Modular Symbiotic Adaptive Neuro Evolution for High Dimensionality Classificatory Problems

Rahul Kala*, Anupam Shukla, Ritu Tiwari
Soft Computing and Expert System Laboratory,
Indian Institute of Information Technology and Management Gwalior

* Corresponding Author
rahulkalaiiitm@yahoo.co.in
Present Address
School of Cybernetics, School of Systems Engineering,
University of Reading, Whiteknights,
Reading, Berkshire, United Kingdom – RG6 6AY
Ph: +44-7424752843

Citation: R. Kala, A. Shukla, R. Tiwari, Modular Symbiotic Adaptive Neural Evolution for High Dimensional Classificatory Problems, *Intelligent Decision Technologies*, 5(4) 309-319.

Final Version Available At:
http://iospress.metapress.com/content/0127t04853847051/?issue=4&genre=article&spage=309&issn=1872-4981&volume=5

Abstract

There has been a considerable effort in the design of evolutionary systems for the automatic generation of neural networks. Symbiotic Adaptive Neuro Evolution (SANE) is a novel approach that carries co-evolution of neural networks at two levels of neuron and network. The SANE network is likely to face problems when the applied data set has high number of attributes or a high dimensionality. In this paper we build a modular neural network with probabilistic sum integration technique to solve this curse of dimensionality. Each module is a SANE network. The division of the problem involves the breaking up of the problem into sub-problems with different (may be overlapping) attributes. The algorithm was simulated for the Breast Cancer database from UCI machine learning repository. Simulation results show that the algorithm, keeping the dimensionality low, was able to effectively solve the problem.

Keywords: Evolutionary Neural Network, Cooperative Evolution, Modular Neural Network, Curse of Dimensionality, Medical Diagnosis, Breast Cancer.

1. Introduction

Decision making is a major issue that is currently under research by multiple research domains. Inability to model the problem effectively and the associated uncertainties, lead to the use of machine learning techniques for the same. Here we aim in extracting hidden data or knowledge from a historical database containing inputs and outputs. With the increase of automation more complicated problems are being attempted to be solved, which require sophisticated methods. We attempt to make a robust decision making system that can solve problem having high number of inputs and representing high complexity, and give good results.

Neural networks are good tools for intelligent system design. Learning involves the extraction of knowledge from the input training data which is represented in the form of internal parameters of the neural network. One of the commonly used models of neural network is the Multi-Layer Perceptron which contains many artificial neurons arranged in a layered manner. Each of the neuron carries some task of processing of the inputs to facilitate the mapping of inputs to the outputs. The task of machine learning is to tune the network weights and biases such that an optimal mapping takes place. Back Propagation Algorithm (BPA) is one of the commonly used algorithms for machine learning. BPA carries out supervised learning of the training database (Konar, 1999). The algorithm is however prone to get struck at some local minima. This calls for better machine learning techniques. The other problem associated with the neural networks is its architecture. The

architecture plays a key role in deciding the performance of the neural network. The architecture is specified by the human designer, who tries numerous architectures, before finalizing the optimal architecture. Due to the human limitations, the results are likely to be sub-optimal.

Evolutionary Algorithms are strong optimizing agents that try to optimize given systems for better performance. The evolutionary algorithms are widely used for system design and evolution (Shukla, Tiwari, & Kala, 2010). These algorithms optimize the system in an iterative manner, where more emphasis is given to the fitter solutions in the entire evolutionary process. Genetic Algorithm comes under the class of Evolutionary Algorithms. This algorithm uses the analogy of the natural evolutionary process. The algorithm maintains a pool of individuals or population that keeps improving with generations. The creation of a higher generation individuals is carried from the lower generation individuals by using genetic operators like selection, mutation, crossover, elite, etc (Mitchell, 1999).

The strong evolutionary powers of these algorithms open doors to an exciting field of evolutionary neural networks, where the problem solving capability of the neural networks is combined with the evolutionary power of the evolutionary algorithms for a more sophisticated hybrid system. The evolutionary neural networks may be fixed architecture or variable architecture. The fixed architecture neural networks use genetic algorithms only for tuning the system weights and biases. The variable architecture evolutionary neural networks, on the contrary, optimize both the network architecture and parameters by the evolutionary algorithms (Nolfi, Parisi, & Elman, 1990; Yao, 1999). This problem is much more challenging and time consuming. 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).

