Biomedical Engineering and Information Systems: Technologies, Tools and Applications

Anupam Shukla
Indian Institute of Information Technology and Management Gwalior, India

Ritu Tiwari
Indian Institute of Information Technology and Management Gwalior, India

Medical Information Science REFERENCE

MEDICAL INFORMATION SCIENCE REFERENCE

Hershey · New York

Director of Editorial Content: Kristin Klinger
Director of Book Publications: Julia Mosemann
Acquisitions Editor: Lindsay Johnston
Development Editor: Dave DeRicco
Publishing Assistant: Natalie Pronio
Typesetter: Michael Brehm
Production Editor: Jamie Snavely
Cover Design: Lisa Tosheff
Printed at: Lightning Source

Published in the United States of America by
Medical Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Biomedical engineering and information systems : technologies, tools, and applications / Anupam Shukla and Ritu Tiwari, editors.
p. ; cm.
Includes bibliographical references and index.
Summary: "Bridging the disciplines of engineering and medicine, this book informs researchers, clinicians, and practitioners of the latest developments in diagnostic tools, decision support systems, and intelligent devices that impact and redefine research in and delivery of medical services"--Provided by publisher.
ISBN 978-1-61692-004-3 (h/c)
1. Biomedical engineering. 2. Medical informatics. I. Shukla, Anupam, 1965- II. Tiwari, Ritu, 1977- [DNLM: 1. Biomedical Engineering--methods. 2. Bioengineering--methods. 3. Biotechnology--methods. 4. Information Systems. W 36 B615 2010]
R856.B495 2010
610.28--dc22
2010009790

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 8
Handling Large Medical Data Sets for Disease Detection

Rahul Kala
Indian Institute of Information Technology and Management Gwalior, India

Anupam Shukla
Indian Institute of Information Technology and Management Gwalior, India

Ritu Tiwari
Indian Institute of Information Technology and Management Gwalior, India

ABSTRACT

The breakthrough in the field of intelligent systems has spread its fruits to the field of biomedical engineering as well; where a series of models are being applied to automatically detect diseases based on some parameters or inputs. The continuous research in this field has resulted in a large amount of database being created for many diseases which becomes very difficult to train. Also the number of attributes is under constant rise. This increases the dimensionality of the problem and ultimately leads to poor performance. In this chapter we deal with the methods to handle these situations. We discuss the mechanism to divide data between different sub-systems. We also discuss the method of division of the attributes to reduce the training time and complexity. The resultant systems are able to train better due to low computational cost and hence give better performance. We validated this with the Breast Cancer database from the UCI Machine Learning repository and found our algorithm optimal.

INTRODUCTION

Computational Intelligence has given rise to automation in numerous spheres. Bio-medical engineering is one of these spheres where the computationally intelligent systems have automated the task of detection of diseases. These systems are able to affectively detect the presence or absence of any disease. These systems take the various parameters as inputs that can potentially affect the decision regarding the presence or absence of disease. The system analyzes these inputs and then makes its decision. Every input or parameter affects the decision of the system to some extent. Some parameters are very important whereas the others may behave rather passive in nature. Ideally the larger the number of parameters, the more is the chance of detection of the disease. This is

DOI: 10.4018/978-1-61692-004-3.ch008

because of the fact that the system is given more information that may be consulted for decision making.

The detection system usually makes use of historic data to formulate rules regarding the analysis. The historic data is a collection of large amount of information regarding the presence or absence of disease in multiple scenarios. The system tries to extract valuable information or rules from this database by the process of machine learning. It then tries to generalize the extracted information or rules to the unknown data. The entire knowledge of the system to detect the disease is based on the learning of the historical data. Hence it can effectively only effectively predict the presence or absence of some disease if the behavior was recorded by the system sometimes earlier and presented at the time of learning as the historical database. In case the same does not hold, the system might not behave as desired and it may give any random or abrupt output. Hence ideally a larger historical database is preferred that can capture as much diversity as possible. The aim is that in the future any unknown input is related to one of the inputs in the historical database.

The large number of attributes in the system as well as the large number of training data size in the historical database seems to be a big boon for the system. However, this may have a negative impact on the system in terms of computational time. The larger number of attributes or data set instances has a multiplicative impact on the time of training. It may hence be possible that the system is unable to train itself in finite time. This would drop the efficiency of the system and the entire system would become sub-optimal. Further, the larger number of attributes makes the system very flexible. The entire system can now assume more behavior by different values of outputs for different inputs. In other words there is an increase in the number of parameters of the systems that require more data and time for training. There is always an associated risk of sub-optimal training as a result of the added attributes. Hence an increase in attributes or the number of inputs is not always desirable.

The larger number of instances of training data in the training data set also has negative affects apart from the time complexity being increased. More data added to the system may add unique characteristics to the system which is to be modeled. This may be an exception to the present state and behavior of the system. As a result the system would have to formulate new rules or learning to accommodate the input that acts as an exception. This would add to the complexity of the system. Further the complex systems are more computationally expensive and difficult to train due to the large system parameters. Also the training may be sub-optimal. This puts a serious limitation to the increase of data.

It may not always be the case that adding attributes or training instances is poor for the system. Some added attributes showing behavior of correlation to one or more of the already existing attributes may not make the system that sensitive. Similarly if all of the added inputs show normal behavior as the rest of the inputs in the database, there would not be much adverse affect on the system performance. In such cases we may consider the addition to play role in time only. It may at the same time help in optimal system tuning.

