Neuro-Fuzzy Approaches to Decision Making: An Application to Check Authorization from Incomplete Information

G. J. Vachtsevanos, S. S. Kim, J. R. Echauz, and V. K. Ramani

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0250

Abstract— **This paper describes the application of several neuro-fuzzy paradigms, such as a multilayer perceptron, a polynomial neural network, and a fuzzy decision model to the problem of check approval from incomplete data. A simple benchmark case is established as a performance metric to compare the various non-linear strategies. An overall improvement of at least 10% was obtained in each of these cases.**

I. Introduction

By the turn of the century, almost half of all consumer payments are expected to be made by checks [1]. This trend is forcing more and more merchants to rely on check-guarantee and check-authorization services provided by specialized companies that assist them to manage the associated increased risk. These companies maintain large databases with information such as customers' names, fraudulent driver's licenses, and Social Security numbers. Stores with connections to an on-line system, can have checks approved or rejected in a matter of seconds.

Despite the steady increase in the availability of these services, the number of check-writing individuals represented in current databases is still a very small fraction of the whole population. In many cases, a new customer shows up for whom there is no information available. Is there any decision scheme, other than pure random guessing, that can reduce the odds for the merchant of accepting a bad check? This problem calls for systems that can generalize from incomplete information, and thus predict the credit worthiness of the new customer. Nonlinear learning networks are prime candidates for this task. We describe the application of a multilayer perceptron, polynomial neural network, and a fuzzy decision model to address the problem of check approval from incomplete data.

II. Problem Description

Consider two sets of data, for training and testing purposes, respectively, typical of those available from a check authorization company. The data set consists of four input variables and an output variable:

x_1 : Day of the week (1=Monday, ... , 7=Sunday)
x_2 : Age of the person
x_3 : Check Number
x_4 : Amount of the data
y : 1 = Accept check, 0 = Reject check

Each set consists of 1950 data points. Out of the 3900 data points only 74 data points have the complete vector of information. Any missing information is coded as zeros. The output of the network is denoted as B for a bounced check and G for a good check, while the predicted output will be represented by $\widehat{B}$ and $\widehat{G}$ respectively.

III. METHODS AND RESULTS

A. *Minimum Distance Classifier*

With two equiprobable classes, the classification accuracy of the nonlinear learning networks outlined in this paper can be neither worse than 50%[1] nor better than 100%. But how can we tell if the added complexity of these networks is justified? How "good" is

[1]Less than 50% is actually "better" by contradicting the output decision.

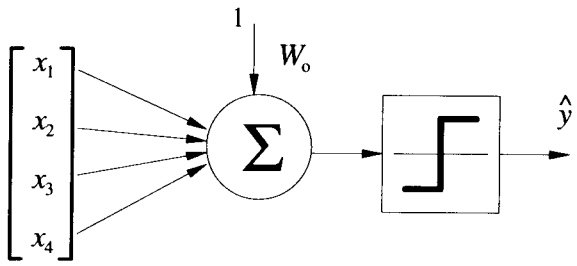

Fig. 1. Thresholded linear discriminant.

their performance relative to simpler, well-established classification methods?

We will establish a benchmark measure of performance defined by a minimum distance classifier (MDC). The decision rule adopted by any such system is simply: "assign $\mathbf{x}$ to the class whose mean feature vector is closest (in the case of a Euclidean distance) to $\mathbf{x}$." Thus a dichotomous decision is given by

$$\begin{cases} \|\mathbf{x}-\overline{\mathbf{x}}_1\| < \|\mathbf{x}-\overline{\mathbf{x}}_2\| \Rightarrow \mathbf{x} \in C_1 \\ \text{else} \quad \mathbf{x} \in C_2 \ . \end{cases} \tag{1}$$

The rule reduces to a bank of linear discriminants followed by a maximum selector [2]. In fact, for a two-class problem in particular, the decision boundary needs only one hyperplanar separatrix. Thus, the classifier further reduces to a single thresholded linear discriminant as shown in Fig. 1. The weight vector $[w_0 \cdots w_4]^T$ is given by

$$\mathbf{w} = \begin{bmatrix} \frac{1}{2}\left(\|\overline{\mathbf{x}}_2\|^2 - \|\overline{\mathbf{x}}_1\|^2\right) \\ \overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2 \end{bmatrix}. \tag{2}$$

This decision model trains virtually instantaneously as its parameters are determined from estimates of the mean feature vectors. It is a reasonable starting point in most cases, and under independent equal-variance gaussianity of the features, it is the optimal (maximum-likelihood) model.

