Appl Intell (2011) 35:345–358
DOI 10.1007/s10489-010-0225-4

Identifying mislabeled training data with the aid of unlabeled data

Donghai Guan · Weiwei Yuan · Young-Koo Lee · Sungyoung Lee

Published online: 26 March 2010
© Springer Science+Business Media, LLC 2010

Abstract This paper presents a new approach for identifying and eliminating mislabeled training instances for supervised learning algorithms. The novelty of this approach lies in the using of unlabeled instances to aid the detection of mislabeled training instances. This is in contrast with existing methods which rely upon only the labeled training instances. Our approach is straightforward and can be applied to many existing noise detection methods with only marginal modifications on them as required. To assess the benefit of our approach, we choose two popular noise detection methods: majority filtering (MF) and consensus filtering (CF). MFAUD/CFAUD is the new proposed variant of MF/CF which relies on our approach and denotes **m**ajority/**c**onsensus **f**iltering with the **a**id of **u**nlabeled **d**ata. Empirical study validates the superiority of our approach and shows that MFAUD and CFAUD can significantly improve the performances of MF and CF under different noise ratios and labeled ratios. In addition, the improvement is more remarkable when the noise ratio is greater.

This research was done when Dr. Donghai Guan worked in Ubiquitous Computing Lab, Kyung Hee University.

D. Guan
College of Automation, Harbin Engineering University,
145 Nantong Street, Nangang District, Harbin 150001, China
e-mail: donghai@oslab.khu.ac.kr

W. Yuan · Y.-K. Lee (✉) · S. Lee
Dept. of Computer Engineering, Kyung Hee University,
1 Seocheon-dong, Giheung-gu, Yongin-si, Gyeonggi-do
446-701, Korea
e-mail: yklee@khu.ac.kr

W. Yuan
e-mail: weiwei@oslab.khu.ac.kr

S. Lee
e-mail: sylee@oslab.khu.ac.kr

Keywords Supervised learning · Identifying mislabeled data · Unlabeled data · Majority filtering · Consensus filtering

1 Introduction

The goal of an inductive learning algorithm is to form a good generalization model constructed on training instances. Generally, two main factors which determine the quality of a generalization model are the quality of the training data and the appropriateness of the biases of the chosen learning algorithm for the training data. When the learning algorithm is given, the quality of the 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 the 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 either of the two approaches: noise tolerance and noise elimination. Noise tolerant 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 the category include rule truncation [1] and tree pruning [2]. On the contrary, noise elimination tries to improve the quality of training data by identifying and eliminating the noisy instances prior to applying the learning algorithm. One typical method in this category is to use an ensemble of classifiers and treat the training instance that is misclassified as noise. It has been argued by [3] that the noise tolerance is less effective than noise elimination. 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. For class noises, they are also called mislabeled noises since they are caused by the mislabeling. 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. Our focus is identifying and eliminating class noises (mislabeled instances), thereby increasing the classifier's predictive accuracy.

Up to now, many research efforts have been made on eliminating mislabeled instances for effective learning. Guyon [5] 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 [6] 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 [7, 8] 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 are 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 above methods can be represented by the expression: $R(t) = f(t, T)$, wherein $f(.)$ denotes the noise identification function which depends on a particular noise measure, such as instances typicality [5], reduction of CLCH [6], number of misclassification [7, 8], improvement of classification accuracy [10] and so on; t denotes the 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 identification function employed, there are always two parameters with the identification function: training set T and the instance t to be evaluated. In other words, given the identification function, the identification result for a training instance is only based on training set and this instance itself. Above methods do not consider the wide availability and potential utility of unlabeled data to noise identification.

The novelty of our approach is to make use of unlabeled data to aid the noise detection in training data. This is in contrast to the previous methods which rely upon only the training set. Let U denote the unlabeled set and then 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 those mentioned above.

To assess the benefit of our approach, two popular methods are chosen which are majority filtering (MF) and consensus filtering (CF). Based on our approach, the variant of MF (and CF) is proposed and called MFAUD (and CFAUD) which means majority (and consensus) filtering with the aid of unlabeled data. MF, CF, and our proposed MFAUD, CFAUD apply to a set of noisy data sets for comparison. Empirical study shows that MFAUD and CFAUD can significantly improve the performances of MF and CF under different noise levels and labeled ratios. In addition, the improvement is in direct proportion to the noise level.

