Neurocomputing 275 (2018) 2374–2383

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A two-stage ensemble method for the detection of class-label noise

Maryam Sabzevari*, Gonzalo Martínez-Muñoz, Alberto Suárez

Computer Science Department, Escuela Politécnica Superior, Universidad Autónoma de Madrid, C/ Francisco Tomás y Valiente, 11, Madrid 28049, Spain

ARTICLE INFO

Article history:
Received 3 August 2017
Revised 3 November 2017
Accepted 5 November 2017
Available online 10 November 2017

Communicated by Prof. Dianhui Wang

Keywords:
Noise detection
Ensemble learning
Subsampling
Robust classification
Random forest

ABSTRACT

The properties of bootstrap ensembles, such as bagging or random forest, are utilized to detect and handle label noise in classification problems. The first observation is that subsampling is a regularization mechanism that can be used to render bootstrap ensembles more robust to this type of noise. Furthermore, appropriate values of the sampling rate can be estimated using out-of-bag data. A second observation is that the ensemble classifiers tend to make more errors in incorrectly labeled instances. Thus, instances for which a sufficiently large fraction of ensemble predictors err are marked as noisy. Suitable values of this threshold, which are problem dependent, are determined by cross-validation using a wrapper method. Instances identified as noisy can then be either filtered (i.e. discarded for training), or cleaned by correcting their class labels. Finally, an ensemble is built afresh on these cleansed training data. Extensive experiments in classification problems from different areas of application show that this procedure is effective to build accurate ensembles, even in the presence of high levels of class-label noise.

1. Introduction

The presence of noise, which is often unavoidable in real-world settings, is an important nuisance factor that needs to be taken into account in the design of learning algorithms [5,20]. Two types of noise can be found in the data used for automatic induction: class-label and feature noise [20]. Except when the latter is very strong, the presence of erroneous class labels is generally more harmful for learning and generalization [20]. According to its statistical properties label noise falls into three categories [5]. In the *Noisy Completely at Random* (NCAR) model the probability of a mislabeled instance is uniform in feature and class spaces. In the *Noisy at Random* (NAR) model, the probability of mislabeled instances is independent of the feature values, but depends on the class. Finally, in the *Noisy Not at Random* (NNAR) models there are dependencies between the class-label noise, and both feature and class values. In this work, we assume that the noise is completely at random. Nevertheless, the method proposed can be readily adapted to take into account other types of class-label noise.

One way of alleviating the effects of noisy data is to incorporate some regularization mechanism into the design of the learning algorithm [2,6,18]. Another approach is to carry out a preprocessing step in which noisy instances are identified. These instances are then either discarded (filtering), or their class label corrected (cleaning) so that they do not interfere in the learning process [5]. In this work, we combine both strategies using randomized ensembles. Our goal is to take advantage of the robustness of bootstrap ensembles, such as bagging and random forest, to handle noise. Specifically, we will use subsampling as a regularization mechanism, which allows one to build ensembles that are robust to class-label noise [13,14]. Another observation is that the individual classifiers in the ensemble tend to make more prediction errors on incorrectly labeled instances. Therefore, the fraction of incorrect ensemble predictions can be used as an indicator to detect noisy instances. Standard ensemble-based approaches to this problem remove instances for which more than half of the votes are incorrect (majority filtering) or where all votes are incorrect (consensus filtering). However, these choices need not be optimal. In this work, we propose to use a wrapper method to determine a near-optimal value for the percentage of incorrect votes necessary to identify a noisy instance.

The structure of the work is as follows: In Section 2, we present a review of previous work on learning from incorrectly labeled data. Particular emphasis is given to the use of subsampling to improve the robustness to class-label noise of bootstrap ensembles. The conclusions of this analysis are applied in Section 3 to the design of a method for noise detection. In Section 4, we present the results of an empirical evaluation of the proposed procedure and a comparison with other state-of-the-art noise detection methods. Finally, the conclusions of this work are summarized in Section 5.

* Corresponding author.
E-mail address: maryam.sabzevari@uam.es (M. Sabzevari).

https://doi.org/10.1016/j.neucom.2017.11.012
0925-2312/

2. Previous work

The problem of induction from noisy data has been extensively addressed in the area of ensemble learning [2,5,6,13,14,18]. Furthermore, as illustrated by recent implementations [8], these methods can be scaled to deal with class-label noise in large problems. Early on in the literature on ensembles, it was noted that the accuracy improvements of an ensemble with respect to a single learner are generally smaller when the training data are contaminated with class-label noise [1]. However, ensembles are generally more robust to this type of noise than single learners. In [3], an ensemble of three classifiers (a univariate decision tree, a k-nearest neighbor and a linear machine) is used to identify noisy instances. The noise detection protocol in this method is as follows: The training set is partitioned into 4-folds. Then, an ensemble is trained using three of these folds. The ensemble is then used to identify noisy instances in the fold not used for training. In this study, two filtering strategies are considered: majority filtering and consensus filtering. In majority filtering, a particular instance of the left-out fold is identified as noisy when the predictions of more than half of the learners are erroneous. The consensus filtering rule is more conservative. An instance is categorized as noisy when all the members of ensemble misclassify it. This process is repeated for each fold to identify noisy instances in all 4 folds. In this study, majority filtering exhibits better overall performance than consensus filtering. In [9] a similar approach is used, albeit with an ensemble of 25 classifiers of different types. The increased diversity of this heterogeneous ensemble reduces the risk of false positives in the noise detection process. In addition, different intermediate strategies between majority and consensus filtering are explored. In the problems investigated, the optimal threshold of erroneous ensemble predictions used to identify an instance as noisy was close to consensus filtering.

In [17], an ensemble of Top-down Induction of Logical Decision Trees (Tilde) is used to filter the training data. In this study either cross-validation or bootstrap sampling are used to generate the training sets on which the base learners are built. Majority and consensus filtering are then applied to identify noisy instances. The best results are obtained with majority filtering. In addition, the use of boosting in noise detection problems is explored. The idea is to filter out instances whose weights are above a threshold after a predefined number of iterations of the boosting algorithm. In the problems investigated, this strategy does not perform well.

In [21], a distributed method for large datasets is presented. In this method, the original training set is partitioned into small subsets. A classifier is induced from each of the these subsets. These classifiers are then used to classify the instances in the complete training set. The local error of an instance is defined as the fraction of classifiers whose training data included that particular instance and misclassified it. The global error for an instance is computed using the classifiers that did not include that instance in their training sets. Majority and consensus filtering are used to identify noisy instances. A necessary condition for an instance to be identified as noisy is that it be misclassified by the base learners whose training sets induced it. The justification is that a classifier has usually higher accuracy on instances that are in its training set. In this work, majority filtering is found to yield better results than consensus filtering.

In [19], a noise detection strategy that combines ideas from bagging and boosting is investigated. In this method, all training instances are initially assigned equal weights. Then, bootstrap samples from the original training data are used to build different classifiers. For each instance a noise count is computed as the number of base learners that have misclassified the instance. The noise count is used in a boosting-like iterative process in which the weight of the instances with higher noise counts is decreased. This process is iterated for a predefined number of rounds. Finally, instances whose noise count values are higher than a threshold are labelled as noisy. Ensemble methods based on ranking have been explored in [15]. In this work, instances are ranked according to the number of base learners that make incorrect predictions. In the medical dataset analyzed, a domain expert singles out the instances with highest ranks for further analysis. Expert knowledge is then used to determine the type of anomaly of the instances with the highest ranks (labeling error, outlier, complex medical case, etc.).

