Eliminating Class Noise in Large Datasets

Xingquan Zhu XQZHU@CS.UVM.EDU
Xindong Wu XWU@CS.UVM.EDU
Qijun Chen QCHEN@CS.UVM.EDU
Department of Computer Science, University of Vermont, Burlington, VT 05405, USA

Abstract

This paper presents a new approach for identifying and eliminating mislabeled instances in large or distributed datasets. We first partition a dataset into subsets, each of which is small enough to be processed by an induction algorithm at one time. We construct good rules from each subset, and use the good rules to evaluate the whole dataset. For a given instance I_k, two error count variables are used to count the number of times it has been identified as noise by all subsets. The instance with higher error values will have a higher probability of being a mislabeled example. Two threshold schemes, majority and non-objection, are used to identify the noise. Experimental results and comparative studies from real-world datasets are reported to evaluate the effectiveness and efficiency of the proposed approach.

1. Introduction

The goal of an inductive learning algorithm is to form a generalization from a set of training instances such that its classification accuracy on previously unobserved instances is maximized. This maximum accuracy is determined by two most important factors: (1) the quality of the training data; and (2) the inductive bias of the learning algorithm. Given a learning algorithm, it's obvious that its classification accuracy depends vitally on the quality of the training data. Accordingly, the most feasible and direct way to improve the system effectiveness is to clean up the noise from the training data. Generally, there are two types of noise sources (Wu, 1995): (a) attribute noise; and (b) class noise. The former is the errors that are introduced in the attribute values of the instances. There are two possible sources for class noise: (1) contradictory examples, i.e., the same examples with different class labels; and (2) misclassifications: instances labeled with wrong classes. Due to the fact that class noise is caused by mislabeling, in this paper, we call it mislabeled errors, and will identify this type of noise.

Quinlan (1986) demonstrated that, for higher levels of noise, removing noise from attribute information decreases the predictive accuracy of the resulting classifier if the same attribute noise is present when the classifier is subsequently used. However, for class noise, the opposite is true: cleaning the training data will result in a classifier with a higher predictive accuracy. The use of pruning and learning ensembles partially addresses the problem, but noise can still drastically affect the accuracy. Hence, many research efforts have been made on eliminating mislabeled errors for effective learning. Generally, the most challenging task in identifying class noise is how to distinguish the mislabeled errors from the exceptions to general rules (Srinivasan *et al*. 1992). When an instance is an exception to general cases, it can also appear as if it is incorrectly labeled. Guyon *et al.* (1996) provided an approach that uses an information criterion to measure an instance's typicality; and atypical instances are then presented to a human expert to determine whether they are mislabeled errors or exceptions. The noise detection algorithm of Gamberger *et al.* (2000) is based on the observation that the elimination of noisy examples reduces the *CLCH* (Complexity of the Least Complex correct Hypothesis) value of the training set. They called their noise elimination algorithm the Saturation filter since it employs the *CLCH* measure to test whether the training set is saturated. Brodley & Friedl (1996; 1999) simplified noise elimination as a filtering operation (John, 1995) where multiple classifiers learned from a noise corrupted dataset are used to identify noise, and the noise is characterized as the instances that are incorrectly classified by the multiple classifiers. Similar schemes have been widely adopted by others (Gamberger *et al.*, 1999; Verbaeten, 2002). Instead of using multiple classifiers learned from the same training set for noise identification, Gamberger *et al*. (1999) suggested a Classification Filter approach, in which the training set E is partitioned into n subsets, a set of classifiers H_y trained from the aggregation of any n-1 subsets are used to classify the instances in the complementary (excluded) subset, and the instances that are incorrectly classified by H_y are identified as noise.

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

To perform noise elimination, most approaches above make two assumptions: (1) the dataset is relatively small for learning at one time; and (2) the dataset is right at hand for learning. This is because that most of them adopt a major set based strategy: Given a dataset E, it is separated into two parts: a major set and a minor set, and the major set is used to induce classifiers to identify the noise from the minor set. Nevertheless, the machine learning community is currently facing the challenge of large and distributed datasets, which might be too large to be handled at one time. Hence, the above assumptions are too strong in realistic situations. One may argue that in the case of a large dataset, scaling-up inductive algorithms (Provost *et al.*, 1999), e.g., boosting and meta-learning, could be adopted. Unfortunately, Chan's (1996) research has shown that in general the classification accuracy from scaling-up algorithms is worse than the accuracy from the whole dataset, especially in noisy environments. This is because that scaling-up algorithms induce rules from sampled data where many exceptions in the sampled data are actually valuable instances in the original dataset. Consequently, while dealing with large datasets, most noise elimination approaches are inadequate.

For distributed datasets, the problem with existing approaches is even more severe. They assume that all data is at hand for processing. Unfortunately, in realistic situations, it's either technically infeasible (e.g., bandwidth limitations) or forbidden (e.g., for security or privacy reasons) to share or download data from other sites. One can execute noise elimination algorithms on each single site respectively, but it may inevitably eliminate some instances that are noise for the current site but useful for other sites.

