New Evolutionary Classifier Based on Genetic Algorithms and Neural Networks: Application to the Bankruptcy Forecasting Problem

M.A. Esseghir

New Evolutionary Classifier Based On Genetic Algorithms And Neural Networks: Application To The Bankruptcy Forecasting Problem

M A Esseghir

Faculty of Sciences of Tunis, Computer Science Department, Campus Universitaire, 1060 Tunis, Tunisia

ABSTRACT

Artificial neural networks (ANNs) have been widely applied in data mining as a supervised classification technique. The accuracy of this model is mainly provided by its high tolerance to noisy data as well as its ability to classify patterns on which they have not been trained. Moreover, the performance of ANN based models mainly depends both on the ANN parameters and on the quality of input variables. Whereas, an exhaustive search on either appropriate parameters or predictive inputs is very computationally expensive. In this paper, we propose a new hybrid model based on genetic algorithms and artificial neural networks. Our evolutionary classifier is capable of: selecting the best set of predictive variables, then, searching for the best neural network classifier and improving classification and generalization accuracies. The designed model was applied to the problem of bankruptcy forecasting, experiments have shown a very promising results for the bankruptcy prediction in terms of predictive accuracy and adaptability.

KEYWORDS: Supervised learning; Artificial neural networks; Genetic Algorithms; Bankruptcy prediction.

INTRODUCTION

Artificial neural networks (**ANNs**) have been widely applied in data mining as a supervised classification technique [1]. Nevertheless, the performance of neural networks (**NNs**) greatly depends on both the problem at hand and on the parameters chosen for it. Neuroevolution searches through the space of behaviors for a network that performs at well for a given task. In fact, **NNs** can be combined with genetic algorithms in order to improve classification accuracy. This hybrid approach, aims to solve complex control problems and represents an exiting alternative as well to conventional learning classifiers as to statistical techniques [2]. Our model consists of a hybrid system which can respond to the following requirements: retrieving the best set of attributes (financial ratios) from a given data set; searching the best **NN** classifier; improving and refining the retained **NN** classification and generalization accuracies. In addition, the problem of efficient bankruptcy prognosis is of great interest both to scientists and practitioners [3, 4]. In fact, owners, managers, investors, creditors and business partners, as well as governmental institutions, have an interest in assessing the financial position of a firm and its propensity for bankruptcy. Besides, the proposed models for bankruptcy prediction unanimously confirm the growing interest to **ANN** approaches [5, 6, 7]. The reminder of this paper is organized as follows: section 1 presents **ANNs** as a learning and a classification techniques. Next, we detail the genetic evolutionary optimization process. In section 3, we survey models developed to forecast firm failure. Section 4 details our model and its two components. Section 5 is dedicated to the assessment of the proposed model and presents the results of the conducted experiments. Finally, we present a summary and we sketch avenues of future research.

Email: M A Esseghir mohamedemir@gawab.com

1 ARTIFICIAL NEURAL NETWORKS

Artificial neural networks are stochastic models that simulate the brain's ability in learning, control and pattern recognition. The decisions taken are always based on previosly learned experiences. In fact, ANNs attempt to model brain's biological structures both in architecture and operations. Most types of **NNs** can be covered by the following definitions:

Definition 1:

A NN is a system composed of many simple processing elements operating in parallel whose function is determined by a network structure, connection strengths and the processing performed at computing elements or nodes[1].

Definition 2 [8]:

According to Haykin, a **NN** is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It mimics the human brain in two aspects:

[1] DARPA Neural Network Study (1988, AFCEA International Press, p. 60)

- Knowledge is acquired by the network through a learning process.
- Interneuron connection strengths known as synaptic weights are used to store the knowledge.

The *neuron* or *unit*, also called *processing element* performs a relatively a simple job: it receives inputs from neighbors and external sources and use them to compute an output signal which is propagated to units in the next layer. The computation procedure is performed in two steps. First, inputs are weighted and summed. Then, the result is used as a parameter for a singular valued function: *the activation function.* The activation function is as follows:

$$y_i = f(h_i) = f(\sum_{j=1}^{n} w_{i,j} x_i + \theta_j) \quad (1)$$

The parameter θ_j is called a bias and can be interpreted as a weight applied to a pseudo input which is generally clamped to a constant value.

The most widely used activation function is the *logistic sigmoidal function*:

$$f(h_i) = (1/(1 + e^{-h_i})) \quad (2)$$

that maps a large input domain into the range of $[0, 1]$. The logistic function is non linear and differentiable, allowing the learning algorithms to model classification problems that are linearly inseparable.
Inputs are propagated through the network, layer by layer, using equations (1) and (2), until reaching the output layer. The decision corresponding to the input vector is provided by the values of the output nodes of the trained ANN.
The next paragraph illustrates how the ANN can learn from examples, by presenting one of the common supervised learning procedure: *the Backpropagation.*

The Backpropagation Algorithm

Considered as the most popular **NN** algorithm, the Backpropagation (**BP**) algorithm was proposed in 1986 by Rumelhart *et al.* [9]. The **BP** performs learning on a multi-layer feed-forward **NN**, through an iterative process of a set of training samples. Each input vector passes through the network until it reaches the output layer where it yields an output vector. The input propagation is called the *"forward"* step. The network prediction derived from the output vector, is then compared the desired output. For each training sample, the weights are adjusted so as to minimize the mean squared error between the desired and the obtained outputs. The corrections, to fit the desired output, are performed in the *"backward"* direction, *i.e*, from the output layer down to the first hidden layer. The process is iteratively applied on the training samples until the satisfaction of the stopping criterion.

The accuracy of ANN model is mainly provided, by its high tolerance to noisy data as well as its ability to classify patterns on which they have not been trained [10].

