International Journal of Computational Intelligence Research.
ISSN 0973-1873 Vol.2, No. 1 (2006), pp. 60-65

Computational Intelligent Techniques for Financial Distress Detection

S. Mukkamala[1], G. D. Tilve[2], A. H. Sung[3], B. Ribeiro[4], and A. S. Vieira[5]

[1,2,3]Department of Computer Science
New Mexico Tech, Socorro, NM 87801, U.S.A.
srinivas|gourav|sung@cs.nmt.edu

[4]Department of Informatics Engineering
University of Coimbra, P-3030-290 Coimbra, Portugal
bribeiro@dei.uc.pt

[5]ISEP and Computational Physics Centre
University of Coimbra, Coimbra, Portugal
armando.vieira@netcabo.pt

***Abstract*: In this paper we apply several computational intelligence techniques to the problem of bankruptcy prediction of medium-sized private companies. Financial data was obtained from Diana, a large database containing financial statements of French companies. Classification accuracy is evaluated for Linear Genetic Programs (LGPs), Classification and Regression Tress (CART), TreeNet, and Random Forests, Multilayer Perceptron (using Back Propogation), Hidden Layer Learning Vector Quantization and several gradient descent methods, conjugate gradient methods, the Levenberg-Marquardt algorithm (LM), the Resilient Backpropogation Algorithm (Rprop), and One Step Secant Method. We analyze 2 datasets, one is balanced and the other unbalanced. TreeNet has the best performance accuracy on unbalanced dataset and LGPs performs the best on balanced dataset. Scaled Conjugate Gradient performs the best among the neural network training functions used for the balanced dataset; and Resilient Back Propagation performs the best among the training functions used for the unbalanced dataset. Our results demonstrate the great potential of using computational intelligent techniques, as an alternative to discriminant analysis, in addressing important economics problems such as bankruptcy prediction.**

Keywords: financial distress detection, neural networks, linear genetic programs, classification and regression trees, random forests, treenet**.**

I. Introduction

Financial distress prediction is of great importance to banks, insurance firms, creditors, and investors, etc. The problem is stated as follows: given a set of parameters (mainly of financial nature) describing the situation of a company over a given period, predict the probability that the company may become bankrupted in the near future, say during the following year.

There has been considerable interest in using financial ratios for predicting financial distress in companies since the seminal work of Beaver [[1]] using univariate analysis and Altman approach with multiple discriminant analysis [[2]]. Despite its limitations [[3]], Multiple Discriminant Analysis (MLD) is still largely used as a standard tool for bankruptcy prediction. Nonlinear models, such as the Logit [[4]] and Probit [[5]], are used with caution as they only slightly improve the accuracy of MLD and may be sensitive to exceptions, common in this problem.

Bankruptcy prediction is a very hard classification problem as it is high-dimensional, most data distribution is non-Gaussian and exceptions are common [[6]]. A nonlinear classifier should be superior to a linear one due to saturation effects and multiplicative factors in the relationships between the financial ratios. For example, an increase of the earnings to total assets ratio from -0.1 to 0.1 is more relevant than an increase from 1.0 to 1.2. One the other the potential for default of a firm with negative cash flow is more amplified if it has large liabilities.

ANNs (Artificial Neural Networks, e.g. multilayer preceptrons) have been increasingly used for default prediction as they generally outperform many other existing methods [7-9]. More recent methods, such as Support Vector Machines, Genetic Algorithms and Genetic Programming have also been applied in this problem with success [10-12]. In general, all these approaches outperform Multiple Discriminant Analysis. However, in most cases the datasets used are very small (sometimes with less than 100 cases) and often highly unbalanced, which does not allow a fair comparison [13,14].

In this work we compare the efficiency of machine learning approaches and neural networks on bankruptcy detection using a large database of French private companies. This database is very detailed as it contains a large set of financial ratios spanning over a period of three years, corresponding to more than one thousand healthy and distressed companies. The approaches used are: Linear Genetic Programs (LGPs), Classification and Regression Tress (CART), TreeNet, Random Forests and Artificial Neural Networks: multilayer perceptron, Hidden Layer Learning Vector Quantization and several gradient descent methods, conjugate gradient methods, the Levenberg-Marquardt algorithm (LM), the Resilient Backpropogation Algorithm (Rprop), and One Step Secant Method.

