# Efficient kNN Classification With Different Numbers of Nearest Neighbors

Shichao Zhang, *Senior Member, IEEE*, Xuelong Li, *Fellow, IEEE*, Ming Zong, Xiaofeng Zhu*, and Ruili Wang

***Abstract*— $k$ nearest neighbor (kNN) method is a popular classification method in data mining and statistics because of its simple implementation and significant classification performance. However, it is impractical for traditional kNN methods to assign a fixed $k$ value (even though set by experts) to all test samples. Previous solutions assign different $k$ values to different test samples by the cross validation method but are usually time-consuming. This paper proposes a kTree method to learn different optimal $k$ values for different test/new samples, by involving a training stage in the kNN classification. Specifically, in the training stage, kTree method first learns optimal $k$ values for all training samples by a new sparse reconstruction model, and then constructs a decision tree (namely, kTree) using training samples and the learned optimal $k$ values. In the test stage, the kTree fast outputs the optimal $k$ value for each test sample, and then, the kNN classification can be conducted using the learned optimal $k$ value and all training samples. As a result, the proposed kTree method has a similar running cost but higher classification accuracy, compared with traditional kNN methods, which assign a fixed $k$ value to all test samples. Moreover, the proposed kTree method needs less running cost but achieves similar classification accuracy, compared with the newly kNN methods, which assign different $k$ values to different test samples. This paper further proposes an improvement version of kTree method (namely, k*Tree method) to speed its test stage by extra storing the information of the training samples in the leaf nodes of kTree, such as the training samples located in the leaf nodes, their kNNs, and the nearest neighbor of these kNNs. We call the resulting decision tree as k*Tree, which enables to conduct kNN classification using a subset of the training samples in the leaf nodes rather than all training samples used in the newly kNN methods. This actually reduces running cost of test stage. Finally, the experimental results on 20 real data sets showed that our proposed methods (i.e., kTree and k*Tree) are much more efficient than the compared methods in terms of classification tasks.**

***Index Terms*— Decision tree, $k$ nearest neighbor (kNN) classification, sparse coding.**

## I. Introduction

DIFFERENT from model-based methods which first learn models from training samples and then predict test samples with the learned model [1]–[6], the model-free $k$ nearest neighbors (kNNs) method does not have training stage and conducts classification tasks by first calculating the distance between the test sample and all training samples to obtain its nearest neighbors and then conducting kNN classification[1] (which assigns the test samples with labels by the majority rule on the labels of selected nearest neighbors). Because of its simple implementation and significant classification performance, kNN method is a very popular method in data mining and statistics and thus was voted as one of top ten data mining algorithms [7]–[13].

Previous kNN methods include: 1) assigning an optimal $k$ value with a fixed expert-predefined value for all test samples [14]–[19] and 2) assigning different optimal $k$ values for different test samples [18], [20], [21]. For example, Lall and Sharma [19] indicated that the fixed optimal-$k$-value for all test samples should be $k = \sqrt{n}$ (where $n > 100$ and $n$ is the number of training samples), while Zhu *et al.* [21] proposed to select different optimal $k$ values for test samples via tenfold cross validation method. However, the traditional kNN method, which assigns fixed kNNs to all test samples (fixed kNN methods for short), has been shown to be impractical in real applications. As a consequence, setting an optimal-$k$-value for each test sample to conduct kNN classification (varied kNN methods for short) has been becoming a very interesting research topic in data mining and machine learning [22]–[29].

A lot of efforts have been focused on the varied kNN methods, which efficiently set different optimal-$k$-values to different samples [20], [30], [31]. For example, Li *et al.* [32] proposed to use different numbers of nearest neighbors

[1]In this paper, we call all kinds of methods, which use the labels of kNNs to classify the test samples, as kNN methods, such as fixed kNN methods and varied kNN methods. The kNN methods usually includes two steps, i.e., the setting of $k$ value and the kNN classification. The $k$ value can be set by experts in the fixed kNN methods while is learned (optimal $k$ value for short) in the varied kNN methods. The kNN classification represents to use the majority rule of kNNs of the test sample to assign it with labels.

Manuscript received February 18, 2016; revised July 28, 2016 and October 20, 2016; accepted February 20, 2017. This work was supported in part by the China "1000-Plan" National Distinguished Professorship, in part by the Nation Natural Science Foundation of China under Grant 61263035, Grant 61573270, and Grant 61672177, in part by the China 973 Program under Grant 2013CB329404, in part by the China Key Research Program under Grant 2016YFB1000905, in part by the Guangxi Natural Science Foundation under Grant 2015GXNSFCB139011, in part by the Research Fund of Guangxi Key Lab of MIMS (16-A-01-01 and 16-A-01-02), in part by the Guangxi High Institutions' Program of Introducing 100 High-Level Overseas Talents, in part by the Guangxi Collaborative Innovation Center of Multi-Source Information Integration and Intelligent Processing, and in part by the Guangxi "Bagui" Teams for Innovation and Research. *(Corresponding author: X. Zhu.)*

S. Zhang, M. Zong and X. Zhu are with the Guangxi Key Laboratory of MIMS, College of Computer Science and Information Technology, Guangxi Normal University, Guilin 541004, China (e-mail: zhangsc@mailbox.gxnu.edu.cn; 920902817@qq.com; xfzhu0011@hotmail.com).

X. Li is with the State Key Laboratory of Transient Optics and Photonics, Center for OPTical IMagery Analysis and Learning, Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an 710119, China (e-mail: xuelongli@opt.ac.cn).

R. Wang is with the Institute of Natural and Mathematical Sciences, Massey University, Auckland 4442, New Zealand (e-mail: r.wang@massey.ac.nz).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2017.2673241

for different categories and Sahigara *et al.* [33] proposed to employ the Monte Carlo validation method to select an optimal smoothing parameter $k$ fore each test sample. Recently, Cheng *et al.* [20] proposed a sparse-based kNN method to learn an optimal-$k$-value for each test sample and Zhang *et al.* [30] studied the kNN method by learning a suitable $k$ value for each test sample based on a reconstruction framework [34]. Previous varied kNN methods usually first learn an individual optimal-$k$-value for each test sample and then employ the traditional kNN classification (i.e., majority rule on $k$ training samples) to predict test samples by the learned optimal-$k$-value. However, either the process of learning an optimal-$k$-value for each test sample or the process of scanning all training samples for finding nearest neighbors of each test sample is time-consuming. Therefore, it is challenging for simultaneously addressing these issues of kNN method, i.e., optimal-$k$-values learning for different samples, time cost reduction, and performance improvement.

To address aforementioned issues of kNN methods, in this paper, we first propose a kTree[2] method for fast learning an optimal-$k$-value for each test sample, by adding a training stage into the traditional kNN method and thus outputting a training model, i.e., building a decision tree (namely, kTree) to predict the optimal-$k$-values for all test samples. Specifically, in the training stage, we first propose to reconstruct each training sample by all training samples via designing a sparse-based reconstruction model, which outputs an optimal-$k$-value for each training sample. We then construct a decision tree using training samples and their corresponding optimal-$k$-values, i.e., regarding the learned optimal-$k$-value of each training sample as the label. The training stage is offline and each leaf node stores an optimal-$k$-value in the constructed kTree. In the test stage, given a test sample, we first search for the constructed kTree (i.e., the learning model) from the root node to a leaf node, whose optimal-$k$-value is assigned to this test sample so that using traditional kNN classification to assign it with a label by the majority rule.

There are two distinguished differences between the previous kNN methods [20], [30] and our proposed kTree method. First, the previous kNN methods (e.g., fixed kNN methods and varied kNN methods) have no training stage, while our kTree method has a sparse-based training stage, whose time complexity is $O(n^2)$ (where $n$ is the sample size). It is noteworthy that the training stage of our kTree method is offline. Second, even though both the varied kNN methods and our proposed kTree method (which can be regarded as one of varied kNN methods) first search the optimal-$k$-values and then conduct traditional kNN classification to classify the test sample with the learned optimal-$k$-values, the previous methods need at least $O(n^2)$ time complexity to obtain the optimal-$k$-values due to involving a sparse-based learning process, while our kTree method only needs $O(log(d) + n)$ (where $d$ is the dimensions of the features) to do that via the learned model, i.e., the kTree. It is also noteworthy that the process of traditional fixed kNN method assigning a fixed $k$ value to all test samples needs at least $O(n^2d)$ via scanning all training samples for each test sample.

Although our kTree method enables to obtain optimal-$k$-values for test samples, it still needs to scan all training samples to conduct kNN classification, which is also a time-consuming process, i.e., at least $O(n^2d)$. We further extend our proposed kTree method to its improvement version (namely, k*Tree method) to speed test stage, by only storing extra information of training samples in the left nodes, such as the training samples, their kNNs, and the nearest neighbors of these nearest neighbors. We call the resulting decision tree as k*Tree. That is, there is one difference between the kTree method and the k*Tree method in the training stage, i.e., the optimal-$k$-values in the leaf nodes for the kTree, while the optimal-$k$-values and the information of training samples for the k*Tree. In the test stage, given a test sample, the k*Tree outputs its optimal-$k$-value and the information of its nearest neighbors in this leaf node, so the traditional kNN classification is conducted using the optimal-$k$-value and a subset of training samples in the left node (i.e., kNNs of the nearest neighbor of the test sample and their corresponding nearest neighbors of these kNNs). In this way, the number of used training samples $s$ is less than the sample size $n$, i.e., $s \ll n$, thus reducing the running cost of test stage.

The rest of this paper is organized as follows. We briefly recall the state-of-the-art kNN methods and describe the detail of the proposed method, respectively, in Sections II and III. We then analyze our experimental results in Section IV and give our conclusion in Section V.

## II. Related Work

While kNN method enables to output remarkable performance and has been proved to approximately achieve to the error rate of Bayes optimization under very mild conditions, it has widely been applied to many kinds of applications, such as regression, classification, and missing value imputation [35]–[41]. The performance of kNN method can be affected by a lot of issues, such as the selection of the $k$ value and the selection of distance measures. To address these issues, a large amount of machine learning techniques have been developed.

