
Data Mining and Knowledge Discovery, 12, 275–308, 2006

DOI: 10.1007/s10618-005-0012-8

Bridging Local and Global Data Cleansing: Identifying Class Noise in Large, Distributed Data 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

Received April 3, 2005; Accepted July 27, 2005

Published online: 4 April 2006

Abstract. To cleanse mislabeled examples from a training dataset for efficient and effective induction, most existing approaches adopt a major set oriented scheme: the training dataset is separated into two parts (a major set and a minor set). The classifiers learned from the major set are used to identify noise in the minor set. The obvious drawbacks of such a scheme are twofold: (1) when the underlying data volume keeps growing, it would be either physically impossible or time consuming to load the major set into the memory for inductive learning; and (2) for multiple or distributed datasets, it can be either technically infeasible or factitiously forbidden to download data from other sites (for security or privacy reasons). Therefore, these approaches have severe limitations in conducting effective global data cleansing from large, distributed datasets.

In this paper, we propose a solution to bridge the local and global analysis for noise cleansing. More specifically, the proposed effort tries to identify and eliminate mislabeled data items from large or distributed datasets through local analysis and global incorporation. For this purpose, we make use of distributed datasets or partition a large dataset into subsets, each of which is regarded as a local subset and is small enough to be processed by an induction algorithm at one time to construct a local model for noise identification. 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 data 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 and eliminate the noisy examples. Experimental results and comparative studies on both real-world and synthetic datasets are reported to evaluate the effectiveness and efficiency of the proposed approach.

Keywords: data cleansing, class noise, machine learning

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. In reality, the procedure of maximizing the accuracy is normally complicated by two important factors: (1) the quality of the training data; and (2) the inductive bias of the learning algorithm. Given any particular learning theory, it is obvious that its classification accuracy depends vitally on the quality of the training data, where various kinds of noise can corrupt the data and decrease the classification

*A preliminary version of this paper was published in the Proceedings of the 20th International Conference on Machine Learning, Washington D.C., USA, 2003, pp. 920–927.
†Corresponding author.

accuracy. Accordingly, the most feasible way to improve the system effectiveness is to cleanse the training data such that noise or imperfect data can be removed or enhanced accordingly. There are two types of noise sources (Wu, 1995; Zhu and Wu, 2004): (a) attribute noise; and (b) class noise. The former is represented by the following four types of errors, where the noise is introduced in the attribute values of the instances: (1) erroneous attribute values, (2) missing attribute values or don't know values, (3) incomplete attributes and uneven data distribution, and (4) redundant data and don't care values. There are two possible sources for class noise:

(1) *Contradictory examples*. The same examples appear more than once and are labeled with different classifications.
(2) *Misclassifications*. Instances are labeled with wrong classes. This type of errors is common in situations that classes have similar symptoms.

Basically, the above two types of class noise are derived from the same source—mislabeling, and due to this fact, in this paper, we call the class noise mislabeled errors, and will identify and eliminate this type of noise.

The challenge of attribute noise cleansing comes from two important aspects: (1) how to distinguish an instance with erroneous (or missing) attribute values; and (2) how to handle an instance containing attribute noise. For most types of attribute noise such as missing attribute values or don't known values, the instance itself will explicitly indicate whether it contains attribute noise or not, e.g, with a "?" to represent a missing attribute value. Noise cleansing for these types of attribute noise mainly focuses on how to handle such a noisy instance. The simplest mechanism in this regard is to ignore the instances containing any missing information, as Friedman (1977) and Breiman et al. (1984) have done in their algorithms. Nevertheless, such a technique has two serious problems: (1) it obviously loses valuable information in the training set; and (2) a test instance with the same kind of attribute noise will not be properly classified. Consequently, many other approaches have been developed, such as filling in unknown values with its most common known value (Clark and Niblett, 1989), using a Bayesian formalism (Kononenko et al., 1984), a decision-tree (Shapiro, 1983) or K nearest neighbors (Huang and Lee, 2001) to predict the missing (or don't know) values. Quinlan (1993) uses the entropy and splits the examples with missing attribute values to all concepts while constructing a decision tree. Comprehensive comparisons on missing attribute value prediction can be found in Quinlan (1989), Grzymala-Busse and Hu (2000) and Bruha and Franek (1996). On the other hand, for instances containing erroneous attribute values, it would be very hard to distinguish them from normal instances, because erroneous values would make an instance act like a new training example containing valuable information. Consequently, rare research efforts have been conducted in cleansing erroneous attribute values (Teng, 1999; Kubica and Moore, 2003; Zhu et al., 2004). Meanwhile, Quinlan (1986) has 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. It indicates that eliminating attribute noise may run a high risk in reducing system accuracy, as long as the same level attribute noise exists in the test set. Accordingly, instead of cleansing attribute noise, most learning theories tend to adopt some special procedures, such as prepruning and postpruning, to deal with attribute noise.

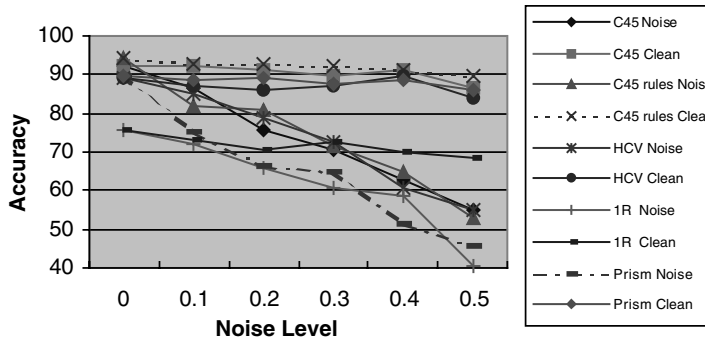

Figure 1. Classification accuracy of various classifiers trained from noise corrupted and manually cleaned training sets, where "X Noise" indicates the classifier "X" trained from a noise corrupted training set and "X Clean" represents the classifier "X" trained from a cleaned training set (Car dataset).

For class noise, the conclusion from Quinlan (1986) may not hold: cleansing the noisy training data often results in a classifier superior to the one from the original massive data (Brodley and Friedl, 1999). As demonstrated in Figure 1, where the *x*-axis indicates the class noise level, and the *y*-axis represents the classification accuracy from different types of classifiers trained from the noise corrupted and manually cleaned training set (evaluated with the same test set), when the noise level increases, all classifiers trained from noise corrupted training set suffer from decreasing the classification accuracy dramatically, where the classification accuracies decline almost linearly with the increase of the noise level (we have used five classification algorithms, C4.5 (Quinlan, 1993), C4.5rules (Quinlan, 1993), HCV (Wu, 1995), 1R (Holte, 1993) and Prism (Cendrowska, 1987) in our experiments. The noise corruption and experiment design details are given in Section 6). On the other hand, the classifiers from the manually cleansed training set (in which instances containing class noise are removed) will have their classification accuracies improved comprehensively. Pruning or classifier ensembles may partially alleviate the noise impact, but noise can still drastically affect the classification accuracy, as long as it exists in the training set. Accordingly, many research efforts have been made on eliminating mislabeled errors for effective learning. A normal practice is to build a macro-model, say T, from the whole original massive data, and then evaluate the consistency of the underlying instances with the model T. The examples which do not comply with T are normally treated as noisy items and removed accordingly. This filtering oriented cleansing procedure performs reasonably well in many situations (Brodley and Friedl, 1999), but when it comes to the process of large or distributed datasets, this mechanism suffers from the following two disadvantages:

1. For large datasets, building the macro-model T is often impractical or hopelessly inefficient, because it requires the users to feed all data into the memory to induce T.
2. For distributed datasets or multiple data sources, existing endeavors fail to come up with a global noise cleansing model, unless the users put all data together to induce the model T, which is either technically infeasible or factitiously forbidden.

The above observations have motivated our design to bridge local and global noise cleansing for better performances (in terms of efficiency and accuracy). Being local, the proposed effort builds micro models from local subsets for noise identification, so it scales up well for large or distributed datasets. Being global, on the other hand, all micro models cooperate with each other and work as a committee to identify noisy examples, so the identified noise is indeed noisy items from the whole dataset perspective. Although well motivated, to ensure success, we need to take the following issues into consideration for a unified design which combines both local and global noise cleansing: (1) How to define and characterize a local micro model for noise identification? (2) How to organize all the local micro modes for global noise cleansing? and (3) How to handle distributed datasets?

In this paper, we present our recent research efforts towards resolving the above concerns. We will propose a *Partitioning Filter* (PF) in noise cleansing from large or distributed datasets, which carries the following three novel features in noise identification:

- It induces a noise identifier from each single subset (minor set), and uses this weak micro model to evaluate all instances in the major set.
- To unify all micro models for global noise cleansing, all noise classifiers work as a committee for noise identification.
- For each single micro model, it only handles the instances in which it has the confidence.

The experimental results and comparative studies from various datasets will demonstrate the effectiveness and efficiency of our approach: with all tested datasets, at any noise level (even 40%), the classifier trained from each dataset that has been processed by PF always shows a remarkably improved classification accuracy. Also when comparing with other schemes, our approach achieves a comprehensive improvement in time efficiency, and is more robust in high-level noisy environments.

The rest of the paper is organized as follows. In the next section, related work on class noise elimination is reviewed. The proposed noise identification approach is described in Section 3. How to deal with distributed datasets is discussed in Section 4. We analyze noise identification errors of two proposed schemes in Section 5. In Section 6, the effectiveness of the proposed schemes is reported with synthetic data from the IBM data generator (IBM Synthetic Data) and with realistic datasets from the UCI data repository (Blake and Merz 1998).

2. Related work

The problem of noise cleansing has been a focus of much attention in machine learning and most inductive learning algorithms have a mechanism for noise handling in the training data (Brodley and Friedl, 1999). Pruning on a decision tree is designed to reduce the chance that the tree is overfitting to noise. However, since the classifiers learned from noisy data have less accuracy, the pruning may have a very limited effect in enhancing the system performance, especially in situations that noise levels are relatively high. As pointed out by Gamberger et al. (2000), removing noise from the data before hypothesis formation has the advantage that noisy examples do not influence

hypothesis construction. There are two types of noise identification approaches, general schemes and domain specific schemes. General schemes are truly general in the sense that they do not use any domain-specific knowledge in their process. In domain specific schemes, domain knowledge can be used to benefit the noise identification (Whitaker and Saroiu, 1999; Li et al., 2002).

For a noise elimination algorithm, regardless whether domain knowledge has been integrated or not, the noise cleansing procedure is normally achieved through two opposite approaches: selecting "good" examples or deleting "bad" instances.

