An Empirical Study of the Behavior of Classifiers on Imbalanced and Overlapped Data Sets

V. García[1,2], J.S. Sánchez[2], and R.A. Mollineda[2]

[1] Lab. Reconocimiento de Patrones, Instituto Tecnológico de Toluca
Av. Tecnológico s/n, 52140 Metepec (México)
[2] Dept. Llenguatges i Sistemes Informàtics, Universitat Jaume I
Av. Sos Baynat s/n, 12071 Castelló de la Plana (Spain)

Abstract. Class imbalance has been reported as an important obstacle to apply traditional learning algorithms to real-world domains. Recent investigations have questioned whether the imbalance is the unique factor that hinders the performance of classifiers. In this paper, we study the behavior of six algorithms when classifying imbalanced, overlapped data sets under uncommon situations (e.g., when the overall imbalance ratio is different from the local imbalance ratio in the overlap region). This is accomplished by analyzing the accuracy on each individual class, thus devising how those situations affect the majority and minority classes. The experiments corroborate that overlap is more important than imbalance for the classification performance. Also, they show that the classifiers behave differently depending on the nature of each model.

Keywords: Imbalance; overlapping; classifiers; performance measures.

1 Introduction

Many traditional approaches to pattern classification assume that the problem classes share similar prior probabilities. However, in many real-world problems, this assumption is grossly violated. Often, the ratios of prior probabilities between classes are extremely skewed. This situation is known as the imbalance problem. A data set is said to be imbalanced when one of the classes (the minority one) is heavily under-represented in comparison to the other (the majority) class.

Most of the research addressing this problem concentrates on balancing the class distribution in the data set. The different proposals can be roughly classified into three categories: assigning distinct costs to the classification errors for positive and negative examples [7], resampling the original training set, either by over-sampling the minority class [5] and/or under-sampling the majority class [11] until the classes are approximately equally represented and, internally biasing the discrimination-based process so as to compensate for the class imbalance [1].

Recently, several works have pointed out that there does not exist a direct correlation between class imbalance and the loss of performance [13]. These studies suggest that the class imbalance is not a problem by itself, but the degradation of performance is also related to other factors. Different works explore the combined effects of class imbalance with the presence of small disjuncts [10], the size of the training set [9, 12], and the degree of overlapping between classes [2, 13].

In the present work, we study the behavior of six classifiers under the presence of both class imbalance and overlap. This study mainly tries to devise the influence of changes in the overlap region on the classifiers, while keeping the majority/minority ratio constant. The aim is to show that these practical situations affect the classifier performance depending on the characteristics of the particular algorithm used, that is, similar situations may produce different results. In order to accomplish this, we experiment with artificial data sets, employing some performance measures that estimate the accuracy on each individual class.

2 The Classifiers

In this section, we briefly describe the classifiers selected for the present experimental study. All these algorithms work under the assumption that there exists a set of n previously labelled examples (training set, TS), say $\mathcal{X} = \{(x_1, \omega_1), (x_2, \omega_2), \ldots, (x_n, \omega_n)\}$, where each element has an attribute vector x_i and a class label ω_i.

2.1 Nearest Neighbor Rule

One of the most popular non-parametric classification approaches corresponds to the k nearest neighbor (kNN) decision rule [6]. In brief, this classifier consists of assigning a new input sample **x** to the class most frequently represented among the k closest instances in the TS, according to a certain dissimilarity measure (generally, the Euclidean distance metric). A particular case is when $k = 1$, in which an input sample is decided to belong to the class indicated by its closest neighbor.

The characteristics of the kNN classifier need the entire TS stored in computer memory, what causes large time and memory requirements. On the other hand, the kNN rule is extremely sensitive to the presence of noisy, atypical and/or erroneously labelled cases in the TS.

2.2 Naïve Bayes Classifier

The naïve Bayes classifier (NBC) [8] is arguably one of the simplest probabilistic schemes, following from Bayesian decision theory. The model constructed by this algorithm is a set of probabilities, each one corresponding to the probability $P(f_i|c)$ that a specific feature f_i appear in the instances of class c. These probabilities are estimated by counting the frequency of each feature value in the instances of a class in the TS. Given a new instance, the classifier estimates the probability that the instance belongs to a specific class, based on the product of the individual conditional probabilities for the feature values in the instance.

NBC is based on an independent feature model, that is, all the attributes are independent given the value of the class variable. Despite the conditional independence assumption is rarely true in most real-world applications, the algorithm tends to perform well in many scenarios and it learns more rapidly than most induction algorithms.