Previous study of kNN method mainly focused on searching for an optimal-$k$-value for all test samples. For example, Zhang *et al.* [42] incorporated certainty factor measure to conduct kNN classification with a fixed $k$ value for all samples [43], while Song *et al.* [44] proposed to select a subset of most informative sample from neighborhoods. Vincent and Bengio [45] designed a $k$-local hyperplane distance and Wang *et al.* [24] defined a new similarity between two data points [46] for conducting kNN classification. Recently, Liu *et al.* [47] proposed an enhanced fuzzy kNN method to adaptively specify the optimal-$k$-values by the particle swarm optimization approach. Gou *et al.* [48]

[2]It is noteworthy that the terms (i.e., "kTree" and "k*Tree") used in this paper are different from the term "k-tree," which is widely used in the graph theory to represent an undirected graph formed by starting with a $(k + 1)$-vertex complete graph and then repeatedly adding vertices in such a way that each added vertex has exactly $k$ neighbors that, together, the $k + 1$ vertices form a clique. More detail on "k-tree" can be found in its Wikipedia page.

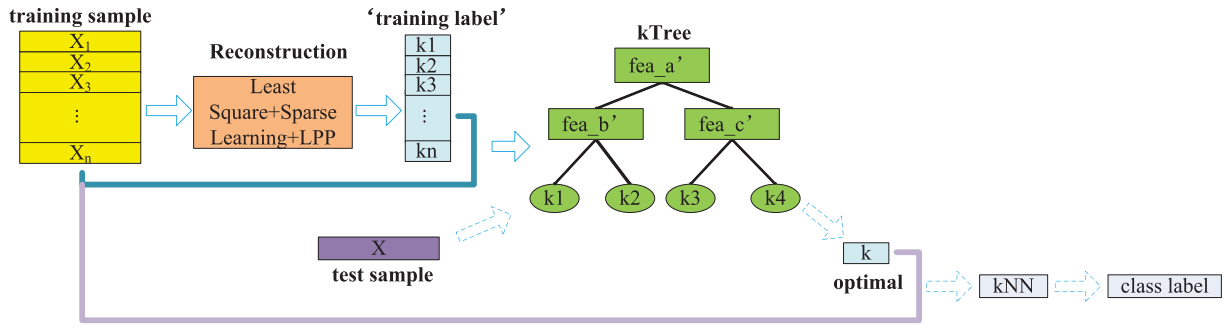


Fig. 1. Flowchart of the proposed kTree method.

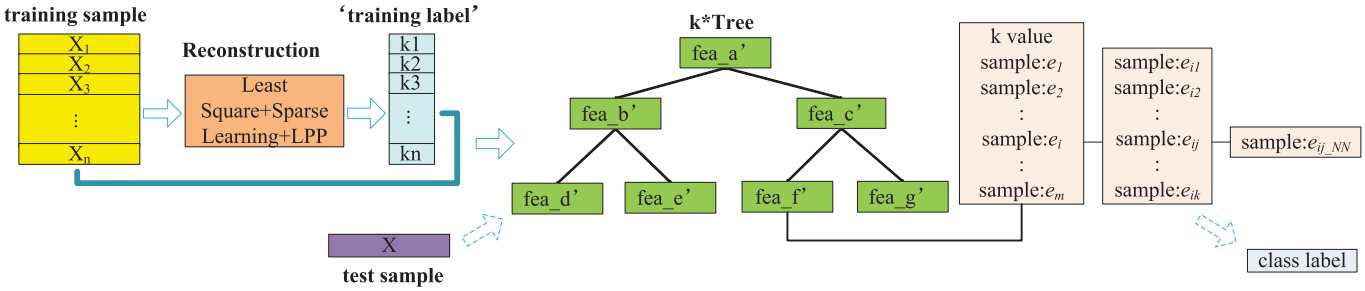


Fig. 2. Flowchart of the proposed k*Tree method.

developed a dual weighted voting scheme for kNN to overcome the sensitivity of the optimal-$k$-values. Premachandran and Kakarala [49] proposed to select a robust neighborhood using the consensus of multiple rounds of kNNs.

As it is impractical for applying for a fixed $k$ value for all test samples in data mining and machine learning, a number of efforts have been focused on designing different $k$ values for different samples. For example, Li *et al.* [16] demonstrated to use different numbers of nearest neighbors for different categories, rather than a fixed number across all categories. Góra and Wojna [15] proposed to combine two widely used empirical approaches, i.e., rule induction and instance-based learning, respectively, to learn the optimal-$k$-values. Guo *et al.* [50] proposed to construct a kNN model to automatically determine the optimal-$k$-value for each sample. Based on statistical confidence, Wang *et al.* [18] proposed to locally adjust the number of nearest neighbors. Manocha and Girolami [51] proposed a probabilistic nearest neighbor method for inferring the number of neighbors, i.e., optimal-$k$-values. Sun and Huang [52] also proposed an adaptive kNN algorithm, for each test sample, by setting the optimal-$k$-value as the optimal $k$ of its nearest neighbor in the training set.

Although the above methods solved the fixed $k$ value problem, their complexity is high for learning the optimal-$k$-value for each test sample.

## III. Approach

### A. Denotations

In this paper, we denote the matrix, the vector, and the scalar, respectively, as a boldface uppercase letter, a boldface lowercase letter, and a normal italic letter. For a matrix $\mathbf{X} = [x_{ij}]$, its $i$th row and $j$th column are denoted as $\mathbf{x}^i$ and $\mathbf{x}_j$, respectively. We denote the Frobenius norms of $\mathbf{X}$, $\ell_2-$norm, $\ell_1-$norm, and $\ell_{21}-$norm, respectively, as $||\mathbf{X}||_F = (\sum_j ||\mathbf{x}_j||_2^2)^{1/2}$ (matrix norms here are entrywise norm), $||\mathbf{X}||_2 = (\sum_i \sum_j |x_{ij}|^2)^{1/2}$, $||\mathbf{X}||_1 = \sum_i \sum_j |x_{ij}|$, and $||\mathbf{X}||_{21} = \sum_i (\sum_j \mathbf{x}_{ij}^2)^{1/2}$. We further denote the transpose operator, the trace operator, and the inverse of a matrix $\mathbf{X}$, respectively, as $\mathbf{X}^T$, $\mathrm{tr}(\mathbf{X})$ and $\mathbf{X}^{-1}$.

### B. Framework

In this section, we describe the proposed kTree method and k*Tree method in detail. Specifically, we first interpret the reconstruction process to learn the optimal-$k$-values for training samples in Section III-C. We then describe the kTree method and the k*Tree method, respectively, in Sections III-D and III-E. Figs. 1 and 2 illustrate the flowcharts of the proposed methods.

### C. Reconstruction

Denote by training samples $\mathbf{X} \in \mathbb{R}^{d\times n} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$, where $n$ and $d$, respectively, represent the number of training samples and features, in this section, we design to use training samples $\mathbf{X}$ to reconstruct themselves, i.e., reconstruct each training sample $\mathbf{x}_i$, with the goal that the distance between $\mathbf{X}\mathbf{w}_i$ and $\mathbf{x}_i$ (where $\mathbf{w}_i \in \mathbb{R}^n$ denotes the reconstruction coefficient matrix) is as small as possible. To do this, we use a least square loss function [53] as follows:

$$\min_{\mathbf{W}} \sum_{i=1}^{n} ||\mathbf{X}\mathbf{w}_i - \mathbf{x}_i||_2^2 = \min_{\mathbf{W}} ||\mathbf{X}\mathbf{W} - \mathbf{X}||_F^2 \tag{1}$$

where $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_n] \in \mathbb{R}^{n\times n}$ denotes the reconstruction coefficient or the correlations between training samples and themselves.

In real applications, an $\ell_2-$norm regularization term is often added into (1) for avoiding the issue of singularity of $\mathbf{X}^T\mathbf{X}$, that is

$$\min_{\mathbf{W}} ||\mathbf{XW}-\mathbf{X}||_F^2+\rho||\mathbf{W}||_2^2 \tag{2}$$

where $||\mathbf{W}||_2$ is an $\ell_2-$norm regularization term and $\rho$ is a tuning parameter. Usually, (2) is called ridge regression [23], [31] with a close solution $\mathbf{W}=(\mathbf{X}^T\mathbf{X}+\rho\mathbf{I})^{-1}\mathbf{X}^T\mathbf{X}$. However, (2) does not output sparse results. In this paper, we expect to generate the sparse reconstruction coefficient (i.e., $\mathbf{W}$) to select parts of training samples to represent each test sample. Following previous literature [34], [54], we employ the following sparse objective function:

$$\min_{\mathbf{W}} ||\mathbf{XW}-\mathbf{X}||_F^2+\rho_1||\mathbf{W}||_1, \quad \mathbf{W}\succeq 0 \tag{3}$$

where $||\mathbf{W}||_1$ is an $\ell_1-$norm regularization term [55] and $\mathbf{W}\succeq 0$ means that each element of $\mathbf{W}$ is nonnegative. Equation (3) has been proved to result in sparse $\mathbf{W}$ [2], [56] and is also called the least absolute shrinkage and selection operator [9], [53]. Moreover, (3) generates the elementwise sparsity, i.e., irregular sparsity in the elements of $\mathbf{W}$. The larger the value of $\rho_1$, the more sparse the $\mathbf{W}$.

