Using Genetic Algorithms to Develop Scoring Models For Alternative Measures of Performance.

Steven Finlay
Lancaster University Management School

Abstract

Most approaches to credit scoring result in the generation of a model that minimises some function of error between actual and predicted values, or that maximises likelihood. Popular approaches include least squares methods such as linear regression and disciminant analysis, and maximum likelihood methods such as logistic regression.

In practice, the criteria by which the parameters of a model are determined and the criteria by which models are then assessed are different. Practioners tend not to be interested in standard statistical measures of model fit, such as the R^2 coefficient for linear regression or the likelihood ratio for logistic regression. Performance will often be assessed using global measures such as the GINI coefficient or KS statistic, or by considering the misclassification rate at different points in the score distribution. For example, a common goal is to minimise the number of mis-classified cases for the cut-off score that yields the desired acceptance rate within the population.

In this paper an approach using genetic algorithms is described, in which a credit scoring model is created in the form of a linear combination of independent variables, without recourse to 'intermediate' measures of performance such as sum of squared error or likelihood. Instead, the training algorithm is used to directly maximise/minimise the measure of interest; that is, the maximisation of the GINI coefficient and the minimisation of the misclassification rate for a range of different acceptance rates. Empirical results are presented, with performance compared to that of a range of models constructed using more traditional approaches including logistic regression and neural networks.

Acknowledgements. The author would like to thank Experian UK for their support for this research.

Introduction

In many real world situations, the objective a lender is trying to optimise through the use of a credit scoring model is different from the objective used during model development. For example, with logistic regression a model is chosen that maximises likelihood, yet what is usually of interest is the relative performance at specific points in the distribution of ranked model scores, rather than some overall measure of predictive accuracy (Thomas et al. 2001). This point can easily be demonstrated by considering a hypothetical example. Imagine that there exists two models that generate probabilistic estimates of credit applications being good credit risks. Two credit applications are scored by each model to produce the results shown in Table 1:

Model	***Model Estimates***		***Likelihood***
	$P(y_1 = \text{good})$	$P(y_2 = \text{good})$	$\boldsymbol{P(y_1 = \text{good})*P(y_2 = \text{good})}$
Model 1	0.75	0.98	0.735
Model 2	0.82	0.85	0.697

Table 1. Hypothetical model estimates

Now assume that both cases are revealed to be good payers. From a maximum likelihood perspective, Model 1 clearly outperforms Model 2. Yet, if a lender was using these models to make credit granting decisions, say on the basis of accepting only those where the estimated probability of being good exceeds 0.8, then Model 2 is better because both cases would be accepted, whereas only one would be accepted using Model 1. Maximising likelihood is therefore, no guarantee of optimal model performance in this case.

A Genetic Algorithm (GA) is a data driven, non-parametric heuristic search process, where the training algorithm can be chosen to optimise a wide range of objective functions. Because the training algorithm is guided only by the performance of competing solutions, GAs have the potential to generate models that outperform other

approaches to scorecard construction in situations where the objective function that a user wishes to optimise, differs from that used within the modelling process.

Previous studies where GAs have been used to develop credit scoring models have reported mixed findings. Fogarty and Ireson (1993/4) took a sample of over fifty thousand accepted credit card applications and compared a GA derived Bayesian classifier with decision rules derived from a number of techniques including a nearest neighbour clustering algorithm, a decision tree and a simple Bayesian classifier. They found that the GA derived classifier performed better than other methods when assessed on classification rates, but did not perform better than a simple decision rule to classify all cases as good. Desai et al. (1997) looked at a three-way classification problem where accounts where classified as good, poor or bad payers. They reported that a GA approach was marginally better at classifying the worst accounts (bad payers) than linear disciminant analysis, logistic regression and a variety of neural network models, but did not perform as well when measured in terms of classification performance on good and poor paying accounts. Yobas et al. (2000) reported that while a GA derived model performed better than neural networks and decision trees on the development sample (no validation sample performance was available for the GA derived model), all three methods were outperformed by linear disciminant analysis. While the results and methodologies applied in these previous studies differs, one feature that they all have in common is that they only considered misclassification performance metrics for which the non-GA approaches used in the study were generally known to provide good levels performance. It is therefore, no surprise that a GA approach was not found to significantly outperform the alternative model development approaches examined. In this paper a GA approach is again explored, but incorporating a number of features that differentiate it from previous studies. First, rather than simply judging performance of competing models on the basis of a single misclassification measure, model performance was assessed on a number of different criteria:

