Design of Simple ANN (SANN) model for Data Classification and its performance Comparison with FLANN (Functional Link ANN)

Gunanidhi Pradhan, Bhubanananda Orissa School of Engineering, Cuttack

Vishal Korimilli, Student Member IEEE, Final Yr CSE,ANITS

Suresh Chandra Satapathy, MIEEE, Anil Neerukonda Institute of Technology & Sciences (ANITS), Vishakapatnam Dist

Dr. Sabyasachi Pattnaik,,FM University, Balasore

Dr Bhabatosh Mitra,FM University, Balasore

Abstract- Data Classification is a prime task in Data mining. Accurate and simple data classification task can help the clustering of large dataset appropriately. In this paper we have experimented and suggested a simple ANN based classification models called as Simple ANN (SANN) for different classification problems. The GA is used for optimally finding out the number of neurons in the single hidden layered model. Further, the model is trained with Back Propagation (BP) algorithm and GA (Genetic Algorithm) and classification accuracies are compared. The designed models are also compared with the Functional Link ANN (FLANN) for Data classification accuracies. It is revealed from the simulation that our suggested model is performing better compared to FLANN model and it can be a very good candidate for many real time domain applications as these are simple with good performances.

Key words: ANN, Genetic Algorithm, Data classification, FLANN

I. Introduction

Data classification is a classical problem extensively studied by statisticians and machine learning researchers. It is an important problem in variety of engineering and scientific disciplines such as biology, psychology, medicines, marketing, computer vision, and artificial intelligence [1]. The goal of the data classification is to classify objects into a number of categories or classes. Given a dataset, its classification may fall into two tasks

i) supervised classification in which given data object is identified as a member of predefined class

ii) Unsupervised classification (or also known as Clustering) in which the data object is assigned to an unknown class.

Manuscript received October 5, 2009
Manuscript revised October 20, 2009

Supervised classification (here onwards to be referred as classification) algorithms have been widely applied to speech, vision, robotics, diseases, and artificial intelligence applications etc where real time response with complex real world data is necessity. There have been wide ranges of machine learning and statistical methods for solving classification problems. Different parametric and non-parametric classification algorithms have been studied [2-9]. Some of the algorithms are well suited for linearly separable problems. Non-linear separable problems have been solved by neural networks [10], support vector machines [11] etc. However, in many cases it is desired to find a simple classifier with simple architecture. There has been wide spectrum of work on developing ANN based classification models consisting of many hidden layers and large number of neurons in the hidden layers. It is obviously understood from the literature of ANN that more number of hidden layers and large number of neurons may sometimes present a good solution for the problem but at the cost of computational cost. There hve been also some attempts made to use FLANN [13] for classification purpose. In this paper SANN (Simple Artificial Neural Network) model is suggested in which we have proposed a ANN model having *m-n-p* as the model parameters wherein *m* is the number of inputs (based on the dataset under investigation) , *n* is the number of neurons in hidden layer (only one hidden layer is used to minimize the computational complexity) and *p* is the number of output neurons (based on the dataset under investigation). The optimal numbers of neurons in the hidden layer are chosen by Genetic Algorithm (GA) [12]. The weights of the SANN are tuned by BP algorithm and GA for different datasets and results are compared. Our results are also compared with FLANN based models. It is revealed that the models suggested by us are less in complexity and better in performance.

The rest of the paper comprises of as follows. In section II, Back Propagation (BP) algorithm is briefly described, which is used to train the SANN. Section III describes the GA approach for optimally finding the values for the number of neurons in the hidden layer. FLANN basics are described in section IV. In section V the datasets are described. Section VI discusses the simulation and results. Conclusion and future research are given in Section VII.

II. Back Propagation Training of SANN

An MLP(Multi Layer Perceptron) network with 2-3-3 neurons (2, 3 and 3 denote the number of neurons in the input layer, the hidden layer and the output layer respectively) with the back-propagation (BP) learning algorithm, is depicted in Fig.1.

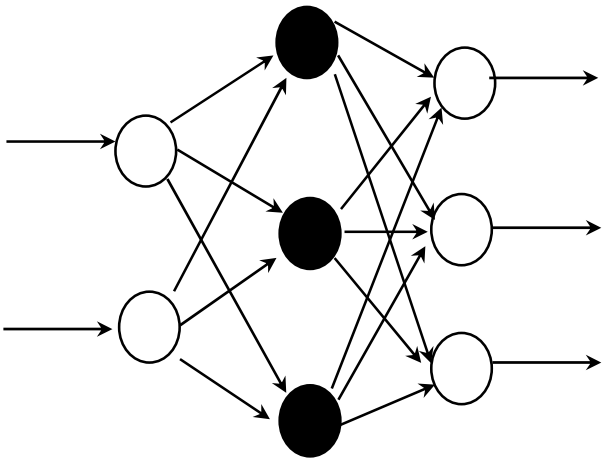


Fig1: MLP network

Initialize the weights

In BP algorithm, initially the weights are initialized to very small random numbers. Each unit has a bias associated with it. The biases are similarly initialized to small random numbers.