To select "good" examples, Wilson (1972) used a *k*-nearest neighbor (*k-NN*) classifier to select instances that were then used to form a *1-NN* classifier, and only those instances that the *k-NN* classified correctly were retained for the *1-NN*. Based on this scheme, many instance selection schemes have been developed for good example selection for machine learning (Tomek, 1976; Aha et al., 1991; Skalak, 1994). Cestnik et al. (1987) proposed an application system ASSISTANT which was able to select good examples only. A comprehensive overview of the research on this topic can be found in Wilson and Martinez (2000). Due to the fact that the quality of the training data vitally determines the accuracy of the classifier, the idea of selecting "good" examples has also been applied to other types of classifiers, where various kinds of selection approaches, such as near misses and the most-on-point cases, are adopted to find the examples that can improve the effectiveness of the classifiers (Winston, 1975).

Instead of selecting "good" examples, some methods conduct noise elimination by deleting "bad" instances. Basically, one of the most challenging tasks in removing "bad" instances is how to distinguish noise from exceptions to general rules. Since when an instance is an exception to general cases, it can also appear as if it is incorrectly labeled. Especially, when the instances belong to the small disjuncts of the training set, they behave more likely to be the exceptions of general cases. In Weiss (1995) and Weiss and Hirsh (1998), the problem of learning with rare cases and small disjuncts has been extensively evaluated, even though no solution of noise elimination was suggested. They concluded that when low levels of attribute noise are applied to both the training and test sets, rare cases make learning only slightly more difficult. However, when low levels of attribute noise are applied only to the training set, rare cases within a domain are primarily responsible for making learning difficult. Moreover, relatively low levels of class noise will increase the percentage of errors contributed by small disjuncts, but this effect will diminish as higher levels of noise are applied. However, high levels of class noise were required to cause rare cases and small disjuncts to be error prone. All these conclusions imply that when there are rare cases contained in the training set, removing "bad" instances run high risks of eliminating the small disjuncts that look like (but not true) noise.

To distinguish "bad" instances from normal cases, various strategies have been designed. Among them, the most general techniques are motivated by the intention of removing outliers in regression analysis (Weisberg 1980). An outlier is a case that does not follow the same model as the rest of the data and appears as though it comes from a different probability distribution. As such, an outlier does not only include erroneous data but also surprisingly correct data. In John (1995), a robust decision tree was presented, and it took the idea of pruning one step further: training examples that are misclassified by the pruned tree, are also globally uninformative. Therefore, after pruning a decision tree, the misclassified training examples should be removed from

the training set and the tree needs to be rebuilt using this reduced set. This process is repeated until no more training examples are removed. With this method, the exceptions to the general rules are likely to be removed without any hesitation; hence, this scheme runs a high risk of removing both exceptions and noise.

Instead of employing outlier filtering schemes, some researchers believe that noise can be characterized by various measures. 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. However, they noted that because their method is an on-line method it suffers from ordering effects. Oka and Yoshida (1993, 1996) designed a method that learns generalizations and exceptions separately by maintaining a record of the correctly and incorrectly classified inputs in the influence region of each stored example. The mechanism for distinguishing noise from exceptions is based on a user-specified parameter, which is used to ensure that each stored example's classification rate is sufficiently high. Unfortunately, as concluded in Brodley and Friedl (1999), this approach has only been tested on artificial datasets. The method in Srinivasan et al. (1992) uses an information theoretic approach to detect exceptions from noise during the construction of a logical theory. Their motivation is that there is no mechanism by which a non-monotonic learning strategy can reliably distinguish true exceptions from noise. The noise detection algorithm of Gamberger et al. (2000) is based on the observation that the elimination of noisy examples, in contrast to the elimination of examples for which the target theory is correct, reduces the *CLCH* value of the training set (*CLCH* stands for the Complexity of the Least Complex correct Hypothesis). They call their noise detection algorithm the Saturation filter since it employs the *CLCH* measure to test whether the training set is saturated, i.e., whether, given a selected hypothesis language, the data set contains a sufficient number of examples to induce a stable and reliable target theory.

In Brodley and Friedl (1996, 1999), noise elimination is simplified as a filtering model as demonstrated in Figure 2, where corrupted datasets are filtered and cleaned by various noise identification strategies, and then the classifiers learned from cleaned datasets are used for data classification. Based on this filtering model, they proposed a noise identification scheme where noise is characterized as the instances that are incorrectly classified by a set of trained classifiers.

As all of the above strategies have indicated, to perform noise elimination, most existing approaches take 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:

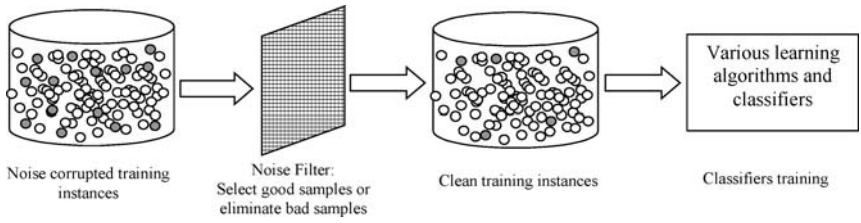

Figure 2. A general filtering scheme in identifying mislabeled training data.

a major set and a minor set, and the major set is used to induce classifiers to identify noise in the minor set. Due to the fact that given a dataset E, the larger the number of training examples the more accurate the induced classifier is, these methods prefer the size of the major set to be close to the size of E (even the whole dataset E is taken as the major set). Nevertheless, with the development of networks and computer hardware, the machine learning community is facing the challenge of large and distributed datasets, which might be too large to be handled in one execution. Hence, the assumptions above are too strong in realistic situations.

When the underlying date volume becomes significantly large, classifier *ensemble* or *committee* based approaches can be adopted to build multiple hypotheses from instance subsets, and then combine their decisions during the classification. Among them, classifier ensemble (Dietterich, 2000) and scaling-up inductive algorithms (Provost et al., 1999) such as boosting (Schapire, 1990) and meta-learning (Chan, 1996) are commonly used for inductive learning from large datasets. As the learner built from a classifier ensemble or committee can act as a noise identifier for large datasets, one may wonder whether those approaches can be directly used to solve our problem. However, there are two problems with this possibility: (1) the theory built from a classifier ensemble or committee is for classification purposes, and usually does not come with noise handing mechanisms. As noise can directly impact to the base learners, a system based on this approach will deteriorate significantly with the increase of the noise level; (2) previous research work from Chan (1996) has shown that although ensemble or committee based mechanisms can handle large datasets effectively, in general, the classification accuracy from these algorithms is worse than the accuracy from the whole dataset, especially in noisy environments. This is because scaling-up algorithms induce rules from sampled data where many exceptions in the sampled data are actually valuable instances in the original dataset. Consequently, when dealing with large datasets, most noise elimination approaches are inadequate, especially when we try to accommodate both local and global noise cleansing for better performances.

For distributed datasets, the problem with existing approaches is even more severe. They assume that all data is right at hand for processing. Unfortunately, in realistic situations, it is either technically infeasible (e.g., bandwidth limitations) or factitiously forbidden (e.g., for security or privacy reasons) to share or download data from other sites (Hall, 2000). One can execute noise elimination algorithms on each single site respectively, however it may inevitably eliminate some instances that are noise for the current site but useful for other sites.

3. Noise elimination from large datasets

The flowchart of our proposed scheme (Partitioning Filter) is depicted in Figure 3, and the various procedures in Figure 3 are given in Figures 4 and 5, where $|\Delta|$ represent the number of instances in set Δ. 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 selected good rules, whereas noisy instances are likely covered by good rules but produce a wrong classification. Based on this assumption, we first partition the whole dataset E into several subsets. For each subset P_i, we use an inductive learning algorithm to learn a micro model, denoted by a set of

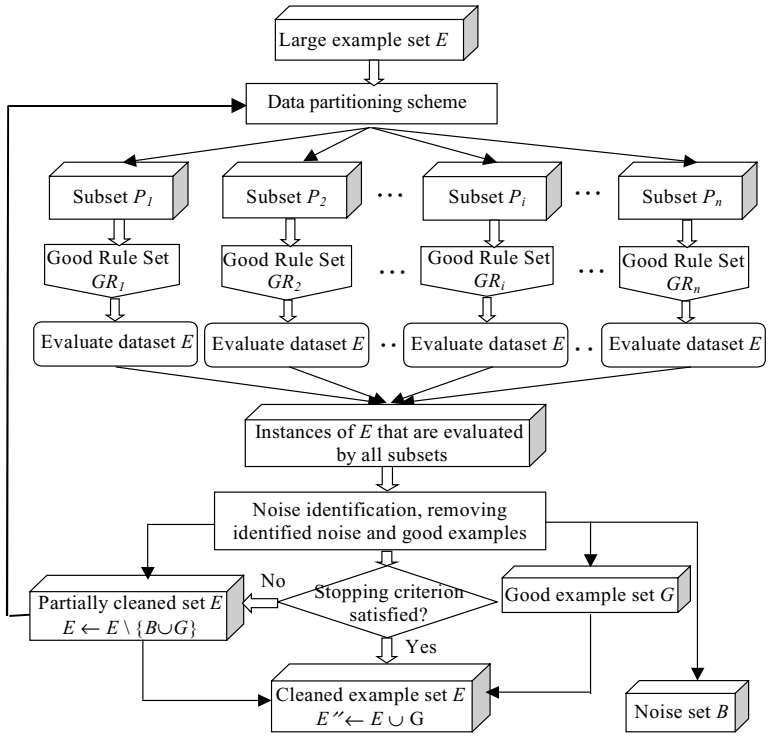

Figure 3. Noise elimination from large datasets (Partitioning Filter).

classification rules R_i. Then we select a good rule set (GR_i) from R_i, and use GR_i to evaluate the whole dataset E. To characterize each local micro model for global noise cleansing, for any 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 class label, we increase its local error count (I_k^{le}) by 1. Meanwhile, if I_k belongs to P_i, we call P_i the host subset of I_k (the number of host subsets of I_k is denoted by variable I_k^{host})
2. If I_k does not belong to P_i and I_k's classification from GR_i is different from its original class label, we increase its global error count (I_k^{ge}) by 1;
3. For other cases, I_k^{le} and I_k^{ge} remain the same values.

