On the Performance of Learning Machines for Bankruptcy Detection

♣A. S. Vieira, *B. Ribeiro, ♦S. Mukkamala, ♦A. H. Sung and +J. C. Neves
♣ISEP and Computational Physics Centre, University of Coimbra, P-3004-516 Coimbra, Portugal
*Department of Informatics Engineering, University of Coimbra, P-3030-290 Coimbra, Portugal
♦Department of Computer Science, New Mexico Tech, Socorro NM 87801, USA
+ISEG - School of Economics, Rua Miguel Lupi 20, 1249-078 Lisboa, Portugal

Abstract
Accurate prediction of the financial health of companies is an important task for stakeholders. In this work we apply several learning machines methods to the problem of bankruptcy prediction of medium-sized private companies. Financial data were obtained from Diana, a large database containing financial statements of French companies. Classification accuracy is evaluated with Artificial Neural Networks, Linear Genetic Programming and Support Vector Machines. We analyze both type I (bankrupted companies misclassified as healthy) and type II (healthy companies misclassified as bankrupted) errors on three datasets containing balanced and unbalanced class distribution. Linear Genetic Programming has the best accuracy in the balanced data while Support Vector Machines is more stable for the unbalanced dataset. Our results demonstrate the potential of using learning machines, with respect to discriminant analysis, in solving important economics problems such as bankruptcy prediction.

Keywords: *Support Vector Machines, Neural Networks, Logic Genetic Programming, Bankruptcy Prediction.*

1. Introduction

Financial distress prediction is of great importance to banks, insurance firms, creditors and investors. 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 a near future, normaly 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. Non-linear 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 approach 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, implemented by multilayer preceptrons, have been increasingly used to default prediction as they generaly outperform 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 [11][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) often highly unbalanced which does not allow a fair comparison [10].

In this work we compare the efficiency of four machine learning approaches on bankruptcy prediction using a large database of French private companies. This database is very detailed as it contains a wide set of financial ratios spanning over a period of three years, corresponding to more than one thousand of healthy and distressed companies. The approaches used are: Linear Genetic Programming, Support Vector Machines and Artificial Neural Networks in two versions: multilayer perceptrons and Hidden Layer Learning Vector Quantization.

This paper is organized as follows: Section 2 presents the Linear Genetic Programming method, Section 3 introduces support vector machines and Section 4 Artificial Neural Networks. Section 5 describes the dataset and Section 6 presents the results and discussion. Finally, Section 7 presents the conclusions.

[*] Corresponding author: Tel:+351-239790087, Fax: +351-239701266, email: bribeiro@dei.uc.pt

2. Genetic Programming

Genetic programming is a branch of genetic algorithms [13][14]. The main difference between genetic programs and genetic algorithms is the representation of the solution. Genetic programs create computer programs (like lisp) that represent the solution whereas genetic algorithms create a string that represents the solution.

The key steps of genetic programming in problem solving include:

- Generate an initial population of random compositions of the functions and terminals of the problem.
- Execute each program in the population and assign it a fitness value according to how well it solves the problem.
- Create a new population of computer programs by
  - copying the best existing programs
  - creating new computer programs by mutation
  - creating new computer programs by crossover
- The best computer program that appeared in any generation, the best-so-far solution, is designated as the result of genetic programming.

Fitness Function

The fitness function determines how well a program is able to solve the problem.

Functions and Terminals

The terminal and function sets are the alphabet of the programs to be made. The terminal set consists of the variables and constants of the programs. In the maze example of Ref. [13], the terminal set would contain three commands: forward, right and left. The function set consists of the functions of the program, including ordinary functions such as addition, subtraction, division, multiplication and more complex functions.

Crossover Operation

In the crossover operation, two parent solutions are combined to form two new solutions or offspring. The parents are chosen from the population according to a fitness function of the solutions.

Three basic methods exist for selecting the solutions for the crossover operation:

- Probability based selection. If $f(S_i(t))$ is the fitness of the solution S_i and

  $$\sum_{1\leq j\leq M} f(S_i(t))$$

  is the sum of all the members of the population, then the probability that the solution S_i will be copied to the next generation is

  $$\frac{f(S_i(t))}{\sum_{1\leq j\leq M} f(S_i(t))}.$$

- Tournament based selection. Typically the genetic program chooses two random solutions where the solution with the higher fitness wins. This method simulates biological mating patterns in which two members of the same sex compete to mate with a third member of a different sex.