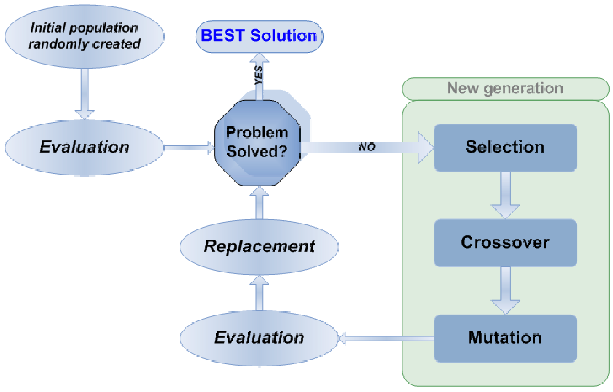

Figure 1: A common genetic evolution process

2 GENETIC ALGORITHMS

Genetic algorithms (**GAs**) are artificial intelligence global search techniques. **GAs** attempt to apply evolutionary techniques to the field of the problem solving notably in function optimization [11]. They been have proven to be valuable in searching large and complex problem spaces [1], especially *NP-hard* problems. A GA process is based on natural selection, crossover and mutation that are repeatedly applied to a population of potential solutions, simulating the *Darwinian evolution* [11]. Each solution is encoded into a binary string called *chromosome.* Over generations, a number of a new solutions is generated, with crossover and mutation operations, and the best solutions are selected. This process iteratively continues until the obtention of a good solution. A common genetic evolution process is sketched by Figure 1.

GAs will be used and referenced, at different levels, in our paper. Each of them tries to tackle a combinatorial optimization problem.

3 BANKRUPTCY FORECASTING MODELS

Numerous models have been developed to predict firms insolvencies [12]. In the reminder, we only put the focus on those related to both NN and hybrid models.

3.1 Neutral networks models

The application of **ANN** in the study of the bankruptcy problem started since 1990 and became a dominant classification research methodology [3]. A variety of studies have been done on the topic of **NN** and bankruptcy prediction. In the following, we survey the **NN** bankruptcy prediction models according to the **NN** type.

3.1.1 MLP-BP

The MLP-BP is a multilayer perceptron (MLP) neural network with a feed-forward architecture and backpropagation as a training algorithm. The MLP-BP is very popular and was successfully applied to a wide range of classification problems [13, 14]. The majority

of bankruptcy studies, that consider ANN as a predictive model, use both multi-layer feed-forward NN as an architecture and **BP** or one of its derivative as a learning algorithm. B. Back *et al.* [5] use networks with one output node. Each firm presented to the network will be classified into a bankrupt or solvent class, according to the activation value of the output node. A varying number of hidden nodes was used in networks trained with 10000 epochs. Four models were tested, using different sets of input ratios as NN input. With 38 training firms samples and the same number of firms on the test set, experiments showed that the near optimal architecture was with five hidden nodes. The first **NN** model achieved a prediction accuracy with 94.47%. Finally, all the four models were tested with one, two and three years old data before bankruptcy. All the four models gave their best results as well with one year before bankruptcy as with the first model based on ratios.
Li *et al.* [6] compared the predictive capabilities of five models: Two statistical techniques based on logistic regression (Logit and Probit), and three feed-forward **NN** architectures. The **BP** learning algorithm is replaced by a global search procedure to avoid rapid convergence in local minima. The GA was used to catch out the near optimal set of **NN** weight values. **GA** was introduced by the authors to search for the global optimum and avoid problems of local minimas caused by the BP algorithm. Two models were of use: one with four input variables, and the other, with six input variables. As a result, the **NN** consistently outperform the other models with an average prediction accuracy around 90% in the fourth variable model, and with 91.96% of accuracy for the six variables model in the training sample. In the test set, the NN generates less out-of-sample errors than probit does with 86.66% and 75.5% of accuracy respectively for the first and the second model. Logit, probit and maximum score outperform **NNs** with four hidden layers. However, when the **NN** was trained with six hidden nodes, it performs the same accuracy than the other models.

3.1.2 Kohonen Self-Organizing-Maps

Self-Organizing-Maps (**SOM**) also known as Kohonen networks [14, 15], are usually composed of two layers: an input layer of N nodes and an output layer. output neurons are usually organized into a two-dimensional grid. The **SOM** tries to separate outputs into categories, called *clusters*. During the training process, input patterns are presented to the network in turn, and all the nodes calculate their activation values.
The winning node, also known as *best matching unit*, and some of the nodes in the neighborhood are then allowed to adjust their weights vector to match the current input vector more closely. Consequently, the winner is the output unit that is the closest to the input vector. The size of the winner's neighborhood is varied throughout the training process. The network has no prior knowledge about the target outputs and the training is characterized by a competition between the output units: the Kohonen learning is unsupervised and competitive [15]. As application of the SOM, Back *et al.* [5] use it as a model to predict the future fail of companies. The data sets and the input variable models are the same as those described in the first example of the previous section. The achieved results vary from 76.31% to 92.1% of accuracy. The best results were achieved with a grid of (3×3) grid using a model based on input ratio variables. Using the optimal network architecture, data from one to three years before the possible bankruptcy were tested. The authors concluded that the used network was unable to predict firm's bankruptcy two and three years ahead.

3.1.3 Support Vector Machines

Support Vector machines (**SVMs**) were introduced by Vapnik in 1995 and numerous studies have applied them in pattern recognition [16]. SVM classification finds hyper-planes in the possible space for maximizing the distance from the hyperplane to the data points, which is equivalent to solving a quadratic optimization problem. A SVM captures the geometric characteristics of feature space without deriving weights of networks from a training data. SVM was applied by Shik in 2005 to predict Korean firms' bankruptcy [17]. The data set contains firm ratios from 1996 to 1999. A statistical scoring method, Multivariate Discriminant Analysis (MDA) was applied in a preprocessing stage to select a set of 10 input variables from 52 ratios. The obtained results showed that the generalization performance of SVM is better than that of ANN trained with the backpropagation algorithm. The learning accuracy on the training set reaches 100% and the generalization performance was around 88%. However, the performance of the model closely depends on the best choice of the model parameters.

