Thesis for the Degree of Doctor of Philosophy

Noisy Training Data Detection: Incorporating the Knowledge of Unlabeled Data

Donghai Guan

Department of Computer Engineering

Graduate School

Kyung Hee University

Seoul, Korea

August, 2009

Noisy Training Data Detection: Incorporating the Knowledge of Unlabeled Data

by

Donghai Guan

Supervised by

Prof. Young-Koo Lee, Ph.D.

Department of Computer Engineering

Graduate School

Kyung Hee University

Seoul, Korea

August, 2009

Noisy Training Data Detection: Incorporating the Knowledge of Unlabeled Data

Donghai Guan

Submitted to
The Faculty of the Graduate School of Computer Engineering
in Partial Fulfillment of the Requirements
of the Degree of
Ph.D.

Thesis Committee:

Professor Chae, Oksam

Professor Chung, Tae-Choong

Professor Kim, Dong-Han

Professor J. d’Auriol, Brian

Professor Lee, Young-Koo

Dedicated to my Family

Acknowledgement

Many different people provided help, support, and input that brought this thesis to fruition. First and foremost, I would like to express my gratitude to my supervisor, Prof. Young-Koo Lee, who gave me an opportunity to do this Ph.D research. He gave me the freedom to try out new ideas and gave me continuous support during the research.

I particularly appreciate Prof. Sungyoung Lee for his guidance and continue support throughout my PhD study. I also would like to thank Prof. Brian J. d'Auriol and Prof. Andrey Gavrilov for their great help on my research.

I am also in debt of my thesis committee whose comments helped me to very much improve the presentation of the thesis.

Many thanks to all members of Activity Recognition Team, as well as the Ubiquitous Computing Group and Database Group, for their collaboration and friendship during my study.

I would like to thank my parents for supporting my decision to continue this study. I also owe a lot of thanks to my younger sister whose efforts of taking over my duty in taking care of our parents have helped to keep my mind on this work.

Last and most importantly, I lovingly thank my sweet fiancée and reliable research partner, Weiwei Yuan, for her love, patient and support.

Donghai Guan

August 2009,

Seoul, Korea

Abstract

The classification learning process consists of different steps: building a training set, training the system, testing its behavior, and finally classifying unknown objects. The quality of the training data is known to be one of the most important factors to determine the learning performance. In real applications, training data are prone to noises for several reasons including subjectivity, data-entry error, or inadequacy of the information used to label each object. Noises in training set tend to degrade the learning performance; handling noisy training data is therefore one of the most important topics in machine learning.

Noise filtering is a popular technique to handle noises in the training data. It identifies and removes noises prior to applying the chosen learning algorithm, so that the noisy data do not influence hypothesis construction. Existing noise filtering methods like edited nearest neighbor, majority/consensus filtering are supervised which only rely upon the training data. In this thesis, we present a new method for noise filtering. Our key idea is to incorporate the knowledge learned from unlabeled data, which are usually easy to obtain since they do not need human labeling effort. Our method is straightforward and general. It can work on any existing noise filtering methods. Existing noise filtering methods can not utilize unlabeled data directly, so we need the method that can convert unlabeled data to labeled data. For this purpose, we propose an ensemble-based semi-supervised learning algorithm to selectively choose and predict labels for unlabeled data. By incorporating the advantage of ensemble learning, this algorithm overcomes the intrinsic limitation of the traditional self-training and co-training algorithm, which require the measure to evaluate the classification “confidence”.

The k-nearest neighbor algorithm is a type of instance-based learning that is highly susceptible to noisy instances in the training set due to the high degree of local sensitivity. Considering its urgent requirement for refined noise filtering techniques, we firstly use our method on three well-known instance-based noise filtering methods: edited nearest neighbor, repeated edited nearest neighbor, and All k-NN. Experimental results demonstrate that all these three methods can achieve improved performance by using our method. This experiment proves the effectiveness of utilizing unlabeled data for instance-based noise filtering methods.

In essence, our method is general and not limited to instance-based noise filtering methods. To show its universality, we use our method with majority/consensus filtering, a well-known general noise filtering method which can be used with any base learning algorithms. We choose three popular algorithms as the base learning algorithm of majority/consensus filtering: naïve Bayes, decision tree, and k-NN. Experimental results indicate that our method can improve the performance of majority/consensus filtering for all the three algorithms.

Based on the above instance-based and general noise filtering methods, the experimental results suggest that our method has the potentiality to provide improved performance for any noise filtering algorithms and any base learning algorithms with only marginal modification required.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Noise detection encompasses aspects of a broad spectrum of technique. Many techniques employed for detecting noises are fundamentally identical but with different names such as noise detection, outlier detection, novelty detection, anomaly detection, deviation detection or exception mining. In this thesis, we use the name noise detection.

Noise detection is important and it can be used in many applications. This study involves evaluation of noisy data in the training set.

The goal of an inductive learning algorithm is to form a good generalization model constructed on the training instances. Generally two main factors determine the quality of generalization model: (1) the quality of the training data, and (2) the appropriateness of the biases of the chosen learning algorithm for the training data. When the learning algorithm is given, the quality of generalization model mainly depends on the quality of the training data. Considering that training data usually include noises which tend to degrade the quality of generalization model, effective noise handling is one of the most important problems in inductive learning.

In order to minimize the downside of noisy training instances, people mainly take one of the two approaches: noise tolerance and noise elimination. Noise tolerance tries to control the negative effect of noisy instances without removing them, usually by designing robust algorithms that are insensitive to noise. The typical methods in this category include rule truncation [1] and tree pruning [2]. For example, pruning in decision trees is designed to reduce the chance that the trees are over fitting to noise in the training data. However, since the classifiers learned from noisy data have less accuracy, the pruning may have very limited effect in enhancing the system performance, especially in the situation that the noise level is relatively high.

On the other hand, noise elimination tries to improve the quality of training data by identifying and eliminating the noisy instances prior to apply the learning algorithm. For a real world dataset, doing the task “by hand” is completely out of the question given the amount of person hours involved. A manual process of noise elimination is also laborious, time consuming, and prone to errors. Useful and powerful tools that automate or greatly assist in noise elimination therefore have been developed. One typical method in this category is to use an ensemble of classifiers and treat the training instance that is misclassified as the noise. It has been argued by [3] that the noise elimination is more effective than noise tolerance. In this work, we focus on noise elimination.

The noisy training instances mainly include two types: attribute noise and class noise. Attribute noises are the errors introduced in the attribute values of the instances. Examples of those external errors include (1) erroneous attribute values, (2) missing or don’t know attribute values, (3) incomplete attributes or don’t care values. The class noises are also called mislabeled noises since they are caused by the mislabeling. Class noise can occur for several reasons including subjectivity, data-entry error, or inadequacy of the information used to label each object. Subjectivity may arise when observations need to be ranked in some way such as disease severity or when the information used to label an object is different from the information to which the learning algorithm will have access. For example, when labeling pixels in image data, the analyst typically uses visual input rather than the numeric values of the feature vector corresponding to the observation. Domains in which experts disagree are natural places for subjective labeling errors. In other domains, the most frequent type of error is mistake made during data-entry. A third cause of labeling error arises when the information used to label each observation is inadequate. For example, in the medical domain it may not be possible to perform the tests necessary to guarantee that a diagnosis is 100% accurate. For domains in which labeling errors occur, an automated method of eliminating or correcting mislabeled observations will improve the predictive accuracy of the classifier formed from the training data.

Quinlan [4] has comprehensively analyzed the two types of noises and demonstrated that, for higher levels of noise, removing noise from attribute information decreases the predictive accuracy of the resulting classifier if the same attribute noise is present when the classifier is subsequently used. However, for class noise, the opposite is true: cleaning the training data will result in a classifier with a higher predictive accuracy. Brodley and Friedl [5][6] have also illustrated that for class noise levels of less than 40%, removing mislabeled instances from the training data resulted in higher predictive accuracy relative to classification accuracies achieved without "cleaning" the training data. Inspired by their works, our study focuses on identifying and eliminating class noises (mislabeled instances), which thereby increasing the classifier's predictive accuracy.

Up to now, many research efforts have been made on eliminating mislabeled instances for effective learning. Guyon [7] 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 instances or exceptions. The noise detection algorithm of Gamberger [8] is based on the observation that the elimination of noisy examples reduces the CLCH (Complexity of the Least Complex correct Hypothesis) value of the training set. They called their noise elimination algorithm the Saturation filter since it employs the CLCH measure to test whether the training set is saturated. Brodley and Friedl [5,6] simplified noise elimination as a filtering operation `where multiple classifiers learned from noisy training data are used to identify noise, and the noise is characterized as the instances that are incorrectly classified by the multiple classifiers. Two major filtering methods they proposed are majority filtering and consensus filtering. In addition, there exists some noise detection methods specially proposed for nearest neighbor classifiers. Wilson [9] used a three-nearest neighbor classifier (3-NN) to select instances that then used to form a 1-NN. Aha, Kibler, and Albert [10] demonstrated that filtering instances based on records of their contribution to classification accuracy in an instance-based classifier improves the accuracy of the resulting classifier.

The noise identification process of the existing methods can be represented by the expression: $R(t) = f(t,T)$, wherein $f(\cdot)$ denotes the noise identification function which depends on a particular measure, such as instances typicality [7], reduction of CLCH [8], number of misclassification [5,6], improvement of classification accuracy [10] and so on; t denotes one training instance in training set T; $R(t)$ denotes the identification & elimination result for instance t that consists of two values: 0 (eliminate) and 1 (retain). For above methods, no matter which kind of noise identification function employed, there are always two parameters with the noise identification function: training set T and the instance t to be evaluated. In other words, given the noise identification function, the identification result (eliminate or retain) for a training instance is only based on the training set and this instance itself.

1.1 Noise filtering by using unlabeled data

Nowadays learning from unlabeled data is a hot topic. Labeled data for machine learning is often very difficult and expensive to obtain, and thus the ability to use unlabeled data holds significant promise in terms of vastly expanding the applicability of learning methods.

Although unlabeled data have been used in many machine learning problems like classification and regression, their potential utility has not been considered by existing noise filtering methods. The novelty of our approach therefore is to make use of unlabeled data to aid the noise filtering in training data. This is in contrast to the existing noise filtering methods which rely upon only the training set. Let U denote the unlabeled set and our approach can be expressed as: $R(t) = f(t,T,U)$. The advantages of our approach include the use of unlabeled data that are often convenient to obtain in many applications with only marginal modifications required to existing methods. Another merit of our approach lies in its wide applicability. It can be used with many various noise detection methods including all the existing methods mentioned in previous paragraph.

1.2 Thesis outline

Below is a summary of the rest of the thesis:

Chapter 2: Semi-supervised classification. Existing noise filtering methods cannot utilize unlabeled data directly. One solution is to predict the labels for unlabeled data by utilizing semi-supervised classification technique. In this chapter, we provide a review of semi-supervised classification, and describe some commonly-used semi-supervised classification approaches, such as self-training and co-training. In addition, we propose an ensemble-based semi-supervised classification approach to solve the limitations of self-training and co-training by incorporating the advantage of ensemble learning.

Chapter 3: Instance-based noise filtering aided by unlabeled data. Instance-based learning like the k-nearest neighbor is susceptible to the noisy training data due to its high degree of local sensitivity. We firstly use our method with instance-based noise filtering methods. Three popular instance-based noise filtering methods are adopted, including edited nearest neighbor, repeated edited nearest neighbor, and All k-NN. The comparisons between the original methods and our proposed methods are evaluated through a set of experiments. The experimental results prove that the performance of instance-based noise filtering can be improved with the aid of unlabeled instances.

Chapter 4: General noise filtering aided by unlabeled data. In essence, our method is not specified to any particular noise filtering method and any learning algorithm. In this chapter, we validate the wide applicability of our method by using majority/consensus filtering. Majority/consensus filtering are general noise filtering methods that can be used with any learning algorithms. In the experiments, we choose three popular learning algorithms: k-nearest neighbor, naïve Bayes, and decision tree. We test whether our proposed method can improve the performance of majority/consensus filtering based on each of these three learning algorithms through a comprehensive empirical study. The study results indicate that our method can improve the performance of majority/consensus filtering for all the three explored algorithms.

Chapter 5: Conclusions and future work. In this chapter, we summarize the main contributions of our work and discuss future research directions for the work presented in this thesis.

Chapter 2
Semi-supervised classification

The key idea of this thesis is to improve the performance of existing noise filtering methods by incorporating the knowledge of unlabeled instances. Existing noise filtering methods are supervised and can only utilize the information of labeled instances. How to utilize unlabeled data is therefore the foremost problem to be considered. Naturally there are two main solutions for this problem. On the one hand, we can adapt existing noise filtering methods for using unlabeled data. On the other hand, we may adapt unlabeled instances for existing noise filtering methods use. In this work, we adopt the latter one as this is a once for all work. Once the unlabeled data are adapted somehow, they can be used by any existing noise filtering method.

To adapt unlabeled data, our solution is to predict the labels for them by using the semi-supervised classification technique, because existing noise filtering methods can only use the information of labeled instances. In this chapter, we provide a brief background on semi-supervised classification and review some common-used semi-supervised classification methods. Followed by, we present our proposed ensemble-based semi-supervised classification method.

2.1 Introduction to semi-supervised learning

Semi-supervised learning has attracted an increasing amount of interest recently. It makes use of both labeled and unlabeled data for learning and therefore falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Many research works have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce considerable improvement in learning performances. The main motivation to develop semi-supervised learning is that in many applications labeled instances are time-consuming and expensive to obtain as they require the efforts of human annotators. For example, obtaining a single labeled example for protein shape classification, which is one of the grand challenges of biological and computational science, requires months of expensive analysis by expert crystallographers.

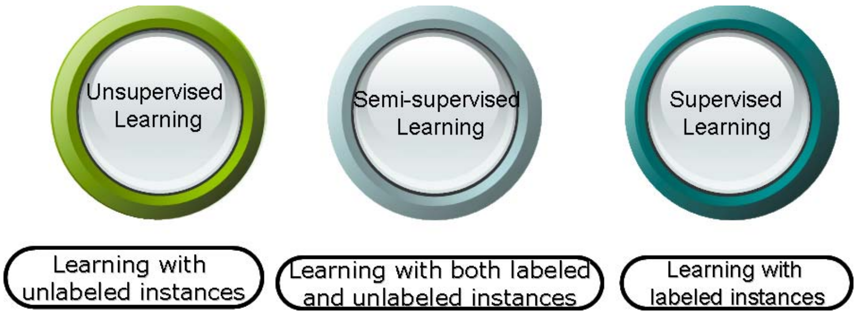

Unsupervised Learning
Semi-supervised Learning
Supervised Learning
Learning with unlabeled instances
Learning with both labeled and unlabeled instances
Learning with labeled instances

Fig. 1. The position of semi-supervised learning

Most existing works on semi-supervised learning focus on the classification task; although, we have noted that unlabeled data have been used to solve other types of problems, including, for example ensemble learning [11][12], dimension reduction [13], active learning [14][15], and feature selection [16][17].

In supervised classification problem we are given as input pairs of variables $(X_1, Y_1) \ldots (X_m, Y_m)$ where the X_i are objects of the type that we want to classify (for example documents or images) and the Y_i are the corresponding labels of the X_i (for example if the X_i are newspaper articles then the Y_i might indicate whether X_i is an article about machine learning). The goal is to minimize error rate on future examples X whose labels are not known. The special case where Y_i can only have two possible values is known as binary classification. Supervised classification problem has been extensively studied in the machine learning community and several algorithms have been proposed. A few of the algorithms which gained broader acceptance are naïve Bayes, neural network, decision trees, k-nearest neighbor, and support vector machines.

In the semi-supervised classification problem, in addition to labeled examples $(X_1, Y_1) \ldots (X_m, Y_m)$ we also receive unlabeled examples $X_{m+1}, \ldots X_n$. Thus we have m labeled examples and $n - m$ unlabeled examples.

Semi-supervised classification began to receive extensive attention in the early 90s. Some of the algorithms that have been proposed for this problem include the Expectation-Maximization algorithm proposed by Dempster, Laird and Rubin [18], the self-training algorithm proposed by Yarowsky [19], the co-training algorithm proposed by Blum and Mitchell [20], the graph mincut algorithm proposed by Blum and Chawla [21], and the Gaussian Fields algorithm proposed by Zhu, Gharamani and Lafferty [22]. The area is still the subject of a very active research effort. A number of researchers have attempted to address the question of "Under what circumstances can unlabeled data be useful" from a theoretical point of view and there also has been great interest from industrial practitioners who would like to make the best use of their unlabeled data.

Among the existing semi-supervised classification algorithms, self-training and co-training are the most popular ones for their straightforwardness and easy implementation. In addition, both of them consist of the procedure to predict the labels for unlabeled data. That is the reason for them to be considered here. Self-training and co-training algorithms are introduced in Chapters 2.1.1 and 2.1.2 respectively.

2.1.1 Self-training

In self-training a classifier is first trained with the small amount of labeled data. The classifier is then used to classify the unlabeled data. Typically the most confident unlabeled points, together with their predicted labels, are added to the training set. The classifier is re-trained and the procedure repeated. Note the classifier uses its own predictions to teach itself. The procedure is also called self-teaching. The generative model and EM approach can be viewed as a special case of "soft" self-training. One can imagine that a classification mistake can reinforce itself. Some algorithms try to avoid this by "unlearn" unlabeled points if the prediction confidence drops below a threshold. Self-training has been applied to several natural language processing tasks. Yarowsky [19] uses self-training for word sense disambiguation, e.g. deciding whether the word "plant" means a living organism or a factory in a give context. Riloff et al. [23] uses it to identify subjective nouns. Maeireizo et al. [24] classify dialogues as "emotional" or "non-emotional" with a procedure involving two classifiers. Self-training has also been applied to parsing and machine translation. Rosenberg et al. [25] apply self-training to object detection systems from images, and show the semi-supervised technique compares favorably with a state of-the-art detector. Self-training is a wrapper algorithm, and is hard to analyze in general. However, for specific base learners, there has been some analyzer's on convergence. A typical self-training algorithm is shown in Table 1.

Table 1 Typical self-training algorithm

```
Algorithm: Typical self-training algorithm
Input: $T$ (training set), $U$ (unlabeled set)
Parameter: $B$ (base learning algorithm)
Output: $H$ (Hypothesis)
(1)    $T_U \leftarrow f_{DL}(T,U)$ // new training data
(2)    $T \leftarrow T \cup T_U$
(3)    Induce $H$ based on algorithm $B$ and training instances $T$
// $f_{DL}(\cdot)$ is the "data labeling function" that generates new training data $T_U$
   by selectively choosing some unlabeled data and predicting their labels
```

Data labeling function is the core of self-training algorithm. It is responsible to choose confident labeling instances. The confidence degree that depends on the ranking of class membership probabilities is commonly used as the selection metric that ranks and selects the unlabeled instances for next training of base learning algorithm. Naïve Bayes is often used as the underlying classifier because its class membership probability estimates have good ranking performance.

2.1.2 Co-training

Co-training [20] assumes that (1) features can be split into two sets; (2) each sub-feature set is sufficient to train a good classifier; (3) the two sets are conditionally independent given the class. Initially two separate classifiers are trained with the labeled data, on the two sub-feature sets respectively. Each classifier then classifies the unlabeled data, and 'teaches' the other classifier with the few unlabeled examples (and the predicted labels) they feel most confident. Each classifier is retrained with the additional training examples given by the other classifier, and the process repeats. In co-training, unlabeled data helps by reducing the version space size. In other words, the two classifiers (or hypotheses) must agree on the much larger unlabeled data as well as the labeled data. We need the assumption that sub-features are sufficiently good, so that we can trust the labels by each learner. We need the sub-features to be conditionally independent so that one classifier's high confident data points are *iid* samples for the other classifier.

Nigam and Ghani [26] perform extensive empirical experiments to compare co-training with generative mixture models and EM. Their result shows co-training performs well if the conditional independence assumption indeed holds. In addition, it is better to probabilistically label the entire unlabeled instances, instead of a few most confident data points. They name this paradigm co-EM. Finally, if there is no natural feature split, the authors create artificial split by randomly break the feature set into two subsets. They show co-training with artificial feature split still helps, though not as much as before. Balcan and Blum [27] show that co-training can be quite effective, that in the extreme case only one labeled point is needed to learn the classifier. Zhou et al. [28] give a co-training algorithm using Canonical Correlation Analysis which also need only one labeled point. Dasgupta et al. [29] provide a PAC-style theoretical analysis. Table 2 is a typical co-training algorithm.

Table 2 Typical co-training algorithm

```
Algorithm: Typical Co-training algorithm
Input: $T$ (training set), $U$ (unlabeled set)
        $T$ is described by two different views $T_{view1}$ and $T_{view2}$
Parameter: $B$ (base learning algorithm)
Output: $H$ (Hypothesis)
(1)     $T_{U1} \leftarrow f_{DL}(T_{view1}, U)$ // new training data obtained from view 1
(2)     $T_{U2} \leftarrow f_{DL}(T_{view2}, U)$ // new training data obtained from view 2
(3)     $T \leftarrow T \cup T_{U1} \cup T_{U2}$
(4)     Induce $H$ based on algorithm $B$ and training instances $T$
// $f_{DL}(\cdot)$ is the "data labeling function" that generates new training data
$T_U$ by selectively choosing some unlabeled data and predicting their
labels
```

2.2 The proposed ensemble-based data labeling function

Although self-training and co-training are widely used, both of them have some limitations. For self-training, the classifier requires some measures to evaluate the "confidence" of unlabeled data. However, it is not easy for many classifiers like k-nearest neighbor to give this measure. What's more, even if the classifier could measure the confidence, its own prediction on label of the unlabeled data is not reliable. In case of co-training, it lacks generality since it only works for the data sets which can be represented by two sufficient and independent views. In addition, the measure of "confidence" is also required. To relieve the requirement of "confidence" measure, we propose an ensemble-based data labeling function. The algorithm of this function is shown in Table 3.

As shown in the algorithm, multiple learning algorithms (more than two) are employed in this data labeling function. Firstly, multiple classifiers will be trained based on the initial labeled data by these various learning algorithms. Afterwards these classifiers will classify the unlabeled instances. The instance (and its predicted label) will be selected only when all these classifiers give the same classification result (predicted label) on it. Using this mechanism, our proposed data labeling function overcomes the limitation of "confidence-based" data labeling function employed in traditional self-training and co-training algorithms. Explicit measure of confidence is not required by our algorithm.

