2. MACHINE LEARNING

This Chapter begins with a brief introduction about machine learning, then some of the supervised learning techniques adopted in this work for pattern classification are presented. Naïve bayes classifier, Decision Tree Rule Induction, Artificial neural network and Support Vector Machine are explained in detail. Ensemble learning and LibD3C classifier is also explicated.

Machine learning technique discovers patterns and trends from diverse datasets, expecting that machines can acquire knowledge from observed data, and aids people to explain and categorize their knowledge. To elucidate new unseen data, machine-learning technique embodies a wide range of procedures for discovering the rules, patterns and relationships in sets of data that result in a generalization of these relationships. The output of a learning scheme is some form of structural description of a dataset, acquired from examples of given data that summarize the knowledge learned by the system and can be represented in various means [65].

Machine learning techniques can be applied in the fields such as machine perception, computer vision, syntactic pattern recognition, natural language processing, detecting credit card fraud, search engines, medical diagnosis, bioinformatics, brain-machine interfaces, cheminformatics, game playing, stock market analysis, classifying DNA sequences, speech and handwriting recognition, object recognition in computer vision, software engineering and robot locomotion. Machine learning provides the basis of data mining which is used to extract information from the raw data in databases, information that is expressed in a comprehensible form and can be used for a variety of purposes [66].

Machine learning involves in searching a very large space of possible hypotheses to decide which the best fits on the observed data. Machine learning algorithms have proven to be of great practical value in a variety of application domains. They are especially useful in (a) data mining problems where large databases may contain valuable implicit regularities that can be discovered automatically (b) poorly understood domains where humans might not have the knowledge needed to develop effective algorithms and (c) domains where the program must dynamically adapt to changing conditions. Machine learning retrieves ideas from a various set of disciplines, that includes artificial intelligence, control theory, probability and statistics, computational complexity, philosophy, information theory, psychology and neurobiology [67].

A well-specified task, performance metric, and source of training experience is required by a well-defined learning problem. Design choices involved in designing a machine learning approach are (a) to choose the type of training experience, (b) the target function to be learned, (c) a representation for this target function, (d) an algorithm for learning the target function from training examples.

Inductive and deductive are the two types of learning where, inductive learning methods extract rules and patterns out of massive data sets. Learning/training and prediction are the core tasks of machine learning. The primary goal is to automatically, acquire effective and accurate model from the training data that provides the domain knowledge which are the characteristics of the domain from which the examples are drawn. When the training set is larger high quality of model is generated [68, 69].

One of the phase in machine learning is prediction, wherein the set of inputs is mapped into the corresponding target values. Creating a model, with good predictive performance on test data is the main challenge of machine learning. Machine learning algorithms are classified into supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning. The function of a supervised learning algorithm is to maps inputs to desired outputs. Each example is associated with a label. If the label is discrete, then the task is classification. If the label is a real value then the task becomes regression problem. The target function is used to predict the label for a new case. Hence, learning is not only a task of remembering examples in the training set but also of generalization of unseen cases [70].

In unsupervised learning the patterns are mined without any explicit description of the target labels. Semi-supervised learning combines both examples to generate an appropriate function. Reinforcement learning learns a rule that where the actions are made based on the observations specified. Transduction predict outputs based on training inputs, training outputs, and test inputs like supervised learning but an explicit model is not constructed.

The primary goal of machine learning research is to develop general purpose algorithms of practical value. The amount of data that is required by the learning algorithm is a precious resource in the context of learning. The reason is that machine learning algorithms are data driven, and are able to examine large amounts of data.

Supervised Learning for Classification

Supervised learning concentrates on modeling input or output relationships which learns automatically. The goal of supervised learning is to identify an optimal functional mapping between the input data X, to the output variable i.e., a class label Y such that $Y = f(X)$. This is performed based on a sample of observations of the input variables X, which are the characteristics of the examples for a given problem.

The task of supervised learning is to built a classifier with a set of classified training samples. A pair consisting of an object and its associated class label is called as a labeled example. The set of labeled examples provided to the learning algorithm is called the training set. The classifier is constructed based on training data supplied to the classification algorithm. The key challenge of the supervised learning algorithm is generalization i.e., the ability to predict the correct label on unseen data.

The performance of the classifier is evaluated by employing a different set of labeled examples called the test set. The reason for using a separate test set for evaluating the classifier is that the most learned classifiers can accurately predict the class label of the training examples, not the new examples. Hence, it is more appropriate to use a different data set for testing the ability of the learned model to generalize new data points. The percentage of correctly classified test examples is called as the classification rate or prediction accuracy. The classification rate estimate is more accurate when the test dataset is larger.

Machine Learning (ML) divides classification tasks onto binary, multi-class, multi-labeled, and hierarchical tasks. In binary, the input is to be classified into one of two non-overlapping classes. In the binary classification, multi-class categorization can be objective or subjective, well-defined or ambiguous. Assigned categories can be objective, independent of manual evaluation or subjective, dependent on manual evaluation. Classes can be well-defined, ambiguous, or both. In Multi-class, the input is to be classified into one of several non-overlapping classes.

In Multi-labeled case, the input is to be classified into several of non-overlapping classes. Classification of functions of yeast genes, identifying scenes from image data or text-database alignment and word alignment in machine translation are some of the examples. Multi-label classification methods are evaluated on OHSUMED, a collection of medical references in text mining domain that holds medical information. Hierarchical, problem reports on relations among

categories which includes their structure into learning targets. The input is to be classified into one and the classes are divided into subclasses or grouped into super classes. The hierarchy is defined and cannot be changed during classification. Text classification and bioinformatics supply many examples such as protein function prediction.

2.1 Supervised Learning Algorithms

An assortment of supervised algorithms was used for solving classification problems. Some of the supervised learning algorithms used in building muscular dystrophy disease identification problem here are Decision tree, Naïve Bayes classifier, Support vector machine and Artificial Neural network. These algorithms are described in this chapter.

2.1.1 Decision tree Classifier

Decision tree classifier is the essential representative in the category of machine learning techniques. The process of learning decision trees from the labeled training examples is termed as Decision tree learning. The output of a decision tree classification algorithm is a binary tree like structure called a decision tree, where the internal node specifies a test on an attribute, each branch denotes an outcome of the test and a class label is located at each leaf node or terminal node. The root node is the topmost node in a tree. The target variable is predicted based on the rules in decision tree model. The class label of a new instance is predicted by testing the attribute values of the instance against the decision tree. A path is traversed from the root to a leaf node, which gives the class label of that data. Decision trees can be easily transformed into classification rules [71].

Tree Induction Algorithm: Input D – set of training instances

- The tree starts with a single node N, representing the training instances in D
- If all the instances in D belong to the same class, then node N becomes a leaf and is labeled with that class and the procedure is terminated. Otherwise an attribute ‘A’ is selected using attribute selection measure based on the splitting criterion and the node N is labeled with the splitting condition. A branch is grown from the node N for each of the decisions of the test condition
- The instances in D are partitioned accordingly

- Apply the algorithm recursively to each of the subsets D_i of D to form a decision tree such that the partitions are as pure as possible.
- The recursive partitioning stops when one of the following conditions is satisfied.
  (i) All the instances in a partition D_i belong to the same class
  (ii) There are no remaining attributes on which the node N may be further partitioned. Convert the node N into a leaf node and label it with most common class of the instances in D_i.
  (iii) A partition D_i is empty. A leaf node is created with the majority class in D_i.

An attribute selection measure is used for selecting the splitting criterion that splits a given data set D, of labeled training instances into individual classes. If D is divided into smaller partitions based on the outcomes of the splitting condition, then all the instances in a given partition would belong to the same class. The manner in which a given node is split are determined with the splitting rules which are also known to be attribute selection measures.

Ranking for each attribute is provided with the attribute selection measure [72]. The splitting attribute for the given training set is chosen with the attribute having the highest rank. If the splitting attribute is continuous-valued, then the split point must be determined as a part of the splitting process. The node N created for a partition D is labeled with the splitting criterion, branches fan out for each outcome of the condition and the instances are partitioned accordingly. There are three possible cases. Let A be the splitting attribute and A has v distinct values, a_1, a_2, ..a_v,

- If A is discrete valued, branches are created for each value a_j of A at node N and labeled with that value. Partition D_j corresponds to the subset of labeled instances in D having value a_j for A. If all the instances in a given partition have the same value for A, then the attribute A need not be considered again for partitioning of the instances.
- If A is continuous-valued, A ≤split-point and A >split-point are the two possible outcomes when testing at node N which has corresponding to the condition. Normally, the split-point, 'a', is taken as the midpoint of two known adjacent values of A and therefore it may not be a pre-existing value of A from the training data. Two branches are grown from N one for each condition A≤split-point, A>split-point and the resultant nodes are labeled accordingly. The

instances are partitioned such that D_1 holds the partition of instances in D for which A≤split-point, whereas D_2 holds the remaining.

