IMA Journal of Mathematics Applied in Business and Industry (2000) **11**, 111–125

Credit scoring using neural and evolutionary techniques

M. B. YOBAS[†], J. N. CROOK[‡] AND P. ROSS[†]

Credit Research Centre, Department of Business Studies, University of Edinburgh, William Robertson Building, 50 George Square, Edinburgh EH8 9JY, UK

[Received 23 December 1997 and in revised form 29 March 1999]

This paper compares the predictive performance of linear discriminant analysis, neural networks, genetic algorithms and decision trees in distinguishing between good and slow payers of bank credit card accounts. Predictive models were built using the evolutionary techniques and the results compared with those gained from the discriminant analysis model published in Crook *et al.* (1992), *The Service Industries Journal* **12** which uses the same dataset. A range of parameters under the control of the investigator was investigated. We found that the predictive performance of linear discriminant analysis was superior to that of the other three techniques. This is consistent with some studies but inconsistent with others.

1. Introduction

The aim of this paper is to compare the predictive ability of linear discriminant analysis (LDA), neural networks (NNs), genetic algorithms (GAs) and decision trees (DTs) in the classification of credit card applicants. All four techniques have been applied to the same dataset and the results of the three evolutionary techniques are compared with the LDA results published in Crook *et al.* (1992).

Whilst many credit scoring agencies have experimented with the latter three techniques, few offer products which use them (Desai *et al.*, 1997) and the same applies to credit granting organisations. Because of the commercial value of particular models there are very few publicly available comparisons between these techniques in the credit scoring context. The published literature suggests mixed results: some find that traditional techniques correctly classify a greater proportion of applications than AI techniques, others the reverse. For example, using data from three credit unions Desai *et al.* (1996) found that when classifying accepted loans into goods and bads neural networks correctly classified a greater percentage of both the total samples, and of the goods, than either linear discriminant analysis or logistic regression. But when *generic* models were estimated neural nets were superior only in the prediction of the bads. King *et al.* (1994) compared a large number of algorithms including linear discriminant analysis, neural networks and decision trees (but not genetic algorithms) and found that linear discriminant analysis predicted less well than various types of decision trees and a smooth additive multiple regression technique, but better than neural networks. However their data oversampled the 'bad' payers and it is known that such oversampling can affect the performance of

[†]Department of Artificial Intelligence, The University of Edinburgh
[‡]Credit Research Centre, The University of Edinburgh

© Oxford University Press 2000

certain techniques. On the other hand, Desai *et al.* (1997) applying the same dataset as in their 1996 paper, to a 3 group classification (goods, poors and bads) and including genetic algorithms into the contest, found that logistic regression was superior to the other methods except for the classification of the poors, where neural networks were best. In classifying the total sample they found that the performance of linear discriminant analysis was almost identical to that of neural networks and slightly better than that of genetic algorithms.

One conclusion which is often expressed by those assessing the accuracy of neural networks at least is that their performance depends on the values of certain parameters which are under the control of the investigor. Whilst there are certain heuristics to aid the choice of these values, the fact that neural networks and genetic algorithms allow for certain nonlinearities which is not possible in the traditional techniques, does not always seem to be reflected in a superior performance. There is therefore a need for further experimentation to support, or otherwise, the small number of findings so far published. In addition no published study has compared all four techniques: traditional, neural networks, genetic algorithms and decision trees using the same credit applicant datasets and using a realistic division of cases between the 'good' and 'bad' groups. It is the aim of this paper to do this.

The structure of this paper is as follows. In the following section we briefly outline the four classification techniques which are to be compared. In section three we describe the data used. In the fourth section we present some implementation details and in section five the results. The final section concludes.

2. Four classification techniques

2.1 *Linear discriminant analysis*

The principles of LDA have been rehearsed elsewhere (Choi (1986); Tatsuoka (1970); Eisenbeis & Avery (1972); Fisher (1936)). Essentially, suppose for each application we have a $(p \times 1)$ column vector of characteristics, $\underline{X}$, and suppose we wish to divide the total population of applications into two groups, J_1 and J_2. If the allocation rule chosen is to minimise the expected costs of misclassifying a case into the group of which it is not a member then we wish to minimise:

$$L = P_1 L_1 \int_{J_2} f_1(\underline{X})\,\underline{dx} + P_2 L_2 \int_{J_1} f_2(\underline{X})\,\underline{dx}$$

where P_1, (P_2) is the prior probability that a case is a member of set 1(2)

L_1, (L_2) is the cost of misclassifying a member of set 1(2) into set 2(1)

$f_1(\underline{X})$, $f_2(\underline{X})$ is the probability density function that a vector $\underline{X}$ will occur given that a case is a member of set 1(2).

