# FUNDAMENTALS OF ALGORITHMIC PROCESSES

# CHAPTER 6: MACHINE LEARNING ALGORITHMS

## 6.1. Introduction

The field of statistics known as computational learning theory studies the computational analysis and performance of machine learning methods. Such algorithms are developed in machine learning to assist the computer in learning. Learning does not always imply consciousness; identifying statistical symmetries or patterns in a set of data is also a form of learning. Human learning algorithms bear no resemblance to machine learning algorithms. Machine learning methods, on the other hand, could provide insight into the relative difficulty of learning in various situations.

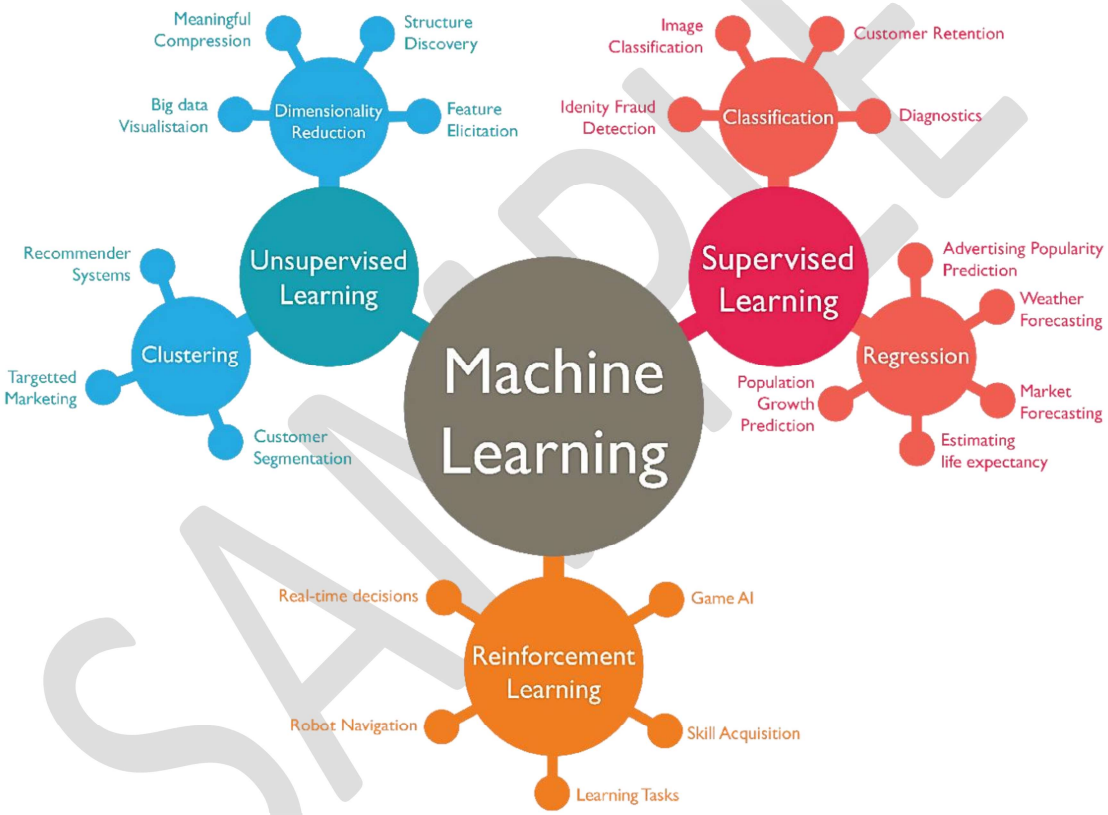


***Figure 6.1. Different types of machine learning algorithms***

*Source: https://towardsdatascience.com/machine-learning-algorithms-in-laymans-terms-part-1-d0368d769a7b?gi=3f432d1ebd11*

Learning algorithms are classified into multiple groups based on the algorithm's preferred outcome. The common learning algorithm types are:

i. Supervised learning ---The algorithm creates a function that maps the inputs to the expected outputs. One of the classic forms of this learning is the categorization issue. The learner must

assimilate information to evaluate the performance of a function in a classification issue. The function displays a vector from many classes by considering various input-output specimens of the function.

ii. Unsupervised learning --- Unsupervised learning models a set of inputs. The instances that are labeled are not available.

iii. Semi-supervised learning --- A suitable function or classifier is produced as a result of combining the labeled and unlabeled instances in this learning process.

iv. Reinforcement learning ---Given the observation of the world, in this case, the algorithm develops a strategy for how to carry out the task. In every action, there is a corresponding effect on the environment, and the environment offers feedback on these effects. The learning algorithm is then escorted by the feedback.

v. Transduction --- This is like supervised learning, except that no function is expressly established. Transduction learning aims to predict new outcomes based on previously learned inputs and outputs, as well as fresh inputs.

vi. Learning to learn --- The program develops its own inductive preference based on its prior experience with this type of problem.

## 6.2. Supervised Learning Approach

In classification problems, supervised learning is quite popular because the purpose is to train a computer to study a classification system that has been built, which is quite common. A good example of this type of learning is digited recognition. Classification learning is appropriate for challenges in which the classification can be determined readily and in which presuming a classification is beneficial is assumed. Some situations where the mediator is able to resolve the classification problem on his or her own make it unnecessary to assign pre-determined classifications to each instance of a problem. In a classification context, this is an example of unsupervised learning, which may be seen here.

In supervised learning, there is a chance that the inputs will be left undefined on a number of occasions. If the necessary inputs are provided, this model is not required. In the absence of some input values, it is not possible to make any predictions regarding the outcomes of the simulation. Unsupervised learning is predicated on the assumption that all observations are begun by a latent variable, and it is predicted that the observations will be at the conclusion of the causal chain in this scenario. The examples of unsupervised and supervised learning are depicted in the diagram to the right.

