Consolidated Tree Classifier Learning in a Car Insurance Fraud Detection Domain with Class Imbalance

Jesús M. Pérez, Javier Muguerza, Olatz Arbelaitz, Ibai Gurrutxaga, and José I. Martín

Dept. of Computer Architecture and Technology, University of the Basque Country,
M. Lardizabal, 1, 20018 Donostia, Spain
{txus.perez, j.muguerza, olatz.arbelaitz,
ibai.gurrutxaga, j.martin}@ehu.es
http://www.sc.ehu.es/aldapa

Abstract. This paper presents an analysis of the behaviour of Consolidated Trees, CT (classification trees induced from multiple subsamples but without loss of explaining capacity). We analyse how CT trees behave when used to solve a fraud detection problem in a car insurance company. This domain has two important characteristics: the explanation given to the classification made is critical to help investigating the received reports or claims, and besides, this is a typical example of class imbalance problem due to its skewed class distribution. In the results presented in the paper CT and C4.5 trees have been compared, from the accuracy and structural stability (explaining capacity) point of view and, for both algorithms, the best class distribution has been searched.. Due to the different associated costs of different error types (costs of investigating suspicious reports, etc.) a wider analysis of the error has also been done: precision/recall, ROC curve, etc.

1 Introduction

The application of machine learning to real world problems has to be done considering two important aspects: class distribution affects to classifiers' accuracy and the explanation is very important in some domains.

In real domains such as illness diagnosis, fraud detection in different fields, customer's behaviour analysis (marketing), customer fidelisation, ... it is not enough to obtain high accuracy in the classification, comprehensibility in the built classifier is also needed [2]. The classifying paradigms used to solve this kind of problems need to be able to give an explanation, for example classification trees.

On the other hand, it is very common to find domains where the number of examples for one of the categories (or classes) of the dependent variable is much smaller than for the rest of the classes. These situations are named class imbalance or skewed class distribution.

Classifiers do not behave well when they are trained with very unbalanced data sets. For example, if 99% of the examples in a data set belong to the same class, for a classifier that labels test cases with the majority class, the accuracy will be 99%. Since most classifiers are designed to minimise the error rate, the classifiers built from this kind of data-sets tend to be very simple and nonsense [3].

Weiss and Provost [8] have shown that each domain has an optimal class distribution to be used for training. Their work shows that, in situations where the class distribution of the training set can be chosen, it is often preferable to use a different distribution to the one expected in reality. So, in environments with skewed class distribution, we can use samples with modified class distribution with the aim of building valid classifiers. Undersampling, eliminating some examples, is the most common strategy to modify the class distribution of a sample. Even if the direct consequence of undersampling is that some examples are ignored, in general, undersampling techniques obtain better results than oversampling techniques (repeating some examples) [3]. In order to avoid the information loss produced by undersampling, multiple classifiers can be built based on subsamples with changed distribution. Most of the information in the original sample can be covered by choosing adequately the number of generated subsamples. Techniques such as bagging and boosting can be a good option in some cases but not in areas where explanation is important. It is clear that "while a single decision tree can easily be understood by a human as long as it is not too large, fifty such trees, even if individually simple, exceed the capacity of even the most patient" [2].

We have developed an algorithm, CTC (Consolidated Tree's Construction Algorithm), that is able to face both problems: several subsamples with the desired class distribution are created from the original training set, but opposite to other algorithms that build multiple trees (bagging, boosting), a single tree is induced, therefore the comprehensibility of the base classifier is not lost.

Fraud detection problems belong to the group of domains where the explanation in the classification is important. We will use the CTC algorithm and compare it to C4.5 [6] in a fraudulent report detection problem from a car insurance company. This is a difficult problem because the experts in the company estimate that the fraud average is in reality higher than 10% or 15% but the fraud examples are very difficult to detect, and, as a consequence, in the data provided by insurance companies this percentage is lower. Many of the examples labelled as not fraudulent in the data-set are actually fraudulent which makes the problem even harder. It is important to provide an insurance company with a tool to know the profile of fraudulent customers or the evidences that make a report suspicious of fraud, in order to investigate them more deeply, so that the fraud does not suppose great financial loss.

The paper proceeds describing how a single tree can be built from several subsamples, CTC algorithm, in Section 2. In Section 3 we describe the main characteristics of the car insurance company's database for fraud detection, and Section 4 contains the methodology used to face the class imbalance problem in the described domain. Section 5 is devoted to describe the results obtained for the fraud detection problem with both algorithms CTC and C4.5. Finally Section 6 is devoted to show the conclusions and further work.

