Improving Classification by Removing or Relabeling Mislabeled Instances

Stéphane Lallich, Fabrice Muhlenbach, and Djamel A. Zighed

ERIC Laboratory – University of Lyon 2
5, av. Pierre Mendès-France
F-69676 BRON Cedex – FRANCE
{lallich, fabrice.muhlenbach, zighed}@univ-lyon2.fr

Abstract. It is common that a database contains noisy data. An important source of noise consists in mislabeled training instances. We present a new approach that deals with improving classification accuracies in such a case by using a preliminary filtering procedure. An example is suspect when in its neighborhood defined by a geometrical graph the proportion of examples of the same class is not significantly greater than in the whole database. Such suspect examples in the training data can be removed or relabeled. The filtered training set is then provided as input to learning algorithm. Our experiments on ten benchmarks of UCI Machine Learning Repository using 1-NN as the final algorithm show that removing give better results than relabeling. Removing allows maintaining the generalization error rate when we introduce from 0 to 20% of noise on the class, especially when classes are well separable.

1 Introduction – Outliers Issue

In this paper, we address the learning process of a categorical variable Y, on the basis of an example database described by p numerical attributes, denoted by X_1, X_2, ... X_p. Our focus is on mislabeled examples that constitute a specific category of outliers. We suggest a filtering strategy which identifies suspect examples in order to improve the generalization performance of the learning algorithm. We consider two options: removal or relabeling. This strategy is based on the *cut weighted edges statistic* [16] defined by geometrical neighborhood [17] and associated with 1-NN prediction.

Identifying outliers is an important step in any instance-based knowledge discovery process [2]. By outliers, we mean examples whose exceptional nature disturbs generalization. Barnett and Lewis [1] define an outlier as "an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data".

Outliers can have different origins, and we will now handle the most common. We will first mention inclusion errors, that happen when an example is wrongly included in the learning set. Either the reference population was wrongly defined or the error happened at the time of sampling, as can be seen in medicine or marketing. Then, we consider appreciation, coding or data-input errors, be it on

M.-S. Hacid et al. (Eds.): ISMIS 2002, LNAI 2366, pp. 5–15, 2002.
© Springer-Verlag Berlin Heidelberg 2002

the predictors or on the class. One should also talk about observations regarding rare examples, which are often associated with variable asymmetry and are an important cause for leverage points. Lastly, the model error can be strongly affected if a relevant attribute has been forgotten and this may wrongly show some examples as outliers.

Outliers disturb the learning process, mainly when the latter includes the variable's statistical moments; the mean and even more so the variance are usually very sensitive to exceptional values. Estimations and especially confidence intervals and the associated p-values may then be distorted. This can lead to faulty conclusions, particularly in a regression.

We distinguish between works which aim at identifying outliers for themselves (e.g., in the cases of fraud detection or records [7]) and those which look into limiting their noisy effects [6,4,5]). Our study belongs to this latter category. In order to remedy to the outliers issue in this second case, we can either remove the suspect examples from the learning set [6,5], or relabel them (cf. relaxation models [8]).

In the learning process, we often prefer to talk about noise, distinguishing the noise on the class from the noise on the attributes [10]. In this last case, Quinlan has shown that when the noise level increases, the fact of removing the noise from the attributes decreases the generalization performance of the classifier if the data to be classified presents the same attribute noise. As regards the class noise, the problem can be seen in different terms, because noise only concerns the learning set. Thus, Brodley and Friedl [4,5] have demonstrated that whatever the base or the filtering strategy experimented, identifying and removing those examples improves substantially the predictive accuracy in generalization as long as the level of noise does not exceed 20%, or even 30 to 40% in some cases.

2 Works Related to Mislabeled Examples

We will now get into the details of the works that deal with class noise. Because our aim is not to reduce the size of the learning set but to filter it, we will mainly discuss the work of Wilson [14], John [6] as well as Brodley and Friedl [4,5].

