster formed. So, we stop. And the final clustering result is as follows :

$C_1(3.75)$	$C_2(18)$	$C_3(34.67)$
$m_1=3.75$	$m_2=18$	$m_3=34.67$
[2,3,4,6]	[12,14,15,16,21,23,25]	[31,35,38]

Python code for K-means

Consider a dummy dataset. Python code for k-means is as follows :

```
# k-means on dummy dataset
   import numpy as np
   import matplotlib. pyplot as plt
   from sklearn. cluster import k-means
# for reproducibility
   np. random.seed(1000)
   np. samples = 1000
   X,_=make_blobs(n_samples=nb_samples,
              n-features = 2,
              centers = 3,
              cluster_std = 1.5,
              random_state = 1000)
```

Algorithm (k-means)

Let $X=\{x_1, x_2, \ldots\ldots\ldots\ldots x_n\}$ be the set of data points & $V = \{V_1, V_2, \ldots\ldots\ldots\ldots V_n\}$ be the set of centers

k-means clustering involves the following steps:

Step-1: **Randomly** select 'c' cluster centres.

Step-2: Compute the **distance** between each data point and cluster centres.

Step-3: Assign the data point to the cluster whose distance from the cluster centre is **minimum** in the value of cluster centres.

Step-4: Calculate the **new cluster centre** using the following formula:

$$v = 1/c_i \, \Sigma \, (j=1 \text{ to } c_i) \, x_i$$

where c_i represents the number of data points in the i^{th} cluster.

Step-5: New **cluster centres** are computed.

Step-6: If no data point is reassigned then stop else repeat from step-3.

As we know that H2O is software for machine learning and data analysis. H2O algorithm has been developed keeping in mind clustering only. But clusters in H2O have some limitations:

(a) Each H2O node on the cluster must be of the same size. It implies that if the smallest machine in our cluster has 2GB free, then every node must be given 2GB even if some of them have 32GB.

(b) You cannot add the machines once the cluster has started.

(c) In case a machine dies then the whole of the cluster must be rebuilt. **Please note that the whole cluster becomes unusable if a single node gets removed. In such a case, you cannot even export data or models from the left over clusters.**

(d) Each node must run the same API version of h2o.jar.

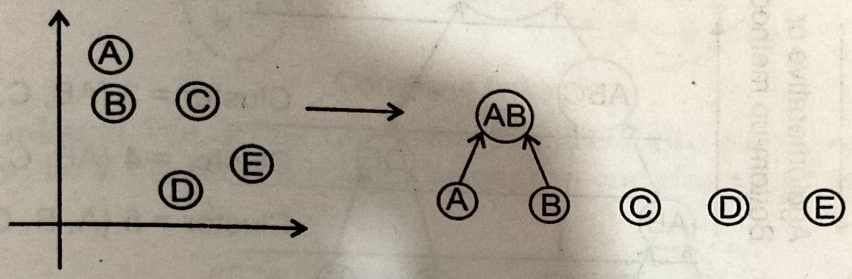
(e) To reduce the network latency, the nodes should be physically close.

You can also give your cluster a name and specify this name when starting on each node. In H2O, the cluster can be created in two ways:

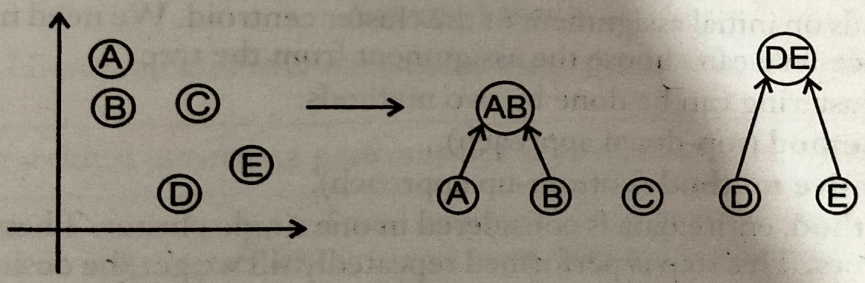
1. **Flat-file:** A simple text file giving the IP address and port number of each node in the cluster. **Note that you must prepare this file identically on each machine.**
2. **Auto-discovery:** When you specify the network address then it searches through that subnet to find all nodes and join them together in a cluster. This is slower than the flat file.