After we execute the same procedure on all subsets, the values of variables I_k^{le} and I_k^{ge} of I_k will record the behaviors of all the local micro models with the instance I_k, and will also indicate the possibility that I_k is mislabeled. As we stated above, due to 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 threshold schemes, majority and non-objection, are adopted to identify noisy examples from the global noise cleansing perspective, as demonstrated in Figure 5. In the majority threshold scheme, an instance is identified as noise only if more than

Main Procedure: ***PartitioningFilter***(E)

Input: E (a noisy training set with S examples)

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

Output: E''(cleansed dataset of E)

(1) $GE \leftarrow \emptyset$ //GE is the subset which contains all removed good examples (from all rounds)
(2) $B \leftarrow \emptyset$, $G \leftarrow \emptyset$, $D \leftarrow \emptyset$ // B is the subset which contains identified nose in the current round
// G is the subset which contains identified good examples in the current round
// D is a temporary subset for good example removing purpose
(3) For any instance $I_k \in E$, $I_k^{le} = I_k^{ge} = I_k^{host} = 0$ // I_k^{host} represents how host subsets I_k has
(4) Partition E into n subsets // the number of n is determined by the size of E and the adopted learning algorithm
(5) **For** $i = 1, \ldots, n$ do
(6) For each subset P_i, learn a rule set R_i from P_i
(7) Select good rules GR_i from R_i (See Section 3.1)
(8) **For** $k = 1, .., S$ do //S is the number of instances in E
(9) Given an instance I_k, $I_k \in E$,
(10) **If** $I_k \in P_i$
(11) I_k^{host}++ // calculate the number of host subsets the instance I_k has
(12) **If** $I_k \in P_i$, I_k fires GR_i and its classification result from GR_i is DIFFERENT from its original class label.
(13) I_k^{le}++ // increase the local error count of I_k
(14) **Else if** $I_k \notin P_i$, I_k fires GR_i and its classification result from GR_i is DIFFERENT from its original class label.
(15) I_k^{ge}++ // increase the global error count of I_k
(16) **Endfor**
(17) **Endfor**
(18) **For** $k = 1, .., S$ do
(19) Given instance I_k, $I_k \in E$,
(20) **If** Noise(I_k^{host}, I_k^{le}, I_k^{ge}, n) = 1 (see Figure 5)
(21) $B \leftarrow B \cup \{I_k\}$ // put I_k in the noisy dataset
(22) **Else if** (($I_k^{le} = 0$) & ($I_k^{ge} = 0$) & ($I_k^{host} \geq 1$))
(23) $G \leftarrow G \cup \{I_k\}$ // put I_k in the good instance set
(24) **Endfor**
(25) Remove identified noise from the data set:
(26) $E \leftarrow E \setminus B$.
(27) Remove a certain portion of good examples from E, where # of removed good examples equals to $\beta \cdot |B|$:
(28) $D \leftarrow \emptyset$
(29) **Repeat** until ($|D| > \beta \cdot |B|$)
(30) $D \leftarrow D \cup \{I_k\}$, $I_k \in G$
(31) $GE \leftarrow GE \cup D$
(32) $E \leftarrow E \setminus D$
(33) **If** the stopping criterion has been satisfied
(34) $E'' \leftarrow E \cup GE$; // add all removed good examples into the final dataset
(35) Exit;
(36) **Else** goto Step (2); // repeat the procedure

Figure 4. The pseudo-code of Partitioning Filter.

Procedure: Noise ($I_k^{host}, I_k^{le}, I_k^{ge}, n$) // n is the number of subsets used to partition set E

(1) **If** ($I_k^{host} \geq 1$ and $I_k^{le} = I_k^{host}$) // only the instance that has at least one host subset and is identified as noise by all its host subsets is considered for further processing.
(2) **If** Majority Scheme:
(3) **If** ($I_k^{ge} \geq [(n - I_k^{le})/2]$), Return (1);
(4) **Else if** Non-objection Scheme:
(5) **If** ($I_k^{ge} \geq (n - I_k^{le})$), Return (1);
(6) **Else** Return (0);

Figure 5. Noise identification (with Majority and Non-objection schemes).

one half of local subsets classify it as noise. In the non-objection scheme, the instance will not be identified as noise unless its classification results from all local subsets are different from its original class 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, i.e., $I_k \in P_i$), as shown in Step (1) of Figure 5, because a classifier usually has a higher accuracy with the instances in its training set.

3.1. Good rule selection

To select good rules from each local subset, a consistency check should be adopted. Intuitively, a rule is considered consistent if it does not cover any negative examples during evaluation. However, in noisy environments, this assumption may not hold, because the noise will likely deny the good rules. Accordingly, 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 the rule r with subset P_i. The criteria we have adopted for good rule detection consist of two parts:

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

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

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

(2) Also, a rule that covers very few examples cannot be selected to evaluate others, and this is ensured by Eq. (2).

$$SubCov(r, P_i) > \mu. \tag{2}$$

With the two criteria above, the parameter α plays an important role in determining the good rule set for each subset P_i. We adopt three schemes to determine this threshold:

- **Adaptive threshold:** The user specifies a value that is relatively close to the actual noise level of the dataset (Assuming the user knows about the noise level).
- **Fixed threshold:** The system assigns a fixed value for α, where the possible values of α are specified by experiences or empirical results.
- **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 $\theta\%$ of this average number. Our experimental results suggest that $\theta = 70$ can usually provide good performances on our evaluated datasets. With the Best-L Rules scheme, the more complicated the dataset, the larger is the number L.

Table 1. Selected good rules from the Car dataset.

Rule ID	Rule	EP
1	(Buying = vhigh) & (Maint = high) → Class 1	20/562
2	(Buying = vhigh) & (Lug_boot = small) & (Safety = med) → Class 1	5/562
3	(Persons = 4) & (Lug_boot = small) & (Safety = med) → Class 2	19/562
4	(Buying = vhigh) & (Persons = more) & (Safety = high)→ Class 2	7/562

3.2. Deduction with the good rule set

Given an instance I_k and a good rule set GR_i from P_i, deduction takes place to evaluate how well GR_i classifies I_k. To this end, we use our previously developed deduction model (Wu, 1998), and meanwhile, some necessary changes have been made.

1) If I_k does not fire any rule in GR_i (which means I_k does not match any rule in GR_i, or no rule in GR_i covers I_k), I_k remains unclassified and GR_i does not cover I_k.
2) If I_k is covered by one or more rules in GR_i, and these rules have the same classification, I_k is assigned with the classification of these rules.
3) If I_k matches more than one rule in GR_i, and there are disagreements among these rules, we calculate the *EP* (*Estimate of Probability*) value of each possible class to determine the final classification of I_k, where the *EP* value of each class, C_l, is determined by the coverage and accuracy of all matched rules that have the class C_l. The deduction model works as follows (Clark and Boswell, 1991; Michalski et al., 1986; Wu, 1995):

The estimate of probability method for handling multiple matches assigns an *EP* value to every class by examining the training set coverage of the satisfied conjunctive rules. The *EP* value for a conjunctive rule r_i which covers the instance I_k in the evaluation set *EV* is defined by Eq. (3).

$$EP(r_i, I_k) = \text{The number of instances covered by } r_i / \text{The number of instances in } EV \tag{3}$$

The *EP* value of a class, C_l, is defined as the probabilistic sum of the *EP* values of all the conjunctive rules that cover I_k and have the class C_l. In the case of two rules r_1 and r_2, the *EP* value is given by Eq. (4). If there are more than two rules for the class C_l, we use this formula recursively. The class with the largest *EP* value is selected as the classification of the instance I_k.

$$EP(C_l, I_k) = EP(r_1, I_k) + EP(r_2, I_k) - EP(r_1, I_k) \cdot EP(r_2, I_k) \tag{4}$$

With the above deduction strategy, the good rule set GR_i only classifies the examples on which GR_i has confidence in its classification. Consequently, exceptions and noise points are likely to be treated in different ways: the exceptions are usually not covered by the good rules, and the noisy examples would likely deny the good rules (covered

but with a different classification). Hence, the error count values of mislabeled instances will receive larger values than non-noisy examples.

Take the results from the Car dataset (Blake and Merz, 1998) as an example. Table 1 provides the selected good rules from two classes (for simplicity, we only present 4 good rules), where the second column shows the details of each rule, and the third column gives the *EP* value of each rule on an evaluation set with 562 examples. Given two instances, I_1 and I_2, as follows, we can easily find that I_1 is covered by both Rule #1 and Rule #4, which come from Class 1 and Class 2 respectively. Since the *EP* value of Rule 1 is larger than that of Rule 4, the induction model will classify instance I_1 as Class 1. For instanceI_2, it simultaneously fires three rules: Rule #1, Rule #2 and Rule #3. Among these three rules, Rule #1 and Rule #2 have the same class label, and the *EP* value of the class *w.r.t.* instance I_2 is 0.044, which is larger than the *EP* value of Class 2 (0.0338). So the induction model will classify I_2 as Class 1 as well.

$$I_1 = \{\text{Buying} = \text{vhigh}; \text{Maint} = \text{high}; \text{Doors} = 3; \text{Person} = \text{more}; \text{Lug\_boot} = \text{med}; \text{Safety} = \text{high}\}$$

$$I2 = \{\text{Buying} = \text{vhigh}; \text{Maint} = \text{high}; \text{Doors} = 4; \text{Person} = 4; \text{Lug\_boot} = \text{small}; \text{Safety} = \text{med}\}$$

3.3. Multi-round execution and stopping criterion

For a relatively large dataset with considerable noise and exceptions, our experimental results in Section 6.5 demonstrate that the first round of noise elimination usually removes about 40% of the noise, and running the same procedure for multiple times will improve the percentage of noise to be identified.

After each round, we remove the identified noise, and also remove a certain number of good examples (this number is less than the number of identified noisy examples, and after noise cleansing has been finalized all removed good examples will be put back to the cleaned dataset). Our experimental results (in Section 6.6) demonstrate that when we keep reducing the noise level, eliminating a small portion of good examples will not have much influence on the system performance. The benefit of eliminating some good examples is that it can shrink the dataset size, and accordingly, 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 new subsets. While removing good examples, the instances with their I_k^{le} and I_k^{ge} both equal to zero are selected as good examples. To maintain the ratio among different classes, the removed good examples will have the same class distribution as the original data set E. In distributed environments, this overall class distribution from all distributed datasets may not be available, so we may need to collect this information separately or simply use the class distribution within a single distributed dataset.

After we repeat the procedure for several times, we need a criterion to determine when to stop. An intuitive strategy is that we can repeat the program until no more examples could be identified as noise. However, this strategy is very time consuming, because at the final stage, it can detect very limited instances in each round. On the

other hand, our experiments have shown that there are situations that in one round, very few noisy examples are identified, but in the next round, a remarkable number of noisy data can be found. This phenomenon indicates that we need to make sure that noise identification is indeed at the final stage before we stop the program. Our stopping criterion is hereby defined as that in T_1 continuous rounds, if the number of identified noise points in each round is less than T_2, the program will stop. In our system, we set $T_1 = 3$ and $T_2 = X \times 0.01$, where X is the number of examples in the training set. Our experimental results show that these thresholds are relatively robust to large datasets (with 10,000–100,000 instances), and in the case of small datasets, we can set $T_2 = 5$ or larger.