The rest of the paper is organized as follows. Section 2 describes the works related to identifying mislabeled training instances, which include majority filtering and consensus filtering. Then we present MFAUD and CFAUD in Sect. 3. Section 4 discusses the experiments by using benchmark data. Section 5 summarizes our conclusions and future work.

2 Related works on identifying mislabeled training data

There are many methods for 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.

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:

Fig. 1 Majority filtering

Algorithm 1: MajorityFiltering (MF)
Input: E (training set)
Parameter: n (number of subjects), y (number of learning algorithms)
$A_1, A_2, \ldots, A_y$ (y kinds of learning algorithms)
Output: A (detected noisy subset of E)
(1) form n disjoint almost equally sized subsets of E_i, where $\bigcup_i E_i = E$
(2) $A \leftarrow \emptyset$
(3) **for** $i = 1, \ldots, n$ **do**
(4) form $E_t \leftarrow E \backslash E_i$
(5) **for** $j = 1, \ldots, y$ **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) $ErrorCounter \leftarrow 0$
(10) **for** $j = 1, \ldots, y$ **do**
(11) **if** H_j incorrectly classifies e
(12) **then** $ErrorCounter \leftarrow ErrorCounter + 1$
(13) **end for**
(14) **if** $ErrorCounter > \frac{y}{2}$, **then** $A \leftarrow A \cup \{e\}$
(15) **end for**
(16) **end for**

(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.

Shown in Fig. 1, 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_t 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 does not give the correct classification are added to A as potentially noisy examples (step 14).

Consensus filtering algorithm is shown in Fig. 2. Its only difference with MF is at step 14. In CF, the example in E_i is regarded as a 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, and which results in fewer instances being eliminated from the training set. The drawback of CF is the added risk in retaining bad data.

3 Our approach: MFAUD and CFAUD

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.

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 noises detection. The two phases are presented in Sects. 3.1 and 3.2 respectively.

3.1 Phase 1: predicting labels for unlabeled data

In machine learning, semi-supervised classification can provide a solution for this phase. Traditional classifiers use only labeled data to train. Alternatively, semi-supervised classification uses unlabeled data together with labeled data to train better classifiers. Semi-supervised classification uses unlabeled data to either modify or reprioritize hypotheses obtained from labeled data alone.

Semi-supervised classification can utilize unlabeled data in a variety of ways: we, however, are only interested in the way which explicitly predicts labels for unlabeled data so that the training set is enlarged.

Generally, there are two straightforward and popular semi-supervised methods following this way. They are self-training and co-training. In self-training [11], initially a

Fig. 2 Consensus filtering

Algorithm 2: ConsensusFiltering (CF)
Input: E (training set)
Parameter: n (number of subjects), y (number of learning algorithms)
$A_1, A_2, \ldots, A_y$ (y kinds of learning algorithms)
Output: A (detected noisy subset of E)
(1) form n disjoint almost equally sized subsets of E_i, where $\bigcup_i E_i = E$
(2) $A \leftarrow \emptyset$
(3) **for** $i = 1, \ldots, n$ **do**
(4) form $E_t \leftarrow E \backslash E_i$
(5) **for** $j = 1, \ldots, y$ **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) $ErrorCounter \leftarrow 0$
(10) **for** $j = 1, \ldots, y$ **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**

base learner is trained on labeled set and then it iteratively chooses to label several instances that it is most confident of in the unlabeled set. After that it enlarges its labeled training set with these self-labeled instances. Co-training [12] requires that features can be split into two sets; each subset is sufficient to train a good classifier; 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 aids the other classifier with the selected unlabeled instances (with their predicted labels) they are most confident of. Each classifier is retrained with the additional training instances given by the other classifier and the process repeats.

However, both self-training and co-training also present some problems. For self-training, on one hand, the classifier requires some measures to evaluate the "confidence" of unlabeled data. But many classifiers like k-nearest neighbors can not give this measure easily. On the other hand, even if the classifier could measure the confidence, its own prediction on the unlabeled data might not be as correct as two classifiers of co-training. 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 this section, we propose ensemble-based co-training (En-co-training) method through combining ensemble learning and co-training.

As shown in Fig. 3, multiple classifiers (more than two) are employed instead of the single classifier of self-training and two classifiers of co-training. En-co-training overcomes the limitations of self-training and co-training. Explicit measures of confidence are not required due to the ensemble voting of these classifiers. In addition, the diversity of two classifiers in co-training is achieved by using different sets of features. This process requires two views of features. But in En-co-training, the diversity of classifiers is achieved by using different algorithms. For example, three various algorithms are employed in our experiment: k-nearest neighbor, naive Bayes and decision tree. At the beginning of En-co-training, a small number of examples U' are randomly selected from U and then at the end of each iteration, more examples are replenished into U'. Related work [12] shows that this kind of setting is better than dealing with the whole U directly.