5.2 Hierarchical Clustering

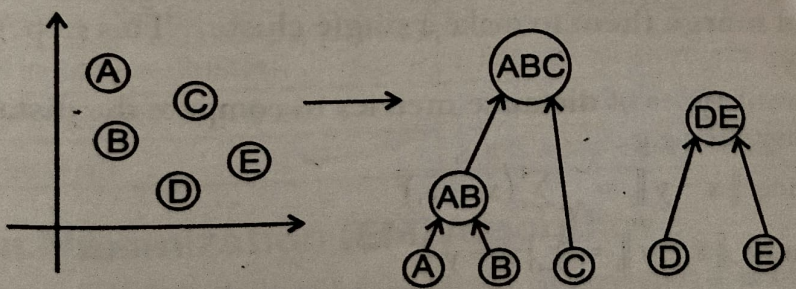
Hierarchical Clustering is an algorithm that builds the **hierarchy of clusters.** To understand this consider the example of points in the space. Each point is considered as a cluster. Say, there are a total of 5 clusters (A,B,C,D,E) as shown below:



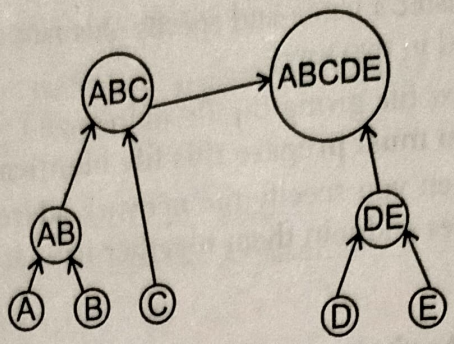
Now, **merge two clusters** which are **closer** to each other **(Rule-1).** So, A and B are merged together. Again find two nearest clusters among the four (AB,C,D,E) clusters. AB cluster center is considered at the **middle** of both the clusters **(Rule-2).**



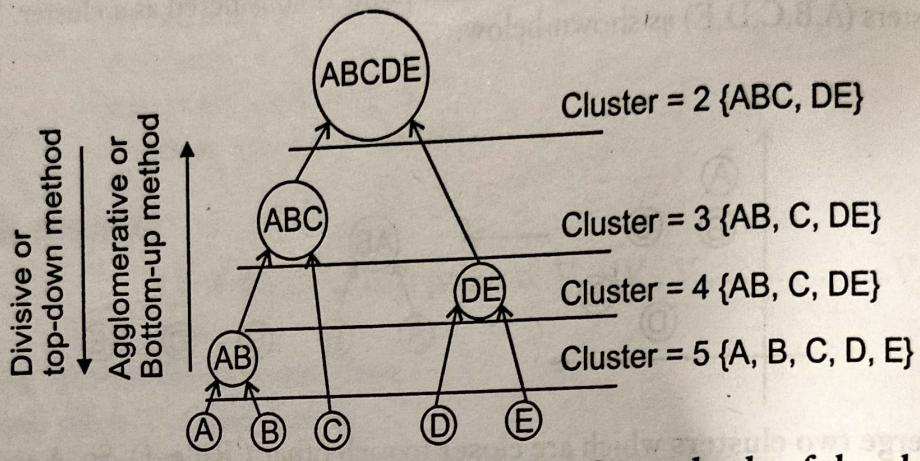
∴Now, D and E are merged together. Again, we repeat there steps and we get the following:



Now, we make the last cluster.



Please note that after building the complete hierarchy, we cut the tree at different length and get the clusters



We can decide the **number of clusters by the tree. General rule of thumb is to cut the tree at the middle of the highest branch of the tree. (Rule-3).** Hierarchical clustering cannot handle huge amount of data. Clusters will be same on different runs unlike K-means clustering which highly depends on initial assignment of the cluster centroid. We need not specify the number of clusters in advance. We can choose the assignment from the tree.

Hierarchical clustering can be done by two methods:

(a) Divisive method (top-down approach).