3.4. Discussion

Partitioning Filter builds micro-models from instance subsets and then conducts noise handing by unifying the results from all micro-models. This design makes it look similar to existing boosting or classifier ensemble mechanisms. There are, however, at least two distinctions:

- The purpose of boosting and classifier ensemble is to build the "best" theory from all base learners. They have no means to accommodate noise handing into their meta-level results, except the results from the final learner. In contrast, Partitioning Filter nicely utilizes the results from micro-models for noise identification and therefore can fulfill the objective of global noise cleansing through local analysis.
- As Partitioning Filter uses two error count variables, I_k^{le} (local error count) and I_k^{ge} (global error count), to record the behavior of each instance I_k with the good rule set from each subset P_i, further modification can be easily made to deal with different thresholding approaches (in addition to the proposed non-objection and majority schemes). This is, however, not available for boosting or classifier ensemble based methods, as they only output a single "best" learner. Therefore, Partitioning Filter can provide more flexibility for noise handing in general.

4. Noise elimination in distributed datasets

In recent years, database systems have undergone major changes, the most significant of which is that databases are now detached from the computing server and are distributed. In some extreme cases, a distributed database system may consist of thousands of partitions which are located all over the world. On the other hand, regardless of the distribution of the data, integrated view of the databases is imperative for some essential data analysis applications, such as noise identification and association mining, because some knowledge is not available only until we take all distributed partitions into a whole view.

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 other sites might be impossible, but sharing the rules is usually allowable. Accordingly, we can take datasets from different sites as a virtual dataset (E'), and execute noise elimination.

In the first stage, the system executes data partitioning on each distributed dataset by considering its original size. After that, the procedure in Section 3 is adopted to induce and select a good rule set from each subset. Then, the selected good rule sets are used to evaluate all data in the virtual dataset E': while using good rule set GR_i from P_i to evaluate data from other sites, e.g., P_j, only GR_i is passed to P_j. Therefore, no data from any site would leak out. After noise identification on all subsets is completed, each dataset 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 until the stopping criterion in Section 3.3 has been satisfied.

With the above strategy, we can handle noise elimination in distributed datasets. Instead of sharing the original data, only induced rules are shared, and 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.

5. Noise identification errors

By taking the micro model from each subset as a committee member to identify noise, our strategy works in a somewhat similar way to the majority voting and consensus filtering schemes in Brodley and Friedl (1999), where they suggested two types of errors to evaluate the system performance: ER_1 and ER_2. The first type of error (ER_1) occurs when a non-noisy instance is tagged as a noisy example; and the second type of error (ER_2) occurs when a noisy instance (M) is tagged as a correctly labeled example. In this section, we will analyze the probability of adopted threshold schemes (majority and non-objection threshold schemes) in making these two types of errors. Given all instances in dataset E, we first calculate the average value of variable I_k^{host} of all sampled instances and denote it by I_{host}, where I_k^{host} denotes the number of host subsets of I_k (see item 3 of Section 3). This value indicates that on average, how many host subsets an instance has. If our partitioning scheme splits E into disjoint (non-overlapping) subsets, it is obvious that $I_{host} = 1$. However, with other partitioning schemes, e.g., random sampling, the value of this variable could be larger than 1, because some instances may appear in multiple subsets, but some instances may never be sampled (while calculating the value of I_{host}, only sampled instances are taken into consideration). Obviously, with a truly random sampling scheme and a relatively large dataset E, the value of I_{host} would be very close to 1.

As depicted in Figure 4, one premise to determine an instance as noise is that the instance has to be identified as noise by its host subset(s), because a classifier usually has a higher accuracy with the instances in its training set. Hence, each learned classifier would have two accuracies: one in the host subset, and the other in all other subsets. Without losing generality, we assume the errors of the classifier learned from each subset are independent and the same, and ER_1 and ER_2 from P_i in classifying the instances in P_i are denoted by $P_i^h(ER_1)$ and $P_i^h(ER_2)$. The ER_1 and ER_2 from P_i in classifying the instances that are not in P_i are denoted by $P_i^o(ER_1)$ and $P_i^o(ER_2)$. The equations in Eq. (5) likely hold in most situations.

$$P_i^h(ER_1) \leq P_i^o(ER_1); \quad P_i^h(ER_2) \leq P_i^o(ER_2) \tag{5}$$

5.1. *The majority threshold scheme*

Given n subsets, the majority threshold scheme falsely identifies good examples as noise in situations that (1) all the host subsets wrongly classify the instance as noise (as denoted by "A" in Eq. (6)), and (2), no less than half of the other subsets identify the instance as noise (as denoted by "B" in Eq. (6)). The definition of the ER_1 errors in the majority threshold scheme is given in Eq. (6):

$$P_M(ER_1) = \underbrace{P_i^h(ER_1)^{I_{host}}}_{A} \cdot \underbrace{\sum_{j \geq [(n-I_{host})/2]}^{j=[n-I_{host}]} \{P_i^o(ER_1)^j \cdot (1 - P_i^o(ER_1))^{[n-I_{host}-j]} \cdot C_{[n-I_{host}]}^j\}}_{B} \quad (6)$$

where $[x]$ indicates the largest integer that is no larger than x. For simplicity and clarity, we can assume that the sampling is truly random and the dataset is relatively large, then most examples will have only one host subset. Accordingly, Eq. (6) can be simplified as Eq. (7).

$$P_M(ER_1) = P_i^h(ER_1) \cdot \sum_{j \geq [(n-1)/2]}^{j=n-1} \{P_i^o(ER_1)^j \cdot (1 - P_i^o(ER_1))^{(n-1-j)} \cdot C_{(n-1)}^j\} \quad (7)$$

The probability of tagging a truly noisy example as a correctly labeled instance (ER_2 errors) occurs in the situation when any of the host subsets or more than one half of other subsets (excluding the host subsets) falsely tagged the noisy example. With the same assumption that the sampling is truly random with a relatively large dataset, the probability that the majority threshold scheme makes the ER_2 errors is given in Eq. (8).

$$P_M(ER_2) = \underbrace{P_i^h(ER_2)}_{A} + \underbrace{(1 - P_i^h(ER_2)) \cdot \sum_{j > [(n-1)/2]}^{j=n-1} \{P_i^o(ER_2)^j \cdot (1 - P_i^o(ER_2))^{(n-1-j)} \cdot C_{(n-1)}^j\}}_{B} \quad (8)$$

In Eq. (8), "A" indicates the probability that the host subset makes the ER_2 errors. As we have shown in Figure 4, once the host subset made the ER_2 errors (to identify a noisy example as a good instance), there is no chance that this instance could be identified as noise by Majority Threshold Scheme. However, even if the host subset does not make ER_2 errors, once more than one half of subsets made ER_2 errors, this noisy instance could still be determined as a good instance, and this type of errors is represented as "B" in Eq. (8).

If we use the noise identification results from a single subset to determine noise from the whole dataset, the probability of the majority threshold scheme making ER_2 errors is given in Eq. (9).

$$P_{M\_Single}(ER_2) = P_i^h(ER_2) + \left(1 - P_i^h(ER_2)\right) \cdot P_i^o(ER_2) \tag{9}$$

From Eqs. (7)–(9), we can note that if the probability of making ER_1 and ER_2 errors were less than 0.5 respectively, the majority threshold scheme would make fewer errors than noise elimination from a single subset. This may also imply that the larger the number of subsets, the better is the noise identification. Unfortunately, with the increase of the subset number, less information would be contained in each subset, and accordingly, the performance of each learned base classifier tends to be worse, and their ER_1 and ER_2 errors will likely increase. This will inevitably reduce the system performance. In Section 6, we will evaluate and discuss the relationship between the number of subsets and the system performance.

5.2. *The non-objection threshold scheme*

For the non-objection threshold scheme, ER_1 errors occur when all subsets fail to identify an instance correctly. In the case that each subset evaluates the instance independently, the probability that all subsets make ER_1 errors is given in Eq. (10).

$$P_N(ER_1) = P_i^h(ER_1) \prod_{j=1}^{n-1} P_j^o(ER_1) \leq \prod_{j=1}^{n} P_j^o(ER_1) \tag{10}$$

where $P_N(ER_1)$ denotes the probability that the non-objection threshold scheme makes ER_1 errors. Obviously, it is far less than the ER_1 errors from a single subset.

The probability that the system falsely tags a mislabeled instance (an ER_2 error) occurs in situations (1) when the instance is noise but its host subset identifies it as a non-noisy examples (as denoted by "A" in Eq. (11)); or (2) when the instance is noise and its host subset does identify it as a noisy example, but one or more of the other subsets fails to identify it as noise (as denoted by "B" in Eq. (11)). That is, the non-objection threshold scheme makes ER_2 errors only if one or more of the subsets make ER_2 errors. This probability equals to one minus the probability that none of the subsets makes the ER_2 errors. Given subset P_i, $1 - P_i^o(ER_2)$ is the probability that it does not make ER_2 errors with the instances that are not in P_i, and the probability that the non-objection threshold scheme makes ER_2 errors is given in Eq. (11).

$$P_N(ER_2) = \underbrace{P_i^h(ER_2)}_{A} + \underbrace{(1 - P_i^h(ER_2)) \cdot \{1 - \prod_{j=1}^{n-1} (1 - P_j^o(ER_2))\}}_{B} \tag{11}$$

One can find that, if both n and $P_i^o(ER_2)$ are relatively large, $\prod_{j=1}^{n-1} (1 - P_j^o(ER_2))$ would be close to zero, and the value of $P_N(ER_2)$ would likely be equal to 1. In this situation, the non-objection threshold scheme will make severe ER_2 errors. Note that, if we use

the identification results from a single subset to determine noise from the whole dataset, the probability that the non-objection threshold scheme makes ER_2 errors is denoted by Eq. (12). As we can see, Eq. (12) is the same as Eq. (9), because when using the identification results from a single subset to determine noise, there is no difference between the majority and the non-objection schemes.

$$P_{N\_Single}(ER_2) = P_i^h(ER_2) + (1 - P_i^h(ER_2)) \cdot P_i^o(ER_2) \tag{12}$$

Equations (11) and (12) indicate that the non-objection threshold scheme has a much higher probability to make ER_2 errors in comparison with using a single subset for noise identification. Actually, the results from Section 6 will demonstrate that this scheme is usually too conservative and hardly eliminates much noise from noisy datasets, especially when the number of subsets is large.

In comparison with two threshold schemes for noise identification, we have the following observations:

(1) The majority threshold scheme tends to make more ER_1 errors. It is more aggressive than the non-objection scheme in identifying noisy data, but consequently, it also has a high probability to remove non-noisy instances.
(2) The non-objection threshold scheme is much safer in keeping good examples, but in many situations, it is too safe to remove even noisy instances.
(3) In the case that the system cannot afford many ER_1 errors (when the size of the dataset is relatively small or the redundancy in the dataset is low, ER_1 errors can critically affect the classification accuracy), the non-objection threshold scheme might be better than the majority scheme. However, in the case that the dataset is relatively large or contains much redundancy, the majority scheme is a better choice. The experimental results in Section 6 will demonstrate that in this case, the majority scheme can achieve a better performance in identifying noisy examples and improving the classification accuracy. Actually, in our system, we set the majority threshold scheme as the default mechanism to identify noise.
(4) Instead of adopting either the majority or non-objection threshold scheme, one can carefully select a threshold so that the merits of both the majority and non-objection schemes can be combined for a better result. However, our experimental results show that the best value of such a threshold varies from dataset to dataset. In this paper, we will consider these two threshold schemes (majority and non-objection).

