Information Sciences 179 (2009) 2273–2282

Contents lists available at ScienceDirect

Information Sciences

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

Nearest neighbor editing aided by unlabeled data

Donghai Guan, Weiwei Yuan, Young-Koo Lee *, Sungyoung Lee

Department of Computer Engineering, Kyung Hee University, 446 701 Yongin, Republic of Korea

ARTICLE INFO

Article history:
Received 15 April 2008
Received in revised form 20 January 2009
Accepted 13 February 2009

Keywords:
Nearest neighbor editing
Unlabeled data
Edited nearest neighbor
Repeated edited nearest neighbor
All *k*-NN

ABSTRACT

This paper proposes a novel method for nearest neighbor editing. Nearest neighbor editing aims to increase the classifier's generalization ability by removing noisy instances from the training set. Traditionally nearest neighbor editing edits (removes/retains) each instance by the voting of the instances in the training set (labeled instances). However, motivated by semi-supervised learning, we propose a novel editing methodology which edits each training instance by the voting of all the available instances (both labeled and unlabeled instances). We expect that the editing performance could be boosted by appropriately using unlabeled data. Our idea relies on the fact that in many applications, in addition to the training instances, many unlabeled instances are also available since they do not need human annotation effort. Three popular data editing methods, including edited nearest neighbor, repeated edited nearest neighbor and All *k*-NN are adopted to verify our idea. They are tested on a set of UCI data sets. Experimental results indicate that all the three editing methods can achieve improved performance with the aid of unlabeled data. Moreover, the improvement is more remarkable when the ratio of training data to unlabeled data is small.

1. Introduction

The *k*-nearest neighbor algorithm (*k*-NN) is an important classification algorithm. This algorithm firstly finds the *k* nearest neighbors to each target instance according to a certain dissimilarity measure and then makes a decision according to the known classification of these neighbors, usually by assigning the label of the most voted class among these *k* neighbors [6]. The asymptotic classification error of *k*-NN (when the number of training data *n* grows to infinity) tends to the optimal Bayes error as $k \to \infty$ and $k/n \to 0$. If $k = 1$, the error is bounded by approximately twice the Bayes error [8]. *k*-NN is a type of lazy learning where the function is only locally approximated and all computation is deferred until classification. Therefore, compared with other classifiers which require training, *k*-NN learns more easily because it only needs to read in the training set without much further processing.

However, due to the high degree of local sensitivity, *k*-NN is highly susceptible to noisy instances in the training set [6]. The noisy instances include those with errors in the input vector or output class, or those not representative of typical cases. The noisy instances can degrade the generalization ability of *k*-NN [17].

In order to minimize the downside of noisy instances in *k*-NN, people mainly take either of the two approaches: indirect or direct. The indirect approach tries to control the negative effect of noisy instances without removing them. The methods in this category include increasing the number of nearest neighbors or utilizing distance weighted voting. On the contrary, the direct approach tries to remove the noisy instances. Nearest neighbor editing [17] is one of the most significant representatives of this approach. It eliminates erroneous instances from the training set and cleans possible overlapping among classes. In this paper, we mainly focus on nearest neighbor editing.

* Corresponding author. Tel.: +82 31 201 3732.
E-mail address: yklee@khu.ac.kr (Y.-K. Lee).

0020-0255/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2009.02.011

We propose a novel nearest neighbor editing method that utilizes both labeled and unlabeled data sets. In particular, this is in contrast with existing editing methods that rely upon only the training data set (labeled data set) and focus upon the two factors of search direction and instance location. The advantages of our method include the use of unlabeled data that are often convenient to obtain in many applications with only marginal modifications required to existing data editing methods.

Our method is motivated by the works on semi-supervised learning which make use of both labeled data and unlabeled data for 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. For instance, unlabeled data have been used for ensemble learning [2,24], dimension reduction [18,22], active learning [4,5,15] and feature selection [11,23].

Our method is applied in two phases wherein the first provides predictive labeling for unlabeled data and the second incorporates the augmented data editing methods.

To evaluate the performance of our method, three popular data editing techniques are adopted. They are edited nearest neighbor, repeated edited nearest neighbor and All k-NN. We use a set of benchmark data sets from the Machine Learning Database Repository to test the data editing performance. Experimental results demonstrate that the performances of all these three methods are improved with the aid of unlabeled data especially in the situation where the ratio between training data to unlabeled data is small.

