UDC:007:681.3:651(082)
Original scientific paper

SELECTING NEURAL NETWORK ARCHITECTURE FOR INVESTMENT PROFITABILITY PREDICTIONS

Marijana Zekić-Sušac, Nataša Šarlija
Faculty of Economics, University of J.J. Strossmayer in Osijek, Croatia
{marijana, natasa}@efos.hr

Mirta Benšić
Faculty of Organization and Informatics
Department of Mathematics, University of J.J. Strossmayer in Osijek, Croatia
mirta@mathos.hr

Abstract: *After production and operations, finance and investments are one of the most frequent areas of neural network applications in business. The lack of standardized paradigms that can determine the efficiency of certain NN architectures in a particular problem domain is still present. The selection of NN architecture needs to take into consideration the type of the problem, the nature of the data in the model, as well as some strategies based on result comparison. The paper describes previous research in that area and suggests a forward strategy for selecting best NN algorithm and structure. Since the strategy includes both parameter-based and variable-based testings, it can be used for selecting NN architectures as well as for extracting models. The backpropagation, radial-basis, modular, LVQ and probabilistic neural network algorithms were used on two independent sets: stock market and credit scoring data. The results show that neural networks give better accuracy comparing to multiple regression and logistic regression models. Since it is model-independant, the strategy can be used by researchers and professionals in other areas of application.*

Keywords: *neural networks, non-linear forward strategy, stock market prediction, credit scoring.*

1. INTRODUCTION

Artificial intelligence based on learning theory improves the ability of using the prior knowledge and data in order to make effective decisions. Neural networks (NNs) can be used for prediction, classification, and association problems in various problem areas. Finance and investing is the second most frequent business area of neural network applications after production/operations [17]. The lack of standardized paradigms that can determine the efficiency of certain NN algorithms and architectures in particular problem domains is emphasized by many authors [5]. The focus of the paper is in improving NN prediction and classification ability by selecting the best NN architecture for a certain problem. Since the aim of the research is to obtain the best model for the observed data, NNs were compared to a method that is classically used for that type of problem (multiple regression for the stock return prediction problem, and logistic regression for the credit scoring problem). The next section provides the description of previous research results regarding NN architecture selection. A brief overview of NN methodology used including five NN algorithms for prediction and classification is described in the third part. Part 4 presents the suggested forward strategy for selecting NN architecture, while the data and results are described in parts 5 and 6. After the conclusion, some guideliness for future research in this area are suggested.

2. DESCRIPTION OF PREVIOUS RESEARCH

Some of the most representative problems being solved by NNs are bankruptcy predictions, risk assessments of mortgage and other loans, stock market predictions (stock, bond, and option prices, capital returns, commodity trade, etc.), financial prognoses (returns on investments) and others [18]. NNs in finance are frequently applied in predicting stock performance and selecting stocks for trading on stock markets. Stock markets are suitable area for NN application because of the existence of nonlinear dependencies assumed on the basis of market microstructure, feedback effects in market prices and empirical observations, and confirmed for most of the world major markets [13]. Linear models cannot serve as efficient forecasting systems in such situations. There are three main groups of problems that NN applications frequently deal with in stock markets: (1) classifying stocks into the classes such as: positive return stocks and negative return stocks, (2) predicting the exact stock price for one or more days in advance, based on previous stock prices and related financial ratios, and (3) modeling stock performance and forecasting. The last group of applications is not only focused on the prediction of future values, but also on the factor significance estimation, sensitivity analysis among the variables that could impact the result, and other analyses of mutual dependencies. NN applications in banking are mostly concentrated on credit assignment problems, bankruptcy predictions, customer segmenting, detecting credit card frauds, and other problems.

The analysis of previous research also shows a variety of NN algorithms and evaluation measures used in financial applications [19]. The backpropagation (multi-layer perceptron) algorithm is the most frequently used, although other algorithms are present in some rare applications. The three-layer structure seems to be effective according to many authors, with the exception of a few applications [13] where the four-layer structure outperforms other structures. The majority of NN applications use the sigmoid transfer function, with changeable learning parameters and momentum. One step toward investigating other NN algorithms than backpropagation is the usage of time-delayed, recurrent and probabilistic networks with great success in false alarm accuracy [14]. The comparison of NN's performance on stock market predictions with statistical multiple regression is done by several authors including White [16] and Refenes et al. [12].