In most of the noise detection methods based on ensembles, majority or consensus filtering are used. Majority filtering was found to be better than consensus filtering in most studies in which homogeneous ensembles were used [3,16,17,21]. By contrast, in small heterogeneous ensembles, agreement rates close to consensus filtering seem to be more effective [9,16]. A qualitative explanation of this observation can be given: In homogeneous ensembles the base learners are of the same type. Diversity is commonly obtained using randomization strategies, such as bootstrap sampling (e.g. in bagging and random forest). Thus, the decision boundaries of the individual classifiers are similar to each other. In fact, the decision boundary is a refinement of the decision border of the individual base learners [12]. Therefore, noisy examples close to this decision boundary can be detected only if majority voting is used. By contrast, in heterogeneous ensembles, the decision borders are more varied because the individual classifiers are of different types. For this reason the dispersion of class labels assignments may simply reflect this variability. In consequence, higher agreement rates should be used to mark an instance as noise. In summary, the optimal agreement rates for noise detection depends on the classification problem and type of ensemble. However, as far as we are aware, the influence of the agreement rate on the effectiveness of the noise detection method has not been systematically evaluated hitherto.

3. A wrapper method for class-label noise detection

In this work, ensembles are used to detect and handle noise in the class labels of the training instances. We focus on randomized ensembles, such as bagging and random forest, in which the individual classifiers are trained on bootstrap samples of the original data. The design of the algorithm leverages two observations: The first one is that reducing the sampling rate in the bootstrapping step can make the ensemble more robust to class-label noise [13,14]. A second observation is that the fraction of erroneous predictions by the ensemble classifiers necessary to detect noisy instances depends on the classification task at hand. Therefore, the value of the threshold θ used to mark an instance as noisy should be estimated for each problem during the training phase. The method proposed proceeds in two stages: First, the optimal sampling rate for the bootstrapping step in the construction of the ensemble is determined using out-of-bag data. Then, the optimal threshold for disagreement between the ensemble predictions and the actual class of the instance is determined by means of a wrapper method. Finally, instances labeled as noise are either removed (*filtering*) or relabeled (*cleaning*) and an ensemble is built afresh from the cleansed training data.

The pseudocode of the method is detailed in Algorithm 1. In the first stage of the algorithm (steps 1–8 in Algorithm 1), ensembles are built using different values of the sampling rate: 0.1, 0.2, 0.4, 0.6, 0.8, 1.0, and 1.2. From these, H_T^*, the ensemble that is expected to generalize best is selected and kept for the next phase. In our implementation, out-of-bag instances are used to estimate the generalization error and carry out this selection. In this way, an unbiased selection is achieved, independently of the amount of noise present in the dataset. According to empirical evidence

Algorithm 1: Noise detection using bootstrap ensembles.

Input: $\mathcal{D}_{train} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N_{train}}$ % Training set
bootstrap_ensemble % Bootstrap ensemble method
T % Ensemble size
wrapper_learner % Wrapper learner method
cleanse_type % either filtering or cleaning
Output: $\mathcal{D}_{cleansed}$ % Cleansed set

1 $min\_error \leftarrow \infty$ % determine optimal sampling rate
2 **foreach** *sampling_rate* **in** [0.1, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2] **do**
3 $H_T \leftarrow bootstrap\_ensemble(\mathcal{D}_{train}, sampling\_rate, T)$
4 $error \leftarrow estimate\_error(H_T, \mathcal{D}_{train})$
5 **if** $error <$ min_error **then**
6 $min\_error \leftarrow error$
7 $sampling\_rate^* \leftarrow sampling\_rate$
8 $H_T^* \leftarrow H_T$
9 min_error $\leftarrow \infty$ % determine optimal disagreement rate
10 **foreach** θ **in** [0.5, 0.6, 0.7, 0.8, 0.9, 1.0] **do**
11 $\mathcal{D}_{cleansed} \leftarrow cleanse\_with\_oob(\mathcal{D}_{train}, cleanse\_type, H_T^*, \theta)$
12 $error \leftarrow cv\_error(wrapper\_learner, \mathcal{D}_{cleansed}, K = 3)$
13 **if** $error <$ min_error **then**
14 $\theta^* \leftarrow \theta$
15 $\mathcal{D}_{cleansed} \leftarrow cleanse\_with\_oob(\mathcal{D}_{train}, cleanse\_type, H_T^*, \theta^*)$

[13,14], the higher the level of class-label noise, the lower the value of the subsampling rate that is selected. Although the possibility that the presence of noisy instances lead to an incorrect selection of the best ensemble cannot be ruled out, we have not observed this effect in the experiments carried out. This is probably due to the fact that random forests are fairly robust to class-label noise, even without cleansing.

In the second stage (steps 9–14 in Algorithm 1), the predictions of the base classifiers of the selected ensemble, H_T^*, are used to identify noisy instances. Different values of the threshold θ for the disagreement rate are considered; namely, 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0. For a given value of θ, instances for which the fraction of incorrect predictions given by the classifiers in the selected ensemble is above this threshold are marked as noise (step 11). Since the data being cleaned and the data used to train H_T^* are the same, only the predictions of base classifiers in H_T^* whose training set does not include that particular instance are used (i.e. out-of-bag instances). Cleansing consists in either correcting the label of noisy instances (*cleaning*), or eliminating them (*filtering*). Then, we use a wrapper method and compute estimates of the generalization error of a learner built on the cleansed training data. In our implementation, this error is estimated using K-fold cross-validation. The optimal value of the disagreement threshold, θ^*, is the one that minimizes this estimate of the generalization error. The details of this selection are as follows: In each of the K iterations of the cross validation procedure, one of the K folds is set apart for validation. Then, a classifier is trained on remaining $K-1$ folds cleansed using the corresponding value of θ. The classifier is then evaluated on the left-out fold, which is not cleansed. Finally, the cross validation error is calculated by averaging the errors of the validation folds in each of the K iterations. The value selected, θ^*, is the threshold that minimizes this cross-validation estimate of the generalization error. This optimal threshold value, θ^*, and the ensemble H_T^* are then used to clean or filter the training data. Note that the type of classifier used in the wrapper step is a parameter of the algorithm. In general, we think it is preferable to use the same classifier as the one that is eventually trained with the cleansed data. In the following section we carry out an extensive empirical analysis of the accuracy and resilience to noise of ensembles built in this manner.

Table 1
Characteristics of the classification problems used in the empirical evaluation.

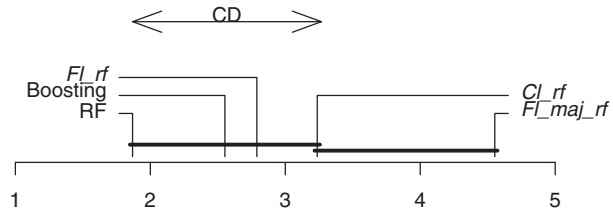
Dataset	Training	Test	Attrib.
Australian	460	230	14
Blood transfusion	499	499	5
Boston	337	169	14
Breast	466	233	9
Chess	2130	1065	37
Crx	460	230	15
Diabetes	512	256	8
German	667	333	20
Heart	178	92	13
Horse-colic	246	122	21
Ionosphere	234	117	34
Liver	230	115	6
Parkinsons	130	65	22
Ringnorm	300	2000	20
Sonar	491	208	60
Spambase	3067	1534	58
Threenorm	300	2000	20
Tic-Tac-Toe	639	319	9
Twonorm	300	2000	20

4. Empirical evaluation

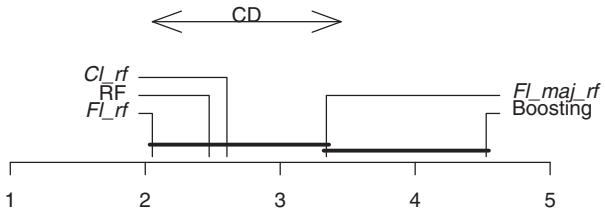
To assess the effectiveness of the proposed noise detection procedure extensive experiments have been carried out in 19 binary classification problems from the UCI repository [11]. The characteristics of the datasets are summarized in Table 1. The implementation makes use of the R *randomForest* package [10] and the R *adabag* package for AdaBoost [7]. In both packages the default settings are used. Specifically, for random forests, the number of variables considered for splitting at the inner nodes of the random trees is the square root of the total number of attributes in the problem. The minimum size of a terminal node is set to 1. Trees in the forest are grown to their maximum possible size. For AdaBoost, weighted resampling is used. The coefficient that controls the weight update is $\alpha = 1/2\ln((1-err)/err)$.