The rest of the paper is organized as follows. Section 2 describes the related works on data editing. Then we present our proposed method in Section 3. Our method is applied in two phases which are presented in Sections 3.1 and 3.2 respectively. Section 4 discusses the experiments by using benchmark data. Section 5 summarizes our conclusions and future work.

2. Related works on nearest neighbor editing

There are many data editing algorithms. Herein, we consider the edited nearest neighbor (ENN) [21], repeated edited nearest neighbor (RENN) [19] and All k-NN (ANN) [19] algorithms due to their wide-spread and popular use 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. 1 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.

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 that the training of the corresponding classifier becomes easier in practice. In fact, it has been shown by Penrod and Wagner [14] that the accuracy of the ENN classifier converges to Bayes error as the number of instances approaches infinity. Table 1 gives the ENN algorithm.

RENN applies the ENN algorithm repeatedly until all remaining instances have a majority of their neighbors with the same class, which continues to widen the gap between classes and smooth the decision boundary of ENN. The RENN algorithm is shown in Table 2.

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 3.

The common factor in all 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

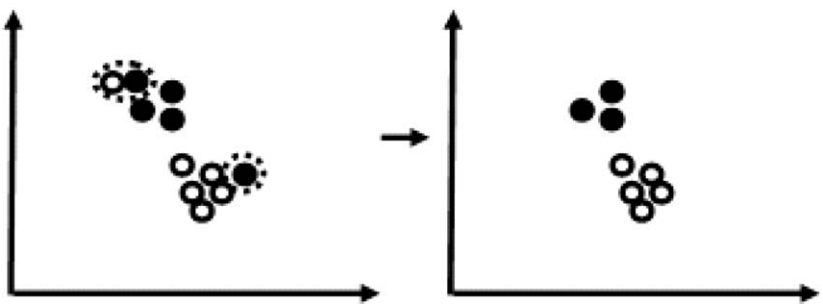

Fig. 1. ENN editing with 1-NN classifier.

Table 1
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 $T_e/\{x_i\}$

Table 2
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 $x_i \in T_e^t$, do
Discard x_i from T_e^t if it is misclassified using the k-NN rule with prototypes in $T_e^t/\{x_i\}$;
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

Table 3
The algorithm of All k-NN (ANN).

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:
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 $T_e/\{x_i\}$, go to Step 2.
2.2.2 Set $m = m+1$

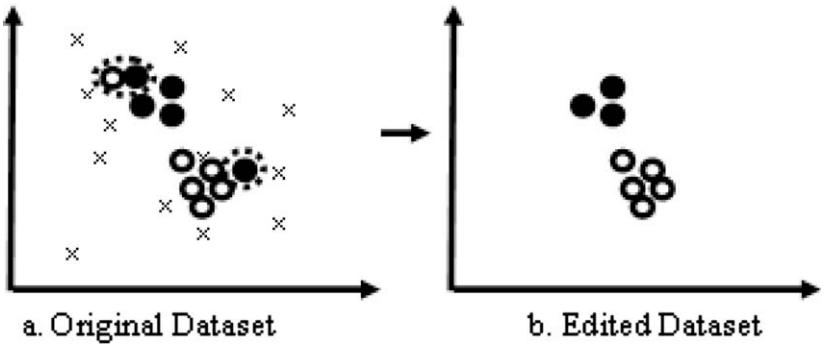

Fig. 2. ENN with 1-NN classifier when unlabeled data are available.

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. 2, "×" 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 exit. The editing result is shown in the right part of Fig. 2.

3. Our proposed method: data editing aided by unlabeled data

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 two phases are presented in Sections 3.1 and 3.2 respectively.

3.1. Phase 1: Predicting labels for unlabeled data

In machine learning area, 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 by either modifying or reprioritizing hypotheses obtained from labeled data only.

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 [16], a base learner is firstly trained on the labeled set. Then, iteratively, it attempts to choose 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 [3] requires that features can be split into two sets; each sub-set 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 few unlabeled instances (and the predicted labels) they feel most confident. Each classifier is retrained with the additional training instances given by the other classifier and the process repeats.

Table 4
The algorithm of ensemble-based co-training (En-co-training).

Given:
- L, A set of labeled training examples, consisting of M classes
- U, A set of unlabeled examples

Create a pool U' of examples by choosing u examples at random from U
Loop for k iterations:
(1) Use L to train a classifier h_1, h_2 and h_3 respectively
(2) For each class C, pick the n_c unlabeled data which classifier h_1, h_2 and h_3 agree with that its class label is C and add it to the collection of labeled examples
(3) Randomly choose $\sum_{c=1}^{M} n_c$ examples from U to replenish U'
Output: The new labeled data generated from step (2).

