Indian J. Innovations Dev., Vol. 1, No. 6 (June 2012) ISSN 2277 - 5390

Machine learning based pattern recognition for chemical spectral data

Dharuman C*

Head Dept. of Mathematics, Pachaiyappa's College, E.V.R.Periyar High Road, Shenoy Nagar, Chennai-600 030, India

Correspondence to: Dharuman C, Head Dept. of Mathematics, Pachaiyappa's College, EVR Periyar High Road, Shenoy Nagar, Chennai-600030, India, Email: cdharuman55@gmail.com

Abstract

The most common use for neural networks is to project what will most likely happen. There are many applications where prediction can help in setting priorities. Know who needs the most time critical help can enable a more successful operation. Basically, all organizations must establish priorities which govern the allocation of their resources. This projection of the future is what drove the creation of networks of prediction. In our study, we was examined the machine learning based pattern recognition for chemical spectral data.

Keywords: Machine learning; Pattern recognition: Chemical-spectral data; Intelligent Information system; Q-learning; Artificial neural network.

Abbreviations: RBF - Radial basis function; GRNN - Generalized Regression networks; ANN - Artificial neural network; PNN - Probabilistic neural networks; SVM - Support vector machine; NN - Neural network; SRM - Structural risk minimization; MLP - Multi-layer perception.

Introduction

Machine learning is a burgeoning new technology with a wide range of applications. It has the potential to become one of the key components of intelligent information systems, enabling compact generalizations, inferred from large databases of recorded information, to be applied as knowledge in various practical ways such as being embedded in automatic processes like expert systems, or used directly for communicating with human experts and for educational purposes. In contrast to performance systems which acquire knowledge from human experts, machine learning systems acquire knowledge automatically from examples, i.e., from source data. It is commonly used in classification tasks (recognizing objects, understanding situations, predicting future data, etc.,) and in expert systems (for example for diagnosis). A commercially and scientifically important area of application is Data Mining, where such algorithms are used to detect relevant information and patterns in large databases. The most frequently used techniques include symbolic, inductive learning algorithms such as ID3, multiple-layered, feed-forward neural networks such as Back propagation networks, and evolution-based genetic algorithms. Many information science researchers have started to experiment with these techniques as well (Gail *et al*., 1991).

Machine learning approaches

Learning is an inherent characteristic of the human beings. By virtue of this, people, while executing similar tasks, acquire the ability to improve their performance. This chapter provides an overview of the principle of learning that can be adhered to machines to improve their performance. Such learning is usually referred to as machine learning. Machine learning can be broadly classified into three categories: i) supervised learning ii) Unsupervised learning and iii) Reinforcement learning. Supervised learning requires a trainer, who supplies the input-output training instances. The learning system adapts its parameters by some algorithms to generate the desired output patterns from a given input pattern. In absence of trainers, the desired output for a given input instance is not known, and consequently the learner has to adapt to its

Indian J. Innovations Dev., Vol. 1, No. 6 (June 2012) ISSN 2277 - 5390

parameters autonomously. Such type of learning is termed unsupervised learning. The third type called the reinforcement learning bridges a gap between supervised and unsupervised categories. In reinforcement learning, the learner does not explicitly know the input-output instances, but it receives some form of feedback from its environment. The feedback signals help the learner to decide whether its action on the environment is rewarding or punishable. The learner thus adapts its parameters based on the states (rewarding / punishable) of its actions. Among the supervised learning techniques, the most common are inductive and analogical learning. The inductive learning technique, presented in the chapter, includes decision tree and version space based learning. Analogical learning is briefly introduced through illustrative examples. The principle of unsupervised learning is illustrated here with a clustering problem. The section on reinforcement learning includes Q-learning and temporal difference learning. A fourth category of learning, which has emerged recently from the disciplines of knowledge engineering, is called inductive logic programming. The principles of inductive logic programming have also been briefly introduced in this research. The research ends with a brief discussion on the computational theory of learning. With the background of this theory, one can measure the performance of the learning behavior of a machine from the training instances and their count. For example, Hopfield networks have been used extensively in the area of global optimization and search; Kohonen networks have been adopted in unsupervised learning and pattern recognition. The other important machine algorithms are support vector machines, genetic algorithms and fuzzy logic. Other important related algorithms are Hidden Markov model and Markov Chain Monte Carlo (Herrero *et al*., 2001).

Artificial neural network

The foundation of the neural networks paradigm was laid in the 1950s and this approach has attracted significant attention in the past decade due to the development of more powerful hardware and neural algorithms. Nearly all connectionist algorithms have a strong learning component. Learning algorithms can be applied to adjust connection weights so that the network can predict or classify unknown examples correctly. Most research effort is directed towards the invention of new algorithms for learning (Swarm optimization), rather than towards gaining experience in applying existing techniques to real problems. Neural networks have been adopted in various engineering, business, military, and biomedical domains. Among these artificial neural networks have had a number of successful applications in biology such as pattern recognition in DNA and proteins, protein structure prediction, analysis and clustering of gene expression data, modeling gene network. An artificial neural network may be described as a set of neurons or nodes Xi, each transforming its total or net input x_ini into an output or activity xi according to an activation

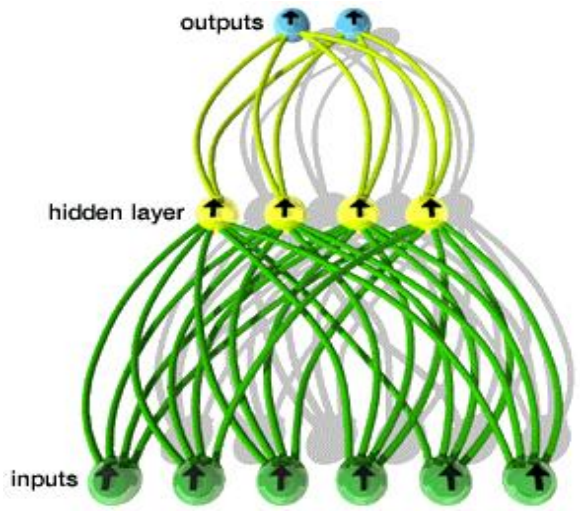

Fig.1. Basic model of a network

function (or transfer function) f(x_ini). Each node Xi sends its output to other units Xj through connections each having a certain effectiveness or weight wij. The net input to any unit Xj is usually modeled as a sum of all the outputs xi from other units (and, in recurrent nets, from itself), weighted by the weights wij of the respective connections. In a neural network model, simple nodes (called neurons) are connected together to form a network

Indian J. Innovations Dev., Vol. 1, No. 6 (June 2012) ISSN 2277 - 5390

of nodes - hence the term "neural network" (Fig.1). While a neural network does not have to be adaptive per se, its practical use comes with algorithms designed to alter the strength (weights) of the connections in the network to produce a desired signal flow (Selaru *et al*., 2002).

The brain is principally composed of about 10 billion neurons; each connected to about 10,000 other neurons. Each of the yellow blobs in the picture above is neuronal cell bodies (soma), and the lines are the input and output channels (dendrites and axons) which connect them. Each neuron receives electrochemical inputs from other neurons at the dendrites. If the sum of these electrical inputs is sufficiently powerful to activate the neuron, it transmits an electrochemical signal along the axon, and passes this signal to the other neurons whose dendrites are attached at any of the axon terminals. These attached neurons may then fire.

It is important to note that a neuron fires only if the total signal received at the cell body exceeds a certain level. The neuron either fires or it does not, if there are no different grades of firing. So, our entire brain is composed of these interconnected electro-chemical transmitting neurons. From a very large number of extremely simple processing units (each performing a weighted sum of its inputs, and then firing a binary signal if the total input exceeds a certain level) the brain manages to perform extremely complex tasks. This is the model on which artificial neural networks are based. Thus far, artificial neural networks haven't even come close to modeling the complexity of the brain, but they have shown to be good at problems which are easy for a human but difficult for a traditional computer, such as image recognition and predictions based on past knowledge.