Wilson [14] has suggested the E-k-NN rule (Edited k Nearest Neighbor rule) which consists in using the k-NN classifier ($k = 3$) to filter the whole of the learning set before proceeding to the prediction by 1-NN. Only instances that the k-NN classifies properly are retained for the 1-NN. This rule edits out mislabeled instances, as well as the ones which are close border cases, leaving smoother decision boundaries while retaining all internal instances. Therefore this algorithm clearly is a filtering one rather than a reducing one. One may apply the E-k-NN rule repeatedly, until all remaining instances have a majority of same class neighbors. An alternative use was introduced by Tomek [12], who applies repeatedly the E-k-NN rule for growing values of k. This process was included by Wilson and Martinez [15] in different techniques of instances selection (DROP 3 for filtering) for instance-based learning algorithms.

As regards decision trees methods, and more particularly C4.5, John [6] suggests to build robust trees. After the complete tree is built, through successive splits of nodes, C4.5 prunes it to make it simpler and avoid overfitting on noisy examples (cf. Quinlan [11]). John defines confusing instances as those incorrectly classified by the final decision tree, obtained after pruning. He removes them from the learning set before rebuilding the tree using the reduced learning set. On the basis of his experiments, John shows a slight but meaningful decrease in the mean error and in the variability of the error rate while reducing the number of nodes by up to 70%.

Brodley and Friedl [4,5] introduce a two-step procedure to improve classification accuracies in case of mislabeled training data. The first step consists in identifying mislabeled examples and removing them from the training set. The authors use m ($m = 3$) learning algorithms to create classifiers that serve to filter training data using a 10-fold cross-validation. An example may be qualified as mislabeled either at the majority (2 out of 3) or by consensus (all 3) of filtering algorithms. The three filtering algorithms are decision trees (C4.5), k-NN and Linear Machine (LM). The LM classifier associates discriminant linear functions related to each class to assign an instance to one of the classes. For the second step, a classification algorithm, 1-NN or decision tree, is applied to the final training set. Experiments have been carried on five databases. Brodley and Friedl introduce from 0 to 40% of random noise on the labels into the training data in order to simulate the most plausible labeling errors. Results show that filtering allows to maintain accuracy near to the baseline accuracy (when 0% of noise) for noise levels up to 20% on all databases, and for noise levels up to 30% on two of the databases. As Brodley and Friedl point it out [5], in order to improve classification accuracy, one needs to proceed in two steps: a filtering step to identify and handle mislabeled examples followed by a prediction step on the reduced training set. The uniqueness of our approach lies on the filtering procedure which calls up *neighborhood graphs* and *cut weighted edges* notions. We will first go into the details of those notions and then introduce our method.

3 Neighborhood Graphs and Cut Weighted Edges

A neighborhood graph [17] can explain that two examples are close or not in the representation space. Such graphs are the Relative Neighborhood Graph (RNG), the Gabriel Graph (GG), the Delaunay Triangulation(DT) or the Minimum Spanning Tree (MST). For our work we have used the Relative Neighborhood Graph of Toussaint [13].

Definition: Let V be a set of points in a real space R^p (with p the number of attributes). The Relative Neighborhood Graph (RNG) of V is a graph with vertices set V and the set of edges of the RNG graph of V which are exactly those pairs (a, b) of points for which $d(a, b) \leq Max\left(d\left(a, c\right), d\left(b, c\right)\right) \forall c, c \neq a, b$, where $d(u, v)$ denotes the distance between two points u and v in R^p.

This definition means that *lune* $L_{(u,v)}$ –constituted by the intersections of hypercircles centered on u and v with range the edge (u, v)– is empty. For exam-

ple, on **Fig. 1** (a), vertices 13 and 15 are connected because there is no vertex on the *lune* $L_{(13,15)}$.

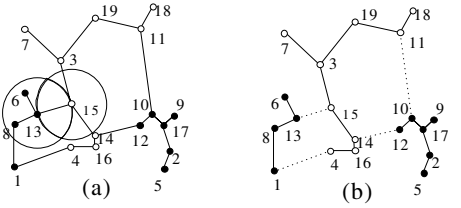


Fig. 1. Relative Neighborhood Graph and clusters with two classes: the black and the white points

Following Zighed et Sebban [17] we introduce the concept of "cluster" to express that a set of close points have the same class. We call *cluster* a connected sub-graph of the neighborhood graph where all vertices have the same class. To construct all clusters required for characterizing the structures of the scattered data points, we proceed in two steps:

1. we generate the geometrical neighborhood graph on the learning set (DT, GG, RNG or MST);
2. we suppress the edges connecting two vertices belonging to different classes, obtaining connected sub-graphs where all vertices have the same class.

