2011 11th IEEE International Conference on Data Mining Workshops

Evolutionary algorithms for selecting the architecture of a MLP Neural Network: A Credit Scoring Case

Alejandro Correa B., Andres Gonzalez M.
Banco Colpatria
Bogotá, Colombia
{correaal, gonzalean}@colpatria.com

***Abstract*—Neural Networks are powerful tools for classification and Regression, but it is difficult and time consuming to determine the best architecture for a given problem. In this paper two evolutionary algorithms, Genetic Algorithms (GA) and Binary Particle Swarm Optimization (BPS), are used to optimize the architecture of a Multi-Layer Perceptron Neural Network (MLP), in order to improve the predictive power of the credit risk scorecards. Results show that both methods outperform the Logistic Regression and a default neural network in terms of predictability, but the GA are more time consuming than the BPS. The predictive power of both methods is similar to the Global Optimum but it is found in a reasonable time.**

Keywords; genetic algorithm; particle swarm optimization; credit scoring; neural networks.

I. 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 [4][6]. This is why it has become common in financial institutions around the world to use scorecards to measure a customer's credit risk [1][8][12]. 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 [11]. The most commonly used method by financial institutions to estimate these models is the Logistic Regression [2], because of its predictive power and ease of interpretation. But there are other methods, such as Neural Networks [7] 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 issues regarding the Neural Network parameters or architecture. In this paper, an optimization of the architecture of the MLP Neural Network is made using two optimization techniques: Genetic Algorithm (GA) and Binary Particle Swarm (BPS) Optimization. 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 whether 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 in the case of the GA [3][18], to our knowledge, these methodologies haven't been applied to improve credit risk scorecards.

This paper is divided into six 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 MLP Neural Networks, GA and BPS techniques. The third section poses the specific definitions for modeling with the MLP and the optimization techniques. Next, the results are shown based on a comparison of the optimization algorithms with a default Neural Network architecture, and a Logistic Regression. Also, all possible architectures of the MLP for our case of study are calculated, and the Global optimum is compared. Then, the algorithm is also used to develop two additional models and the impact generated by optimization methodologies is shown. Finally, a discussion is made regarding the approaches used in this paper and some specific issues.

TABLE I. 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

Using information provided by a local bank, 125,557 clients with active credit cards are used for the model construction, and with 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. The original variables have been

978-0-7695-4409-0/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDMW.2011.80

standardized and TABLE I displays the descriptive statistics of the seven variables, while TABLE II presents the correlation between them. The maximum correlation between the seven variables is 0.017. Finally, TABLE III shows how the original data is randomly divided into three different datasets used for the scorecard development and validation.

TABLE II. CORRELATION MATRIX

	X1	*X2*	*X3*	*X4*	*X5*	*X6*	*X7*
X1	1	0.001	-0.005	0.004	0.017	0.003	0.000
X2	0.001	1	-0.002	0.014	-0.003	0.000	0.000
X3	-0.005	-0.002	1	0.011	-0.005	-0.001	-0.009
X4	0.004	0.014	0.011	1	-0.002	0.001	-0.009
X5	0.017	-0.003	-0.005	-0.002	1	-0.002	-0.001
X6	0.003	0.000	-0.001	0.001	-0.002	1	0.001
X7	0.000	0.000	-0.009	-0.009	-0.001	0.001	1

TABLE III. 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%

II. GENERAL CONCEPTS

A. 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 [9]. 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. Information is passed from one layer to the next. Then, the output units receive a weighted sum of the hidden units output and apply an activation function to this sum. 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 (MLP) and it has some specific characteristics. In order to easily explain the MLP Neural Network structure, Fig. 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 Fig. 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.

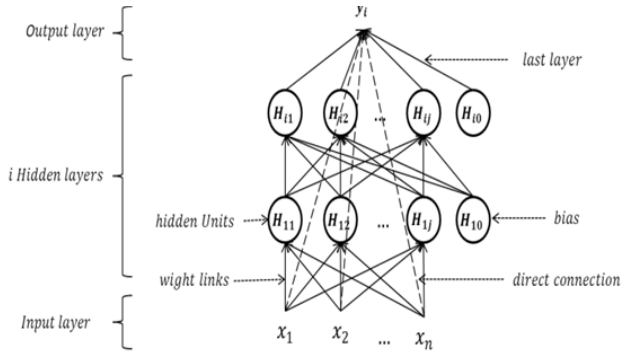

Figure 1. MLP Neural Network structure.