- The GINI coefficient (A measure of the area under the receiver operator curve)

- The proportion of bads scoring above the cut-off for reject rates of 5,10,25 and 50%

In each case a GA was applied to generate a scoring model that maximised each objective independently, whereas a single competing model was constructed and assessed using the competitor approaches. Second, a large real world data set is used, whereas previous studies have been based on relatively low dimensional data sets and small samples (with the exception of Fogarty and Ireson's study). Thirdly, solutions are considered with and without seeding - the process whereby a genetic algorithm is initialised using a number of pre-existing solutions found using some alternative technique. The GA is then applied in an attempt to improve upon the performance of the original seed solution(s).

Empirical results are presented, with the performance of the GA derived models compared to models constructed using logistic regression, multiple OLS regression and a single hidden layer neural network.

Overview of Genetic Algorithms

The theory of GAs was developed in the late 1960s and early 1970s by John Holland (1975) as a means to study evolutionary process in nature, but were quickly adopted as a heuristic approach applicable to a wide range of optimisation problems (Hollstien 1971; De Jong 1975). The general principles of GAs are analogous to Darwinian principles of natural selection and survival of the fittest, and the terminology employed to describe GA training and selection is taken from the biological analogy. In nature, the structure of a living organism is defined by a set of genes, each of which can assume a number of states or values (termed alleles). Chromosomes are defined as the set(s) of genes that describe a complete individual. According to natural selection, within a population of individuals those that are fitter; that is, better adapted to their environment, tend to survive and reproduce while the weaker members of the population tend to die out. Sexual reproduction acts as a mechanism by which genes from different individuals are combined and passed on to subsequent generations, giving offspring the potential to be fitter than either of the individual parents - should the best genes from each parent be passed on. Over time the general fitness of the population tends to improve because the genes from the fittest individuals have a

greater propensity to be propagated from one generation to the next than genes from less fit individuals.

While the exchange of genes through sexual reproduction leads to improved fitness within the population, a secondary evolutionary driver is mutation – where spontaneous changes to genes occur (for example, as a result of a gene being in collision with a high energy particle such as a cosmic ray) resulting in new and previously unseen features appearing within the population. Mutation is an important mechanism for driving the evolutionary process for two reasons. First, in some situations the population will only contain a relatively small number of all possible gene/allele combinations. Second, it is possible through inbreeding for the members of the population to become genetically very similar. Mutation therefore provides a method by which diversity can be maintained or reintroduced to a population.

With GAs a set of possible solutions to a given problem is analogous to a population of individuals in the natural world. The goal of the GA is to combine together and mutate different solutions so that over time fitter (better) solutions evolve. Each individual solution within the population is represented in the form of a finite length string, comprising a finite alphabet where the string and its component characters are analogous to chromosomes and genes respectively (Goldberg 1989a). From an initial (usually randomly generated) population of strings, new populations are created over a number of generations (iterations) through the application of the following genetic operators:

- Selection: From the existing population, a number of strings are selected for breeding, with selection favouring those strings that represent the best solutions found to date.

- Crossover: From the selected population, pairs of strings are matched for breeding. ‘Child’ strings are created by selecting and combining different characters from each of the parent strings.

- Mutation: Each character within a string has the chance to undergo mutation, based on some random selection process. If selected, then the value of the character within the string is randomly reassigned to one of the possible values defined by the encoding alphabet.

The algorithm terminates when either a given number of generations have occurred, or when the improvement from one generation to the next falls below some pre-determined threshold value. Despite the relative simplicity of genetic algorithms, they have been successfully applied to a wide range of diverse and complex optimisation problems (Mitchell 1996; Coley 1999).