Biological neurons

Neurons are body cells specialized for signal transmission and signal processing. Fig.2 shows the typical structural characteristics of a neuron. It has a cell body (or soma) and root-like extensions called neuritis. Amongst the neuritis, one major outgoing trunk is the axon, and the others are dendrites. The signal processing capabilities of a neuron is its ability to vary its intrinsic electrical potential (membrane potential) through special electro-physical and chemical processes. A single neuron receives signals from many other neurons, (typically in order of 10,000 for mammals) at specialized sites on the cell body or on the dendrites, known as synapses. Synapses receive signals from a pre-synaptic neuron and alter the state of the postsynaptic neuron (the receiver neuron) and eventually trigger the generation of an electric pulse, the action potential (a spike), in the postsynaptic neuron. This action potential is initiated at the rooting region of the axon, the axon-hillock, and it subsequently travels along the axon sending information signal to the other parts of the nervous system.

Among the numerous artificial neural networks which have been proposed recently, Back propagation networks have been extremely popular for their unique learning capability. Back propagation networks are fully connected, layered, feed-forward models. Activations flow from the input layer through the hidden layer, then to the output layer. A Back propagation network typically starts out with a random set of weights. The network adjusts its weights each time it sees an input-output pair. Each pair is processed at two stages, a forward pass and a backward pass. The forward pass involves presenting a sample input to the network and letting activations flow until they reach the output layer. During the backward pass, the network's actual output is compared with the target output and error estimates are computed for the output units. The weights connected to the output units are adjusted in order to reduce the errors (a gradient descent method). The error estimates of the output units are then used to derive error estimates for the units in the hidden layer. Finally, errors are propagated back to the connections stemming from the input units. The Back propagation network updates its weights incrementally until the network stabilizes.

Feed forward neural network

Indian J. Innovations Dev., Vol. 1, No. 6 (June 2012) ISSN 2277 - 5390

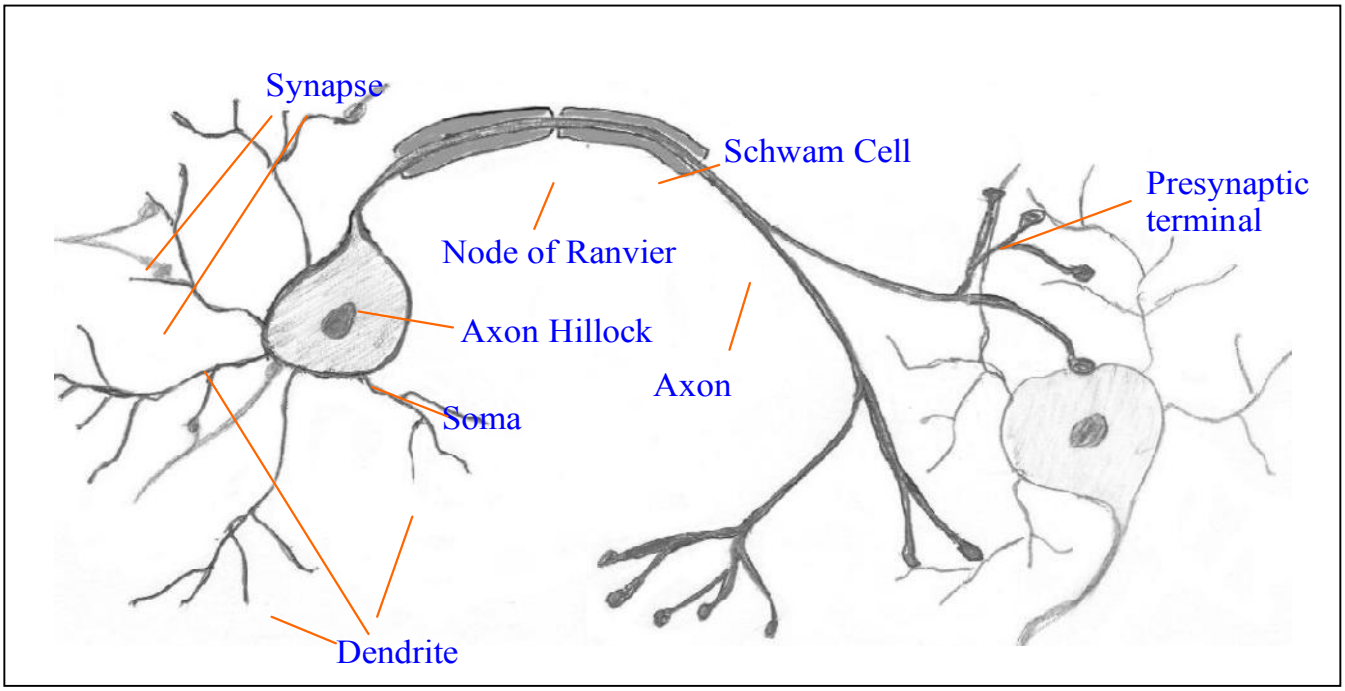

Fig.2. Diagram of generic neuron

A feed forward neural network is an artificial neural network where connections between the units do not form a directed cycle. This is different from recurrent neural networks. The feed forward neural network was the first and arguably simplest type of artificial neural network devised. A feed forward neural network is a biologically inspired classification algorithm. It consists of a (possibly large) number of simple neuron-like processing units, organized in layers. Every unit in a layer is connected with all the units in the previous layer. These connections are not all equal; each connection may have a different strength or weight. The weights on these connections encode the knowledge of a network. Often the units in a neural network are also called nodes. Data enters at the inputs and passes through the network, layer by layer, until it arrives at the outputs. During normal operation, that is when it acts as a classifier, there is no feedback between layers. This is why they are called feed forward neural networks (Fig.3).

Feed-forward networks have the following characteristics:

1. Perceptrons are arranged in layers, with the first layer taking in inputs and the last layer producing outputs. The middle layers have no connection with the external world, and hence are called hidden layers.
2. Each perceptron in one layer is connected to every perceptron on the next layer. Hence information is constantly fed forward from one layer to the next, and this explains why these networks are called feed-forward networks.
3. There is no connection among perceptrons in the same layer.

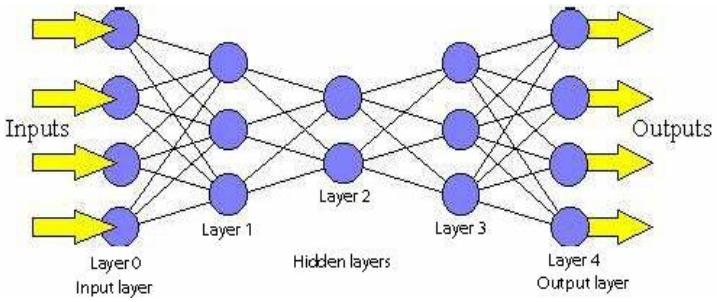

Fig.3. A feed-forward neural networks

FFNN Phases

During the learning phase the weights in the FF Net will be modified. All weights are modified in such a way that when a pattern is presented, the

Indian J. Innovations Dev., Vol. 1, No. 6 (June 2012) ISSN 2277 - 5390

output unit with the correct category, hopefully, will have the largest output value. In the classification phase the weights of the network are fixed. A pattern, presented at the inputs, will be transformed from layer to layer until it reaches the output layer (Fig.4). Now classification can occur by selecting the category associated with the output unit that has the largest output value. The traditional two-phase approach to FNN development. Phase I consist of the following procedures:

1) Determining dependent variable and independent variables for the problem.
2) Collecting data according to specified dependent and independent variables
3) Identifying significant independent variables using multivariate analysis method (e.g., multiple linear regression)
4) Separating data into two sets namely training data set and validation data set.
5) Determining set of FNNs with different architectures.

