Interdisciplinary Advances in Information Technology Research

Mehdi Khosrow-Pour
Information Resources Management Association, USA

Information Science
REFERENCE

Interdisciplinary Advances in Information Technology Research

Mehdi Khosrow-Pour
Information Resources Management Association, USA

Information Science
REFERENCE

Chapter 3
Breast Cancer Diagnosis Using Optimized Attribute Division in Modular Neural Networks

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

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

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

ABSTRACT

The complexity of problems has led to a shift toward the use of modular neural networks in place of traditional neural networks. The number of inputs to neural networks must be kept within manageable limits to escape from the curse of dimensionality. Attribute division is a novel concept to reduce the problem dimensionality without losing information. In this paper, the authors use Genetic Algorithms to determine the optimal distribution of the parameters to the various modules of the modular neural network. The attribute set is divided into the various modules. Each module computes the output using its own list of attributes. The individual results are then integrated by an integrator. This framework is used for the diagnosis of breast cancer. Experimental results show that optimal distribution strategy exceeds the well-known methods for the diagnosis of the disease.

INTRODUCTION

There has been a vast amount of research in the use of neural networks for problem solving. The neural networks are extensively used for a variety of problems including biometrics, bioinformatics, robotics, and so forth (Shukla, Tiwari, & Kala, 2010a). The ease of modelling and use makes the neural networks good problem solving agents. The neural networks carry the task of machine learning. Here a training database is given to the system. This database is a source of large amount of information regarding patterns, trends, and knowledge about the problem domain. The task of the learning

DOI: 10.4018/978-1-4666-3625-5.ch003

algorithm is to extract this knowledge and use it as per the system knowledge representation. In the neural network this knowledge is in the form of weights between the various neurons and the individual neuron biases. A commonly used architecture of the neural networks is the Multi-Layer Perceptron. Here the various neurons are arranged in a layered manner, the first layer being the input layer and the last being output layer. The input and output layer may be separated by a number of hidden layers. Back Propagation Algorithm is commonly used for training the neural networks. This algorithm works over the gradient descent approach for fixing the various weights and biases. The back propagation algorithm is however likely to get struck at some local minima, considering the very complex nature of the search space over which it operates (Konar, 1999).

The weakness in the various soft computing paradigms has led to the emergence of the field of hybrid soft computing. Here we mix two similar or different paradigms so as to magnify the advantages of each of these and diminish their disadvantages. This coupling of individual systems may result in complementation of the limitations of the systems, for an overall enhanced performance. The evolutionary neural networks are commonly used hybrid systems, where neural modelling fuses with evolutionary computation to result in good problem solving agents.

The architecture of the neural networks is a major criterion that decides the system performance. The traditional neural networks use human expertise to design the optimal architecture, which may then be trained by the training algorithm. This however is a human-intensive task which may hence yield sub-optimal results. The training algorithm in turn may get struck at some local minima, with very poor exploration of the search space. The evolutionary algorithms are very strong optimizing agents that optimize the given problem in an iterative manner, and fix all the values of the parameters so as to optimize the final objective (Mitchell, 1999). Evolutionary neural networks hence use the optimization potential of the evolutionary algorithms for evolving the complete architecture of the neural networks, along with the weights and biases (Nolfi, Parisi, & Elman, 1990; Yao, 1999). Many times the evolutionary process may be assisted by a local search strategy like BPA or simulated annealing to search for local minima in the vicinity of the current location of the evolutionary individual in the search space (Yao, 1993).

Classification is a fundamental problem of study. The classification system is given a set of features as inputs, and is expected to return the class to which the input belongs as the output. The classifier is supposed to build the decision boundaries in the feature space that separates the various classes. Ideally the features must be such that the various instances of the classes have a high inter-class separation and low intra class separation. This makes it very easy for the classifier to construct decision boundaries across the various classes, separating them from each other. Every input attribute in this classifier is a dimension in the feature space. The additional dimensions usually make the task of construction of the decision boundary by the classifier easier. Two classes lying very close to each other may get separated by the addition of some dimension. This however may require more training instances, and would result in immense increase of computation time. The decision boundaries, across various dimensions, may become very complex and difficult to model and train (Shukla, Tiwari, & Kala, 2010b; Kala, Shukla, & Tiwari, 2009). Hence the number of inputs to the classifier needs to be limited in nature.

Modular Neural Network is advancement over the conventional neural networks. Here we try to introduce modularity into the structure and working of the neural network. This leads to the creation of multiple modules that together solve the entire problem. The results generated by the different modules are integrated using an integrator. Each of the modules of the modular neural network is a neural network that aids in

the solution building. These networks can hence model very complex problems and give effective decisions in smaller times (Fu et al., 2001; Gruau, 1995; Jenkins & Yuhas, 1993). A related concept is ensemble, where the same problem is solved by a number of experts. Each of them computes the output to the problem which is then integrated using an integration mechanism (Dietterich, 2000; Hansen & Salamon, 2000; Jacobs, et al., 1991).

