Learning *k* for kNN Classification

SHICHAO ZHANG, Guangxi Key Lab of MIMS & Guangxi Normal University
XUELONG LI, Chinese Academy of Sciences
MING ZONG, XIAOFENG ZHU, and DEBO CHENG, Guangxi Key Lab of MIMS & Guangxi Normal University

The *K* Nearest Neighbor (kNN) method has widely been used in the applications of data mining and machine learning due to its simple implementation and distinguished performance. However, setting all test data with the same *k* value in the previous kNN methods has been proven to make these methods impractical in real applications. This article proposes to learn a correlation matrix to reconstruct test data points by training data to assign different *k* values to different test data points, referred to as the Correlation Matrix kNN (CM-kNN for short) classification. Specifically, the least-squares loss function is employed to minimize the reconstruction error to reconstruct each test data point by all training data points. Then, a graph Laplacian regularizer is advocated to preserve the local structure of the data in the reconstruction process. Moreover, an ℓ_1-norm regularizer and an $\ell_{2,1}$-norm regularizer are applied to learn different *k* values for different test data and to result in low sparsity to remove the redundant/noisy feature from the reconstruction process, respectively. Besides for classification tasks, the kNN methods (including our proposed CM-kNN method) are further utilized to regression and missing data imputation. We conducted sets of experiments for illustrating the efficiency, and experimental results showed that the proposed method was more accurate and efficient than existing kNN methods in data-mining applications, such as classification, regression, and missing data imputation.

CCS Concepts: • **AI Technology** → **Machine Learning**; • **Data Mining** → *Classification algorithm;*

Additional Key Words and Phrases: kNN method, sparse learning, missing data imputation

ACM Reference Format:
Shichao Zhang, Xuelong Li, Ming Zong, Xiaofeng Zhu, and Debo Cheng. 2017. Learning *k* for kNN classification. ACM Trans. Intell. Syst. Technol. 8, 3, Article 43 (January 2017), 19 pages.
DOI: http://dx.doi.org/10.1145/2990508

1. INTRODUCTION

Classification is one of most important research topics in data mining (especially for big data mining) [Sun and Reddy 2013; Wu et al. 2014, 2015; Zhu et al. 2016b; Li et al.

This work was supported in part by the China Key Research Program (Grant no: 2016YFB1000905), the National Natural Science Foundation of China (Grant Nos: 61263035, 61573270, and 61672177), the China 973 Program (Grant no: 2013CB329404), the Guangxi Natural Science Foundation (Grant no: 2015GXNSFCB139011), the Guangxi Higher Institutions' Program of Introducing 100 High-Level Overseas Talents, the Guangxi Collaborative Innovation Center of Multi-Source Information Integration and Intelligent Processing, and the Guangxi "Bagui" Teams for Innovation and Research.
Authors' addresses: S. Zhang, M. Zong, X. Zhu (corresponding author), and D. Cheng, Guangxi Key Lab of MIMS and College of Computer Science and Information Technology, Guangxi Normal University, Guilin, Guangxi, PR China; emails: zhangsc@mailbox.gxnu.edu.cn, 920902817@qq.com, xfzhu0011@hotmail.com and 729483509@qq.com; X. Li is with the Center for OPTical IMagery Analysis and Learning (OPTIMAL), State Key Laboratory of Transient Optics and Photonics, Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an 710119, Shaanxi, P. R. China; email: xuelong_li@opt.ac.cn.

ACM 2157-6904/2017/01-ART43 $15.00
DOI: http://dx.doi.org/10.1145/2990508

2015]. The main task of classification is to predict the labels of test data points by inducing all training data points. Over the past few decades, a great many classification methods have been developed in real applications [Luo et al. 2016; Zhu et al. 2016a; He et al. 2016; Li et al. 2016], among of which k Nearest Neighbors (kNN) classification has been regarded as one of the top 10 data-mining algorithms [Wu et al. 2008], due to its simplicity and efficiency. Thus the kNN method has been successfully developed in data-mining applications, such as classification, regression, and missing value imputation. The key idea of a standard kNN method is to predict the label of a test data point by the majority rule, that is, the label of the test data point is predicted with the major class of its k most similar training data points in the feature space [Cheng et al. 2015].

As is well known, kNN classification has at least two open issues to be addressed [Zhang 2010; Zhu et al. 2007], that is, the similarity measurement between two data points and the selection of the k value. Many methods have been proposed to address the first issue, such as Euclidean distance, Mahalanobis distance, and Minkowsky distance and their variants. The common conclusion of the first issue is that different applications need different distance measurements [Qin et al. 2007; Zhang et al. 2006; Zhu et al. 2011]. Accordingly, this article focuses on the second issue, that is, the selection of k value by simply employing Euclidean distance to calculate the similarity (or distance) between two data points.

Previous kNN classification methods select the k value by either setting a fixed constant for all test data or conducting cross-validation to estimate the k value for each test data point. This often leads to low prediction rate in real classification applications due to the fact that these methods do not give a consideration to the distribution of the data [Qin et al. 2007; Zhang et al. 2006]. We illustrate this issue by using two examples in Figures 1 and 2.

In Figure 1, by setting $k = 5$ for the whole problem space, two test data points are assigned to positive class according to the majority rule. From the distribution, $k = 5$ is suitable for predicting the label of the left test data point, but unsuitable for the right one. The right test data point should be predicted to negative class. This can be carried out with $k = 3$. It indicates that different test data points should take different numbers of nearest neighbors.

For a similar scenario of regression (or missing value imputation) in Figure 2, it is reasonable to take $k = 3$ and $k = 2$ for the left test data point and the right one, respectively. This scenario also indicates that different test data points should take different numbers of nearest neighbors in real kNN prediction applications. It says that setting k as a fixed constant for all test data points (the whole problem space) can often lead to low prediction rates in real classification applications.

This article proposes a new kNN method by extending our conference version in Zhang et al. [2014][1] to address the above issue. This proposed method, referred to as the Correlation Matrix kNN (CM-kNN for short), is devised to learn different k values for different test data points by following the distribution of training data. The goal of our proposed method is to make the best use of prior knowledge inherent in training data, including the correlation among data points, the removal of noisy data, and the preservation of the local structures of the data. Specifically, we first design a reconstruction process between training data and test data to obtain the k value of

[1]Compared to our conference version in Zhang et al. [2014], we have improved the abstract and introduction, added the newly literatures to the related work, rearranged the GS-kNN method [Zhang et al. 2014], and added three comparison methods and 12 real datasets to the experimental part to examine the robustness and scalability of the proposed approach.

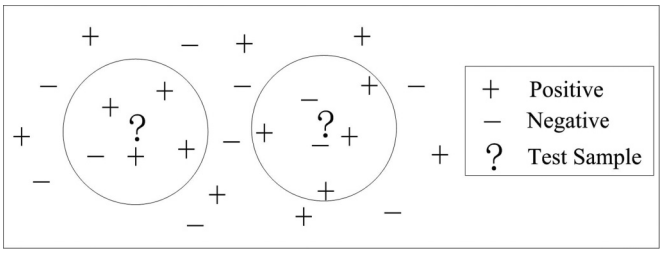

Fig. 1. Binary classification using the kNN method with a fixed k value, that is, $k = 5$.

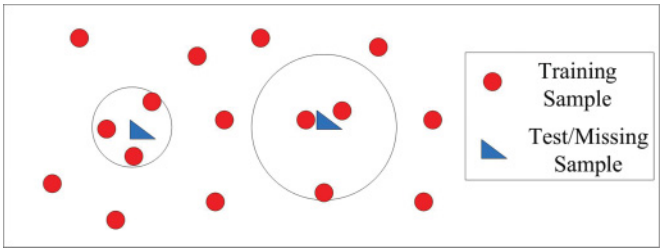

Fig. 2. Regression and missing value imputation using the kNN method with a fixed k value, that is, $k = 3$.

the kNN method for each test data point. We then use the learned k value to conduct classification, regression, and missing value imputation by using a standard kNN method. In the proposed reconstruction process, we advocate an ℓ_1-norm regularizer to result in element-wise sparsity [Luo et al. 2014; Liu et al. 2015; Ye and Li 2016; Li and Pang 2009] to generate different k values for different test data points. Then we use an $\ell_{2,1}$-norm regularizer to generate the row sparsity to remove the impact of noisy data points [Yang et al. 2012; Zhu et al. 2013a, 2013b, 2014]. In addition, we employ a Locality Preserving Projection (LPP) [Niyogi 2004] regularizer (that is, a graph Laplacian regularizer) to preserve the local structure of training data in the reconstruction process.

The rest of this article is organized as follows. Section 2 briefly recalls the reports on the kNN method from research areas of classification, regression, and missing value imputation. Then the CM-kNN classification method is described in Section 3. The proposed approach is evaluated by conducting sets of experiments in Section 4. This research is concluded in Section 5.