In all the classification tasks, with the exception of *Ringnorm, Threenorm* and *Twonorm*, which are synthetic problems, the labeled instances are randomly assigned to the training and test sets. The sizes of these sets are 2/3 and 1/3 of the original set, respectively. For synthetic problems, 300 examples are used for training and 2000 for testing. In all cases, stratified sampling is used. The results reported are averages over 50 executions. In these executions, different random partitions of the data into training and test sets, or, in synthetic problems, independent realizations of the data are used. The following protocol is followed for each classification task and execution:

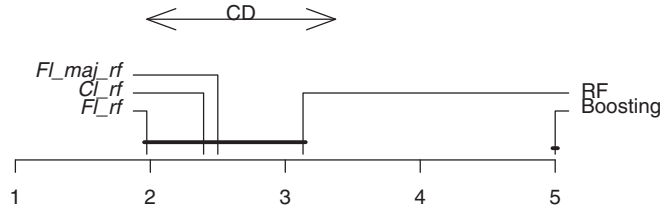
1. First, noise is injected in the data by randomly switching the class label of a random subset of the training instances. Different noise rates are considered: 0 (no noise is injected), 10, 20, 30 and 40%. This type of class-label noise is known as completely at random noise (NCAR) [5].
2. For each noise level, ensembles composed of 501 trees are trained using the following bootstrap sampling ratios: 10, 20, 40, 60, 80, 100(standard) and 120%. Then, the out-of-bag error is computed for each ensemble. The ensemble with the best out-of-bag accuracy (H_T^*) is kept for the next step. Both random forests and bagging ensembles of unpruned CART trees have been considered. According to the results of our experiments, they are both equally effective to identify noisy instances. Therefore, since random forest are generally more ac-