The immense increase in computation has led to automation in the diagnosis of disease. A considerable effort is being given for the use of computation and engineering principles in the field of medical sciences. This leads to an exciting field of Biomedical Engineering. The automatic diagnosis of disease helps in the early detection of diseases, and hence suitable preventive measures may be taken. These systems may be used to assist the doctors for decision making (Bronzino, 2006; Shukla & Tiwari, 2010a, 2010b). Breast Cancer is an emerging problem which has attracted the interests of a large number of researchers. A number of diagnostic techniques are built for the diagnosis of this disease. These systems take the various attributes observed in the patient and try to predict the presence or absence of disease (Breastcancer.org, 2010; Janghel et al., 2009, 2010; Shukla et al., 2009a, 2009b).

Attribute division is a novel concept where we use multiple neural networks for solving the problem in place of a single neural network. Each of the neural networks is given a part of the complete set of attributes. As a result every neural network or module gets a problem of limited complexity. Each network tries to train itself so as to make the best prediction as per the attributes given. Each network has a limited view of the feature space, comprising of limited dimensions. This is unlike the view of the complete feature space. Each network tries to make the best prediction as per its view. The predictions of all the neural networks are integrated using an integration mechanism, and the final decision is made. Here we try to integrate the views of the various modules, to optimally construct the global view of the complete feature space. As per the working guidelines of ensemble, it is known that the various modules must all give high performance, and must disagree as much as possible (Pedrajas & Fyne, 2008). These contradictory conditions, when held together solve the problem efficiently. The various modules give diverse view to the integrator which makes the final decision. The detail not seen by some module gets compensated by another module.

In this paper we propose a method of deciding the distribution of attributes among the various modules of the modular neural network. A large set of attributes is given as the input, using which a set of modules need to be made. An attribute can be given to any number of modules or to none at all. The ultimate aim is to make network that gives the best performance over the testing database. For this reason we penalize the network structure with large number of attributes which may give a higher training accuracy but a low generality. Diversity in attributes is further encouraged.

RELATED WORK

A considerable amount of work is done into the domain of evolutionary and modular neural networks in the past decade. A review of the various evolutionary approaches, operators, and other concepts behind the evolution of fixed and variable architecture neural networks can be found in the work of Yao (1999). The use of Evolutionary Programming for a behavioural evolution of the neural network is found in (Yao, 1997). Symbiotic Adaptive Neuro-Evolution (SANE) is a major development into the field (Moriarty, 1997; Moriarty & Miikkulainen, 1997). This algorithm uses the concept of co-evolution for carrying the task of optimization of the neural network architecture and parameters (Potter, 1997; Rosen & Belew, 1996; Stanley & Miikkulainen, 2004). Here it is assumed that the neural network consists of a single hidden layer. A hidden layer may be connected to

any number of neurons from the input and output layer, which is optimized as the algorithm runs. The evolution takes place at two levels. The first level is the neuron level. Here the genetic individual is a neuron connected to some other neurons with some weights. The next level is the network level. Here the individual is a collection of neurons that makes the complete neural network. Pedrajas (2003) proposed a novel framework of using co-operative evolution or co-evolution for the task of modular neural network evolution. Their solution used two levels of evolution. The first level was the nodule level. Here the individuals corresponded to the individual neural networks. The next level was the network level which tried to figure out the best combination of nodules for an effective overall recognition. The fitness function used in this algorithm encouraged the individuals to possess a good overall fitness, as well as rewarded them for adding unique characteristics to the network. The results showed a better performance of these networks over the conventional approaches.

Fieldsend and Singh (2005) used Multi-objective optimization to evaluate the evolutionary neural network for a set of error functions into a pareto front. The use of multiple error functions enabled strong check against generalization loss or over-fitting. This neural network was further extended to use a validation data set to avoid overfitting, and booststrapping to make use of a number of small data sets for training and validation. The net decision was made using the training on validation errors in all these sets. Jung and Reggia (2006) present another interesting approach where the users are provided with a language specification they can use to tell the system about the general architecture of the neural network. The architectural parameters to be optimized may be explicitly specified. The system carries the rest of the evolution as per the user set architectural specifications. In this manner the human expertise and evolutionary optimizing potential interact at user front for the generation of optimal neural network. Rivera *et al*. (2007) used co-operative evolution for the task of generation of Radial Basis Function network. Each individual here was a neuron of this network or a radial basis function. The impact of the neuron was measured against its performance, error and overlapping with the other neurons. The final evaluation was done using a Fuzzy Rule Based system. This enabled the neurons to attain diverse roles, which collectively made an effective network. Cho and Shimohara (1998) used Genetic Programming for the generation of Modular Neural Network. Here the chromosome was framed to model the architecture and parameters of the various modules of the modular neural network. Different types of modules were used to performed different functions.

Boosting is another novel concept applied for effective machine learning. Here we assign different weights to the different data instances, which denote their ease of being learned. The more difficult instances have a greater impact on the final network error. These weights are updated as per the system readings of errors (Freund, 1995; Freund & Schapire, 1996). Pedrajas (2009) presents an interesting application to boosting. In his approach multiple classifiers are made. Computation of the boosting weights takes place as the system learns. The projection of input space to the hidden layer space (outputs of the hidden layer) is passed as inputs to next classifier.