6. Experimental evaluations

6.1. Experiment settings

In this section, we design a series of experiments to test our system performance on various factors. To construct classification rules and base classifiers, C4.5 rules (Quinlan, 1993) is adopted in our system. We have evaluated our noise elimination strategy extensively on both synthetic data from IBM (IBM Synthetic Data) and real-world datasets from UCI repository (Blake and Merz, 1998), and the sizes of all adopted datasets vary from several hundred examples to more than sixty thousands. A summary of these datasets is given in Table 2.

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 an $x \cdot 100\%$ chance to be corrupted and mislabeled as Y, so does an instance of class Y. We use this method because in reality, only certain types of classes are likely to be mislabeled. Meanwhile, with this scheme, 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 instead of corrupting the class "Neither" which contains 50% of the total instances, we corrupt the classes EI and IE, each of which contains 25% of the total instances). Meanwhile, we only report the value x (not the actual noise level in the dataset) in all tables and figures below.

For each experiment, we perform 10 times 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 (using C4.5). In the figures and tables below, the original dataset means the noise corrupted training set, and the classification accuracy of the original dataset represents the accuracy of the classifier learned from the noise corrupted training set (without noise elimination scheme).

For a better evaluation, we adopt three factors to evaluate the system performance: ER_1, ER_2 and Noise Elimination Precision (NEP). ER_1 occurs when a non-noisy instance is tagged as a noisy example; and ER_2 occurs when a noisy instance is tagged as a correctly labeled example. Their definitions are given by Eq. (13).

Table 2. Datasets used in our experiments.

Dataset name	Total instances	Number of symbolic attributes	Number of continuous attributes	Number of classes
Adult	48842	8	6	2
Car	1728	6	0	4
Connect-4	67557	42	0	3
IBM-Synthetic	12000	3	6	4
Krvskp	3196	36	0	2
Splice	3190	61	0	3
Mushroom	8124	22	0	2
WDBC	569	0	30	2
Credit-app	690	9	6	2
Nursery	12960	8	0	5
Monk-3	432	6	0	2
Tic-tac-toe	958	9	0	2
Sick	3772	22	7	2
Contraceptive method choice (CMC)	1473	7	2	3

Table 3. Noise elimination results from the majority threshold scheme (Mushroom dataset).

Noise (%)	Filtered (%)	Noise & filtered (%)	*NEP*	$P(ER_1)$	$P(ER_2)$
5	6.211	4.9800	0.801	0.0131	0.0104
10	11.060	9.8890	0.894	0.0160	0.0130
15	15.715	14.749	0.921	0.0231	0.0195
20	20.434	19.156	0.938	0.0294	0.0472
25	25.377	24.130	0.943	0.0358	0.0695
30	28.879	28.336	0.926	0.0825	0.1072
35	33.057	22.026	0.681	0.1584	0.2104
40	41.945	26.843	0.662	0.2289	0.2949

$$P(ER_1) = \frac{|F \cap G|}{|G|} \qquad P(ER_2) = \frac{|\tilde{F} \cap M|}{|M|} \qquad NEP = \frac{|F \cap M|}{|F|} \tag{13}$$

where $|\Delta|$ denotes the number of examples in set Δ. 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 noisy examples.

6.2. *Threshold schemes for noise identification*

We use two threshold schemes to identify noise: majority and non-objection. To check which method is more efficient with our system, we compare their performance in noise elimination and in classification accuracy improvement. 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 6.4). To select good rules, we use an adaptive threshold scheme (in Section 3.1): when processing a dataset with a noise level of x, we randomly select a value for α with the constraint in Eq. (14).

$$\alpha = \begin{cases} 1 - Random\{[(x - 0.1), (x + 0.1)]\}; & (x - 0.1) \geq 0 \\ 1 - Random\{[0, (x + 0.1)]\}; & (x - 0.1) < 0 \end{cases} \tag{14}$$

We tabulate the results of noise elimination in Tables 3–6 where the first column indicates the noise level, the second column denotes the percentage of eliminated instances, and the third column gives the ratio between the number of correctly identified noisy examples and the total number of instances in the dataset. We also depict the accuracy improvements in Figures 6 and 7, where the x-axis denotes the noise level and the y-axis represents the classification accuracy of the classifier trained from the original/cleaned dataset (and evaluated with the test set).

From Tables 3–6, 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 Figures 6 and 7, 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, as long as this scheme can keep its ER_1 errors at an acceptable level. From Tables 2 and 4, 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, the classifier learned from the dataset that is processed by the majority threshold scheme receives more improvement on its classification accuracy. From Figures 6 and 7, take the noise level at 20% as an example. In comparison with the non-objection scheme, the classification accuracy improvement from the majority scheme has about 2% and 4% more improvement for the Mushroom and Adult datasets respectively. When the noise level goes higher (over 30%), the accuracy improvements from both schemes tend to be worse, because with such an environment, the noise and non-noisy instances are more likely to be confused. The induction algorithm is going to be less efficient. However, the results demonstrate that both schemes still receive a

Table 4. Noise elimination results from the non-objection threshold scheme (Mushroom dataset).

Noise (%)	Filtered (%)	Noise & filtered (%)	*NEP*	$P(ER_1)$	$P(ER_2)$
5	5.721	4.771	0.873	0.0082	0.0075
10	10.287	9.787	0.959	0.0078	0.0087
15	15.315	14.267	0.945	0.0107	0.0582
20	18.677	18.181	0.961	0.0119	0.0965
25	22.967	22.347	0.965	0.0123	0.1231
30	25.592	24.737	0.951	0.0177	0.1754
35	12.961	11.205	0.867	0.0292	0.5214
40	16.447	14.121	0.833	0.0426	0.6955

Table 5. Noise elimination results from the majority threshold scheme (Adult dataset)

Noise (%)	Filtered (%)	Noise& filtered (%)	*NEP*	$P(ER_1)$	$P(ER_2)$
5	6.417	3.124	0.512	0.0335	0.3527
10	10.914	6.748	0.619	0.0463	0.3252
15	15.463	10.506	0.687	0.0563	0.2996
20	20.302	14.596	0.718	0.0725	0.2681
25	23.472	18.082	0.764	0.0738	0.2723
30	30.719	23.015	0.751	0.1107	0.2328
35	35.923	26.179	0.734	0.1488	0.2223
40	41.762	30.061	0.702	0.2629	0.2486

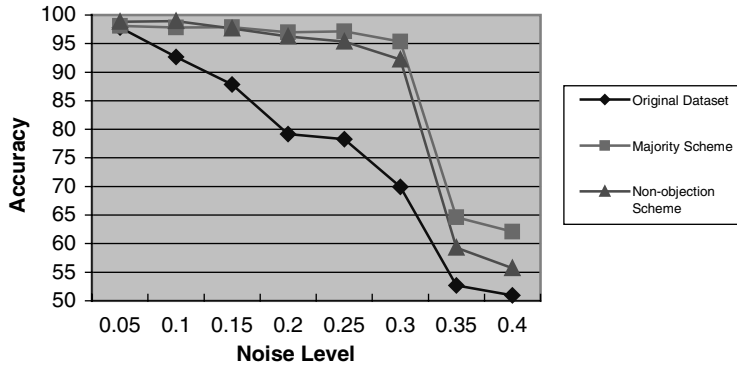

Figure 6. Classification accuracies on Mushroom dataset (5 subsets).

remarkable classification improvement in comparison with the original noise corrupted training dataset (without any noise elimination). In the sections below, unless specified otherwise, we will use the majority scheme for noise identification.

6.3. Good rule selection schemes

To identify noise, we heavily rely on the good rule set from each subset. Therefore, a good rule selection strategy should be designed and evaluated. As we introduced in Section 3.1, three schemes have been proposed. In this subsection, we compare the system performance with these schemes: (1) the adaptive threshold scheme, with Eq. (14) for determining the value for α; (2) the fixed threshold scheme, with three typical fixed values $\alpha = 0.9, 0.75$ and 0.6; and (3) the Best-L rules scheme, with $\theta = 70$ which usually generates $L = 10$ and $L = 20$ for the Mushroom and Adult datasets respectively. Figures 8 and 9 present the experimental results from the Mushroom and Adult datasets.

From Figure 8, one can easily conclude that the adaptive threshold scheme is the best in almost all situations. When the noise level is worse than $1-\alpha$, the

Table 6. Noise elimination results from the non-objection threshold scheme (Adult dataset)

Noise (%)	Filtered (%)	Noise & filtered (%)	*NEP*	$P(ER_1)$	$P(ER_2)$
5	2.081	1.743	0.795	0.0047	0.6643
10	4.801	3.963	0.831	0.0093	0.6040
15	7.281	6.665	0.891	0.0095	0.5686
20	9.397	8.475	0.902	0.0115	0.5664
25	12.258	11.261	0.918	0.0133	0.5495
30	14.031	13.070	0.934	0.0137	0.5644
35	14.394	13.289	0.943	0.0121	0.6092
40	15.167	14.395	0.946	0.0128	0.6563

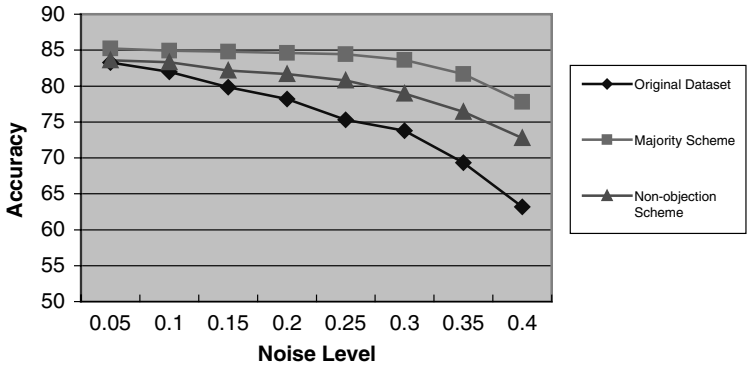

Figure 7. Classification accuracies on Adult dataset (5 subsets).