2. RELATED WORK

Because of its high performance in real applications [Blanzieri and Melgani 2008; Ni and Nguyen 2009; Fan et al. 2015] and nonparametric setting [Mary-Huard and Robin 2009; Li et al. 2008], the kNN method was regarded as one of the top 10 algorithms in data mining and has been widely developed in data-mining applications, such as classification, regression, and missing value imputation [Dong et al. 2015b]. For simplification, this section briefly recalls the related work on kNN classification, kNN imputation, and kNN regression.

2.1. kNN Classification

The kNN classifier has shown remarkable performance on data with a large example size, such as approaching infinity, in which its error rate approximately reaches the Bayes optimization under very mild conditions. However, the performance of the kNN

classification can be affected by some issues, such as the selection of the k value, the selection of distance measures, and so on. Recently, many techniques have been developed to overcome these issues. For example, the kNN incorporating Certainty Factor (kNN-CF) classification method can incorporate the certainty factor measure into the conventional kNN method so it can be applied to the beginning of the kNN classification to meet the need of imbalanced learning [Zhang 2010]. Moreover, the kNN-CF classification method can be easily extended to the dataset with skewed class distribution. Song et al. proposed two novel kNN approaches, that is, Locally Informative-kNN and Globally Informative-kNN, respectively, via designing new measure metrics for selecting a subset of the most informative data point from neighborhoods [Song et al. 2007]. Vincent and Bengio modified the conventional kNN method to be the K-local Hyperplane Distance Nearest Neighbor (HkNN) method, which applied the collection of 15–70 nearest neighbors from each class to span a linear subspace for that class, followed by conducting classification based on distance to the linear subspaces [Vincent and Bengio 2001]. Wang proposed a new measure to define the similarity between two data points using the number of neighborhoods for conducting a new kNN classifier [Wang 2006]. Zhang et al. [2016] proposed a novel k Nearest Neighbor algorithm, which is based on sample self-representation, sparse learning, and the technology of decision tree. Sun et al. [2015] studied a new type of query based on the k-Nearest Neighbor temporal aggregate, which organizes the locations by integrating the spatial and temporal aggregate information. Tang et al. [2011] studied a new type of query that finds the k Nearest Neighboring Trajectories (k-NNT) with the minimum aggregated distance to a set of query points.

2.2. kNN Missing Data Imputation

Missing data can be found everywhere in real applications and often lead to inaccurate results, lessening the learning performance in machine learning and data mining. Recently, missing value imputation has been shown to be a very important solution to deal with missing data, especially kNN-based methods. For example, Zhang et al. proposed a Grey-Based kNN Iteration Imputation method [Zhang et al. 2007], which efficiently reduced the time complexity and got over the slow convergence rate of the classical missing value imputation method, that is, the EM (Expectation Maximization) algorithm. Based on nearest-neighbor imputation, Chen and Shao proposed some jackknife variance estimators, which are asymptotically unbiased and consistent for the example means [Chen and Shao 2001]. Meesad and Hengpraprohm proposed an imputation method combing the kNN-based feature selection with kNN-based imputation. Differing from the conventional kNN method, their method first conducts feature selection, and then estimates missing values [Meesad and Hengpraprohm 2008]. García-Laencina et al. proposed to employ mutual information to design a feature-weighted distance metric for conducting kNN [García-Laencina et al. 2009]. Most recently, Zhang proposed a Shell Neighbors imputation method to select the left and right nearest neighbors of missing data for imputing missing data [Zhang 2011].

2.3. kNN Regression

The kNN method for solving regression problems is still popular in machine learning and data mining. However, the drawbacks of conventional kNN regression usually include low efficiency, ignoring feature weights in distance calculation, and so on. To address these issues, many approaches have been designed. For example, Hamed et al. proposed an interval regression method based on the conventional kNN method by taking advantage of the possibility distribution to choose the value of k of kNN method due to the limited example size [Hamed et al. 2012]. Based on the observation that conventional kNN regression is sensitive to the selection of similarity metric,

Yao proposed a general kNN framework to infer the similarity metric as a weighted combination of a set of base similarity measures [Yao and Ruzzo 2006]. Navot et al. proposed a new nearest neighbor method to capture complete dependency of the target function [Navot et al. 2006].

From the above three research directions and publications, previous studies on kNN methods always separately focused on classification, regression, and missing value imputation. In this article, we study a kNN approach for taking into account the drawbacks of the conventional kNN method, such as the fixed k value in the kNN method, the removal of noisy data points, and the preservation of the local structures of data. In particular, we apply the proposed kNN apporach to simultaneously conduct classification, regression, and missing value imputation.

3. CM-KNN APPROACH

3.1. Notations

In this article, we denote matrices as bold uppercase letters, vectors as bold lowercase letters, and scalars as normal italic letters, respectively. For a matrix $\mathbf{X} = [x_{ij}]$, its ith row and jth columns are denoted as $\mathbf{x}^i$ and $\mathbf{x}_j$, respectively. Also we denote the Frobenius norm, ℓ_2-norm, ℓ_1-norm, and $\ell_{2,1}$-norm of a matrix $\mathbf{X}$, respectively, as $||\mathbf{X}||_F = \sqrt{\sum_i ||\mathbf{x}^i||_2^2} = \sqrt{\sum_j ||\mathbf{x}_j||_2^2}$, $||\mathbf{X}||_2 = \sqrt{\sum_i \sum_j |x_{ij}|^2}$, $||\mathbf{X}||_1 = \sum_i \sum_j |x_{ij}|$, and $||\mathbf{X}||_{2,1} = \sum_i ||\mathbf{x}^i||_2 = \sum_i \sqrt{\sum_j \mathbf{x}_{ij}^2}$. We further denote the transpose operator, the trace operator, and the inverse of a matrix $\mathbf{X}$ as $\mathbf{X}^T$, $Tr(\mathrm{X})$, and $\mathbf{X}^{-1}$, respectively.

3.2. Reconstruction

Let $\mathbf{X} \in R^{n \times d}$ denote the set of training data points, where n is the number of training data points and d is the dimensionality of features, and assume $\mathbf{Y} \in R^{d \times m}$ denotes the matrix of test data, where m is the number of test data. In this article, we use training data points $\mathbf{X}$ to reconstruct each test data $\mathbf{y}_i$ so the distance between $\mathbf{X}^T\mathbf{w}_i$ and $\mathbf{y}_i$, where $\mathbf{w}_i \in R^n$ denotes the reconstruction weights of training data points, is as small as possible. This leads to the least-squares loss function [Zhu et al. 2014] as follows:

$$\min_{\mathbf{W}} \sum_{i=1}^{m} ||\mathbf{X}^T\mathbf{w}_i - \mathbf{y}_i||_2^2 = \min_{\mathbf{W}} ||\mathbf{X}^T\mathbf{W} - \mathbf{Y}||_F^2, \tag{1}$$

where $||.||_F$ denotes the Frobenius matrix norm and $\mathbf{W} \in R^{n \times m}$ denotes reconstruction weight matrix or the correlations between training data points and test data.

Obviously, the optimization function in Equation (1) is convex and smooth, and the optimal weight matrix $\mathbf{W}$ is obtained as $\mathbf{W}^* = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{Y}$. However, $\mathbf{X}\mathbf{X}^T$ is not always invertible in real applications. To this end, the conventional objection function is added to the smooth regularization term, for example, an ℓ_2-norm,

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

where $||\mathbf{W}||_2$ is an ℓ_2-norm regularization term and ρ is a tuning parameter. Usually, Equation (2) is called ridge regression, and its close solution is $\mathbf{W}^* = (\mathbf{X}\mathbf{X}^T + \rho\mathbf{I})^{-1}\mathbf{X}\mathbf{Y}$. Although the ridge regression in Equation (2) can solve the singular issue in Equation (2), the solution $\mathbf{W}^*$ does not satisfy our requirement because each element in $\mathbf{W}^*$ may be non-zero value. However, we expect to generate the sparsity for the correlation between training data points and the test data. In this way, all of the test data can be represented by partial training data points. Therefore, an ℓ_1-norm regularization term

is proposed to replace the ℓ_2-norm regularization term, and it is defined as follows:

$$R_1(\mathbf{W}) = ||\mathbf{W}||_1. \tag{3}$$

The ℓ_1-norm has been proved to lead the optimal sparse $\mathbf{W}^*$ [Zhu et al. 2016a; Wang et al. 2014; Zhang et al. 2011]. Moreover, the corresponding objective function is also called the Least Absolute Shrinkage and Selection Operator (LASSO) [Zhu et al. 2013a, 2014; Dong et al. 2015a]. It can generate element-wise sparsity in the optimal $\mathbf{W}^*$, that is, irregular sparsity in the elements of the matrix $\mathbf{W}^*$.