- Rank based selection. Selection is based on the rank and not the numerical value of the fitness values of the solutions.

An important improvement that genetic programs display over genetic algorithms is its ability to create two new solutions from the same solution, thus increasing diversity.

Mutation

Mutation is an important feature of genetic programming. Two types of mutations are possible. In the first kind a function can only replace a function or a terminal can only replace a terminal. In the second kind an entire subtree can replace another subtree.

2.1. Linear Genetic Programming

Linear Genetic Programming (LGP) is a variant of the genetic programming technique that acts on linear genomes [10]. 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.

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.

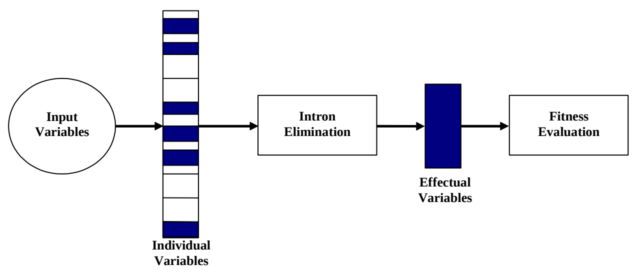


Fig. 1. Intron elimination in GP.

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

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

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) [15].

3. Support Vector Machines

Support Vector Machines (SVMs) [16][17] are a learning-by-example paradigm spanning through a wide range of classification, regression and density estimation problems. Many applications of SVM in science and engineering have been reported – see Ref. [18] for a review. This technique has its roots in statistical learning theory [16] and is an efficient approach to build optimum classifiers according a given criterion, usually the maximal margin criterion.

Outfitted with a sound mathematical background, SVMs are able to solve the problem of minimizing the complexity in learning while keeping a good generalization capability. This trade-off between complexity and accuracy leads to a range of principles that guarantee an optimal compromise. Cortes and Vapnik [17] have shown that the generalization is bounded by the sum of the training error and a term depending on the Vapnik-Chervonenkis (VC) dimension of the learning machine, leading to the formulation of the principle of Structural Risk Minimization (SRM). Good generalization can be achieved by minimizing above upper bound, which mainly depends on the classifier margin.

A brief review of Support Vector Classification (SVC) follows. For a detailed review see [16]. SVC was first proposed in pattern recognition for binary patterns where $y_i \in Y \equiv \{\pm 1\}$. The learning method uses input-output training examples from the data set $D = \{(x_i, y_i) \in X \subseteq \Re^N \times Y : 1 \le i \le l\}$ such that the decision function f classifies correctly test data ($\boldsymbol{x}$, y) that is generated from the same underlying probability distribution $P(\mathbf{x},y)$. In this framework, SVMs with Reproducing Kernel $k(\mathbf{.},\mathbf{.})$ finds the minimizer of:

$$\frac{1}{2}\sum_{i=1}^{l} V(y_i, f(x_i)) + \lambda \|f\|_{F_k}^2 \qquad (2)$$

where V is a loss function that measures the discrepancy of the interpolating function output $f(x_i)$ with respect to the given output y_i, F_k is the Reproducing Kernel Hilbert Space (RKHS) with Reproducing Kernel k and λ a positive parameter. The minimizer of (2) has the form:

$$f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i k(\mathbf{x}, \mathbf{x}_i) + b \qquad (3)$$

with $\alpha_i, b \in \Re$. The learning problem can be formulated as minimizing the function f using the loss function defined by $V(y_i, f(\mathbf{x}_i)) = |1 - y_i f(\mathbf{x}_i)|_+$. The equivalent quadratic programming problem originally proposed in [20] is:

$$\min_{f \in \mathbf{F}, \xi} \Phi(f, \xi) = \frac{C}{l}\sum_{i=1}^{l} \xi_i + \frac{1}{2}\|f\|_k^2 \qquad (4)$$

subjected to the constraints:

$$\begin{aligned} y_i f(x_i) &\ge 1 - \xi_i & i &= 1,\ldots,l \\ \xi_i &\ge 0 & i &= 1,\ldots,l \end{aligned} \qquad (5)$$

where C is the penalty constant (regularization parameter) and ξ the slack variable. The first term in Eq. (4) is the empirical error measured by $\sum_{i=1}^{l} \xi_i$. The second term controls the learning machine capacity measured in terms of the norm of f and corresponds to minimizing the machine VC-dimension.