The number of clusters generated gives a partial information on the class separability. If a number of clusters is low –at least the number of classes–, the classes are well separable and we can find a learning method capable of exhibit the underlying model. For example on **Fig. 1** (b), after having cut the four edges connecting vertices of different colors (in dotted line), we have three clusters for the two classes. But if this number tends to increase, close to the number of clusters that we could have in a random situation, the classes can not be learned.

Actually, this number of clusters cannot ever characterize some little situations that seems intuitively different. For the same number of clusters, the situation can be very different depending on wether the clusters are easily isolated in the neighborhood graph or not. As soon as $p > 1$, rather than study the number of clusters, we will prefer to take an interest in the edges cut for building the clusters and we will calculate the relative weight of these edges in the edges set. In our example on **Fig. 1** (b), we have cut four edges for isolating three clusters. To take into account the distances between the considered example i and its neighbors (i.e., j), it is possible to weight the edges. The weight, denoted $w_{i,j}$, can be based on the distances or ranks. When $w_{i,j} = 1$ the weight matrix W is equivalent to the connection matrix. If W is not symmetrical we can make it symmetrical by computing its transposed matrix and making the average between the components of the two matrices. Since the term *cut weighted edges statistic* is introduced [16].

4 New Method

Dealing with mislabeled points raises three types of problems:

- identification of suspect examples;
- handling of suspect examples;
- prediction based on the filtered training set.

The uniqueness of our method lies on both the identification of mislabeled points which exploits locally *cut weighted edges statistic* and in the handling of such examples. In a previous work we chose to remove them [9]. As suggested by Brodley and Friedl [5] another option may be contemplated which consists in relabeling *spurious* examples. The contribution of our article is to propose a method that associates both options *remove* and *relabel* while letting the user the possibility to adjust the definition of both options.

4.1 Identification of Suspect Examples

An example label is subject to caution when its class is different to the one of the examples belonging to its geometrical neighborhood. And this even more as classes are well discernible in the representation space. In order to identify such an example, we offer to calculate the sum of the cut edges weights that run from this example (i.e., edges which link this example to other examples in its geometrical neighborhood that do not have the same label).

Let i be an example whose class is $y_{r(i)}$. We denote by $\pi_{r(i)}$ the global proportion of the class $y_{r(i)}$ in the learning set. An example i is considered as a *good* example if in its geometrical neighborhood the proportion of examples that do not have the same label $y_{r(i)}$ is significantly smaller than $1 - \pi_{r(i)}$.

We denote H_0 the matching null hypothesis. Thus, for a good example the weight of cut edges is significantly smaller than its expected value under H_0.

Under H_0, the probability for an example in the neighborhood of i not to be of the same class than i is $1 - \pi_{r(i)}$. We note n_i the number of examples belonging to the neighborhood of i, w_{ij} is the weight of the edge connecting the vertices i and j, and J_i is the absolute weight of cut edges running from i.

$$J_i = \sum_{j=1}^{n_i} w_{ij} I_i(j)$$

where $I_i(j)$ are independent and identically distributed random variables, according to the Bernouilli law of parameter $1 - \pi_{r(i)}$, under H_0. The mean E and the variance Var of J_i under H_0 are given by:

$$E(J_i/H_0) = \left(1 - \pi_{r(i)}\right) \sum_{j=1}^{n_i} w_{ij}$$

$$Var(J_i/H_0) = \pi_{r(i)} \left(1 - \pi_{r(i)}\right) \sum_{j=1}^{n_i} w_{ij}^2$$

We propose to rank examples i, $i = 1, 2, ..., n$, according to the level of the normal repartition function to which the realization of J_i corresponds under H_0 (left unilateral p-value). In doing so, we define ideally three categories of examples: *good* examples located in the left rejection region (significantly less cut

edges than expected under H_0, and consequently more neighbors of same label), *doubtful* examples that do not contradict H_0 and *spurious* examples located in the right rejection region (significantly more cut edges than expected under H_0, and consequently less neighbors of same label).

