Evolutionary Training Set Selection to Optimize C4.5 in Imbalanced Problems

Salvador García, Francisco Herrera
University of Granada
Department of Computer Science and Artificial Intelligence
18071 Granada, Spain
{salvagl,herrera}@decsai.ugr.es

Abstract

Classification in imbalanced domains is a recent challenge in machine learning. We refer to imbalanced classification when data presents many examples from one class and few from the other class, and the less representative class is the one which has more interest. One of the most used techniques to tackle this problem consists in preprocessing the data previously to the learning process. This preprocessing could be done through under-sampling; removing examples, mainly belonging to the majority class; and over-sampling, by means of replicating or generating new minority examples. This contribution proposes an under-sampling procedure based on evolutionary algorithms to perform a training set selection for optimizing the models obtained by the C4.5 decision tree. The proposal has been compared with other under-sampling and over-sampling techniques and the results are very competitive in terms of accuracy, and the obtained models are more interpretable.

1. Introduction

In the last years, the class imbalance problem is one of the emergent challenges in data mining [24]. The problem appears when the data presents a class imbalance, which consists in containing many more examples of one class than the other one and the less representative class is the most interesting [8]. Imbalance in class distribution is pervasive in a variety of real-world applications, including but not limited to telecommunications, WWW, finance, biology and medicine.

Usually, the instances are grouped into two type of classes: the majority or negative class, and the minority or positive class. The minority or positive class is often of interest and also accompanied with a higher cost of making errors. A standard classifier might ignore the importance of the minority class because its representation inside the data set is not strong enough. As a classical example, if the ratio of imbalance presented in the data is 1:100 (that is, there is one positive instance versus one hundred negatives), the error of ignoring this class is only 1%.

Many approaches have been proposed to deal with the class imbalance problem. They can be divided into algorithmic approaches and data approaches. The first ones assume modifications in the operation of the algorithms, making them cost-sensitive towards the minority class [18, 15]. The data approaches modify the data distribution, conditioned on an evaluation function. Re-sampling of data could be done by means of under-sampling, by removing instances from the data, and over-sampling, by replicating or generating new minority examples. There have been numerous papers and case studies exemplifying their advantages [3, 6, 22, 11, 7].

Evolutionary Algorithms (EAs) have been used for data reduction with promising results. They have been successfully used for feature selection [23, 14, 21] and instance selection [4, 12]. They have also been applied for under-sampling the data in imbalanced domains in instance-based learning [13]. EAs also have a good behaviour for Training Set Selection (TSS) in terms of getting a trade-off between precision and interpretability with classification rules [5].

In this contribution, we propose the use of EAs for TSS in imbalanced data sets. Our objective is to increase the accuracy of the well-known C4.5 decision tree [19] classifier by means of removing instances mainly belonging to the majority class. We compare our approach with other under-sampling, over-sampling methods and hybridization proposals of over-sampling and under-sampling [3] studied in the literature. The empirical study has been contrasted via non-parametrical statistical testing.

To achieve this objective, the rest of the contribution is organized as follows: Section 2 gives an explanation about the measure used for evaluating imbalanced classification. In Section 3, the evolutionary TSS issues are explained, together with a description of the used model. In Section 4 the experimentation framework and the results and their analy-

978-0-7695-3326-1/08 $25.00 © 2008 IEEE
DOI 10.1109/HIS.2008.67

sis are presented. Finally, in Section 5, we point out our conclusion.

2. Evaluation Measure Used for Evaluating Imbalanced Classification

When we want to evaluate a classifier over imbalanced domains, classical ways of evaluating, such as classification accuracy, have no sense. A standard classifier that uses accuracy rate may be biased towards the majority class due to the bias inherent in the measure, which is directly related to the ratio between the number of instances of each class.