Propagate the inputs forward:

First, the training tuple is fed into the input layer of the network. The inputs pass through the input units, unchanged. Next, the net input and output of each unit in the hidden and output layers are computed. The net input to a unit in the hidden or output layers is computed as a linear combination of its inputs. To compute the net input to the unit, each input connected to the unit is multiplied by its corresponding weight, and is summed. Given a unit in a hidden layer or output layer, the net input, I j, to unit j is

$$I_j = \sum_i W_{ij} O_i + \theta_j$$

Where W_{ij} is the weight of the connection from the unit I in the previous layer to unit j

O_i output of unit I from the previous layer

θ_j is the bias of the unit

The bias acts as a threshold in that it serves to vary the activity of the unit.

Each unit in the hidden and output layers takes its net input and then applies an activation function. The function symbolizes the activation of neuron represented by the unit. The logistic or sigmoid function is used and the output of unit j, is computed as

$$O_j = \frac{1}{1 + e^{-I_j}}$$

This function is also referred to as squashing function, because it maps a large input domain onto the smaller range of 0 to 1.

Back propagate the error:

The error is back propagated backward by updating the weights and biases to reflect the networks prediction. For a unit j in the output layer, the error is computed by

$$Err_j = O_j(1 - O_j)(T_j - O_j)$$

Where O_j s the actual output of unit j, and

T_j is the known target value of the given training tuple.

To compute the error of a hidden layer unit j, the weighted sum of the errors of the units connected to unit j in the next layer is considered. The error of a hidden layer unit j is,

$$Err_j = O_j(1 - O_j)\sum_k Err_k w_{jk}$$

Where W_{jk} is the weight of the connection from unit j to a unit k in the next higher layer, and Err_k is the error of unit k.

The weights and biases are updated to reflect the propagated errors. Weights are updated by following equation,

$$w_{ij} = w_{ij} + (l) Err_j O_i$$

The variable l is the learning rate, a constant having a value betwen 0 to 1. Back propagation learns using a method of gradient descent to search for a set of weights that fits the training data so as to minimize the mean squared error. The learning rate helps avoid getting stuck at a local minimum and encourages finding global minimum.
Biases are updated by following equation,

$$\theta_j = \theta_j + (l) Err_j$$

Terminating condition:

Training stops when

- All weights in the previous epoch were so small as to below some specific threshold
- The percentage of tuples misclassified in the previous epoch is below some threshold
- A prespecified number of epochs has expired

A classifier which gives a higher accuracy value is considered as a good classifier.

Algorithm:

1. Initialize weights and biases in the network.

2. Propagates inputs forward in the usual way, i.e.
   - All outputs are computed using sigmoid threshold of the inner product of the corresponding weight and input vectors.
   - All outputs at stage *n* are connected to all the inputs at stage *n*+1

3. Propagates the errors backwards by apportioning them to each unit according to the amount of this error the unit is responsible for.

4. Terminating condition (Error is minimum or till the iterations are exhausted).

III. GA for optimally finding Neurons in hidden layer of SANN

Genetic algorithms (GA) are an evolutionary optimization approach which is an alternative to traditional optimization methods. GAs are most appropriate for complex non-linear models where location of the global optimum is a difficult task. It may be possible to use GA techniques to consider problems which may not be modeled as accurately using other approaches. Therefore, GA appears to be a potentially useful approach. GA follows the concept of solution evolution by stochastically developing generations of solution populations using a given fitness statistic (for example, the objective function in mathematical programs). They are particularly applicable to problems which are large, non-linear and possibly discrete in nature, features that traditionally add to the degree of complexity of solution. Due to the probabilistic development of the solution, GA does not guarantee optimality even when it may be reached. However, they are likely to be close to the global optimum. This probabilistic nature of the solution is also the reason they are not contained by local optima. The standard GA process consists of an initialization step and the iterative generations [12]. The SGA process is shown in the Fig. 2. First, a population of chromosomes is created. Second ,the chromosomes are evaluated by a problem defined fitness function. Third, some of the chromosomes are selected for performing genetic operations. Fourth, genetic operations of crossover and mutations are performed. The offspring produced out of genetic operations replace their parents in their initial population. This GA process repeats until a user defined criterion is met.

```
Procedure of the standard GA
begin
i=0    // i : number of iteration
initialize P(i)  // P(i): population for iteration i
evaluate f(P(i))   // f: fitness function
while ( not termination condition) do
        begin
          i=i+1
          select 2 parents p1and p2 from P(i-1)
 perform genetic operations (crossover and mutation)
        reproduce a new P(i)
        evaluate f(P(i))
       end
end
```

Fig2: GA procedure

In our work chromosomes are nothing but string of integers within a random range depicting the possible values for the number of neurons in the hidden layer. Depending on the dataset complexity i.e the dimension and number of data objects in the dataset this range has been chosen for simulation. The fitness value is nothing but the classification accuracy of the model. More the percentage of correct classifications better the model. The population size is chosen based on the dataset. However, a population size up to 30 is a good choice for our simulations. In our work as we are aiming for Simple ANN having only one hidden layer, the binary GA is implemented for the simulation. In Binary GA the chromosomes initialized by integer is converted to binary values for applying the GA operators. In entire of our work the one point cross over is adopted with 0.8 crossover probabilities and with 0.01 mutation probability.