Cooperative evolution or co-evolution is a novel concept that gives good results to most complex problems. This form of evolution takes its analogy from the social living nature of humans where we not only try to develop ourselves, but also the other members of the society. This results in a better overall development. In this form of evolution the individuals represent part solutions, rather than complete solutions. This representation reduces the problem dimensionality by the generation of smaller individuals. The fitness evaluation of the individuals takes place by analysis of the individual performance in multiple complete solutions. The individual is assigned a high fitness count, even if it aids other individuals in attaining high fitness count. The aim is to use evolutionary pressures to develop distinct and high performance components of the solutions or individuals. This is perceived as a better mechanism than the conventional mechanisms of diversity preservation in evolutionary computation (Potter, 1997; Rosen and Belew, 1996; Stanley & Miikkulainen, 2004).

Dimensionality is a major problem associated with evolutionary algorithms. The algorithm performance reduces drastically by the addition of parameters. Each parameter of the evolutionary algorithm represents a dimension into its search space. It is evident that the additional dimensions have a large effect on algorithm performance, convergence, and computational requirements. The problem is even more when we deal with problem of neural network evolution. Any addition of input attribute may result in the need of additional hidden layer neurons. This increases the total number of

connections in the neural network by large amounts, which are all dimensions in the evolutionary algorithm search space. This creates a large problem in the evolution, making the evolutionary process very time consuming. It is hence important to carefully select the attributes and their numbers, before giving it to an evolutionary neural network (Kala, Shukla, & Tiwari, 2010).

The higher number of dimensions is further problematic from a neural perspective as well. The additional dimensions or input attributes in the problem result in the creation of a large input space which is extremely difficult to model for the neural network. The control of the neural network architecture is an extremely difficult task as the number of attributes increase. This further requires more training instances for training and a very large training time. Hence the attributes must always be limited in a neural network (Kala, Shukla, & Tiwari 2009).

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 and 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).

Biomedical Engineering is a very exciting field where computational intelligence has found application. A lot of work is done for automatic diagnosis of the diseases. The automatic diagnostic system can be an excellent tool for aiding the doctors in making effective decisions very early. These tools take the data for any patient as an input and produce the diagnosis result as the output. The mapping of the input to the output, or the diagnosis is generally based on the learning of the system from the training database. The artificially designed expert systems behave just like real doctors to carry out the diagnosis task (Bronzino, 2006; Shukla & Tiwari, 2010a, 2010b). Breast Cancer is specifically an emerging problem for which numerous diagnosis techniques are being engineered. The growing rise of this disease is a major point of concern for engineers (Breastcancer.org, 2010).

In this paper we propose a method for the automatic diagnosis of breast cancer by using Modular Symbiotic Adaptive Neuro Evolution (SANE). The approach involves adaptation of the Symbiotic Adaptive Neuro Evolution (Moriarty, 1997; Moriarty and Miikkulainen, 1997) algorithm for the classificatory problems. The resulting system is prone to have reasonably large execution time due to very large problem complexity. Further the convergence is likely to be poor. We hence then build a modular neural network framework over SANE. The resulting system is capable of handing very complex problems with large number of attributes. This makes the resultant algorithm robust, at the same time displaying large diagnostic accuracy.

This paper is organized as follows. In Section 2 we present some of the related works. Section 3 presents the classificatory model of the SANE algorithm. The next task is the modularization of the formulated SANE network by division of attributes. This is

discussed in Section 4. The simulation results are given in Section 5. Section 6 gives the conclusion remarks.

2. 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). 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 against a set of error functions against a pareto front. The use of multiple errors functions enabled strong check against generalization loss or over-fitting. This neural network was further extended to use a validation data set to avoid over-fitting, 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 and 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 concept 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..

3. Classificatory Symbiotic Adaptive Neural Evolution

Symbiotic Adaptive Neuro Evolution (SANE) uses co-evolution as a means to evolve the optimal architecture of the neural network. Complete details regarding SANE may be found at (Moriarty, 1997; Moriarty and Miikkulainen, 1997). In this approach two evolutionary approaches are used, at the neuron level and network level. It is assumed that the neural network to be evolved is a multi-layer perceptron where the different neurons are arranged in a layered manner. The input and the output layers are fixed whose number of neurons is provided by the user. It is assumed that the neural network may have only a single hidden layer. The number of neurons in this layer is however evolved by the SANE algorithm. Any neuron of the hidden layer may be connected to any number of hidden or output layer neurons. Hence it is not assumed that the architecture is fully connected.

The first genetic algorithm is used for the neurons. Here each genetic individual is a hidden layer neuron. This neuron has certain connections with the input and output layer neurons. Information regarding all these connections and the corresponding weights, along with the bias is stored in these neurons. These neurons try to attain optimal values to carry out effective processing. Since these neurons store reasonably small information, these are easy to be optimized by the evolutionary process, which is the basic motive behind the co-evolution. The individuals are used to denote part solutions rather than the individual solutions. These part solutions co-operate with each other to make the complete solution.