If n_i is great enough and the weights not too unbalanced, one can use the normal approximation of J_i^s (J_i standardized). Otherwise, we can proceed to a simulation in order to calculate the associated p-values to each J_i. In order to efficiently characterize those three types of examples, we have chosen to control two parameters that are fixed by the user:

- θ_1, the left risk, poses the frontier between *good* and *doubtful* examples;
- θ_2, the complement of the right risk, poses the frontier between *doubtful* and *spurious* examples.

An example will be considered as *good*, *doubtful* or *spurious*, depending on J_i^s left unilateral p-value being smaller than θ_1, between θ_1 and θ_2 or greater than θ_2. By playing with θ_1 and θ_2, we can modulate the definition of each type of example. The closer θ_1 is to 0, the more we are severe for the definition of *good* examples. The closer θ_2 is to 1, the more we are severe for the *spurious* examples. The closer θ_2 is to θ_1, the less is the place for *doubtful* examples.

4.2 Suspect Examples Processing: Remove or Relabel

To process *suspect* examples (respectively *doubtful* or *spurious*), we can either remove these examples [6,5] or relabel them [8]. Our formalization allows to associate in a general way *doubtful* example removing and *spurious* examples relabeling when playing on θ_1 and θ_2 parameters (option "*Remove* and *Relabel* (θ_1, θ_2)").

Another possibility consists in considering only two categories of examples, by gathering the *doubtful* and *spurious* types in one *suspect* type ($\theta = \theta_1 = \theta_2$), and using only one of the two processing. We then have the choice between removing (option "*Remove* (θ)" used in a preceding work [9]) and relabeling (option "*Relabel* (θ)") as suggested by Brodley and Friedl in the conclusion of their work [5].

In practice, considering the bad results of relabeling, whatever the chosen option and the value of control parameters, we introduce and experiment a further option (option "*Relabel else Remove* (θ)"). This one consists in relabeling a *suspect* example according to the majority label amongst the neighbors of the very example which are *good* examples. If the *suspect* example does not have neighbors that are *good* examples, the very example is removed (**Fig. 2**).

4.3 Prediction from Filtered Training Set

Once *spurious* or *doubtful* exemples are identified, one can apply a prediction algorithm to the final training set stemming from the filtering procedure. For our experiments, we used 1-NN algorithm. Other prediction algorithms can be preferred in order to extract more comprehensible knowledge like decision trees.

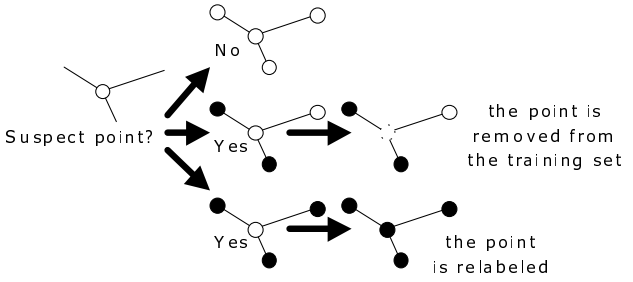


Fig. 2. How to deal with the *suspect* points

4.4 Complexity of the Method

The method needs different steps. Computing the matrix distance is in $O(k \times n^2)$, with n the number of examples and k the classes. Building the neighborhood graph in R^p is in $O(n^3)$ (with p the number of attributes). Removing or relabeling the suspect points is in $O(n^2)$. The method is then in $O(n^3)$ due to the neighborhood graph computation.

5 Experimental Validation

5.1 Organization of the Experiments

We have performed experiments on a collection of ten domains from the UCI Repository of Machine Learning Databases [3]. The domains have been chosen for having only numerical attributes –continuous or discrete or boolean (transformed in 0/1 form) data– with a qualitative class attribute. Because the geometrical neighborhood are sensitive to the scale measure, we have standardized all attributes of the domains. The noise is introduced on the training set only otherwise we couldn't see if we have correctly removed or relabeled the data when we generalize the model.

For each domain, we have repeated the following process:

- first, we have split the base in two sets of the same size: the training set and the testing set of data;
- on the training set, we have introduced some noise by relabeling the class of the data by another one for 0 to 20% of the total of instances;
- we have used the Relative Neighborhood Graph of Toussaint [13] for generating a geometrical neighborhood graph;
- with this graph, we have used our statistical test based on the *cut weighted edges* for detecting the suspect points;
- we then composed our learning model (1) without filtering, (2) by *removing* only the suspect instances or (3) by *relabeling else removing* the suspect instances, with $\theta = 0.1$;
- we have tested our three models on the testing set using 1-NN;
- we have repeated this 25 times.