We also consider removing the noisy data points that are almost irrelevant to all test data during the reconstruction process. Consider the $\ell_{2,1}$-norm regularization term: It leads to the reconstruction process to generate the sparseness through the whole rows of $\mathbf{W}$, that is, row sparsity for short [Zhu et al. 2016b; Chen et al. 2016; Li et al. 2016]. It is defined as follows:

$$R_2(\mathbf{W}) = ||\mathbf{W}||_{2,1}. \tag{4}$$

Moreover, this article considers to hold the local consistency of the structures of the data during the reconstruction process, in particular to preserve the local consistency of the structures of the features in the data points [Shi et al. 2013]. To this end, a LPP regularization term is added to the objective function Equation (2). It is defined as follows:

$$R_3(\mathbf{W}) = Tr(\mathbf{W}^T\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{W}), \tag{5}$$

where $\mathbf{L} \in \mathbb{R}^{d\times d}$ is a Laplacian matrix, and L = D - S with a similarity matrix $\mathbf{S} \in \mathbb{R}^{d\times d}$ and a diagonal matrix $\mathbf{D} \in \mathbb{R}^{d\times d}$. The LPP is a nonlinear dimensionality reduction method, and the goal of LPP is to ensure that k Nearest Neighbors of original data are correspondingly preserved in the new space after conducting dimensionality reduction.

To meet all of the above requirements, our final objective function is thus formulated as follows:

$$\min_{\mathbf{W}} ||\mathbf{X}^T\mathbf{W} - \mathbf{Y}||_F^2 + \rho_1 R_1(\mathbf{W}) + \rho_2 R_2(\mathbf{W}) + \rho_3 R_3(\mathbf{W}). \tag{6}$$

After optimizing Equation (6) with the proposed optimization method shown in Section 3.3, we obtain the optimal of $\mathbf{W}^*$, that is, the reconstruction weights or the correlations between training data points and test data. The element w_{ij} of $\mathbf{W}^*$ denotes the correlation between ith training data point and jth test data. The positive weight (that is, $w_{ij} > 0$) indicates that there is positive correlation between ith training data point and jth test data, whereas the negative weight (that is, $w_{ij} < 0$) means negative correlation. In particular, the zero weight (that is, $w_{ij} = 0$) means that there is no correlation between ith training data point and jth test data. In this case, the ith training data point should not be used for predicting the jth test data. That is, we use only those related training data points, that is, the training data points with nonzero coefficient, to predict a test data, rather than using all training data points to predict the test data. In this way, Equation (6) takes into account the distribution of data and prior knowledge for selecting the k value for each test data.

To better understand the characteristics of the proposed method, we assume the optimal $\mathbf{W}^* \in R^{5\times 3}$ as follows:

$$\mathbf{W}^* = \begin{bmatrix} 0 & 0.2 & 0 \\ 0 & 0 & 0 \\ 0 & 0.7 & 0.6 \\ 0.3 & 0.9 & 0 \\ 0.4 & 0 & 0 \end{bmatrix}.$$

In this example, we have five training data points and three test data. According to the proposed method, the values in the first column of $\mathbf{W}^*$ indicates the correlations

ALGORITHM 1: The Pseudo of Objective Function Equation (6).

Input: $\mathbf{X}$, $\mathbf{Y}$;
Output: $\mathbf{W}^{(t)} \in R^{n\times m}$;
Initialize $\mathbf{W}^1 \in R^{n\times m}$, $t = 1$;
while *not converge* **do**
  Calculate the diagonal matrices $\mathbf{D}_i^{(t)}(1 \le i \le m)$ and $\tilde{\mathbf{D}}^{(t)}$, where the kth diagonal element of $\mathbf{D}_i^{(t)}$ is $\frac{1}{2|w_{ki}^{(t)}|}$ and the kth diagonal element of $\tilde{\mathbf{D}}^{(t)}$ is $\frac{1}{2||(\mathbf{w}^{(i)})^k||_2}$;
  For each $i(1 \le i \le m)$, $\mathbf{w}_i^{(t+1)} = (\mathbf{X}\mathbf{X}^T + \rho_1\mathbf{D}_i^{(t)} + \rho_2\tilde{\mathbf{D}}^{(t)} + \rho_3\mathbf{X}\mathbf{L}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{y}^i$;
  $t = t + 1$;
end

between the first test data and five training data points. As a result, there are only two non-zero values in the first column, that is, w_{14} and w_{15}. This indicates that the test data are only related to the last two training data points, that is, the fourth training data point and the fifth training data point. More specifically, in the kNN algorithm, we only need to regard the last two training data points as the nearest neighbors of the first test data, that is, the corresponding value of k as 2. Meanwhile, according to the values of the second column of $\mathbf{W}^*$, we only need to regard three training data points as the nearest neighbors of the second test data, that is, the corresponding value of k is 3. Obviously, for the third test data, it should be predicted by the third training data point. The corresponding value of k is 1. In this way, the nearest neighbors of each test data are obtained. Moreover, the value of k in the kNN algorithm differs and is learned according to the distribution of data.

Furthermore, on one hand, we find the sparsity of $\mathbf{W}^*$ is irregular, that is, there is sparsity in elements of the matrix $\mathbf{W}^*$. On the other hand, we also find that the values in the second row of $\mathbf{W}^*$ are all zero, and this indicates that the second training data point is unrelated to all test data. We can regard the second training data points as the noisy training data point. Actually, the ℓ_1-norm in Equation (6) ensures that we product zero values in elements, while the $\ell_{2,1}$-norm in Equation (6) ensuresthat we remove the impact of the noisy training data points. Furthermore, the LPP term in Equation (6) ensures that we further improve the performance of the kNN algorithm.

3.3. Optimization

The objective function Equation (6) is convex but non-smooth, and this article employs the framework of Iteratively Reweighted Least Square [Daubechies et al. 2010] to optimize Equation (6) as follows.

We first take the derivative with respect to each row $\mathbf{w}_i(1 \le i \le m)$ and then set it to zero as follows:

$$\mathbf{X}\mathbf{X}^T\mathbf{w}_i - \mathbf{X}\mathbf{y}^i + \rho_1\mathbf{D}_i\mathbf{w}_i + \rho_2\tilde{\mathbf{D}}\mathbf{w}_i + \rho_3\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{w}_i = 0, \tag{7}$$

where $\mathbf{D}_i(1 \le i \le m)$ is a diagonal matrix with the kth diagonal element as $\frac{1}{2|w_{ki}|}$, and $\tilde{\mathbf{D}}$ is a diagonal matrix with the kth diagonal element as $\frac{1}{2||\mathbf{w}^k||_2}$. We further change Equation (7) to Equation (8) as follows:

$$\mathbf{w}_i = (\mathbf{X}\mathbf{X}^T + \rho_1\mathbf{D}_i + \rho_2\tilde{\mathbf{D}} + \rho_3\mathbf{X}\mathbf{L}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{y}^i. \tag{8}$$

In Equation (8), $\mathbf{D}$ and $\tilde{\mathbf{D}}$ are unknown and depend on $\mathbf{W}$. According to Zhu et al. [2013b], we design an iteration method listed in Algorithm 1 and give the proof of the convergence of the proposed Algorithm 1 as follows.

THEOREM 3.1. *Algorithm 1 decreases the objective value in Equation (6) in each iteration.*

PROOF. According to Step 2 in Algorithm 1, we have

$$\mathbf{W}^{(t+1)} = \min_{\mathbf{W}} Tr((\mathbf{X}^T\mathbf{W}-\mathbf{Y})^T(\mathbf{X}^T\mathbf{W}-\mathbf{Y})) + \rho_1\sum_{i=1}^{m}\mathbf{w}_i^T\mathbf{D}_i^{(t)}\mathbf{w}_i + \rho_2 Tr\mathbf{W}^T\tilde{\mathbf{D}}^{(t)} + \rho_3\mathbf{X}\mathbf{L}\mathbf{X}^T.$$