More recent research shows that combining NNs with other data mining techniques is very effective. Lam [6] used a backpropagation NN in combination with a rule extraction technique to predict stock return for 364 S&P companies on the basis of fundamental and technical data. Her results indicate that NN using one year's or multiple years' financial data significantly outperform the minimum benchmark (the overall market average return), but not the maximum benchmark (the top one-third returns in the market). Enke and Thawornwong [2] introduce a combination of neural networks and data mining techniques in forecasting stock market return. The information gain technique is used to evaluate the predictive relationships of input variables, while the NN models were then used to forecast the future values.

2.1. PREVIOUS RESEARCH ON NEURAL NETWORK TOPOLOGY AND ALGORITHM SELECTION

Previous research in finding the best NN architecture was mostly focused on network topology, resulted with some heuristic rules [7], [8], [10], and optimizing algorithms such as A* algorithm [1], cascading, pruning, genetic algorithms and others. It is proved (by Hornik et al., 1989. in [8]) that three-layer networks are able to approximate any discontinuous function. In many cases, more layers slow down the training time because the gradient of the error is more unstable and there is a higher risk of local minima. Mozer (in [11]) showed that processing units (nodes) vary in their functional importance for solving a problem, indicating that the relevance of a unit can be determined by comparing performance of the NN when the unit is included to the performance of the NN when the unit is removed. The overview of structure optimization techniques is given by Quinlan [11], who synthesizes a number of optimization techniques into two main groups: (a) backpropagation derivatives, and (b) algorithms for discrete unit networks. Backpropagation derivatives are further divided into: (1) techniques based on units removal (skeletonisation, artificial programmed cell death, and gauging the optimal size of network in terms of generalization capability), (2) pruning techniques (Thodberg's technique of pruning the connections with small weights, and optimal brain damage), (3) adding units and connections (Meiosis networks), (4) both adding and pruning units and connections (Bartlett's technique, and Hirose's technique). The second group of algorithms for discrete unit networks includes the techniques such as tiling algorithm, pointing and tower algorithms, neural tree approach, and upstart algorithm. Most of those techniques are based on two common principles: gradually adding new nodes or gradually excluding the nodes, using different criteria. Another shared characteristic of those techniques is that they make the network dynamic in the sense of changing its structure. The idea of self-pruning is in simultaneous minimization of output error and minimization of the number of hidden neurons. The main disadvantage of both cascading and pruning procedures is that the upper limit of number of hidden nodes should be determined in advance, while there is no rule of how to determine it. One way to overcome this limitation is to use genetic algorithms that start with different, randomly chosen topologies. A*- algorithm, proposed by Nilson, 1980 (in [1]) applies graph theory to find the optimal path that presents the best NN structure. Its advantage is in the fact that it does not imply any restrictions on generated structure i.e. there is no limited number of hidden neurons.

Besides dynamic optimizing techniques, there are some static heuristic rules for determining a NN structure, such as Masters' formula [8] that computes the number of hidden units on the basis of the number of input and output units. Some of the above techniques and rules were tested in relation to underlying data in a stock prediction model [20], and it was shown that Masters' rule performs good on the training set of the models, but those NNs have low performance on the validation set. It was also shown that cascading technique performs well on data with small variance, while pruning technique performs well on most of the models, and generalization capability of such models is acceptable.

The selection of NN algorithm is primarily subject to input and output functions built in an algorithm. They determine the main selection of algorithms according to the type of the problem: prediction, classification, association, data compression, data conceptualization, etc. Some general-purpose algorithms allow adding specific functions that will adjust them to a specific type of problem. For example, backpropagation and radial-basis algorithms that are primarily aimed for prediction can be used for classification by adding a softmax activation function to the output layer [10]. Another criterion for algorithm selection is the nature of the data, since researchers report different results depending on the dataset and models used. Masters suggests the probabilistic neural network as a good selection for classification problems when there are outliers in data [8], and when the learning speed is important, since this algorithm does not learn iteratively, but use only one pass through the dataset. NeuralWare [10] brings some heuristic rules for selecting NN algorithm based on a problem type and data constraints. According to [10] the backpropagation and the modular algorithms are the most suitable for noisy data, they are also recommended together with LVQ for the models with many input variables, which was also shown by Zekic and Klicek [20]. Although none of the algorithms is evaluated as excellent for sparse data, the counter propagation, general regression, LVQ, and probabilistic are suggested as good choices in such datasets. When the data is nonstationary, the general regression is selected as the best solution, followed by fuzzy ARTMAP and modular algorithms. The same authors [20] test eight NN algorithms according to the nature of data in a stock market dataset, and obtains that the performance of NN algorithms significantly (0.5 level) differs for the models with sparse data and data with outliers. In the prediction type of problem, the modular network is extracted as the best for such data followed by backpropagation on the basis of the average trade result (ATR). In case of the existence of nonstationary variables in the model, the general regression network showed the worst result, and is not suggested for such type of data. None of the NN algorithms significantly outperforms others in classification problems, although the LVQ showed the lowest result. These findings reveal that the selection of algorithms depends on both data nature and model used; therefore they support the strategy for selecting NN algorithms based on testing more algorithms on the same dataset.