IV. Bascis of FLANN

Pao [13]. was originally proposed the FLANN architecture. They have shown that, their proposed network may be conveniently used for function approximation and pattern classification with faster convergence rate and lesser computational load than an MLP structure. The FLANN is basically a flat net and the need of the hidden layer is removed and hence, the learning algorithm used in this network becomes very simple. The functional expansion effectively increases the dimensionality of the input vector and hence the hyper planes generated by the FLANN provide greater discrimination capability in the input pattern space.

To bridge the gap between the linearity in the single layer neural network and the highly complex and computation intensive multi layer neural network, the FLANN architecture is suggested. The FLANN architecture uses a single layer feed forward neural network and to overcome the linear mapping, functionally expands the input vector. Let each element of the input pattern before expansion be represented as z(i),1 < i < d where each element z(i) is functionally expanded as ( ) n z i ,1 < n < N , where N = number of expanded points for each input element. Expansion of each input pattern is done as follows.

$$x1(i) = z(i),\ x2(i) = f1(z(i)),\ldots..,\ xN(i) = fN(z(i))$$

where, z(i),1 < i < d , d is the set of features in the dataset.

These expanded input patterns (shown in Fig 3) are then fed to the single layer neural network and the network is trained to obtain the desired output. The set of functions considered for function expansion may not be always suitable for mapping the nonlinearity of the complex task. In such cases few more functions may be incorporated to the set of functions considered for expansion of the input dataset. However dimensionality of many problems itself are very high and further increasing the dimensionality by to a very large extent may not be an appropriate choice. So, it is advisable to choose a small set of alternate functions, which can map the function to the desired extent.

FLANN CLASSIFIER:

In this, a single layer model based on trigonometric expansion is presented. Let each element of the input pattern before expansion be represented as z(i),1 < i < I where each element z(i) is functionally expanded as zn (i),1 < n < N , where
N = number of expanded points for each input element. In this, N = 5 and I = total number of features in the dataset as been taken. Expansion of each input pattern is done as follows:

$$x(i) = z(i),\ x(i) = \sin \pi(z(i)),\ x(i) = \sin 2\pi(z(i)),\ x(i) = \cos \pi(z(i)),$$
$$x(i) = \cos 2\pi(z(i))$$

where, z(i),1 < i < d , d is the set of features in the dataset.

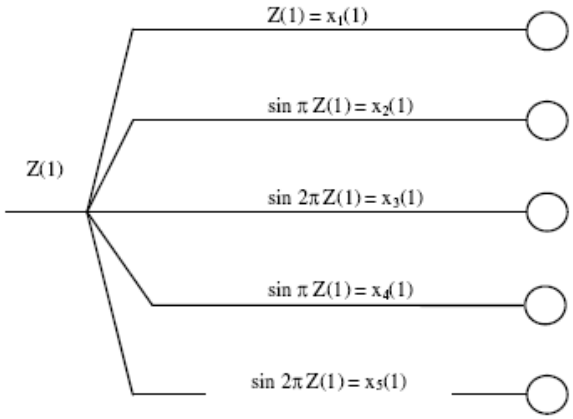


Fig3: Functional expansion of each unit

These nonlinear outputs are multiplied by a set of random initialized weights from the range [-0.5, 0.5] and then summed to produce the estimated output. This output is compared with the corresponding desired output and the resultant error for the given pattern is used to compute the change in weight in each signal path P, given by

$$\Delta W_j(k) = \mu * xf_j(k) * e(k)$$

where, xf j(k) is the functionally expanded input at kth iteration.

Then the equation, which is used for weight update, is given by

$$Wj(k+1) = Wj(k) + DWj(k)$$

where, Wj(k) is the jth weight at the kth iteration, m is the convergence coefficient, its value lies between 0 to 1 and 1<j<J, J = M´d. M is defined as the number of functional expansion unit for one element.

$$e(k) = y(k) - \hat{y}(k)$$

where, y(k) is the target output and $\hat{y}(k)$ is the estimated output for the respective pattern an and is defined as:

$$\hat{y}(k) = \sum xf_j(k) . w_k$$

where, xfj is the functionally expanded input at kth iteration and Wj(k) is the jth weight at the kth iteration and Wj(0) is initialized with some random value from the range [-0.5, 0.5]. The FLANN for classification shown in Fig4.

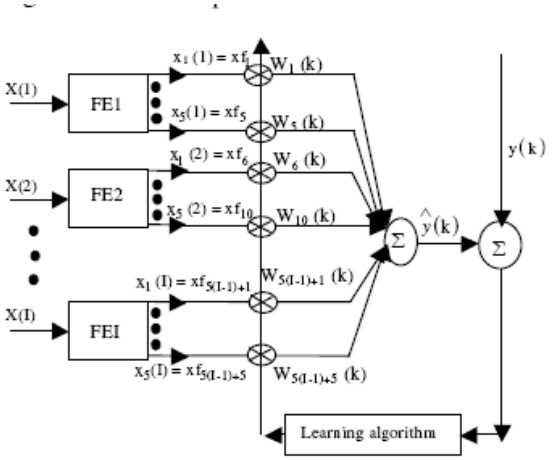


Fig4: FLANN nonlinear model for classification

V. Dataset Description

In our work we have used 3 different data sets for training and testing the neural network model. In this section we shall describe the details of the data sets. The data sets we used are IRIS dataset, DIABETES dataset and BLOOD TRANSFUSION dataset. And one bio-informatics dataset ECOLI dataset is also used (UCI machine learning repository and archive.ics.**uci**.edu/ml/datasets.html

IRIS Dataset:

This is perhaps the best known database to be found in the pattern recognition literature.The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other. The predicted attribute is class of iris plan.