If it is assumed that $L_1 = L_2 = 1$, that the values of $\underline{X}$ are normally distributed with the same covariance matrix, C, in both sets, then it can be shown (Thomas, 1997) that it is optimal to classify a case into set 1 if

$$(\underline{X} - 0{\cdot}5(\underline{m}_1 + \underline{m}_2))^{\top} C^{-1}(\underline{m}_1 - \underline{m}_2) > \log\left(\frac{P_2}{P_1}\right) \tag{1}$$

where $m_1(m_2)$ is the $(p \times 1)$ vector of mean values of the characteristics for set 1(2).

Fisher (1936) deduced the same linear rule as equation (1) but in a different way. He argued that the greatest difference between the groups occurred when the between groups sums of squares divided by the within groups sums of squares was maximised. This may be represented as:

$$\lambda = \frac{W^{\top} B W}{W^{\top} A W} \tag{2}$$

where W is a column vector of weights, B is the between group sums of squares and cross products matrix and A is the within groups sums of squares and cross products matrix (Tatsuoka, 1970). Differentiating equation (2) with respect to each weight separately and equating each resulting equation to zero gives, eventually, in the case of two groups a single set of weights for which λ is maximised (Tatsuoka, 1970).

2.2 *Neural networks*

A multilayer perceptron feed forward neural network consists of an input layer, a number of hidden layers and an output layer. In the credit scoring context the input layer consists of p characteristics, X_j, $j = 1, \ldots, p$. A weight is applied to each characteristic and the resulting products are summed to give

$$U_k^{[1]} = \sum_{j=1}^{p} W_{kj}^{[1]} X_j \tag{3}$$

where $W_{kj}^{[1]}$ is the weight connecting variable j and neuron k in layer 1. Each $U_k^{[1]}$ value is transformed non linearly to give an output of neuron k:

$$y_k^{[1]} = f(U_k^{[1]}). \tag{4}$$

The $y_k^{[1]}$ values are then weighted by a new set of weights and the procedures repeated to yield a new layer of neurons in layer 2. In the credit classification problem, with two groups, the final layer would consist of a single neuron and a case would be classified according to whether $y_n \lessgtr c$ where c is the critical value of O_n such that cases with $y_n > c$ are allocated into one group and cases where $y_n < c$ are allocated into the second group.

In general we may write equations (3) and (4) as

$$U_k^{[s]} = \sum_{j}^{z^{[s-1]}} W_{kj}^{[s]} y_j^{[s-1]}$$

and

$$y_k^{[s]} = f(U_k^{[s]}) \qquad \text{where } k = 1 \ldots z^{[s]}$$

where $z^{[s]}$ = number of neurons in layer $[s]$ and $z^{[s-1]}$ = number of neurons in layer $[s-1]$.

The only requirement of equation (4) is that it is differentiable. Examples are the logistic and hyperbolic tangent functions. In supervised learning networks the predicted group membership is compared with the observed membership to produce an error, $e(i)$:

$$e(i) = d(i) - y(i) \tag{5}$$

where $d(i)$ denotes the observed group membership, and $y(i)$ the predicted group membership, for case i. The error is transformed to give

$$\varepsilon(i) = 0{\cdot}5e(i)^2 \tag{6}$$

and the mean value of $\varepsilon(i)$across all N cases is calculated to give the 'average squared error', E:

$$E = \frac{1}{N}\sum_{i=1}^{N} 0{\cdot}5e^2(i).$$

Initially a set of weights in each neuron is chosen randomly. The value of $\varepsilon(i)$ is calculated for each case and the weights are altered. The procedure is repeated for each case with the aim of adjusting the weights to minimise E.