However, both of self-training and co-training still present some problems. For self-training, on one hand, the classifier requires some measures to evaluate the "confidence" of unlabeled data. In fact, many classifiers like k-nearest neighbor 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 shown in Table 4 by combining ensemble learning and co-training.

Three classifiers (algorithms) are used instead of the individual 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 majority voting of these three classifiers. In addition, the diversity of the 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 instance, in our experiment, k-nearest neighbor, naive Bayes and decision tree are used to train three different classifiers, respectively. At the beginning of En-co-training, u instances are randomly selected from the unlabeled set U to be processed. Related work [3] shows that this kind of setting is better than dealing with the whole U directly. Then this algorithm will run k times to iteratively select more unlabeled instances to process. In En-co-training, k is employed to control the number of unlabeled instances used to aid the data editing process. With k growing, more unlabeled instances are likely to be selected. k is set to 2 in the following experiments. With respect to u, in general, the output of En-co-training tends to be better when u is smaller. However, a small u will lead to the long execution time of En-co-training. Empirically it is an adequate trade-off between the performance of En-co-training and its execution time when u equals to the number of labeled data.

3.2. Phase 2: Utilizing the augmented data set in data editing

The nearest neighbors of a training instance obtained from a search of the training set and those from the whole data set may be different. The variation of nearest neighbors might lead to a change of editing result. The effect of our method is illustrated in Fig. 3. Let T denote the training set and T_U denote the unlabeled set with their predicted labels from Phase 1. 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.

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 5–7. Tables 5–7 show

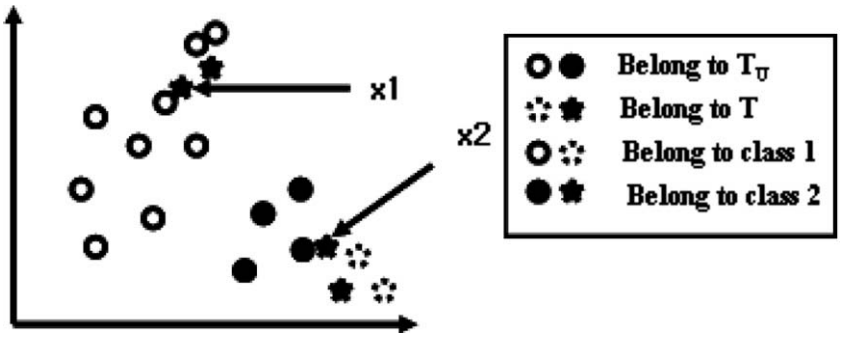

Fig. 3. The effect of unlabeled data for data editing.

Table 5
The algorithm of ENN aided by unlabeled data.

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 $(T_e/\{x_i\}) \cup T_U$

Table 6
The algorithm of RENN aided by unlabeled data.

1. Let $T_e = T \cup T_U//T$ is the original training set, and T_e is the edited set
REPEAT
2. At iteration t, for each $x_i \in T_e^t$, do
Discard x_i from T_e^t if it is misclassified using the k-NN rule with prototypes in $T_e^t/\{x_i\}$
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 7
The algorithm of ANN aided by unlabeled data.

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:
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 $(T_e/\{x_i\}) \cup T_U$, go to Step 2.
2.2.2 Set $m = m+1$

two different usages of T_U. In Tables 5 and 7, T_U is used to optimize the editing performance of T while T_U is never edited. In Table 6, T_U and T are edited together. But as shown in Step 3 of Table 6, $T_e = T_e \cap T$, 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.

Although semi-supervised classification could predict the labels for unlabeled data, it is intractable to predict labels without errors. That is the reason the predicted labels usually include some errors. The erroneous labels could potentially degrade the editing performance. Considering the following three cases (Fig. 4) 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 En-co-training 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 $T_{e(c)}$.