The dataset used for analysis is described in section 2. An introduction to neural networks and the types of neural network training functions used are given in section 3. A brief introduction Linear Genetic Programs (LGPs) is given in section 4. CART and a tree generated for classifying normal vs. bankruptcy in Diana dataset is explained in section 5. TreeNet is briefly described in section 6. Random Forests are briefly explained in section 7. In section 8, we analyze classification accuracies of LGPs, CART, TreeNet, Random Forests and Neural Networks. Conclusions of our work are given in section 9.

II. Data Used for Analysis

We used a sample obtained from Diana, a database containing financial statements of about 780,000 French companies. The initial sample consisted of financial ratios of 2,800 industrial French companies, for the years of 1998, 1999 and 2000, with at least 35 employees. From these companies, 311 were declared bankrupted in 2000 and 272 presented a restructuring plan ("Plan de redressement") to the court for approval by the creditors. We decided not to distinguish these two categories as both signal companies in financial distress. The sample used for this study has 583 financial distressed firms, most of them small to medium size, with a number of employees from 35 to 400, corresponding to the year of 1999 - thus we are making bankruptcy prediction one year ahead. This dataset includes companies from a wide range of industrial sectors with 30 financial ratios defined by COFACE[1] and included in the Diana database.

Most companies on the verge of bankruptcy have heterogeneous patterns which are difficult to identify by any learning machine. To study the effect of unbalanced datasets, we randomly added healthy companies in order to get the following ratios of bankrupted to healthy firms: balanced dataset (50/50), and un balanced dataset (28/72). Lower ratios put stronger bias towards healthy firms, deteriorating the generalization capabilities of the network and increasing flase positives which are undesirable.

[1] Coface is a French credit risk provider

III. Neural Networks

The Hidden Layer Learning Vector Quantization (HLVQ) is an algorithm recently proposed for classification of high dimensional data [10,11]. HLVQ is implemented in three steps. First, a multilayer perceptron is trained using back-propagation. Second, supervised Learning Vector Quantization is applied to the outputs of the last hidden layer to obtain the code-vectors $\vec{w}_{ci}$ corresponding to each class c_i in which data are to be classified. Each example, $\vec{x}_i$, is assigned to the class c_k having the smallest Euclidian distance to the respective code-vector:

$$k = \min_j \left\| \vec{w}_{c_j} - \vec{h}(\vec{x}) \right\| \quad (1)$$

where $\vec{h}$ is a vector containing the outputs of the hidden layer and $\|\cdot\|$ denotes the usual Euclidian distance. In the third step the MLP is retrained but with two differences regarding conventional multilayer training. First the error correction is not applied to the output layer but directly to the last hidden layer being the output layer ignored from now on. The second difference is in the error correction back propagated to each hidden node:

$$E = \frac{1}{2}\sum_{i=1}^{N_h}\left(\vec{w}_{ck} - \vec{h}(\vec{x}_i)\right)^2 \quad (2)$$

where N_h is the number of hidden nodes. After retraining the MLP a new set of code-vectors,

$$\vec{w}_{c_i}^{new} = \vec{w}_{ci} + \Delta\vec{w}_{ci} \quad (3)$$

is obtained according to the following training scheme:

$$\Delta\vec{w}_{ci} = \alpha(n)(\vec{x} - \vec{w}_{c_i}) \text{ if } \vec{x} \in \text{class } c_i,$$

$$\Delta\vec{w}_{c_i} = 0 \quad \text{if } \vec{x} \notin \text{class } c_i \quad (4)$$

The parameter α is the learning rate, which should decrease with iteration n to guarantee convergence. Steps two and three are repeated following an iterative process. The stopping criterion is met when a minimum classification error is found.

The distance of given example $\vec{x}$ to each prototype is:

$$d_i = \left\| \vec{h}(\vec{x}) - \vec{w}_{c_i} \right\| \quad (5)$$

Which is a proximity measure to each class?

After HLVQ is applied, only a small fraction of the hidden nodes is relevant for the code-vectors. Therefore HLVQ simplifies the network thus reducing the risk of over fitting.

