Journal of Computer Science 3 (12): 948-955, 2007

Functional Link Artificial Neural Network for Classification Task in Data Mining

[1]B.B. Misra and [2]S. Dehuri
[1]Department of Computer Science,
College of Engineering Bhubaneswar, Orissa, India
[2]Department of Information and Communication Technology,
Fakir Mohan University, Vyasa Vihar-756019, India

Abstract: In solving classification task of data mining, the traditional algorithm such as multi-layer perceptron takes longer time to optimize the weight vectors. At the same time, the complexity of the network increases as the number of layers increases. In this study, we have used Functional Link Artificial Neural Networks (FLANN) for the task of classification. In contrast to multiple layer networks, FLANN architecture uses a single layer feed-forward network. Using the functionally expanded features FLANN overcomes the non-linearity nature of problems, which is commonly encountered in single layer networks. The features like simplicity of designing the architecture and low-computational complexity of the networks encourages us to use it in data mining task. An extensive simulation study is presented to demonstrate the effectiveness of the classifier.

Key words: Data mining, classification, functional link artificial neural networks

INTRODUCTION

For the past few years, there have been many studies[1] focused on the classification task in the emerging field of data mining. In classification, we are given a set of example records, called a training set, where each record consists of several fields or attributes. Attributes are either continuous, coming from an ordered domain or categorical coming from an unordered domain. One of the attributes called the classifying attribute indicates the class to which each example belongs. The objective of classification is to build a model of the classifying attribute based upon the other attributes.

Several classification models have been proposed over the years, e.g. statistical models like linear/quadratic discriminates[2], genetic models[3], decision trees[4,5] and neural networks[6]. Among these we found very rare literatures on neural networks specifically FLANN for classification task of data mining. Since it is sometimes difficult to search the optimal nonlinear boundary for a classification problem other than neural network models. Hence the nonlinear learning capabilities of Artificial Neural Networks (ANNs) have become a powerful tool for many complex applications including functional approximation, nonlinear system identification and control, unsupervised classification and optimization. The ANN's are capable of generating complex mapping between the input and the output space and thus these networks can form arbitrarily complex nonlinear decision boundaries. The traditional algorithms also takes longer time to optimize the weight vectors and their complexity increases as the number of layers increases. Hence, to resolve few of the issues, in this study we use functional link artificial neural network for solving the classification problem.

Pao *et al.*[7] 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.

Introduction to artificial neural networks: Over the past decade, Artificial Neural Network (ANN) has become increasingly popular in many disciplines as a problem-solving tool. ANN has the ability to solve extremely complex problems with highly non-linear relationships. ANN's flexible structure is capable of approximating almost any input output relationships. Particularly ANN has been extensively used as a tool in

Corresponding Author: B.B. Misra, Department of Computer Science, College of Engineering Bhubaneswar, Orissa, India

many disciplines to solve different types of problems such as forecasting, identification and control, classification and optimization. Complex and heterogeneous systems are extremely difficult to model mathematically. However, it has been proved that ANN's flexible structure can provide simple and reasonable solutions to various problems.

A formal computational model of neural network: Let us first recall a general model of an artificial neural network that consists of s simple computational units or neurons, indexed as V = {1, . . . , s}, where s = |V| is called the network size. Some of these units may serve as external inputs or outputs and hence we assume that the network has n input and m output neurons, respectively. The remaining ones are called hidden neurons. The units are densely connected into an oriented graph representing the architecture of the network, in which each edge (i, j) leading from neuron i to j is labeled with a real (synaptic) weight $w(i,j) = w_{ji} \in \Re$ The absence of a connection within the architecture corresponds to a zero weight between the respective neurons.

The computational dynamics of a neural network determines for each neuron $j \in V$ the evolution of its real state (output) $y_j^{(t)} \in \Re$ as a function of time $t \geq 0$. This establishes the network state $y^{(t)} = \left(y_1^{(t)},...,y_S^{(t)}\right) \in \Re^s$ at each time instant $t \geq 0$. At the beginning of a computation, the neural network is placed in an initial state $y^{(0)}$, which may also include an external input. Typically, a network state is updated by a selected subset of neurons collecting their inputs from the outputs of their incident neurons via the underlying weighted connections and transforming these input values into their current states. Finally, a global output from the network is read at the end of computation, or even in the course of it.