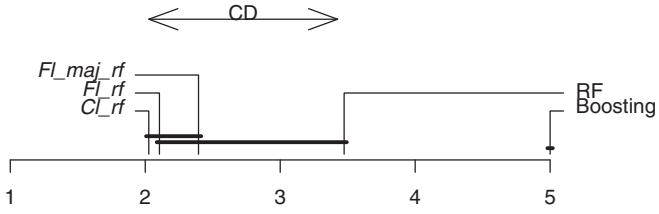
(a) Without injected noise



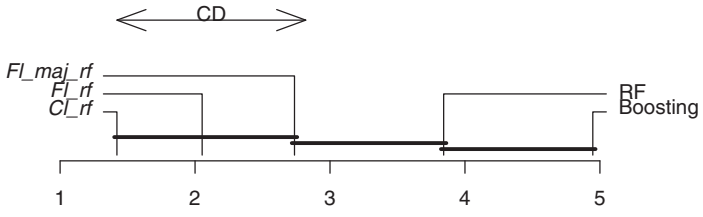
(b) 10% injected noise



(c) 20% injected noise



(d) 30% injected noise

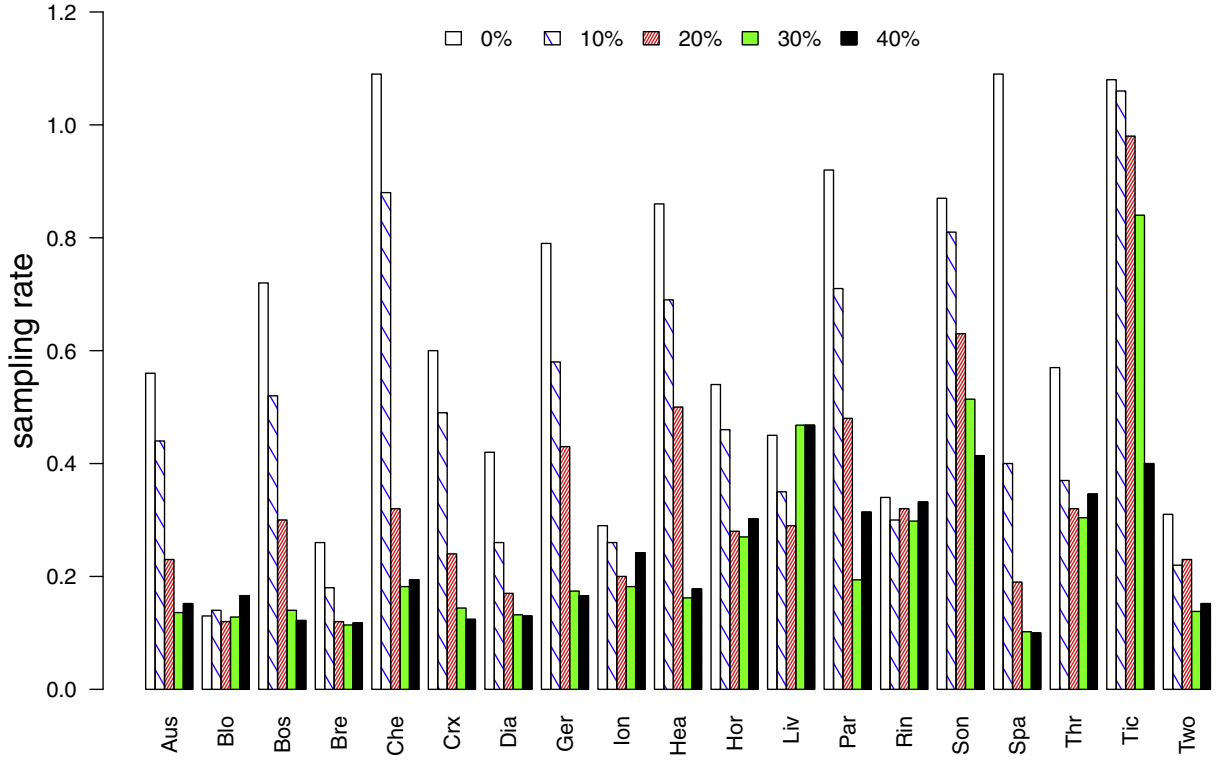


(e) 40% injected noise

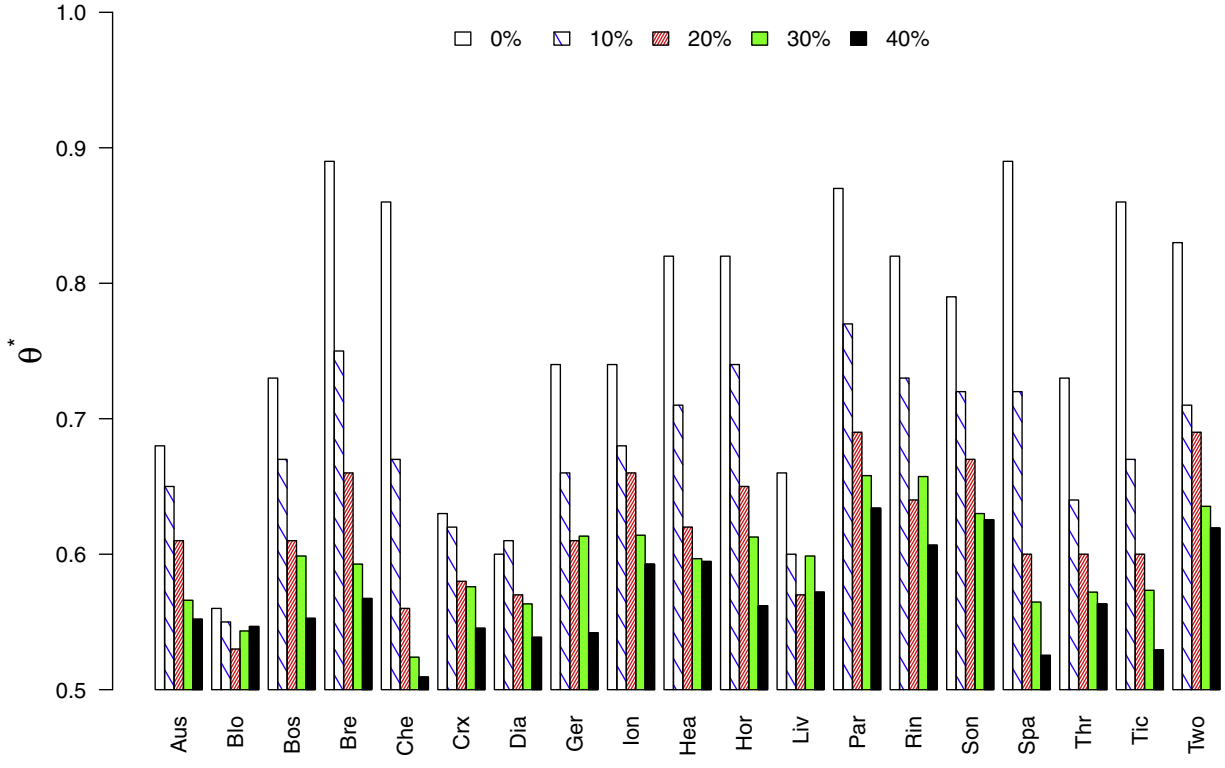
Fig. 1. Comparison of the average ranks of the different types of ensembles for different levels of class-label noise: (a) without injected noise; (b) 10%; (c) 20% ; (d) 30%; and (e) 40% noise. Horizontal lines connect methods whose average ranks are not significantly different according to a Wilcoxon signed-ranks test (p-value $< .05$).

curate than bagging ensembles, only results for the former are reported.

3. For each instance in the training set, we compute a tally of votes (class label predictions). In this tally, only the predictions of those classifiers in H_T^* whose training sets do not include that particular instance are used.
4. Instances are tentatively marked as noisy if the percentage of incorrect predictions by the individual ensemble classifiers is above a specified threshold θ. Noisy instances are either cleaned (i.e. their class labels are corrected by assigning the majority label in the out-of-bag predictions) or filtered (i.e. removed from the training set). The following values of the threshold θ are tested: 0.5 (majority filtering), 0.6, 0.7, 0.8, 0.9 and 1.0 (consensus filtering). As described in the previous section, the optimum value θ^* is selected by K-fold cross-validation within the training set by means of a wrapper method. From the results of exploratory experiments with different values of K, reliable estimates are obtained with $K = 3$, which is the value used in the experiments. Random forest composed of 501 trees is used in this wrapper stage.
5. The training instances for which the fraction of incorrect out-of-bag predictions is above θ^* are definitively marked as noise. These noisy instances are then either corrected or removed from the training set.
6. Finally, a random forests of 501 random trees, is built on the cleansed training set. The labels *Fl_rf* for random forest with filtering and *Cl_rf* for random forest with cleaning will be used throughout the experiments.



(a) Sampling rate



(b) Threshold for cleansing

Fig. 2. Optimal hyperparameters for random forest with filtering (*Fl_rf*).

Table 2
Test errors of the different methods for different noise levels (I).

Dataset	Noise (in %)	Fl_rf	Cl_rf	Fl_maj_rf	Boosting	RF
Australian	0	13.3 ± 1.9	13.5 ± 1.9	13.8 ± 2.0	**13.1 ± 1.9**	**13.1 ± 1.9***
	10	13.6 ± 1.9	13.7 ± 1.9	13.7 ± 2.0	18.2 ± 2.2	**13.5 ± 1.9**
	20	14.1 ± 1.8	14.0 ± 2.0	**13.8 ± 2.1**	23.8 ± 3.1	15.3 ± 2.2
	30	16.3 ± 2.9	16.1 ± 2.8	**15.8 ± 2.6**	33.6 ± 3.6	21.7 ± 3.0
	40	26.0 ± 6.3	**24.5 ± 5.2***	26.1 ± 5.1	41.9 ± 3.2	34.3 ± 3.5
Blood transfusion	0	21.9 ± 1.6	**21.8 ± 1.9**	22.1 ± 1.8	25.7 ± 2.3	24.7 ± 2.1
	10	22.2 ± 1.5	**22.0 ± 1.7**	22.9 ± 2.0	28.0 ± 2.8	26.4 ± 2.1
	20	23.3 ± 2.3	**23.2 ± 2.4**	24.9 ± 2.6	31.2 ± 3.2	29.8 ± 2.8
	40	34.0 ± 4.6	**32.6 ± 4.9***	38.5 ± 4.5	43.2 ± 4.3	42.1 ± 4.5
Boston	0	13.2 ± 2.2	13.5 ± 2.2	14.7 ± 2.7	**12.4 ± 2.4**	12.8 ± 2.1
	10	**13.7 ± 2.6**	13.8 ± 2.6	14.6 ± 2.2	16.3 ± 2.8	**13.7 ± 2.6**
	20	15.5 ± 2.8	15.5 ± 2.4	**15.2 ± 2.8**	23.6 ± 3.9	17.8 ± 2.7
	30	19.2 ± 3.8	**18.9 ± 3.7**	20.2 ± 4.0	32.5 ± 4.2	25.1 ± 4.4
	40	26.8 ± 6.1	**26.1 ± 6.5**	28.8 ± 6.5	41.1 ± 4.1	36.2 ± 5.1
Breast	0	**3.1 ± 1.1***	**3.1 ± 1.0***	3.3 ± 1.0	3.4 ± 1.1	**3.1 ± 1.1***
	10	3.6 ± 1.2	3.6 ± 1.1	**3.5 ± 1.2**	8.7 ± 1.8	4.1 ± 1.3
	20	**4.3 ± 1.5**	4.4 ± 1.8	**4.3 ± 1.4**	14.9 ± 2.5	6.7 ± 1.9
	30	**6.6 ± 3.4***	**6.6 ± 3.2***	7.6 ± 3.5	24.1 ± 3.8	12.7 ± 3.8
	40	15.5 ± 6.4	**14.8 ± 5.8**	18.4 ± 4.8	33.8 ± 4.2	26.4 ± 4.7
Chess	0	1.7 ± 0.4	1.7 ± 0.4	2.6 ± 0.4	**0.4 ± 0.2***	1.6 ± 0.4
	10	**2.2 ± 0.4***	2.3 ± 0.4	3.0 ± 0.6	4.1 ± 0.8	2.3 ± 0.5
	20	**3.4 ± 0.7***	3.6 ± 0.8	3.7 ± 0.8	5.8 ± 0.8	4.1 ± 0.6
	30	**5.5 ± 1.2**	6.0 ± 1.1	5.7 ± 1.0	11.0 ± 1.7	10.0 ± 1.1
	40	**16.6 ± 3.0***	17.7 ± 3.2	17.7 ± 2.3	24.6 ± 2.2	25.7 ± 1.7
Crx	0	12.9 ± 1.8	13.0 ± 2.1	13.2 ± 2.1	13.8 ± 1.8	**12.6 ± 1.9**
	10	13.7 ± 1.8	**13.6 ± 1.8**	**13.6 ± 2.0**	18.8 ± 2.5	14.0 ± 1.9
	20	14.1 ± 2.2	14.2 ± 2.3	**14.0 ± 2.2**	25.3 ± 3.1	16.2 ± 2.1
	30	16.9 ± 3.3	**16.2 ± 2.9**	17.0 ± 3.0	33.4 ± 3.8	22.3 ± 3.3
	40	24.5 ± 5.9	**24.3 ± 5.9**	25.8 ± 5.0	41.5 ± 3.4	33.8 ± 3.7
Diabetes	0	24.1 ± 2.0	24.3 ± 1.8	24.3 ± 1.9	27.5 ± 2.1	**24.0 ± 1.9**
	10	24.6 ± 2.3	24.5 ± 2.3	**24.3 ± 2.2**	30.1 ± 2.9	25.4 ± 2.2
	20	25.4 ± 2.5	25.4 ± 2.4	**25.3 ± 2.4**	34.4 ± 3.0	27.8 ± 2.5
	30	27.3 ± 2.6	**26.7 ± 2.7***	27.3 ± 2.7	39.6 ± 3.3	31.7 ± 3.0
	40	**32.5 ± 4.9**	**32.5 ± 3.8**	33.4 ± 4.7	44.4 ± 4.0	39.3 ± 4.5