2.3 C4.5 Decision Tree

The C4.5 algorithm [14] uses a greedy technique to induce decision trees. A decision-tree model is built by analyzing training data and the model is used to classify unseen data. The nodes of the tree evaluate the existence or significance of individual features. Following a path from the root to the leaves of the tree, a sequence of such tests is performed resulting in a decision about the appropriate class of new objects.

The decision trees are constructed in a top-down fashion by choosing the most appropriate attribute each time. An information-theoretic measure is used to evaluate features, which provides an indication of the ”classification power” of each feature. Once a feature is chosen, the training data are divided into subsets, corresponding to different values of the selected feature, and the process is repeated for each subset until a large proportion of the instances in each subset belongs to a single class.

2.4 Multilayer Perceptron

The multilayer perceptron (MLP) neural network [3] usually comprises one input layer, one or more hidden layers, and one output layer. Input nodes correspond to features, hidden layers are used for computations, and output layers are the problem classes. A neuron is the elemental unit of each layer. It computes the weighted sum of its inputs, adds a bias term and drives the result thought a generally nonlinear (commonly, sigmoid) activation function to produce a single output.

The most popular training algorithm for MLP is the backpropagation, which takes a set of training instances for the learning process. For the given feedforward network, the weights are initialized to small random numbers. Each training instance is passed through the network and the output from each unit is computed. The target output is compared with the output estimated by the network to calculate the error, which is fed back through the network. To adjust the weights, backpropagation uses gradient descent to minimize the squared error between the target output and the computed output. At each unit in the network, starting from the output unit and moving down to the hidden units, its error value is used to adjust weights of its connections so as to reduce the error. This process of adjusting the weights is repeated for a fixed number of times or until the error is small or it cannot be reduced.

2.5 Radial Basis Function

The radial basis function (RBF) [4] neural network, which has three layers, can be seen as an especial kind of multilayer feedforward networks. Each unit in the hidden layer employs a radial basis function, such as Gaussian kernel, as the activation function. The output units implement a weighted sum of hidden unit outputs. The input into an RBF network is nonlinear. The output is linear. The kernel is centered at the point specified by the weight vector associated with the unit. Both the positions and the widths of these kernels are learned from training instances. Each output unit implements a linear combination of these radial basis functions.

An RBF is trained to learn the centers and widths of the Gaussian function for hidden units, and then to adjust weights in the regression model that is used at the output unit. To learn the centers of the Gaussian functions, the k-means algorithm can be used, obtaining k Gaussian functions for each attribute in the instance. After the parameters for the Gaussian function at the hidden units have been found, the weights from these units to the output unit are adjusted using a linear regression model.

2.6 Support Vector Machine

Support vector machines (SVMs) [15] are a set of related supervised learning methods used for classification and regression. They belong to a family of generalized linear classifiers. A special property of SVMs is that they simultaneously minimize the empirical classification error and maximize the geometric margin; hence they are also known as maximum margin classifiers.

SVMs map input vectors to a higher dimensional space where a maximal separating hyperplane is constructed. Two parallel hyperplanes are constructed on each side of the hyperplane that separates the data. The separating hyperplane is the hyperplane that maximizes the distance between the two parallel hyperplanes. An assumption is made that the larger the margin or distance between these parallel hyperplanes the better the generalization error of the classifier will be.

3 Performance Evaluation in Class Imbalance Problems

In the literature, there appear many different proposals for measuring the performance of learning algorithms in imbalanced domains. However, they all agree in the need of measuring the classification performance over each individual class, instead of the overall accuracy, because the misclassification costs can be different for each of them. Most of performance measures for two-class problems are built over a 2×2 confusion matrix as illustrated in Table 1.

Table 1. Confusion matrix for a two-class problem

	Positive prediction	Negative prediction
Positive class	True Positive (TP)	False Negative (FN)
Negative class	False Positive (FP)	True Negative (TN)

From the confusion matrix, apart from other more elaborated performance measures, one can obtain several simple metrics on the positive (minority) and negative (majority) classes, along with the overall accuracy and the overall error rate:

True Positive Rate: $TPR = \frac{TP}{TP+FN}$ True Negative Rate: $TNR = \frac{TN}{TN+FP}$

False Positive Rate: $FPR = \frac{FP}{FP+TN}$ False Negative Rate: $FNR = \frac{FN}{TP+FN}$

Accuracy: $Acc = \frac{TP+TN}{TP+FP+FN+TN}$ Error Rate: $Err = \frac{FP+FN}{TP+FP+FN+TN}$

In the present work, since the purpose is to analyze the classifier behavior on each individual class, we adopt the TPR (or a^+) and the TNR (or a^-).