In this paper, we present a new strategy to eliminate mislabeled instances from large datasets, and we also analyze its availability for distributed datasets. Because it's a partition-based scheme, we call it Partitioning Filter (PF). Our experimental results on real-world datasets will demonstrate the effectiveness of our approach: with datasets from the UCI repository (Blake and Merz, 1998), at any noise level (even 40%), the classifier learned from the dataset that has been processed by PF always shows remarkably improved classification accuracy.

2. Noise Elimination from Large Datasets

The flowchart of our proposed scheme (Partitioning Filter) is depicted in Figure 1, and the various procedures in Figure 1 are given in Figures 2 and 3. The intuitive assumption behind our approach in distinguishing exceptions and noise is that once we can select good rules from induced classifiers, the behaviors of the exceptions and noise would be different with the good rules: exceptions are usually not covered by the selected good rules, whereas noisy instances are likely covered by the good rules but produce a wrong classification. Based on this assumption, we first partition the whole dataset E into several subsets. Given any subset P_i, we learn a set of classification rules R_i, then select a good rule set (GR_i) from R_i, and use the GR_i to evaluate the whole dataset E. Given an instance I_k in E, we use two error count variables, I_k^{le} (local error count) and I_k^{ge} (global error count), to record the behavior of I_k with GR_i from P_i:

(1) If I_k belongs to P_i and I_k's classification from GR_i is different from its original label, we increase its local error count (I_k^{le}) by 1;

(2) If I_k does not belong to P_i and I_k's classification from GR_i is different from its original label, we increase its global error count (I_k^{ge}) by 1;

(3) Otherwise, I_k^{le} and I_k^{ge} remain the same values.

After we execute the same procedure on all subsets, the variables I_k^{le} and I_k^{ge} of I_k will indicate the possibility that I_k is mislabeled. As we stated above, due to the different behaviors of exceptions and noise with good rules, mislabeled instances will hopefully receive large values on I_k^{le} and I_k^{ge} than exceptions. Then two noise identification schemes, majority and non-objection, are adopted, as shown in Figure 3. In the majority threshold scheme, an instance is identified as noise only if more than one half of the subsets identify it as noise. In the non-objection scheme, the instance won't be judged as noise until its classification results from all subsets are different from its original label. For both schemes, one premise to identify I_k as a mislabeled error is that I_k should be classified as noise by its host subset (subset P_i to which I_k belongs), as shown in Step (1) of Figure 3, because a classifier usually has a higher accuracy with the instances in its training set.

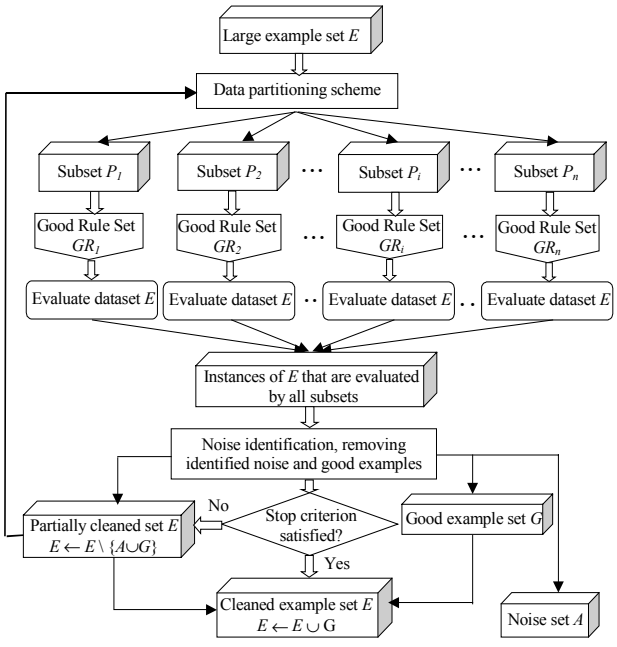

Figure 1. Noise elimination from large datasets

Procedure: ***PartitioningFilter*** ()

Input: E (training set with S examples)

Parameters: (1) The scheme and threshold to select good rules (default: Best-L rules); (2) β, the rate of good examples to be removed in each round (default: $\beta = 0.5$).

Output: A (detected noise subset of E)

(1) $B \leftarrow \emptyset$, for any instance $I_k \in E$, $S_k^{le} = S_k^{ge} = S_k^{lused} = 0$;
(2) Partition E into N subsets.
(3) For $i=1, \ldots, N$ do
(4) For each subset P_i, learn a rule set R_i from P_i.
(5) GoodRulesSelection (GR_i, R_i). // See Section 2.1
(6) For k=1,.., S do
(7) Given instance I_k, $I_k \in E$, if $I_k \in P_i$
(8) S_k^{lused} ++;
(9) If $I_k \in P_i$, I_k fires GR_i and its classification from GR_i is different from its class label,
(10) S_k^{le} ++;
(11) Else if $I_k \notin P_i$
(12) If I_k fires GR_i and its classification from GR_i is different from its class label,
(13) S_k^{ge} ++;
(14) End for
(15) End for
(16) For $k=1$,.., S do
(17) Given instance I_k, if Noise (S_k^{lused}, S_k^{le}, S_k^{ge}, N) = 1
(18) $B \leftarrow B \cup \{I_k\}$
(19) End for
(20) Remove identified noise and a certain portion ($\beta \cdot \|B\|$) of good examples (G).
(21) $E \leftarrow E \setminus \{B \cup G\}$; $A \leftarrow A \cup B$;
(22) Exit if the stopping criterion has been satisfied.
(23) Otherwise: Goto Step (1); // Repeat the procedure