For each case, i, the weights in each neuron in each layer are altered by a magnitude which is proportional to their marginal effect on $\varepsilon(i)$, that is, by a magnitude which is proportional to $\partial\varepsilon(i)/\partial W_{kj}^{[s]}$. Thus

$$\Delta W_{kj} = \eta\frac{\partial\varepsilon(i)}{\partial W_{kj}^{[s]}}. \tag{7}$$

The precise formula for the modification of W_{kj} depends on whether the weights are in the final layer or a hidden layer. Given equations (3)–(6), the $\partial\varepsilon/\partial W$ term in equation (7) can be written as

$$\frac{\partial\varepsilon}{\partial W_{kj}^{[s]}} = \frac{\partial\varepsilon}{\partial e_k}\frac{\partial e_k}{\partial y_k}\frac{\partial y_k}{\partial u_k}\frac{\partial u_k}{\partial W_{kj}} \tag{8}$$

where all variables relate to case i and so the '(i)' term is dropped. Calculating the partial derivatives, equation (8) gives

$$\frac{\partial\varepsilon}{\partial W_{kj}^{[s]}} = -e_k^{[s]} f^1(u_k^{[s]}) y_j^{[s-1]}. \tag{9}$$

In the output layer, d_k is known and so, therefore, is e_k. Given the functional form of equation (4), $f^1(U_k^{[s]})$ can be found and the $y_j^{[s-1]}$ is known, so equation (9) can be evaluated, and given η, ΔW_{kj} can be calculated.

In a hidden layer e is not observable. Instead the error values from the final layer are back propagated to the weights in earlier layers. The formula used is

$$\Delta W_{kj}^{[s]} = \eta\delta_k^{[s]} y_j^{[s-1]} \tag{10}$$

and

$$\delta_k^{[s]} = f_k^{1[s]} \sum_l \delta_l^{[s+1]} W_{lk}^{[s+1]} \tag{11}$$

where $\delta_l^{[s+1]} = -\frac{\partial \varepsilon}{\partial U_l^{[s+1]}} = e_l^{[s+1]} f_l'^{[s+1]}$ which is derived from applying the chain rule to equations (4)–(6) and substituting into (8). Thus if $[s+1]$ were the final layer, and $[s]$ a hidden layer, the value of δ_l would be known. If both $[s+1]$ and $[s]$ were hidden layers then from equation (11) it can be seen that $\delta_k^{[s]}$ is derived from $\delta_l^{[s+1]}$ and so, recursively, from $\delta^{[s+n]}$ where $[s+n]$ is the final output layer. Equation (10) is known as the 'delta rule' and δ the 'local gradient'.

As explained, the back propagation technique leads the weights to converge on those values which minimise E. However, for various reasons the rate of convergence is slow (Haykin, p. 190). Jacobs (1988) has outlined four heuristics to accelerate the rate of convergence. The method used in this paper is to allow the learning coefficient, η, to alter according to the delta-bar-delta method as new cases are fed through the network†.

2.3 *Genetic algorithms*

Consider a population of possible alternative solutions to a problem. Each solution may be represented as a collection, or string, of numbers. Each such number is known as a 'gene'. Each specific value which a gene can take on is known as an 'allele'. A collection of such numbers, arranged in specific positions, is known as a 'chromosome'. Genetic algorithms essentially evaluate the success, called 'fitness', of each possible solution in achieving a given objective, and by the processes of mutation and crossover, they alter potential solutions to improve their fitness. Crossover involves selecting, say, two potential solutions and exchanging portions of these solutions. The probability that a solution is selected is positively related to the fitness of the solution. The resulting 'children' replace members of the population with the lowest fitness and the procedure is repeated. At some stages various values (called 'alleles') in a chromosome are mutated: they are replaced by a different value. The probability that a string is selected for crossover is proportional to its fitness. The probability that an allele is mutated is also stochastic. GAs were first devised by Holland (1975) and have been used in a variety of contexts: optimisation, classification and clustering problems (Albright, 1993; Goldberg, 1989).

In the credit applicant classification problem two approaches have been followed. Albright (1993) estimated the parameters of a polynomial scoring equation where each chromosome consisted of estimated parameters. A case was classified according to whether or not the value of the function exceeded a critical value. An alternative method is to

† To explain this method, define

$$P_{ji}(n) = (1-\lambda)\frac{\partial \varepsilon(n-1)}{\partial W_{ji}(n-1)} + \lambda P_{ji}(n-1).$$

The change in η between one case and the following case is then equal to a constant if $P_{ji}(n-1)$ and $\partial \varepsilon^{(y)}/\partial W_{ji}(n)$ have different signs, equals $-\beta \eta_{ji}$ if P and $\frac{\partial \varepsilon}{\partial W_{ji}}$ have the same signs, and equals zero otherwise. These properties fulfil two of Jacob's heuristics.

estimate ranges of values of each characteristic and a switching value which indicates whether a characteristic is to enter the classification or not.