The most correct way of evaluating the performance of classifiers is based on the analysis of the confusion matrix. In Table 1, a confusion matrix is illustrated for a problem of two classes, with the values for the positive and negative classes. From this matrix it is possible to extract a number of widely used metrics to measure the performance of learning systems, such as *Error Rate*, defined as $Err = \frac{FP+FN}{TP+FN+FP+TN}$ and *Accuracy*, defined as $Acc = \frac{TP+TN}{TP+FN+FP+TN} = 1 - Err$.

	Positive Prediction	*Negative Prediction*
Positive Class	True Positive (TP)	False Negative (FN)
Negative Class	False Positive (FP)	True Negative (TN)

Table 1. Confusion matrix for two-class problem.

In relation to the use of error (or accuracy) rate, another type of metric in the domain of the imbalanced problems is considered more correct. Concretely, from Table 1 it is possible to obtain four metrics of performance that measure the classification performance for the positive and negative classes independently:

- **False negative rate** $FN_{rate} = \frac{FN}{TP+FN}$ is the percentage of true positive cases misclassified as negative.
- **False positive rate** $FP_{rate} = \frac{FN}{FP+TN}$ is the percentage of true negative cases misclassified as positive.
- **True negative rate** $TN_{rate} = \frac{TN}{FP+TN}$ is the percentage of true negative cases correctly classified as negative.
- **True positive rate** $TP_{rate} = \frac{TP}{TP+FN}$ is the percentage of true positive cases correctly classified as positive.

The goal of a classifier is to minimize the false positive and false negative rates or, in a similar way, to maximize the true positive and true negative rates.

In [2] it was indicated a metric called *Geometric Mean (GM)*, defined as $g = \sqrt{a^+ \cdot a^-}$, where a^+ denotes accuracy in positive examples (TP_{rate}), and a^- is accuracy on negative examples (TN_{rate}). This measure tries to maximize accuracy in order to balance both classes at the same time. It is an evaluation measure that allows to simultaneously maximize the accuracy in positive and negative examples with a good trade-off. We focus our study in the GM metric.

3. Evolutionary Training Set Selection in Imbalanced Classification

Let us assume that there is a training set TR with N instances which consists of pairs $(x_i, y_i), i = 1, ..., N$, where x_i defines an input vector of attributes and y_i defines the corresponding class label. Each of the N instances has M input attributes and they should belong to positive or negative class. Let $S \subseteq TR$ be the subset of selected instances resulted in the execution of an algorithm.

TSS can be considered as a search problem in which EAs can be applied. Our approach will be denoted by Evolutionary Under-Sampling for Training Set Selection (EUSTSS). We take into account two important issues: the specification of the representation of the solutions and the definition of the fitness function.

- *Representation:* The search space associated is constituted by all the subsets of TR. This is accomplished by using a binary representation. A chromosome consists of N genes (one for each instance in TR) with two possible states: 0 and 1. If the gene is 1, its associated instance is included in the subset of TR represented by the chromosome. If it is 0, this does not occur.
- *Fitness Function:* Let S be a subset of instances of TR and be coded by a chromosome. We define a fitness function based on the GM measure evaluated over TR.

$$Fitness(S) = GM. \quad (1)$$

This fitness function relates the proposal to the Evolutionary Under-Sampling guided by Classification Measures (EUSCM) proposed in [13]. The C4.5 decision tree is used for measuring the accuracy associated with the tree induced by using the instances selected in S. The accuracy independently computed in each class is useful to obtain GM value associated to the chromosome. The objective of the EAs is to maximize the fitness function defined: maximize the GM rate.