3.2 Phase 2: Utilizing unlabeled data and their predicted labels

Let T_U denote the output of phase 1, which includes the selected unlabeled instances and their predicted labels by En-co-training. 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 Figs. 4 and 5, T_U is utilized by our proposed MFAUD and CFAUD with only marginal modifications on MF and CF are required.

Always keep it in mind that there is no free lunch and it is usually 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 a training set. The reason is as follows: although semi-supervised method

Fig. 3 En-co-training algorithm

Algorithm 3: Ensemble-based Co-training (En-co-training)
Input: E (training set), U (unlabeled set)
Parameter: k (number of iterations), y (number of learning algorithms)
n (number of initially selected unlabeled instances)
$A_1, A_2, \ldots, A_y$ (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) $T_U \leftarrow \emptyset$
(3) **for** $i = 1, \ldots, n$ **do**
(4) $U \leftarrow U \backslash U'$, $num_{before} \leftarrow |T_U|$
(5) **for** $j = 1, \ldots, y$ **do**
(6) induce H_j based on instances in L and algorithm A_j
(7) **end for**
(8) **for** every $t \in U'$ **do**
(9) **for** $j = 1, \ldots, y$ **do**
(10) $pl_j(t) \leftarrow H_j(t)$// predicted label of H_j on t
(11) **end for**
(12) **if** $pl_1(t) = pl_2(t) =, \ldots, pl_y(t)$
(13) **then** $T_U \leftarrow t \cup pl_1(t)$, $U' \leftarrow U' \backslash t$
(14) **end for**
(15) $L \leftarrow L \cup T_U$
(16) $num_{after} \leftarrow |T_U|$, $\Delta num \leftarrow num_{after} - num_{before}$
(17) **if** $|u| >= \Delta num$
(18) **then** randomly choose Δnum 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**

including our En-co-training can predict the labels for some selected unlabeled data, the predicted labels are usually with errors. 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 the training set. With this caveat in mind, we now proceed to an empirical evaluation.

4 Empirical study

The main objective of the empirical study is to assess the benefit of unlabeled data for noise detection in training data. Section 4.1 explains the experimental setup. Afterwards Sect. 4.2 presents the experimental results.

4.1 Experimental setup

Existing MF, CF, and our proposed MFAUD, CFAUD are tested on the benchmark data sets from the Machine Learning Database Repository [13]. Information of these data sets is tabulated in Table 1. These data sets are collected from different real-world applications in various domains. Each data set is divided into training set and test set. A noise detection method works on the training set and outputs a filtered training set. Afterwards the test set is classified by the classifiers which are trained on 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.

Fig. 4 Majority filtering with the aid of unlabeled data

Algorithm 4: Majority Filtering with the Aid of Unlabeled Data (MFAUD)
Input: E (training set), U (unlabeled set)
Parameter: n (number of subjects), y (number of learning algorithms)
k (refer to Alg, of En-co-training)
n (refer to Alg, of En-co-training)
$A_1, A_2, \ldots, A_y$ (y kinds of learning algorithms)
Output: A (detected noisy subset of E)
(1) form n disjoint almost equally sized subsets of E_i, where $\bigcup_i E_i = E$
(2) $A \leftarrow \emptyset$
(3) **for** $i = 1, \ldots, n$ **do**
(4) form $E_t \leftarrow E \backslash E_i$
(5) $T_U = \text{En-co-training}(E_t, U, k, u, A_1, A_2, \ldots, A_y)$
(6) $E_t \leftarrow E_t \cup T_U$
(7) **for** $j = 1, \ldots, y$ **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) $ErrorCounter \leftarrow 0$
(12) **for** $j = 1, \ldots, y$ **do**
(13) **if** H_j incorrectly classifies e
(14) **then** $ErrorCounter \leftarrow ErrorCounter + 1$
(15) **end for**
(16) **if** $ErrorCounter > \frac{y}{2}$, **then** $A \leftarrow A \cup \{e\}$
(17) **end for**
(18) **end for**