B. Genetic Algorithm

A 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 the progressive evolution of the species.

GA aims to improve the solution to a problem by keeping the best combination of input variables. The flow diagram presented in Fig. 2 describes the process. 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 [5]. 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.

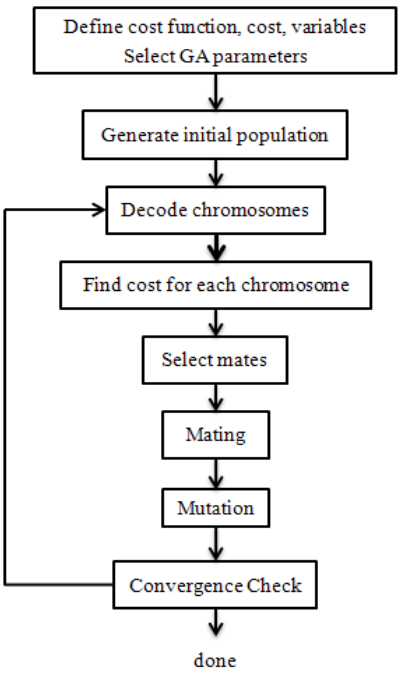

Figure 2. GA Flow diagram [5].

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 [5]. In this paper, two parents create one child.

In order to transfer the parent's binary information to the child, there are also different kinds of approaches such as the one-point crossover. 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 *parent1* transfers its binary code from the first gene to the crossover point. Then the *parent2* 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 [5].

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.

C. Binary Particle Swam Optimization

Particle Swarm (PS) is a population based algorithm that was introduced by Eberhart and Kennedy [14][15] to simulate the social behavior and movements of animals when they are together in a swarm or a shoal. Then, the algorithm was used as a computation technique to optimize the solution of a problem using a population of candidate solutions called "particles". These particles move along the search space based on mathematical calculations regarding their position and velocity. Each particle next movement is affected by the inertia of the current movement, the best position it has explore so far and the global best position explored in the search space by all the particles of the swarm. This method seeks to move the swarm toward the best solution.

For the specific case of this paper, a binary implementation of the particle swarm (BPS) optimization algorithm proposed by Khanesar, Teshnehlab and Shoorehdeli [17] is used instead of the original binary algorithm of Kennedy and Eberhart [16], because the original algorithm presents some limitations regarding the parameters and the memory of the particles [17].

The flow diagram presented in Fig. 3 describes the process for the BPS. It begins with the designation of the cost function based on the problem to solve and the parameters *w*, *c1* and *c2*, where *w* is the inertia of the current velocity, and *c1*, *c2* are fixed variable defined by the user.

The next step is to randomly initialize the *n* particles of the swarm within the search space. To do this, each binary value of the particles (input variable), called bit, is randomly set to 0 or 1. Once the initialization is done, each particle is decoded to the real values of the input variables and the performance is evaluated in the cost function. Then the performance of each particle is compared with its best solution found so far (P_{ibest}) and also with the best global solution found by the swarm (P_{gbest}). For each case, if the current position of the particle presents a better solution than its best solution found so far, then (P_{ibest}) is updated with the new particle position. Likewise, if the best solution of the current positions of the particles is higher than the best global solution found by the swarm, then (P_{gbest}) is updated.

The next step is to update the velocity V_i^c for each particle. The velocity refers to how fast the particle moves in though the search space. Then the new position of each particle is calculated based on the previous value of the particle (inertia) and the velocity.

Finally, the convergence criterion is verified. If the algorithm meets the convergence criterion them it stops, otherwise, the performance of the new positions of the particles are evaluated in the cost functions and are used in the next iteration of the algorithm. The BPS process can iterate for a fixed number of times or until a satisfactory

solution is reached. For more detail information about the formulas used in the BPS refer to Khanesar, Teshnehlab and Shoorehdeli [17].

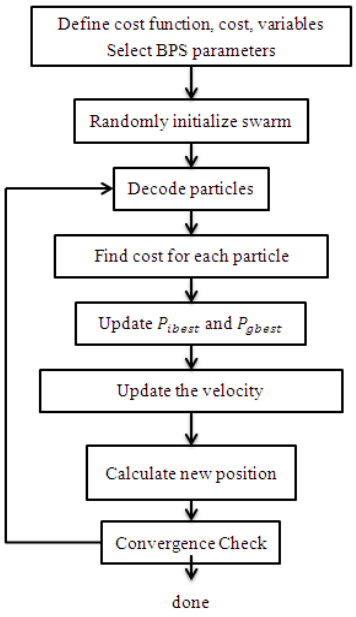

Figure 3. BPS Flow diagram.