Since we use training samples to reconstruct themselves, it is natural to expect that there exist relations among features or samples. Generally, if two features are highly related to each other, then it is reasonable to have the corresponding predictions also related [43], [53]. To this end, we devise a regularization term with the assumption that, if some features, e.g., $\mathbf{x}^i$ and $\mathbf{x}^j$ are involved in regressing, then the corresponding predictions are also related to each other. Thus, their corresponding predictions (i.e., $\mathbf{y}^i=\mathbf{x}^i\mathbf{W}$ and $\mathbf{y}^j=\mathbf{x}^j\mathbf{W}$) should have the same or similar relation. To utilize such relation, we penalize the loss function with the similarity between $\mathbf{y}^i$ and $\mathbf{y}^j$. Specifically, we impose the relation between two training samples in $\mathbf{X}$ to be reflected in the relation between their predictions by defining the following embedding function [43]:

$$\frac{1}{2}\sum_{i,j}^{d} s_{ij}\|\mathbf{x}^i\mathbf{W}-\mathbf{x}^j\mathbf{W}\|_2^2 \tag{4}$$

where $s_{ij}$ denotes an element in the feature similarity matrix $\mathbf{S}=[s_{ij}]\in\mathbb{R}^{d\times d}$ which encodes the relation between feature vectors. With respect to the similarity measure between vectors of $\mathbf{a}$ and $\mathbf{b}$, throughout this paper, we first use a radial basis function kernel as defined as follows:

$$f(\mathbf{a},\mathbf{b})=\exp\left(-\frac{\|\mathbf{a}-\mathbf{b}\|_2^2}{2\sigma^2}\right) \tag{5}$$

where $\sigma$ denotes a kernel width. As for the similarity matrix $\mathbf{S}$, we first construct a data adjacency graph by regarding each feature as a node and using kNNs along with a heat kernel function defined in (5) to compute the edge weights, i.e., similarities. For example, if a feature $\mathbf{x}^j$ is selected as one of the kNNs of a feature $\mathbf{x}^i$, then the similarity $s_{ij}$ between these two features or nodes is set to the value of $f(\mathbf{x}^i,\mathbf{x}^j)$; otherwise, their similarity is set to zero, i.e., $s_{ij}=0$ [43], [53], [57].

After simple mathematical transformation, we obtain the following regularization term:

$$R(\mathbf{W})=\mathrm{Tr}(\mathbf{W}^T\mathbf{X}^T\mathbf{LXW}) \tag{6}$$

where $\mathbf{L}\in\mathbb{R}^{d\times d}$ is a Laplacian matrix. We should note that the definition of $\mathbf{L}$ is different from [43] and [58], whose Laplacian matrix indicates the relational information between samples, while our Laplacian matrix indicates the relational information between features, which has been successfully used in many manifold learning methods [50], [57], [58]. Finally, we define the final objective function for the reconstruction process as follows:

$$\min_{\mathbf{W}} ||\mathbf{XW}-\mathbf{X}||_F^2+\rho_1||\mathbf{W}||_1+\rho_2 R(\mathbf{W}), \quad \mathbf{W}\succeq 0. \tag{7}$$

Equation (7) sequentially includes the least square loss function, the $\ell_1$-norm regularization term, the graph Laplacian regularization term, and the nonnegative constraint. According to [56] and [59], both the least square loss function and the graph Laplacian regularization term are convex and smooth, while either the $\ell_1$-norm regularization term or the nonnegative constraint is convex but not differentiable in all the range of $\mathbf{W}$ and thus being convex but nonsmooth. Therefore, our final objective function is convex but nonsmooth. According to [30] and [59], we can use an iterative method to optimize (7). As the objective function (7) is convex, the $\mathbf{W}$ satisfying (7) is a global optimum solution. Moreover, it will converge to the global optimum of the objective function (7). In this paper, we omit the proof, since it can be directly obtained according to [30, Th. 1].

After optimizing (7), we obtain the optimal solution $\mathbf{W}^*$, i.e., the weight matrix or the correlations between training samples and themselves. The element $w_{ij}$ of $\mathbf{W}^*$ denotes the correlation between the $i$th training sample and the $j$th training sample. The positive weight (i.e., $w_{ij}>0$) indicates that the $i$th training sample and the $j$th training sample are positively correlated, and the negative weight (i.e., $w_{ij}<0$) means that their correlation is negative. In particular, the zero weight (i.e., $w_{ij}=0$) means that there is no correlation between the $i$th training sample and the $j$th training sample. In other words, the $i$th training sample should not be used for predicting the $j$th training sample. Consequently, we only use those correlated training samples (i.e., the training samples with nonzero coefficient) to predict each training sample, rather than using all training samples.

To better understand the characteristics of the proposed reconstruction method, we assume the optimal solution $\mathbf{W}^*\in R^{5\times 5}$ as follows:

$$\mathbf{W}^*=\begin{bmatrix} 0.2 & 0.05 & 0 & 0 & 0 \\ 0 & 0.7 & 0 & 0.6 & 0.1 \\ 0 & 0.02 & 0.9 & 0 & 0 \\ 0.1 & 0.3 & 0 & 0.8 & 0 \\ 0.02 & 0 & 0.1 & 0 & 0.3 \end{bmatrix}.$$

In this example, we have five training samples. According to our proposed method, the values in the first column of $\mathbf{W}^*$ indicate the correlations between the first training sample and all five training samples. Due to that there are only three nonzero values in the first column, i.e., $w_{11}$, $w_{41}$,

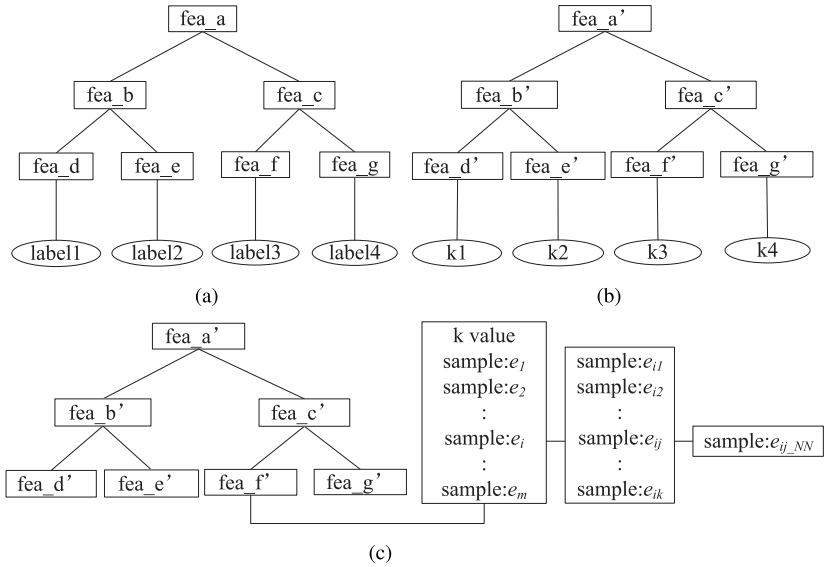


Fig. 3. Illustration of three different kinds of decision trees, i.e., ID3 method, kTree method, and k*Tree method, respectively. Note that these three decision trees are constructed with the same rule but storing different items in leaf nodes. Specifically, ID3 and kTree, respectively, store the class labels and the optimal-$k$-values, of training samples in the leaf node, while k*Tree stores the optimal-$k$-value (i.e., $k$ value), a subset of training samples (i.e., $\mathbf{e}_1, \ldots, \mathbf{e}_m$, the corresponding nearest neighbors of these training samples (e.g., the kNNs $\mathbf{e}_{i1}, \ldots \mathbf{e}_{ij}, \ldots, \mathbf{e}_{ik}$ of training sample $\mathbf{e}_i$, $i = 1, \ldots, m$), and the nearest neighbor of these nearest neighbors (e.g., the nearest neighbor $\mathbf{e}_{ij_{NN}}$ of $\mathbf{e}_{ij}$, $i = 1, \ldots, m$ and, $j = 1, \ldots, k$). (a) Decision tree. (b) kTree. (c) k*Tree.

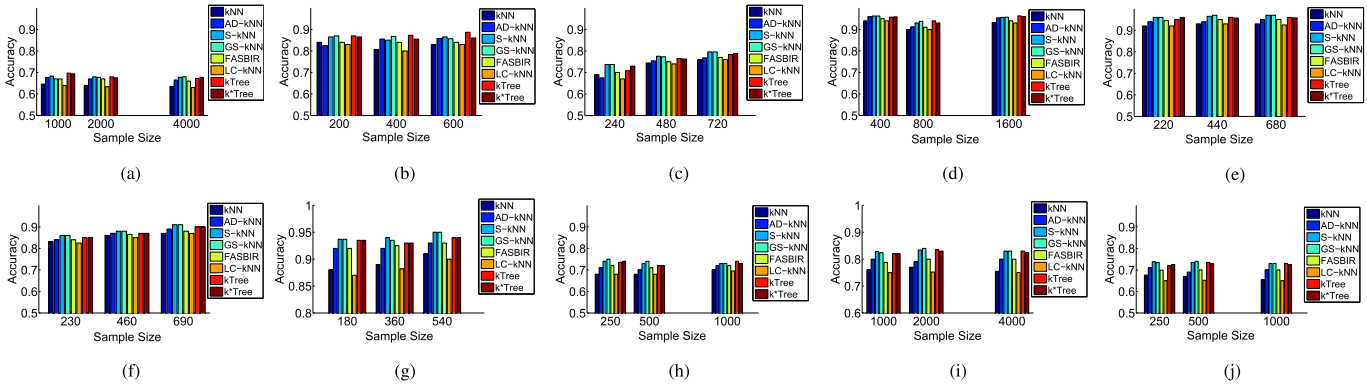


Fig. 4. Classification accuracy on ten data sets with different sample size. (a) Abalone ($\rho_1 = 10^{-4}, \rho_2 = 10^{-5}$). (b) Balance ($\rho_1 = 10^{-3}, \rho_2 = 10^{-5}$). (c) Blood ($\rho_1 = 10^{-3}, \rho_2 = 10^{-4}$). (d) Car ($\rho_1 = 10^{-3}, \rho_2 = 10^{-1}$). (e) BreastOri ($\rho_1 = 10^{-3}, \rho_2 = 10^{-5}$). (f) Australian ($\rho_1 = 10^{-3}, \rho_2 = 10^{-3}$). (g) Climate ($\rho_1 = 10^{-4}, \rho_2 = 10^{-5}$). (h) German ($\rho_1 = 10^{-4}, \rho_2 = 10^{-4}$). (i) DDCclients ($\rho_1 = 10^{-3}, \rho_2 = 10^{-5}$). (j) MicePE ($\rho_1 = 10^{-4}, \rho_2 = 10^{-4}$);.

