ISSN: 0128-7680
Pertanika J. Sci. & Technol. **18 (1)**: 139– 154 (2010)

Grid Base Classifier in Comparison to Nonparametric Methods in Multiclass Classification

M. R. Mohebpour[*], **Adznan B. J. and M. I. Saripan**
Department of Computer and Communication System Engineering,
Faculty of Engineering, Universiti Putra Malaysia,
43400 UPM, Serdang, Selangor, Malaysia
[*]*E-mail:m_mohebpour@yahoo.com*

ABSTRACT

In this paper, a new method known as Grid Base Classifier was proposed. This method carries the advantages of the two previous methods in order to improve the classification tasks. The problem with the current lazy algorithms is that they learn quickly, but classify very slowly. On the other hand, the eager algorithms classify quickly, but they learn very slowly. The two algorithms were compared, and the proposed algorithm was found to be able to both learn and classify quickly. The method was developed based on the grid structure which was done to create a powerful method for classification. In the current research, the new algorithm was tested and applied to the multiclass classification of two or more categories, which are important for handling problems related to practical classification. The new method was also compared with the Levenberg-Marquardt back-propagation neural network in the learning stage and the Condensed nearest neighbour in the generalization stage to examine the performance of the model. The results from the artificial and real-world data sets (from UCI Repository) showed that the new method could improve both the efficiency and accuracy of pattern classification.

Keywords: Grid, decision boundaries, pattern recognition, supervised, nonparametric, multiclass

INTRODUCTION

There are growing needs for developing both powerful and robust methods to model a pool of data. In many applications, useful models can be defined in terms of their generalization capabilities. Various applications in engineering, statistics, computer sciences, and health sciences are concerned with estimating good predictive models from available data. In such problems, the goal is to estimate a model from unknown data patterns, and use this knowledge to predict future trends.

In statistical pattern recognition, the goal is to classify samples accurately based on a set of descriptive predictor variables (features). The design of pattern recognition systems involves two major tasks, as shown in *Fig. 1*. The description task transforms data collected from the environment into features, i.e. any value which can be derived from and is representative of the data that are used in the classification task to arrive at identification (Ho and Basu, 2002 ; Duda *et al.,* 2001). The end result of the description task is a set of features, commonly known as a *feature vector,* which constitutes a representation of the data. The classification task uses a classifier to map a feature vector to a group. Hence, after the observed data from the patterns to be recognized have been expressed in the form of pattern points or measurement vectors in the pattern space, the researchers want the machine to decide to which pattern group these data belong to.

Received: 9 September 2008
Accepted: 7 August 2009
[*]Corresponding Author

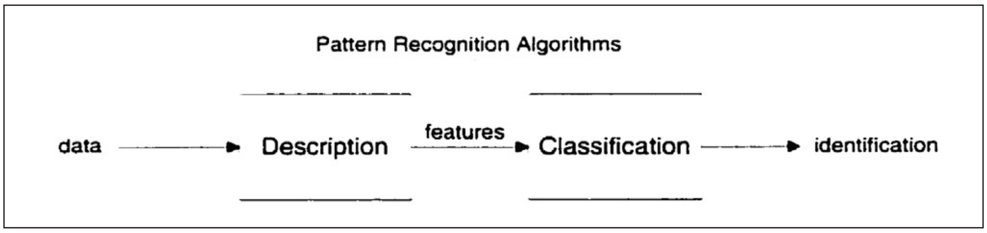

Fig. 1: Two separate tasks commonly used within pattern recognition systems

The two major approaches to building statistical pattern recognition systems are *parametric* and *non-parametric* methods (Duda *et al.,* 2001; Fukunaga, 1990). In the parametric methods, the forms of class-conditional distributions are assumed to be known and the unknown parameters of the distribution are estimated by maximum likelihood, maximum a posteriori, or Bayesian techniques based on a training set. Meanwhile, in the non-parametric methods, no global method is assumed for the class-conditional distributions; the class-conditional densities are therefore estimated in the regions of feature space determined by the training set (such as Parzen Windows and Nearest Neighbour). In other non-parametric methods, the forms of decision boundary are assumed to be known and their parameters are found as a solution to an optimization problem (such as Neural Network). The optimization criterion is, in general, composed of an error term (error on the training set) and regularization term (a smoothness constraint on the decision boundary). Then, the error rate on the training set serves as a good predictor of the error rate on the test set.

There are a large number of classifiers which have been presented in the literature; these include Nearest Neighbour, Parzen Window, Neural Networks, Decision Tree, etc. which are further categorized into two groups, namely *Eager* and *Lazy* methods. The eager classification uses training data to create a single-rule set during the training phase, and this particular rule set is used to classify all test instances. On the other hand, the lazy classification does not create a rule set during the training phase and the rule set generation is delayed until a test instance is given. Nevertheless, the choice between the eager and lazy algorithms is not always simple. Using a single-rule set to perform all the predictions, one may not take advantage of specific characteristics of a test instance. On the other hand, building a specific rule set for each test instance may require excessive computational efforts (Ho, 2004; Veloso and Meira, 2005).

