Comparison with Parametric Optimization in Credit Card Fraud Detection

Manoel Fernando Alonso Gadi
Grupo Santander
Abbey National plc
Santander Analytics
Milton Keynes, United Kingdom
manoel.gadi@abbey.com

Xidi Wang
Citigroup
Citibank
New York, USA
xidi.wang@citi.com

Alair Pereira do Lago
Depart. de Ciência de Computação
Inst. de Matemática e Estatística
Universidade de São Paulo
05508-090, São Paulo, SP, Brazil
+55 11 3091-6135, alair@ime.usp.br

Abstract

We apply five classification methods, Neural Nets(NN), Bayesian Nets(BN), Naive Bayes(NB), Artificial Immune Systems(AIS) [4] and Decision Trees(DT), to credit card fraud detection. For a fair comparison, we fine adjust the parameters for each method either through exhaustive search, or through Genetic Algorithm(GA) [9]. Furthermore, we compare these classification methods in two training modes: a cost sensitive training mode where different costs for false positives and false negatives are considered in the training phase; and a plain training mode.

The exploration of possible cost-sensitive metaheuristics to be applied is not in the scope of this work and all executions are run using Weka, a publicly available software.

Although NN is claimed to be widely used in the market today, the evaluated implementation of NN in plain training leads to quite poor results. Our experiments are consistent with the early result of Maes in [13] which concludes that BN is better than NN.

Cost sensitive training substantially improves the performance of all classification methods apart from NB and, independently of the training mode, DT and AIS with, optimized parameters, are the best methods in our experiments.

1 Introduction

The lack of publicly available databases has limited the publications on financial fraud detection [15], particularly credit card transactions. In fact, only a few publications on the subject actually have a real contribution based on experiments. For example, the method AdaCost [17, 6], which lead to cost sensitive metaheuristics (see [5] and in [12]), was developed from Adaboost [16] for credit card fraud detection. It can be applied for many applications where there are different costs for false positives and false negatives which is similar to credit card fraud detection. In fact, most machine learning methods are developed in order to minimize the number of classification errors. This leads to biased and unsatisfactory results if there are different costs associated with false positives and false negatives. In the same way, we take into account these different costs in order to not only evaluate the classifier performance more accurately but also to alternatively train the classification model under a cost sensitive procedure. In another publication in the field, comparative studies between Neural Networks (NN) [8] and Bayesian Networks (BN) [2] in credit card fraud detection were reported [13], with better results for BN.

In the same way as NN in the past, many *bio-inspired* algorithms are arising for solving the classification problems. An example of this is in [3], where an Artificial Immune System (AIS) is described. Since AIS is quite unknown, we briefly introduce it in this section. An AIS is a class of *bio-inspired* adaptive or learning algorithms, which includes the Artificial Immune Recognition System (AIRS) [18], a supervised learning that has shown significant success on a broad range of classification problems. One may see it as a rule-based algorithm whose function is to prepare a pool of memory cells (the recognition rules) which are representative of the training data the model is exposed to, and one that is suitable for classifying unseen data. In 2002, the journal Nature [11] published an article on AIS indicating many kinds of applications, including the detection of fraudulent financial transactions. Even though this article previewed a possible commercial application in 2003 by a British company, we are not aware of any subsequent publication on AIS in financial fraud detection which reported good experimental results. The current paper brings back our successful application of AIS on credit card fraud detection previously presented in [7], and reports more complete experiments and comparisons.

Besides AIS, BN, and NN, we compare the methods of Decision Trees (DT) [14] and Naive Bayes (NB), always performing a parametric optimization with the aim of a fair comparison. Furthermore, we develop models under two

training modes in this work: a plain training mode where different costs for false positives and false negatives are not considered; and a cost sensitive training mode where they are taken into account.

Background: Since the beginning of banking, the bank's main job was to protect the money of their client, to safety transfer money from one city to another and to certify/guarantee its customer payment. In this way, bank business has always been related to *fraud prevention*. New technologies such as automated teller machines (ATMs), credit cards and internet banking have amplified the number of transactions banks have to deal with. Analyzing whether each transaction is legitimate or not is very expensive. Confirming whether a transaction was done by a client or a fraudster, for example by phoning all card holders, is cost prohibitive if we check them in all transactions. In this scenario machine learning techniques play a very important role.