2 Consolidated Trees' Construction Algorithm

Consolidated Trees' Construction Algorithm (CTC) uses several subsamples to build a single tree [4]. The consensus is achieved at each step of the tree's building process and only one tree is built. The different subsamples are used to make proposals about the feature that should be used to split in the current node. The split function used in each subsample is the gain ratio criterion (the same used by Quinlan in C4.5). The decision about which feature will be used to make the split in a node of the Consolidated Tree (CT) is accorded among the different proposals by a voting process (not weighted) node by node. Based on this decision, all the subsamples are divided using the same feature. The iterative process is described in Algorithm 1.

Algorithm 1. Consolidated Trees' Construction Algorithm (CTC)

```
Generate $Number\_Samples$ subsamples ($S^i$) from $S$ with $Resampling\_Mode$ method.
CurrentNode := RootNode
for  $i$ := 1 to $Number\_Samples$
   $LS^i := \{S^i\}$
end for
repeat
   for  $i$ := 1 to $Number\_Samples$
       $CurrentS^i := First(LS^i)$
       $LS^i := LS^i - CurrentS^i$
       Induce the best split $(X,B)^i$ for $CurrentS^i$
   end for
   Obtain  the consolidated pair $(X_c,B_c)$, based on $(X,B)^i$, $1 \le i \le Number\_Samples$
   if $(X_c,B_c) \ne Not\_Split$
      Split CurrentNode based on $(X_c,B_c)$
      for $i$ := 1 to $Number\_Samples$
         Divide $CurrentS^i$ based on $(X_c,B_c)$ to obtain $n$ subsamples $\{S_1^i, \dots S_n^i\}$
         $LS^i := \{S_1^i, \dots S_n^i\} \cup LS^i$
      end for
   else consolidate CurrentNode as a leaf
   end if
CurrentNode := NextNode
until $\forall i$, $LS^i$ is empty
```

The algorithm starts extracting a set of subsamples (*Number_Samples*) from the original training set. Based on previous experimentation we find that to use 30 subsamples can be a good trade-off among efficiency and computational cost . The subsamples are obtained based on the desired resampling technique (*Resampling_Mode*). For example, the class distribution of the original training set can be changed or not, examples can be drawn with or without replacement, different subsample sizes can be chosen, etc.

Decision tree's construction algorithms divide the initial sample in several data partitions. In our algorithm, LS^i contains all the data partitions created from each subsample S^i. When the process starts, the only existing partitions are the initial subsamples.

The pair $(X,B)^i$ is the split proposal for the first data partition in LS^i. X is the feature selected to split and B indicates the proposed branches or criteria to divide the data in the current node. In the consolidation step, X_c is the feature obtained by a voting process among all the proposed X. Whereas B_c will be the median of the proposed *Cut* values when X_c is continuous and all the possible values of the feature when X_c is discrete. In the different steps of the algorithm, the default parameters of C4.5 have been used as far as possible.

The process is repeated while LS^i is not empty. The Consolidated Tree's generation process finishes when in the last subsample in all the partitions in LS^i, most of the proposals are not to split it, so, to become a leaf node. When a node is consolidated as a leaf node, the a posteriori probabilities associated to it are calculated averaging the a posteriori obtained from the data partitions related to that node in all the subsamples.

Once the consolidated tree has been built, it works the same way a decision tree does for testing, pruning, etc. This way the explanation of the classifier is not lost even if several subsamples are used to build it.

3 Car Insurance Database for Fraud Detection

One of the factors affecting the price of car insurance policies is the large amount of fraudulent reports that a company is not able to detect is. The company has to assume all the increase in costs produced by this fraud, and, as a consequence, the insurance policies become more expensive. The experts in the companies think that at least 10% or 15% of the produced reports are fraudulent, and, however, about 5% of them is detected. So the databases in insurance companies have the following characteristics: the examples labelled as fraudulent belong to the minority class (class imbalance) and, on the other hand, they are the only 100% reliable data, because among the examples labelled as not fraudulent there are some fraudulent examples that the company has not been able to detect. Therefore the information provided to the algorithm is not correct which makes the machine learning problem difficult to solve.