III. MODELING

Now that the general concepts of MLP Neural Networks, GA and BPS 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 MLP 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 TABLE IV and TABLE V present the activation and combination functions used in this paper. Second, the discussed network finds the weights through a backpropagation algorithm [13].

TABLE IV. HIDDEN LAYERS FUNCTIONS

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

TABLE V. HIDDEN LAYERS FUNCTIONS

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

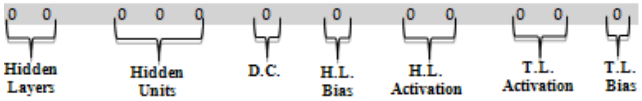

Figure 4. Chromosome/Particle Structure.

In addition, for both optimization procedures the ROC curve (Receiver Operating Characteristic) is chosen as the 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 structure of the chromosome in the case of the GA and the structure of the particle in the case of the BPS are carried out. These definitions are the same in both algorithms; the only difference is the way of referring to them. Seven input variables are selected to form the chromosome/particle that is going to have a total of 12 genes/bits which can generate a total of 4,096 (2^{12}) possible combinations. The chromosome/particle structure is defined in Fig 4.

TABLE VI. VARIABLES ENCODING

Hidden Layers	***Hidden Units***	***Direct Connection***	***Hidden Layers Bias***
00 = 1	000 = 1	0 = No	0 = No
01 = 2	001 = 2	1 = Yes	1 = Yes
10 = 3	…		
11 = 4	111 = 8		

Hidden Layers Activation Function	***Target Layer Activation Function***	***Target Layer Bias***
00 = Logistic	00 = Logistic	0 = No
01 = Linear	01 = MLogistic	1 = Yes
10 = Act Tan	10 = Softmax	
11 = Tan H	11 = Gauss	

Likewise, the variables of the chromosomes/particle are encoded as shown in TABLE VI.

Finally, there are some key definitions that are specific to each of the optimization algorithms. For the GA there is 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 the case of the BPS there is the population size that is equal to 10, the number of iterations is set to 10, the inertia weight is 0.6, and both *c1* and *c2* are set to 0.6.

IV. RESULTS

A. Experimental results

In this section we present the results of the GA and the BPS used to select the best architecture of the MLP Neural Network. Results of a 30 iterations run with the GA, 10 iterations with the BPS are compared with the results of a Neural Network using the default parameters of SAS Enterprise Miner™ [10] (1 hidden layer, 3 hidden units, no direct connection, hidden layer bias, hyperbolic tangent hidden layer activation function, logistic target layer activation function, target layer bias), a Logistic Regression (most common algorithm in credit scoring) [2], and the global optimum.

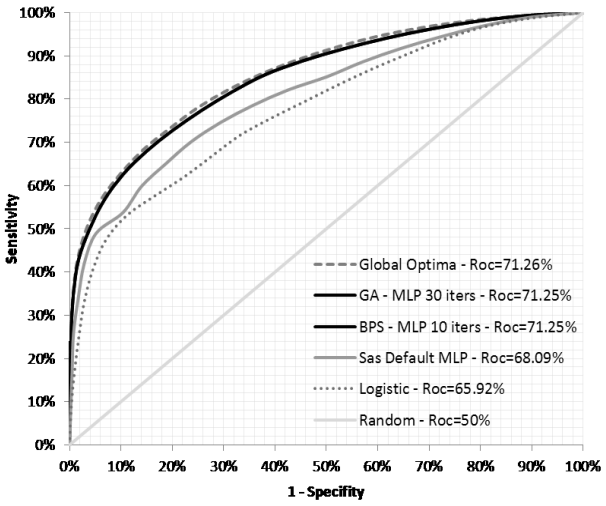

Figure 5. Comparison ROC curve

The comparison of the ROC curves obtained by the GA, and the BPS in the MLP network and the other three alternatives are exhibited in Fig. 5. The area under the ROC curve of the GA and the BPS is (71.25%) wish is significantly larger than that of the MLP Neural Network using the default parameters (68.09%) and the Logistic Regression (65.92%). This difference indicates that the GA and the BPS in the MLP Neural Network has a greater predictive power at all risk levels. The only alternative that slightly exceeds the GA and the BPS performance is the global optimum (71.26%) but the difference is so small that it doesn´t represent a significant improvement in predictive power.