A mechanism to avoid overlearning in training data is needed in the fitness function. Although C4.5, in its standard definition, incorporates a pruning mechanism

Data set	#Examples	#Attributes	Class (min., maj.)	%Class(min.,maj.)
Abalone9-18	731	9	(18, 9)	(5.75, 94.25)
EcoliCP-IM	220	7	(im,cp)	(35.00, 65.00)
EcoliIM	336	7	(im,remainder)	(22.92, 77.08)
EcoliIMU	336	7	(iMU, remainder)	(10.42, 89.58)
EcoliOM	336	7	(om, remainder)	(6.74, 93.26)
German	1000	20	(1, 0)	(30.00, 70.00)
GlassBWFP	214	9	(build-window-float-proc, remainder)	(32.71, 67.29)
GlassBWNFP	214	9	(build-window-non_float-proc, remainder)	(35.51, 64.49)
GlassNW	214	9	(non-windows glass, remainder)	(23.93, 76.17)
GlassVWFP	214	9	(Ve-win-float-proc, remainder)	(7.94, 92.06)
Haberman	306	3	(Die, Survive)	(26.47, 73.53)
New-thyroid	215	5	(hypo,remainder)	(16.28, 83.72)
Pima	768	8	(1,0)	(34.77, 66.23)
VehicleVAN	846	18	(van,remainder)	(23.52, 76.48)
Vowel0	990	13	(0, remainder)	(9.01, 90.99)
YeastCYT-POX	483	8	(POX, CYT)	(4.14, 95.86)

Table 2. Imbalanced Data Sets.

to avoid overfitting, the inclusion of the induction tree process within an evolutionary cycle can direct the resulting tree to an optimal model for training data, loosing the generalization ability. We incorporate a simple and effective mechanism which consists of providing to the classification costs a higher weight (W) to the instances that are no included in S than to the instances included in S. An instance of TR well classified scores a value W if it is not included in S and a value of 1 if it is included in S. This procedure encourages the reduction ability of the selected subset, due to the fact that it is more beneficial to evaluate chromosomes with a higher number of examples out of the selected ones. Obviously, the instance causes a substraction on accuracy of the same magnitude in case of misclassification. Our empirical studies have determined that a value of W equal to 3 works appropriately.

- *Crossover operator for data reduction:* In order to achieve a good reduction rate, Heuristic Uniform Crossover (HUX) implemented for CHC undergoes a change that makes more difficult the inclusion of instances inside the selected subset. Therefore, if a HUX switches a bit on in a gene, then the bit could be switched off depending on a certain probability (its value will be specified in Section 4.1, Table 3).

- As the evolutionary computation method, we have used the CHC model [10, 5]. CHC is a classical evolutionary model that introduces different features to obtain a trade-off between exploration and exploitation; such as incest prevention, reinitialization of the search process when it becomes blocked and the competition among parents and offspring into the replacement process.

  During each generation the CHC develops the following steps.

  – It uses a parent population of size N to generate an intermediate population of N individuals, which are randomly paired and used to generate N potential offspring.

  – Then, a survival competition is held where the best N chromosomes from the parent and offspring populations are selected to form the next generation.

CHC also implements a form of heterogeneous recombination using HUX, a special recombination operator. HUX exchanges half of the bits that differ between parents, where the bit position to be exchanged is randomly determined. CHC also employs a method of incest prevention. Before applying HUX to the two parents, the Hamming distance between them is measured. Only those parents who differ from each other by some number of bits (mating threshold) are mated. The initial threshold is set at $L/4$, where L is the length of the chromosomes. If no offspring are inserted into the new population then the threshold is reduced by one.

No mutation is applied during the recombination phase. Instead, when the population converges or the search stops making progress (i.e., the difference threshold has dropped to zero and no new offspring are being generated which are better than any member of the parent population) the population is reinitialized to introduce new diversity to the search. The chromosome representing the best solution found over the course of the search is used as a template to reseed the population. Reseeding of the population is accomplished by randomly changing 35% of the bits in the template chromosome to form each of the other $N-1$ new chromosomes in the population. The search is then resumed.

4. Experimental Framework and Results

This section describes the methodology followed in the experimental study of the re-sampling compared techniques. We will explain the configuration of the experiment: used data sets and parameters for the algorithms. The algorithms used in the comparison are: OSS [16], NCL [17], SMOTE [6], SMOTE + Tomek Links (TL) and SMOTE + ENN [3].