TABLE I

RESULT OF CLASSIFICATION ACCURACY/RUNNING COST (MEAN)

| Dataset (#(samples)) | kNN | AD-kNN | S-kNN | GS-kNN | FASBIR | LC-kNN | kTree | k*Tree |
|---|---|---|---|---|---|---|---|---|
| Abalone (4177) | 0.640／5.36 | 0.671／2000 | 0.685／42000 | 0.684／42050 | 0.667／10.4 | 0.632／1.50 | **0.683／5.30** | **0.682／1.21** |
| Balance (625) | 0.823／0.23 | 0.846／ 5 | 0.860／ 87 | 0.865／ 92 | 0.845／0.45 | 0.820／0.19 | **0.875／0.22** | **0.857／0.18** |
| Blood (748) | 0.723／0.19 | 0.737／ 7 | 0.769／ 51 | 0.770／ 44 | 0.740／0.40 | 0.720／0.17 | **0.753／0.19** | **0.759／0.14** |
| Car (1728) | 0.923／0.91 | 0.942／ 19 | 0.950／ 678 | 0.951／ 794 | 0.937／2.10 | 0.923／0.52 | **0.954／0.78** | **0.950／0.40** |
| BreastOri (683) | 0.926／0.11 | 0.943／ 4 | 0.965／ 48 | 0.967／ 50 | 0.948／0.20 | 0.925／0.11 | **0.957／0.12** | **0.958／0.09** |
| Australian (690) | 0.850／0.21 | 0.867／ 8 | 0.883／ 38 | 0.883／ 31 | 0.862／0.42 | 0.847／0.12 | **0.874／0.21** | **0.874／0.08** |
| Climate (540) | 0.893／0.13 | 0.924／ 13 | 0.942／ 5 | 0.941／ 6 | 0.925／0.31 | 0.885／0.12 | **0.935／0.19** | **0.935／0.04** |
| German (1000) | 0.686／0.63 | 0.710／ 15 | 0.733／ 32 | 0.740／ 33 | 0.712／1.31 | 0.685／0.35 | **0.732／0.42** | **0.731／0.13** |
| DDCclients (4000) | 0.763／15.2 | 0.800／3000 | 0.828／62000 | 0.824／63000 | 0.788／30.2 | 0.750／3.04 | **0.820／7.10** | **0.820／3.02** |
| MicePE (1080) | 0.670／4.00 | 0.710／ 29 | 0.741／ 1678 | 0.740／ 1794 | 0.699／6.21 | 0.650／2.50 | **0.720／2.78** | **0.725／1.45** |
| AVERAGE | **0.789／2.67** | **0.815／ 510** | **0.835／10661** | **0.836／10789** | **0.812／5.14** | **0.784／0.85** | **0.830／1.72** | **0.829／0.67** |

and $w_{51}$, the first training sample is only related to the last two training samples except itself, i.e., the fourth training sample and the fifth training sample. More specifically, in the kNN classification step, we only need to regard the last two training samples as the nearest neighbors of the first training sample, i.e., the corresponding optimal-$k$-value is 2. Meanwhile, according to the values of the second column of $\mathbf{W}^*$, we only need to regard three training samples as the nearest neighbors of the second training sample, i.e., the corresponding optimal-$k$-value is 3. Obviously, for the third training sample, it should be predicted by the fifth training sample. The corresponding optimal-$k$-value is 1. In this way,

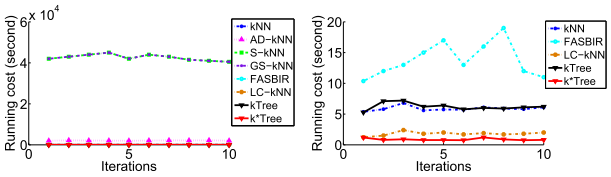


Fig. 5. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on Abalone with a sample size of 4000.

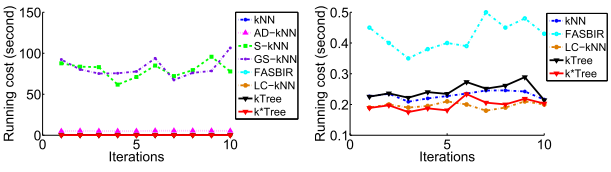


Fig. 6. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on Balance with a sample size of 600.

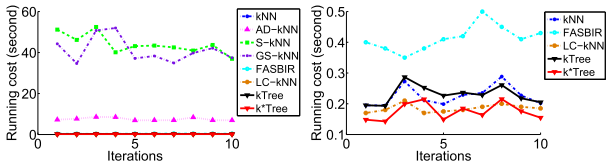


Fig. 7. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on Blood with a sample size of 720.

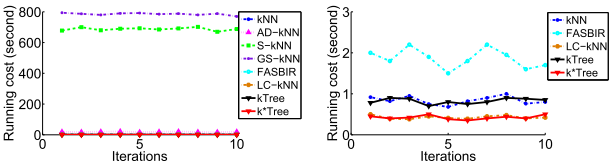


Fig. 8. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on Car with a sample size of 1600.

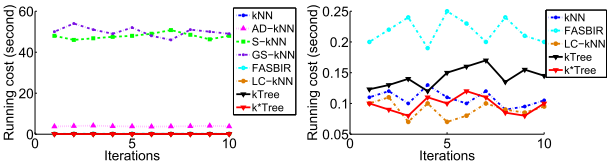


Fig. 9. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on Breast with a sample size of 680.

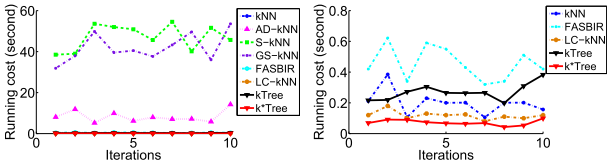


Fig. 10. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on Australian with a sample size of 690.

the nearest neighbors of each training sample are obtained by learning the proposed reconstruction model. Moreover, the optimal-$k$-value in the kNN algorithm are different for different samples. Hence, (7) takes the distribution of data and prior knowledge into account for selecting the optimal-$k$-value for each training sample.

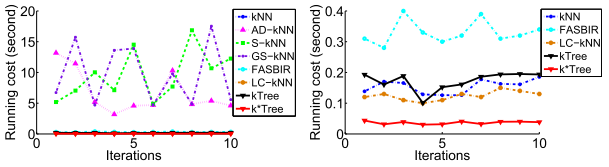


Fig. 11. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on Climate with a sample size of 540.

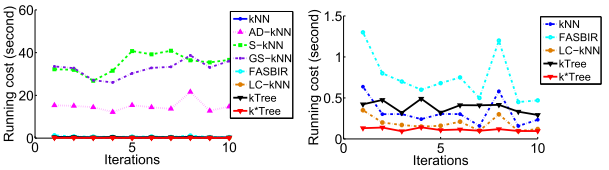


Fig. 12. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on German with a sample size of 1000.

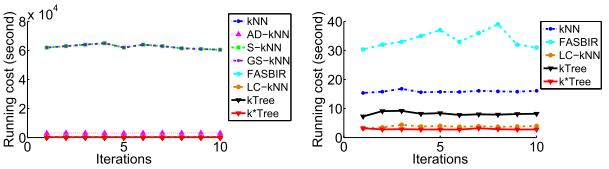


Fig. 13. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on DDCclients with a sample size of 4000.

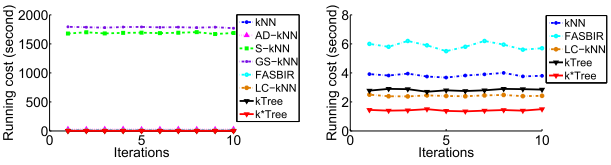


Fig. 14. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on MicePE with a sample size of 1000.

### *D. kTree Method*

The kNN based on graph sparse reconstruction (GS-kNN) method in [30] used (7) to reconstruct test samples by training samples to yield good performance. However, it is time-consuming, i.e., at least $O(n^2)$ for predicting each test sample, where $n$ is the number of training samples. To overcome this, we propose a training stage to construct a $k$-decision tree (namely, kTree) between training samples and their corresponding optimal-$k$-values. The motivation of our method is that we expect to find the relationship between training samples and their optimal-$k$-values so that the learned kTree enables to output an optimal-$k$-value for a test sample in the test stage. In this way, our test stage with time complexity $O(log\ \ (d) + n)$ is faster than both the GS-kNN method in [30] and the fixed kNN method, with the time complexity at least $O(n^2d)$. It should be noteworthy that our proposed method thus results in a training stage involving two steps, i.e., optimizing (7) to yield the optimal-$k$-values for all training samples and constructing the kTree, respectively. Fortunately, both of them are offline.

In the training stage, our kTree method first uses (7) to learn the optimal $\mathbf{W}$ to obtain optimal-$k$-values of training samples, i.e., the numbers of nonzero coefficients in each

column of $\mathbf{W}$ for each training sample. Then, we regard the learned optimal-$k$-values as labels to construct a kTree between training samples and their corresponding optimal-$k$-values. That is, we follow the idea of the state-of-the-art methods such as ID3 [12], [60], [61] to greedily construct a top–down recursive divide-and-conquer decision tree. The difference between our method and ID3 method is that our kTree regards the optimal-$k$-values of training samples as their labels, while ID3 method uses the labels of training samples to construct its decision tree. This results in different items being stored in the leaf nodes, where ID3 stores the labels of training samples and our kTree stores the optimal-$k$-values of training samples. We illustrate their difference in Fig. 3(a) and (b).

In the test stage, we easily obtain the optimal-$k$-values of test samples in the leaf nodes of kTree. Then, we conduct the kNN classification step to classify test samples between training samples and the learned optimal-$k$-values of test samples. Such a test process only needs $O(\log\ (d) + n)$ and is faster than both the varied kNN methods (such as GS-kNN method in [30]) and the fixed kNN method with the time complexity at least $O(n^2 d)$. We list the pseudo of the proposed kTree method in Algorithm 1.

Although our kTree method enables to obtain optimal-$k$-values for test samples, it still needs to conduct kNN classification on all training samples. To further reduce time complexity, we extend our kTree method to an improvement version (namely, k*Tree method) with the time complexity of test stage $O(\log\ (d)+s)$, where $s$ is the cardinality of a subset of training samples, i.e., $s \ll n$.