Table 3 Ensemble-based data labeling function

```
Algorithm: Ensemble-based Data Labeling Function
Input: $T$ (training set), $U$ (unlabeled set)
Parameter: $k$ (number of iterations), $y$ (number of learning algorithms)
        $u$ (number of initially selected unlabeled instances)
        $A_1, A_2, ..., A_y$ ($y$ kinds of learning algorithms)
Output: $T_U$ (selected unlabeled instances from $U$ with predicted labels)
(1)   create $U'$ by choosing $u$ instances at random from $U$
(2)   $T_U \leftarrow \varnothing$
(3)   for $i = 1, ..., k$ do
(4)     $U \leftarrow U \setminus U'$, $num_{before} \leftarrow |T_U|$ //size of $T_U$ when iteration starts
(5)     for $j = 1, ..., y$ do
(6)       induce $H_j$ based on instances in $T$ and algorithm $A_j$
(7)     end for
(8)     for every $t \in U'$ do
(9)       for $j = 1, ..., y$ do
(10)        $pl_j(t) \leftarrow H_j(t)$ // predicted label of $H_j$ on $t$
(11)      end for
(12)      if $pl_1(t) = pl_2(t) =, ..., pl_y(t)$
(13)      then $T_U \leftarrow t \cup pl_1(t)$, $U' \leftarrow U' \setminus t$
(14)    end for
(15)    $T \leftarrow T \cup T_U$
(16)    $num_{after} \leftarrow |T_U|$, $\Delta num \leftarrow num_{after} - num_{before}$ // num. of selected data
(17)    if $|U| \geq \Delta num$
(18)    then randomly choose $\Delta num$ instances from $U$ to replenish $U'$
(19)    if $0 < |U| < \Delta num$
(20)    then choose all instances of $U$ to replenish $U'$
(21)    if $|U| = 0$
(22)    then exit;
(23)  end for
```

2.2.1 The performance of ensemble-based data labeling function

As shown in Table 1 and 2, data labeling function is the core of self-training and co-training. Its quality (predictive accuracy on unlabeled data) has heavy impact on the performance of semi-supervised classification algorithms. Intuitively the predicted labels of data labeling function, including both traditional confidence-based and our ensemble-based, are prone to errors. Although we mainly care for whether noise filtering performance could be improved with the aid of these self-labeled instances by data labeling function (in the following chapters), we would like to have the quantitative knowledge about what is the predictive accuracy of this ensemble-based data labeling function. Moreover, based on this data labeling function, whether semi-supervised classification method works. In the experiment, we will test the performance of our method based on the self-training method based on k-NN algorithm (1-NN). We choose k-NN as it is a typical method which requires noise filtering technique since it is sensitive to noises. Regarding to the ensemble-based data labeling function, three learning algorithms are employed, including 3-NN, naïve Bayes, and decision tree.

The experiments are based on the benchmark data sets from the Machine Learning Repository [30]. Information of these data sets is tabulated in Table 4. These data sets are collected from different real-world applications in various domains, such as breast cancer (breast) and iris plant database (iris). Note that “magic” used here is just part of the original one in UCI. Originally, “magic” has 19020 instances consisting of 2 classes. To reduce the experiment time, we extract the first 1000 instances from each class. Therefore, 2000 samples are used here.

Table 4 UCI data sets used in the experiments

Data set	**Attribute**	**Size**	**Class**	**Class distribution**
iris	4	150	3	50/50/50
voting	16	435	2	267/168
heart2	13	294	2	188/106
horse	15	368	2	232/136
sonar	60	208	2	111/97
wine	13	178	3	59/71/48
breast	9	1000	2	700/300
yeast	8	1484	10	463/429/244/163/51/44/37/30/20/5
australian	14	690	2	383/307
bupa	6	345	2	145/200
diabetes	8	768	2	500/268
echo	7	131	2	88/43
german	24	1000	2	700/300
glass	9	214	6	70/76/17/13/9/30
magic	10	2000	2	1000/1000
credit	15	690	2	307/383
spect	44	267	2	212/55
wdbc	31	569	2	357/212
ecoli	7	336	8	143/77/52/35/20/5/2/2
ionosphere	34	351	2	225/126
haberm	3	306	2	225/81

Each data set is divided into training set, test set, and unlabeled set. Self-training algorithm (with our proposed ensemble-based data labeling function) works on the unlabeled set and outputs the self-labeled training set, which then combines with the original training set to become an augmented training set. Afterwards the test set is classified by the supervised learning algorithm (based on the original training set) and the self-training algorithm (based on augmented training set) respectively. Classification accuracy is the measure to evaluate the performance of data labeling function and self-training algorithm.

The detailed process for each data set is as follows:

(1) D is randomly partitioned into two parts: labeled set L and unlabeled set U.

(2) Ten trials derived from ten-fold cross-validation on L are used to evaluate the performance of data labeling function and self-training algorithm. At each trial, 90% of L is firstly selected and it is denoted by T, used as training set.

The remaining 10% of L is used as test set to be classified by 1-NN (based on T) and 1-NN based self-training algorithm (based on T and U). At each trial, we also calculate the predictive accuracy of ensemble-based data labeling function on U .

(3) The average classification accuracies of data labeling function and self-training algorithm are obtained by averaging ten trials' classification accuracies.

(4) Considering that the partition of data set could influence this average classification accuracy, we execute the partition five times and get five classification accuracies (execute step 1-3 five times)

(5) Finally the reported classification accuracy is the further averaged value of these five values.

Two major parameters are able to influence this experiment. The first parameter determines data partitioning and it is the ratio between labeled data to whole data, referred to labeled ratio. It is set to 10% in the experiment. The second parameter is the number of iterations for ensemble-based data labeling function. We have performed several experiments varying the iteration number, including 1, 2, 3, and 4. Referring to the algorithm of ensemble-based data labeling, one obvious function of iteration number k is: more unlabeled instances are expected to be selected and added to the training set when k increases.

Table 5 presents the accuracy of predicted labels (for unlabeled instances) by ensemble-based data labeling function. This table consists of two parts: 1) the predictive accuracy of each data set, 2) the average accuracy across all the data sets.

The observations from Table 5 include: (1) data labeling function cannot provide noise-free predictive labels (for unlabeled instances). The accuracy varies for different data sets and different iteration numbers. (2) The predictive accuracy does not increase when the iteration number increases.

Table 6 presents the performance of 1-NN based self-training. It shows that the performance of 1-NN could be improved by using self-labeled instances predicted by ensemble-based data labeling function. On average, the classification accuracy of 1-NN is 72.4%. The performances of 1-NN based self-training are 73.8%, 74.3%, 74.3%, and 74.9% when iteration number is 1, 2, 3, and 4 respectively.

Table 5 Performance of ensemble-based data labeling function

Dataset	Iteration times of data labeling function			
	1	2	3	4
iris	0.909	0.920	0.887	0.882
voting	0.964	0.953	0.948	0.952
heart2	0.826	0.804	0.818	0.827
horse	0.806	0.816	0.809	0.793
sonar	0.785	0.804	0.804	0.804
wine	0.953	0.939	0.932	0.872
breast	0.965	0.974	0.965	0.965
yeast	0.688	0.691	0.651	0.653
australian	0.904	0.903	0.893	0.884
bupa	0.611	0.604	0.580	0.575
diabetes	0.795	0.769	0.769	0.768
echo	0.727	0.750	0.750	0.750
german	0.778	0.782	0.765	0.765
glass	0.585	0.546	0.546	0.546
magic	0.832	0.803	0.781	0.778
credit	0.901	0.874	0.870	0.874
spect	0.888	0.818	0.814	0.832
wdbc	0.980	0.975	0.973	0.969
ecoli	0.878	0.851	0.862	0.861
ionosphere	0.903	0.901	0.894	0.894
haberm	0.738	0.754	0.729	0.719
Ave.	0.829	0.821	0.811	0.808

Table 6 Performance of 1-NN based self-training

Dataset	Iteration times of data labeling function				
	1-NN	1	2	3	4
iris	0.750	0.756	0.717	0.744	0.778
voting	0.852	0.885	0.905	0.908	0.918
heart2	0.733	0.789	0.778	0.833	0.756
horse	0.707	0.704	0.700	0.726	0.674
sonar	0.683	0.622	0.589	0.572	0.667
wine	0.989	0.989	0.970	0.985	0.978
breast	0.902	0.914	0.947	0.939	0.943
yeast	0.461	0.474	0.510	0.489	0.506
australian	0.782	0.814	0.814	0.813	0.804
bupa	0.653	0.622	0.642	0.619	0.633
diabetes	0.674	0.693	0.705	0.681	0.712
echo	0.740	0.700	0.820	0.700	0.833
german	0.657	0.677	0.670	0.690	0.660
glass	0.442	0.508	0.483	0.450	0.500
magic	0.708	0.708	0.707	0.732	0.725
credit	0.776	0.797	0.799	0.803	0.797
spect	0.641	0.670	0.624	0.737	0.674
wdbc	0.954	0.968	0.948	0.957	0.948
ecoli	0.675	0.700	0.713	0.678	0.729
ionosphere	0.764	0.735	0.764	0.761	0.731
haberm	0.656	0.773	0.809	0.786	0.759
Ave.	0.724	0.738	0.743	0.743	0.749

The performance of ensemble-based data labeling is preliminarily studied in this chapter. As shown in Table 5, the predicted labels of unlabeled data by data labeling function are prone to errors (around 20% are errors). These noisy instances (unlabeled data with their noisy labels) will be used to help the noise detection of original training set. We are interested in the question: since these unlabeled instances cannot get noise-free predicted labels, can they contribute to noise filtering in training set? The following chapters aim to answer this question.

Chapter 3
Semi-supervised classification

Instance-based learning techniques work essentially by keeping typical attribute examples for each class. Instance-based learning algorithms in general have three characteristics:

- A similarity function. This tells the algorithm how close together two instances are. Although this sounds easy, there is a great deal of complexity in choosing the similarity function, especially in situations where some of the inputs are enumerated. For example, if you were trying to match people, and one attribute was hair color, what does distance mean in the context of hair color?
- A "typical instance" selection function. This tells the algorithm which of the instances to keep as examples. How do you know which instances are "typical" and which are atypical?
- A classification function. This function is the one that when given a new case, decides how it relates to the learned cases. For example, this function might be the instance to which it is closest in location.

The k-nearest neighbor (k-NN) algorithm is a typical instance-based learning. We will introduce some background knowledge of k-NN, and then present some k-NN related noise filtering techniques.

3.1 The k-nearest neighbor algorithm

The k-nearest neighbor is generally considered as a good classifier. It has a number of advantages, including:

- It can easily be implemented and it is conceptually simple.
- Its behavior is asymptotically optimal [31].

- Its expected error is bounded [32].

Here, we briefly explain some features of these three advantages. Firstly we highlight its easiness of implementation and its conceptual simplicity. Imagine that two new fruits are first shown to someone. Then, when another unknown piece of fruit is presented, the individual will try to classify the new one by comparing it to the firstly shown pieces. So, the idea behind algorithms based on proximity is as follows. The classification of a new item x could be estimate based on the already known classifications of the elements sufficiently near to x, because observations that are close to each other will have a high probability to belong to the same class.

Let $\{X,\Theta\}=\{(x_1,\theta_1),(x_2,\theta_2),...,(x_n,\theta_n)\}$ be a training set with n instances $\{x_i\}_{i=1}^{n}$ and their labels $\{\theta_i\}_{i=1}^{n}$. Let x be a new sample with an unknown class label. Assume $(x',\theta')\in\{X,\Theta\}$ is the nearest instance to the sample x. Then, the NN rule would be: $\delta_{NN}(x)=\theta' \Leftrightarrow d(x,x')=\min_{i=1..n} d(x,x_i)$

Considering the asymptotic optimal behavior of the NN rule, we must say that, in addition to its conceptual simplicity, the NN rule has a good behavior when applied to non-trial problems. In fact, the k-NN rule is asymptotically optimal in the Bayes sense [33]. In other words, the k-NN rule performs as well as any other possible classifier, provided that there is an arbitrary large number of representative prototypes available and the volume of the k-neighborhood of x is arbitrarily close to zero for all x.

Given that the above conditions are fulfilled, the NN rule expected error is never worse than twice the Bayes error rate. In this sense, at least half of the classification information in an infinite data set resides in the nearest neighborhoods.

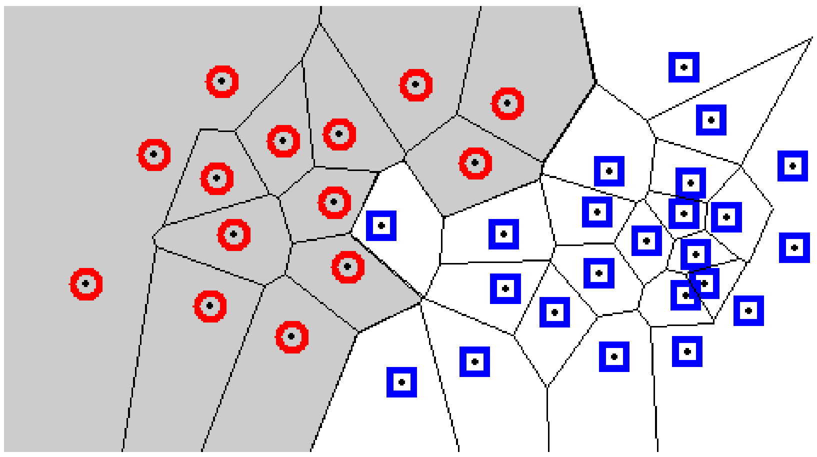


Fig. 2. Decision regions of nearest neighbor rule

The decision rule of nearest neighbor determines that the decision regions of nearest neighbor are the cells. As shown in Fig. 2, each cell contains one sample, and every location within that cell is closer to that sample than to any other samples. Every query point will be assigned the classification of the sample within that cell.

Although nearest neighbor (and k-NN) has many advantages. However, due to this kind of decision region, nearest neighbor has high degree of local sensitivity, which makes it highly susceptible to noisy training instances. For example, if a point in Fig. 2 is mislabeled, then the points to be classified will be misclassified if they are located in the cell of that mislabeled point.

Nearest neighbor editing techniques have been proposed to deal with the mislabeled instances. We will introduce them in the following parts.

3.2 Nearest neighbor editing

Nearest neighbor editing is the step in charge of increasing the accuracy of predictions, when there is a great amount of noise in the training data. A basic editing algorithm removes noisy instances, as well as close border cases, eliminating a possible overlap between the regions from different classes and leaving smoother decision boundaries. Wilson introduced the first editing method [9]. Briefly, the k-NN rule is used to estimate the class of each example in the training set followed by removing those examples whose true class labels do not agree with the ones judged by the k-NN rule.

Many researchers have addressed the problem of editing by proposing alternative schemes. Some representative works are introduced here. Tomek [34] proposed to apply the idea of the Wilson's algorithm repeated until no more instances can be removed. Tomek also proposed the All k-NN editing scheme. It uses a set of the l-NN rules, with l ranging from 1 to k. In general, both algorithms achieve a higher storage reduction than the Wilson's editing, but similar in the classification accuracy. They are however higher at the computational efforts.

The generalized editing [35] consists of removing some "suspicious" instances from the training set and also changing the class labels of some of them. Its purpose is to cope with all types of imperfections of the training instances (mislabeled, noisy and atypical cases). Recently, the generalized editing and Wilson's algorithm have been jointly used for the depuration method [36].

In the case of editing algorithms based on the leaving-one-out error estimate (the Wilson's scheme and its relatives), the statistical independence between test and training instances cannot be assumed because their functions are interchanged. In order to achieve this statistical independence, classification of instances can be performed in a hold-out manner. Thus, the Holdout editing [37] consists of randomly partitioning the initial training set into $b > 2$ blocks of instances, $B_{1,}...B_{b,}$ and then eliminating cases from each block using only two independent blocks at the same time. [37] also introduced the Multiedit algorithm, which basically corresponds to an iterative version of the Holdout scheme using the 1-NN rule.

A genetic algorithm [38] was also applied to define an edited set for the NN rule. Two different criteria were employed as the fitness function: the apparent error rate and a criterion based on the certainty of the classification. The empirical results show that the latter criterion led to a subset of the initial training set that provides higher classification accuracy in comparison to the whole original set, with random selection and with the Wilson's technique.

The work [39] presented an editing algorithm based on proximity graphs, such as the Gabriel graph and the relative neighborhood graph. The first one computes the corresponding graph structure and then eliminates instances incorrectly classified by its graph neighbors. On the other hand, a combined editing-condensing scheme was also introduced to remove internal instances as well as border cases by using the concept of graph neighbors.

The rationale of the k-NN editing rule proposed by [40] is very similar to that of the Wilson's scheme. In this method, the condition for an instance x to be included in the edited set is that all the k nearest neighbors must be from the class to which x belongs. Accordingly, this condition is much more severe than that in Wilson's algorithm and, as a consequence, the number of instances in the resulting edited set is equal to or less than in the Wilson's edited set.

The ACC filtering technique introduced by [41] tries to find centre instance of compact regions by considering the classification performance of each example in the training set. Each training instance is classified by its nearest neighbor. If it is correctly classified, then classification accuracy of its nearest neighborhood will be increased. After processing all the training instances, the algorithm discards examples with the accuracy lower than a certain threshold. As center instances are usually neighbours of other instances from the same class, they generally gain a high accuracy, thus are being retained by ACC.

Among all above works, we consider the Wilson editing (i.e. edited nearest neighbor (ENN)), repeated nearest neighbor (RENN), and All k-NN (ANN) algorithms due to their wide-spread and more popular usage in the literature.

ENN is the base of the other two algorithms. It removes all instances which have been misclassified by the k-NN rule from the training set. Fig. 3 shows the effect of ENN. In this figure, the hollow rounds and the solid rounds represent instances which belong to two different classes. The left part shows a hypothesis training set where misclassified instances using the 1-NN rule are marked with dotted circles around them. The right part shows the reduced training set after applying ENN.

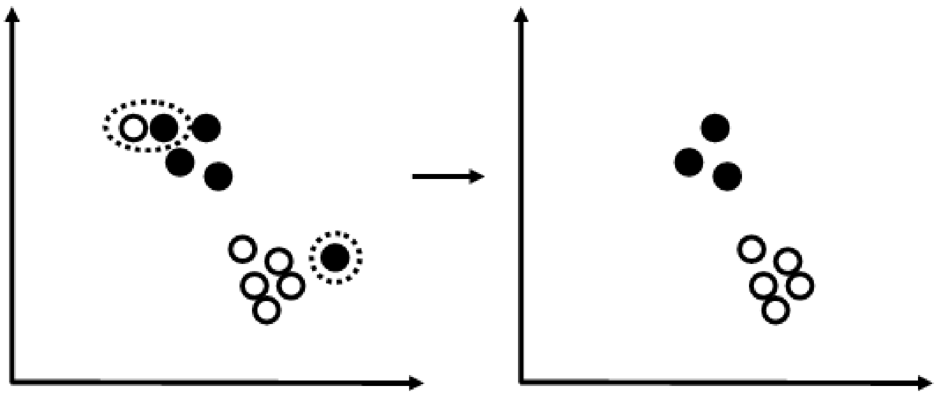


Fig. 3. ENN with 1-NN classifier

The idea of ENN relies on the fact that one can optimally eliminate outliers and possible overlap among classes from a given training set so the training of the corresponding classifier becomes easier in practice. In fact, it has been shown by Penrod and Wagner [42] that the accuracy of the ENN classifier converges to Bayes error as the number of instances approaches infinity. Table 7 gives the ENN algorithm.

Table 7 The algorithm of edited nearest neighbor (ENN)

```
1. Let $T_e = T$ // $T$ is the original training set, and $T_e$ is the edited set
2. For each $x_i \in T_e$, do:
    Discard $x_i$ from $T_e$ if it is misclassified using the k-NN rule
    with prototypes in $T_e \setminus \{x_i\}$
```

RENN applies the ENN algorithm repeatedly until all remaining instances have a majority of their neighbors with the same class, which continue to widen the gap between classes and smooth the decision boundary of ENN. The RENN algorithm is shown in Table 8.

Table 8 The algorithm of repeated edited nearest neighbor (RENN)

```
1. Let $T_e = T$ // $T$ is the original training set, and $T_e$ is the edited set
REPEAT
2. At iteration $t$, for each $x_i \in T_e^t$ (edited set at iteration $t$), do
   Discard $x_i$ from $T_e^t$ if it is misclassified using the k-NN rule with prototypes in $T_e^t \setminus \{x_i\}$;
UNTIL $T_e^t = T_e^{t-1}$ // $T_e^t$ and $T_e^{t-1}$ denote the edited data set of $T$ at iteration $t$ and $t-1$ respectively
```

The ANN algorithm is similar with the iterative ENN with the only exception that the value k is increased after each iteration. Its algorithm is given in Table 9.

Table 9 The algorithm of All k-NN (ANN)

```
1. Let $T_e = T$ // $T$ is the original training set, and is the edited set
2. For each $x_i \in T_e$, do:
   2.1 set $m = 1$
   2.2 while $m < k + 1$ do:
       2.2.1 Discard $x_i$ from $T_e$ if it is misclassified using the m-NN rule with prototypes
             in $T_e \setminus \{x_i\}$, go to Step 2.
       2.2.2 Set $m = m + 1$
```

The common factor in all the three data editing methods discussed above is that they edit each instance based on the voting of other instances in the training set. In many applications, as the matter of fact, it is common that the size of the unlabeled data set is greater than that of the labeled data set (e.g. mainly due to the fact that unlabeled data do not require human labeling effort, and are easy to obtain).

In Fig. 4, "$\times$" represents unlabeled instance. Existing data editing methods, such as ENN, do not consider these unlabeled data and they edit the training set as if the unlabeled data do not exist. The editing result is shown in the right part of Fig. 4.

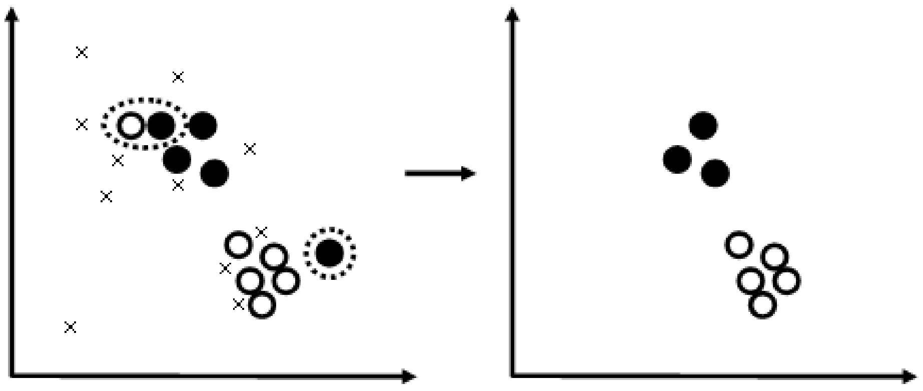


Fig. 4. ENN with 1-NN classifier when unlabeled data are available

3.3 Nearest neighbor editing aided by unlabeled data (NNEAUD)

Our intuitive idea is to extend the searching scope of neighbors from the training set to the whole data set which includes both the training set and the unlabeled set. Considering that the labels of unlabeled data are not available, in order to utilize unlabeled data, the first phase therefore is to predict the labels for them. Then the second phase is to utilize this augmented set in data editing.

The first phase employs the ensemble-based data labeling function introduced in Chapter 2.

The nearest neighbors of a training instance obtained from a search of the training set and those from the whole set might be different. The variation of nearest neighbors might lead to a changing of editing result. The effect of our method is illustrated in Fig. 5. Let T denote the training set and T_U denote the unlabeled set with their predicted labels from ensemble-based data labeling function. Traditionally, when editing T using ENN with 1-NN rule, instance x_1 will be retained since its nearest neighbor in T has the same label with it.

Instance x_2 will be removed since its nearest neighbor in T has different label with it. However, when aided by T_U, the editing results of x_1 and x_2 are changed. Now x_1 will be removed as its nearest neighbor searched from the whole data set is in T_U which has the different label with x_1. In contrast, x_2 will be retained as its nearest neighbor based on the whole data set is from T_U which has the same label with it.