Division of the entire input set into multiple input sets for easier and better recognition is a commonly used task. A good use of modular neural network can be found in the work of Melon and Castilo (2005). Here the authors carried out the task of multi-modal biometric fusion. Each biometric modality was handled separately by a modular neural network, which were all integrated using fuzzy integration. Each of the biometric identification system was a modular neural network consisting of one module for each part of the biometric modality. Each modality input was broken into three parts, each part being performed by a different neural network. The parts were also integrated using fuzzy integration. A similar con-

cept was applied by Kala et al. (2010) for bi-modal biometric recognition. Here the entire pool of attribute set from both modalities was distributed into four recognition neural networks. All outputs were integrated using probabilistic sum technique.

METHODOLOGY

The basic motive behind the approach is to devise an optimal distribution of the attributes of the problem, into the classifiers. We assume here that the number of classifiers is already fixed. Let there be *n* classifiers into the system. Let the complete attribute set be given by $<p_1, p_2, p_3, ...p_N>$. Here *N* is the total number of attributes in the problem of consideration. It is assumed that any classifier may have any number of attributes. Further it is assumed that any attribute may be given to any number of classifiers. Once a classifier gets the stated attributes, it trains itself with the training data set. Training algorithm may be specific to the classifier. We propose the use of Genetic Algorithm for carrying out this distribution. The Genetic Algorithm Framework is discussed. The construction and training of the neural network is then presented. The fitness evaluation is also given.

Genetic Algorithm

The first task in the implementation of the genetic algorithm is the individual representation technique. The individual in our case consists of *n* number of vectors $< V_1, V_2, V_3, V_4, ... V_n>$. Each vector V_i contains the set of attributes that are given to classifier *i*. Each set is a collection of integers denoting the attributes it is assigned. It is natural that the number of attributes may vary from 1 to *N*. Further we assume that the various neural networks used would have a single hidden layer. This is a valid assumption considering the fact that most problems give best results when trained and tested with a small number of neurons and a single hidden layer. The number of neurons in the neural network for every module constitutes the structure of the module or neural network. Hence every vector is appended with the number of neurons present in the module. This can be any integer between 1 and neu_{max}. neu_{max} is the maximum allowable number of neurons that is specified by the user. Based on the same mechanism an initial population is designated. Random attribute numbers are assigned to the various vectors. The number of attributes in the vectors is itself determined randomly. The number of neurons of various modules is also assigned randomly.

The next task is the use of genetic operators to aid in the evolutionary process. For this we use a variety of genetic operators. These are (i) crossover, (ii) mutation, (iii) elite, (iv) add attribute, (v) delete attribute, (vi) mutate number of neurons, and (viii) repair.

The crossover operator uses two parents to generate two children. The crossover is carried for each of the *n* attribute sets, the results are then combined to get the final genetic individual. Let the two attribute sets be V^x_j and V^y_j. Here *x* and *y* are the two parents, and *j* is the attribute set. Let n^x_j and n^y_j be the number of attributes in these sets. The major problem in crossover is the difference in the number of attributes in each of the attribute sets. For this we first make a pool of distinct attributes from both these attribute sets V^x_j and V^y_j. The number of attributes in the two children are kept as $ceil((n^x_j \text{ and } n^y_j)/2)$ and $floor((n^x_j \text{ and } n^y_j)/2)$. Where *ceil(.)* is the greatest integer greater than function, and *floor(.)* is the greatest integer less than function. These many attributes are randomly given to the two children using a scattered crossover technique. Every attribute from this attribute set is given to the first parent with a probability of 0.5, and to the other parent with a probability of 0.5. The left spaces in the children are filled with random attributes, provided they are not currently in the possession of the children. The number of neurons for the children per module is changed using the same phenomenon. If neu^x_j and

neu^y_j denote the number of neurons of module *j* of the parents *x* and *y*, the number of neurons in children is given by $ceil((neu^x_j \text{ and } neu^y_j)/2)$ and $floor((neu^x_j \text{ and } neu^y_j)/2)$.

The mutation operator simply replaces an attribute randomly with a new attribute. The replacement is governed by the mutation rate that follows a normal distribution. Elite passes the best individual of a generation directly to the next generation. Add attribute selects an attribute set of the individual randomly. It then adds a new distinct attribute to this attribute set. The delete attribute genetic operator performs in a similar manner. It however deletes an attribute from the available list. The mutate neurons selects a module randomly and changes the number of neurons in it. The change follows a normal distribution.

The repair operator checks the individuals and removes any anomalies in them. One of the constraints in the individuals is that the various vectors cannot contain the same attribute more than once. This is natural since multiple occurrence of an input attribute would not be useful for the neural network training. Hence the repeated attributes are deleted by this operator. Further the various attributes are stored in a sorted order in their attribute sets. The sorting of the attributes is done by this operator. The number of neurons must also be within the designated limits. Every attribute also needs to be within the limited values. The number of attributes must be greater between 1 to *N*.