In order to detect fraud, suspicious reports have to be investigated but this has associated costs: the costs concerning to the investigation itself (staff, resources, etc.), and the cost coming from investigating not fraudulent customers [1]. The company's image can be severely affected by customers that are annoyed when they realise that they are being investigated. When evaluating a report, it will be important for the insurance broker to know the fraud probability assigned to it by the classification system, as well as the factors that have affected to the decision. The explanation given by the classifier about the decision made could be used by the broker to investigate the case. So, if the aim is to have a tool that will help in the detection of fraudulent reports, it is absolutely necessary to use classification paradigms that are able to give an explanation, for example, decision trees. This paper analyses the behaviour of different classifiers with real data from a car insurance company, the kind of problem we have just described. The data-set has 108,000 examples, and just 7.40% of them are fraudulent cases. This database is clearly an example of class imbalance problem with imprecise information for one of the classes.

The database has 31 independent variables that contribute to the report with information of different nature about the accidents: date of the accident (when it happened and when it was communicated), insured person (age, sex, marital status,...), insurance policy and vehicle (fully comprehensive insurance or not, driving experience, kind and use of the vehicle, power,…). When solving this problem with supervised classification, the dependent variable or class will have two categories: fraud and not fraud.

4 Experimental Methodology

The original class distribution of the collected data does not always coincide with the best one to build the classifier when a problem with class imbalance has to be faced. We will make a sweep with different percentages of fraud examples in order to find the class distribution we should use to induce the tree. Based on the methodology proposed by Weiss and Provost in [8] the tried percentages will be 2%, 5%, 7.40% (original distribution), 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 95%. We will use the methodology described in [8] to compare the CTC algorithm (with parameters mentioned in Section 2) with the well known C4.5 with default settings for the different class distributions. In order to make a fair comparison, the used training sets need to have the same size even if the used percentages are changed. The size of the training set is fixed to 75% of the minority class examples (6,000 examples), so that subsamples of all the mentioned percentages can be used in the experimentation. The remaining 25% of both, majority and minority classes, will be used for test.

Two trees, one with C4.5 and the other with CTC, have been built for each one of the proposed percentages. Even if Weiss and Provost did not prune the trees, the results obtained with pruned trees in this database are substantially better for both algorithms. Therefore, we will present the results obtained by pruning the trees based on the training sample and C4.5 standard pruning. To prune the C4.5 trees we have used the corrector proposed by Weiss and Provost for estimating the a posteriori probability of the leaf nodes, so that they are adapted to the distribution expected in reality. This has to be done because the class distribution of the training set and the class distribution existing in reality (test) do not coincide. Nevertheless, the corrector needs not to be used when pruning CT trees, because the pruning is done with the whole training set (the percentages are the ones expected in reality), and the a posteriori probabilities are corrected due to the backfitting process.

As a validation method, the experimentation has been repeated 10 times.

5 Experimental Results

In the problem described in previous sections, the behaviour of classifiers can not be analysed based just on the error rate. Other aspects will help us to complete our comprehension about the classifier's behaviour: the structural stability of the generated trees and the complexity of the trees will give us information about the quality of the explanation, the ratio among True Positive and False Positive examples (ROC curve) will give us information about the behaviour of the classifier in different environments, etc.

Table 1 shows, for CT trees built with 30 subsamples and C4.5, the error rates and standard deviation (*Error* and σ columns) and the average complexity, measured as the number of internal nodes of the trees (*Compl.* column). The values in different

Table 1. Error, standard deviation and complexity for different class distributions

	C4.5			CT(30)		
	Error	σ	*Compl.*	*Error*	σ	*Compl.*
2%	7.40	0.00	0.0	7.30	0.00	3.8
5%	7.40	0.00	0.0	7.31	0.03	4.8
7.4%	7.40	0.00	0.0	7.31	0.03	4.3
10%	7.40	0.00	0.0	7.31	0.03	4.1
20%	7.57	0.34	12.3	7.30	0.05	5.6
30%	9.75	0.68	103.6	7.30	0.05	6.6
40%	11.70	1.06	183.7	7.31	0.03	6.2
50%	10.87	0.95	141.4	7.29	0.06	7.1
60%	8.69	0.71	44.6	7.30	0.05	6.9
70%	8.59	0.60	28.2	7.30	0.05	7.2
80%	8.27	0.52	17.2	7.29	0.06	6.6
90%	8.59	0.70	11.2	7.29	0.06	6.0
95%	7.40	0.00	0.0	7.31	0.07	3.4

rows are related to different class distributions. Results show that in every case the error is smaller for CTC than for C4.5. The trees built with C4.5 are not able to reduce the error rate of 7.40% that would achieve a trivial classifier that labels all the examples with the majority class (no fraud). The values belonging to average complexity confirm that when the error for C4.5 trees is 7.40% the built trees are trivial classifiers: they are just the root node. For the rest of the percentages, the built classifiers do not make sense because of the achieved error rate and complexity. The values of standard deviation show that the error rates achieved with CT trees are very stable (best values are obtained when class distributions are 50%, 80% and 90%). Besides, all of them are under the threshold of 7.40% (7.30% in average) and with small complexity in average; therefore, giving a simple explanation. These results confirm that CT trees are better situated in the learning curve and also according to the principle of parsimony (Occam's razor).

