A Semi-Supervised Approach Based on k-Nearest Neighbor

Zhiliang Liu
School of Automation Engineering, University of Electronic Science and Technology of China, Chengdu, P. R. China
Email: zhiliang_liu@uestc.edu.cn

Xiaomin Zhao
Department of Mechanical Engineering, University of Alberta, Edmonton, Canada
Email: xiaomin1@ualberta.ca

Jianxiao Zou and Hongbing Xu
School of Automation Engineering, University of Electronic Science and Technology of China, Chengdu, P. R. China
Email: {jxzou, hbxu}@uestc.edu.cn

***Abstract*—k-nearest neighbor (k-NN) needs sufficient labeled instances for training in order to get robust performance; otherwise, performance deterioration occurs if training sets are small. In this paper, a novel algorithm, namely ordinal semi-supervised k-NN, is proposed to handle cases with a few labeled training instances. This algorithm consists of two parts: instance ranking and semi-supervised learning (SSL). Using SSL, the performance of k-NN with small training sets can be improved because SSL enlarges the training set by including unlabeled instances with their predicted labels. Instance ranking is used to pick up the unlabeled instances that are included to the training set. It gives priority to the unlabeled instances that are closer to class boundaries because they are more likely to be correctly predicted (these instances are called high confidence prediction instances). Thus, SSL benefits from high confidence prediction instances if they are added into the training set early. Experimental results demonstrate that the proposed algorithm outperforms the semi-supervised k-NN and the conventional k-NN algorithms if the training set is small.**

***Index Terms*—Instance ranking; semi-supervised learning; k-nearest neighbor; weighted mean**

NOTATION

U	the given dataset, including the labeled dataset U_l and the unlabeled dataset U_u in this paper, $U=U_l \cup U_u$;
n	the number of features that describe an instance in U;
$\mathbf{x}$ (or $\mathbf{z}$)	an instance (a point) which is an n-dimensional column vector in U, and each dimension represents a feature;
N_j	the number of instances of the jth class in U_l;
N_l	the number of instances in U_l;
N_u	the number of instances in U_u;
L	the number of classes (labels) in U_l;
$\mathbf{x}_i^{(j)}$	the ith instance within the jth class;
k	the number of nearest neighbors used for prediction in k-NN;
$d(\mathbf{x},\mathbf{z})$	the Euclidean distance between two instances $\mathbf{x}$ and $\mathbf{z}$;
$NN_i(\mathbf{x})$	the ith nearest neighbor with respect to an unlabeled instance $\mathbf{x}$, i=1,2, …, k;
$\mathrm{CF}_{\min}$	the predefined threshold of confidence factor for high confidence prediction instances;
$class(\mathbf{x})$	returns the true class of instance $\mathbf{x}$, and a hat on it indicates a prediction class;
κ	kernel functions; in this paper, Gaussian radial basis function is adopted;
σ	the parameter, namely the width of features, in the Gaussian radial basis function;
R^*	the optimal Bayes probability of error;
K	the number of folds, that is a parameter in K-fold cross-validation.

Manuscript received May 5, 2012; revised August 23, 2012; accepted August 29, 2012.

I. INTRODUCTION

The nearest neighbor rule (NNR) was originally proposed by Fix and Hodges [1] for solving discrimination problems in 1951. The theory behind the NNR is that an unlabeled instance in the input space is likely to have the same class as its close instances. Under this rule, the nearest neighbor (1-NN) classifier, a supervised learning algorithm, identifies a class of an unlabeled instance by finding its closest training instance, taking note of its class, and predicting this class for the unlabeled instance. Instead of looking at one nearest instance only, the k-nearest neighbor (k-NN) classifier aims to find k closest instances in the training set and carries out the k nearest neighbor rule (kNNR), typically by the majority voting for the conventional k-NN, to make a prediction decision. In fact, the nearest neighbor classifier is a special case of k-NN if k=1. In references [1] and [2], it is concluded that k-NN reaches the optimal

Bayes probability of error R^* in such a manner that $k/N_l \to 0$ if underlying probability distributions are assumed on the training set. Later on Cover and Hart [3] proved that the asymptotic optimality of the 1-NN is in the following interval,

$$R^* \le R_{NN} \le R^*\left(2 - \frac{L}{L-1}R^*\right), \quad (1)$$

where R_{NN} is the 1-NN probability of error. They also pointed out that the nearest neighbor contains half classification information within an infinite training set. From Eq. (1), 1-NN reaches asymptotic optimality the same as the optimal Bayes probability of error if $R^*=0$. In such condition, class boundaries are clearly separated with each other, and no instances from different classes overlap. Considering joint distribution cases, k-NN was proposed to address the sub-optimality problem of NNR [4]. Mathematically, the algorithm of k-NN can be expressed as

$$\hat{class}(\mathbf{x}) = \underset{j=1,2,\ldots,L}{\arg\max} \sum_{i=1}^{k} f(\mathbf{x}, NN_i(\mathbf{x}))\delta(class(NN_i(\mathbf{x})), j), \quad (2)$$

where $\delta(i,j)$ is the Kronecker Delta Function defined as

$$\delta(i,j) = \begin{cases} 1, & i = j \\ 0, & otherwise \end{cases}, \quad (3)$$

which returns one if the two inputs are identical and zero; otherwise, $f(\mathbf{x},\mathbf{z})$ is a voting weight function that defines how much impact of a nearest neighbor **z** on **x**. For example, $f(\mathbf{x},\mathbf{z})$ is specified as $f(\mathbf{x},\mathbf{z})=\delta(class(\mathbf{x}), class(\mathbf{z}))$, which means that k-NN assigns an equal weight of one to each of the k nearest neighbors for the majority voting.