Multiclass classifiers are important for handling classification problems with more than two categories. There are a lot of applications like handwritten recognition, object classification, speech recognition, bioinformatics, information retrieval, and text categorization which require multiclass classifier. Despite the well-known two-class classification problems, the multiclass classification problems are relatively less investigated. Many of the multiclass classifiers were designed by developing the existing binary class classifiers utilizing 'one versus one', or 'one versus rest' (Guobin and Murphey, 2007). The aim of this paper is to improve the efficiency and accuracy of training and generalization phases in the multiclass classification. To accomplish this goal, the study presents a method which, as compared to some previous methods, gives us a very efficient approach to modelling a multi-class pattern classification problem, decision modules, learning complexity, and system generalization.

Problem Definition

As a branch of artificial intelligence, pattern recognition has extensively been studied in computer science. The design of a complete pattern recognition system consists of multiple stages, which include: 1) data acquisition and pre-processing; 2) feature extraction and selection; 3) learning from examples; 4) testing and evaluation. These pre-processing step and feature extraction and selection are generally dependent very much on the specific application. In contrast, the statistical issues addressed by the classification step (learning and testing) are generally independent of the application.

This paper focuses on the last two stages, namely learning from examples and testing the classifier for evaluation (classification step). It is assumed that all tasks involving pre-processing (such as noise removal), as well as feature extraction and selection (such as dimension reduction) are applied to the dataset if they are required. Therefore, the system proposed in the present study would be given a dataset of labelled objects, whereby each object which is represented by a set of attributes belongs to one of a finite number of classes. The task is to find a rule which can be used to categorize an object into its right class, when only its attribute values are given. In relation to the fact that the classification steps are generally independent of the application, the virtual data made by random function in the MATLAB are generally used to cover various datasets with different complexities. Computer programs were also designed for the proposed classifier using the MATLAB programming tools to be applied on those virtual datasets and compared with other methods. The real datasets, such as the common problem XOR and dataset from UCI repository (Murphy and Aha, 1992), were also provided to test and evaluate the proposed classifier.

Among the current methods, some are efficient in the learning stage, but they are rather disorganized and time-consuming in the generalization stage as compared to the others which are rather time-consuming in the learning step but are effective in classification.

This study also proposed a new non-parametric classification method based on the grid structure, with the aim to improve the efficiency and accuracy of the training and generalization phases in the multiclass classification.

MATERIALS AND METHODS

In this section, a new pattern classification method called the Grid Base Classifier (GBC) based on the grid structure is proposed. Just like the eager method, the proposed model also uses training data to create a decision boundary during the training phase, but this procedure occurs in a very short time as compared to the current eager methods. In addition, predictions are performed using a decision boundary without excessive computational efforts, unlike the lazy method. Therefore, it is capable in both steps and radically improves the performance of the recognizer. Moreover, the GBC can be applied to the design of any kind of statistical recognizer and can be used in any pattern recognition application.

Grid-Based Classifier

The design of a pattern recognition system should consider two important aspects of the pattern recognition operation, namely "learning" and "generalization". In general, the system proposed in this study should be able to learn and generalize well. There is an important trade-off between learning and generalization that arises quite generally.

In the same vein, the better the time complexity of an algorithm, the faster the algorithm will practically carry out the work. Apart from time complexity, its space complexity is also important. This is essentially the number of memory cells required by an algorithm. In addition, a good algorithm also keeps this number as small as possible.

In general, the lazy methods take a short time to learn because they do not use any training data to create a rule as a decision boundary (low time complexity). Nevertheless, they take a long time to classify new pattern because they may require excessive computational efforts (high time complexity), and it is normally necessary to keep all the training data in memory (high space complexity).

On the other hand, the eager methods require a long time to learn in order to obtain a decision boundary because of their time-consuming algorithms (high time complexity). However, after the learning stage, the training data can be omitted (low space complexity), and they can classify new pattern by simply using the decision boundary very quickly (low time complexity).

Therefore, the proposed method follows the advantages of both methods as described above. Using the simple algorithm illustrated below, this method can find a decision boundary without any misclassification in a short time for the learning phase (low time complexity). Thus, keeping the training data is no longer required (low space complexity), and the classification can be done much quickly, i.e. by simply comparing it to the decision boundary (low time complexity).

The following section describes how the proposed method can learn and classify the multiclass (e.g. 4-class) problem.

Learning

The basic classification rules partition $\mathbb{R}$ into some cells $A_1, A_2, \ldots$ and classifies in them in each cell according to the majority vote among the labels of the falling in the cell containing. Formally,

$$\phi_n(z) = \begin{cases} 1 & \mathit{if} \ \sum_{i=1}^{n} I\{Z_i \in A(z)\} I\{Y_i = 1\} > \sum_{i=1}^{n} I\{Z_i \in A(z)\} I\{Y_i = 0\} \\ 0 & \mathit{otherwise} \end{cases} \tag{1}$$

Where *A(z)* denotes the cell containing *z*.