Skewed data and other discussions: Fraud detection models are among the most complicated models used in the credit card industry. Skewness of data, search space dimensionality, different cost of false positive and false negative, durability of the model and short time-to-answer are among the problems one has to face in developing a fraud detection model. In this article we focus our attention on dealing with the search space dimensionality by GA and multiresoltion search, and taking into account the different cost of false positive and false negative in the training phase by using the metaheuristics CostSensitiveClassifier[1].

Fraud definition: A transaction is considered *fraudulent* if, in the 2 months following the date of the transaction (which is called performance period), either the client queries the transaction, or the bank distrusts it as a legitimate transaction and confirms it does not belong to the client; otherwise the transaction is tagged as *legitimate*.

Sampling: Our database was obtained from a large Brazilian bank, with registers in the time window between Jul/14/2004 through Sep/12/2004. Each register represents a credit card authorization, including only approved transactions and excluding the denied ones. Before the database reached us, the sampling of transactions was done in two steps: firstly, one randomly samples card numbers to be analyzed in this period, irrespective of whether the card has a fraud transaction in the historical period or not, Secondly, there is a weighted sampling of the class where 10% of legitimate transactions are selected and 100% fraudulent transactions selected. In the end, the database that we have received from the bank contained 41647 registers, of which 3.74% were fraudulent.

[1]The problem of durability and short time-to-answer needs a special investigation which we intend to start to analyze next year.

Categorization: We preprocess the database in two steps.

In step 1, we have 17 independent variables to bind and 1 dependent variable. All variables apart from Merchant Category Code (MCC) are categorized in at most 10 groups[2].

In step 2, we generate 9 splits (also known as samples) from the databases. Each split contains a pair of databases: 70% of transactions for development (training set), and 30% of transaction for validation (testing set, holdout sample). Table 1 shows the average number of registers and the standard deviation of these splits.

data set	average (#): splits 1 to 9	standard deviation(%)
development frauds	1093	1.25%
development legitimates	28017	0.33%
validation frauds	466	2.93%
validation legitimates	12071	0.77%

Table 1: Average fraud/legitimate distribution

The software Weka [19] (version 3-4-11) is used for all of our studies. Implementations used for DT, BN, NB and NN are built in Weka. The only plugged in implementation was the AIS (AIRS2 version 1.6) by Jason Brownlee [1], originally designed and revised by Watkins et al. [18].

Comparison measure: For this work, we found that we would obtain more applicable results using a function independent of the business, but in the same way capable of pricing the business singularities. We also considered a cost function in which we adopted an average cost of $1 for every verification, and an average loss of $100 for every undetected fraud.

If we denote *tp*, *fp* and *fn* as the number of true positives (true frauds), false positive and false negatives, and bearing in mind the received database had only 10% of legitimate and 100% of fraudulent transactions, the cost function is:

$$\$cost = \$100 \times fn + \$10 \times fp + \$1 \times tp. \quad (1)$$

Parameter space: In the implementations used the number of parameters are[3]: NB has no parameter; DT has 2 parameters (C, M); BN has 3 parameters (D, Q, E) and 3 sub parameter (P, S, A); NN has 7 parameters (L, M, N, V, S, E, H); AIS has 9 parameters (S, F, C, H, R, V, A, E, K).

As a reference, the training plus testing time for every method at each in the plain execution was around: NB (2 seconds), DT (70 seconds), BN (12 seconds), NN (800 seconds), AIS (2300 seconds). It almost doubles when we shift to the cost sensitive metaheuristics.

[2]variable name = # of categories: mcc = 33, mcc_previous = 33, zip_code = 10, zip_code_previous = 10, value_trans = 10,value_trans_previous = 10, pos_entry_mode = 10, credit_limit = 10, brand = 6, variant = 6, score = 10, type_person = 2, type_of_trans = 2, # of statements = 4, speed = 8, diff_score = 6, credit_line = 9, flag_fraud = 2.

[3]Please refer to Weka documentations for more details [19, 20].

