Paper 149-2011

Genetic Algorithm Optimization for Selecting the Best Architecture of a Multi-Layer Perceptron Neural Network: A Credit Scoring Case

Alejandro Correa, Banco Colpatria
Andrés González, Banco Colpatria
Camilo Ladino, Banco Colpatria

ABSTRACT

Neural Networks are powerful tools for classification and regression, but it is difficult and time costly to determine the best architecture for a given problem. In this paper, Genetic Algorithms (GA) are used to optimize the architecture of a Multi-Layer Perceptron Neural Network (MLP) in SAS®, in order to improve the predictive power of the credit risk scorecards. The objective function to maximize is the ROC curve and the input variables are the number of hidden layers and units, activation function, use or not of bias and whether it will be a direct connection between the initial and the final layer. Results show that this method outperforms logistic regression and the default neural network architecture of SAS Enterprise Miner™. The predictive power of this method is similar to the Global Optimum but in a reasonable time.

INTRODUCTION

In order to mitigate the impact of credit risk and make more objective and accurate decisions, financial entities have created new and better tools to predict and control their losses (Anderson, 2007) (Lawrence & Solomon, 2002). This is why it has become common in financial institutions around the world to use scorecards to measure a customer's credit risk (Abranhams & Zhang, 2009) (Mays, 2004) (Thomas, 2009). A scorecard is a statistical model that allows attributing a rating (score) to a client, which indicates the predicted probability that the customer reflect a certain behavior. What is sought with scorecards is to create an estimated measure of a customer's risk, i.e., the probability of a customer having a good payment habit if a loan is granted, based on past experiences (Thomas, 2002). The most commonly used method by financial institutions to estimate these models is the logistic regression (Allison, 2003), because of its predictive power and ease of interpretation. But there are other methods, such as Neural Networks, which have a higher level of complexity that could improve the predictive power of the scorecards.

Neural Networks aren't widely used in credit scoring due to two main reasons, i) the difficulty with interpretability and ii) the complexity in model development. When developing a Multi-Layer Perceptron (MLP) Neural Network, analysts have to address several kinds of architectural issues. In this paper, an optimization of the architecture of the Multi-Layer Perceptron Neural Network is made using an implementation of Genetic Algorithm in the SAS system (SAS Institute Inc., 2010).The objective function to maximize is the ROC curve(Receiver operating characteristic) and the decision variables are the number of hidden layers and their activation function, the number of hidden units in each layer, the activation function of the target layer, and whenever or not to use bias or to have a direct connection between the input and output layer. Although, a similar optimization methodology has been developed in other fields (Carstern & Paasch, 2008), to our knowledge, this methodology haven't been applied to improve credit risk scorecards.

This paper is divided into five sections. First, a description of the data and the variables used for the model development is made. Subsequently, there is an introduction to general concepts of Multi-Layer Perceptron Networks (MLP) and Genetic Algorithms (GA) methodology. The third section poses the specific definitions for modeling with the MLP and the GA. Next, the results are shown based on a comparison with a default Neural Network architecture developed in SAS Enterprise Miner™ (Matignon, 2005), and a logistic regression. Also, all possible architectures of the MLP for our case of study are calculated, and the best one (Global optimum) is compared. Finally, conclusions are presented.

DATA DESCRIPTION

For the model construction, 125.557 clients with active credit cards in June 2009 are used. Using the bank's default definition over the performance period, clients are classified into good and bad. Variables names are changed to X1 … X7 by request of the financial institution (clients credit information is confidential).The original variables have been standardized and Table 1 displays the descriptive statistics of the seven variables, while Table 2 presents the correlation between them. The maximum correlation between the seven variables is 0,0176. Finally, Table 3 shows how the original data is randomly divided into three different datasets used for the scorecard development and validation.

Table 1. Descriptive statistics of the variables

Variable	N	Mean	Std. Dev.	Minimum	Maximum
X1	125.557	-0,027	5,823	-5,615	53,637
X2	125.557	0,002	1,629	-66,097	52,220
X3	125.557	0,003	1,511	-13,606	20,037
X4	125.557	0,000	1,376	-13,579	33,091
X5	125.557	0,014	2,108	-10,655	21,943
X6	125.557	-0,014	1,710	-8,699	38,439
X7	125.557	-0,002	1,656	-63,313	122,732