Figure 2. Partitioning Filter

Procedure: **Noise** (S_k^{lused}, S_k^{le}, S_k^{ge}, n)

(1) If ($S_k^{lused} \geq 1$ and $S_k^{le} = S_k^{lused}$)
(2) If Majority Scheme:
(3) If ($S_k^{ge} \geq [(n - S_k^{lused})/2]$), Return (1);
(4) Else if Non-objection Scheme:
(5) If ($S_k^{ge} \geq (n - S_k^{lused})$), Return (1);
(6) Else Return (0);

Figure 3. Noise identification (with Majority and Non-objection schemes)

2.1 Good Rule Selection

To select good rules from each subset, a consistency check should be adopted. For each rule r learned from subset P_i, we compute two factors, $SubPrec(r, P_i)$ and $SubCov(r, P_i)$ which indicate the classification precision and coverage of r with subset P_i. The criteria we have adopted for selecting good rules consist of two parts:

(1) The estimated $SubPrec(r, P_i)$ has to be significantly above the given threshold. This is ensured by Eq.(1).

$$SubPrec(r, P_i) - SE\{SubPrec(r, P_i)\} > \alpha \quad (1)$$

Where α is the threshold to select good rules, $SE\{p\} = \sqrt{p(1-p)/n}$ is the standard error of classification (Breiman *et al*., 1984).

(2) A rule that covers very few examples cannot be selected to evaluate other subsets. This is ensured by Eq.(2).

$$SubCov(r, P_i) > \mu. \quad (2)$$

With the two criteria above, the parameter α plays an important role in selecting good rules for each subset P_i. We adopt three schemes to determine this threshold: (1) **Adaptive Threshold**: The user specifies a value that is relatively close to the actual noise level (assuming the user knows about the noise level in the dataset); (2) **Fixed Threshold**: The system automatically assigns a value for α, where possible values of α are specified by experiences or empirical results; and (3) **Best-L Rules**: We rank all rules of P_i by their $SubPrec(r, P_i)$ and select the best L rules with the highest precisions. Meanwhile, all selected rules should have their $SubPrec(r, P_i)$ higher than a threshold α_w (we set $\alpha_w = 0.6$ in our system). To determine L for a given dataset, we calculate the average number of rules induced from each subset, and L is assigned as θ% of this average number. Our experimental results suggest that θ=70 can usually provide a good performance on our evaluate datasets. With the Best-L Rules scheme, the more complicated the dataset, the larger is the number L.

2.2 Deduction with the Good Rule Set

Given an instance I_k and a good rule set GR_i, deduction takes place to evaluate how well GR_i classifies I_k as follows. If I_k is not covered by any rule in GR_i, I_k remains unclassified; If I_k is covered by at least one rule in GR_i, and these rules have the same classification, I_k is classified. However, in the case that I_k matches more than one rule in GR_i, and there are disagreements among these rules, the confidence of each rule is used to determine the classification of I_k, where the confidence is determined by the coverage and accuracy of each rule (Clark & Boswell 1991; Wu, 1998).

With the above deduction strategy, the good rule set GR_i only classifies the instances on which GR_i has confidence in its classification. Consequently, exceptions and noise are likely to be treated in different ways: the exceptions are usually not covered by the good rules, but the noisy instances would likely deny the good rules (covered but with a different classification). The error count variables of misclassified examples will receive larger values than non-noisy examples.

2.3 Multi-Round Execution and Stopping Criterion

Instead of trying to accomplish noise elimination at one time, we identify noise in multiple rounds until the following stopping criterion has been satisfied: in T_1 continuous rounds, if the number of identified noisy examples in each round is less than T_2, noise elimination will stop. In our system, we set T_1=3 and $T_2 = X \cdot 0.01$, where X is the number of examples in the training set. After each round, we remove the identified noisy examples, and also remove a certain number of good examples (this number is smaller than the number of identified noisy examples) from the training set. Our experimental results (Zhu *et al.*, 2003) demonstrate that when we keep reducing the noise level, eliminating a small portion of good examples will not have much influence with the system performance. The benefit of eliminating good examples is that it can shrink the dataset size, and consequently, when executing data partitioning in the next round, we can have a smaller number of subsets. Therefore, the exceptions in the subsets of the last round may form new rules in the next round. While removing good examples, the instances with I_k^{le} and I_k^{ge} both equal to zero are selected as good examples.