Alternatively, the equivalent optimization problem, the so-called dual problem, needs to be solved whose solution in the binary classification is the solution of (5) under the indicated constraints.

$$\max_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \qquad (6)$$

with respect to α_i, where $0 \le \alpha_i \le \frac{C}{l}$, $i = 1,\ldots,l$ and $\sum_{i=1}^{l} \alpha_i y_i = 0$, the solution has again the form of Eq. (3). The input data points $\mathbf{x}_i$ for which α_i is non-zero are the so-called support vectors. The bias term b follows from the Karush-Kuhn-Tucker (KKT) conditions.

4. Neural Networks

The Hidden Layer Learning Vector Quantization (HLVQ) is an algorithm recently proposed for classification of high dimensional data [21][23]. 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 $\vec{w}_{c\,i}$ 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:

$$k = \min_j \left\| \vec{w}_{c_j} - \vec{h}(\vec{x}) \right\| \tag{7}$$

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 backpropagated to each hidden node:

$$E = \frac{1}{2} \sum_{i=1}^{N_h} \left(\vec{w}_{c\,k} - \vec{h}(\vec{x}_i) \right)^2 \tag{8}$$

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

$$\vec{w}_{c_i}^{\,new} = \vec{w}_{c\,i} + \Delta \vec{w}_{c\,i} \tag{9}$$

is obtained according to the following training scheme:

$$\begin{aligned} \Delta \vec{w}_{c\,i} &= \alpha(n)(\vec{x} - \vec{w}_{c_i}) \text{ if } \vec{x} \in \text{class } c_i\,, \\ \Delta \vec{w}_{c_i} &= 0 \qquad\qquad \text{if } \vec{x} \notin \text{class } c_i \end{aligned} \tag{10}$$

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:

$$d_i = \left\| \vec{h}(\vec{x}) - \vec{w}_{c_i} \right\| \tag{11}$$

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 overfitting.

5. Dataset and feature selection

5.1 Dataset

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.

5.2 Feature selection

In this work, we start by considering all the financial ratios produced by Coface. These ratios allow a very comprehensive financial analysis of the firms including the financial strength, liquidity, solvability, productivity of labor and capital, margins, net profitability and return on investment. Although, in the context of linear models, some of these variables have small discriminatory capabilities for default prediction, the non-linear approaches here used can extract relevant information contained in these ratios to improve the classification accuracy without compromising generalization.

Feature selection is an important issue in bankruptcy prediction, as in other problems where a large set of attributes is available. Of the large number of features available for predicting the financial health of a company, which are truly useful? Which of those features are significant and which are useless? These questions are relevant since elimination of useless features may enhance the accuracy of detection while reducing the amount of time in processing the data.

---

[1] Coface is a French credit risk provider

Table 1: Variables used for the problem.

#	**Variable definition**
1	Number of employees
2	Financial equilibrium ratio
3	Equity to Stable Funds
4	Financial autonomy
5	Current ratio
6	Collection period
7	Interest to sales (%)
8	Debt ratio
9	Financial Debt to Cash earnings
10	Working capital requirements in sales days
11	Value added per employee
12	Value added to assets
13	EBITDA margin
14	Margin before extra items and taxes
15	Return on equity
16	Value added margin
17	Percentage of value added for employees
18	Working capital to current assets

The feature selection problem for bankruptcy prediction is similar in nature to various engineering problems that are characterized by:

- Having a large number of input variables **x** = (x_1, x_2, …, x_n) of varying degrees of importance to the output **y**; it is known that i) some elements of **x** are essential while others less important; ii) some of components may not be mutually independent and iii) some may be completely useless or noise for determining the value of **y**.
- Lacking an analytical model that provides the basis for a mathematical formula that precisely describes the input-output relationship: **y** = *F* (**x**).
- Having available a finite set of experimental data, based on which a model is built for simulation and prediction.