The MDC was trained on the raw training set, and the overall classification accuracy was 56.4% on the test set. This performance is only slightly better than the "constant-output classifier" or the "coin-flip classifier," whose expected accuracy is 50% assuming equal a priori probabilities for each class. An obvious factor that cannot be directly handled by this classifier is the use of zeros in places of missing data. The linear decision boundary implemented by the dichotomous MDC cannot "interpret" zeros as special cases, but rather takes them as customers of zero age writing checks numbered 000 – clearly an artifact.

	B	G
$\widehat{B}$	532 (53.8%)	245 (25.5%)
$\widehat{G}$	456 (46.2%)	717 (74.5%)

Overall correct = 64.1%

(a)

	B	G
$\widehat{B}$	\$61,468 (29.5%)	\$31,416 (14.0%)
$\widehat{G}$	\$147,195 (70.5%)	\$192,849 (86.0%)

Overall correct = 58.7%

(b)

Table 1. (a) Performance of MDC (b) Performance of MDC in dollars

A simple preprocessing of the raw database can help alleviate the anomaly caused by missing data. One approach is to replace each zero with the mode or the mean of the corresponding input variable. The latter can be easily estimated from presumably representative nonzero data in an unbiased and consistent manner using the arithmetic average. In this case, the unknown values are replaced with guesses that produce near-minimal mean squared errors. Once these guesses are computed from the training set, they become part of the classifier and are applied, without any change, to the test set.

The above scheme was employed on the MDC and the results are summarized in the confusion matrix on Table 1(a). The columns represent the true output classes whereas the rows represent the model assessments based on input information. Each entry contains the number of cases and the percentage relative to its column. The overall classifier accuracy (trace of the matrix over sum of its entries) increased by almost 8%, and we take this new figure as a worst case bound for the more involved methods described next. It is interesting to note that the exact same performance was obtained upon deletion of the first input variable), indicating that the day of the week is irrelevant at least in an MDC sense.

In the context of check authorization systems, it is

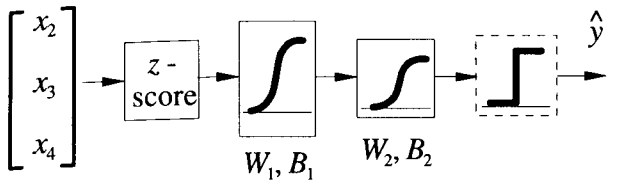

Fig. 2. Two-layer perception.

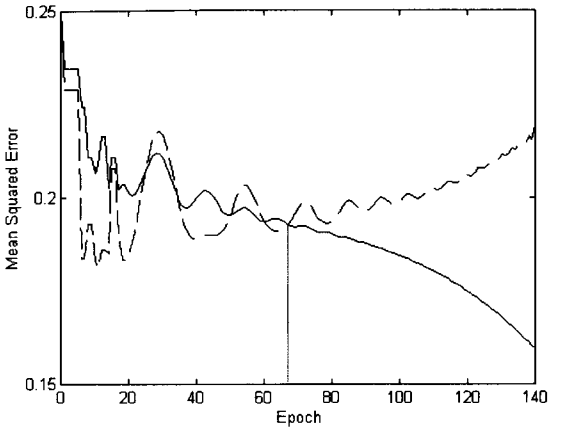

Fig. 3. Training with cross validation.

also significant to monitor the confusion matrix expressed in dollar terms as shown in Table 1(b). This performance measure gives more weight to larger check amounts, recognizing the proportionately larger impact of fraud in such transactions.