Modular Neural Networks (MNNs) (Shadabi, Sharma and Cox, 2006) are neural networks that exploit the modularity in the problem. The whole problem is divided into a set of modules which are Artificial Neural Networks (ANNs) in themselves. All the modules behave independent of each other. Each module tries to solve its part of the problem independently. After all the modules have solved the problem, the results are communicated to the integrator. The integrator is responsible for combining the results of the individual modules to form the result of the entire problem. In such a manner the problem is solved by the combined effect of the various modules. Each module is ultimately a neural network that has to undergo training before it can be put to use. The training of

the various modules is independent of each other and is carried out by their part of the problem. Modular neural networks are effective means to solve problem where the conventional system is unable to solve the problem due to its complex size or other limitations. These networks give a good performance boost to the system with a very little loss of generality.

Large number of attributes or data instances is not always desirable. However not considering any of them would make the system miss out valuable piece of information, which might be needed for optimal training. Hence in this chapter we study the manner in which we can make the system perform well in computationally feasible time, at the same instance not delete any valuable piece of information. We study both the problems independently one by one. First we would be discussing the problems of a large number of attributes in the training data set. For this we would distribute the attributes among different modules of a modular neural network. All these compute their result and the same are used by the integrator to compute the final result.

The second problem is one large training data size. For this problem also we make use of modular neural network. The first step is to cluster the training data into clusters. Each cluster has a separate neural network of its own. The training data set is divided into cluster and each cluster is trained by the inputs that belong to that cluster. For unknown input we first compute the cluster to which it belongs. We then use the neural network of the same cluster to compute the final output. This is the output of the system.

This chapter is organized as follows. In section 2 we present some of the works from literature. Section 3 discusses the problem of division of attributes. In section 4 we discuss about the clustering based division of training data set. The results to solve the problem of Breast Cancer are given in Section 5. We give the directions for future research in section 6 and the conclusions in section 7.

BACKGROUND

Modular Neural Networks are hybrid systems that effectively solve the problem. A lot of work is being done in use of hybrid systems for problem solving. Rutwoski (2004) did a major work to build flexible neuro fuzzy systems. Here he presented various models build of neuro fuzzy systems. Kuo et al (2008) proposed a hybrid model for learning fuzzy if-then rules. These have been applied to variety of problems (Grigore and Gavat 1998; Lee et al 1999, Taur and Tao, 2000). A lot of work is going on in the various parts of these algorithms like clustering, genetic optimizations, rule forming, parameter updating etc (Er and Zhou 2008; Mu et al 2006; Sandhu, Salaria and Singh, 2008). The algorithms try to optimize the performance in clustering by designing various models based on the architecture of neuro-fuzzy systems (Chaudhuri and Bhattacharya, 2000; Lin, Chun and Cheng 2007, Lin and Hong, 2007; Pinero et al 2007, Vieira and Barradas, 2003). Adams (2009) proposed the use of Genetic Algorithms to decide the connectivity of associative memory that can be generalized to artificial neural networks. Maravall et al (2009) also proposed a hybrid model of Genetic Algorithms with reinforcement learning for robotic controllers.

The problem of detection of disease is classificatory in nature where we need to classify the inputs to either of the two output classes that stand for the presence or absence of disease. Classification is a major problem of study in literature. People are trying to better adapt the present algorithms to make them more suitable for classificatory problems. Amin et al. (2009) presented a model of single layered complex valued artificial neural network for classificatory problem. Here they had made use of new activation functions. Hu (2008) used Choquet Integral with neural network and genetic algorithms for pattern classification. Estudillo et al (2008) proposed multiplicative nodes instead of additive in neural networks to build neural network classifiers. Ang, Tan and

Al-Mamum (2008) used probability to genetically evolve a neural network from smallest size to larger sizes. They proposed variable probabilities.

Genetic Algorithms are being constantly studied and modified for better performances at various situations. One of the major landmark here is the evolution of Particle Swarm Optimizations. Nani (2009) used the Nearest Neighbor approach for prototype reduction using Particle Swarm Optimization and the found the Nearest Neighbor approach to be good. Grammatical Evolution is another area of work. Tsoulus et al (2008) proposed grammatical evolution for neural network construction and training. O'Neill et al (2001 and 2005) has highlighted various developments in these fields in terms of representation and working of the algorithm. Ryann, Collins and O'Neil (2005) proposed these algorithms for program generation in arbitrary language.

A lot of work is being done in the field of neural network for validating, generalizing and better training of the neural network (Daghici, 1997; Gerlot, Flink and Thomas 2006; Graves et al 2008; Shadabi, Sharma and Cox, 2006). The problems also employ the concept of Hidden Markov Models (Hewavitharana, Fernando and Kodikara 2002; Sandhu, Salaria and Singh, 2008; Shi, Shu and Liu 1998). These are statistical models that can be used to predict the consequence of the unknown input. These models have found a variety of use in handwriting recognition, speech recognition etc. Instantaneously trained neural networks (Kak 1998; Rajagopal 2003) are a good approach for faster training with a smaller generalization capability. These networks require very less training time as the weights are decided just by seeing the inputs. Self organizing maps have also been used extensively for the problem solving. Various other mathematical models have been proposed. These employ mathematical techniques like point to point matching for solving the problem.

DIVISION OF ATTRIBUTES

The first problem of study is where we have a large number of attributes on which the decision regarding the presence or absence of disease depends upon. Consider the case of skin diseases it has 34 attributes that are considered for deciding the presence or absence of the disease (Ilter, Guvenir and Altey, 1998). Such a large number of attributes need to be computed which has its own advantages and disadvantages as discussed. We hence build a modular neural network over this system that helps in solving the problem.

In this problem, we first divide the problem into modules. Here the attributes are divided into modules. Every module gets a set of attributes. It is possible that an attribute is given to more than one module. Similarly it is possible that an attribute is not given to any of the attributes at all. This however must be avoided as it leads to a loss of valuable information.