3.2 Hybrid models based on feature selection

By targeting more accurate results and less prediction errors, several methodologies have attempted to take advantage of more than one model's possibilities. Lee *et al.* [18] proposed two hybrid models: MDA-assisted NN and ID3-assisted Neural Network. These models are based on a NN operating with the input variables selected, respectively, by the MDA and ID3 [19]. The intention underlying these hybrid models is that MDA or ID3 was used as preprocessing mechanism for selecting most predictive input variables. The data sample was based on the Korean firms failing in the period 1979-1992. Initially, fifty seven financial variables were selected as significant in predicting bankruptcy. The preprocessing stage yields with 10, 18, 17 and 7, 7, 9 financial ratios respectively with MDA and ID3 respectively. The results of the MDA was 68%, 68.57%, and 70% of accuracy. However, the MDA-assisted NN showed more accurate results with 70%, 80%, and 80%. Thus, hybrid model perfiormances outperform those obtained by each model apart. Compared to MDA, hybrid model the ID3-

assisted NN is more powerful. Besides, it realizes such performance with a fewer number of input variables.

Back *et al.* proposed in [7] a comparison between GAs, logistic regression (Logit) and discriminant analysis. The three models were applied to search the best bankruptcy predictors. The study showed that the three techniques lead to different failure prediction models. The best results were found with GAs evolving a NN as a wrapper.

4 THE PROPOSED MODEL

Numerous studies have showed that ANNs can be an alternative methodology for classification problems to those statistical methods have been of extensive use. [3, 5, 6]. Nevertheless, numerous theoretical and experimental studies carried out on supervised learning reported several drawbacks in building and using the backpropagation [10, 20, 21]. Adding to this, the fact that there is no automatic way to design the **NN** architecture for a given classification problem. Aiming to give the best of the neural network classification capabilities, we introduce a new model that is able to: design near optimal **NN** architecture, look for a best set weight values yielding high learning and classification accuracies, and provide a compact set of relevant features as a **NN** input.

Believing that combining classifiers and boosting methods can lead to improvements in performance, we propose, in this article, a new hybrid model based on mainly two stochastic techniques, which are combined at different levels of the proposed model. The following paragraphs illustrate how **ANNs** and **GAs** are combined and applied to improve classifiers performance accuracies.

The model to present in this study consists in two-stage evolutionary optimization processes. The first step selects the best set of inputs having a predictive relationship to the target. It will be done using the first sub-model: Input Selection Model (**ISM**). The purpose of the ISM is to provide, from the whole set of available features, a relevant set of features discarding noisy ones. The second step consists of the design and the optimization of the classifier based on NNs. The Neural Network Optimizer (**NNO**) chooses the best architecture of the neural networks using the inputs selected by the first sub-model **ISM**.

4.1 The Input Selection Model

At this level, **ISM** plays the role of a feature selection method. In fact, ISM aims to select a reduced feature subset to feed the **NN** inputs in the second stage. The selection of a relevant subset of features is a combinatorial problem involving a search within 2^n possible subsets. **ISM** uses a **GA** as a global search procedure and a NN as a wrapper for the evaluation of retained subsets. The GA designed by **ISM** is based on the standard algorithm of Goldberg [11]. We apply it to select a compact set of input variables for the prediction of firm bankruptcy. A set of candidate solutions are evolved through a fixed number of generations.

Hence, possible solutions to the problem are encoded as binary strings, where each bit corresponds to a feature. When a bit is set to '1', it means that the variable corresponding to this position is selected to build a NN, otherwise, the variable is not retained. The pseudo-code of the ISM is illustrated by *Algorithm 1.* In our case, dataset variables are represented by financial ratios.

4.1.1 Initialization

An initial set of solutions is randomly generated. For each individual, a number of ratios is randomly selected by setting their corresponding bits to 1. Once the initial population is created, the evaluation process starts to evaluate the classification accuracy of the retained subsets (*c.f Algorithm 2*). A fitness value is assigned to each chromosome. The first generation of solutions is derived by applying a tournament selection to the evaluated set.

4.1.2 Evaluation

At this level newly generated chromosomes are evaluated to acquire a fitness value and the wrapper plays, here, a key role to evaluate features corresponding to each chromosome. There are two steps that must be performed to evaluate each chromosome. First, a NN, with the selected variables in the chromosome as input, is built and trained. Next, the trained network is evaluated using a test set which is constituted of unseen data at the training stage. The predictive power of a ANN is assessed on the basis of both the number of Correctly Classified Instances (CCI) and the MSE obtained on the test set. The chromosome evaluation assesses the predictive generalization ability of the NN and consequently of the set of involved ratios. The idea of the use of the test set in the evaluation was retrieved from the work of Back *et al.* [7]. Besides, Aleixandre *et al.* [22] used the number of incorrect classified instances in the evaluation of the training set. In fact, and as illustrated by equation (3), our fitness involves two evaluation criteria: the proportion of incorrectly classified instances and the mean square error in the test set.

$$fitness = (ICI + TMSE)/2 \qquad (3)$$

equation (3), ICI and $TMSE$ denote, respectively, the percentage of incorrectly classified instances and the mean square error found on the test set.

Input:
S: Set of features
N_i: initial population size
N: population size
t: tournament size
p_{mut}: mutation probability
p_{cross}: Crossover probability
Maxgen: number of generations
h: number of hidden nodes
it: number of training iteration
η: learning rate
m: moment value
Output: S1: Best subset of features
Begin
  Population P_0, P,P_{tmp}
  P_0=P=P_{tmp}= $\emptyset$;
  P_0=GenerateInitialPopulation(N_i)
  Evaluate (P_0, h, it, η, m)
  P=Select (P_0, N,t)
  i=0
  While $i <$ *Maxgen* **do**
    P_{tmp}=Select (P_0, N,t)
    Crossover(P_{tmp}, p_{cross})
    Mutate(P_{tmp}, p_{mut})
    Evaluate(P_{tmp}, h, it, η, m)
    Replace(P_{tmp}, P)
    i=i+1
  S1=P.bestChromosome().decode()
  Return(S1)
End

Algorithm 1: Input Selection Model Algorithm (ISM)