Design and Implementation of Genetic Algorithms

Although all GAs contain the key features of selection, crossover and mutation, there are a number of parameters and procedures that need to be selected for GA training, and as with methods such as neural networks, the parameters that deliver the best solution tend to be problem specific. Consequently, a good deal of trial and error can be exerted to try and find the most appropriate training parameters for a given problem. In this section some general research findings into design and implementation of GAs are discussed.

The first question is how best to encode solutions to a given problem. The most commonly adopted approach favoured by Holland (1975) is to encode possible solutions as binary strings using a two character alphabet. For example, if the objective is to optimise some function of two independent variables x_1 and x_2, where x_1 and x_2 can take positive integer values between 0 and 31, then any possible combination of values of x_1 and x_2 can be encoded using a string of ten binary characters representing a base 2 numbering system. The first five characters represent x_1 and the second five x_2 (or vice versa). So if the values of x_1 and x_2 were 28 and 9 respectively, they could be encoding as the following string:

x_1 x_2

1 1 1 0 0 0 1 0 0 1.

However, other forms of encoding are possible including integer and real number representations, and some studies have shown that a high cardinality alphabet can lead to better GA performance than using binary encoding (Wright 1991).

After choosing the encoding method to be employed, the population size is determined. If the population is too small then the solution space will not provide adequate coverage of the problem domain and the GA is likely to under perform. Too large, and an acceptably good solution may not be found in realistic time. Some theoretical work has indicated that the relationship between optimal population size and string length is exponential (Goldberg 1989b), which would suggest that for highly dimensional real world problems a GA approach may be impractical. However, empirical research has reported that population sizes as small as 30 can lead to reasonable solutions for many problems, and a population size of between l and $2l$, where l is the string length, is generally sufficient to allow an optimal solution to be found for binary encoded problems (Reeves 1995b).

The population is usually initialised to a set of random solutions, but an alternative is to use seeding, where the population is initialised using a set of existing solutions found using some other method. It has been reported that that seeding can be an important determinant of the quality of the final solution (Ahuja and Orlin 1997; Ahuja et al. 2000) and that seeding can be lead to reduced run times in what can be a computationally expensive exercise (Reeves 1995a). However, it has also been reported that seeding can lead to sub-optimal solutions (Kapsalis et al. 1993; Levine 1997). This is because the area of search may focus on the regions of the problem domain close to the seed solutions, and thus the GA may converge on a local rather than global optimum.

A variety of methods are available for selecting individuals for crossover. The simplest, truncation selection, is just to select the fittest x% percent of the population and discard the rest. However, this is not popular as there may be desirable characteristics within the poorer part of the population that have no chance of being propagated to the next generation (Coley 1999). More widely used selection methods are based on some form of random selection, where the probability of selection is a function of a solution's fitness. With fitness proportional (roulette wheel) selection the

probability of selection is directly proportional to the relative fitness of the individual compared to the population average. So an individual whose performance is twice average will have twice the chance of being selected than an average performing one. One drawback of this approach is that for problems where many similarly good solutions exist, or where the flat maximum effect is prevalent - as is found in many credit scoring problems (Lovie and Lovie 1986), fitness proportional selection can be little better than uniform random sampling; i.e. the principle of survival of the fittest is lost. While it is possible to apply scaling or a non-linear transformation to place greater emphasis on fitter individuals, a more popular approach is rank selection. Individuals are ranked by fitness and a probability of selection is calculated based on some linear or non-linear function of rank, with one of the commonest rank selection functions being:

$$P_{selection} = \frac{(n-r)}{\sum_{i=1}^{n} i}$$

where r is the rank and n is the population size. Another popular selection method is tournament selection. Pairs of individuals are chosen at random. A threshold value t, $0.5 < t <= 1$ is chosen. If a uniform random number R in the range (0-1) is greater than t then the fittest individual is selected; otherwise the less fit individual is selected. Ranking and tournament selection methods tend to perform equally well and both have been found to outperform fitness proportional selection (Goldberg and Deb 1991).