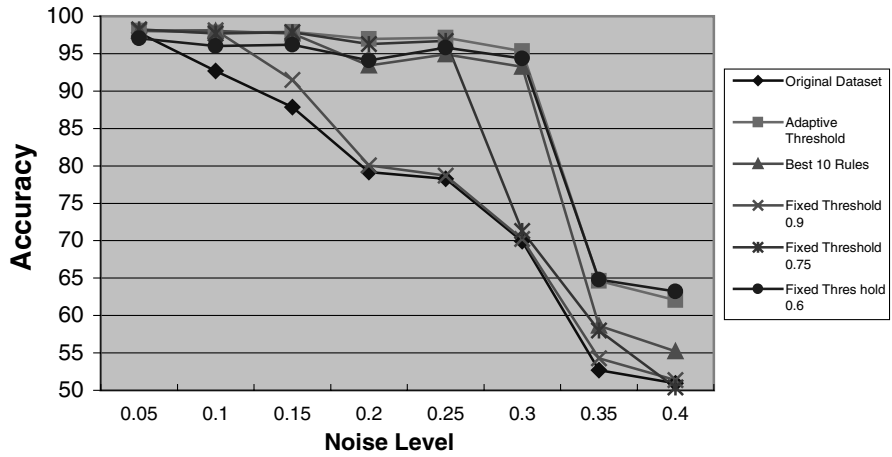

Figure 8. Good rule selection schemes (Mushroom dataset, 5 subsets, majority threshold scheme).

performance from the fixed threshold scheme is less 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 $\alpha = 0.9$ and 0.75). Our analysis shows that when the noise level x is less than the threshold $1 - \alpha$, the system tends to take more insignificant rules into the good rule set, and as a result, more good examples are identified as noise. Take $\alpha = 0.6$ as an example. The performance from this dataset with 10% noise is worse (about 2%) than the adaptive threshold scheme; on the other hand, when the noise level is higher than α, the good rules' accuracy would tend to be less than α (the noise corrupts the good rules' accuracy), 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 Figures 8 and 9, the results from $\alpha = 0.6$ are acceptable to some degree, even when worse performances have been found from the lower noise levels. Therefore,

in the case that the user has no idea about the noise level, we may need to 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 imagine 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.

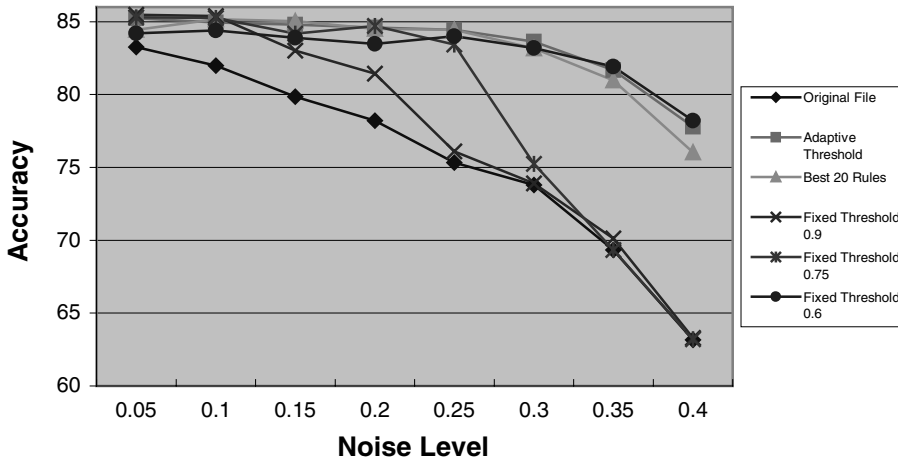

Figure 9. Good rule selection schemes (Adult dataset, 5 subsets, majority scheme).

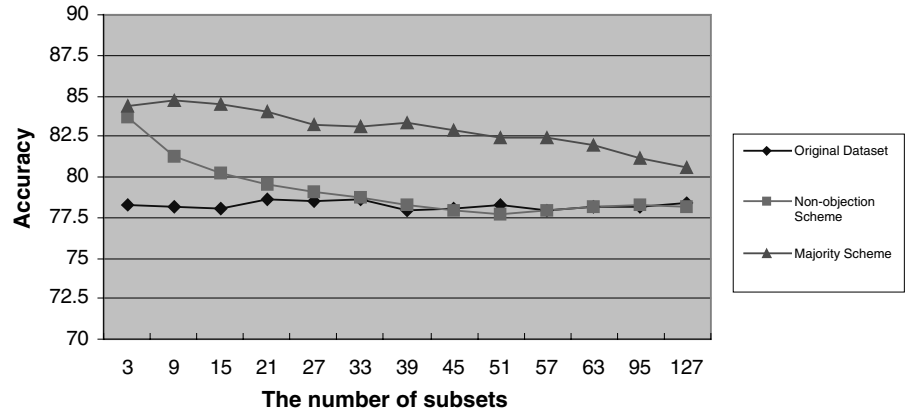

Figure 10. Classification improvement with the number of subsets (Adult dataset, 20% noise level).

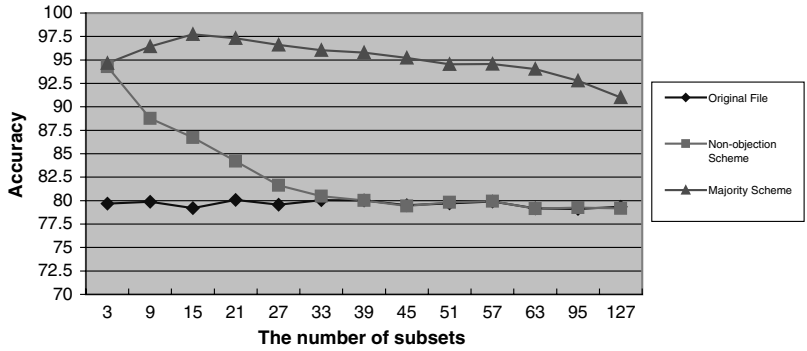

Figure 11. Classification improvement with the number of subsets (Mushroom dataset, 20% noise level).

6.4. *Number of subsets on system performance*

Our approach uses the classifier learned from the minor set (one single subset) to evaluate the instances in 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 function 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 the classification improvement and noise elimination are given in Figures 10 and 11 and Table 7 respectively.

Table 7. Noise elimination results with various numbers of subsets (Adult dataset).

	$P(ER_1)$		$P(ER_2)$	
Subsets	Non-objection	Majority	Non-objection	Majority
3	0.0303	0.0582	0.4265	0.3067
9	0.0096	0.0788	0.6374	0.2558
15	0.0040	0.0700	0.7232	0.2882
21	0.0011	0.1391	0.7744	0.1895
27	0.0009	0.1504	0.8244	0.2355
33	0.0005	0.1627	0.8709	0.2464
39	0.0002	0.1462	0.8916	0.2118
45	0.0001	0.1502	0.9475	0.2643
51	0	0.1726	0.9571	0.2424
57	0	0.1794	0.9675	0.2488
63	0	0.1869	0.9781	0.2947
95	0	0.1648	0.9971	0.5120
127	0	0.1729	0.9992	0.6594

Table 7 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 data set would have very limited improvement (sometimes no improvement), as demonstrated in Figures 10 and 11. The classification accuracy of the non-objection scheme decreases rapidly with the increasing number of subsets. However, the results in Table 7 indicate that the majority scheme sacrifices a small portion of ER_1 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 in Table 7 with 3 and 33 subsets respectively, the ER_1 errors of the majority scheme increases from 5.82% to 16.27%, but we still receive remarkable improvement on the classification accuracy.

The experimental results from Figures 10 and 11 demonstrate that by using the majority threshold scheme, the number of subsets plays a very limited role in affecting the system performance. 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 the noise), and meanwhile, the weaker is the ability of each committee member (since with the increase of the subset number, less information is contained in each subset). From Figures 10 and 11, 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 classification accuracy increases with the increase of the subset number, as shown in Figure 11 with 3 to 15 subsets, but after a certain number it begins to drop). However, the experiments from this subsection indicate that even in the extreme case, e.g., 127 subsets, our strategy can still attain good classification results from both the Mushroom and Adult datasets. In most of our experiments in next sections, we use 5 subsets.

6.5. *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

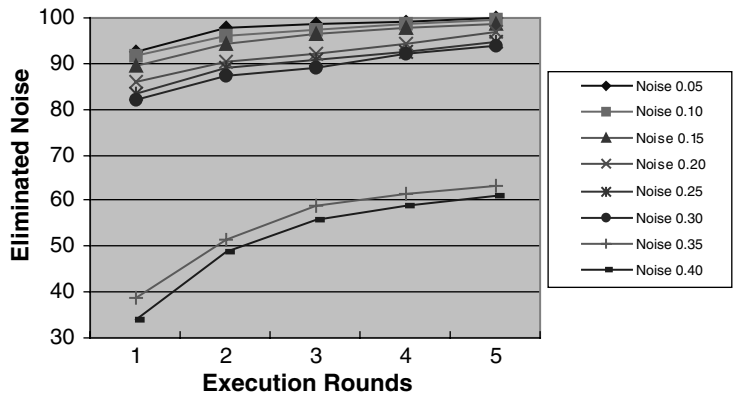

Figure 12. Noise elimination results from multiple rounds of execution (Mushroom dataset, 5 subsets, Majority and Best-10 rules schemes).

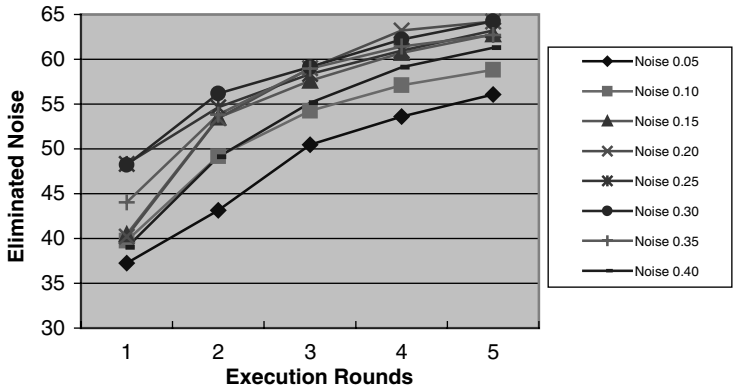

Figure 13. Noise elimination from multiple rounds of execution (Adult dataset, 5 subsets, Majority and Best-20 rules schemes).

ask whether it is 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 demonstrate the results in Figures 12 and 13, where the x-axis denotes the number of execution rounds and the y-axis represents the percentage of identified noise until this round. For the Mushroom dataset, 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 is necessary to execute the same procedure for more than once. As shown in Figure 12, when the noise level is 35% or 40%, the first round eliminates only 35% of the noise, and after we repeat the procedure for 5 rounds, this number goes up to about 60%. The results from Figure 13 are more convincing: 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, 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 3.3, another possible empirical operation to stop noise elimination is to execute the procedure for a certain number of rounds.