Input:
P: population
h: number of hidden nodes
it: number of training iteration
η: learning rate
m: moment value
Output: P: Population evaluated
Begin
  Foreach *Chromosome ch* $\in$ *P* **do**
    I=ExtractRatiosIndexes(ch)
    TestSet=GenerateTestSet(I)
    TrainSet=GenerateTrainSet(I)
    N=new Network(I, h, 1)
    N.train(TrainSet, it, η, m)
    Eval(N,TestSet, TMSE , ICI)
    ch.fitness=(TMSE+ICI)/2
  Return(P)
End

Algorithm 2: Evaluate procedure

4.2 Neural Network Optimizer (NNO)

As a second part of the model, we propose to develop a sub-model capable of delivering a classifier with near optimal train and generalization accuracies. To achieve this goal, we have decided to combine NN learning skills with those of GA search capabilities.
It is well known that the ANN architecture has a significant impact on the information processing ability of the model. Moreover, there is no systematic way to design an optimal architecture for a particular task [23]. Besides, the **BP** learning algorithm suffers from the problem of local minima, which depend on both initial weight values and the learning rate parameter (η).

The **NNO** sub-model proposes a new hybrid algorithm which can evolve weights and architectures, and refines the learning capabilities. Our approach consists in two stages: *global* and *local search procedures*. The first stage designs the main aspects of the **NN** classifier, and the second concentrates on refining learning capabilities.

Input:
S: set of Financial ratios
N_i: initial population size
N: population size
t: tournament size
p_{mut} : mutation probability
p_{cross} : crossover probability
Maxgen : number of generations
Data: normalized dataset ratios
MP: An array of mutation operator probabilities
N_m: number of bits to change with mutation
N_{it}: iteration number of the train operator
η: learning rate of the train operator
m: moment value of the train operator
Output: Net_{opt}: Near optimal trained NN
Begin
  Population P_0, P, P_{tmp}
  P_0= P= P_{tmp}=$\emptyset$
  TrainSet=GenerateTrainSet(Data, S)
  TestSet=GenerateTestSet(Data, S, TrainSet)
  P_0=GenerateInitialPopulation(N_i)
  Evaluate (P_0, TrainSet, TestSet)
  P=Select (P_0, N, t)
  Wheel=InitWheel(MP)
  i=0
  While $i <$ *Maxgen* **do**
    P_{tmp}=Select (P_0, N, t)
    Crossover(P_{tmp}, p_{cross})
    Mutate(P_{tmp},p_{mut}, Wheel,N_m, it, η, m)
    Evaluate(P_{tmp}, TrainSet, TestSet)
    Replace(P_{tmp}, P)
    i=i+1
  Net=P.bestChromosome()
  Net_{opt}=EvolveWeights(Net)
  Return(Net_{opt})
End

Algorithm 3: Neural Network Optimizer algorithm (**NNO**)

4.2.1 The first stage: The global optimization procedure

The GA evolves through generations a set of NNs, where each one represented by a chromosome. In each generation, new individuals (NN genotypes) are created and added to the population using the repro-

duction operators (crossover, mutation and selection). Networks are evaluated according to their training and test error levels.

Chromosome representation

In deciding on an appropriate representation, we tried to choose the one that can be suitable to the evolution of both weights and architectures. Applying a binary encoding allows the NN to operate almost like the standard GA [24]. A direct encoding scheme is applied in this evolution process. The chromosome can be seen as the concatenation of binary representation of weight values. The node oriented representation groups the input weights for each neuron together. The network is encoded by a set of all possible connections like a fully connected network. However, only the existing connections are activated. Each connection encodes the weight value and its own state (enabled, disabled).

Genetic operators

Since the solutions that encode NN topologies and weights use binary representation, then, we can apply to them classical operators. Nevertheless, and because crossover of networks with different topologies can frequently lead to a loss of functionality [24]; we decided to reduce the crossover probability and to impose a granularity parameter. Individuals are spitted at the end of a node or connection according to the representation scheme adopted. The new chromosomes generated by crossover, passes through a validation stage. If the chromosome can be mapped into a valid NN (inputs connected, layers connected, nodes connected,...), then it can be added to the new generation. Otherwise, the generation process restarts. The mutation operators introduce innovation to the genetic material and allow the evolution process to explore new regions in the search space. Trying to diversify the exploration of new regions in the search space, we proposed a new mutation procedure based on the wheel selection mechanism. Each individual elected for mutation starts by the selection of the operator that will be applied on. In the aim to diversify the mutation procedure importance of each chromosome, we propose to use the wheel technique in the selection of mutation operators. The respective importance of each operator is randomly distributed on the wheel. In a such a way we can use the diversity of the mutation operators with the strength of the wheel technique. The applied mutation procedure is detailed by *Algorithm* 4.

Chromosome evaluation

The fitness function measures the quality of each chromosome. This value is particularly used during the selection of the individuals for the new generation. Three steps are necessary to compute the fitness value. First, the binary string is mapped into a network. Next, the mean error of the mean square errors and the incorrect classified instances are computed for both training and test sets. Finally, the mean value obtained by the two error measures is assigned to the chromosome as a fitness value.

Input:
P: population of chromosomes P_{mut} :
Mutation probability
N_m: number of bits to change with mutation
it: iterations to achieve with the train operator
η: learning rate of the train operator
m: moment value of the train operator
Wheel: the wheel represents the mutations operators probabilities
Output: P: Population with new chromosomes

```
Begin
    Random r
    Chromosome C
    Foreach C ∈ P do
        If r.getValue()<P_mut then
            choice=Wheel.getOperator()
            switch choice do
                case 1
                    mutateWheights(C,N_m)
                case 2
                    train(it,η,m,C)
                case 3
                    addConnection(C)
                case 4
                    deleteConnection(C)
                case 5
                    addNode(C)
                case 6
                    deleteNode(C)
    Return(P)
End
```

Algorithm 4: Mutation Procedure

4.2.2 The second stage: Refining the learning

Once the main aspects of the near optimal NN are defined by the global search procedure, the learning capabilities of the resulting classifier can be refined by a local search procedure. Consequently, the refine learning procedure will adjust the values of weights with one of the known learning approaches: the BP is generally used for this purpose [25, 26]. We propose, here, another GA that searches for the best set of weights using the network topology and architecture provided by the NNO used in the first stage.