- If A is discrete and binary valued, two branches are grown from N. The left branch of N is labeled as 'yes' such that D_1 corresponds to the partition of instances in D that satisfy the test. The right branch of N is labeled 'no' so that the partition D_2 contains instances of D that does not satisfy the test.

An example decision tree is shown in Fig.2.1.

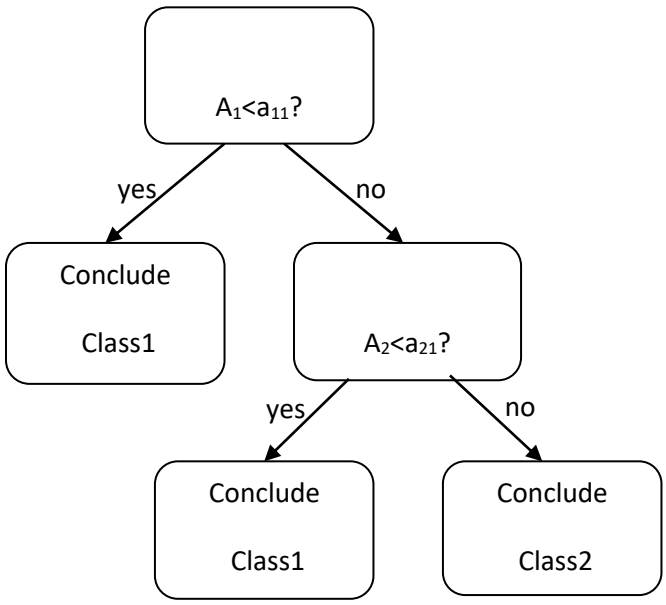

Fig.2.1 Example Decision Tree

There are three attribute selection measures namely information gain, gain ratio and gini index basically used in decision tree construction. The tree induction algorithm ID3 uses information gain as its attribute selection measure.

Information Gain

Splitting attribute for node N is chosen with the highest information gain, which minimizes the information that is needed to classify the instances in the resulting partitions. Such an approach

minimizes the expected number of tests needed to classify a given instance and guarantees that a simple tree is found. The expected information needed to classify an instance in D is given by

$$Info(D) = - \sum_{i} p_i \log_2 p_i$$

(2.1)

where p_i is the probability that an arbitrary instance in D belonging to class C_i. Info(D) is also known as the entropy of D. If D is partitioned into D_1, D_2 … D_v based on v distinct values a_1, a_2, a_3…a_v of an attribute A, then D_j contains those instances in D that have value a_j and these partitions correspond to the branches grown from node N. This partitioning produces the exact classification of the instances. The information needed to arrive at an exact classification after partitioning based on attribute A is given by

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \text{ x } Info(D_j)$$

(2.2)

The purity of the partitions is higher when the expected information required is smaller. The difference between the original information requirement and obtained after partitioning on A is termed as information gain. Thus the gain is

$$Gain(A) = Info(D) - Info_A(D)$$

(2.3)

Domain knowledge is not required in the construction of a decision tree and hence it is more suitable for extracting the hidden predictive knowledge. The algorithm also scales well in case of high dimensional data. The learning and classification phases of decision tree induction are simple and fast. In general, decision tree classifiers show good accuracy. The speed of this learning algorithm is reasonably high, as is the speed of the resulting decision tree classification system.

2.1.2 Naïve Bayes Classifier

The Naïve Bayes classifier (NB) is an effective classifier which is used in number of applications such as natural language processing and information retrieval. The Naïve Bayes Classifier technique works based on Bayesian theorem, which is suits in the environments where the dimensionality of the inputs is more. The effect of a variable value on a given class is independent of the values of other variable in Naïve Bayes classifiers. The Naïve-Bayes inducers compute conditional probabilities of the classes when the instance is given and the class with the highest posterior is picked up. Naïve Bayes classifiers can be trained very efficiently in a supervised learning setting, depending on the precise nature of the probability model.

A Naïve Bayes classifier is a simple probabilistic classifier based on Bayes theorem with strong independence assumptions. Below given is the formulation of Naïve bayes classifier,

$$Given\ classes\ \omega_j\ and\ dataset\ x$$

$$P(\omega_j \mid x) = \frac{p(x \mid \omega_j)P(\omega_j)}{p(x)}$$

$$where$$

$$p(x) = \sum_j p(x \mid \omega_j)P(\omega_j)$$

(2.4)

Bayesian classification model is given by,

- The *prior* probability reflects knowledge of the relative frequency of instances of a class
- The *likelihood* is a measure of the probability that a measurement value occurs in a class
- The *evidence* is a scaling term

$$posterior = \frac{likelihood\ x\ prior}{evidence}$$

(2.5)

The classification of unseen data x is performed by calculating P (C_i / x) for each class and assigning **x** to class i if P (C_i / x) > P (Cj / x) for all i ≠ j [73].

The probability model for a classifier is a conditional model on several feature variables F1 through Fn is given by p(C | F1,….Fn) over a dependent class variable C with a small number of outcomes or classes. If the number of features is large or if a feature can take on a large number of values, then arises an issue where basing such a model on probability tables is not feasible and therefore the model is reformulated as below,

$$p(C|F_1,\ldots,F_n) = \frac{p(C)\ p(F_1,\ldots,F_n|C)}{p(F_1,\ldots,F_n)}. \tag{2.6}$$

Using Bayesian Probability terminology, the above equation can be written as

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}. \tag{2.7}$$

Building a classifier from the probability model

The naive Bayes classifier combines this probability model with a decision rule such as the maximum posteriori or MAP decision rule that picks up the hypothesis, which is most probable. The corresponding classifier, a Bayes classifier, is the function defined as follows:

$$\text{classify}(f_1,\ldots,f_n) = \operatorname*{argmax}_c\ p(C=c)\prod_{i=1}^{n} p(F_i=f_i|C=c). \tag{2.8}$$

Parameter estimation and event models

A class prior is calculated by assuming equally probable classes such as priors = 1 / (number of classes), or by calculating an estimate for the class probability from the training set

prior for a given class = (number of samples in the class) / (total number of samples)

To estimate the parameters for a feature's distribution, assume a distribution or generate nonparametric models for the features from the training set. The event model of the Naive Bayes classifier are the assumptions on distributions of features. For discrete features encountered in document classification, multinomial and Bernoulli distributions are popular.

In naive Bayes classifiers, every feature plays a role in determining the label to a given input value. To select a label for an input value, the naive Bayes classifier begins by computing

the prior probability of each label that is determined by examining frequency of each label in the given training set. The impact of each feature is then combined with this prior probability to estimate likelihood for each label. The highest likelihood estimate is then assigned to the input value.

To create a labeled input, the model a label is chosen first and each of the input's features is generated based on that label. Every feature is expected to be entirely independent of every other feature, given the label. Based on this assumption, calculate an expression for P(label|features), the probability that an input will have a particular label given that it has a particular set of features.

$$P(\text{label}|\text{features}) = P(\text{features}, \text{label})/P(\text{features})$$

In the next case, P(features) will be the same for every choice of label, and it simple for finding the most likely label, it serves to calculate P(features, label), which is called as the label likelihood.

$$P(\text{features}) = \Sigma l \text{ in}| \text{ labels } P(\text{features}, \text{label})$$

The label likelihood is extended out as the probability of the label times the probability of the features given the label

$$P(\text{features}, \text{label}) = P(\text{label}) \times P(\text{features}|\text{label})$$

The features are all independent of one another separate the probability of each individual feature:

$$P(\text{features}, \text{label}) = P(\text{label}) \times \text{Prod}f \text{ in}| \text{ features} P(f|\text{label})$$

2.1.3 Artificial Neural Network

Artificial Neural network is one of the supervised learning algorithm that analyzes the training data and produces an inferred function, which is called a classifier. The classifier is then used for predicting the accurate output value for any valid unseen input object.

An Artificial Neural Network is a computational model that is stimulated by the biological nervous system. Dendrites, soma, axon and synapses are the four components of biological neuron that receive signals from other neurons. Synapses regulate the signals and forward the signals to axon. Axon directs signals away from the cell body to other neurons. The biological neuron structure is shown in Fig.2.2.