4.1. Experimental Framework

Performance of the algorithms is analyzed by using 16 data sets taken from the UCI Machine Learning Database Repository [1]. Multi-class data sets are modified to obtain two-class non-balanced problems, defining one class as positive and one or more classes as negative.

The data sets are sorted by their IR values in an incremental way. The main characteristics of these data sets are summarized in Table 2. For each data set, it shows the number of examples (#Examples), number of attributes (#Attributes) and class name (minority and majority).

The data sets considered are partitioned using the *ten fold cross-validation (10-fcv)* procedure. The parameters of the used algorithms are presented in Table 3.

Algorithm	Parameters
SMOTE EUSTSS	$k = 5,\ Balancing\ Ratio = 1:1$ $Pop = 50,\ Eval = 10000,$ $Prob.\ inclusion\ HUX = 0.25,\ W = 3$

Table 3. Parameters considered for the algorithms.

4.2. Results and Analysis

Table 4 shows the results in test data obtained by the algorithms compared by means of GM evaluation measure. The column denoted by *none* corresponds to the case in which no re-sampling is performed previous to C4.5. The best case in each data set is remarked in bold.

Table 5 shows the average number of rules (or leafs) obtained by C4.5 in each data set.

Observing Tables 4 and 5, we can make the following analysis:

- EUSTSS proposal obtains the best average result in GM measure. It clearly outperforms the other under-sampling methods (OSS and NCL) and it improves the accuracy even when comparing with over-sampling approaches.
- Over-sampling techniques obtain better accuracy than under-sampling procedures in combination with C4.5 [3], but they cannot improve EUSTSS proposal.
- Except for OSS, EUSTSS produces decision trees with lower number of rules than the remaining methods. Although the combination OSS + C4.5 yields less rules, the accuracy in GM is the worst of all the re-sampling methods.
- Over-sampling techniques force C4.5 to produce many rules. This fact is not desirable when our interest lies in interpretable models.

We have included a second type of table accomplishing a statistical comparison of methods over multiple data sets. Demšar [9] recommends a set of simple, safe and robust non-parametric tests for statistical comparisons of classifiers. One of them is Wilcoxon Signed-Ranks Test [20]. Table 6 collects results of applying Wilcoxon's test between our proposed methods and the rest of re-sampling algorithms studied in this paper over the 16 data sets considered. This table is divided into two parts: In the first part, the measure of performance used is the accuracy classification in test set through GM. In the second part, we accomplish Wilcoxon's test by using as performance measure the number of rules yielded by C4.5. Each part of this table contains one column, representing our proposed methods, and N_a rows where N_a is the number of algorithms considered in this study. In each one of the cells can appear three symbols: +, = or -. They represent that the proposal outperforms (+), is similar (=) or is worse (-) in performance than the algorithm which appears in the column (Table 6). The value in brackets is the p-value obtained in the comparison and the level of significance considered is $\alpha = 0.10$.

algorithm	EUSTSS GM	EUSTSS num. rules
none	+ (.001)	+ (.088)
OSS	+ (.003)	- (.052)
NCL	+ (.047)	+ (.074)
SMOTE	+ (.052)	+ (.000)
SMOTE + TL	= (.501)	+ (.000)
SMOTE + ENN	= (.363)	+ (.000)

Table 6. Wilcoxon's test results over GM and number of rules

We make a brief analysis of results summarized in Table 6:

- The use of Wilcoxon's test confirms the improvement caused by EUSTSS over OSS and NCL under-sampling methods. Curiously, it statistically outperforms SMOTE, but it does not the hybrids of SMOTE.