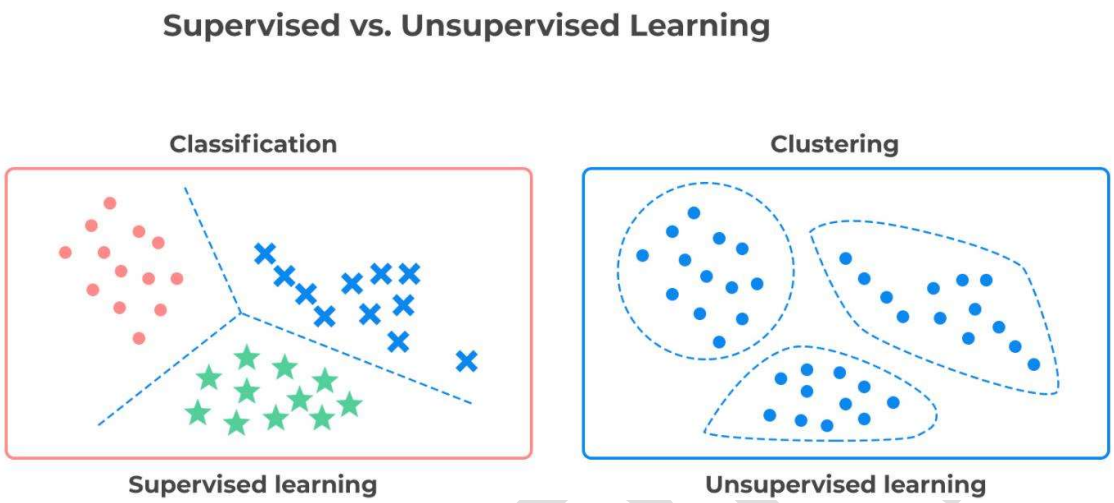


***Figure 6.2. Illustration of Supervised Learning and Unsupervised Learning Systems***

*Source: https://www.intechopen.com/books/new-advances-in-machine-learning/types-of-machine-learning-algorithms*

The most widely used technique for training neural networks and decision trees is supervised learning, which is also the most effective. Both neural networks and decision trees rely heavily on the information provided by pre-determined classifications in order to function properly. As a result of the classification, errors in neural networks are determined. The classification then adjusts the network in order to decrease errors. In contrast, in decision trees, the qualities that provide the most information that aids in the solution of the classification enigma are decided by the classification process itself. Both of these strategies will be addressed in greater depth later on in this article. It is sufficient for knowledge to know that these examples thrive under "supervision" in the form of pre-determined categorization and that this is true.

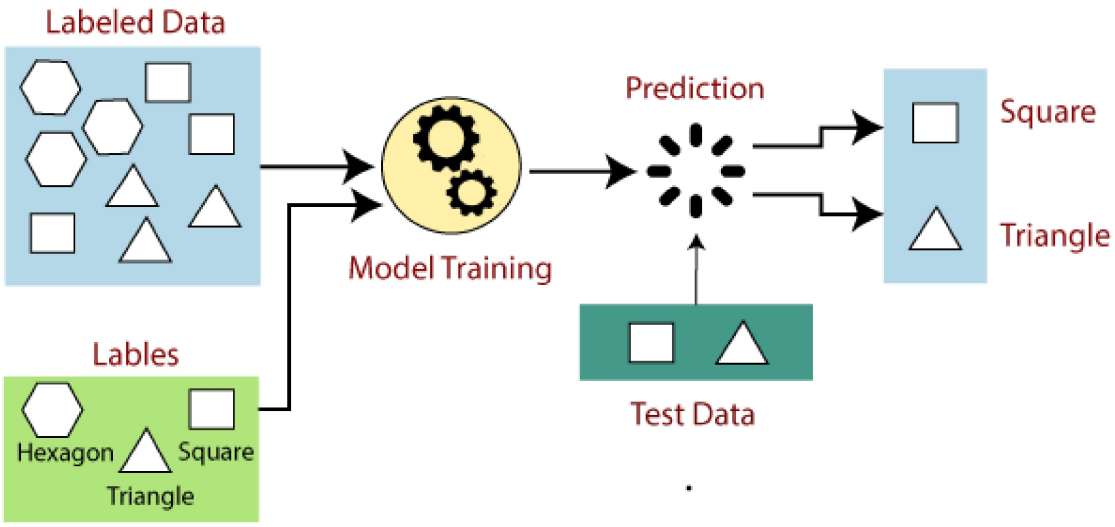


***Figure 6.3. Supervised learning algorithm***

*Source: https://geekycodes.in/what-is-supervised-learning/*

Inductive machine learning is the process of learning a set of rules from instances or, to put it another way, developing a classifier that aids in generalizing from fresh examples. The parts that follow describe how supervised machine learning is applied to a real-world situation. The initial phase entails gathering the data. If an essential specialist is available, she or he can advise on the most useful traits and features. If the expert is not available, a "brute-force algorithm" can be utilized, in which everything obtainable is measured in the hopes of making the appropriate features inaccessible. The dataset gathered using the "brute-force algorithm" is insufficient for induction. According to Zhang, it usually contains the needed information (2002).

The data preparation and pre-processing is the next phase. Researchers have a variety of strategies to address missing data depending on the circumstances (Batista, 2003). Hodge (2004) presented a survey of current noise detection strategies. These researchers also shed insight on the benefits and drawbacks of these strategies. Instance selection is employed to deal with noise as well as the challenges of learning from huge datasets. In these datasets, the optimization challenge is instance selection, which aims to preserve mining quality while minimizing the sample size. It reduces the amount of data and uses a data mining method to work with enormous datasets effectively. For sampling the instances from the huge dataset, a variety of procedures are possible (Allix, 2000; Mooney, 2000).

In the field of feature subset selection, the method of identifying and deleting as many incorrect and discarded characteristics as possible is referred to as the selection of features (López 2001; Lopez et al. 2002; Yu 2004). As a result of this procedure, the dimensionality of data is reduced, allowing data mining algorithms to operate more quickly and efficiently than before. It is the fact that several

features are reliant on one another that has a significant impact on the precision of supervised Machine language classification models, as previously stated. It is possible to find a solution to this challenge by developing creative features from a basic feature set of features. This technique is referred to as the feature construction technique. The recently developed features may prove useful in the development of classifiers that are more concise and exact. The finding of the relevant traits contributes to the enhanced clarity of the classifier that has been built, as well as the improved understanding of the notion. The implementation of Markov models and Bayesian networks in voice recognition is dependent on a few aspects of supervision in order to adjust the parameters in such a way that the mistakes may be minimized when given inputs are provided (Mostow, 1983; Fu & Lee, 2005; Ghahramani, 2008).