2 Cost Sensitive

Cost sensitive training is implemented in Weka as a metaheuristic and any classifier can be turned into a cost sensitive one by some kind of appropriated data set preparation. In order to perform such a task, Weka disposes of two methods, *Metacost* and *CostSensitiveClassifier*. Since we were interested in applying GA parameter optimization and the obtained results were almost the same, we adopted the second option because of its speed and also its simplicity. In its default behavior, it makes its base classifier cost-sensitive using one of two methods: either reweighting training instances, if the method allows; or otherwise simply replicating/resampling training instances. (See Figure 1) The right weights and proportions to be used are

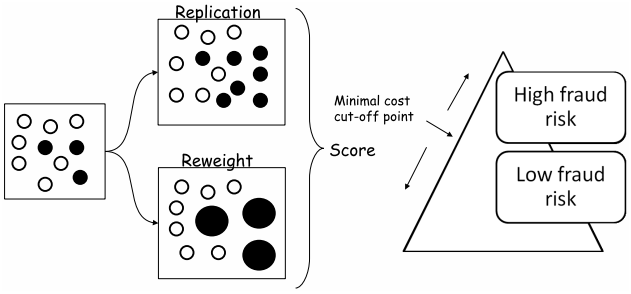

Figure 1: Cost Sensitive meta-heuristics scheme.

critical and depend greatly on the costs associated with false/true positives/negatives. For instance, for the cost function described in equation (1), fraud instances from our dataset should be reweighted 9.9 times, according to equation (2) from [5]. Based on Theorem 1 from [5], this indirectly simulates the minimal cost cut-off point in Figure 1 through reweighting.

3 Optimization of parameters

The parameter spaces of the methods Decision Tree, Bayesian Network and Naive Bayes are small enough to the extent that an exhaustive exploration of the whole parameter space is possible. However, this is not the case for Neural Networks and Artificial Immune Systems. In order to find an optimized parameter set for these methods, we performed a parameter set optimization based on a Genetic Algorithm (GA).

As shown in Figure 2, we start with an initial pool of 50 random executions, followed by 20 Genetic Algorithm (GA) generations. Each GA generation combines two randomly selected candidates among the best 15 from the previous generation. This combination performs: cross over, mutation, random change or no action for each parameter independently. As generations go by, the chance of no action increases. At the end, we perform a local search around the optimized parameter set founded by GA optimization.

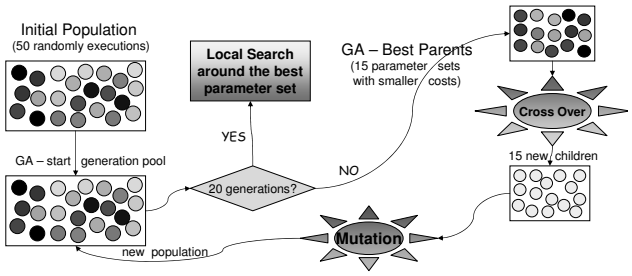

Figure 2: Genetic Algorithm for parameters optimization

Notice that the final solution is usually suboptimal. Given a classification method M, we denote it by $Optimized(M)$.

4 Robustness of the parameters

After the parameter optimization, all optimized parameters may be independent of the split. In this case we say that this parameter set is *robust* and we name it $Robust(M)$, for the classification method M.

When this does not happen, the optimization process is not as strong since the obtained optimized parameter set does not have generalization power. In this case, we decided to sacrifice prediction to gain robustness in the parameter set. In order to rewrite the optimization function that should be used in a GA algorithm, we used a visualization procedure with computed costs for lots of equally spaced parameter sets in the parameter space. After we had defined a good optimization function, we did not proceed with another GA optimization because of time constraints. We did, nonetheless, reuse the runs used in the visualization, with the following kind of *multiresolution optimization* [10]:

- we identify those parameters that have not changed, and we freeze the values for these respective parameters;
- with any other parameter, we screen the best 20 parameter sets for every split and identify a reasonable range;
- for all non-robust parameters, we choose an integer step *s* so the searching space does not explode;
- we evaluate the costs for all possible combinations according to the searching space defined above, and find the parameter set P that brings the minimum average cost among all the different used splits;
- if the parameter set P is on the border of the searching space, we shift this searching space by one step in the direction of this border and repeat the last step until we find this minimum P inside the searching space;
- we zoom the screen to the neighborhood of P, refine steps *s*, and repeat the process from then on, until no refinement is possible.

After this process (refer to Figure 3 for an example), we call this parameters set $Robust(M)$ *robust*. Notice that we could also have used a GA optimization instead.