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. Class:
   -- Iris Setosa
   -- Iris Versicolour
   -- Iris Virginica

PIMA INDIAN DIABETES Dataset:

Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Attribute Information:

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

ECOLI Dataset:

The reference below describes a predecessor to this dataset and its development. Reference: "Expert Sytem for Predicting Protein Localization Sites in Gram-Negative Bacteria", Kenta Nakai & Minoru Kanehisa, PROTEINS: Structure, Function, and Genetics 11:95-110, 1991.

Attribute Information:

1. Sequence Name: Accession number for the SWISS-PROT database

2. mcg: McGeoch's method for signal sequence recognition.
3. gvh: von Heijne's method for signal sequence recognition.
4. lip: von Heijne's Signal Peptidase II consensus sequence score. Binary attribute.
5. chg: Presence of charge on N-terminus of predicted lipoproteins. Binary attribute.
6. aac: score of discriminant analysis of the amino acid content of outer membrane and periplasmic proteins.
7. alm1: score of the ALOM membrane spanning region prediction program.
8. alm2: score of ALOM program after excluding putative cleavable signal regions from the sequence.

To overcome the skewing/biasing effect we have divided the Ecoli dataset into two different datasets naming as Ecoli (1) and Ecoli (2).

BLOOD TRANSFUSION Dataset:

To demonstrate the RFMTC marketing model (a modified version of RFM), this study adopted the donor database of Blood Transfusion Service Center in Hsin-Chu City in Taiwan. The center passes their blood transfusion service bus to one university in Hsin-Chu City to gather blood donated about every three months. To build a FRMTC model, we selected 748 donors at random from the donor database. These 748 donor data, each one included R (Recency - months since last donation), F (Frequency - total number of donation), M (Monetary - total blood donated in c.c.), T (Time - months since first donation), and a binary variable representing whether he/she donated blood in March 2007 (1 stand for donating blood; 0 stands for not donating blood).

Attribute Information:

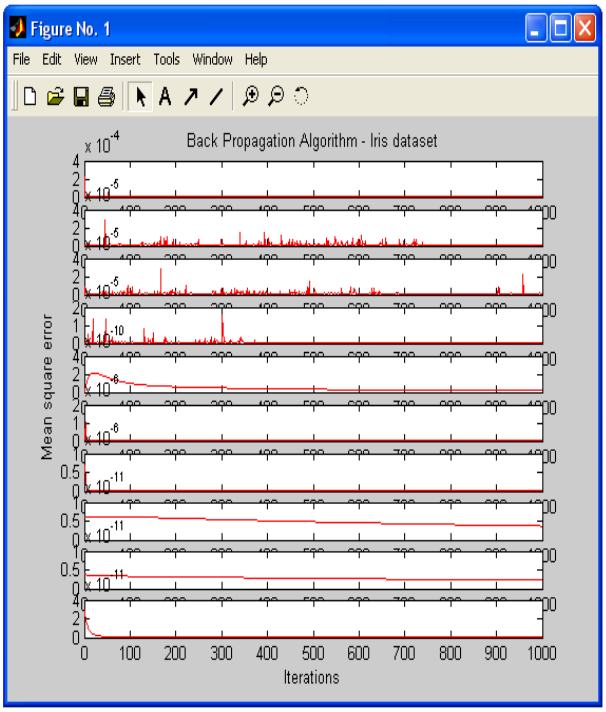
Given is the variable name, variable type, the measurement unit and a brief description. The "Blood Transfusion Service Center" is a classification problem. The order of this listing corresponds to the order of numerals along the rows of the database.
R (Recency - months since last donation),
F (Frequency - total number of donation),
M (Monetary - total blood donated in c.c.),
T (Time - months since first donation), and
a binary variable representing whether he/she donated blood in March 2007 (1 stand for donating blood; 0 stands for not donating blood).
Variable Data Type Measurement Description min max mean std Recency quantitative Months Input 0.03 74.4 9.74,8.07 Frequency quantitative Times Input 1 50 5.51 5.84 Monetary quantitative c.c. blood Input 250 12500 1378.68,1459.83 Time quantitative Months Input 2.27 98.3,34.42,24.32 Whether he/she donated blood in March 2007 binary 1=yes 0=no Output 0 1 1 (24%) 0 (76%) .