During crossover, pairs of strings from the selected population are mated, resulting in a new population of strings being created. A variety of crossover methods have been proposed. With k-point crossover, k cut-points are chosen at k different loci along the length of the two parent strings. The characters within each string are then copied and exchanged. An example of k-point crossover for k=2 is shown in Figure 1.

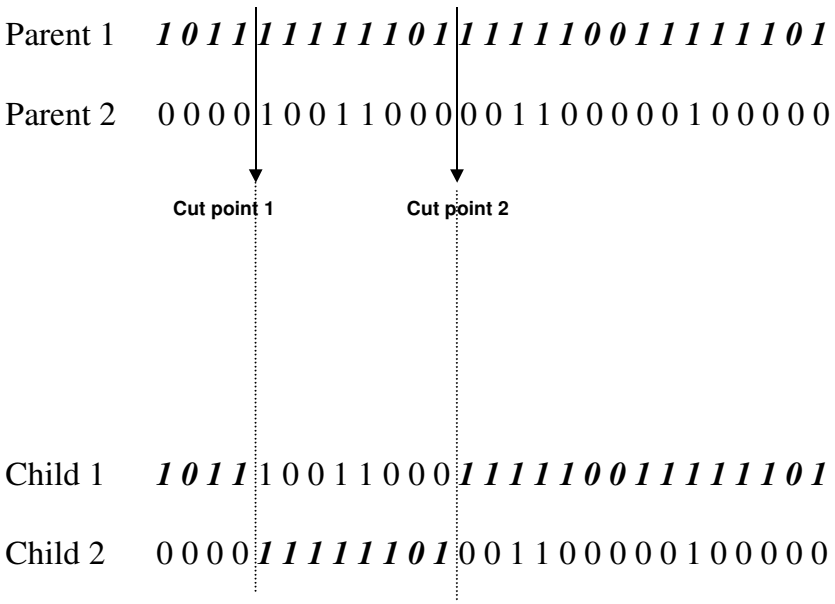

Figure 1. Example of k point crossover (k=2)

While *k* is commonly chosen to take a value of 1 or 2, with uniform crossover the value of *k* is set equal to *l*, the string length, so that the maximum diversity is achieved in the offspring. The perceived benefit of uniform crossover is that it removes representational bias that can arise when *k* is low. However, it creates a high degree of disruption which can lead to movement away from the optimal solution. A less disruptive approach is parameterised multi-point crossover where a probability is assigned for crossover occurring at each locus in the string (Spears and De Jong 1991).

After selection and crossover have occurred mutation is randomly applied to each character of each string within the population. If the probability of mutation is too high, then it is likely that good quality solutions will be lost. However, if the probability of mutation is low, then the population may tend to stagnate and progression towards the optimal solution will be slow. Like most other aspects of GAs the ideal mutation rate is problem specific. However, mutation rates between $1/l$ and $1/(n*\sqrt{l})$ where *l* is the string length and *n* is the population size, have been widely used (Coley 1999).

An additional element often applied to the selection and mutation process is elitism. If elitism is present then the very best solution(s) will always be selected for propagation into the next generation without mutation being applied. This ensures that the best solution found to date is always present within the current population.

Overall GAs tend to be remarkably robust, and a wide range of parameter values will lead to good solutions for many problems (Reeves 1995b). Therefore, from a practical perspective when applying a GA approach to real world problems, it is not usually necessary for an analyst to go to great lengths to find the best possible set of parameters to use, but rather to identify the range of possible parameter values for which good solutions are likely to exist, and to then (perhaps rather arbitrarily!) choose values that lie somewhere within these ranges.

Data

A data set of UK credit applications with known performance from April to June 2002 was supplied by Experian UK. Account performance data was supplied as at 12 months after the application date. After removal of outliers and indeterminates the sample contained 88,792 observations of which 75,528 where classified as good (no more than 1 month in arrears) and 13,264 as bad (3 months or more in arrears).