In order to run the multiresolution optimization, we

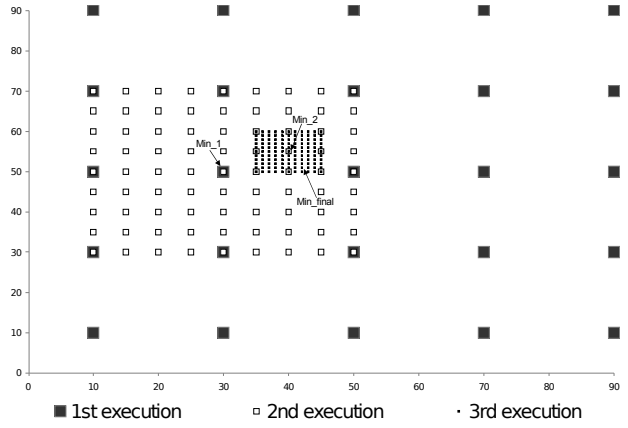

Figure 3: An example of the multiresolution optimization that was applied in order to find robust parameters. In this example we can see two parameters in the searching space, and three steps of this multiresolution optimization. For the parameter represented in horizontal line, the searching space in first step ranges from 10 to 90 with step 20 and the minimum was found for 30. In second step, it ranges from 10 to 50 with step 5 and the minimum was found for 40. In third step, it ranges from 35 to 45, with step 1, which is equivalent to a local exhaustive search in this neighborhood.

elected 6 splits (2,3,4,5,6 and 7) as the *robustization split group*, and 3 others (8,9 and 1) as the *evaluation split group* for posterior evaluation and comparison of all methods.

5 Results

We compare five classification methods: Naive Bayes (NB), Neural Network (NN), Bayesian Network (BN), Artificial Immune System (AIS) and Decision Tree(DT) in two different training modes: plain (cost-blind) and cost-sensitive execution. For any method M, for any of the two training modes, we have applied three different parameter sets, which performs six different strategies for any method M. The first three strategies

$$Default(M), Optimized(M), Robust(M)$$

are executed in plain training mode and use respectively: the default parameters provided by Weka; the optimized set of parameters; and the optimized robust set of parameters. The three last strategies

$$Default{+}{+}(M), Optimized{+}{+}(M), Robust{+}{+}(M)$$

are similar, up to the fact that they are executed in cost-sensitive training mode. Notice that the presence of the symbol $+\!+$ stands for a cost sensitive training mode, in contrast to its absence that stands for plain training mode.

One can see in Figure 4 and Table 2 the final costs[4] of

[4]In relative terms, one should divide all numbers by \$46.2 K, which is the average cost for validation part of splits 8, 9 and 1 for a permissive classifier that classify all transactions as legitimate (frauds are unwatched).

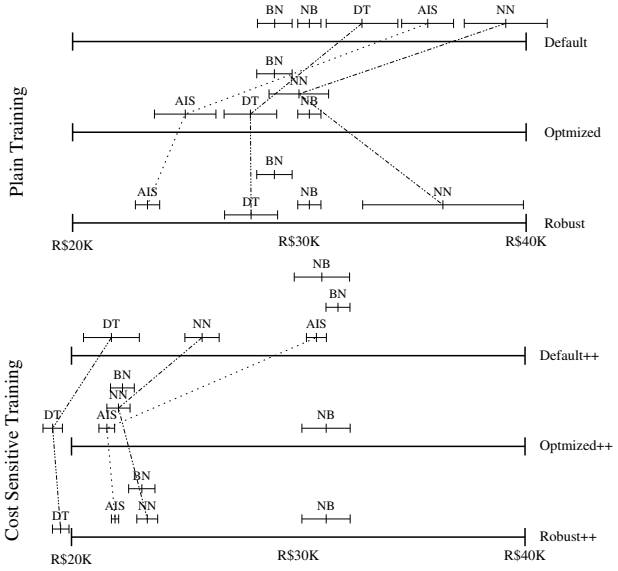

Figure 4: Summary results for the five methods in all six strategies. Average and standard deviation (obtained in three evaluation splits reserved for evaluation of robust parameter sets) are represented by small error-bars. The six horizontal lines with their error-bars (the statistics) are ordered by the development evolution and are stacked in order to compare the six strategies. Some error-bars are vertically shifted for avoiding overlaps.

the classification methods obtained for all strategies. From these results one can notice the analysis that follows.

Method NB does not have any parameter, hence it had no variation in our executions. For this reason, we excluded it from our analysis and therefore the comments bellow are not valid for NB.