(b) Agglomerative method (bottom-up approach).

In **divisive method**, entire data is considered in one single cluster. Then it is **partitioned** into **two separate clusters.** This step is performed repeatedly till we get the desired number of clusters or we create **one cluster for each point.**

In **agglomerative method**, we consider each point as a single cluster. In each step, we select **two closest point** and **merge them** to make a **single cluster.** This step is repeated till we are left with **one cluster only.**

We can use different types of **distance metrics** to compute the **distance between points in hierarchical** clustering. For e.g :

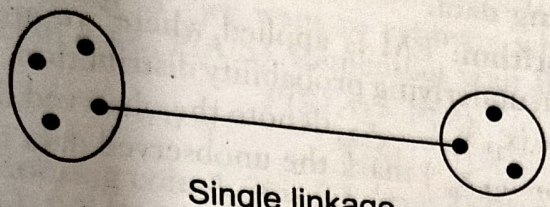
Euclidean Distance $\|x-y\|_2 = \sqrt{\sum_i (x_i - y_i)^2}$

Manhattan Distance $\|x-y\|_1 = \sum_i |x_i - y_i|$

Maximum Distance $\|x-y\|_\infty = \max_i |x_i - y_i|$

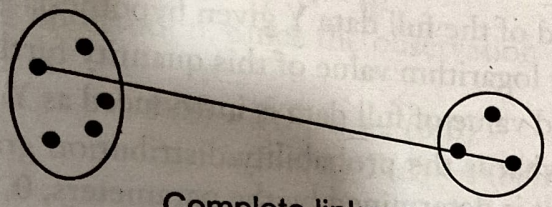
Also there are different ways to define **linkage between two clusters.** Using different linkage techniques, we get different cluster assignments. It is of 3 types

(a) Single Linkage: We define the distance between two clusters as the **minimum distance** between any points in both the clusters. One point belongs to one cluster and second point beliongs to another cluster. That is,



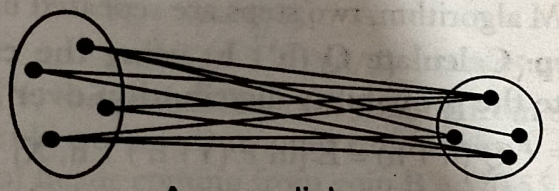
Single linkage

(b) Complete Linkage: We define the distance between two clusters as the **maximum distance** between any points in both the clusters. One point belongs to one cluster and second point belongs to another cluster.



Complete linkage

(c) Average Linkage: We define the distance between two clusters as the **mean distance** between all points in both cluster. One point belongs to one cluster and second point belong to another cluster.



Average linkage

Applications: Hierarchical clustering is used during social media analysis, market segmentation etc.

> **Note:** Hierarchical clustering performs better as compared to k-means clustering.

A **dendrogram** is a commonly used tree structure of hierarchical clustering. It shows how the clusters are **merged** or **split** iteratively to arrive at an **optimal clustering solution.**

Let us compare the two techniques.

Agglomerative Clustering	Divisive Hierarchical Clustering
1. Initially, each item is in its own cluster.	1. Initially, all items are in one cluster.
2. Iteratively clusters are merged together.	2. Large clusters are successively divided (split)
3. It is a bottom-up approach.	3. It is a top-down approach.

5.3 Expectation Maximization (EM) Algorithm

Many a times the presence of unobserved variables is observed as a problem of learning. In case of some variable that can be observed sometimes and cannot be observed at other times then the time when it was observed can be used to learn its values when it was not observed. So, EM is

(a) Binary Adaptive Resonance Theory (ART1).
(b) Analog adaptive Resonance Theory (ART2).
(c) Discrete Hopfield (DH).
(d) Continuous Hopfield (CH).
(e) Discrete Bi-directional Associative Memory (BAM).
(f) Temporal Associative Memory (TAM).
(g) Adaptive Bi-directional Associative Memory (ABAM).
(h) Kohonen Self-Organizing Map/ Topology Preserving Map (SOM/TPM).
(i) Competitive learning.

2. Feed-forward only Nets:
   (a) Learning Matrix (LM).
   (b) Driver-reinforcement Learning (DR).
   (c) Counter Propagation (CPN).

