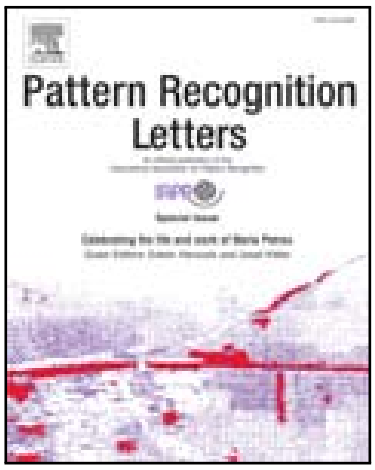

A novel kNN Algorithm with Data-driven k Parameter Computation

Shichao Zhang, Debo Cheng, Zhenyun Deng, Ming Zong, Xuelian Deng

PII: S0167-8655(17)30356-2
DOI: 10.1016/j.patrec.2017.09.036
Reference: PATREC 6948

To appear in: *Pattern Recognition Letters*

Received date: 1 June 2017
Revised date: 1 September 2017
Accepted date: 25 September 2017

Please cite this article as: Shichao Zhang, Debo Cheng, Zhenyun Deng, Ming Zong, Xuelian Deng, A novel kNN Algorithm with Data-driven k Parameter Computation, *Pattern Recognition Letters* (2017), doi: 10.1016/j.patrec.2017.09.036

Highlights

- Existing kNN approximate prediction algorithm is with a fixed k value for the whole problem space. The S-kNN algorithm identifies an optimal k value for each test sample, *i.e.,* the parameter k can be different for different test samples.
- Different from conventional Least Absolute Shrinkage and Selection Operator (LASSO), our approach takes the local structures of samples into account.
- This paper proposes a novel optimization method to solve the designed objective function.

Pattern Recognition Letters
journal homepage: www.elsevier.com

A novel *k*NN Algorithm with Data-driven *k* Parameter Computation

Shichao Zhang[a], Debo Cheng[a,b], Zhenyun Deng[a], Ming Zong[c], Xuelian Deng[d]

[a]*Guangxi Key Lab of Multi-source Information Mining & Security, Guangxi Normal University, Guilin, Guangxi, China.*
[b]*Information Technology and Mathematical Sciences, University of South Australia, Adelaide, Australia*
[c]*Institute of Natural and Mathematical Sciences, Massey University, Auckland, New Zealand*
[d]*College of Public Health and Management, Guangxi University of Chinese Medicine, Nanning, Guangxi, China*

ABSTRACT

This paper studies an example-driven *k*-parameter computation that identifies different *k* values for different test samples in *k*NN prediction applications, such as classification, regression and missing data imputation. This is carried out with reconstructing a sparse coefficient matrix between test samples and training data. In the reconstruction process, an ℓ_1-norm regularization is employed to generate an element-wise sparsity coefficient matrix, and an LPP (Locality Preserving Projection) regularization is adopted to keep the local structures of data for achieving the efficiency. Further, with the learnt *k* value, *k*NN approach is applied to classification, regression and missing data imputation. We experimentally evaluate the proposed approach with 20 real datasets, and show that our algorithm is much better than previous *k*NN algorithms in terms of data mining tasks, such as classification, regression and missing value imputation.

1. Introduction

The *k*NN (*k* Nearest Neighbors) algorithm is a non-parametric, or an instance-based, or a lazy method, and has been regarded as one of the simplest method in data mining and machine learning (Qin et al., 2013)(Zhang et al., 2017a)(Zhang et al., 2017b). The principle of *k*NN algorithm is that the most similar samples belonging to the same class have high probability. Generally, the *k*NN algorithm first finds *k* nearest neighbors of a query in training dataset, and then predicts the query with the major class in the *k* nearest neighbors. Therefore, it has recently been selected as one of top 10 algorithms in data mining (Wu et al., 2008).