The other major task associated with the problem is the manner in which these part solutions integrate to make the complete solution. We may not select the best performing part solutions as the best complete solutions, as they might not collectively give a high performance. In our case the part solutions are the neurons that are collectively supposed to make the complete neural network. The selection of neurons that make the neural network is an important decision. Multiple combinations of the various neurons may be possible. This problem is solved by another genetic algorithm. This algorithm does the task of optimal selection of the neurons of the first genetic approach. Hence an individual in this level stores the selected neurons hat make the network.

These two evolutionary approaches run simultaneously in a co-operative manner for the evolution of the optimal neural network. The neuron genetic algorithm tries to generate good neurons that can carry effective computations for getting the desired output. Good neurons are source of good neural networks for the network construction. The task of

assembling these neurons and their selection is done by the other genetic algorithm. Since the system possesses good neurons as per the problem requirements, it is natural that the complete network may have good performance. The optimal neuron selection and arrangement further maximizes the performance. In this manner a very complicated problem is solved optimally.

Fitness evaluation is different for both these levels of evolution. The fitness evaluation for the network is its performance for the designated data. The fitness evaluation of the neuron is however its fitness in the best 5 networks it participates in. This judges the capability of the neuron is contributing good features to the network. If the contributed features are good, these would be combined with the other neurons resulting in a high network performance. The average over 5 networks ensures that the specific neuron capability is judged, as specific neural networks may be dominated by other neurons, and fitness may not necessarily be from the neuron being assessed.

The evolutionary operators carry the task of construction of a higher generation population from a lower generation population. The evolution is as well different for both the evolutionary approaches. At the neuron level, the evolution is done by crossover and mutation operators. The crossover operator randomly selects the parents from the top 25% of the population. This generates two children. The first child is the result of one-point crossover. The other child is one of the participating parents. This is used for convergence control. These two children replace the worst fit individuals from the population. The mutation is also applied at constant rate. Half the population individuals are replaced at every generation. The operations are similar for evolution at the network genetic algorithm as well. The network contains a set of neurons. The crossover carries

the exchange of these neurons between the parents to generate the children. The mutation operation carries the change of neurons in the network. Another mutation is used to assign newly created neurons at any generation to the networks. The general architecture of the SANE algorithm is given in figure 1.