Table 2.Correlation Matrix

Variable	X1	X2	X3	X4	X5	X6	X7
X1	1	0,0019	-0,0051	0,0046	0,0176	0,0039	-0,0003
X2	0,0019	1	-0,0026	0,0142	-0,0033	-0,0009	-0,0004
X3	-0,0051	-0,0026	1	0,0111	-0,0059	-0,0015	-0,0094
X4	0,0046	0,0142	0,0111	1	-0,0022	0,0014	-0,0093
X5	0,0176	-0,0033	-0,0059	-0,0022	1	-0,0023	-0,0013
X6	0,0039	-0,0009	-0,0015	0,0014	-0,0023	1	0,001
X7	-0,0003	-0,0004	-0,0094	-0,0093	-0,0013	0,001	1

Table 3.Development and validation datasets

Data	N	Percentage of the total population	Bad Rate
Train	50.223	40,00%	56,48%
Test	37.667	30,00%	56,68%
Validation	37.667	30,00%	56,84%
Total	125.557	100,00%	56,65%

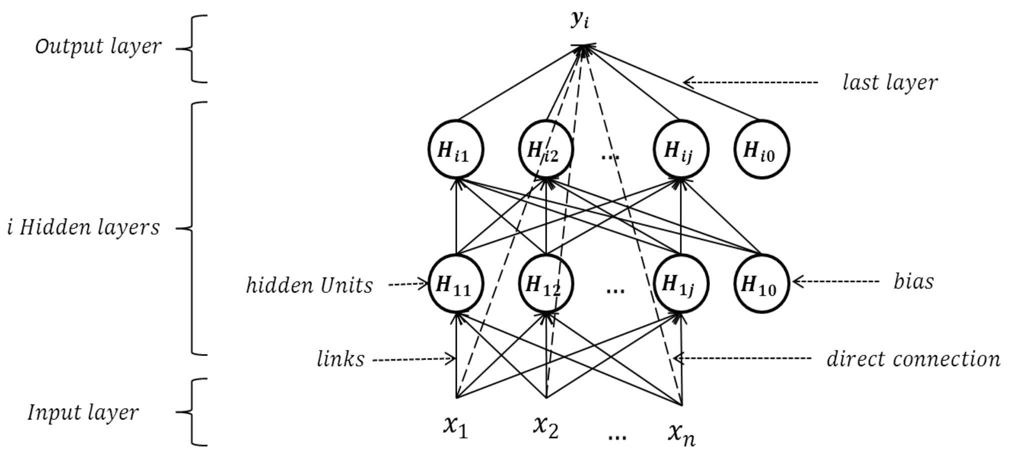
GENERAL CONCEPTS

- ### MULTI-LAYER PERCEPTRON NEURAL NETWORK

An artificial neural network is a mathematical/computational model that tries to imitate the structure and functionality of biological neural networks (Rosenblatt, 1962). It is composed by a set of simple computational units that are highly interconnected. These units are called nodes, and each one represents a biological neuron. In a neural network, the hidden units receive a weighted sum of the inputs and apply an activation function to it. Then, the output units receive a weighted sum of the hidden units output and apply an activation function to this sum. Information is passed from one layer to the next. The neural network finds the weights by an iterative process through different types of algorithms.

The network discussed in this paper is called a Multi-Layer Perceptron Neural Network and it has some specific characteristics. In order to easily explain the MLP neural network structure, Figure 1 shows the main components. It has an input layer that represents the input variables to be used in the neural network model and it can be connected directly with the output layer. It also has *i* hidden layers and each layer contains *j* hidden units. In Figure 1 the hidden units are represented by circles. The connections between units are unidirectional and are represented by directed lines. Each connection has an associate scalar called weight *w*. The hidden units have a variety of hidden activation functions and also a linear combination function. Finally, the MLP has an output layer that computes for the result of the process. The output layer also has a target activation function. Both, the hidden layers and the output layer could have the bias option activated. A bias term can be treated as a connection weight from a special unit with a constant, nonzero activation value. The term "bias" is usually used with respect to a "bias unit" with a constant value of one.
The single bias unit is connected to every hidden or output unit that needs a bias term. Hence the bias terms can be learned just like other weights.

Figure 1. MLP Neural Network structure



- **GENETIC ALGORITHM**

  Genetic Algorithm (GA) is an optimization technique that attempts to replicate natural evolution processes in which the individuals with the considered best characteristics to adapt to the environment are more likely to reproduce and survive. These advantageous individuals mate between them, producing descendants similarly characterized, so favorable characteristics are preserved and unfavorable ones destroyed, leading to a progressive evolution of the species.