* p-value < 0.05.

7. The generalization error of the resulting ensembles is estimated on the unperturbed test set.

As a benchmark, a second *cleansed* dataset is obtained using majority filtering following the proposals of previous studies [3,17,21], and, in particular, for homogeneous ensembles [16]. In this benchmark, the data are filtered using K-fold cross-validation. At each iteration, an ensemble is trained using data from $K-1$ folds. Then, the ensemble is used to predict the labels of the instances in the remaining fold. The examples of the holdout fold that are incorrectly classified are removed from the dataset. The process is repeated to filter the other folds. We implemented this method using random forest of size 501 and $K = 3$. Finally, a random forest composed of 501 random trees is built on the training set cleaned with majority filtering. The label *Fl_maj_rf* is used to denote this benchmark. As an additional reference, we also present the results of standard random forest (labelled as RF) and AdaBoost (labeled as Boosting) trained on the original uncleaned training set.

4.1. Predictive accuracy

The results of the experiments carried out to compare the accuracies of the different methods considered are summarized in Tables 2–4. The test errors reported correspond to averages over 50 realizations of the training and test sets. These averages followed by their standard deviations after the ± sign. To assess the significance of these observations, the results of an overall comparison of the different methods are summarized in Fig. 1 using the methodology introduced in [4]. In these plots, the average ranks of the different methods are compared. The differences between the average ranks of two methods are statistically significant at a level $\alpha = 0.05$ if they are above a critical distance (CD). Methods whose average ranks are not significantly different are linked by a horizontal line. Average rank plots are presented for experiments with different levels of class-label noise injected: 0, 10, 20, 30 and 40%.

From the results presented in Fig. 1, Tables 2–4, one concludes that random forests with optimal filtering or cleaning (*Fl_rf* and *Cl_rf*) are among the most accurate ensembles at all noise levels. When no noise is injected, boosting and random forest trained on the original data are slightly better than *Fl_rf* and *Cl_rf*. However, the differences are not statistically significant. Because of its progressive emphasis on incorrectly classified instances, boosting is not robust to errors in the class labels. This is apparent from the degradation of its performance in problems with higher levels of noise. In fact, boosting has the worst average rank for 10–40% noise levels. In these cases, the differences with most other methods are statistically significant. The differences between filtering and cleaning are not statistically significant. Filtering seems to have a slightly better performance at lower noise levels. When either 30% or 40% of the class labels of the training instances are perturbed, cleaning is slightly better than filtering. This is probably a consequence of the loss of information that results form discarding instances in filtering. Finally, we observe that selecting adequate values of the threshold θ for noise filtering or cleaning is superior to using standard majority filtering at all noise levels. These improvements are more pronounced at lower noise levels.

Table 3
Test errors of the different methods for different noise levels (II).

Dataset	Noise (in %)	*Fl_rf*	*Cl_rf*	*Fl_maj_rf*	Boosting	RF
German	0	24.5 ± 2.2	24.6 ± 1.9	26.5 ± 2.1	25.1 ± 2.1	**24.2 ± 2.0**
	10	25.0 ± 1.9	25.6 ± 2.4	26.7 ± 2.4	28.1 ± 2.2	**24.8 ± 1.9**
	20	**26.5 ± 2.7**	26.7 ± 2.3	27.1 ± 2.2	32.7 ± 3.1	27.1 ± 2.7
	30	28.9 ± 2.7	28.6 ± 2.8	**28.3 ± 2.6**	38.3 ± 3.0	31.0 ± 2.7
	40	32.2 ± 4.2	**31.7 ± 3.9**	32.8 ± 4.0	43.3 ± 2.9	37.9 ± 3.6
Ionosphere	0	6.9 ± 2.0	6.9 ± 2.0	7.3 ± 1.9	**6.4 ± 1.8**	6.8 ± 1.9
	10	**7.7 ± 2.0**	7.8 ± 2.2	8.0 ± 2.4	11.3 ± 3.3	7.8 ± 2.1
	20	9.4 ± 3.1	9.5 ± 3.7	**9.3 ± 3.0**	19.0 ± 4.2	11.3 ± 3.7
	30	14.8 ± 4.5	14.1 ± 4.0	**13.8 ± 4.6**	28.0 ± 5.3	18.4 ± 3.8
	40	32.2 ± 4.2	**31.7 ± 3.9**	32.8 ± 4.0	43.3 ± 2.9	37.9 ± 3.6
Heart	0	**17.5 ± 3.4**	18.1 ± 3.8	17.9 ± 3.8	20.9 ± 3.4	18.1 ± 3.6
	10	19.6 ± 4.3	19.7 ± 4.3	**18.9 ± 4.8**	26.3 ± 4.1	20.3 ± 3.8
	20	21.7 ± 4.0	**21.4 ± 4.5**	21.5 ± 4.2	32.1 ± 5.8	23.5 ± 4.3
	30	24.2 ± 5.6	24.2 ± 5.5	**24.0 ± 5.3**	37.1 ± 5.3	28.7 ± 5.7
	40	34.2 ± 7.6	**33.7 ± 7.2**	34.5 ± 7.3	43.2 ± 6.6	38.4 ± 6.2
Horse-colic	0	15.6 ± 2.5	**15.5 ± 2.7**	17.6 ± 3.6	15.7 ± 2.6	**15.5 ± 2.7**
	10	**17.5 ± 3.0**	**17.5 ± 3.1**	18.2 ± 3.4	22.3 ± 3.4	**17.5 ± 2.9**
	20	**20.2 ± 3.6**	20.5 ± 4.1	20.3 ± 4.0	28.3 ± 4.9	20.7 ± 3.7
	30	26.2 ± 4.7	26.5 ± 4.9	**26.0 ± 5.5**	35.4 ± 4.5	29.0 ± 4.9
	40	36.0 ± 6.3	**35.2 ± 6.5**	36.0 ± 6.5	43.6 ± 4.7	39.7 ± 4.5
Liver	0	28.4 ± 4.2	29.7 ± 3.8	31.2 ± 4.4	29.6 ± 3.4	**27.4 ± 3.8***
	10	**31.4 ± 4.3***	32.7 ± 4.9	33.6 ± 4.8	34.4 ± 3.9	**31.4 ± 4.4***
	20	**34.5 ± 4.6**	35.5 ± 4.2	36.0 ± 4.7	37.7 ± 4.6	34.6 ± 4.5
	30	38.0 ± 5.9	38.5 ± 5.3	38.8 ± 5.7	40.5 ± 5.5	**37.7 ± 5.8**
	40	43.8 ± 5.6	**43.1 ± 5.4**	44.6 ± 4.6	44.9 ± 4.6	43.5 ± 5.3
Parkinsons	0	11.9 ± 3.5	11.4 ± 3.8	16.3 ± 4.0	**8.1 ± 3.5***	11.0 ± 3.5
	10	13.5 ± 3.8	13.5 ± 3.8	16.4 ± 4.0	**12.8 ± 3.9**	13.0 ± 3.7
	20	**17.8 ± 4.7**	18.1 ± 5.2	18.1 ± 4.5	22.0 ± 5.0	**17.8 ± 4.7**
	30	21.0 ± 5.2	21.5 ± 4.7	**20.7 ± 5.4**	29.8 ± 7.1	23.9 ± 5.5
	40	31.8 ± 8.2	31.8 ± 8.0	**29.3 ± 7.5***	39.0 ± 6.2	34.6 ± 7.1
Ringnorm	0	6.1 ± 1.0	6.2 ± 1.0	8.3 ± 1.4	**4.4 ± 0.4***	6.0 ± 1.0
	10	6.8 ± 1.3	6.9 ± 1.1	8.7 ± 1.5	8.6 ± 1.1	**6.7 ± 1.1**
	20	8.4 ± 1.8	**8.3 ± 1.6**	9.9 ± 2.2	15.2 ± 1.7	**8.3 ± 1.7**
	30	11.8 ± 2.7	**11.5 ± 3.1**	12.2 ± 3.2	24.4 ± 2.7	12.8 ± 2.4
	40	21.1 ± 5.3	**18.2 ± 5.3***	20.8 ± 6.4	36.3 ± 3.1	24.7 ± 3.5