The most crucial thing to keep in mind is that the purpose of the learning process in a classification problem is to lower the error in accordance with the particular inputs being considered. These inputs, referred to as the "training set," are the specimens that aid the agent in its learning process. However, it is not always necessary to become familiar with these inputs. Suppose I want to demonstrate the exclusive-or and exhibit the combinations consisting of one true and the other false, for example (Getoor & Taskar, 2007; Rebentrost et al., 2014). The combination of both false and true is never presented; in this case, it is possible to learn the rule that states that the response is true at all times. Additionally, with machine learning algorithms, the problem of overfitting the data and basically remembering the training sets instead of adopting a more broad categorization strategy is a problem that is commonly encountered (Rosenblatt, 1958; Durbin & Rumelhart, 1989). The inputs to each of the training sets are incorrectly categorized. This is a serious problem. This can result in issues if the implemented method is dominant enough to remember "special situations" that are not fit for the general principles of the algorithm in question. Overfitting can occur as a result of this. It is quite difficult to discover algorithms that are both powerful enough just to absorb complex functions and vigorous enough to yield outcomes that are generalizable over a wide range of situations (Hopfield,

1982; Timothy Jason Shepard, 1998).

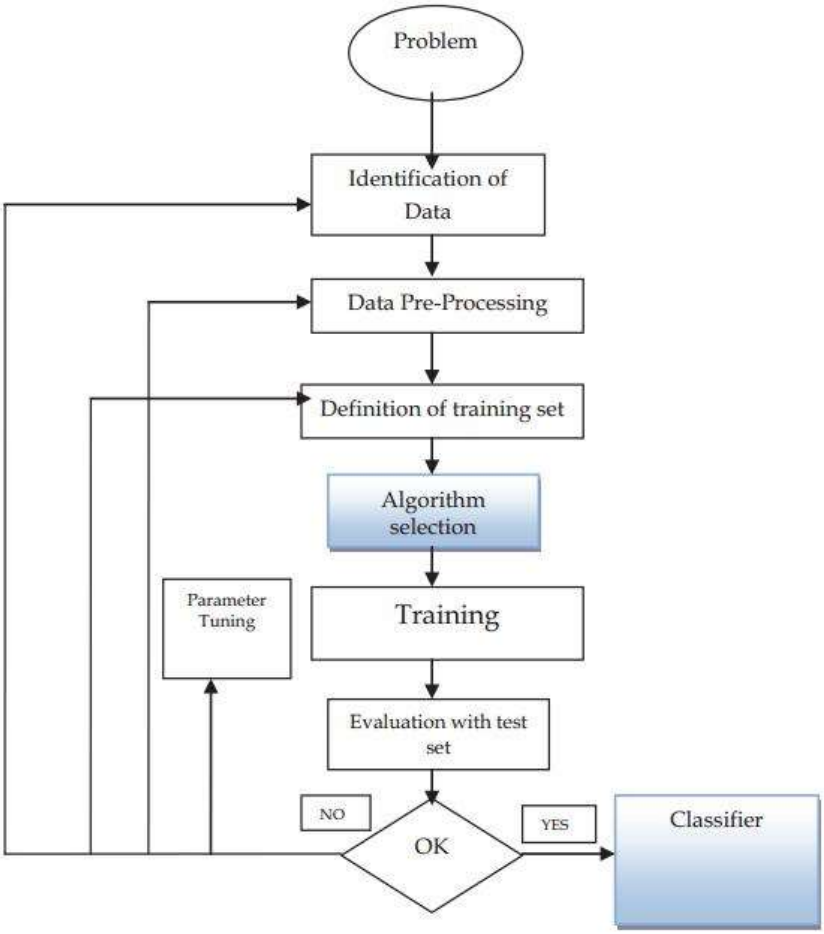


***Figure 6.4. Schematic illustration of Machine Learning Supervise Procedure***

*Source: https://www.intechopen.com/books/new-advances-in-machine-learning/types-of-machine-learning-algorithms*

## 6.3. Unsupervised Learning

This strategy appears to be considerably more difficult: the purpose of this approach is to teach the computer how to do a task without our assistance. Unsupervised learning can be accomplished in two ways. The first way entails training the agent by implementing a reward system that shows achievement rather than by offering unambiguous categorizations. Because the purpose is to make decisions that will enhance the rewards rather than to produce a classification, this form of training is usually appropriate for the decision issue frame. This strategy is well-suited to the actual world, where agents may be rewarded for certain activities and chastised for others (LeCun et al., 1989; Bilmes, 1998; Alpaydm, 1999). Unsupervised learning is a type of supportive learning in which the agent's behaviors are based on previous punishments and rewards deprived of necessarily learning information about the direct ways in which they affect the real environment. This knowledge is useless in the sense that once the

agent has been accustomed to the reward function, he or she knows exactly what action to take without having to think about it. The agent is well aware of the exact reward that must be obtained for each activity.

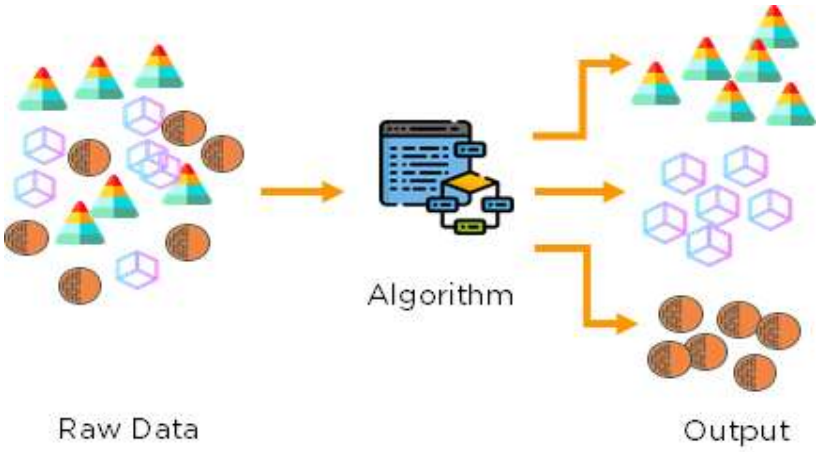


***Figure 6.5. Illustration of unsupervised learning***

*Source: https://medium.com/analytics-vidhya/beginners-guide-to-unsupervised-learning-76a575c4e942*

The use of this method can be particularly advantageous in situations where the calculation of each alternative takes an inordinate amount of time. However, learning via trial and error can be a very time-consuming process on the other hand. However, this sort of learning has the potential to be extremely effective because it does not rely on the classification of samples that have already been discovered. In some instances, our classifications are not the most accurate that could be achieved (Rumelhart et al., 1985; 1986; Schwenker & Trentin, 2014). Consider the case of backgammon, where the predictable knowledge in the game was turned on its head when computer programs that learned through unsupervised learning outperformed the best human chess players in terms of strength and endurance. There were specific ideas that these programs picked up that astounded the backgammon professionals, and they performed significantly better than the backgammon programs educated on the pre-classified samples. Clustering is another sort of unsupervised learning. The goal of this type of learning is to discover resemblances between training data and test data rather than to maximize the utility function (Xu & Jordan, 1996; Mitra et al., 2008).