A variety of neural network training algorithms are used to detect the financial distress of the companies described in section 2 of this paper. Training algorithms used include, several gradient descent methods, conjugate gradient methods, the Levenberg-Marquardt algorithm (LM), the Resilient Backpropogation Algorithm (Rprop), and One Step Secant Method.

IV. Linear Genetic Programs

If you are using *Word*, use either the Microsoft Equation Linear Genetic Programming (LGP) is a variant of the genetic programming technique that acts on linear genomes [15]. The linear genetic programming technique used for our current experiment is based on machine code level manipulation and evaluation of programs. Its main characteristic, in comparison to tree-based GP, is that the evolvable units are not the expressions of a functional programming language (like LISP); instead, programs of an imperative language (like C) are evolved [15,16,17].

In the automatic induction of machine code by GP, individuals are manipulated directly as binary code in memory and executed directly without passing through an interpreter during fitness calculation. The LGP tournament selection procedure puts the lowest selection pressure on the individuals by allowing only two individuals to participate in a tournament. A copy of the winner replaces the loser of each tournament. The crossover points only occur between instructions. Inside instructions the mutation operation randomly replaces the instruction identifier.

In GP an intron is defined as part of a program that has no influence on the fitness calculation of outputs for all possible inputs. Fitness F of an individual program p is calculated as

$$F(p) = \frac{1}{nm}\sum_{j=1}^{n}\left(o_{ij}^{pred} - o_{ij}^{des}\right)^2 + \frac{w}{n}CE = MSE + wMCE \qquad (6)$$

i.e., the mean square error (MSE) between the predicted output (o_{ij}^{pred}) and the desired output (o_{ij}^{des}) for all n training samples and m outputs. The classification error (CE) is defined as the number of misclassifications. Mean classification error (MCE) is added to the fitness function while its contribution is determined by the absolute value of weight (w) [20].

V. Classification and Regression Trees

CART builds classification and regression trees for predicting continuous dependent variables (regression) and categorical predictor variables (classification) [18,19].

The decision tree begins with a root node t derived from whichever variable in the feature space minimizes a measure of the impurity of the two sibling nodes. The measure of the impurity or entropy at node t, denoted by i(t), is as shown in the following equation:

$$i(t) = -\sum_{f=1}^{k} p(w_j / t) \log p(w_j / t) \qquad (7)$$

Where p(wj | t) is the proportion of patterns xi allocated to class wj at node t. Each non-terminal node is then divided into two further nodes, tL and tR, such that pL , pR are the proportions of entities passed to the new nodes tL, tR respectively. The best division is that which maximizes the difference given in:

$$\Delta i(s,t) = i(t) - pi_L(t_L) - pi_R(t_R) \qquad (8)$$

The decision tree grows by means of the successive sub-divisions until a stage is reached in which there is no significant decrease in the measure of impurity when a further additional division **s** is implemented. When this stage is reached, the node **t** is not sub-divided further, and automatically becomes a terminal node. The class wj associated with the terminal node t is that which maximizes the conditional probability p(wj | t). No of nodes generated and terminal node values for each class are for the sample data set obtained from Diana, a database containing financial statements of about 780,000 French companies described in section 2, are presented in Table 1. Figure 1 and Figure 2 represent classification trees generated for datasets described in section 2 for classifying normal vs. bankrupt. Each of the terminal node describes a data value; each record is classifies into one of the terminal node through the decisions made at the non-terminal node that lead from the root to that leaf.