II. Supervised learning (*i.e.* with a teacher):

1. Feedback Nets:
   (a) Boltzmann Machine. (BM).
   (b) Mean Field Annealing (MFT).
   (c) Recurrent Cascade Correlation (RCC).
   (d) Learning Vector Quantization (LVQ).
   (e) Back Propagation through Time (BPTT).
   (f) Real-time Recurrent Learning (RTRL).
2. Feed-forward only Nets:
   (a) Perceptron.
   (b) Adaline, Madaline.
   (c) Backpropagation (BP).
   (d) Cauchy Machine (CM).
   (e) Artmap.
   (f) Cascade Correlation (CasCor).

Q38. Comparison Agglomerative and Divisive Hierarchical Clustering.

Ans. Comparisons of **Agglomerative** and **divisive Hierarchical clustering**—

Agglomerative	Divisive Hierarchical clustering
1: Initially, each item is in its own cluster.	1: Initially, all items in one cluster.
2: Iteratively clusters are merged together.	2: Large clusters are successively divided.
3: It uses a bottom-up approach.	3: It uses a top-down approach.

Q39. Illustrate k-means algorithm with the help of the 3D dataset of 10 points given below:

(1,1,1), (1,1,2), (1,3,2), (2,1,1), (6,3,1), (6,4,1), (6,6,6), (6,6,7), (6,7,6), (7,7,7).
Consider initial seeds to be (1,1,1), (6,3,1), (6,6,6).

Ans. The first iteration of the k–means algorithm:

Cluster 1	Cluster 2	Cluster3
(1,1,1)	(1,1,2)	(1,3,2)
(2,1,1)		(6,3,1)
		(6,4,1)
		(6,6,6)
		(6,6,7)
		(6,7,6)
		(7,7,7)

Centroids are as follows:

C_1=(1,1,1)

C_2=(6,3,1)

C_3=(6,6,6)

$$D^0 = \begin{bmatrix} 0 & 1 & 2.23 & 1 & 5.38 & 5.23 & 8.66 & 9.27 & 9.27 & 10.39 \\ 5.38 & 5.47 & 5.09 & 4.24 & 0 & 1 & 5.83 & 8.24 & 6.40 & 7.28 \\ 8.66 & 8.12 & 707 & 8.12 & 5.83 & 5.83 & 0 & 1 & 1 & 1.73 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$\therefore$ C_1 = (5/4, 6/4, 6/4) = (1.25, 1.5, 1.5)

C_2 = (12/2, 7/2, 2/2) = (6, 3.5, 1)

C_3 = (25/4, 26/4, 26/4) = (6.25, 6.5, 6.5)

Exercise Questions

Q1. Differences between supervised and unsupervised learning.

Q2. Explain PCA and ICA.

Q3. Define clustering.

Q4. Write k-means algorithm.

Q5. What is Factor Analysis? Explain.

Q6. What is LSI?

Q7. Write short notes on HMM.

Q8. What is ICA? Discuss

Q9. Discuss k-means clustering with examples.

Q10. (a) What is PCA? Discuss its steps in details.

(b) What is spectral clustering? Discuss any one spectral clustering algorithm.

Q11. Write short notes on HMM.

Q12. (a) Explain PCA and ICA.

(b) Explain spectral clustering.

Q13. (a) Explain HMM in detail.

(b) Discuss differences between Markov and HMM models.

Q14. Apply k-means to find the clusters assume k=2.

Data sample is

Height	Weight
185	72
170	56
168	60
179	68
182	72
188	77
180	71
180	70
183	84
180	88
180	67
177	76

Q15. What is zero centering?

[Hint: During feature normalization, we can do two things:

1. **Centering:** It means moving the entire dataset so that it is centered around the origin.
2. **Scaling:** It means re–scaling each feature so that one of the following holds:
   (a) Each feature has variance 1 across the training data.
   (b) Each feature has maximum absolute value 1 across the training data.

Note that the goal of centering is to make sure that no features are arbitrarily large.
The goal of normalization is to make it easier for your learning algorithm to learn.]