In general the models that we use to solve complex problems are multi-layer neural network. There are many algorithms to train the neural network models. However the models being complex in nature, one single algorithm cannot be claimed as best for training to suit different scenarios of the complexities of real life problems. Depending on the complexities of the problems, the number of layer and number of neuron in the hidden layer need to be changed. As the number of layers and the number of neurons in the hidden layer increases, training the model becomes further complex. Very often different algorithms fail to train the model for a given problem set. However we try to find an alternative algorithm, which will train the model to provide us with an output possibly not good enough to our expectation. In the process we develop one model containing many hidden layers and neurons, which is very complex to train and computation intensive.

FLANN architecture: To overcome the complexities associated with multi-layer neural network, single layer neural network can be considered as an alternative approach. But the single layer neural network being linear in nature very often fails to map the complex nonlinear problems. The classification task in data mining is highly nonlinear in nature. So solving such problems in single layer feed forward artificial neural network is almost an impossible task.

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[1]. 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 $z_n(i), 1 < n < N$, where N = number of expanded points for each input element. Expansion of each input pattern is done as follows.

$$\underbrace{x_1(i) = z(i),\ x_2(i) = f_1(z(i)),....,\ x_N(i) = f_N(z(i))} \qquad (1)$$

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

These expanded input pattern 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.

Classification: The digital revolution has made digitized information easy to capture and fairly inexpensive to store[8,9]. With the development of

computer hardware and software and the rapid computerization of business, huge amount of data have been collected and stored in databases. The rate at which such data stored is growing at a phenomenal rate. As a result, traditional ad-hoc mixtures of statistical techniques and data management tools are no longer adequate for analyzing this vast collection of data.

Raw data is rarely of direct benefit. Its true value is predicated on the ability to extract information useful for decision support or exploration and understanding the phenomenon governing the data source. In most domains, data analysis was traditionally a manual process. One or more analysts would become intimately familiar with the data and with the help of statistical techniques, provide summaries and generate reports. In effect, the analyst acted as a sophisticated query processor. However, such an approach rapidly breaks down as the size of data grows and the number of dimensions increases. When the scale of data manipulation, exploration and inferencing goes beyond human capacities, people look to computing technologies for automating the process.

All these have prompted the need for intelligent data analysis methodologies, which could discover useful knowledge from data. The term KDD refers to the overall process of knowledge discovery in databases. Data mining is a particular step in this process, involving the application of specific algorithms for extracting patterns (models) from data[10]. Supervised pattern classification is one of the important tasks of data mining.

Supervised pattern classification can be viewed as a problem of generating appropriate class boundaries, which can successfully distinguish the various classes in the feature space[11]. In real-life problems, the boundaries between different classes are usually nonlinear. It is known that using a number of hyperplanes can approximate any nonlinear surface. Hence, the problem of classification can be viewed as searching for a number of linear surfaces that can appropriately model the class boundaries while providing minimum number of misclassified data points.

The goal of pattern classification[12] is to assign input patterns to one of a finite number, M, of classes. In the following, it will be assumed that input patterns consist of static input vectors x containing N elements or continuous valued real numbers denoted x_1, x_2,..., x_N. Elements represent measurements of features selected to be useful for distinguishing between classes. Input patterns can be viewed as points in the multidimensional space defined by the input feature measurements. The purpose of a pattern classifier is to partition this multidimensional, space into decision regions that indicate to which class any input belongs. Conventional Bayesian classifiers characterize classes by their probability density functions on the input features and use Bayes' decision theory to form decision regions from these densities[13,14]. Adaptive non-parametric classifiers do not estimate probability density functions directly but use discriminant functions to form decision regions.

Application of a pattern classifier first requires selection of features that must be tailored separately for each problem domain. Features should contain information required to distinguish between classes, be insensitive to irrelevant variability in the input and also be limited in number to permit efficient computation of discriminant functions and to limit the amount of training data required. Good classification performance requires selection of effective features and also selection of a classifier that can make good use of those features with limited training data, memory and computing power. Following feature selection, classifier development requires collection of training and test data and separate training and test or use phases. During the training phase, a limited amount of training data and a priori knowledge concerning the problem domain is used to adjust parameters and/or learn the structure of the classifier. During the test phase, the classifier designed from the training phase is evaluated on new test data by providing a classification decision for each input pattern. Classifier parameters and/or structure may then be adapted to take advantage of new training data or to compensate for nonstationary inputs, variation in internal components, or internal faults. Further evaluations require new test data.