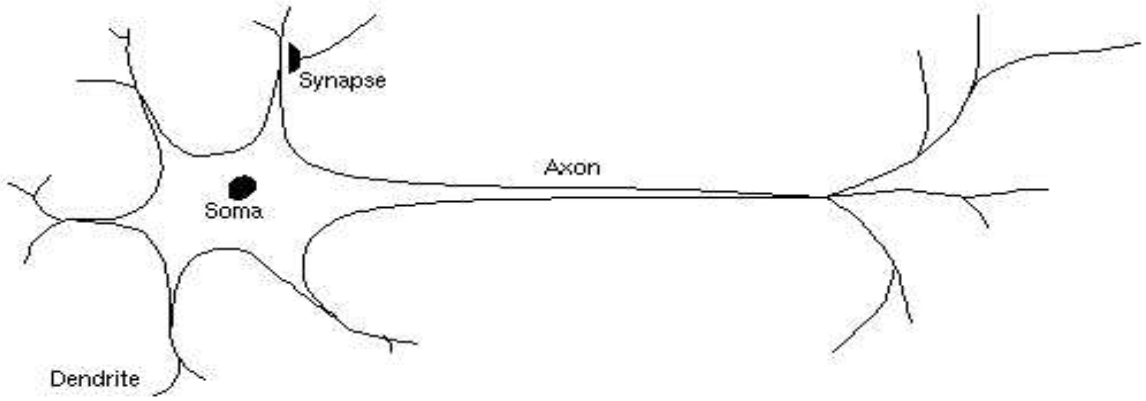

Fig.2.2 Biological Neuron

The artificial neuron inspired from biological neuron is represented in Fig 2.2.

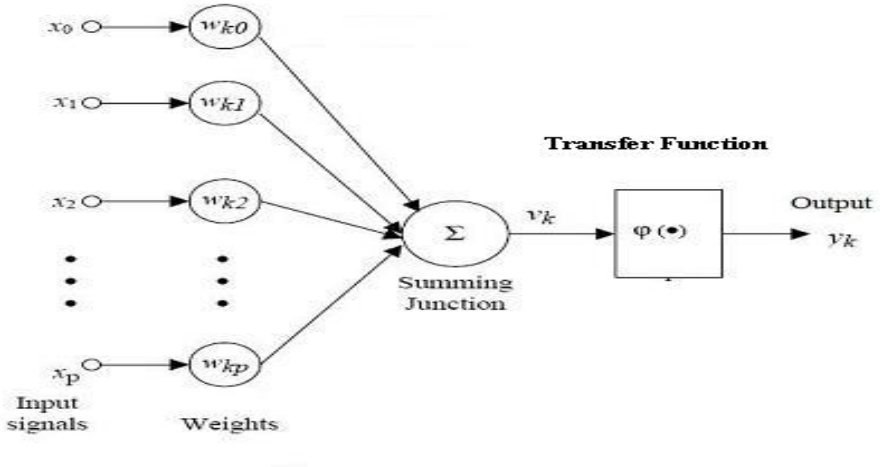

Fig.2.3 A Perceptron

In artificial neuron, the inputs that are the features are received using dendrites. As synapses modulate in biological neurons, the inputs are multiplied by its weight and the product is cumulated and fed into the transfer function to produce the output.

Multiplayer Perceptron

Multilayer Perceptron network (MLP) is the general purpose, flexible, nonlinear models which consists of a number of units organized into multiple layers. By varying the number of layers and the number of units in each layer the complexity of the MLP network is reduced. With sufficient hidden units and enough data, it has been shown that MLPs can approximate virtually any

function to any desired accuracy. MLPs are appropriate in problems when knowledge about the form of the relationship between input vectors and their corresponding outputs is not known explicitly.

Activation function

If a multilayer perceptron has a linear activation function in all neurons, then it is proved with linear algebra that number of layers can be reduced to the standard two-layer input-output model. Multilayer perceptron is different by, each neuron uses a nonlinear activation function which was developed to model the frequency of action potentials. This function is modeled in several ways, but must always be normalizable and differentiable.

At present, the applications uses the two main activation functions where both are sigmoid, and are described by

$$\phi(y_i) = \tanh(v_i) \quad \text{and} \quad \phi(y_i) = (1 + e^{-v_i})^{-1} \tag{2.9}$$

The first function is a hyperbolic tangent that ranges from -1 to 1, and the second one is the logistic function, ranges from 0 to 1. y_i is the output of the ith node and v_i is the weighted sum of the input synapses. include radial basis functions is more specialized activation function that are used in another class of supervised neural network models.

Layers

The multilayer perceptron consists of three or more layers such as an input and an output layer with one or more hidden layers of nonlinearly activating nodes. Each node in one layer connects with a certain weight w_{ij} to every node in the next layer.

Learning through back propagation

The perceptron learns the inputs by changing connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result. This is an example of supervised learning, and is carried out through backpropagation, a generalization of the least mean squares algorithm in the linear perceptron.

It represent the error in output node $j$in the nth data point by $e_j(n) = d_j(n) - y_j(n)$, where $d$is the target value and $y$is the value produced by the perceptron. Corrections to the weights of the nodes are made and based on those corrections the error is minimized in the entire output and is formulated as,

$$\mathcal{E}(n) = \frac{1}{2}\sum_{j} e_j^2(n) \tag{2.10}$$

Using gradient descent, the change in each weight will be

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial v_j(n)} y_i(n) \tag{2.11}$$

Where y_i is the output of the previous neuron and η is the learning rate, which is carefully selected to ensure that the weights converge to a response fast enough, without any fluctuations. This parameter normally ranges from 0.2 to 0.8 in programming applications.

2.1.4 Support Vector Machine

Support vector machines (SVM) are learning systems that implements a learning bias derived from statistical learning theory which use a hypothesis space of linear functions in a high dimensional feature space, trained with a learning algorithm from optimization theory. Learning classification and regression rules from data are learned using the support vector machine training algorithm.

The geometrical elucidation of support vector classification is that the algorithm searches for the optimal separating surface, which is the hyper plane that is equidistant from the two classes and can be extended to multi-class problems. In order to construct non-linear decision surfaces kernel functions are introduced [74].

The techniques of mathematical programming and kernel functions are two key elements in the implementation of SVM. The parameters are found by solving a quadratic programming problem with linear equality and inequality constraints. The kernel functions used in case of non linear SVM are Polynomial, Gaussian, Neural networks.

Standard linear and non-linear Support vector Machine

Implementations of statistical inference principles are learning machine algorithms. Normally, the machine is presented with a set of training examples, (x_i, y_i) where the x_i is the real world data instances and the y_i are the labels indicating which class. For the two class pattern

recognition problem, $y_i = +1$ or $y_i = -1$. A training example (x_i, y_i) is called positive if $y_i = +1$ and negative otherwise.

An optimal hyper plane between two classes of examples is constructed by the support vector machine using the geometric point of view, which given by $w^T x - \gamma = 0$. A vector of weights **w** and a threshold value **γ** are the parameters and W is orthogonal to the hyper plane.

Margin is a quantity that separates hyper plane from each training example. $\eta_i = y_i (w^T x - \gamma)$ is the functional margin. The hyper plane correctly classifies the data point when $\eta_I > 0$. The geometric margin measures the distance of data points from the hyper plane in Euclidean space when the weight vector w of the functional margin is normalized ($w = w/||w||$). The maximum geometric margin over all possible hyper planes is referred using the expression margin of a training set. A maximal margin hyper plane is defined by the maximum geometric margin which has a unique hyper plane.

The two planes parallel to the hyper plane, if w is weight vector recognizing functional margin 1 on the positive point X^+ and on the negative point X^- , that passes through one or more points called bounding hyper planes and are given by,

$$w^T x - \gamma = 1$$

$$w^T x - \gamma = -1$$

Distance of the bounding plane, $w^T x - \gamma = 1$ from the origin is elucidated as $|- \gamma + 1|/||w||$ and the distance of the bounding plane, $w^T x - \gamma = -1$ from the origin is summarized as $|- \gamma - 1|/||w||$.

Support vectors are the points falling on the bounding planes that are vital in the theory. The data points x belonging to two classes A+ and A- are classified based on the condition

(2.14)

$$\mathbf{w}^T\mathbf{x}_i - \gamma \geq 1, \quad \text{for all } \mathbf{x}_i \in A^+$$
$$\mathbf{w}^T\mathbf{x}_i - \gamma \leq -1, \quad \text{for all } \mathbf{x}_i \in A^-$$
$$D_{ii}(\mathbf{w}^T\mathbf{x}_i - \gamma) \geq 1, \text{for all } \mathbf{x}_i$$

These disproportion constraints can be united to give, where $D_{ii} = 1$ for A^+ and $D_{ii}=-1$ for A^- · An optimal hyper plane is found as the learning problem <w, γ>, $w^T x - \gamma = 0$ which separates A^+ from A^- by maximizing the distance between the bounding hyper planes.