VI. Simulation and Results

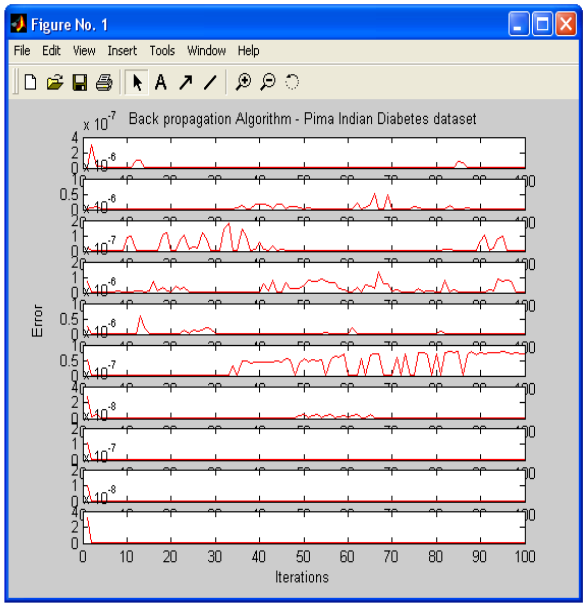
The neural network, which we are using in back propagation algorithm, is *m*-*n*-1 type network. It represents that input layer would contain 'n' nodes, which will be equal to the number of attributes in the dataset we are using. Say for example in Iris dataset we have 4 attributes so for construction neural network for this we require 4 nodes in its input layer. In the above notation m represents nodes in hidden layer (only one hidden layer we are considering). We can have any number of nodes in hidden layer and in output layer we are considering only one node.

For training and testing we have adopted 10-fold cross-validation for Iris, Pima Indian Diabetes, and Blood transfusion datasets in which we divide tuples in the dataset into 10 equal divisions. We apply back propagation algorithm for first 9 divisions and train the network for certain number of iterations. After training the network we then apply same algorithm without propagation of errors back and find the accuracy of the 10^{th} division. We repeat this process for all the remaining divisions by placing the last division on the top moving down the remaining tuples Such that each division will take part in training the network. However, we have used 3 fold-cross validations for Ecoli dataset.

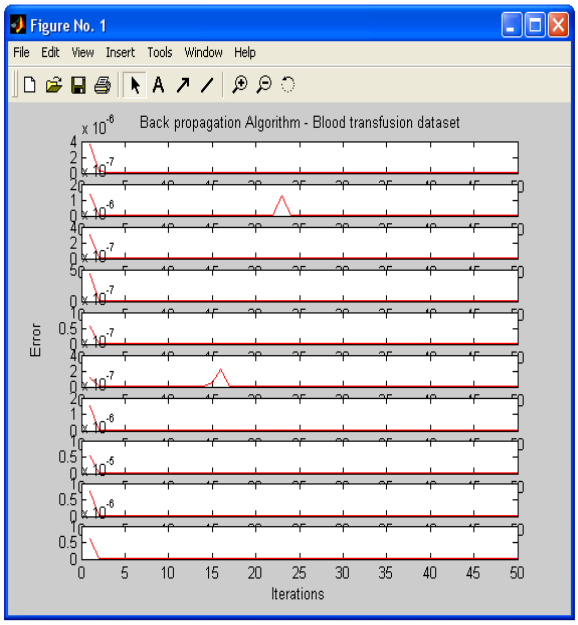
GA is used for optimally selecting the *n* values. For Iris dataset the range of probable *n* values are taken from 2 to 15. And after applying the GA it is found that the classifier provides best result with the SANN model described by 4-4-1, or in other words having 4 neurons in hidden layer. The percentage of correct classification obtained for this model with learning rate 0.9 and momentum constant 0.9 with 500 epoch and 10 simulations is 98.667 (best value),96.8667 (average value) with 0.0965 standard deviation. The mean square error plot is shown below for Iris dataset.



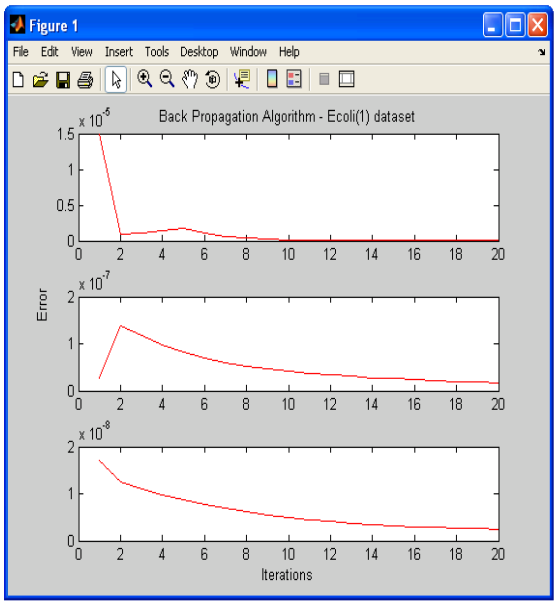
For Pima dataset the SANN gives the best accuracy with 5 neurons in the hidden layer. Best accuracy being 72% with average accuracy of 72.2%. The MSE is at 1.6838e-004. The error plot is shown below.



Similarly, for Blood Transfusion dataset the SANN is arrived at 4-4-1. The model provides 76.1333 % of best correct classification with 76.0533% average over 500 epoch and 10 simulation runs. The standard deviation is being 0.0891. The MSE plot is shown below.