TABLE VII. MEASURES OF COMPARISON

Model	*ROC*	*CPU time (m)*	*Function calls*
Default MLP	68.09%	2	1
Logistic Regression	65.92%	1	1
GA - MLP	71.25%	559	274
BPS - MLP	71.25%	204	100
Global Optimum	71.26%	8,356	4,096

Besides the statistical evidence presented in the ROC curve, TABLE VII 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 function was called. Since the Logistic Regression and the default MLP Neural Network 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 BPS spent 3.4 hours (204 minutes) in the 10 iteration doing a total of 100 function calls. Finally 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 tree 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 while the BPS spent and 3,994.4% less time and function calls than the global optimum finding method and 173.5% less time than the GA.

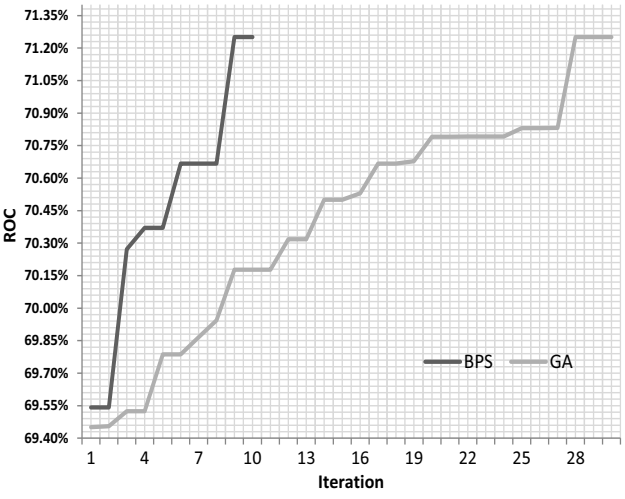

Figure 6. Leaning curve ROC versus Iteration

Additionally, a third comparison was made regarding the evolution of the optimization function (ROC curve) at each iteration. The learning curves of the GA and BPS are shown in Fig. 6. The learning curve of the GA has a slope of 0.0006 while the one of the BPS has slope of 0.0018 which translates into an average increase of 208.65% in the speed of evolution of the resulting networks ROC on each iteration of the BPS over the GA.

Likewise it is important to note that even from the first iteration, both optimization algorithms surpass the predictive power of the final Logistic Regression and the default MLP Neural Network.

B. Real world Impact

Besides the data used to develop the credit card origination models referred in this case of study, the optimization algorithms were used to develop two additional models with completely different datasets that are also currently being use by the bank: *i)* A credit card behavior model and a *ii)* collection model [4]. For each model was calculated a Logistic Regression, a default Neural Network, a BPS and a GA. For both cases the results of the optimization algorithms were the same and therefore only one is displayed.

In order to measure the impact generated in the bank caused by the application of the BPS/GA resulting network over the Logistic Regression model and the default Neural Network model, two comparisons were carried out. The first comparison refers to the difference between predictive power measured by the ROC curve. As TABLE VIII exhibits, for the credit card behavior model the GA/BPS network exceeds the logistic regression predictive power in 1.83% and the default MLP in 1.06%. Likewise, in the collection model the GA/BPS model surpasses the Logistic Regression and the default MLP predictive power in 1.03% and 0.66% respectively.

TABLE VIII. ROC FOR EACH MODEL

Model	*Logistic Regression*	*SAS Default MLP*	*GA/PSO MLP*
Behavior	82.10%	82.72%	83.60%
Collections	88.91%	89.24%	89.83%

For the second comparison, the impact produced by the improvement of the predictive power in the models is measured in savings of money. This comparison is the most important for the senior management given that at the moment of truth is the economic impact that creates value for the companies. The calculations and assumptions applied to measure the economic impact of each of the model are presented in the Appendix. In the case of the credit card behavior model, the annual savings in expected loss obtained by using the GA/BPS final model over the Logistic Regression model were US$528,890, while the saving in expected loss obtained by using the GA/BPS network over the default MLP were US$336,502. Now, for the collection model the savings were calculated as the difference in annual collection fees. The savings achieved by additional predictive power of the GA/BPS models over the Logistic Regression were US$33,490. Equally, the savings of the GA/BPS network over the MLP Neural Network model were US$12,997.