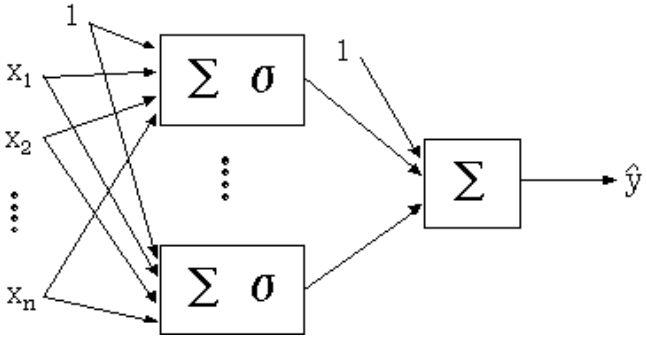

Fig.4. A network showing input vectors, weight vector and activation function

Phase II consists of the following procedures:

1. Training each FNN using training data set, and choosing FNN with highest percentage of correction classification.
2. Validating the chosen FNN using validation data set.

Supervised learning

The FF net uses a supervised learning algorithm; besides the input pattern, the neural net also needs to know to what category the pattern belongs. Learning proceeds as follows: a pattern is presented at the inputs. The pattern will be transformed in its passage through the layers of the network until it reaches the output layer. The units in the output layer all belong to a different category. The outputs of the network as they are now are compared with the outputs as they ideally would have been if this pattern were correctly classified: in the latter case the unit with the correct category would have had the largest output value and the output values of the other output units would have been very small. On the basis of this comparison all the connection weights are modified a little bit to guarantee that, the next time this same pattern is presented at the inputs, the value of the output unit that corresponds with the correct category is a little bit higher than it is now and that, at the same time, the output values of all the other incorrect outputs are a little bit lower than they are now. The differences between the actual outputs and the idealized outputs are propagated back from the top layer to lower layers to be used at these layers to modify connection weights. This is why the term back propagation network is also often used to describe this type of neural network.

Mathematically the functionality of a hidden neuron is described by Eqn.(3.1)

$$\sigma = \sum_{j=1}^{n} w_j x_j + b_j \quad (3.1)$$

where the weights $w_j\ b_j$ are symbolized with the arrows feeding into the neuron. The network output is formed by another weighted summation of the outputs of the neurons in the hidden layer. This summation on the output is called the output layer. In Fig.4 there is only one output in the output layer since it is a single-output problem. Generally, the number of output neurons equals the number of outputs of the approximation problem. The neurons in the hidden layer of the network in Fig.3 are similar in structure to those of the perception, with the exception that their activation functions can be any differential function. The output of this network is given by Eqn.(3.2).

Indian J. Innovations Dev., Vol. 1, No. 6 (June 2012) ISSN 2277 - 5390

$$\hat{y}(\theta) = g(\theta, x) = \sum_{i=1}^{nh} w_i^2 \sigma\left[\sum_{j=1}^{n} w_{ij}^1 x_j + b_{ji}^1\right] + b^2 \quad (3.2)$$

where n is the number of inputs and nh is the number of neurons in the hidden layer. The variables w_{ij}^1, b_{ji}^1, w_i^2, b^2 are the parameters of the network model that are represented collectively by the parameter vector θ. In general, the neural network model will be represented by the compact notation g(θ,x) whenever the exact structure of the neural network is not necessary in the context of a discussion.

Note that the size of the input and output layers are defined by the number of inputs and outputs of the network and, therefore, only the number of hidden neurons has to be specified when the network is defined. The network in Fig.1 is sometimes referred to as a three-layer network, counting input, hidden, and output layers. However, since no processing takes place in the input layer, it is also sometimes called a two layer network. In training the network, its parameters are adjusted incrementally until the training data satisfy the desired mapping as well as possible; that is, until $\hat{y}(\theta)$ matches the desired output y as closely as possible up to a maximum number of iterations. The nonlinear activation function in the neuron is usually chosen to be a smooth step function. The default is the standard sigmoid by Eqn.(3.3).

$$\text{sigmoid}[x] = \frac{1}{1 + e^{-x}} \quad (3.3)$$

Back propagation algorithm

Back propagation, or propagation of error, is a common method of teaching artificial neural networks how to perform a given task. It was first described by Werbos in 1974, but it wasn't until 1986, through the work of Rumelhart, et al., that it gained recognition, and it led to a renaissance in the field of artificial neural network research. It is a supervised learning method, and is an implementation of the Delta rule. It can calculate the desired output for any given input. It is most useful for feed-forward networks (networks that have no feedback, or simply, that have no connections that loop). The term is an abbreviation for backwards propagation of errors. Back propagation requires that the activation function used by the artificial neurons is differentiable.

The algorithm of the back propagation technique is as follows:

- Present a training sample to the neural network.
- Compare the network's output to the desired output from that sample. Calculate the error in each output neuron.
- For each neuron, calculate what the output should have been, and a scaling factor, how much lower or higher the output must be adjusted to match the desired output. This is the local error.
- Adjust the weights of each neuron to lower the local error.
- Assign blame for the local error to neurons at the previous level, giving greater responsibility to neurons connected by stronger weights.
- Repeat from step 3 on the neurons at the previous level, using each one's blame as its error.

As the algorithm's name implies, the errors (and therefore the learning) propagate backwards from the output nodes to the inner nodes. So technically speaking, back propagation is used to calculate the gradient of the error of the network with respect to the network's modifiable weights. This gradient is almost always then used in a simple stochastic gradient descent algorithm to find weights that minimize the error. Often the term back propagation is used in a more general sense, to refer to the entire procedure encompassing both the calculation of the gradient and its use in stochastic gradient descent. Back propagation usually allows quick convergence on satisfactory local minima for error in the kind of networks to which it is suited.

It is important to note that back propagation networks are necessarily multilayer perceptions (usually with one input, one hidden, and one output layer). In order for the hidden layer to serve any useful function, multilayer networks must have

Indian J. Innovations Dev., Vol. 1, No. 6 (June 2012) ISSN 2277 - 5390

non-linear activation functions for the multiple layers: a multilayer network using only linear activation functions is equivalent to some single layer, linear network. Non-linear activation functions that are commonly used include the logistic function, the soft max function, and the Gaussian functions. The back propagation algorithm for calculating a gradient has been rediscovered a number of times, and is a special case of a more general technique called automatic differentiation in the reverse accumulation mode.

Supervised patterns-network selection

Because all artificial neural networks are based on the concept of neurons, connections, and transfer functions, there is a similarity between the different structures, or architectures, of neural networks. The majority of the variations stems from the various learning rules and how those rules modify a network's typical topology. The following sections outline some of the most common artificial neural networks. They are organized in very rough categories of application. These categories are not meant to be exclusive, they are merely meant to separate out some of the confusion over network architectures and their best matches to specific applications. Basically, most applications of neural networks fall into the following five categories namely: 1) Prediction 2) Classification 3) Data association 4) Data conceptualization 5) Data filtering.

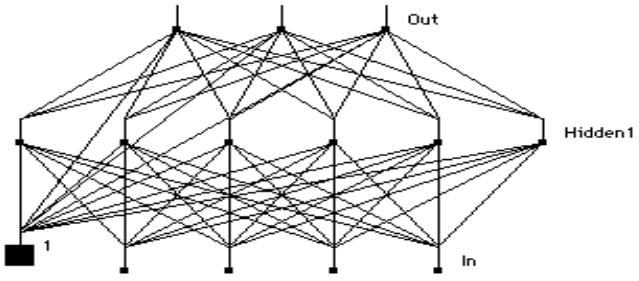

Fig.5. Feed forward Back-propagation Neural Network

The most common use for neural networks is to project what will most likely happen. There are many applications where prediction can help in setting priorities. For example, the emergency room at a hospital can be a hectic place. To know who needs the most time critical help can enable a more successful operation. Basically, all organizations must establish priorities which govern the allocation of their resources. This projection of the future is what drove the creation of networks of prediction (Yegnanarayana, 1994).