37 independent variables were identified on the development database. These included common application form characteristics such as age, residential status and income, as well as a set of commonly used credit bureau variables including number, value and time since most recent CCJ/bankruptcy, current and historical account performance, recent credit searches, Electoral Roll and MOSAIC postcode level classifiers. A full list of the independent variables used in the study is contained in Appendix A. The independent variables were a mixture of categorical and continuous variables. These were pre-processed to code them as a set of dummy variables. For categorical variables each category was coded as a separate dummy variable. For continuous and semi-continuous variables a separate dummy variable was defined for each 10% of the population. The dummy variable approach is generally found to provide a good approximation to non-linear features of a data set (Fox 2000) and is a standard practice in the development of credit scoring models (Hand and Henley,

1997). It has also been observed that the coding of continuous variables as a set of dummy variables can lead to greater discrimination in credit scoring models than using either raw or transformed versions of the continuous variables (Hand and Adams, 2000). After pre-processing 218 dummy variables were available for model construction.

Methodology

The goal of the exercise was to apply a GA to create a linear scoring function in the form $Y=B^TX$ where X is a column vector of independent variables and B a column vector of parameter coefficients. It should be noted that the resulting score, Y, should not be interpreted as an estimate of individual performance, but merely as the relative score of each observation within the dataset. Although other types of scoring function exist such as a multi-layer perception, a linear function was considered appropriate because linear scoring rules have continuing popularity with credit scoring practioners, perform well when compared to a wide variety of alternative modelling techniques (Baesens et al. 2003) and are desirable for a number of reasons in addition to their overall predictive performance. This includes having an intuitive structure so that model parameters can be used to provide a qualitative explanation of the model to the layperson in support of legislative requirements, and being easily implementable within the decision engines employed by lenders.

Two encoding schema were considered. One binary, the other integer. For the binary representation parameter coefficients, B, where defined as integer values in the range ±32,767 ($\pm 2^{15}$-1) and encoded as a 16 bit binary number, resulting in a total string length of 3504 (16*218 plus 16 for the model constant). For the integer encoding schema, model parameters were represented as a string of signed integers, each of which was allowed to take values in the range ±32,767. The string length for the integer encoding was therefore 219 (218 variables plus the model constant).

After considering the findings within the literature, and performing a number of experimental training runs, the parameters chosen for the GA training for both encoding schemas were as shown in Table 2.

Parameter	***Value/Method applied***	***Range of values/methods for which 'good' solutions where found***
Population size (n)	1500	>400
Selection method	Proportional ranking with replacement. $P_{selection} = \frac{(n-r)}{\sum_{i=1}^{n} i}$ where n = population size and r = population rank ($r = 1$ for best, $r = n$ for worst)	N/A
Crossover method	Parameterised multi-point crossover $P_{crossover} = 0.5$	$2/l < P_{crossover} < 1.0$ (where l=string length)
Mutation rate per character	$P_{mutate}=0.003$ (*Integer encoding*) $P_{mutate}=0.0005$ *(Binary encoding)*	$0.5/l < P_{mutate} < 3/l$ (where l=string length)
Elitism	Elitism applied.	N/A
Stopping criteria	1,000 generations, or no improvement in validation sample performance seen for 50 generations.	>100 generations

Table 2. GA Parameters

The maximum number of generations was set at 1,000, after results from a number of test runs showed performance tended to plateau after about 200-400 generations. Where seeding was applied, 100 seed models were constructed using a number of variations of logistic regression. This included forward and backward stepwise procedures, as well as a developing a set of models using 99 percent subsets of the main development dataset, with a different 1% of observations excluded for each model. For the binary encoding, mutation was applied by flipping the value of the mutated character to its alternative value; i.e. from 1 to 0 or vice versa. Where an integer encoding was used, mutation was applied through the use of a uniform random number, allowing the mutated character an equal probability of being assigned any value within the range of possible values (±32,767).

The performance of the GA derived model was compared with that of five competitor models constructed/selected by the following means:

C1. OLS regression (stepwise).