If we want to evaluate the stability of the explanation given by CT trees we need to measure the structural stability. A structural distance, *Common*, based on a pair to pair comparison among all the trees of the compared set has been defined with this aim. *Common* is calculated starting from the root and covering the tree, level by level. The common nodes among two trees are counted if they coincide in the feature used to make the split, the proposed branches or stratification and the position in the tree [5]. Normalising the *Common* value with the complexity of the tree (as defined before) and making the analysis for the CT trees built for different class distributions, we find a maximum value of 74.27% and minimum of 35.19% being the average 49.36%. So we can say that in average half of the structure of the trees, and, as a consequence the explanation, is maintained. However for C4.5 the *%Common* is in average 10.31%.

The division of the error in false positive (FP) and false negative (FN) is important in this kind of applications. FP quantifies the amount of unnecessary investigations of customers whereas the FN quantifies the fraudulent customers that are not detected. Evidently it is of capital importance not to investigate honest customers in order to achieve a good company image. Even if the objective is to detect all the fraudulent reports, the quantification of the percentage of investigated reports is also important due to the costs and the trouble to the customers it implies. We will analyse these aspects with results of *precision*, *recall* or *sensitivity*, *breakeven point*, ROC curve and AUC [7]. To be brief, we will present results for class distribution of 50% (results for other distributions are similar).

Classification trees can work on a more or less conservative way by modifying the threshold needed to label a node as fraudulent. Table 2 shows results for a wide range of thresholds. When the threshold is 0% (trivial acceptor) all the examples will be classified as fraudulent; this would be the most liberal operation mode. On the other hand, the most restrictive operation mode will be when the threshold is 100%. Only the fraudulent examples belonging to homogeneous nodes would be classified as fraudulent. The adequate threshold is usually selected so that the best trade off among costs of FP and FN is found.

Table 2. Results for conservative and liberal operation mode for CTC and C4.5 algorithms

	CTC			*C4.5*		
Threshold	*Precision*	*Recall*	*Reports*	*Precision*	*Recall*	*Reports.*
0%	7.41	100.00	27000	7.41	100.00	27000
5%	11.21	89.66	15995	12.20	76.39	12523
10%	12.17	66.53	10932	12.60	70.17	11135
15%	**53.98**	**5.12**	**190**	13.50	41.28	6115
20%	55.03	5.09	185	14.11	22.87	3241
25%	57.84	4.89	169	14.52	16.80	2314
30%	59.92	4.71	157	14.95	14.32	1916
35%	60.31	4.68	155	15.61	13.33	1708
40%	60.64	4.62	152	15.78	11.82	1498
45%	63.23	3.59	113	14.73	10.13	1375
50%	63.39	3.46	109	14.85	9.91	1335
55%	63.51	3.42	108	13.88	8.92	1285
60%	66.43	2.80	84	13.31	8.29	1245
65%	67.05	2.62	78	12.13	7.28	1199
70%	70.11	2.24	64	10.69	6.14	1148
75%	**70.39**	**1.61**	**46**	9.82	5.48	1116
80%	50.00	0.09	3	9.83	5.38	1094
85%	50.00	0.09	3	9.83	5.38	1094
90%	50.00	0.09	3	9.83	5.22	1062
95%	50.00	0.09	3	9.83	5.22	1062
100%	--	0.00	0	--	0.00	0

The *precision* and *recall* are two parameters that can be used to measure the effectiveness of the classifier on basis of the threshold. Examples of conservative and liberal operation mode appear in bold in Table 2. We can observe that if the classifier based on CTC would work in a conservative way (*threshold* 75%), the company would revise only 46 reports (*reports*) and 70.39% of them (*precision*) would be fraudulent (the probability to find a fraudulent report has been increased from 7.4% to 70.39% and the disturbed customers have been very few). If we would like to detect more fraudulent reports, increasing the *recall* but still without disturbing a lot of not fraudulent customers we could decrease the threshold to 15% (liberal example). As a consequence 190 reports would be revised and more than half of them would be fraudulent. Table 2 shows that the trees induced with C4.5 achieve higher *recall* values, but the amount of reports to investigate and the low precision obtained make grow considerably the costs related to investigations and incorrectly revised customers.