The above explained optimization problem is solved using Lagrangian duality and the Lagrangian is given as

$$L(\mathbf{w}, b, u) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} u_i \left[D_{ii}(\mathbf{w}^T\mathbf{x}_i - \gamma) - 1\right]$$

(2.15)

where u_i is the lagrange multiplier. The clarification to this quadratic programming problem is given by maximising *L* with respect to u and minimising with respect to w.

For a given u, the problem is solved using Lagrangian duality to find values of w and γ which minimizes $L(\mathbf{w}, \gamma, \mathbf{u})$. The decision function of the quadratic programming problem yields $u_i s$ and is given by

$$\text{sign}(\mathbf{w}^T\mathbf{x} - \gamma) = \text{sign}(\sum_{i=1}^{m} u_i d_i \mathbf{x}_i{}^T\mathbf{x} - \gamma)$$

(2.16)

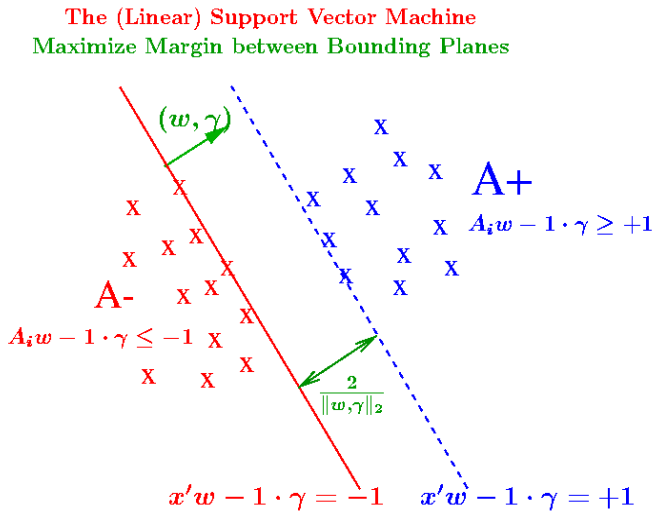

Fig. 2.4 Linear Support Vector Machine

Where γ is calculated for $u_i \neq 0$ and by holding a point on the margin. Corresponding to each data point $\mathbf{x}_i$, there is a dual variable u_i. Since, the decision function f(**x**) is a function of data points, most of those points of u_i s are zero and when the data points corresponding to nonzero u_i s then those are called support vectors.

SVM formulation with soft margin

The situation of noisy data is depicted in Fig.2.5, where it is tedious to identify a separating plane, where region A+, and A- are totally separated and poor generalisation occurs when training error is zero. The only probable way is to catch a maximally separating plane with a few number of points that fall on the wrong side. Error is the deviation of such a point from the respective bounding plane.

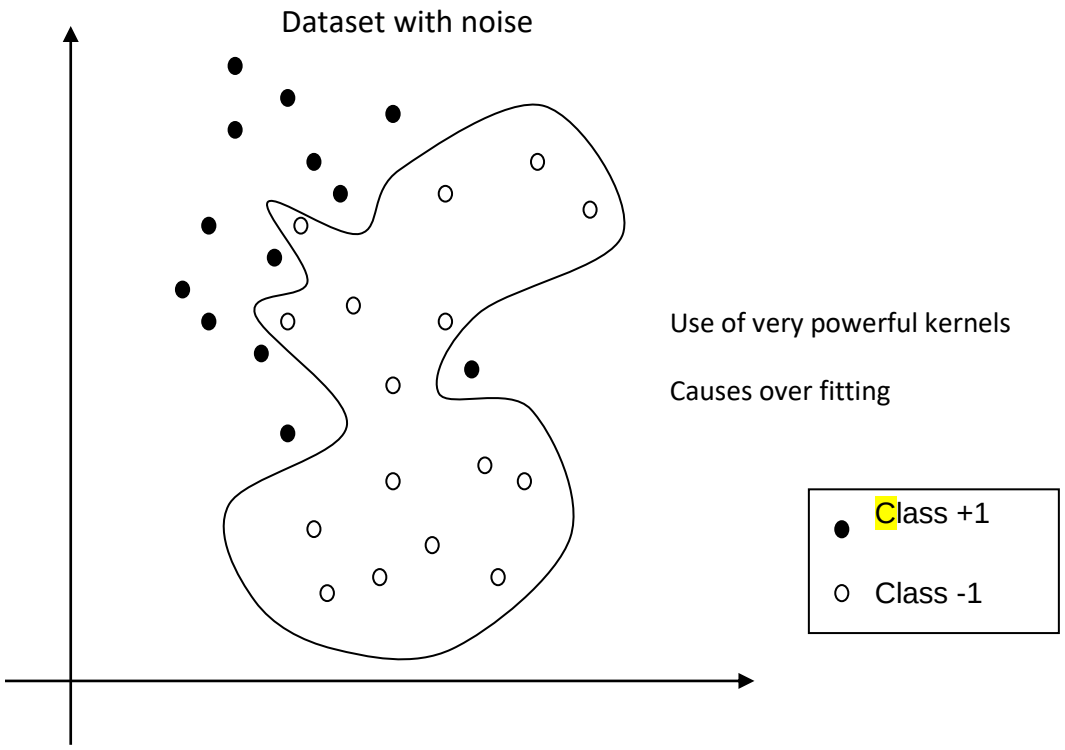

Fig.2.5 Data which require Linear SVM Formulation with Soft Margin

SVM looks for a classifier through training where the margin between the bounding planes is maximum and the number of points that contributes error is minimum. Error is caused with more points in the maximum margin. A control parameter C controls the two contradicting requirements using the weight assigned.

The amount of violation of the constraints introduced is measure using a vector of slack variables ξi and is termed as SVM with soft margin and the construction is as follows,

$$\min_{\mathbf{w},\gamma,\boldsymbol{\xi}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\xi_i$$
$$\text{subject to } d_i(\mathbf{w}^T\mathbf{x}_i - \gamma) + \xi_i - 1 \geq 0, \quad 1 \leq i \leq m$$
$$\xi_i \geq 0, \quad 1 \leq i \leq m$$

(2.17)

Non-linear SVM and Kernel Trick

The elucidation of SVM requires Q, d and C where Q is attained from $\boldsymbol{A}*\boldsymbol{A}^T$ and $\boldsymbol{d}*\boldsymbol{d}^T$.

$$\boldsymbol{AA}^T = \begin{bmatrix} \boldsymbol{x}_1{}^T\boldsymbol{x}_1 & \boldsymbol{x}_1{}^T\boldsymbol{x}_2 & . & \boldsymbol{x}_1{}^T\boldsymbol{x}_m \\ . & . & \boldsymbol{x}_i{}^T\boldsymbol{x}_j & . \\ . & . & . & . \\ \boldsymbol{x}_m{}^T\boldsymbol{x}_1 & \boldsymbol{x}_m{}^T\boldsymbol{x}_2 & & \boldsymbol{x}_m{}^T\boldsymbol{x}_m \end{bmatrix} = \boldsymbol{K}$$

The $(i, j)th$ element of $\boldsymbol{AA}^T$ is $\boldsymbol{x}_i{}^T\boldsymbol{x}_j$ and is called the linear kernel matrix which infers that the information necessary for training is taken in the form of dot products of the training vectors. A function $\phi(.)$ is used to map each data point $\boldsymbol{x}_i$ in to a higher dimensional space when the data is not linearly separable as shown if Fig.2.6, which tries to find a maximally separating hyper plane in that space as a classifier.

The SVM training requires $\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$ for all i and j when such a separation is done. The kernel trick is used for obtaining such dot products without explicitly mapping the data points into higher dimensional space. Training algorithm as same as that of the linear classifier, the problem of 'curse of dimensionality' is tackled in a simple way are the advantages of non-linear SVM classifiers.