6.6. *Removing good examples*

While repeating the same procedure for multiple rounds, in addition to removing identified noise, some good examples are also removed. Assuming we eliminate $|B|$ noisy examples in the current round (where $|\Delta|$ means the number of instances in set Δ), we also remove $\beta \times |B|$ good examples ($\beta \in [0, 1]$). The benefit of this scheme has been discussed in Section 3.3. However, intuitively the more the good examples the higher is the classification accuracy of the learned classifier. So it seems that we could sacrifice the classifier's accuracy by removing good examples. Nevertheless, our experimental

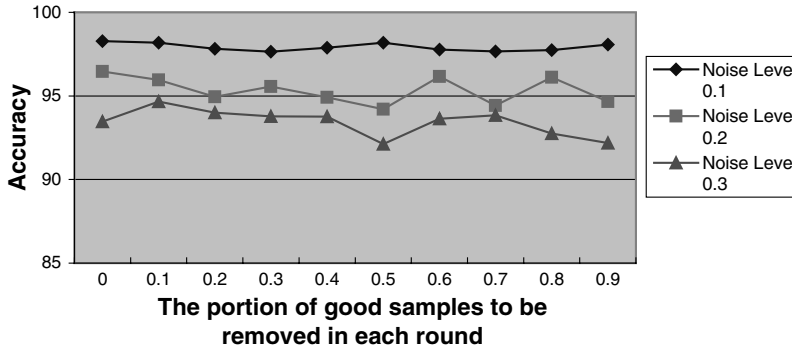

Figure 14. Classification accuracy with different portions of good examples to be removed in each round (Mushroom dataset, 5 subsets, Majority and Best-10 rules schemes).

results demonstrate that eliminating a small portion of good examples usually has less impact on the system performance. Actually, it may benefit noise elimination occasionally, as shown in Figure 14, where the x-axis denotes the value of β and the y-axis indicates the classification accuracy.

From the experimental results in Figure 14, we can find that when the noise level in the dataset is less than 50%, removing a small portion ($\beta \in [0, 1]$) of good examples does no impact much to the system performance, and the classification accuracy is still somewhat a horizontal line (if the noise level in the dataset is larger than 50%, such a removing procedure could possibly increase the noise level in the dataset). One possible reason is that induction algorithms usually use some heuristic strategies to construct rules, and changing good examples or bad examples in a certain range may get a different result (either better or worse). This indicates that as long as the noise level in the dataset declines, we can eliminate a small portion of good examples and make the dataset shrink to a smaller size quickly. We set $\beta = 0.5$ in our experiments.

To remove good examples, we also run the risk that noise is taken as good instances and is removed (which is different from ER_1 errors). We have tabulated the average percentage of noise that is identified as good examples and removed at each noise level, and demonstrate the results in Table 8. We can find that the percentage of noisy instances in removed good examples is very small.

6.7. *More experimental results*

In addition to the results from the Mushroom and Adult datasets, we have evaluated our approach on 12 other datasets (with 5 subset and Best L rules on all datasets), and the results are shown in Tables 9–11. Also, for comparative studies, we have implemented the Classification Filter proposed by Gamberger et al. (1999), where the given training

Table 8. Portion of noise eliminated as good examples at different noise levels

Noise Level	5%	10%	20%	30%	40%
Portion of noise in good examples (%)	0.124	0.173	0.654	1.233	6.173

set is partitioned into 10 non-overlapping subsets (as recommended by these authors). The classifier H_y trained from the aggregation of any 9 subsets (using C4.5 rules) is used to identify noise in the complementary one subset. The instances that are incorrectly classified by H_y are then identified as noise. In Tables 9– 11, the first column indicates the noise levels x, and all other columns denote the accuracies from different datasets, where OG indicates the classification accuracy of the classifier learned from the original noisy training set (without any noise elimination), CF represents the accuracy from the Classification Filter, and PF denotes the results from our approach (Partitioning Filter). The texts in boldface indicate that Partition Filter (*PF*) has better performance than Classification Filter (*CF*).

From the results in Tables 9–11, the system performance can always improve with noise elimination. With all twelve datasets, noise elimination can contribute from 1% to over 20% to the classification accuracy improvement, varying from different 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 7 out of 12 datasets. However, when the noise level becomes higher, the performance from CF decreases rapidly. When the noise level is at 35%, for example, CF outperforms PF on only 3 out of 12 datasets. Further analysis provides an answer for this phenomena: since CF learns its classifiers from the major sets and uses them to identify mislabeled instances, 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 trained classifiers tend to make

Table 9. Experimental results with the classification accuracy (Krvskp, Car, Nursery and WDBC, 5 subsets, Best L rules scheme).

	Krvskp (%)			Car (%)			Nursery (%)			WDBC (%)		
Noise (%)	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**
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**
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**
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**
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**

Table 10. Experimental results with the classification accuracy (Splice, Credit-app, Connect-4 and Tic-tac-toe, 5 subsets, Best L rules scheme).

	Splice (%)			Credit-app (%)			Connect-4 (%)			Tic-tac-toe (%)		
Noise (%)	OG	CF	PF	OG	CF	PF	OG	CF	PF	OG	CF	PF
5	89.1	92.6	91.8	81.9	85.3	**85.6**	73.2	75.8	75.7	83.5	83.9	**83.8**
15	85.6	92.1	91.4	73.7	84.6	**86.7**	68.2	74.7	**75.1**	76.3	79.2	**78.8**
25	82.1	91.2	89.7	66.7	83.4	**85.2**	61.6	71.8	**72.5**	69.1	72.5	**73.4**
35	77.6	89.1	86.4	61.5	80.5	**83.9**	55.8	68.8	**69.7**	61.8	62.6	**64.7**
40	75.5	87.4	80.9	58.2	79.1	**81.4**	51.6	66.5	**67.9**	57.8	61.1	**62.7**

Table 11. Experimental results with the classification accuracy (Monks-3, IBM-Synthetic, Sick and CMC, 5 subsets, Best L rules scheme).

	Monks-3 (%)			IBM-Synthetic (%)			Sick (%)			CMC (%)		
Noise (%)	OG	CF	PF	OG	CF	PF	OG	CF	PF	OG	CF	PF
5	96.8	99.2	97.3	88.5	92.7	91.8	97.0	98.1	98.1	49.2	52.2	**53.5**
15	89.2	98.0	96.9	83.6	91.4	90.9	93.2	97.6	**97.9**	48.8	52.5	**52.8**
25	82.7	91.9	90.8	76.4	89.2	**90.3**	91.4	96.3	95.8	44.9	49.3	**49.7**
35	67.3	79.2	**80.1**	63.7	83.6	80.2	83.7	95.5	94.7	42.8	47.1	**47.8**
40	63.1	71.4	67.5	53.1	63.7	**66.3**	77.5	88.6	86.9	43.3	46.0	**47.6**

Table 12. Execution time comparisons (Mushroom dataset).

	Execution time at different noise levels (seconds)				
Methods	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

more and more mistakes, which will in turn either falsely classify some good instances as noise or keep noisy examples as good instances. Our approach selects good rules, and the increase of the noise has less impact on the system performance. Moreover, instead of fully relying on any single classifier (as the CF scheme does), we take each subset as a committee member and consider the identification results 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 12 shows the execution times of the two methods at various noise levels (on the Mushroom dataset), where the first column denotes different methods, and all other columns indicate the average execution times of different methods at each noise levels. We have used a PC with Pentium 4, a 2 GHz processor, and 512 MB memory to implement both methods. One can find that, the PF scheme is much better than the CF approach in terms of time efficiency. When the noise level is 0, PF is about 3.4 times faster than CF, and when the noise level increases to 40%, PF is about 40 times faster than CF. Our results from other datasets indicate that on average, PF is about 10 times faster than CF. Actually, the larger the dataset, the higher the noise level, the more efficient is PF compared to CF. Part of the reason is that the time complexity of C4.5 rules nonlinearly increases with the noise level and the number of training instances. If adopting different inductive algorithms, the impact of noise and the number of instances may be less significant, but we believe that the PF scheme will still be much more efficient than CF.

6.8. Discussion

Our approach is related to the Classification Filter (CF) scheme proposed in Gamberger et al. (1999). However, there are three key distinctions between them even though they

Table 13. Noise elimination comparisons (Car dataset).

Noise (%)	Partitioning filter (PF)					Classification filter (CF)				
	Filtered	Filtered & Noise	*NEP*	ER_1	ER_2	Filtered	Filtered & Noise	*NEP*	ER_1	ER_2
5	0.1242	0.0412	0.338	0.087	0.198	0.1592	0.05	0.315	0.115	0.026
15	0.1856	0.105	0.570	0.093	0.218	0.2528	0.1231	0.487	0.150	0.083
25	0.2623	0.1816	0.708	0.105	0.237	0.3652	0.2011	0.553	0.215	0.155
35	0.3243	0.2216	0.685	0.147	0.272	0.4305	0.2489	0.581	0.262	0.186
40	0.3328	0.2095	0.636	0.196	0.438	0.4706	0.252	0.535	0.348	0.325

both adopt a partitioning scheme: (1) CF learns the base classifiers from the aggregation of any $n - 1$ subsets (the major set) to identify noise from the complementary (excluded) subset, while our approach learns classifiers 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 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 in which it has the confidence. Consequently, our approach is more efficient in handling large and very noisy datasets.

In addition, further analysis shows another disadvantage of the CF scheme: the results of the CF scheme critically depend on the performance of adopted learning algorithms. Assuming an inductive algorithm has a $\omega\%$ classification accuracy with a noise-free dataset E'', the CF scheme will inevitably identify about $100\%-\omega\%$ of the instances as noise, even if there is no noise in E''. The reason is that it unconditionally trusts each base classifier, and any instance that is incorrectly classified by any base classifier is directly identified as noise. This strategy obviously neglects the fact that inductive algorithms cannot work perfectly, even with noise-free datasets. Our experimental results indicate that the CF scheme is usually more aggressive than our approach, eliminates more instances and makes more ER_1 errors. As shown in Tables 13–15, where the first column indicates the noise level, the column "filtered" indicates the percentage of eliminated

Table 14. Noise elimination comparisons (Connect-4 dataset).

Noise (%)	Partitioning filter (PF)					Classification filter (CF)				
	Filtered	Filtered & Noise	*NEP*	ER_1	ER_2	Filtered	Filtered & Noise	*NEP*	ER_1	ER_2
5	0.2025	0.035	0.172	0.176	0.243	0.2764	0.038	0.138	0.249	0.161
15	0.2463	0.1025	0.4	0.178	0.266	0.3609	0.1144	0.317	0.286	0.181
25	0.2587	0.1452	0.561	0.146	0.355	0.4245	0.1767	0.416	0.320	0.216
35	0.3546	0.2097	0.592	0.209	0.322	0.473	0.2352	0.497	0.344	0.239
40	0.3373	0.210	0.691	0.199	0.418	0.5086	0.2564	0.501	0.394	0.288

Table 15. Noise elimination comparisons (Monks-3 dataset).