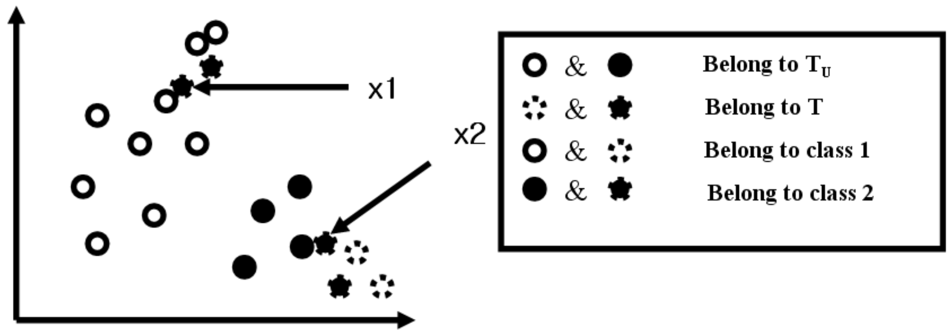


Fig. 5. The effect of unlabeled data for data editing

Now the variants of ENN, RENN, and ANN in the case of using unlabeled data to aid data editing on T are considered. Without changing the data editing rules much, ENN, RENN, and ANN can use T_U easily as shown in Tables 10, 11, and 12. Tables 10 through 12 show two different usages of T_U. In Tables 10 and 12, T_U is used to optimize the editing performance of T while T_U is never edited. In Table 11, T_U and T are edited together. But as shown in Step 3 of Table 7, $T_e = T_e \cap T$, the final edited data are only extracted from T. Therefore, all the three methods regard unlabeled data as the activator of data editing which will not be included in the final edited set.

Table 10 The algorithm of ENN aided by unlabeled data

```
1. Let $T_e = T$
2. For each $x_i \in T_e$, do:
   Discard $x_i$ from $T_e$ if it is misclassified using the k-NN rule with prototypes in
     $(T_e \setminus \{x_i\}) \cup T_U$
```

Table 11 The algorithm of RENN aided by unlabeled data

```
1. Let $T_e = T \cup T_U$
REPEAT
2. At iteration $t$, for each $x_i \in T_e^t$ (edited set at iteration $t$), do
-- Discard $x_i$ from $T_e^t$ if it is misclassified using the k-NN rule with prototypes in $T_e^t \setminus \{x_i\}$
UNTIL ($T_e^t = T_e^{t-1}$, $T_e^t$ and $T_e^{t-1}$ denote the edited data set of $T$ at iteration $t$ and $t-1$ respectively)
3. $T_e = T_e \cap T$
```

Table 12 The algorithm of ANN aided by unlabeled data

```
1. Let $T_e = T$ ($T$ is the original training set, and $T_e$ will be the edited set)
2. For each $x_i \in T_e$, do:
   2.1 set $m = 1$
   2.2 while $m < k + 1$ do:
          2.2.1 Discard $x_i$ from $T_e$ if it is misclassified using the m-NN rule with
             prototypes in $(T_e \setminus \{x_i\}) \cup T_U$, go to Step 2.
          2.2.2 Set $m = m + 1$
```

Although ensemble-based data labeling function could predict the labels for unlabeled data, as shown in the experiments in Chapter 2, the predicted labels may include some errors. The erroneous labels could potentially degrade the editing performance. Considering the following three cases (Fig. 6) based on the ENN algorithm:

Case A: Traditional ENN. The edited data set is denoted by $T_{e(a)}$.

Case B: Assume that an oracle who can predict labels for the unlabeled set U correctly exists. Then this idealistic noise-free labeled data set, denoted by T_{ideal}, is used to aid the editing on T. The edited data set is denoted by $T_{e(b)}$.

Case C: Our proposed ensemble-based data labeling function is used to predict labels for the unlabeled set U and then this realistically obtained, but possibly noisy labeled data set, denoted by $T_{realistic}$, is used to aid the editing on T. The edited data set is denoted by $T_{e(c)}$.

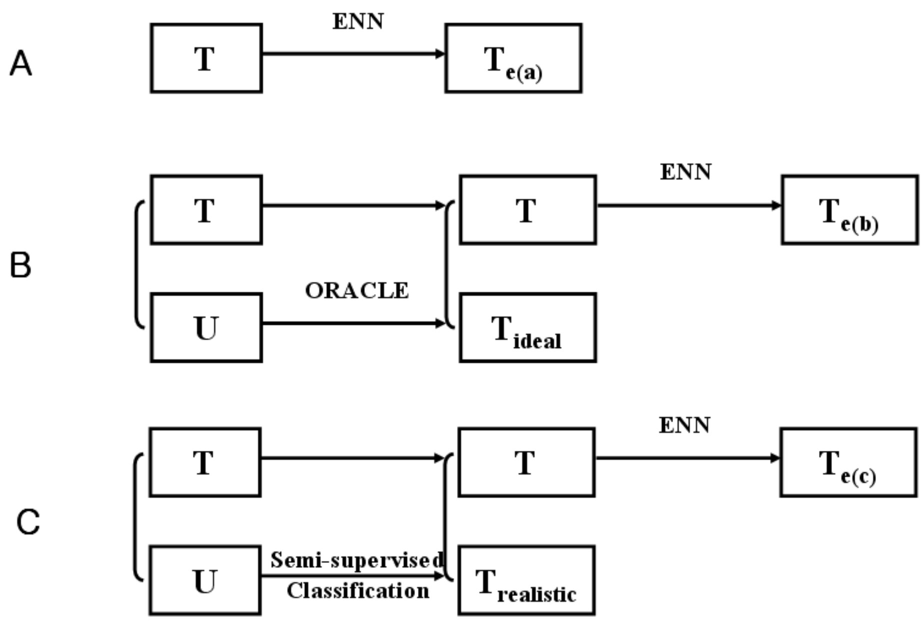


Fig. 6. (A) standard ENN (B) ENN aided by unlabeled data (processed by the Oracle) (C) ENN aided by unlabeled data (processed by semi-supervised classification)

It is expected that the editing performance of case B is the best since T_{ideal} is noise free which provides reliable extra information to aid the editing process. Heuristically, the functions of $T_{realistic}$ are two-fold. On one hand, the noise-free instances in $T_{realistic}$ could improve the editing performance as T_{ideal} . On the other hand, the noisy instances in $T_{realistic}$ could degrade the editing performance. Therefore, the comparison between case A and case C is significant. The success of our proposed method (case C) depends on the comparison between the positive effects and the negative effects generated by $T_{realistic}$. The comparison between case A and case C will be validated in next section.

3.4 Empirical study

The objective of the empirical study is to validate the benefit of our method compared with existing methods in the instance-based noise filtering.

3.4.1 Experimental setup

Three data editing techniques are used and tested on the benchmark data sets from the Machine Learning Database Repository. These methods are Wilson's edited nearest neighbor (ENN), Tomek's Repeated ENN (RENN), and All k-NN (ANN). Aided by unlabeled data, their variants are represented by ENN+, RENN+, and ANN+ respectively.

Information of the data sets used in the experiment is tabulated in Table 4. Each data set is divided into training set and test set. Data editing method works on the training set and generates the edited training set. Then, the test set is classified by the edited training set with the k-NN algorithm. Classification accuracy is the measure to evaluate the performance of data editing methods, where

$$\text{classification accuracy}=\frac{\text{No. of correct classifications on testing instances}}{\text{No. of testing instances}}$$

When two data editing methods are applied to the same data set with the same k-NN algorithm, higher classification accuracy means that the data editing performance is better. To obtain the classification accuracy, each data set D is processed as follows:

(1) Data set D is randomly partitioned into two parts: labeled set L and unlabeled set U .

(2) Ten trials derived from ten-fold cross-validation on L are used to evaluate the performance of editing methods. At each trial, 90% of L is firstly selected and it is denoted by T_{temp} . Most data sets here are experimental data sets where the ratio between noisy data to the whole data might be small. However, the performance of noise filtering need to be evaluated on the noisy data sets. To this end, we artificially generate some noises in T_{temp} by selecting some instances at random and then changing their labels. The number of selected instances, that is the number of generated noises, is based on the defined noise ratio, which is

the ratio between noisy data to the data in T_{temp} . Let T denote the data after adding noises in T_{temp} . T is used as training set and it will be edited by the various editing methods as mentioned above. The remaining 10% of L is used as test set to evaluate the performance of various edited sets of T .

(3) The average classification performance is obtained by averaging ten trials' results.

(4) Considering that the partition of data set could influence this average classification result, we execute the partition five times and get five classification values (execute step 1-3 five times).

(5) Finally the report classification result is the further averaged value of these five values.

In this experiment, ensemble-based data selection method is configured as follows. Three classifiers are generated by: 3-nearest neighbor, naïve Bayes, and decision tree respectively. Initially, the size of U', u is equal to the size of labeled set, namely $u = |L|$. Iteration number k is 2. In addition to the parameters in this semi-supervised classification method, there are other three major parameters which can influence the experiment. The first parameter determines data partitioning (step 1 of above experiment flow) and it is the ratio between labeled data to whole data, referred to labeled ratio. The second parameter determines nearest neighbor editing. Refer to the nearest neighbor editing methods, k-NN algorithm is the base of them. Therefore, different choice of nearest neighbor number k can influence the experiment. 3-NN is the most popular setting in nearest neighbor editing and it is adopted. The third parameter determines the noise level in the training set T and it is noise ratio (step 2 of above experiment flow). Considering that the data sets obtained from the real applications might have different labeled ratios and noise levels, we have performed several experiments varying these two values to make the experiments comprehensively.

3.4.2 Experimental results: varying the labeled ratio

The objectives of experiments in this part are two-fold: (1) testing whether our proposed method works well under different labeled ratios, and (2) whether there is any relationship between the labeled ratio and the performance of our method. In this part, noise ratio is 0 (i.e. we do not add any noises in the training set). Note that this does not mean that there are no any noises in the training set. The training set from UCI datasets unavoidable includes some noises. The labeled ratios are varied including 5%, 10%, 15%, 20%, 30%, 40%, and 50%.

Table 13 shows the results when labeled ratio is 5%. In each cell, the value without bracket represents the classification accuracy. In addition, we also give the data retention rate that is below the accuracy and in brackets. Data retention rate is defined as

$$\text{data retention rate}=\frac{\text{No. of labeled instances after editing}}{\text{No. of training instances}}$$

This rate captures the contribution of unlabeled data for training data (labeled data) editing. The main purpose of data editing is to improve the classification accuracy. Therefore, in the experiment, classification accuracy is the only measure to evaluate the editing performance. However, if two data editing methods give the same classification accuracy, the one with smaller data retention rate is more attractive as it has smaller number of training data leading to higher classification speed.

In Table 13 and the following results in this part, “Ave.” denotes the average classification accuracy across all data sets. “B/W” denotes the number of data sets for which our method is statistically better or worse than the original method. “S.B/W” denotes the number of data sets for which our method is significantly better (threshold is 2%) or worse than the original method. In each row, the better accuracy of each pair (ENN & ENN+, RENN & RENN+, ANN & ANN+) is shown in bold. The significant better accuracy is shown in bold with underline.

Several observations can be made from the results in this table. As we expect, our methods defeat the original data editing methods on average. For example, the classification accuracy comparison between ENN+ and ENN is 0.743/0.709. "B/W" is 18/2. "S.B/W" is 11/0. In addition, the accuracies of ENN+ and ANN+ are similar and better than RENN+. If considering retention rate, ANN+ is better than ENN+.

For nine of the data sets explored, using data editing methods can improve the classification accuracy of k-NN. However, for other twelve data sets, the data editing methods generate a negative effect. These data sets are iris, vote, heart2, horse, wine, etc. One important observation is that our proposed methods can significantly reduce the negative effect for those data sets. For instance, for eight of these twelve data sets (iris, vote, heart2, horse, breast, yeast, glass, and ionosphere), ENN+ significantly improves the performance of ENN.

As well, Table 14 through 19 show the experiment results when labeled ratio is 10%, 15%, 20%, 30%, 40%, and 50%.

Table 13 Experimental results when labeled ratio is 5%

Dataset	**KNN**	**ENN**	**ENN+**	**RENN**	**RENN+**	**ANN**	**ANN+**
iris	0.893	0.800 (87.6%)	**0.895** (95.1%)	0.763 (82.5%)	**0.893** (94.4%)	0.780 (85.4%)	**0.895** (94.7%)
vote	0.832	0.743 (80.5%)	**0.838** (88.1%)	0.733 (76.9%)	**0.848** (87.3%)	0.742 (77.2%)	**0.832** (85.6%)
heart2	0.752	0.742 (78.2%)	**0.775** (83.9%)	0.737 (75.3%)	**0.775** (82.5%)	0.742 (71.4%)	**0.765** (80.5%)
horse	0.730	0.679 (80.1%)	**0.739** (85.1%)	0.646 (74.3%)	**0.729** (83.6%)	0.679 (71.4%)	**0.760** (79.6%)
sonar	0.617	0.733 (66.0%)	0.733 (72.4%)	0.733 (66.0%)	0.733 (72.4%)	0.733 (46.5%)	0.733 (65.7%)
wine	0.815	**0.781** (57.4%)	0.767 (62.3%)	0.641 (52.7%)	**0.752** (57.6%)	0.741 (51.6%)	**0.800** (53.5%)
breast	0.952	0.935 (93.6%)	**0.957** (95.6%)	0.930 (93.1%)	**0.957** (95.4%)	0.930 (92.1%)	**0.957** (95.1%)
yeast	0.438	0.423 (41.0%)	**0.503** (52.4%)	0.376 (31.0%)	**0.482** (46.9%)	0.418 (32.2%)	0.494 (45.2%)
australian	0.800	0.802 (82.2%)	**0.835** (86.3%)	0.810 (80.0%)	**0.833** (85.6%)	0.790 (75.7%)	**0.826** (82.7%)
bupa	0.574	0.567 (58.8%)	**0.574** (67.2%)	0.527 (47.0%)	**0.584** (58.6%)	0.567 (49.8%)	**0.574** (59.3%)
diabetes	0.703	0.736 (73.1%)	**0.763** (78.0%)	0.711 (68.3%)	**0.746** (75.3%)	0.744 (64.0%)	**0.767** (72.9%)
echo	0.632	0.752 (65.1%)	**0.756** (75.7%)	0.752 (63.7%)	0.752 (70.2%)	**0.752** (55.4%)	0.748 (64.3%)
german	0.627	0.707 (67.6%)	**0.720** (73.9%)	**0.733** (62.6%)	0.720 (71.3%)	0.707 (60.4%)	**0.720** (69.0%)
glass	0.602	0.441 (52.4%)	**0.526** (60.2%)	0.365 (41.5%)	**0.480** (52.7%)	0.435 (47.2%)	**0.516** (55.2%)
magic	0.682	0.675 (63.8%)	**0.691** (68.4%)	0.638 (55.6%)	**0.667** (62.6%)	0.670 (53.1%)	**0.691** (61.2%)
credit	0.716	0.728 (77.0%)	**0.747** (80.9%)	0.730 (74.3%)	**0.748** (80.0%)	0.720 (68.5%)	**0.741** (75.3%)
spect	0.674	0.740 (74.6%)	**0.758** (80.4%)	0.772 (69.6%)	**0.800** (77.8%)	**0.782** (63.3%)	0.766 (73.1%)
wdbc	0.890	0.893 (90.2%)	**0.905** (92.6%)	0.868 (87.1%)	**0.910** (92.3%)	0.898 (87.8%)	**0.904** (91.2%)
ecoli	0.735	**0.718** (68.8%)	0.715 (74.6%)	0.670 (67.6%)	**0.715** (72.7%)	0.698 (61.7%)	**0.715** (70.9%)
ionosphere	0.778	0.670 (73.6%)	**0.734** (75.6%)	0.689 (63.3%)	**0.744** (70.9%)	0.689 (69.0%)	**0.753** (73.0%)
haberm	0.612	0.630 (66.6%)	**0.675** (75.0%)	0.627 (61.9%)	**0.668** (72.1%)	0.640 (55.9%)	**0.667** (67.0%)
Ave.	0.717	0.709 (71.3%)	0.743 (77.1%)	0.688 (66.4%)	0.740 (74.4%)	0.708 (63.8%)	0.744 (72.1%)
B/W		2/18		1/18		2/18	
S.B/W		0/11		0/17		0/14	

Table 14 Experimental results when labeled ratio is 10%

Dataset	KNN	ENN	ENN+	RENN	RENN+	ANN	ANN+
iris	0.893	0.800 (87.6%)	**0.895** (95.1%)	0.763 (82.5%)	**0.893** (94.4%)	0.780 (85.4%)	**0.895** (94.7%)
vote	0.905	0.890 (89.4%)	**0.907** (90.8%)	0.849 (87.5%)	**0.902** (89.7%)	0.881 (85.1%)	**0.913** (88.1%)
heart2	0.743	0.793 (79.0%)	**0.827** (83.1%)	0.803 (77.0%)	**0.823** (82.3%)	0.797 (72.4%)	**0.820** (79.5%)
horse	0.727	0.756 (85.9%)	**0.774** (87.6%)	0.733 (84.0%)	**0.771** (86.8%)	0.738 (77.2%)	**0.764** (82.0%)
sonar	0.700	0.543 (61.4%)	**0.643** (71.5%)	0.533 (51.0%)	**0.630** (64.8%)	0.558 (53.5%)	**0.652** (66.8%)
wine	0.815	**0.781** (57.4%)	0.767 (62.3%)	0.641 (52.7%)	**0.752** (57.6%)	0.741 (51.6%)	**0.800** (53.5%)
breast	0.893	0.917 (93.0%)	**0.992** (94.2%)	0.917 (92.2%)	**0.919** (94.0%)	0.921 (89.4%)	**0.923** (92.6%)
yeast	0.504	0.552 (50.8%)	**0.565** (56.9%)	0.516 (42.0%)	**0.549** (51.2%)	**0.548** (40.9%)	0.545 (48.6%)
australian	0.747	0.791 (81.5%)	**0.807** (85.5%)	0.792 (80.6%)	**0.807** (84.6%)	0.785 (74.3%)	**0.807** (80.2%)
bupa	0.583	0.563 (56.4%)	**0.596** (61.6%)	0.546 (48.0%)	**0.583** (55.8%)	0.546 (44.4%)	**0.602** (53.6%)
diabetes	0.698	0.681 (69.2%)	**0.708** (76.0%)	0.666 (62.3%)	**0.703** (72.6%)	0.688 (60.3%)	**0.714** (70.2%)
echo	0.632	0.752 (65.1%)	**0.756** (71.7%)	0.752 (63.7%)	0.752 (70.2%)	**0.752** (55.4%)	0.748 (64.3%)
german	0.634	0.670 (65.0%)	**0.676** (69.6%)	0.674 (59.6%)	**0.678** (66.8%)	**0.668** (54.1%)	0.666 (62.5%)
glass	0.602	0.441 (52.4%)	**0.526** (60.2%)	0.365 (41.5%)	**0.480** (52.7%)	0.435 (47.2%)	**0.516** (55.2%)
magic	0.706	**0.719** (67.4%)	0.716 (70.8%)	0.695 (60.5%)	**0.696** (65.6%)	**0.718** (57.7%)	0.709 (64.3%)
credit	0.798	0.824 (82.9%)	**0.829** (86.1%)	0.828 (81.4%)	**0.831** (85.4%)	0.814 (76.6%)	**0.828** (82.2%)
spect	0.703	0.751 (74.9%)	**0.753** (79.8%)	**0.757** (67.4%)	0.724 (74.4%)	**0.753** (63.3%)	0.748 (72.4%)
wdbc	0.931	0.928 (92.9%)	**0.931** (94.8%)	0.923 (91.6%)	**0.931** (94.5%)	0.925 (91.0%)	**0.930** (94.2%)
ecoli	0.778	0.679 (74.8%)	**0.766** (80.2%)	0.620 (70.3%)	**0.757** (77.7%)	0.644 (68.7%)	**0.751** (76.1%)
ionosphere	0.754	0.726 (75.1%)	**0.735** (76.6%)	0.698 (69.1%)	**0.718** (72.6%)	0.716 (73.3%)	**0.729** (75.5%)
haberm	0.615	0.627 (64.9%)	**0.691** (71.3%)	0.636 (58.2%)	**0.687** (68.7%)	0.636 (54.1%)	**0.674** (64.8%)
Ave.	0.731	0.723 (72.7%)	0.752 (77.4%)	0.700 (67.8%)	0.742 (74.4%)	0.716 (65.5%)	0.749 (72.4%)
B/W		2/19		1/19		5/16	
S.B/W		0/8		1/12		0/12	

Table 15 Experimental results when labeled ratio is 15%

Dataset	KNN	ENN	ENN+	RENN	RENN+	ANN	ANN+
iris	0.860	0.830	**0.870**	0.732	**0.875**	0.832	**0.878**
		(90.9%)	(94.9%)	(85.6%)	(94.5%)	(86.5%)	(93.6%)
vote	0.863	0.842	**0.849**	0.837	**0.841**	0.852	0.852
		(88.0%)	(89.2%)	(85.9%)	(87.6%)	(81.7%)	(85.6%)
heart2	0.803	**0.844**	0.841	**0.846**	0.841	0.838	**0.839**
		(82.7%)	(84.8%)	(81.1%)	(82.9%)	(75.7%)	(81.9%)
horse	0.703	0.744	**0.760**	0.744	**0.756**	0.764	**0.772**
		(82.3%)	(83.9%)	(81.2%)	(83.2%)	(73.8%)	(77.9%)
sonar	0.703	0.581	**0.649**	0.589	**0.639**	0.567	**0.661**
		(61.8%)	(70.0%)	(52.5%)	(63.7%)	(53.8%)	(64.4%)
wine	0.914	0.802	**0.889**	0.752	**0.836**	0.772	**0.855**
		(68.9%)	(70.5%)	(66.5%)	(66.9%)	(61.7%)	(63.9%)
breast	0.948	0.953	**0.957**	0.951	**0.957**	0.949	**0.955**
		(95.3%)	(95.7%)	(94.3%)	(95.3%)	(93.6%)	(94.7%)
yeast	0.521	0.570	**0.581**	0.554	**0.565**	0.556	**0.577**
		(49.4%)	(58.4%)	(43.3%)	(53.9%)	(40.3%)	(51.3%)
australian	0.763	0.818	**0.822**	0.820	**0.826**	0.818	**0.820**
		(81.1%)	(84.1%)	(79.8%)	(82.9%)	(73.9%)	(79.7%)
bupa	0.599	0.569	**0.621**	0.556	**0.606**	0.567	**0.625**
		(62.1%)	(68.0%)	(53.1%)	(61.4%)	(50.8%)	(58.9%)
diabetes	0.671	0.706	**0.736**	0.693	**0.722**	0.705	**0.734**
		(70.8%)	(76.8%)	(65.8%)	(74.1%)	(60.5%)	(70.9%)
echo	0.615	0.705	**0.715**	0.685	**0.705**	0.700	**0.705**
		(67.1%)	(72.7%)	(61.9%)	(69.6%)	(55.6%)	(66.2%)
german	0.637	0.666	**0.687**	0.686	**0.697**	0.675	**0.683**
		(67.5%)	(70.6%)	(62.4%)	(67.2%)	(56.3%)	(61.9%)
glass	0.602	0.441	**0.526**	0.365	**0.480**	0.435	**0.516**
		(52.4%)	(60.2%)	(41.5%)	(52.7%)	(47.2%)	(55.2%)
magic	0.729	**0.738**	0.736	**0.729**	0.710	**0.744**	0.734
		(69.6%)	(72.2%)	(63.2%)	(67.6%)	(58.9%)	(66.0%)
credit	0.789	0.824	**0.831**	0.821	**0.830**	0.820	**0.831**
		(83.7%)	(85.9%)	(82.8%)	(85.3%)	(76.5%)	(82.2%)
spect	0.688	**0.709**	0.688	**0.710**	0.671	0.691	**0.697**
		(72.3%)	(78.0%)	(64.2%)	(71.7%)	(63.7%)	(70.8%)
wdbc	0.936	0.932	**0.933**	**0.935**	0.934	0.929	**0.936**
		(94.5%)	(94.6%)	(93.2%)	(93.9%)	(92.1%)	(93.3%)
ecoli	0.730	0.734	**0.779**	0.687	**0.769**	0.730	**0.795**
		(75.6%)	(82.6%)	(71.9%)	(81.2%)	(70.6%)	(79.3%)
ionosphere	0.752	0.716	**0.735**	0.677	**0.723**	0.712	**0.728**
		(73.8%)	(74.5%)	(67.7%)	(72.7%)	(70.5%)	(72.6%)
haberm	0.645	0.710	**0.730**	0.688	**0.729**	0.700	**0.724**
		(68.8%)	(74.8%)	(63.0%)	(72.8%)	(57.8%)	(68.6%)
Ave.	0.737	0.735	0.759	0.717	0.748	0.731	0.758
		(74.2%)	(78.2%)	(69.6%)	(75.3%)	(66.7%)	(73.3%)
B/W		3/18		4/17		1/19	
S.B/W		1/9		1/9		0/9	