Now, the assumption is that the clusters that have been discovered will correspond to the intuitive classification in a logical manner. For example, the clustering of persons based on demographics may result in the grouping of the wealthy in one set and the impoverished in the other set, depending on the circumstances. Despite the fact that the algorithm will not assign names to these clusters, it will be able to generate them and subsequently, by using these clusters, will be able to allocate fresh samples into

one of the clusters (Herlihy, 1998; Gregan-Paxton, 2005). The data-driven technique described above is effective when a large amount of data is accessible; for example, social information filtering algorithms, which are employed by Amazon.com to propose books, are an example of such an approach (Stewart & Brown, 2004; Sakamoto et al., 2008). It is the concept of these algorithms that new customers are assigned to equivalent groups of people after they have been discovered and classified by them. It is enough for the algorithm to produce relevant results when social information filtering is used to gather information about other members of a cluster of people to be considered. In the remaining circumstances, clusters are simply a valuable tool for expert analysts to use in their analysis. Unfortunately, the unsupervised learning method is equally susceptible to the overfitting problem. There is no easy route to avoiding the problem of overfitting because the algorithm that can adapt to changes in its inputs must be powerful enough to overcome the problem (Nilsson, 1982; Vancouver, 1996; Sanchez, 1997).

Unsupervised learning methods are designed to remove structure from data samples without the assistance of a supervisor. Typically, the cost function is employed to determine the value of a structure, and it is minimized in order to derive the most favorable parameters that illustrate the hidden structure in data. The guarantee that the structures extracted are typical for the source is required for consistent and robust inference; for example, the structures extracted from the second model set of a comparable data source must be the same as those retrieved from the first model set of a similar data source (Bateson, 1960; Campbell, 1976; Kandjani et al., 2013). Overfitting is a term used in the statistical and machine learning literature to describe a lack of robustness in a model. Currently, the overfitting phenomena are described by a group of histogram clustering dummies, which play a key role in information recovery, linguistics, and computer vision applications, among other things. Because of the enormous deviations in the outcomes, learning algorithms with the potential to simulate fluctuations have emerged as the ultimate entropy principle for the process of learning (Pollack, 1989; Pickering, 2002; Turnbull, 2002).

Many successes have been produced by unsupervised learning, such as:

i. World champion-caliber backgammon program.
ii. Machines skilled in driving cars.

When there is a simple way to assign values to activities, unsupervised learning can be a powerful tool. Clustering is helpful when there is enough data to construct clusters, and especially when supplementary data about a cluster's adherents are used to generate additional outcomes due to data dependencies (Acuna & Rodriguez, 2004; Farhangfar et al., 2008). When it is known whether the classifications are true or are just random objects that the computer can recognize, classification learning is useful. Classification learning is typically required in scenarios when the algorithm's decision is required as input in another context. Clustering and classification learning are both beneficial, and the best strategy to use depends on the following factors (Dutton & Starbuck, 1971; Lakshminarayan et al., 1999).

i. Kind of a problem being solved.
ii. Time allotted for solving it.
iii. Either supervised learning is possible.

## 6.4. Algorithm Types

In the extent of supervised learning where which mostly classification is dealt with, the following are the types of algorithms:

i. Linear Classifiers
ii. Naïve Bayes Classifier
iii. Logical Regression
iv. Support Vector Machine
v. Quadratic Classifiers
vi. Perceptron
vii. Boosting
viii. Decision Tree
ix. K-Means Clustering
x. Neural networks
xi. Random Forest
xii. Bayesian Networks

### 6.4.1. Linear Classifiers

Classification is used in the machine learning process to arrange objects with similar feature values into groups. As per Timothy et al. (1998), linear classifiers achieve this with the help of better decisions (Grzymala-Busse & Hu, 2000; Grzymala-Busse et al., 2005). The categorization decision is influenced by the linear combination input. The output is supplied as if the real vector $\vec{x}$ were the input to the classifier.:

$$y = f(\vec{w} \cdot \vec{x}) = f\left(\sum_j w_j x_j\right),$$

Where, $\vec{w}$ → real weights vector,f→ function in which the dot product of two vectors is translated into the preferred output. A set of marked training samples helps to deduce the vector $\vec{w}$. The function f is usually used to translate values over a certain threshold to first class and the remainder of the data to second class. The complex function f determines the likelihood that an object belongs to a specific category (Li et al., 2004; Honghai et al., 2005; Luengo et al., 2012).

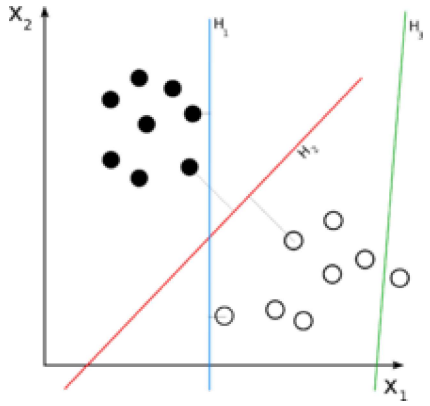


***Figure 6.6. Graphical representation of linear classifiers***

*Source: https://en.wikipedia.org/wiki/Linear_classifier*

**Note:**

Any number of linear classifiers can correctly classify the solid and empty dots in this example. H1 (blue) and H2 both appropriately classify them (red). H2 is "better" in the sense that it is the furthest away from both groupings. The dots are incorrectly classified by H3 (green).