The feed forward, back-propagation architecture was developed in the early 1970's by several independent sources. This independent co-development was the result of a proliferation of articles and talks at various conferences which stimulated the entire industry. Currently, this synergistically developed back-propagation architecture is the most popular, effective, and easy to learn model for complex, multi-layered networks. This network is used more than all others combined. It is used in many different types of applications. This architecture has spawned a large class of network types with many different topologies and training methods. Its greatest strength is in non-linear solutions to ill defined problems. The typical back-propagation network has an input layer, an output layer, and at least one hidden layer. There is no theoretical limit on the number of hidden layers but typically there is just one or two. Some work has been done which indicates that a maximum of four layers (three hidden layers plus an output layer) are required to solve problems of any complexity. Each layer is fully connected to the succeeding layer, as shown in Fig.5. The in and out layers indicate the flow of information during recall. Recall is the process of putting input data into a trained network and receiving the answer. Back-propagation is not used during recall, but only when the network is learning a training set.

The number of layers and the number of processing elements per layer are important decisions. These parameters to a feed forward, back-propagation topology are also the most ethereal. They are the art of the network designer. There is no quantifiable, best answer to the layout of the network for any particular application. There are only general rules picked up over time and followed by most researchers and engineers applying this architecture to their problems.

Rule One: As the complexity in the relationship between the input data and the desired output

Indian J. Innovations Dev., Vol. 1, No. 6 (June 2012) ISSN 2277 - 5390

increases, then the number of the processing elements in the hidden layer should also increase.
Rule Two: If the process being modeled is separable into multiple stages, then additional hidden layer(s) may be required. If the process is not separable into stages, then additional layers may simply enable memorization and not a true general solution.

Rule Three: The amount of training data available sets an upper bound for the number of processing elements in the hidden layer(s). To calculate this upper bound, use the number of input / output pair examples in the training set and divide that number by the total number of input and output processing elements in the network. Then divide that result again by a scaling factor between five and ten. Larger scaling factors are used for relatively noisy data. Extremely noisy data may require a factor of twenty or even fifty, while very clean input data with an exact relationship to the output might drop the factor to around two. It is important that the hidden layers have few processing elements. Too many artificial neurons and the training set will be memorized. If that happens then no generalization of the data trends will occur, making the network useless on new data sets.

Once the above rules have been used to create a network, the process of teaching begins. This teaching process for a feed forward network normally uses some variant of the Delta rule, which starts with the calculated difference between the actual outputs and the desired outputs. Using this error, connection weights are increased in proportion to the error times a scaling factor for global accuracy. Doing this for an individual node means that the inputs, the output, and the desired output all have to be present at the same processing element. The complex part of this learning mechanism is for the system to determine which input contributed the most to an incorrect output and how does that element get changed to correct the error. An inactive node would not contribute to the error and would have no need to change its weights. To solve this problem, training inputs are applied to the input layer of the network, and desired outputs are compared at the output layer. During the learning process, a forward sweep is made through the network, and the output of each element is computed layer by layer. The difference between the output of the final layer and the desired output is back-propagated to the previous layer(s), usually modified by the derivative of the transfer function, and the connection weights are normally adjusted using the Delta Rule. This process proceeds for the previous layer(s) until the input layer is reached. There are many variations to the learning rules for back-propagation networks. Different error functions, transfer functions, and even the modifying method of the derivative of the transfer function can be used. The concept of momentum error was introduced to allow for more prompt learning while minimizing unstable behavior. Here, the error function, or delta weight equation, is modified so that a portion of the previous delta weight is fed through to the current delta weight. This acts, in engineering terms, as a low-pass filter on the delta weight terms since general trends are reinforced whereas oscillatory behaviour is cancelled out. This allows a low, normally slower, learning coefficient to be used, but creates faster learning. Another technique that has an effect on convergence speed is to only update the weights after many pairs of inputs and their desired outputs are presented to the network, rather than after every presentation. This is referred to as cumulative back-propagation because the delta weights are not accumulated until the complete set of pairs is presented. The number of input-output pairs that are presented during the accumulation is referred to as an epoch. This epoch may correspond either to the complete set of training pairs or to a subset.

There are limitations to the feed forward, back-propagation architecture. Back-propagation requires lots of supervised training, with lots of input / output examples. Additionally, the internal mapping procedures are not well understood, and there is no guarantee that the system will converge to an acceptable solution. At times, the learning gets stuck in local minima, limiting the best solution. This occurs when the network system

Indian J. Innovations Dev., Vol. 1, No. 6 (June 2012) ISSN 2277 - 5390

finds an error that is lower than the surrounding possibilities but does not finally get to the smallest possible error. Many learning applications add a term to the computations to bump or jog the weights past shallow barriers and find the actual minimum rather than a temporary error pocket. Typical feed forward back-propagation applications include speech synthesis from text, robot arms, evaluation of bank loans, image processing, knowledge representation, forecasting and prediction, and multi-target tracking. Each month more back-propagation solutions are announced in the trade journals.

Higher-order neural network or functional-link network

Either name is given to neural networks which expand the standard feed forward, back-propagation architecture to include nodes at the input layer which provide the network with a more complete understanding of the input. Basically, the inputs are transformed in a well understood mathematical way so that the network does not have to learn some basic math functions. These functions do enhance the network's understanding of a given problem. These mathematical functions transform the inputs via higher-order functions such as squares, cubes, or sines. It is from the very name of these functions, higher-order or functionally linked mappings, that the two names for this same concept were derived. This technique has been shown to dramatically improve the learning rates of some applications. An additional advantage to this extension of back propagation is that these higher order functions can be applied to other derivations - delta bar delta, extended delta bar delta, or any other enhanced feed forward, back-propagation networks.

There are two basic ways of adding additional input nodes. First, the cross-products of the input terms can be added into the model. This is also called the output product or tensor model, where each component of the input pattern multiplies the entire input pattern vector. A reasonable way to do this is to add all interaction terms between input values. For example, for a back-propagation network with three inputs (A, B and C), the cross-products would include: AA, BB, CC, AB, AC, and BC. This example adds second-order terms to the input structure of the network. Third-order terms, such as ABC, could also be added. The second method for adding additional input nodes is the functional expansion of the base inputs. Thus, a back-propagation model with A, B and C might be transformed into a higher-order neural network model with inputs: A, B, C, Sin(A), Cos(B), Log(C), Max(A,B,C), etc., In this model, input variables are individually acted upon by appropriate functions. Many different functions can be used. The overall effect is to provide the network with an enhanced representation of the input. It is even possible to combine the tensor and functional expansion models together. No new information is added, but the representation of the inputs is enhanced. Higher-order representation of the input data can make the network easier to train. The joint or functional activations become directly available to the model. In some cases, a hidden layer is no longer needed. However, there are limitations to the network model. Many more input nodes must be processed to use the transformations of the original inputs. With higher-order systems, the problem is exacerbated. Yet, because of the finite processing time of computers, it is important that the inputs are not expanded more than is needed to get an accurate solution.

Radial basis function network

Radial basis function (RBF) networks have a static Gaussian function as the nonlinearity for the hidden layer processing elements. The Gaussian function responds only to a small region of the input space where the Gaussian is centered. The key to a successful implementation of these networks is to find suitable centers for the Gaussian functions. This can be done with supervised learning, but an unsupervised approach usually produces better results.

Indian J. Innovations Dev., Vol. 1, No. 6 (June 2012) ISSN 2277 - 5390

The simulation starts with the training of an unsupervised layer. Its function is to derive the Gaussian centers and the widths from the input data. These centers are encoded within the weights of the unsupervised layer using competitive learning. During the unsupervised learning, the widths of the Gaussians are computed based on the centers of their neighbors. The output of this layer is derived from the input data weighted by a Gaussian mixture. Once the unsupervised layer has completed its training, the supervised segment then sets the centers of Gaussian functions (based on the weights of the unsupervised layer) and determines the width (standard deviation) of each Gaussian. Any supervised topology (such as a MLP) may be

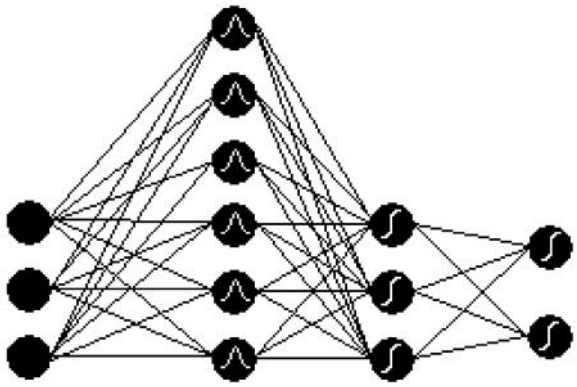

Fig.6. Radial Basis Function Network

used for the classification of the weighted input. The advantage of the radial basis function network is that it finds the input to output map using local approximators. Usually the supervised segment is simply a linear combination of the approximators. Since linear combiners have few weights, these networks train extremely fast and require fewer training samples.