As well known, *k*NN algorithm is often sensitive to the selection of the *k* value. Although efforts have been focused on this topic for a long time, setting *k* value is still very challengeable in *k*NN algorithm (Zhang et al., 2010). Lall and Sharama mentioned that setting a suitable *k* should satisfy $k = \sqrt{n}$ for training datasets with sample size larger than 100 (Lall and Sharma, 1996). Ghosh investigated a Bayesian method for guiding us well in selecting k mainly (Ghosh, 2006). Mitra *et al.* thought it is without any theories to guarantee that $k = \lfloor \sqrt{n} \rfloor$ is suitable for each test sample. Liu *et al.* pointed out, it has been proved that a fixed *k* value is not suitable for many test samples in a given training dataset (Liu et al., 2010).

We now illustrate the above limitations of *k*NN algorithm with a fixed *k* value in Figures 1 and 2. Figure 1 is an example of a binary classification task, where the classes training samples are marked as '+' and '-' respectively, and the labels of test samples are marked with the symbol '?'. Figure 2 is an example of a missing data imputation, where the symbol '?' stands for data with missing values.

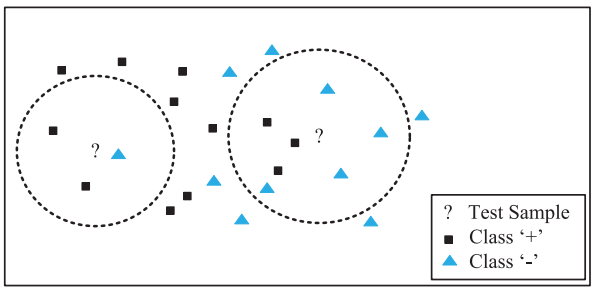

Fig. 1. Training examples for *k*NN classification

**Corresponding author: Shichao Zhang
e-mail: zhangsc@mailbox.gxnu.edu.cn (Shichao Zhang)

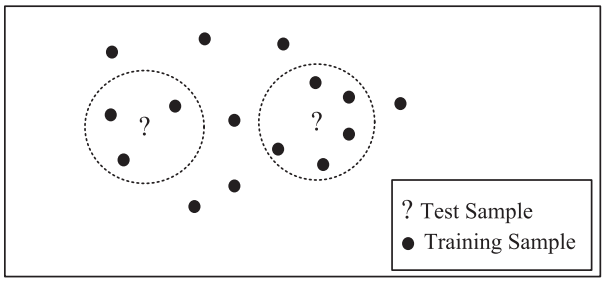

Fig. 2. Training examples for *k*NN regression/ missing value imputation.

In Figure 1, when setting k=5 for the kNN algorithm, there are two test samples that are predicted to '+' class according to the kNN rule. And the left test sample is incorrectly predicted. When setting k=1, the test samples are both incorrectly predicted. From the training examples, it is reasonable to take k=3 and k=7 for the left test sample and the right one, respectively.

For a similar scenario of missing value imputation in Figure 2, the right and left test samples should be assigned different k, *i.e.,* k=3 and k=5 respectively. This scenario also indicates that different test samples should take different numbers of nearest neighbors in real kNN prediction applications. That is to say, setting a fixed constant for all test samples may often lead to low prediction rates in real classification applications.

Motivated by the above facts, this paper proposes a k-parameter computation for kNN approximate prediction based on Sparse learning, called S-kNN [1] (Cheng et al., 2014). The k-parameter computation can identify different k values for predicting different test samples with kNN algorithm. This is carried out by reconstructing a sparse coefficient matrix between test samples and training data (Zhu et al., 2017d)(Zhu et al., 2016b). With the matrix, an optimal k value can be obtained for each test sample one by one. In the reconstruction, a least square loss function is applied to achieve the minimal reconstruction error, and an norm regularization is utilized to result in the element-wise sparsity (*i.e.,* the sparse codes appear in the element of the coefficient matrix) for generating various k values for different test samples (Zhu et al., 2017c)(Zhu et al., 2017b). We also employ the Locality Preserving Projection (LPP) regularization to preserve the local structures of data during the reconstruction process, aiming to further improve the reconstruction performance (He and Niyogi, 2003)(Hu et al., 2017). The proposed S-kNN algorithm is experimentally evaluated against data mining tasks, such as classification, regression and missing value imputation. Comparing with previous kNN algorithms, the main contributions are as follows.