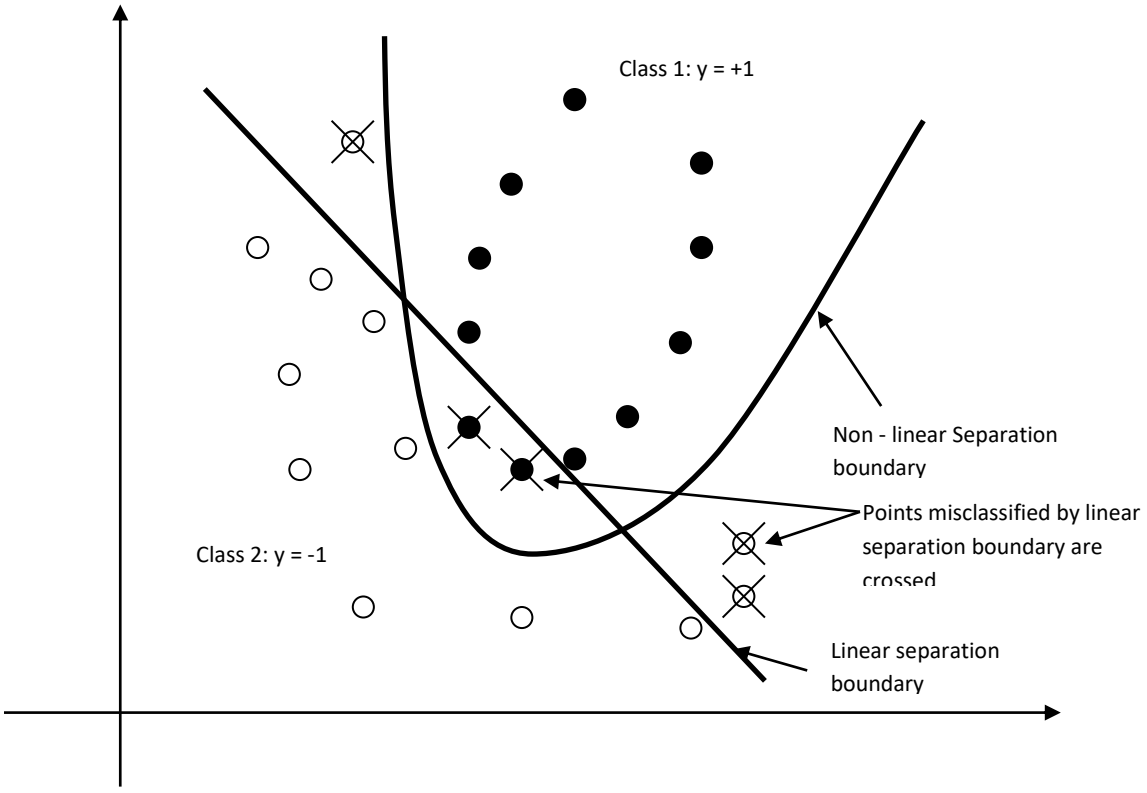

Fig.2.6 Data that require Non linear classifier

A function in input space is given by the Kernel function $K(x_i, x_j)$ and the advantage in using kernel function $K(x_i, x_j)$ is a mapping $\phi(x)$ is avoided. For given training data vectors in an input space, the required scalar products in a feature space $\phi(x_i)^T \phi(x_j)$, are considered directly by computing kernels $K(x_i, x_j)$. An SVM that operates in an infinite dimensional space can be constructed by using the chosen kernel $K(x_i, x_j)$. A function *K* is a kernel when,

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \tag{2.18}$$

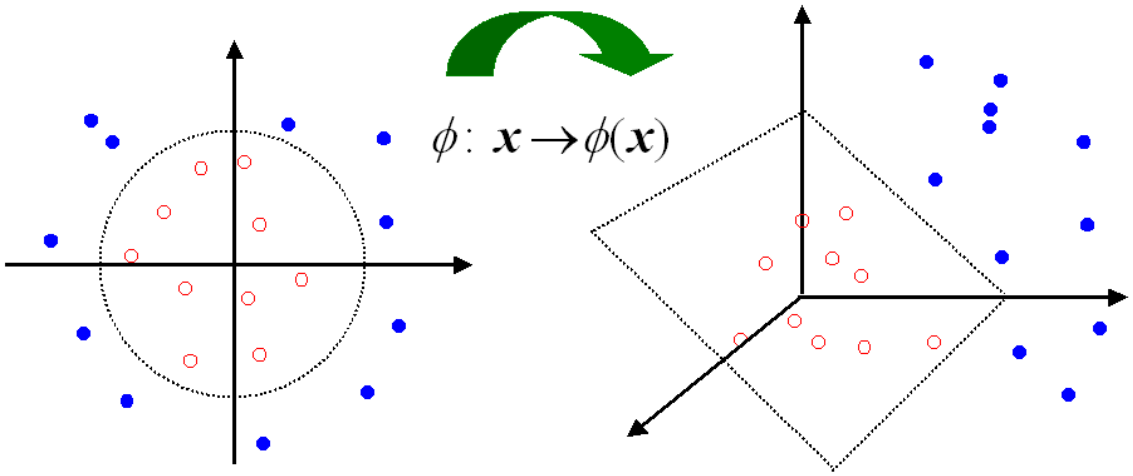

Fig.2.7 Non-linear Mapping into Feature space

Linear, Polynomial, Radial Basis Function (RBF) are the possible kernels and the simplest is a linear kernel is given as $K(x_i, x_j) = (x_i)^T (x_j)$. The polynomial kernel takes the form $K(x, y) = (x^T y + 1)^d$.

The RBF kernel is given by $k(x, y) = \exp(-\sigma \|x - y\|^2)$ where σ is a positive parameter controlling the radius. A dot product in a high-dimensional space is the kernel trick used by the Mercer's condition that states that any positive semi-definite kernel K (x, *y*) can be stated.

Normalization of kernels

The training of support vector machines very challenging when the number of variables is hefty, and the value of the kernel may become very little or too large [76]. Each variable and the kernel is normalized to overcome this issue. Variables can be normalized to the [0,1] or [-3 , +3] range by using the formula $\left(x_i - \mu_i\right)/\sigma_i$, where μ_i and σ_i are the mean and standard deviation of the i[th] predictor variable. The maximum value of the kernel is if each variable is normalized in the range [0, 1]. Maximum values $k(\boldsymbol{x},\boldsymbol{x})$ for a polynomial kernel is $\left(n+1\right)^p$ when data points has all its variable values as 1, then the normalized kernel values are found as $k(\boldsymbol{x},y) = \left(\frac{\boldsymbol{x}^T y+1}{n+1}\right)^p$, where the degree of the polynomial kernel p., the maximum value of $\|\boldsymbol{x} - \boldsymbol{y}\|^2$ is n for an RBF kernel. So, a normalized kernel may be computed as $k(\boldsymbol{x},\boldsymbol{y}) = exp(-\frac{\sigma}{n}\|\boldsymbol{x} - \boldsymbol{y}\|^2)$.

SVM Formulation of Non-linear Kernels with Soft Margin

The SVM formulation for Non-linear kernels with soft margin can be given as:

$$\min_{\mathbf{w},\gamma,\boldsymbol{\xi}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\xi_i$$
$$\text{subject to } d_i(\mathbf{w}^T\phi(\mathbf{x}_i) - \gamma) + \xi_i - 1 \geq 0, \quad 1 \leq i \leq m$$
$$\xi_i \geq 0, \quad 1 \leq i \leq m$$

(2.19)

The optimization problem can be solved using lagrangian duality and the dual problem in matrix form is written as below:

$$\text{Minimize } L_D(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T\boldsymbol{Q}\mathbf{u} - \mathbf{e}^T\mathbf{u}$$

(2.20)

$$\mathbf{d}^T\mathbf{u} = 0, \ \mathbf{0} \leq \boldsymbol{u} \leq C\boldsymbol{e}$$

Where $\boldsymbol{Q} = \boldsymbol{DKD}$ and K is kernel matrix. $\boldsymbol{Q}$ can be computed as $\boldsymbol{Q} = \mathrm{K} \bullet *(\boldsymbol{d} * \boldsymbol{d}^{\mathrm{T}})$

For linear SVM, the dual formulation is:

$$\begin{aligned} &\min_{\boldsymbol{u}} L(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^T \boldsymbol{D}(\boldsymbol{A}\boldsymbol{A}^T + \frac{\boldsymbol{I}}{C})\boldsymbol{D}\boldsymbol{u} - \boldsymbol{e}^T\boldsymbol{u} \\ &\text{subject to} \quad \boldsymbol{d}^T\boldsymbol{u} = 0 \\ &\qquad\qquad\quad \boldsymbol{u} \geq \boldsymbol{0} \end{aligned} \tag{2.21}$$

So for a non-linear kernel, the dual formulation is:

$$\begin{aligned} &\min_{\boldsymbol{u}} L(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^T \boldsymbol{D}(\boldsymbol{K} + \frac{\boldsymbol{I}}{C})\boldsymbol{D}\boldsymbol{u} - \boldsymbol{e}^T\boldsymbol{u} \\ &\text{subject to} \quad \boldsymbol{d}^T\boldsymbol{u} = 0 \\ &\qquad\qquad\quad \boldsymbol{u} \geq \boldsymbol{0} \end{aligned}$$

Or,

$$\begin{aligned} &\min_{\boldsymbol{u}} L(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^T \boldsymbol{Q}\boldsymbol{u} - \boldsymbol{e}^T\boldsymbol{u} \\ &\text{subject to} \quad \boldsymbol{d}^T\boldsymbol{u} = 0 \\ &\qquad\qquad\quad \boldsymbol{u} \geq \boldsymbol{0} \end{aligned}$$

$\boldsymbol{Q}$ can be computed as $\boldsymbol{Q}=(\boldsymbol{K} + \boldsymbol{I}/C).*(\boldsymbol{d} * \boldsymbol{d}^T)$

(2.22)

Multiclass Support Vector Machine

Multi Classification is a typical data mining task, with origins in machine learning which categorize the data into a set of known classes. A set objects whose class label is known and a classification model is constructed based on the features of data in the training set. A combination of classification rules is generated from the classification model, which can be used to classify future data and develop a better understanding of each class in the database.