  Artificial genetic algorithm aims to improve the solution to a problem by keeping the best combination of input variables. It starts with the definition of the problem to optimize, generating an objective function to evaluate the possible candidate solutions (chromosomes), i.e., the objective function is the way of determining which individual produces the best outcome.

  The next step is to generate an initial random population of *n* individuals called chromosomes that are symbolized by binary strings, where each binary position of the chromosome is called a gene and denotes a specific characteristic (input variable). Therefore the combination of all the different characteristics encoded in the string represents an individual who is a candidate for the solution.

  Each chromosome is evaluated in the objective function and the best individuals are selected to survive for mating (parents), while the worse ones are discarded to make room for new descendants. There are many ways of pairing the selected chromosomes (Haupt & Haupt, 2004). In this paper, a weighted cost pairing is used, which consists of assigning a selection probability according to each chromosome cost. That is, a chromosome with the higher cost has a greater probability of mating because cost maximization is desired.

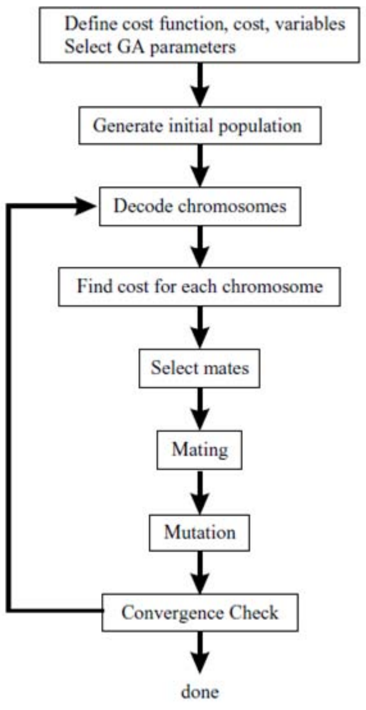
  After selecting the parent chromosomes with the chosen pairing method, the next step is to create a second generation of individuals, based on the information of the parents. There are several ways of mating (Haupt & Haupt, 2004).In this paper, two parents create one child.

  In order to transfer the parent's binary information to the child, there are genetic operators such as crossover point and mutation. The one-point crossover technique consists in selecting one random point on the parent's string. The child is created in the following way: First, the parent$_1$ transfers its binary code from the first gene to the crossover point. Then the parent$_2$ transfers its binary code from the crossover point to the last gene of the chromosome. New parents are randomly selected for each new child and the process continues until the chromosome population grows back to the original size *n*.

  Once the breeding process is completed, random mutation is used to alter a certain percentage of the genes of the chromosomes. The purpose of mutation is to introduce diversity into the population, allowing the algorithm to avoid local minima by generating new gene combinations in the chromosomes. The most common mutation procedure is the one called single point mutation. It's implemented by generating a random variable that indicates the position of the gene that will be modified, from the population of chromosomes. Generally, mutation is not allowed in the best solution chromosomes because these "elite" individuals are destined to propagate unchanged. In genetic algorithm this is called elitism (Haupt & Haupt, 2004).

  Finally, after mutation is done the new generation of chromosomes is evaluated with the objective function and used in the next iteration of the described algorithm. The algorithm iterates until a maximum number of chromosome generations are created or a satisfactory solution is reached.

A flowchart of the described process is presented below:


Define cost function, cost, variables
Select GA parameters
Generate initial population
Decode chromosomes
Find cost for each chromosome
Select mates
Mating
Mutation
Convergence Check
done

Flow diagram extracted from (Haupt & Haupt, 2004)

MODELING

Now that the general concepts of MLP neural Networks and genetic algorithms have been covered, it is time to focus on the specific case of this paper. First, a definition about the activation and combination functions in the neural network is presented. Given that in credit scoring the objective is to obtain a predicted probability to reflect a certain behavior of a client, the MLP neural network target activation functions have been bounded to functions with range between 0 and 1. Table 4.a. and Table 4.b. present the activation and combination functions used in this paper. Second, the discussed network finds the weights through a Backpropagation algorithm (Warner & Misra, 1996).