Table 16 Experimental results when labeled ratio is 20%

Dataset	**KNN**	**ENN**	**ENN+**	**RENN**	**RENN+**	**ANN**	**ANN+**
iris	0.927	**0.900**	0.880	**0.893**	0.874	**0.887**	0.880
		(92.3%)	(96.0%)	(90.6%)	(95.8%)	(89.0%)	(95.7%)
vote	0.907	0.910	**0.917**	0.908	**0.914**	0.912	**0.915**
		(92.6%)	(92.3%)	(90.2%)	(91.1%)	(90.0%)	(90.6%)
heart2	0.739	0.793	**0.796**	0.795	**0.796**	**0.804**	0.793
		(76.4%)	(80.4%)	(75.5%)	(79.8%)	(70.0%)	(76.7%)
horse	0.730	**0.776**	0.771	**0.783**	0.766	**0.777**	0.775
		(83.9%)	(87.4%)	(82.1%)	(86.6%)	(77.6%)	(82.7%)
sonar	0.722	0.695	**0.728**	0.643	**0.698**	0.702	**0.708**
		(62.2%)	(16.6%)	(52.1%)	(51.3%)	(54.1%)	(48.1%)
wine	0.944	0.862	**0.940**	0.769	**0.907**	0.827	**0.902**
		(65.6%)	(69.6%)	(60.9%)	(64.4%)	(58.9%)	(64.6%)
breast	0.941	0.959	0.959	0.954	**0.959**	0.957	**0.959**
		(95.3%)	(96.4%)	(94.1%)	(95.7%)	(93.3%)	(95.3%)
yeast	0.532	0.566	**0.597**	0.566	**0.589**	0.560	**0.595**
		(53.2%)	(59.7%)	(46.7%)	(57.0%)	(44.2%)	(52.2%)
australian	0.800	0.832	**0.842**	0.833	**0.847**	0.839	**0.849**
		(84.0%)	(85.2%)	(82.9%)	(84.3%)	(78.2%)	(81.4%)
bupa	0.504	0.576	**0.579**	**0.596**	0.595	0.573	**0.611**
		(60.8%)	(66.2%)	(51.7%)	(61.8%)	(49.3%)	(57.8%)
diabetes	0.705	0.732	**0.740**	0.696	**0.731**	0.716	**0.730**
		(71.9%)	(76.3%)	(66.3%)	(73.9%)	(63.7%)	(71.4%)
echo	0.611	0.638	**0.646**	0.624	**0.640**	0.638	**0.648**
		(62.1%)	(68.1%)	(52.4%)	(62.8%)	(50.0%)	(59.0%)
german	0.684	0.697	**0.701**	0.700	**0.703**	0.693	**0.703**
		(65.1%)	(69.7%)	(61.1%)	(66.1%)	(54.9%)	(62.1%)
glass	0.580	0.494	**0.551**	0.474	**0.519**	0.484	**0.536**
		(54.1%)	(58.7%)	(46.1%)	(53.8%)	(49.5%)	(54.2%)
magic	0.735	**0.740**	0.737	**0.734**	0.731	**0.745**	0.742
		(70.9%)	(73.7%)	(64.7%)	(69.5%)	(61.5%)	(67.6%)
credit	0.763	0.802	**0.823**	0.805	**0.824**	0.795	**0.813**
		(83.3%)	(85.1%)	(81.7%)	(84.1%)	(74.8%)	(79.6%)
spect	0.707	**0.735**	0.729	**0.729**	0.717	0.759	**0.761**
		(73.7%)	(75.0%)	(68.5%)	(70.4%)	(62.6%)	(65.8%)
wdbc	0.964	0.950	**0.957**	0.939	**0.957**	0.950	**0.956**
		(96.0%)	(97.0%)	(94.9%)	(96.6%)	(95.0%)	(96.2%)
ecoli	0.787	0.804	**0.821**	0.770	**0.821**	0.808	**0.828**
		(81.2%)	(86.0%)	(78.3%)	(85.0%)	(74.4%)	(82.9%)
ionosphere	0.820	0.776	0.776	0.744	**0.761**	0.776	0.776
		(80.8%)	(81.4%)	(76.4%)	(78.8%)	(78.6%)	(79.9%)
haberm	0.679	0.733	**0.760**	0.747	**0.750**	0.737	**0.757**
		(72.6%)	(77.0%)	(67.1%)	(75.2%)	(62.8%)	(72.4%)
Ave.	0.751	0.760	0.774	0.748	0.767	0.759	0.773
		(75.2%)	(76.1%)	(70.7%)	(73.2%)	(68.2%)	(71.1%)
B/W		4/15		5/16		4/16	
S.B/W		0/6		0/6		0/6	

Table 17 Experimental results when labeled ratio is 30%

Dataset	**KNN**	**ENN**	**ENN+**	**RENN**	**RENN+**	**ANN**	**ANN+**
iris	0.917	**0.919** (93.1%)	0.917 (97.4%)	0.909 (92.0%)	**0.917** (97.3%)	0.919 (92.8%)	**0.922** (97.1%)
vote	0.919	0.920 (93.8%)	**0.923** (94.3%)	**0.917** (91.2%)	0.912 (93.3%)	0.920 (90.5%)	**0.922** (92.3%)
heart2	0.754	0.779 (79.7%)	**0.795** (83.4%)	0.789 (77.7%)	**0.804** (82.8%)	0.784 (73.0%)	**0.804** (79.8%)
horse	0.760	0.805 (84.9%)	**0.813** (86.2%)	0.814 (82.7%)	**0.820** (85.7%)	0.806 (77.6%)	**0.807** (82.7%)
sonar	0.742	0.669 (65.2%)	**0.711** (69.4%)	0.656 (58.5%)	**0.694** (63.2%)	0.687 (58.5%)	**0.728** (63.2%)
wine	0.933	0.889 (63.9%)	**0.911** (69.2%)	0.809 (59.5%)	**0.889** (63.9%)	0.867 (57.6%)	**0.904** (63.7%)
breast	0.959	0.968 (96.5%)	0.968 (97.0%)	0.966 (95.7%)	0.966 (96.7%)	**0.967** (94.9%)	0.966 (96.3%)
yeast	0.534	0.578 (52.7%)	0.578 (59.4%)	**0.586** (47.1%)	0.577 (56.1%)	0.574 (42.9%)	0.574 (52.1%)
australian	0.801	0.838 (84.5%)	**0.840** (85.6%)	0.838 (82.9%)	**0.842** (84.5%)	**0.842** (78.2%)	0.837 (82.1%)
bupa	0.606	0.605 (61.4%)	**0.608** (63.4%)	**0.596** (53.6%)	0.584 (57.5%)	0.596 (50.3%)	**0.603** (54.6%)
diabetes	0.661	0.690 (69.8%)	**0.721** (76.3%)	0.694 (63.8%)	**0.726** (73.7%)	0.702 (60.1%)	**0.722** (70.4%)
echo	0.620	0.640 (59.9%)	**0.670** (67.3%)	0.640 (51.7%)	**0.670** (63.1%)	0.665 (46.8%)	**0.695** (57.4%)
german	0.644	0.683 (63.9%)	**0.689** (67.6%)	0.678 (58.5%)	**0.685** (64.7%)	0.679 (52.9%)	**0.690** (59.6%)
glass	0.669	0.624 (59.4%)	**0.641** (63.9%)	0.594 (53.6%)	**0.597** (59.3%)	0.609 (55.6%)	**0.648** (60.8%)
magic	0.723	**0.737** (70.7%)	0.732 (73.0%)	**0.735** (65.2%)	0.717 (68.7%)	**0.735** (61.4%)	0.732 (67.1%)
credit	0.782	0.837 (83.3%)	**0.842** (85.6%)	0.834 (82.3%)	**0.845** (84.9%)	0.831 (75.1%)	**0.842** (81.0%)
spect	0.739	**0.758** (76.1%)	0.727 (75.7%)	0.718 (71.1%)	**0.721** (69.8%)	**0.748** (65.2%)	0.726 (68.6%)
wdbc	0.949	0.949 (95.5%)	**0.959** (97.0%)	0.948 (96.8%)	**0.958** (96.5%)	0.951 (93.6%)	**0.960** (95.6%)
ecoli	0.779	0.807 (81.8%)	**0.827** (86.7%)	0.766 (77.8%)	**0.815** (85.6%)	0.791 (74.3%)	**0.819** (82.4%)
ionosphere	0.833	0.784 (81.2%)	**0.804** (82.5%)	0.758 (75.1%)	**0.800** (80.6%)	0.788 (78.9%)	**0.811** (80.7%)
haberm	0.645	0.701 (70.0%)	**0.737** (75.4%)	0.735 (64.9%)	**0.741** (74.8%)	0.713 (59.3%)	**0.734** (68.0%)
Ave.	0.760	0.770 (75.5%)	0.782 (78.9%)	0.761 (71.4%)	0.775 (76.3%)	0.770 (68.6%)	0.783 (74.1%)
B/W		3/16		4/16		4/16	
S.B/W		1/7		0/6		1/8	

Table 18 Experimental results when labeled ratio is 40%

Dataset	**KNN**	**ENN**	**ENN+**	**RENN**	**RENN+**	**ANN**	**ANN+**
iris	0.933	0.930	**0.937**	0.930	**0.937**	0.937	**0.940**
		(95.4%)	(96.4%)	(95.2%)	(96.0%)	(93.5%)	(95.6%)
vote	0.902	**0.904**	0.903	**0.903**	0.894	**0.913**	0.906
		(91.9%)	(92.2%)	(90.0%)	(91.3%)	(88.9%)	(90.2%)
heart2	0.769	0.807	**0.815**	**0.817**	0.815	0.808	**0.819**
		(77.4%)	(80.2%)	(75.1%)	(78.9%)	(71.7%)	(76.6%)
horse	0.752	0.785	**0.786**	**0.785**	0.783	0.787	**0.790**
		(82.6%)	(84.8%)	(80.6%)	(86.5%)	(75.8%)	(79.1%)
sonar	0.810	0.700	**0.738**	0.670	**0.715**	0.696	**0.734**
		(70.8%)	(77.4%)	(64.2%)	(73.5%)	(64.1%)	(74.4%)
wine	0.950	**0.940**	0.925	0.917	**0.922**	**0.933**	0.925
		(66.2%)	(70.3%)	(62.8%)	(65.5%)	(61.4%)	(66.5%)
breast	0.955	**0.969**	0.968	0.968	0.968	**0.969**	0.968
		(96.7%)	(97.5%)	(96.2%)	(97.1%)	(95.6%)	(96.8%)
yeast	0.528	0.582	**0.600**	0.597	**0.603**	0.592	**0.603**
		(53.1%)	(59.9%)	(47.5%)	(57.9%)	(44.2%)	(53.2%)
australian	0.797	**0.851**	0.849	**0.853**	0.848	0.852	0.852
		(82.5%)	(85.5%)	(81.7%)	(84.0%)	(76.4%)	(80.9%)
bupa	0.599	0.615	**0.627**	**0.621**	0.613	0.608	**0.617**
		(61.5%)	(62.1%)	(53.8%)	(56.7%)	(49.6%)	(53.5%)
diabetes	0.706	0.727	**0.746**	0.729	**0.741**	0.724	**0.747**
		(72.3%)	(77.2%)	(67.0%)	(74.4%)	(64.3%)	(71.7%)
echo	0.617	0.666	**0.681**	0.671	**0.698**	0.686	**0.689**
		(66.1%)	(69.0%)	(59.4%)	(63.9%)	(55.8%)	(60.4%)
german	0.657	0.704	**0.713**	0.704	**0.713**	0.705	**0.711**
		(67.1%)	(71.2%)	(62.7%)	(68.4%)	(57.3%)	(63.3%)
glass	0.652	0.554	**0.600**	0.550	**0.577**	0.561	**0.596**
		(56.2%)	(61.1%)	(49.4%)	(56.1%)	(51.7%)	(56.8%)
magic	0.736	**0.757**	0.750	**0.751**	0.738	**0.750**	0.748
		(73.1%)	(74.8%)	(67.4%)	(71.6%)	(64.1%)	(69.2%)
credit	0.804	**0.856**	0.855	0.853	**0.857**	0.858	**0.863**
		(85.3%)	(87.0%)	(84.0%)	(86.0%)	(78.8%)	(83.1%)
spect	0.709	**0.756**	0.725	**0.778**	0.697	**0.760**	0.722
		(75.4%)	(76.6%)	(69.4%)	(70.8%)	(65.2%)	(69.5%)
wdbc	0.941	0.952	**0.953**	0.947	**0.952**	**0.953**	0.951
		(95.4%)	(95.8%)	(94.6%)	(95.3%)	(93.1%)	(93.9%)
ecoli	0.803	0.820	**0.852**	0.818	**0.853**	0.813	**0.851**
		(83.8%)	(88.3%)	(81.4%)	(86.9%)	(77.1%)	(84.2%)
ionosphere	0.857	0.825	**0.834**	0.817	**0.825**	0.824	**0.834**
		(83.9%)	(84.4%)	(80.5%)	(83.3%)	(81.9%)	(82.5%)
haberm	0.697	0.732	**0.761**	0.764	**0.772**	0.740	**0.763**
		(72.9%)	(76.4%)	(69.1%)	(73.3%)	(64.2%)	(71.5%)
Ave.	0.770	0.783	0.791	0.783	0.787	0.784	0.792
		(76.7%)	(79.4%)	(73.0%)	(77.0%)	(78.4%)	(79.2%)
B/W		7/14		7/13		6/14	
S.B/W		1/4		1/4		1/5	

Table 19 Experimental results when labeled ratio is 50%

Dataset	**KNN**	**ENN**	**ENN+**	**RENN**	**RENN+**	**ANN**	**ANN+**
iris	0.936	0.935 (95.3%)	**0.944** (96.6%)	0.935 (94.9%)	**0.944** (96.6%)	0.941 (94.5%)	**0.946** (96.4%)
vote	0.923	**0.924** (92.4%)	0.922 (93.0%)	**0.919** (91.0%)	0.918 (92.2%)	**0.921** (90.7%)	0.919 (91.6%)
heart2	0.790	0.773 (78.7%)	**0.781** (80.3%)	0.771 (75.9%)	**0.776** (77.8%)	0.775 (72.8%)	**0.784** (75.8%)
horse	0.755	0.785 (82.5%)	**0.795** (83.2%)	0.795 (80.5%)	**0.798** (85.5%)	0.784 (74.2%)	**0.792** (77.3%)
sonar	0.811	0.738 (75.4%)	**0.780** (80.4%)	0.705 (70.9%)	**0.766** (78.2%)	0.722 (72.1%)	**0.774** (77.9%)
wine	0.929	0.902 (66.8%)	**0.920** (71.5%)	0.851 (62.3%)	**0.923** (68.4%)	0.901 (61.0%)	**0.914** (66.7%)
breast	0.958	0.962 (97.0%)	**0.963** (97.4%)	0.962 (96.5%)	0.962 (97.0%)	**0.964** (95.4%)	0.962 (96.4%)
yeast	0.567	0.593 (52.0%)	**0.597** (56.9%)	0.597 (46.3%)	**0.602** (54.0%)	0.595 (43.1%)	**0.598** (49.9%)
australian	0.793	**0.848** (84.2%)	0.846 (85.5%)	**0.848** (83.3%)	0.845 (84.7%)	**0.845** (76.9%)	0.842 (80.3%)
bupa	0.598	**0.628** (65.3%)	0.627 (67.3%)	**0.636** (59.6%)	0.633 (62.9%)	0.612 (51.7%)	**0.632** (55.4%)
diabetes	0.693	0.716 (70.8%)	**0.734** (76.1%)	0.712 (65.4%)	**0.729** (72.6%)	0.717 (61.9%)	**0.729** (69.8%)
echo	0.600	0.672 (64.1%)	**0.688** (68.9%)	0.648 (54.1%)	**0.692** (65.4%)	0.663 (51.7%)	**0.692** (57.8%)
german	0.658	0.694 (66.5%)	**0.702** (68.3%)	**0.698** (61.0%)	0.696 (64.4%)	0.699 (56.6%)	**0.707** (60.5%)
glass	0.690	0.622 (63.3%)	**0.655** (66.2%)	0.607 (58.0%)	**0.637** (60.6%)	0.623 (57.1%)	**0.665** (61.0%)
magic	0.741	**0.752** (72.3%)	0.747 (74.0%)	**0.746** (67.4%)	0.737 (69.9%)	**0.752** (63.4%)	0.748 (66.9%)
credit	0.778	0.831 (83.1%)	**0.840** (84.8%)	0.836 (81.6%)	**0.841** (83.5%)	0.827 (75.8%)	**0.832** (79.7%)
spect	0.699	**0.707** (70.6%)	0.686 (73.4%)	**0.719** (65.6%)	0.699 (66.0%)	**0.709** (61.1%)	0.687 (65.5%)
wdbc	0.942	0.948 (95.4%)	**0.950** (95.8%)	0.943 (94.4%)	**0.948** (95.5%)	0.950 (93.5%)	**0.951** (94.2%)
ecoli	0.805	0.825 (82.7%)	**0.842** (86.7%)	0.826 (80.3%)	**0.839** (85.6%)	0.831 (76.4%)	**0.843** (82.7%)
ionosphere	0.840	0.818 (83.7%)	**0.822** (84.5%)	0.801 (81.2%)	**0.825** (83.7%)	0.814 (80.6%)	**0.822** (82.1%)
haberm	0.640	0.718 (67.9%)	**0.735** (72.8%)	0.730 (64.9%)	**0.746** (77.1%)	0.727 (56.5%)	**0.737** (63.6%)
Ave.	0.769	0.781 (76.7%)	0.790 (79.2%)	0.775 (73.0%)	0.788 (77.2%)	0.780 (69.9%)	0.789 (73.9%)
B/W		5/16		6/14		5/16	
S.B/W		1/2		1/5		1/4	

Table 20 Experimental results summarization when labeled ratio is 5%, 10%, 15%, 20%, 30%, 40%, and 50%

		labeled ratio						
		5%	10%	15%	20%	30%	40%	50%
ENN+ & ENN	Accu.	**0.743** **0.709** **(0.034)**	0.752 0.723 (0.029)	0.759 0.735 (0.024)	0.774 0.760 (0.014)	0.782 0.770 (0.012)	0.791 0.783 (0.008)	0.790 0.781 (0.009)
	B/W	18/2 (16)	**19/2** **(17)**	18/3 (15)	15/4 (11)	16/3 (13)	14/7 (7)	16/5 (11)
	S.B/W	**11/0** **(11)**	8/0 (8)	9/1 (8)	6/0 (6)	7/1 (6)	4/1 (3)	2/1 (1)
RENN+ & RENN	Accu.	**0.740** **0.688** **(0.052)**	0.742 0.700 (0.042)	0.748 0.717 (0.031)	0.767 0.748 (0.019)	0.775 0.761 (0.014)	0.787 0.783 (0.004)	0.788 0.775 (0.013)
	B/W	18/1 (17)	**19/1** **(18)**	17/4 (13)	16/5 (11)	16/4 (12)	13/7 (6)	14/6 (8)
	S.B/W	**17/0** **(17)**	12/1 (11)	9/1 (8)	6/0 (6)	6/0 (6)	4/1 (3)	5/1 (4)
ANN+ & ANN	Accu.	**0.744** **0.708** **(0.036)**	0.749 0.716 (0.033)	0.758 0.731 (0.027)	0.773 0.759 (0.014)	0.783 0.770 (0.013)	0.792 0.784 (0.008)	0.789 0.780 (0.009)
	B/W	18/2 (16)	16/5 (11)	**19/1** **(18)**	16/4 (12)	16/4 (12)	14/6 (8)	16/5 (11)
	S.B/W	**14/0** **(14)**	12/0 (12)	9/0 (9)	6/0 (6)	8/1 (7)	5/1 (4)	4/1 (3)

We summarize the results from different labeled ratios in Table 20. In this Table, only classification accuracy is considered and data retention ratio is neglected. In each cell, not only the performance comparison, but the degree of improvement is shown in brackets. The greatest improvement among different labeled ratios is shown in bold.

From the experimental results, we obtain the similar observations as before. These experiments further validate the superiority of our methods since they work well for different labeled ratios.

There is another useful result shown in experiments. In the case that labeled ratio is 5%, 10%, or 15%, greater improvement will be achieved by unlabeled data compared to 20%, 30%, 40%, and 50%. That is to say, unlabeled data will give more remarkable improvement for data editing in case that labeled ratio is small (labeled data number is small). Existing work on data editing [43] shows that the small set-size editing failure mainly stem for the inability of the nearest neighbor instances to achieve sufficiently reliable estimates for this instance. Using unlabeled data seems to be promising to solve this intrinsic limitation of data editing.

3.4.3 Experimental results: varying the noise ratio

In this part, we will fix the labeled ratio to 50% and vary the noise ratio including 10%, 20%, 30%, and 40%.

Nearest neighbor editing (NNE) is originally proposed to increase the generality ability of k-nearest neighbor by removing noisy training instances. Attracted by its straightforwardness and good performance, we intend to explore whether nearest neighbor editing can be used with other machine learning algorithms. We choose naïve Bayes and decision tree in the experiments.

In addition to this motivation, there are other two objectives for this experiment: 1) test whether our method works well under different noise ratios, and 2) whether there is any relationship between the noise ratio and the performance of our method.