B. *Multilayer Perceptron*

The results obtained using the MDC served to guide in the design of a (3,100,1) two-layer perceptron trained with backpropagation. The choice of two layers, with at least the hidden layer being nonlinear, is adequate in many problems and gives the network the capability of generating arbitrary decision boundaries [3]. The three inputs to the network are the ones previously found most essential. The number of neurons follows the rough rule-of-thumb of being one tenth the number of training exemplars, which is roughly 1000 because half of the "training" set was actually used for cross-validation. The single output is a "soft" decision in the interval [0,1] during training, but is hard-limited in its final implementation, returning only values in {0,1}. The network architecture is shown in Fig. 2 in vector/matrix notation. The activation functions are logistic sigmoids and the hard-limiter has a threshold of 0.5.

	B	G
$\hat{B}$	704 (71.3%)	274 (28.5%)
$\hat{G}$	284 (28.7%)	688 (71.5%)

Overall correct = 71.4%

(a)

	B	G
$\hat{B}$	\$136,296 (65.3%)	\$45,185 (20.1%)
$\hat{G}$	\$72,367 (34.7%)	\$179,080 (79.9%)

Overall correct = 72.8%

(b)

Table 2. (a) Performance of MLP (b) Performance of MLP in dollars

Each input variable has different meanings and incommensurate ranges of values. A standardization of the inputs in terms of z-scores does not change the final performance of a trained network, but does tend to make the learning procedure faster. The mean and standard deviations are estimated from the nonzero values in the training set.

The network was trained until the mean squared error on the validation set started its upward trend – the point at which generalization performance starts to degrade. The optimal stopping point occurred after 67 passes of the training data as shown in Fig. 3.

The performance of this neural network on the test set is summarized in Tables 2(a) and (b) in terms of number of cases and dollar amount.

C. *PNN Algorithm*

The inherent property of the Polynomial Neural Network (PNN) or the Group Method of Data Handling (GMDH) is to model complex systems using simple building blocks [4]. This led us to implement the PNN strategy on the check validation problem.

As there was a lot of uncertainty associated with the data available, both for training and validation

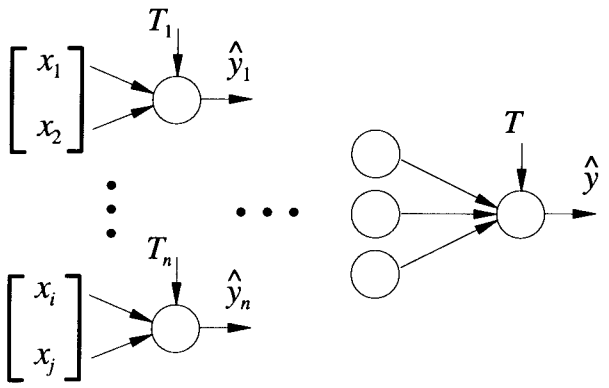

Fig. 4. PNN model.

purposes, two different strategies were implemented. In the first case, to establish a benchmark measure of performance, the PNN methodology was implemented on the testing and validation data sets [5]. In the second case, the data was preprocessed. Preprocessing of the data was considered a necessary step because of the uncertainty (lack of information) associated with the data. Preprocessing was further necessitated by the fact that more than 98% of the training and validation data had at least one input variable missing.

It must be noted here that the computational complexity of the simulation is the same in both cases, as the PNN methodology was implemented on the same number of data points.

In a PNN technique, a simple function is combined at each node of a neural network to obtain a more complex function. This function represents the model for the given set of input-output data. A simple function of the following form is used to combine two inputs at each node of the neural network.

$$y = A + Bx_i + Cx_j + Dx_i^2 + Ex_j^2 + Fx_ix_j \ , \quad (3)$$

where y is the output and x_i and x_j are the two inputs. As shown in Fig. 4, the outputs obtained from each of these nodes are then combined to obtain a higher degree polynomial so that the best model may be achieved which represents the input-output data.