Modular Neural Network

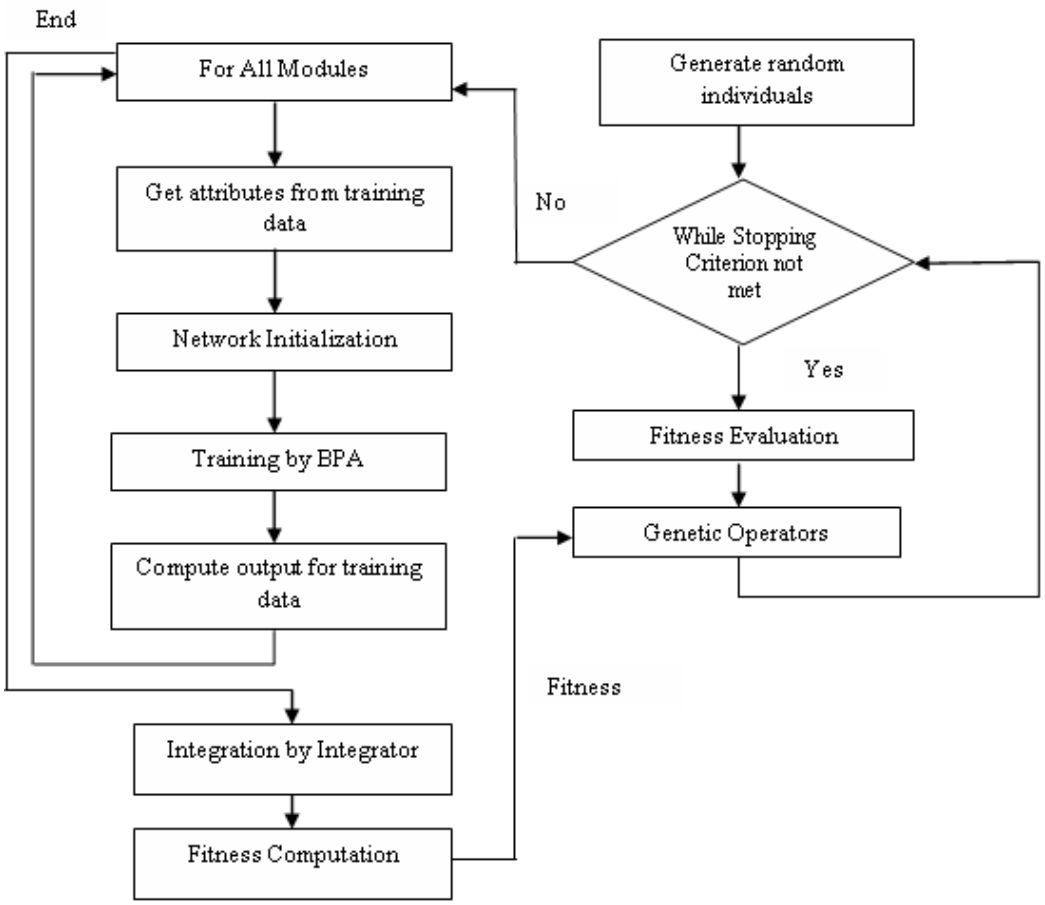
The other major task is the formulation of modular neural network using the specifications of the genetic individual. This would enable the computation of the fitness of the genetic individual as per previous discussions. Each of the modules of the modular neural network is a multi-layer perceptron that is trained using back propagation algorithm. Each neural network gets the designated attributes as represented by the genetic individual. The network is supposed to output the diagnosis of the disease.

Each individual of the genetic algorithm is decoded to make the designated number of modules. Each module further has designated number of neurons. Each network is trained using back propagation algorithm. Only the selected attributes are given to the network for training, out of all the available attributes. Figure 1 shows the general evolutionary and neural architecture used in the algorithm. It may be easily seen from figure that the entire algorithm is based on evolutionary principles where Genetic Algorithms carry optimization. Each individual, for fitness computation, is decoded which results in many modules, each being a neural network. All modules are trained by the specific attributes represented by the individual from training database. Trained modules are then used for performance evaluation over training database. Integration of results for various modules is done by the integrator. Figure 2 specifically stresses upon the mechanism of division of attributes and the further processing with the neural network. The entire set of attributes available in the problem is divided into a set of modules. Each attribute may be given to none, one, or more than one module. Each module is a neural network that takes the selected attributes as input and produces a probability set as output. The integrator sums the probabilities recorded by various modules and declares the class with maximum probability as the winning class.

Fitness Evaluation

The last task associated with the use of genetic algorithm is to devise a fitness function. The fitness function is an evaluation of the performance of the individual. Here we try to simultaneously optimize four objectives. These are (i) Diagnostic Performance (DP), (ii) Left Attributes (LA), (iii) Average Module Size (AMS), and (iv) Difference

Figure 1. The general algorithm framework



in attributes within modules (DA). Diagnostic performance is a measure of the number of cases correctly diagnosed by the system and is measured as given in Equation (1)

DP = Correctly Diagnosed Cases/Total Number of Cases (1)

This is a number between 0 and 1.

The Left Attributes (LA) measures the total number of attributes that did not go to any of the modules. This means that the resultant system is deprived of these attributes, which may lead to sub-optimal performance. We hence try to make this number as small as possible. LA is normalized to lie between 0 and 1 by dividing by the total number of attributes *N*.