Furthermore, using the *grid based* idea, as described in the following section, partitioning $\mathbb{R}$ is done into cells containing one label. Thus, after the pre-processing stage of features extraction and selection, the training data are plotted using the Cartesian plane. After a grid is inserted into the space vector and divided into cells, each cell is then checked if it contains only one kind of data. Otherwise, the grid size must be decreased into smaller cells, and it will therefore be rechecked. Both the sizes of the grid are continuously decreased until a situation where only one kind of pattern remaining in each cell (see *Fig. 2*) is obtained.

When the size of the grid has been determined, the decision boundary is sought based on the cells, as shown in *Fig. 3.* After the decision boundary has been found, the training data are no longer required. Hence, they can be eliminated and the decision boundary can then be used as a decision function which separates the *M* pattern classes.

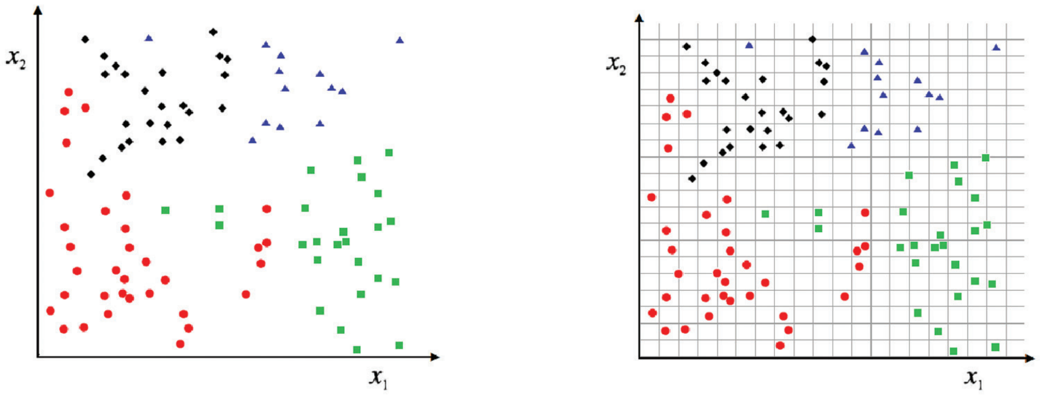

Fig. 2: (L) Multiclass features space and training data; (R) Desirable grid which guarantees that each cell contains only one kind of pattern

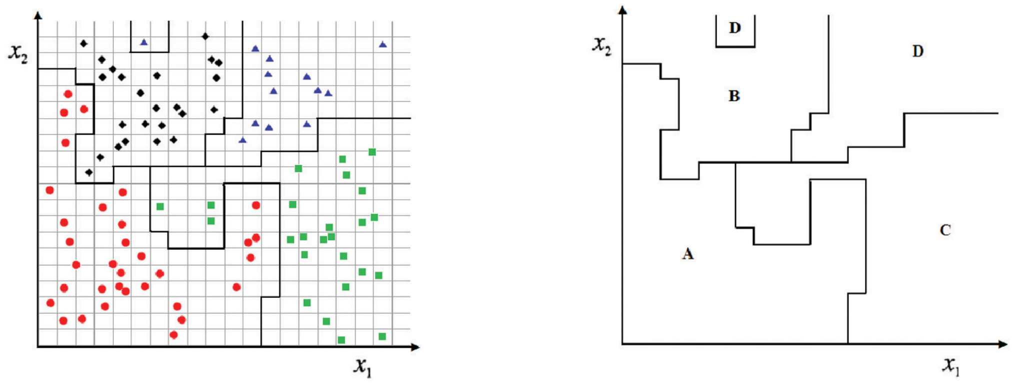

Fig. 3: (L) Finding the decision boundary based on boundary cells; (R) Eliminate the training data and keep the boundary cells

The above description is the main idea used to find the decision boundary in the method proposed in this study. There are various ways which can be used to find the grid size for this particular idea. Here, an optimal and practical way to create the appropriate grid used in the above algorithm is also proposed. Steps taken in obtaining the size of the grid are shown below:

To find α which is considered as the non-zero minimum of distances between the different patterns among the available classes and presents λ as the grid size, the following equation is used:

$$\alpha = \min\left\{\left\| m_i - n_j \right\|; m \neq n, m \in X, n \in Y, i, j = 1..K \right\} \tag{2}$$

where X, Y ∈ {Patterns Group]

Now, two different patterns with a minimum distance are determined; for instance, for *M* and *N,* λ is defined for the desirable grid size.

$$\lambda = \max\left\{ \left|M_{x_1} - N_{x_2}\right|, \left|M_{x_2} - N_{x_2}\right| \right\}/2 \tag{3}$$

Finding the grid size using this method guarantees that different patterns are separately located in the different cells. A grid is basically of a double array, with all the points falling in one cell are stored in the same place. All the points will be stored in the same entry if a cell contains more than one point. The next step is to scan the grid cells in order to find the decision boundaries and store them for the generalization phase. At this stage, the training data can be eliminated and the memory can be released.