We have tried different values of θ_1 and θ_2 for the option "Remove and Relabel" and different values of θ for the option "Relabel" but the results were not fruitful, thus we present the results obtained for the option "Relabel else Relabel" with one value of $\theta = 0.1$.

5.2 Experimental Results

On **Fig. 3** we show the number of removed and relabeled instances (related to the total size of the learning set of the domain) when varying the noise from 0 to 10% on the learning set for the ten domains.

Dataset	Total points	Removing only						Relabeling else Removing											
		Removed points						*Removed points*						*Relabeled points*					
	Noise	0%	2%	4%	6%	8%	10%	0%	2%	4%	6%	8%	10%	0%	2%	4%	6%	8%	10%
Breast cancer	341	19%	20%	25%	29%	31%	34%	13%	12%	15%	17%	18%	21%	4%	4%	4%	4%	4%	4%
House votes 84	217	28%	33%	38%	41%	43%	48%	13%	17%	20%	22%	24%	27%	10%	10%	8%	8%	6%	7%
Image segmentation	105	16%	18%	22%	24%	26%	29%	8%	10%	13%	15%	16%	19%	6%	6%	6%	6%	5%	5%
Ionosphere	175	67%	70%	73%	76%	76%	78%	61%	64%	67%	71%	71%	74%	2%	2%	2%	1%	2%	1%
Iris plants	75	19%	25%	27%	28%	34%	39%	12%	18%	19%	21%	26%	31%	6%	6%	6%	6%	5%	5%
Iris (Bezdek)	75	19%	25%	27%	28%	34%	39%	13%	18%	20%	21%	26%	31%	6%	7%	6%	5%	5%	5%
Musk clean 1	238	60%	62%	66%	67%	70%	70%	45%	49%	53%	54%	58%	59%	10%	10%	9%	8%	7%	6%
Pima Indians diabetes	384	71%	72%	75%	75%	76%	78%	58%	60%	63%	64%	65%	68%	8%	6%	6%	6%	5%	5%
Waveform (Breiman)	500	32%	36%	39%	42%	44%	46%	14%	16%	18%	19%	22%	24%	13%	13%	13%	12%	12%	12%
Wine recognition	89	23%	23%	29%	31%	35%	39%	14%	13%	18%	20%	23%	27%	2%	2%	2%	2%	2%	2%

Fig. 3. Percentage of removed or relabeled instances for a noise rate from 0 to 10%

The results on **Fig. 4** and **Fig. 5** show the error rate and standard deviation on generalization when varying the noise rate from 0 to 20%. On these figures, we present the results obtained when we test **All** instances (no filtering) when we remove all suspect instances (**Removed** points) or when we do the **r**elabeling else **r**emoving process to the suspect instances (**R else R**).

For our experiments, we have chosen only databases that have a reduced error rate in generalization (less than 30%) because our filtering method can only deal with databases that have a sufficient class separability. At 0% of noise, our filtering method –with the "Removing" or "Relabeling else Removing" option– is better than using all the database on 6 bases out of 10. From 4% of noise, filtering is better on 9 bases out of 10.

Except for the *Breast cancer* database, the results are surprisingly better when we remove all suspect points instead of relabeling else removing. The *Breast cancer* database is the easiest to learn, the class values are well separable, the results after filtering are better for *Relabel else Remove* than for *Remove*, and this even without noise in the database.

As one can see on **Fig. 3**, for each domain the number of removed examples is much higher than the number of relabeled examples. This can be explained by the fact that we only relabel examples whom neighbors are considered as well labeled. When the neighbors of the example considered are from heterogeneous classes, the relabeling cannot be performed and the very example is removed.