Due to the lack of an analytical model, the relative importance of the input variables can only be estimated through empirical methods. A complete analysis would require examination of all possibilities, e.g., taking two variables at a time to analyze their dependence or correlation, then taking three at a time, etc. This, however, is both infeasible (requiring 2^n experiments!) and not completely error free since the available data may be of poor quality in sampling the whole input space.

Feature Selection with ANN

In a first step, highly correlated variables were automatically excluded. In a second step, elimination was based on the sensitivity of the neural network output to each variable.

The sensitivity of each variable x_i was then estimated by adding a small perturbation to the input $(x_1,...,x_i+\Delta x,...,x_n)$, and evaluating the correspondent variation of the output *y*. After repeating this evaluation for all data available we obtain an average $\bar{S}_i$ and standard variation, ΔS_i, of the sensitivity of each variable *i*.

The average sensitivity measures the linear and additive influence of the variables to the output. However, this quantity does not take into account non-linear influences and possible interactions with other variables. These interactions may be identified with the standard deviation of the sensitivity. Thus variables with *both* $\bar{S}_i$ and ΔS_i small were eliminated.

From the 30 initial ratios we select the 18 most relevant and normalized them to zero mean and unity variance. Table 1 presents the eighteen variables used for prediction.

Feature ranking by LGP

Of the features selected by ANN, table 2 presents the ranking obtained using the LGP feature ranking algorithm [11] which measures the relevance of each variable for output prediction. In the top five, for all datasets, we have value added to assets, percentage of value added for employees and debt ratio which is consistent with other studies – for example see Ref. [2].

Table 2: Feature important as ranked by LGP.

Feature Rank	Dataset 1	Dataset 2	Dataset 3
1	12	12	6
2	17	8	17
3	8	17	1
4	1	6	8
5	4	4	12
6	18	18	19
7	19	9	7
8	6	3	10
9	3	5	11
10	10	1	3
11	16	7	18
12	2	19	13
13	5	2	2
14	9	13	4
15	13	11	9
16	15	14	15
17	7	16	16
18	11	10	5
19	14	15	14

5.3 Error analysis

There are two types of errors for this classification problem: type I error and type II error. Type I error is the number of cases classified as healthy when they are really bankrupted, N_{01}, divided by the number of bankrupt companies N_1:

$$e_I = \frac{N_{10}}{N_1}. \qquad (12)$$

Type II error is the number of companies classified as bankrupt when in reality they are healthy, N_{01}, divided by the total number of healthy companies, N_0:

$$e_{II} = \frac{N_{01}}{N_0}. \qquad (13)$$

The total error is just:

$$e_{Total} = \frac{N_{10} + N_{01}}{N_0 + N_1}. \qquad (14)$$

For a balanced dataset with $N_0 = N_1$, the total error is average of both errors. The accuracy is defined as 1-e_{Total}.

Most companies on the verge of bankruptcy have heterogeneous patterns which are difficult to identify by any learning machine. Therefore type I error is in general higher than type II. Since the cost associated with this type of error is in general higher, in real applications global accuracy may not be the best performance indicator of the algorithm.

To study the effect of unbalanced datasets, we randomly added healthy companies in order to get the following ratios of bankrupted to healthy firms: dataset 1 (50/50), dataset 2 (36/64) and dataset 3 (28/72). Lower ratios put stronger bias towards healthy firms, deteriorating the generalization capabilities of the network and increasing type I error which is undesirable.

6. Results

We applied all methods to three datasets, one balanced (dataset 1) and two unbalanced – dataset 2 and 3. We used 10-fold cross validation to evaluate their generalization errors.

6.1 LGP

Table 3 show the accuracy obtained with LGP. On the balanced dataset both error types are similar, but on the unbalanced dataset error I is very large, although the overall accuracy is above 80%.