As a very popular classification technique, k-NN has been well studied in the past and widely applied to fields such as text processing [5] and fault diagnosis [6]. Reported studies on k-NN can be roughly grouped into three topics: on similarity measurement, on k nearest neighbor rule (kNNR), and on computation cost. Conventional metrics for similarity measurement of k-NN [7] include the Minkowski distance, Euclidean distance, Manhattan distance, and Chebyshev distance. A few new methods of similarity measurement have been proposed recently. Wang et al. [8] developed an adaptive distance metric to improve the performance of k-NN. Yu et al. [9] extended the kernel method into the k-NN and termed it as kernel distance metric. Lei et al. [6] proposed a feature-weighted method (TFSWT) that weights each feature differently to overcome the shortcomings of feature scaling sensitivity. The kNNR, the second topic on k-NN, is a strategy of prediction using k nearest neighbors. It is so critical to the performance of k-NN that attracts attentions from many researchers. The majority voting, one of the most popular and intuitive kNNRs, assigns an unlabeled instance a class that has a majority vote among k nearest neighbors. Dudani [10] proposed a distance-weighted function that weights the nearest neighbors closer to the unlabeled instance more heavily than others. The function is expressed as

$$f(\mathbf{x},\mathbf{z}) = \begin{cases} 1 & d(\mathbf{x}, NN_k(\mathbf{x})) = d(\mathbf{x}, NN_1(\mathbf{x})) \\ \dfrac{d(\mathbf{x}, NN_k(\mathbf{x})) - d(\mathbf{x},\mathbf{z})}{d(\mathbf{x}, NN_k(\mathbf{x})) - d(\mathbf{x}, NN_1(\mathbf{x}))} & otherwise \end{cases}. \quad (4)$$

Shepard [11] further utilized the exponential function on Euclidean distances to weight the voting for the nearest neighbors. Zeng et al. [12] proposed the pseudo NNR that combines the distance weighted kNNR with the local mean learning [13]. Because k-NN is an instance-based learner and time consuming, the third topic is also valuable to focus on. Instance selection is an option to address such a problem. Brighton and Mellish [14] reviewed instance selection methods over the past thirty years. A condensed NNR proposed by Hart [15] tends to retain training instances along the class boundaries and abandon instances that are inside a training set. By removing certain training instances, computation cost is improved, to a certain extent. Nowadays, the k-NN is particularly popular because poor run-time performance is alleviated with the today's available computational power [7].

The varieties of k-NN reviewed above seem robust if there are sufficient training data; but challenges such as classification deterioration, arise if a few training data available because labeled training data are sometimes difficult, expensive or time-consuming to obtain [4]. As a result, performance of classifiers deteriorates [12] [16]. The reason of deterioration is that small size training sets contain only partial information and are not sufficient to represent an intrinsic structure of classes. That is to say classifiers tend to be under-fitting with a few training data. The semi-supervised learning (SSL) provides a possible solution to this problem because it enlarges the training set with qualified unlabeled instances, namely high confidence prediction instances [17], and their predicted labels. Xie et al. [24] proposed a semi-supervised k-nearest neighbor classifier named De-YATSI. Lv [25] proposed a semi-supervised learning approach using local regularizer and unit circle class label representation. Wajeed and Adilakshmi [26] proposed a semi-supervised method to increase accuracy of k-NN for text classification. Giannakopoulos and Petridis [27] applied a semi-supervised version of Fisher discriminant analysis together with k-nearest neighbor to a diarization system.

In this paper, we utilize instance ranking to sequence unlabeled instances according a certain criterion. It is believed that the performance could be improved further by using an ordinal/sequential unlabeled set rather than an arbitrary one. The idea behind instance ranking is that instance closer to the class boundaries are more likely to be in the same class, that is, we have a higher confidence in classifying them correctly, and it is better to predict them as early as possible. These predicted instances (more likely to be correct) can then be utilized by SSL for further training. Simply speaking, instance ranking gives those important instances priority of being predicted first; by doing SSL sequentially rather than randomly, the training set is enlarged iteratively from the initial class boundaries and becomes more and more representative of the intrinsic structure of the given problem. The performance of classifiers, therefore, can be improved.

The rest of the paper is structured as follows. *Section* II detailed introduces instance ranking and how it works with SSL in the proposed ordinal semi-supervised k-NN. *Section* III illustrates the idea of the proposed algorithm using simulated datasets and validates the proposed algorithm using benchmark datasets; Conclusions are made in *Section* IV.

II. The Oridinal Semi-Supervised k-NN

As stated in Section I, k-NN suffers from poor classification performance if the training data is limited. To alleviate this problem, the ordinal semi-supervised k-NN is proposed, as shown in Fig. 1. The proposed method contains two parts: instance ranking and semi-supervised learning. In the first part, instance ranking, unlabeled instances are sorted by their distances to class boundaries that are estimated on the training set. The second part called semi-supervised learning continues to predict the top one unlabeled instance until all the unlabeled instances are classified. In another words, the instance of the top one ranked in the first part is classified by SSL and updates the training set with its predicted label if it is considered as the high confidence instance. The training set may increase iteratively, and the procedure terminates until all the unlabeled instances are processed.