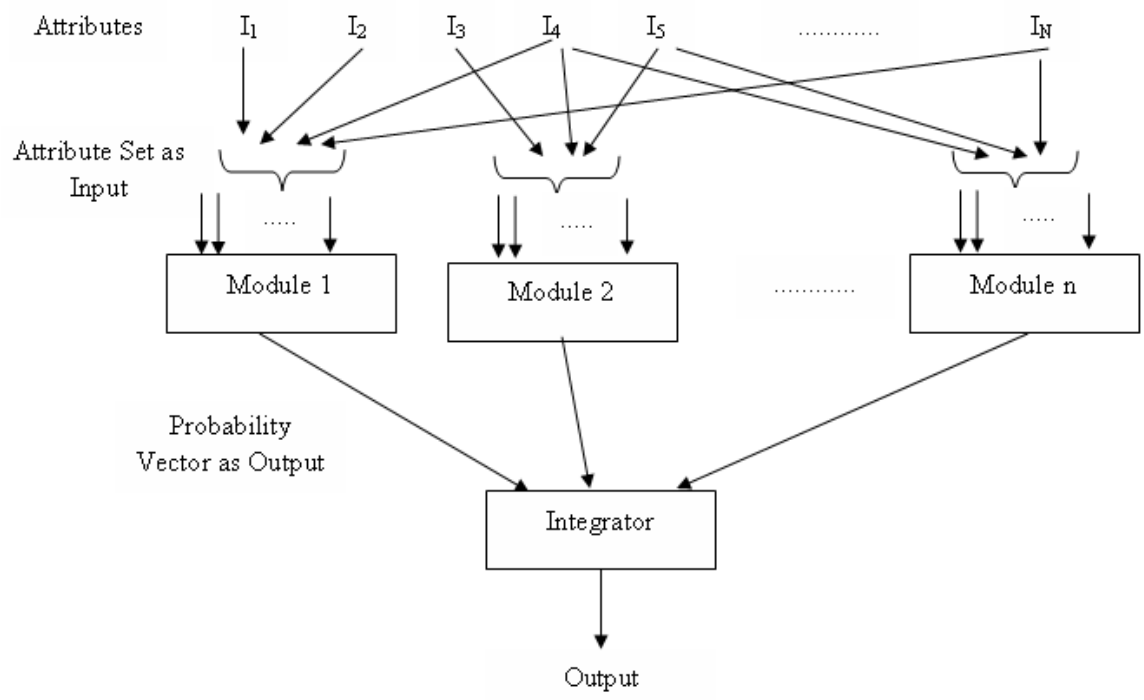
Average Module Size (AMS) is measured as given by Equation (2). This is also normalized to lie between 0 and 1 by division by *N*.

$$AMS = \frac{\sum_{i=1}^{n} n_i}{n} \quad (2)$$

Here n_i is the number of attributes in module *i*. *n* is the total number of modules.

The difference in number of attributes (DA) is an indication of the variance in the size of the various modules. We keep this number as small as possible, so that the different modules are similar

Figure 2. Attribute division in the modules



in size and action. DA is measured as given in Equation (3).

$$DA = \sum_{i=1}^{n} \sum_{j=1, j>i}^{n} \left| n_i - n_j \right| \quad (3)$$

The fitness function is a weighted sum of all these objectives given by Equation (4). The signs denote maximization or minimization.

$$\text{Fit} = - \alpha_1 \text{DP} + \alpha_2 \text{LA} + \alpha_3 \text{AMS} + \alpha_4 \text{DA} \quad (4)$$

Here $\alpha_1, \alpha_2, \alpha_3,$ and α_4 are the multi-objective optimization constants.

RESULTS

The discussed model was applied over the problem of Breast Cancer diagnosis. 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, & Aha, 1992). This database consists of 30 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 (perimeter^2/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.

JAVA was used as the simulation platform. The evolutionary algorithm used for the same was coded in the same platform. Back Propagation algorithm was taken as a library from (Cole, 2009). The various modules of evolutionary algorithm, individual, fitness evaluation, data manager, and

neural networks were made and integrated. The system had three modules that were kept constant throughout the simulation. The initial population was initially generated randomly, that was later optimized using the evolutionary algorithm.

The entire data set is broken for both training as well as testing data sets. Approximately 70% of the instances are randomly chosen for training and the rest 30% and kept for testing. Each data item gets designated to training or testing datasets. The evolutionary parameters used constitute of population size of 25 and stopping criterion of 25 generations. At any generation 40% individuals came from crossover, 20% from mutation, 5% from elite, 10% from mutate neuron, 10% from add attribute, and 10% from delete attribute. The maximum number of neurons in any module was limited to 20. The four multi-objective weights had a value of 0.4, 0.4, 0.1, and 0.1. Back Propagation Algorithm was used as a local search strategy. Learning rate was fixed to 0.1, and momentum was kept as 0.7. The training was carried for 1000 generations. On training and testing the algorithm gave an accuracy of 98.63% for the testing data and 98.10%for the testing data. The algorithm took approximately 10 hours for the evolution. The results of the algorithm are summarized in Table 1.