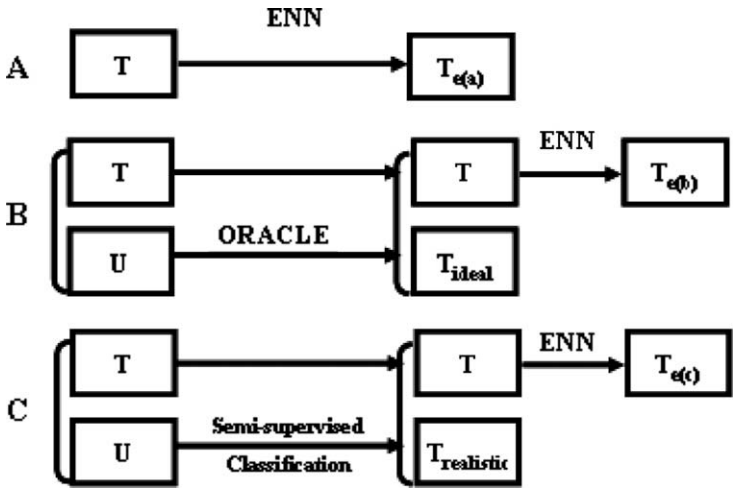

Fig. 4. (a) Standard ENN (b) ENN aided by unlabeled data (processed by the Oracle) (c) ENN aided by unlabeled data (processed by semi-supervised classification).

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 twofold. 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.

4. Experimental results

The objective of the experiments is to validate the benefit of our method compared with existing methods described in Section 2. Section 4.1 explains the experimental setup. Followed by Section 4.2 presents the experimental results.

4.1. Experimental setup

Three data editing techniques are used and tested on the benchmark data sets from the Machine Learning Database Repository [20]. 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 these data sets is tabulated in Table 8. 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 19,020 instances consisting of two classes. To reduce the experiment time, we extract the first 1000 instances from each class. Therefore, 2000 samples are used. 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 tenfold cross-validation on L are used to evaluate the performance of data editing methods. At each trial, 90% of L, that is T, is used as training set. T is edited by various data 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 accuracy is obtained by averaging ten trials' accuracies.

Table 8
UCI data sets used in the experiments.

Dataset	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

(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 accuracy is the further averaged value of these five values.

In this experiment, En-co-training (Section 3) is adopted and its configuration is as follows. Three classifiers are generated by: 3-nearest neighbor, naive Bayes and decision tree respectively. Initially, the size of U', u is equal to the size of the training set, namely $u = |L|$. Iteration number k is 2.

When the setting of semi-supervised classification is fixed, there are three major parameters which can influence the experiment. The first parameter determines data partitioning and it is the ratio between labeled data to whole data, referred to labeled ratio. The second parameter determines data editing. Refer to the data editing methods mentioned before, k-NN algorithm is the base of data editing methods. Therefore, different choices of k can influence the experiment. 3-NN and 1-NN are the popular settings in the data editing area. The third parameter determines data testing. One popular setting is 1-NN for testing. We have performed several experiments varying these parameters.

Experiment A: The labeled ratio is 5%. This value is used to represent the applications in which the number of unlabeled data is much greater than the number of labeled data. In fact, this applies to most applications since unlabeled data are often much easier to obtain as compared with labeled data. In addition, 3-NN is used for data editing and 1-NN is used for data testing.

Experiment B: The labeled ratios are 10%, 15%, 20%, 30%, 40%, and 50%. This is the same as Experiment A that 3-NN is used for data editing and 1-NN is used for data testing. The objectives of Experiment B are twofold. First, we further validate the performance of our methods by varying the labeled ratio. Second, we want to know whether there are some relationships between the labeled ratio and the data editing performance.

Experiment C: The labeled ratios are 5%, 10%, 15%, 20%, 30%, 40%, and 50%. Different from Experiments A and B, here 1-NN is used for data editing and data testing. As a complement to Experiments A and B, this experiment is used to validate the robustness of our method when k-NN in data editing is changed.

4.2. Experimental results

Experiment A: Table 9 shows the results of Experiment A. In each cell, the value without bracket represents the classification accuracy. In addition to that, we also give the retention rate that is below the accuracy and in brackets. Data retention rate is defined as

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

Table 9
Data editing performance when labeled ratio is 5% (3-NN for data editing, 1-NN for testing). The value without bracket represents the classification accuracy; the value with bracket represents the retention ratio; in each row, the better accuracy of each pair is shown in bold; the significant better accuracy is shown in bold with underline.

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	

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 9 and the following tables, "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.

Experiment B: This experiment is implemented for labeled ratios 10%, 15%, 20%, 30%, 40%, and 50%. We summarize the results from different labeled ratios in Table 10. Here, 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. This experiment further validates the superiority of our methods since they work well for different labeled ratios.