Thus, the best individual of the first optimization stage is first mapped into a valid network. Then, the network is remapped into a new chromosome in the initial population of the second stage. Hence, the best solution provided by the first stage defines the parameters of the second stage. The network weights are evolved using a binary representation and standard genetic operators. The evaluation procedure is the same as the first stage optimization. This evolution process is presented in *Algorithm 3* by the **NNO** by the $EvolveWeights(.)$ function (*line* 19).

5 EXPERIMENTS AND EVALUATION

5.1 Available data

5.1.1 data description

For our experiments, we use data sets that have been used to forecast firm bankruptcy. The data set represents Tunisian firms results one year before bankruptcy. Results were derived from financial balance sheets of the year 2002 exercise. The data consists of 88 firms, which filed for bankruptcy 38 cases, and non-bankruptcy 50 cases. Each firm in the data set is represented by 31 values: 30 ratios and a binary variable representing the firms state (0: non-bankrupt, 1: bankrupt). The available ratios are described in Table 1.

5.1.2 training and test datasets

To evaluate the model training accuracy and generalization performance in unseen cases; we should divide the available sample instances into two sets: train and test set. Given the small size of our sample (88 instances), we have decided to dynamically change the instances in both data sets at each new learning task. Each time when we have to train or evaluate a NN, both datasets are built with a random distribution from the original data set generated. Although data sets instances are randomly selected, the bankrupt and non-bankrupt firms are proportionally distributed in each data set.

5.1.3 Data preprocessing

Modelling tools based on ANN can not be trained or assessed by a raw data set [27]. The available data must be normalized. In our case, all the available variables values have a numerical representation (ratios).

Ratio	Formula
R1	Current assets/Short-term debts
R2	(realizable values + liquid assets)/Short-term debts
R3	Quick ratio Liquid assets/Short-term debts
R4	long term capital/fixed assets
R5	long and medium term liabilities/long term capital
R6	long and medium term liabilities/Share holders' equity
R7	Share holder's equity/Total debts
R8	Share holder's equity/Short-term debts
R9	financial costs/ EBE
R10	Cash-flows/financial costs
R11	Working capital/total assets
R12	Working capital/current assets
R13	Working capital/working capital requirements
R14	Turnover/total assets
R15	Turnover/Net fixed assets
R16	Turnover/Stocks
R17	Depreciation provision/fixed assets
R18	Staff costs/added values
R19	financial costs/added value
R20	Net fixed assets/total fixed assets
R21	realizable values/total assets
R22	liquid assets/total assets
R23	Share holder's equity/ Total liabilities
R24	long and medium term liabilities/Total liabilities
R25	long term liabilities/Total liabilities
R26	$Log(balancesheet)$
R27	EBE /Total assets
R28	Net return+financial costs/long term capital
R29	Net return/Share holder's equity
R30	Cash flow/total assets

Table 1: List of available ratios

Normalization accepts values that span one range and represents them in another range. ANN requires data to be close to the range of 0 to 1. The selected method for normalization is the *linear transform scaling* [27]:

$$\nu_n = \frac{\nu_i - min(\nu_1..\nu_n)}{max(\nu_1..\nu_n) - min(\nu_1..\nu_n)} \quad (4)$$

where ν_n and ν_i respectively represent the normalized and instance values. The main advantage of the linear scaling is that it introduces no distortion to the variable distributions.

5.2 Artificial neural networks in the forecast of bankruptcy

As a first step, experiments were performed with different NN wrappers configurations (architecture, and learning parameters) using the inputs selected by the MDA in a previous study. The ratios used as a NN input are summarised in Table 2:

Ratio	Formula
R5	long and medium term liabilities/long term capital
R26	Log(balance sheet)
R27	EBE /Total assets
R30	Cash flow/total assets

Table 2: List of ratios selected by MDA

We trained four models. Each model used the same set of ratios as described above. However, each model differs form the others by the number of hidden nodes. Also, we have applied the standard **BP** algorithm with momentum [9, 28]. Built networks are feed foreword and fully connected. We have tested the **ANN** using different data sets and the best results were obtained with the following parameters: the learning rate and momentum was respectively fixed to 0.25 and 0.5. The initial values of weights were randomly selected in the range of [-0.1,0.1]. Using these **ANN** parameters, the models was trained with 5000 epochs. The train and the test errors were computed using the following formulas:

$$Train\_error = \frac{(MSE + ICI)}{2} \quad (5)$$

$$Test\_error = \frac{(MSE + ICI)}{2} \quad (6)$$

Where ICI and MSE correspond respectively to the proportion of correct classified instances in the data set and the value of the mean square error.

Globally, the four ANN models achieve around 80% of accuracy in tests and 85% in train. We can also remark that the train error is less than the test error. Although, the best train error was obtained by the first model, the worst test error was also delivered by the same model. We can conclude, for this case, that the network with five hidden nodes was enable to learn all the the data set enfolded behaviors. The test error decreases as long as the number of hidden nodes increases. Moreover, the evolution of the error levels can be seen as two stage evolution. The four models decrease their respective train errors until reaching their convergence levels. In fact, as far as the train errors stagnate, the test errors start to increase after reaching their lowest levels.

5.3 Input Selection Model: evaluation

Trying to give to the ANN the best predictors, a GA is launched by the ISM model to select the set of ratios with the high predictability potential. The GA parameters are summarized in Table 3, where p_{mut} and p_{cross} represent respectively the operators probabilities. The GA searches for the best solution during 100 generations. $Size$ and $Size_{ng}$ represent respectively the population size and the number of individuals elected to the new generation. The ANN parameters are used during the computation of the fitness value.

The best set of predictors selected with ISM are summarized in Table 4.