Fig. 5 Consensus filtering with the aid of unlabeled data

Algorithm 5: Consensus Filtering with the Aid of Unlabeled Data (CFAUD)
Input: E (training set), U (unlabeled set)
Parameter: n (number of subjects), y (number of learning algorithms)
k (refer to Alg, of En-co-training)
n (refer to Alg, of En-co-training)
$A_1, A_2, \ldots, A_y$ (y kinds of learning algorithms)
Output: A (detected noisy subset of E)
(1) form n disjoint almost equally sized subsets of E_i, where $\bigcup_i E_i = E$
(2) $A \leftarrow \emptyset$
(3) **for** $i = 1, \ldots, n$ **do**
(4) form $E_t \leftarrow E \backslash E_i$
(5) $T_U = \text{En-co-training}(E_t, U, k, u, A_1, A_2, \ldots, A_y)$
(6) $E_t \leftarrow E_t \cup T_U$
(7) **for** $j = 1, \ldots, y$ **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) $ErrorCounter \leftarrow 0$
(12) **for** $j = 1, \ldots, y$ **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**

Table 1 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

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 and even can be neglected. However, the performance of noise removing methods need to be evaluated on 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 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 the training set 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 (1-NN), naive Bayes, and decision tree. The additional parameters in MFAUD and CFAUD are for En-co-training algorithm 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.

After filtering noises from the training set, a classification algorithm is trained from the filtered training set and then classfies the test data to evaluate the performance of each noise detection method. Generally when evaluating the performance of noise detection, the simple classification algorithms are more convincing than sophisticated algorithms. This is because the simple algorithm is not robust to noises; therefore, if a simple classification algorithm can achieve better performance with a certain noise detection method, it indicates that the performance of that noise detection method is better. Although the simple classification algorithms are preferred here, our work can be stronger and more persuasive if it is tested by both simple and sophisticated algorithms. Therefore, we select both simple and sophisticated algorithms to assess our work. Three simple and widely used classification algorithms are selected, which include k-nearest neighbor (1-NN), naive

Bayes, and decision tree. On the other hand, we also select three more sophisticated classification algorithms, including support vector machines [14], Multilayer Perception, and KStar [15].

In addition to above parameters, 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 whole data, referred to labeled ratio. The second 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 the two parameters to make the experiments comprehensively. The experiments include two parts:

Experiment A: We fix the labeled ratio to 50% and vary the noise ratios including 10%, 20%, 30%, and 40%. The experiments here show the comparisons between original noise detection methods and our proposed ones under different noise ratios. The objective is to test whether our proposed methods are robust to the noise ratio.

Experiment B: We fix the noise ratio to 20% and vary the labeled ratios including 10%, 20%, 30%, 40%, and 50%. The experiment in B shows the comparisons between original noise detection methods and our proposed ones under different labeled ratios and the objective is to test whether our proposed methods are robust to the labeled ratio.

4.2 Experimental results

Experiment A: Evaluation under different noise ratios

A-1: MF versus MFAUD under different noise ratios

The experiments in this part include the comparisons between MF and MFAUD under four different noise ratios. To avoid the tedious result presentation for each noise ratio with details, we select the minimal noise ratio 10% as the representative for analyzing purpose. Considering the simple classification algorithms are more suitable for assessing the performance of noise detection, we mainly present the experimental results based on the three simple algorithms (1-NN, naive Bayes, and decision tree). Afterwards, the experimental results from the sophisticated algorithms will be presented concisely.

In Table 2 we show the classification error for each data set of the classifiers formed by each of the three algorithms tested 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 simple algorithms. Table 2 shows for each of the three simple 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 naive Bayes, the improvement is 1.3%; for decision tree, the improvement is 4.0%.

Table 3 summarizes the comparisons between MF and MFAUD under four different noise ratios. As shown in this table, both MF and MFAUD could improve the classification performance of the three simple algorithms under any noise ratios, while MFAUD is even better than MF. 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%.

We further summarize the comparisons based on the other three relatively sophisticated algorithms in Table 4. The similar observations as Table 3 are obtained: MFAUD outperforms MF over different noise ratios. Moreover, the improvement of MFAUD on MF is more remarkable when the noise ratio is greater.

A-2: CF versus CFAUD under different noise ratios

Same as before, we select the minimal noise ratio 10% as representative for analyzing purpose. In Table 5 we show the classification error for each data set of the classifiers formed by each of the three algorithms tested 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%.

Table 6 shows the summarizations of CFAUD versus CF under four different noise ratios from 10% to 40% based on the three simple algorithms. This table indicates that both CF and 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. Averaged on the four noise ratios, the improvement of CFAUD on CF is around 12% for each of the three classifiers.