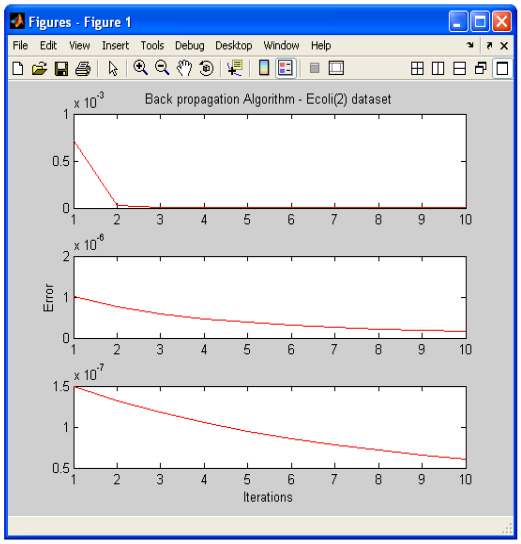


For Ecoli(1) dataset the SANN model settles at 7-3-1 with 96.2991 best accuracy and 96.3413 average accuracy. The epoch size is 20 and 10 simulation runs are done for this dataset. The std obtained is 0.3993. the MSE plot is shown below.



For Ecoli(2) the 7-2-1 SANN shows best accuracy of 68.1881 and average accuracy of 56.3636 with 0.5

learning parameter and 0.5 momentum constant. The MSE plot is as shown below.

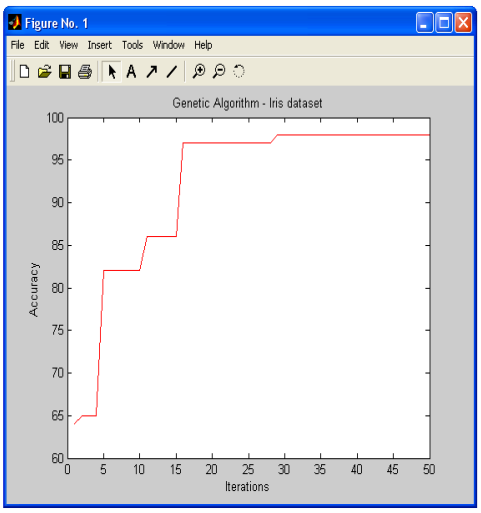


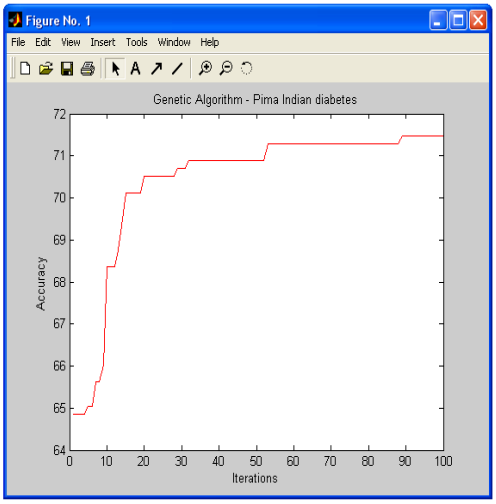
The results obtained using BP algorithm for all the investigated dataset reveal that the networks may have fall into the local minima. It appears that results can be improvised if some randomized optimization techniques is used for training the SANN i.e optimizing weights of the SANN. This has motivated us to explore the use of GA for optimizing the weights of SANN. In our experiments we have used the same SANN which we have obtained with GA and trained with BP. For example for Iris dataset we take the SANN 4-4-1 network, for Pima the SANN is 8-5-1, for Blodd Transfusion the SANN is 4-4-1 and for E-coli(1) it is 7-2-1 and Ecoli(2) 7-3-1. The weights of these networks are optimized using GA. The chromosomes of GA are the strings of real numbers randomly chosen in a range. The length of chromosome is determined by the number of weights to be optimized. The number of weights to be optimized is $(m*n+n*p)$. For example, in Iris dataset SANN model total weights to optimized are (4*4 + 4*1=20) 20. In all our simulation the population size is taken to be 30. The performances of these models are shown in the table 1given below.