Radial basis networks may require more neurons than standard feed-forward back propagation networks, but often they can be designed in a fraction of the time it takes to train standard feed-forward networks. They work best when many training vectors are available (Chen et al., 2005). This chapter discusses two variants of radial basis networks, Generalized Regression networks (GRNN) and Probabilistic neural networks (PNN) construction of RBF network for tuberculosis prediction.

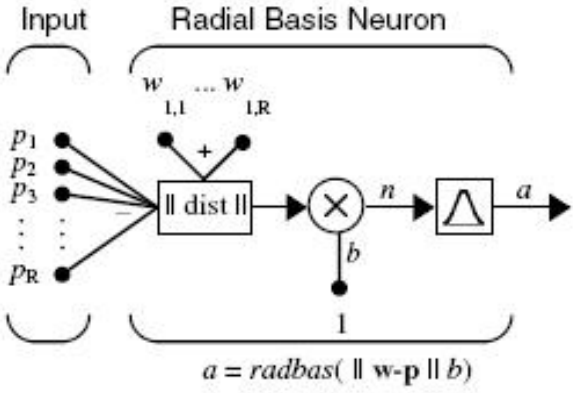

Fig.7. Neuron Model with R inputs

A generalized regression neural network (GRNN) is often used for function approximation. It has been shown that, given a sufficient number of hidden neurons, GRNNs can approximate a continuous function to an arbitrary accuracy. Probabilistic neural networks (PNN) can be used for classification problems. Their design is straightforward and does not depend on training. A PNN is guaranteed to converge to a Bayesian classifier providing it is given enough training data. These networks generalize well. The GRNN and PNN have many advantages, but they both suffer from one major disadvantage. They are slower to operate because they use more computation than other kinds of networks to do their function approximation or classification (Fig.6 & 7).

Radial basis function (RBF) networks typically have three layers: an input layer, a hidden layer with a non-linear RBF activation function and a linear output layer. The output, $\varphi = R^n \to R$, of the network is thus by Eqn. (3.4).

$$\varphi = (x) = \sum_{i=1}^{N} a_i \rho(\|x - c_i\|) \quad (3.4)$$

where N is the number of neurons in the hidden layer, Ci is the center vector for neuron i, and ai are the weights of the linear output neuron. In the basic form all inputs are connected to each hidden neuron. The norm is typically taken to be the

Indian J. Innovations Dev., Vol. 1, No. 6 (June 2012) ISSN 2277 - 5390

Euclidean distance and the basis function is taken to be Gaussian by Eqn. (3.5).

$$\rho(\|x - c_i\|) = \exp(-\beta\|x - c_i\|^2) \quad (3.5)$$

The Gaussian basis functions are local in the sense that by Eqn. (3.6)

$$\lim_{\|x\|\to\infty} \rho(\|x - c_i\|) = 0 \quad (3.6)$$

i.e., changing parameters of one neuron has only a small effect for input values that are far away from the center of that neuron.

RBF networks are universal approximators on a compact subset of Rn. This means that a RBF network with enough hidden neurons can approximate any continuous function with arbitrary precision. The weights ai, ci and β are determined in a manner that optimizes the fit between φ and the data (Vishal Gupta *et al*., 2009).

Normalized architecture

In addition to the above non-normalized architecture, RBF networks can be normalized. In this case the mapping is

$$\varphi(X) \overset{def}{=} \frac{\sum_{i=1}^{N} a_i \rho(\|X - c_i\|)}{\sum_{i=1}^{N} \rho(\|X - c_i\|)} = \sum_{i=1}^{N} a_i u(\|X - c_i\|) \quad (3.7)$$

where $u(\|X - c_i\|) \overset{def}{=} \frac{\rho(\|X - c_i\|)}{\sum_{i=1}^{N} \rho(\|X - c_i\|)}$

is known as a normalized radial basis function.

Training

In a RBF network, there are three types of parameters that need to be chosen to adapt the network for a particular task: the center vectors ci, the output weights wi, and the RBF width parameters βi. In the sequential training of the weights are updated at each time step as data streams in. For some tasks it makes sense to define an objective function and select the parameter values that minimize its value. The most common objective function is the least squares function

$$K(w) \overset{def}{=} \sum_{t=1}^{\infty} K_t(w) \quad (3.8)$$

where $K(w) \overset{def}{=} [y(t) + \varphi(x(t), w)]^2$.

We have explicitly included the dependence on the weights. Minimization of the least squares objective function by optimal choice of weights optimizes accuracy of fit. There are occasions in which multiple objectives, such as smoothness as well as accuracy, must be optimized. In that case it is useful to optimize a regularized objective function such as

$$H(w) \overset{def}{=} K(w) + \lambda S(w) \overset{def}{=} \sum_{t=1}^{\infty} H_t(w) \quad (3.9)$$

where $S(w) \overset{def}{=} \sum_{t=1}^{\infty} S_t(w)$

and $H_t(w) \overset{def}{=} K_t(w) + \lambda S_t(w)$

where optimization of S maximizes smoothness and λ is known as a regularization parameter.

Logistic regression

Logistic regression is a model used for prediction of the probability of occurrence of an event by fitting data to a logistic curve. It makes use of several predictor variables that may be either numerical or categories. For example, the probability that a person has a heart attack within a specified time period might be predicted from knowledge of the person's age, sex and body mass index. Logistic regression is used extensively in the medical and social sciences as well as marketing applications such as prediction of a customer's capacity to purchase a product or cease a subscription. Logistic regression analyzes binomially distributed data of the form

$$Y_i \sim B(n_i, p_i) \quad ; \quad \text{for } i = 1,..., m \quad (3.10)$$

Where the numbers of Bernoulli trials ni are known and the probabilities of success pi are unknown. An example of this distribution is the fraction of seeds (pi) that germinate after ni are planted. The model proposes for each trial (value of i) there is a set of explanatory variables that might inform the final

Indian J. Innovations Dev., Vol. 1, No. 6 (June 2012) ISSN 2277 - 5390

probability. These explanatory variables can be thought of as being in a k vector Xi and the model then takes the form

$$p_i = E\left(\frac{Y_i}{n_i}\middle| X_i\right) \quad (3.11)$$

The logits of the unknown binomial probabilities (i.e., the logarithms of the odds) are modelled as a linear function of the Xi.

$$\log it(p_i) = In\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_{1,i} + ... + \beta_k x_{k,i} \quad (3.12)$$

Note that a particular element of Xi can be set to 1 for all i to yield an intercept in the model. The unknown parameters βj are usually estimated by maximum likelihood. The interpretation of the βj parameter estimates is as the additive effect on the log odds ratio for a unit change in the jth explanatory variable. In the case of a dichotomous explanatory variable, for instance gender, eβ is the estimate of the odds ratio of having the outcome for, say, males compared with females. The model has an equivalent formulation

$$p_i = \frac{1}{1+e^{-(\beta_0+\beta_1 x_{1,i}+...+\beta_k x_{k,i})}} \quad (3.13)$$

This functional form is commonly called a single-layer perceptron or single-layer artificial neural network. A single-layer neural network computes a continuous output instead of a step function. The derivative of pi with respect to X = x1,...,xk is computed from the general form:

$$y = \frac{1}{1+e^{-f(X)}} \quad (3.14)$$

Where f(X) is an analytic function in X. With this choice, the single-layer network is identical to the logistic regression model. This function has a continuous derivative, which allows it to be used in back propagation. This function is also preferred because its derivative is easily calculated:

$$y' = y(1-y)\frac{df}{dx} \quad (3.15)$$

Unsupervised Neural methods

In the unsupervised learning, adjustment of synaptic weights may be carried through the use of neurobiological principles such as Hebbian learning and competitive learning. In this section we will describe specific applications of these two approaches.