- Existing kNN approximate prediction algorithm is with a fixed k value for the whole problem space. The S-kNN algorithm identifies an optimal k value for each test sample, *i.e.,* the parameter k can be different for different test samples.
- Different from conventional Least Absolute Shrinkage and Selection Operator (LASSO) (Kang and Cho, 2008; Tibshirani, 1996), our approach takes the local structures of samples into account.
- This paper proposes a novel optimization method to solve the designed objective function.

The remainder of the paper is organized as follows. Section 2 briefly reviews related kNN methods for classification, regression and missing value imputation. Section 3 is the main body of our S-kNN method. The proposed method is experimentally evaluated with real datasets in Section 4. Finally, this research is concluded in Section 5.

2. Related Work

The study of kNN method has been a hot research topic in data mining and machine learning since the algorithm was proposed in 1967 (Cover and Hart, 1967). In this section, we briefly review the applications of kNN algorithm in data mining tasks, such as classification, regression and missing value imputation.

2.1. Classification

KNN classification algorithm first selects k closest samples(*i.e., k* nearest neighbors) for a test sample from all the training samples, and then predicts the test sample with a simple classifier, *e.g.,* majority classification rule. Liu *et al.* designed a new anomaly removal algorithm under the framework of kNN classification (Liu et al., 2010), which adopts mutual nearest neighbors whose advantage is that pseudo nearest neighbors can be identified instead of k nearest neighbors to determine the class labels of unknown samples. Weinberger *et al.* used semi-definite programming to learn a Mahanalobis distance metric for kNN classification, and adopted the target that k nearest neighbors always belong to the same class to optimize the measure metric, which samples from different classes are separated by a large margin (Weinberger and Saul, 2009). Moreover, Goldberger *et al.* proposed a novel non-parametric kNN classification that learns a new quadratic distance metric and calls neighborhood component analysis (NCA) method (Goldberger et al., 2004). This method focuses on the learned distance to be low-rank, so as to saving the storage and search costs. Jamshidi and Kaburlasos proposed an effective synergy of the Intervals' Number k-nearest neighbor classifier, and the gravitational search algorithm (GSA) for stochastic search and optimization (Jamshidi and Kaburlasos, 2014). Saini *et al.* presented an application of k-Nearest Neighbor (kNN) algorithm as a classifier for detection of QRS-complex in ECG (Saini et al., 2013). This algorithm uses a digital band-pass filter to reduce the interference present in ECG false detection signal. For avoiding the influence of k value, Varmuza *et al.* used the repeated double cross validation method to search an optimum k for k nearest neighbor classification (Varmuza et al., 2014).

[1] In this manuscript, we rewrote the parts (*i.e.,* Section 1 and Section 4) and added the parts (*i.e.,* Section 2.1, Section 2.2, Section 2.3, Section 3.3, and Section 3.4), compared to our former conference version.

2.2. Regression

The *k*NN regression has been widely used and studied for many years in pattern recognition and data mining. In regression analysis, Burba *et al.* utilized kernel estimator based some asymptotic properties of the *k*NN to improving the performance of *k*NN regression (Burba et al., 2009). Moreover, the purpose of their work utilized local adaptive bandwidth to study the non-parametric *k*NN algorithm. Ferraty and Vieu utilized the functional version of the Nadaraya-Watson kernel type estimator to construct the non-parametric characteristics of *k*NN algorithm for estimation, classification and discrimination on high dimensional data (Ferraty and Vieu, 2006). In the theory of *k*NN algorithm, Mack studied the L^2 convergence and the asymptotic distribution (Mack, 1981), and Devroye proved the strong consistency and the uniform convergence of *k*NN algorithm (Devroye et al., 1981). Hu *et al.* proposed a data-driven method for the battery capacity estimation, and used a non-linear kernel regression model based on the *k*NN to capture the dependency of the capacity on the features. This work also utilizes the adaptation of particle swarm optimizations to find the feature weights for the *k*NN regression model (Hu et al., 2014). Goyal *et al.* took the interrelatedness of these metrics into account and statistically established the extent to improve the explanatory power of multiple linear regression. And then they conducted stepwise regression to identify influential metrics to avoid over fitting of data, and proposes suitability of *k*NN regression in the development of fault prediction model (Goyal et al., 2014). Cycle time of wafer lots for semiconductor fab was a critical task, therefore, Ni *et al.* combined the particle swarm optimization with a Gaussian mutation operator and a simulated weight of the features for *k*NN regression, and then used it to predict the cycle time of wafer fab (Ni et al., 2012). Zhou proposed semi-supervised regression with co-training (Zhou and Li, 2005), which employed two *k*NN regressors with different distance metrics, each of which labeled the unlabeled data for the others during the learning process.