3. FORWARD STRATEGY FOR SELECTING NN ARCHITECTURE

The selection of NN architecture needs to take into consideration the type of the problem, the nature of the data in the model, as well as some cross-validating strategies based on result comparison. The type of the problem (classification, prediction, function approximation, association) makes the general selection based on the theoretical foundation of NN algorithms, although there are a number of general-purpose algorithms that can be used for all types of problems (such as backpropagation, radial basis function, general regression). Besides that, most of the prediction problems can be also defined as classification problems, with some modifications in expressing the output variable (for example, prediction of stock prices can be easily modified as classification of stocks into poorly-performed and good-performed stocks). For the above reasons, it was necessary to define a strategy for selecting NN architecture depending on the model tested. The suggested strategy is to build NN models by starting from a single variable, then gradually add another one and test if it improves the model performance. The variables that improve the performance of the model are included in the model. The strategy is not only used for modeling purpose, but also for the purpose of finding the best NN algorithm and structure. Prediction algorithms were backpropagation, radial-basis, general regression, and modular, while the following four algorithms were tested for classification problems: backpropagation and radial basis function network (with softmax activation function in the output layer in order to obtain probabilities), probabilistic and learning vector quantization. Each more-layered NN algorithm (backpropagation and radial-basis with and without softmax) is structured separately including 1 hidden layer and 2 hidden layers (in order to see if the additional hidden layer can improve the detection of nonlinear dependencies among the data). Additional structure and parameter optimization is done in the training phase of the network. The strategy can be described in the following steps:

1) select the first NN algorithm based on the problem type

2) perform forward modeling

The modeling procedure starts by using a single input variable. The NN is trained on in-sample and tested on out-of-sample data, while the first test result serves as the baseline measure (or reference) for the next NN. The second NN model adds another exogenous variable. The out-of-sample test result is compared to the reference, and if the new result is better than or equal to the reference, the variable is sustained in the model and the new result becomes the reference. If the result is worse than the reference, the variable is not included in the model, and the next variable is tested. Such iterative process is repeated until all the exogenous variables are tested and the best model is saved.

3) test the model with all available variables

After the forward modeling strategy extracts the model, the whole universe of input variables is tested together in one model and the performance of such maximal model is compared to the best result obtained in step 2.

4) repeat steps 1-3 for the next NN algorithm (or for the next parameter within the same algorithm) until all available NN architectures were tested

In this step, it is possible to change only one parameter in a NN algorithm used in step 1 (for example to change the transfer function in a layer, or to change a learning parameter etc.), or to change a NN algorithm type.

Completion of the first three steps of the above process extracts the best model using one NN architecture. Since the change of one parameter in the architecture can produce different results regarding the variable selection, it is necessary to test all available NN architectures suitable for a problem. The overall best NN model is identified at the end of the procedure by choosing the model with the overall best result. All the examined models were tested on the out-of-sample data. The above iterative strategy is conducted a number of times in our experiments testing different NN architectures. The above procedure is graphically presented in Figure 1. In order to perform the strategy a VB program is created, which controls the whole iterative process of selecting the NN architectures and variables, evaluating the results, and presenting it to the user, while the neural computation is done by the NeuralWare software.