FUZZY artificial neural network

It may be mentioned that human reasoning is somewhat fuzzy in nature. The utility of fuzzy set (Kli and Folger 1989, Pal and Dutta Majumder 1986) lies in their ability to model the uncertain or ambiguous data so often encountered in real life. Hence, to enable a system to tackle real-life situations in a manner more like humans, one may incorporate the concept of fuzzy sets into the neural network. It is to be noted that although fuzzy logic is a natural mechanism for propagating uncertainty, it may involve in some cases an increase in the amount of computation. (compared with a system using classical binary logic). This can be suitably offset by using fuzzy neural network models having the potential for parallel computation with high flexibility. Fuzzy concepts have already been incorporated into neural nets in control problems, in modeling output possibility distributions (Ishibuchi and Tanaka 1990), in learning and extrapolating complex relationships between antecedents and consequents of rules, and in fuzzy reasoning.

The fuzzy attempts to build a fuzzy version of the multilayer perceptron are using the gradient-descent-based back-propagation algorithm, by incorporating concepts from fuzzy sets at various stages. Besides, conventional two-state neural net models generally deal with the ideal condition, where an input feature is either present or absent and each pattern belongs to either one class or another. They do not consider cases where an input feature may possess a property with a certain degree of confidence, or where a pattern may

Indian J. Innovations Dev., Vol. 1, No. 6 (June 2012) ISSN 2277 - 5390

belong to more than one class with a finite degree of belongingness.

Broadly, the network passes through two phases, viz., training and testing. During the training phase supervised learning is used to assign output membership values lying in the range (0, 1) to the training vectors. Hence each output class mode in the output layer, may be assigned a nonzero membership instead of choosing the single class (node) with the highest activation. This allows modeling of fuzzy data when the feature space involves overlapping pattern classes such that a pattern point may belong to more than one class with nonzero membership.

During training, each error in membership assignment is fed back and the connection weights of the network are appropriately updated. The back propagated error is computed with respect to each desired output, which is a membership value denoting the degree of belongingness of the input vector to that class. Hence the error which is back-propagated for weight updating, has inherently more weight in case of nodes with higher membership values. The contribution of ambiguous or uncertain vectors to the weight correction is automatically reduced. This is natural as vectors that are more typical of their class should have more influence in determining the position and shape of the decision surface.

The utility of the approach proposed here for the modeling of output values may be further appreciated by considering a point lying in a region of overlapping class in the feature space. In such cases its membership in each of these classes may be nearly equal. Then there is no reason why we should follow the crisp approach of classifying this pattern as belonging to the class corresponding to that output neuron with a slightly higher activation, and thereby neglect the smaller yet significant responses obtained for the other overlapping class.
After a number of cycles the neural net may converge to a minimum error solution. The network now encodes the input space information in its connection weights. In the second phase, a part of the same fuzzy data with is kept aside for testing during random selection of the training set is applied as input and the network. Here each test datum contributes a count at a particular position in this matrix, its row corresponding to the class to which it belongs (as determined from the hard labels attached to the input data set) and its column indicating the class corresponding to the neuronal output providing the best match.

The proposed fuzzy neural network model is capable of handling input feature presented in quantitative and linguistic form. The components of the input vector consist of the membership values to the overlapping partitions of linguistic properties low, medium and high corresponding to each input feature. This provides scope for incorporating linguistic information in both the training and the testing phases of the said model and increase its robustness in tackling imprecise or uncertain input specifications.

During training, the learning rate and the damping coefficient are gradually decreased until the network hopefully converges to a minimum error sultan. This heuristic helps to avoid spurious local minima and usually prevents oscillations of the mean square error in the weight space. In the process, the network undergoes a maximal number of sweeps through the training set.

Support Vector Machines

SVMs will be presented in a gentle way-starting with linear separable problems, through the classification tasks having overlapped classes but still a linear separation boundary, beyond the linearity assumptions to the nonlinear separation boundary, and finally to the linear and nonlinear regression problems. The adjective parsimonious denotes an SVM with a small number of support vectors. The scarcity of the model results from a sophisticated learning that matches the model capacity to the data complexity ensuring a good performance in the future, previously unseen, data. Same as the neural networks or similarly to them, SVMs possess the well known ability of being universal approximators of any multivariate function to any desired degree of accuracy. Consequently, they are of particular interest for modeling the unknown, or partially known, highly

Indian J. Innovations Dev., Vol. 1, No. 6 (June 2012) ISSN 2277 - 5390

nonlinear, complex systems, plants or processes. SVMs have been developed in the reverse order to the development of neural networks (NNs). SVMs evolved from the sound theory to the implementation and experiments, while the NNs followed more heuristic path, from applications and extensive experimentation to the theory. It is interesting to note that the very strong theoretical background of SVMs did not make them widely appreciated at the beginning. The publication of the first papers by Vapnik, Chervonenkis and co-workers in 1964 / 65 went largely unnoticed till 1992. This was due to a widespread belief in the statistical and/or machine learning community that, despite being theoretically appealing, SVMs are neither suitable nor relevant for practical applications. They were taken seriously only when excellent results on practical learning benchmarks were achieved in digit recognition, computer vision and text categorization. Today, SVMs show better results than (or comparable outcomes to) NNs and other statistical models, on the most popular benchmark problems. The learning problem setting for SVMs is as follows: there is some unknown and nonlinear dependency (mapping, function) y = f (x) between some high-dimensional input vector x and scalar output y (or the vector output y as in the case of multiclass SVMs). There is no information about the underlying joint probability functions. Thus, one must perform a distribution-free learning. The only information available is a training data set D = {(xi , yi) ∈ X×Y }, i = 1, l, where l stands for the number of the training data pairs and is therefore equal to the size of the training data set D. Often, yi is denoted as di, where d stands for a desired (target) value. Hence, SVMs belong to the supervised learning techniques.

Note that this problem is similar to the classic statistical inference. However, there are several very important differences between the approaches and assumptions in training SVMs and the ones in classic statistics and/or NNs modeling. Classic statistical inference is based on the following three fundamental assumptions:

1. Data can be modeled by a set of linear in parameter functions; this is a foundation of a parametric paradigm in learning from experimental data.
2. In the most of real-life problems, a stochastic component of data is the normal probability distribution law, that is, the underlying joint probability distribution is a Gaussian distribution.
3. Because of the second assumption, the induction paradigm for parameter estimation is the maximum likelihood method, which is reduced to the minimization of the sum-of-errors-squares cost function in most engineering applications.

All three assumptions on which the classic statistical paradigm relied turned out to be inappropriate for many contemporary real-life problems because of the following facts:

1. Modern problems are high-dimensional, and if the underlying mapping is not very smooth the linear paradigm needs an exponentially increasing number of terms with an increasing dimensionality of the input space X (an increasing number of independent variables). This is known as the curse of dimensionality.
2. The underlying real-life data generation laws may typically be very far from the normal distribution and a model-builder must consider this difference in order to construct an effective learning algorithm.
3. From the first two points it follows that the maximum likelihood estimator (and consequently the sum of error squares cost function) should be replaced by a new induction paradigm that is uniformly better, in order to model non-Gaussian distributions.

In addition to the three basic objectives above, the novel SVMs' problem setting and inductive principle have been developed for standard contemporary data sets which are typically high-dimensional and sparse (meaning, the data sets contain small number of the training data pairs).