* p-value < 0.05.

4.2. Optimal values of the hyperparameters

In this section, we analyze the trends in the values selected for the sampling rate and the threshold, θ^*. We consider first the values of the sampling rate. In plot (a) of Fig. 2 the average sampling rates obtained by the proposed cleaning procedure for each dataset and noise level are displayed. One can observe that in the original unperturbed problems (white bars in the plot) the optimal values for this hyperparameter are strongly problem dependent. For instance, in *Chess, Spambase*, and *Tic-tac-toe* these values are above the standard 100% resampling rate. It is likely that for these problems the variability random forests is too large, and that oversampling is an effective mechanism to reduce it. In the remaining problems subsampling, which increases the variability of the ensemble, seems to be more effective. In some cases, such as *Blood Transfusion, Breast, Ionosphere*, and *Twonorm*, the optimal values of the sampling ratios are fairly low (around or below 30%, on average). In general, as more class-label noise is injected in the data, the optimal sampling ratios become smaller. This confirms the observation that subsampling becomes more effective as the amount of class-label noise increases [13,14].

In the method proposed, an instance is marked as noisy when it is incorrectly labeled by a fraction of classifiers that exceeds a specified threshold θ^*. The optimal value of this parameter is also strongly problem-dependent. However, from the results presented in plot (b) of Fig. 2, it is apparent that as more noise is injected, the values of θ^* become smaller and approach 0.5, which corresponds to majority voting. This is consistent with the observation that majority filtering becomes more effective at higher noise levels.

4.3. Noise detection

The value of the threshold used to mark an instance as noisy is determined on the basis of the accuracy of the wrapper classifier trained on cleansed data. The question remains whether this procedure is actually effective for the detection of noisy instances. To investigate this issue, we have recorded the fraction of instances marked as noise for the different classification tasks and with different levels of injected noise. The results of these experiments are presented in the plots of Fig. 3 for the following noise levels: 0% (first row), 10% (second row) and 30% (third row). Each plot of this figure shows the average percentage of instances that are marked as noise with a white bar. From those instances, the ones corresponding to the artificially injected noise are marked in red. The results for the proposed cleaning procedure using filtering are summarized in the plots in the left column of this figure. For reference, the results of majority filtering benchmark are displayed in the plots in the right column.

An analysis of the results for the unperturbed classification tasks (first row of Fig. 3) reveals that the levels of detected noise vary significantly. In the synthetic problems, which, by construction, are noiseless, the proposed cleaning procedure using filtering discards only a small percentage instances. By contrast, in some problems (such as *Blood transfusion, Diabetes, German* and *Liver*) a significant fraction of instances are identified as noise. Filtering these instances does not appear to be detrimental. As a matter

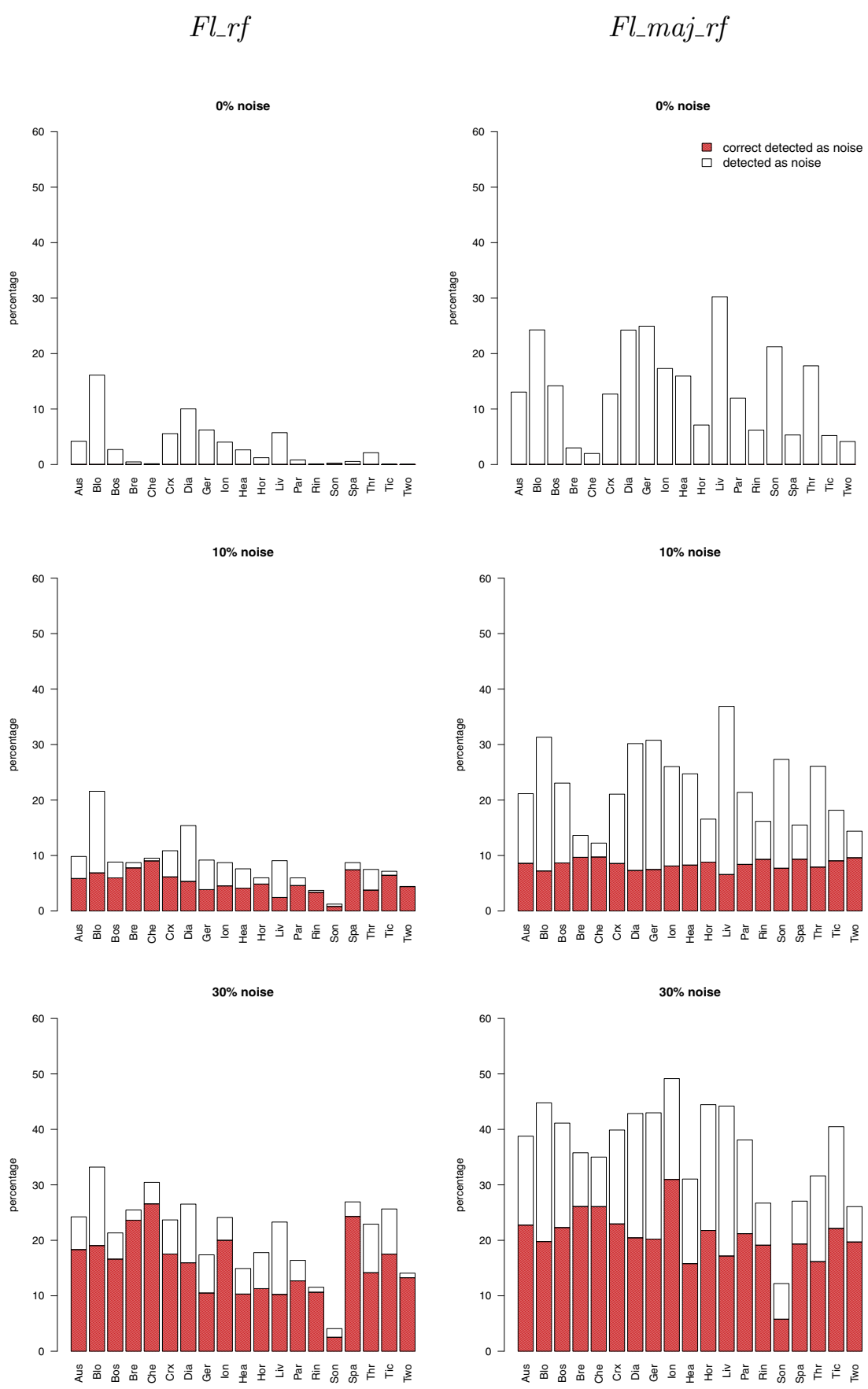


Fig. 3. Percentage of filtered examples (white bars) and filtered examples that correspond injected noise (red part of the bars) for different noise levels: without injected noise (first row), 10% (second row), and 30% (third row) for the proposed cleansing procedure (left column) and majority filtering (right column). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4
Test errors of the different methods for different noise levels (III).