It is important to note that test data should never be used to estimate classifier parameters or to determine classifier structure. This will produce an overly optimistic estimate of the real error rate. Test data must be independent data that is only used to assess the generalization of a classifier, defined as the error rate on never-before-seen input patterns. One or more uses of test data, to select the best performing classifier or the appropriate structure of one type of classifier, invalidate the use of that data to measure generalization. In addition, input features must be extracted automatically without hand alignment, segmentation, or registration. Errors caused by these processes must be allowed to affect input parameters as they would in practical applications where extensive hand-tuning is normally impossible. Unfortunately, these simple guidelines, restricting use of test data and limiting hand-tuning and also other important common-sense guidelines discussed in[15], are frequently broken by pattern recognition researchers.

Supervised training, unsupervised training, or combined unsupervised/supervised training can be used to train neural net classification and clustering algorithms. Classifiers trained with supervision require data with side information or labels that specify the correct class during training. Clustering or vector quantization algorithms use unsupervised training and group unlabeled training data into internal clusters. Classifiers that use combined unsupervised/supervised training typically first use unsupervised training with unlabeled data to form internal clusters. Labels are then assigned to clusters and cluster centroid locations and sizes are often altered using a small amount of supervised training data. Although combined unsupervised/supervised training mimics some aspects of biological learning, it is of interest primarily because it can reduce the amount of labeled training data required. Much of the expense and effort required to develop classifiers results from the necessity of collecting and hand-labeling large amounts of training data. Combined unsupervised/supervised training can simplify data collection and reduce expensive hand labeling.

Back-propagation classifier: Back-propagation classifiers form nonlinear discriminant functions using single- or multi-layer perceptrons with sigmoidal nonlinearities. They are trained with supervision, using gradient-descent training techniques, called back-propagation. Which minimize the squared error between the actual outputs of the network and the desired outputs. Patterns are applied to input nodes that have linear transfer functions. Other nodes typically have sigmoid nonlinearities. The desired output from output nodes is low (0 or <0.1) unless that node corresponds to the current input class, in which case it is high (1.0 or >0.9). Each output node computes a nonlinear discriminant function that distinguishes between one class and all other classes. Good introductions to back-propagation classifiers are available in many studys. including[9,20]. Early interest in back-propagation training was caused by the presupposition that it might be used in biological neural nets.

Figure 1 shows how the multi-layer perceptron can form three nonlinear input/output functions using back-propagation training. The multi-layer perceptron shown has n linear input node, p nodes with sigmoidal nonlinearities in the first hidden layer and one linear output node

One major characteristic of back-propagation classifiers is long training times. Training times are typically longer when complex decision regions are required and when networks have more hidden layers. As with other classifiers, training time is reduced and performance improved if the size of the network is tailored to be large enough to solve a problem but not so large that too many parameters must be estimated with limited training data.

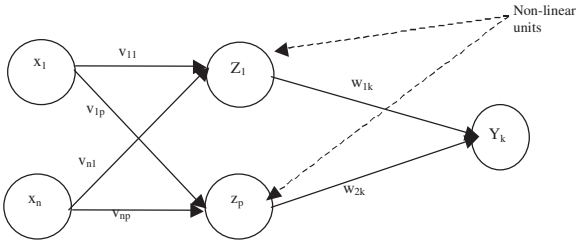


Fig. 1: Multi layer feed forward artificial neural network

FLANN classifier: In this study, 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 $z_n(i), 1 < n < N$, where N = number of expanded points for each input element. In this study, N = 5 and I = total number of features in the dataset has been taken.

Expansion of each input pattern is done as follows:

$$\underbrace{x_1(i) = z(i), x_2(i) = \sin \pi(z(i)), x_3(i) = \sin 2\pi(z(i)), \quad x_4(i) = \cos \pi(z(i)), x_5(i) = \cos 2\pi(z(i))}_{} \tag{2}$$

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

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

$$\Delta W_j(k) = \mu \times xf_j(k) \times e(k) \tag{3}$$

where, $xf_j(k)$ is the functionally expanded input at k^{th} iteration.