4 Experimental Data Sets and Results

With the aim of analyzing the behavior of each classifier in situations of imbalance and overlap, two groups of experiments have been carried out. In both cases, pseudo-random bivariate patterns have been generated following a uniform distribution in a square of length 100. There are 400 negative examples and 100 patterns from the minority class, keeping the overall majority/minority ratio equal to 4 in all cases.

The experiments have been performed using the Weka toolkit [16] with the learning algorithms described in Sect. 2, that is, 1-NN, NBC, MLP, J48 (a reimplementation of C4.5), RBF, and SVM. We have adopted a 10-fold cross-validation method: each data set was divided into ten equal parts, using nine folds as the TS and the remaining block as an independent test set. This process has been repeated ten times and the results correspond to the average over the 100 runs.

4.1 Constant Imbalance with Increasing Overlap

This experiment will be over a collection of six data sets with increasing overlap. In all cases, the positive examples are defined on the X-axis in the range [50..100], while the majority class instances are generated in [0..50] for 0% of overlap, [10..60] for 20%, [20..70] for 40%, [30..80] for 60%, [40..90] for 80%, and [50..100] for 100% of overlap. Fig. 1 illustrates two of these data sets. It is expected a similar behavior of all classifiers since in the overlap region, the minority class is also under-represented.

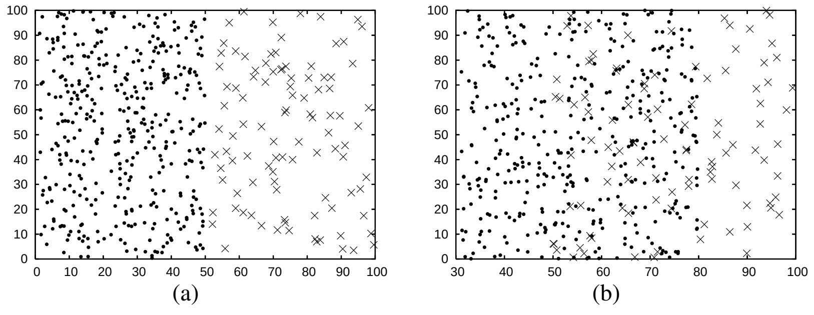

Fig. 1. Two different levels of class overlapping: (a) 0% and (b) 60%

The analysis of Fig. 2 allows to point up a number of results. First, it should be reminded that the majority/minority ratio keeps constant for all data sets. Despite this, the accuracy on the minority class varies in function of the overlap level. While a^+ is close to 100% over the data set with no overlap, it drastically degrades as the overlapping increases (independently of the algorithm used). This corroborates the conclusions of other investigations stating that the loss of classifier performance is not especially due to class imbalance, but to other factors (e.g., overlapping).

Another important observation is the different behavior of the 1-NN classifier on the majority class when compared to the rest of algorithms. In the case of 1-NN, the increase of overlapping produces a degradation of the accuracy on the majority class (a^-), due to the local nature of this classifier. Oppositely, the algorithms based on a more global learning keep the a^- close to 100% independently of the overlap degree.