Table 2 MF versus MFAUD when noise ratio is 10%

Dataset	1-NN			Naive 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%	

Table 3 Summarizations of MF versus MFAUD under different noise ratios on the simple classification algorithms

Algorithm		Noise Ratio				
		10%	20%	30%	40%	**Ave.**
1-NN	None	0.280	0.340	0.386	0.442	0.362
	MF	0.221	0.240	0.262	0.324	0.262
	MFAUD	0.211	0.220	0.222	0.241	0.224
	Imp.	4.5%	8.1%	15.4%	25.6%	13.4%
NB	None	0.254	0.264	0.276	0.319	0.278
	MF	0.234	0.245	0.254	0.299	0.258
	MFAUD	0.231	0.238	0.246	0.247	0.241
	Imp.	1.3%	2.5%	3.2%	17.1%	6.03%
DT	None	0.242	0.277	0.320	0.396	0.309
	MF	0.218	0.228	0.251	0.319	0.254
	MFAUD	0.209	0.216	0.223	0.240	0.222
	Imp.	4.0%	5.4%	11.2%	24.8%	11.4%

We further summarize the comparisons between CF and CFAUD based on the other three relatively sophisticated algorithms in Table 7. The similar observations as Table 6 are obtained: CFAUD outperforms CF over different noise ratios. Moreover, in most cases, the improvement of MFAUD on MF increases when the noise ratio is greater.

Experiment B: Evaluation under different labeled ratios

In Experiment A, the labeled ratio is fixed, and the noise ratio is changed. While in Experiment B, the noise ratio is fixed, and the labeled ratio is altered. This is the only difference between these two experiments. To avoid tedious ex-

Table 4 Summarizations of MF versus MFAUD under different noise ratios on the sophisticated classification algorithms

Algorithm		Noise Ratio				
		10%	20%	30%	40%	**Ave.**
SVM	None	0.251	0.268	0.300	0.369	0.297
	MF	0.228	0.234	0.255	0.313	0.258
	MFAUD	0.223	0.226	0.230	0.264	0.236
	Imp.	2.1%	3.4%	9.7%	15.8%	7.75%
MP	None	0.240	0.278	0.328	0.387	0.308
	MF	0.213	0.229	0.249	0.324	0.254
	MFAUD	0.205	0.217	0.227	0.267	0.229
	Imp.	3.8%	5.1%	9.0%	17.6%	8.88%
KStar	None	0.288	0.331	0.385	0.433	0.359
	MF	0.237	0.257	0.296	0.335	0.281
	MFAUD	0.229	0.241	0.264	0.269	0.251
	Imp.	3.4%	6.4%	10.7%	19.7%	10.1%

Table 5 CF versus CFAUD when noise ratio is 10%

Dataset	1-NN			Naive 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%	

perimental results, we concisely give the summarizations of MF versus MFAUD and CF versus CFAUD under five different labeled ratios in Tables 8 and 9 respectively.

Tables 8 and 9 clearly show that our proposed noise detection methods MFAUD & CFAUD outperform the traditional MF & CF under all the five explored labeled ratios. In other words, our methods are robust to the labeled ratio.

Experiments summarization The results in Experiments A and B show that MFAUD and CFAUD could consistently improve MF and CF under different noise ratios and differ-

Table 6 Summarizations of CF versus CFAUD under different noise ratios on the simple classification algorithms

Algorithm		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%

Table 7 Summarizations of CF versus CFAUD under different noise ratios on the sophisticated classification algorithms

Algorithm		Noise Ratio				
		10%	20%	30%	40%	**Ave.**
SVM	None	0.251	0.268	0.300	0.369	0.297
	CF	0.235	0.252	0.273	0.346	0.277
	CFAUD	0.227	0.231	0.256	0.269	0.246
	Imp.	3.4%	8.3%	6.2%	22.3%	10.1%
MP	None	0.240	0.278	0.328	0.387	0.308
	CF	0.211	0.232	0.262	0.318	0.256
	CFAUD	0.206	0.215	0.229	0.257	0.227
	Imp.	2.4%	7.3%	12.6%	19.2%	10.4%
KStar	None	0.288	0.331	0.385	0.433	0.359
	CF	0.237	0.262	0.295	0.358	0.288
	CFAUD	0.223	0.249	0.266	0.283	0.255
	Imp.	5.9%	5.0%	9.8%	20.9%	10.4%