Multiclass classification is applied in many real world applications. The formulation to solve multi-class SVM problems in one step has variables proportional to the number of classes. Therefore, for either multiclass SVM methods, several binary classifiers have to be constructed or a larger optimization problem is needed. Hence, in general it is computationally more

expensive to solve a multi-class problem than a binary problem with the same number of data. Up to now experiments are limited to small data sets.

SVM is basically a binary classifier where the formulation of SVM was based on a two-class problem. To regulate the K-class pattern classification problem, several different schemes can be applied to the simple SVM algorithm. The K-class pattern classification problem is given below:

- For i=1… *l* is a feature vector of length d and the class label for data point.
- Find a classifier with the decision function, *f(x)* such that *y=f(x,* where y is the class label for x.

The performance of the classifier is assessed in terms of the total classification error over a set of testing data that is similar to that of the binary classifier.

The schemes which have been used for solving the multi-class problem are as listed below:

- Using *k* one-to-rest classifiers.
- Using *k(k-1)/2* pair wise classifiers with one of the voting scheme listed below:
  - Majority voting
  - Pair wise coupling
- Extending the formulation of SVM to support the k-class problem.
  - Construct the decision function by considering all classes at once.
  - Construct a decision function for each class by only considering the training data points belong to that particular class.

2.2 Ensemble learning

Learning algorithms that build a set of classifiers and categorizes new data points with a weighted vote of their predictions are termed as Ensemble methods. In order to create a stronger overall prediction, ensemble learning combines multiple predictions that is derived from various techniques [76]. For example, the combination of predictions of a random forest, a support vector machine, and a simple linear model will create a stronger final prediction set. Models can be varied from each other for a range of reasons, from the population they are built upon to the modeling used for building the model. are Difference in population, Difference in hypothesis, Difference in modeling technique, Difference in initial seed are the top four reasons for a model to be distinct.

Three fundamental reasons that represent an ensemble work better than a single classifier are Statistical, Computational and Representational [77]. Statistical is the first reason where, to identify the best hypothesis in the space, a learning algorithm can be viewed as searching a space H of hypotheses. This problem arises when the amount of training data available is too small compared to the size of the hypothesis space. The algorithm can average their votes by constructing an ensemble out of all of the precise classifiers and risk is reduced by choosing the wrong classifier which is depicted in Fig.2.8. The hypothesis space H is denoted with the outer curve. The set of hypotheses on the training data is given in the inner curve. The point labeled f is the true hypothesis, and it is shown that by averaging the accurate hypotheses.

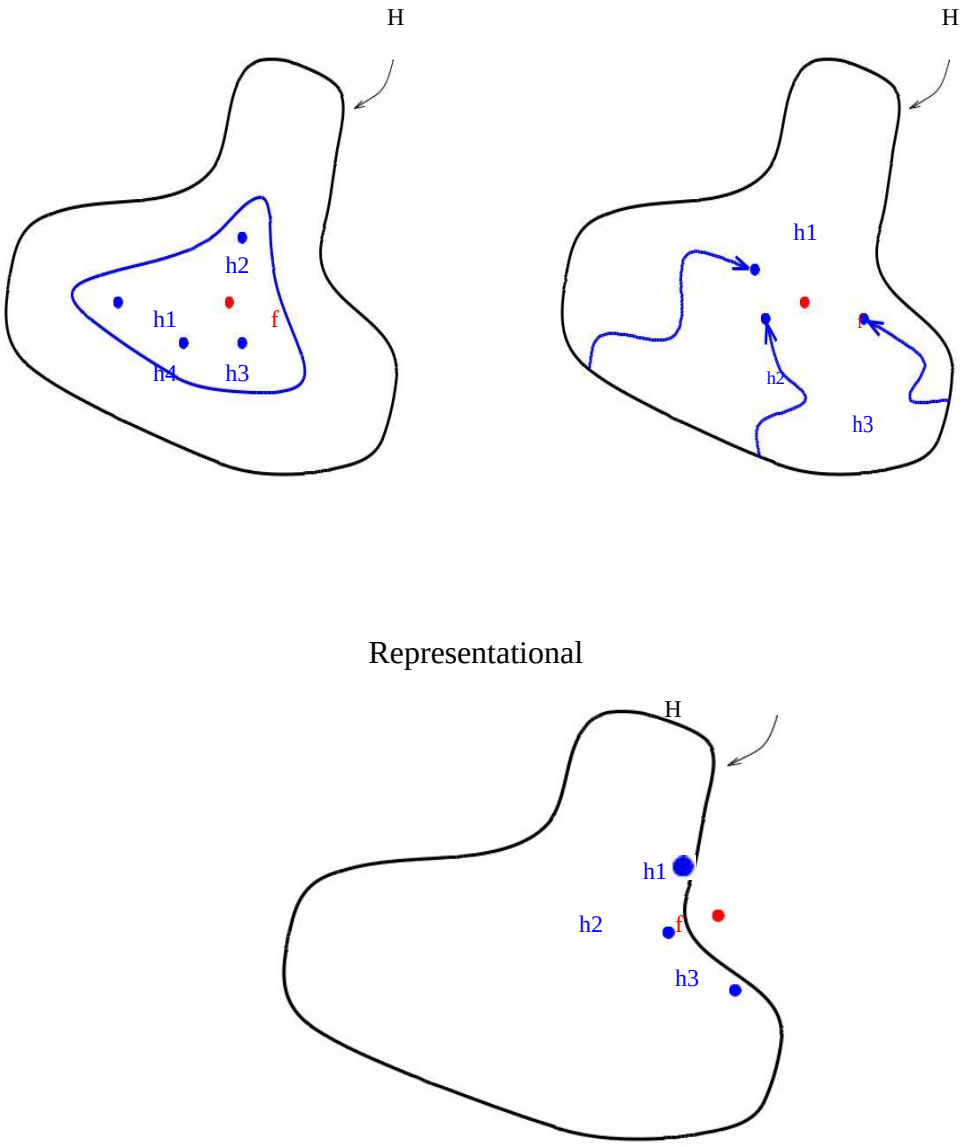

Fig.2.8 Modeling Techniques of Ensemble Classifier

The next reason is computational where many learning algorithms are worked by executing local search that may be caught in local optima. To minimize an error function over the training data, Neural Network algorithms employ gradient descent and greedy splitting rule is employed by decision tree algorithms to raise the decision tree. in cases when sufficient training data is not exists, the statistical problem will be absent. An enhanced approximation to the accurate unknown function is constructed by running the local search from many different starting points. The third reason is representational, where in maximum applications of machine learning the true

function f cannot be represented by any of the hypotheses in H, by forming weighted sums of hypotheses drawn from H it may be possible to expand the space of representable functions.

The three most popular methods for combining the predictions from different models are:

Bagging: Bagging, finds a average of all the predictions and tries to implement similar learners on small sample populations. In simplified bagging, to reduce the variance error different learners can be used on different population. The term bagging means bootstrap aggregation.

Working of bagging for a Given set, D, of d tuples. For iteration i (i = 1, 2,..., k), a training set, Di , of d tuples is sampled with replacement from the original set of tuples, D. Each training set is a bootstrap sample because sampling with replacement is used, some of the original tuples of D may not be included in Di , and some may occur beyond once.

For each training set, Di a classifier model, Mi is learned. Each classifier, Mi, returns its class prediction, to classify an unknown tuple, X. Bagging can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple. The algorithm is summarized below.

Algorithm:

The bagging algorithm—create an ensemble of models (classifiers or predictors) for a learning scheme where each model gives an equally-weighted prediction.

Input: D, a set of d training tuples;
k, the number of models in the ensemble;
a learning scheme (e.g., decision tree algorithm, backpropagation, etc.)
Output: A composite model, M*.
Method:
for i = 1 to k do // create k models:
create bootstrap sample, Di, by sampling D with replacement;
use Di to derive a model, Mi;
end for
To use the composite model on a tuple, X:
if classification then
let each of the k models classify X and return the majority vote;
if prediction then
let each of the k models predict a value for X and return the average predicted value;

The accuracy attained by the bagged classifier is pointedly larger than a single classifier derived from the original training data D. The accuracy is increased occurs because the composite model reduces the variance of the individual classifiers. It was proven that a bagged predictor will always have better accuracy against a single predictor that is derived from D, for predicting a variable.