Then each module independently calculates the output. These use ANNs for the task. At the end each ANN returns the probabilities of the occurrence of each of the class. These probabilities lie in the range of -1 to 1. A '-1' means that the given class is definitely absent. Similarly a '1' means that the class is present for sure. The numbers denote the certainty of the ANN in the occurrences of the class as the final output class.

At the end the integrator does the task of combining the individual solutions to give the final output. This is in the form of 1st averaging the various probabilities returned by the individual systems. Here the performance of the systems at the time of training is used as weights. Then the summation takes place to return the final class that is declared as the final output. The basic working of the system is shown in Figure 1. We discuss each of the steps in the next section.

Figure 1. The general structure of the algorithm

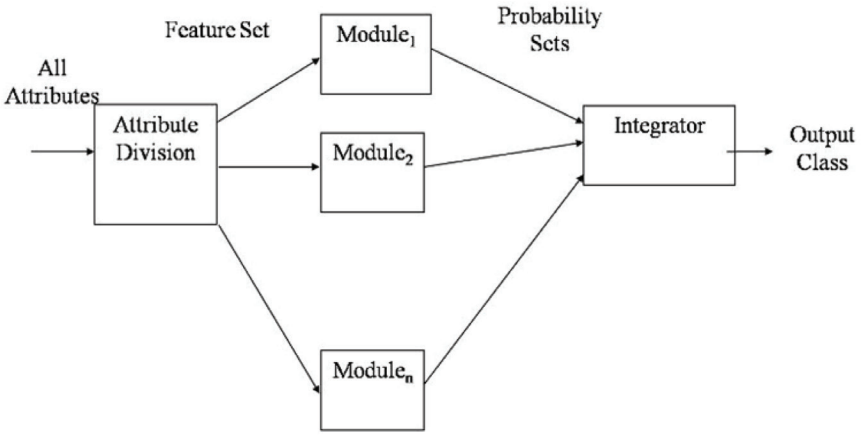
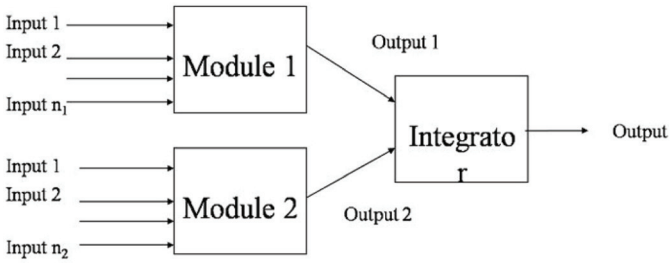


Figure 2. The division of inputs between modules



Modules

Here we are supposed to exploit the modularity in the features. We would be dividing the entire feature set into the various modules. The basic motive is to ensure that each module after getting its feature set must be in a condition to appreciably solve the problem. It must have the related attributes to enable it to do so. Hence the attributes given to any module must be diverse and must collectively supply the entire information. Redundancy in features may be removed. The redundant features would unnecessary increase computation. Also loading too many features to an ANN would be not desirable. This would also increase the computation on it unnecessarily. Here we also try to ensure that all attributes collectively get the complete feature set. This would avoid the loss of information from the system.

Consider that the disease being considered had '*a*' attributes $X<X_1, X_2, X_3, ..., X_a>$. Let us say we are to divide this attribute set X into b modules. In such a case we form b new modules each consisting of a set of X_is such that each attribute X_i belongs to some or the other module Y_j. It must be additionally kept in mind that the distribution must be such that each module Y_j must be able to give a decent performance even by its own. There must be a very high correlation between attributes of different modules and a very low correlation between the attributes of same module.

The structure in general related to the division of the features among modules is shown in Figure 2. Here the input comes to the system and is divided among these modules. Each module

calculates its output and gives it to the integrator to decide the final output.

Artificial Neural Networks

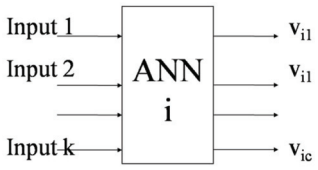
The job of classification of the inputs is carried out with the help of ANNs with Back Propagation Algorithm as the training algorithm. The ANNs are a natural choice because of their ability to learn from the historical data and to generalize the results. The ANNs map any input to some class or person here. We use a classificatory model of ANN here. This has as many output neurons as the number of classes. Each output neuron stands for some person or class. The output at this neuron for any input *i* is the probability of occurrence of this person or class according to the ANN. Hence the ANN gives as its output c number of probabilities in the output vector. Let this output vector for any input *i* be represented by $<v_{i1}, v_{i2}, v_{i3}, \ldots v_{ic}>$.

The probabilities here are measured in the range of -1 to 1. A probability of 1 means that according to the ANN this is the output class with full confidence. On the other hand a probability of -1 means that according to the ANN this class is not the output with full confidence. Hence the ideal output for any input for the ANN should be a 1 for any one element of the output vector and a -1 for all the other elements. However due to practical reasons, the output lies anywhere in the complete region. Figure 3 shows the system inputs and outputs with the ANN as a black box.

Integrator

The last part to implement according to the entire algorithm is the integrator. This is the major part of the whole system that does the task of finding the final output class after getting inputs from the individual modules. The integrator analyzes all the outputs by the various modules and then decides the final output class that according to the system is the output.

Figure 3. The ANN inputs and outputs



The input given to the integrator is the solution vector of every module of ANN. Let the vector of the module *i* be $<v_{i1}, v_{i2}, v_{i3}, \ldots v_{ic}>$ where each v_{ik} is the probability of the occurrence of the class *k* measured on a scale of -1 to 1. The integrator decides the output by first taking the weighted averages of the probabilities given to it and then selecting the class with the maximum probability. This class is declared as the winner. This is shown in Figure 4.