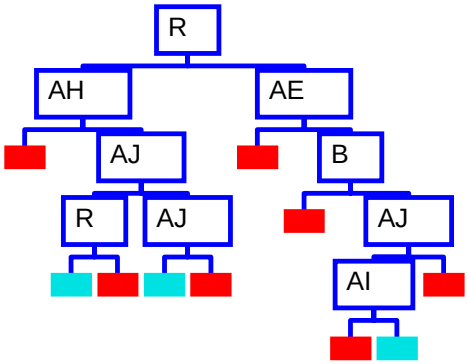

Figure 1. Tree for classifying normal vs. bankruptcy balanced dataset

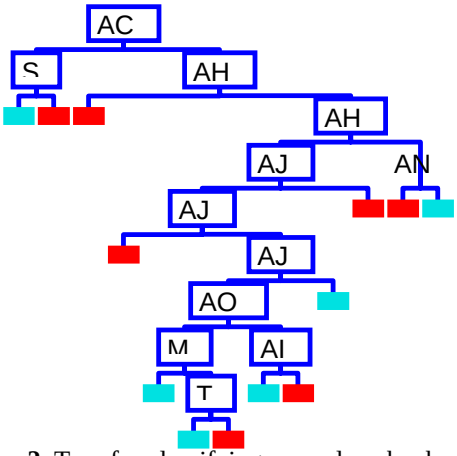

Figure 2. Tree for classifying normal vs. bankruptcy unbalanced dataset.

Class	No of Nodes	Terminal Node Value
Balanced Dataset	38	0.58
Unbalanced Dataset	52	0.63

Table 1. Summary of terminal nodes

VI. TreeNet

In a TreeNet model classification and regression models are built up gradually through a potentially large collection of small trees. Typically consist from a few dozen to several hundred trees, each normally no longer than two to eight terminal nodes. The model is similar to a long series expansion (such as Fourier or Taylor's series) - a sum of factors that becomes progressively more accurate as the expansion continues. The expansion can be written as [18,20]:

$$F(X) - F_0 + \beta_1 T_1(X) + \beta_2 T_2(X) ... + \beta_M T_M(X) \quad (9)$$

Where T_i is a small tree

Each tree improves on its predecessors through an error-correcting strategy. Individual trees may be as small as one split, but the final models can be accurate and are resistant to over fitting.

VII. Random Forests

A random forest is a classifier consisting of a collection of tree structured classifiers $\{\mathbf{h(x,\Theta_k), k=1, \ldots}\}$ where $\{\mathbf{\Theta_k}\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class of input $\mathbf{X}$. The common element in random trees is that for the **kth** tree, a random vector $\mathbf{\Theta_k}$ is generated, independent of the past random vectors $\mathbf{\Theta_1, \ldots\ \Theta_{k-1}}$ but with the same distribution; and a tree is grown using the training set and $\mathbf{\Theta_k}$, resulting in a classifier $\mathbf{h(x,\Theta_k)}$ where $\mathbf{x}$ is an input vector. For instance, in bagging the random vector $\mathbf{\Theta}$ is generated as the counts in $\mathbf{N}$ boxes resulting from $\mathbf{N}$ darts thrown at random at the boxes, where N is number of examples in the training set. In random split selection $\mathbf{\Theta}$ consists of a number of independent random integers between 1 and K. The nature and dimensionality of $\mathbf{\Theta}$ depends on its use in tree construction. After a large number of trees are generated, they vote for the most popular class [18,21].

The random forest error rate depends on two things:

- ✓ The ***correlation*** between any two trees in the forest. Increasing the correlation increases the forest error rate.
- ✓ The ***strength*** of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.

VIII. Results

We applied all methods to two datasets, balanced and unbalanced, respectively. Detection rates and false alarms are evaluated for Diana, a database containing financial statements of about 780,000 French companies and the obtained results are used to form the ROC curves. The point (0,1) is the perfect classifier, since it classifies all positive cases and negative cases correctly. Thus an ideal system will initiate by identifying all the positive examples and so the curve will rise to (0,1) immediately, having a zero rate of false positives, and then continue along to (1,1). In each of these ROC plots, the x-axis is the false positive rate, calculated as the percentage of normal companies considered as bankrupt; the y-axis is the detection rate, calculated as the percentage of bankrupt companies detected. A data point in the upper left corner corresponds to optimal high performance, i.e, high detection rate with low false alarm rate. Overall classification accuracies are summarized in Table 2. Figure 3 and Figure 4 show the ROC curves of the detection models of LGP, TreeNet and Random Forests for balanced dataset and unbalanced dataset.