Dataset	Noise (in %)	*Fl_rf*	*Cl_rf*	*Fl_maj_rf*	Boosting	RF
Sonar	0	18.8 ± 3.9	19.4 ± 4.8	24.5 ± 3.8	**14.4 ± 3.8***	18.5 ± 3.7
	10	20.7 ± 5.2	21.5 ± 4.9	25.2 ± 4.6	20.1 ± 4.4	**19.9 ± 4.7**
	20	24.2 ± 4.8	24.3 ± 5.4	26.1 ± 5.2	26.9 ± 5.5	**23.7 ± 5.5**
	30	31.7 ± 7.1	31.9 ± 7.2	33.1 ± 7.7	35.1 ± 6.6	**31.2 ± 7.6**
	40	**38.6 ± 6.2**	39.7 ± 7.5	39.6 ± 7.6	42.6 ± 7.7	39.4 ± 7.4
Spambase	0	5.1 ± 0.6	5.1 ± 0.6	6.3 ± 0.5	**4.9 ± 0.4**	5.0 ± 0.5
	10	**5.9 ± 0.5***	6.1 ± 0.7	6.5 ± 0.5	9.4 ± 0.8	6.6 ± 0.6
	20	**6.7 ± 0.6**	**6.7 ± 0.6**	6.9 ± 0.7	13.7 ± 1.1	9.3 ± 0.9
	30	**7.8 ± 0.8***	**7.8 ± 0.8***	8.6 ± 0.9	20.3 ± 1.4	14.6 ± 1.1
	40	11.8 ± 2.2	**11.1 ± 1.5***	14.3 ± 1.5	32.0 ± 1.6	25.6 ± 1.3
Threenorm	0	17.0 ± 1.0	17.2 ± 1.3	19.2 ± 1.4	16.8 ± 0.9	**16.5 ± 0.9**
	10	18.8 ± 1.5	19.1 ± 1.6	20.5 ± 1.8	20.7 ± 1.4	**18.4 ± 1.2***
	20	20.9 ± 2.2	21.2 ± 2.6	21.7 ± 2.5	25.5 ± 1.6	**20.8 ± 1.8**
	30	**25.6 ± 2.6**	25.7 ± 2.5	26.7 ± 2.5	32.3 ± 2.1	26.5 ± 2.1
	40	33.4 ± 4.5	**33.0 ± 4.7**	34.3 ± 5.1	40.6 ± 2.9	35.7 ± 3.1
Tictactoe	0	2.5 ± 0.9	2.4 ± 1.1	7.2 ± 2.5	**0.6 ± 0.5***	2.3 ± 1.0
	10	**5.7 ± 2.2**	6.2 ± 2.3	12.0 ± 2.6	10.4 ± 1.7	5.8 ± 1.6
	20	12.5 ± 2.9	13.8 ± 2.8	17.4 ± 2.8	21.3 ± 2.3	**12.4 ± 2.3**
	30	22.1 ± 2.7	23.5 ± 2.6	24.1 ± 3.3	30.5 ± 2.6	**21.7 ± 2.2**
	40	**34.4 ± 3.7**	**34.4 ± 3.5**	35.1 ± 3.5	40.5 ± 3.7	35.7 ± 3.2
Twonorm	0	3.9 ± 0.5	3.9 ± 0.5	4.3 ± 0.5	**3.7 ± 0.3**	3.8 ± 0.4
	10	4.6 ± 0.7	**4.5 ± 0.6**	**4.5 ± 0.5**	7.7 ± 1.0	4.9 ± 0.7
	20	5.6 ± 1.2	5.5 ± 0.9	**5.2 ± 0.9**	13.7 ± 1.6	6.5 ± 1.0
	30	8.2 ± 2.9	7.7 ± 2.8	**7.3 ± 2.3**	23.7 ± 2.4	10.8 ± 1.8
	40	17.5 ± 6.3	16.7 ± 6.9	**15.2 ± 5.5**	35.7 ± 3.2	23.2 ± 3.3

* p-value < 0.05.

of fact, in these problems the proposed cleaning procedure yields competitive or better accuracy rates with respect to random forest trained on the uncleaned data (see Tables 2 and 3).

For the unperturbed classification tasks, majority filtering is much more aggressive and marks many more instances as noise than the proposed procedure. In problems without noise in the class labels, such as *Threenorn, Tic-tac-toe, Ringnorm, Twonorm*, around or above 5% of the training instances are discarded (see upper right plot on Fig. 3). As a result, there is a significant decrease of the accuracy for random forest trained on these cleansed data (see Tables 3 and 4). In real-world problems, it is not possible to know the level of intrinsic noise. Nonetheless, majority filtering is likely to discard too many instances as well. Specifically, this method marks more than 20% of the instances as noise in five of the datasets analyzed (*Blood transfusion, Diabetes, German, Liver* and *Sonar*). The accuracy of random forest trained on the data cleansed by majority filtering in *Blood transfusion* and in *Diabetes* is fairly good. However, in *Liver* and *Sonar* it is the least accurate among the methods considered.

The results of experiments on classification tasks perturbed with 10% and 30% class-label noise, are displayed in the second and the third rows of Fig. 3, respectively. The red part of the bars is the percentage of instances whose class-label has been switched in the noise injection process that are identified as noisy. In most cases for the proposed cleansing procedure based on optimal filtering, the height of the red bar is well below the level of noise injected. This means that a significant fraction of perturbed instances are not detected. An extreme example is *Sonar*. We conjecture that this lack of sensitivity is due to the strong overlap of the distributions for the two classes. For this reason, it is difficult to single out noisy instances located in regions where such overlap is high. Using majority filtering, which is more aggressive, it is possible to detect most of the injected noisy instances. In fact, the relative performance of majority filtering improves when the levels of class-label noise are high. Still, even at high noise levels, the precision of the method is rather low: the percentage of instances that are marked as noise by majority voting is significantly larger than the level of noise injected. By contrast, even though the proposed optimal filtering procedure fails to identify some noisy instances, those that are identified by this strategy are more likely to be noise in actuality. In some datasets (*Boston, Breast, Chess, Parkinsons, Ringnorm, Spambase, Tictactoe* and *Twonorm*) the proposed procedure detects a fairly high percentages of the injected noise without removing a significant number of noiseless instances. For these datasets, random forests trained on data cleaned with the proposed procedure are more accurate than those trained based on data cleaned majority filtering, except in *Breast* and *Twonorm*, where the differences are not statistically significant (see Tables 2–4).