The weighted average is calculated for each and every class in the system on which the output can map to. The weights of the various modules or ANNs here are their performances on the training data. Each ANN was given the same data set for the training purposes with a different feature set. The performance here is calculated as a ratio between the numbers of elements the ANN correctly classifies in the validation data set by the total number of elements in the validation data set. The higher the performance of the ANN in the validation data set, the more would be its weight and more dominant it would be to decide the final output at the time of integration.

Using these calculated weights, the integrator calculates the weighted average for all the classes. This gives the final probability vector comprising of c probabilities with each probability associated with some class. These probabilities again lie in the range of -1 to 1 with the same meaning of the probabilities.

The next task of the integrator is to find out the final output class. For this the integrator selects the output class with the largest value of the probability out of all the available classes. The class or the person corresponding to this largest

Figure 4. The Integrator

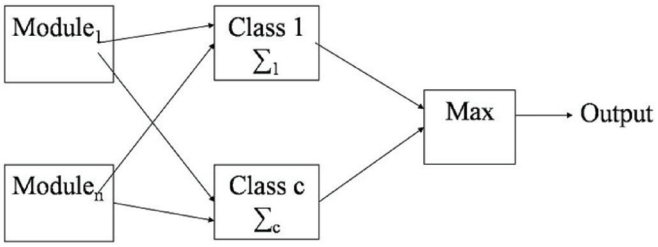
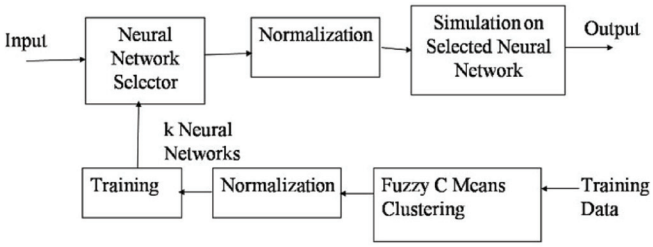


Figure 5. The hierarchical training algorithm



probability is declared as the winner and this is the output that the system gives.

DIVISION OF TRAINING DATA

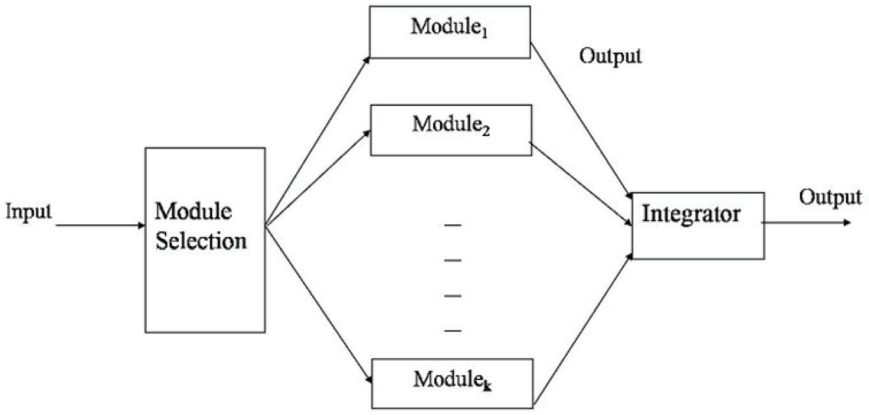
The other problem that we have considered is of division of the training data into a set of modules. We know that a large amount of training data has an adverse affect on training. Besides it exhibits a complex relation between the inputs and the outputs which is difficult to model. In this approach we try to divide the entire problem into independent modules that can be independently trained and tested. Here each module is a cluster in the input space. In other words we may say that we divide the entire multi-dimensional input space into a set of clusters. Each cluster has a vague boundary which demarcates it from the other clusters. Further, each cluster has a separate neural network of itself through which it is trained.

A conventional Artificial Neural Network (ANN) gives the inputs to the system and the output is calculated by the processing by the various nodes of the system. Every time an input is given from the input nodes and the output is collected after processing from the output nodes. The system consists of a number of parallel processing nodes that work on the input to generate the output.

In this approach we first cluster the input training data into *k* number of clusters. Each of the *k* clusters is a module of the modular neural network. It represents an ANN. This module or ANN caters to the needs of inputs that lie within its cluster range. Hence every neural network or cluster has a boundary. Any input that lies within this boundary is processed by this neural network and the corresponding output is generated solely by this network. This is decided by the clustering of the input test data.

The major steps of the algorithm are given in Figure 5. As it can be seen from figure, the algorithm has two phases, training and testing. The purpose of training is to make the *k* modules and decide their clusters. The training data is clustered using Fuzzy C Means clustering. This gives us the center of every cluster. We also get the members of every cluster. Every cluster has

Figure 6. The hierarchical structure of the algorithm



its own ANN. This ANN is trained independently of all other networks using the members of this cluster. When this process is over for all the *k* ANNs, we have the networks ready for the testing. In testing phase, whenever an input is given, we select the best cluster or module that is closest to the input. We then process the input by this ANN or module. The general architecture of the system is given in Figure 6.

We discuss more about the various phases of the algorithm in detail in the next few subsections. We discuss the two phases of training and testing, one by one.

Training

The purpose of this stage is to form the *k* ANNs or modules, each trained by its set of inputs. Each module has its own range in which it operates. Fuzzy C Means Clustering is a clustering algorithm that is extensively used for clustering of input data in various applications. In this algorithm, we need to specify the number of clusters in advance. The algorithm clusters the input data into these many clusters. We get the centers of the clusters at the end. This clustering algorithm is called a fuzzy algorithm, as any data element is not a member of just one cluster, rather it is associated with more than one clusters each with a different membership degree. This clustering algorithm also gives us the membership degrees of the various inputs.