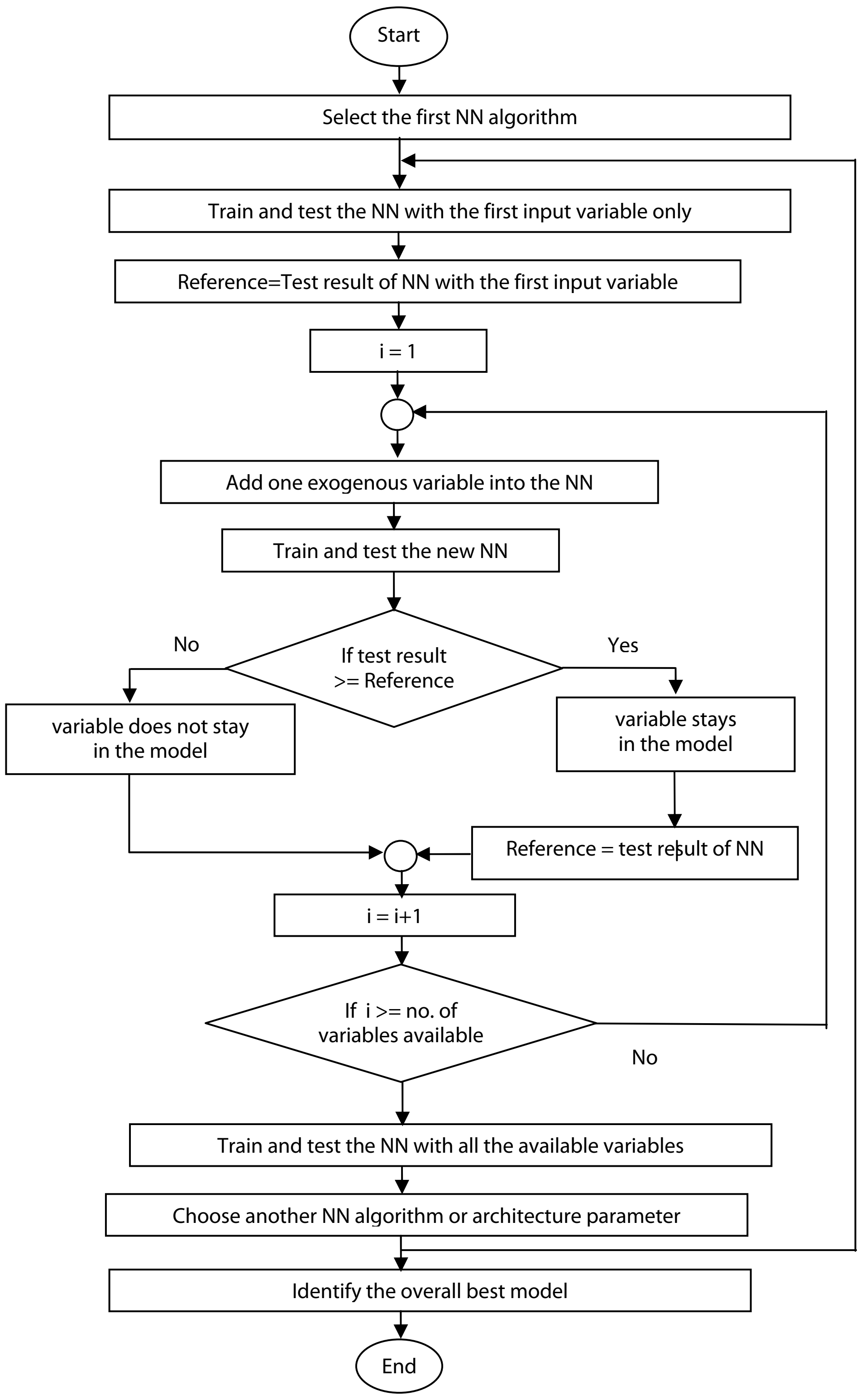
Start
Select the first NN algorithm
Train and test the NN with the first input variable only
Reference=Test result of NN with the first input variable
i = 1
Add one exogenous variable into the NN
Train and test the new NN
No
If test result >= Reference
Yes
variable does not stay in the model
variable stays in the model
Reference = test result of NN
i = i+1
If i >= no. of variables available
No
Train and test the NN with all the available variables
Choose another NN algorithm or architecture parameter
Identify the overall best model
End

Figure 1: Forward strategy for selecting NN architecture and model

4. NEURAL NETWORK METHODOLOGY

Four prediction and four classification algorithms were tested. They were selected due to their well-established and confirmed computational foundation. In the prediction type of problem, backpropagation, radial basis, general regression and modular (proposed by Jacobs et al, 1991) algorithms were used for predicting stock return rate. For classification type of problem, softmax activation function is added to the standard backpropation and radial-basis function algorithms in order to obtain the probabilities in the output classes. LVQ, as a supervised version of the Kohonen algorithm with unsupervised kernel in the hidden layer is also tested. In order to make the learning process in backpropagation algorithm faster, the extended delta-bar-delta (EDBD) rule is used instead of the standard delta rule. The initial learning rates for the EDBD learning rule used in our experiments were set as follows: 0.3 for the first hidden, 0.2 for the second hidden layer, 0.15 for the output layer, and the inital momentum term was set to 0.5. The tangent hyperbolic transfer function was used in the hidden layer. To overcome the problem of local minima in the backpropagation, a stochastic method of simulated annealing is applied by adding noise into the result of the input function before sending output to other units in the network. The range of noise is determined by the temperature (-0.01*t, 0.01*t). The temperature is initially set to a high value, then decreased gradually in order to lead the transfer function to the area of the global minimum. Overtraining is avoided by the so-called “save best” procedure, a sort of cross-validation process which alternatively trains and tests the network (using a separate test sample) until the performance of the network on the test sample does not improve for *n* number of attempts. The maximum number of iterations in our experiments was set to 100000, *n*=10, while the number of iterations in each training session was 5000 for the first dataset, and 1000 for the second dataset. After the best network is selected, it is tested on a new validation sample to determine its generalization ability. Saturation of weights is prevented by adding a small F’offset value to the derivative of the sigmoid transfer function. It is experimentally proved that the value 0.1 is adequate for the sigmoid, and 0.3 for the tangent hyperbolic transfer function (Fahlmann in [10]).