Default vs. Optimized vs. Robust set of parameters: As one can expect, either with plain or cost-sensitive training modes, for any method M, it can be observed that the cost decreases from the default set of parameters ($Default(M)$ or $Default{+}{+}(M)$) to the corresponding optimized set of parameters ($Optimized(M)$ or $Optimized{+}{+}(M)$). Moreover, a small increase of cost from this optimized set to the robust set is acceptable ($Robust(M)$ or $Robust{+}{+}(M)$). This is particularly true for the cost-sensitive training mode, where no surprises are seen. With regards to the plain training mode, two unexpected facts can be noted. Firstly, there is an abrupt cost increase for $Robust(NN)$ in relation to $Optimized(NN)$, which shows the over-fitting tendency of method NN with optimized parameters. Secondly, there is a cost reduction for $Robust(AIS)$ in relation to $Optimized(AIS)$. We suppose that this reduction happens due to the fact that AIS has more parameters and also the largest parametric search space. In this way, when the parametric space is reduced, after freezing some of the parameters during the parameters

Strategy	DT	AIS	BN	NN	NB
	32.76	35.66	**28.91**	39.10	30.44
$Default$	(4.83%)	(3.21%)	(2.69%)	(4.68%)	(1.68%)
	27.84	**24.97**	28.91	29.98	30.44
$Optimized$	(4.16%)	(5.43%)	(2.69%)	(4.38%)	(1.68%)
	27.87	**23.30**	28.91	36.33	30.44
$Robust$	(4.21%)	(2.29%)	(2.69%)	(9.75%)	(1.68%)
	21.76	30.80	31.75	25.76	31.04
$Default{+}{+}$	(5.67%)	(1.42%)	(1.67%)	(2.94%)	(3.94%)
	19.17	21.56	22.25	22.07	31.23
$Optimized{+}{+}$	(2.25%)	(1.60%)	(2.37%)	(2.33%)	(3.42%)
	19.52	21.92	23.10	23.34	31.23
$Robust{+}{+}$	(1.88%)	(0.72%)	(2.51%)	(1.98%)	(3.42%)
Percentage of cost reduction					
$Def \rightarrow Rob$	14.9%	**34.7%**	0.0%	7.1%	0.0%
$Def{+}{+} \rightarrow Rob{+}{+}$	10.3%	**28.8%**	27.2%	9.4%	-0.6%
$Def \rightarrow Def{+}{+}$	33.6%	13.6%	-9.8%	**34.1%**	-2.0%
$Opt \rightarrow Opt{+}{+}$	**31.1%**	13.7%	23.0%	26.4%	-2.6%
$Rob \rightarrow Rob{+}{+}$	30.0%	5.9%	20.1%	**35.8%**	-2.6%
$Def \rightarrow Rob{+}{+}$	**40.4%**	**38.5%**	20.1%	**40.3%**	-2.6%

Table 2: Summary results for the five methods in all six strategies. Average costs and standard deviations are obtained in three evaluation splits, reserved for evaluation of the robust parameter sets. Average costs (the smaller, the better) are in thousand of Brazilian currency. Standard deviations are in parenthesis (num%). $A \rightarrow B$ represents the cost reduction $(1 - cost(B)/cost(A)) \times 100\%$.

robustization process, a more efficient optimization can be observed. This phenomenon is offen reffered to as "Curse of Dimensionality". These unexpected facts are not observed for cost-sensitive training. Moreover, the cost variations for suboptimal parameter sets in these training modes are observed to be much smaller than for the plain training mode, which suggests that the cost-sensitive training mode increases the tendency of robustness of optimal and suboptimal parameter sets.

Cost-sensitive vs. plain training modes: When we come to analyse the use of a cost-sensitive training ($+{+}$), all methods (excluding NB) lead to smaller costs if cost sensitive training is used. DT in particular shows the biggest decrease of 33.58%. Moreover, the smallest cost achieved by a plain training method $Robust(AIS)$ is not an improvement on any cost obtained with any method method (except NB) under cost sensitive training mode and robust parameters set ($Robust{+}{+}(DT)$, $Robust{+}{+}(AIS)$, $Robust{+}{+}(BN)$, $Robust{+}{+}(NN)$). Even $Default{+}{+}(DT)$ is better.