The differences of experimental setup with the experiments in Section 3.4.2 include: 1) the number of iteration for ensemble-based data labeling function is 3, and 2) we use classification error rate to evaluate the performance of our method.

$$\text{classification error rate} = \frac{\text{No. of incorrect classifications on testing instances}}{\text{No. of testing instances}}$$

The experiments can be divided into two parts:

Experiment A: the noise ratios are 10%, 20%, 30%, and 40%. The objective of Experiment A is to test whether traditional NNE methods can improve the classification performance of 1-NN, naïve Bayes, and decision tree under different noise ratios.

Experiment B: the noise ratios are 10%, 20%, 30%, and 40%. The objective of Experiment B is to test whether our proposed method can outperform traditional nearest neighbor editing under different noise ratios when applied to 1-NN, naïve Bayes, and decision tree.

Experiment A: Fig. 7 shows the detailed result when noise ratio is 10%. The values in this table and following tables represent the classification error rates. The result in Fig. 7 includes three parts: 1) the classification error rates of each classification algorithm (including both before and after applying nearest neighbor editing methods) on each data set, 2) the average classification error rates of each algorithm (including both before and after applying nearest neighbor editing methods) across all the data sets, and 3) the improvement of each NNE method acting on each classification algorithm with respect to the reduction of the average classification error.

Data	k-NN				Naive Bayes				Decision Tree			
	kNN	ENN	RENN	ANN	NB	ENN	RENN	ANN	DT	ENN	RENN	ANN
iris	0.145	0.076	0.081	0.082	0.084	0.059	0.064	0.056	0.091	0.077	0.069	0.076
vote	0.165	0.078	0.081	0.086	0.098	0.095	0.096	0.09	0.046	0.045	0.042	0.045
heart2	0.262	0.208	0.200	0.211	0.197	0.195	0.206	0.191	0.212	0.197	0.200	0.196
horse	0.284	0.204	0.202	0.22	0.269	0.209	0.193	0.207	0.202	0.167	0.17	0.179
sonar	0.231	0.294	0.343	0.305	0.341	0.326	0.378	0.324	0.343	0.345	0.345	0.333
wine	0.139	0.086	0.095	0.09	0.051	0.107	0.145	0.122	0.161	0.212	0.265	0.234
breast	0.125	0.036	0.034	0.037	0.037	0.042	0.043	0.043	0.062	0.047	0.05	0.05
yeast	0.49	0.419	0.405	0.421	0.46	0.405	0.394	0.403	0.439	0.408	0.400	0.407
aus.	0.266	0.171	0.16	0.172	0.215	0.184	0.175	0.181	0.178	0.149	0.155	0.147
bupa	0.421	0.384	0.367	0.406	0.459	0.449	0.431	0.447	0.41	0.383	0.372	0.403
dia.	0.339	0.302	0.288	0.29	0.249	0.262	0.278	0.263	0.279	0.274	0.281	0.284
echo	0.403	0.34	0.358	0.325	0.283	0.277	0.352	0.307	0.378	0.338	0.358	0.343
german	0.351	0.307	0.299	0.296	0.253	0.26	0.273	0.261	0.304	0.292	0.304	0.285
glass	0.433	0.433	0.472	0.446	0.523	0.508	0.502	0.515	0.397	0.448	0.457	0.448
magic	0.309	0.258	0.259	0.261	0.332	0.31	0.300	0.308	0.246	0.242	0.244	0.243
credit	0.275	0.184	0.176	0.181	0.248	0.213	0.207	0.206	0.199	0.154	0.157	0.153
spect	0.37	0.306	0.277	0.304	0.413	0.259	0.236	0.274	0.312	0.249	0.274	0.247
wdbc	0.138	0.059	0.055	0.059	0.068	0.065	0.065	0.068	0.114	0.072	0.077	0.079
ecoli	0.288	0.195	0.193	0.195	0.182	0.209	0.219	0.221	0.231	0.2	0.226	0.21
ionos	0.224	0.209	0.23	0.204	0.186	0.165	0.24	0.164	0.164	0.146	0.219	0.157
Ave.	0.283	0.227	0.229	0.229	0.247	0.23	0.24	0.233	0.238	0.222	0.233	0.226
Imp.		19.8%	19.1%	19.1%		6.9%	2.8%	5.7%		6.7%	2.1%	5.0%

Fig. 7. ENN, RENN, and ANN on k-NN, naïve Bayes, and decision tree when noise ratio is 10%

An important observation obtained from this table is that NNE methods can improve the performance not only for k-NN, but also for naïve Bayes and decision tree. In detail, the average improvements of ENN, RENN, and ANN on k-NN are 19.8%, 19.1%, and 19.1% respectively. The average improvements of ENN, RENN, and ANN on naïve Bayes are 6.9%, 2.8%, and 5.7% respectively. Their average improvements on decision tree are 6.7%, 2.1%, and 5.0% respectively.

Rather than presenting the experimental results for every noise ratio with details, afterwards Table 21 summarizes the results under four different noise ratios. Table 21 includes three parts: 1) the average classification error rates of each algorithm under each noise ratio across all the data sets, 2) the further average classification error rates of each algorithm across four noise ratios, and 3) the improvement of each nearest neighbor editing method acting on each classification algorithm with respect to the reduction of the average classification error.

Table 21 Summarization of ENN, RENN, and ANN on k-NN, naïve Bayes, and decision tree

Alg. / noise ratio	k-NN (kNN)				Naïve Bayes (NB)				Decision Tree (DT)			
	kNN	ENN	RENN	ANN	NB	ENN	RENN	ANN	DT	ENN	RENN	ANN
10%	0.283	0.227	0.229	0.229	0.247	0.23	0.24	0.233	0.238	0.222	0.233	0.226
20%	0.334	0.247	0.243	0.245	0.256	0.241	0.247	0.242	0.275	0.243	0.246	0.245
30%	0.382	0.295	0.284	0.292	0.285	0.265	0.272	0.273	0.326	0.283	0.281	0.291
40%	0.439	0.366	0.353	0.371	0.318	0.312	0.317	0.318	0.395	0.36	0.349	0.366
Ave.	0.359	0.284	0.277	0.284	0.276	0.262	0.268	0.266	0.308	0.277	0.277	0.282
Imp.		20.9%	22.8%	20.9%		5.1%	2.9%	3.6%		10.1%	10.1%	8.4%

This table shows that with any of the noise ratios, all the three NNE methods could improve the performance of k-NN, naïve Bayes, and decision tree.

Therefore, NNE methods could improve the performance of these classification algorithms consistently and robustly. Finally by averaging the classification error rates of four noise ratios, we find that the improvements of ENN, RENN, and ANN on k-NN are 20.9%, 22.8%, and 20.9% respectively; on naïve Bayes are 5.1%, 2.9%, and 3.6% respectively; on decision tree are 10.1%, 10.1%, and 8.4% respectively. In addition to this, Table 9 indicates that without noise removing, among the three classifiers, on average naïve Bayes has the lowest classification error (0.276). Next is decision tree (0.308), k-NN has the highest classification error (0.359). For all the three NNE methods, they have the best improvement on k-NN; next is decision tree; then naïve Bayes.

Experiment B: The benefit of existing NNE methods on k-NN, naïve Bayes, and decision tree has been tested by Experiment A. In this part, we aim to evaluate whether our proposed method could defeat the original NNE and has better improvement on the three classifiers. Rather than presenting the result for every noise ratios with details, we present the result when noise ratio is 40% for analyzing purpose. In addition, the results for all the noise ratios are summarized. The experimental results in this part are presented according to the classification algorithms (B1) naïve Bayes, (B2) decision tree, (B3) k-nearest neighbor.

(B1) Naïve Bayes. Table 22 shows the comparison between NNE and NNEAUD on naïve Bayes when noise ratio is 40%.

It indicates across all the data sets, on average, NNEAUD-based methods can significantly improve the performance of traditional NNE methods. For example, the average improvement of ENN+ on ENN is 18.3%

Table 23 summarizes the results of NNE and NNEAUD on naïve Bayes under four different noise ratios. It shows that NNEAUD methods could consistently improve the performance of NNE methods on almost all the noise ratios (with the only exception: ENN+ has no improvement on ENN under noise ratio is 10%). This suggests that our propose NNEAUD can be used as an effective noise removing method for naïve Bayes.

Table 22 Comparison between NNE and NNEAUD on naïve Bayes when noise ratio is 40%

Nearest neighbor editing on naïve Bayes, 40%						
Dataset	ENN	ENN+	RENN	RENN+	ANN	ANN+
iris	0.182	0.072	0.179	0.065	0.199	0.07
vote	0.13	0.112	0.139	0.111	0.147	0.107
heart2	0.278	0.214	0.292	0.224	0.276	0.22
horse	0.326	0.235	0.314	0.217	0.332	0.236
sonar	0.463	0.383	0.458	0.361	0.451	0.359
wine	0.22	0.133	0.309	0.15	0.281	0.165
breast	0.044	0.038	0.052	0.052	0.046	0.039
yeast	0.435	0.405	0.44	0.402	0.449	0.403
australian	0.247	0.202	0.236	0.188	0.256	0.193
bupa	0.494	0.484	0.478	0.46	0.481	0.48
diabetes	0.338	0.272	0.35	0.268	0.338	0.276
echo	0.417	0.303	0.402	0.308	0.425	0.322
german	0.36	0.295	0.363	0.292	0.361	0.295
glass	0.541	0.498	0.588	0.518	0.569	0.499
magic	0.35	0.357	0.348	0.357	0.348	0.356
credit	0.272	0.208	0.26	0.2	0.289	0.204
spect	0.455	0.405	0.375	0.317	0.409	0.369
wdbc	0.112	0.068	0.107	0.064	0.124	0.069
ecoli	0.312	0.209	0.303	0.191	0.3	0.219
ionos	0.27	0.218	0.346	0.212	0.283	0.202
Ave.	0.312	0.255	0.317	0.248	0.318	0.254
Imp.	18.3%		21.8%		20.1%	

Table 23 Summarization of comparisons between NNE and NNEAUD on naïve Bayes

Naïve Bayes									
	ENN	ENN+	**Imp**.	RENN	RENN+	**Imp.**	ANN	ANN+	**Imp.**
10%	0.230	0.230	0%	0.238	0.230	3.4%	0.233	0.232	0.4%
20%	0.241	0.233	3.3%	0.247	0.230	6.9%	0.242	0.230	5.0%
30%	0.265	0.241	9.1%	0.272	0.239	12.1%	0.273	0.242	11.4%
40%	0.312	0.255	18.3%	0.317	0.248	21.8%	0.318	0.254	20.1%
Ave.	0.262	0.240	7.7%	0.269	0.237	11.1%	0.267	0.240	9.2%

(B2) Decision tree. Table 24 shows the comparison between NNE and NNEAUD on decision tree when noise ratio is 40%.

Table 25 shows the summarization of comparisons on decision tree. These two tables demonstrate that no matter what noise ratio, NNEAUD methods always outperform NNE methods. Averaged across all the four noise ratios, the improvement of ENN+ on ENN is 14.8%; the improvement of RENN+ on RENN is 15.7%; the improvement of ANN+ on ANN is 16.1%. This observation suggests that NNEAUD is capable to effectively remove noises in the training set for decision tree.

Table 24 Comparison between NNE and NNEAUD on decision tree when noise ratio is 40%

Nearest neighbor editing on decision tree, 40%						
Dataset	ENN	ENN+	RENN	RENN+	ANN	ANN+
iris	0.196	0.073	0.205	0.065	0.216	0.079
vote	0.276	0.151	0.238	0.096	0.284	0.151
heart2	0.343	0.231	0.32	0.216	0.352	0.247
horse	0.354	0.208	0.326	0.182	0.342	0.209
sonar	0.467	0.403	0.452	0.405	0.449	0.383
wine	0.335	0.244	0.408	0.28	0.372	0.242
breast	0.206	0.095	0.189	0.071	0.232	0.096
yeast	0.453	0.412	0.442	0.409	0.465	0.405
australian	0.323	0.167	0.305	0.172	0.328	0.172
bupa	0.462	0.402	0.435	0.39	0.458	0.403
diabetes	0.385	0.285	0.374	0.275	0.37	0.271
echo	0.44	0.328	0.403	0.328	0.41	0.325
german	0.419	0.309	0.39	0.301	0.411	0.308
glass	0.507	0.469	0.559	0.5	0.55	0.494
magic	0.356	0.295	0.361	0.304	0.351	0.289
credit	0.363	0.185	0.308	0.151	0.349	0.176
spect	0.399	0.311	0.367	0.318	0.419	0.318
wdbc	0.269	0.112	0.235	0.08	0.292	0.112
ecoli	0.308	0.208	0.312	0.207	0.327	0.227
ionos	0.348	0.202	0.345	0.189	0.346	0.197
Ave.	0.36	0.254	0.349	0.247	0.366	0.255
Imp.	29.4%		29.2%		30.3%	

Table 25 Summarization of comparisons between NNE and NNEAUD on decision tree

Decision tree									
	ENN	ENN+	**Imp**.	RENN	RENN+	**Imp.**	ANN	ANN+	**Imp.**
10%	0.222	0.212	4.5%	0.233	0.216	7.3%	0.226	0.212	6.2%
20%	0.243	0.225	7.4%	0.246	0.226	8.1%	0.245	0.225	8.2%
30%	0.283	0.232	18.0%	0.281	0.230	18.1%	0.291	0.233	19.9%
40%	0.360	0.254	29.4%	0.349	0.247	29.2%	0.366	0.255	30.3%
Ave.	0.277	0.231	14.8%	0.277	0.230	15.7%	0.282	0.231	16.1%

(B3) The k-nearest neighbor algorithm. Table 26 shows the comparison between NNE and NNEAUD on k-nearest neighbor when noise ratio is 40%.

Table 27 shows the summarization of comparison on k-NN. Same with previous experiments, we find that NNEAUD can consistently defeat NNE on k-NN.

Table 26 Comparison between NNE and NNEAUD on k-NN when noise ratio is 40%

Nearest neighbor editing on k-NN, 40%						
Dataset	ENN	ENN+	RENN	RENN+	ANN	ANN+
iris	0.184	0.082	0.17	0.067	0.176	0.073
vote	0.296	0.214	0.264	0.184	0.285	0.198
heart2	0.325	0.228	0.327	0.218	0.348	0.219
horse	0.374	0.264	0.34	0.235	0.378	0.268
sonar	0.436	0.33	0.451	0.327	0.444	0.343
wine	0.235	0.137	0.277	0.152	0.257	0.123
breast	0.273	0.162	0.213	0.122	0.279	0.147
yeast	0.468	0.426	0.448	0.413	0.471	0.42
australian	0.36	0.204	0.334	0.182	0.363	0.209
bupa	0.48	0.461	0.477	0.445	0.499	0.455
diabetes	0.417	0.313	0.4	0.285	0.418	0.312
echo	0.465	0.372	0.413	0.31	0.45	0.35
german	0.41	0.335	0.388	0.311	0.413	0.326
glass	0.513	0.483	0.568	0.499	0.554	0.479
magic	0.389	0.318	0.374	0.313	0.387	0.314
credit	0.361	0.207	0.322	0.181	0.364	0.203
spect	0.427	0.344	0.424	0.356	0.402	0.349
wdbc	0.296	0.114	0.245	0.066	0.289	0.101
ecoli	0.257	0.179	0.265	0.172	0.268	0.181
ionos	0.362	0.219	0.356	0.197	0.378	0.231
Ave.	0.366	0.27	0.353	0.252	0.371	0.265
Imp.	26.2%		28.6%		28.6%	

Table 27 Summarization of comparisons between NNE and NNEAUD on k-NN

k-NN									
	ENN	ENN+	**Imp.**	RENN	RENN+	**Imp.**	ANN	ANN+	**Imp.**
10%	0.227	0.215	5.3%	0.229	0.217	5.2%	0.229	0.217	5.2%
20%	0.247	0.220	10.9%	0.243	0.219	9.9%	0.245	0.218	11.0%
30%	0.295	0.239	19.0%	0.284	0.232	18.3%	0.292	0.238	18.5%
40%	0.366	0.270	26.2%	0.353	0.252	28.6%	0.371	0.265	28.6%
Ave.	0.284	0.236	15.4%	0.277	0.230	15.5%	0.284	0.235	15.8%

There is another useful result shown in the experiments. Figure 8 shows the relationship between noise ratio and improvement of ENN+ acting on ENN. We find that for all the three classifiers, the improvement increases in direct proportion to the noise ratio. Similarly, Figure 9 shows the improvement of RENN+ acting on RENN increases in direct proportion to the noise ratio. Figure 10 shows the improvement of ANN+ acting on ANN also increases in direct proportion to the noise ratio.

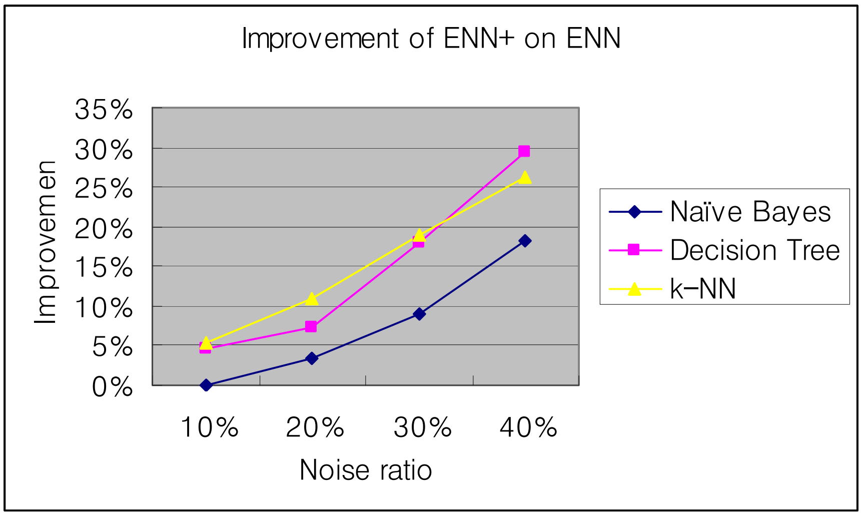


Fig. 8. Improvement of ENN+ on ENN

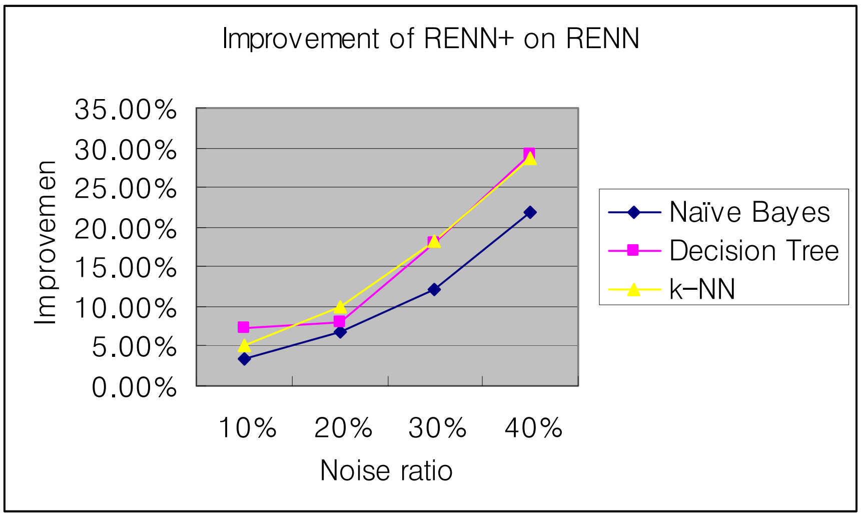


Fig. 9. Improvement of RENN+ on RENN

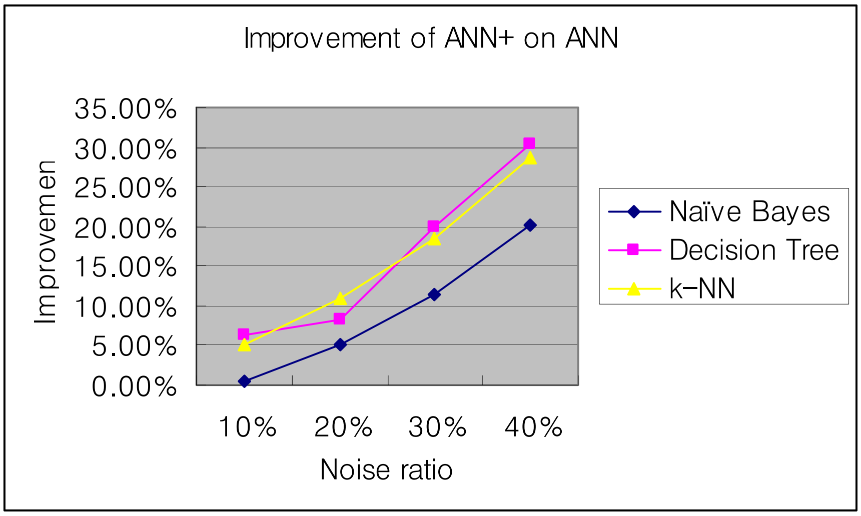


Fig. 10. Improvement of ANN+ on ANN

Experiments summarization: Experiment A and B indicate that by using nearest neighbor editing methods, the classification performance of k-NN, naïve Bayes, and decision tree can be improved. This suggests that although nearest neighbor editing is originally proposed for k-NN, it is not limited to k-NN. It also has the ability to effectively remove the noises for naïve Bayes and decision tree. In other words, nearest neighbor editing has the capability to serve as a general noise removing method. Moreover, by using unlabeled data, we develop NNEAUD methodology. Based on this novel methodology, the variants of edited nearest neighbor, repeated edited nearest neighbor, and All k-NN are devised. Experimental results show that all the three variants significantly outperform the original approaches. This means NNEAUD is more competent than NNE with respect to removing noises for k-NN, naïve Bayes, and decision tree. Moreover, the superiority of NNEAUD is more remarkable when noise ratio in the training data is greater.

3.5 Discussions

The experiments in this Chapter show that our method can improve the performance of instance-based noise filtering methods.

Suppose x is a noisy instance in the original training set T; x_a is the actual label of x; x_g is the given label of x in the training set; $T^{'}$ is the augmented data set ($T \subseteq T^{'}$); R is the k nearest neighbors of x in T (the distance is calculated based on the Euclidean distance of two feature vectors); $R^{'}$ is the k nearest neighbors of x in $T^{'}$. Let us consider a two-class classification problem wherein x_a=+, x_g=-.

ENN can identify x as noise if the label predicted by R is different with that of x_g. Therefore, the probability that ENN could identify x as noise is:

$P_T(R = + \mid x_a = +, x_g = -)$.