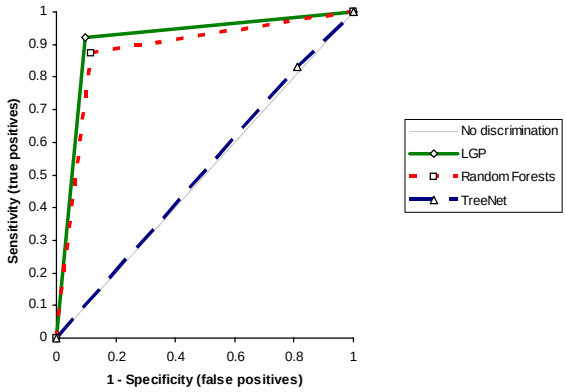

Figure 3. Classification accuracies of (LGP, RF, TreeNet) normal vs. bankruptcy (balanced dataset)

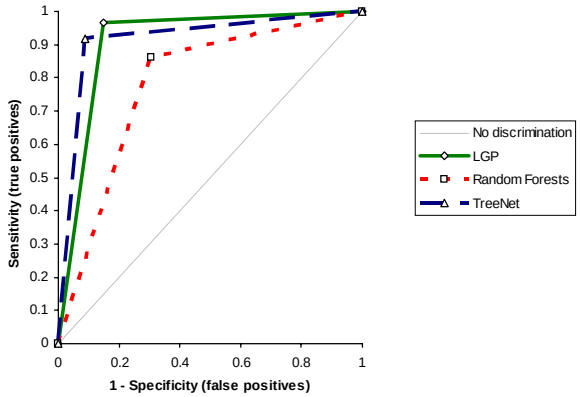

Figure 4. Classification accuracies of (LGP, RF, TreeNet) normal vs. bankruptcy (unbalanced dataset)

Classifier	Balanced Dataset	Unbalanced Dataset
Linear Genetic Programs	**91.2**	90.9
Classification and Regression Trees	85.5	87.0
TreeNet	50.9	**91.5**
Random Forests	87.7	77.7
Neural Network Training Functions		
Hidden Layer Learning Vector Quantization	77.03	83.06
Multilayer Perceptron (Back Propogation)	75.20	84.97
Scaled Conjugate Gradient	**83.63**	87.84
Powell-Beale Conjugate Gradient	83.41	87.91
Fletcher-Reeves Conjugate Gradient	83.23	87.42
Polak-Ribiere Conjugate Gradient	83.16	83.33
Resilient Back Propagation	82.22	**89.14**
One Step Secant Method	81.03	88.50
Gradient Descent	71.37	76.04
Gradient Descent with Momentum	72.46	73.83
Adaptive Learning Rate	80.33	81.01
By-Weight-and-Bias	72.0	74.64

Table 2. Summary of classification accuracies (%)

IX. Conclusions

Although the performance of all the methods used is largely comparable, we found that LGPs, CART, Scaled Conjugate Gradient and Resilient Back Propagation achieved consistently the best results.

TreeNet has the best performance accuracy on unbalanced dataset and LGPs performs the best on balanced dataset. Scaled Conjugate Gradient performs the best among the neural network training functions used for the balanced dataset and Resilient Back Propagation performs the best among training functions used for the unbalanced dataset.

For unbalanced samples the overall accuracy improves. However, false negatives (bankrupted companies misclassified as healthy), the more costly error, degrades in all machines including LGPs and CART. Therefore unbalanced samples should be avoided.

Bankruptcy prediction is an important, interesting but difficult problem and further investigation is still needed. As future work we plan to use a more complete data set including annual variations of important ratios from two or more years. Also, as more inputs are added, the feature selection problem will become more important and will need to be addressed.

Acknowledgment

A. S. Vieira thanks Fundação da Ciência and Tecnologia for financial support and João C. Neves for providing us with the datasets. The first and third, would like to acknowledge the support received from ICASA (the Institute for Complex Additive Systems Analysis, a division of New Mexico Tech).

References

[1] W. K. Beaver. “Financial Ratios as Predictors of Failure, Empirical Research in Accounting”, *Journal of Accounting Research*, X, pp. 71-102, 1996.

[2] E. I. Altman. “Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy”, *Journal of Finance*, XXII, pp. 589-609, 1968.

[3] R. A. Eisenbeis. “Pitfalls in the Application of Discriminant Analysis in Business, Finance and Economics”, Journal of Finance, XXII (3), pp. 875-900, 1977.

[4] D. Martin. “Early Warning of Bank Failure: A Logit Regression Approach”, *Journal of Banking and Finance*, pp. 249-276, 1977.