SVMs are the so-called nonparametric models. Nonparametric does not mean that the SVMs' models do not have parameters at all. On the contrary, their learning (selection, identification, estimation, training or tuning) is the crucial issue here. However, unlike in classic statistical inference, the parameters are not predefined and

Indian J. Innovations Dev., Vol. 1, No. 6 (June 2012) ISSN 2277 - 5390

their number depends on the training data used. In other words, parameters that define the capacity of the model are data-driven in such a way as to match the model capacity to data complexity. This is a basic paradigm of the structural risk minimization (SRM) introduced by Vapnik and Chervonenkis and their coworkers that led to the new learning algorithm. Namely, there are two basic constructive approaches possible in designing a model that will have a good generalization property.

1. Choose an appropriate structure of the model (order of polynomials, number of HL neurons, number of rules in the fuzzy logic model) and, keeping the estimation error (a.k.a. confidence interval, a.k.a. variance of the model) fixed in this way, minimize the training error (i.e., empirical risk), or
2. Keep the value of the training error (a.k.a. an approximation error, a.k.a. an empirical risk) fixed (equal to zero or equal to some acceptable level), and minimize the confidence interval.
3. Classic NNs implement the first approach (or some of its sophisticated variants) and SVMs implement the second strategy. In both cases the resulting model should resolve the trade-off between under-fitting and over-fitting the training data. The final model structure (its order) should ideally match the learning machines capacity with training data complexity. This important difference in two learning approaches comes from the minimization of different cost (error, loss) functional.

Table 1 tabulates the basic risk functional applied in developing the three contemporary statistical models di stands for desired values, w is the weight vector subject to training, λ is a regularization parameter, P is a smoothness operator, Lε is a loss function of SVMs', h is a VC dimension and Ω is a function bounding the capacity of the learning machine. In classification problems Lε is typically 0-1 loss function, and in regression problems Lε is the so-called Vapnik's ε-insensitivity loss (error) function.

$$L_\varepsilon = |y - f(x,w)|_\varepsilon = \begin{cases} 0 & ; \quad \text{if } |y - f(x,w)| \le \varepsilon \\ |y - f(x,w)| - \varepsilon & ; \quad \text{otherwise} \end{cases} \tag{3.16}$$

Where ε is a radius of a tube within which the regression function must lie, after the successful learning. (Note that for ε=0, the interpolation of training data will be performed). It is interesting to note that (Girosi, 1997) has shown that under some constraints the SV machine can also be derived from the framework of regularization theory rather than SLT and SRM. Thus, unlike the classic adaptation algorithms (that work in the L2 norm), SV machines represent novel learning techniques which perform SRM. In this way, the SV machine creates a model with minimized VC dimension and when the VC dimension of the model is low, the expected probability of error is low as well. This means good performance on previously unseen data, i.e., a good generalization. This property is of particular interest because the model that generalizes well is a good model and not the model that performs well on training data pairs. Too good a performance on training data is also known as an extremely undesirable over fitting. In the simplest pattern recognition tasks, support vector machines use a linear separating hyper plane to create a classifier with a maximal margin. In order to do that, the learning problem for the SV machine will be cast as a constrained nonlinear optimization problem. In this setting the cost function will be

Table 1. Basic models and their error (risk) functional

Multilayer Perception(NN)	**Regularization Network (Radial Basis Function Network)**	**Support Vector Machine**
$R=\sum_{i=1}^{l}\underbrace{(d_i - f(x_l,w))^2}$ Closeness to data	$R=\sum_{i=1}^{l}\underbrace{(d_i - f(x_l,w))^2}$ Closeness to data $+\lambda \underbrace{\|Pf\|^2}$ Smooth to data	$R=\sum_{i=1}^{l}\underbrace{L_\varepsilon}$ Closeness to data $+ \underbrace{\Omega\ (l+h)}$ Capacity of α machine

Indian J. Innovations Dev., Vol. 1, No. 6 (June 2012) ISSN 2277 - 5390

quadratic and the constraints linear (i.e., one will have to solve a classic quadratic programming problem). In cases when given classes cannot be linearly separated in the original input space, the SV machine first (non-linearly) transforms the original input space into a higher dimensional feature space. This transformation can be achieved by using various nonlinear mappings; polynomial, sigmoidal as in multilayer perceptions, RBF mappings having as the basis functions radially functions. After this nonlinear transformation step, the task of an SV machine in finding the linear optimal separating hyper plane in this feature space is relatively trivial. Namely, the optimization problem to solve in a feature space will be of the same kind as the calculation of a maximal margin separating hyper plane in the original input space for linearly separable classes.

In a probabilistic setting, there are three basic components in all learning from- data tasks: A generator of random inputs x, a system whose training responses y (d) are used for training the learning machine, and a learning machine which, by using inputs xi and system's responses yi, should learn (estimate, model) the unknown dependency.

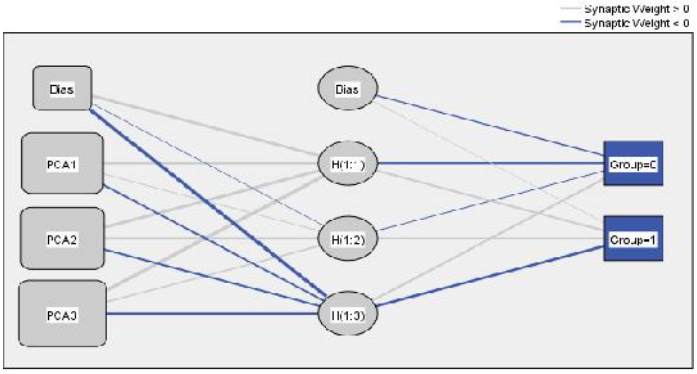

Fig.8. FFNN model for Diabetes data;
Hidden layer activation function: Sigmoid
Output layer activation function: Sigmoid

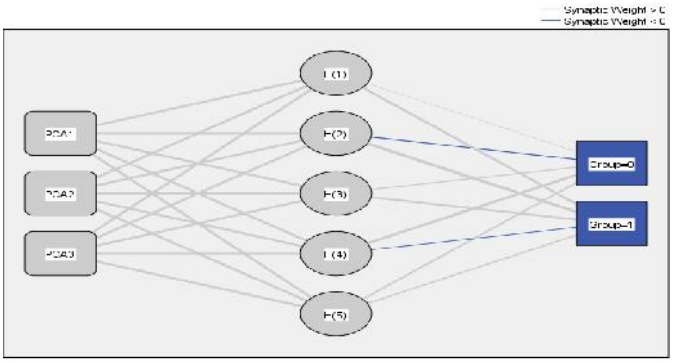

Fig.9. RBFN model for Diabetes data;
Hidden layer activation function: Softmax
Output layer activation function: Identity

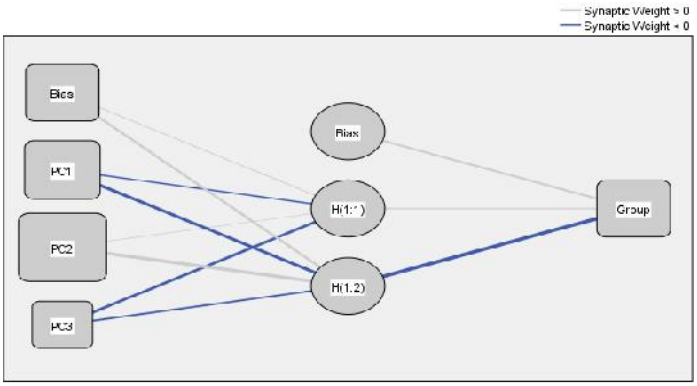

Fig.10. FFNN model for Bromide data;
Hidden layer activation function: Sigmoid
Output layer activation function: Sigmoid