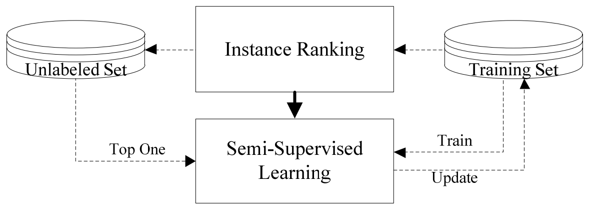


Figure 1. Scheme of the ordinal semi-supervised k-NN

A. Instance Ranking

Instance ranking aims to rank unlabeled instances by their relative distances to class boundaries. Weighted mean is a tool to represent the class boundary. That is, the distance between an unlabeled instance and its corresponding weighted mean of class *j* is an approximate measurement of the distance to the boundary of class *j*. Compared with the nearest neighbor, the weighted mean is more robust to detect class boundaries particularly if the nearest neighbor is an outlier, because it considers not only the nearest neighbor but also other instances in a class. The weighted mean was first used in [18] and [19] for the nonparametric weighted feature extraction.

Consider a given dataset U having L classes, and in U_l class j ($1\le j \le L$) has N_j instances. The weighted mean of an unlabeled instance $\mathbf{x}$ in U_u with respect to class j is defined as

$$M_j(\mathbf{x}) = \sum_{i=1}^{N_j} \omega(\mathbf{x},\mathbf{x}_i^{(j)})\mathbf{x}_i^{(j)} , \tag{5}$$

where $\omega(\mathbf{x},\mathbf{x}_i^{(j)})$ is the weight of $\mathbf{x}_i^{(j)}$ with respect to $\mathbf{x}$, and it is defined as

$$\omega(\mathbf{x},\mathbf{x}_i^{(j)}) = \frac{\kappa(\mathbf{x},\mathbf{x}_i^{(j)})}{\sum_{t=1}^{N_j}\kappa(\mathbf{x},\mathbf{x}_t^{(j)})} , \tag{6}$$

where $\kappa(\mathbf{x},\mathbf{z})$, the Gaussian radial basis function (Gaussian RBF), denotes an evaluation function for the distance between $\mathbf{x}$ and $\mathbf{z}$, and it is expressed as

$$\kappa(\mathbf{x},\mathbf{z}) = \exp(-\frac{d(\mathbf{x},\mathbf{z})^2}{2\sigma^2}) , \tag{7}$$

where $0< \sigma <\infty$, and $0< \kappa(\mathbf{x},\mathbf{z}) \le 1$.

An example in Fig. 2 illustrates how to calculate the weighted mean. Class 1 and class 2 are marked with red and green colors, respectively. Each class has four labeled instances for training, denoted as $\mathbf{x}_i^{(1)}$ and $\mathbf{x}_i^{(2)}$, respectively, where $1\le i \le 4$. $\mathbf{x}$ is one unlabeled instance. First, we calculate distance $d(\mathbf{x}, \mathbf{x}_i^{(1)})$ between $\mathbf{x}$ and every labeled instance $\mathbf{x}_i^{(1)}$, where i = 1,2,3,4. Second, the weight $\omega(\mathbf{x}, \mathbf{x}_i^{(1)})$ is available after evaluating distances between $\mathbf{x}$ and every instance in class 1 by $\kappa(\mathbf{x},\mathbf{z})$ with a predefined σ. At last, the weighted mean $M_1(\mathbf{x})$ is computed according to Eq. (5). From Fig. 2, we notice that the distance between $\mathbf{x}$ and $M_1(\mathbf{x})$ is kind of orthogonal distance to the corresponding class boundary. The weighted mean of $\mathbf{x}$ with respect to class 2, $M_2(\mathbf{x})$, is obtained following the same procedure.

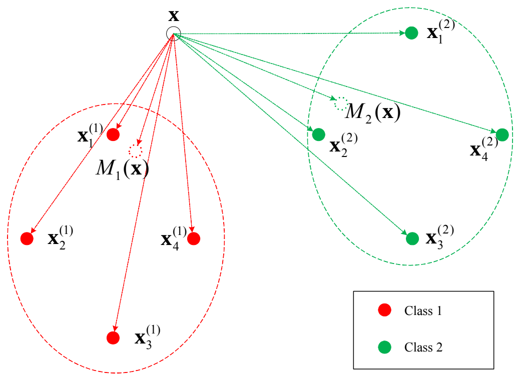


Figure 2. An example of how to calculate the weighted mean in a two-class case

Sigma, σ, in the Gaussian RBF is important to the weighted mean because it changes weight distribution among instances. Exceptionally, the weighted mean becomes the arithmetic mean, that is,

$$M_j(\mathbf{x}) = \frac{1}{N_j}\sum_{i=1}^{N_j}\mathbf{x}_i^{(j)} , \tag{8}$$

if $\sigma\to\infty$. It is because $\kappa(\mathbf{x},\mathbf{z}) \to 1$ if $\sigma\to\infty$, and then $\omega(\mathbf{x}, \mathbf{x}_i^{(j)})$ tends to be identical of $1/N_j$ for $1 \le i \le N_j$. Let us take an example to illustrate the impact of sigma in the weighted mean. Suppose $\mathbf{x}_1^{(1)}$=(2,3), $\mathbf{x}_2^{(1)}$=(1,2), $\mathbf{x}_3^{(1)}$=(2,1), $\mathbf{x}_4^{(1)}$=(3,2), $\mathbf{x}_1^{(2)}$=(5,4), $\mathbf{x}_2^{(2)}$=(4,3), $\mathbf{x}_3^{(2)}$=(5,2), $\mathbf{x}_4^{(2)}$=(6,3), and $\mathbf{x}$=(2.5,4) in Fig. 2. The weight distribution and Euclidean distances between an unlabeled instance $\mathbf{x}$ and eight training instances are shown in Fig. 3. Three different values, sigma=0.5, 1.0, 5.0, are considered in computing the weight distribution of $\mathbf{x}$. No matter what the value of sigma is, the nearest neighbor is always weighted heaviest because $\kappa(\mathbf{x},\mathbf{z})$ is monotonic with respect to $d(\mathbf{x},\mathbf{z})$, such as $\mathbf{x}_1^{(1)}$ in class 1 and $\mathbf{x}_2^{(2)}$ in class 2. $\kappa(\mathbf{x},\mathbf{z})$ with a smaller sigma weights the nearest neighbor more than that with a bigger sigma, such as $\mathbf{x}_1^{(1)}$ with sigma = 0.5 and sigma = 5.0. In a word, a small sigma comes with discriminant weights among a class while a big sigma tends to evaluate instances in a class identically.