3. Noise Elimination from Distributed Datasets

To eliminate noise from distributed datasets, an intuitive way is to apply the mechanism above on the dataset at each site separately. However, with this scheme, it may not be rare that instances eliminated from one site might be useful for other sites. Meanwhile, for various reasons, sharing or downloading the data from each site might be impossible, but sharing the rules is usually allowable. Accordingly, we can treat all datasets from different sites as a virtual dataset (E'), and perform the noise elimination. In the first stage, the system executes data partitioning on each distributed dataset by considering the dataset size. After that, the procedure in Section 2 is adopted to induce and construct a good rule set from each data subset. Then, the good rule sets are used to evaluate all data in E': while using a good rule set GR_i from P_i to identify noise from other sites, e.g., P_j, only the GR_i itself is passed to P_j. Therefore, no data from any site would leak out. After noise identification on all subsets is completed, each data site removes identified noise and a small portion of good examples from its own dataset. After the same procedure has been executed on all datasets, it will repeat the same procedure until the stopping criterion has been satisfied.

With the above strategy, only induced rules are shared, the privacy and security of the data are maintained. Moreover, the noise is determined by not only the local site but also all other distributed datasets.

4. Experimental Evaluations

4.1 Experiment Settings

To construct classification rules and base classifiers, C4.5rules (Quinlan, 1993) is adopted in our system. We have evaluated our noise elimination strategy extensively on both synthetic and realistic datasets, and more details can be found in Zhu *et al*. (2003). Due to size restrictions, here we will mainly report the results of two relatively large datasets from the UCI data repository, Adult and Mushroom.

To add noise, we adopt a pairwise scheme: given a pair of classes (X, Y) and a noise level x, an instance with its label X has a $x \cdot 100\%$ chance to be corrupted and mislabeled as Y, so does an instance of class Y. We use this method because in realistic situations, only certain types of classes are likely to be mislabeled. Using this method, the percentage of the entire training set that is corrupted will be less than $x \cdot 100\%$ because only some pairs of classes are considered problematic. In the sections below, we corrupt only one pair of classes (usually the pair of classes having the highest proportion of instances, except the Splice dataset, where we corrupt the classes EI and IE) and report only the value x in all tables and figures.

For each experiment, we perform 10-fold cross-validation and use the average accuracy as the final result. In each run, the dataset is randomly (with a proportional partitioning scheme) divided into a training set and a test set, and we corrupt the training set by adding noise with the above method, and use the test set to evaluate the system performance. In the sections below, the original dataset means the noise corrupted training set.

For a better evaluation, we adopt three factors: ER_1, ER_2 and Noise Elimination Precision (*NEP*). ER_1 occurs when a non-noisy instance is tagged as noise; and ER_2 occurs when a noisy example is tagged as a correctly labeled instance. Their equational definitions are given by Eq. (3).

$$P(ER_1) = \frac{\| F \cap G \|}{\| G \|} \quad P(ER_2) = \frac{\| \tilde{F} \cap M \|}{\| M \|} \quad NEP = \frac{\| F \cap M \|}{\| F \|} \quad (3)$$

where $\|X\|$ denotes the number of examples in set X; F is the set of examples that are identified as noise and have been removed; $\tilde{F}$ is F's complement; G is the set of non-noisy examples and M is the set of noise.

4.2 Threshold Schemes for Noise Identification

In this subsection, we compare the performances of the majority and non-objection threshold schemes in noise identification. We first split the original dataset into 5 subsets by using a proportional partitioning scheme (a detailed analysis on the number of subsets will be provided in Section 4.4). To select good rules, we use an adaptive threshold scheme (in Section 2.1): when processing a dataset with a noise level of x, we randomly select a value for α with the constraint in Eq. (4).

$$\alpha = \begin{cases} 1 - Random\{[(x-0.1),(x+0.1)]\}; & (x-0.1) \geq 0 \\ 1 - Random\{[0, \quad (x+0.1)]\}; & (x-0.1) < 0 \end{cases} \quad (4)$$

We tabulate the results of noise elimination in Tables 1 and 2 and depict the accuracy improvements in Figure 4.

From Tables 1 and 2, we can find that the non-objection threshold scheme is much safer than the majority scheme, and rarely (usually less than 2%) identifies good examples as noise (ER_1 errors). The noise elimination precision (*NEP*) from the non-objection scheme is much better than the majority scheme. However, as a tradeoff, it shows a relatively severe problem in preventing too much noise from being removed (ER_2 errors). One of the serious problems with ER_2 errors is that they cause the classifier learned from the cleaned dataset to still suffer from the low accuracy problem. As shown in Figure 4, its classification improvement is relatively limited in comparison with the majority scheme. These observations suggest that we may not always want the *NEP* to be maintained at a high level, and that a more aggressive scheme is needed if this scheme can keep its ER_1 errors at an acceptable level. From Tables 1 and 2, we can find that the majority scheme provides a remarkable tradeoff between the system performance and the *NEP*. This scheme can remove about 80% noise from most datasets at different noise levels (the results from the Adult dataset are relatively poor). As a result, it receives more improvement on its classification accuracy. From Figure 4, take the noise level at 20% as an example. Although both majority and non-objection schemes receive remarkable accuracy improvements compared with the original noise corrupted training set, but the majority scheme has about 2% and 4% more improvements for the Mushroom and Adult datasets respectively. In the sections below, unless specified otherwise, we will use the majority scheme for noise identification.