Using unlabeled data, the probability that ENN could identify x as noise is:

$P_{T^{'}}(R^{'} = + \mid x_a = +, x_g = -)$. If the elements in R are same with that in $R^{'}$, then these two probabilities are also same. It means unlabeled data has no any effect on noise detection on x. If the elements in R are different with that in $R^{'}$, then one or some neighbors which are closer to x are found in $T^{'}$. When the number of samples is very large, it is reasonable to assume that $x^{'}$ is sufficiently close to x that the label of x and $x^{'}$ are same. Therefore it is reasonable to assume that:

$P_{T'}(R' = + \mid x_a = +, x_g = -) \geq P_T(R = + \mid x_a = +, x_g = -)$ (inequation 1)

Above hypothesis is established when the new added training data are noise-free. When the points $R' \setminus \{R \cap R'\}$ are noisy, it is possible that:

$P_{T'}(R' = + \mid x_a = +, x_g = -) \leq P_T(R = + \mid x_a = +, x_g = -)$ (inequation 2)

Based on the theory in [44], the performance of the classifier is expected to be improved if enough noisy training data are added. In [44], m is the sample size; η is the noise rate in the training set; ε is the classification error; c is a constant under which that $\varepsilon = \sqrt{\frac{c}{m(1-2\eta)^2}}$. Let ε_1 and ε_2 denote the classification error based on the original training set and the augmented training set respectively; η_1 and η_2 denote the noise rates of the original training set and the augmented training set; m_1 and m_2 denote the original training set size and the augmented training set size. In the worst case that $\eta_2 > \eta_1$, if the sample number in $m_2 \setminus m_1$ are big enough, still $\varepsilon_2 > \varepsilon_1$. This proved that the probability of inequation 1 holds is greater than that of inequation 2 holds. The positive effect of our method on average can defeat the negative effect.

Chapter 4

General Noise Filtering

We have proved that our method is effective in instance-based noise filtering methods. We further prove that our method is general and not limited to any noise filtering methods in this chapter. There are many methods for general identifying and eliminating mislabeled training instances. Herein, we consider Brodley's majority filtering (MF) and consensus filtering (CF) due to their wide-spread and popular use in the literature.

4.1 Majority & Consensus filtering

The general idea of MF and CF is as follows: they employ ensemble classifier to detect mislabeled instances by constructing a set of base-level classifiers and then using their classifications to identify mislabeled instances. The general approach is to tag an instance as mislabeled if x of the m base-level classifiers cannot classify it correctly. MF tags an instance as mislabeled if more than half of the m base level classifiers classify it incorrectly. CF requires that all base-level classifiers must fail to classify an instance as the class given by its training label for it to be eliminated from the training data.

The reason to employ ensemble classifiers in MF and CF is that ensemble classifier has better performance than each base-level classifier on a dataset if two conditions hold: (1) the probability of a correct classification by each individual classifier is greater than 0.5, and (2) the errors in predictions of the base-level classifiers are independent.

As Shown in Table 28, majority filtering begins with n equal-sized disjoint subsets of the training set E (step 1) and the empty output set A of detected noisy examples (step 2). The main loop (steps 3-16) is repeated for each training subset E_i. In step 4, subset E_i is formed which includes all examples from E except those in E_i, which then is used as the input an arbitrary inductive learning algorithm that induces a hypothesis (a classifier) H_j (step 6). Those examples from E_i for which majority of the hypotheses do not give the correct classification are added to A as potentially noisy examples (step 14).

Table 28 Majority filtering

```
Algorithm: MajorityFiltering (MF)
Input: $E$ (training set)
Parameter: $n$ (number of subsets), $y$ (number of learning algorithms)
$A_1, A_2, ..., A_y$ ( $y$ kinds of learning algorithms)
Output: $A$ (detected noisy subset of $E$ )
(1)   form $n$ disjoint almost equally sized subsets of $E_i$, where $\cup_i E_i = E$
(2)   $A \leftarrow \varnothing$
(3)   for $i = 1, ..., n$ do
(4)     form $E_t \leftarrow E \setminus E_i$
(5)     for $j = 1, ...y$ do
(6)       induce $H_j$ based on examples in $E_t$ and $A_j$
(7)     end for
(8)     for every $e \in E_i$ do
(9)       $ErrorCounter \leftarrow 0$ // the num. of classifiers which misclassify e
(10)      for $j = 1, ..., y$ do
(11)        if $H_j$ incorrectly classifies $e$
(12)        then $ErrorCounter \leftarrow ErrorCounter + 1$
(13)      end for
(14)      if $ErrorCounter > y/2$ , then $A \leftarrow A \cup \{e\}$
(15)    end for
(16)  end for
```

Consensus filtering algorithm is shown in Table 29. Its difference with MF is at step 14. In CF, the example in E_i is regarded as noisy example only when all the hypotheses incorrectly classify it. Compared with MF, CF is more conservative due to the severer condition for noise identification, which results in fewer instances being eliminated from the training set. The drawback of CF is the added risk in retaining bad data.

Table 29 Consensus filtering

```
Algorithm: ConsensusFiltering (CF)
Input: $E$ (training set)
Parameter: $n$ (number of subsets), $y$ (number of learning algorithms)
$A_1, A_2, ..., A_y$ ( $y$ kinds of learning algorithms)
Output: $A$ (detected noisy subset of $E$ )
(1)   form $n$ disjoint almost equally sized subsets of $E_i$, where $\cup_i E_i = E$
(2)   $A \leftarrow \varnothing$
(3)   for $i = 1,...,n$ do
(4)     form $E_t \leftarrow E \setminus E_i$
(5)     for $j = 1,...y$ do
(6)       induce $H_j$ based on examples in $E_t$ and $A_j$
(7)     end for
(8)     for every $e \in E_i$ do
(9)       $ErrorCounter \leftarrow 0$ // the num. of classifiers which misclassify e
(10)      for $j = 1,...,y$ do
(11)        if $H_j$ incorrectly classifies $e$
(12)        then $ErrorCounter \leftarrow ErrorCounter + 1$
(13)      end for
(14)      if $ErrorCounter = y$ , then $A \leftarrow A \cup \{e\}$
(15)    end for
(16)  end for
```

Both majority filtering and consensus filtering employ multiple classifiers to detect the noisy instances through n-cross-validation. In cross i , subset i is extracted and checked. The combination of other subsets is used as training data to construct a set of classifiers based on the learning algorithms, which further classify the instances in subset i to detect the noises. The reliability of these classifiers therefore is crucial and the noise detection performance is expected to improve when the classification accuracies of these classifiers are increased. Our approach is to utilize the unlabeled data to increase the classification accuracies of these classifiers.

4.2 Majority & Consensus filtering aided by unlabeled data

As the learning algorithms in MF and CF are supervised, in order to utilize unlabeled data, the first phase therefore is to predict the labels for them. Then the second phase is to employ the unlabeled data with predicted labels to augment the classifiers' predictive ability, thereby improving the performance of noise detection.

The work in the first phase is based on our proposed ensemble-based data labeling function which can be referred to Chapter 2.

Table 30 Majority filtering aided by unlabeled data

```
Algorithm: Majority Filtering with the Aid of Unlabeled Data (MFAUD)
Input: $E$ (training set), $U$ (unlabeled set)
Parameter: $n$ (number of subsets), $y$ (number of learning algorithms)
$A_1, A_2, ..., A_y$ ( $y$ kinds of learning algorithms)
Output: $A$ (detected noisy subset of $E$ )
(1)   form $n$ disjoint almost equally sized subsets of $E_i$, where $\cup_i E_i = E$
(2)   $A \leftarrow \varnothing$
(3)   for $i = 1, ..., n$ do
(4)     form $E_t \leftarrow E \setminus E_i$
(5)     $T_U = En-co-training(E_t, U, k, u, A_1, A_2, ..., A_y)$
(6)     $E_t \leftarrow E_t \cup T_U$
(7)     for $j = 1, ...y$ do
(8)       induce $H_j$ based on examples in $E_t$ and $A_j$
(9)     end for
(10)    for every $e \in E_i$ do
(11)      $ErrorCounter \leftarrow 0$ // the num. of classifiers which misclassify e
(12)      for $j = 1, ..., y$ do
(13)        if $H_j$ incorrectly classifies $e$
(14)        then $ErrorCounter \leftarrow ErrorCounter + 1$
(15)      end for
(16)      if $ErrorCounter > y/2$ , then $A \leftarrow A \cup \{e\}$
(17)    end for
(18)  end for
```

Table 31 Consensus filtering aided by unlabeled data

```
Algorithm: Consensus Filtering with the Aid of Unlabeled Data (CFAUD)
Input: $E$ (training set), $U$ (unlabeled set)
Parameter: $n$ (number of subsets), $y$ (number of learning algorithms)
$A_1, A_2, ..., A_y$ ( $y$ kinds of learning algorithms)
Output: $A$ (detected noisy subset of $E$ )
(1)  form $n$ disjoint almost equally sized subsets of $E_i$ , where $\cup_i E_i = E$
(2)  $A \leftarrow \varnothing$
(3)  for $i = 1, ..., n$ do
(4)    form $E_t \leftarrow E \setminus E_i$
(5)    $T_U = En - co - training(E_t, U, k, u, A_1, A_2, ..., A_y)$
(6)    $E_t \leftarrow E_t \cup T_U$
(7)    for $j = 1, ...y$ do
(8)      induce $H_j$ based on examples in $E_t$ and $A_j$
(9)    end for
(10)   for every $e \in E_i$ do
(11)     $ErrorCounter \leftarrow 0$ // the num. of classifiers which misclassify e
(12)     for $j = 1, ..., y$ do
(13)       if $H_j$ incorrectly classifies $e$
(14)       then $ErrorCounter \leftarrow ErrorCounter + 1$
(15)     end for
(16)     if $ErrorCounter = y$ , then $A \leftarrow A \cup \{e\}$
(17)   end for
(18) end for
```

Let T_U denote the output of phase 1, which includes the selected unlabeled instances and their predicted labels. In this part, the variants of majority filtering and consensus filtering are devised that use T_U to aid the noise detection in training data T . As shown in the above Tables, T_U is utilized by our proposed MFAUD and CFAUD with only marginal modifications on MF and CF are required.

Always keep it mind that there is no free lunch and it is actually risky to use unlabeled data. We shouldn't take for granted that the using of unlabeled data can definitely improve the performance of noise detection in training set. The reason is as follows: although semi-supervised method including our ensemble-based data labeling function can predict the labels for some selected unlabeled data, the predicted labels are not noise-free (refer to Chapter 2). The unlabeled instances with correctly predicted labels tend to improve the noise detection performance, yet the noisy labels from prediction could potentially degrade the classifiers' predictive accuracies, which further leads to the performance degradation of noise detection in training set. With this caveat in mind, we now proceed to an empirical evaluation.

4.3 Empirical study

The main objective of the empirical study is to assess the benefit of unlabeled data for noise detection in training data. Chapter 4.3.1 explains the experimental setup. Afterwards Chapter 4.3.2 presents the experimental results.

4.3.1 Empirical setup

Existing MF, CF, and our proposed MFAUD, CFAUD are tested on the benchmark datasets from the Machine Learning Database Repository. Information of these data sets is tabulated in Table 4. Each data set is divided into training set and test set. Noise detection method works on the training set and outputs the filtered training set. Afterwards the test set is classified by the classifiers which are trained on the various filtered training sets. Classification error rate is the measure to evaluate the performance of each noise detection method on the classifier, where

$$\text{classification error rate}=\frac{\text{No. of incorrect classifications on testing instances}}{\text{No. of testing instances}}$$

When two noise detection methods are applied to the same data set with the same learning algorithm, lower classification error rate indicates that the noise detection performance is better. To obtain the classification error rate, each data set D is processed as follows:

(1) D is randomly partitioned into two parts: labeled set L and unlabeled set U .

(2) Ten trials derived from ten-fold cross-validation on L are used to evaluate the performance of each noise detection method. At each trial, 90% of L is firstly selected and it is denoted by T_{temp}. Most data sets here are experimental data sets where the ratio between noisy data to the whole data might be very small. However, the performance of noise removing method need to be evaluated on the noisy data sets. To this end, we artificially generate some noises in T_{temp} by selecting some instances at random and then incorrectly changing their labels. The number of selected instances, that is the number of generated noises, is based the defined noise ratio, which is the ratio between noisy data to the data in T_{temp}. Let T denote the data after adding noises in T_{temp}. T is used as training data and it will be processed by MF, CF, MFAUD, and CFAUD respectively. The remaining 10% of L is used as test set to be classified by the algorithms that are trained on each filtered set of T.

(3) The average classification error rates of each algorithm with different noise filtering methods are obtained by averaging ten trials' error rates.

(4) Considering that the partition of data set could influence this average classification error rate, we execute the partition five times and get five classification error rates (execute step 1-3 five times).

(5) Finally the reported error rates of each algorithm with different noise filtering methods are the further averaged value of these five values.

In this experiment, the four noise detection methods (MF, CF, MFAUD, and CFAUD) follow the same configuration which is as follows: n, that is the number of subsets, is set to 5; y, that is the number of learning algorithms, is set to 3; A1, A2, and A3, representing three learning algorithms, refer to k-nearest neighbor, naïve Bayes, and decision tree. The additional parameters in MFAUD and CFAUD are for ensemble-based data labeling function and configured as follows: k, that is the number of iterations, is set to 4; u, that is the number of initially selected unlabeled instances, is equal to the number of training data.

In addition to the parameters in the noise detection methods, there are two major parameters in above experiment flow which can influence the experiment. The first parameter determines data partitioning (step 1 of above experiment flow) and it is the ratio between labeled data to the whole data, referred to labeled ratio. In the experiment, labeled ratio is set to 0.5. The second and most important parameter determines the noise level in the training set T and it is the noise ratio (step 2 of above experiment flow). Considering that the data sets obtained from the real application might have different noise levels, we have performed several experiments varying the noise ratio to make the experiments comprehensively. The experiments include two parts:

Experiment A: The noise ratios include 10%, 20%, 30%, and 40%. The experiments here show the comparisons between MF and MFAUD under different noise ratios. The objective is to test whether unlabeled data are able to improve the performance of MF.

Experiment B: The noise ratios include 10%, 20%, 30%, and 40%. The experiments in B show the comparisons between CF and CFAUD under different noise ratios and the objective is to test whether unlabeled data are able to improve the performance of CF.

4.3.2 Experimental results

Experiment A: MF versus MFAUD. The experiments in this part include the comparisons between MF and MFAUD under four different noise ratios.

In Fig. 11 we show the classification error for each data set of the classifiers formed by each of the three algorithms using no filter (None), majority filtering (MF), and majority filtering aided by unlabeled data (MFAUD) when noise ratio is 10%. The second last row reports the average classification error across all the data sets of above classifiers. The last row reports the average improvement of MFAUD over MF with respect to reduction of classification error when they are used by each of the three algorithms. Fig. 11 shows for each of the three algorithms, on average its performance with noise filtering is better than that without noise filtering. Moreover MFAUD is better than MF for each of the algorithms. Concretely, for 1-NN, the improvement of MFAUD over MF is 4.5%; for naïve Bayes, the improvement is 1.3%; for decision tree, the improvement is 4.0%.

As well, Fig. 12, 13, and 14 show the classification results when noise ratio is 20%, 30%, and 40%. Among them, we analyze Fig. 14 for analyzing purpose.

Fig. 14 reports the comparison between MF and MFAUD when noise ratio is 40%. It shows similar result trend as Fig. 11 that both MF and MFAUD could improve the performance of all the three algorithms, and that MFAUD outperforms MF. Note that the improvement of MFAUD over MF is significantly increased when noise ratio increases from 10% to 40%. Concretely, when noise ratio is 10%, the improvement is less than 5%; the improvement however reaches to more than 20% when noise ratio is 40%.

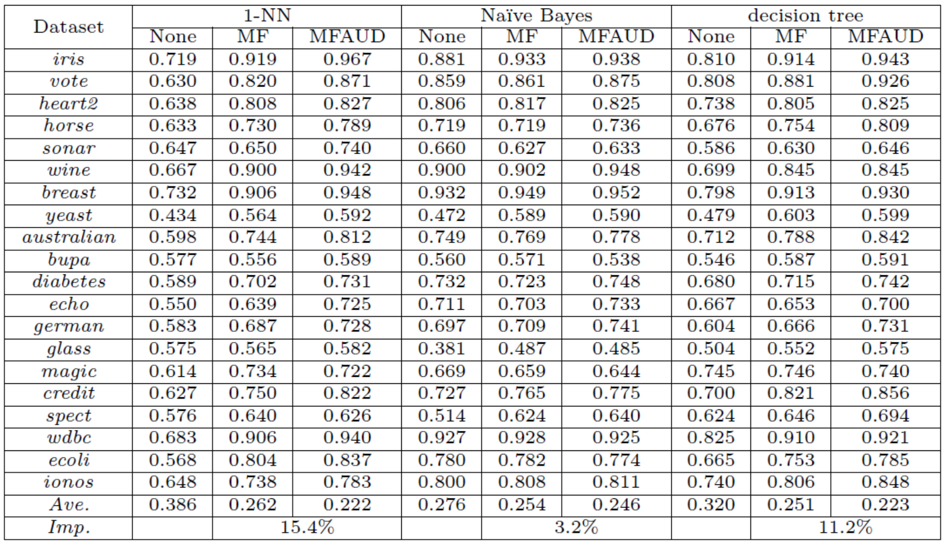
Fig. 15 summarizes the comparisons between MF and MFAUD under four different noise ratios. As shown in this table, MFAUD outperforms MF for each of the three algorithms under different noise ratios. In addition, for each of the algorithm, the improvement of MFAUD over MF is in direct proportion to the noise ratio. For example, for 1-NN, the improvement of MFAUD over MF is 4.5%when noise ratio is 10%, while the improvement is significantly increased to 25.6% when noise ratio is 40%. Averaged on the four noise ratios, for 1-NN, the improvement of MFAUD on MF is 13.4%; for naïve Bayes, the improvement of MFAUD on MF is 6.0%; for decision tree, the improvement of MFAUD on MF is 13.7%.

Dataset	1-NN			Naïve Bayes			decision tree		
	None	MF	MFAUD	None	MF	MFAUD	None	MF	MFAUD
iris	0.174	0.068	0.049	0.092	0.073	0.054	0.109	0.095	0.090
vote	0.165	0.133	0.151	0.161	0.125	0.132	0.109	0.102	0.087
heart2	0.296	0.221	0.203	0.212	0.209	0.214	0.215	0.218	0.171
horse	0.328	0.224	0.211	0.269	0.217	0.226	0.242	0.169	0.170
sonar	0.25	0.268	0.270	0.360	0.345	0.350	0.302	0.304	0.298
wine	0.143	0.055	0.051	0.020	0.025	0.023	0.183	0.097	0.103
breast	0.128	0.056	0.056	0.067	0.069	0.066	0.119	0.079	0.079
yeast	0.508	0.410	0.401	0.471	0.420	0.422	0.450	0.410	0.400
australian	0.273	0.174	0.162	0.207	0.198	0.196	0.176	0.159	0.160
bupa	0.435	0.377	0.376	0.466	0.450	0.443	0.380	0.374	0.382
diabetes	0.359	0.269	0.259	0.260	0.267	0.257	0.293	0.263	0.270
echo	0.419	0.322	0.275	0.286	0.286	0.269	0.381	0.322	0.306
german	0.336	0.29	0.263	0.247	0.243	0.239	0.316	0.282	0.252
glass	0.329	0.376	0.350	0.547	0.477	0.449	0.4	0.423	0.403
magic	0.308	0.253	0.259	0.342	0.343	0.352	0.236	0.234	0.239
credit	0.239	0.195	0.171	0.234	0.214	0.216	0.177	0.157	0.149
spect	0.329	0.328	0.315	0.408	0.286	0.280	0.271	0.254	0.230
wdbc	0.117	0.051	0.054	0.077	0.081	0.080	0.107	0.078	0.075
ecoli	0.278	0.163	0.158	0.183	0.165	0.158	0.219	0.193	0.179
ionos	0.194	0.176	0.176	0.171	0.188	0.195	0.161	0.144	0.140
Ave.	0.280	0.221	0.211	0.254	0.234	0.231	0.242	0.218	0.209
Imp.		4.5%			1.3%			4.0%	

Fig. 11. MF versus MFAUD when noise ratio is 10%

Dataset	1-NN			Naïve Bayes			decision tree		
	None	MF	MFAUD	None	MF	MFAUD	None	MF	MFAUD
iris	0.764	0.902	0.927	0.911	0.913	0.942	0.880	0.940	0.923
vote	0.763	0.878	0.883	0.848	0.865	0.865	0.878	0.906	0.922
heart2	0.656	0.787	0.817	0.794	0.804	0.805	0.731	0.796	0.811
horse	0.634	0.750	0.787	0.768	0.762	0.780	0.729	0.783	0.834
sonar	0.678	0.671	0.711	0.663	0.660	0.671	0.592	0.679	0.635
wine	0.767	0.915	0.917	0.919	0.950	0.957	0.737	0.872	0.878
breast	0.776	0.947	0.954	0.922	0.950	0.944	0.876	0.917	0.930
yeast	0.461	0.577	0.592	0.532	0.575	0.576	0.511	0.593	0.598
australian	0.686	0.803	0.822	0.763	0.791	0.793	0.761	0.854	0.852
bupa	0.559	0.591	0.614	0.540	0.556	0.567	0.612	0.579	0.620
diabetes	0.595	0.713	0.738	0.729	0.727	0.730	0.721	0.725	0.735
echo	0.531	0.669	0.714	0.692	0.703	0.706	0.625	0.672	0.703
german	0.619	0.697	0.727	0.744	0.732	0.760	0.665	0.723	0.752
glass	0.591	0.573	0.619	0.442	0.498	0.499	0.582	0.555	0.577
magic	0.639	0.730	0.723	0.655	0.657	0.632	0.735	0.751	0.743
credit	0.670	0.771	0.810	0.745	0.776	0.782	0.756	0.810	0.827
spect	0.607	0.678	0.657	0.517	0.656	0.655	0.679	0.750	0.732
wdbc	0.760	0.934	0.939	0.920	0.929	0.931	0.877	0.915	0.926
ecoli	0.630	0.801	0.810	0.786	0.785	0.814	0.735	0.772	0.807
ionos	0.753	0.820	0.838	0.820	0.820	0.824	0.782	0.854	0.882
Ave.	0.340	0.240	0.220	0.264	0.245	0.238	0.277	0.228	0.216
Imp.		8.1%			2.5%			5.4%	

Fig. 12. MF versus MFAUD when noise ratio is 20%



Dataset	1-NN			Naïve Bayes			decision tree		
	None	MF	MFAUD	None	MF	MFAUD	None	MF	MFAUD
iris	0.719	0.919	0.967	0.881	0.933	0.938	0.810	0.914	0.943
vote	0.630	0.820	0.871	0.859	0.861	0.875	0.808	0.881	0.926
heart2	0.638	0.808	0.827	0.806	0.817	0.825	0.738	0.805	0.825
horse	0.633	0.730	0.789	0.719	0.719	0.736	0.676	0.754	0.809
sonar	0.647	0.650	0.740	0.660	0.627	0.633	0.586	0.630	0.646
wine	0.667	0.900	0.942	0.900	0.902	0.948	0.699	0.845	0.845
breast	0.732	0.906	0.948	0.932	0.949	0.952	0.798	0.913	0.930
yeast	0.434	0.564	0.592	0.472	0.589	0.590	0.479	0.603	0.599
australian	0.598	0.744	0.812	0.749	0.769	0.778	0.712	0.788	0.842
bupa	0.577	0.556	0.589	0.560	0.571	0.538	0.546	0.587	0.591
diabetes	0.589	0.702	0.731	0.732	0.723	0.748	0.680	0.715	0.742
echo	0.550	0.639	0.725	0.711	0.703	0.733	0.667	0.653	0.700
german	0.583	0.687	0.728	0.697	0.709	0.741	0.604	0.666	0.731
glass	0.575	0.565	0.582	0.381	0.487	0.485	0.504	0.552	0.575
magic	0.614	0.734	0.722	0.669	0.659	0.644	0.745	0.746	0.740
credit	0.627	0.750	0.822	0.727	0.765	0.775	0.700	0.821	0.856
spect	0.576	0.640	0.626	0.514	0.624	0.640	0.624	0.646	0.694
wdbc	0.683	0.906	0.940	0.927	0.928	0.925	0.825	0.910	0.921
ecoli	0.568	0.804	0.837	0.780	0.782	0.774	0.665	0.753	0.785
ionos	0.648	0.738	0.783	0.800	0.808	0.811	0.740	0.806	0.848
Ave.	0.386	0.262	0.222	0.276	0.254	0.246	0.320	0.251	0.223
Imp.		15.4%			3.2%			11.2%	