The linear classifier's functionality in two-class classification can be viewed as dividing the high-dimensional input vector with a hyperplane. The points on one side of the hyperplane are labeled "yes," while those on the other are labeled "no." Because linear classifiers are the fastest classifiers, especially when the real vector is sparse, they are frequently utilized in situations where classification speed is a concern. Decision trees can also be made to run quicker (Hornik et al., 1989; Schaffalitzky & Zisserman, 2004). When the number of dimensions in the real vector is big, linear classifiers typically perform well. Every element in the real vector in document categorization is usually the count of a specific word in the document. In these situations, the classifier must be well-regularized (Dempster et al., 1997; Haykin & Network, 2004).

According to (Luis et al.), a support vector machine performs classification by building an N-dimensional hyperplane that divides data into categories as efficiently as possible. Support vector machine models and neural networks are inextricably linked. The sigmoid kernel function is utilized in the Support vector machine model, which is similar to the two-layer (PNN) perceptron neural network (Rosenblatt, 1961; Dutra da Silva et al., 2011).

Traditional multilayer (PNN) perceptron neural networks are closely connected to SVM models. Support vector machine models, which use a kernel function as a training scheme, are a good alternative to polynomials, radial basis functions, and multilayer perceptron neural networks classifiers. Instead of tackling a non-convex and unconstrained minimization issue like in normal neural network instruction,

the weight of the network is determined by solving a quadratic programming problem with linear constraints in SVM (Gross et al., 1969; Zoltan-Csaba et al., 2011; Olga et al., 2015).

The attribute is a predictor variable in the support vector machine literature, and a feature is a changed attribute that specifies the (HP) hyperplane. Feature selection is the process of selecting the best appropriate representation. A vector is a collection of features that describe one of the situations. The purpose of SVM (support vector machine) modeling is to find the best hyperplane for splitting vector groups so that cases belonging to one category of the target variable are on one side of the plane and cases belonging to the other category are on the other (Block et al., 1962; Novikoff, 1963; Freund & Schapire, 1999). The support vectors are the vectors that are nearest to the hyperplane. The next section gives an overview of the support vector machine procedure.

### 6.4.2. A Two-Dimensional Case

Before we look at N-dimensional hyperplanes, let's have a look at a simple 2-dimensional illustration. Consider the following scenario: we want to perform classification, and the data we have available has a specified target variable with two categories. Also, consider the possibility that two predictor variables with continuous values are available as well. If the data points are plotted on the X and Y axes with the values of the predictor on the X and Y axes, the image displayed below may be the result. The rectangles indicate one category, and the ovals represent the other category, as shown below (Lula, 2000; Otair & Salameh, 2004; Minsky & Papert, 2017).

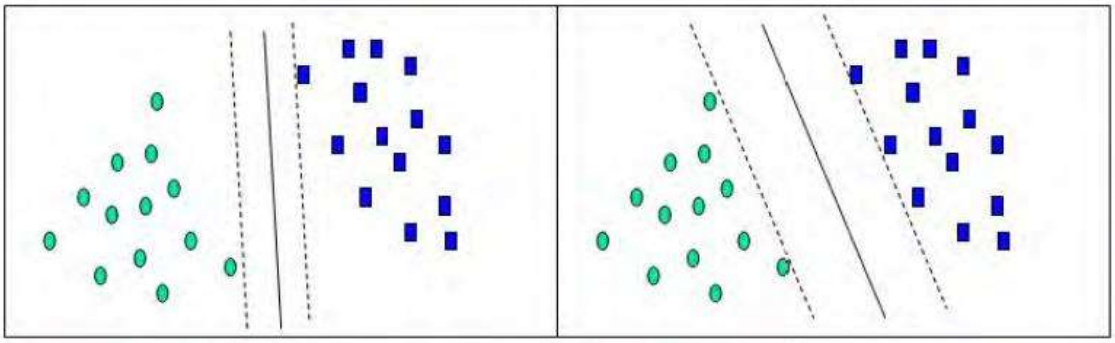

***Figure 6.7. The demonstration of SVM analysis for determining 1D hyperplane (i.e., line) which differentiates the cases because of their target categories***

*Source: https://www.intechopen.com/books/new-advances-in-machine-learning/types-of-machine-learning-algorithms*

In this instance, the cases of one category are in the lower-left corner, while the examples of another category are in the upper right corner. Both scenarios are completely distinct from one another. The one-dimensional hyperplane that separates both cases on the basis of their target categories is sought by

the Support Vector Machine analysis. There are an infinite number of alternative lines; the two candidate lines are depicted above. Now the question is, which line is far superior, and how is the optimum line defined? (Caudill & Butler, 1993; Hastie et al., 2009; Noguchi & Nagasawa, 2014).

The dotted lines parallel to the dividing line indicate the distance between the dividing line and the nearest vectors to the line. The space between dotted lines is the margin. The points that define the margin's size are known as support vectors. It is depicted in the diagram below.

The line, or, in particular, the hyperplane that is angled to maximize the margin among the support vectors, is discovered using the Support Vector Machine analysis (Luis Gonz, 2005; Hall et al., 2009).

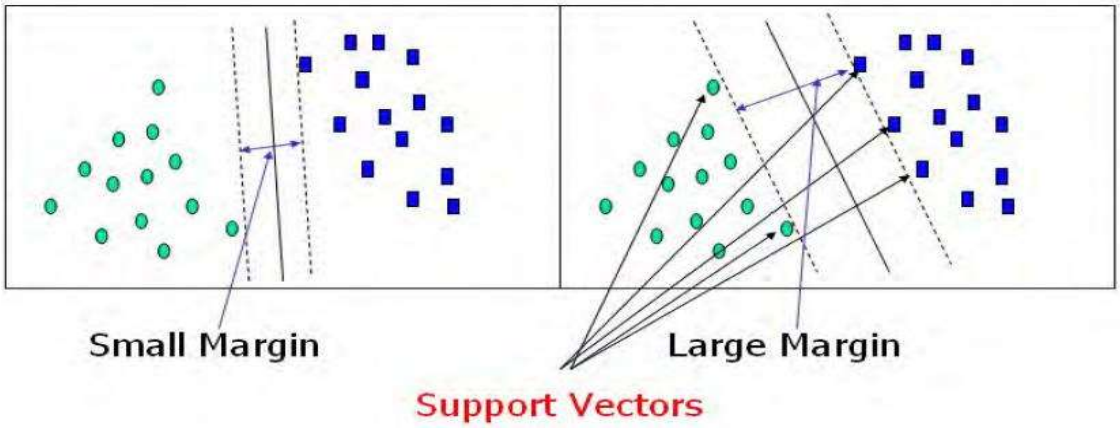


***Figure 6.8. Illustration of an SVM Analysis containing dual-category target variables possessing two predictor variables having the likelihood of the division for point clusters***

*Source: https://www.intechopen.com/books/new-advances-in-machine-learning/types-of-machine-learning-algorithms*

The line in the right section is superior to the line in the other section, as shown in the diagram above. Life would be stress-free if all analyses included two category goal variables, two predictor variables, and a straight line that could separate the group of points (Neymark et al., 1970; Hammes & Wieland, 2012). Unfortunately, this is not usually the case; therefore, the Support Vector Machine must deal with:

a. Handling of the cases where more than the two predictor variables distinct the points with the non-linear curves.
b. Handling of the cases where groups cannot be separated completely.
c. Handling of the classifications having more than the two categories.