Therefore, we have

$$\begin{aligned}
&Tr((\mathbf{X}^T\mathbf{W}^{(t+1)}-\mathbf{Y})^T(\mathbf{X}^T\mathbf{W}^{(t+1)}-\mathbf{Y})) + \rho_1\sum_{i=1}^{m}(\mathbf{w}_i^{(t+1)})^T\mathbf{D}_i^{(t)}\mathbf{w}_i^{(t+1)} \\
&\quad + \rho_2 Tr(\mathbf{W}^{(t+1)})^T\tilde{\mathbf{D}}^t\mathbf{W}^{(t+1)} + \rho_3\mathbf{X}\mathbf{L}\mathbf{X}^T \\
&\le Tr((\mathbf{X}^T\mathbf{W}^{(t)}-\mathbf{Y})^T(\mathbf{X}^T\mathbf{W}^{(t)}-\mathbf{Y})) + \rho_1\sum_{i=1}^{m}(\mathbf{w}_i^{(t)})^T\mathbf{D}_i^{(t)}\mathbf{w}_i^{(t)} \\
&\quad + \rho_2 Tr(\mathbf{W}^{(t)})^T\tilde{\mathbf{D}}^t\mathbf{W}^{(t)} + \rho_3\mathbf{X}\mathbf{L}\mathbf{X}^T \\
&\Rightarrow Tr((\mathbf{X}^T\mathbf{W}^{(t+1)}-\mathbf{Y})^T(\mathbf{X}^T\mathbf{W}^{(t+1)}-\mathbf{Y})) \\
&\quad + \rho_1\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\frac{(\mathbf{w}_{ij}^{(t+1)})^2}{2||\mathbf{w}_{ij}^{(t)}||} - ||\mathbf{w}_{ij}^{(t+1)}|| + ||\mathbf{w}_{ij}^{(t+1)}||\right) \\
&\quad + \rho_2\sum_{k=1}^{d}\left(\frac{||(\mathbf{w}^{(t+1)})^k||_2^2}{2||(\mathbf{w}^{(t)})^k||_2} - ||(\mathbf{w}^{(t+1)})^k||_2 + ||(\mathbf{w}^{(t+1)})^k||_2\right) + \rho_3\mathbf{X}\mathbf{L}\mathbf{X}^T \\
&\le Tr((\mathbf{X}^T\mathbf{W}^{(t)}-\mathbf{Y})^T(\mathbf{X}^T\mathbf{W}^{(t)}-\mathbf{Y})) \\
&\quad + \rho_1\sum_{i=1}^{n}\sum_{j=1}^{n}\left(||\mathbf{w}_{ij}^{(t)}|| + \frac{(\mathbf{w}_{ij}^{(t)})^2}{2||\mathbf{w}_{ij}^{(t+t)}||} - ||\mathbf{w}_{ij}^{(t)}||\right) \\
&\quad + \rho_2\sum_{k=1}^{d}\left(||(\mathbf{w}^{(t)})^k||_2 + \frac{||(\mathbf{w}^{(t)})^k||_2^2}{2||(\mathbf{w}^{(i)})^k||_2} - ||(\mathbf{w}^{(t)})^k||_2\right) + \rho_3\mathbf{X}\mathbf{L}\mathbf{X}^T \\
&\Rightarrow Tr((\mathbf{X}^T\mathbf{W}^{(t+1)}-\mathbf{Y})^T(\mathbf{X}^T\mathbf{W}^{(t+1)}-\mathbf{Y})) + \rho_1\sum_{i=1}^{d}\sum_{j=1}^{m}||\mathbf{w}_{ij}^{(t+1)}|| \\
&\quad + \rho_2\sum_{K=1}^{d}||(\mathbf{w}^{(t+1)})^k||_2 + \rho_3\mathbf{X}\mathbf{L}\mathbf{X}^T \\
&\le Tr((\mathbf{X}^T\mathbf{W}^{(t)}-\mathbf{Y})^T(\mathbf{X}^T\mathbf{W}^{(t)}-\mathbf{Y})) + \rho_1\sum_{i=1}^{d}\sum_{j=1}^{m}||\mathbf{w}_{ij}^{(t)}|| \\
&\quad + \rho_2\sum_{k=1}^{d}||(\mathbf{w}^{(t)})^k||_2 + \rho_3\mathbf{X}\mathbf{L}\mathbf{X}^T.
\end{aligned}$$

According to the literature [Zhu et al. 2013a, 2014], for any vectors $\mathbf{w}$ and $\mathbf{w}_0$, we have

$$||\mathbf{w}||_2 - \frac{||\mathbf{w}||_2^2}{2||\mathbf{w}_0||_2} \leq ||\mathbf{w}_0||_2 - \frac{||\mathbf{w}_0||_2^2}{2||\mathbf{w}_0||_2}.$$

So Algorithm 1 decreases the objective value in each iteration. $\mathbf{W}^{(t)}$ and $\mathbf{D}_i^{(t)}(1 \leq i \leq n)$ will satisfy the Equation (8) at the convergence. As the objective function Equation (6) is convex, the $\mathbf{W}$ satisfying Equation (6) is a global optimum solution. Therefore, Algorithm 1 will converge to the global optimum of the objective function Equation (6). □

3.4. Algorithm

In the proposed method, we first use the learned k values for kNN method, that is, the CM-kNN method, and then apply it for three different tasks, classification, regression, and missing value imputation. The pseudo of CM-kNN is presented in Algorithm 2.

ALGORITHM 2: The Pseudo of the CM-kNN Algorithm.

```
   Input: X, Y;
   Output:
   switch task do
       case 1
           Class labels;
       endsw
       case 2
           Predicted value;
       endsw
       case 3
           Imputation value;
       endsw
   endsw
 1 Normalizing X and Y;
 2 Optimizing Equation (6) to obtain the optimal solution W;
 3 Obtaining the optimal k value for test data based on W;
 4 switch task do
 5     case 1
 6         Obtaining class labels via majority rule;
 7     endsw
 8     case 2
 9         Obtaining prediction value via Equation (9);
10     endsw
11     case 3
12         Obtaining imputation value via Equation (9);
13     endsw
14 endsw
```

First, the proposed CM-kNN method employs the majority rule for predicting the class label of the test data.

Second, in both the regression task and the missing value imputation task, the proposed CM-kNN method considers that the bigger the correlation between the test data and its nearest neighbors, the larger the contribution of this nearest neighbor to the test data. Therefore, this article proposes a weighted method for both the regression and missing value imputation. Specifically, the weighted predictive value of the jth test

Table I. Simulation Configuration

Dataset	Instance	Feature	Class	Application
Climate	540	17	2	Classification
German	1,000	24	2	Classification
Blood	748	4	2	Classification
Australian	690	14	2	Classification
Seeds	210	7	3	Classification
SPECTF	267	44	2	Classification
Balance	625	4	3	Classification
Fertility	100	9	2	Classification
Gisette	13,500	5,000	2	Classification
DCCclients	30,000	24	2	Classification
Mpg	398	7	—	Regression/Imputation
Housing	506	13	—	Regression/Imputation
Bodyfat	252	14	—	Regression/Imputation
ConcreteSlump	103	7	—	Regression/Imputation
Abalone	4,177	8	—	Regression/Imputation
Pyrim	74	27	—	Regression/Imputation
Triazines	186	60	—	Regression/Imputation
Mg	1,385	6	—	Regression/Imputation
Buzz	140,000	77	—	Regression/Imputation
KEGG	53,414	24	—	Regression/Imputation

data is defined as follows:

$$predictValue\_weight = \sum_{i=1}^{n} \left(\frac{\mathbf{w}_{ij}}{\sum_{i=1}^{n} \mathbf{w}_{ij}} \times \mathbf{y}_{train(i)} \right), \tag{9}$$

where n is the number of training data points and $\mathbf{y}_{train(i)}$ represents the true value of the ith training data point.

4. EXPERIMENTS

We evaluate the proposed CM-kNN method with the state-of-the-art kNN methods on 20 datasets for three data-mining applications, classification, regression, and missing value imputation.

4.1. Experimental Setting

The used datasets were mainly downloaded from the UCI (University of California Irvine) dataset[2] and the LIBSVM (A Library for Support Vector Machines) website.[3] These datasets include all different types of data, such as a low-dimensional dataset anda high-dimensional dataset, binary datasets and multi-class datasets, imbalance datasets, and so on, and are used to evaluate the robust of the proposed method. Among of them, both the Climate dataset containing 46 positive samples and 494 negative samples and the German dataset including 700 positive samples and 300 negative samples can be regarded as imbalance datasets. It is noteworthy that there are no missing values in the original datasets, and we randomly selected some independent values to be missed according the literature on missing value imputations [Zhang et al. 2007]. Table I summarize these datasets as follows.

We employed the 10-fold cross-validation method on all methods. Specifically, we first randomly partitioned the whole dataset into 10 subsets and then selected one subset

[2]UCI Repository of Machine Learning Datasets (http://archive.ics.uci.edu).
[3]LIBSVM Data: Classification, Regression, and Multi-label (http: //www.csie.ntu.edu.tw).

for testing and the remaining 9 subsets for training. We repeated the whole process 10 times to avoid the possible bias during dataset partitioning for cross-validation. The final result was computed by averaging results from all experiments; that is, we repeated the experiments on each dataset 10 times, and regarded the average performance as the reported results. We used classification accuracy as the evaluation for the classification task. The higher the accuracy of the algorithm, the better the performance of the classification. We used correlation coefficient and root mean square error (RMSE) [Qin et al. 2007; Zhu et al. 2011] to evaluate the performance of both regression and missing value imputation. The correlation coefficient indicates the correlation between prediction and observation. Generally, the larger the correlation coefficient, the more accurate the prediction. RMSE is defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{\text{i}=1}^{n}(\mathbf{y}_i - \hat{\mathbf{y}}_i)^2}, \tag{10}$$

where n represents the number of data points, $\mathbf{y}_i$ represents the ground truth of the ith test data, and $\hat{\mathbf{y}}_i$ represents the predicted value. The smaller the RMSE, the better the algorithm.

4.2. Competing Methods

In our experiments, we selected state-of-the-art methods, including the standard kNN method, the kNN with cross-validation determined parameter k (we briefly denote it as CV-kNN), the L-kNN method [Zhang et al. 2014, 2016] (compared to CM-kNN, it doesn't consider the importance of removing the noisy data points), the LL-kNN method (compared to CM-kNN, it doesn't consider the importance of preserving the local consistency of the structures of the data), the kNN-based Applicability Domain approach (AD-kNN) [Sahigara et al. 2014], and the Large Margin Nearest Neighbor approach (LMNN) [Weinberger and Saul 2006].