Boosting: Boosting works on iterative method where the weight of an observation is adjusted based on the classification that is made recently. When an incorrect classification is made, it tries to increase the weight of this observation and vice versa. Boosting in general builds strong predictive models and the bias error is decreased.

Weights are assigned to each train a tuple to get work with boosting algorithm. A sequence of k classifiers is iteratively learned and after a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to pay more attention to the training tuples that were misclassified by M_i. The weight of each classifier's vote is a function of its accuracy and the final boosted classifier, M*, combines the votes of each individual classifier. The boosting algorithm can be extended for the prediction of continuous values. In the classification of tuples, the widely used boosting algorithm is AdaBoost.

Stacking: Stacking is used to combine models where a learner is used to combine output from different learners. Depending on the combining learner, bias or variance error can be decreased.

An ensemble with two techniques that are alike will not perform well such as Bayesian model combination and stacking, that attempts to weight the models before combining them. The hybrid approach has been an emerging research area in machine learning, to gain a better accuracy of classification or prediction problems over a single learning approach. Combining two different machine learning techniques is termed as Hybridization. The motivation of the hybrid model is that a hybrid classification model can be composed of one unsupervised learner to preprocess the training data and one supervised learner to learn the clustering result.

In machine learning, the hybrid approach has been an active research area for improving classification or prediction performance over a single learning approach. In general, hybridization is based on combining two different machine learning techniques. The rationale behind the hybrid model is that a hybrid classification model can be composed of one unsupervised learner to preprocess the training data and one supervised learner to learn the clustering result. The pruned sub-ensemble was built by first modifying the order in which the

classifiers were aggregated in the ensemble and then selecting the first classifiers in the ordered sequence. The double pruning algorithm was used to reduce the storage requirements, speed up the classification process and improve the performance of the parallel ensembles. Most of the previous work focused on the improvement of selective ensemble techniques,while little attention has been devoted to the development of the hybrid approach of selective ensemble techniques. Motivated by the performance of the hybrid model, an approach that integrates two types of elective ensemble techniques is proposed and called dynamic selection and circulating combination-based clustering (D3C) [78].

Dynamic selection and circulating combination

This approach is presented in terms of the adaptive framework of selective ensemble learning. In [79], a framework with selective ensemble learning based on a complementarily factor was proposed, and the experimental results of handwritten digit recognition showed that it was more flexible and efficient compared to other classifier selection methods. In this paper, the framework with the sequential methods is proposed, and the sequential methods include sequential forward search (SFS) and sequential backward search (SBS).

In the beginning, the performance in the overproduction phase is used to rank all of the classifiers that were obtained by ensemble pruning based on k-means clustering. With the aim of improving the final performance the most accurate classifier is added from the selected subset by SFS and the least accurate classifier is deleted by SBS iteratively. At each step, only one or a few classifiers is added or deleted and as a result, the complexity of the search is not high which is efficient even for large-scale problems [80].

The selection criterion is very important for sequential search. Furthermore, the inter rater agreement κ mentioned above can be the selection criterion. However, there is no explicit theory that shows a measure of diversity that is as straightforward as accuracy. In addition, majority voting (MV), which only works with nominal classes, is utilised as the combination. κ acts as a selection criterion. In practical applications, the majority voting error is the most common selection criterion and is a measure for the majority voting error rate.

Assuming that t classifiers vote for the instance$(\vec{X}j, w_j)$, each classifier classifies the instance and outputs the probability distribution for the instance. Suppose that there is a set of class labels $\Omega = \{1, \ldots, c\}$ and the probability distribution of the instance is a vector $\vec{P} = \{p_1, \ldots p_c\}$. Then, the label with the highest probability obtains a vote. If multiple labels have the same

probability, then all of those labels receive a vote. Once all of the classifiers cast the vote, the label with the most votes is selected as the label for the test instance. If multiple labels have the same number of votes, then one of those labels will be selected at random. The MVE can be formulated as

$$\text{MVE} = 1 - \frac{1}{m}\sum_{j=1}^{m} MV\left(\vec{X}j, w_j\right) \tag{2.33}$$

Where $\text{MV}\left(\vec{X}j, w_j\right)$represents the correct/incorrect decision of the majority voting and is an oracle type of output.

The sequential search method is embedded into the frame-work. This method can be seen as a multilayer sequential search method. In addition, the multilayer classifier subset selection can take full advantage of each base classifier and preserve more useful information than a selective ensemble that obtains only one optimal selection as an ensemble result. The whole framework is composed of many layers. In this framework, the selective ensemble method of one layer can be seen as a multimodal optimization problem where each layer will generate an output, and the search method of the current layer is based on the output of the previous layer. In this situation, each classifier will have the opportunity to participate in one ensemble. A schematic diagram of a multilayer search method is shown in Fig.2.9 and the algorithm is shown in 2.10.

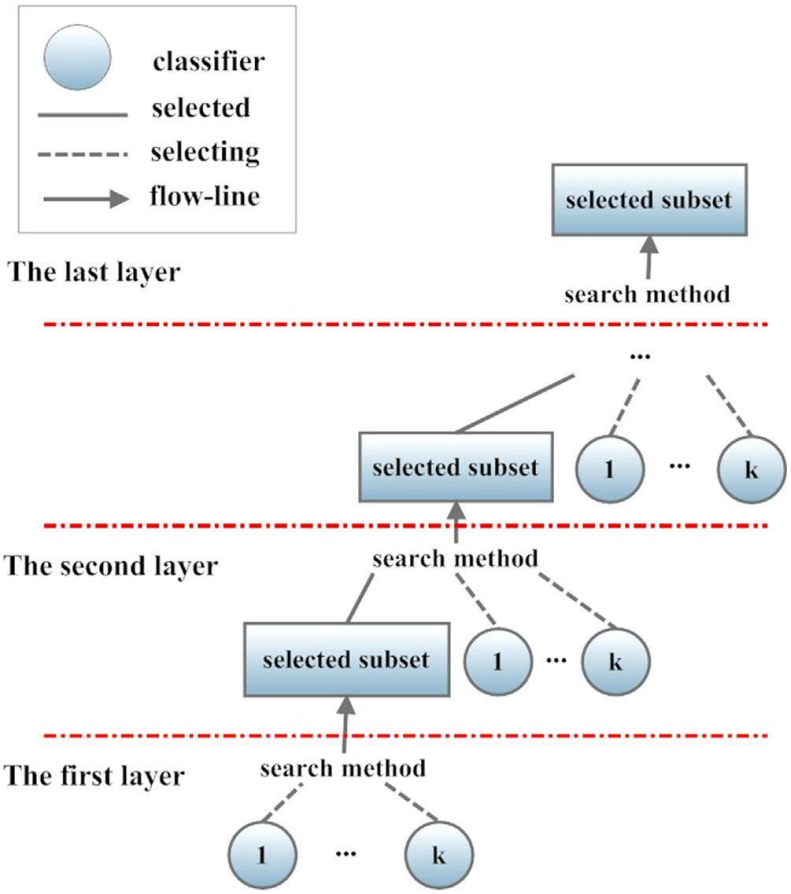

Fig. 2.9 Multilayer Search Method Schematic Diagram

Input: training set $S = \{(\vec{x}_1, \omega_1),..., (\vec{x}_m, \omega_m)\}$, set of classifiers $H' = \{h'_1, ..., h'_k\}$, expected accuracy λ, interval $\Delta\lambda$, threshold $N_{threshold}$, range α, and selected subset $V = \Phi$ *for* *SFS* ($V = H'$ *for* *SBS*);

Output: selected subset V, $MVE(V)$

Procedure:

```
1.  begin
2.      while λ ≥ 0 do
3.          if 1 − MVE(V) > λ
4.              return V;
5.          end if
6.          if Num(V) > N_threshold
7.              V_suboptimal = V ;
8.          end if
9.          for h'_1 to h'_k do
10.             change V to V';
11.             if |κ(V) − κ(V')| < α && MVE(V) > MVE(V')
12.                 commit the changes;
13.                 if 1 − MVE(V') > λ
14.                     return V';
15.                 else
16.                     if κ(V') < κ(V)
17.                         V_suboptimal = V';
18.                     end if
19.                 end if
20.             else
21.                 rollback to V;
22.             end if
23.         end for
24.         λ = λ − Δλ;
25.     end while
26. End
```

Fig. 2.10 The Framework of Dynamic Selection and Circulating Combination

The popular ensemble learning methods used for classification problems are bagging, boosting and random forests. The principal impulse is the production of multi-level models that can be perceived by humans. A model shared subspace boosting algorithm was constructed to

reduce the information redundancy within multi-label learning, [81] which automatically finds shared subspace models, where every model was made to learn from the random feature subspace, bootstrap data and combined a number of base models through multiple labels.