A way to find a balance among *precision* and *recall* is to establish a threshold, *breakeven point*, so that both parameters are made equal. Although this is a too general measure for the purposes of our study, CTC clearly beats C4.5; the estimated values are 34.45% for CTC and 15.25% for C4.5. Evidently being the aim the maximisation of *precision* and *recall* when larger the *breakeven point* is, better the behaviour of the algorithm is.

We have also calculated the ROC curves (they are not shown due to lack of space) of both classifiers. The aim is to maximize the TP with a minimum FP, and as a consequence, to maximize the Area Under the ROC Curve (AUC). We have calculated the average AUC for all the analyzed class distributions and the average values obtained are: 68.87% for CTC and 60.71% for C4.5. This indicates that CT trees have better global behaviour than C4.5 trees.

6 Conclusions and Further Work

This paper presents the analysis of the influence of class distribution in a fraud detection problem from a car insurance company for two tree induction algorithms: C4.5 and CTC. The behaviour of both algorithms for different class distributions has been analysed based on the methodology presented in [8]. Thanks to this methodology we have been able to build non trivial C4.5 trees, but results have been better for CT trees. Moreover, both algorithms build a single tree, that is to say, they maintain the explanation in the classification which is essential in real problems of this kind where an explanation added to the classification made is compulsory. The results presented in Section 5 confirm that CT trees behave better than C4.5 trees in many aspects: accuracy, structural stability or explanation, ROC curve, *precision*/*recall*, etc.

The results obtained in this experimentation could be compared to other strategies that do not lose the explanation even if they use several subsamples to build the classifier. For example the procedure presented in [2], which is able to extract explanation to bagging, and our proposal could be compared. As we mentioned when describing CTC algorithm many parameters can be varied. CTC algorithm can be tested using other base algorithm different to C4.5 such as CHAID, CART, …

Related to the real application, fraud detection in car insurance companies, the difficulty finding fraud examples make us think that the characteristics provided by the experts in the company might not be suitable, so the extraction of more discriminating information could be studied.

Acknowledgements

The work described in this paper was partly done under the University of Basque Country (UPV/EHU), project: 1/UPV 00139.226-T-15920/2004. It was also funded by the Diputación Foral de Gipuzkoa and the European Union. We also want to thank the collaboration of the company ADHOC Synectic Systems, S.A. The *lymphography* domain was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. Thanks go to M. Zwitter and M. Soklic for providing the data.

References

1. Chan P.K., Stolfo S.J.: Toward Scalable Learning with Non-uniform Class and Cost Distributions: A Case Study in Credit Card Fraud Detection, Proc. of the 4th Int. Conference on Knowledge Discovery and Data Mining, (1998) 164-168.
2. Domingos P.: Knowledge acquisition from examples via multiple models. Proc. 14th International Conference on Machine Learning Nashville, TN (1997) 98-106.
3. Japkowicz N.: Learning from Imbalanced Data Sets: A Comparison of Various Strategies, Proceedings of the AAAI Workshop on Learning from Imbalanced Data Sets, Menlo Park, CA, (2000).
4. Pérez J.M., Muguerza J., Arbelaitz O., Gurrutxaga I.: A New Algorithm to Build Consolidated Trees: Study of the Error Rate and Steadiness. Advances in Soft Computing, Proceedings of the International Intelligent Information Processing and Web Mining Conference (IIS: IIPWM´04), Zakopane, Poland (2004), 79-88.
5. Pérez J.M., Muguerza J., Arbelaitz O., Gurrutxaga I., Martín J.I.: Analysis of structural convergence of Consolidated Trees when resampling is required. Proc. of the 3rd Australasian Data Mining Conf. (AusDM04), Australia (2004), 9-21.
6. Quinlan J.R.: C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers Inc.(eds), San Mateo, California, (1993).
7. Sebastiani F.: Machine Learning in Automated Document Categorisation. Tutorial of the 18th Int. Conference on Computational Linguistics, Nancy, Francia, 2000.
8. Weiss G.M., Provost F.: Learning when Training Data are Costly: The Effect of Class Distribution on Tree Induction, Journal of Artificial Intelligence Research, Vol. 19, (2003) 315-354.