The other interesting observation obtained from this table is that although improvement is not strictly inversely proportional to the labeled ratio, the greatest improvement is usually obtained when labeled ratio is small (5% or 10% in the experiments).

Experiment C: Different with above experiments, 1-NN is used for data editing in Experiment C. Rather than giving the experiment in detail, we summarize the results into Table 11. When 1-NN is used for data editing, ENN is same with ANN, therefore ANN and ANN+ are ignored in this table.

Table 11 has similar experimental observations with Tables 10 and 9. This suggests that our methods work well independently with *k*-NN algorithm of data editing.

Experiments A, B, and C indicate that when using ENN, RENN and ANN for data editing, the data editing performance could achieve consistent improvement with the aid of unlabeled data. Recall the discussion in Section 3, intuitively, it is like a double-edged sword to use unlabeled data for data editing. On one hand, unlabeled data (labeled by semi-supervised classification) might improve editing performance by providing extra information. On the other hand, the provided information

Table 10
Data editing performance summarization when labeled ratio is 5%, 10%, 15%, 20%, 30%, 40%, and 50% (3-NN for data editing, 1-NN for testing).

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

Table 11
Data editing performance summarization when labeled ratio is 5%, 10%, 15%, 20%, 30%, 40%, and 50% (1-NN for data editing, 1-NN for testing).

		Labeled ratio						
		5%	10%	15%	20%	30%	40%	50%
ENN+ & ENN	Accu.	**0.727**	0.741	0.760	0.774	0.783	0.791	0.790
		0.694	0.715	0.742	0.757	0.767	0.786	0.783
		(**0.033**)	(0.026)	(0.018)	(0.017)	(0.016)	(0.005)	(0.007)
	B/W	18/2	**20/1**	18/2	13/5	17/2	15/6	16/5
		(16)	(**19**)	(16)	(8)	(15)	(9)	(11)
	S.B/W	**14/1**	10/1	7/0	7/0	6/0	4/2	4/0
		(**13**)	(9)	(7)	(7)	(6)	(2)	(4)
RENN+ & RENN	Accu.	**0.728**	0.743	0.760	0.773	0.781	0.791	0.790
		0.688	0.707	0.738	0.754	0.764	0.785	0.784
		(**0.040**)	(0.036)	(0.022)	(0.019)	(0.017)	(0.006)	(0.006)
	B/W	**20/0**	20/1	18/3	16/5	15/6	14/6	15/6
		(**20**)	(19)	(15)	(11)	(9)	(8)	(9)
	S.B/W	**14/0**	13/0	13/1	6/0	6/0	4/2	3/1
		(**14**)	(13)	(12)	(6)	(6)	(2)	(2)

might degrade the editing performance as there might be some mislabeled data generated by semi-supervised classification. The experimental results show that the positive effect of using unlabeled data defeats its negative effect. One possible explanation is as follows: ENN (RENN or ANN) edits an instance based on the voting of its nearest neighbors. An instance is regarded as noisy instance to remove when its label is different from its label obtained by the voting of its nearest neighbors, which is called voting label here. Heuristically, data editing works well only when the voting label of an instance is same with its real label. In other words, data editing performance is expected to be improved when the training instances' labels are predicted more correctly by their nearest neighbors. In k-NN, this means that data editing performance is in direct ratio to the generalization ability. Existing works on semi-supervised learning have already shown that the generalization ability can be improved by using unlabeled data. Based on PAC theory [1], although there might be some noisy instances when predicting the labels for unlabeled data, the negative effect of noisy instances could be compensated if the amount of newly labeled examples is sufficient.

There is another useful result shown in the 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 data ratio is small (labeled data number is small). Existing work on data editing [9] 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.

Note that, since the relative but not the absolute performance of the proposed methods is concerned, the configuration 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 classifiers and iterations. Therefore, our proposed method is expected to work better in the real applications for which En-co-training is finely tuned.

5. Conclusions and future work

In this work, we propose to utilize unlabeled data to boost the performance of data editing on the training set. Our method is applied in two phases. The first phase is to predict labels for the unlabeled data and then the second phase is to utilize the augmented data set in data editing.

Wilson's Edited Nearest Neighbor (ENN), Tomek's Repeated ENN (RENN), and All k-NN (ANN) are used with 21 UCI data sets to verify our method. Experimental results show that all the three editing techniques' performances are improved with the aid of the unlabeled data. In addition, more improvement is obtained in the case that the size of training set is small.