The rate of examples removed or relabeled varies from a domain to another but the number of removed points grows linearly as a fonction of the noise

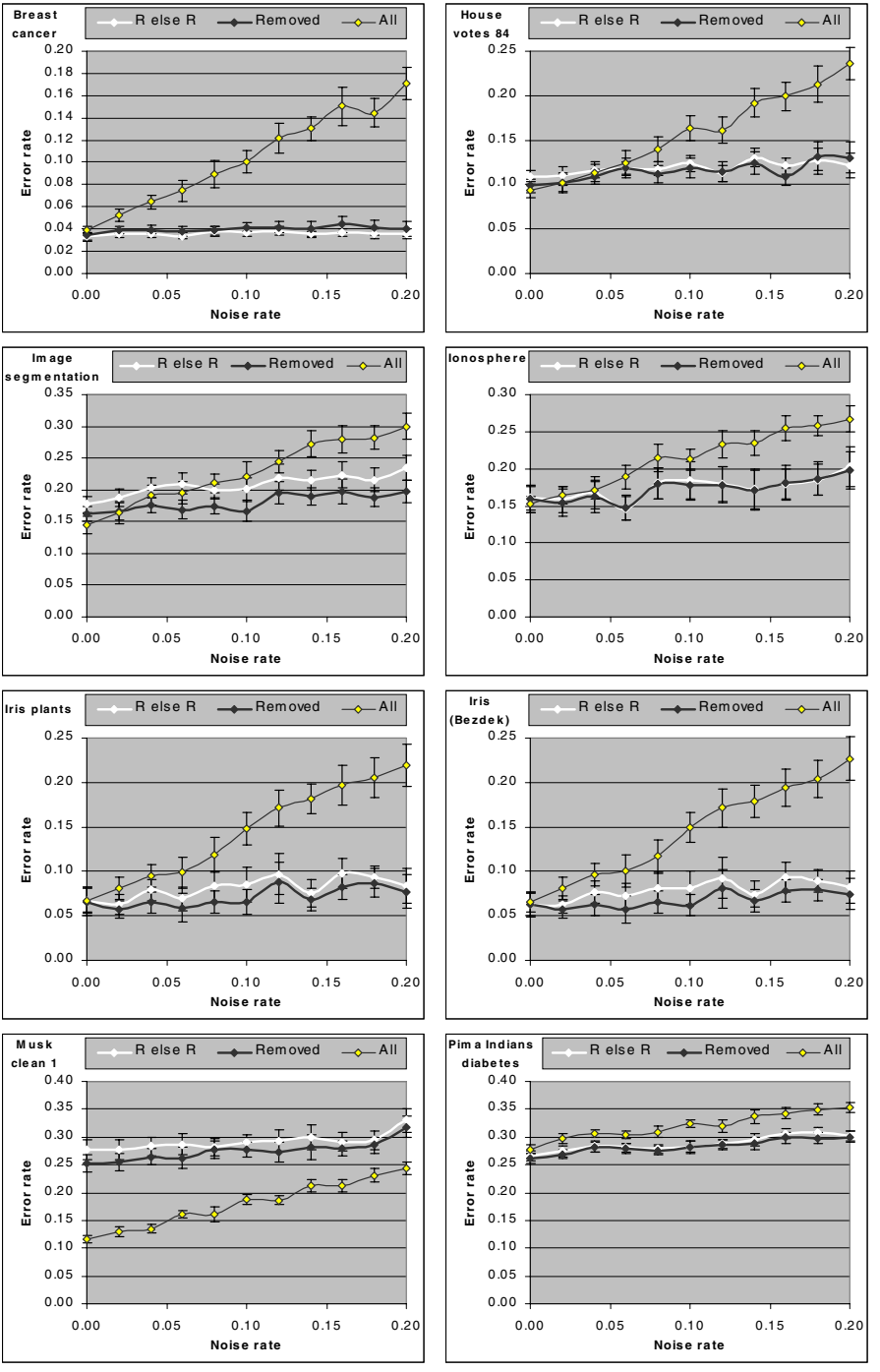


Fig. 4. Error rate on the domains *Breast cancer*, *House vote 84*, *Image segmentation*, *Ionosphere*, *Iris plants*, *Iris Bezdek*, *Musk "clean 1"* and *Pima Indians diabetes*