If there are p patterns to be applied then average change in each weight is given by

$$\overline{\Delta W_j(k)} = \frac{1}{p}\sum_{i=1}^{P} \Delta W_j^i(k) \tag{4}$$

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

$$W_j(k+1) = W_j(k) + \Delta W_j(k) \quad (5)$$

where, $W_j(k)$ is the j^{th} weight at the k^{th} iteration, μ is the convergence coefficient, its value lies between 0 to 1 and $1<j<J$, $J = M \times d$. M is defined as the number of functional expansion unit for one element.

$$e(k) = y(k) - \hat{y}(k) \quad (6)$$

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

$$\hat{y}(k) = \sum_{j=1}^{J} xf_j(k).w_j(k)$$

where, xf_j is the functionally expanded input at k^{th} iteration and $W_j(k)$ is the j^{th} weight at the k^{th} iteration and $W_j(0)$ is initialized with some random value from the range [-0.5, 0.5]

Figure 2 and 3 shows the functional expansion unit for one element and FLANN architecture respectively.

Experimental studies: The performance of the FLANN model is evaluated using the five benchmark classification databases. Out of these, the most frequently used in the area of neural networks and of neuro-fuzzy systems are IRIS, WINE, PIMA, BUPA Liver Disorders and HEART Disease datasets. All these databases are taken from the UCI machine repository[27] and its corresponding site is ftp: //ftp.ics.uci. edu/pub/machine-learning-databases/. In addition, we have compared the results of FLANN with other competing classification methods using the aforesaid datasets.

Description of the datasets: Let us briefly discuss the datasets, which we have taken for our experimental setup.

IRIS dataset: a classification data set based on characteristics of a plant species (length and thickness of its petal and sepal) divided into three distinct classes (Iris Setosa, Iris Versicolor and Iris Virginica).

WINE dataset: data set resulting from chemical analyses performed on three types of wine produced in Italy from grapevines cultivated by different owners in one specific region.

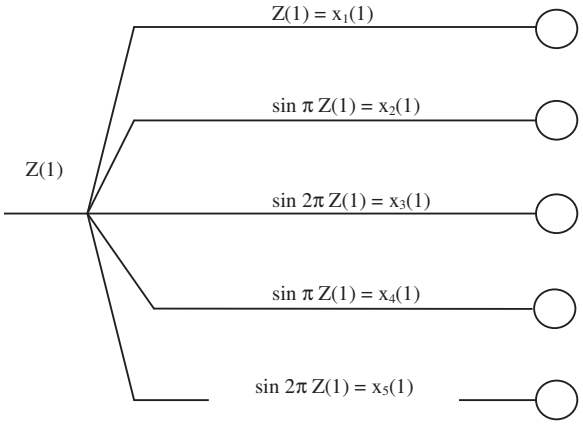


Fig. 2: Functional expansion of the first element

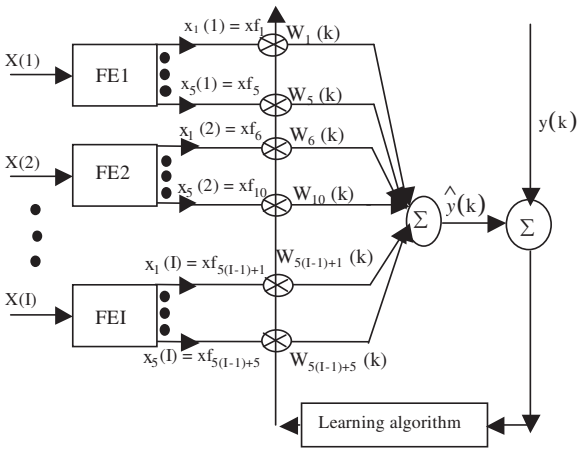
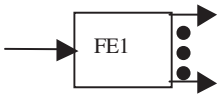


Fig. 3: Proposed nonlinear model for classification



Block form representation of Fig. 1

PIMA Indians diabetes database: data set related to the diagnosis of diabetes (with or without the disease) in an Indian population that lives near the city of Phoenix, Arizona.

BUPA liver disorders: data set related to the diagnosis of liver disorders and created by BUPA Medical Research, Ltd.

Heart disease: data set related to diagnoses of people with heart problems.

Table 1 presents a summary of the main features of each database that has been used in this study.