ent labeled ratios. In other words, the mislabeled instances detection performance could achieve consistent improvement with the aid of unlabeled data. Recall the discussion in Sect. 3, using unlabeled data is not free because their predicted labels might be noisy. On one hand, the unlabeled data with correctly predicted labels tend to improve the classification accuracy of ensemble classifiers, thereby upgrading the performance of mislabeled instances detection. On the other hand, the unlabeled data with incorrectly predicted labels tend to degrade the classification performance of ensemble classifiers, thereby degrading 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 as follows: 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 already shown that the predictive accuracy of each classifier (also ensemble classifiers) can be improved by using unlabeled data. Based on PAC theory [16], although there might be some noisy predicted labels for unlabeled data, the negative effect of them could be compensate if the amount of newly labeled examples is sufficient.

There is another useful result shown in the experiments. When the noise ratio increases, the improvement of MFAUD over MF and CFAUD over CF also increases. That is to say, unlabeled data give more remarkable improvement of noise detection in case that noise ratio is greater.

Note that, since the relative but not the absolute performance of the proposed methods is concern, the configuration

Table 8 Summarizations of MF versus MFAUD under different labeled ratios

Algorithm		Labeled ratio					
		10%	20%	30%	40%	50%	**Ave.**
1-NN	None	0.394	0.350	0.344	0.351	0.340	0.356
	MF	0.337	0.289	0.275	0.268	0.240	0.282
	MFAUD	0.307	0.255	0.244	0.240	0.220	0.253
	Imp.	8.9%	11.8%	11.3%	10.4%	8.1%	10.1%
NB	None	0.340	0.287	0.286	0.279	0.264	0.291
	MF	0.334	0.274	0.261	0.261	0.245	0.275
	MFAUD	0.305	0.251	0.243	0.251	0.238	0.258
	Imp.	8.7%	8.4%	6.9%	3.8%	2.5%	6.06%
DT	None	0.375	0.310	0.308	0.303	0.277	0.315
	MF	0.342	0.290	0.268	0.258	0.228	0.277
	MFAUD	0.318	0.262	0.242	0.237	0.216	0.255
	Imp.	7.0%	9.7%	9.7%	8.1%	5.4%	7.98%
SVM	None	0.327	0.294	0.289	0.285	0.268	0.293
	MF	0.324	0.283	0.278	0.257	0.234	0.275
	MFAUD	0.303	0.250	0.249	0.241	0.226	0.254
	Imp.	6.5%	11.7%	10.4%	6.2%	3.4%	7.64%
MP	None	0.355	0.311	0.315	0.301	0.278	0.312
	MF	0.331	0.273	0.261	0.242	0.229	0.267
	MFAUD	0.301	0.246	0.240	0.229	0.217	0.247
	Imp.	9.1%	9.9%	8.0%	5.4%	5.1%	7.5%
KStar	None	0.394	0.355	0.345	0.335	0.331	0.352
	MF	0.367	0.295	0.284	0.279	0.257	0.296
	MFAUD	0.332	0.257	0.261	0.258	0.241	0.270
	Imp.	9.5%	12.9%	8.1%	7.5%	6.4%	8.88%

of En-co-training has not been finely tuned. Based on our observations in experiments, En-co-training could achieve better performance by tuning the parameters such as the number of learning algorithms and iterations. Therefore, our proposed method is expected to work better in the real applications for which En-co-training is finely tuned.

Although the advantage of our proposed methods have been verified through a set of experiments, it does not mean that our methods are free of limitations. The first step of our methods is to obtain the initial classifier based on the training data and then predict the labels for some "confident" unlabeled data. The new labeled data are then used to aid the noise detection in the training data. With such a work flow, it is easy to understand that the quality of new labeled data (accuracy of the predicted labels with them) is an important factor in our method, while this factor is determined by the quality of the initial classifier. Heuristically, a better initial classifier leads to more accurate predicted labels (for unlabeled data), which consequently leads to better noise detection performance. Although we prefer the initial classifiers with high performance, it is true that these ideal initial classifiers are sometimes difficult to obtain since the amount of training data is not enough in many applications. Because the initial classifiers and their new labeled data are only used as the assistant of original training data, we only have an easy, rather than difficult, requirement on them. That is, the number of training data cannot be too small to build the initial classifiers with at least acceptable performance. For various applications, the number of training data to achieve the acceptable initial classifiers are also various. But through the experimental analysis, we find that it is usually easy to meet the requirement about the number of training data.