Finally, since the acquisition credit card model developed for the study of this paper is also being used by the bank, the savings calculations were also measure by the annual expected loss. The savings of the GA/BPS over the Logistic Regression and the default MLP were US$225,216 and US$181,152 correspondingly.

Lastly, in order to show that it is necessary to change the architecture of the MLP neural network to best fit each model and therefore is efficient to use the optimization algorithm to do so. TABLE IX displays the different final architectures of the Neural Networks found with the GA/BPS algorithms for every model.

TABLE IX. GA/PSO 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*
Acquisition	2	6	0	1	TAN	SOF	0
Behavior	3	3	0	1	TANH	Logistic	0
Collections	1	5	0	1	TAN	Logistic	1

V. DISCUSSION

During the last years financial institutions started using scores not only for client acquisition but also for others processes inside the bank such us collections, marketing, credit maintenance among others. This due to the success of the scores as cost savings and efficient decision making tools. This tendency creates more pressure on the analysts to develop algorithms faster and with a higher predictive power, leading him to more complex techniques and hence to new problems.

The first problem is that analysts don't have enough time to leave aside the simplicity of the logistic regression and learn a new technique in order to develop more advanced models.

The second problem refers to the way of explaining the models to the senior management. Given that the logistic regression interpretation is very straightforward, the senior management can understand easily the impact of every variable in the model as well as the logic for the business. On the other hand, more complex techniques such as Neural Networks are difficult to explain and understand because the interaction and impact of the variables are unknown and the model turns into a black box. This generates distrust in senior management.

This paper focuses on the first issue, and provides to the analyst a standard procedure to estimate Neural Networks parameters in an efficient way, but there are other issues that

must be address like how to select variables before using then for the Neural Network development and how to simplify the interpretation of complex methodologies in order to present to senior management.

VI. Conclusion

This paper has shown the use of GA and BPS in credit risk modeling as techniques 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. This additional predictive power is reflected in significant savings of money for the bank compared with the two benchmark algorithms (Logistic Regression and default MLP).

Additionally, it is interesting to show that for each model the final Neural Network architecture computed with the GA/BPS varies, demonstrating that is not a good practice to use a single architecture to develop different models.

Also, the experimental results have shown that with far less computational effort the GA and the BPS 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 the optimization algorithms and the global optimum is negligible, we illustrate that it doesn´t represent an improvement in the scorecard predictive power.

It is as well important to say that the GA and the BPS outperformed the results of the Logistic Regression and the results of the default MLP Neural Network.

Finally, although both optimization algorithms presented the same final ROC value, the BPS outperformed the GA in CPU time and function calls.

Appendix

Calculation of the savings for each model using bank´s internal information:

A. Acquisition model

For the acquisition model the savings are defined as the difference of expected loss between models. In order to calculate the expected loss the following information is needed: number of applicants, average credit limit, portfolio severity, and average credit card utilization.

TABLE X. Acquisition model assumptions

Variable	*Value*
Number of applications	68,123
Average credit line	$ 1,500
Severity	85.0%
Average utilization	48.0%

A fixed approval rate of 53.08% was found after determining the statistical cut point in the Logistic Regression model [8]. Then for each model, the bad rate above the statistical cut point is computed.

As shown in [8] the expected loss of one client is defined as follow:

$$EL_i = Prob_i * Balance_i * Severity \quad (1)$$

Since we don't have the balance for each client, it is estimated as the average credit limit multiplied by the average utilization.

$$EL = \#Expected\ Bads * Average\ Utilization * Average\ Credit\ Line * Severity \quad (2)$$

Finally using (2) the expected loss is calculated for each model.

TABLE XI. Acquisition Model Expected Loss Calculation

Model	*Logistic*	*Default MLP*	*GA/PSO MLP*
Approval rate	53.1%	53.1%	53.1%
Bad rate above the cut off	7.12%	6.92%	6.10%
Population above the cut off	36,156	36,156	36,156
Expected bad clients	2,574	2,502	2,206
Expected loss	$ 1,575,288	$ 1,531,224	$ 1,350,072

B. Behavior model

The process for calculating the savings of the behavior model is similar to the acquisition model, in which savings are defined as the differences of the expected loss. Given the different usage of the behavior model versus de acquisition model, the expected loss is calculated as the balance of the bad clients that were above the cut point at the moment of the credit limit increase strategy.