Table 1: Description of the features of the databases employed

	No.of patterns	No. of attributes	No. of classes	No. of patterns in class1	No. of patterns in class 2	No. of patterns in Class 3
Iris dataset	150	4	3	50	50	50
Wine dataset	178	13	3	71	59	48
Pima indian diabetes dataset	768	8	2	500	268	
Bupa liver disorders	345	6	2	145	200	
Heart disease	270	13	2	150	120	

Table 2: Results obtained with the FLANN model for the classification of four databases: IRIS dataset, WINE data, PIMA Indians diabetes database and bupa liver disorders

Data set used for testing	Hit % in the training set	Hit in the test set
iris1.dat	97.333	97.333
iris2.dat	100	100
Average IRIS	98.665	98.665
Wine1.dat	93.258	92.135
wine2.dat	98.876	98.876
Average WINE	96.058	95.506
pima1.dat	77.604	76.826
pima2.dat	79.948	79.427
Average PIMA	78.776	78.127
liver1.dat	73.837	75
liver2.dat	78.613	77.457
Average LIVER	76.225	76.229

Classification performance: In the case of the IRIS Dataset, WINE Dataset, PIMA Indians Diabetes Database and BUPA Liver Disorders, in order to generate the training and test sets, the total set of patterns was randomly divided into two equal parts (database1.dat and database2.dat). Each of these two sets was alternately used either as a training set or as a test set. Table 2 summarizes the results obtained in the classification of these four data sets with the use of the FLANN model.

The average values (in bold type) of each application will be used for comparisons with other classification methods (Table 5).

As for the Heart Disease database (extracted from the StatLog project[16]), the tests were carried out with the use of the 9-Fold Cross Validation methodology[16], the same approach used by all algorithms that were analyzed by the StatLog project. This method consists of partitioning the database into nine subsets (heart1.dat, heart2.dat, heart3.dat, heart4.dat, heart5.dat, heart6.dat, heart7.dat, heart8.dat and heart9.dat), where eight subsets are used for training and the remaining subset is used for testing (validation). The process is repeated nine times in such a way that each time a different subset of data is used for testing.

Thus, the database was randomly segmented into nine subsets with 30 elements each. Each subset contains about 56% of Class1 records (without heart disease) and 44% of Class 2 records (with heart disease).

Table 3: Cost matrix of the results

	Model classification	
Real classification	Class 1 (Absence)	Class 2 (Presence)
Class 1 (Absence)	0	1
Class 2 (Presence)	5	0

The methodology also makes use of a cost matrix, which is described in Table 3. The purpose of such a matrix is to penalize wrongly classified records in different ways, depending on the class. The weight of the penalty for Class 2 records that are classified as Class 1 records is 5, while the weight of the penalty for Class 1 records that are classified as Class 2 records is 1.

Therefore, the cost of wrongly classifying the patterns in the training and test data sets is given by (7) and (8), respectively, as follows:

$$C_{Tr} = \frac{P1*5 + P2*1}{P_{Tr}} \quad (7)$$

$$C_{Te} = \frac{P1*5 + P2*1}{P_{Te}} \quad (8)$$

where:

C_{Tr} = Cost in the training set
C_{Tr} = Cost in the test set
P_1 = Number of patterns that were wrongly classified as belonging to Class 1
P_2 = Number of patterns that were wrongly classified as belonging o Class 2
P_{Tr} = Total number of patterns in the training set
P_{Te} = Total number of patterns in the test set

Table 4 presents the errors and costs of the training and test sets for the FLANN model. Upon closer inspection of Table 4, it may be observed that the configuration of the heart8.dat database as a test subset obtained lower errors and consequently a lower cost for the FLANN model.

Comparison with other models: The results obtained for the Iris Dataset, Wine Data, Pima Indians Diabetes

Table 4: Results obtained by the FLANN model for the classification of the heart disease database

Dataset used for testing	Error in the training set		Error in the test set		Cost in the training set	Cost in the test set
	Class 1	Class 2	Class 1	Class 2		
heart1.dat	13/133	14/107	1/17	1/13	0.34583	0.2
heart2.dat	14/133	12/107	2/17	1/13	0.30833	0.23333
heart3.dat	13/134	15/106	4/16	2/14	0.36667	0.46667
heart4.dat	13/133	10/107	1/17	4/13	0.2625	0.7
heart5.dat	13/133	16/107	3/17	2/13	0.3875	0.43333
heart6.dat	13/134	14/106	6/16	0/14	0.34583	0.2
heart7.dat	15/133	13/107	0/17	3/13	0.33333	0.5
heart8.dat	18/133	17/107	1/17	0/13	0.42917	0.033333
heart9.dat	20/134	9/106	2/16	1/14	0.27083	0.23333
Average					0.338888	0.333333