By scrutinizing the variables (ratios) included in each model, we pay attention to the number of variables selected by ISM. The number varies from 6 to 12. Also, the number of selected ratios is independent of the number of hidden nodes. Six and eight variables are selected respectively with 5 and 20 hidden nodes, while 10 and 15 nodes, eleven and twelve ratios were selected. Variables included in Table 4 show that there is only one variable R5 (long and medium term liabilities/long term capital) that is selected in the four models.

by comparing performance of each model (Figure 3), we can see that the worst results were given with the model with the fewest number of variables. Nevertheless, the best results were not obtained with a model with the largest number of variables. Compared to the variables selected with MDA, the first difference was in the number of retained variables. The four models derived with ISM have a greater number of ratios. Two of the four models (10-15 hidden nodes) contain all the selected variables in the MDA (R5, R26, R27, R30) and all the four sets include at least two MDA ratios.

Once the best set of predictors has been selected, a new ANN is built for each model and trained for 5000 iterations. The model parameters and results are detailed in Tables 5 and 3.

The results show a considerable improvement in almost cases. The best results were given by the third model which, that reached 100% of success for the correct classified instances and lower value of global training error equal to 3×10^{-4}%. The test level is also kept at his low level which is equal to 8%. The fourth model achieves a slightly greater error level. However, its global performance is better than those obtained with ANNs trained without ISM feature selection.

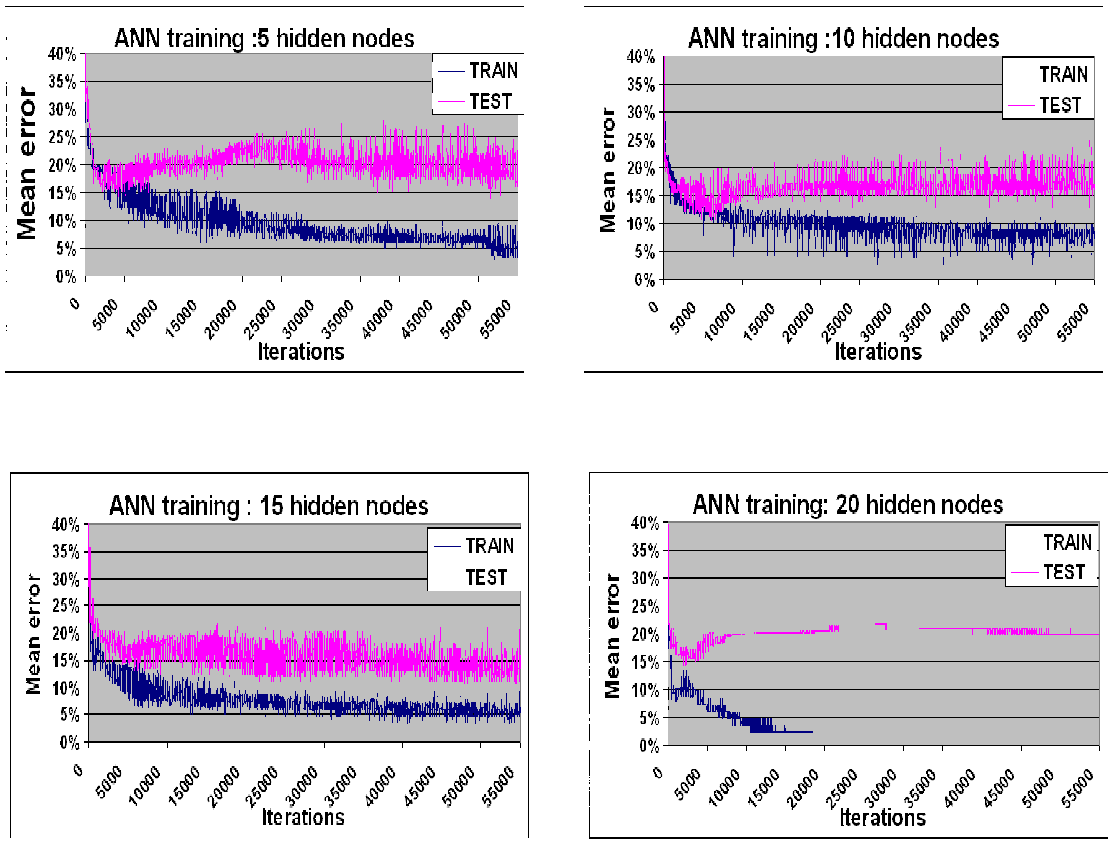

Figure 2: ANN results after 5000 epochs

Configuration	GA parameters					ANN parameters		
	Iterations	p_{mut}	p_{cross}	Size	Size_{ng}	Hidden nodes	Iterations	Lr
H5	100	5%	85%	30	5	5	100	0.25
H10	100	5%	85%	30	5	10	100	0.25
H15	100	5%	85%	30	5	15	100	0.25
H20	100	5%	85%	30	5	20	100	0.25

Table 3: ISM parameters

Configuration	Ratio number	List of ratios
H5	6	R1, R5, R6, R10, R16, R26
H10	11	R3, R4, R5, R6, R7, R16, R18, R21, R26, R27, R30
H15	12	R5, R17, R21, R22, R23, R24, R25, R26, R27, R28, R29, R30
H20	8	R5, R9, R14, R17, R20, R22, R27, R28

Table 4: List of selected ratios

Configuration	Hidden nodes	Train error	Test error
H5	5	8.391 %	18.697 %
H10	10	6.007 %	14.693 %
H15	15	0.039 %	8.154 %
H20	20	5.938 %	17.977 %