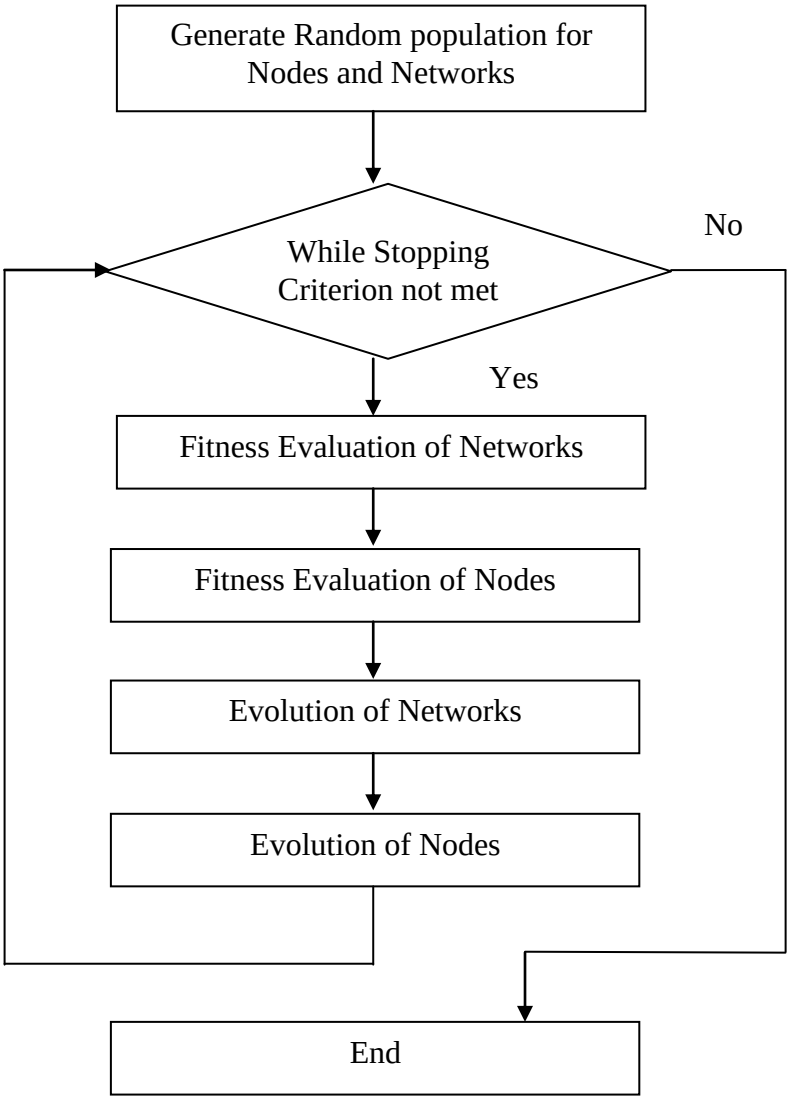

Figure 1: The general architecture of SANE algorithm

SANE architecture evolves the neural network as per the problem requirements. The problem of our consideration is classificatory in nature. We must hence devise a mechanism to handle classificatory outputs in SANE. Classification deals with the

determination of the class to which the applied input maps to, out of all the various available classes. Let the system have *n* classes. In this model we take *n* outputs of the neural network, with each output denoting the probability of the occurrence of the class. This lies between 0 and 1. The output vector may hence be taken as $<o_1, o_2, o_3, ..., o_n>$ Here o_i denotes the probability of the input belonging to class *i*. While testing the class that has the largest probability is returned as output. This may be given by (1).

$$\text{Output} = \{i: o_i < o_j \vee 0 \leq i, j \leq n, i \neq j\} \quad (1)$$

In this model we may easily see that every output tries to identify the region in the input space where the associated class is present. It returns output close to 1 for the areas in the input space where class is likely to be present, and returns outputs close to 0 for all other areas in the input space.

The error measurement of this network is a measure of the classification accuracy of the classifier. An error is recorded if the network incorrectly classifies an input. Let the training database contain *N* instances. Let the output of any instance *i* be O_i. Let the target be T_i. The error is measured by (2).

$$Error = \frac{\sum_{i=1}^{N} O_i \neq T_i}{N} \quad (2)$$

4. Modularity in Attributes

Any machine learning or classification task is highly dependent on the number of attributes that make inputs to the intelligent system. Attributes especially play a vital role in deciding the algorithm performance. Very less attributes may not give a good

recognition score as it may be very difficult to separate the classes by the decision boundaries. This is expecting the attributes were not idealistic in nature having high inter-class separation and low intra-class separation. However the system may not behave well either if the number of attributes is very large. In such a case there are a large amount of decision boundaries possible, and figuring out the optimal one is a time consuming task. There might further be the requirement of a large amount of training database to tune the various weights and parameters in the network. This greatly reduces the execution time of the algorithm. In case the algorithm is not executed for the entire length, the algorithm may be trained sub-optimally (Kala, Shukla, & Tiwari, 2010).

In order to control the problem of dimensionality in this network, we use the concept of modularity into the classificatory SANE algorithm. The attributes of the algorithm are distributed into a number of modules (Kala *et al*. 2010; Kala, Shukla, & Tiwari 2010b). Each module gets a part of the total number of attributes available with the system. The division of attributes amongst the modules is done in such a manner that the various attributes occur in some module or the other. An attribute may simultaneously be given to multiple modules of the network. The attribute division is done so that the various modules individually give high performance with the least number of attributes. It is expected the errors of one module may be eliminated by the other modules when they collectively work in a ‘mixture of experts’ architecture.

Integration of the results is an essential part of the entire modular network architecture. The various modules working with different attributes act like experts in judging the inputs. The results of each of the experts need to be integrated by the integrator to make the final decision regarding the output class of the system. The integration in this

algorithm is performed by a probabilistic sum technique. Each module *j* returns as output a set of probabilities corresponding to the occurrence of each of the classes in the system. Let these probabilities be $O^j <o^j_1, o^j_2, o^j_3, ... o^j_i, ... o^j_n>$. Here o^j_i denotes the probability of occurrence of class *i* as per expert *j*. The probabilistic addition technique takes the weighted addition of all these probabilities. This forms the final probability vector $P<P_1, P_2, P_3, ... P_i, ...P_n>$. Here P_i denotes the probability of occurrence of class *i*. This may be given by (3).

$$P_i = \sum_j w_j o_i^j \quad (3)$$

Here w_j denotes the weight given to expert *j*. The weights must obey equation (4).

$$\sum_j w_j = 1 \quad (4)$$

The resultant probability vector is then checked for the computation of the final class. The class corresponding to the maximum probability of occurrence is stated as the class to which the output classifies to, as given in equation (5).

$$\text{Output} = \{k: p_k < p_j,\ i \neq j\} \quad (5)$$

Each of the modules of the problem is a classificatory SANE neural network. It takes the stated input attributes and produces as output the probability vector denoting the possibility of occurrences of the various classes. The training and the testing data sets are hence divided into the various attributes. This forms the inputs of the training and testing

data sets of the various modules which are processed as per the SANE evolutionary methodology. The complete architecture of the system is given in figure 2.