Table 4.a. Hidden Layers Functions

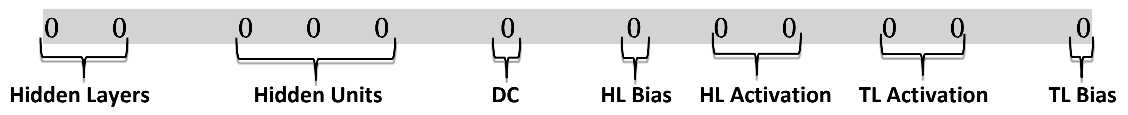
Hidden Combination Function	**Hidden Activation Function**	**Hidden Activation Function Range**
Linear: $bias + \quad w_i x_i$	Linear: x	$(-\infty, \infty)$
	Logistic: $\frac{1}{1 + exp^{-x}}$	$(0,1)$
	arctan: $Tan(x)$	$(-1,1)$
	Hyperbolic angent: $Tanh\ x = \frac{e^x - e^{-x}}{e^x - e^{-x}}$	$(-1,1)$

Table 4.b. Target Layer Functions

Target Combination Function	**Target Activation Function**
Linear: $bias + \quad w_i x_i$	Logistic: $\frac{1}{1 + exp^{-x}}$
	Mlogistic: $\frac{e^x}{exponentials}$
	Softmax: $\frac{e^x}{exponentials}$
	Gauss: e^{-x^2}

In addition, for the genetic algorithm modeling the ROC curve (Receiver operating characteristic) is chosen as the GA objective function to maximize because it measures the neural network capability to assign and rank relatively more low scores to loans that eventually become bad than to loans that continue with a good behavior. The ROC is also known as the swap curve since it represents the exchange between good clients and bad clients, i.e., the percentage of bad clients to be allowed in order to accept a certain percentage of the good clients.

Subsequently, the definition of the input variables and the chromosome structure is carried out. Seven input variables are selected to form the chromosome that is going to have a total of 12 genes which can generate a total of 4.096 (2^{12}) possible combinations. The chromosome structure is defined as follow:



Likewise, the variables of the chromosomes are encoded in the following manner:

Hidden Layers	Hidden Units	Direct Connection	Hidden Layers Bias	Hidden Layers Activation Function	Target Layer Activation Function	Target Layer Bias
00 = 1	000 = 1	0 = No	0 = No	00 = Logistic	00 = Logistic	0 = No
01 = 2	001 = 2	1 = Yes	1 = Yes	01 = Linear	01 = Mlogistic	1 = Yes
10 = 3	…			10 = Act Tan	10 = Softmax	
11 = 4	111 = 8			11 = Tan H	11 = Gauss	

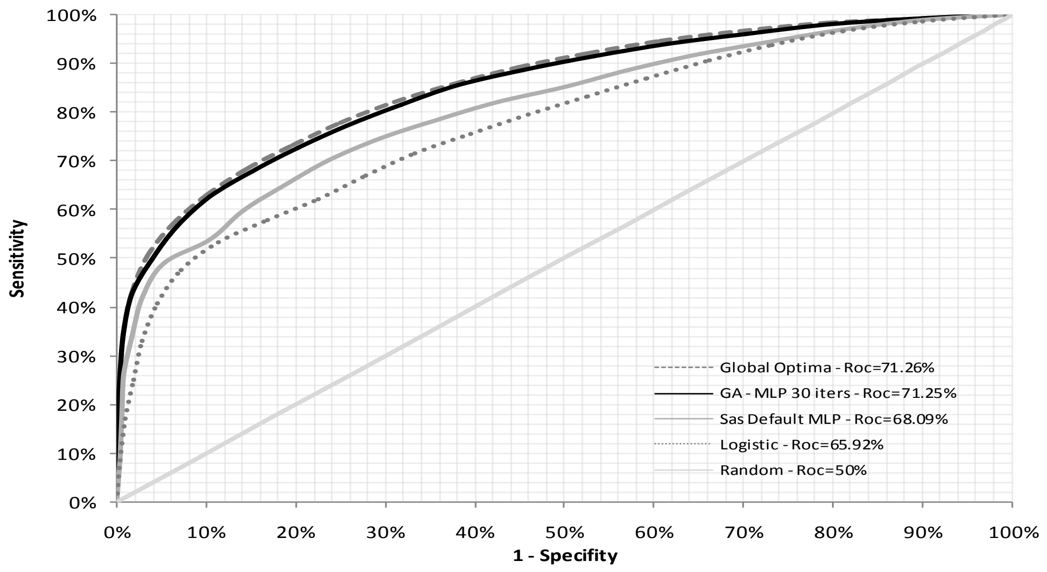
Finally, other key definitions are the total size of the population, the number of “elite” individuals and the percentage of genes to mutate from the entire chromosome population. Correspondingly, the population size is 16 individuals (chromosomes), the best four solution chromosomes will remain unchanged and the percentage of mutation is 2% of the genes of the total population.
In order to facilitate the understanding of the genetic algorithm (GA) iterative process, the pseudo-code is attached.