—kNN: kNN is a classical classification method. It uses the k nearest neighbors to classify the test sample. In the experiments, we set $k = 5$.
—CV-kNN: CV-kNN is an improved kNN method. It uses cross-validation to determine parameter k, that is, $k = 1, 2, \ldots, 10$, and the square root of the sample size.
—The L-kNN method [Zhang et al. 2014, 2016], that is, Equation (6) with the setting $\rho_2 = 0$, on which we would like to show the importance of removing the noisy data points.
—The LL-kNN method, that is, Equation (6) with the setting $\rho_3 = 0$, on which we would like to show the importance of preserving the local consistency of the structures of the data.
—AD-kNN: AD-kNN integrates salient features of the kNN approach and adaptive kernel methods for conducting probability density estimation. Following the literature, we set the parameter k of AD-kNN with the Monte Carlo validation method by setting the maximum number of neighbors as 20 [Sahigara et al. 2014].
—LMNN: LMNN learns a Mahanalobis distance metric for k nearest neighbor (kNN) classification by semi-definite programming. The metric is trained with the goal that the k nearest neighbors always belong to the same class while examples from different classes are separated by a large margin [Weinberger and Saul 2006].

4.3. Experimental Analysis

4.3.1. Classification. We listed the classification performance (including the mean of classification accuracy in 10 iterations and the corresponding STandard Deviation (STD)) of all algorithms on the 10 datasets in Table II, the last line in the table

Table II. The Result of Classification Accuracy (Mean±STD)

Dataset	kNN	CV-kNN	L-kNN	LL-kNN	AD-kNN	LMNN	CM-kNN
Climate	0.896 ± 0.03	0.904 ± 0.03	0.927 ± 0.03	0.927 ± 0.03	0.922 ± 0.03	0.924 ± 0.03	**0.940 ± 0.03**
German	0.673 ± 0.06	0.688 ± 0.05	0.711 ± 0.06	0.710 ± 0.05	0.695 ± 0.06	0.683 ± 0.06	**0.724 ± 0.05**
Blood	0.751 ± 0.05	0.760 ± 0.06	0.782 ± 0.06	0.775 ± 0.06	0.768 ± 0.06	0.761 ± 0.05	**0.801 ± 0.06**
Australian	0.782 ± 0.05	0.800 ± 0.05	0.813 ± 0.05	0.810 ± 0.04	0.797 ± 0.05	0.794 ± 0.05	**0.842 ± 0.05**
Seeds	0.842 ± 0.07	0.862 ± 0.07	0.885 ± 0.04	0.881 ± 0.04	0.866 ± 0.06	0.859 ± 0.07	**0.890 ± 0.04**
SPECTF	0.744 ± 0.05	0.740 ± 0.04	0.781 ± 0.07	0.751 ± 0.05	0.770 ± 0.07	0.767 ± 0.08	**0.807 ± 0.08**
Balance	0.820 ± 0.04	0.835 ± 0.03	0.850 ± 0.04	0.838 ± 0.02	0.843 ± 0.04	0.838 ± 0.03	**0.850 ± 0.04**
Fertility	0.850 ± 0.08	0.877 ± 0.08	0.880 ± 0.10	0.880 ± 0.10	0.873 ± 0.14	0.870 ± 0.09	**0.880 ± 0.10**
Gisette	0.822 ± 0.04	0.832 ± 0.03	0.858 ± 0.04	0.859 ± 0.03	0.841 ± 0.03	0.845 ± 0.05	**0.894 ± 0.03**
DCCclients	0.763 ± 0.06	0.788 ± 0.05	0.808 ± 0.06	0.814 ± 0.04	0.800 ± 0.65	0.786 ± 0.06	**0.837 ± 0.03**
AVERAGE	**0.794 ± 0.05**	**0.809 ± 0.05**	**0.829 ± 0.06**	**0.825 ± 0.05**	**0.818 ± 0.12**	**0.813 ± 0.06**	**0.847 ± 0.05**

showed the average performance of each method on 10 datasets. We also reported the classification accuracy in each iteration in Figure 3.

According to Table II and Figure 3, we have the following observations:

—The proposed CM-kNN method achieved the best classification accuracy, compared to kNN, CV-kNN, L-kNN, LL-kNN, AD-kNN, and LMNN. For example, the CM-kNN method improved 5.3% (vs. kNN), 3.8% (vs. CV-kNN), 1.8% (vs. L-kNN), 2.2% (vs. LL-kNN), 2.9% (vs. AD-kNN), and 3.4% (vs. LMNN) on average for classification accuracy on 10 datasets. In addition, according to Figure 3, CM-kNN almost had higher accuracy than five comparison methods in each iteration.
—CM-kNN outperformed L-kNN because our CM-kNN used an $\ell_{2,1}$-norm regularizer to remove noisy data points. For example, our method improved by 3.6% and 2.9%, respectively, compared to L-kNN, on the Gisette dataset and the DCCclients dataset. This indicates that both the Gisette dataset and DCCclients dataset may contain noisy data points. Moreover, the Gisette dataset might have more noisy data points.
—CM-kNN outperformed LL-kNN because our CM-kNN used the LPP term to preserve the local consistency of the structures of the data. For example, the CM-kNN method improved by 3.5% and 3.2%, respectively, compared to LL-kNN, on the Gisette dataset and the Australian dataset.
—The methods (including CM-kNN, CV-kNN, L-kNN, LL-kNN, AD-kNN, and LMNN) were better than the standard kNN method. This indicates that using different k values in kNN classification (such as CM-kNN, CV-kNN, L-kNN, LL-kNN, and AD-kNN) can achieve better classification performance.

4.3.2. Regression and Missing Value Imputation. Regression is similar to missing value imputation, so we discuss them in the same section. Tables III and IV showed the results of both RMSE (mean±STD) and correlation coefficient (mean±STD) of all algorithms on 10 datasets; the last lines in the tables showed the average performance of each method on 10 datasets. Figure 4 shows the prediction performance of RMSE in each iteration and Figure 5 shows the prediction performance of correlation coefficient in each iteration.

From Tables III and IV, we found that the proposed CM-kNN had the best prediction performance, followed by L-kNN, LL-kNN, AD-kNN, CV-kNN, or LMNN and kNN. Figures 4 and 5 also intuitively showed that the proposed CM-kNN achieved the best performance in each iteration, in terms of either RMSE or correlation coefficient, on 10 datasets.

In terms of RMSE, the proposed CM-kNN on average reduced 0.392, 0.231, 0.107, 0.106, 0.153, and 0.166 on 10 datasets, compared to kNN, CV-kNN, L-kNN, LL-kNN, AD-kNN, and LMNN, respectively. In particlular, the most distinguish improvement

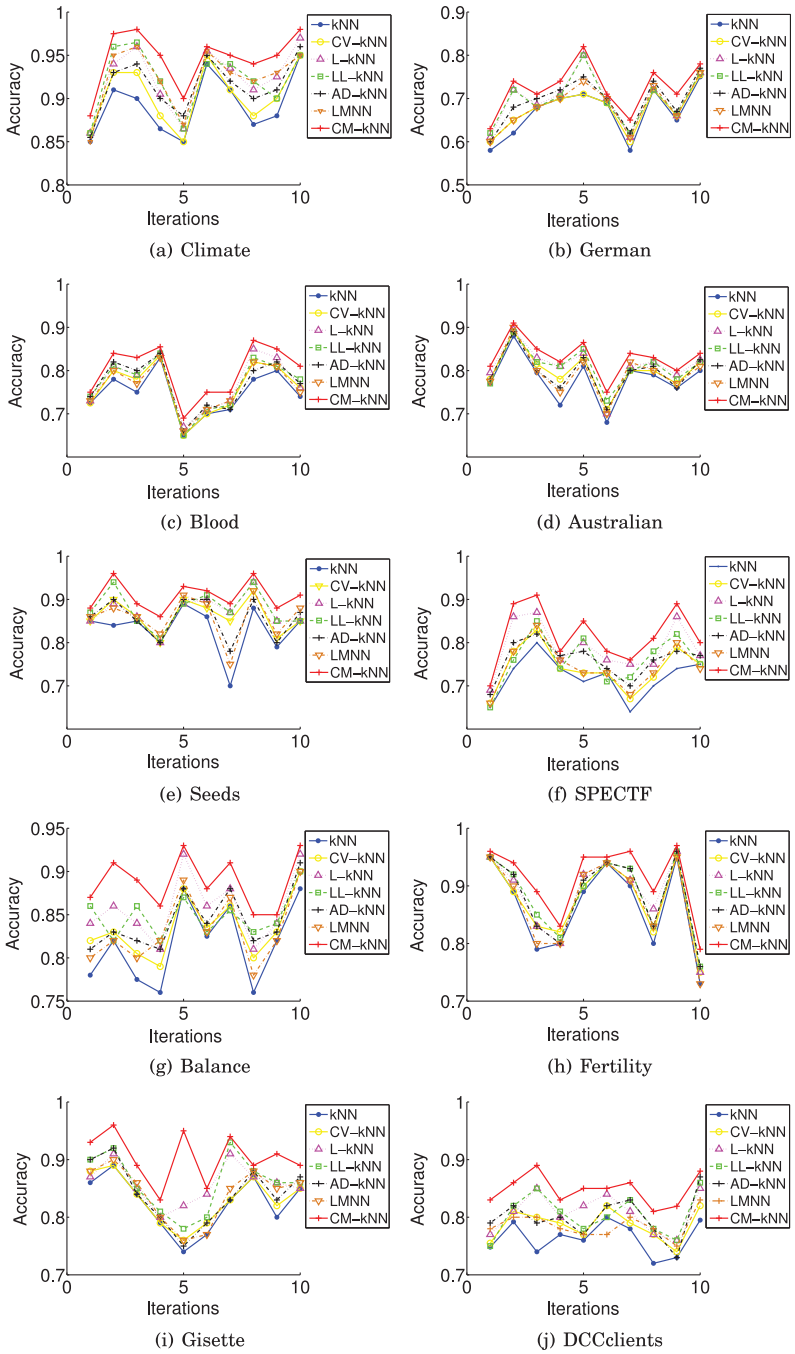

Fig. 3. Classification accuracy of each iteration in 10 iterations of all methods at 10 datasets.