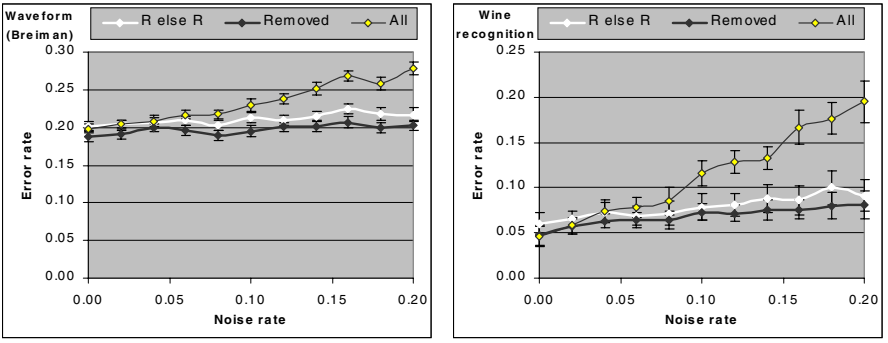


Fig. 5. Error rate on the domains *Waveform* and *Wine recognition*

whereas the number of relabeled points is the same or decreases just a little. Our filtering method is not adapted to *Musk "clean 1"* database probably because it contains many attributes (166) for a reduced number of instances (238 for the learning set) even if the base is not so difficult to learn. The filtering procedure removes many instances (like for *Pima Indians diabetes* but the results are not so bad for this base). This is actually a classical problem that occurs when the classes are not well separable. Finally when we apply our filtering process on some benchmarks we remove or relabel too many instances, even if there is no noise. We can correct this by taking a higher value for our θ criterium used in the test for detecting the suspect points.

6 Conclusion and Future Work

The results obtained seem quite promising, especially when the database can be successfully learn by geometrical neighborhood.

One can notice that the filtering procedure is more efficient when we simply remove suspect examples. The filtering procedure proposed can be associated to other classifiers than the nearest neighbor, such as decision trees.

Otherwise filtering procedure allows to correct not only the class noise but also the noise on attributes, particularly if we use a robust standardization. In fact, an example having a noisy value for an attribute will be misplaced in R^p. This example will be suspected if its neighbors are not of the same class. In this case it seems more logical to remove it than to relabel it.

The more the class values are separable, the more successful is our method. Our method is all the more so successful since the class values are separable. To improve this method we intend to link the value of the parameter θ used in the test and the result of a prior test of classificability, for example our *cut weighted edges test* [16]. In that way, the user applying the method to real-life data can adjust the θ parameter, which controls the definition of suspect examples.

References

1. V. Barnett and T. Lewis. *Outliers in statistical data*. Wiley, Norwich, 1984.
2. R. J. Beckman and R. D. Cooks. Oulier...s. *Technometrics*, 25:119–149, 1983.
3. C. L. Blake and C. J. Merz. UCI repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science [http://www.ics.uci.edu/∼mlearn/MLRepository.html], 1998.
4. C. E. Brodley and M. A. Friedl. Identifying and eliminating mislabeled training instances. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 799–805, Portland OR, 1996. AAI Press.
5. C. E. Brodley and M. A. Friedl. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11:131–167, 1999.
6. G. H. John. Robust decision trees: removing outliers from data. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 174–179, Montréal, Québec, 1995. AAI Press.
7. E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *The VLDB Journal*, 8(3):237–253, February 2000.
8. C. Largeron. *Reconnaissance des formes par relaxation : un modèle d'aide à la décision*. PhD thesis, Université Lyon 1, 1991.
9. F. Muhlenbach, S. Lallich, and D. A. Zighed. Amélioration d'une classification par filtrage des exemples mal étiquetés. *ECA*, 1(4):155–166, 2001.
10. J. R. Quinlan. Induction of decisions trees. *Machine Learning*, 1:81–106, 1986.
11. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.
12. I. Tomek. An experiment with the edited nearest neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, 6(6):448–452, 1976.
13. G. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern recognition*, 12:261–268, 1980.
14. D. Wilson. Asymptotic properties of nearest neighbors rules using edited data. *In IEEE Transactions on systems, Man and Cybernetics*, 2:408–421, 1972.
15. D. R. Wilson and T. R. Martinez. Reduction techniques for exemplar-based learning algorithms. *Machine Learning*, 38:257–268, 2000.
16. D. A. Zighed, S. Lallich, and F. Muhlenbach. Séparabilité des classes dans R^p. In *Actes des 8èmes Rencontres de la SFC*, pages 356–363, 2001.
17. D. A. Zighed and M. Sebban. Sélection et validation statistique de variables et de prototypes. In M. Sebban and G. Venturini, editors, *Apprentissage automatique*. Hermès Science, 1999.