The degree of the polynomial increases by two at each layer of the neural network. The neural network was restricted to two layers as it had to be trained

	B	G
$\hat{B}$	659 (66.7%)	456 (47.4%)
$\hat{G}$	329 (33.3%)	506 (52.6%)

Overall correct = 59.7%

(a)

	B	G
$\hat{B}$	\$96,893 (46.4%)	\$63,922 (28.5%)
$\hat{G}$	\$111,770 (53.6%)	\$160,343 (71.5%)

Overall correct = 59.4%

(b)

Table 3. (a) Performance of PNN (b) Performance of PNN in dollars

for a dichotomous decision only. The best models at each layer were obtained by using a thresholding (T_i) error value. The output of these best models were then combined at the next layer to obtain a higher degree polynomial. The best output model, in terms of, the minimum predicted squared error, was used as the model for the system.

• Case I: PNN Implementation

A straightforward implementation of the PNN on the training data was carried out to obtain a model for the data. Zeros were used in place of missing information. This was done to obtain a worst case measure for the neural net model. Some other methods, like averaging, etc. could have been adopted to fill the missing information, but that would have skewed the results and would not have allowed us to use it as a benchmark.

A model is constructed based on the training data. This model is then tested using the validation data. The results obtained by the implementation of the PNN strategy are summarized in the confusion matrix in Table 3(a).

The overall predictability measure was 59.7% and this is taken to be the worst case bound for the PNN methodology. In dollar terms the confusion matrix obtained was as shown in Table 3(b). Thus 59.4% signifies the dollar amount that was correctly predicted in the first case.

• CASE II: PNN Implementation after pre- processing

In this case, the data was pre-processed. The entire

Model Number	Input missing	No. of Data points
M I	None	37
M II	Age	736
M III	Check #	918
M IV	Age & Check #	278

Table 4. Model Clusters

Model Number	Overall Correct Data (%)	Overall correct Amount (%)
M I	75.68	82.48
M II	72.87	75.79
M III	73.37	73.29
M IV	70.50	65.10

Table 5. Performance of PNN with preprocessing (in clusters)

training and validation set was divided into clusters based on the information available. If all the input variable values were available then all such data was put in a cluster, if variable 2 was not available then the entire such data was put in a separate cluster. In the check validation example four models were obtained. The break up for the training data was as shown in Table 4.

The PNN methodology was then implemented on each cluster separately. The results were then tested on the preprocessed tested data. If the test data contains elements in a cluster which was not present in the training set, then no prediction can be made. The implication is that there is a lack of training data for such a set to predict the output.

The confusion matrix, both for the data points and amount of dollars, were obtained for all four clusters. The Overall correct percentages for the data and dollar amount for each model cluster are given in Table 5.

The overall correct percentage for the entire data was also obtained by combining the values calculated from each model cluster, and is shown in Table 6(a) and the dollar values are shown in Table 6(b).

A marked improvement in the performance of the Polynomial Neural Network strategy was observed, when it was combined with a preprocessing stage. The overall correct percentage for the prediction of the output was increased by 13.1% while the overall correct percentage in dollar terms was increased by 14.3%.

	B	G
$\hat{B}$	663 (67.1%)	205 (21.3%)
$\hat{G}$	325 (32.9%)	757 (78.7%)

Overall correct = 72.8%

(a)

	B	G
$\hat{B}$	\$129,533 (62.1%)	\$34,617 (15.4%)
$\hat{G}$	\$79,130 (37.9%)	\$189,648 (84.6%)

Overall correct = 73.7%

(b)

Table 6. (a) Performance of PNN with preprocessing (b) Performance of PNN with preprocessing in dollars

Thus, the preprocessing improves the performance significantly without increasing the computational effort. The only drawback of the second case is that it will not be able to predict an output if that particular model is not available in the training data set.

D. Fuzzy Decision Model

An important concern for decision-making problems is how to devise a method that would aid judgments from a strategic point of view by means of summarized information. In this section, a fuzzy model is used to assist the human decision process.