Table 1. Noise identification schemes (Mushroom dataset)

Noise	Majority Scheme			Non-objection Scheme		
(%)	*NEP*	$P(ER_1)$	$P(ER_2)$	*NEP*	$P(ER_1)$	$P(ER_2)$
5	0.801	0.0131	0.0104	0.873	0.0082	0.0075
15	0.921	0.0231	0.0195	0.945	0.0107	0.0582
25	0.943	0.0358	0.0695	0.965	0.0123	0.1231
35	0.681	0.1584	0.2104	0.867	0.0292	0.5214
40	0.662	0.2289	0.2949	0.833	0.0426	0.6955

Table 2. Noise identification schemes (Adult dataset)

Noise	Majority Scheme			Non-objection Scheme		
(%)	*NEP*	$P(ER_1)$	$P(ER_2)$	*NEP*	$P(ER_1)$	$P(ER_2)$
5	0.512	0.0335	0.3527	0.795	0.0047	0.6643
15	0.687	0.0563	0.2996	0.891	0.0095	0.5686
25	0.764	0.0738	0.2723	0.918	0.0133	0.5495
35	0.734	0.1488	0.2223	0.943	0.0121	0.6092
40	0.702	0.2629	0.2486	0.946	0.0128	0.6563

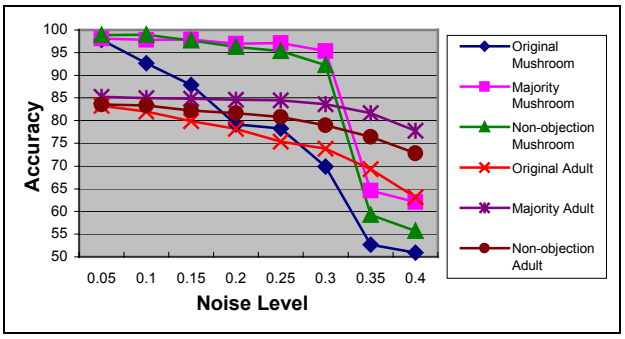

Figure 4. Classification accuracies on Adult and Mushroom datasets (5 subsets)

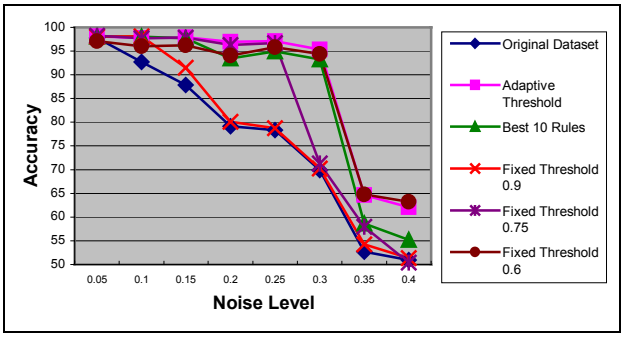

Figure 5. Good rule selection schemes (Mushroom dataset, 5 subsets, Majority scheme)

4.3 Good Rule Selection Schemes

In this subsection, we compare the system performance with three good rule selection schemes in Section 2.1: (1) the adaptive threshold scheme, with Eq. (4) for determining the value for α; (2) the fixed threshold scheme, with three typical fixed values α=*0.9, 0.75* and *0.6*; and (3) the Best-*L* rules scheme, with θ=70 which usually generates *L*=10 and *L*=20 for the Mushroom and Adult datasets respectively. Figure 5 presents the experimental results.

From Figure 5, one can easily conclude that the adaptive threshold scheme is the best in almost all situations. With the fixed threshold scheme, when the noise level is less than 1-α, its performance is a little worse than the performance of the adaptive threshold scheme; however, once the noise level is higher than the given threshold, the system performance declines rapidly (as demonstrated when α=*0.9* and *0.75*). Our analysis shows that when the noise level x is less than 1-α, the system tends to take more insignificant rules into the good rule set, and as a result, more good examples are identified as noise. On the other hand, when the noise level is higher than 1-α, the good rules' precision would tend to be less than 1-α (the noise corrupts the good rules' precision), and most good rules would not be taken into the good rule set. Hence, less noise is eliminated, which critically affects the

system performance. As indicated in Figure 5, the results from α=*0.6* are acceptable to some degree, even when worse performances have been found from the lower noise levels. Hence, in the case that the user has no idea about the noise level, we may specify a small value for α. One can find that the results from the Best-*L* rules scheme are very attractive. Usually, it has a slightly worse performance than the adaptive threshold scheme. These situations usually happen when the noise level is relatively high. For the Mushroom dataset, when *L*=10 and the noise level is *30%*, the Best-*L* rules scheme receives about 2% less improvement than the adaptive scheme and is similar to the fixed threshold scheme, but in low noise level environments, it is better than the fixed threshold scheme. The same conclusion can be drawn from other datasets. One can image that in general situations, the user usually has no idea about the noise level, and from this viewpoint, the Best-*L* rules scheme might be the best choice to select good rules. In the following sections, unless specified otherwise, we use the Best-*L* rules scheme to select good rule for each subset.