RESULTS

In this section we present the experimental results of the genetic algorithm used to select the best architecture of the MLP neural network. Results of a run of 30 iterations with the GA are compared with the results of a neural network using the default parameters of SAS Enterprise Miner™ (1 hidden layer, 3 hidden units, no direct connection, hidden layer bias = yes, Tan H hidden layer activation function, logistic target layer activation function, target layer bias = yes), a logistic regression (most common algorithm in credit scoring) using SAS PROC LOGISTIC with stepwise variable selection and the global optimum.

The comparison of the ROC curves obtained by the Genetic Algorithm in the MLP network and the other three alternatives are exhibited in Figure 2. The area under the ROC curve of the GA (71,25%) is significantly larger than that of the MLP neural network using the default parameters of SAS Enterprise Miner™ (68,09%) and the logistic regression (65,92%). This difference indicates that the GA in the MLP neural network has a greater predictive power at all risk levels. The only alternative that slightly exceeds the GA performance is the global optimum (71,26%) but the difference is so small that it doesn´t represent a significant improvement in predictive power to justify the great additional effort (see Table 5).

Figure 2. Comparison of ROC curve



In addition to the statistical evidence presented in the ROC curve, Table 5 shows the computational effort of each alternative through two measures, the total time spent in minutes (CPU time in minutes) and the number of times that the SAS function was called. Since the logistic regression and the MLP neural network using the default parameters of SAS Enterprise Miner™ are run only once, both have only one function call and spent 1 and 2 minutes respectively. As presented above, this two alternatives show the worst predictive power measured by the ROC curve.

The GA used to optimize the MLP neural network architecture spent 9,3 hours (559 minutes) in a 30 iteration run and made 274 function calls while the model used to find the global optimum took 139,2 hours (8.356 minutes) and made 4.096 function calls (all possible combinations). These last two alternatives have approximately the same predictive power and the difference in computational effort is evident. The GA spent 1.395% less time and function calls than the global optimum finding method.

Table 5. Measures of comparison

Model	ROC	CPU time (m)	Function calls
Default MLP	68,09%	2	1
Logistic Regression	65,92%	1	1
GA - MLP 30 iterations	71,25%	559	274
Global Optimum	71,26%	8.356	4.096

Finally, table 6 displays the final architecture of the neural networks found with the GA and the global optimum model.

Table 6. Best MLP neural network architecture

Model	Hidden Layers	Hidden Units	Direct Connection	Hidden Layers Bias	Hidden Layers Activation Function	Target Layer Activation Function	Target Layer Bias
GA - MLP 30 iters	2	6	0	1	TAN	0	SOF
Global Optimum	2	5	1	1	LOG	0	LOG

CONCLUSION

This paper have shown the use of Genetic Algorithms in credit risk modeling as a technique to optimize the process of choosing the architecture of a MLP neural network that maximizes the area under the ROC curve and therefore the scorecard predictive power.

The experimental results have shown that with far less computational effort the GA used to optimize the MLP neural network came to a result approximately equal to the global optimum. Also, since the difference between the ROC curves of these two alternatives is negligible, we illustrate that it doesn´t represent an improvement in the scorecard predictive power. It is also important to say that the GA outperformed the results of the logistic regression and the results of the MLP neural network using the default parameters of SAS Enterprise Miner™.