Table III. The Results of RMSE (Mean±STD)

Dataset	kNN	CV-kNN	L-kNN	LL-kNN	AD-kNN	LMNN	CM-kNN
Mpg	4.129 ± 0.49	3.835 ± 0.31	3.737 ± 0.26	3.834 ± 0.39	3.788 ± 0.33	3.921 ± 0.27	**3.599 ± 0.29**
Housing	5.750 ± 1.16	5.157 ± 0.92	4.887 ± 0.84	5.067 ± 0.82	4.973 ± 0.97	5.032 ± 0.92	**4.782 ± 0.74**
Bodyfat	0.005 ± 0.00	0.004 ± 0.00	0.004 ± 0.00	0.004 ± 0.00	0.005 ± 0.00	0.005 ± 0.00	**0.002 ± 0.00**
ConcreteSlump	5.855 ± 1.18	5.453 ± 1.47	5.085 ± 1.38	4.921 ± 1.33	5.168 ± 1.27	5.334 ± 1.25	**4.708 ± 1.43**
Abalone	2.891 ± 0.17	2.785 ± 0.26	2.814 ± 0.18	2.749 ± 0.20	2.864 ± 0.21	2.794 ± 0.22	**2.698 ± 0.19**
Pyrim	0.070 ± 0.01	0.066 ± 0.08	0.061 ± 0.01	0.056 ± 0.01	0.055 ± 0.64	0.057 ± 0.05	**0.053 ± 0.01**
Triazines	0.148 ± 0.03	0.133 ± 0.19	0.129 ± 0.03	0.123 ± 0.03	0.131 ± 0.02	0.134 ± 0.03	**0.114 ± 0.03**
Mg	0.151 ± 0.01	0.168 ± 0.20	0.139 ± 0.01	0.141 ± 0.01	0.146 ± 0.01	0.148 ± 0.01	**0.135 ± 0.01**
Buzz	1.751 ± 0.34	1.585 ± 0.32	1.126 ± 0.42	1.093 ± 0.25	1.287 ± 0.32	1.105 ± 0.22	**0.903 ± 0.24**
KEGG	0.377 ± 0.78	0.324 ± 0.49	0.296 ± 0.89	0.282 ± 0.76	0.322 ± 0.72	0.335 ± 0.80	**0.211 ± 0.71**
AVERAGE	**2.113 ± 0.42**	**1.951 ± 0.42**	**1.828 ± 0.04**	**1.827 ± 0.38**	**1.874± 0.45**	**1.887 ± 0.38**	**1.721 ± 0.37**

Table IV. The Results of Correlation Coefficient (Mean±STD)

Dataset	kNN	CV-kNN	L-kNN	LL-kNN	AD-kNN	LMNN	CM-kNN
Mpg	0.834 ± 0.04	0.846 ± 0.04	0.862 ± 0.03	0.860 ± 0.03	0.854 ± 0.03	0.842 ± 0.03	**0.880 ± 0.02**
Housing	0.766 ± 0.10	0.809 ± 0.09	0.843 ± 0.06	0.830 ± 0.06	0.821 ± 0.05	0.825 ± 0.06	**0.859 ± 0.05**
Bodyfat	0.976 ± 0.01	0.982 ± 0.01	0.985 ± 0.01	0.983 ± 0.01	0.980 ± 0.01	0.980 ± 0.01	**0.993 ± 0.01**
ConcreteSlump	0.698 ± 0.07	0.736 ± 0.06	0.750 ± 0.07	0.779 ± 0.06	0.750 ± 0.07	0.730 ± 0.07	**0.792 ± 0.06**
Abalone	0.701 ± 0.02	0.721 ± 0.04	0.737 ± 0.02	0.742 ± 0.02	0.734 ± 0.02	0.725 ± 0.02	**0.751 ± 0.01**
Pyrim	0.813 ± 0.08	0.847 ± 0.09	0.888 ± 0.05	0.908 ± 0.04	0.862 ± 0.09	0.870 ± 0.08	**0.917 ± 0.03**
Triazines	0.531 ± 0.07	0.650 ± 0.12	0.676 ± 0.10	0.709 ± 0.09	0.680 ± 0.07	0.672 ± 0.03	**0.754 ± 0.08**
Mg	0.722 ± 0.06	0.733 ± 0.06	0.747 ± 0.04	0.746 ± 0.05	0.741 ± 0.04	0.740 ± 0.05	**0.767 ± 0.04**
Buzz	0.756 ± 0.07	0.790 ± 0.06	0.815 ± 0.08	0.824 ± 0.07	0.813 ± 0.73	0.805 ± 0.06	**0.847 ± 0.07**
KEGG	0.809 ± 0.23	0.834 ± 0.24	0.875 ± 0.35	0.860 ± 0.21	0.872 ± 0.25	0.852 ± 0.32	**0.896 ± 0.20**
AVERAGE	**0.761 ± 0.08**	**0.795 ± 0.08**	**0.818 ± 0.08**	**0.816 ± 0.06**	**0.811 ± 0.14**	**0.804 ± 0.07**	**0.846 ± 0.06**

is on the ConcreteSlump dataset, in which the proposed CM-kNN reduced 1.147 (vs. the kNN method), 0.745 (vs. CV-kNN), 0.377 (vs. L-kNN), 0.213 (vs. LL-kNN), 0.46 (vs. AD-kNN), and 0.626 (vs. LMNN).

Regarding the evaluation of correlation coefficient, our CM-kNN averagely improved by 8.5%, 5.08%, 2.78%, 2.15%, 3.49%, and 4.15% on 10 datasets, compared to kNN, CV-kNN, L-kNN, LL-kNN, AD-kNN, and LMNN. In particular, CM-kNN achieved the most remarkable improvement (22.4%, 10.4%, 7.8%, 4.5%, 7.4%, and 8.3% on the \textit{Tr}iazines dataset), compared to kNN, CV-kNN, L-kNN, LL-kNN, AD-kNN, and LMNN, respectively.

In a nutshell, in the three tasks, the proposed CM-kNN showed a distinct difference versus the competing methods, according to the last row of Tables II–IV, which represent the averaged result over 10 datasets of the competing methods. Specifically, the proposed CM-kNN outperformed L-kNN because it considers removing the noisy data points during the process of selecting k values for test data, and outperformed LL-kNN because it considers preserving the local consistency of the structures of the data. Moreover, the methods (including CM-kNN, CV-kNN, L-kNN, LL-kNN, and AD-kNN) are superior to kNN, which indicates that the methods tha used a varied k value are preferred in real applications.