2.3. Missing value imputation

In real data mining applications, missing data is often inevitable. There are many techniques to deal with missing data that can mainly be divided into two categories, missing instance deletion and missing value imputation (Qin et al., 2007)(Zhang et al., 2011)(Zhang, 2011), in which the *k*NN imputation is an important approximate solution in real applications. For instance, Zhang *et al.* utilized grey-based distance measure to replace Euclidean distance in conventional *k*NN algorithm, which can improve the performance of *k*NN imputation algorithm, referred Grey-Based *k*NN Iteration Imputation (GBKII) (Zhang et al., 2007). It is an instance-based and a non-parametric imputation algorithm. Chen and Shao proposed the naive jackknife variance estimators that treat imputed values as observed data produces serious underestimation based on nearest-neighbor imputation. This method is a nonparametric variance estimation technique, asymptotically unbiased and consistent for the sample means (Chen and Shao, 2001). Meesad and Hengpraprohm first proposed a methodology to impute missing values in microarray, which combines KNN-based feature selection and KNN-based imputation, and then estimated missing values by the *k*NN algorithm (Meesad and Hengpraprohm, 2008). Recently, Zhang developed a kerne-based missing value imputation algorithm that takes the attribute correlations within data, so as to making optimal statistical parameters: mean, distribution function after missing-data are imputed. Thus, the method improves the nearest neighbors of missing data the performance of the previous *k*NN algorithm (Zhang et al., 2006). Hoef *et al.* proposed two methods *i.e.,* the spatial linear model and *k* nearest neighbor for mapping and estimating totals. In order to enhance prediction and understanding, they employed a Bayesian approach account for the covariance parameters (Hoef and Hailemariam, 2013). For adjusting the estimated missing values to the overall size of the compositional parts of the neighbors, Hron *et al.* proposed that the distance utilizes the *k* nearest neighbor to procedure based on the Aitchison distance and uses an iterative model-based imputation technique to search the result of the proposed *k*-nearest neighbor procedure (Hron et al., 2010).

From the above subsections, existing *k*NN methods use a fixed *k* value for the whole problem space and often lead to poor predictions.

3. Proposed Method

In this section, we introduce some basic concepts used in our proposed method. And then describe the S-*k*NN method. Finally, we improve the S-*k*NN method by optimizing the objective function.

3.1. Notation

Throughout the paper, we denote matrices as boldface uppercase letters, vectors as boldface lowercase letters and scalars as normal italic letters. 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. The Frobenius norm, the ℓ_2-norm and the ℓ_1-norm are represented as $\|\mathbf{X}\|_F = \sqrt{\sum_i \|\mathbf{x}^i\|_2^2} = \sqrt{\sum_j \|\mathbf{x}_j\|_2^2}$, $\|\mathbf{x}_j\|_2 = \sqrt{\sum_i^n x_{i,j}^2}$, and $\|\mathbf{X}\|_1 = \sum_i \sum_j |x_{i,j}|$, respectively. The transpose operator, trace operator and inverse of a matrix $\mathbf{X}$ are expressed as $\mathbf{X}^T$, $tr(\mathbf{X})$, and $\mathbf{X}^{-1}$, respectively.

3.2. Reconstruction and LPP (locality preserving projection)

Given $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n \in R^{n\times d}$ and $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^m \in R^{m\times d}$, where n, m, and d stand for the numbers of training samples, test samples, and feature dimension, respectively. For obtaining the reconstruction coefficient matrix $\mathbf{W} \in R^{n\times m}$, we reconstruct the test samples with the training samples. Thus, the objective function defined as follows (Yager and Petry, 2014).

$$\arg\min_{\mathbf{W}} \sum\nolimits_i \left\| \mathbf{w}_i{}^T \mathbf{x}_i - \mathbf{y}_i \right\| \tag{1}$$

where $w_{i,j}$ is utilized to measure the correlation among $\mathbf{y}_i$ and training samples $\mathbf{x}_j$. The larger the value of $w_{i,j}$ is, the more relevant between *i*th test sample and *j*th training sample is. In particular, the case of $w_{i,j} = 0$ denotes that there is uncorrelation between $\mathbf{y}_i$ and $\mathbf{x}_j$.