The topology of the backpropagation NN consisted of maximum 105 hidden units in the first hidden layer and 52 hidden units in the second hidden layer. The same was used in the radial-basis function network. The general regression networks consisted of 1 hidden layer with maximum 105 hidden units. The modular network consisted of 1 hidden layer, 3 local experts (consisted of backpropagation networks) with maximum 105 hidden units in each, while the gate network had 35 hidden units.

Probabilistic network is a classifier that uses Parzen windows for clustering and produces a number of classes in the output. The parameter σ is optimized by cross-validating a heuristically set range of σ values, such that each σ value is used to train and test the network. The minimum value of sigma was 0.1; the maximum value was 3, with the step of 0.2. Euclidean summation function in the pattern unit is used, radius of influence was 0.25, and output mode for the output layer was competitive. LVQ network consisted of a Kohonen layer, and an output layer with predetermined classes. The number of Kohonen units was set to 10% of the size of the train sample. The number of learning iterations in the LVQ1 phase is set to 25000, the number of learning iterations in the LVQ2 phase is set to 10000, the initial learning rate for LVQ1 was 0.06, and the initial learning rate for LVQ2 was 0.03. The LVQ2 width parameter was 0.2, the option “in-class winner always learns” is used, with the conscience factor 1.

The output layer in classification networks consisted of three neurons representing three classes of stocks in the stock market dataset, and two neurons representing two classes of applicants in the credit scoring dataset.

According to Masters' suggestions for appropriate evaluate measures in financial applications (1998), we have evaluated stock market neural networks using the ATR, although the total return, average return per trade, and correlation between the computed and the desired output were also computed. The performance of credit scoring neural networks was measured by the total hit rate. The NeuralWare software was used to run neural networks.

5. DATA

The first dataset consisted of daily data on IBM stock return, financial ratios and macroeconomic variables. Daily close prices were collected by i-Soft Inc.[1], financial ratios and dividends were computed or collected from the IBM quarterly balance sheets and income statements provided by Value Line Inc., while macroeconomic variables were provided by Econ*magic*.com, an economic time series source[2]. The total data sample, after data pre-processing, consisted of 848 cases. The train sample consists of the first 80% of the cases, while the other 20% of the cases covering the rest of the period were used to form the out-of-sample test data. For the purposes of optimizing NN structure and training time, the train sample is further divided into the train1 sample (85% of the train sample), and the test1 sample (15% of the train sample). Test1 sample is needed in NNs in order to avoid the influence of out-of-sample data to the optimizing procedure in NNs, which is necessary according to [9]. The available space of input variables is created on the basis of the arbitrage pricing theory or the factoral model [12], consisted of variables: V1 - moving average term (MA(1)) - obtained by ARIMA, V2 - volume, V3 - weekly stock volatility, V4 - beta, V5 - return on investment (ROI), V6 - capital intensiveness (CI), V7 - financial leverage (FL), V8 - receivables intensiveness (RI), V9 - inventory intesiveness (II), V10 - current ratio (CR), V11 - cash position (C), V12 - 30-year mortgage rate (MR), V13 - 3-month US treasury bill rate (TBR), V14 - bank prime loan rate (BPLR), V15 - exchange rate of Japanese yen according to US dollar (ER), V16 - federal funds rate (FFR), V17 - S&P 500 total return rate (S&P500 TRR), V18 - industrial production index (IPI), V19 - unemployment rate (UR), V20 - index of leading indicators (ILI), and V21 - consumer price index (CPI). The output variable was the stock return, defined as the actual rate of return realized over some evaluation period. It was represented in two ways: (a) as the absolute value of stock return for the next day *(t+1)* in prediction problems (first differences are used to eliminate the trend influence) computed according to White (1996):

$$R_{t+1}=(p_{t+1}-p_t+d_{t+1})/p_t\,, \qquad (1)$$

where p_t is the closing price on day t, d_t is the dividend paid in day t, and (b) as a classification problem, with three output variables representing three classes of stock return: positive, neutral, and negative stock return. In order to classify the stock return rate into three classes, 1 of N binary code is used together with the threshold values -0.5 and 0.5. The rate of return is classified as positive if it is greater than 0.5, negative if it is less than –0.5, and neutral if it is greater than or equal to –0.5 and less than or equal to 0.5.