Once cost sensitive training has achieved good results in our analysis, we investigated and concluded that the main reason for this good performance was related to the optimal cut-off definition, as result of the cost sensitive classifier formulation [5], but it was also clear that for some algorithms it caused behavior changes, which can introduce shifts of probabilities among two different transactions[5].

[5]It was seeing in AIS when the weight creates an environment where rules to indicate fraud (B-cell) are more like to survive than in an no weighted environment, making the new algorithm to change prediction from legitimate to fraudulent, and specially, not keeping the previous order of probability among different transactions.

Short comments on robust parameters: Before we compare the methods themselves, we report in Table 3 the set of optimized robust parameters obtained for each method. At a first glance, we can observe that for DT we

DEFAULT EXECUTION

Meth	Avg Cost	Robust parameters in command line display
DT	$ 32.76 K	-C 0.25 -M 2
NB	$ 30.44 K	n/a
BN	$ 28.91 K	-D -Q weka.classifiers.bayes.net.search.local.K2 – -P 1 -S BAYES-E weka.classifiers.bayes.net. estimate.SimpleEstimator – -A 0.5
NN	$ 39.10 K	-L 0.30 -M 0.2 -H a -E 20 -V 0 -N 500 -S 0
AIS	$ 35.66 K	-C 10 -R 150 -E 1 -K 3 -F 0.2 -H 2 -V 0.9 -A -1 -S 1

PLAIN TRAINING

Meth	Avg Cost	Robust parameters in command line display
DT	$ 27.87 K	-C 0.49 -M 1
NB	$ 30.44 K	n/a
BN	$ 28.91 K	-D -Q weka.classifiers.bayes.net.search.local.K2 – -P 1 -S BAYES-E weka.classifiers.bayes.net. estimate.SimpleEstimator – -A 0.5
NN	$ 36.33 K	-L 0.40 -M 0.12 -H 20 -E 0 -V 0 -N 500 -S 0
AIS	$ 23.30 K	-C 30 -R 177 -E 5 -K 1 -F 0 -H 10 -V 1 -A -1 -S 1

COST SENSITIVE TRAINING

Meth	Avg Cost	Robust parameters in command line display
DT	$ 19.52 K	-C 0.01 -M 1
NB	$ 31.23 K	n/a
BN	$ 23.10 K	-D -Q weka.classifiers.bayes.net.search.local.K2 – -P 8 -S BAYES-E weka.classifiers.bayes.net. estimate.SimpleEstimator – -A 1.0
NN	$ 23.34 K	-L 0.01 -M 0.08 -H 20 -E 0 -V 0 -N 500 -S 0
AIS	$ 21.87 K	-C 24 -R 77 -E 8 -K 1 -F 0 -H 10 -V 1 -A -1 -S 1

Table 3: Summary of default and robust parameters. Parameters N,S for NN and A,S for AIS were not iterated. Parameters E,V for NN and K,F,H,V for AIS were frozen for the multiresolution optimization. Parameters L,M,H for NN and C,R,E for AIS needed a multiresolution optimization. Parameter H=20 in NN is the number of attributes + number of classes + 1.

have a tree that performs better in our data with minimum pruning (1, default value is 2) according to parameter M, the minimum number of instances per leaf. On the other hand, the confidence parameter C defaults to 0.25, but performs better on the extremes of evaluated intevals: 0.49 for plain training and 0.01 for cost-sensitive training mode. For NN, we see that the parameter L, the learning rate that defaults to 0.3, has decreased dramatically from plain to cost-sensitive training, reaching the allowed minimum (zero does not make sense). The closer to zero, the smaller the impact of the incoming information to be learnt. Parameter V, the percentage size of the validation set from the training to use, performs better for 0, which is also its default value. This shows that no test set is needed in the training phase. In other words, more risk of over-fitting is better for the developed neural nets. This is in accordance with the empirical experience reported in personal discussions with us by a consulting company. In plain training, BN was already

optimal with default parameters. In fact, for plain training, the default parameters performed almost in the same way as with the optimized parameters. However, when cost is taken into account, the parameter P, the maximum number of node parents, influences the final topology and the probability tables, reducing the final costs considerably. For cost sensitive training, where the cut-off thresholds are adjusted to values other than 0.5, the model improves as the number of parent nodes increases, reaching a plateau when P is greater than or equal to 8. Finally, the multiresolution optimization phase was very difficult due to the time required for the cost sensitive training and the insignificant improvement when making small adjustments in parameters. In the end, a very robust result was obtaned as one can see by the AIS cost sensitive standard deviation. One of the most surprising results was that the number of voting rules K ends equals to 1, which means that no voting is necessary: the rule with more affinity decides the class. One can also notice that the number of rules R in AIS decreased from 177 in plain training to 77 in cost sensitive training. What suggests that fewer rules are relevant when different costs for false positives and false negatives are taken into account, whereas, for real applications, it is really an advantage.