To carry out the reconstruction, LPP(locality preserving projection) (He and Niyogi, 2003) is applied to obtain an optimal linear transformation $\mathbf{W}$. Lpp technique can preserve the local structure of original data in the new space, *i.e.,* $\mathbf{W}$ converts the high-dimensional data $\mathbf{X}$ into the low-dimensional data $\mathbf{Y}$ with the following definition:

$$\mathbf{y}_j = \mathbf{W}^T\mathbf{x}_i, i = 1, 2, \cdots, n \tag{2}$$

To this end, the objective function of LPP can be defined as follows:

$$\begin{aligned}&\min_{\mathbf{W}} \sum_{i,j} (\mathbf{W}^T\mathbf{x}_i - \mathbf{W}^T\mathbf{x}_j)^2 s_{i,j}\\ &= \min_{\mathbf{W}} \sum_{i,j} (\mathbf{W}^T\mathbf{x}_i - \mathbf{W}^T\mathbf{x}_j)^2 \mathbf{S}\end{aligned} \tag{3}$$

where $\mathbf{S}$ is the weight matrix and each element of $\mathbf{S}$ is defined by a heat kernel $s_{i,j} = exp(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma})$ [2]. σ is a tuning parameter. Without loss of generality, we set $\sigma = 1$ in our experiments which the justification for this choice of weights can be traced back to (Belkin and Niyogi, 2001).

By plugging Eq.(2) into Eq.(3), and some algebraic transformation operations, we obtain:

$$\begin{aligned}&\frac{1}{2}\sum_{i,j} (\mathbf{W}^T\mathbf{x}_i - \mathbf{W}^T\mathbf{x}_j)^2 s_{i,j}\\ =\ & \sum_i (\mathbf{W}^T\mathbf{x}_i d_{i,j} \mathbf{x}_i{}^T\mathbf{W}) - \sum_{i,j} (\mathbf{W}^T\mathbf{x}_i s_{i,j} \mathbf{x}_i{}^T\mathbf{W})\\ =\ & tr(\mathbf{W}^T\mathbf{X}\mathbf{D}\mathbf{X}^T\mathbf{W}) - tr(\mathbf{W}^T\mathbf{X}\mathbf{S}\mathbf{X}^T\mathbf{W})\\ =\ & tr(\mathbf{W}^T\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{W})\end{aligned} \tag{4}$$

where $\mathbf{D}$ is a diagonal matrix and the ith diagonal element of $\mathbf{D}$ is defined as $d_{i,i} = \Sigma_j s_{i,j}$. Hence, $\mathbf{L} = \mathbf{D} - \mathbf{S}$ is the Laplacian matrix.

3.3. Approach

When we reconstruct test samples $\mathbf{Y}$ with training samples $\mathbf{X}$ to obtain the linear transformation matrix $\mathbf{W}$, it expects to map $\mathbf{X}$ into the space of $\mathbf{Y}$ via $\mathbf{W}$ and make the distance between $\mathbf{Y}$ and $\mathbf{W}^T\mathbf{X}$ as small as possible. Accordingly, we employ the least square loss function to control the reconstruction error (Arefi and Taheri, 2015)(Zhu et al., 2016a).

$$\left\|\mathbf{W}^T\mathbf{X} - \mathbf{Y}\right\|_F^2 = \left\|\hat{\mathbf{Y}} - \mathbf{Y}\right\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{m} (\hat{y}_{i,j} - y_{i,j})^2 \tag{5}$$

where $\hat{\mathbf{Y}}$ is the new representation of $\mathbf{X}$ in the space of $\mathbf{Y}$, *i.e.,* $\hat{\mathbf{Y}} = \mathbf{W}^T\mathbf{X}$. $\|\mathbf{W}^T\mathbf{X} - \mathbf{Y}\|_F^2$ denotes the reconstruction error. Thanks to the Eq.(5) is convex, we can easily obtain its global solution $\mathbf{W} = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{Y}$. Because of $\mathbf{X}\mathbf{X}^T$ is not always invertible in real applications, an ℓ_2-norm is added to remove the issue of invertible. Thus, the objective function is changed to the ridge regression.