Apart from the three data editing techniques we used, there are many other related ones. Therefore, one future work will investigate the using of unlabeled data for other data editing methods. Moreover, data editing is a kind of instance selection technique [7,12,13] which consists of selecting an appropriate reduced sub-set of training instances and applying the k-NN rule using only the selected instances. In addition to data editing, data condensing [10] is the other instance selection technique with focuses on data reduction more than generalization ability. Another future work is to investigate the using of unlabeled data for data condensing.

Acknowledgements

Many thanks to Prof. Brian J. d'Auriol for proofreading of our paper. We also thank the reviewers for their constructive comments. 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-2009-(C1090-0902-0002)). This work was also supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MOST) (No. 2008-1342).

References

[1] D. Angluin, P. Laird, Learning from noisy examples, Machine Learning 2 (2) (1988) 343–370.
[2] K. Bennett, A. Demiriz, R. Maclin, Exploiting unlabeled data in ensemble methods, in: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada, 2002, pp. 289–296.
[3] A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, in: Proceedings of 11th Annual Conference on Computational Learning Theory, Wisconsin, USA, 1998, pp. 92–100.
[4] J. Cheng, K. Wang, Active learning for image retrieval with Co-SVM, Pattern Recognition 40 (1) (2007) 330–334.
[5] C. Constantinopoulos, A. Likas, Semi-supervised and active learning with the probabilistic RBF classifier, Neurocomputing 71 (13–15) (2008) 2489–2498.
[6] T. Cover, P. Hart, Nearest neighbor pattern recognition, IEEE Transaction on Information Theory 13 (1967) 21–27.
[7] R.K. De, S.K. Pal, A connectionist model for selection of cases, Information Sciences 132 (1–4) (2001) 179–194.
[8] P. Devijver, J. Kittler, Pattern Recognition: A Statistical Approach, Prentice Hall, Eaglewood Cliffs, NJ, 1982.
[9] F. Ferri, E. Vidal, Small sample size effects in the use of editing techniques, in: Proceedings of 11th International Conference of Pattern Recognition, Hague, The Netherlands, 1992, pp. 607–610.
[10] D.J. Hand, B.G. Batchelor, Experiments on the edited condensed nearest neighbor rule, Information Sciences 14 (3) (1978) 171–180.
[11] J. Handl, J. Knowles, Semi-supervised feature selection via multiobjective optimization, in: Proceedings of 2006 International Joint Conference on Neural Networks, Vancouver, Canada, 2006, pp. 3319–3326.
[12] C.H. Liu, L.S. Chen, C.C. Hsu, An association-based case reduction technique for case-based reasoning, Information Sciences 178 (17) (2008) 3347–3355.
[13] H. Liu, H. Motoda, Instance Selection and Construction for Data Mining, Kluwer Academic Publishers, Massachusetts, 2001.
[14] C. Penrod, T. Wagner, Another look at the edited nearest neighbor rule, IEEE Transactions on Systems, Man, and Cybernetics 7 (1977) 92–94.
[15] T. Qin, X.D. Zhang, T.Y. Liu, An active feedback framework for image retrieval, Pattern Recognition Letters 29 (5) (2008) 637–646.
[16] E. Riloff, J. Wiebe, T. Wilson, Learning subjective nouns using extraction pattern bootstrapping, in: Proceedings of the 7th Conference on Natural Language Learning, Edmonton, Canada, 2003, pp. 25–32.
[17] B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge Univ. Press, Cambridge, U.K, 1996. pp. 198–201.
[18] Y. Song, F. Nie, C. Zhang, S. Xiang, A unified framework for semi-supervised dimensionality reduction, Pattern Recognition 41 (9) (2008) 2789–2799.
[19] I. Tomek, An experiment with the edited nearest-neighbor rule, IEEE Transactions on Systems, Man, and Cybernetics 6 (6) (1976) 448–452.
[20] UCI KDD Archive. <http://kdd.ics.uci.edu>.
[21] D.L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Transaction on Systems, Man, and Cybernetics 2 (3) (1972) 431–433.
[22] D. Zhang, Z. Zhou, S. Chen, Semi-supervised dimensionality reduction, in: Proceedings of the 7th International Conference on Data Mining, Omaha, USA, 2007, pp. 629–634.
[23] J. Zhao, K. Lu, X. He, Locality sensitive semi-supervised feature selection, Neurocomputing 71 (10–12) (2008) 1842–1849.
[24] X. Zhu, A lazy bagging approach to classification, Pattern Recognition 41 (10) (2008) 2980–2992.