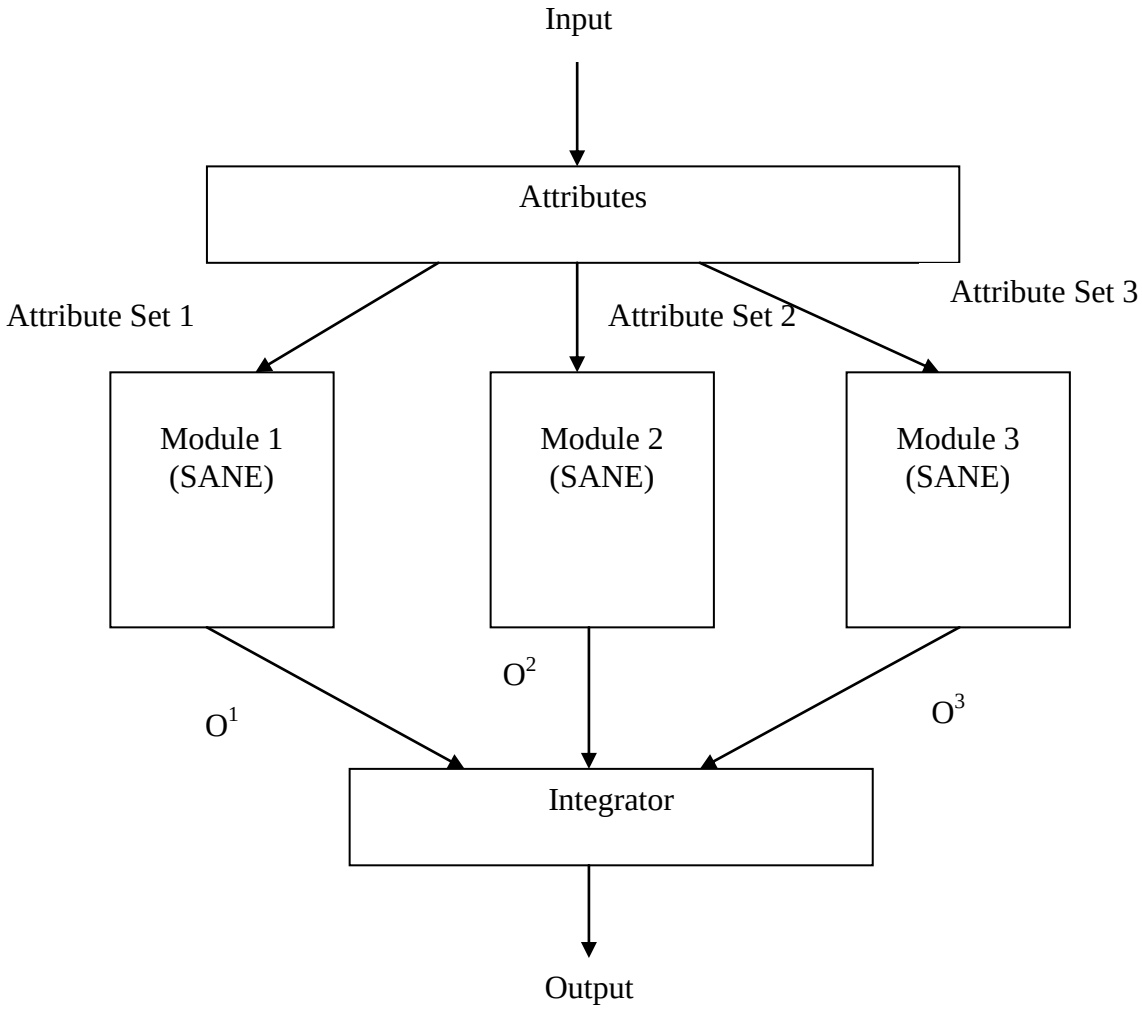

Figure 2: The Modular SANE architecture

5. 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 and 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 (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). Here three cells are considered for analysis which makes sure that all information is captured that may affect the diagnostic decision. Choosing lesser number of cells clearly implies loss of some valuable information. The database chosen is recorded from real world data and hence it carries all the characteristics of the actual problem. Hence the developed system may be directly applied for diagnosis, subjected to the condition that same attributes are considered for diagnosis. The entire data set consists of a total of 357 benign and 212 malignant cases, totalling to 569 instances in the database.