PSEUDO - CODE

The SAS Code may be sent on request to authors.

```
Variable Integer Num_Chromosomes
Variable Integer Num_Genes
Variable Integer Num_Elitism
Variable Integer Num_iterations
Variable Integer Per_Mutations
Variable Boolean Chromosomes ( Num_Chromosomes , Num_Genes)
Variable Double  Cost ( Num_Chromosomes)
Variable Double  Prob(Num_Chromosomes)
Variable Integer Father (Num_chromosomes – Num_Elitism)
Variable Integer Mother ( Num_chromosomes – Num_Elitism)
Variable Boolean Child ( Num_chromosomes – Num_Elitism , Num_Genes)
Variable Boolean Elitism ( Num_Elitism , Num_Genes)

Function Decoding
// Function that convert a chromosome into Neural Network parameters

Function Calculate_Cost
```

```
    Architecture = Call Decoding( Chromosome )
    Net    = Call Neural_Networks( Architecture )
    Return  CallRoc ( Net )
End Function

Function Genetic_Algorithm

Cromosomes (all,all) = Call Create_random_population
For iter = 1 to Num_iterations
    For  i = I to Num_Chromosomes
          Cost( i )  = Call Calculate_Cost ( Cromosomes ( i , all ) )
    End For

    // Calculate cumulative matching probability
    For i = I to Num_Chromosomes
          Prob(i) = Prob(i-1) + Cost( i )  / sum ( Cost( all) )
    End For

    // Selection
    For k = 1 to Num_Chromosomes - Num_Elitism
          Rand = Call Random(Uniform,0,1)
          Father( k ) = 0
          For i  = 1 to Num_Chromosomes
                If Rand <= Prob( i ) And Father ( k ) = 0  Then
                      Father( k ) = i
                End If
          End For
          Rand = Call Random(Uniform,0,1)
          Mother( k ) = 0
          For i  = 1 to Num_Chromosomes
                If Rand <= Prob( i ) And Mother ( k ) = 0  Then
                      Mother( k ) = i
                End If
          End For
    End For

    // Mating
    For k = 1 to Num_Chromosomes - Num_Elitism
          Rand = Call Random(Uniform,1,Num_Genes)
          For j = 1 to Num_Genes
                If j <= Rand Then
                      Child( k , j ) = Chromosome ( Father (k) , j )
                Else
                      Child( k , j ) = Chromosome ( Mother (k) , j )
                End If
          End For
    End For

    // Elitism
    Call Sort ( Chromosomes by Cost )
    For k=1 to Num_Elitism
          Elitism( k , all ) = Chromosomes ( k , all )
    End For

    // Replace
    For i = 1 to Num_Chromosomes
          If i<= Num_Elitism Then
                Chromosomes( i , all ) = Elitism ( i , all )
          Else
                Chromosomes( i , all ) = Child ( i + Num_Elitism , all)
          End If
    End For

    // Mutations
    For m = 1 to Per_Mutations * Num_Chromosomes * Num_Genes
          R1 = Call Random(Uniform,1,Num_Chromosomes)
          R2 = Call Random(Uniform,1,Num_Genes)
          Chromosomes( R1, R2 ) = (Chromosomes(R1,R2) - 1 )  ^ 2
    End For
End For
```

```
   End Function
```

REFERENCES

Abranhams, C. & Zhang, M. (2009). Fair Lending Compliance. NewJersey: John Wiley & Sons, Inc.
Allison, P. D. (2003). Logistic Regression using the SAS system: Theory and Application. Cary, United States of America: SAS Institute and Wiley.
Anderson, R. (2007). The credit scoring toolkit: Theory and practice for retail credit risk management and decision automation. New York: Oxford University press Inc.
Carsten, A.W. Paasch (2008). Credit Card Fraud Detection Using Artificial Neural Networks Tunned By Genetic Algorithms.The Hong Kong University of Science and Technology.
Haupt, Randy. Haupt, Sue (2004). Practical Genetic Algorithms. Second edition. New Jersey: John Wiley & Sons.
Lawrence, D., & Solomon, A. (2002). Managing a consumer lending business. New York: Solomon
Matignon, Randall (2005). Neural Network Modeling using SAS Enterprise Miner. Author House.
Mays, E. (2004). Credit Scoring for Risk Managers. The Handbook for Lenders. Mason, Ohio, United States of America: Thomson South-Western.
Rosenblatt, F. (1962). Principles of Neurodynamics. Washington, DC: Spartan.
SAS Institute Inc. (2010). SAS help and documentation. PROC NEURAL. Cary, North Carolina, United States of America.
Thomas, L. C. (2002). Credit Scoring and its applications. Philadelphia: Siam.
Thomas, L. C. (2009). Consumer Credit Models: Pricing, Profit, and Portfolios. New York: Oxford
Warner,B. & Misra, M. (1996). Understanding Neural Networks as Statistical Tools. The American Statistician Association. URL:http://www.jstor.org/stable/2684922 (Accessed: 03/01/2011).

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Name:	Alejandro Correa Bahnsen
Enterprise:	Banco Colpatria
City:	Bogotá, Colombia
Phone:	(+57) 3208306606
E-mail:	correaal@colpatria.com

Name:	Andrés González
Enterprise:	Banco Colpatria
City:	Bogotá, Colombia
Phone:	(+57) 3103595239
E-mail:	gonzalean@colpatria.com

Name:	Camilo Ladino
Enterprise:	Banco Colpatria
City:	Bogotá, Colombia
Phone:	(+57) 3185865390
E-mail:	ladinoi@colpatria.com