In our algorithm, we first of all need to decide the number of clusters. This value depends of the level of modularity of the problem. The higher value of *k* would result in larger number of ANNs. The inputs would be distributed over a larger number of modules. This would improve the training time but result in a loss of generalization. By analyzing the training data set and the system requirements, we decide the value of *k*. The other parameters that we need to fix in this algorithm are Maximum number of iterations, Minimum improvement and exponent. Once we run the algorithm, the data is analyzed and the various clusters are formed. At the end of the algorithm, we get to know the centers of the *k* clusters that we had intended to find.

Suppose that our training data is $<I_{11}, I_{12}, I_{13}, I_{14}....I_{1n}>$, $<I_{21}, I_{22}, I_{23}, I_{24}....I_{2n}>$.....$<I_{m1}, I_{m2}, I_{m3}, I_{m4}....I_{mn}>$ Using Fuzzy C Means Clustering we get the *k* centers as $<C_{11}, C_{12}, C_{13}, C_{14}....C_{1n}>$, $<C_{21}, C_{22}, C_{23}, C_{24}....C_{2n}>$ ..... $<C_{k1}, C_{k2}, C_{k3}, C_{k4}....C_{kn}>$.

Now we allocate every input to a corresponding cluster. This is decided based on the distance between the input and the cluster. We select the cluster with the least distance from the given input. The distance between any two points $X<x_1$,

$x_2, x_3, x_4 .... x_n>$ and $Y<y_1, y_2, y_3, y_4 .... y_n>$ is given in equation (1).

$$d(X,Y) = \sum_{i-1}^{n} (x_i - y_i)^2 \quad (1)$$

The cluster p to which any given input I belongs is given by equation (2). We select only one value of p.

Cluster(I) = { p: d(I, C_p) ≤ d(I, C_i) for all 1≤ i ≤ k } (2)

We look at all the training data and decide the cluster to which they belong. We group up data and hence at the end each cluster has some input training data associated with it. Through this step, we have clustered the input data into various clusters.

The input data may be having very high ranges. Since we are using back propagation algorithm, we need to make sure that data in every step lies within workable ranges. If the input data is very high, it is possible that it may overshoot the workable range in any step of internal processing. This is a common observation while working with the ANN toolkit of MATLAB. If the input data sizes are even slightly higher, the results are badly affected. Hence it is usually preferred to keep all inputs between -1 to 1. This improves the performance. In order to make sure that the inputs are within range, we apply a step of normalization. In this step every input is divided by the maximum value that the input can take. Usually we take this maximum value as the maximum value that is present in the training input. This is given by equation (3).

$$I_{new} = \frac{I_{old}}{\prod_{i=1}^{cluster\ size} Max\left(\left|I_i\right|\right)} \quad (3)$$

If the outputs are also quite large, then we need to normalize them as well. This can be done by dividing them with the maximum output, as we did for the input. This would ensure that the output is within the range of –1 to 1, under which the artificial neural network would be able to perform well.

These maximum values need to be preserved, as we would need them in the testing phase as well. While testing also, whenever an input is applied, we first find out the cluster to which it belongs. Then we divide the input by the same value as we did for all the training data of that cluster.

The most commonly used method for training is the Back Propagation Algorithm (BPA). In this algorithm, the inputs are applied one after the other and the outputs are calculated. The actual results to each of these inputs are known. The difference is calculated between the actual and the observed outputs. This error is back propagated to all the layers. The layers uses the concepts of steepest descend to calculate the new weights. The weights are changed to the new weights. Next time when same inputs are applied, the errors are smaller. This process is repeated for a few iterations or epochs.

We have the k clusters ready, each with a set of normalized data sets from the previous step. We use the back propagation algorithm in all these k ANNs and train them independently. The limited inputs are given and hence training is a lot better and faster.

Testing

Once the k modules are ready, we need to test them for unknown inputs. This is done in the testing phase of the algorithm. We need to select one of the k modules. This selection is done by finding the cluster that is closest to the given input. This selection is the same as given in equations (1) and (2). We need to normalize the input as we did for the training data. Every cluster had a maximum value per input which we used to divide the input to make it lie within the range of -1 to 1. We divide the input by the same value as we did for all the training data of that cluster. This value is given in equation (3).

Once we have selected the module and the normalized input to be processed. We now apply the input to this module or ANN using the forward pass of the artificial neural network. The various nodes process the inputs and pass the results to the subsequent layers. The final answer is given by the output node. This is the output to the given input that may be interpreted as per the requirements of the system. It may be noted that the integrator always gets a response from a single module in the presented approach. The integrator simply outputs this value as the final system output.

RESULTS

In this section we present an experimental framework for the problems discussed. The problem that we choose for the experimental purposes is the detection of Breast Cancer. Here we are given a set of attributes and these needs to be classified into malignant or benign. We take the breast cancer data from the UCI Machine Learning Repository for this purpose (Wolberg, Mangasarian and Aha, 1992). This database consists of 29 real valued inputs. These correspond to the following features for each cell nucleus: radius (mean of distances from center to points on the perimeter), texture (standard deviation of gray-scale values), perimeter, area, smoothness (local variation in radius lengths), compactness (perimeter2 / area - 1.0), concavity (severity of concave portions of the contour), concave points (number of concave portions of the contour), symmetry, fractal dimension ("coastline approximation" - 1). The entire data set consists of a total of 357 benign and 212 malignant cases, totaling to 569 instances in the database.

The basic methodology consists of first division of the database into training and testing data sets. We keep a total of approximately 70% data for the training purpose and the rest 30% for the testing purpose. The entire algorithm is applied as per sections 3 and 4 on the training and the testing data sets. The final performance for both of these is noted. We discuss the results to both the experiments of division of attributes and division of training data sets one by one in the next sub-sections.