In fact, the above requirement on the number of training data is common for most existing noise detection methods. This is because most noise detection algorithms compose of two phases, wherein the first phase is to build a classifier with some of the training data; this classifier is then used in the second phase to detect the noises in the remaining samples of the training data according to a certain strategy. Therefore to use noise elimination, the number of training data also must reach some value to generate the classifier with at least acceptable performance.

Table 9 Summarizations of CF versus CFAUD under different labeled ratios

Algorithm		Labeled ratio					
		10%	20%	30%	40%	50%	**Ave.**
1-NN	None	0.394	0.350	0.344	0.351	0.343	0.357
	CF	0.337	0.279	0.255	0.253	0.263	0.277
	CFAUD	0.305	0.251	0.238	0.244	0.230	0.254
	Imp.	9.5%	10.0%	6.7%	3.6%	12.4%	8.44%
NB	None	0.340	0.287	0.286	0.279	0.264	0.291
	CF	0.345	0.281	0.264	0.259	0.252	0.280
	CFAUD	0.321	0.257	0.246	0.249	0.244	0.263
	Imp.	7.0%	8.5%	6.8%	3.9%	3.3%	5.90%
DT	None	0.375	0.310	0.308	0.303	0.277	0.315
	CF	0.346	0.296	0.264	0.251	0.241	0.280
	CFAUD	0.319	0.258	0.241	0.233	0.218	0.254
	Imp.	7.8%	12.8%	8.7%	7.2%	9.3%	9.16%
SVM	None	0.327	0.294	0.289	0.285	0.268	0.293
	CF	0.339	0.279	0.265	0.258	0.252	0.279
	CFAUD	0.303	0.248	0.241	0.239	0.231	0.252
	Imp.	10.6%	11.1%	9.1%	7.4%	8.3%	9.30%
MP	None	0.355	0.311	0.315	0.301	0.278	0.312
	CF	0.320	0.261	0.250	0.238	0.232	0.260
	CFAUD	0.249	0.229	0.233	0.226	0.215	0.230
	Imp.	22.2%	12.3%	6.8%	5.0%	7.3%	10.7%
KStar	None	0.394	0.355	0.345	0.335	0.331	0.352
	CF	0.372	0.292	0.282	0.274	0.262	0.296
	CFAUD	0.321	0.269	0.261	0.258	0.249	0.272
	Imp.	13.7%	7.9%	7.4%	5.8%	5.0%	7.96%

5 Conclusions and future works

This article presents a new approach for identifying mislabeled instances, which considers the wide availability and potential utility of unlabeled instances and makes use of them to improve the performance of existing methods.

Majority filtering (MF) and consensus filtering (CF) are two popular mislabeled instance detection methods and they are chosen to assess the benefit of our approach. To this end, firstly we proposed the variants of MF and CF which are aided by unlabeled data and called MFAUD and CFAUD. The results of an empirical evaluation demonstrated that MFAUD and CFAUD can significantly improve the classification performance of MF and CF. Moreover, our experiments show that as the noise ratio increases, the improvements of MFAUD over MF and CFAUD over CF increase.

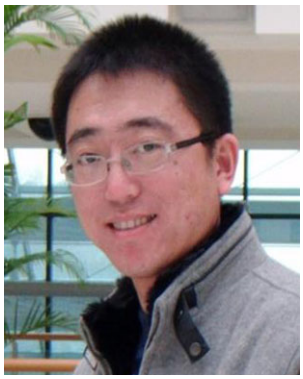
In this work the superiority of our methodology is validated on MF and CF. Our future work is to further validate our approach based on other mislabeled instance detection methods. Moreover, we will consider how our work can contribute to other machine learning topics, including, for instance, active learning, ensemble learning, and feature selection. The reason is that these topics also need to deal with noisy training instances to achieve good performance.

Acknowledgements This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Advancement) (IITA-2010-(C1090-1002-0003)), also was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MEST) (No. 2009-1515).