Generalization

Another important issue related to the conventional pattern recognition design methods which must be addressed is generalization - the performance of the designed recognizer on the data outside the training set, i.e. correctly classify the test patterns it has never seen before, so that the new pattern (novel) belongs to which class can be recognized using the decision boundary which can be found in the previous stage. As the decision boundaries are stored in a hash table, testing the novel can be done using any kind of hash function very quickly. In comparison to the generalization step in the lazy algorithms (which involves checking the relation by all training data or some part of the data that are sensitive) to find a good estimation for the novel, this method seems to be very effective in computational and time complexity.

RESULTS AND DISCUSSION

The strength of a given classification technique is measured by the number of different distinguishable categories (Sarlashkar, Bodruzzaman and Malkani, 1998). It is also measured by computational complexity, execution time of algorithms, and the number of patterns which can be classified correctly despite any distortion (Lu, 1996; McGuire *et al.,* 2001). The experiments have been carried out to evaluate the performance of the proposed method. In this section, the new method was tested on thirty artificial datasets with different numbers of classes and complexities. Then the new method are described and compared with the neural network. The two methods were also tested on ten datasets out of the thirty so as to examine the performance of each classification method when they are trained on different training sets. To demonstrate the competitiveness of the proposed model, the GBC model was tested on the real-world data set which is publicly available from University of California at Irvine (UCI) data repository (Murphy and Aha, 1992).

Experimental Results for the Artificial Datasets

In the experiment for the multiclass problems, two-dimensional data samples were employed including the multiclass samples. Classes A, B, C, and D were generated by random functions. The procedure for the classification was repeated 10 times for all 30 data sets. Some of the results are shown in *Fig. 4*.

When testing a hypothesis on the artificial data is involved, there is a need to test the proposed method on at least 30 data sets. Thus, the above data sets with different numbers of data points were generated randomly using the random function in MATLAB (*randint()*) for this purpose and these data points were divided into 2, 3, and 4 groups, with different levels of complexity to test the proposed method in various conditions. The number of data points, the number of classes, and the distribution of data points change to cover all possible complicated conditions as we move from the first figure to the last one. The results show that no matter how complicated the distribution of the input training samples is, and no matter how many classes and how many samples there are in each class, the proposed model gives a promising performance.