Table 1. Analysis of the results of the algorithm

S. No.	Property	Value
1.	Mean Training Accuracy	98.10%
2.	Mean Testing Accuracy	98.63%
3.	Approx. Training Time	10 hours
4.	Mean Correctly Identified Instances (Training)	391
5.	Mean Incorrectly Identified Instances (Training)	7
6.	Mean Correctly Identified Instances (Testing)	168
7.	Mean Incorrectly Identified Instances (Testing)	3

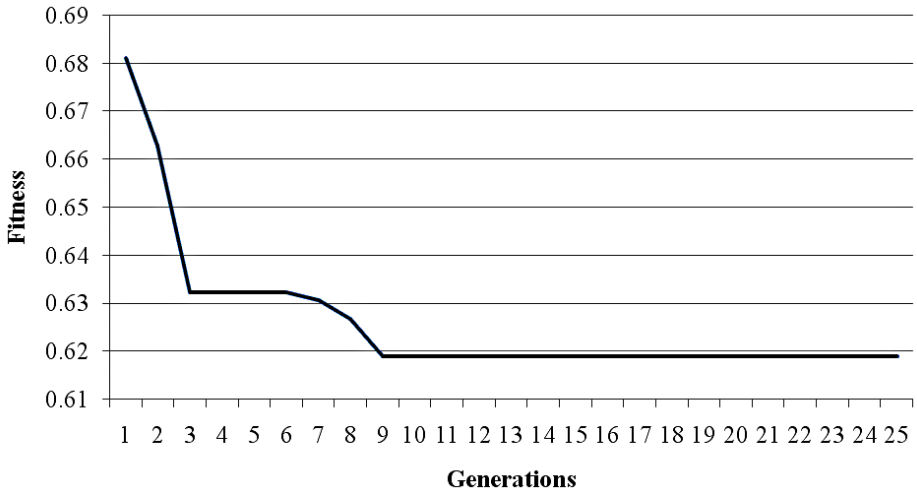
We further study the convergence of the algorithm by plotting the fitness of the best individual of the population along with time. This is given in Figure 3. The figure clearly shows a large improvement in the performance value in the initial few generations. This improvement however becomes small as the generations increase. Towards the later stages, the algorithm converges to the optimal value. It may be noted that the database taken had limited instances, and hence the training accuracy can only increase by some discrete amounts, corresponding to the increase in accuracy due to a single data instance.

The high accuracies achieved in the use of proposed algorithm encourage a high usage of the algorithm for medical diagnosis. In order to fully test the performance of the algorithm, we compare the proposed algorithm with a number of algorithms available in literature. In all these algorithms the data was broken down into training and testing data sets. The training data set was used for network tuning the network parameters. The testing dataset was used for the testing purposes.

The first method applied was the conventional neural network model Multi-Layer Perceptron trained with Back Propagation Algorithm. Here MATLAB was used for the simulation purposes. The network had 1 hidden layer with 18 neurons. Learning rate was fixed to be 0.05. Momentum was fixed to 0.7. The network was trained for 3500 epochs. The resultant network gave a training accuracy of 97.01% and a testing accuracy of 94.61%.

The other model trained was a fixed architecture evolutionary neural network. Here the neural network architecture was the same as discussed in the previous approach. The weights and biases were optimized using genetic algorithm. The genetic algorithm consisted of 10 individuals, each trained in 15 generations. The various weights and biases could vary from -2 to 2. Rank based scaling with stochastic uniform selection was used. Elite count was kept as 2. Crossover rate was 0.8. Gaussian mutation with a spread and scale of 1 each was used. The genetic algorithm used back propaga-

Figure 3. Performance of best networks against generations



tion algorithm as the local search strategy. The BPA had a learning rate of 0.05, and momentum of 1. Training was carried for 30 epochs. The algorithm had a training accuracy of 93.92% and testing accuracy of 95.40%.

The next algorithm we use to test the accuracy with is the variable architecture Evolutionary Neural Network. Here we follow a connectionist approach. The neural network is assumed to be consisting of one hidden layer. In place of an all-connected architecture, we assume that only some connections are allowed from the input layer to hidden layer and hidden layer to output layer. The information regarding the connections is stored into the genetic individual. The various parameters used in this approach are the same as the ones used in the fixed architecture neural network. Extra connections were penalized by assigning a penalty of 0.01 per connection. The hidden layer could have a maximum of 30 neurons. The accuracy in this case was 97.01% for the training data and 95.21% for testing data.

The next approach tried to test the accuracy of the algorithm was ensemble. Here we made 3 experts, each being a neural network with similar architecture as discussed in the previous approaches. These three neural networks gave their probability vectors corresponding to the various classes as outputs. A probabilistic sum of these vectors was taken, to get the final output vector. The winning class was determined from this vector. On training and testing, the system gave an accuracy of 97.98% on the training data and 95.95% on the testing data.

The other method applied for testing the accuracy of the algorithm was modular neural network, where the complete feature space was partitioned into three modules. Each of these modules was given a separate neural network for training. The

Table 2. Comparisons between various algorithms

S. No.	Algorithm	Training Accuracy	Testing Accuracy
1.	Proposed Algorithm	98.10%	98.63%
2.	MLP with BPA	97.01%	94.61%
3.	Fixed Architecture Evolutionary ANN	93.92%	95.40%
4.	Variable Architecture Evolutionary ANN	97.01%	95.21%
5.	Ensemble	97.98%	95.95%
6.	Feature Space Modular ANN	96.49%	95.08%
7.	Attribute Modular ANN	97.19%	96.03%

architecture of the neural network was the same as discussed earlier. After training and testing, the system gave an accuracy of 96.49% on the training data set and 95.08% on the testing data set.

The next method of application was a random attribute division using Back Propagation Algorithm. Here we try to control the curse of dimensionality by using multiple neural networks in a modular manner. Three modules were made. Every attribute was given randomly to two of the three modules. The three modules were trained and tested against the respective data sets. On simulation, the training accuracy was found to be 97.19% and testing accuracy was found to be 96.03%.