**Algorithm 1** Pseudo of the Proposed kTree Method

**Input:** training samples $\mathbf{X}$, test samples $\mathbf{Y}$
**Output:** Class labels of $\mathbf{Y}$

* Training Stage *

1. Learning the optimal-$k$-values of all training samples by Eq. (7);
2. Using ID3 method to construct kTree with training samples and their corresponding optimal-$k$-values;
3. Storing *the optimal-k-values of training samples* in leaf nodes;

* Test Stage *

1. Obtaining the optimal-$k$-values of test samples (i.e., $k$) using kTree;
2. Predicting test labels using traditional kNN method with learnt optimal-$k$-values on *all training samples*;

### *E. k*Tree Classification*

In the training stage, the proposed k*Tree method constructs the decision tree (namely, k*Tree) by using the same steps of kTree described in Section III-D. Their difference is the information in the leaf nodes. That is, kTree stores the optimal-$k$-value in leaf nodes, while k*Tree stores the optimal-$k$-value as well as other information in the leaf nodes, including a subset of training samples located in this leaf node, the kNNs of each sample in this subset, and the nearest neighbor of each of these kNNs. Specifically, in the constructed k*Tree, each leaf node contains an optimal-$k$-value (e.g., $k_j$) and a subset of training samples (i.e., $\mathbf{X}' = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_m\}$) which regard $k_j$ as their optimal-$k$-values. Besides these, we also store the $k_j$ nearest neighbors of each sample in $\mathbf{X}'$, denoted as $\mathbf{X}'_i = \{\mathbf{x}'_{i1}, \ldots, \mathbf{x}'_{ik_j}\}$ (where $i = 1, \ldots, m$), and the nearest neighbor of each $\mathbf{x}'_{ik}$ as $\mathbf{x}''_{ik}$ ($k = 1, \ldots, k_j$), denoted as $\mathbf{X}''_i = \{\mathbf{x}''_{i1}, \ldots, \mathbf{x}''_{ik_j}\}$. In this way, each leaf node contains the optimal-$k$-values, $\mathbf{X}'$, $\{\mathbf{X}'_1, \ldots, \mathbf{X}'_m\}$, and $\{\mathbf{X}''_1, \ldots, \mathbf{X}''_m\}$. We list the illustration in Fig. 3(c).

In the test stage, given a test sample (e.g., $\mathbf{x}_t$), the proposed k*Tree method first searches the constructed k*Tree to output its optimal-$k$-value (e.g., $k_t$) as well as its nearest neighbors in the leaf node (e.g., $\mathbf{x}'_t$). With these, the proposed k*Tree method selects $k_t$ nearest neighbors from the subset of training samples, including $\mathbf{x}'_t$, its $k_t$ nearest neighbors $\mathbf{X}'_t = \{\mathbf{x}'_{t1}, \ldots, \mathbf{x}'_{tk_t}\}$, and the nearest neighbors of $\mathbf{X}'_t$, i.e., $\mathbf{X}''_t = \{\mathbf{x}''_{t1}, \ldots, \mathbf{x}''_{tk_t}\}$, and further assigns $\mathbf{x}_t$ with a label according to the majority rule of $k_t$ nearest neighbors. In the proposed k*Tree method, the kNN classification is conducted by select nearest neighbors from the set $\mathbf{S} = \{\mathbf{x}'_t, \mathbf{X}'_t, \mathbf{X}''_t\}$. We denoted the cardinality of $\mathbf{S}$ as $s$, i.e., $s \leq 2 \times k_t + 1$ in this example. The pseudo of k*Tree method is presented in Algorithm 2.

**Algorithm 2** Pseudo of the Proposed k*Tree Method

**Input:** training samples $\mathbf{X}$, test samples $\mathbf{Y}$
**Output:** Class labels of $\mathbf{Y}$

* Training Stage *

1. Learning the optimal-$k$-values of all training samples by Eq. (7);
2. Using ID3 method to construct k*Tree with training samples and their corresponding optimal-$k$-values;
3. Storing *the optimal-k-values of training samples,* $\mathbf{X}'$, $\{\mathbf{X}'_1, \ldots, \mathbf{X}'_m\}$, *and* $\{\mathbf{X}''_1, \ldots, \mathbf{X}''_m\}$, in leaf nodes;

* Test Stage *

1. Obtaining the optimal-$k$-values of test samples (i.e., $k$) using k*Tree;
2. Predicting test labels using traditional kNN method with learnt optimal-$k$-values on $\mathbf{X}'$, $\{\mathbf{X}'_1, .., \mathbf{X}'_m\}$ *and* $\{\mathbf{X}''_1, \ldots, \mathbf{X}''_m\}$;

The principle of kNN method is based on the intuitive assumption that samples in the same class should be closer in the feature space [19]. As a result, for a given test sample of unknown class, we can simply compute the distance between this test sample and all the training samples, and assign the class determined by kNNs of this test sample. In the proposed k*Tree method, we reduce the training set from all training samples to its subset, i.e., the neighbors of the nearest neighbors of the test sample (i.e., $\mathbf{X}'_t$) and the nearest neighbor of all neighbors of the test samples (i.e., $\mathbf{X}''_t$). In this way, we expect that the set $\mathbf{S}$ almost includes all the nearest neighbors in the whole training samples. Actually, our experimental results listed in Section IV verified this assumption, since the proposed k*Tree method achieved similar classification accuracy to the kTree method and the traditional kNN method.

The training complexity of k*Tree method is the same as the kTree method, i.e., $O(n^2)$. In the test stage, the k*Tree method conducts kNN classification on a subset of training samples, i.e., $\mathbf{S} = \{\mathbf{x}'_t, \mathbf{X}'_t, \mathbf{X}''_t\}$, thus resulting in the time complexity of test stage as at most $O(\log\ (d) + s)$ (where $s \ll n$).

## IV. Experiments

### A. Experimental Setting

We used 20 public data sets from UCI Repository of Machine Learning Data sets,[3] whose data sets have been widely used for academic research, to evaluate the proposed methods and the competing methods on the classification task, in terms of classification accuracy and running cost. These data sets include all different types of data, such as low-dimensional data set and high-dimensional data set, binary data sets and multiclass data sets, and imbalance data sets, and are used to evaluate the robust of the proposed methods. In our experiments, we used ten of them (e.g., Abalone, Balance, Blood, Car, Breast, Australian, Climate, and German) for the experiments of different sample size, while the rest (e.g., Madelon, LSVT, CNAE, Gisette, Hill, Libras, Dbworld, and Arcene) for the experiments of different feature numbers. Among of them, both Climate data set containing 46 positive samples and 494 negative samples and German data set including 700 positive samples and 300 negative samples can be regarded as imbalance data sets.

We employed the tenfold cross validation method on all methods. Specifically, we first randomly partitioned the whole data set into ten subsets and then selected one subset for testing and the remaining nine subsets for training. We repeated the whole process ten times to avoid the possible bias during data set partitioning for cross-validation. The final result was computed by averaging results from all experiments. For the model selection of our method, we considered the parameter spaces of $\rho_1 \in \{10^{-5}, 10^{-4}, \ldots, 10^{1}\}$ and $\rho_2 \in \{10^{-5}, 10^{-4}, \ldots, 10^{-1}\}$ in (7).

### B. Competing Methods

In this paper, we selected the state-of-the-art methods, including kNN [19], kNN-based applicability domain approach (AD-kNN) [33], kNN method based on sparse learning (S-kNN) [20], GS-kNN [30], filtered attribute subspace-based bagging with injected randomness (FASBIR), ensembles of nearest neighbor classifiers) [62], [63], and Landmark-based spectral Clustering kNN (LC-kNN) [64] as the competing methods. We list their details as follows.

1) *k-Nearest Neighbor:* kNN is a classical classification method. Following the literature in [19], we set $k = 1$, 5, 10, 20, and the squared root of sample size, respectively, and reported the best result.
2) *kNN-Based Applicability Domain Approach [33]:* AD-kNN integrates salient features of the kNN approach and adaptive kernel methods for conducting probability density estimation. Following the literature [33], we set the parameter $k$ of AD-kNN with the Monte Carlo validation method by setting the maximum number of neighbors as 20.
3) *kNN Method Based on Sparse Learning [20]:* S-kNN learns different $k$ values for different test samples by sparse learning, where a least square loss function is applied to achieve the minimal reconstruction error, an $\ell_1-$norm regularization term is utilized to obtain the elementwise sparsity, and a Laplacian regularization term is used to preserve the local structures of data. Following the literature in [20], we used the cross validation method to conduct model selection by setting $\beta_1$ and $\beta_2$ in the ranges of $\{10^{-5}, 10^{-4}, \ldots, 10^{1}\}$.
4) *kNN Based on Graph Sparse Reconstruction [30]:* GS-kNN first uses training samples to reconstruct test samples to obtain the optimal $k$ values, and then uses the traditional kNN method to conduct classification tasks. Following the literature in [30], we used the cross validation method to conduct model selection by setting the parameter spaces of $\gamma_1 \in \{10^{-5}, 10^{-4}, \ldots, 10^{1}\}$, $\gamma_2 \in \{10^{-5}, 10^{-4}, \ldots, 10^{-1}\}$ and $\gamma_3 \in \{10^{-5}, 10^{-4}, \ldots, 10^{-2}\}$.
5) FASBIR [62], [63] was designed for building ensembles of nearest neighbor classifiers. FASBIR works through integrating the perturbations on the training data, input attributes and learning parameters together.
6) Landmark-based spectral Clustering kNN (LC-kNN) [64] was proposed to first conduct $k$-means clustering to separate the whole data set into several parts and then select the nearest cluster as the training samples for conducting the kNN classification.

### C. Experimental Results on Different Sample Sizes

In this section, we conducted classification tasks with all methods at different sample size on ten UCI data sets, aim at avoiding the bias of imbalanced sample size. We reported the classification accuracy (i.e., the averaging classification accuracy of ten iterations) of all methods in Fig. 4, where the horizontal axis indicates the sample size and the vertical axis represents the classification accuracy. We also listed the running cost (in time) of all methods in each of iteration in Figs. 5–14, where the horizontal axis indicates the number of iterations and the vertical axis represents the running cost. We also list the accuracy and the running cost in Table I.