Table 5: Comparison of the average performance of several classification systems

	Iris dataset	Wine data	Pima Indians diabetes database	Bupa liver disorders
NN		95.2%	65.1%	60.4%
KNN		96.7%	69.7%	61.3%
FSS		92.8%	73.6%	56.8%
BSS		94.8	67.7	60.0
MFS1		97.6	68.5	65.4
MFS2		97.9	72.5	64.4
CART			74.5	
C4.5	94.0		74.7	
FID3.1	96.4		75.9	
MLP			75.2	
NEF class	96.0			
HNFB	98.67	98.31	77.08	74.49
HNFB fixed	98.67	97.8	78.0	
HNFB adaptive	98.67	97.8	78.6	
HNFQ	98.67	98.88	77.08	75.07
$HNFB^{-1}$	98.67	99.44	78.26	73.33
FLANN	98.67	95.51	78.13	76.23

Table 6: Table comparing the average cost in the training and test set of several classification systems evaluated for the heart disease database

Algorithm	Cost in test	Cost in training
FLANN	0.339	0.333
$HNFB^{-1}$	0.366	0.594
Bayes	0.374	0.351
Dicrim	0.393	0.315
LogDisc	0.396	0.271
Alloc80	0.407	0.394
QuaDisc	0.422	0.274
Castle	0.441	0.374
Cal5	0.444	0.330
Cart	0.452	0.436
Cascade	0.467	0.207
KNN	0.478	0
Smart	0.478	0.264
Dipol92	0.507	0.429
Itrule	0.515	-
BayTree	0.526	0.111
Default	0.560	0.560
BackProp	0.574	0.381
LVQ	0.600	0.140
IndCart	0.630	0.261
Kohonen	0.693	0.429
Ac2	0.744	0
Cn2	0.767	0.206
Radial	0.781	0.303
C4.5	0.781	0.439

Database and Bupa Liver Disorders data sets were compared with the results described in[17] where the performance of several models is presented: NN (nearest neighbor), kNN (k- nearest neighbor, FSS (nearest neighbor with forward sequential selection of feature) and BSS (nearest neighbor with backward sequential selection of feature). In addition, FLANN model has also been compared with other methods such as MFS (multiple feature subsets)[18], CART (CART decision tree)[19], C4.5 (C4.5 decision tree)[20], FID3.1 (FID3.1 decision tree)[21], MLP (multilayer perceptron)[22] and NEFCLASS[23]. Finally, the performance of FLANN was compared with the hierarchical neurofuzzy BSP models (Table 6) HNFB, HNFB fixed (which is the HNFB model with the same variable for all cells in the same level), HNFB_adaptive (the HNFB model with different variables for cells in the same level) and the Hierarchical Neuro-Fuzzy Quadtree (NFHQ) model[24], which uses the Quadtree partition of the input space[25]. The results were also compared with Inverted Hierarchical Neuro-Fuzzy BSP System ($HNFB^{-1}$)[26]. Table 5 presents a summary of the results obtained by the various different models. The best performance for each data set, measured in terms of each model's hit percentage, is highlighted in bold type.

The classification results found for the Heart Disease data set were compared with the results found in the StatLog project[16]. According to the StatLog project methodology, comparison consists of calculating the average cost produced by the nine data subsets used for validation. Table 6 presents the average cost for the nine training and test subsets. The result of the FLANN model is highlighted in bold.

CONCLUSION

In this study, we have evaluated the Functional Link Artificial Neural Network (FLANN) model for the task of pattern classification in data mining. The FLANN model functionally expands the given set of inputs. These inputs are fed to the single layer feed forward artificial neural network. The network is trained like Back propagation training methods. The experimental studies demonstrated that the FLANN model performs the pattern classification task quite well. In most cases, the results obtained with the FLANN model proved to be as good as or better than the best results found by the other models and algorithms with which it has compared. The performance of the FLANN models is remarkable in terms of processing time, which is also treated as one of the crucial aspect in data mining community. For all the databases described in the experimental studies, the models converged in an order of magnitude of less than one min of processing time on a Pentium IV 500 MHz computer.