TABLE XII. Behavior model assumptions

Variable	*Value*
Number of Clients	844,177
Average Credit Line	$ 1,500
Average Credit Line Increaced %	58.0%
Average Credit Line Increaced	$ 870
Severity	85.0%
Average Utilization	48.0%

TABLE XIII. Behavior model Expected Loss Calculation

Model	*Logistic*	*Default MLP*	*GA/PSO MLP*
Approval rate	80.2%	80.2%	80.2%
Bad rate above the cut off	4.39%	4.31%	4.17%
Population above the cut off	677030	677030	677030
Expected bad clients	29722	29180	28232
Expected loss	$10,550,121	$10,357,733	$10,021,231

C. Collection model

For the collection model, the savings are going to be calculated as the difference in collection fees. An internal policy states that every client with a default probability higher than 15% must be contacted by the collection team. Therefore the savings are computed as the difference between the number of clients below the cut point multiplied by the cost of one collection action.

TABLE XIV. COLLECTION MODEL ASSUMPTIONS

Variable	*Value*
Average number of clients per month	541,234
Cost per client per month	$ 1.06

TABLE XV. COLLECTION MODEL COSTS CALCULATION

Model	*Logistic*	*Default MLP*	*GA/PSO MLP*
Population below cut point	13.0%	12.7%	12.5%
Clients with collection actions per month	70,415	68,802	67,779
Clients with collection actions per year	844,975	825,620	813,345
Cost per year	$ 894,679	$ 874,186	$ 861,189

REFERENCES

[1] C. Abranhams, M. Zhang, Fair Lending Compliance. John Wiley & Sons, Inc. 2009.

[2] P. D. Allison. Logistic Regression using the SAS system: Theory and Application. Sas Institute and Wiley.

[3] Paasch, Carsten A. W. Credit Card Fraud Detection Using Artificial Neural Networks Tuned By Genetic Algorithms. The Hong Kong University of Science and Technology. 2008.

[4] R. Anderson. The credit scoring toolkit: Theory and practice for retail credit risk management and decision automation. Oxofrd University press Inc, New York. 2007.

[5] R. Haupt, S. Haupt. Practical Genetic Algorithms, second edition. John Wiley & Sons. New York. 2004.

[6] D. Lawrence, A. Solomon. Managing a consumer lending business. Solomon, New York. 2002.

[7] R. Matignon. Neural Network Modeling using SAS Enterprise Miner. Aithor House. 2005.

[8] E. Mays. Credit Scoring for Risk Managers. The Handbook for Lenders. Thomson South-Western. Mason, Ohio. 2004.

[9] F, Rosenblatt. Principles of Neurodynamics. Spartan, Washington, DC. 1962.

[10] SAS Institute Inc. Sas help and documentation, Proc Neural. SAS Institute, Cary, NC. 2010.

[11] L. C. Thomas. Credit Scoring and its applications. Siam, Philadelphia. 2002.

[12] L. C. Thomas. Consumer Credit Models: Pricing, Profit, and Portafolios. Oxford, New York. 2009.

[13] B. Wamer, M. Misra. Understanding Neural Networks as Statistical Tools. The American Statistician Association. 1996.

[14] R. Eberhart, and J. Kennedy, A New Optimizer Using Particles Swarm Theory, Proc. Sixth International Symposium on Micro Machine and Human Science (Nagoya, Japan), IEEE Service Center, Piscataway, NJ, pp. 39-43, 1995.

[15] J. Kennedy, and R. Eberhart, "Particle Swarm Optimization", IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ, IV, pp. 1942-1948, 1995.

[16] J. Kennedy, and R. Eberhart, A discrete binary version of the particle swarm algorithm, IEEE International Conference on Systems, Man, and Cybernetics, 1997.

[17] M. A. Khanesar, M. Teshnehlab, M. A. Shoorehdeli, A novel binary particle swarm optimization, IEEE 15th Mediterranean Conference on Control & Automation, 2009.

[18] Frank H. F. Leung, Member, IEEE, H. K. Lam, S. H. Ling, and Peter K. S. Tam, Tuning of the Structure and Parameters of a Neural Network Using an Improved Genetic Algorithm, IEEE Transactions on neural networks, VOL. 14, NO. 1, January 2003.