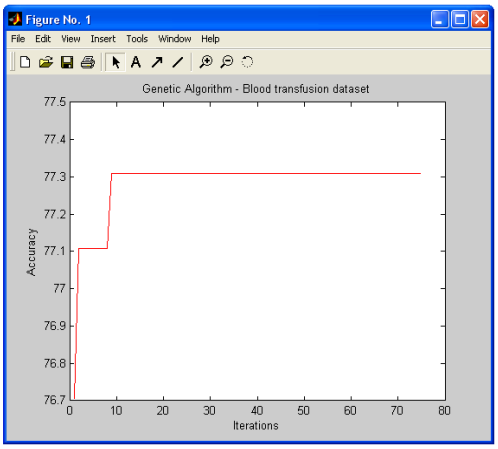
Table:1

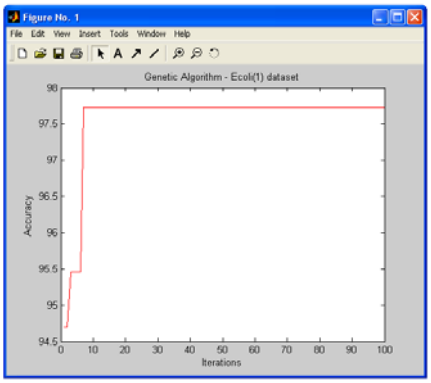
Model/dataset	Best Accuracy	Avg accuracy(std)
4-4-1/Iris	100	98.467(0.023)
8-5-1/Pima	73.438	71.212 (0.324)
4-4-1/Blood Tansfusion	81.93	78.0154(0.256)
7-2-1/Ecoli(1)	92.428	88.4848(0.453)
7-3-1/Ecoli(2)	85.714	80.857(3.234)

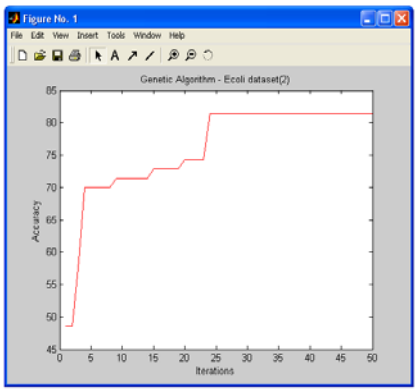
The results reveal GA based optimization for SANN is far more accurate with comparison to BP based training. However, for Ecoli(2) dataset the BP algorithm provides better result. The fitness curves for each datasets are given below.









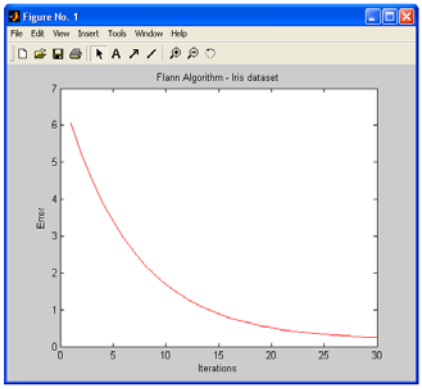


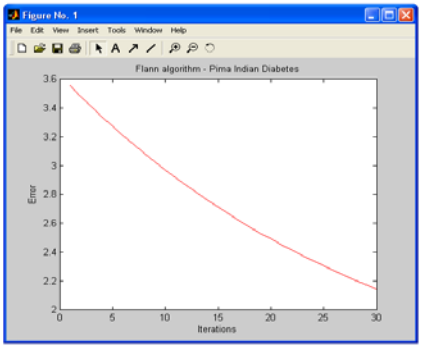
The simulated results of FLANN models are shown below in a tabular format in Table 2. All experimenst are done for 30 epochs and 10 such simulations are considered for finding average correct classifications..

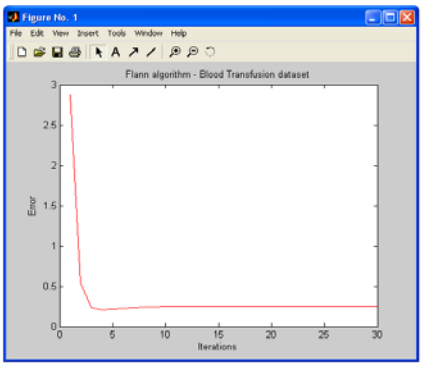
Table: 2

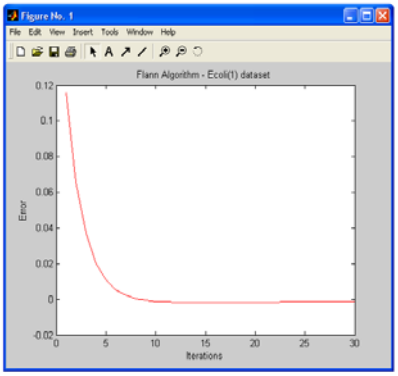
Dataset	Convergence co-efficient	Best accuracy	Avg accuracy
Iris	8e-006	98	74.6
Pima	8e-006	71.845	59.76
Blood Trans	8e-006	77.108	76.14
Ecoli(1)	8e-005	93.443	74.09
Ecoli(2)	8e-005	82.857	67.34

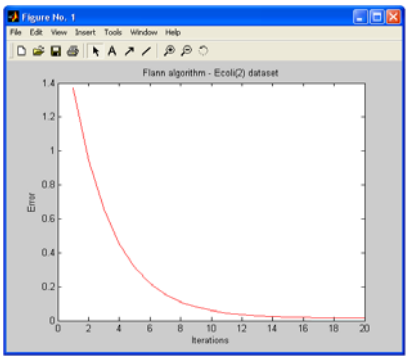
The error curves for FLANN for all models are shown below.











VII. Conclusion and further Research

This paper has explored the design of a simple ANN for data classifications. The simple ANN termed as Simple ANN (SANN) is envisioned by using GA for optimally deciding the number of neurons in a single hidden layer architecture. The weights of such SANNs are trained using BP algorithm and GA algorithm respectively. The results shown in the paper give very clear impression of the simplicity of the model without sacrificing the cost of accuracy. Contrary to the views of many researchers it is felt that we can have simple ANN model or Simple ANN model with only one hidden layer and having very few neurons in the hidden layer. Even our designed SANN outperforms FLANN classification models in all dataset except Ecoli(2). However, in Ecoli(2) SANN gives a competition to FLANN as regard to the percentage of correct classification. Due to less computational cost at the hidden layer SANN can have applications in the real time domain.