REFERENCES

1. Agrawal, R., T. Imielinski and A. Swami, 1993. Database mining: A performance perspective. IEEE Trans. Knowledge Data Eng., 5: 914-925.
2. James, M., 1985. Classification Algorithms. Wiley.
3. Goldberg, D.E., 1989. Genetic algorithms in search, optimization and machine learning. Morgan Kaufmann.
4. Breiman, L., J.H. Friedman, R.A. Olshen and C.J. Stone, 1984. Classification and Regression Trees. Wodsworth, Belmont.
5. Quinlan, J.R., 1993. C4.5: Programs for Machine Learning. Morgan Kaufman.
6. Lippmann, R., 1987. An introduction to computing with neural networks. IEEE ASSP Mag., 4: 22.
7. Pao, Y.-H., S.M. Phillips and D.J. Sobajic, 1992. Neural-net computing and intelligent control systems. Int. J. Contr., 56: 263-289.
8. Fayyad, U. and R. Uthurusamy, 1996. Data mining and knowledge discovery in databases. Commun. ACM, 39: 24-27.
9. Inmon, W.H., 1996. The data warehouse and data mining. Commun. ACM, 39: 49-50.
10. Mitra, S., S.K. Pal and P. Mitra, 2002. Data mining in soft computing framework: A survey. IEEE Trans. Neural Networks, 13: 1.
11. Bandyopadhyay, S., S.K. Pal and B. Aruna, 2004. Multiobjective GAs, quantitative, indices and pattern classification. IEEE Trans. Syst. Man Cybernetics, Part-B, 34: 5.
12. Lippmann, R.P., 1989. Pattern classification using neural networks. IEEE Commun. Mag., pp: 47-64.
13. Duda, R.O. and P.E. Hart, 1973. Pattern Classification and Scene Analysis. NY: John Wiley and Sons.
14. Fukunaga, K., 1972. Introduction to Statistical Panern Recognition. NY: Academic Press.
15. Nagy, G., 1983. Candide's practical principles of experimental pattern recognition. IEEE Trans. Panern Anal. Mach. Intel., PAMI-5: 199-200.
16. Heart Disease Dataset. http:// www.ncc. up.pt /liacc /ML/ statlog/datasets/heart/heart.doc.html
17. Aha, D.W. and R.L. Bankert, 1994. Feature selection for case-based classification of cloud types: An empirical comparison. Proc. Am. Assn. for Artificial Intelligence (AAAI-94)-Workshop Case-Based Reasonings, pp: 106-112.
18. Bay, S.D., 1999. Nearest neighbor classification from multiple feature subsets. Intell. Data Anal., 3: 191-209.
19. Pattern Recognition and Neural Networks. http://129.186.1.21/~dicook/stat501/97/lectures/4.17.html
20. Quinlan, J.R., 1996. Improved use of continuous attributes in C4.5. J. Artif. Intell. Res., 4: 77-90.
21. Janikow, C.Z. and M. Faifer, 1999. Fuzzy partitioning with FID3.1. Proc. IEEE 18th Int. Conf. North Am. Fuzzy Inform. Processing Soc., pp: 467-471.
22. Haykin, S., 1999. Neural Networks-A Comprehensive Foundation. Englewood Cliffs, NJ: Prentice-Hall.
23. Klawonn, F., D. Nauck and R. Kruse, 1995. Generating rules from data by fuzzy and neuro-fuzzy methods. Proc. Fuzzy-Neuro-Syst., pp: 223-230.
24. de Souza, F.J., M.M.B.R. Vellasco and M.A.C. Pacheco, 2002. Hierarchical neuro-fuzzy quad tree models. Fuzzy Sets Syst., 130/2, pp: 189-205.
25. Finkel, R.A. and J.L. Bentley, 1974. Quad trees, a data structure for retrieval on composite keys. Acta Informatica, 4: 1-9.
26. Goncalves, L.B., M.M.B.R. Vellasco, F.J. de Souza, M.A.C. Pacheco, 2006. Inverted hierarchical neuro-fuzzy BSP system: A novel neuro-fuzzy model for pattern classification and rule extraction in databases. IEEE Trans. Syst. Man Cybernetics-Part C: Appl. Rev., 36: 2.
27. Blake, C.L. and C.J. Merz, UCI Repository of Machine Learning Databases. http://www.ics. uci.edu/~mlearn/MLRepository.html