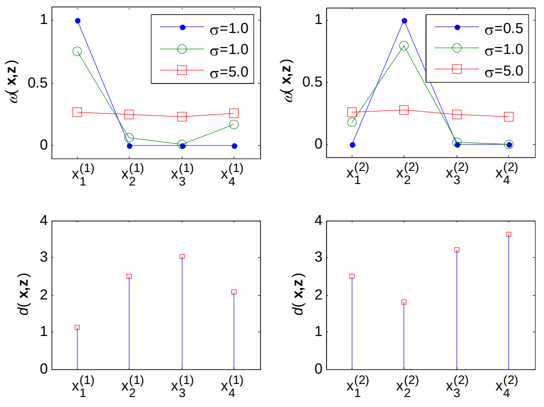


Figure 3. The weight distributes of instances with respect to three different sigma: 0.5, 1.0, and 5.0

A distance factor (DF) is then proposed to represent the relative closeness of the unlabeled instance **x** to its closest class boundary among all the boundaries measured by the weighted means. It is defined as

$$\mathrm{DF}(\mathbf{x}) = \min_{j=1,2,\cdots L} \frac{d(\mathbf{x}, M_j(\mathbf{x}))}{\sum_{t=1}^{L} d(\mathbf{x}, M_t(\mathbf{x}))}, \qquad (9)$$

where 0≤DF≤1/*L*. There are *L* distances between **x** and *L* class boundaries. The distance factor is the minimum one after normalized by the summation of *L* distances with respect to **x**. So far we have a measurement of DF that indicates how close of an instance to its nearest class boundary. A smaller DF implies that the instance is closer to its nearest class boundary. If all DFs of unlabeled instances are ready, they are sorted in ascending order and ready to be used in the next step.

B. Semi-supervised Learning

Semi-supervised learning is a framework of algorithms proposed to improve the performance of supervised algorithms through the use of both labeled and unlabeled data. Semi-supervised learning leads to be developed mainly because in some cases with limited labeled data, it needs great effort and cost to label large amount of training data [20]. Generally, semi-supervised learning may be achieved by either co-training or self-training [21]. Co-training assumes that features can be divided into two independent sets, and each subset is sufficient to train a classifier. Two classifiers are trained on two views separately so that classifiers can help with each other by adding one's most confidence prediction instance into the other's training set. Assumptions of co-training are too strict to be satisfied in practice particularly in cases with a few training data. Moreover, co-training is complicated because more than one classifier is involved in classification. On the other hand, self-training is simple to achieve because it uses only one classifier to train and select high confidence instances in the unlabeled set. In this work, we implement SSL by the self-training approach. The only classifier used is k-NN. Following the principles of self-training, we define a confidence measure in prediction**,** confidence factor (CF), that is mathematically expressed as

$$\mathrm{CF}(\mathbf{x}) = \frac{\sum_{i=1}^{k} \delta\left(class(NN_i(\mathbf{x})), \hat{class}(\mathbf{x}) \right) d(\mathbf{x}, NN_i(\mathbf{x}))}{\sum_{i=1}^{k} d(\mathbf{x}, NN_i(\mathbf{x}))}, \qquad (10)$$

where $\hat{class}(\mathbf{x})$ is the label predicted by Eq. (2) using the majority voting; 0≤CF≤1. In other words, CF is a summation of the distances between the unlabeled instance (**x**) and those same-class neighbors within the group of the k nearest divided by a summation of distances between the unlabeled instance and all those k nearest neighbors whether in the same class or not. If CF has a large value, there is a strong evidence to trust the predicted class of the unlabeled instance.

The rule of self-learning is that an unlabeled instance is considered as a high confidence instance and added to the training set if its CF is greater than or equal to a predefined threshold denoted by $\mathrm{CF}_{\min}$. Once the high confidence instances are added into the training set, it is considered the same as the initial training instances. By involving more unlabeled instances and their predicted labels, the training set is updated iteratively, and class boundaries are thus updated as well. In the next iteration, instance ranking is repeated on the current unlabeled and training sets. k-NN makes a prediction for the next unlabeled instance of top ranking on the latest training set.

C. Algorithm Summary

TABLE I.
PSEUDO CODES OF THE ORDINAL SEMI-SUPERVISED K-NN

```
INPUT: Specify the values of parameters: $\sigma$, $CF_{min}$, and k;
OUTPUT: The predicted labels of unlabeled instances;
MAIN ALGORITHM
    For index of unlabeled instances $i = 1, 2, \ldots, N_u$ % $N_u$ refers to the number of the initial $U_u$
        Module 1;
        Module 2;
    End

Module 1: Instance Ranking
1.1 For class index $j = 1, 2, \ldots, L$
        1.1.1 Calculate $d(\mathbf{x}, \mathbf{x}_i^{(j)})$, the Euclidean distance between each unlabeled instance x in $U_u$ and each instance of class j in $U_l$;
        1.1.2 Calculate the corresponding weights $\omega(\mathbf{x}, \mathbf{x}_i^{(j)})$;
        1.1.3 Calculate the weighted mean for each instance x in $U_u$ with respect to class j;
    End
1.2 Compute the distance factor DF(x) by Eq. (9);
1.3 Sort all the DF values in ascending order and return the top ranking instance possesses the smallest DF;

Module 2: Semi-Supervised Learning
2.1 Predict the class of current instance x of top ranking in Module 1 by Eq. (2);
2.2 Compute confidence factor of x, CF(x) using Eq. (10);
2.3 If $CF(\mathbf{x}) \geq CF_{min}$
        2.3.1  $U_l = U_l \cup \mathbf{x}$;
2.4 $U_u = U_u \backslash \mathbf{x}$;            % remove x from $U_u$
```