However, this study is too early to claim the universal ness of the model. It left for further study to see how the other models like Bayesian classifier, Decision tree etc behave with comparison to our suggested models. Also we can further investigate to improvise the classification accuracies using some other randomized optimization techniques. Performance comparisons with some other well known approach for data classification will also be a good direction for future work.

References

[1] A.K.Jain, R.P.W. Duin, and J.Mao, Statistical Pattern Recognition: A Review, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 22(1), January 2000, pp.4-37.

[2] R.O.Duda and P.E.Hard, Pattern classification and Scene Analysis, John wiley & Sons, NY, USA, 1973.

[3]Breitman,L.,Friedman,J.H.,Olshen,R.A.,C.J., Classification and Regression tress,Wadsworth,Belmont,CA,1984.

[4] Buntine,W.L., Learning classification trees, Statistics and Computing, 1992,pp. 63-73.

[5] Cover,T.M.,Hart,P.E., Nearest neighbors pattern classification, IEEE Trans on Information Theory, vol. 13, ,1967,pp. 21-27.

[6] Hanson R.,Stutz,J.,Cheeseman,P., Bayesian classification with correlation and inheritance, Proceedings of the 12th International Joint Conference on Artificial Intelligence 2, Sydney,Australia,Morgan KaufSANN, 1992,pp. 692-698.

[7] Michie,D. et al , Machine Learning, Neural and Statistical Classification, Ellis Horwood,1994.

[8] Richard,M.D, LippSANN,R.P., Neural network classifiers estimate Bayesian a-posterior probabilities, Neural Computation ,vol.3, ,1991,pp. 461-483.

[9] Tsoi, A.C et al, Comparison of three classification Techniques, CART,C4.5 and multilayer perceptrons , Advances in Neural Information Processing Systems, vol. 3, 1991 pp.963-969.

[10] C. Bishop, *Neural Networks for Pattern Recognition*. New York: Oxford Univ. Press, 1995.

[11] V.N.Vapnik, A.Y. Chervonenkis, On the uniform convergence of relative frequencies of events to

their probabilities, Theory of Probability and its Applications, 1971, pp.264-280.
[12]D.E.Goldberg, "Genetic Algorithms in Search, Optimization and machine Learning", Addision-Wesley, New York, 1989.
[13] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks*, Reading, MA: Addison-Wesley, 1989.

Gunanidhi Pradhan has completed M E. in Computer Sc. From N.I.T., Rourkela and M.B.A. in Human Resource Management from North Orissa University, Baripada. Currently perusing Ph.D. in the Department of Information & Communication Technology, Fakir Mohan University, Balasore. He hass about 20 years of Teaching experience and 1 year of Industrial experience in a Software Firm. Research interest areas are Soft Computing, Bio-Informatics and Data Mining. Authored two text books in Computer Networks and Programming in C. Awarded the Best Teacher award for the year 2008 by Indian Society for Technical Education, New Delhi. Presently working as the HOD (Information Technology) at BOSE, Cuttack.

Vishal Korimilli, pursuing his Final Year Computer Science and Engineering in ANITS, Vishakapatnam, AP, INDIA. His interests include Data Mining, Soft Computing and Swarm Intelligence applications. He is also a Student Member of IEEE. He is very active in fundamental research activities in the Dept of CSE of ANITS.

Suresh Chandra Satapathy has submitted his PhD to JNTU, Hyderabad, Andhra pradesh, India on pattern Classification using Swarm Intelligence Techniques in 2009. He has completed his M.Tech from NIT, Rourkela, Orissa, India. Presently he is working as professor in computer science and engineering in Anil Neerukonda Institute of Technology and Sciences, Vishakapatnam Dist, AP, India. His research area includes PSO, GA. Neural network and Data mining. He is reviewer for many international journals including from Elsevier sciences.

Dr. Sabyasachi Pattnaik has done his B.E in Computer Science, M Tech.from IIT Delhi. He has received his PhD degree in Computer Science in the year 2003, now working as Reader in the Department of Information and Communication Technology, in Fakir Mohan University, Vyasavihar, Balasore, Orissa, India. He has got 15 years of teaching and research experience in the field of neural networks, soft computing techniques. He has got 22 publications in national & international journals and conferences. He has published three books in office automation, object oriented programming using C++ and artificial intelligence. At present he is involved in guiding 6 scholars in the field of neural networks in cluster analysis, bio-informatics, computer vision & stock market applications. He has received the best paper award & gold medal from Orissa Engineering congress in 1992 and institution of Engineers in 2009.

Dr B Mittra, Reader, School of Biotechnology, F.M.University, Orissa, working in the area of proteomics and Bio-informatics. He has fifteen years of research experiences and produced research papers in many international journals related to molecular biology, immunotechnology, and proteomics.