$$\arg\min_{\mathbf{W}} \left\|\mathbf{W}^T\mathbf{X} - \mathbf{Y}\right\|_F^2 + \delta \|\mathbf{W}\|_2^2 \tag{6}$$

where δ is a tuning parameter. The optimal solution of Eq.(6) can be described as a closed solution $\mathbf{W} = (\mathbf{X}\mathbf{X}^T + \delta\mathbf{I})^{-1}\mathbf{X}\mathbf{Y}$, where $\mathbf{I} \in R^{n\times n}$ is an identity matrix.

The regularization term ℓ_1-norm has been proved to generate zero elements in a matrix, *i.e.,* lead to sparsity (Zhu et al., 2017a), while many studies have shown that the ℓ_2-norm do not surely generate sparse result. In this paper, the element $w_{i,j}$ in $\mathbf{W}$ indicates the correlation between the ith test sample and the jth training sample. We expected that each test sample is only represented by part of training samples, *i.e.,* many zero elements on each column in $\mathbf{W}$. Therefore, it makes sense for us to use an ℓ_1-norm term to replace the ℓ_2-norm. Meanwhile, we also employ the LPP to preserve the local structures of data after the reconstruction process. Thus, we defined the objective function of the proposed S-kNN method as follows.

$$\arg\min_{\mathbf{W}} \frac{1}{2}\left\|\mathbf{W}^T\mathbf{X} - \mathbf{Y}\right\|_F^2 + \rho_1 tr(\mathbf{W}^T\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{W}) + \rho_2\|\mathbf{W}\|_1 \tag{7}$$

where ρ_1 is a tuning parameter and designed to balance the magnitude between $tr(\mathbf{W}^T\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{W})$ and $\left\|\mathbf{W}^T\mathbf{X} - \mathbf{Y}\right\|_F^2$. Moreover, the larger the value of ρ_1 is, the larger the contribution of LPP in Eq.(7) will be. In particular, Eq.(7) can shrink to LASSO when setting $\rho_1 = 0$.

Different from LASSO, the proposed S-kNN gives a consideration to preserve the local structures of data via LPP. Moreover, the S-kNN is utilized to learn the k value for kNN algorithm. Different from that conventional kNN algorithms often use a fixed k value for all test samples, or learn the k value for each test sample without respect to the correlation among test samples. The proposed S-kNN algorithm learns an optimal k value for each test sample with the above reconstruction process. During the reconstruction process, the proposed method considers the correlation between test samples and training samples. It first considers the correlations of test samples through generating the k values for all test samples. And then, the correlations of training samples are taken into account by adding the LPP regularization term into the reconstruction process. Consequently, the proposed S-kNN method is a data-driven method for selecting the optimal k values.

In Eq.(7), each element $w_{i,j}$ of the matrix $\mathbf{W}$ can be understood as the correlation between ith test sample and jth training sample. If $w_{i,j} > 0$, their correlation is positive; if $w_{i,j} < 0$, the correlation is negative; and if $w_{i,j}$=0, the ith test sample is unrelated to the jth training sample. To understand the optimization of Eq.(7), assume we have the following optimal $\mathbf{W}$.

$$\mathbf{W} = \begin{pmatrix} 0 & 0 & 0.7 & 0.3\\ 0.1 & 0 & 0.3 & 0\\ 0.6 & 0 & 0 & 0\\ 0 & 0.7 & 0 & 0.1\\ 0 & 0 & 0 & 0.5\\ 0.5 & 0 & 0 & 0.7 \end{pmatrix}$$

In this example, there are six training samples and four test samples. According to $\mathbf{W}$, we know that the first column has three nonzero elements. That is, the test sample has three nearest neighbors, *i.e.,* the value k=3. Moreover, the greater the correlation value is, the closer the correlations between the

[2] σ is a tuning parameter. For simplicity, we set $\sigma = 1$ in our experiments.