The second dataset consisted of a small business credit scoring data, and was collected randomly in a Croatian savings and loan association. The sample size was 166 credit applicants – small entrepreneurs - that entered the bank requesting for a loan. The selection of input variables was guided by the previous research (see for details [21]), including both business and personal characteristics of entrepreneur. The input variables were: main activity of the firm, new firm (yes/no), number of employees, entrepreneur's occupation,

[1] StockWiz, a registered trade mark of i-Soft Inc. is used for collecting daily stock prices

[2] URL: http://www.economagic.com

entrepreneur's age, business location, credit request (first time or not), way of interest payment, grace period, principal payment, repayment period, interest rate, credit amount, planned value of the reinvested profit, cleat vision of business, better than competition, sale on goods and services, advertising, and awareness of competition. The output variable was the credit scoring in the form of a binary variable with one category representing good applicants and the other one representing bad applicants. An applicant is classified as good if there have never been any payments overdue for 45 days or more, and bad if the payment has at least once been overdue for 46 days or more. The sample consisted of 66% goods and 34% bads. The accuracy of models is measured by total hit rates for correctly classified applicants using the threshold 0.5, but individual hit rates for good and bad applicants were also captured. The dataset was divided into the in-sample data (app.75% of data), and the out-of sample data (25% of data) used for the final validation. NNs, logistic regression, and CART decision trees were applied using the same in-sample and out-of-sample data.

6. RESULTS

6.1. RESULTS ON THE STOCK MARKET DATASET

In the stock market dataset the comparison was conducted separately for prediction and classification problem. The best overall NN result is obtained by a 3-layered backpropagation starting with 105, ending with 50 hidden units after pruning (see Table 1). The ATR of 0.841 is accompanied by a very high correlation of 0.9697. Variables V1 (moving average term), V2 (volume), V3 (volatility), V4 (beta), V6 (capital intensiveness), V8 (receivable intensiveness), and V9 (cash position) entered the best model. It is interesting that the forward strategy has selected only stock-related variables and two financial ratios, and none of the macroeconomic variables, indicating their insignificant influence to the stock return rate. Regarding the network topology, our findings are that 1 hidden layer is enough in all architectures to predict stock market return with a very high accuracy (83.5% by modular and 75.3% by radial-basis network). Both modular and radial-basis networks ended with 50 hidden units after pruning.

Table 1: Results of different NN architectures tested on the stock return prediction model

NN architecture	1 hidden layer			2 hidden layers		
	Correlation	Av. trade result	Input vars	Correlation	Av. trade result	Input vars
Backprop	0.9697	0.841*	V1,V2,V3,V4,V6,V8,V9	0.9175	0.771	V1,V2, V4,V5, V6,V8
Radial-basis function	0.9193	0.753	V1,V2,V4	0.9099	0.741	V1,V4
General regression	0.5103	0.488	V1,V4	-	-	
Modular	0.9733	0.835	V1,V2,V4,V6,V8,V9	-	-	

* the best average trade result {ATR) obtained in experiments

The results of the best NN architectures obtained by the four NN algorithms for classification are presented in Table 2.

Table 2: Results of different NN architectures tested on the the stock classification model

NN architecture	1 hidden layer			2 hidden layers		
	Correlation	Av. trade result	Input vars	Correlation	Av. trade result	Input vars
Backprop	0.511974	0.624	V1,V3	0.481883	0.612	V1,V3
Radial-Basis	0.50731	0.629	V1,V4, V6, V8, V10, V14, V16, V17, V19	0.51636	0.629	V1,V3, V4,V5
Probabilistic	0.448204	0.635*	V1,V4	-	-	
LVQ	0.470044	0.529	V1,V3,V4,V6,V9, V10, V11	-	-	

* the best average trade result {ATR) obtained in experiments

The best classification result on the model is obtained by the probabilistic network (ATR of 0.635), followed by radial-basis and backpropagation networks. It consisted of 1 hidden layer with 576 pattern units. The optimal value of parameter sigma was 2.1. The performance of the radial-basis network was not dependent on the number of hidden layers, while the backpropagation was better when only 1 hidden layer is used. Backpropagation and LVQ have selected a relatively large number of input variables, while others have chosen only 2 variables. The best architecture (obtained by the probabilistic network) uses only moving average term and beta as relevant inputs. The results obtained by classification of stock returns are surprisingly worse than by prediction. Although it is expected that the network will more easily identify classes than actual values of stock return, the classification NN algorithms seem to be less efficient. When looking into NN accuracy for each class: positive, hold and negative trade class, the hit rates obtained by the best NN model were the following: 90% for positive trades (buy action), 67.74% for hold trades (hold action), and 84.13% for negative trades (sell action). It can be seen that the individual trade actions accuracy is higher for buy and sell actions, and lower for hold actions. In order to see if NNs perform better than statistical methods, the best overall NN result is compared to the multiple stepwise linear regression. The best ATR obtained by the regression is 0.724 while the total return rate was 162.20. Model fitting was also high (R^2=96.2%) and F-test shows that all the selected variables are significant at the 15% level.