In summary, the proposed cleansing procedure achieves high specificity at the expense of not being able to detect some noisy instances. By contrast, majority filtering detects most noisy instances, but also incorrectly discards a high percentage of valid ones. As shown in Fig. 2 (b), θ^*, the optimal threshold for filtering, becomes closer to 0.5 (majority filtering) as the level of class-label noise increases.

5. Conclusions

In this paper, we have proposed a two-stage method for the detection of class-label noise based on the robustness to noise of randomized ensembles that use resampling [13,14]. Near-optimal values of the sampling rate can be determined using out-of-bag data. Typically, the selected sampling ratios become smaller as the level of class label noise increases. Another important observation is that the classifiers in the ensemble tend to make more errors on noisy instances. Therefore, the fraction of incorrect predictions can be used as an indicator for noise detection: if this quantity is above a threshold, θ, the instance considered is marked as noisy. Standard values for the threshold are $\theta = 0.5$ (majority) or $\theta = 1$ (consensus). In this work, we have shown that the best results are obtained at a value θ^* that is intermediate between these extremes. A simple wrapper procedure is proposed to determine θ^*. This optimal value depends on the problem under consideration and the

amount of class-label noise. Values of θ^* closer to majority filtering are generally obtained for noisy problems. However, majority cleansing tends to discard instances that are correctly labeled. This has been shown to be disadvantageous, specially in problems with low levels of noise. In general, adjusting the threshold used to detect noisy instances allows us to build more accurate ensembles at all levels of class-label noise.

Once the noisy instances have been identified, they can be removed from the training data (filtering) or corrected (cleaning). Filtering is slightly superior at low and medium noise levels. Cleaning tends to be more accurate when the noise levels are high. The reason for this behavior is that filtering discards training instances. This involves some information loss, which could be detrimental if too many instances are discarded.

Acknowledgments

The authors acknowledge financial support from the *Spanish Ministry of Economy, Industry and Competitiveness*, projects TIN2013-42351-P, TIN2016-76406-P, and TIN2015-70308-REDT, and of the *Comunidad de Madrid*, project CASI-CAM-CM (S2013/ICE-2845).

References

[1] K.M. Ali, M.J. Pazzani, Error reduction through learning multiple descriptions, Mach. Learn. 24 (3) (1994) 173–202.
[2] J. Bi, T. Zhang, Support vector classification with input data uncertainty, in: L.K. Saul, Y. Weiss, L. Bottou (Eds.), Advances in Neural Information Processing Systems 17, MIT Press, 2005, pp. 161–168.
[3] C.E. Brodley, M.A. Friedl, Identifying mislabeled training data, J. Artif. Intell. Res. 11 (1) (1999) 131–167.
[4] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[5] B. Frénay, M. Verleysen, Classification in the presence of label noise: a survey, IEEE Trans. Neural Netw. Learn. Syst. 25 (5) (2014) 845–869.
[6] Y. Freund, An adaptive version of the boost by majority algorithm, Mach. Learn. 43 (3) (2001) 293–318.
[7] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: Proceedings of the Thirteenth International Conference on Machine Learning (ICML), 1996, pp. 148–156.
[8] D. García-Gil, J. Luengo, S. García, F. Herrera, Enabling smart data: noise filtering in big data classification, Comput. Res. Repos. (2017). http://arxiv.org/abs/1704.01770.
[9] T.M. Khoshgoftaar, S. Zhong, V. Joshi, Enhancing software quality estimation using ensemble-classifier based noise filtering, Intell. Data Anal. 9 (1) (2005) 3–27.
[10] A. Liaw, M. Wiener, Classification and regression by randomforest, R News 2 (3) (2002) 18–22. http://CRAN.R-project.org/doc/Rnews/.
[11] M. Lichman, UCI machine learning repository, 2013. http://archive.ics.uci.edu/ml.
[12] G. Martnez-Muoz, A. Surez, Switching class labels to generate classification ensembles, Pattern Recognit. 38 (10) (2005) 1483–1494.
[13] M. Sabzevari, G. Martínez-Muñoz, A. Suárez, Improving the robustness of bagging with reduced sampling size, in: Proceedings of the Twenty Second European Symposium on Artificial Neural Networks, ESANN, Bruges, Belgium, 2014. April 23–25.
[14] M. Sabzevari, G. Martnez-Muoz, A. Su rez, Small margin ensembles can be robust to class-label noise, Neurocomputing 160 (Supplement C) (2015) 18–33.
[15] B. Sluban, D. Gamberger, N. Lavrač, Ensemble-based noise detection: noise ranking and visual performance evaluation, Data Mining Knowl. Discov. 28 (2) (2014) 265–303.
[16] B. Sluban, N. Lavra, Relating ensemble diversity and performance: a study in class noise detection, Neurocomputing 160 (Supplement C) (2015) 120–131.
[17] S. Verbaeten, A. Van Assche, Ensemble Methods for Noise Elimination in Classification Problems, Springer, Berlin, Heidelberg, 2003, pp. 317–325.
[18] D. Wang, M. Li, Robust stochastic configuration networks with kernel density estimation for uncertain data regression, Inf. Sci. 412–413 (Supplement C) (2017) 210–222.
[19] S. Zhong, W. Tang, T.M. Khoshgoftaar, Boosted noise filters for identifying mislabeled data, Technical report, Department of Computer Science and engineering, Florida Atlantic University, 2005.
[20] X. Zhu, X. Wu, Class noise vs. attribute noise: a quantitative study, Artif. Intell. Rev. 22 (3) (2004) 177–210.
[21] X. Zhu, X. Wu, Q. Chen, Eliminating class noise in large datasets, in: Proceedings of the Twentieth International Conference on Machine Learning (ICML), 2003, pp. 920–927.

Maryam Sabzevari received her B.Sc (2008) degree in Computer Science and M.Sc (2010) in Artificial Intelligence from Azad University of Mashhad, Iran. Later, she received another M.Sc (2015) in Neuroinformatics from Universidad Autónoma de Madrid (UAM), Spain. Currently, she is conducting her Ph.D. in Computer Science in Universidad Autónoma de Madrid (Spain), where she is also a Teaching Assistant. Her current research interests include machine learning, pattern recognition, neural networks, ensemble learning and learning methods in the presence of noise.

Gonzalo Martínez-Muñoz received the university degree in Physics (1995) and Ph.D. degree in Computer Science (2006) from the Universidad Autónoma de Madrid (UAM). From 1996 to 2002 he worked in industry. Until 2008 he was an interim assistant professor in the Computer Science Department of the UAM. During 2008/2009, he worked as a Fulbright postdoc researcher at Oregon State University in the group of Professor Thomas G. Dietterich. He is currently a professor at Computer Science Department at UAM. His research interests include machine learning, computer vision, pattern recognition, neural networks, decision trees, and ensemble learning.

Alberto Suárez received the degree of Licenciado in Chemistry from the Universidad Autónoma de Madrid, Spain, in 1988, and the Ph.D. degree in Physical Chemistry from the Massachusetts Institute of Technology (MIT), Cambridge, MA, in 1993. After holding postdoctoral positions at Stanford University (USA), at the Université Libre de Bruxelles (Belgium), as a research fellow financed by the European Commission within the program "Training and Mobility of Researchers", and at the Katholieke Universiteit Leuven (Belgium), he is currently a professor in the Computer Science Department of the Universidad Autónoma de Madrid (Spain). He has also held appointments as "Senior Visiting Scientist" at the International Computer Science Institute (Berkeley, CA) and at MIT (Cambridge, MA). He has worked on relaxation theory in condensed media, stochastic and thermodynamic theories of nonequilibrium systems, latticegas automata, and automatic induction from data. His current research interests include machine learning, computational statistics, quantitative finance,time series analysis and information processing in the presence of noise. He is a member of IEEE.