Table 3: Performance Accuracy of LGP

Dataset	LGP Best Program		LGP Best Team	
	Accuracy (%)		Accuracy (%)	
1	78.88	81.69	77.25	78.96
2	84.50	85.42	84.00	83.65
3	86.44	87.29	85.75	86.04

6.2 SVM

For SVMs we have to select the best model for each dataset. The common kernel functions, e.g., Gaussian, polynomial and sigmoidal, tend to project data onto a high dimensional space, often increasing the risk of overfitting. This can be controlled using the capacity of the machine through the maximization of the margin and hyperparameter selection. While the choice of the kernel still remains a research issue, an RBF Gaussian kernel has been used since good results in many practical problems have been reported so far.

Using a grid search (see Figure 2) we seeks the optimum values of the constraint penalty for the method's solution and the kernel width (C,γ) has been performed. The best model parameters have been obtained for the three data sets under test as shown in Table 4. Model selection has thus been accomplished avoiding the costly trial and error procedure [21].

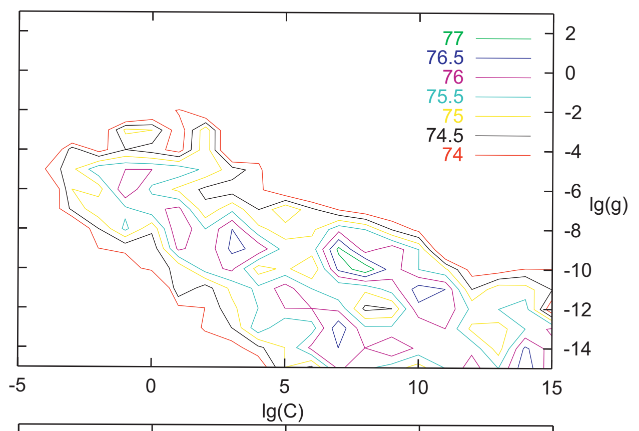
Table 4: Best SVM parameters for the three datasets.

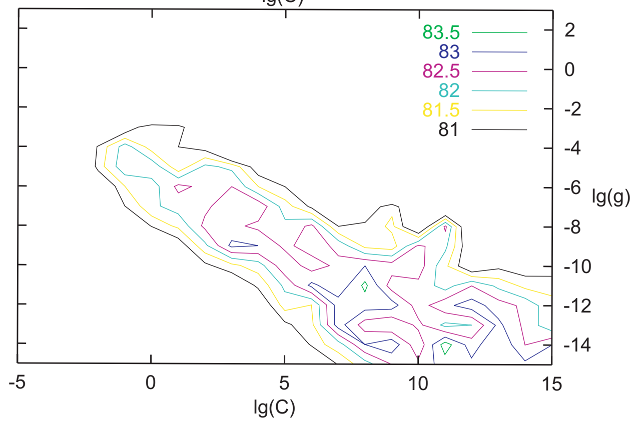
Parameters	**C**	γ
Best model 1	2^0	2^{-10}

Best model 2	2^8	2^{-11}
Best model 3	2^8	2^{-9}

Results of the performance accuracies and type I and II errors attained by using the SVMs as learning machines are presented in tables 5 and 6.

Receiver Operating Characteristics (ROC) curves are used to evaluate the classifiers' performance and provide information on the trade-off between the hit rate or true positives, and the false alarm rate or false positives – Fig. 3. In this application negative examples abound while positive cases are rare and uncharacteristic. The ROC curves were obtained using this set of positive and negative examples [24]. To circumvent the bias induced by unbalanced data, weighting factors have been used in the SVMs algorithm [16].





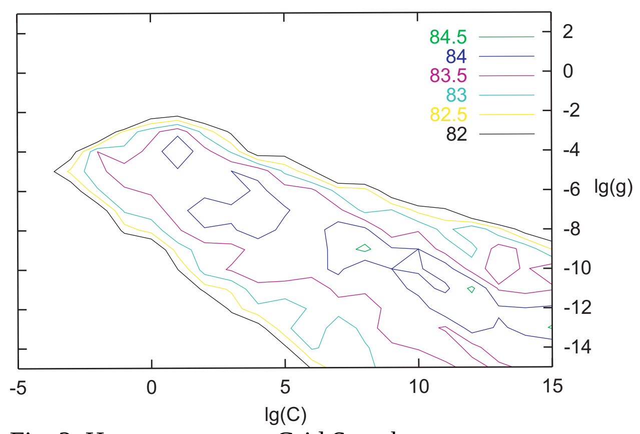


Fig. 2. Hyperparameters Grid Search:
$C = 2^{-5} \cdots 2^{15}, \gamma = 2^{-15} \cdots 2^{1}$