Fig. 13. MF versus MFAUD when noise ratio is 30%

Dataset	1-NN			Naïve Bayes			decision tree		
	None	MF	MFAUD	None	MF	MFAUD	None	MF	MFAUD
iris	0.397	0.163	0.054	0.152	0.128	0.057	0.344	0.157	0.066
vote	0.419	0.270	0.131	0.205	0.220	0.134	0.336	0.275	0.091
heart2	0.434	0.333	0.201	0.337	0.302	0.237	0.369	0.324	0.201
horse	0.436	0.356	0.262	0.351	0.303	0.261	0.428	0.335	0.217
sonar	0.454	0.404	0.329	0.436	0.446	0.356	0.432	0.423	0.397
wine	0.381	0.131	0.089	0.114	0.139	0.048	0.381	0.164	0.120
breast	0.359	0.15	0.052	0.098	0.075	0.046	0.272	0.170	0.087
yeast	0.607	0.499	0.428	0.543	0.440	0.425	0.546	0.457	0.422
australian	0.415	0.326	0.201	0.345	0.277	0.226	0.397	0.313	0.198
bupa	0.478	0.474	0.441	0.478	0.488	0.460	0.484	0.464	0.425
diabetes	0.435	0.326	0.256	0.310	0.298	0.239	0.326	0.323	0.234
echo	0.447	0.372	0.292	0.344	0.35	0.258	0.403	0.361	0.283
german	0.441	0.384	0.289	0.334	0.335	0.237	0.422	0.353	0.285
glass	0.547	0.444	0.392	0.601	0.52	0.496	0.569	0.497	0.432
magic	0.435	0.360	0.313	0.359	0.366	0.378	0.382	0.342	0.299
credit	0.402	0.307	0.198	0.349	0.286	0.215	0.390	0.272	0.175
spect	0.468	0.438	0.426	0.462	0.430	0.409	0.439	0.402	0.402
wdbc	0.386	0.170	0.071	0.117	0.094	0.077	0.284	0.173	0.079
ecoli	0.483	0.205	0.158	0.195	0.215	0.177	0.385	0.244	0.192
ionos	0.409	0.373	0.237	0.241	0.259	0.213	0.335	0.324	0.189
Ave.	0.442	0.324	0.241	0.319	0.299	0.247	0.396	0.319	0.240
Imp.		25.6%			17.1%			24.8%	

Fig. 14. MF versus MFAUD when noise ratio is 40%

		noise ratio				
		10%	20%	30%	40%	**Ave.**
1-NN	None	0.280	0.343	0.386	0.442	0.363
	CF	0.230	0.263	0.288	0.362	0.286
	CFAUD	0.217	0.230	0.233	0.273	0.238
	Imp.	5.6%	12.4%	18.8%	24.6%	15.4%
NB	None	0.254	0.264	0.276	0.319	0.278
	CF	0.241	0.252	0.269	0.304	0.267
	CFAUD	0.232	0.244	0.250	0.256	0.246
	Imp.	3.5%	3.3%	7.3%	16.0%	7.5%
DT	None	0.242	0.277	0.320	0.396	0.309
	CF	0.219	0.241	0.278	0.349	0.272
	CFAUD	0.211	0.218	0.230	0.255	0.229
	Imp.	3.7%	9.3%	17.2%	27.1%	14.3%

Fig. 15. Summarization of MF versus MFAUD

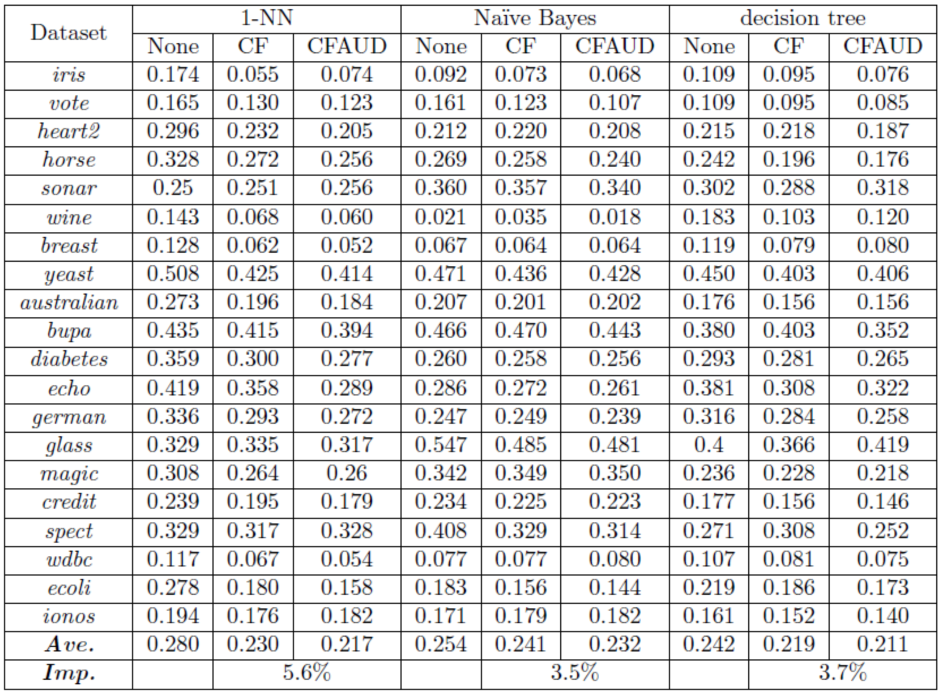
Experiment B: CF versus CFAUD. In Fig. 16 we show the classification error for each data set of the classifiers formed by each of the three algorithms testing using no filter (None), consensus filtering (CF), and consensus filtering aided by unlabeled data (CFAUD) when noise ratio is 10%. As shown in this table, both CF and CFAUD can provide improvement on the three algorithms. When CF and CFAUD are compared, we find that CFAUD defeats CF. However, the improvement of CFAUD over CF is not very remarkable and limited in 6%.

Fig. 17 to 19 show the experimental results when noise ratio is 20%, 30%, and 40% respectively. Here we choose Fig. 19 for analyzing.

Fig. 19 reports the classification comparison between CF and CFAUD when noise ratio is 40%. Under this noise ratio, all of the three algorithms achieve improved classification performance when using CF or CFAUD. Moreover, CFAUD significantly outperforms CF and its improvement over CF is much greater than that when noise ratio is 10%.

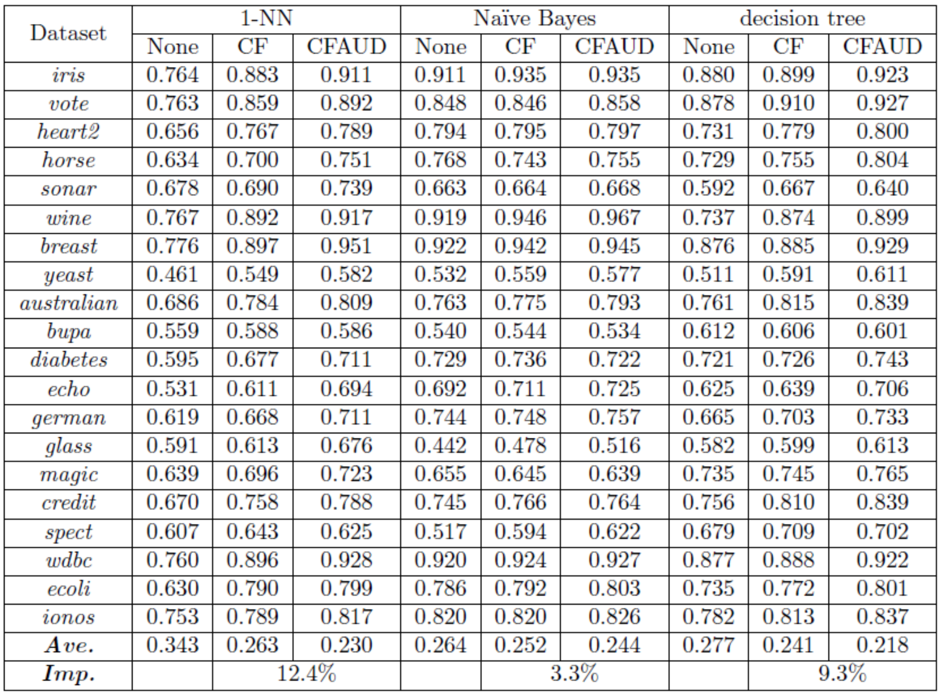
Fig. 20 shows the summarization of CFAUD versus CF under for different noise ratios from 10% to 40%. This table indicates that CFAUD can provide consistent improvement on each of the three algorithms under each of the four noise ratios. In addition, for each of the three algorithms, the improvement of CFAUD over CF increases as the noise ratio increases. When comparing the three algorithms without noise filtering, naïve Bayes gives the best classification. When comparing the three algorithms with noise filtering, on average decision tree with CFAUD gives the best classification performance.

Experiments summarization: The results in Experiment A and B show that MFAUD and CFAUD could consistently improve MF and CF under different noise ratios. In other words, the mislabeled instances detection performance could achieve consistent improvement with the aid of unlabeled data.



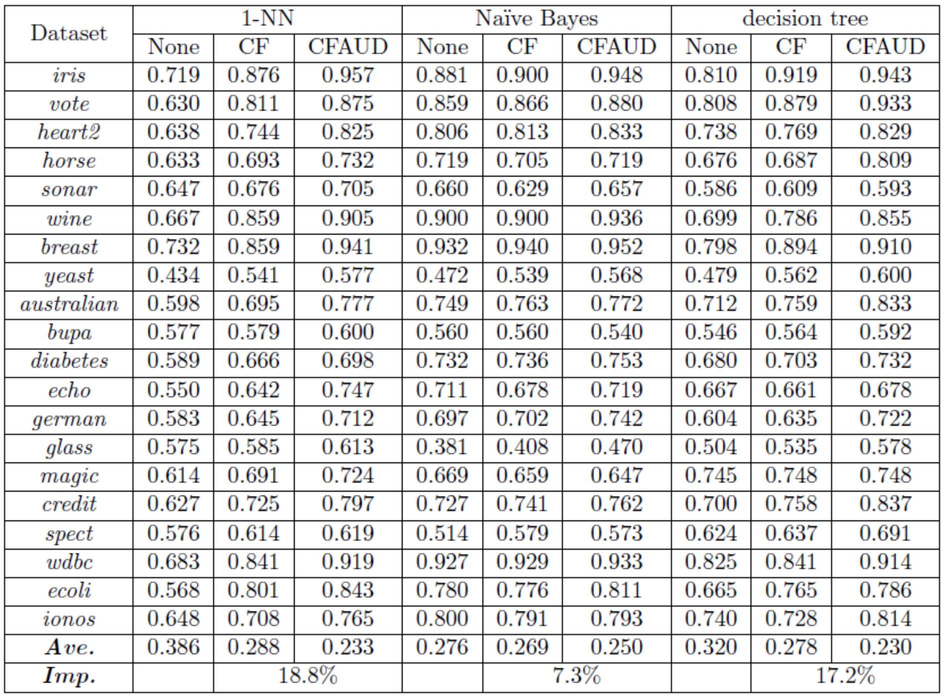
Dataset	1-NN			Naïve Bayes			decision tree		
	None	CF	CFAUD	None	CF	CFAUD	None	CF	CFAUD
iris	0.174	0.055	0.074	0.092	0.073	0.068	0.109	0.095	0.076
vote	0.165	0.130	0.123	0.161	0.123	0.107	0.109	0.095	0.085
heart2	0.296	0.232	0.205	0.212	0.220	0.208	0.215	0.218	0.187
horse	0.328	0.272	0.256	0.269	0.258	0.240	0.242	0.196	0.176
sonar	0.25	0.251	0.256	0.360	0.357	0.340	0.302	0.288	0.318
wine	0.143	0.068	0.060	0.021	0.035	0.018	0.183	0.103	0.120
breast	0.128	0.062	0.052	0.067	0.064	0.064	0.119	0.079	0.080
yeast	0.508	0.425	0.414	0.471	0.436	0.428	0.450	0.403	0.406
australian	0.273	0.196	0.184	0.207	0.201	0.202	0.176	0.156	0.156
bupa	0.435	0.415	0.394	0.466	0.470	0.443	0.380	0.403	0.352
diabetes	0.359	0.300	0.277	0.260	0.258	0.256	0.293	0.281	0.265
echo	0.419	0.358	0.289	0.286	0.272	0.261	0.381	0.308	0.322
german	0.336	0.293	0.272	0.247	0.249	0.239	0.316	0.284	0.258
glass	0.329	0.335	0.317	0.547	0.485	0.481	0.4	0.366	0.419
magic	0.308	0.264	0.26	0.342	0.349	0.350	0.236	0.228	0.218
credit	0.239	0.195	0.179	0.234	0.225	0.223	0.177	0.156	0.146
spect	0.329	0.317	0.328	0.408	0.329	0.314	0.271	0.308	0.252
wdbc	0.117	0.067	0.054	0.077	0.077	0.080	0.107	0.081	0.075
ecoli	0.278	0.180	0.158	0.183	0.156	0.144	0.219	0.186	0.173
ionos	0.194	0.176	0.182	0.171	0.179	0.182	0.161	0.152	0.140
Ave.	0.280	0.230	0.217	0.254	0.241	0.232	0.242	0.219	0.211
Imp.		5.6%			3.5%			3.7%	

Fig. 16. CF versus CFAUD when noise ratio is 10%



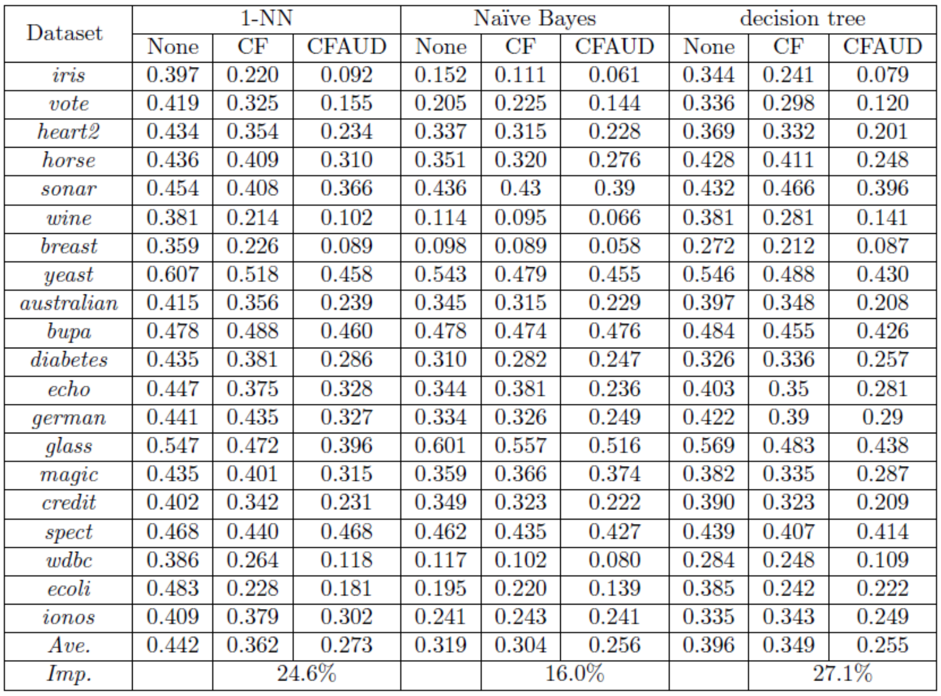
Dataset	1-NN			Naïve Bayes			decision tree		
	None	CF	CFAUD	None	CF	CFAUD	None	CF	CFAUD
iris	0.764	0.883	0.911	0.911	0.935	0.935	0.880	0.899	0.923
vote	0.763	0.859	0.892	0.848	0.846	0.858	0.878	0.910	0.927
heart2	0.656	0.767	0.789	0.794	0.795	0.797	0.731	0.779	0.800
horse	0.634	0.700	0.751	0.768	0.743	0.755	0.729	0.755	0.804
sonar	0.678	0.690	0.739	0.663	0.664	0.668	0.592	0.667	0.640
wine	0.767	0.892	0.917	0.919	0.946	0.967	0.737	0.874	0.899
breast	0.776	0.897	0.951	0.922	0.942	0.945	0.876	0.885	0.929
yeast	0.461	0.549	0.582	0.532	0.559	0.577	0.511	0.591	0.611
australian	0.686	0.784	0.809	0.763	0.775	0.793	0.761	0.815	0.839
bupa	0.559	0.588	0.586	0.540	0.544	0.534	0.612	0.606	0.601
diabetes	0.595	0.677	0.711	0.729	0.736	0.722	0.721	0.726	0.743
echo	0.531	0.611	0.694	0.692	0.711	0.725	0.625	0.639	0.706
german	0.619	0.668	0.711	0.744	0.748	0.757	0.665	0.703	0.733
glass	0.591	0.613	0.676	0.442	0.478	0.516	0.582	0.599	0.613
magic	0.639	0.696	0.723	0.655	0.645	0.639	0.735	0.745	0.765
credit	0.670	0.758	0.788	0.745	0.766	0.764	0.756	0.810	0.839
spect	0.607	0.643	0.625	0.517	0.594	0.622	0.679	0.709	0.702
wdbc	0.760	0.896	0.928	0.920	0.924	0.927	0.877	0.888	0.922
ecoli	0.630	0.790	0.799	0.786	0.792	0.803	0.735	0.772	0.801
ionos	0.753	0.789	0.817	0.820	0.820	0.826	0.782	0.813	0.837
Ave.	0.343	0.263	0.230	0.264	0.252	0.244	0.277	0.241	0.218
Imp.		12.4%			3.3%			9.3%	

Fig. 17. CF versus CFAUD when noise ratio is 20%



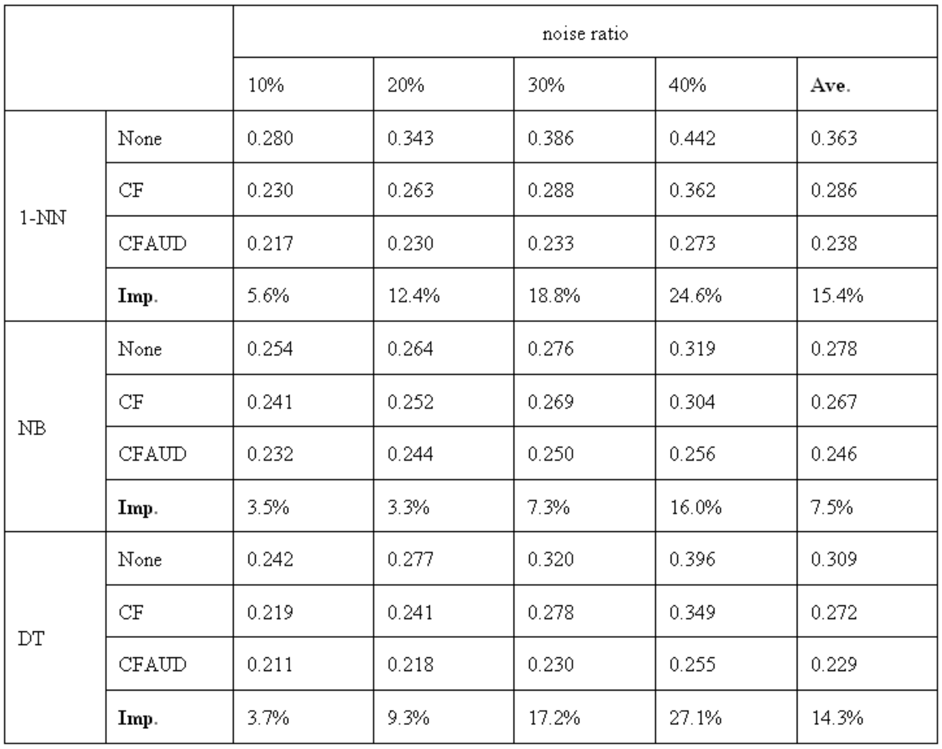
Dataset	1-NN			Naïve Bayes			decision tree		
	None	CF	CFAUD	None	CF	CFAUD	None	CF	CFAUD
iris	0.719	0.876	0.957	0.881	0.900	0.948	0.810	0.919	0.943
vote	0.630	0.811	0.875	0.859	0.866	0.880	0.808	0.879	0.933
heart2	0.638	0.744	0.825	0.806	0.813	0.833	0.738	0.769	0.829
horse	0.633	0.693	0.732	0.719	0.705	0.719	0.676	0.687	0.809
sonar	0.647	0.676	0.705	0.660	0.629	0.657	0.586	0.609	0.593
wine	0.667	0.859	0.905	0.900	0.900	0.936	0.699	0.786	0.855
breast	0.732	0.859	0.941	0.932	0.940	0.952	0.798	0.894	0.910
yeast	0.434	0.541	0.577	0.472	0.539	0.568	0.479	0.562	0.600
australian	0.598	0.695	0.777	0.749	0.763	0.772	0.712	0.759	0.833
bupa	0.577	0.579	0.600	0.560	0.560	0.540	0.546	0.564	0.592
diabetes	0.589	0.666	0.698	0.732	0.736	0.753	0.680	0.703	0.732
echo	0.550	0.642	0.747	0.711	0.678	0.719	0.667	0.661	0.678
german	0.583	0.645	0.712	0.697	0.702	0.742	0.604	0.635	0.722
glass	0.575	0.585	0.613	0.381	0.408	0.470	0.504	0.535	0.578
magic	0.614	0.691	0.724	0.669	0.659	0.647	0.745	0.748	0.748
credit	0.627	0.725	0.797	0.727	0.741	0.762	0.700	0.758	0.837
spect	0.576	0.614	0.619	0.514	0.579	0.573	0.624	0.637	0.691
wdbc	0.683	0.841	0.919	0.927	0.929	0.933	0.825	0.841	0.914
ecoli	0.568	0.801	0.843	0.780	0.776	0.811	0.665	0.765	0.786
ionos	0.648	0.708	0.765	0.800	0.791	0.793	0.740	0.728	0.814
Ave.	0.386	0.288	0.233	0.276	0.269	0.250	0.320	0.278	0.230
Imp.		18.8%			7.3%			17.2%	

Fig. 18. CF versus CFAUD when noise ratio is 30%



Dataset	1-NN			Naïve Bayes			decision tree		
	None	CF	CFAUD	None	CF	CFAUD	None	CF	CFAUD
iris	0.397	0.220	0.092	0.152	0.111	0.061	0.344	0.241	0.079
vote	0.419	0.325	0.155	0.205	0.225	0.144	0.336	0.298	0.120
heart2	0.434	0.354	0.234	0.337	0.315	0.228	0.369	0.332	0.201
horse	0.436	0.409	0.310	0.351	0.320	0.276	0.428	0.411	0.248
sonar	0.454	0.408	0.366	0.436	0.43	0.39	0.432	0.466	0.396
wine	0.381	0.214	0.102	0.114	0.095	0.066	0.381	0.281	0.141
breast	0.359	0.226	0.089	0.098	0.089	0.058	0.272	0.212	0.087
yeast	0.607	0.518	0.458	0.543	0.479	0.455	0.546	0.488	0.430
australian	0.415	0.356	0.239	0.345	0.315	0.229	0.397	0.348	0.208
bupa	0.478	0.488	0.460	0.478	0.474	0.476	0.484	0.455	0.426
diabetes	0.435	0.381	0.286	0.310	0.282	0.247	0.326	0.336	0.257
echo	0.447	0.375	0.328	0.344	0.381	0.236	0.403	0.35	0.281
german	0.441	0.435	0.327	0.334	0.326	0.249	0.422	0.39	0.29
glass	0.547	0.472	0.396	0.601	0.557	0.516	0.569	0.483	0.438
magic	0.435	0.401	0.315	0.359	0.366	0.374	0.382	0.335	0.287
credit	0.402	0.342	0.231	0.349	0.323	0.222	0.390	0.323	0.209
spect	0.468	0.440	0.468	0.462	0.435	0.427	0.439	0.407	0.414
wdbc	0.386	0.264	0.118	0.117	0.102	0.080	0.284	0.248	0.109
ecoli	0.483	0.228	0.181	0.195	0.220	0.139	0.385	0.242	0.222
ionos	0.409	0.379	0.302	0.241	0.243	0.241	0.335	0.343	0.249
Ave.	0.442	0.362	0.273	0.319	0.304	0.256	0.396	0.349	0.255
Imp.		24.6%			16.0%			27.1%	

Fig. 19. CF versus CFAUD when noise ratio is 40%



		noise ratio				
		10%	20%	30%	40%	**Ave.**
1-NN	None	0.280	0.343	0.386	0.442	0.363
	CF	0.230	0.263	0.288	0.362	0.286
	CFAUD	0.217	0.230	0.233	0.273	0.238
	Imp.	5.6%	12.4%	18.8%	24.6%	15.4%
NB	None	0.254	0.264	0.276	0.319	0.278
	CF	0.241	0.252	0.269	0.304	0.267
	CFAUD	0.232	0.244	0.250	0.256	0.246
	Imp.	3.5%	3.3%	7.3%	16.0%	7.5%
DT	None	0.242	0.277	0.320	0.396	0.309
	CF	0.219	0.241	0.278	0.349	0.272
	CFAUD	0.211	0.218	0.230	0.255	0.229
	Imp.	3.7%	9.3%	17.2%	27.1%	14.3%

Fig. 20. Summarization of CF versus CFAUD

4.4 Discussions

The experiments in this Chapter show that our method can improve the performance of majority and consensus filtering methods.