Table 3: Results of the best neural network and multiple regression models

Model	Estimated equation	Av. trade result	Pos. hit rate (buy) %	Hold hit rate (hold) %	Neg. hit rate (sell) %	TRR*
Backpropagation NN (input vars: V1,V2,V3,V4,V6,V8,V9)	-	0.841	90.00	67.74	84.13	171.77
stepwise multiple regression (input vars: V1, V2, V4)	y = 0.0005 + 1.932 * MA(1) + (- 0.000002) * Volume + (-104.757) * Beta R^2 = 0.96187181, F = 5667.74	0.724	80.26	41.94	77.78	162.20

* TRR is the total return rate

The results show that NNs outperform multiple regression yielding better average trade result, higher total return rate, as well as higher hit rates for each class of the stocks.

6.2. RESULTS ON THE CREDIT SCORING DATASET

When the suggested strategy for selecting NN architectures is applied on the second dataset, only NN algorithms for classification were tested, since the problem was to classify credit applicants into two categories: "good" and "bad". The backpropagation and radial-basis algorithms were used with additional softmax function in the output layer in order to obtain probabilities. The probabilistic and LVQ algorithms were also tested as standard classifiers. The strategy extracted the best NN architecture for each one of the four algorithms. Logistic regression and CART decision trees were conducted in order to compare the results with NNs (see more detailed description of this research in [21]), and the hit rates of all models obtained on the same out-sample data are shown in Table 4.

Table 4. NN, LR and CART results on the credit scoring dataset

Model	total hit rate (%)	hit rate of bads (%)	hit rate of goods (%)
Backprop NN, 6-50-2	73.80	53.33	85.19
RBFN, 5-50-2	71.40	73.33	70.37
Probabilistic NN, 10-106-2,	83.30*	80.00	85.19
LVQ NN, 2-20-2	61.90	13.33	88.89
Logistic regression	57.14	66.67	51.85
CART	66.67	66.67	66.67

* the highest total hit rate obtained in experiments

As in the first dataset, the overall highest result is obtained by the probabilistic NN model (total hit rate of 83.3%). This network also showed the highest hit rate in classifying bad applicants (bads hit rate of 80%). Among other NN architectures, the LVQ was the worst in perfomance, unable to classify more than 13.33% of bad applicants, while the backpropagation was the second best, followed by the RBFN. The strategy showed that the best structure for the probabilistic NN applied in this dataset was 10 input units and 106 pattern units. Concerning the variable selection, the best NN model extracted 10 input variables as important. The selection of variables showed that both personal and business characteristics are relevant in small business credit scoring systems.

7. CONCLUSION

The motivation for this research was the lack of paradigms in the area of NN methodology for selecting NN architectures, as well as a variety of results obtained by the researchers depending on the problem areas and algorithms used. The paper describes some previous efforts to overcome this limitation, and suggests a non-linear forward strategy for selecting NN architectures that is problem and algorithm independent. The strategy was tested on two independent datasets: the stock market data, and the credit scoring data, using four NN algorithms for prediction and four NN algorithms for classification type of problems. Among different NN architectures for prediction of stock return tested in our research, the most efficient NN architecture selected on the basis of the forward strategy was the 3-layered Backpropagation with 7 input, 50 hidden, and 1 output neurons, hyperbolic tangent transfer function and the EDBD learning rule. The results show that NNs' best result is 11.7% higher than the best stepwise regression's result, which can be considered as an important difference for investors on stock markets. Applied on the credit scoring dataset, the strategy extracted the probabilistic neural network, with 10 input units and 106 pattern units as the best NN architecture with the hit rate of 83.3%. The model obtained by this architecture showed the best overall result, comparing to the multiple regression and CART decision tree models.

It is obvious that there is no unique method which can be considered the best for prediction or classification. The choice of method, as well as the result depend on the observed data. The fact that NNs give better result on the observed data confirms the advantage of NNs over the multivariant linear regression in stock return data and over the logistic regression in credit scoring data. Although tested on limited data, the suggested selecting strategy enables an extensive test of NN architectures on any theoretical model, therefore providing a useful tool for NN researchers and practitioners.