The pseudo codes of the proposed method are summarized in TABLE I. Basically instance ranking and semi-supervised learning are implemented by *Module* 1 and *Module* 2, respectively. Instance ranking finds the top one instance of unlabeled according to the distance factor. Then the top one instance is classified by SSL and is determined whether it is added to the training set with its predicted label. The number of iterations is the number of unlabeled instances, that is, N_u. In the next section, we illustrate the algorithm visually on simulated datasets and compare it with other methods.

III. Experimental Results

In this section the proposed method is applied to two groups of datasets. The first group contains two simulated datasets of two-dimensional which are used to visually illustrate the mechanism of the ordinal semi-supervised k-NN. The second group contains four benchmark datasets which are used for validation of our algorithm in the practical application. Experimental results and discussions are given below.

A. Experiment 1: The Simulated Datasets

Two simulated datasets are artificially generated to visually illustrate the mechanism of the proposed method. One is a two-class and Gaussian distributed dataset, where the mean of two classes are $[2, 5]^T$ and $[6, 5]^T$, respectively, and the standard deviations of two classes are both $[1, 2]^T$. The other is the well-known two-moon dataset [22] which is non-Gaussian distributed. We use two half circles to simulate the dataset: one is with a center of $[6, 5.5]^T$ and radius of 5, the other is with a center of $[11, 6.5]^T$ and radius of 5. Two datasets are summarized in TABLE II.

TABLE II.
SUMMARY OF SIMULATED DATASETS

No.	Dataset	Classes	Instances	Features
1	Simulated dataset #1	2	400	2
2	Simulated dataset #2	2	102	2

Fig. 4 shows six intermediate iterations on the simulated dataset #1: 0th iteration, 80th iteration, 160th iteration, 240th iteration, 320th iteration, and 398th iteration (the last iteration). White curves are decision boundaries of the current iteration. Colors indicate the class of instances, for example, red is of class 1 while green is of class 2. Instances with big size dots are used as training instances while others with small dots are considered as unlabeled instances. A circle around a dot means the instance is predicted already, and the color of circles indicates the predicted class. At 0th iteration, two instances are in the training set: one red instance is for class 1, and one green instance is for class 2. The rest 398 (400-2) instances are considered to be unlabeled. 1-NN is the classifier used here. According to the class boundary in Fig. 4 (a), there are lots of instances wrongly predicted since the decision boundary of 1-NN is exactly in the middle of the two training instances. After 80 iterations, in Fig. 4 (b), the decision boundary is slightly changed, and prediction is conducted to instances along class boundaries. In Fig. 4 sequential prediction continues until 398th iteration. The decision boundary evolves by aid of the ordinal semi-supervised k-NN and unlabeled instances. It is eventually representative enough for the structure of the simulated dataset #1 in the last iteration shown in Fig. 4 (f).

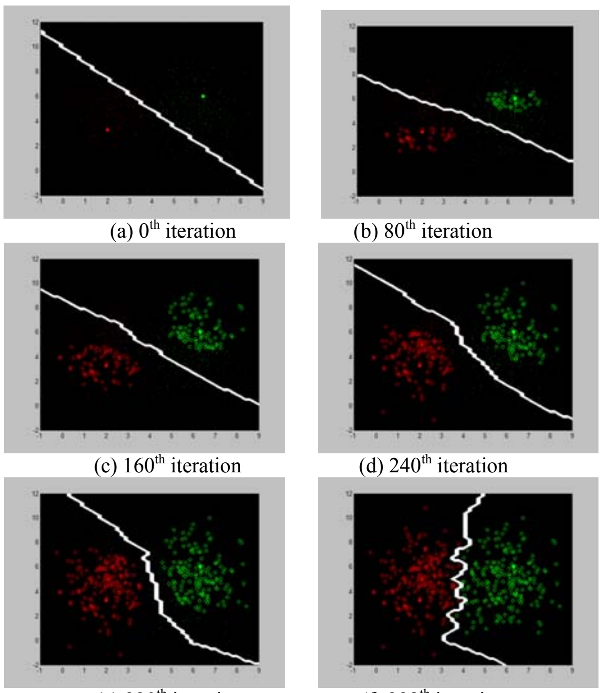
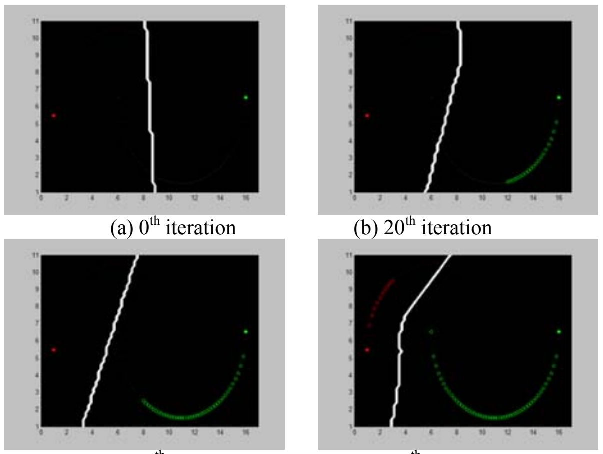
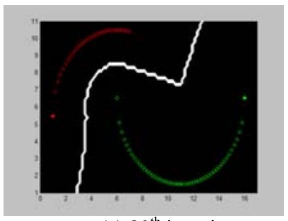