4.4 Number of Subsets on System Performance

Due to the fact that our approach uses the minor set (one single subset) to identify noise from the major set (all other subsets), we need to evaluate how the number of subsets affects the system performance. If the system performance is very sensitive to the number of subsets, it would imply that this method might not work well in realistic situations. For any given dataset at a certain noise level (20%), we use a proportional partitioning scheme to construct various numbers of subsets, and use both the non-objection and majority threshold schemes to identify noise. The results of noise elimination and classification accuracy improvements are given in Table 3.

Table 3 demonstrates that when using more subsets, the non-objection scheme prevents more noise from being eliminated. One of the worst consequences of keeping much noise is that the classifier learned for the dataset would have very limited improvement (sometimes no improvement). The classification accuracy of the non-objection scheme decreases rapidly with the increasing number of subsets. The majority scheme, on the other hand, sacrifices a small portion of ER_1 errors but gains considerable improvement on ER_2 errors. With this scheme, the number of subsets has less impact on the system performance. Comparing the results with 3 and 39 subsets respectively, the ER_1 errors of the majority scheme increases from 5.82% to 14.62%, but we still receive remarkable improvement on the classification accuracy. Actually, there is a contradiction between the number of subsets and the system performance: the more the subsets, the larger is the number of decision committee members (because we take each subset as a decision maker to identify noise), and meanwhile, the weaker is the ability of each committee member (because with the increase of the subset number, less information is contained in each subset). From Table 3, one can intuitively conclude that the larger the number of subsets, the worse is the system performance. Actually, at the beginning, there is a trend that the accuracy increases with the increase of the subset number, but after a certain number it begins to drop. However, even in the extreme case, e.g., 127 subsets, Table 3 indicates that the majority scheme can still attain a good performance. In most of our experiments, we use 5 subsets.

Table 3. Noise elimination results with various subset numbers (Adult dataset)

Sub sets	$P(ER_1)$		$P(ER_2)$		Accuracy (%)		
	Non-obj.	Maj.	Non-obj.	Maj.	Org.	Non-obj.	Maj.
3	0.0303	0.0582	0.4265	0.3067	78.3	83.7	84.4
15	0.0040	0.0700	0.7232	0.2882	78.1	80.3	84.5
27	0.0009	0.1504	0.8244	0.2355	78.5	79.1	83.3
39	0.0002	0.1462	0.8916	0.2118	77.9	78.4	83.4
51	0	0.1726	0.9571	0.2424	78.3	77.9	82.4
63	0	0.1869	0.9781	0.2947	78.2	78.2	82.1
95	0	0.1648	0.9971	0.5120	78.2	78.2	80.9
127	0	0.1729	0.9992	0.6594	78.3	78.3	80.1

4.5 Multiple Rounds of Execution

While most existing methods perform noise identification in one round, we execute the procedure for multiple rounds until the stopping criterion is satisfied. One may ask whether it's necessary to perform multiple rounds if one round of execution can attain satisfactory results. We have recorded the percentage of eliminated noise after the execution of each round (at different noise levels), and show the results in Figure 6, where the *x*-axis denotes the number of execution rounds and the *y*-axis represents the percentage of identified noise until this round. Figure 6 demonstrates that with the Adult dataset, at any noise level, the first round can only eliminate about 40% of noise; however, running the same procedure for 5 rounds will increase this percentage to 60%. Actually, our experimental results from other datasets (e.g., Mushroom) indicate that when the noise level is relatively low (less than 30%), the results from one round of execution are basically good enough (about 90% of the noise could be eliminated). However, when the noise level goes higher, it's necessary to execute the same procedure more than once. Generally, since the Adult dataset contains many exceptions, its classification accuracy is relatively low (85%) even without any mislabeled error. Hence, we believe the results from the Adult dataset are representative for large datasets.

More experiments show that in most situations, the first five rounds usually eliminate most noise (over 85%) that the system can identify. Accordingly, instead of using the stopping criterion in Section 2.3, another possible empirical operation to stop the program is to execute the procedure for a certain number of rounds.