Attribute Division

First we took the data from the data set and divided it into 2 separate modules. There were a total of 30 attributes in the data set. We took the first 15 attributes for the first module and the next 15 attributes for the second part. There was no repetition or sharing of attributes between these two modules. The whole data set comprising of training and testing data set was hence partitioned vertically into separate parts. Each of these was trained separately. The integrator was made that carried forward the task of combining the results of these two modules.

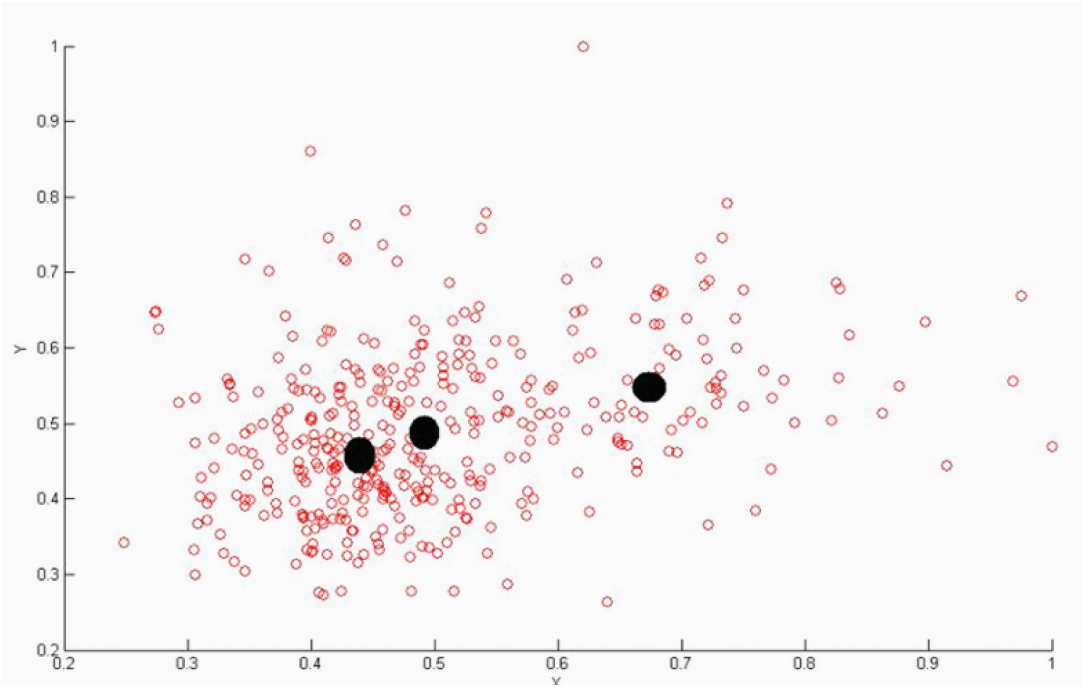
Each of the two ANNs used in the problem had the same architecture. They were separately trained using Back Propagation Algorithm. There was 1 hidden layer consisting of 4 neurons. The hidden layer had an activation function of '*tansig*' and the output layer had an activation function of '*purelin*'. The learning rate was fixed to be 0.05. Momentum was kept as 0.8. The maximum number of epochs were 3500.

The system so formed with two ANNs and an integrator, making a complete modular neural network gave a performance of 98.21% in the training data where it could identify 385 of the 392 cases. The efficiency of the testing data was 94.35% with the identification of 167 out of 177 cases presented. The system hence gave a good performance.

Training Data Division

The next problem that we have is where we need to divide the training data set into clusters and assign each ANN a different cluster. We make 3 clusters where the data would be clustered into.

Figure 7. Fuzzy C means clustering of training input data



Fuzzy C means clustering was used for the clustering purposes. 100 iterations were used with a minimum improvement of 10^{-5} and exponent of 2. The clustered plot of two of the axis from the input space is shown in Figure 7. Here the black dots denote the cluster center.

From this we get the training centers for further processing. 3 different ANNs are made based on these clusters. Each of them is given its share of the training input to train with. Later the testing is performed. Here also we first select the cluster or the ANN and then find the output corresponding to the input.

All the three ANNs used in the problem had the same architecture. They were separately trained using Back Propagation Algorithm. There was 1 hidden layer consisting of 6 neurons. The hidden layer had an activation function of '*tansig*' and the output layer had an activation function of '*purelin*'. The learning rate was fixed to be 0.05. Momentum was kept as 0.8. The maximum number of epochs were 3500.

The entire system in the stated configuration gave an accuracy of 91.3% in the training data set and 90.5% in the testing data set. Again it may be seen that the system was able to effectively solve the problem of diagnosis of the breast cancer.

FUTURE RESEARCH DIRECTIONS

In this chapter we presented solutions to the common problems of handling large data sizes. The solutions proposed were effective in diagnosis of breast cancer where we achieved large computational speedup with a minute loss of generality. The data set chosen for this problem was itself quite small in nature. For future, work may be carried out on data sets which are massively large in nature and fail to train by a single method. This would be the ideal testing of the proposed algorithm. These data sets may be large horizontally or vertically. The small computational availability has been the reason for low pace of research in

numerous fields. The proposed methods promise an effective solution where the data can be easily partitioned to solve the problem. The algorithm may be used over newly built large datasets of large sizes.

We have proposed two independent solutions to the problem of large data sizes. The former divides the data vertically by division of the attributes. At the same time the other divides the data horizontally by the division of instances. These methods may be combined in an intelligent manner to solve complex problems. The design, training and testing of this combined approach may be carried out in future. This would effectively solve the problem of data sets which are very vast in nature. As more and more problems are coming under automation, there is a growth in the number of attributes as well as the entire data set size. The combined method would prove to be very beneficial in such scenarios.