The various algorithm results have been summarized in Table 2.

From Table 2, it is clear that the proposed algorithm gave the best generalization and testing accuracy as compared to all the methods presented. Hence using this approach we have been successful in removing the adverse effects of curse of dimensionality in an effective manner using an evolutionary approach. The high recognition score illustrates an effective diagnostic system.

CONCLUSION

Curse of dimensionality is a major problem that the neural networks face. Modularity is a welcome step to remove the curse of dimensionality problem with the least loss of information. One of the primary ways of implementation of modularity is by attribute division. In this paper we used genetic algorithms for devising an effective division of attributes amongst the various modules of the modular neural network. The system so formed tried to make as effective modules as possible, with limited size and limited number of attributes. The overall modular neural network used these modules for the decision making. Each module was a multi-layer perceptron which was trained using back propagation algorithm. The modules returned the probabilities of the occurrence of the various classes, which was combined using an integrator using sum rule. This made the final decision. The genetic algorithm not only worked for formulating the effective attribute division strategy, but also evolved the optimal neural network architecture.

This algorithm was applied to the problem of diagnosis of Breast Cancer. The attributes needed to be divided into a set of three modules. On training and testing, the resultant system recorded an accuracy of 98.10% on the training data and 98.63% on the testing data. The performance of the algorithm was compared with a number of algorithms known in literature. We observed that the proposed algorithm gave the best performance in terms of both the training and testing data. This shows the high diagnostic capability of the network. Hence we have been able to remove the problem of dimensionality from the conventional neural network to a good extent by working over effective modularity.

The present algorithm is used over a single database from biomedical engineering. The algorithm may be further extended to more databases from diverse applications. This would enable use of the algorithm in different fields. Time complexity is another major problem associated with the algorithm. Effective models may be built to control execution time. This would enable an even better optimization. The approach is primarily used using the genetic algorithm with multi-layer perceptron model. More combinations of neural and genetic models may be tried for better perforce. All this may be done in future.

REFERENCES

Breastcencer.org. (2010). *Understanding the Breast Cancer*. Retrieved February 1, 2010, from http://www.breastcancer.org

Bronzino, J. D. (2006). *Biomedical Engineering Fundamentals*. Boca Raton, FL: CRC Press.

Cho, S. B., & Shimohara, K. (1998). Evolutionary Learning of Modular Neural Networks with Genetic Programming. *Applied Intelligence*, *9*, 191–200. doi:10.1023/A:1008388118869

Cole, G. (2009). *Backprop1*. Retrieved February 1, 2010, from http://sourceforge.net/projects/backprop1

Dietterich, T. (2000). Ensemble methods in machine learning. In J. Kittler & F. Roli (Eds.), *Multiple Classier Systems,* Cagliari, Italy (LNCS 5519, pp. 1-15).

Fieldsend, J. E., & Singh, S. (2005). Pareto Evolutionary Neural Networks. *IEEE Transactions on Neural Networks*, *16*(2), 338–354. doi:10.1109/TNN.2004.841794

Freund, Y. (1995). Boosting a weak learning algorithm by majority. *Information and Computation*, *121*, 256–285. doi:10.1006/inco.1995.1136

Freund, Y., & Schapire, R. E. (1996). Experiments with a New Boosting Algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning,* Bari, Italy (pp. 148-156). San Francisco: Morgan Kaufmann.

Fu, H. C., Lee, Y. P., Chiang, C. C., & Pao, H. T. (2001). Divide-and-Conquer Learning and Modular Perceptron Networks. *IEEE Transactions on Neural Networks*, *12*(2), 250–263. doi:10.1109/72.914522

Gruau, F. (1995). Automatic definition of modular neural networks. *Adaptive Behavior*, *3*(2), 151–183. doi:10.1177/105971239400300202

Hansen, L. K., & Salamon, P. (2000). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(10), 993–1001. doi:10.1109/34.58871

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, *3*, 79–87. doi:10.1162/neco.1991.3.1.79

Janghel, R. R., Shukla, A., & Tiwari, R. (2010). Decision Support system for fetal delivery using Soft Computing Techniques. In *Proceedings of the Fourth International Conference on Computer Sciences and Convergence Information Technology,* Seoul, Korea (pp. 1514-1519). Washington, DC: IEEE.

Janghel, R. R., Shukla, A., Tiwari, R., & Tiwari, P. (2009). Clinical Decision support system for fetal Delivery using Artificial Neural Network. In *Proceedings of the 2009 International Conference on New Trends in Information and Service Science,* Gyeongju, Korea (pp. 1070-1075). Washington, DC: IEEE.

Jenkins, R., & Yuhas, B. (1993). A simplified neural network solution through problem decomposition: The case of the truck backer-upper. *IEEE Transactions on Neural Networks*, *4*(4), 718–722. doi:10.1109/72.238326

Jung, J. Y., & Reggia, J. A. (2006). Evolutionary Design of Neural Network Architectures Using a Descriptive Encoding Language. *IEEE Transactions on Evolutionary Computation*, *10*(6), 676–688. doi:10.1109/TEVC.2006.872346

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, IACC '09,* Patiala, India (pp. 541-545). Washington, DC: IEEE.