The first task to be carried out is the division of attributes. The 30 attributes need to be distributed into modules. We make a total of 3 modules. The various attributes are distributed amongst these three modules. The distribution, by design of the algorithm, may be done in a manner that each attribute is given to some or the other neuron. Further an attribute may be given to multiple modules. Hence we distribute the attributes such that each attribute is given randomly to two modules. The entire database 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 set gets designated to training or testing datasets.

The next task is the use of classificatory SANE algorithm for evolution. For this we use the SANE code available at (Texas, 2010). This forms the base code over which the other

modules are built as per the requirements stated in section 3 and section 4. One module of the program did the task of collection of the data from a text file, its normalization, and random division into training and testing data sets. The attribute division was done randomly within the program. The training and testing data sets are passed into the JAVA SANE program for each of the modules. The algorithm evolved the neural network using the co-evolutionary principles. The various parameters used for the execution of the algorithm include maximum number of connections per neuron as 24, evolutionary population size of 1000, maximum neurons in hidden layer as 12, SANE elite value of 200, mutation rate of 0.2, and number of generations as 100. These parameters were same for all the three modules that were executed one after the other. The output of each of the module was taken. These values were then used by a separate module that carried out the task of integration to give the final results.

On training and testing the system as per this methodology, the network achieved a good accuracy for both training and testing dataset. The values of the weights of the three modules were fixed so as to maximize the performance in training dataset. The accuracies were taken as a mean of 20 executions with the same parameter set. The net performance of the system was 96.90% using training data set and 96.59% using testing dataset. The algorithm took approximately 2 minutes and 30 seconds for the evolution. It may be easily seen that the time required is significantly less than the conventional approaches over data sets of same size and complexity which may sometimes even take hours to be trained. The reduction in time is first due to use of co-evolutionary technique that simplifies problem by large degree, and second due to further simplification by use of modularity. In the absence of these factors training time may be in order of an hour. The results of the algorithm for an average of 20 runs are summarized in table 1.

Convergence is an important factor in the evolutionary approaches that gives an idea of the manner in which the individuals of the algorithm behave in regard to the fitness along with generations. We plot the fitness of the best network for all the three modules along with time. Here fitness is taken to be the accuracy over the training data. The resultant convergence is shown 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.

Table 1: Analysis of the Results of the algorithm

S. No.	Property	Value
1.	Mean Training Accuracy	96.90%
2.	Mean Testing Accuracy	96.60%
3.	Standard Deviation in Training Accuracy	0.933
4.	Standard Deviation in Testing Accuracy	1. 65
5.	Approx. Mean Training Time	2 mins 30 secs
6.	Mean Correctly Identified Instances (Testing)	165
7.	Mean Incorrectly Identified Instances (Testing)	6

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.