Three major machine learning techniques are explained in this chapter with examples and how these techniques perform in reality. These are:

i. K-Means Clustering

ii. Neural Network
iii. Self-Organized Map

### 6.4.3. K-Means Clustering

This technique consists of a few simple steps. In the beginning, K (the cluster number) is established, and the cluster center is assumed. The primary center might be any random item, or the first K entities in the structure can alternatively serve as the primary center (Teather, 2006; Yusupov, 2007). The K means algorithm then performs the three steps listed below till convergence. Iterate until the result is stable.

i. Determining the center coordinate
ii. Determining the distance of every object from the center
iii. Grouping the objects based on the minimum distance

K- Means flowchart is shown in the figure below

K-means clustering is the simplest unsupervised learning technique that may be used to solve the well-known segmentation problem. It is also the most widely used algorithm. The procedure uses a straightforward approach to categorize the given data set into one of a pre-determined number of clusters. It is necessary to define K centroid values for each cluster. This is important since different results can be achieved from different locations (Franc & Hlavá, 2005; González et al., 2006). These centroids must be placed in a strategic manner. The more appropriate option is to place the centroids as far apart as possible from one another. It is next necessary to take every point from the provided data and associate it with the centroid nearest to that point in the data set. When there are no more points, it signifies that the first stage has been completed, and an initial grouping is carried out (Oltean & Dumitrescu, 2004; Fukunaga, 2008; Chali, 2009).

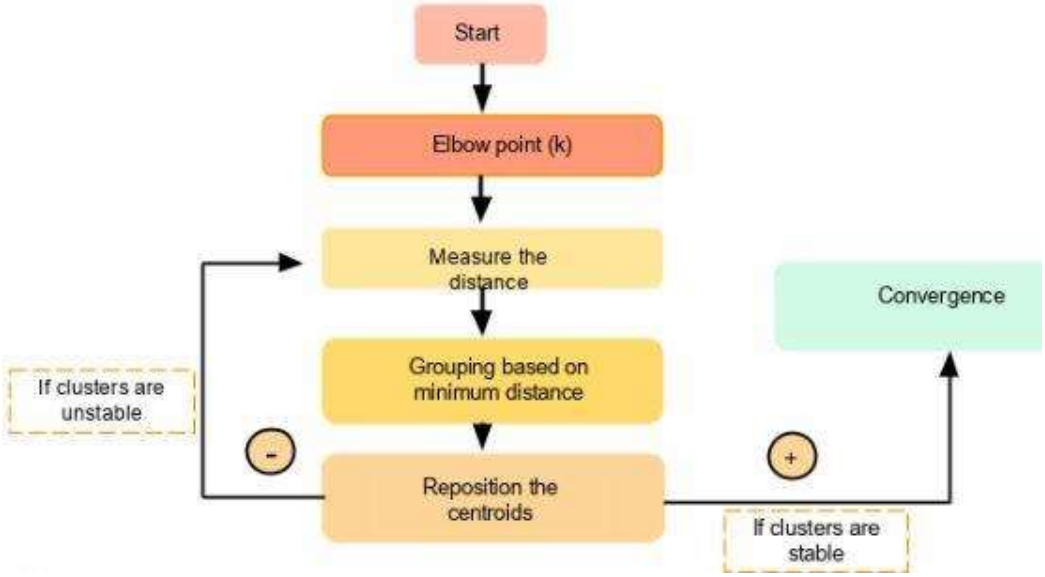


***Figure 6.9. Schematic illustration of K-means iteration***

*Source: https://www.intechopen.com/books/new-advances-in-machine-learning/types-of-machine-learning-algorithms*

It is now necessary to recalculate 'knew centroids. After acquiring the new centroids, a new binding must be performed between the same points and the nearby new centroid. The result is a loop. The k centroids gradually shift their position because of this loop because there are no more changes. As a result, the centroids stop moving (Oltean & Groşan, 2003; Oltean, 2007). This algorithm aims to reduce an objective function, in this case, a squared error function. The role of the objective:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$

,

$\left\| x_i^{(j)} - c_j \right\|^2$ It is a selected distance measure between $x_i^{(j)}$ (data point) and $_{CJ\,(}$cluster center). It is a sign of distance of *n* data points with respect to their particular cluster centers.

The algorithm for this technique consists of the steps defined below:

i. K points are placed into the space denoted by objects which are being clustered. These points denote primary group centroids.

ii. Each object is assigned to the group which has the nearest centroid.

iii. The positions of K centroids are recalculated after all of the objects have been allocated.

iv. Steps ii and iii are repeated as long as the centroids are moving. This creates a partition of objects into the groups from where the minimized metric can be calculated.

Despite the fact that it can be determined that the process will always finish, the k-means clustering algorithm does not fundamentally identify the most optimal configuration that is identical to the minimum global objective function in most cases (Burke et al., 2006; Bader-El-Den & Poli, 2007). It is also important to note that the K-means method is extremely sensitive to the major cluster centers that are arbitrarily selected. This procedure is repeated multiple times in order to reduce the impact of this effect. The K-means method has been adapted to a wide range of problem domains. When it comes to working with fuzzy feature vectors, this approach is a strong contender for modification (Tavares et al., 2004; Keller & Poli, 2007).