There are few direct comparisons of classificatory accuracy in the credit scoring context involving GAs. Fogarty & Ireson (1993/4) compared the performance of a GA, a decision tree approach, a nearest neighbour approach and a Bayesian method and concluded that the GA was superior to the other methods, although with only 6% of the sample classified as bads none of these methods were superior to classifying all cases as goods. Albright (1993), in an application of a GA to 918 credit union loans, correctly classified 76% of the sample. Desai *et al.* (1997) found that GAs were inferior to traditional techniques and neural networks. King *et al.* (1994) using a dataset with 8900 cases found that a decision tree method gave a higher percentage correctly classified than either neural networks, GA or a range of many different statistical techniques.

2.4 *Decision trees*

In these techniques a characteristic is chosen and a particular value of the characteristic is chosen to partition the cases into two subsets. The characteristic becomes a decision node and each decision, indicated by a particular value of the characteristic, forms a branch. Each branch leads to a different characteristic. Again a particular value of this characteristic is chosen to partition the subsets into further subsets and so on. After successive partitions the members of the final subsets will be members of only one group. This is known as a Recursive Partitioning Algorithm (RPA). New applicants can be classified by successively applying the partitioning criteria to identify the predicted group membership.

Different tests have been used to determine the appropriate characteristics and critical values for each decision node. In this paper we used an information criterion (Quinlan, 1993) whereby different characteristics and different critical values of each are examined, and those which maximise the difference between the information required to partition a set before the test and that information after the test are chosen for partitioning. The details are as follows.

Suppose a test with k outcomes partitions a training set into $T_i (i = 1 \ldots k)$ subsets. Call the probability that a randomly chosen member, i, of any subset, T_k, will be a member of a particular class, C_j, $P(C_j)_{i \in T_k}$. According to Quinlan 'the information conveyed by a message depends on its probability'. Hence the information relating to class membership is:

$$I(T_k) = -\sum_{j=1}^{l} P_i(C_j) \cdot \log_2(P_i(C_j)) \qquad \text{bits.}$$

This is the entropy of set T_k.

If applied to the whole training set, T, we gain the average information needed to correctly classify a randomly chosen case.

Suppose a test with n outcomes is used to divide the training sample, T, into T_k subsets $(k = 1 \ldots n)$. Then the average information needed to correctly classify a case, $I_x(T)$, is:

$I_x(T) = \sum_{k=1}^{n} p_k \cdot I(T_k)$ where p_k is the proportion of all cases which is in subset T_k, and $I(T_k)$ is as above.

The method then chooses the characteristic and cutoff values to maximise:

$$\text{Gain Ratio}_x = G_x / S_x$$

where

$$G_x = I(T) - I_x(T)$$

and

$$S_x = -\sum_{k=1}^{n} p_k \cdot \log_2 p_k$$

G_x is the information gained by partitioning the training set using test X over that gained from allocating members of the training set at random. S_x is the information gained by randomly dividing the training set into n groups.

The RPA will create splits until either no further improvement is possible or each leaf contains only one group: goods or bads. Quinlan argues that such a tree may 'overfit' the data in the sense that the optimal partitions for the training sample may not give the best classification in the test sample. To reduce the degree of overfit the optimal tree is constructed for the training sample and branches are then pruned off.

Quinlan (1993) describes a technique to transform a decision tree into propositional-like production rules. The rules are in the form: $L \rightarrow R$ where the left-hand side is a conjunction of attribute based tests and the right hand side is a class. The technique used involved four stages as follows. First the initial rules are obtained by following the path from the root to each leaf. The rule's left-hand-side contains the conditions and the right hand side is the class of that leaf. The path from the root to a leaf gives an initial rule. Second, the initial rules are simplified by removing conditions that do not seem to be useful in discriminating this class from others. Third all the rules for a class are examined and the ones that do not contribute to the accuracy of the set of rules as a whole are removed. This selection process is applied to each class. Finally the set of rules for the classes are ordered in order to minimise the false positive errors and then a default class is chosen.

3. Data preparation

3.1 *Sample*

The data set consists of 1001 individuals who had been issued with a credit card, who had used it and whose repayment history was known. A case was defined as a 'slow' if the individual had missed one or more repayments in the sample period and 'good' otherwise. This is a relatively stringent criterion; most credit grantors would be concerned only if three consecutive payments were missed, and would then target the account for collection procedures. However, given the small sample size available such a criterion would give a very small number of 'bad' cases, whereas the discrimination between *slows* and goods results in 38.1% of cases in the smaller group.