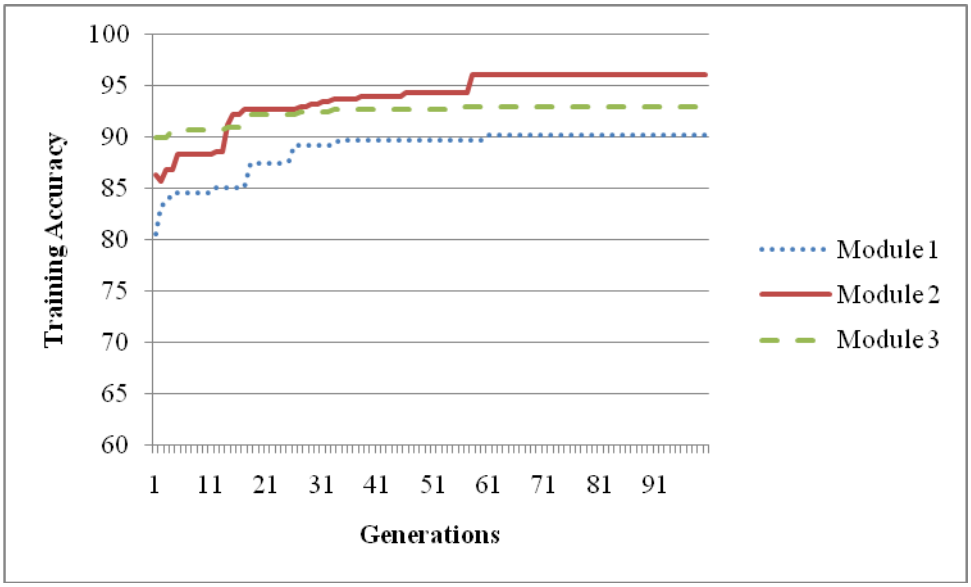

Figure 3: Performance of best networks for all three modules against generations

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 propagation 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 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 an 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 last method we adopt for the same problem is the convenital SANE algorithm. The modifications were made to this algorithm to enable it act for the classificatory problems. The parameters of this algorithm were similar to the parameters used in the execution of the proposed algorithm. The algorithm on execution gave a training accuracy of 94.73% and a testing accuracy of 93.52%.

The various algorithm results have been summarized in table 2.

Table 2: Comparisons between various algorithms

S. No.	**Algorithm**	**Training Accuracy**	**Testing Accuracy**
1.	**Modular Classificatory SANE**	**96.90%**	**96.60%**
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%
8.	Classificatory SANE	96.40%	95.39%

From the above table, it is clear that the proposed algorithm gave the best generalization and testing accuracy as compared to all the methods presented. A higher training accuracy was displayed by a variety of methods, but the ultimate aim of the network is a high performance over the testing data. The larger networks may produce a better training accuracy, but not a better testing accuracy. The higher generalization capability of the proposed algorithm is a warrant that it would carry effective diagnosis, whenever applied to the real life scenarios. The high recognition score illustrates an effective diagnostic system.

6. Conclusions

The present work dealt with devising a solution to the problem of curse of dimensionality. Evolutionary neural networks have been known to be strong agents of machine learning with good capability of generalizing the learning from the learnt data to new data or the testing data. We took SANE algorithm as the development platform for this purpose. The problem of dimensionality was solved by building a modular neural network over SANE. The modular neural network carried out attribute division. Each module, with its designated attributes, computed the output. The outputs were integrated using an integrator, which used a probabilistic sum for output computation. The class corresponding to the maximum likeness of occurrence was designated as the output class to the input applied.

The approach was applied over the problem of breast cancer diagnosis. The entire attribute set, comprising of 30 attributes, was divided into 3 modules. Each attribute was randomly given to two of the three modules. All the three modules were trained using the

training database. Each of the modules was SANE which used co-evolution as the basic evolutionary methodology. In the testing mode the outputs of the various modules was combined using the integrator. This gave the final output.

The algorithm over this database achieved an accuracy of 96.90% for the training data and 96.60% for the testing data. This accuracy was compared to a variety of methods commonly used in literature. In all these we found that the proposed algorithm gave the highest accuracy for the testing database. This shows a very high learning capability, as well as a very high generalizing capability of these networks. The high generalizing capability shows that the proposed algorithm can be effectively used for the Breast Cancer diagnosis.

Present study is limited to only a single database of Breast Cancer. Future attempts may be made to carry experimentation with this algorithm on multiple databases from multiple domains. The rising automation has brought many new domains into the outreach of neural networks. The crux of this algorithm is its ability to handle a large number of attributes. Extensive research over different problems with large dimensions may be useful to fully experiment the algorithm capability of handling dimensions. Further research may build optimal attribute division strategy, as well as strategy to decide the number and roles of modules. The problems are getting more complex and difficult to solve. The increasing complexity largely results in an increase in the total number of attributes. This further motivates the need of making systems that can take a large number of attributes. While the present approach is effective in introduction of modularity at the attribute level, further sincere attempts are needed to make the systems scalable to an even higher level.

References

[1] Breastcencer.org. *Understanding the Breast Cancer*. available at http://www.breastcancer.org Accessed Feb 2010

[2] J. D. Bronzino, *Biomedical Engineering Fundamentals*, CRC Press, 2006

[3] S. B. Cho, K. Shimohara. Evolutionary Learning of Modular Neural Networks with Genetic Programming. *Applied Intelligence* 9: 191–200, 1998

[4] T. Dietterich. Ensemble methods in machine learning. In: J. Kittler, F. Roli (eds.) Multiple Classier Systems. *Lecture Notes in Computer Science*, pp. 1-15, Cagliari, Italy, 2000

[5] J. E. Fieldsend, S. Singh. Pareto Evolutionary Neural Networks. *IEEE Transactions on Neural Networks*, 16(2): 338:354, 2005

[6] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation* 121: 256–285, 1995

[7] Y. Freund, R. E. Schapire. Experiments with a New Boosting Algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning*, pp 148-156, 1996

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

[9] F. Gruau. Automatic definition of modular neural networks. Adaptive Behavior. 3(2): 151–183, 1995.