dataset	none	OSS	NCL	SMOTE	SMOTE + TL	SMOTE + ENN	EUSTSS
abalone9-18	0.3763	0.4761	0.4963	0.6023	**0.6724**	**0.6724**	0.6697
ecoliCP-IM	**0.9787**	0.9486	**0.9787**	0.9751	0.9748	**0.9787**	**0.9787**
ecoliIM	0.8167	0.8882	0.8860	0.8795	**0.9060**	0.8811	0.8809
ecoliIMU	0.7709	0.7600	0.8092	0.8661	0.8137	**0.8671**	0.8579
ecoliOM	0.8073	0.8220	0.8749	0.8412	0.8010	0.8725	**0.9291**
german	0.5759	0.6437	**0.6753**	0.6410	0.6636	0.6658	0.6419
glassBWFP	0.8138	0.6652	0.7551	0.8216	0.7599	0.7971	**0.8425**
glassBWNFP	0.6934	0.5648	0.7353	0.7511	**0.7631**	0.7427	0.7235
glassNW	0.8942	0.8101	**0.9505**	0.9239	0.9373	0.9344	0.9321
glassVWFP	0.5286	0.6755	0.6884	0.6994	0.7572	0.4930	**0.7816**
haberman	0.4280	0.4329	0.6089	**0.6832**	0.6292	0.6022	0.6206
new-thyroid	0.9048	0.9132	0.8810	0.9193	**0.9492**	0.9414	0.9463
pima	0.6908	0.6457	0.7161	0.7155	0.6990	**0.7181**	0.7179
vehicle	0.9172	0.8737	0.9118	0.9202	0.9216	**0.9241**	0.9239
vowel0	**0.9808**	0.9360	**0.9808**	0.9657	0.9764	0.9671	0.9734
yeastCYT-POX	0.0699	0.7245	0.1000	0.5585	0.6156	0.6176	**0.6489**
AVERAGE	*0.7030*	*0.7363*	*0.7530*	*0.7977*	*0.8025*	*0.7922*	**0.8168**

Table 4. Results obtained by C4.5 using GM evaluation measure over test data

dataset	none	OSS	NCL	SMOTE	SMOTE + TL	SMOTE + ENN	EUSTSS
abalone9-18	8.10	6.50	7.30	57.50	57.30	52.60	6.30
ecoliCP-IM	2.00	2.50	2.00	2.90	3.10	2.00	2.00
ecoliIM	5.30	5.10	6.20	10.40	10.10	10.40	6.00
ecoliIMU	10.00	5.80	6.50	16.70	13.10	14.00	5.40
ecoliOM	3.90	3.40	4.40	7.80	6.60	6.80	5.40
german	91.00	35.30	57.60	159.90	121.00	82.40	33.60
glassBWFP	12.20	5.80	6.70	15.70	10.40	10.40	7.00
glassBWNFP	12.40	5.50	11.60	19.90	15.90	15.90	9.60
glassNW	6.70	4.10	4.40	9.70	6.90	7.10	5.60
glassVWFP	7.50	6.10	8.40	13.40	13.10	13.50	6.90
haberman	2.60	3.90	8.70	16.10	18.20	18.00	5.70
new-thyroid	4.10	2.60	4.30	4.90	4.90	5.00	4.30
pima	22.40	16.10	24.60	39.50	38.90	34.90	14.50
vehicle	20.60	12.50	16.30	28.40	23.40	22.50	11.10
vowel0	7.80	5.00	7.80	10.70	11.40	10.50	7.90
yeastCYT-POX	1.70	3.70	2.30	23.30	19.70	21.20	7.60
AVERAGE	13.64	7.74	11.19	27.30	23.38	20.45	8.68

Table 5. Average number of rules obtained by C4.5 decision tree

We have seen in Table 4 that SMOTE obtains a higher average GM than SMOTE + ENN, but Wilcoxon's test indicates us that SMOTE has an irregular behaviour depending on the data sets.

- In the case of interpretability, Wilcoxon's test again confirms the results observed in Table 5. The combination EUSTSS + C4.5 yields a low number of rules.
- EUSTSS outperforms OSS, NCL and SMOTE in GM measure and behaves similarly to SMOTE + TL and SMOTE + ENN. However, the number of rules produced by C4.5 when it is applied after EUSTSS is much lower than the produced by them. EUSTSS allows C4.5 to induce very precise trees with few rules.