Furthermore, such a small sample size may be all that would be available, in practice, from some sectors of the US consumer credit market, such as credit unions. In addition, it might represent the size of the first available sample after the introduction of a new product.

The same fourteen characteristics were used in all four methods as described in Table 1. However many of the variables were transformed in different ways between the four algorithms. These transforms are explained below.

TABLE 1
Characteristics variables

Applicant's employment status
Years at bank
Home mortgage value
Number of children
Years at present employment
Residential status
Types of account
Other cards held
Outgoings
Estimated value of home
Home phone
Applicant's income
Spouse's income
Major credit cards held

3.2 *LDA*

The LDA results have been reported in Crook *et al.* (1992) using the same dataset as has been used with the evolutionary techniques in this paper. One way of transforming nominal level data is to use dummy variables. But given the relatively small degrees of freedom which would result we used an alternative approach, the weights of evidence method. Values of nominal variables were banded together according to similarity of $g_i/(g_i + b_i)$ where g_i and b_i are, the number of good and slow cases, respectively, in the nominal category i. The alphabetic values were then replaced by X_j where:

$$X_j = \log(g_i/b_i) + \log(B_T/G_T)$$

where $b_j(g_j)$ = number of slow (good) cases in band j
$B_T(G_T)$ = total number of slow (good) cases in the dataset.
In the case of ratio level data, for consistency, a similar procedure was followed.

3.3 *Neural networks*

Large numeric values will make the values of $U_k^{[1]}$ equation (3) very large and so the value of f^1 in equation (9) close to zero. This would result in slow learning. Therefore to speed up learning the data was scaled to give a fairly even distribution in the [0 : 0·9] range. The scaling procedure varied between the variables. As an example, in the case of income where the vast majority of values were less than £35 000, the following transformation was used:

$$x' = \log\left(\frac{x}{35\,000}\right) + 1 \qquad \text{if income } \leqslant £31\,699$$

$$x' = 0{\cdot}9 \qquad \text{if income } > £31\,699$$

In the case of nominal level data, one-of-N coding was used. That is where such a variable had N possible alphabetic values, N separate inputs were presented to the network with the digit 1 in one of the N digits used to represent the value for each input variable.

3.4 *Decision trees*

No transformations were required with this technique since the minimum level of measurement required is nominal.

3.5 *Genetic algorithms*

GAs require numeric values at nominal level or higher. Therefore alphabetic variables were replaced by adjacent natural numbers. For example, residential status, with six possible alphabetic values, was recoded with a numeric value between 0 and 5 replacing each alphabetic value. Finally four of the ratio level variables were scaled by dividing by a constant.

4. Implementation issues

The LDA results, drawn from Crook *et al.* (1992) were derived after applying a stepwise routine to an initial data set of 21 variables. The 14 significant variables which resulted were used in the alternative methods reported in this paper. The validity of the model was tested using the jacknife method. That is the models was estimated for all cases except for one which was used as the test sample. This was repeated for all 1001 cases. The LDA was estimated using BMDP.

4.1 *Neural networks*

Different network topologies, learning and momentum rates, activation functions and epoch numbers were experimented with. The number of iterations ranged between 40 000 and 200 000. The learning rule used was delta—bar—delta as explained in Section 2 above. The number of nodes used in the input layer was usually 29 although in a few experiments we used 23 when only eight fields were included. Either one or two hidden layers were used. The number of nodes in the first hidden layer ranged between 18 and 5; the number in the second hidden layer ranged between 3 and 5. The number of nodes in the output layer was 2. The number of epochs ie number of times the datasets were presented to the network ranged between 50 and 500.

The networks were build using a randomly selected sample of 501 cases and tested on the remaining 500 cases. The networks were estimated using NeuralWorks Professional Plus.

4.2 *Genetic algorithms*

In this application each chromosome consisted of a string of adjacent blocks of adjacent digits (genes). Each block related to a particular applicant characteristic, such as years at address. In the case of continuously measured variables each block consisted of three

genes: the maximum, minimum and outcome value respectively. The values of the outcome variables were coded 0 for 'do not use this variable to imply an outcome', or 1 for 'use this variable such that if a case meets the conditions indicated by the range in the preceding genes, classify the case as a good'. In the case of a binary variable a block consisted of a single value and an outcome value. The reason why the outcome variable was used to imply two joint conditions (whether to use and, if used, group membership) is that this allows the GA to generate predictive conditions which relate only to a subset of the attributes, rather than all of them.