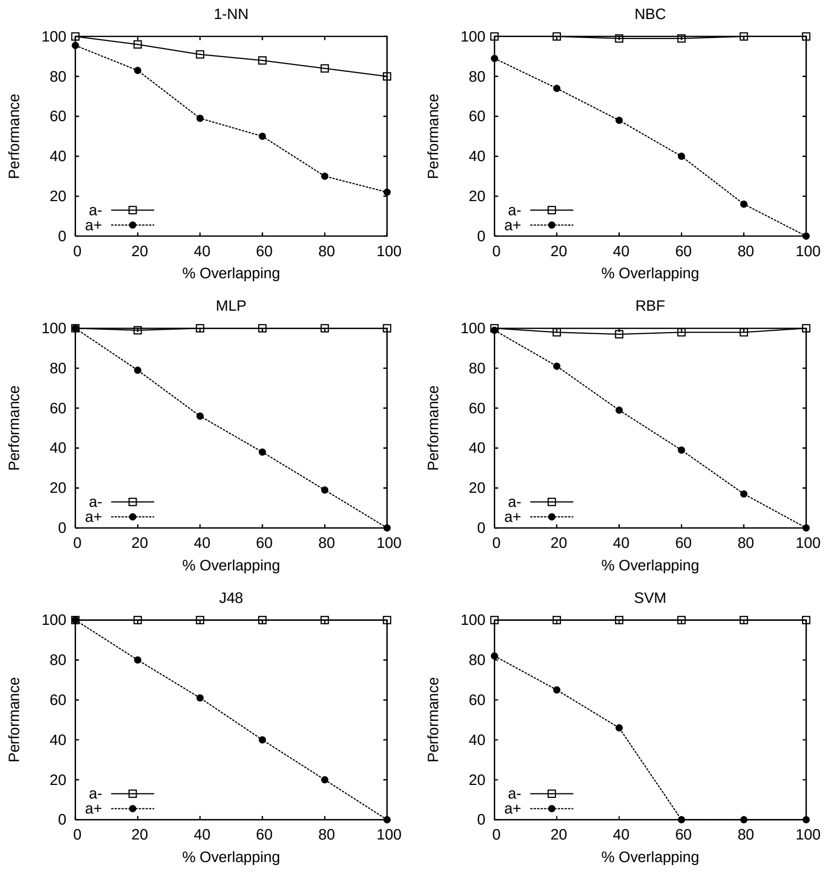

Fig. 2. Classifier performances on each class for the first experiment

In order to measure whether differences between each pair of classifiers are significant or not, a one-tailed paired t-test has been used as implemented in Weka. Table 2 gives a comparison among the learning algorithms for each individual class. The three values in each entry of this table refer to how many times the algorithm of the column has been significantly-better/same/significantly-worse than the classifier of the row. As can be seen, there exist statistically significant differences when comparing 1-NN with the remaining algorithms. For example, NBC, MLP, J48 and SVM on the majority class have been better than 1-NN in five data sets, whereas on the minority class they have been clearly worse. Comparisons with NBC, MLP, RBF and J48 reveal that in general, differences are not statistically significant. On the minority class, SVM has been significantly worse than any other algorithm.

Table 2. Statistical comparison of the classifiers used in the first experiment

	NBC		MLP		RBF		J48		SVM	
	a^-	a^+	a^-	a^+	a^-	a^+	a^-	a^+	a^-	a^+
1NN	5/1/0	0/1/5	5/1/0	0/3/3	5/1/0	0/3/3	5/1/0	0/3/3	5/1/0	0/0/6
NBC			0/6/0	1/5/0	0/4/2	2/4/0	0/6/0	2/4/0	0/6/0	0/1/5
MLP					0/3/3	0/6/0	0/6/0	1/5/0	0/6/0	0/1/5
RBF							3/3/0	0/6/0	4/2/0	0/1/5
J48									0/6/0	0/1/5

4.2 Changing the Imbalance in the Overlap Region

The second experiment uses a collection of five data sets in which the minority class becomes more representative than the majority one in the overlap region. The aim of this is to analyze the behavior of the classifiers when the overall imbalance is different from the imbalance in the overlap region. To this end, the 400 negative examples have been defined on the X-axis to be in the range [0..100] in all data sets, while the 100 positive cases have been generated in the ranges [75..100], [80..100], [85..100], [90..100], and [95..100] (see Fig. 3 for two examples). The number of elements in the overlap region varies from no local imbalance in the first case, where both classes have the same number of patterns and density, to a critical inverse imbalance in the last case, where the minority class appears as majority in the overlap region due to an increase in density.

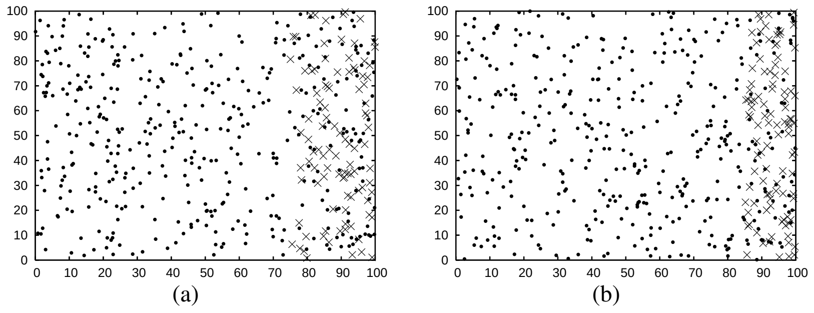

Fig. 3. Two data sets used in the second experiment: (a) [75..100] and (b) [85..100]. Note that the minority class is much denser in the latter case where its samples are confined in a smaller space