In the proposed algorithm we made use of ANN with Back Propagation Algorithm. Various other models may be tried like Self Organizing Maps, Learning vector Quantization, etc. Also the application of Hybrid Systems including Neuro-Fuzzy Systems, Genetic Training and Evolution of the System may be studied in the future.

CONCLUSION

In this chapter we discussed the problem and solution of having large data set. The size could either be large due to a high number of input attributes that caused a problem in the system training and testing. The size could also be large due to the size of the training data instances. These problems were solved by the use of modular neural networks. For the first problem we divided the input attributes in different modules. This division resulted in a reduced workload as well as reduced the problem complexity. For the second problem we made the division horizontally where the data set was divided by clustering into different data segments or clusters. Each cluster was trained by a separate ANN.

The algorithm was validated by executing it against the breast cancer database. The database contained 30 attributes and 569 instances. The attributes were divided into two sets for the first part of the problem. For the second part of the problem, the data was divided into three clusters using Fuzzy C Means clustering. This made the problem less complex that could easily be solved.

We divided the data set available into training and testing data sets. The training data set was used for training the system with the designed algorithm. The testing was performed by the testing data set. The efficiency of the algorithm to correctly identify the disease from the testing dataset is given as the final diagnostic efficiency. Using this system we could get a sufficiently large efficiency of diagnosis of the disease. The computational time was much less. As a result the system could optimally train itself. This solved the problem of training and testing of large data sets that is becoming an issue of concern for many problems.

ACKNOWLEDGMENT

This work has been supported and sponsored by Indian Institute of Information Technology and Management Gwalior.

REFERENCES

Adams, R., Calcraft, L., & Davey, N. (2009). Using a genetic algorithm to investigate efficient connectivity in associative memories. *Neurocomputing*, *72*(4-6), 732–742. doi:10.1016/j.neucom.2008.05.017

Amin, M. F., & Murase, K. (2009). Single-layered complex-valued neural network for real-valued classification problems. *Neurocomputing*, *72*(4-6), 945–955. doi:10.1016/j.neucom.2008.04.006

Ang, J. H., Tan, K. C., & Al-Mamun, A. (2008). Training neural networks for classification using growth probability-based evolution. *Neurocomputing*, *71*(16-18), 3493–3508. doi:10.1016/j.neucom.2007.10.011

Chaudhuri, B. B., & Bhattacharya, U. (2000). Efficient training and improved performance of multilayer perceptron in pattern classification. *Neurocomputing*, *34*, 11–27. doi:10.1016/S0925-2312(00)00305-2

Conor, R., Collins, J. J., & Neill, M. O. (n.d.). *Grammatical Evolution - Evolving Programs for an Arbitrary Language*

Draghici, S. (1997). A neural network based artificial vision system for license plate recognition, International Journal of Network Security. *International Journal of Neural Systems*, *8*(1). doi:10.1142/S0129065797000148

Er, M. J., & Zhou, Y. (2008). A novel framework for automatic generation of fuzzy neural networks. *Neurocomputing*, *21*(4-6), 584–591. doi:10.1016/j.neucom.2007.03.015

Estudillo, F. J., Martinez, C. H., Gutierrez, P. A., & Estudillo, A. C. (2008). Evolutionary product-unit neural networks classifiers. *Neurocomputing*, *72*(1-3), 548–561. doi:10.1016/j.neucom.2007.11.019

Gernot, A., & Fink, P. & Thomas (2006). Unsupervised Estimation of Writing Style Models for Improved Unconstrained Off-line Handwriting Recognition. In *Proceedings of the Tenth International Workshop on Frontiers in Handwriting Recognition.*

Graves, A., Fernandez, S., Liwicki, M., Bunke, H., & Schmidhuber, J. (2008). Unconstrained Online Handwriting Recognition with Recurrent Neural Networks, In Proceedings of Advances in Neural Information Processing Systems.

Grigore, O., & Gavat, I. (1998). Neuro-fuzzy Models for Speech Pattern Recognition in Romanian Language. In *Proceedings of European Symposium on Intelligent Techniques* (CONTI'98). Timisoara, Romania, (pp. 165–172).

Hewavitharana, S., Fernando, H. C., & Kodikara, N. D. (2002). Off-line Sinhala Handwriting Recognition using Hidden Markov Models

Hu, Y.-C. (2008). Nonadditive grey single-layer perceptron with Choquet integral for pattern classification problems using genetic algorithms. *Neurocomputing*, *72*(1-3), 331–340. doi:10.1016/j.neucom.2008.01.008

Ilter, N., Guvenir, H., & Altay (1998). *UCI Machine Learning Repository*. Rerieved from http://www.ics.uci.edu/~mlearn/MLRepository.html

Kak, S. C. (1998). On generalization by neural networks. *Information Sciences*, *111*, 293–302. doi:10.1016/S0020-0255(98)10009-9

Kala, R., Shukla, A., & Tiwari, R. (2009). Fuzzy Neuro Systems for Machine Learning for Large Data Sets. In *Proceedings of the IEEE International Advance Computing Conference*, (pp. 541-545) Patiala, India.

Kuo, R. J., Hong, S. M., Lin, Y., & Huang, Y. C. (2008). Continuous genetic algorithm-based fuzzy neural network for learning fuzzy IF–THEN rules. *Neurocomputing*, *71*(13-15), 2893–2907. doi:10.1016/j.neucom.2007.07.013

Lee, S. G., et al. (1999). A Neuro-Fuzzy Classifier for Land Cover Classification. In *Proceedings of the 1999 IEEE International Fuzzy Systems Conference Proceedings*, Seoul, Korea.