A chromosome therefore consisted of l genes where $l = 2 \cdot b + 3 \cdot c$ where b is the number of binary attributes and c is the number of continuously measured attributes. A (simplified) example is shown below

No of children			Credit card		Years at present employment		
3	0	0	0	1	6	12	1

This chromosome indicates that having between 0 and 3 children implies the condition relating to the number of children is not to be used to predict group membership, and that having no credit card and between 6 and 12 years at present employment implies a case is a good.

A chromosome thus represents a solution in the form of a number of predictions. The predictions take the form whereby firstly, the values of the variables with an indicator value of 1 implying a joint condition associated with a case being used to make a prediction and that prediction being that the case is a good, and secondly, values of variables with an indicator value of 0 implying that the attribute is not to be used to predict group membership. Mutation occurred by changing any of the three (or two in the case of binary variables) genes representing a characteristic.

The fitness of the model was estimated using the following steps. First, identify those characteristics where the outcome gene has the value 1, that is it indicates that the characteristic should be used and that a case which fulfils the condition would be classified as a good. Second, for each case compare the observed characteristic values with those of the condition in the chromosome. If the observed values fulfil the conditions in the chromosome predict that the case is a good, and if this is a correct prediction increase the number of hits by one. If the observed values do not fulfil the conditions in the chromosome but the case is actually a slow, also increase the number of hits by one. This implies that the success of the rule contained in the chromosome is equally 'high' if it correctly classifies goods as if it correctly classifies slows. The fitness (Φ) of the chromosome is then calculated as: $\Phi = (r_h)^{0.5} - \alpha/N_h$ where r_h is the ratio of the number of hits to the total number of training cases, N_h is the total number of hits and α is a constant. The second term represents a penalty for achieving a small number of hits, implying unreliability of the first term.

The GAs were estimated using a package called PGA (Ross & Ballinger, 1993).

4.3 *Decision trees*

The models were tested using ten way cross-validation. The data was divided into $N = 10$ datasets such that each set contained an equal distribution of goods and slows. Then N

different models are built, in each case using an aggregation of $N - 1$ sets as the training set and the remaining single set as the test set. This ensures that each case appears in only one test set. The overall proportion of test cases correctly classified is then the average value over the N test sets.

The trees were estimated using $C4{\cdot}5$ (Quinlan, 1993).

5. Results

5.1 *Comparisons within techniques*

As explained in Section 4 a large number of experiments were performed with the evolutionary techniques.

In the case of neural networks we experimented with the transformation function (tanh or sigmoid), the topology (number of input nodes and number of hidden layers), the epoch number and the number of iterations. The greatest total proportion of cases correctly classified came from a sigmoid transformation function with all variables included, one hidden layer and an epoch number of 500 and 40 000 iterations.

Our experiments suggest a number of general conclusions, as follows.

1. Whilst increasing the number of iterations above 40 000 did not always increase the proportion of total test cases correctly classified, it often increased the proportion of goods and decreased the proportion of slows correctly classified.
2. Increasing the epoch size to 500 generally, but not always, increased the predictive accuracy. For example, given a sigmoid transfer function, two hidden layers and 40 000 iterations, increasing the epoch size from 50 to 100 increased the proportion of the test sample which was correctly classified from 61.90% to 63.31%. Increasing the epoch size to 250 and to 500 resulted in the performance deteriorating to 61.90% correctly classified.
3. Increasing the number of inputs in the first hidden layer and removing the second hidden layer increased the proportion of cases which were correctly classified. For example, again consider a sigmoid transfer function, an epoch number of 500 and 40 000 iterations. Changing from a 29-8-3-2 (number of nodes in the input layer, in the first hidden layer, in the second hidden layer and in the output layer, respectively) to a 29-12-2, increased the proportion correctly classified from 61.90% to 64.20%. (However, changing from 29-8-3-2 to 29-6-2 led to a marginal reduction in predictive performance.)
4. The tanh transformation function generally gave inferior results compared to the sigmoid function.

Turning to the RPA, ten different trees were built on 901 cases each and each tested on the remaining 100 cases. For each tree a set of production rules were derived and tested, again on test datasets of 100 cases. The size of the trees ranged from 421 to 470 leaves and decision nodes in the training samples. After pruning, the number of leaves and decision nodes ranged from 154 to 233. In 8 out of the 10 trees pruning reduced the proportion of the test set misclassified. This suggests that the prepruned trees overfitted the training dataset.