Although in general, the behavior of all classifiers on the minority class improves when this class is denser, thus suggesting the influence of the imbalance in the overlap region, Fig 4 provides some interesting, even surprising results. While the 1-NN performance on the majority class is clearly better than that on the minority one, the other classifiers show a very different behavior. It is especially worth pointing up the results achieved by NBC, in which a^+ is higher than a^- over all data sets. A similar behavior is shown by J48, except when both classes are equally represented in the overlap region (75–100). In this case, the accuracy on the minority class abruptly drops down about 50 points. Results on SVM show very high accuracy on the minority class for the case [95–100], and close to 0% when this class becomes less dense.

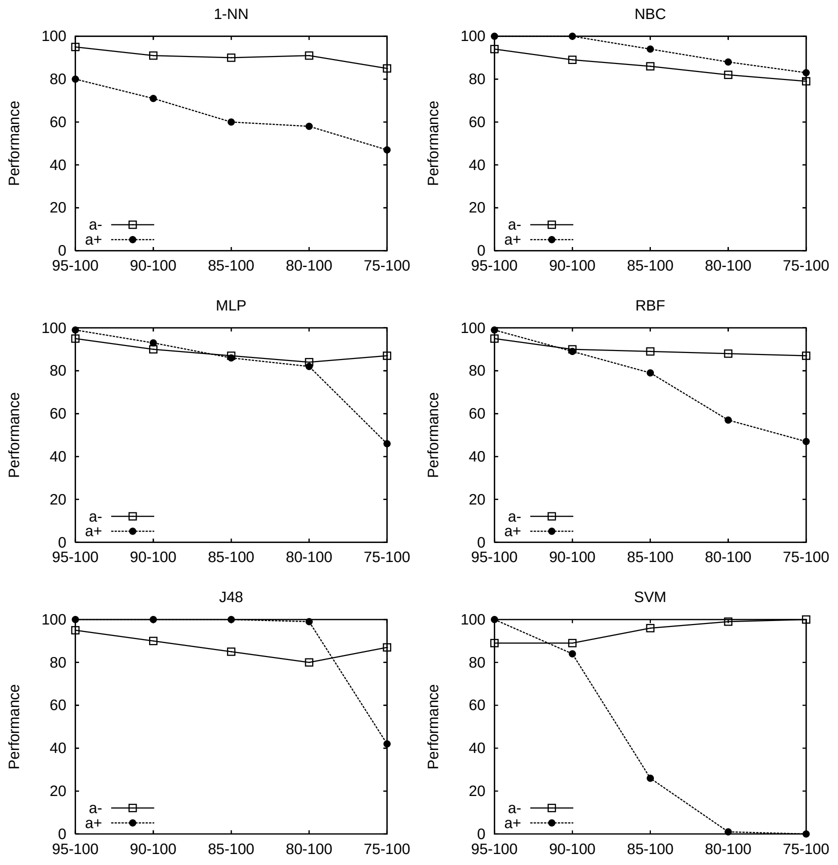

Fig. 4. Classifier accuracies on each class for the second experiment

With respect to MLP and RBF, a^+ keeps superior to a^- for the cases in which the majority class is under-represented in the overlap region. Nevertheless, when the number of negative examples is close to the number of positive patterns, the accuracy on the minority class rapidly degrades. In summary, one can observe that in most cases, the performance of NBC, MLP, RBF and J48 on the minority class is higher than that of the majority class, whereas 1-NN and SVM exhibit the opposite behavior. This can be explained by the data characteristics and the operation of both these classifiers. The most surprising effect is on SVM, where a^+ drops down rapidly as the imbalance ratio in the overlap region approaches the overall imabalce.

Analyzing the results in Table 3, it has to be noted that in most cases all classifiers are significantly better than 1-NN in a^+, while there are no significant differences in a^- (except with respect to NBC and SVM). On the other hand, when comparing NBC, MLP, RBF and J48, it seems that there are very few differences. It can be mentioned that the RBF accuracy on the minority class has been significantly worse than that of the NBC four times, while significantly better in the case of a^- also four times. In general, SVM shows the most significant difference in both a^+ (worse) and a^- (better).