Suppose that n-sample feature vectors ($x_1$, $x_2$, $x_3$..., x n) are available, all of which belong to the same class, and that they all fall into one of the k dense clusters where k n is the number of samples available. Assume that mi represents the mean of the cluster i. Because they are well separated, we can use the minimal distance classifier to divide the clusters and use it to split the data. It is possible to say that the x is included in cluster I only if || x - mi || represents the shortest of all k distances between the two points. This recommends the following process in order to find the k means:

i. Supposing the values for means ($\mathbf{m}_1$, $\mathbf{m}_2$,$\mathbf{m}_3$ ..., $_{MK}$ ) unless no modifications in any of the mean occur
ii. Classifying the samples into the clusters using the estimated means
iii. For the cluster **i** from the range 1 to k
iv. Swap $\mathbf{m}_i$ by mean of all the samples for the cluster **i**
v. end_for
vi. end_until

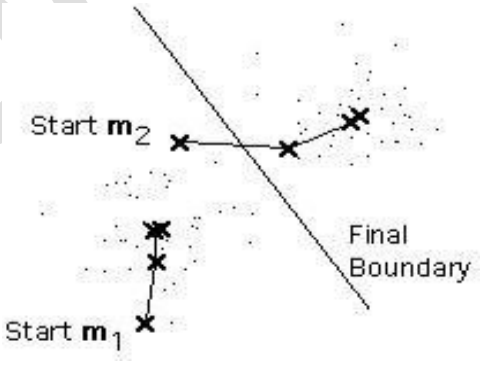


***Figure 6.10. Demonstration of the motion for m1 and m2 means at the midpoint of two clusters.***

*Source: https://www.intechopen.com/books/new-advances-in-machine-learning/types-of-machine-learning-algorithms*

This is a simplified version of the k-means algorithm. It can be thought of as a greedy technique for grouping n models into k clusters in order to minimize the total number of squared distances between cluster centers. There are certain flaws in this algorithm:

i. The procedure for determining the means was not specified. One well-known method for starting the mean is to select k of the models at random.
ii. The results are determined by the fundamental mean values, and it is usual for suboptimal partitions to emerge. The conventional technique is to choose a variety of different beginning positions.
iii. It's conceivable that a set of models close to mi is empty, preventing mi from being updated. This is an annoyance that must be addressed during implementation.
iv. The results are influenced by the measure used to quantify || x - mi ||. The traditional solution is to normalize each variable by its standard deviation, albeit this is not always desired.

### 6.4.4. Neural Network

In fact, neural networks may perform multiple reversal and classification tasks at the same time, albeit they usually only do one at a time (Forsyth, 1990; Bishop C. M., 1995; Poli et al., 2007). In most instances, the network's output variables are limited to just one. This may be related to the number of outcome units in several state classification issues. If you draw a network with a lot of output variables, crosstalk is a possibility. Hidden neurons have a hard time learning since they are trying to simulate at least two functions. The optimum way is to train a separate network for each output and then integrate them into one so that they can operate as a unit. The following sections explain how to use neural networks:

#### 6.4.4.1. Multilayer Perceptron

In today's world, this is the most widely acknowledged network architecture, having been discovered by (Rumelhart & McClelland, 1986) and thoroughly detailed in most neural network textbooks. It is briefly mentioned in the preceding sections that this type of network has the following characteristics: the units, in turn, implement the weighted sum of the inputs and, with the help of a transfer function, pass through this activation level in order to generate the output. Each of the units is configured in a covered feedback control topology. The form of the input-output model, as well as the weights of the multilayer perceptron, can be explained in a straightforward manner. As a result, the free variables in the model are biassed by this network. By varying the number of layers and the number of units in each layer, such networks may be used to construct functions of random complexity, with the complexity of the function determined by the number of levels and the number of units in each layer. When designing

MLPs (Multilayer Perceptrons), one of the essential problems to consider is the specification of hidden nodes as well as the number of units included in these hidden levels (Bishop, 1995; Michie, 1994; Anevski et al., 2013).

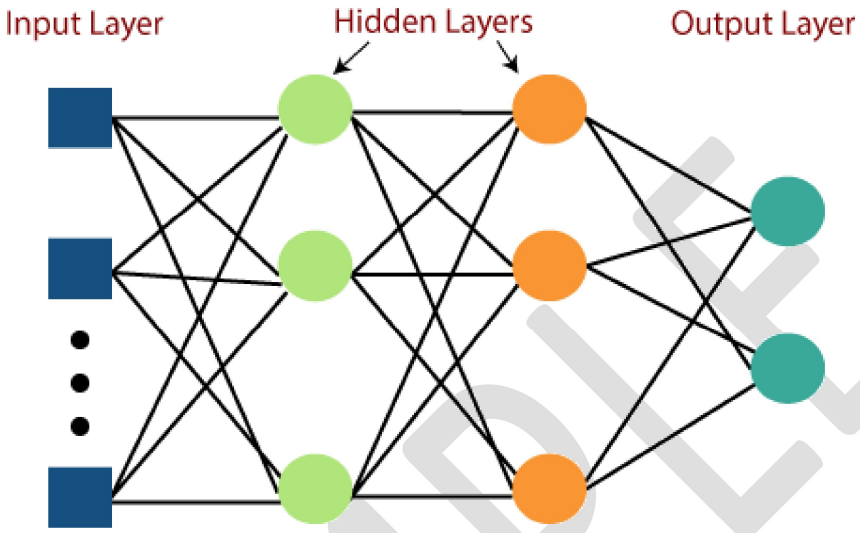


***Figure 6.11. Example of Multi-layer Perceptron in TensorFlow***

*Source: https://www.javatpoint.com/multi-layer-perceptron-in-tensorflow*

The problem interprets the total number of input-output units. There may be some ambiguity about which inputs to use, which will be explained later. Let's pretend for the time being that the inputs are chosen intuitively and are significant. The number of hidden units to be used is obvious. Using a single hidden layer with several units equal to half of the total number of input-output units can be a suitable place to start. Later, we'll go through how to pick an acceptable amount (Nilsson, 1982; Orlitsky et al., 2006).

#### 6.4.4.2. Training Multilayer Perceptron

The weights and criteria of the network must be established after the number of layers and units inside each layer have been determined to reduce the network's prediction error. This is done through the training algorithms. The historical cases are used to automatically control the weights and thresholds to reduce the mistake (Mohammad & Mahmoud, 2014). This approach is identical to fitting the network-defined model to the provided training data. Running all the training instances throughout the network and comparing the real output with the desired outputs can reveal the individual configuration's fault (Ayodele, 2010; Singh & Lal, 2013). With the use of an error function, the differences are pooled together to form the network error. The most well-known error functions are the following:

i. For regression situations, the sum squared error is employed. The discrete errors of the output nodes are squared and then added together as a total squared error.
ii. Cross-entropy functions are used for the maximum likelihood classification.