The research can be further improved by a sensitivity analysis that could bring additional information for effective modelling. The stability of NN results should also be considered, as well as the inclusion of other NN algorithms. The selection of methods can be extended by including other data mining techniques, such as support vector machines, regularization and other algorithms that should be tested in order to find some latent characteristics of the observed data.

REFERENCES

[1] A.Doering, M. Galicki, H. Witte. Structure Optimization of Neural Networks with the A*-algorithm. IEEE Transactions on Neural Networks, Vol. 8, No. 6, 1997, pp. 1434-1445.

[2] D. Enke, S. Thawornwong, The use of data mining and neural networks for forecasting stock market returns, Expert Systems with Applications, Vol. 29, No. 4, 2005, pp. 927-940.

[3] R.A. Jacobs, M.I. Jordan, S. Nowlan, G.E. Hinton. Adaptive Mixtures of Local Experts. Neural Computation, No. 3, 1991, pp. 79-87.

[4] O. Korn, U. Anders. Model selection in neural networks. Neural Networks, Vol. 12, 1999, pp. 309-323.

[5] E.Y. Li. Artificial Neural Networks and Their Business Applications. Information & Management, Vol. 27, 1993, pp. 303-313.

[6] M. Lam. Neural network techniques for financial performance prediction: integrating fundamental and technical analysis. Decision Support Systems, Vol. 37, No. 4, pp. 567-581.

[7] D. Marcek. Neural Network Model for Stock Prices Forecasting. Neural Networks World, No. 3, 1997, pp. 347-352

[8] T. Masters. Advanced Algorithms for Neural Networks. A C++ Sourcebook, John Wiley & Sons, New York, 1995.

[9] T. Masters. Just what are we optimizing anyway?, International Journal of Forecasting, Vol. 14, 1998, pp. 277-290.

[10] NeuralWare. Neural Computing, A Technology Handbook for NeuralWorks Professional II/Plus and NeuralWorks Explorer, NeuralWare, Aspen Technology, Pittsburgh, 1998.

[11] P.T. Quinlan. Structural Change and Development in Real and Artificial Neural Networks. Neural Networks, No. 11, 1998, pp. 577-599.

[12] A.N. Refenes, A.N. Burges, Y. Bentz. Neural Networks in Financial Engineering: A Study in Methodology, IEEE Transactions on Neural Networks, Vol. 8, No. 6, 1997, pp. 1222-1267.

[13] A.N. Refenes, A. Zapranis, G. Francis. Stock Performance Modeling Using Neural Networks: A Comparative Study with Regression Models, Neural Networks, Vol. 7, No. 2, 1994, pp. 375-388.

[14] E.W. Saad, D.V. Prokhorov, D.C. Wunsch. Comparative Study of Stock Trend Prediction Using Time Delay, Recurrent and Probabilistic Neural Networks, IEEE Transactions on Neural Networks, Vol. 9, 1998, pp.1456-1470.

[15] R.R. Trippi, E. Turban. Neural Networks in Finance and Investing. Irwin Professional Publishing, Chicago, 1998.

[16] H. White. Economics Prediction Using NN; The Case of IBM Daily Stock Returns. In R.R. Trippi, E. Turban, ed., Neural Networks in Finance and Investing, Irwin Professional Publishing, Chicago, 1996, pp. 469-481.

[17] B.K. Wong, T.A. Bodnovich, Y. Selvi. Neural Network Applications in Business: A Review and Analysis of the literature (1988-95), Decision Support Systems, Vol. 19, 1997, pp. 301-320.

[18] F. Zahedi. A Meta-Analysis of Financial Applications of Neural Networks, International Journal of Computational Intelligence and Organizations, Vol. 1, No. 3, 1996, pp. 164-178.

[19] M. Zekic. Neural Network Applications in Stock Market Predictions – A Methodology Analysis, Proceedings of the 9th International Conference on Information and Intelligent Systems '98, Varazdin, 1998., pp. 255-263.

[20] M. Zekic-Susac, B. Klicek. A Nonlinear Strategy of Selecting NN Architectures for Stock Return Predictions. Finance, Proceedings from the 50th Anniversary Financial Conference Svishtov, Bulgaria, 11-12 April, 2002, pp. 325-355.

[21] M. Zekic-Susac, N. Sarlija, M. Bensic. Small Business Credit Scoring: A Comparison of Logistic Regression, Neural Network, and Decision Tree Models, Proceedings of the 26th International Conference on Information Technology Interfaces, June 7-10, 2004, Cavtat, pp. 265-270.

Received: 20 January 2005
Accepted: 18 November 2005