From Fig. 4 and Table I, we knew that: 1) the proposed methods (i.e., the kTree method and the k*Tree method) improved the classification accuracies on average by 4% (versus kNN), 1.5% (versus AD-kNN), 1.8% (versus FASBIR), and 4.5% (versus LC-kNN), on all ten data sets, while our methods (i.e., the kTree method and the k*Tree method) have almost the same accuracy with GS-kNN and S-kNN and 2) the methods learning optimal-$k$-values for different samples (e.g., our k*Tree method, our kTree method, S-kNN, GS-kNN, and AD-kNN) outperformed the method with fixed expert-predefined value (e.g., kNN). For example, kNN method reduced on average the classification accuracy on ten data sets by 2.6% (versus AD-kNN), which is the worst method learning optimal-$k$-values for different samples in our experiments.

[3] http://archive.ics.uci.edu

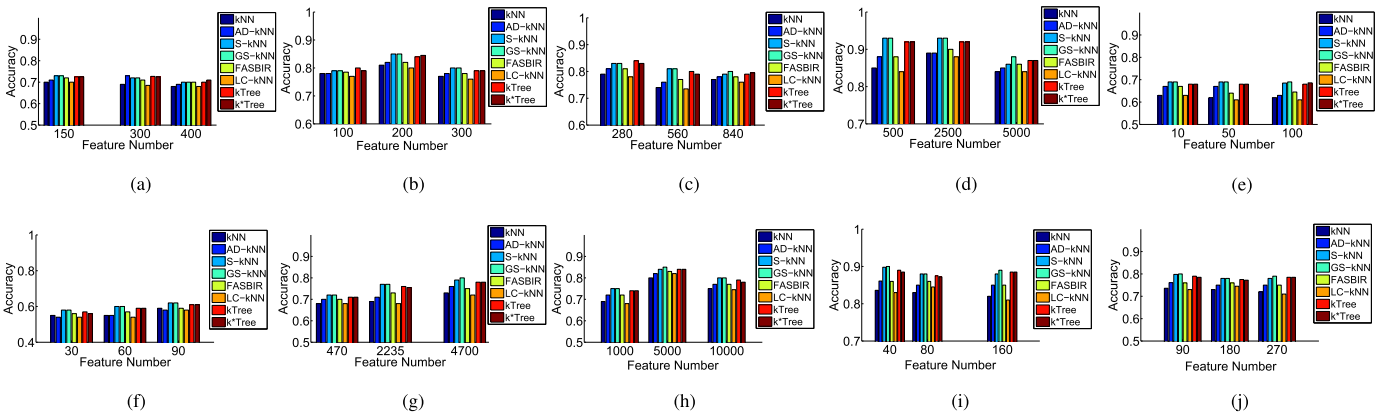


Fig. 15. Classification accuracy on ten data sets with different number of features. (a) Madelon ($\rho_1 = 10^{-3}, \rho_2 = 10^{-3}$). (b) LSVT ($\rho_1 = 10^{-3}, \rho_2 = 10^{-3}$). (c) CNAE ($\rho_1 = 10^{-2}, \rho_2 = 10^{-2}$). (d) Gisette ($\rho_1 = 10^{-1}, \rho_2 = 10^{-5}$). (e) Hill ($\rho_1 = 10^{-3}, \rho_2 = 10^{-3}$). (f) Libras ($\rho_1 = 10^{-3}, \rho_2 = 10^{-4}$). (g) DBworld ($\rho_1 = 10^{-4}, \rho_2 = 10^{-3}$). (h) Arcene ($\rho_1 = 10^{-2}, \rho_2 = 10^{-3}$). (i) Musk ($\rho_1 = 10^{-4}, \rho_2 = 10^{-4}$). (j) Arrhythmia ($\rho_1 = 10^{-3}, \rho_2 = 10^{-5}$).

Regarding the running cost in Figs. 5–14, we have the following observations.

1) Our k*Tree method achieved the minimal running cost, followed by LC-kNN, kNN, our kTree, FASBIR, AD-kNN, GS-kNN, and S-kNN. For example, the k*Tree method was four times faster than kNN on Abalone data set in our experiments. The reason is that the proposed k*Tree method scanned a small subset of training samples to conduct kNN classification, while both kNN and the kTree method conducted kNN classification by scanning all training samples. It is noteworthy that the running cost of LC-kNN is similar to our k*Tree, since LC-kNN conducts $k$-means clustering to separate the whole data set into several parts, i.e., only scanning a subset of training data set. However, our k*Tree outperformed LC-kNN in terms of classification accuracy. Moreover, it is very difficult for LC-kNN to find a suitable number of the parts so that achieving the similar performance as the standard kNN [64].
2) The methods (such as S-kNN, GS-kNN, and AD-kNN) took more running cost than either our methods (e.g., the kTree method and the k*Tree method) or kNN, since both S-kNN and GS-kNN must use training samples to reconstruct test samples to obtain the optimal $k$ values, while AD-kNN took expensive cost to calculate AD of training samples and verified if test samples were inside or outside the AD.

From Table I, our proposed kTree and k*Tree outperformed kNN, AD-kNN and FASBIR in terms of classification accuracy, and also took less running cost. For example, k*Tree improved about 5.7% (versus kNN), 2% (versus AD-kNN), and 3.2% (versus FASBIR), respectively, on DDCclients data set, while k*Tree was about 5 times, 1000 times and 10 times, faster than kNN, AD-kNN, and FASBIR, respectively, in terms of running cost. On the other hand, our k*Tree did not achieve the similar performance as either GS-kNN or S-kNN, but was faster than each of them in terms of running cost. For example, k*Tree was on average about 15 000 times faster than either GS-kNN or S-kNN, but only reducing the classification

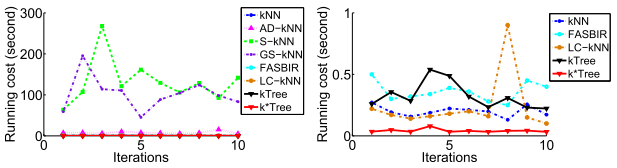


Fig. 16. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on Madelon with a feature number of 450.

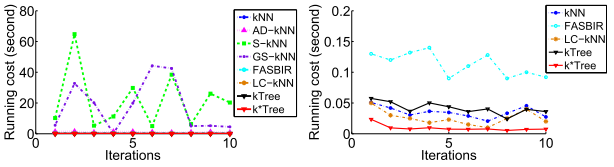


Fig. 17. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on LSVT with a feature number of 300.

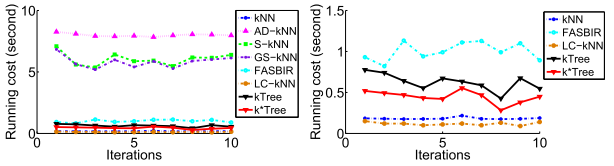


Fig. 18. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on CNAE with a feature number of 840.

accuracy by about 0.6% on all data sets. The reason is that the proposed k*Tree method only scanned a small subset of training samples to conduct kNN classification, while bo GS-kNN and S-kNN scanned all training samples.

### *D. Experimental Results on Different Feature Number*

In this section, we first employed the state-of-the-art method Fisher score [65] to rank all features of the data, and then selected the most informative features for kNN classification. Our goal is to analysis the robustness of all methods with different feature numbers. Fig. 15 listed the classification accuracy of all methods on ten data sets and Figs. 16–25 reported the running cost of each iteration for all methods. We also list the accuracy and the running cost in Table II.

Fig. 15 and Table II clearly indicated that our methods (i.e., the kTree method and the k*Tree method) still

TABLE II
RESULT OF CLASSIFICATION ACCURACY/RUNNING COST (MEAN)

| Dataset (#(features)) | kNN | AD-kNN | S-kNN | GS-kNN | FASBIR | LC-kNN | kTree | k*Tree |
|---|---|---|---|---|---|---|---|---|
| Madelon (500) | 0.691／0.27 | 0.710／7.50 | 0.720／63.1 | 0.721／59.1 | 0.711／0.52 | 0.681／0.22 | **0.717／0.25** | **0.720／0.03** |
| LSVT (310) | 0.780／0.05 | 0.791／1.42 | 0.820／10.2 | 0.820／5.52 | 0.800／0.13 | 0.773／0.05 | **0.812／0.05** | **0.801／0.02** |
| CNAE (856) | 0.762／0.18 | 0.783／8.31 | 0.812／7.10 | 0.813／6.80 | 0.787／0.93 | 0.758／0.15 | **0.810／0.77** | **0.805／0.51** |
| Gisette (5000) | 0.860／0.22 | 0.873／8.12 | 0.907／8.02 | 0.913／8.00 | 0.880／0.70 | 0.853／0.05 | **0.903／0.26** | **0.903／0.15** |
| Hill (100) | 0.621／0.04 | 0.656／2.00 | 0.688／11.0 | 0.690／19.2 | 0.652／0.07 | 0.611／0.05 | **0.682／0.08** | **0.682／0.07** |
| Libras (90) | 0.561／0.03 | 0.556／1.10 | 0.601／6.31 | 0.602／6.12 | 0.573／0.05 | 0.552／0.03 | **0.590／0.04** | **0.586／0.06** |
| DBworld (4702) | 0.702／0.03 | 0.723／1.51 | 0.762／0.42 | 0.763／0.42 | 0.726／0.06 | 0.693／0.03 | **0.752／0.07** | **0.748／0.04** |
| Arcene (10000) | 0.746／0.07 | 0.770／2.52 | 0.796／7.20 | 0.802／7.11 | 0.738／0.12 | 0.742／0.06 | **0.793／0.13** | **0.786／0.12** |
| Musk (168) | 0.836／0.15 | 0.861／4.40 | 0.898／20.1 | 0.900／15.2 | 0.860／1.16 | 0.830／0.14 | **0.894／0.55** | **0.885／0.52** |
| Arrhythmia (279) | 0.736／0.13 | 0.761／4.51 | 0.798／5.61 | 0.801／4.05 | 0.761／0.76 | 0.732／0.13 | **0.795／0.17** | **0.785／0.14** |
| AVERAGE | **0.729／0.12** | **0.748／4.12** | **0.780／13.8** | **0.780／13.1** | **0.749／0.45** | **0.721／0.13** | **0.773／0.20** | **0.770／0.16** |

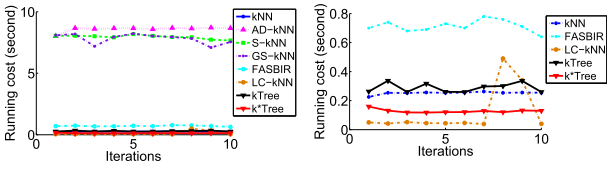


Fig. 19. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on Gisette with a feature number of 5000.