(a) 0th iteration (b) 80th iteration
(c) 160th iteration (d) 240th iteration
(e) 320th iteration (f) 398th iteration

Figure 4. Intermediate boundary change on the simulated dataset #1, classifier: 1-NN

Fig. 5 shows intermediate results on the simulated dataset #2 that is the two-moon dataset. Unlike the simulated dataset #1, distributions in this dataset are non-Gaussian. The same as the case of the simulated dataset #1, only two training instances are selected in the 0th iteration, so the rest 100 (102-2) instances are used as the unlabeled instances. The decision boundary, similarly in the simulated dataset #1, continues to update itself by extracting and absorbing information from unlabeled sets. At the last iteration, the 100th iteration shown in Fig. 5 (f), the decision boundary could be good enough for the following prediction. That is, a good model is generated then.


(a) 0th iteration (b) 20th iteration
(c) 40th iteration (d) 60th iteration


(e) 80th iteration

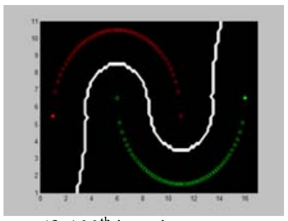

(f) 100th iteration

Figure 5. Intermediate boundary change on the simulated dataset #2, classifier: 1-NN

From two cases studied above, it demonstrates that the ordinal semi-supervised k-NN has the ability to improve the performance of k-NN by using unlabeled sets if a few training instances are available, such as in the simulated dataset #2 only one training instance available for one class. Its effectiveness is proved not only on Gaussian distributed cases, such as the simulated dataset #1, but also on non-Gaussian distributed cases, such as the simulated dataset #2. Before applying our algorithm, assumptions [22] are made on it like other techniques of SSL, and they are:

(1) *The Smooth Assumption* (SA). It states instances in high density region should have similar labels, that is, labels should change in low density regions;

(2) *The Cluster Assumption* (CA). It says two instances from the same cluster should have similar labels;

(3) *The Manifold Assumption* (MA). It presumes data lies roughly on a low-dimensional manifold.

We demonstrate the effectiveness of the proposed method in this section, and in next section the proposed method is applied to practical benchmark datasets.

B. *Experiment 2: The Benchmark Datasets*

The ordinal semi-supervised k-NN, namely *method* 1, is compared with two other varieties of k-NN: the semi-supervised k-NN and the conventional k-NN, namely *method* 2 and *method* 3, respectively. The difference between method 1 and method 2 is that method 1 is implemented with instance ranking while no instance ranking, or random ranking, is implemented in method 2. The k-NN classifier is implemented by *knnclassify* from the Bioinformatics Toolbox of MATLAB. The parameters in the three methods are predefined as follows: σ=1, k=1, CF_{min}=1. These three methods are compared with each other on four benchmark datasets downloaded from the University of California Irvine (UCI) repository [23]. They are summarized in TABLE III.

TABLE III.
SUMMARY OF BENCHMARK DATASETS

No.	Dataset	Classes	Instances	Features
1	Vehicle	4	846	18
2	Ionosphere	2	351	34
3	Parkinson	2	195	22
4	Wine	3	178	13

We split each dataset in TABLE III into two subsets because of limited instances available in the datasets, one is considered as a labeled set for training; the other, although it is labeled, is considered as an unlabeled set for SSL. In method 3, unlabeled sets are directly used to test method 3. In method 2 and method 3, after applying SSL on the unlabeled set, unlabeled sets, considered as test sets, are re-used to test these two methods. The K-fold cross-validation technique is used to do data partitioning. The capital K is a specified integer from 2 to 10. For each K value, one fold is used for unlabeled sets and the other K-1 folds are used for initial training sets, totally there are K combinations of selecting one out of K. We run the computer programs for each combination, obtain the classification accuracy of each run, and then average them. Classification accuracy is the only measurement for evaluating performance of methods. It is calculated with the test set by 0-1 loss function. That is to say, classification accuracy is the ratio between the total number of truly predicted instances in the test set and the total number of the test unlabeled set. A labeled ratio is defined by the size of training set divided by all instances, i.e. (K−1)/K in this case. Since K ranges from 2 to 10, the ratio ranges from 1/2 to 9/10, namely, 1/2, 2/3, 3/4, …, 9/10. Next, we do the same procedure as above except considering one fold as the training set and the rest K−1 folds as the unlabeled set. The labeled ratio is therefore calculated by 1/K and has a range from 1/10 to 1/2, namely, 1/10, 1/9, 1/8, …, 1/2. If all calculations are completed, we have 17 average values of classification accuracy with respect to 17 ratios from 1/10 to 9/10.