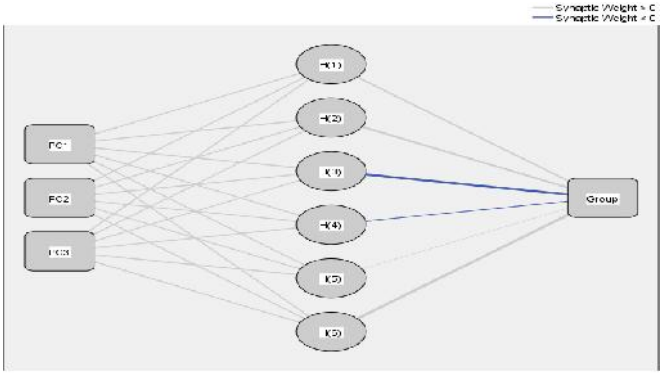

Fig.11. RBFN model for Bromide data:
Hidden layer activation function: Softmax
Output layer activation function: Identity

Application to FTIR pattern analysis

FTIR Spectroscopy is a form of vibrational spectroscopic and the spectrum reflects both molecular structure and molecular environment. A molecule when exposed to the radiation produced by the thermal emission of a hot source absorbs only at frequencies corresponding to its molecular mode of vibration in the region of the

Indian J. Innovations Dev., Vol. 1, No. 6 (June 2012) ISSN 2277 - 5390

Table 2. Parameter estimates for FFNN model (Diabetes data)

Predictor		Predicted				
		Hidden Layer 1			Output Layer	
		H(1:1)	H(1:2)	H(1:3)	[Group=0]	[Group=1]
Input Layer	(Bias)	3.585	-0.024	-3.675		
	PCA1	3.644	0.665	-2.195		
	PCA2	3.594	1.480	-2.233		
	PCA3	3.713	1.647	-3.091		
Hidden Layer 1	(Bias)				-1.052	0.133
	H(1:1)				-2.621	2.388
	H(1:2)				-0.662	0.741
	H(1:3)				2.406	-2.632

Table 3. Parameter Estimates for RBFN Model (Diabetes data)

Predictor		Predicted						
		Hidden Layer[a]					Output Layer	
		H(1)	H(2)	H(3)	H(4)	H(5)	[Group=0]	[Group=1]
Input Layer	PCA1	1.998	.088	-.446	-.957	-.592		
	PCA2	2.000	.084	-.445	-.955	-.592		
	PCA3	2.003	.081	-.442	-.954	-.594		
Hidden Unit Width		.091	.253	.213	.188	.075		
Hidden Layer	H(1)						3.001E-38	1.000
	H(2)						-.015	1.015
	H(3)						.266	.734
	H(4)						1.011	-.011
	H(5)						.682	.318

a. Displays the center vector for each hidden unit

Table 4. Parameter Estimates for FFNN Model (Bromide data)

Predictor		Predicted		
		Hidden Layer 1		Output Layer
		H(1:1)	H(1:2)	Group
Input Layer	(Bias)	0.040	0.347	
	PC1	-0.104	-0.667	
	PC2	0.031	1.345	
	PC3	-0.615	-0.112	
Hidden Layer 1	(Bias)			0.211
	H(1:1)			0.095
	H(1:2)			-1.543

electromagnetic spectrum between visible (real) and short waves (micro waves). These changes in vibrational motion give rise to bands in the vibrational spectrum; each spectral band is characterized by its frequencies and amplitude.

The IR region (1 to 100 μM) is subdivided in three zones, far (100 to 25 μM), mid (25 to 2.5 μM) and near IR (2.5 to 1 μM), The mid IR depicts primary molecular vibration and in the most common and widely used employed region for the analysis of substances in chemistry and forensics. All molecules present characteristic absorbance peaks in a section of this region (1350 cm–1 to 1000 cm–1, 1μM=104 cm–1), thus this physical property in considered as a molecules finger print. The far and near IR are not frequently employed, because only skeletal and secondary vibration occurs in these regions producing spectra that are difficult to interpret many of the IR bands of biological interest occurs in the frequency range between 4000 to 1000cm–1. The FTIR spectrum of a cell

Indian J. Innovations Dev., Vol. 1, No. 6 (June 2012) ISSN 2277 - 5390

will exhibit contribution from all cellular macromolecules including protein, liquid, carbohydrates and DNA. Although the spectra are complex, protein liquids
and DNA provide characteristic non-overlapping contributions of the FTIR spectrum. IR spectroscopy was greatly improved by the use of a new component the interferometer and by the application of fast Fourier Transform algorithm which allow for simultaneous detection of all transmitted energy.

The multi-layer perception (MLP) and radial basis function networks (RBF) are two non-linear feed forward net works and both are universal approximators. Hence we can find an RBF neural network capable of accurately mimicking a specified MLP or vice versa. However the two

methods differ each other. The most common RBF networks has a single hidden layer where as MLP may have more than one. The RBF network trains extremely fast and require fewer training samples then MLP: Also the MLP constitutes global approximations to non linear input-output mapping, but RBF networks uses exponentially decaying localized non-linearities to construct the approximators. The data set is small and we have divided 75 percent for training and 25 percent for testing MLP and RBF networks. The binary logistic regression analysis was also fitted for comparison. All the models were fitted using SPSS 16.0 package. The efficiency of the models is evaluated by sensitivity, specificity and accuracy.

For MLP network architecture, a single hidden layer with sigmoid activation function, which is optional for the diclorotomone out, is chosen. A back propagation algorithm based on conjugate gradient optimization technique was used the model MLP. The RBF network considered for this application was a single hidden layer with Gaussian kernel and the activation function used is symmetric. The cross validation layer error correction method is used. The logistic regression

Table 5. Parameter Estimates for RBFN Model (Bromide data)

Predictor		Predicted						
		Hidden Layer[a]						Output Layer
		H(1)	H(2)	H(3)	H(4)	H(5)	H(6)	Group
Input Layer	PC1	-0.747	-0.550	0.833	-1.316	0.800	-0.399	
	PC2	-2.261	-1.455	0.141	0.721	0.118	0.963	
	PC3	-1.701	.936	0.103	-0.559	-0.507	0.950	
Hidden Unit Width		0.220	.474	0.716	0.887	0.888	0.220	
Hidden Layer	H(1)							1.101
	H(2)							1.163
	H(3)							-1.685
	H(4)							-0.863
	H(5)							0.669
	H(6)							1.898

a. Displays the center vector for each hidden unit.

Table 6. Comparative predications of the Models

Database	Model	Sensitivity (%)	Specificity (%)	Correct Prediction (%)
Diabetes data	Logistic	81.8	77.8	80.3
	MLPNN	81.8	83.3	82.4
	RBFNN	90.9	88.9	90.1
	SVM	90.9	94.4	92.2
Organic data	Logistic	66.7	73.3	70.8
	MLPNN	77.7	80.0	79.1
	RBFNN	88.9	86.7	87.5
	SVM	88.9	93.3	91.7

Indian J. Innovations Dev., Vol. 1, No. 6 (June 2012) ISSN 2277 - 5390

model was fitted using the same input vectors as in the neural network.

The network model for the diabetic organic chemical data and chemical data are given in the Fig.8-11. The corresponding parameter estimates for connection weights are given in the Tables 2-6. The FFNN architecture of diabetic and organic data consists of one hidden layer each. The numbers of nodes in the hidden layer are three and two respectively for diabetic and organic data. The sigmoid activation function is used for hidden and output nodes. The input vectors are rescaled using standardized method and output units are rescaled using normalization method. The error function used is sum of squares.

The RBF network model for diabetic data consists of five hidden nodes and organic data consists of six hidden nodes. For both the networks the inputs were standardized. The softmax activation function was for hidden nodes and identify activation function was used for the output unit. The sum of squares error function was used to adjust the weights.

References

1. Gail A Carpenter, Stephen Grossberg and Rosen DB (1991) Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4, 759-771.
2. Herrero J, Valencia A and Dopazo J (2001) A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics*, 17, 126-136.
3. Selaru FM, Xu Y and Yin J (2002) Artificial neural networks distinguish among subtypes of neoplastic colorectal lesions. *Gastroenterology*.122, 606-613.
4. Vishal Gupta (2009) A Survey of Text Mining Techniques and Applications. *J Emerg. Technol. web Intelligen*. 1(1), 60-76.
5. Yegnanarayana (1994) Artificial neural networks for pattern recognition, 19(2), 189-238.