Lin, C. J., Chung, I. F., & Chen, C. H. (2007). An entropy-based quantum neuro-fuzzy inference system for classification applications. *Neurocomputing*, *70*(13-15), 2502–2516. doi:10.1016/j.neucom.2006.08.008

Lin, C. J., & Hong, S. J. (2007). The design of neuro-fuzzy networks using particle swarm optimization and recursive singular value decomposition. *Neurocomputing*, *71*(1-3), 297–310. doi:10.1016/j.neucom.2006.12.016

Maravall, D., Lope, de, Martin, J., & Antonio, J. (2009). Hybridizing evolutionary computation and reinforcement learning for the design of almost universal controllers for autonomous robots. *Neurocomputing*, *72*(4-6), 887–894. doi:10.1016/j.neucom.2008.04.058

Mu, C. S., Chien, H. C., Eugene, L., & Jonathan, L. (2006). A new approach to fuzzy classifier systems and its application in self-generating neuro-fuzzy systems. *Neurocomputing*, *69*(4-6), 586–614. doi:10.1016/j.neucom.2004.11.033

Nani, L., & Lumini, A. (2009). Particle swarm optimization for prototype reduction. *Neurocomputing*, *72*(4-6), 1092–1097. doi:10.1016/j.neucom.2008.03.008

O' Neill, M., & Brabazon, A. (2005). Recent Adventures in Grammatical Evolution. In *Proceedings of CMS 2005 Computer Methods and Systems.* (Vol. 1, pp. 245-253).

O' Neill, M., & Ryan, C. (2001). Grammatical Evolution. *IEEE Transactions on Evolutionary Computation*, *5*(4).

Pinero, P. (2004). Sleep stage classification using fuzzy sets and machine learning techniques. *Neurocomputing*, *58–60*, 1137–1143. doi:10.1016/j.neucom.2004.01.178

Rajagopal, P. (2003). *The Basic Kak Neural Network with Complex Inputs* (Master of Science in Electrical Engineering Thesis). Tamil Nadu, India: University of Madras

Rutkowski, L. (2004). *Flexible Neuro-Fuzzy Systems, Structures, Learning and Performance Evaluation*. Amsterdam: Kluwer Academic Publishers.

Sandhu, P. S., Salaria, D. S., & Singh, H. (2008). A Comparative Analysis of Fuzzy, Neuro-Fuzzy and Fuzzy-GA Based Approaches for Software Reusability Evaluation. In *Proceedings of Worlds Academy of Science* (*Vol. 29*). Engineering and Technology.

Shadabi, F., Sharma, D., & Cox, R. (2006). Learning from Ensembles: Using Artificial Neural Network Ensemble for Medical Outcomes Prediction, *Innovations in Information Technology*, 1-5.

Shi, D., Shu, W., & Liu, H. (1998). Feature Selection for Handwritten Chinese Character Recognition Based on Genetic Algorithms. In *Proceedings of the Intl' Conf. on Systems, man and Cybernetics*, (Vol. 5, No 5, pp. 4201-4206).

Som, T., & Saha, S. (2008). Handwritten character recognition by using Neural-network and Euclidean distance metric. *Social Science Research Network*. Retrieved from http://ssrn.com/abstract=1118755

Soryani, M., & Rafat, N. (2006). Application of Genetic Algorithms to Feature Subset Selection in a Farsi OCR. In *Proceedings of World Academy of Science* (*Vol. 18*). Engineering and Technology.

Taur, J. S., & Tao, C. W. (2000). A New Neuro-Fuzzy Classifier with Application to On-Line Face Detection and Recognition. *The Journal of VLSI Signal Processing*, *26*, 397–409. doi:10.1023/A:1026515819538

Tsoulos, I., Dimitris, G., & Glavas, E. (2008). Neural network construction and training using grammatical evolution. *Neurocomputing*, *72*(1-3), 269–277. doi:10.1016/j.neucom.2008.01.017

Vieira, A., & Barradas, N. (2003). A training algorithm for classification of high-dimensional data. *Neurocomputing*, *50*, 461–472. doi:10.1016/S0925-2312(02)00635-5

Wolberg, W. H., Mangasarian, O. L., & Aha, D. W. (1992). UCI Machine Learning Repository, University of Wisconsin Hospitals. Retrieved from http://www.ics.uci.edu/~mlearn/MLRepository.html

KEY TERMS AND DEFINITIONS

Artificial Neural Networks: Mathematical model consisting of parallel processing neurons that perform information processing in a layered manner. They are excellent agents for regression, functional prediction or classification.

Classification: Problems in which the input always lies in one of the available classes based on the input attributes.

Fuzzy C Means Clustering: A clustering algorithm that forms fuzzy cluster where each member is associated to every cluster by some degree of membership that lies in a scale of 0 to 1.

Generalization: Ability of a system to give correct outputs to unknown inputs.

Hybrid Systems: Systems made by a combination of soft computing systems (Artificial neural Networks, Evolutionary Algorithms or Fuzzy Logic) along with heuristics.

Input Space: A multi-dimensional space plotting the output values of the Artificial Neural Network (or any other system) for every combination of inputs for a specific system state.

Integrator: Part of the Modular Neural Network that carries forward the task of division of problem into modules and the computation of the final output by the individual module output.

Machine Learning: Forming rules, trends or patterns in the historical database, so that the correct outputs may be computed whenever this data is presented again to the systems.

Modular Neural Networks: Neural Networks that work on the modularity in the problem. The problem is divided into modules and given to various sub-systems that independently solve part of the problem. The overall result is communicated to an integrator and the final output is computed.

Modules: Independent Neural Networks that solve a part of the whole problem in a Modular Neural Network and communicate their results to the integrator.