Fuzzy model identification based on fuzzy implications and fuzzy reasoning [6] is one of the most important aspects of fuzzy system theory [7]. In this paper the membership function of a fuzzy set A is represented by $\mu_A(x)$, $x \in A$ and the fuzzy sets are associated with triangular shaped membership functions. The structure of a fuzzy decision model based upon input-output information is defined as a finite set of linguistic relations or rules, $\{R^i;\ i = 1, ..., m\}$, which together form an algorithm

$$R^i: \quad \text{If } x_1(k) \text{ is } A_1^i \text{ and } \cdots \text{ and } x_n(k) \text{ is } A_n^i \quad \text{Then } y(k) \text{ is } B^i\ , \tag{4}$$

where $x_1, \ldots, x_n$ are inputs, $A_1^i, \ldots, A_n^i$ are the fuzzy sets in $X_1, \ldots, X_n$ and B^i is the fuzzy set in Y with appropriate membership functions. X_j $(j = 1, \ldots, n)$ and Y are the universes of discourse of x_j and y, respectively. After constructing the fuzzy model using linguistic rules, the compositional rule of inference [8]

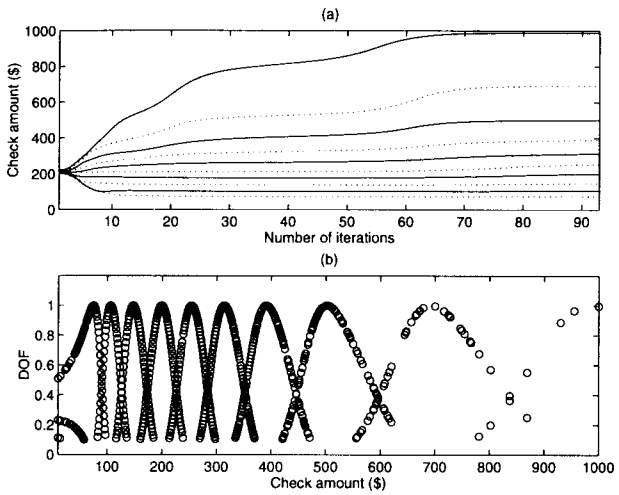

Fig. 5. (a) Convergence of the centers of the 10 clusters in Check amount variable. (b) The final result of the FCM (c=10, m=2).

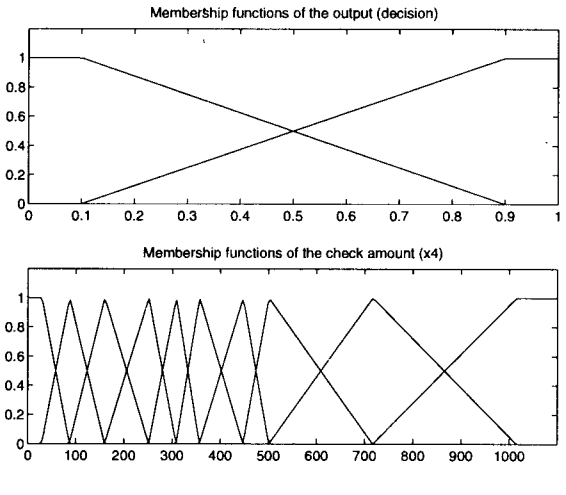

Fig. 6. Membership functions of the decision (two fuzzy sets) and check amount (ten fuzzy sets).

is called upon to infer the output fuzzy variable from given input information. The output fuzzy set is calculated from the expression:

$$\mu_{B'}(y) = \max\min[\{\mu_{A_1^i}(x_1^0), \cdots, \mu_{A_n^i}(x_n^0)\}, \mu_{B^i}(y)], \quad y \in Y, \tag{5}$$

where x_j^0, $j = 1, \cdots, n$, is a given input singleton. The centroid defuzzification method is used to arrive at a crisp output value , y^0 [9]:

$$y^0 = \frac{\int y \mu_{B'(y)}}{\int \mu_{B'(y)}}, \; y \in Y \; . \tag{6}$$

We construct membership functions from the collected data set using a fuzzy c-means (FCM) clustering method [10]. A fuzzy clustering of X (crisp data set) into n clusters is a process of assigning a grade of membership for each element to every cluster. The fuzzy clustering problem is formulated as:

$$\text{minimize} \quad J_m(U,V) = \sum_{k=1}^{n}\sum_{i=1}^{c}(u_{ik})^2 (d_{ik})^2 \tag{7}$$

subject to

$$\sum_{i=1}^{c} u_{ik} = 1, \; u_{ik} \geq 0, \; 1 \leq i \leq c, \; 1 \leq k \leq n \; , \tag{8}$$

where n is the number of data points to be clustered, c is the number of clusters, m is a scalar ($m > 1$), $d_{ik} = \|\mathbf{x}_k - \mathbf{v}_i\|$, the Euclidean distance between each data point $\mathbf{x}_k$ in X, $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, and the center of the cluster, $\mathbf{v}_i$; u_{ik} is the membership grade of the kth data in the cluster i. If $m = 1$, it becomes a hard clustering problem, whereas if $m > 1$, the clusters become fuzzy. More fuzziness can be obtained by increasing the value of m. The center $\mathbf{v}_i$ of the fuzzy cluster is calculated by

$$\mathbf{v}_i = \sum_{k=1}^{n}(u_{ik})^m \mathbf{x}_k / \sum_{k=1}^{n}(u_{ik})^m, \quad 1 \leq i \leq c \; . \tag{9}$$

Four inputs ($x_1, \cdots, x_4$: day of week, age, check number and check amount) and one output (y: decision) variable are defined in the data set. The day of week (x_1) is a crisp value that can not be fuzzified. An FCM clustering algorithm is applied to the training data, except x_1, resulting in ten clusters. Fig. 5(a) shows the convergence of the centers $\mathbf{v}_i$, $i = 1, \cdots, 10$, in the check amount (x_4) variable, and Fig. 5(b) shows the membership grades from the training data, x_4. Thus, ten linguistic terms for the input variables (x_2, x_3, and x_4) are derived as shown in Fig. 6. Possible values for the output are either 1, (the check is approved) or 0 (the check is turned down) and we assign two linguistic terms for the decision (Fig. 6). If

	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}
A_1	1	1	1	1	0	1	0	0	0	1
A_2	0	0	0	0	1	1	0	1	1	1
A_3	0	0	1	0	1	0	0	1	0	1
A_4	1	0	1	1	1	0	1	0	1	0
A_5	0	0	0	1	1	0	0	1	0	1
A_6	1	1	1	0	0	1	0	0	0	1
A_7	0	0	0	0	0	1	1	0	1	0
A_8	0	1	0	0	0	0	0	0	1	1
A_9	0	0	0	1	1	1	1	0	1	1
A_{10}	1	0	0	1	0	1	1	0	0	1

Table 7. Rule base between check amount and check number with fixed day and age (N_i: Check number, A_i: Check amount)

	B	G
$\widehat{B}$	741 (77.0%)	273 (27.6%)
$\widehat{G}$	221 (23.0%)	715 (72.4%)

Overall correct = 74.7%

(a)

	B	G
$\widehat{B}$	\$174,576 (77.8%)	\$57,932 (27.8%)
$\widehat{G}$	\$49,689 (22.2%)	\$150,731 (72.2%)

Overall correct = 75.1%

(b)

Table 8. (a) Performance of FDM (b) Performance of FDM in dollars

the output of the fuzzy decision is greater than 0.5, the final decision is 'approved', and if less than 0.5, the action is 'turned down' with unbiased threshold.

The rule base for the fuzzy decision model contains rules of the form as in (4). Table 7 shows 100 rules for the check amount and check number in the case of fixed day (crisp variable) and age (fuzzy variable). The training data set includes a lot of missing information that is represented as 0, but it can be easily handled after assigning membership functions to it. In the case of an incomplete rule base, the empty rule cells are filled by observing the underlying pattern [11]. The complete rule set consists of 7,000 rules (7×10^3) and includes the day of the week.