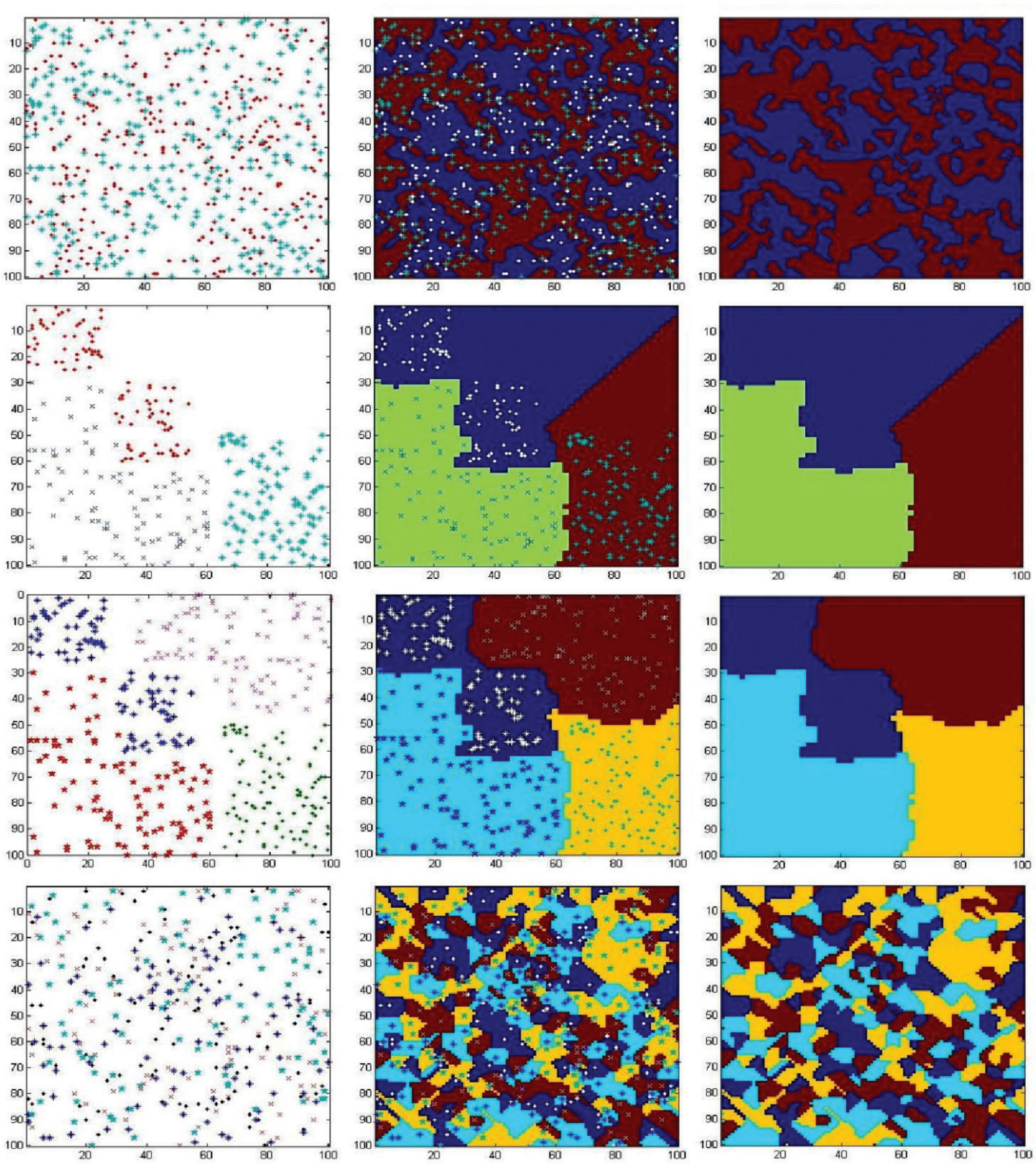

Fig. 4: Data sets (Left) applying the training procedure (Middle) training result (Right)

TABLE 1
Accuracy (%) and training time (s) for finding decision boundary with 2, 3, and 4 classes of artificial data

Artificial data set	No. of classes	Accuracy (%)	Training time (s)
1	2	100	0.28
2	2	100	0.28
3	2	100	0.28
4	2	100	0.28
5	2	100	0.28
6	2	100	0.28
7	2	100	0.28
8	2	100	0.28
9	3	100	0.30
10	3	100	0.30
11	4	100	0.31
12	4	100	0.31
13	4	100	0.31
14	4	100	0.31
15	4	100	0.31
16	4	100	0.31
17	4	100	0.31
18	4	100	0.31
19	4	100	0.31
20	4	100	0.31
21	4	100	0.31
22	4	100	0.31
23	4	100	0.31
24	4	100	0.31
25	4	100	0.31
26	4	100	0.31
27	4	100	0.31
28	4	100	0.31
29	4	100	0.31
30	4	100	0.31

As can be seen in all cases, the GBC method found the decision boundary without any misclassification in a very short time, and therefore, the performance is as impressive as it is expected to be. The accurate results obtained on the test data for the learning step was 100% (or without any misclassification), and the best result for the training time are reported in Table 1. The classification time is almost constant and it is very small for all the cases because a simple hash function is applied on a hash table.

As can be seen from the results, the proposed method has a high performance in the training stage, whereby it can detect any single datum correctly (without any misclassification) and it is also very efficient in terms of training and classification speed. Thus, the training time slightly increases in the case where the data set has more classes, although the amount of data samples is already fixed.

A Comparison between the Lazy and Eager Methods

A comparison was carried out with the neural network (for learning time and accuracy) and the nearest neighbour classifier (for generalization time) on 10 of the above data sets to examine the performance of the new method, and the results are also reported. The evaluation of the generalization accuracy for the real-world example is given in the subsequent section.

Backpropagation neural network in the learning phase

One of the powerful networks for the multiclass classifier is the *backpropagation network*. The backpropagation can train the multilayer feed-forward networks with differentiable transfer functions so as to perform function approximation, pattern association, and pattern classification (Ruck *et al.,* 1990) (note that other types of network can be trained as well, although the multilayer network is most commonly used). The term ‘backpropagation’ refers to the process by which derivatives of network error, with respect to the weights and biases of the netword, can be computed. This process can be used with a number of different optimization strategies. Input vectors and the corresponding target vectors are used to train a network so that it can be used to approximate a function, associate input vectors with specific output vectors, or classify input vectors in an appropriate way as defined by the user. Networks with biases, a sigmoid layer, and a linear output layer are capable of approximating any function with a finite number of discontinuities. Among the training functions, the Levenberg-Marquardt algorithm is the fastest training algorithm for the networks of moderate size; hence, the most extensively used is the Levenberg-Marquardt backpropagation algorithm, partly because it is much faster than any other algorithms. Therefore, the following code is used to create such a network:

```
net = newff ( minmax(p), [5,1], {‘tansig’,’purelin’}, ‘trainlm’);
```

The network was applied on the datasets. Using trial and false method during training, some network parameters (such as the number of hidden layers and performance goal) could be determined to achieve a proper performance. The performance goal parameter (indicating how many errors would be acceptable) determines when the training should be stopped. The training is ended when the performance function drops below the goal. If this parameter is set too high, the algorithm may become unstable. On the contrary, the algorithm will take a longer time to converge if this parameter is too small. Therefore, it is not practical to determine the optimal setting for the performance goal prior to any training, and for this reason, this parameter is set with the trial and false method to achieve an optimal performance. The performance also degrades as the performance goal parameter increases. After adjusting the parameter, the network was applied 10 times for each data set and the best results are presented in Table 2.