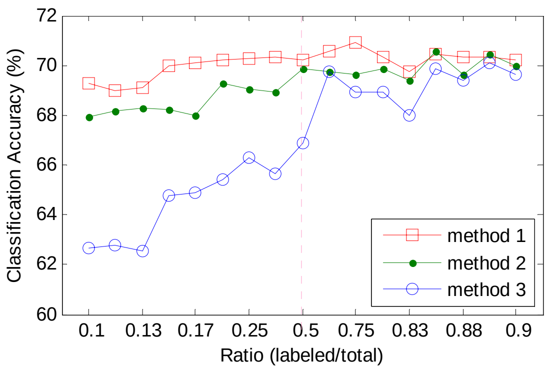


Figure 6. Classification accuracy on the Vehicle dataset

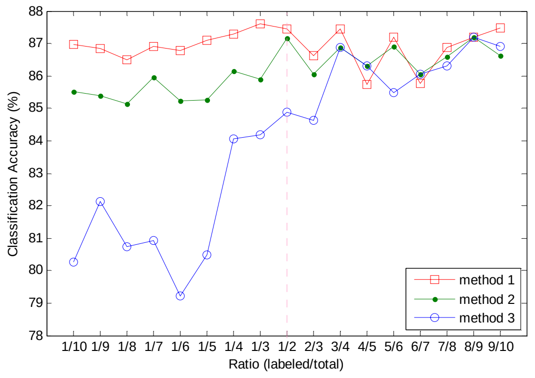


Figure 7. Classification accuracy on the Ionosphere dataset

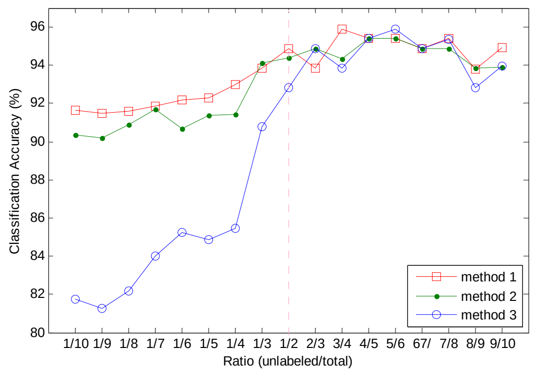


Figure 8. Classification accuracy on the Parkinson dataset

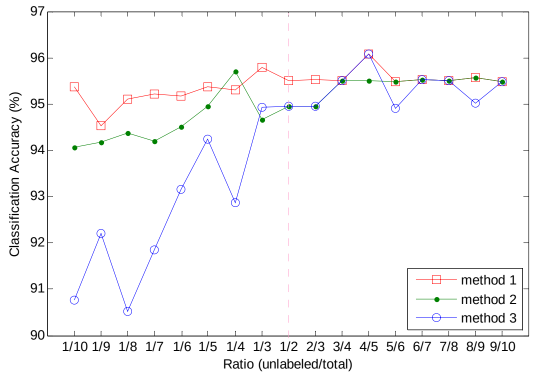


Figure 9. Classification accuracy on the Wine dataset

Results are provided in Fig. 6 to Fig. 9 in terms of classification accuracy of three methods at different ratios. We need to compare the results on the case of the labeled ratio>1/2 (large training datasets) and the case of the ratio≤1/2 (small training datasets). From Fig. 6 to Fig. 9, it can be said that method 1 outperforms other two methods if the ratio ranges from 1/10 to 1/2 (i.e. small training sets). The conventional k-NN is sensitive to the ratio and deteriorates a lot as mentioned in section I if the ratio is smaller than 1/2. With a small ratio (≤1/2), the training sets are insufficient and are not capable of providing enough knowledge for a robust performance in classifying the test set. Comparing with the conventional k-NN, use of the semi-supervised k-NN improves the classification accuracy. In general, classification accuracy of the semi-supervised k-NN, however, is inferior in comparison with the proposed method, as the latter benefited from the use of a sequential unlabeled set. If the ratio>1/2, classification accuracy of three methods is quite comparable. This clearly shows that the proposed method is advantageous for small training sets and not for large training sets.

TABLE IV provides two statistical measures of classification accuracy for insufficient cases (i.e. ratio≤1/2), namely the mean (MEAN) and the standard deviation (STD). Both the mean and the standard deviation are calculated over the nine averaged values of classification accuracy corresponding to nine ratios less or equal to 0.5. It can be seen that the proposed method possesses not only the highest mean of the accuracy but also the smallest standard deviation on any of the four datasets, which also demonstrates the effectiveness of the proposed method in insufficient cases.

TABLE IV.
SUMMARY OF EXPERIMENT RESULTS

Dataset	Method 1		Method 2		Method 3	
	MEAN	STD	MEAN	STD	MEAN	STD
Vehicle	*69.83*	*0.55*	68.63	0.67	64.66	1.63
Ionosphere	*87.05*	*0.35*	85.74	0.64	81.87	2.03
Parkinson	*92.53*	*1.17*	91.69	1.5	85.38	3.98
Wine	*95.26*	*0.34*	94.62	0.52	92.83	1.66

Note: Italic and shading text denotes either the largest value of mean or the smallest value of standard deviation.

IV. CONCLUSIONS

In this work, we propose a classification method named the ordinal semi-supervised k-NN to deal with insufficient training instances. The proposed method is a nonparametric classification approach and includes two parts, namely instance ranking and semi-supervised learning. By instance ranking, a sequential unlabeled set is obtained. By semi-supervised learning, the top ranking instance is classified and included into the training set with its predicted label if it is considered as the high confidence instance. The proposed method works for both Gaussian and non-Gaussian distributed cases. Experimental results on four benchmark datasets demonstrate that the proposed method outperforms the conventional k-NN and the semi-supervised k-NN especially if the training instances are insufficient (ratio≤1/2), and it is comparable with the other two methods if training sets are sufficient (ratio>1/2). But the proposed method requires more computational time than the conventional k-NN because the training set continuously increases while it is fixed in the conventional k-NN.