References

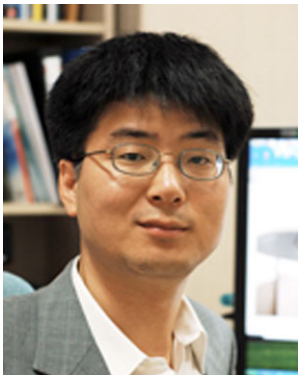
1. Mingers J (1989) An empirical comparison of pruning methods for decision tree induction. Mach Learn 4(2):227–243
2. Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann, San Francisco
3. Gamberger D, Lavrac N, Dzeroski S (2000) Noise detection and elimination in data proprocessing: Experiments in medical domains. Appl Artif Intell 14(2):205–223
4. Quinlan JR (1986) Induction of decision trees. Mach Learn 1(1):81–106

5. Guyon I, Matic N, Vapnik V (1996) Discovering information patterns and data cleaning. In: Advances in knowledge discovery and data mining. AAAI/MIT Press, Cambridge, pp 181–203
6. Gamberger D, Lavrac N, Groselj C (1999) Experiments with noise filtering in a medical domain. In: Proceedings of 16th international conference on machine learning, pp 143–151
7. Brodley CE, Friedl MA (1996) Identifying and eliminating mislabeled training instances. In: Proceedings of 13th national conference on artificial intelligence, pp 799–805
8. Brodley CE, Friedl MA (1999) Identifying mislabeled training data. J Artif Intell Res 11:131–167
9. Wilson D (1972) Asymptotic properties of nearest neighbor rules using edited data. IEEE Trans Syst Man Cybern 2:408–421
10. Aha D, Kibler D, Albert M (1991) Instance-based learning algorithms. Mach Learn 6(1):37–66
11. Riloff E, Wiebe J, Wilson T (2003) Learning subjective nouns using extraction pattern bootstrapping. In: Proceedings of the 7th conference on natural language learning, pp 25–32
12. Blum A, Mitchell T (1998) Combining labeled and unlabeled data with co-training. In: Proceedings of 11th annual conference on computational learning theory, pp 92–100
13. UCI KDD archive. http://kdd.ics.uci.edu
14. Keerthi S, Shevade S, Bhattacharyya C, Murthy K (2001) Improvements to Platt's SMO algorithm for SVM classifier design. Neural Comput 13(3):637–649
15. John G, Leonard E (1995) An instance-based learner using an entropic distance measure. In: Proceedings of the 12th international conference on machine learning, pp 108–114
16. Angluin D, Laird P (1988) Learning from noisy examples. Mach Learn 2(2):343–370



Donghai Guan received his B.S. in College of Automation from Harbin Engineering University (HEU), Harbin, China in 2002. He got his M.S. degree in Computer Science from Kumoh National Institute of Technology (KIT), Gumi, South Korea in 2004. He got his Ph.D. degree in Computer Science from Kyung Hee University, South Korea in 2009. From 2009, he was a Post Doctoral Fellow at Computer Science Department, Kyung Hee University. His research interests are Machine Learning, Pattern Recognition, Data Mining, Activity Recognition, and Trust modeling.

Weiwei Yuan received B.S. and M.S. in Automation and Computer Engineering in 2002 and 2005 respectively from Harbin Engineering University, China. Currently she is a Ph.D. candidate in the Department of Computer Engineering, Kyung Hee University, South Korea. Her research interests are trust models, reputation systems, information security and machine learning.



Young-Koo Lee received his BS, MS, and Ph.D. in Computer Science from Korea Advanced Institute of Science and Technology (KAIST), Korea in 1988, 1994 and 2002, respectively. Since 2004, he has been an assistant professor at the Dept. of Computer Engineering, College of Electronics and Information, Kyung Hee University, Korea. From 2002 to 2004, he was a Post Doctoral Fellow Advanced Information Technology Research Center(AITrc), KAIST, Korea, and a Postdoctoral Research Associate at Dept. of Computer Science, University of Illinois at Urbana- Champaign, USA. His research interests are Ubiquitous Data Management, Data Mining, Activity Recognition, Bioinformatics, Online Analytical Processing, DataWarehousing, Database Systems, Spatial Databases, and Access Methods.

Sungyoung Lee received his B.S. from Korea University, Seoul, South Korea. He got his M.S. and Ph.D. degrees in Computer Science from Illinois Institute of Technology (IIT), Chicago, Illinois, USA in 1987 and 1991 respectively. He has been a professor in the Department of Computer Engineering, Kyung Hee University, South Korea since 1993. He is a founding director of the Ubiquitous Computing Laboratory, and has been affiliated with a director of Neo Medical ubiquitous- Life Care Information Technology Research Center, Kyung Hee University since 2006. Before joining Kyung Hee University, he was an assistant professor in the Department of Computer Science, Governors State University, Illinois, USA from 1992 to 1993. His current research focuses on Ubiquitous Computing and applications, Context-aware Middleware, Sensor Operating Systems, Real-Time Systems, and Embedded Systems. He is a member of the ACM and IEEE.