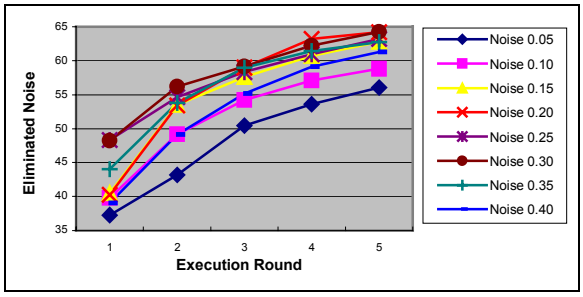

Figure 6. Noise elimination from multiple rounds (Adult dataset, 5 subsets, Majority and Best-20 Rules schemes)

4.6 More Experimental Results

In addition to the results from the Mushroom and Adult datasets, Table 5 shows more experiments on 7 other datasets. More details can be found in Zhu *et al.*, (2003). Also, for comparative studies, we have implemented the Classification Filter proposed by Gamberger *et al*. (1999) (with 10 subsets, as recommended by these authors). In Table 5, the first column indicates the noise level and the other columns present accuracies from different datasets, where OG indicates the classification accuracy of the classifier learned from the original dataset, CF represents the accuracy from the Classification Filter, and PF denotes the results from our approach (Partitioning Filter).

From the results in Table 5, the system performance can always improve with noise elimination. With all seven datasets, noise elimination can contribute from 1% to over 20% to the system performance improvment, varying from noise levels and datasets. One can also find that when the noise level is relatively low (less than 15%), the CF method outperforms the PF scheme on 4 out of 7 datasets. However, when the noise level becomes higher, the performance of CF decreases rapidly. When the noise level is at 35%, for example, CF outperforms PF on only 1 out of 7 datasets. Actually, this only dataset is Splice, in which the instances of the corrupted classes occupy only 50% of the instances in the whole dataset, i.e., the actual noise in the dataset is one half of the noise level *x*. Further analysis provides an answer for this phenomenon: since CF learns its classifiers from the major sets and uses them to identify noise in the minor sets, when the noise level is low, the learned base classifiers would behave well and have relatively good performances in identifying noise; but when the noise level increases, the learned classifiers tend to make more mistakes, which will in turn either wrongly identify some good instances as noise or keep noisy examples as good instances. Our approach selects good rules, and the increase of noise has less impact on the system performance. Moreover, instead of fully relying on any single classifier (as the CF scheme does), we perform noise identification from all subsets, and our scheme is more robust in noisy environments.

In addition to the classification accuracy, we also compare the efficiency between CF and PF. Table 4 shows the execution times of the two methods at different noise levels. We used a PC with Pentium 4, a 2GHz processor, and 512 MB memory to implement both methods. The PF scheme is much better than the CF approach in terms of time efficiency. When the noise level is 0 and 40%, PF is about 3.4 and 40 times faster than CF respectively. Our results from other datasets indicate that on average, PF is about 10 times faster than CF. Actually, the larger the dataset, and the higher the noise level, the more efficient is PF compared to CF. Part of the reason is that the time complexity of C4.5rules nonlinearly increases with the noise level and the number of instances. If adopting different inductive algorithms, the impact of noise might be less significant, but we believe PF will still be much more efficient than CF.

Table 4. Execution times in seconds (Mushroom dataset)

Methods	Execution time at different noise levels				
	0%	10%	20%	30%	40%
CF	18.2	159.3	468.6	868.4	1171.2
PF	5.3	12.8	19.8	22.8	29.6

4.7 Discussion

Our approach is related to the Classification Filter (CF) proposed in Gamberger *et al.* (1999). However, there are three key distinctions between them even though they both adopt a partitioning scheme: (1) CF learns the base classifier from the aggregation of any *n*-1 subsets (the major set) to identify noise from the complementary (excluded) subset, while our approach learns a classifier from each single subset (the minor set) to evaluate the whole dataset; (2) when identifying noisy examples, CF fully trusts each base classifier, i.e., the instances

Table 5. Experimental results and comparisons from seven datasets of UCI data repository

Noise (%)	Krvskp (%)			Car (%)			Nursery (%)			WDBC (%)			Splice (%)			Credit-app (%)			Connect-4 (%)		
	OG	CF	PF	OG	CF	PF	OG	CF	PF	OG	CF	PF	OG	CF	PF	OG	CF	PF	OG	CF	PF
5	96.6	98.5	97.9	91.5	91.8	91.3	95.8	96.9	96.2	92.6	92.2	93.9	89.1	92.6	91.8	81.9	85.3	85.6	73.2	75.8	75.7
15	88.1	97.5	96.3	82.7	88.7	88.6	90.4	96.5	94.3	90.6	91.5	92.4	85.6	92.1	91.4	73.7	84.6	86.7	68.2	74.7	75.1
25	76.7	96.4	95.2	76.8	83.8	86.4	83.5	94.9	93.3	88.3	90.1	91.1	82.1	91.2	89.7	66.7	83.4	85.2	61.6	71.8	72.5
35	68.3	93.1	93.6	67.5	78.1	82.7	77.5	90.4	92.7	82.7	84.7	84.9	77.6	89.1	86.4	61.5	80.5	83.9	55.8	68.8	69.7
40	60.7	83.1	84.8	61.8	69.7	81.8	72.7	83.1	92.3	78.6	79.2	79.7	75.5	87.4	80.9	58.2	79.1	81.4	51.6	66.5	67.9