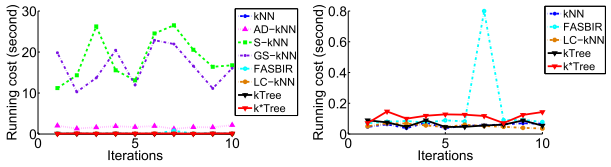


Fig. 20. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on Hill with a feature number of 100.

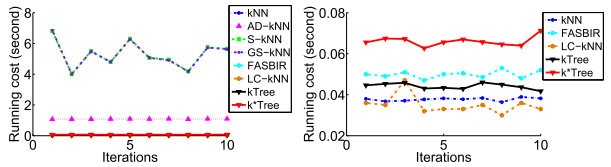


Fig. 21. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on Libras with a feature number of 90.

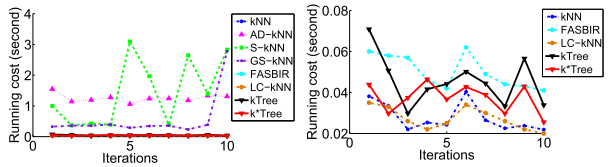


Fig. 22. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on DBworld with a feature number of 4700.

achieved the best classification accuracy, compared with other comparison methods. For example, our k*Tree method improved the classification accuracies on average over ten data sets by 2.6% (versus AD-kNN), 4.2% (versus kNN), 2.3% (versus FASBIR), and 5.1% (versus LC-kNN). Moreover, the proposed kTree method and the proposed k*Tree method had similar results with GS-kNN and S-kNN, in terms of classification accuracy.

Figs. 16–25 intuitively showed that the k*Tree method is faster than the kTree method and kNN on Madelon data set, while slower than kNN on data sets, such as CNAE and

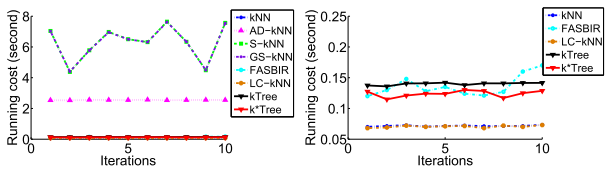


Fig. 23. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on Arcene with a feature number of 10000.

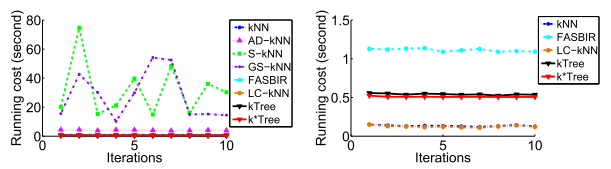


Fig. 24. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on Musk with a feature number of 160.

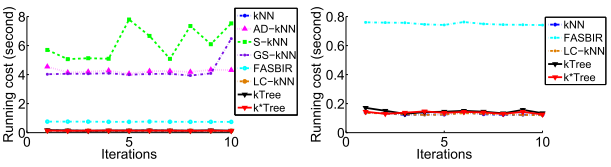


Fig. 25. Running cost of all methods (left) and five methods zoomed in from the left subfigure (right) on Arrhythmia with a feature number of 270.

Musk. The reason is that when the dimensions of the data sets are small and the number of training samples is big, i.e., $n \gg d$, the k*Tree method is much faster than kNN. While the dimensions are large, or even larger than the number of samples, i.e., $d \geq n$, the k*Tree method will be slower than kNN.

From Table II, our proposed kTree and k*Tree still outperformed kNN, AD-kNN, and FASBIR with different feature numbers. Moreover, our k*Tree did not achieve the similar performance as either GS-kNN or S-kNN, but was faster than each of them in terms of running cost. The reason is that the proposed k*Tree method only scanned a small subset of training samples to conduct kNN classification, while both GS-kNN and S-kNN scanned all training samples. In particular, although both k*Tree and LC-kNN were designed to scan a subset of training samples, our k*Tree increased by on average 4.9% (classification accuracy) and also was faster twice than LC-kNN. The reason is that it is difficult for LC-kNN to find a suitable number of the parts, which was concluded in [64]. All the above experimental results showed that the proposed

method k*Tree can be used to improve the performance of kNN method in terms of classification accuracy and running cost.

## V. Conclusion and Future Work

In this paper, we have proposed two new kNN classification algorithms, i.e., the kTree and the k*Tree methods, to select optimal-$k$-value for each test sample for efficient and effective kNN classification. The key idea of our proposed methods is to design a training stage for reducing the running cost of test stage and improving the classification performance. Two set of experiments have been conducted to evaluate the proposed methods with the competing methods, and the experimental results indicated that our methods outperformed the competing methods in terms of classification accuracy and running cost.

In future, we will focus on improving the performance of the proposed methods on high-dimensional data [11], [66]–[68].

## References


[1] S. Zhang, "Shell-neighbor method and its application in missing data imputation," *Appl. Intell.*, vol. 35, no. 1, pp. 123–133, 2011.

[2] L. Shao, L. Liu, and X. Li, "Feature learning for image classification via multiobjective genetic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 7, pp. 1359–1371, Jul. 2014.

[3] D. Tao, X. Tang, X. Li, and X. Wu, "Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 7, pp. 1088–1099, Jul. 2006.

[4] J. Yu, X. Gao, D. Tao, X. Li, and K. Zhang, "A unified learning framework for single image super-resolution," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 4, pp. 780–792, Apr. 2014.

[5] Q. Zhu, L. Shao, X. Li, and L. Wang, "Targeting accurate object extraction from an image: A comprehensive study of natural image matting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 2, pp. 185–207, Feb. 2015.

[6] K. Huang, D. Tao, Y. Yuan, X. Li, and T. Tan, "Biologically inspired features for scene classification in video surveillance," *IEEE Trans. Syst., Man, Cybern., Part B*, vol. 41, no. 1, pp. 307–313, Feb. 2011.

[7] S. Zhang, "Nearest neighbor selection for iteratively $K$NN imputation," *J. Syst. Softw.*, vol. 85, no. 11, pp. 2541–2552, 2012.

[8] T. Wang, Z. Qin, S. Zhang, and C. Zhang, "Cost-sensitive classification with inadequate labeled data," *Inf. Syst.*, vol. 37, no. 5, pp. 508–516, 2012.

[9] X. Zhu, X. Li, and S. Zhang, "Block-row sparse multiview multilabel learning for image classification," *IEEE Trans. Cybern.*, vol. 46, no. 2, pp. 450–461, Feb. 2015.

[10] X. Wu *et al.*, "Top 10 algorithms in data mining," *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, 2008.

[11] K. Zhang, X. Gao, D. Tao, and X. Li, "Single image super-resolution with multiscale similarity learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 10, pp. 1648–1659, Oct. 2013.

[12] D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Trans. Syst., Man Cybern.*, vol. 21, no. 3, pp. 660–674, May 1991.

[13] H. Liu, X. Li, and S. Zhang, "Learning instance correlation functions for multi-label classification," *IEEE Trans. Cybern.*, vol. 47, no. 2, pp. 499–510, Feb. 2017.

[14] S. Zhang, "Parimputation: From imputation and null-imputation to partially imputation," *IEEE Intell. Inform. Bull.*, vol. 9, no. 1, pp. 32–38, Jan. 2008.

[15] G. Góra and A. Wojna, "RIONA: A classifier combining rule induction and $k$-NN method with automated selection of optimal neighbourhood," in *Proc. ECML*, 2002, pp. 111–123.

[16] B. Li, Y. W. Chen, and Y. Q. Chen, "The nearest neighbor algorithm of local probability centers," *IEEE Trans. Syst., Man, B*, vol. 38, no. 1, pp. 141–154, Feb. 2008.

[17] X. Zhu, H.-I. Suk, and D. Shen, "Multi-modality canonical feature selection for Alzheimer's disease diagnosis," in *Proc. MICCAI*, 2014, pp. 162–169.

[18] J. Wang, P. Neskovic, and L. N. Cooper, "Neighborhood size selection in the k-nearest-neighbor rule using statistical confidence," *Pattern Recognit.*, vol. 39, no. 3, pp. 417–423, 2006.

[19] U. Lall and A. Sharma, "A nearest neighbor bootstrap for resampling hydrologic time series," *Water Resour. Res.*, vol. 32, no. 3, pp. 679–693, 1996.

[20] D. Cheng, S. Zhang, Z. Deng, Y. Zhu, and M. Zong, "$K$NN algorithm with data-driven $k$ value," in *Proc. ADMA*, 2014, pp. 499–512.

[21] X. Zhu, S. Zhang, Z. Jin, Z. Zhang, and Z. Xu, "Missing value estimation for mixed-attribute data sets," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 1, pp. 110–121, Jan. 2011.

[22] H. A. Fayed and A. F. Atiya, "A novel template reduction approach for the $K$-nearest neighbor method," *IEEE Trans. Neural Netw.*, vol. 20, no. 5, pp. 890–896, May 2009.

[23] X. Zhu, Z. Huang, H. T. Shen, and X. Zhao, "Linear cross-modal hashing for efficient multimedia search," in *Proc. ACM MM*, 2013, pp. 143–152.

[24] H. Wang, "Nearest neighbors by neighborhood counting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 6, pp. 942–953, Jun. 2006.

[25] Q. Liu and C. Liu, "A novel locally linear KNN method with applications to visual recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published.