TABLE 2
Training time (s) and performance goal of the grid-based classifier and neural network

Data Set	Grid Base Classifier (GBC)		Neural Network (NN)	
	Training time (s)	Performance goal	Training time (s)	Performance goal
3	0.28	0	0.52	0.05
6	0.28	0	0.85	0.25
7	0.28	0	0.90	0.25
9	0.30	0	0.91	0.05
11	0.31	0	1.00	0.10
12	0.31	0	0.98	0.10
15	0.31	0	1.06	0.20
16	0.31	0	1.12	0.70
22	0.31	0	1.56	0.90
27	0.31	0	2.45	1.20

The results show when the new method was used, the training time seemed to slightly increase due to the increasing number of classes, but it was found to remain the same because of the rising level of complexity of input data. On the other hand, when the inputs were trained by the neural network method, the training time seemed to grow considerably because of the complicated input data and the rise in the number of classes, even when the effects of increasing the mean square error were ignored.

Condensed nearest neighbour in the generalization phase

The best-known of the lazy methods is the *condensed nearest neighbour,* which is a greedy algorithm that aims to minimize training error and complexity measured by the size of the stored subset (Hart, 1968; Devijier and Kittler, 1980). Both the time and space complexities of the non-parametric methods are proportional to the size of the training set, and condensing methods have been proposed to decrease the number of stored instances without degrading the performance.

To examine the performance of the proposed model in the generalization phase, it was compared with the condensed nearest neighbour as a candidate of the lazy methods. The fact that the evaluation of the test accuracy seems to be meaningless in the absence of the real-world dataset, checking the real-world problems is discussed in the next section. In this study, only the testing time evaluation of the above artificial datasets was taken into consideration. Table 3 shows the comparison in the speed of both the algorithms. For this purpose, the test vector containing 100 unknown samples was applied to the classifiers.

TABLE 3
Testing time (s) for the grid-based classifier and condensed nearest neighbour

Data set	Grid Base Classifier (GBC)	Condensed Nearest Neighbour
	Testing time (s)	Testing time (s)
3	0.002	1.42
6	0.002	1.57
7	0.002	1.8
9	0.002	1.61
11	0.002	1.69
12	0.002	1.67
15	0.002	1.71
16	0.002	1.75
22	0.002	2.3
27	0.002	2.92

As indicated earlier, there will be a grid after the learning stage, which each cell assigned to one of the classes within this grid. This grid or just a part of it is then kept (the boundary cells) in the hash table. Thus, a function is required in order to categorize the new sample into its right class when only its attribute values are given. In a quick procedure, this function takes these values as an argument and determines this sample to place it in a cell from the grid. Here, the proposed function is very simple and the values are simply divided by the grid size to determine the specific cell, so that the class label of the sample comes from the class label of this particular cell. Hence, the classification complexity for the proposed method is *O (1)* in which there is a single computation for the test pattern which is the same as the procedure of the eager methods. In contrast, the time complexity for the condensed nearest neighbour is *O(nN),* which includes computation of distances from the test pattern to all the condensed training patterns, in which each distance computation is *O(n)*. This is the reasonable cause of the considerable difference between the results presented in Table 3.

Experimental Results for the Real-world Datasets

In the following section, some real data sets are used to examine the classification performance of the new method.

The XOR problem

To find the non-linearly separable problems, going into complicated situations is not required. The well-known *Exclusive OR (XOR)* Boolean function is a typical example of such a problem. The Boolean functions (AND, OR, XOR) can be interpreted as the classification tasks. Indeed, based on the values of the input binary data $[x_1, x_2, ..., x_n]^T$, the output is either 0 or 1, and x is classified into one of the two classes, $A(1)$ or $B(0)$.

Fig. 5 (left) shows the position of the classes in space. It is apparent from this figure that no single straight line exists separates the two classes. Therefore, the major concern is to tackle the XOR problem in order to evaluate the new method. *Fig. 5* (right) shows how the GBC method finds decision boundary based on this problem.

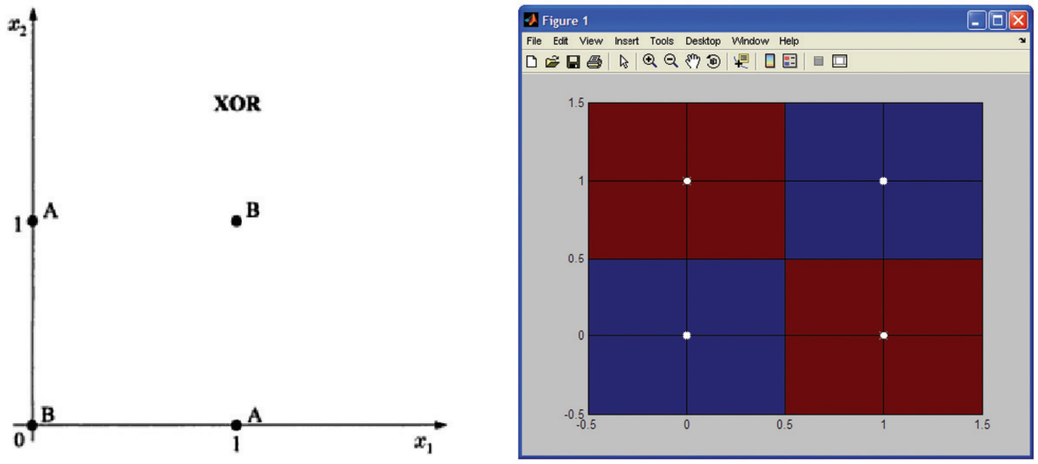

Fig. 5: (L) Classes A and B for the XOR problem; (R) Decision boundary by GBC for the XOR problem