5. CONCLUSION

In this article, we have proposed a new kNN method, referred to as CM-kNN, for the applications of classification, regression, and missing data imputation. Compared to the conventional kNN classification, there are two significant improvements in our CM-kNN approach. First, the proposed method learned different k values for each test data point by making the best use of prior knowledge of the data. Second, the proposed method is robust to the noisy datasets. Experiments on real datasets have

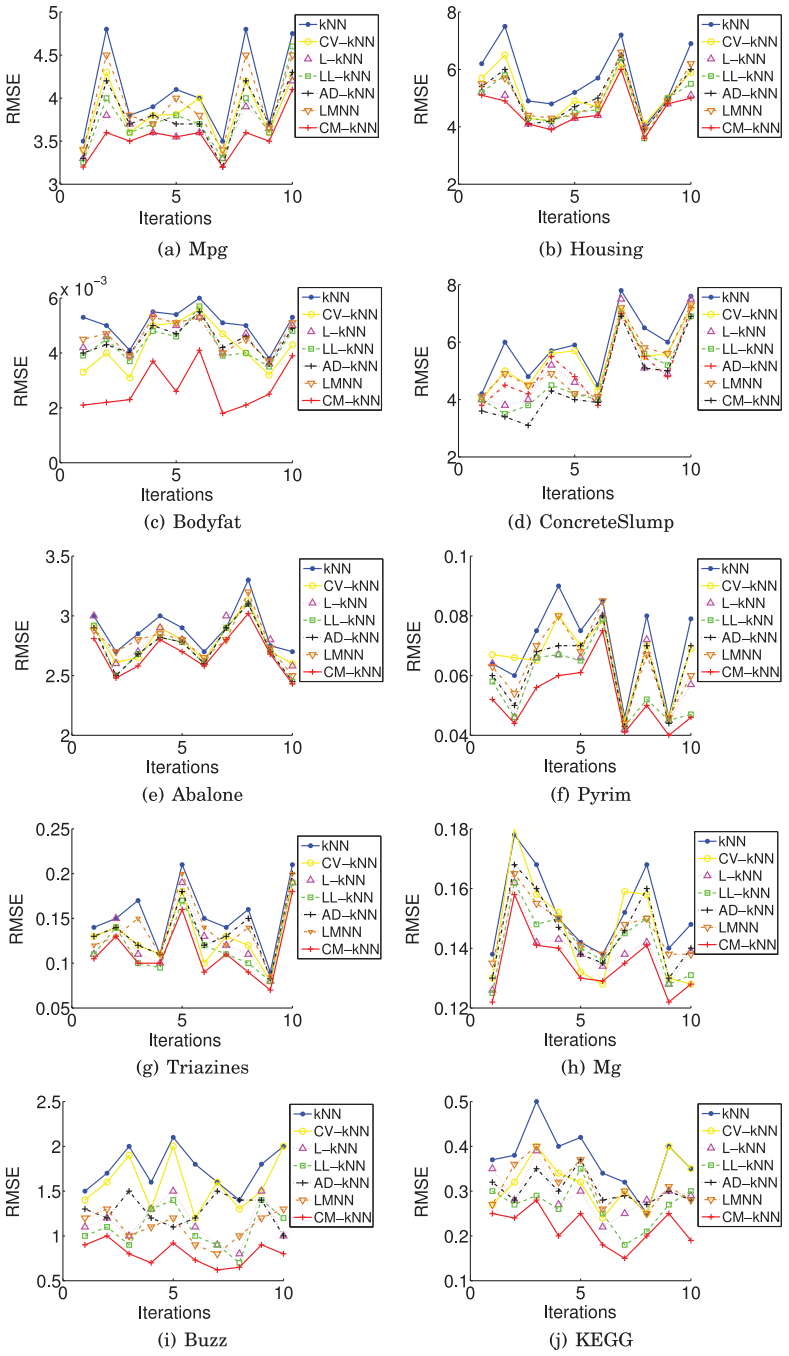

Fig. 4. RMSE of each iteration in 10 iterations of all methods at 10 datasets.

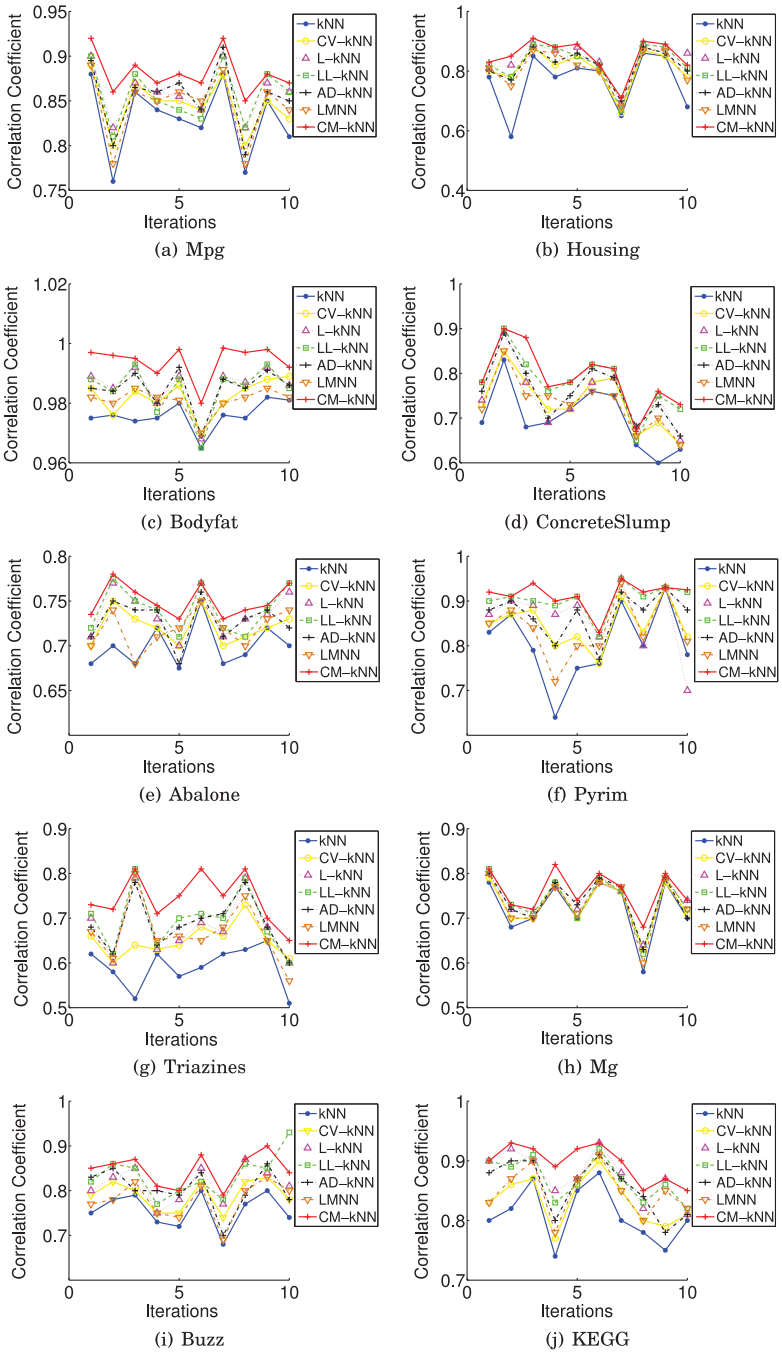

Fig. 5. Correlation Coefficient of each iteration in 10 iterations of all methods at 10 datasets.

demonstrated that, compared with extant kNN methods, the proposed method achieved high accuracy and efficiency in the applications of classification, regression, and missing data imputation.

In the future work, we will focus on designing a nonlinear transformation matrix to learn the correlation between test data and training data.

REFERENCES

Enrico Blanzieri and Farid Melgani. 2008. Nearest neighbor classification of remote sensing images with the maximal margin principle. *IEEE Trans. Geosci. Remote Sens.* 46, 6 (2008), 1804–1811.

Jiahua Chen and Jun Shao. 2001. Jackknife variance estimation for nearest-neighbor imputation. *J. Am. Statist. Assoc.* 96, 453 (2001), 260–269.

Xiai Chen, Zhi Han, Yao Wang, Yandong Tang, and Haibin Yu. 2016. Nonconvex plus quadratic penalized low-rank and sparse decomposition for noisy image alignment. *Sci. Chin. Infor. Sci.* 5 (2016), 1–13.

Debo Cheng, Shichao Zhang, Xingyi Liu, Ke Sun, and Ming Zong. 2015. Feature selection by combining subspace learning with sparse representation. *Multimedia Syst.* (2015), 1–7.

Ingrid Daubechies, Ronald DeVore, Massimo Fornasier, and C. Sinan Güntürk. 2010. Iteratively reweighted least squares minimization for sparse recovery. *Commun. Pure Appl. Math.* 63, 1 (2010), 1–38.

Yongsheng Dong, Dacheng Tao, and Xuelong Li. 2015b. Nonnegative multiresolution representation-based texture image classification. *ACM Trans. Intell. Syst. Technol.* 7, 1 (2015), 4.

Zhen Dong, Wei Liang, Yuwei Wu, Mingtao Pei, and Yunde Jia. 2015a. Nonnegative correlation coding for image classification. *Sci. Chin. Infor. Sci.* 59, 1 (2015), 1–14.

Jianping Fan, Jinye Peng, Ling Gao, and Ning Zhou. 2015. Hierarchical learning of tree classifiers for large-scale plant species identification. *IEEE Trans. Image Process.* 24, 11 (2015), 4172–84.

Pedro J. García-Laencina, José-Luis Sancho-Gómez, Aníbal R. Figueiras-Vidal, and Michel Verleysen. 2009. K nearest neighbours with mutual information for simultaneous classification and missing data imputation. *Neurocomputing* 72, 7 (2009), 1483–1493.

Mohammad Ghasemi Hamed, Mathieu Serrurier, and Nicolas Durand. 2012. Possibilistic knn regression using tolerance intervals. In *Advances in Computational Intelligence*. 410–419.

Xiaofei He, Chiyuan Zhang, Lijun Zhang, and Xuelong Li. 2016. A-optimal projection for image representation. *IEEE Trans. Pattern Anal. Mach. Intell.* 38, 5 (2016), 1009–1015.