Since, in many cases, we have more healthy than bankrupted companies, weighting factors were introduced in the SVMs algorithm. To assess the performance of the classifiers, probabilistic outputs for SVMs, which require the class probabilities, have been calculated and ROC curves computed – Fig. 3.

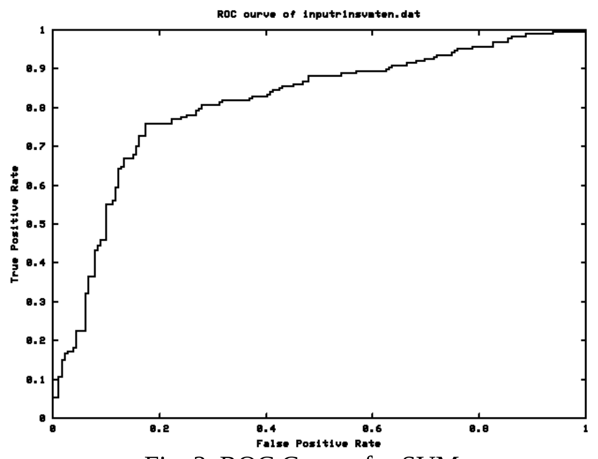


Fig. 3. ROC Curves for SVM.

Table 5: Error type I and II for all methods used.

	Data set 1		Data set 2		Data set 3	
	Error I	Error II	Error I	Error II	Error I	Error II
LGP	25.13	16.75	42.22	3.37	32.08	4.33
SVM	29.95	18.99	34.44	13.28	36.57	12.03
HLVQ	25.55	20.39	33.84	10.75	39.80	8.05
MLP	27.75	21.86	35.95	12.83	36.10	6.83

Table 6: Overall Accuracy

	Data set 1	Data set 2	Data set 3
LGP	79.06	82.64	87.90
SVM	75.53	79.10	81.10

HLVQ	77.03	80.94	83.06
MLP	75.20	78.85	84.97

6.3 Neural Networks

Multilayer Perceptrons (MLP) containing a single hidden layer from 5 to 20 nodes were tested in this problem. The best performing set was a hidden layer of 15 neurons trained by backpropagation with a learning rate of 0.1 and a momentum term of 0.25.

HLVQ was applied upon this MLP with a very fast convergence - only 8 iterations. Results obtained with MLP and HLVQ are presented in table 5 and 6.

Fig 4 presents the code vectors obtained by HLVQ corresponding to the two categories: healthy and bankrupt companies. Note that of the total 15 components, five are very similar, thus redundant. The remaining ten components are the effective features used by HLVQ to classify data.

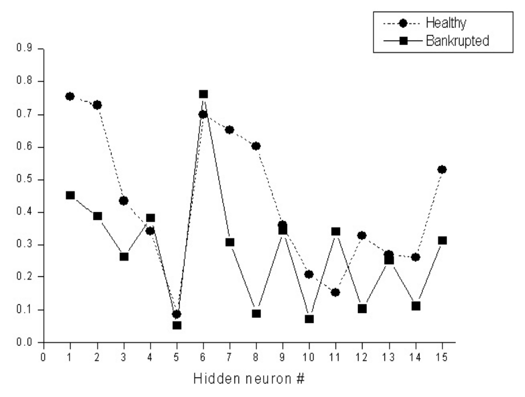


Fig. 4: HLVQ code-vectors

7. Discussion and Conclusions

Although the performance of the four methods are comparable in all datasets, we found that LGP achieved consistently the best results. Hidden Layer Learning Vector Quantization algorithm performs very closely to Support Vector Machines and is more robust than multilayer perceptrons.

For unbalanced samples the overall accuracy improves. However, error type I, the most costly for banks, degrades in all machines including LGP. Therefore unbalance samples should be avoided.