[26] X. Zhu, S. Zhang, J. Zhang, and C. Zhang, "Cost-sensitive imputing missing values with ordering," in *Proc. AAAI*, 2007, pp. 1922–1923.

[27] J. Hou, H. Gao, Q. Xia, and N. Qi, "Feature combination and the kNN framework in object classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 6, pp. 1368–1378, Jun. 2016.

[28] S. Zhang, X. Li, M. Zong, X. Zhu, and D. Cheng, "Learning k for kNN classification," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 3, pp. 1–19, 2017.

[29] D. Tao, J. Cheng, X. Gao, X. Li, and C. Deng, "Robust sparse coding for mobile image labeling on the cloud," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 1, pp. 62–72, Jan. 2017.

[30] S. Zhang, M. Zong, K. Sun, Y. Liu, and D. Cheng, "Efficient kNN algorithm based on graph sparse reconstruction," in *Proc. ADMA*, 2014, pp. 356–369.

[31] X. Zhu, L. Zhang, and Z. Huang, "A sparse embedding and least variance encoding approach to hashing," *IEEE Trans. Image Process.*, vol. 23, no. 9, pp. 3737–3750, Sep. 2014.

[32] B. Li, S. Yu, and Q. Lu. (2003). "An improved k-nearest neighbor algorithm for text categorization." [Online]. Available: https://arxiv.org/abs/cs/0306099

[33] F. Sahigara, D. Ballabio, R. Todeschini, and V. Consonni, "Assessing the validity of QSARS for ready biodegradability of chemicals: An applicability domain perspective," *Current Comput.-Aided Drug Design*, vol. 10, no. 2, pp. 137–147, 2013.

[34] X. Zhu, Z. Huang, Y. Yang, H. T. Shen, C. Xu, and J. Luo, "Self-taught dimensionality reduction on the high-dimensional small-sized data," *Pattern Recognit.*, vol. 46, no. 1, pp. 215–229, 2013.

[35] E. Blanzieri and F. Melgani, "Nearest neighbor classification of remote sensing images with the maximal margin principle," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 6, pp. 1804–1811, Jun. 2008.

[36] X. Zhu, X. Li, S. Zhang, C. Ju, and X. Wu, "Robust joint graph sparse coding for unsupervised spectral feature selection," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published.

[37] H. Liu and S. Zhang, "Noisy data elimination using mutual k-nearest neighbor for classification mining," *J. Syst. Softw.*, vol. 85, no. 5, pp. 1067–1074, 2012.

[38] K. S. Ni and T. Q. Nguyen, "An adaptable-nearest neighbors algorithm for MMSE image interpolation," *IEEE Trans. Image Process.*, vol. 18, no. 9, pp. 1976–1987, Mar. 2009.

[39] Y. Pang, Z. Ji, P. Jing, and X. Li, "Ranking graph embedding for learning to rerank," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 8, pp. 1292–1303, Aug. 2013.

[40] T. Mary-Huard and S. Robin, "Tailored aggregation for classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 11, pp. 2098–2105, Nov. 2009.

[41] Y. Qin, S. Zhang, X. Zhu, J. Zhang, and C. Zhang, "Semi-parametric optimization for missing data imputation," *Appl. Intell.*, vol. 27, no. 1, pp. 79–88, 2007.

[42] C. Zhang, Y. Qin, X. Zhu, and J. Zhang, "Clustering-based missing value imputation for data preprocessing," in *Proc. IEEE Int. Conf.*, Aug. 2006, pp. 1081–1086.

[43] X. Zhu, H.-I. Suk, and D. Shen, "A novel multi-relation regularization method for regression and classification in ad diagnosis," in *Proc. MICCAI*, 2014, pp. 401–408.

[44] Y. Song, J. Huang, D. Zhou, H. Zha, and C. L. Giles, "IKNN: Informative K-nearest neighbor pattern classification," in *Knowledge Discovery in Databases*. Berlin, Germany: Springer, 2007, pp. 248–264.
[45] P. Vincent and Y. Bengio, "K-local hyperplane and convex distance nearest neighbor algorithms," in *Proc. NIPS*, 2001, pp. 985–992.
[46] Z. Don, W. Liang, Y. Wu, M. Pei, and Y. Jia, "Nonnegative correlation coding for image classification," *Sci. CHINA Inf. Sci.*, vol. 59, no. 1, pp. 1–14, 2016.
[47] D.-Y. Liu, H.-L. Chen, B. Yang, X.-E. Lv, L.-N. Li, and J. Liu, "Design of an enhanced fuzzy k-nearest neighbor classifier based computer aided diagnostic system for thyroid disease," *J. Med. Syst.*, vol. 36, no. 5, pp. 3243–3254, 2012.
[48] J. Gou, T. Xiong, and Y. Kuang, "A novel weighted voting for k-nearest neighbor rule," *J. Comput.*, vol. 6, no. 5, pp. 833–840, 2011.
[49] V. Premachandran and R. Kakarala, "Consensus of k-NNs for robust neighborhood selection on graph-based manifolds," in *Proc. CVPR*, Jun. 2013, pp. 1594–1601.
[50] G. Guo, H. Wang, H. D. Bell, Y. Bi, and K. Greer, "KNN model-based approach in classification," in *On The Move to Meaningful Internet Systems: CoopIS, DOA, and ODBASE* (Lecture Notes in Computer Science), vol. 2888, R. Meersman, Z. Tari, and D. C. Schmidt, Eds. Berlin, Germany: Springer, 2003, pp. 986–996.
[51] S. Manocha and M. Girolami, "An empirical analysis of the probabilistic k-nearest neighbour classifier," *Pattern Recognit. Lett.*, vol. 28, no. 13, pp. 1818–1824, 2007.
[52] S. Sun and R. Huang, "An adaptive k-nearest neighbor algorithm," in *Proc. FSKD*, vol. 1. 2010, pp. 91–94.
[53] X. Zhu, H.-I. Suk, and D. Shen, "Matrix-similarity based loss function and feature selection for Alzheimer's disease diagnosis," in *Proc. CVPR*, 2014, pp. 3089–3096.
[54] J. Zhang, M. Wang, S. Zhang, X. Li, and X. Wu, "Spatiochromatic context modeling for color saliency analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 6, pp. 1177–1189, Jun. 2016.
[55] X. Li, Y. Pang, and Y. Yuan, "L1-norm-based 2DPCA," *IEEE Trans. Syst., Man, Cybern. B*, vol. 40, no. 4, pp. 1170–1175, Aug. 2010.
[56] X. Zhu, Z. Huang, H. Cheng, J. Cui, and H. T. Shen, "Sparse hashing for fast multimedia search," *ACM Trans. Inf. Syst.*, vol. 31, no. 2, p. 9, 2013.
[57] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang, "Face recognition using Laplacianfaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 3, pp. 328–340, Mar. 2005.
[58] X. He and P. Niyogi, "Locality preserving projections," in *Proc. Neural Inf. Process. Syst.*, vol. 16. 2004, p. 153.
[59] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[60] H. Li and J. Sun, "Majority voting combination of multiple case-based reasoning for financial distress prediction," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 4363–4373, 2009.
[61] A. Bahety, "Extension and Evaluation of ID3—Decision Tree Algorithm," *Entropy*, vol. 2, no. 1, p. 1, 2014.
[62] Z. H. Zhou and Y. Yu, "Ensembling local learners throughmultimodal perturbation," *IEEE Trans. Syst. Man, B*, vol. 35, no. 4, pp. 725–735, Apr. 2005.
[63] Z. H. Zhou, *Ensemble Methods: Foundations and Algorithms*. London, U.K.: Chapman & Hall, 2012.
[64] Z. Deng, X. Zhu, D. Cheng, M. Zong, and S. Zhang, "Efficient kNN classification algorithm for big data," *Neurocomputing*, vol. 195, pp. 143–148, Jun. 2016.
[65] K. Tsuda, M. Kawanabe, and K.-R. Müller, "Clustering with the fisher score," in *Proc. NIPS*, 2002, pp. 729–736.
[66] X. Zhu, H.-I. Suk, and D. Shen, "A novel matrix-similarity based loss function for joint regression and classification in AD diagnosis," *NeuroImage*, vol. 100, pp. 91–105, Oct. 2014.
[67] Y. Pang, Y. Yuan, and X. Li, "Effective Feature Extraction in High-Dimensional Space," *IEEE Trans. Syst., Man, B*, vol. 38, no. 6, pp. 1652–1656, Dec. 2008.
[68] S. Zhang and X. Wu, "Large scale data mining based on data partitioning," *Appl. Artif. Intell.*, vol. 15, no. 2, pp. 129–139, 2001.

**Shichao Zhang** (SM'04) is currently a China 1000-Plan Distinguished Professor with Guangxi Normal University, Guangxi, China. His current research interests include data mining and partitioning.

Prof. Zhang is a member of the ACM

**Xuelong Li** (M'02–SM'07–F'12) is currently a Full Professor with the State Key Laboratory of Transient Optics and Photonics, Center for OPTical IMagery Analysis and Learning, Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an, China.

**Ming Zong** received the master's degree in computer science from Guangxi Normal University, Guangxi, China. He is currently pursuing the Ph.D. degree with the Institute of Natural and Mathematical Sciences, Massey University, Auckland, New Zealand.

His current research interests include data mining, sparse coding, and deep learning.

**Xiaofeng Zhu's** current research interests include large-scale multimedia retrieval, feature selection, sparse learning, data preprocess, and medical image analysis.

Dr. Zhu is a Faculty Member of Guangxi Normal University, Guangxi, China.

**Ruili Wang** received the Ph.D. degree in computer science from Dublin City University, Dublin, Ireland.

He is currently an Associate Professor with the Institute of Natural and Mathematical Sciences, Massey University, Auckland, New Zealand, where he is the Director of the Centre of Language and Speech Processing. His current research interests include speed processing, language processing, image processing, data mining, and intelligent systems.

Dr. Wang serves as a member and an Associate Editor of the editorial boards for international journals, such as *Knowledge and Information Systems* and *Applied Soft Computing*. He was a recipient of the most prestigious research grants in New Zealand, i.e., the Marsden Fund.