[5] C. Tan, H. A. Dihardjo. “Study on Using Artificial Neural Networks to Develop an Early Warning Predictor for Credit Union Financial Distress with Comparison to the Probit Model”, *Journal of Managerial Finance*, XXVII (4), pp. 56-77, 2001.

[6] C. Zavgren. “The Prediction of Corporate Failure: The State of the Art”, *Journal of Accounting Literature*, II, pp. 1-38, 1983.

[7] P. K. Coats, L. F. Fant. “Recognizing Financial Distress Patterns Using a Neural Network Tool”, *Financial Management* (autumn), pp. 142-155, 1996.

[8] F. Atiya. “Bankruptcy prediction for credit risk using neural networks: A survey and new results”, *IEEE Transactions on Neural Networks*, IV, pp. 12-16, 2001.

[9] G. Udo. “Neural Network Performance on the Bankruptcy Classification Problem”, *Computers and Industrial Engineering*, XV, pp. 377-380, 1995.

[10] A. Vieira, N. P. Barradas. "A training algorithm for classification of high dimensional data", Neurocomputing, XXXXX (C), pp. 461-472, 2003.

[11] A. Vieira, P. Castillo, J. Merelo. "Comparison of HLVQ and GProp in the problem of bankruptcy prediction", *In Proceedings of International Workshop on Artificial Neural Networks* (IWAN), pp. 665-662, 2003.

[12] A. S. Vieira, B. Ribeiro, S. Mukkamala, J. C. Neves, A. Sung. "On the Performance of Learning Machines for Bankruptcy Detection", *In Proceedings of IEEE International Conference on Computational Cybernetics (ICCC)*, pp. 323-327, 2004.

[13] J. S. Grice, M. T. Dugan. "The Limitations of Bankruptcy Prediction Models: Some cautions for the researcher", *Review of Quant. Finance and Account*, XVII (2), pp. 151, 2001.

[14] F. Cucker, S. Smale. "On the Mathematical Foundations of learning", *Bulletin of the American Mathematical Society*, XXXIX, (1), pp. 1-49, 2001.

[15] J. R. Koza. "*Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge", MA: The MIT Press, 1992.

[16] D. E. Goldberg. "*Genetic Algorithms in Search, Optimization and Machine Learning*", Reading, MA: Addison-Wesley, 1989.

[17] AIM Learning Technology, http://www.aimlearning.com.

[18] Salford Systems. TreeNet, CART, MARS, Random Forests Manual.

[19] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone. "*Classification and regression trees*", Wadsworth and Brooks/Cole Advanced Books and Software, 1986.

[20] J. H. Friedman. "Stochastic Gradient Boosting", *Journal of Computational Statistics and Data Analysis*, Elsevier Science, XXXVIII, pp. 367-378, 2002.

[21] L. Breiman. "Random Forests", *Journal of Machine Learning*, XXXXV, pp. 5-32, 2001.

Author Biographies

Srinivas Mukkamala received his Ph.D. in 2005 and M.S. in 2002, in Computer Science from New Mexico Tech; and Bachelor of Engineering (B.E.) in Computer Science and Engineering from University of Madras in 1999. As a Senior Research Scientist with the Institute for Complex Additive Systems Analysis (ICASA) of New Mexico Tech, he currently works in the areas of information assurance, network security, digital forensics, applied soft computing, data mining and bioinformatics.

Gourav D. Tilve received his Bachelor of Engineering in Electronics and Communications Engineering from Vishweswaraya Technological University in 2004. He is currently a master's student in Computer Science at New Mexico Tech. His research interests are information security, network security, computer networks and data mining.

Andrew H. Sung received his Ph.D. in Computer Science from the State University of New York at Stony Brook in 1984. He is currently a professor and chairman of the Computer Science department of New Mexico Tech.

Bernardete M. Ribeiro received her Ph.D. in Electrical Engineering from the University of Coimbra in 1993. She is currently an Associate Professor of the Department of Informatics Engineering at the University of Coimbra, Portugal.

Armando S. Vieira received his Ph.D. in Physics from University of Coimbra in 1996. He is currently with ISEP and Computational Physics Centre, University of Coimbra, Portugal.