Bankruptcy prediction is an important, interesting but difficult problem and further investigation is still needed. As a future work we plan to use a more complete data set including annual variations of important ratios from two or more years. As more inputs are added, feature selection will have to follow a more stringent scrutiny.

Acknowledgements

A. S. Vieira thanks Fundação da Ciência and Tecnologia for financial support. The second, third, and fourth authors would like to acknowledge the support received from ICASA (the Institute for Complex Additive Systems Analysis, a division of New Mexico Tech). The work of the second author was performed during a sabbatical visit to the Computer Science Department of New Mexico Tech.

References

[1] W. K. Beaver, Financial Ratios as Predictors of Failure, Empirical Research in Accounting: Selected Studies, 1966, supplement to volume 5, Journal of Accounting Research (1996) 71-102.

[2] Altman, E. I. Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy, Journal of Finance, 23 (1968) 589-609.

[3] R. A. Eisenbeis, Pitfalls in the Application of Discriminant Analysis in Business, Finance and Economics, Journal of Finance, 32 (3), June, (1977) 875-900.

[4] D. Martin, Early Warning of Bank Failure: A Logit Regression Approach, Journal of Banking and Finance, 1 (1977) 249-276.

[5] C. Tan, and 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, Managerial Finance, 27 (4), (2001) 56-77.

[6] C. Zavgren, The Prediction of Corporate Failure: The State of the Art, Journal of Accounting Literature, 2 (1983) 1-38.

[7] P.K. Coats and L.F. Fant, Recognising Financial Distress Patterns Using a Neural Network Tool, Financial Management (Autumn), (1996) 142-155.

[8] F. Atiya, Bankruptcy prediction for credit risk using neural networks: A survey and new results, IEEE Trans. Neural. Net., 4 (2001) 12-16.

[9] G. Udo Neural Network Performance on the Bankruptcy Classification Problem, Computers and Industrial Engineering, 25 (1993) 377-380.

[10] J. S. Grice and M. T. Dugan, The limitations of bankruptcy prediction models: Some cautions for the researcher, Rev. of Quant. Finance and Account., 17 no. 2 (2001) 151.

[11] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and Its Applications. Morgan Kaufmann Publishers, Inc., 1989.

[12] B. Back, T. Laitinen, K. Sere, M. V. Wezel, Choosing Bankruptcy Predictors Using Discriminant Analysis, Logit Analysis, and Genetic Algorithms, Turku Centre for Computer Science, Technical Report N. 40, ISBN 951-650-828-6, ISSN 1239-1891, 1996.

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

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

[15] Brameier, W. Banzhaf, A Comparison of Linear Genetic Programming and Neural Networks in Medical Data Mining, IEEE Transactions on Evolutionary Computation 5 (2001) 17-26.

[16] V. Vapnik, The Nature of Statistical Learning Theory. New York: Springer Verlag, 1995.

[17] C. Cortes and V. Vapnik, Support vector networks, Machine Learning 20 (1995) 273-297.

[18] V. David Sánchez, Advanced support vector machines and kernel methods, Neurocomputing – Special Issue on Support Vector Machines 55 (2003) 5-20.

[19] F. Cucker and S. Smale, On the mathematical foundations of learning, Bulletin of the American Mathematical Society 39, n 1 (2001) 1-49.

[20] N. Cristianini and J. Shawe-Taylor, Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, 2000.

[21] C.-C. Chang and C.-J. Lin, LIBSVM: a library for support vector machines, Technical Report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2000.

[22] A. Vieira and N. P. Barradas, A training algorithm for classication of high dimensional data Neurocomputing, 50C, (2003) 461-472.

[23] A. Vieira, P. Castillo and J. Merelo, Comparison of HLVQ and GProp in the problem of bankruptcy prediction, IWANN03 - International Workshop on Artificial Neural Networks, L.J. Mira, ed., Springer-Verlag (2003) 665-662.

[24] J. P. Egan, Signal detection theory and ROC analysis. New York, Academic Press, 1975.

[25] T. Evgeniou, M. Pontil, and T. Poggio, Regularization networks and support vector machines, Advances in Computational Mathematics 13, n. 1 (2000) 1-50.