The Haberman dataset

In this section, the experimental results of the proposed method on the *Haberman's Survival* data set from the UCI Machine Learning Repository are presented and discussed. The dataset contains cases from a study conducted at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer.

The experiments carried out in this study were conducted using the ten-fold cross-validation technique, which is a common technique used to evaluate the performance of a pattern classification algorithm [2]. The researchers conducted the ten-fold cross-validation in the following ways: each data set was randomly partitioned into ten groups, with each group having approximately the same number of points. Then, the GBC (and NN) model was run ten times, and one different group of data was held out to be used as the test set each time, whereas the other nine groups were used as the training set. The reported results are the average values for the ten runs.

The learning procedure used to find the decision boundary for the Haberman dataset is shown in *Fig. 6*.

The numerical results for the GBC, in comparison with BNN (backpropagation neural network), are given in the following tables. In Table 4, the results for learning time and accuracy are listed, while the time and accuracy for the testing phase are given in Table 5.

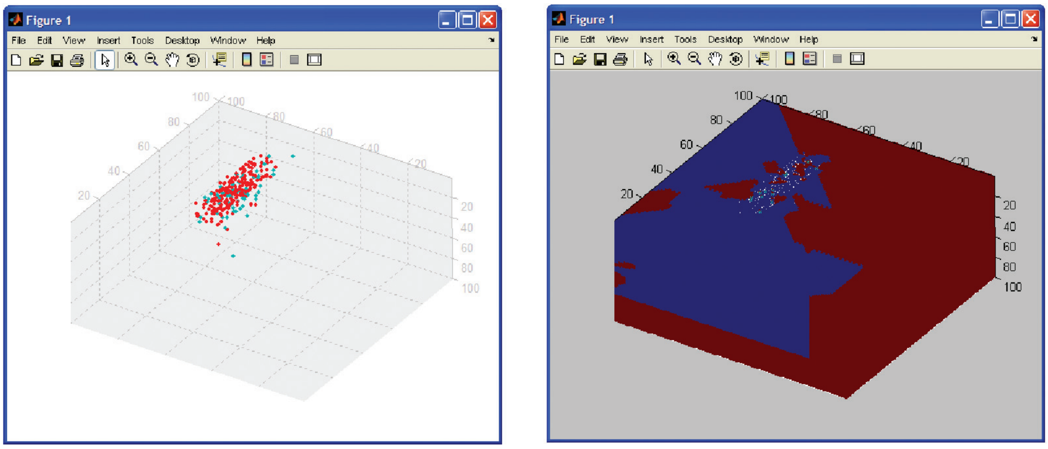

Fig. 6: The Haberman dataset is plotted in 3-dimensional space and decision boundary made by the GBC

TABLE 4
Training time (s) and Training accuracy (%) of the Grid-Based Classifier and Neural Network

	Grid Base Classifier (GBC)		Neural Network (NN)	
	Training time (s)	Accuracy (%)	Training time (s)	Accuracy (%)
1	0.51	98	1.31	96.5
2	0.51	98	1.40	96.3
3	0.51	98.4	1.15	97
4	0.51	98.7	1.52	96
5	0.51	98	1.51	98
6	0.51	98	1.45	98.5
7	0.51	98.4	1.06	96.7
8	0.51	98.4	1.12	97
9	0.51	98	1.56	98.5
10	0.51	98.4	1.45	97.5
Average	0.51	98.2	1.35	97.25

As can be seen from Table 4, the performance of these two models is comparable. From this test, it can therefore be concluded that the GBC model completes the training stage much faster than the neural network, and this was expected for the proposed model. In addition, the accuracy of the training part was also found to be better for the proposed model than the neural network, but it is not as perfect (100%) as the results derived from the artificial datasets, as there is 1.8% error which was caused by some overlapping data points. In this experiment, the data set is composed of 306 data points. Among these data points, which are categorized into two different classes, there are 6 repeated data points within the both classes (overlapped). When these elements are assigned to one of the classes, user will face an error on the other class and this is the cause of this insolvable error.

As can be seen in Table 5, the results for both methods are almost similar in terms of the overall generalization time and accuracy comparison. From this comparison, as expected, the results are in good agreement and the proposed method carries the advantages of the eager methods in the generalization phase.