Fig. 7 shows the final result on the testing data set (1950). The shaded region represents incorrect decisions. The final analysis based upon the testing data set is shown in Table 8. The overall correct decision in terms of number of cases is 74.7% and in dollar amount

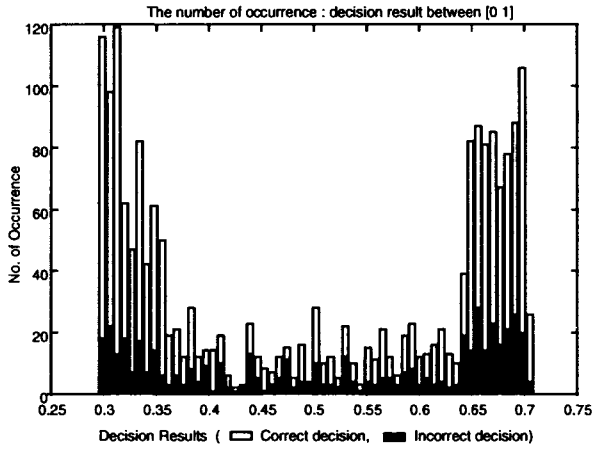

Fig. 7. Correct and incorrect decision result from fuzzy decision model (FDM).

75.1%.

IV. Discussion and Conclusions

Based on the overall classification accuracy, we may conclude that linear minimum distance classifiers are not capable of directly handling missing data, thus yielding only slightly better performance than pure random guessing. However, preprocessing of the database by replacing the zeros with their estimated mean values improves the overall performance by almost 8%.

The two-layer perceptron is able to bring the overall classification accuracy to a little more than 70% and has much better balance (diagonal elements of the confusion tables are closer to each other) than the benchmark method.

The polynomial neural network with preprocessing of the data was able to improve the model accuracy by nearly 14% over the benchmark problem. This was obtained without any additional computational effort. This is a reasonably fast process, as it only requires a regression of a quadratic equation at each node.

In the fuzzy decision model case, the result in terms of the number of cases and dollar amount, is slightly better than other methods presented in this paper. We can conclude from these results, that a fuzzy system, as a universal approximator, is efficient in handling

imprecise information.

Considering the importance of achieving maximum recognition rates in the check approval application, it is seen that the marked improvement in performance from all three nonlinear decision makers in this study justifies the significant additional effort involved in defining and training such systems, and warrants further investigation.

References

[1] L. Sloane, "Checking out checks: The methods stores use," *The New York Times*, Col. 1, vol. 140, pp. 44(L), Nov. 24, 1990.

[2] N. Ahmed and K.R. Rao, *Orthogonal Transforms for Digital Signal Processing.* New York: Springer-Verlag, 1975.

[3] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359-366, 1989.

[4] S.J. Farlow, *The GMDH Algorithm: Self Organizing Methods in Modeling. GMDH type Algorithms*, New York: Marcel Dekker. 1984.

[5] R. L. Barron, *et. al.*, "Applications of Polynomial Neural Networks to FDIE and Reconfigurable Flight Control," *IEE Proc. of National Aerospace and Electronics Conf.* , vol. 2, pp. 507-519. 1990.

[6] Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. Syst. Man and Cybern.*, vol. SMC-3, no. 1, pp. 28-44, Jan. 1973.

[7] G. J. Klir and T. A. Folger, *Fuzzy Sets, Uncertainty and Information*, New Jersey: Prentice Hall, 1988.

[8] M. Mizumoto, "Fuzzy controls under various fuzzy reasoning methods," *Information Science*, Vol. 45, pp. 129-151, 1988.

[9] C. C. Lee. "Fuzzy logic control systems: fuzzy logic controller, Parts I and II," *IEEE Trans. Syst. Man Cybern.*, vol.20, no.2, pp. 404-435, 1990.

[10] C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, New York, Plenum Press, 1981.

[11] R. M. Tong, "Synthesis of fuzzy models for industrial processes: Some recent results," *Int. J. General Systems*, vol. 4, pp. 143-162, 1978.