[10] L. K. Hansen , P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis Machine Intelligence*. 12(10): 993-1001, 2000

[11] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation* 3:79-87, 1991.

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

[13] J. Y. Jung, J. A. Reggia. Evolutionary Design of Neural Network Architectures Using a Descriptive Encoding Language. *IEEE Transaction on Evolutionary Computation* 10(6): 676-688, 2006

[14] R. Kala, A. Shukla, R. Tiwari. Clustering Based Hierarchical Genetic Algorithm for Complex Fitness Landscapes. *International Journal of Intelligent Systems Technologies and Applications*, 2010a, accepted forthcoming

[15] R. Kala, A. Shukla, R. Tiwari. Handling Large Medical Data Sets for Disease Detection, Shukla, Anupam & Tiwari, Ritu (eds.) *Biomedical Engineering and Information Systems: Technologies, Tools and Applications*, IGI Global, 2010b

[16] R. Kala, A. Shukla, R. Tiwari. Fuzzy Neuro Systems for Machine Learning for Large Data Sets, *Proceedings of the IEEE International Advance Computing Conference, IACC '09*, pp 541-545, Patiala, India, 2009

[17] R. Kala, H. Vazirani, A. Shukla, R. Tiwari. Fusion of Speech and Face by Enhanced Modular Neural Network. *Proceedings of the Springer International Conference on Information Systems, Technology and Management, ICISTM 2010*, pp. 363-372, Bankok, Thailand, 2010

[18] A. Konar. *Artificial Intelligence and Soft Computing: Behavioral and Cognitive Modeling of the Human Brain*. CRC Press,1999

[19] P. Melin, O. Castilo. *Hybrid Intelligent Systems for Pattern Recognition Using Soft Computing*, Springer-Verlag Berlin Heidelberg, 2005

[20] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1999

[21] D. E. Moriarty. *Symbiotic Evolution of NeuralNetworks in SequentialDecision Tasks*. Ph.D. thesis, Department of Computer Science, University of Texas, Austin, Texas, 1997.

[22] D. E. Moriarty, R. Miikkulainen. Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, 5(4):373–399, 1997.

[23] S. Nolfi, J. L. Elman, D. Parisi. *Learning and Evolution in Neural Networks*. CRL Technical Report 9019, LaJolla, CA, USA, University of California at San Diego, 1990

[24] M. A. Potter. *The design and analysis of a computational model of cooperative coevolution*. PhD thesis, George Mason University, 1997

[25] N. G. Pedrajas. COVNET: A Cooperative Coevolutionary Model for Evolving Artificial Neural Networks. *IEEE Transactions on Neural Networks*, 14(3): 575:596, 2003

[26] N. G. Pedrajas. Supervised projection approach for boosting classifiers. *Pattern Recognition* 42: 1742-1760, 2009

[27] A. J. Rivera, I. Rojas, J. Ortega, M. J. del Jesus. Anew hybrid methodology for cooperative-coevolutionary optimization of radial basis function networks. *Soft Computing*, 11: 655–668, 2007

[28] C. Rosin, R. Belew. New Methods for Competitive Coevolution. *Evolutionary Computation*, 5: 1-29, 1996.

[29] A. Shukla, R. Tiwari. *Intelligent Medical technologies and Biomedical Engineering: Tools and Applications*. IGI Global Publishers, 2010a

[30] A. Shukla, R. Tiwari. *Biomedical Engineering and Information Systems: Technologies, Tools and Applications*. IGI Global Publishers, 2010b

[31] A. Shukla, R. Tiwari, R. Kala. *Real Life Applications of Soft Computing*. CRC Press, 2010

[32] K. O. Stanley, R. Miikkulainen. Competitive Coevolution through Evolutionary Complexification, *Journal of Artificial Intelligent Research*, 21: 63-100, 2004.

[33] Texas, University of, Neural Network Group, http://nn.cs.utexas.edu/, Accessed January, 2010

[34] W. H. Wolberg, O. L. Mangasarian, D. W. Aha. *UCI Machine Learning Repository* [http://www.ics.uci.edu/~mlearn/MLRepository.html], University of Wisconsin Hospitals, 1992.

[35] X. Yao. A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, 8(4): 539-567, 1993

[36] X. Yao. Evolving Artificial Neural Networks. *Proceedings of the IEEE*. 87(9): 1423-1447, 1999

[37] X. Yao. A New Evolutionary System for Evolving Artificial Neural Networks. *IEEE Transactions on Neural Networks*, 8(3): 694-713, 1997.