C2. OLS regression (non-stepwise).

C3. Logistic regression (stepwise).

C4. The best of a logistic regression model (non-stepwise) developed using the full development sample, and the set of seed models used to initialise the GA when seeding was applied.

C5. A single hidden layer neural network, trained using back propagation to minimise the overall mis-classification rate, with equal costs assigned for misclassifying goods and bads.

For models C1, C2, C3 and C5 a single model was constructed and used in all performance comparisons. For C4, the model selected for comparison was reviewed for each performance measure evaluated. This was to ensure that the GA derived model was always compared to the best competitor from the set of seed models, regardless of the performance metric chosen.

Preliminary analysis showed that validation performance of GA derived models tended to peak before that of the development sample and then tail off, which would suggest that a development/validation/holdout methodology was more appropriate than a simpler development/validation methodology applied in some empirical studies – including most of the previous GA studies relating to credit scoring. The population was segmented 60/20/20 and in all cases models were developed using the 60% development sample. Where a choice of models existed (for example with the GA and neural network approaches) the best model was selected based on performance on the 20% validation sample. Model performance for all competing models was then compared using the 20% holdout sample.

Results

Results are summarised in Table 3, with the best performing model for each measure highlighted in bold.

	GA derived models				Competitor models				
Performance Measure	***GA1***	***GA2***	***GA3***	***GA4***	***C1***	***C2***	***C3***	***C4***	***C5***
GINI	**72.55**	72.52	72.34	72.16	71.89	72.10	72.23	72.48	71.98
5 % reject rate	**72.11%**	72.41%	72.90%	73.09%	72.39%	72.41%	72.26%	72.23%	72.30%
10% reject rate	56.03%	56.11%	56.45%	57.08%	56.64%	56.41%	56.26%	56.07%	**55.73%**
25% reject rate	28.22%	28.26%	28.82%	28.64%	**27.70%**	27.81%	28.41%	28.15%	28.49%
50% reject rate	**8.40%**	8.70%	9.60%	9.71%	9.22%	9.18%	8.85%	8.55%	9.52%

Notes:

1. *GA1. Integer encoding, seeding applied.*
   GA2. Binary encoding, seeding applied.
   GA3. Integer encoding, seeding not applied.
   GA4. Binary encoding, seeding not applied.
2. *For reject rate measures performance is calculated as the proportion of total bads scoring above the cut-off that delivers the required reject rate.*
3. *For GINI measure higher = better. For reject rate measures lower= better.*

Table 3. Performance of Competing Models

The first point to note from Table 3, is how similar the performance of the competing models was across all measures. It is also the case that no one model could be said to dominate the others entirely, with different models performing slightly better/worse in relative terms for each measure. This is a positive finding for practioners because it supports the case for the flat maximum effect, and suggests that standard techniques such as logistic and linear regression produce models that are close to optimal when measured across a range of different performance criteria, even when the decisions being made are not perfectly aligned to the metrics used in model construction.

The GA derived models using seeding did show a propensity to outperform other approaches in some situations, albeit by a very small margin, when the measure of interest was GINI or bads scoring above the cut-off for a 5% and 50% reject rate.

However, in other situations the performance was worse than that of the competitor models. The results would suggest that while a GA approach can produce models that are competitive with traditional methods that include linear regression, logistic regression and neural networks, they do not significantly outperform these methods, and in some cases perform less well. It may be the case that alternative formulations of the GA could lead to better solutions, and this view is supported by the comparisons of the models derived from using binary and integer encodings. In nine out of the ten comparisons between a high cardinality integer encoded alphabet and a binary encoded alphabet, the models produced from integer encoding outperformed those produced from binary encoding. This is a significant difference at a 5% significance level, although not at the 1% level. However, given the robust nature of GAs to the parameters used, it is likely that any potential improvement in performance will be small. Where seeding was applied, the seeded GA models outperformed the non-seeded GA models in all cases, although in some cases this was simply due to the GA retaining the best seed model.