Table 5: ISM: train and test results after 5000 epochs

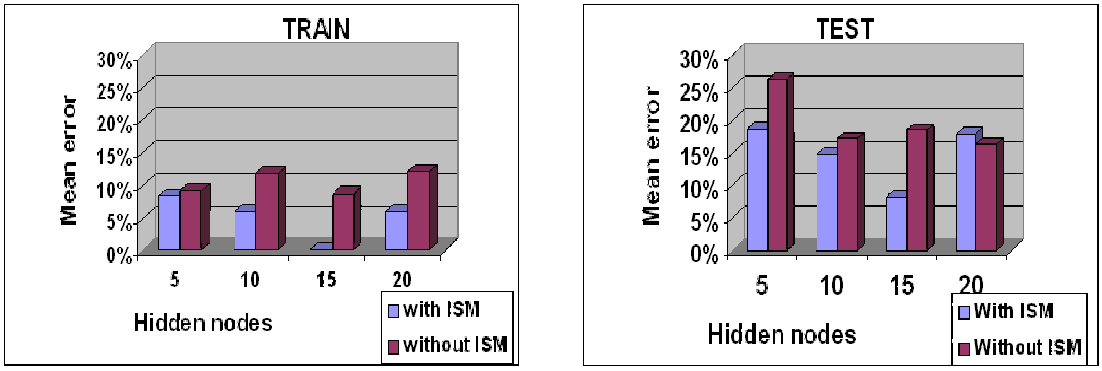

Figure 3: Train and test results of the ANN trained with 5000 epochs

5.4 Neural network optimizer (NNO)

Once the best input predictors are selected, the **NNO** uses theses inputs to search the best ANN architecture and weights suited to the problem at hand. In the following, we experiment different configurations of the **NNO** parameters.

Figure 4 presents and compares the results obtained in two experiments. The first one is done with NNs using the ratios selected by ISM. These models are trained for 55000 iterations. The results showed in the figure are labelled "ISM Convergence" and corresponds to NN classifiers only applying the first model stage. In the second experiment, the best set of predictors selected within ISM are presented to **NNO** to search their best ANN properties. **NNO** is applied for each set of ratios selected by ISM and the results presented in this figure illustrate the error level achieved by each model. All these classifiers used ISM as a feature selection model and are trained with for 50000 iterations. The evolutionary learning approach NNO applies 1500 iterations in the global search stage and 1500 iterations with the local search stage using the best chromosome retrieved from the first one.

By comparing the obtained results, it is clear that the results obtained with ISM and **NNO** are better than those by only applying ISM with fully connected networks. The four models confirm the superiority of our model. Also, this figure shows the effect of the selection of the appropriate and adjusted network weights and architecture. The training error is kept at a low level $(2\%, 0.1\%, 10^{-4}\%, 9\%)$. The accuracy in generalization is considerably improved: in the second model this test error passes from 24% to 11%. The lowest accurate model provided by **NNO** is the fourth. The accuracy lost in training is recovered by generalization. The best performance is given by the third model with more than 99% of accuracy in both training and tests.

5.5 Whole model evaluation

Figure 5 presents a recapitulation of the experiments stages. In each stage, the error of both train and test are represented for each model. We can point out that the improvement of the classifiers' performance when the redundant and irrelevant features are discarded by ISM. Besides, the combination of the two sub-models improves the resulting classifiers accuracy as well in training as in test. We can conclude that our evolutionary classifier based on feature selection and evolutionary learning techniques outperforms as well the conventional ANNs trained with BP as ANNs using ISM.

6 CONCLUSION

In this paper, we have shown how feature selection based on both GA as a global search procedure and NN as a wrapper can be effective to improve NN classifiers accuracy. Global search procedure based on GA was also applied in the second stage model as learning algorithm and NN architecture designer. Experiments have shown globally very promising results of our proposed tow stage model for the forecasting of firms insolvency, in terms of predictive accuracy and predictability. Nevertheless, more robust and reliable model can be obtained by equipping it by meta-learning capabilities allowing our model to adjust its parameters without any human interaction.

REFERENCES

[1] A. Cornujols, L. Miclet, Y. Kodratoff and T. Mitchell. *Apprentissage artificiel: concepts et algorithmes*. Eyrolles, August 2002.

[2] K. O. Stanley and R. Miikkulainen. "Evolving Neural Networks through Augmenting Topologies". *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[3] F. A. Atiya. "Bankruptcy Prediction for Credit Risk Using Neural Networks: A Survey and New Results". *IEEE Transactions on Neural Networks*, vol. 12, no. 4, July 2001.

[4] T. K. Sung and N. G. Lee. "Dynamics of modeling in data mining: interpretive approach to bankruptcy prediction". *Journal of Management Information Systems*, vol. 16, pp. 63–85, Summer 1999.

[5] B. Back, G. Oosterom, K. Sere and M. van Wezel. "A Comparative Study of Neural Networks in Bankrupty Prediction". 12, pp. 140–148. Proc. Conference on Artificial Intelligence Res. in Finland, Finnish Artificial Intelligence Society, Helsinki, Finland, 1994.