Recall the discussion in Chapter 2, using unlabeled data is not free because their predicted labels are prone to noises. On one hand, the unlabeled data with correctly predicted labels tend to improve the classification accuracy of ensemble classifiers, thereby updating the performance of mislabeled instances detection. On the other hand, the unlabeled data with incorrectly predicted labels tend to degrade the performance of mislabeled instances detection. The experimental results show that the positive effect of using unlabeled data defeats its negative effect. One possible explanation is that: both MF and CF employ n-cross-validation method for noise detection. For each cross, all the data except this cross are used to train ensemble classifiers based on different learning algorithms and then these constructed ensemble classifiers classify the data in this cross to detect noises. This process determines that the performance of noise detection is expected to improve when the predictive accuracy of ensemble classifiers increase. Existing works on semi-supervised learning have shown that the predictive accuracy of each classifier (also ensemble classifiers) can be improved by using unlabeled data. Based on PAC theory [44], although there might be some noisy predicted labels for unlabeled data, the negative effect of them could be compensated if the amount of newly labeled examples is sufficient.

Above discussion also explains the reason why our method works well for instance-based noise filtering methods in Chapter 3. In essence, the instance-based noise filtering methods also employ the “n-cross-validation” method for noise detection, in which, the number n is the number of training instances in their methods.

Chapter 5
Conclusion and future work

In this thesis, we present a novel noise filtering methodology which incorporates the knowledge of unlabeled data. This is in contrast with existing noise filtering methods that are usually supervised and can only utilize the knowledge of training instances. To utilize unlabeled instances, we propose an ensemble-based semi-supervised classification method to predict the labels for some selected unlabeled instances.

To test the performance of our method, we firstly use it on three instance-based noise filtering methods: edited nearest neighbor, repeated edited nearest neighbor, and All k-NN. A set of experiments prove the superiority of our method that all these three methods can achieve improved performance when using our method.

Afterwards, to show the generality of our method, we also use it on the general noise filtering methods: majority filtering and consensus filtering. Three popular algorithms are adopted, including k-NN, naïve Bayes, and decision tree. The experimental results show that by using our method, majority/consensus filtering can be improved for all of these three methods.

This study indicates that the performance of traditional supervised noise filtering methods could be boosted with the aid of unlabeled instances.

Noise filtering is one of the techniques for noise handling. One of our future works is to consider using unlabeled data in other noise handling techniques, including, for example noise tolerance and noise correction.

In addition, although we claim that our method can be used to improve the performance of noise filtering; currently five noise filtering methods are tested.

We will test the performance of our method on other noise filtering techniques in the future.

Noise handling including noise filtering aims to refine the training data to improve the learning performance. Another future work is to use unlabeled data in other techniques with the aim of refining training data. The techniques we are interested include feature selection and valuable training data selection.

Finally, the usage of unlabeled data in current work is by predicting the labels for them. In the future, we will try to extract some other type of information from unlabeled data, such as density information, partition information, and so on.

Bibliography

[1] J. Mingers, "An Empirical Comparison of Pruning Methods for Decision Tree Induction," *Machine Learning*, vol. 4, Nov. 1989, pp. 227-243.

[2] J.R. Quinlan, *C4.5: programs for machine learning*, Morgan Kaufmann Publishers Inc., 1993.

[3] D. Gamberger, N. Lavrac, and S. Dzeroski, "NOISE - DETECTION - AND - ELIMINATION - IN - DATA - PREPROCESSING: EXPERIMENTS - IN - MEDICAL - DOMAINS - PB - Taylor & Francis," *Applied Artificial Intelligence*, vol. 14, 2000, p. 205.

[4] J.R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, Mar. 1986, pp. 81-106.

[5] C.E. Brodley and M.A. Friedl, "Identifying and eliminating mislabeled training instances," *IN AAAI/IAAI*, vol. 1, 1996, pp. 799--805.

[6] C.E. Brodley, P. Uiversity, M.A. Friedl, B. Uiversity, and B.P. Edu, "Identifying Mislabeled Training Data," *JOURNAL OF ARTIFICIAL INTELLIGENCE RESEARCH*, vol. 11, 1999, pp. 131--167.

[7] I. Guyon, N. Matic, and V. Vapnik, "Discovering informative patterns and data cleaning," *Advances in knowledge discovery and data mining*, American Association for Artificial Intelligence, 1996, pp. 181-203.

[8] D. Gamberger, N. Lavrac, and C. Groselj, "Experiments with Noise Filtering in a Medical Domain," *Proceedings of the Sixteenth International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., 1999, pp. 143-151.

[9] D.L. Wilson, "Asymptotic Properties of Nearest Neighbor Rules Using Edited Data," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 2, 1972, pp. 408-421.

[10] D.W. Aha, D. Kibler, and M.K. Albert, "Instance-Based Learning Algorithms," *Mach. Learn.*, vol. 6, 1991, pp. 37-66.

[11] K.P. Bennett, A. Demiriz, and R. Maclin, "Exploiting unlabeled data in ensemble methods," *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, Edmonton, Alberta, Canada: ACM, 2002, pp. 289-

296.

[12] X. Zhu and Y. Yang, “A lazy bagging approach to classification,” *Pattern Recogn.*, vol. 41, 2008, pp. 2980-2992.

[13] Y. Song, F. Nie, C. Zhang, and S. Xiang, “A unified framework for semi-supervised dimensionality reduction,” *Pattern Recognition*, vol. 41, Sep. 2008, pp. 2789-2799.

[14] J. Cheng and K. Wang, “Active learning for image retrieval with Co-SVM,” *Pattern Recognition*, vol. 40, Jan. 2007, pp. 330-334.

[15] C. Constantinopoulos and A. Likas, “Semi-supervised and active learning with the probabilistic RBF classifier,” *Neurocomput.*, vol. 71, 2008, pp. 2489-2498.

[16] J. Handl and J. Knowles, “Semi-supervised feature selection via multiobjective optimization,” *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, 2006, pp. 3319-3326.

[17] J. Zhao, K. Lu, and X. He, “Locality sensitive semi-supervised feature selection,” *Neurocomputing*, vol. 71, Jun. 2008, pp. 1842-1849.

[18] A.P. Dempster, N.M. Laird, and D.B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, vol. 39, 1977, pp. 1--38.

[19] D. Yarowsky, “Unsupervised word sense disambiguation rivaling supervised methods,” *IN PROCEEDINGS OF THE 33RD ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, 1995, pp. 189--196.

[20] A. Blum and T. Mitchell, “Combining labeled and unlabeled data with co-training,” Proceedings of the eleventh annual conference on Computational learning theory, Madison, Wisconsin, United States: ACM, 1998, pp. 92-100.

[21] A. Blum and S. Chawla, “Learning from Labeled and Unlabeled Data using Graph Mincuts,” Proceedings of the Eighteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., 2001, pp. 19-26.

[22] X. Zhu, Z. Ghahramani, and J. Lafferty, “Semi-supervised learning using Gaussian fields and harmonic functions,” IN ICML, 2003, pp. 912--919.

[23] E. Riloff, J. Wiebe, and T. Wilson, “Learning subjective nouns using extraction pattern bootstrapping,” *Proceedings of the seventh conference on*

Natural language learning at HLT-NAACL 2003 - Volume 4, Edmonton, Canada: Association for Computational Linguistics, 2003, pp. 25-32.

[24] B. Maeireizo, D. Litman, and R. Hwa, “Co-training for predicting emotions with spoken dialogue data,” *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, Barcelona, Spain: Association for Computational Linguistics, 2004, p. 28.

[25] C. Rosenberg, M. Hebert, and H. Schneiderman, “Semi-Supervised Self-Training of Object Detection Models,” *Application of Computer Vision, 2005. WACV/MOTIONS '05 Volume 1. Seventh IEEE Workshops on*, 2005, pp. 29-36.

[26] K. Nigam and R. Ghani, “Analyzing the effectiveness and applicability of co-training,” *Proceedings of the ninth international conference on Information and knowledge management*, McLean, Virginia, United States: ACM, 2000, pp. 86-93.

[27] Balcan, M.F., Blum, A. An augmented pac model for semi-supervised learning. Semi-supervised learning, MIT Press. (2006)

[28] Zhou, Z.H., Zhan, D.C., and Yang, Q. Semi-supervised learning with very few labeled training examples. 22nd AAAI conference on Artificial Intelligence. (2007)

[29] Dasgupta, S., Littman, M.L., and McAllester, D. PAC generalization bounds for co-training. (2001)

[30] UCI KDD Archieve, http://kdd.ics.uci.edu

[31] T.M. Cover and P.E. Hart, Nearest neighbor pattern recognition, IEEE Trans. on Information Theory 13 no. 1 (1967) 21-27.

[32] R.O. Duda, P.E. Hart and D.G. Stork, Pattern Classification, 2nd ed. (John Wiley & Sons, Inc., 2001)

[33] B.V. Dasarathy, Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques, IEEE Computer Society Press (Los Alamitos, CA, 1991)

[34] “An Experiment with the Edited Nearest-Neighbor Rule,” *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 6, 1976, pp. 448-452.

[35] J. Koplowitz and T.A. Brown, “On the relation of performance to editing in

nearest neighbor rules," Pattern Recognition, vol. 13, 1981, pp. 251-255.

[36] R. Barandela and E. Gasca, "Decontamination of Training Samples for Supervised Pattern Recognition Methods," Advances in Pattern Recognition, 2000, pp. 621-630.

[37] P.A. Devijver, J. Kittler, Pattern Recognition: a Statistical Approach, (Prentice Hall, Englewood Cliffs, N.J. 1982)

[38] J.S. Sánchez, F. Pla, and F.J. Ferri, "Prototype selection for the nearest neighbour rule through proximity graphs," *Pattern Recogn. Lett.*, vol. 18, 1997, pp. 507-513.

[39] L.I. Kuncheva, "Editing for the k-nearest neighbors rule by a genetic algorithm," *Pattern Recognition Letters*, vol. 16, Aug. 1995, pp. 809-814.

[40] K. Hattori and M. Takahashi, "A new edited k-nearest neighbor rule in the pattern classification problem," *Pattern Recognition*, vol. 33, Mar. 2000, pp. 521-528.

[41] C. Keung and W. Lam, "Prototype Generation Based on Instance Filtering and Averaging," *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications*, Springer-Verlag, 2000, pp. 142-152.

[42] "Another Look at the Edited Nearest Neighbor Rule," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 7, 1977, pp. 92-94.

[43] F. Ferri and E. Vidal, "Small sample size effects in the use of editing techniques," *Pattern Recognition, 1992. Vol.II. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on*, 1992, pp. 607-610.

[44] D. Angluin and P. Laird, "Learning from noisy examples," *Machine Learning*, vol. 2, Apr. 1988, pp. 343-370.

List of publications

Journals

[1] Donghai Guan, Weiwei Yuan, Young-Koo Lee, Andrey Gavrilov, and Sungyoung Lee. **Improving Supervised Learning Performance by Using Fuzzy Clustering Method to Select Training Data**. Journal of Intelligent & Fuzzy Systems, Vol 19, pp. 321-334, 2008. (SCIE)

[2] Donghai Guan, Weiwei Yuan, Young-Koo Lee, and Sungyoung lee. **Nearest Neighbor Editing Aided by Unlabeled Data**. Information Sciences, Vol 179, Issue 13, pp. 2273-2282, 2009. (SCI, impact factor 3.095)

Conferences (25)

[1] Donghai Guan, Yong-Koo Han, Young-Koo Lee, Sungyoung Lee and Chongkug Park, "Refining Classifier from Unsampled Data", Proceedings of 2009 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2009), Korea, (accepted), August, 2009

[2] Donghai Guan, Young-Koo Lee and Sungyoung Lee, "Activity Recognition with the Aid of Unlabeled Samples", Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication (ICUIMC’09), Korea, pp.720-724, January, 2009

[3] Donghai Guan, Weiwei Yuan, Young-Koo Lee, and Sungyoung Lee, “Training data selection based on Fuzzy C-means”, Proc. of Fuzz-IEEE (WCCI2008), Hong Kong, China. ISBN: 978-1-4244-1818-3, pp. 761-765.

[4] Donghai Guan, Weiwei Yuan, Young-Koo Lee, and Sungyoung Lee, “Semi-supervised Nearest Neighbor Editing”, Proc. of IJCNN-IEEE (WCCI2008), Hong Kong, China. ISBN: 978-1-4244-1820-6, pp. 1183-1187.

[5] Weiwei Yuan, Donghai Guan, and Sungyoung Lee, “Trust Management for Ubiquitous Healthcare,” Parallel and Distributed Processing with Applications, 2008. ISPA '08. International Symposium on, 2008, pp. 63-70.

[6] Donghai Guan, Weiwei Yuan, Sungyoung Lee and Young-Koo Lee, “Context Selection and Reasoning in Ubiquitous Computing”, The 2007 International

Conference on Intelligent Pervasive Computing (IPC-07), October 11th ~ 13th, 2007, in Jeju Island, Korea, ISBN: 978-0-7695-3006-0, pp. 184-187

[7] Donghai Guan, Weiwei Yuan, Seong Jin Cho, Andrey Gavrilov, Young-Koo Lee, Sungyoung Lee:, "Devising an Information Gain-based Reasoning Engine for Context-aware Ubiquitous Computing Middleware", Proc. of International Conference on Ubiquitous Intelligence and Computing (UIC 2007, LNCS), Hong Kong, China, July, 2007. ISBN: 978-3-540-73548-9, pp. 849-857

[8] Donghai Guan, Andrey V. Gavrilov, Weiwei Yuan, Young-Koo Lee and Sungyoung Lee, "A Novel Hybrid Neural Network for Data Clustering", The 2007 International Conference on Machine Learning, Models, Technologies and Applications, WorldComp 2007, June 25-28, Las Vegas, USA. ISBN 1-60132-027-2. pp. 284-288

[9] Donghai Guan, Weiwei Yuan, Young-Koo Lee, Andrey Gavrilov and Sungyoung Lee, "Combining Multi-layer Perceptron and K-means for Data Clustering with Background Knowledge", The 2007 International Conference on Intelligent Computing (ICIC2007, Springer), August 21-24, Qingdao, China. ISSN 1865-0929 (Print) 1865-0937 (Online). pp. 1220-1226

[10] Donghai Guan, Weiwei Yuan, Young-Koo Lee, Andrey Gavrilov and Sungyoung Lee, "Data Selection Based on Fuzzy Clustering", The 12th International Conference on Fuzzy Theory & Technology (JCIS 2007), July 18-24, USA. DOI No: 10.1142/9789812709677_0174, Source: INFORMATION SCIENCES 2007, pp 1231-1237.

[11] Donghai Guan, Weiwei Yuan, Young-Koo Lee, Andrey Gavrilov and Sungyoung Lee, "Activity Recognition Based on Semi-supervised Learning", The 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, August 21-24, 2007, Korea. ISSN: 1533-2306.ISBN: 978-0-7695-2975-2.pp. 469-475

[12] Donghai Guan, Andrey Gavrilov, Weiwei Yuan, Sungyoung Lee and Young-Koo Lee, "Data Clustering Using Hybrid Neural Network", the 27th KIPS Spring Conference, Korea. pp. 457-458.

[13] Weiwei Yuan, Donghai Guan, Sungyoung Lee, and Heejo Lee, "Using Reputation System in Ubiquitous Healthcare", The 9th IEEE International Conference on e-Health Networking, Application & Services (Healthcom 2007), Taipei, June 19-22, 2007. ISBN: 1-4244-0942-x, pp 182-186.

[14] Weiwei Yuan, Donghai Guan, Sungyoung Lee, and Young-Koo Lee, “The Role of Trust in Ubiquitous Healthcare”, the 9th IEEE International Conference on e-Health Networking, Application & Services (Healthcom 2007), Taipei, June 19-22, 2007. ISBN: 1-4244-0942-x, pp 312-315.

[15] Weiwei Yuan, Donghai Guan, Sungyoung Lee and Young-Koo Lee, “A Reputation System based on Computing with Words”, International Wireless Communications and Mobile Computing Conference 2007 (IWCMC 2007), August 12-16, 2007, Honolulu, Hawaii. ISBN:978-1-59593-695-0. pp, 132-137

[16] Weiwei Yuan, Donghai Guan, Sungyoung Lee, “The Role of Reputation in Ubiquitous Healthcare System”, the 27th KIPS Spring Conference, Korea. pp. 847-848.

[17] Weiwei Yuan, Donghai Guan, Sungyoung Lee and Heejo Lee, “Bayesian Memory-Based Reputation System”, International Mobile Multimedia Communications Conference (MobiMedia 2007), ISBN:978-963-06-2670-5, Article No. 9.

[18] Donghai Guan, Qing Li, Sungyoung Lee and Young-Koo Lee, "A Context-aware Music Recommendation Agent in Smart Office", The Second International Conference on Natural Computation (ICNC'06) and the Third International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'06), Xi'an, China, ISBN: 978-3-540-45916-3, ISSN: 0302-9743, LNCS 4223, pp. 1201-1204, http://www.icnc-fskd2006.org, September 24-28, 2006

[19] Donghai Guan, Weiwei Yuan, Andrey Gavrilov, Young-Koo Lee, Sungyoung Lee and Sang Man Han, "Using Fuzzy Decision Tree to Handle Uncertainty in Context Deduction", LNAI, SCIE. 2006 International Conference on Intelligent Computing, ISBN: 978-3-540-37274-5, ISSN: 0302-9743, LNAI 4114, pp. 63-72.

[20] Donghai Guan, Weiwei Yuan, Mohammad A. U. Khan, Young-Koo Lee, Sungyoung Lee and Sangman Han, "Utilizing a Hierarchical Method to Deal with Uncertainty in Context-aware Systems", LNAI, SCIE. 2006 International Conference on Intelligent Computing, ISBN: 978-3-540-37255-4, ISSN: 0170-8643, LNCIS 344, pp. 741-746.

[21] Weiwei Yuan, Donghai Guan, Sungyoung Lee, and Youngkoo Lee, “A Context-Based Architecture for Reliable Trust Model in Ubiquitous

Environments”, The 14th IEEE International Conference on Networks (ICON2006), Singapore, Sep 13-15, 2006. ISBN: 0-7803-9746-0, pp236-240.

[22] Weiwei Yuan, Donghai Guan, Le Xuan Hung, Sungyoung Lee, and Youngkoo Lee, “A Trust Model with Dynamic Decision Making For Ubiquitous Environments”, The 14th IEEE International Conference on Networks (ICON2006), Singapore, Sep 13-15, 2006. ISBN: 0-7803-9746-0, pp230-235.

[23] Weiwei Yuan, Donghai Guan, Sungyoung Lee, and Youngkoo Lee, "A Dynamic Trust Model Based on Naive Bayes Classifier for Ubiquitous Environments", LNCS, SCIE. The 2006 International Conference on High Performance Computing and Communications (HPCC-06). LNCS 4208, ISBN 3-540-39368-4, pp.562-571.

[24] Weiwei Yuan, Donghai Guan, Sungyoung Lee and Youngkoo Lee, “Finding Reliable Recommendations for Trust Model”, LNCS, SCIE. The 7th International Conference on Web Information Systems Engineering (WISE 2006). LNCS 4255, ISBN 3-540-48105-2. pp 375-386.

[25] Weiwei Yuan, Donghai Guan, Sungyoung Lee, Young-Koo Lee, and Heejo Lee, “Filtering out Unfair Recommendations for Trust Model in Ubiquitous Environments”, LNCS, Second International Conference on Information Systems Security (ICISS 2006) 17-21 December 2006, Kolkata, India. LNCS 4332, ISBN 3-540-68962-1, pp 357-360.

List of abbreviations

SSL	Semi-supervised Learning
ENN	Edited Nearest Neighbor
RENN	Repeated Edited Nearest Neighbor
ANN	All k-NN
NNE	Nearest Neighbor Editing
ENN+	Edited Nearest Neighbor Aided by Unlabeled Data
RENN+	Repeated Edited Nearest Neighbor Aided by Unlabeled Data
ANN+	All k-NN Aided by Unlabeled Data
MF	Majority Filtering
CF	Consensus Filtering
MFAUD	Majority Filtering Aided by Unlabeled Data
CFAUD	Consensus Filtering Aided by Unlabeled Data
k-NN	The k-Nearest Neighbor Algorithm
NB	Naïve Bayes
DT	Decision Tree