Final comparison of all methods and training modes: From Table 2, one can notice that, in changing from default to robust optimized parameter set, AIS is the most cost reducing method, either for plain training (34.7%) or cost-sensitive training (28.8). In contrast, DT and NN obtain the worst cost reductions. On the other hand, changing from plain to cost sensitive training, DT and NN obtain the best cost reductions and AIS the worst. In the end, by changing from default parameters and plain training mode to robust optimized parameters and cost sensitive training mode, the three methods (DT, NN, AIS) are all similarly at effectively reducing their costs.

Notice that the costs and their deviations obtained with a robust set of parameters, shown in Figure 4, suggest that the plain training of DT and BN, and cost sensitive training of BN and NN could have the same costs. Hence, we performed ten Statistical Student's t-tests with 100 new random splits in the same proportion that were specially created for these tests. We tested if

$$\begin{aligned}
Robust{+}{+}(DT) - Robust{+}{+}(AIS) &= 0,\\
Robust{+}{+}(AIS) - Robust{+}{+}(BN) &= 0,\\
Robust{+}{+}(BN) - Robust(AIS) &= 0,\\
Robust{+}{+}(BN) - Robust{+}{+}(NN) &= 0,\\
Robust(AIS) - Robust{+}{+}(NN) &= 0,\\
Robust{+}{+}(NN) - Robust(DT) &= 0,\\
Robust(DT) - Robust(BN) &= 0,\\
Robust(BN) - Robust(NB) &= 0.\\
Robust(NB) - Robust{+}{+}(BN) &= 0.\\
Robust{+}{+}(BN) - Robust(NN) &= 0.
\end{aligned}$$

Not surprisingly, with 99.9% of certainty, almost all $H0$'s were rejected, which means that the great majority of them are not equal. Only the executions with robust parameter sets $Robust{+}{+}(BN)$ and $Robust{+}{+}(NN)$ have equivalent costs, which in turn are also equal to the plain execution with $Robust(AIS)$.

The average costs for an optimized and robust set of parameters defines the rank of methods. From Figure 4, we can notice that DT cost sensitive produced the overall best classifiers, followed by AIS cost sensitive. Then there is a tie among three combinations: BN cost sensitive, AIS plain and NN cost sensitive. Finally, follows DT plain, BN plain, NB plain, NB cost sensitive and NN plain respectively.

6 Future Work

We intend to analyze indeepth the optimized parameters in the coming future, and try to reach better relationship between the values of each parameter, the skewness of the data and cost sensitiveness. We are also considering the inclusion of Support Vector Machines (SVM) in the pool of compared methods. Given we are using AIS, a suitable comparison method would be k nearest neighbor. We also want to include ensembles.

We intend to apply the models for unseen out-of-date datasets to compare stability and durability.

7 Conclusions

In this paper, we present a comparative study of five classification methods (Decision Tree, Neural Network, Bayesian Network, Naive Bayes and Artificial Immune System). In all our executions, except for NB (no parameter needed), we concluded that the best results had not been reached with default sets of parameters as given in Weka. Particularly for AIS and NN, the results achieved using default parameters are very poor if compared with those obtained after a parametric adjustment using GA and our multiresolution optimization that brings robust parameters. We also report optimized and robust parameter sets that can be quite a good start for similar applications.

It is clear that models produced using cost sensitive training, as defined in Section 2, have better performances.

Our tests results show that BN is better than NN, the most used method in real application today, which reproduces the results from Maes [13]. In addition, we found that AIS and DT also surpass both BN and NN. Perhaps DT, because it is a classic classification method, has been forgotten in recent works; however, it still reveals itself as one of the best methods, with sufficiently competitive results.

To sum up, even though some methods show they take more advantage of cost-sensitive training (DT,NN) and others of robust optimization (AIS), applying both cost sensi-

tive training and robust optimization, in conjunction, produce better classifiers. Furthermore, the bests models are obtained with DT, second best models are obtained with AIS. Next, BN and NN, follow equally, and finally, NB.