In all cases the error rate in the training sample increased after pruning as one would expect. The range of proportions of the training sets correctly classified by production rules

TABLE 2
Comparison of results: percentage correctly classified

Techniques	Total test sample	Goods	Slows
LDA[1]	68·4	85	40
Neural networks	64·2	79	39
Decision trees	62·3		
Genetic algorithms	64·5		
C_{prop}	46·8		

Notes: [1] from Crook *et al.* (1992)

was 72.9% to 61.9%. The corresponding figures for the test sets were 67.0% to 59.0%. These differences are due to differences between the selected training and test samples.

In the case of GAs various experiments were conducted. We initially used steady state reproduction whereby children were added to the population of possible solutions. We then used various combinations of (a) one child or twins produced by each crossover, (b) generational reproduction, whereby the new child is placed in a new population which replaces the old one when n children (n being the number of solutions in the population) are produced (c) with or without mutation and (d) with or without crossover.

The maximum fitness was gained with generational reproduction, with twins, with mutation and crossover. The lowest fitness, occurred with steady state reproduction, without twins, no mutation and with crossover turned on. In general, producing twins rather than not doing so, and applying crossover rather than not applying it, increased the fitness of the resulting solution. The maximum proportion of the test cases correctly classified occurred over several combinations of parameters including that which gave the greatest fitness.

The advantage of generational reproduction is that to some extent it improves a weakness of steady state reproduction. This weakness is that in steady state reproduction a fit solution takes over the population very quickly and does not allow variations to stay in the population long enough to contribute to the final solution.

5.2 *Comparison across techniques*

Table 2 shows the relative performance of each technique and the proportion of cases which could be correctly classified by chance (C_{prop}). In the case of LDA the proportion correctly classified was calculated using the jacknife technique. For NNs the proportion given is the highest proportion given by any experiment. For decision trees the proportion is the mean proportion over ten trees. In the case of GAs the porportion is that corresponding to the highest mean fitness over 50 runs of each experiment.

The most successful technique in terms of the proportion of the total number of cases correctly classified is that reproduced from Crook *et al.* (1992): LDA. GAs and NNs appear to classify approximately the same proportion of cases, but the success of GAs is probably overemphasised because the reported result is based on the classification of the training

sample. This was a constraint imposed by the software. Decision trees performed least well of the four methods.

Comparisons of the proportions of goods and slows correctly classified is possible only between LDA and NNs. Nets were found to be marginally less successful than LDA in classifying slows and considerably less successful at predicting the goods. Given that nets are computationally more time consuming, the findings of this paper suggest that LDA appears to dominate nets in terms of both accuracy and cost.

Given that neural networks and GAs allow for far more nonlinearities in the relationship between the individual's characteristics and his or her repayment performance than LDA, the superior performance of LDA may seem surprising. Possible explanations for these differences are: that the training samples and test samples differ between the techniques, that the transformations applied to the characteristics variables in LDA result in superior predictive performance, and that the evolutionary techniques have found local, but not global, optimum solution sets. Considering the first possibility it is likely that, everything else equal, the larger training sample used (1000 cases) in the case of LDA would give estimated parameters with lower variances about the population values than the smaller samples (500 cases) used to estimated the neural nets. This explanation would apply to a lesser extent to the decision trees where the training samples consisted of repeated samples of 900 cases each. The explanation would not apply to the GAs since they were trained and tested in the training sample and this is known to over estimate the predictive performance.

The validity of the second possible explanation for the different results is difficult to assess. On the one hand aggregating responses across different alphabetic values for each characteristic constrains the estimated coefficients compared with using dummy variables for each alphabetic value; on the other hand, the transformed value used relates to the value closely correlated with the dependent variable in the discriminating equation. Therefore the effect of the transformation on the predictive power of LDA versus GAs or trees is difficult to disentangle because neither of the latter two methods involved transformed values.

The third explanation is always applicable to any use of all of these techniques. We experimented with a number of parameters under our control in all three evolutionary techniques as explained above.

Our results are consistent with those of certain other studies when credit scoring data is used, but inconsistent with others. For example in a three way classification, Desai *et al.* (1997) found that when considering the predictive performance for their total sample, LDA was almost identical to that of neural networks and slightly better than GAs. But neural networks (with a best neuron rule) gave the greatest predictive performance when predicting poor payers (rather than goods or chargeoffs and bankrupts) and GAs were superior when predicting chargeoffs and bankrupts. King *et al.* (1994) found that LDA gave a greater predictive performance than neural networks, but a poorer performance compared with decision trees. On the other hand, Desai *et al.* (1996) in a two way classification, found that LDA was *inferior* to neural networks at predicting both good and bad payers (separately). In addition Khoylou & Stirling (1993) using a sample of two thousand cases, found that neural networks performed considerably better than multiple linear regression, although the latter was not included in our comparison. In terms of ranking decision trees and GAs, the ranking we obtained is exactly the same as that gained by Fogarty & Ireson (1993/4).