incorrectly classified by the base classifier are directly identified as noise, but our approach takes each base classifier as a committee member for noise identification; and (3) the base classifier in our approach processes only instances on which it has the confidence. Consequently, our approach is more efficient in handling large and very noisy datasets. Meanwhile, one can find that the result of the CF scheme critically depends on the performance of adopted learning algorithms. Assuming an inductive algorithm has a ω% classification accuracy with a noise-free dataset E'', the CF scheme will inevitably identify about 100%-ω% of the instances as noise, even if there is no noise in E''. This is because it unconditionally trusts each base classifier. This strategy obviously neglects the fact that inductive algorithms cannot work perfectly, even with noise-free datasets. Our experimental results (Zhu *et al.*, 2003) indicate that the CF scheme is usually more aggressive than our approach, eliminates more instances, and makes more ER_1 errors. This might be another reason that it occasionally works better than PF when the noise level is low, because removing a certain portion of exceptions likely increases the classification accuracy of some datasets.

5. Conclusions

An inherent disadvantage of existing approaches in identifying class noise is that they are less efficient in handling large, distributed datasets. In this paper, we have presented a Partitioning Filter approach for noise identification from large, distributed datasets. Given a dataset E, we first partition it into N subsets. For any subset P_i, a set of good rules are induced and selected to evaluate all instances in E. For any instance I_k in E, two error count variables are used to record how this instance behaves with the good rule sets from all subsets. Due to the fact that exceptions usually do not fire the good rules and noise more likely denies the good rules, the noise has a higher probability of receiving large error values in comparison with non-noisy examples. We have adopted different threshold schemes to identify noise. After each round, the identified noise and a certain portion of good examples are removed, and the same procedure is repeated until the stopping criterion is satisfied. Experimental evaluations and comparative studies have shown that our proposed approach is effective and robust in identifying noise and improving the classification accuracy.

Acknowledgements

This research is supported by the U.S. Army Research Laboratory and the U.S. Army Research Office under grant number DAAD19-02-1-0178. The authors would like to thank the anonymous reviewers for their constructive comments on an earlier version of this paper.

References

Blake, C.L. & Merz, C.J. (1998). *UCI Repository of machine learning databases*.

Breiman, L., Friedman, J.H., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees*. Wadsworth & Brooks, CA.

Brodley, C.E. & Friedl, M.A. (1996). Identifying and eliminating mislabeled training instances, *Proc. of 13th National conf. on artificial intelligence*, pp.799-805.

Brodley, C.E. & Friedl, M.A. (1999). Identifying mislabeled training data, *Journal of Artificial Intelligence Research*, vol. 11: 131-167.

Chan, P. K.-W. (1996). An extensive meta-learning approach for scalable and accurate inductive learning, *Ph.D Thesis, Columbia University*.

Clark, P., & Boswell, R. (1991). Rule induction with CN2: some recent improvement, *Proc. of 5th ECML*, Berlin, Springer-Verlag.

Gamberger, D., Lavrac, N., & Groselj C. (1999). Experiments with noise filtering in a medical domain, *Proc. of 16th ICML*, pp. 143-151, San Francisco, CA.

Gamberger, D., Lavrac, N., & Dzeroski, S. (2000) Noise detection and elimination in data preprocessing: experiments in medical domains. *Applied Artificial Intellig.*, vol. 14: 205-223.

Guyon, I., Matic, N., & Vapnik, V. (1996). Discovering information patterns and data cleaning. *Advances in Knowledge Discovery and Data Mining*, pp.181-203. AAAI/MIT Press.

John, G. H. (1995). Robust decision trees: Removing outliers from databases. *Proc. of 1st international conf. on knowledge discovery and data mining*, pp.174-179.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning*, 1(1): 81-106.

Quinlan, J.R. (1993). *C4.5: programs for machine learning*, Morgan Kaufmann, San Mateo, CA.

Provost, F., & Kolluri, V. (1999). A Survey of Methods for Scaling Up Inductive Algorithms. *Data mining and knowledge discovery*, 3(2):131-169.

Srinivasan, A., Muggleton, S., & Bain, M. (1992). Distinguishing exception from noise in non-monotonic learning. *Proc. of 2th ILP Workshop*, pp.97-107.

Verbaeten, S. (2002). Identifying mislabeled training examples in ILP classification problems, *Proc. of Machine learning conf. of Belgium and the Netherlands*.

Wu, X. (1995), *Knowledge acquisition from database*, Ablex Pulishing Corp., USA.

Wu, X. (1998), Rule Induction with Extension Matrices, *American Society for Inform. Science*, 49(5):435-454.

Zhu, X., Wu, X. & Chen Q. (2003). Identifying class noise in large, distributed datasets, *Technical Report, http://www.cs.uvm.edu/tr/CS-03-12.shtml*.