5. Concluding Remarks

The purpose of this paper is to present a proposal of Evolutionary Training Set Selection Algorithm for C4.5 in imbalanced data sets. The results shows that our proposal allows to C4.5 to obtain very accurate trees with a low number of rules or leafs. The accuracy of the model is very competitive with respect to advanced hybrids of over-sampling and under-sampling, and the interpretability of the models obtained is increased.

Acknowledgement

This work was supported by TIN2005-08386-C05-01.

References

[1] A. Asuncion and D. Newman. UCI machine learning repository, 2007.

[2] R. Barandela, J. S. Sánchez, V. García, and E. Rangel. Strategies for learning in class imbalance problems. *Pattern Recognition*, 36(3):849–851, 2003.

[3] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations*, 6(1):20–29, 2004.

[4] J. R. Cano, F. Herrera, and M. Lozano. Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study. *IEEE Transactions on Evolutionary Computation*, 7(6):561–575, 2003.

[5] J. R. Cano, F. Herrera, and M. Lozano. Evolutionary stratified training set selection for extracting classification rules with trade-off precision-interpretability. *Data and Knowledge Engineering*, 60:90–108, 2007.

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[7] N. V. Chawla, D. A. Cieslak, L. O. Hall, and A. Joshi. Automatically countering imbalance and its empirical relationship to cost. *Data Mining and Knowledge Discovery. In press. DOI: 10.1007/s10618-008-0087-0*, 2008.

[8] N. V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explorations*, 6(1):1–6, 2004.

[9] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Rechearch*, 7:1–30, 2006.

[10] L. J. Eshelman. The CHC adaptive search algorithm: How to safe search when engaging in nontraditional genetic recombination. In G. J. E. Rawlings, editor, *Foundations of genetic algorithms*, pages 265–283. 1991.

[11] A. Estabrooks, T. Jo, and N. Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20(1):18–36, 2004.

[12] S. García, J. R. Cano, and F. Herrera. A memetic algorithm for evolutionary prototype selection: A scaling up approach. *Pattern Recognition*, 41(8):2693–2709, 2008.

[13] S. García and F. Herrera. Evolutionary under-sampling for classification with imbalanced data sets: Proposals and taxonomy. *Evolutionary Computation. In press.*, 2008.

[14] C. Guerra-Salcedo, S. Chen, D. Whitley, and S. Smith. Fast and accurate feature selection using hybrid genetic strategies. In *CEC*, pages 177–184, 1999.

[15] K. Huang, H. Yang, I. King, and M. R. Lyu. Imbalanced learning with a biased minimax probability machine. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 36(4):913–923, 2006.

[16] M. Kubat and S. Matwin. Addressing the course of imbalanced training sets: One-sided selection. In *ICML*, pages 179–186, 1997.

[17] J. Laurikkala. Improving identification of difficult small classes by balancing class distribution. In *AIME '01: Proceedings of the 8th Conference on AI in Medicine in Europe*, pages 63–66, 2001.

[18] A. Orriols-Puig and E. Bernadó-Mansilla. Evolutionary rule-based systems for imbalanced data sets. *Soft Computing. In press. DOI: 10.1007/s00500-008-0319-7*, 2008.

[19] R. J. Quinlan. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann, 1993.

[20] D. Sheskin. *Handbook of parametric and nonparametric statistical procedures.* Chapman & Hall/CRC, 2006.

[21] X. Wang, J. Yang, X. Teng, W. Xia, and R. Jensen. Feature selection based on rough sets and particle swarm optimization. *Pattern Recognition Letters*, 28(4):459–471, 2007.

[22] G. M. Weiss and F. J. Provost. Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315–354, 2003.

[23] D. Whitley, R. Beveridge, C. Guerra, and C. Graves. Messy genetic algorithms for subset feature selection. In *Proceedings of the International Conference on Genetic Algorithms*, pages 568–575, 1998.

[24] Q. Yang and X. Wu. 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, 5(4):597–604, 2006.