Practical Considerations

A GA is a computationally expensive process, and to be of use to practioners needs to be applicable to real world problems in realistic time. Therefore, it is prudent to briefly discuss resource requirements for running a GA for those in the credit industry who may be interested in applying a GA approach to model development. The GA program used in the study was developed using Microsoft Visual C++ All programs were run using a single processor AMD Athlon 64 3500+ PC with 1GB of RAM. Typical run times for a GA containing a population of 1500 models and run for 250 generations was around 5 hours, with little difference between the two encoding methods applied. As generally reported in the academic literature, the vast majority of CPU time (estimated to be 90% or more) was involved in calculating model scores or sorting observations by score to facilitate the calculation of the performance metrics (GINI coefficient and number of bads above score). The core GA processes of selection, crossover and mutation were relatively undemanding in terms of CPU, taking up no more than 10% of the total run time. This compared to about 12 minutes to produce a stepwise logistic regression using SAS PROC Logistic, and about 1 hour for the neural network model using SAS enterprise miner. Therefore, while the GA approach is computationally expensive when compared traditional methods of

scorecard construction, applying a GA approach to real world credit scoring problems using standard hardware/software is feasible. GAs are also well suited to parallel implementations, allowing those with access to large scale multi-processor systems to explore the problem domain to a far greater extent than the GAs developed in this study.

In terms of the models produced, while it is not possible for reasons of commercial sensitivity to display the full set of model parameters, the model structure produced from the GA approach were for the main part, what would be considered sensible; i.e. the pattern of scores allocated for different dummy variables was generally consistent with the univariate distribution of odds for each characteristic. However, as often found with standard modelling techniques, there were a number of instances where the model parameters did not follow the expected pattern, which is believed to be due to some moderate multicollinearity in the data. Similar patterns were also present within the non-stepwise linear and logistic regression models.

Discussion and Concluding Remarks

In this paper, a GA modelling approach to developing credit scoring models has been presented, and it has been shown that in some situations GA derived models can perform as well as, if not marginally better than, a range of other approaches to model development when measured in terms of GINI and mis-classification rates for a range of cut-off scenarios, but in other cases the performance can be marginally worse. This finding is broadly in line with those of previous studies into the application of GAs to credit scoring which concluded that a GA approach did not provide a consistent and significant benefit over alternative methodologies.

Two different encoding schema were examined, one a binary encoding the other a high cardinality integer encoding. In a significant number of cases, models developed using an integer encoding outperformed those developed using a binary encoding, which would support the case for using high cardinality alphabets when applying a GA approach to credit scoring problems. When seeding was used, slight improvements to model performance was observed in some cases, but the level of improvement was slight. However, seeding is easy to apply and GAs using seeding

can be less computationally demanding than non-seeded GAs which may justify the application of a seeding approach.

This research was performed using a single credit scoring data set. It would be desirable to repeated the research using other datasets, should they be become available. Another objective of any future research would be to undertake a more thorough investigation into the sensitivity of model performance to the parameter settings used.

References

Ahuja, R. K. and Orlin, J. B. (1997). "Developing Fitter Genetic Algorithms." INFORMS Journal On Computing **9**(3): 251-53.

Ahuja, R. K., Orlin, J. B., et al. (2000). "A greedy genetic algorithm for the quadratic assignment problem." Computers & Operations Research **27**(3): 917-34.

Baesens, B., Gestel, T. V., et al. (2003). "Benchmarking state-of-the-art classification algorithms for credit scoring." Journal of the Operational Research Society **54**(5): 627-35.

Coley, D. A. (1999). An Introduction to Genetic Algorithms for Scientists and Engineers, World Scientific.

De Jong, K. A. (1975). An analysis of the behavior of a class of genetic adaptive systems. Michigan, University of Michigan, Ann Arbor.

Desai, V. S., Conway, D. G., et al. (1997). "Credit-scoring models in the credit union environment using neural networks and genetic algorithms." IMA Journal of Mathematics Applied in Business and Industry **8**: 323-346.