Kala, R., Shukla, A., & Tiwari, R. (2010a). Clustering Based Hierarchical Genetic Algorithm for Complex Fitness Landscapes. *International Journal of Intelligent Systems Technologies and Applications*, *9*(2), 185–205. doi:10.1504/IJISTA.2010.034320

Kala, R., Shukla, A., & Tiwari, R. (2010b). Handling Large Medical Data Sets for Disease Detection. In Shukla, A., & Tiwari, R. (Eds.), *Biomedical Engineering and Information Systems: Technologies, Tools and Applications*. Hershey, PA: IGI Global.

Kala, R., Vazirani, H., Shukla, A., & Tiwari, R. (2010). Fusion of Speech and Face by Enhanced Modular Neural Network. In *Proceedings of the Springer International Conference on Information Systems, Technology and Management, ICISTM 2010,* Bankok, Thailand (pp. 363-372). Washington, DC: IEEE.

Konar, A. (1999). *Artificial Intelligence and Soft Computing: Behavioral and Cognitive Modeling of the Human Brain*. Boca Raton, FL: CRC Press. doi:10.1201/9781420049138

Melin, P., & Castilo, O. (2005). *Hybrid Intelligent Systems for Pattern Recognition Using Soft Computing*. Berlin: Springer.

Mitchell, M. (1999). *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press.

Moriarty, D. E. (1997). *Symbiotic Evolution of NeuralNetworks in SequentialDecision Tasks*. Unpublished doctoral dissertation, Department of Computer Science, University of Texas, Austin, TX.

Moriarty, D. E., & Miikkulainen, R. (1997). Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, *5*(4), 373–399. doi:10.1162/evco.1997.5.4.373

Nolfi, S., Elman, J. L., & Parisi, D. (1990). *Learning and Evolution in Neural Networks* (CRL Tech. Rep. 9019). La Jolla, CA: University of California at San Diego.

Pedrajas, N. G. (2003). COVNET: A Cooperative Coevolutionary Model for Evolving Artificial Neural Networks. *IEEE Transactions on Neural Networks*, *14*(3), 575–596. doi:10.1109/TNN.2003.810618

Pedrajas, N. G. (2009). Supervised projection approach for boosting classifiers. *Pattern Recognition*, *42*, 1742–1760. doi:10.1016/j.patcog.2008.12.023

Pedrajas, N. G., & Fyne, C. (2008). Construction of classifier ensembles by means of artificial immune systems. *Journal of Heuristics*, *14*, 285–310. doi:10.1007/s10732-007-9036-0

Potter, M. A. (1997). *The design and analysis of a computational model of cooperative coevolution*. Unpublished doctoral dissertation, George Mason University, Fairfax, VA.

Rivera, A. J., Rojas, I., Ortega, J., & del Jesus, M. J. (2007). Anew hybrid methodology for cooperative-coevolutionary optimization of radial basis function networks. *Soft Computing*, *11*, 655–668. doi:10.1007/s00500-006-0128-9

Rosin, C., & Belew, R. (1996). New Methods for Competitive Coevolution. *Evolutionary Computation*, *5*, 1–29. doi:10.1162/evco.1997.5.1.1

Shukla, A., & Tiwari, R. (Eds.). (2010a). *Intelligent Medical technologies and Biomedical Engineering: Tools and Applications*. Hershey, PA: IGI Global Publishers.

Shukla, A., & Tiwari, R. (Eds.). (2010b). *Biomedical Engineering and Information Systems: Technologies, Tools and Applications*. Hershey, PA: IGI Global Publishers.

Shukla, A., Tiwari, R., & Kala, R. (2010a). *Real Life Applications of Soft Computing*. Boca Raton, FL: CRC Press. doi:10.1201/EBK1439822876

Shukla, A., Tiwari, R., & Kala, R. (2010b). *Towards Hybrid and Adaptive Computing: A Perspective*. Berlin: Springer.

Shukla, A., Tiwari, R., & Kaur, P. (2009). Intelligent System for the Diagnosis of Epilepsy. In *Proceedings of the IEEE World Congress on Computer Science and Information Engineering,* Los Angeles, CA (pp. 755-758). Washington, DC: IEEE.

Shukla, A., Tiwari, R., Kaur, P., & Janghel, R. R. (2009). Diagnosis of Thyroid Disorders using Artificial Neural Networks. In *Proceedings of the IEEE International Advanced Computing Conference,* Patiala, India (pp. 1016-1020). Washington, DC: IEEE.

Stanley, K. O., & Miikkulainen, R. (2004). Competitive Coevolution through Evolutionary Complexification. *Journal of Artificial Intelligence Research*, *21*, 63–100.

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

Yao, X. (1993). A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, *8*(4), 539–567. doi:10.1002/int.4550080406

Yao, X. (1997). A New Evolutionary System for Evolving Artificial Neural Networks. *IEEE Transactions on Neural Networks*, *8*(3), 694–713. doi:10.1109/72.572107

Yao, X. (1999). Evolving Artificial Neural Networks. *Proceedings of the IEEE*, *87*(9), 1423–1447. doi:10.1109/5.784219

This work was previously published in the Journal of Information Technology Research, Volume 4, Issue 1, edited by Mehdi Khosrow-Pour, pp. 34-47, copyright 2011 by IGI Publishing (an imprint of IGI Global).