Table 3. Statistical comparison of the classifiers used in the second experiment

	NBC		MLP		RBF		J48		SVM	
	a^-	a^+	a^-	a^+	a^-	a^+	a^-	a^+	a^-	a^+
1NN	0/1/4	5/0/0	0/3/2	4/1/0	0/5/0	3/2/0	0/3/2	4/1/0	3/1/1	2/0/3
NBC			2/3/0	0/3/2	4/1/0	0/1/4	0/4/1	2/2/1	3/1/1	0/1/4
MLP					1/4/0	0/4/1	0/3/2	3/2/0	3/1/1	0/2/3
RBF							0/3/2	3/2/0	3/1/1	0/2/3
J48									3/1/1	0/1/4

5 Conclusions and Future Work

In the framework of imbalanced data sets, recent studies have concluded that performance degradation is not solely caused by class imbalance, but is also related to class overlapping and other data characteristics. In the present work, we have conducted an empirical analysis with the aim of establishing the behavior of several classifiers under combined situations of class imbalance and class overlap. The algorithms here studied correspond to representatives of widely-used models in order to draw more general conclusions: 1-NN, NBC, MLP, RBF, J48, and SVM.

The experiments carried out have revealed some interesting results. First, it has been corroborated that the class imbalance is not a problem by itself, but the loss of performance is also related to other critical factors (class overlap, small disjoints). Second, the combined effects of imbalance and overlap strongly depend on the characteristics of each classifier. Third, it has been observed that 1-NN and SVM are very sensitive to noise in the data set, while NBC seems to be the most tolerant to it. Finally, it has been shown that in certain uncommon situations (like that in the second experiment), the results are quite different from those expected in an imbalanced scenario, that is, the performance on the majority class appears poorer than that on the minority class.

Future research includes an extension of this study to real-world problems. Also, the present work should be complemented with the analysis of other important factors that can strongly affect the classifier performance in imbalanced domains. In particular, we are especially interested in exploring the effects of feature dimensionality. Finally, the use of classifier ensembles with the aim of exploiting the main characteristics of each individual classifier seems an interesting field to be further investigated.

Acknowledgment

This work has been partially supported by grants DPI2006–15542 from the Spanish CICYT and SEP-2003-C02-44225 from the Mexican CONACyT.

References

1. Barandela R, Sánchez JS, García V, Rangel E (2003) Strategies for learning in class imbalance problems. Pattern Recognition 36:849–851
2. Batista GE, Prati RC, Monard MC (2005) Balancing strategies and class overlapping. In: Proc. 6th Intl. Symposium on Intelligent Data Analysis, Madrid, Spain, pp. 24–35.
3. Bishop C (1995) Neural Networks for Pattern Recognition. Oxford University Press, New York
4. Buhmann M, Albowitz M (2003) Radial Basis Functions: Theory and Implementations. Cambridge University Press
5. Chawla NV, Bowyer KW, Hall L, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. Journal of Artificial Intelligence Research 16:321–357.
6. Cover TM, Hart PE (1967) Nearest neighbor pattern classification. IEEE Trans. on Information Theory 13:21–27.
7. Domingos P (1999) Metacost: a general method for making classifiers cost-sensitive. In: Proc. 5th Intl. Conf. on Knowledge Discovery and Data Mining, San Diego, CA, pp. 155–164.
8. Duda RO, Hart PE, Stork DG (2001) Pattern Classification and Scene Analysis. John Wiley & Sons, New York
9. Japkowicz N, Stphen S (2002) The class imbalance problem: a systematic study. Intelligent Data Analysis 6:40–49.
10. Jo T, Japkowicz N (2004) Class imbalances versus small disjuncts. ACM SIGKDD Explorations Newsletters 6:40–49.
11. Kubat M, Matwin S (1997) Adressing the curse of imbalanced training sets: one-sided selection. In: Proc. 14th Intl. Conf. on Machine Learning, Nashville, USA, pp. 179–186.
12. Orriols A, Bernadó E (2005) The class imbalance problem in learning classifier systems: a preliminary study. In: Proc. of Conf. on Genetic and Evolutionary Computation, Washington DC, USA, pp. 74–78.
13. Prati RC, Batista GE, Monard MC (2004) Class imbalance versus class overlapping: an analysis of a learning system behavior. In: Proc. 3rd Mexican Intl. Conf. on Artificial Intelligence, Mexico City, Mexico, pp. 312–321.
14. Quinlan JR (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA
15. Vapnik V, Kotz S (2006) Estimation of Dependences Based on Empirical Data. Springer, New York.
16. Witten IH, Frank E (2005) Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, San Francisco, CA.