TABLE 5
Testing time (ms) and Testing error (%) of the Grid-Based Classifier and Neural Network using the ten-fold cross-validation technique

	Grid Base Classifier (GBC)		Neural Network (NN)	
	Testing time (ms)	Test error (%)	Testing time (ms)	Test error (%)
1	0.6	10	0.8	31
2	0.6	30	0.7	25
3	0.6	35	0.7	29.3
4	0.6	39.9	0.6	32
5	0.6	23.6	0.7	25
6	0.6	36.2	0.8	30.7
7	0.6	30	0.7	35
8	0.6	31.7	0.7	30
9	0.6	24	0.7	30.5
10	0.6	15.3	0.8	26
Average	0.6	27.6	0.7	29.4

However, the results for the testing accuracy are not so impressive because of the overlapping data and the noisy nature of this data set. On the contrary, it is highly comparable with the results previously reported for the other methods. Table 6 reveals the performance of the statistical method known as the *Support Vector Machine (SVM)* and some ensemble methods (multi classifier methods) such as *Bacing*, *Bagging*, *Adaboost* and *Arcing* on the Haberman data set (Zhang and Street, 2008). The error estimates listed are over ten-fold cross-validation runs.

TABLE 6
The best generalization performance reported for the Haberman dataset

	SVM	Bagging	Adaboost	Arcing	Bacing
Haberman	28.10	27.77	25.81	25.89	25.50

In comparison with these results, the proposed method has been shown to give a better performance as compared to the other statistical methods such as the neural network and support vector machine; this is what the researchers were looking for the experiments conducted in this study. In addition, the proposed model also shows highly acceptable results in terms of testing accuracy as compared to the ensemble methods.

CONCLUSION

In the learning phase, the model was given a set of training samples, in which each sample was represented by a set of attributes. It is better to perform compression in that by fitting a rule to the data, and the explanation obtained is that it is simpler than the data, as it requires less memory for storing and less computation for processing. Therefore, the task is to find a decision boundary which can be used to categorize a sample into its right class when only its attribute values are given. The results indicate that the GBC model can find the decision boundary in a very short time as compared to the other statistical methods, and that the accuracy of this phase was 100%, except in the case of the overlapping data which have caused insignificant error. Based on the results, it is therefore concluded that the outcomes of this stage are independent (or are not determined by) of the size of the data sets, the number of classes and the distribution of data.

In the generalization phase, there is always a certain non-zero probability that the decision for a new sample is wrong when a decision boundary is made from a finite number of the training samples. Therefore, it is desirable to determine the percentage of the error when making a decision for the new instances. The results obtained for the real-world data set show that the proposed model has achieved a better performance in time and accuracy as compared to the other statistical models.

REFERENCES

Devijver, P. A. and Kittler, J. (1980). On the edited nearest neighbor rule. *Proceeding of the fifth International Conference on Pattern Recognition* (pp. 72-88).

Duda, R.O., Hart, P.E. and Stork, D.G. (2001). *Pattern Classification* (2nd ed.). San Diego: Wiley-Interscience.

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition*. Academic Press, Inc.

Guobin, O. and Murphey, Y.L. (2007). Multi-class pattern classification using neural networks. *Pattern Recognition, 40*, 4-18.

Hart, P. E. (1968). The condensed nearest neighbor rule. *IEEE Transaction on Information Theory*, *14*(5), 515-516.

Ho, T.K. (2004). Promises and challenges in automatic pattern recognition. *ASP Conference Series*, *314*, 239.

Ho, T.K. and Basu, M. (2002). Complexity measures of supervised classification Problems. *IEEE Transaction on Pattern Analysis and Machine Intelligence, 24*(3), 289-300.

Lu, Y. (1996). Knowledge integration in a multiple classifier system. *Applied Intelligence*, *6*(2),75-86.

McGuire, P. and Eleuterio, G. M. T. D. (2001). Eigenpaxels and a neural network approach to image classification. *IEEE Transaction on Neural Network*, *12*(3), 625-635.

Murphy, P. M. and Aha, D. W. (1992). *UCI repository of machine learning databases*. Retrieved from http://www.ics.usi.edu/~mlearn/MLRepository.html.

Ruck, D. W., Rogers, S. K., Kabirisky, M., Oxley, M. E. and Sutter, B. W. (1990). The multilayer perceptron as an approximation to a bayes optimal discriminant function. *IEEE Transaction on Neural Network, 1*(4), 296-298.

Sarlashkar, M. N., Bodruzzaman, M. and Malkani, M. J. (1998). Feature extraction using wavelet transform for neural network based image classification. *Proceeding of the Thirtieth Southeastern Symposium on System Theory* (pp. 412-416).

Veloso, J. and Meira, W. (2005). Eager, lazy and hybrid algorithms for multi-criteria associative classification. In *Proceeding of the Data Mining Algorithms Workshop,* Uberlandia, MG.

Zhang, Y. and Street, W. N. (2008). Bagging with adaptive costs. *IEEE Transaction on Knowledge and Data Engineering*, *20*(5).