	Partitioning filter (PF)					Classification filter (CF)				
Noise (%)	Filtered	Filtered & Noise	*NEP*	ER_1	ER_2	Filtered	Filtered & Noise	*NEP*	ER_1	ER_2
5	0.0598	0.0376	0.771	0.023	0.331	0.0638	0.0558	0.883	0.008	0.011
15	0.1561	0.1159	0.752	0.048	0.293	0.1857	0.1557	0.839	0.036	0.046
25	0.1631	0.1407	0.862	0.029	0.408	0.3057	0.1986	0.653	0.141	0.167
35	0.1372	0.1228	0.904	0.022	0.635	0.4039	0.2494	0.619	0.233	0.258
40	0.0908	0.0776	0.764	0.021	0.728	0.4373	0.2556	0.587	0.308	0.367

instances, and the column "filtered & noise" represents the percentage of identified true noise (in comparison with the whole dataset). From the results in these three datasets (Car, Connect-4, and Monks-3), one can find that CF tends to make more ER_1 errors and less ER_2 errors. This might be another reason that it occasionally works better than PF on some datasets. Some datasets may contain relatively large redundancy, and removing a small portion of non-noisy examples as noise will not have much influence on the classification accuracy. Actually, removing a certain portion of exceptions (or outliers) will most likely increase the classification accuracy on some datasets.

We do not claim that our approach outperforms the CF scheme in all situations, and we have noticed that the CF scheme has achieved better performances on some datasets (e.g., Splice, Monks-3 and Sick). Actually, when the noise level is low, the classifiers learned from the major set are usually better than the classifiers from the minor set. This is why scaling-up inductive learning algorithms usually have less accuracy than the classifiers learned from the whole dataset (Chan, 1996; Provost et al., 1999). However, as we have indicated, the merit of our method comes from the fact that it can deal with large, distributed datasets, where most existing mechanisms are inadequate or have less efficiency. We believe from this point of view, the method proposed in this paper provides a new solution in handling large, distributed noisy datasets.

7. Conclusions

Most inductive learning algorithms critically depend on the quality of the training data to achieve high performances, and an effective noise cleansing model is often regarded as one of the major steps in real-world data mining applications. Existing endeavors often try to induce noise identifiers from the whole massive data, and are therefore less efficient in handling large or distributed datasets. In this paper, we have presented a Partitioning Filter approach which nicely incorporates both local and global procedures for noise cleansing from large, distributed datasets. Given a dataset E, we first partition it into N subsets. For any local subset P_i, a set of good rules are induced and selected to classify 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 set from each local subset. Due to the fact that exceptions usually do not fire the good rules and noise will likely deny 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.

The novel feature that distinguishes the proposed framework from existing research efforts is threefold: First, our strategy conducts noise elimination from a local subset which is more reasonable to scale up to large or distributed datasets. Second, unlike other strategies where induced noise classifiers are directly used to identify noise, we take the micro models from all local subsets as a committee for noise identification, and each committee member only handles the data it has the confidence. Third, while existing multi-round noise elimination mechanisms remove the identified noise only, we remove both identified noisy examples and good instances in each round. Experimental evaluations and comparative studies have shown that our proposed approach is effective and robust to identify noise and improve the classification accuracy.

References

Aha, D., Kibler, D., and Albert, M. 1991. Instance-based learning algorithms. Machine Learning, 6(1):37–66.

Blake, C.L. and Merz, C.J. 1998. UCI Repository of Machine Learning Databases.

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

Brodley, C.E. and Friedl, M.A. 1996. Identifying and eliminating mislabeled training instances. Proc. of 13th National Conf. on Artificial Intelligence, pp.799–805.

Brodley, C.E. and Friedl, M.A. 1999. Identifying mislabeled training data. Journal of Artificial Intelligence Research, 11:131–167.

Bruha, I. and Franek, F. 1996. Comparison of various routines for unknown attribute value processing the covering paradigm. IJPRAI 10(8):939–955

Cestnik, B., Kononenko, I., and Bratko, I. 1987. ASSISTANT 86: A knowledge elicitation tool for sophisticated users, Proc. of 2nd European Working Session on Learning, Sigma Press, 1987. pp. 31–45.

Cendrowska, J. 1987. Prism: An algorithm for inducing modular rules. International Journal of Man-Machines Studies, 27:349–370.

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

Clark, P. and Niblett, T. 1989. The CN2 induction algorithm. Machine Learning, 3(4):261–283.

Clark, P. and Boswell, R. 1991. Rule induction with CN2: Some recent improvement. Proc. of 5th ECML, Berlin, Springer-Verlag.

Dietterich, T. 2000. Ensemble methods in machine learning. In Lecture Notes in Computer Science Vol. 1867, J. Kittler and F. Roli, (Eds.), Springer, Berlin: pp. 1–15.

Gamberger, D., Lavrac, N., and Groselj, C. 1999. Experiments with noise filtering in a medical domain. Proc. of 16th ICML Conference, San Francisco, CA, pp. 143–151.

Gamberger, D. Lavrac, N., and Dzeroski, S. 2000. Noise detection and elimination in data preprocessing: Experiments in medical domains. Applied Artificial Intelligence, 14:205–223.

Grzymala-Busse, J.W. and Hu, M. 2000. A comparison of several approaches to missing attribute values in data mining. Rough Sets and Current Trends in Computing, pp. 378–385.

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

Friedman, J.H. 1977. A recursive partitioning decision rule for nonparametric classification. IEEE Transaction on Computers, 26(4):404–408.

Hall, L., Bowyer, K., Kegelmeyer, W., Moore, T., and Chao, C. 2000. Distributed learning on very large data sets, KDD-00 Workshop on Distributed and Parallel Knowledge Discovery, pp. 79–84.

Holte, R.C. 1993. Very simple classification rules perform well on most commonly used datasets. Machine Learning, 11.

Huang, C.C. and Lee, H.M. 2001. A grey-based nearest neighbor approach for predicting missing attribute values. Proc. of 2001 National Computer Symposium, Taiwan, NSC-90-2213-E-011-052.

IBM Synthetic Data. IBM Almaden Research, Synthetic classification data generator, http://www.almaden.ibm.com/software/quest/Resources/datasets/syndata.html#classSynData.

John, G.H. 1995. Robust decision trees: Removing outliers from databases. Proc. of the First International Conference on Knowledge Discovery and Data Mining, AAAI Press, pp. 174–179.

Kononenko, I., Bratko, I., and Roskar, E. 1984. Experiments in automatic learning of medical diagnostic rules, Technical Report, Jozef Stefan Institute, Ljubljana, Yugoslavia.

Kubica, J. and Moore, A. 2003. Probabilistic noise identification and data cleaning. Proc. of ICDM, FL, USA

Lewis, D. and Catlett, J. 1994. Heterogeneous uncertainty sampling for supervised learning. Proc. of the 11th ICML Conference, NJ, Morgan Kaufmann: pp. 148–156.

Li, Q. Li, T., Zhu, S., and Kambhamettu, C. 2002. Improving medical/biological data classification performance by wavelet preprocessing. Proc. of International Conference on Data Mining (ICDM 2002), Japan.

Michalski, R.S., Mozetic, I., Hong, J., and Lavrac, N. 1986. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. Proceedings of AAAI, pp. 1041–1045.

Oak, N. and Yoshida, K. 1993. Learning regular and irregular examples separately. Proc. of IEEE International Joint Conference on Neural Networks, pp. 171–174.

Oak, N. and Yoshida, K. 1996. A noise-tolerant hybrid model of a global and a local learning model. Proc. of AAAI-96 Workshop: Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithm, pp. 95–100.

Provost, F., Jensen, D., and Oates, T. 1999. Efficient progressive sampling. Proc. of the 5th ACM SIGKDD, pp. 23–32.

Provost, F. and Kolluri, V. 1999. A survey of methods for scaling up inductive algorithms. Data Mining and Knowledge Discovery, 3(2):131–169.

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

Quinlan, J.R. 1989. Unknown attribute values in induction. Proceedings of the 6th International Workshop on Machine Learning, pp. 164–168.

Quinlan, J.R. 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA.

Schapire, R.E. 1990. The strength of weak learnability. Machine Learning, 5(2):197–227.

Shapiro, A. 1983. The role of structured induction in expert systems, Ph.D Thesis, University of Edinburgh.

Skalak, D. 1994. Prototype and feature selection by sampling and random mutation hill climbing algorithms, Proc. of 11th ICML Conference, New Brunswick, NJ. Morgan Kaufmann, pp. 293–301.

Srinivasan, A., Muggleton, S., and Bain, M. 1992. Distinguishing exception from noise in non-monotonic learning. Proc. of 2nd Inductive Logic Programming Workshop, pp. 97–107.

Teng, C.M. 1999. Correcting noisy data. Proc. of International Conference on Machine Learning, pp. 239–248.

Tomek, I. 1976. An experiment with edited nearest-neighbor rule. IEEE Trans. on Sys. Man and Cyber., 6(6):448–452.

Verbaeten, S. 2002. Identifying mislabeled training examples in ILP classification problems. Proc. of Benelearn, Annual Machine Learning Conf. of Belgium and the Netherlands.

Weisberg, S. 1980. Applied Linear Regression, John Wiley and Sons, Inc.

Weiss, G.M. 1995. Learning with rare cases and small Disjunctions. Proc. of 12th International Conference on Machine Learning, Morgan Kaufmann, pp. 558–565.

Weiss, G.M. and Hirsh, H. 1998. The problem with noise and small disjuncts. Proc. of 15th International Conference on Machine Learning, San Francisco, CA, pp. 574–578.

Whitaker, A. and Saroiu, S. 1999. Cleaning mislabeled training data using SMART FILTER. Second Project Report for CSE573–Artificial Intelligence, Prof. Pedro Domingos.

Wilson, D. 1972. Asymptotic properties of nearest neighbor rules using edited data. IEEE Trans. on SMC, 2:408–421.

Wilson, D. and Martinez, T.R. 2000. Reduction techniques for examplar-based learning algorithms. Machine Learning, 38(3):257–268.

Winston, P. 1975. Learning structural descriptions from examples. The Psychology of Computer Vision, McGraw-Hill, New York.

Wu, X. 1995. Knowledge Acquisition from Database, Ablex Pulishing Corp., USA.

Wu, X. 1998. Rule induction with extension matrices. American Society for Information Science, 49(5):435–454.

Zhao, Q. and Nishida, T. 1995. Using qualitative hypotheses to identify inaccurate data. Journal of Artificial Intelligence Research, 3:119–145.

Zhu, X., Wu, X., and Yang, Y. 2004. Error detection and impact-sensitive instance ranking in noisy datasets. Proceedings of the 19th National Conference on Artificial Intelligence (AAAI-2004), July 25–29, San Jose, California.

Zhu, X. and Wu, X. 2004. Class noise vs attribute noise: A quantitative study of their impacts. Artificial Intelligence Review, 22(3-4):177–210.