Fogarty, T. C. and Ireson, N. S. (1993/4). "Evolving Bayesian classifiers for credit control - a comparison with other machine learning methods." IMA Journal of Mathematics Applied in Business and Industry **5**: 63-75.

Fox, J. (2000). Nonparametric Simple Regression, Sage.

Goldberg, D., E. (1989a). Genetic Algorithms in Search Optimisation & Machine Learning, Addison-Wesley.

Goldberg, D., E. (1989b). Sizing populations for serial and parallel genetic algorithms. Proceedings of the 3rd International Conference on Genetic Algorithms. J. D. Schaffer. Los Atlos, CA, Morgan Kaufmann Publishers.

Goldberg, D., E. and Deb, K. (1991). A comparative Analysis of Selection Schemes Used in Genetic Algorithms. Foundations of Genetic Algorithms. G. Rawlins, J. E. San Mateo, California, Morgan Kaufmann Publishers.

Holland, J., H. (1975). Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor.

Hollstien, R. B. (1971). Artificial genetic adaptation in computer control systems. Michigan, University of Michigan.

Kapsalis, A., Smith, G. D., et al. (1993). "Solving the graphical steiner problem using GAs." Journal of the Operational Research Society **44**: 397-406.

Levine, D. (1997). "Genetic Algorithms: A practioner's View." INFORMS Journal on Computing **9**(3): 256-9.

Lovie, A. and Lovie, P. (1986). "The flat maximum effect and linear scoring models for prediction." Journal of Forecasting **5**(3): 159-68.
Mitchell, M. (1996). An Introduction to Genetic Algorithms, The MIT Press.
Reeves, C. R. (1995a). "A genetic algorithm for flowshop sequencing." Computers & Operations Research **22**(1): 5-13.
Reeves, C. R., Ed. (1995b). Modern Heuristic Techniques for Combinatorial Problems, McGraw-Hill Book Company.
Spears, W. M. and De Jong, K. A. (1991). An Analysis of Multi-Point Crossover. Foundations of Genetic Algorithms. G. Rawlins, J. E. San Mateo, California, Morgan Kaufmann.
Thomas, L. C., Banasik, J., et al. (2001). "Recalibrating scorecards." Journal of the Operational Research Society **52**(9): 981-88.
Wright, A., H. (1991). Genetic Algorithms for Real Parameter Optimization. Foundations of Genetic Algorithms. G. Rawlins, J. E. San Mateo, California, Morgan Kaufmann Publishers**:** 205-18.
Yobas, M. B., Crook, J., et al. (2000). "Credit scoring using neural and evolutionary techniques." IMA Journal of Mathematics Applied in Business and Industry **11**: 111-125.

Appendix A: Variables used in model construction

Application form variables

1 Age of applicant
2 Accommodation status
3 Council tax banding
4 Employment status
5 Employment type
6 Gross annual income
7 Home phone indicator
8 Marital status
9 Mortgage indicator
10 Number of credit cards
11 Number of dependents
12 Time at address
13 Time with bank
14 Time in current employment

UK credit bureau variables (same person)

15 Director indicator (Y/N)
16 Electoral roll confirmation indicator
17 MOSAIC postcode level classifier
18 Number of active credit accounts (excluding mail-order)
19 Number of active credit accounts (including mail-order)
20 Number of credit account status 8/9s
21 Number of credit searches in the last 3 months
22 Number of credit searches in the last 6 months
23 Number of CCJs/bankruptcies
24 Number of delinquent accounts
25 Number of settled good credit accounts
26 Number of settled good credit accounts in last 12 months
27 Outstanding balance on all active credit accounts (excluding mortgages)
28 Outstanding balance on all active credit accounts (mortgages)
29 Time registered on Electoral Roll at current address
30 Time since most recent CCJ/bankruptcy
31 Time since most recent credit account status 8/9s
32 Time since most recent delinquent account
33 Value of outstanding CCJs/ bankruptcy
34 Value of outstanding credit account status 8/9s
35 Value of delinquent accounts
36 Worst current arrears status on all active credit accounts
37 Worst arrears status last 6 months on all active credit accounts