8 Acknowledgments

This work was partially supported by Projects PRONEX-FAPESP # 03/09925-5 and CNPq # 306217/2004-0 and also by Santander Analytics, the credit risk model development team of Grupo Santander. We would like to acknowledge the banks Santander and Citibank for supporting the development of this research by their employees. We would also like to thank Roberto Cesar Marcondes, Nina Hirata, Casimir Kulikowski, Ilya Muchnik, João Eduardo Ferreira and Yoshiharu Kohayakawa for their comments and support.

References

[1] Jason Brownlee. Artificial immune recognition system v.2 (airs2) - a review and analysis. Technical report, Victoria, Australia: Centre for Intelligent Systems and Complex Processes (CISCP), Faculty of Information and Communication Technologies (ICT), Swinburne University of Technology, Jan 2005.

[2] Eugene Charniak. Bayesians networks without tears. *AI Magazine*, 1991.

[3] Dipankar DasGupta. *Artficial Immune Systems and Their Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[4] Leandro Nunes de Castro and J. Timmis. *Artificial Immune Systems: A Novel Paradigm to Pattern Recognition*. University of Paisley, 2002.

[5] Charles Elkan. The foundations of cost-sensitive learning. In *IJCAI*, pages 973–978, 2001.

[6] Wei Fan, Salvatore J. Stolfo, Junxin Zhang, and Philip K. Chan. AdaCost: misclassification cost-sensitive boosting. In *Proc. 16th International Conf. on Machine Learning*, pages 97–105. Morgan Kaufmann, San Francisco, CA, 1999.

[7] Manoel Fernando Alonso Gadi, Xidi Wang, and Alair Pereira do Lago. Credit card fraud detection with artificial immune system, 7th international conference, icaris 2008, phuket, thailand, august 10-13, 2008, proceedings. In *ICARIS*, volume 5132 of *Lecture Notes in Computer Science*, pages 119 – 131. Springer, 2008.

[8] Simon Haykin. *Neural Networks: A Comprehensive Foundation, 2/E*. Prentice Hall, Ontario, Canada, 1999.

[9] John Holland. *Adaptation in Natural and Artificial Systems*. 1st MIT press, 1975.

[10] Jinwoo Kim and Bernard P. Zeigler. A framework for multiresolution optimization in a parallel/distributed environment: simulation of hierarchical gas. *J. Parallel Distrib. Comput.*, 32(1):90–102, 1996.

[11] Erica Klarreich. Inspired by immunity. *Nature*, (415):468–470, January 2002. doi:10.1038/415468a.

[12] Jin Li, Xiaoli Li, and Xin Yao. Cost-sensitive classification with genetic programming. In *Congress on Evolutionary Computation*, pages 2114–2121, 2005.

[13] Sam Maes, Karl Tuyls, Bram Vanschoenwinkel, and Bernard Manderick. Credit card fraud detection using bayesian and neural networks. In *Proceedings of NF2002, 16-19 January*, Havana, Cuba, 2002.

[14] Sreeram K. Murthy, Enno Ohlebusch, and Stefan Kurtz. Automatic construction of decision trees from data: A multi-disciplinary survey. In *Data Mining and Knowledge Discovery*, volume 2, pages 345 – 389, USA, 1998. Kluwer Academic Publishers.

[15] Clifton Phua, Vincent Lee, Kate Smith, and Ross Gayler. A comprehensive survey of data mining-based fraud detection research. In *Artificial Intelligence Review (submitted for publication)*, 2005.

[16] Robert E. Schapire and Yoram Singer. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[17] S. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. Chan. Credit card fraud detection using meta-learning: Issues and initial results, 1997. Working notes of AAAI Workshop on AI Approaches to Fraud Detection and Risk Management.

[18] Andrew Watkins, Jon Timmis, and Lois Boggess. Artificial immune recognition system (AIRS): An immune-inspired supervised machine learning algorithm. *Genetic Programming and Evolvable Machines*, 5(3):291–317, September 2004.

[19] Ian H. Witten and Eibe Franku. *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*. Elsevier, 2005.

[20] Ian H. Witten and Eibe Franku. Software documentation: Weka, 2008. file `/Weka-3-4-11/doc/weka/classifiers/functions/MultilayerPerceptron.html` from Weka package.