Boyu Li, Yun Wen Chen, and Yan Qiu Chen. 2008. The nearest neighbor algorithm of local probability centers. *IEEE Trans. Syst. Man Cybernet. B* 38, 1 (2008), 141–154.

Xuelong Li, Qun Guo, and Xiaoqiang Lu. 2016. Spatiotemporal statistics for video quality assessment. *IEEE Trans. Image Process.* 25, 7 (2016), 3329–3342.

Xuelong Li, Lichao Mou, and Xiaoqiang Lu. 2015. Scene parsing from an MAP perspective. *IEEE Trans. Cybernet.* 45, 9 (2015), 1876–1886.

Xuelong Li and Yanwei Pang. 2009. Deterministic column-based matrix decomposition. *IEEE Trans. Knowl. Data Eng.* 22, 1 (2009), 145–149.

Xuelong Li, Zhigang Wang, and Xiaoqiang Lu. 2016. Surveillance video synopsis via scaling down objects. *IEEE Trans. Image Process.* 25, 2 (2016), 740–755.

Fan Liu, Jinhui Tang, Yan Song, Liyan Zhang, and Zhenmin Tang. 2015. Local structure-based sparse representation for face recognition. *ACM Trans. Intell. Syst. Technol.* 7, 1 (2015), 2.

Chen Luo, Jia Zeng, Mingxuan Yuan, Wenyuan Dai, and Qiang Yang. 2016. Telco user activity level prediction with massive mobile broadband data. *ACM Trans. Intell. Syst. Technol.* 7, 4 (2016), 63.

Minnan Luo, Fuchun Sun, and Huaping Liu. 2014. Joint block structure sparse representation for multi-input–multi-output (MIMO) T–S fuzzy system identification. *IEEE Trans. Fuzzy Syst.* 22, 6 (2014), 1387–1400.

Tristan Mary-Huard and Stephane Robin. 2009. Tailored aggregation for classification. *IEEE Trans. Pattern Anal. Mach. Intell.* 31, 11 (2009), 2098–2105.

Phayung Meesad and Kairung Hengpraprohm. 2008. Combination of knn-based feature selection and knnbased missing-value imputation of microarray data. In *ICICIC*. 341–341.

Amir Navot, Lavi Shpigelman, Naftali Tishby, and Eilon Vaadia. 2006. Nearest neighbor based feature selection for regression and its application to neural activity. (2006).

Karl S. Ni and Truong Q. Nguyen. 2009. An adaptable-nearest neighbors algorithm for MMSE image interpolation. *IEEE Trans. Image Process.* 18, 9 (2009), 1976–1987.

X. Niyogi. 2004. Locality preserving projections. In *NIPS*, Vol. 16. 153.

Yongsong Qin, Shichao Zhang, Xiaofeng Zhu, Jilian Zhang, and Chengqi Zhang. 2007. Semi-parametric optimization for missing data imputation. *Appl. Intell.* 27, 1 (2007), 79–88.

F. Sahigara, D. Ballabio, R. Todeschini, and V. Consonni. 2014. Assessing the validity of QSARs for ready biodegradability of chemicals: An applicability domain perspective. *Curr. Comput.- Aid. Drug Des.* 10, 10 (2014), 137–147.

Ziqiang Shi, Jiqing Han, and Tieran Zheng. 2013. Audio classification with low-rank matrix representation features. *ACM Trans. Intell. Syst. Technol.* 5, 1 (2013), 15.

Yang Song, Jian Huang, Ding Zhou, Hongyuan Zha, and C. Lee Giles. 2007. Iknn: Informative k-nearest neighbor pattern classification. In *PKDD*. 248–264.

Jimeng Sun and Chandan K. Reddy. 2013. Big data analytics for healthcare. In *KDD*. 1525–1525.

Yu Sun, Jianzhong Qi, Yu Zheng, Zhang, and Rui. 2015. K-nearest neighbor temporal aggregate queries. *Inproceedings* (2015).

Lu An Tang, Yu Zheng, Xing Xie, Jing Yuan, Xiao Yu, and Jiawei Han. 2011. *Retrieving k-Nearest Neighboring Trajectories by a Set of Point Locations*. Springer, Berlin, 223–241 pages.

Pascal Vincent and Yoshua Bengio. 2001. K-local hyperplane and convex distance nearest neighbor algorithms. In *NIPS*. 985–992.

Hui Wang. 2006. Nearest neighbors by neighborhood counting. *IEEE Trans. Pattern Anal. Mach. Intell.* 28, 6 (2006), 942–953.

Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel time estimation of a path using sparse trajectories. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 25–34.

Kilian Q. Weinberger and Lawrence K. Saul. 2006. Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.* 10, 1 (2006), 207–244.

Xindong Wu, Huanhuan Chen, Gongqing Wu, Jun Liu, Qinghua Zheng, Xiaofeng He, Aoying Zhou, Zhong-Qiu Zhao, Bifang Wei, Ming Gao, and others. 2015. Knowledge engineering with big data. *IEEE Intell. Syst.* 30, 5 (2015), 46–55.

Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, S. Yu Philip, and others. 2008. Top 10 algorithms in data mining. *Knowl. Infor. Syst.* 14, 1 (2008), 1–37.

Xindong Wu, Xingquan Zhu, Gong-Qing Wu, and Wei Ding. 2014. Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering* 26, 1 (2014), 97–107.

Chunlei Yang, Jialie Shen, Jinye Peng, and Jianping Fan. 2012. Image collection summarization via dictionary learning for sparse representation. *Pattern Recogn.* 46, 3 (2012), 948–961.

Zizhen Yao and Walter L. Ruzzo. 2006. A regression-based K nearest neighbor algorithm for gene function prediction from heterogeneous data. *BMC Bioinfor.* 7, Suppl. 1 (2006), S11.

Renzhen Ye and Xuelong Li. 2016. Compact structure hashing via sparse and similarity preserving embedding. *IEEE Trans. Cybernet.* 46, 3 (2016), 718–729.

Chengqi Zhang, Yongsong Qin, Xiaofeng Zhu, Jilian Zhang, and Shichao Zhang. 2006. Clustering-based missing value imputation for data preprocessing. In *2006 4th IEEE International Conference on Industrial Informatics*. 1081–1086.

Chengqi Zhang, Xiaofeng Zhu, Jilian Zhang, Yongsong Qin, and Shichao Zhang. 2007. GBKII: An imputation method for missing values. In *PAKDD*. 1080–1087.

Shizhao Zhang. 2010. KNN-CF approach: Incorporating certainty factor to kNN classification. *IEEE Intell. Infor. Bull.* 11, 1 (2010), 24–33.

Shichao Zhang. 2011. Shell-neighbor method and its application in missing data imputation. *Appl. Intell.* 35, 1 (2011), 123–133.

Shichao Zhang, Debo Cheng, Ming Zong, and Lianli Gao. 2016. Self-representation nearest neighbor search for classification. *Neurocomputing* 195 (2016), 137–142.

Shichao Zhang, Ming Zong, Ke Sun, Yue Liu, and Debo Cheng. 2014. Efficient kNN algorithm based on graph sparse reconstruction. In *ADMA*. 356–369.

Yuejie Zhang, Lei Cen, Cheng Jin, Xiangyang Xue, and Jianping Fan. 2011. Learning inter-related statistical query translation models for English-Chinese bi-directional CLIR. In *International Joint Conference on Artificial Intelligence*. 1915–1920.

Xiaofeng Zhu, Zi Huang, Hong Cheng, Jiangtao Cui, and Heng Tao Shen. 2013a. Sparse hashing for fast multimedia search. *ACM Trans. Infor. Syst.* 31, 2 (2013), 9.

Xiaofeng Zhu, Zi Huang, Yang Yang, Heng Tao Shen, Changsheng Xu, and Jiebo Luo. 2013b. Self-taught dimensionality reduction on the high-dimensional small-sized data. *Pattern Recogn.* 46, 1 (2013), 215–229.

Xiaofeng Zhu, Xuelong Li, and Shichao Zhang. 2016a. Block-row sparse multiview multilabel learning for image classification. *IEEE Trans. Cybernet.* 46, 2 (2016), 450–461.

Xiaofeng Zhu, Xuelong Li, Shichao Zhang, Chunhua Ju, and Xindong Wu. 2016b. Robust joint graph sparse coding for unsupervised spectral feature selection. *IEEE Trans. Neur. Netw. Learn. Syst.* (2016).

Xiaofeng Zhu, Heung-Il Suk, and Dinggang Shen. 2014. Matrix-similarity based loss function and feature selection for alzheimer's disease diagnosis. In *CVPR*. 3089–3096.

Xiaofeng Zhu, Shichao Zhang, Zhi Jin, Zili Zhang, and Zhuoming Xu. 2011. Missing value estimation for mixed-attribute data sets. *IEEE Trans. Knowl. Data Eng.* 23, 1 (2011), 110–121.

Xiaofeng Zhu, Shichao Zhang, Jilian Zhang, and Chengqi Zhang. 2007. Cost-sensitive imputing missing values with ordering. In *AAAI*. 1922–1923.

Received May 2016; revised August 2016; accepted August 2016