REFERENCES

[1] E. Fix, and J. L. Hodges, Jr., “Discriminatory analysis. nonparametric discrimination: consistency properties,” *U.S. Air Force Sch. Aviation Medicine, Randolf Field, Tex.*, Project 21-49-004, Contract AF 41 (128)-31, Rep.4, 1951.
[2] E. Fix, and J. L. Hodges Jr., “Discriminatory analysis: small sample performance,” *U.S. Air Force Sch. Aviation Medicine*, Randolph Field, Tex., Project 21-49-004, Rept. 11, 1952.
[3] T. M. Cover, and P. E. Hart, “Nearest neighbor pattern classification,” *IEEE Transaction on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.
[4] Y. Wang, X. Xu, H. Zhao, and Z. Hua, “Semi-supervised learning based on nearestneighbor rule and cut edges,” *Knowledge-Based Systems*, vol. 23, no. 6, pp. 547-554, August 2010.
[5] Reynaldo Gil-García and Aurora Pons-Porrata, “A new nearest neighbor rule for text categorization,” *Lecture Notes in Computer Science*, vol. 4225, pp. 814-823, 2006.
[6] Y. Lei, and M. J. Zuo, “Gear crack level identification based on weighted k nearest neighbour classification algorithm”. *Mechanical Systems and Signal Processing*, vol. 23, no. 5, pp. 1535-1547, July 2009.
[7] P. Cunningham, and S. J. Delany, “k-nearest neighbor classifiers,” *Technical Report UCD-CSI-2007-4*, March 2007.

[8] J. Wang, P. Neskovic, and L. N. Cooper, “Improving nearest neighbor rule with a simple adaptive distance measure,” *Pattern Recognition Letters*, vol. 28, no. 2, pp. 207-213, Janurary 2007.

[9] K. Yu, L. Ji, and X. Zhang, “Kernel nearest-neighbor algorithm,” *Neural Processing Letter*, vol. 15, no. 2, pp. 147-156, 2002.

[10] S. A. Dudani, “The distance-weighted k-nearest neighbor rule,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 6, no. 4, pp. 325–327, April 1976.

[11] R. N. Shepard, “Toward a universal law of generalization for psychological science,” *Science*, vol. 237, no. 4820, pp.1317–1323, September 1987.

[12] Y. Zeng, Y. Yang, and L. Zhao, “Pseudo nearest neighbor rule for pattern classification,” *Expert Systems with Applications*, vol. 36, no. 2, pp. 3587-3595, March 2009.

[13] Y. Mitani, and Y. Hamamoto, “A local mean-based nonparametric classifier,” *Pattern Recognition Letters*, vol. 27, no. 10, pp. 1151–1159, July 2006.

[14] H. Brighton, and C. Mellish, “Advances in instance selection for instance-based learning algorithms,” *Data Mining and Knowledge Discovery*, vol. 6, no. 2, pp. 153-172, April 2002.

[15] P. E. Hart, “The condensed nearest neighbor rule,” *IEEE Transactions on Information Theory* , vol. 16, no. 3, pp. 515-516, May 1968.

[16] A. M. Martinez, and A. C. Kak, “PCA versus LDA,” *IEEE Transactions on Pattern Analysis and Machine Intelligence* , vol.23, no.2, pp.228-233, Feb. 2001.

[17] X. Zhu, “Semi-Supervised Learning Literature Survey,” Computer Sciences TR 1530. University of Wisconsin–Madison, July 2008.

[18] B. C. Kuo, and D. A. Landgrebe, “Nonparametric weighted feature extraction for classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 42, no. 5, pp.1096-1104, May 2004.

[19] B. C. Kuo, C. H. Li, and J. M. Yang, “Kernel nonparametric weighted feature extraction for Hyperspectral Image Classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 4, pp. 1139-1155, April 2009.

[20] A. Albalate, and W. Minker, “Semi-supervised and unsupervised machine learning: novel strategies”, John Wiley & Sons Inc, Hoboken, USA, 2011.

[21] D. Guan, W. Yuan, Y. K. Lee, and S. Lee, “Nearest neighbor editing aided by unlabeled data,” *Information Sciences*, vol. 179, no. 13, pp. 2273-2282, June 2009.

[22] Z. Bodó, and Z. Minier, “On Supervised and Semi-supervised k-Nearest Neighbor Algorithms,” *Proceedings of the 7th Joint Conference on Mathematics and Computer Science,* vol. LIII, pp. 79–92. Studia Universitatis Babe¸s-Bolyai, Series Informatica, 2008.

[23] A. Frank, and A. Asuncion. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2010.

[24] Y. Xie, Y. Jiang, and M. Tang, “A semi-supervised k-nearest neighbor algorithm based on data editing,” Proceedings of Chinese control and decision conference, 2011.

[25] J. Lv, “Semi-supervised learning using local regularizer and unit circle class label representation,” *Journal of Software*, vol. 7, no. 5, pp. 951-958, 2012.

[26] M. A. Wajeed, and T. Adilakshmi, “Semi-supervised text classification using henhanced KNN algorithm,” Proceedings of World Congress on Information and Communication Technologies, 2011.

[27] T. Giannakopoulos, and S. Petridis, “Fisher linear semi-Discriminant analysis for speaker diarization,” *IEEE Transaction on Audio, Speech, and Laguage Processing*, vol. 20, no. 7, pp. 1913-1922, 2012.

Zhiliang Liu received the B.S. degree in school of electrical and information engineering, Southwest University for Nationalities, Chengdu, in 2006. He visited University of Alberta as a visiting scholar from 2009 to 2011. He is currently a Ph.D candidate in the school of automation enigeering, University of Electronic Science and Technology of China, Chengdu. His research interests mainly include pattern recognition, data mining, fault diagnosis and prognosis.