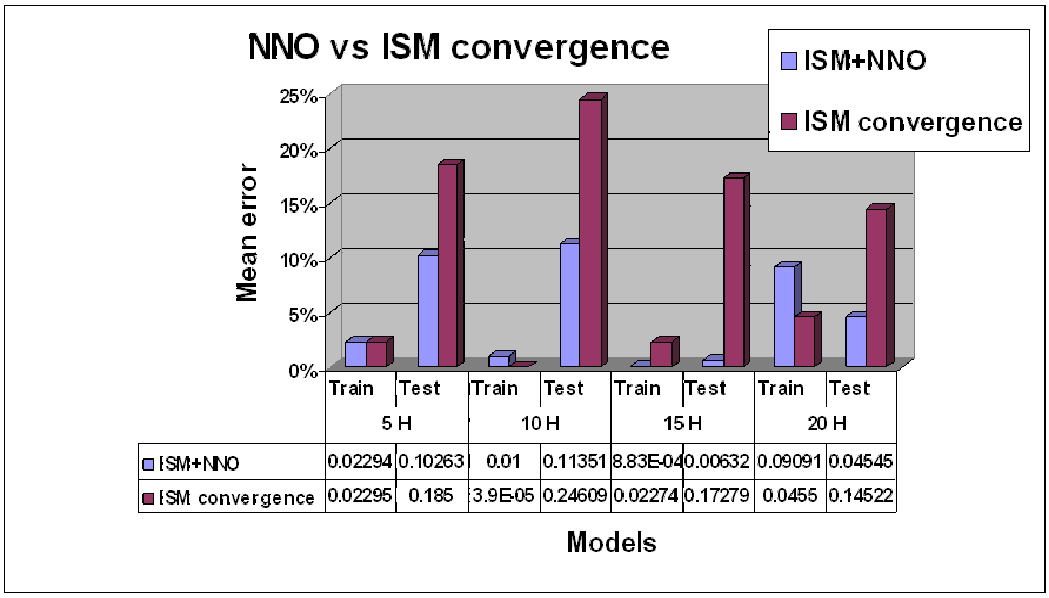

Figure 4: NNO results compared to ISM convergence

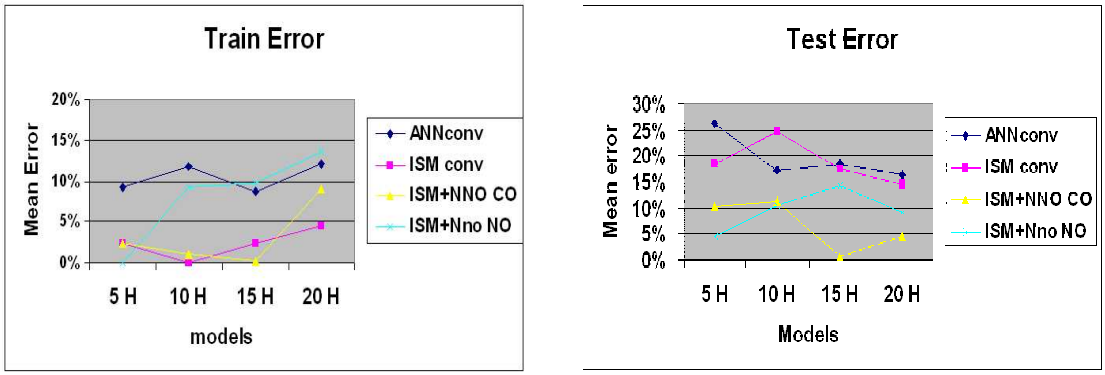

Figure 5: Whole model evaluation

[6] X. Li and N. D. Gupta. "Neural networks in financial failure prediction: A revisit". *Decision Sciences Institute*, vol. 87, pp. 233–238, 2002.

[7] B. Back, T. Leitinen, K. Sere and M. V. Wezel. "Choosing the best set of bankruptcy predictors' using diascriminant analysis, Logit Analysis and genetic algorithms". Technical report ISBN 951-650-828-6 40, Turku Center for computer science, september 1996.

[8] S. Haykin. *Neural Networks: a Comprehensive Foundation.* Upper Saddle River, NJ: Prentice Hall, 2nd edn., 1999.

[9] D. E. Rumelhart, G. E. Hinton and R. J. Williams. "Learning Internal Representations by Error Propagation". In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, chap. 8. MIT Press, Cambridge, 1986.

[10] S. Sexton and N. Gupta. "Comparative evaluation of genetic algorithms and backpropagation for training neural networks". *Information Sciences*, vol. 129, pp. 45–49, 2000.

[11] D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning.* Addison Wesley, 1989.

[12] J. S. Grice and R. W. Ingram. "Test of the generalizability of Altman's bankruptcy prediction model". *Journal of Business Research*, vol. 54, pp. 53–61, 2001.

[13] E. Bodt and E. F. Henrion. *Les réseaux de neurones en finance: Conception et applications.* Facto publications, 1995. ISBN 2-9600049-5-7.D.

[14] A. Faure. *Cybernétique des réseaux neuronaux.* Edition Hermès, 1998.

[15] T. Kohonen. *Self-Organizing Maps.* Berlin, Heidelberg. Springer-Verlag, 2 edn., 2001.

[16] V. Vapnik. *Statistical Learning Theory.* Wiley, 1998.

[17] K. S. Shin, T. S. Lee and H. J. Kim. "An application of support vector machines in bankruptcy prediction model". *Expert Systems with Applications*, vol. 28, pp. 127–135, 2005.

[18] K. C. Lee, I. Han and Y. Kwon. "Hybrid Neural Networks models for bankruptcy prediction". *Decision Support System*, vol. 18, pp. 63–62, 1996.

[19] J. R. Quinlan. "Induction of decision trees". *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[20] Y. Lui and X. Yao. "A population-based learning algorithm which learns both architectures and weights of neural networks". *Chinese Journal of Advanced Software Research*, vol. 3, pp. 54–65, 1996.

[21] P. C. Pendhrkar. "An empirical study of design and testing of hybrid evolutionnary-neural approach for classification." *The International journal of Management Science*, vol. 29, pp. 361–374, April 2001.

[22] M. Aleixandre, I. Sayago and M. Horrilo. "Analysis of neural networks and analysis of feature selection with genetic algorithm to discriminate among pollutant gaz". *Sensors and Actuators*, vol. B, no. 103, pp. 122–128, June 2004.

[23] X. Yao. "Evolving Artificial Neural Networks". *PIEEE: Proceedings of the IEEE*, vol. 87, pp. 1423–1447, 1999.

[24] K. O. Stanley and R. Miikkulainen. "Evolving neural networks through augmenting topologies". Tecnical report tr-ai-01-290, The University of Texas, Department of Computer Science, June 2001.

[25] M. Gheith and A. A. Badr. "Stochastic and Gradient techniques for optimal supervised learning". 3rd International Conference on Artificial Intelligence Applications, Egypt (1995).

[26] D. Curran and C. OŔiordan. "Applying Evolutionnary computation to designing neural networks: A study of the state of the Art". Technical report, nuig-111002, National University of Ireland, Galway, October 2002.

[27] D. Pyle. *Data Preparation for Data Mining.* 1999. ISBN 1558605290.

[28] M. Riedmiller and H. Braun. "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP algorithm". In *Proc. of the IEEE Intl. Conf. on Neural Networks*, pp. 586–591. San Francisco, CA, 1993.