The differences between our results and those of other studies could be explained

in a number of different ways: differences in the types of individuals in the samples, differences in the sample sizes, differences in the ranges of controllable parameters used in the experiments, differences in the transformations applied to the data and so on.

6. Conclusion

We have compared the predictive performance of LDA, neural networks, GAs and decision trees using a small sample of credit scoring data. Like other studies we found that LDA was superior to GAs. However consistent with some studies but unlike others we found that neural networks were inferior to LDA. Unlike other studies we found that neural networks were almost identical to LDA at predicting the slow payers, and that LDA was superior at predicting good payers. Further research is needed particularly to incorporate the relative costs of rejected good applicants and accepted slow payers, using larger datasets which preserve the proportions of good and slow payers in the population.

References

ALBRIGHT, H. 1993 *Construction of a Polynomial Classifier for Consumer Loan Applications Using Generic Algorithms*, Mimeo, Department of Systems Engineering, University of Virginia.

CHOI, S. C. 1986 Discrimination and classification: an overview. *Comput. Math. Appl.* **12**, 173–177.

CROOK, J. N., HAMILTON, R., & THOMAS, L. C. 1992 A comparison of a credit scoring model with a credit performance model. *The Service Industries Journal* **12**.

DESAI, V. S., CROOK, J. N., & OVERSTREET, G. A. 1996 A comparison of neural networks and linear scoring models in the credit union environment. *Europ. J. Oper. Res.* **95**, 24–37.

DESAI, V. S., CONWAY, D. G., CROOK, J. N., & OVERSTREET, G. A. 1997 Credit scoring models in the credit union environment using neural networks and genetic algorithms. *IMA J. Math. Appl. Business and Industry*, Forthcoming.

EISENBEIS, R. A. & AVERY, R. B. 1972 *Discriminant Analysis and Classification Procedures: Theory and Applications*. Lexington, MS: DC Heath.

FISHER, R. A. 1936 The use of multiple measurement in taxonomics problems. *Ann. Eugenics* **7**, 179–188.

FOGARTY, T. C. & IRESON, N. S. 1993/4 Evolving Baysian classifiers for credit control—a comparison with other machine learning methods. *IMA J. Math Appl. Business and Industry* **5**, 63–75.

GOLDBERG, D. E. 1989 *Genetic Algorithms in Search, Optimization, and Machine Learning*. Massachusetts: Addison-Wesley.

HAYKIN, S. 1994 *Neural Networks: A Comprehensive Foundation*. New York: Macmillan.

HOLLAND, J. H. 1975 *Adaptation in Artificial and Natural Systems*. Ann Arbor: The University of Michigan Press.

JACOBS, R. A. 1988 Increased rates of convergence through learning rate adaption. *Neural Networks* **1**, 295–307.

KHOYLOU, J. & STIRLING, M. 1993 *Credit Scoring and Neural Networks*, Presented at Credit Scoring and Credit Control conference, Edinburgh.

KING, R. D., HENERY, R., FENG, C., & SUTHERLAND, A. 1994 A comparative study of classification algorithms: statistical, machine learning and neural network. *Machine Intelligence*, vol 13, (K. Furukwa, D. Michie & S. Muggleton eds). Oxford: Oxford University Press.

QUINLAN, J. R. 1993 *C4.5: Programs for Machine Learning*. San Mateo, California: Morgan Kaufman.

ROSS, P. & BALLINGER 1993 *PGA*, Software program, University of Edinburgh.

TATSUOKA, M. M. 1970 *Discriminant Analysis*, Selected ropics in advanced statistics: an elementary approach, Number 6, Illinois: Institute for Personality and Ability Testing.

THOMAS, L. C. 1997 Methodologies for classifying applicants for credit. *Statistics in Finance*. (D. Hand & S. Jacka eds). Edward Arnold.

YOBAS, M. B. 1996 Credit Scoring Using Neural and Evolutionary Techniques, *MSc Thesis*, Department of Artificial Intelligence, University of Edinburgh.