In traditional modeling approaches, such as linear modeling, it is likely that the design of the model will be determined algorithmically to decrease this mistake to an absolute minimum. The trade-off made in exchange for the non-linear modeling power of neural networks is that we can never be guaranteed that the error cannot be minimized any longer under any conditions (Jayaraman et al., 2010; Punitha et al., 2014).

The concept of error surface is one that is interesting to explore in this context. It is necessary to engage each one of the N weights and network starting points in order to be the dimension in space. The network error is often represented as the N+1th dimension. Any possible configuration of the weights can result in the error being intrigued in the N+1th dimension, resulting in the formation of an error surface. Network training is primarily concerned with finding the lowest point on a multidimensional surface, which is the primary goal. The error surface of a linear model with a sum squared error function is a parabola in the case of the sum squared error function. In other words, the error surface is shaped like a bowl with a single minimum value. As a result, determining the bare minimum is straightforward (Kim & Park, 2009; Chen et al., 2011; Anuradha & Velmurugan, 2014).

In the case of neural networks, the error surfaces are more complicated and are frequently characterized by a large number of non-supportive properties such as:

i. Local Minima
ii. Flat-spot and plateau
iii. Saddle-points
iv. Long narrow ravine

As a result, it is unlikely that an analytical determination can be made as to where the general minimum of error surface is located; hence, the training of neural networks involves an exploration of the error surface. The training algorithms attempt to find the global minimum step by step, starting with a basically arbitrary setup of the weights and thresholds and progressing from there. Typically, the gradient of the error surface is considered at the most recent position and is then employed to make the downward maneuver. A final stop is reached by the algorithm at the lowest position, which may be either the local minimum or the global minimum (Fisher, 1987; Lebowitz, 1987; Gennari et al., 1988**)**.

#### 6.4.4.3. Back Propagation Algorithm

The backpropagation technique is the best example of neural network training (Haykin, 1994; Patterson, 1996; Fausett, 1994). Conjugate gradient descent and the Levenberg-Marquardt (Bishop, 1995; Shepherd, 1997) are two contemporary second-order algorithms that are significantly faster to use in

many tasks. The backpropagation method still offers numerous advantages in various scenarios and can be the easiest algorithm to understand. This algorithm will only be introduced; more complex algorithms will be explored later. The slope vector is calculated in the backpropagation technique. The slope vector from the present position points to the steepest descent line. The mistake will be reduced if we move a modest distance along the slope vector. A series of such maneuvers will eventually yield a minimum of some sort. The difficult part is deciding how big the step should be (Michalski & Stepp, 1982; Stepp & Michalski, 1986).

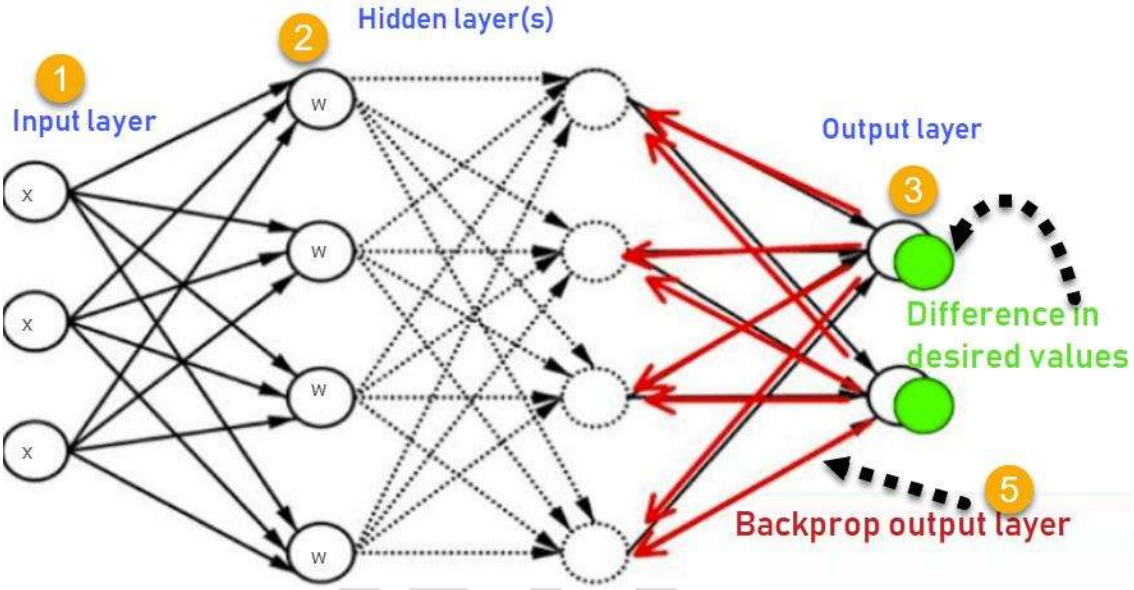


***Figure 6.12. Backpropagation Algorithm Working flowchart***

*Source: https://www.guru99.com/backpropogation-neural-network.html*

Large steps may be taken fast, yet these steps may be in excess of the solution or on the incorrect route altogether. One illustration of this is the circumstance where the algorithm goes extremely slowly over a steep and narrow valley, reflecting from one side to the other. Small steps, as compared to large leaps, may move the needle in the right direction, but these steps will require a number of rounds. The step size is proportional to the slope and the rate of acquisition. The specific setting for the particular constant learning rate is always dependent on the application and is typically determined by the experimenter during the experiment. It is also possible that the learning rate is time-varying and that it decreases with the expansion of the algorithm (Knill & Richards, 1996; Mamassian et al., 2002).

The addition of a momentum component to the algorithm is widely used to increase its performance. This accelerates the travel in a specific direction, and if a sufficient number of steps are taken in that direction, the algorithm becomes more efficient. It is because of this speed that the algorithm is capable of escaping the local minimum on occasion, as well as moving fast over the flat patch and plateau (Weiss & Fleet, 2002; Purves & Lotto, 2003).