Ensemble classification is a technique that combines multiple basic classifiers that each has its own decision-making capacity. The prediction ability of an ensemble classifier is tremendous to that of a single classifier because the ensemble classifier can address the differences produced by the single classifier more efficiently with different problems when challenged.

LibD3C Classifier

LibD3Cis an ensemble classifier, based on hybrid model of ensemble pruning approach. This kind of classifier is based on k-means clustering and the framework of dynamic selection and circulating in combination with a sequential search method. Ensemble classifier pruning becomes useful in some applications, where the number of independent classifiers that are needed to achieve reasonable accuracy is enormously large [82].

LibD3C is a kind of ensemble classifiers with a clustering and dynamic selection strategy. A method that blends two types of discriminating ensemble techniques called as dynamic selection and circulating combination-based clustering (D3C). LibD3C employs two types of selective ensemble techniques, such as ensemble pruning based on k-means clustering and dynamic selection and circulating combination. LibD3C is a selective ensemble classifier, where various candidate classifiers are trained, and a set of classifiers that are accurate and diverse are selected to rectify the problem [83].

2.3 Training and Testing

A training set is used to fit the models and the validation set or development test set used to estimate test error for model selection. The test set or evaluation test set used for assessment of the generalization error of the finally chosen model [84]. In supervised learning the induced function is usually evaluated on a separate set of inputs and function values for them called the testing set. A hypothesized function is said to generalize when it guesses well on the testing set. Both mean squared error and the total number of errors are common measures. In this research work, the dataset is trained using supervised learning algorithms namely Decision tree classifier, Naïve bayes classifier, artificial neural network, Support Vector Machine and LibD3C classifier.

Testing and Evaluating Classifier Accuracy

Accuracy estimate is used to measure, how accurately a given classifier will be able to identify the class label of the future data. Accuracy estimates also help in the comparison of different classifier. The accuracy of a classifier on a given test set is the percentage of test instances that are correctly classified by the classifier. Error rate is the proportion of errors made over the number of testing instances. When the test dataset is more, estimating the error will be more accurate. The common techniques to assess the accuracy of a classifier are,

- Hold-Out Method.
- Cross-Validation Method.
- K-fold cross validation Method.
- Leave-one-out cross validation Method.

Hold-Out Method

The holdout method is the simplest kind of cross validation where the data set is separated into two sets, called the training set and the testing set. Using the training set, the function approximator fits a function and then the function approximator is asked to predict the output values for the data in the testing set. The test error is used to evaluate the model and the errors are accumulated as before to give the mean absolute test set error. The advantage of this method is that it is usually preferable to the residual method and takes no longer to compute. The evaluation may depend heavily on which data points end up in the training set and which end up in the test set, and thus the evaluation may be significantly different depending on how the division is made.

Cross-Validation Method

Cross validation is a model evaluation method that is better than residuals. The problem with residual evaluations is that one does not give an indication of how well the learner will do when it is asked to make new predictions for data it has not already seen. One way to overcome this problem is to not use the entire data set when training a learner. Some of the data is removed

before training begins. Then when training is done, the data that was removed can be used to test the performance of the learned model on ``new'' data. This is the basic idea for a whole class of model evaluation methods called cross validation.

K-fold Cross-Validation Method

K-fold cross validation is one way to improve over the holdout method. The data set is divided into k subsets, and the holdout method is repeated k times. Each time, one of the k subsets is used as the test set and the other k-1 subsets are put together to form a training set. Then the average error across all k trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set k-1 times. The variance of the resulting estimate is reduced as k is increased. The disadvantage of this method is that the training algorithm has to be rerun from scratch k times, which means it takes k times as much computation to make an evaluation.

Leave-one-out Cross Validation Method

Leave-one-out cross validation is used in the field of machine learning to determine how accurately a learning algorithm will be able to predict data that it was not trained on. When using the leave-one-out method, the learning algorithm is trained multiple times, using all but one of the training set data points. The form of the algorithm is as follows:

- For k = 1 to R where R is the number of training set points
- Temporarily remove the k^{th} data point from the training set.
- Train the learning algorithm on the remaining R - 1 point.
- Test the removed data point and note your error.
- Calculate the mean error over all R data points.

Leave-one-out cross validation is useful because it does not waste data. When training, all but one of the points are used, so the resulting regression or classification rules are essentially the same as if they had been trained on all the data points. The main drawback to the leave-one-out method is that it is expensive - the computation must be repeated as many times as there are training set data points. Leave-one-out cross validation is K-fold cross validation taken to its

logical extreme, with K equal to N, the number of data points in the set. That means that N separate times, the function approximator is trained on all the data except for one point and a prediction is made for that point.

In the proposed work the 10 fold cross validation is performed to test the performance of the muscular dystrophy disease identification models. The cross validation models of the results are compared and discussed.

A common practice in evaluating the performance of the classifiers is to divide the data into two parts, one for training the classifier, the aforementioned training set and a second part called test set that it used for testing it. This testing is done by using the classifier to predict the classes for the samples in the test set and comparing them to their actual classes. There are several measurements that can be used for comparison and in this thesis Accuracy, Precision, Recall and F – score, Cohen's Kappa and Time taken to build the model are focussed. They are defined below as measurements for a specific class.

Accuracy

Accuracy is the percentage of correctly classified instances over all classified instances and is calculated from formula given as follows

$$\textbf{Accuracy} = \frac{\textbf{Truepositive} + \textbf{Truenegative}}{\textbf{Truepositive} + \textbf{Truenegative} + \textbf{Falsepositive} + \textbf{Falsenegative}}$$

Precision

Precision for a given class is the ratio between the amount of the instances that were correctly predicted to it and the total amount of instances that were predicted to it. It is the proportion of the samples that truly have class A among all those which were classified as class A. Precision can be calculated from the formula given as follows

$$\textbf{Precision} = \frac{\textbf{True positive}}{\textbf{True positive} + \textbf{False positive}}$$

Recall

Recall for a given class is the ratio between the amount of the instances that were correctly predicted to it and the total amount of instances that should belong to that class. It is the proportion of samples of a particular class A correctly classified as belonging to that class A. It is equivalent to True Positive Rate (TPR). Recall can be calculated from the formula.,

$$\textbf{Recall} = \frac{\textbf{True positive}}{\textbf{True positive+False negative}}$$

F-score

F-measure discriminates the correct classification of document labels within different classes. In essence, the effectiveness of the algorithm is assessed on a single class, and the higher it is, the better is the clustering. It is defined as follows:

F = 2*(Precision*Recall) / (Precision + Recall)

Cohen's Kappa

Cohen's Kappa statistic is used to measure the inter-annotator agreement. The score computed by this agreement is used to measure the level of agreement between two annotators on a classification problem.

$$\boldsymbol{K = (p_0 - p_e)/(1 - p_e)}$$

where p_o is the empirical probability of agreement on the label assigned to any sample , and p_e is the expected agreement when both annotators assign labels randomly. p_e is estimated using a per-annotator empirical prior over the class labels.

Precision- Recall Curve

The output of the binary classifier is typically studied with the Precision-recall curves. With the intention of extending the precision-recall curve and average precision to multi-class classification, it is essential to binarize the output of the classifiers. One curve can be drawn per label, and a precision-recall curve can be drawn by considering each element of the label indicator matrix as a binary prediction. In this multi-class classification work, the target attribute

values are binarized and the class values are shaped into 1. The precision- recall curve is computed by adding some noisy features and the micro-average ROC and ROC area is calculated.

Receiver Operating Characteristic (ROC) Curve

ROC analysis is performed to identify the presence of disease in response to classification tasks using sensitivity and specificity measures. ROC curves are also typically used in binary classification to study the output of the classifier. ROC curves typically feature true positive rate on the Y axis, and false positive rate on the X axis. The “steepness” of ROC curves is vital as it is perfect to maximize the true positive rate while minimizing the false positive rate.

2.4 Summary

This chapter presents a brief note on DataMining, Datamining in bioinformatics, Machine learning, Ensemble learning. Also, it describes various supervised algorithms like Decision tree classifier, Naïve bayes classifier, Artificial neural network, support vector machines, LibD3C classifier. In general the performance of the classifier depends greatly on the characteristics of the data to be classified. These learning algorithms are considered for building a muscular dystrophy disease classification model, since machine learning technique automatically learns by taking intelligent hints from the training data and predicts the output more accurately.