test sample and the training samples are. The second column has one nonzero element. It means that the test sample has only one nearest neighbor, *i.e.,* the value k=1. With the same rule, the third test sample and the fourth test samples have two nearest neighbors (*i.e.,* the value k=2) and four nearest neighbors (*i.e.,* the value k=4), respectively. Note that, the sparsity (*i.e.,* there are many zero elements in $\mathbf{W}$) is generated due to the introduction of the ℓ_1-norm in Eq.(7). This leads to that our algorithm outputs different k values for different test samples. Moreover, the LPP is introduced for further improving the performance of the reconstruction process in Eq.(7). However, almost all conventional kNN algorithms employ a fixed k value decided by users or experts for all test samples.

3.4. Optimization

Note that the Eq.(7) is a convex but non-smooth function. In this subsection, we address this by designing a new accelerated proximal gradient method (Zhu et al., 2014). We first conduct the proximal gradient method on Eq.(7) by letting:

$$f(\mathbf{W}) = \frac{1}{2}\left\|\mathbf{W}^T\mathbf{X}-\mathbf{Y}\right\|_F^2+\rho_1 tr(\mathbf{W}^T\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{W}) \quad (8)$$

$$\vartheta(\mathbf{W}) = f(\mathbf{W})+\rho_2\|\mathbf{W}\|_1 \quad (9)$$

We known that that $f(\mathbf{W})$ is convex and differentiable. Thus, we used the proximal gradient method to optimize $\mathbf{W}$, and iteratively update it by means of the following optimization rule.

$$\mathbf{W}(t+1) = \arg\min_{\mathbf{W}} \mathbf{G}_{\eta(t)}(\mathbf{W},\mathbf{W}(t)) \quad (10)$$

where $\mathbf{G}_{\eta(t)}(\mathbf{W},\mathbf{W}(t)) = f(\mathbf{W}(t))+\langle\nabla f(\mathbf{W}(t)),\mathbf{W}-\mathbf{W}(t)\rangle+\frac{\eta(t)}{2}\|\mathbf{W}-\mathbf{W}(t)\|_F^2+\rho_2\|\mathbf{W}\|_1$, $\nabla f(\mathbf{W}(t)) = (\mathbf{X}\mathbf{X}^T+\rho_1\mathbf{X}\mathbf{L}\mathbf{X}^T)\mathbf{W}(t)-\mathbf{X}\hat{\mathbf{Y}}^T$, $\langle\cdot,\cdot\rangle$ is an inner product operator, $\eta(t)$ determines the step size of the t-iteration, $\mathbf{W}(t)$ is the value of $\mathbf{W}$ obtained at the t-iteration, and ρ_2 is a tuning parameter.

By ignoring the terms independent of $\mathbf{W}$ in Eq.(10), we can rewrite it as follows.

$$\mathbf{W}(t+1) = \pi_{\eta(t)}(\mathbf{W}(t)) = \arg\min \frac{1}{2}\|\mathbf{W}-\mathbf{U}(t)\|_2^2+\frac{\rho_2}{\eta(t)}\|\mathbf{W}\|_1 \quad (11)$$

where $\mathbf{U}(t) = \mathbf{W}(t)-\frac{1}{\eta(t)}\nabla f(\mathbf{W}(t))$ and $\pi_{\eta(t)}(\mathbf{W}(t))$ is the Euclidean projection of $\mathbf{W}(t)$ onto the convex set $\eta(t)$. Taking into account the separability of $\mathbf{W}(t+1)$ on each row, *i.e.,* $\mathbf{w}^i(t+1)$, we update the weights for each row individually as follows.

$$\mathbf{w}^i(t+1) = \arg\min_{\mathbf{w}^i}\frac{1}{2}\left\|\mathbf{w}^i-\mathbf{u}^i(t)\right\|_2^2+\frac{\rho_2}{\eta(t)}\left\|\mathbf{w}^i\right\|_2 \quad (12)$$

where $\mathbf{u}^i(t) = \mathbf{w}^i(t)-\frac{1}{\eta(t)}\nabla f(\mathbf{w}^i(t))$ and $\mathbf{w}^i(t)$ are the ith row of $\mathbf{u}(t)$ and $\mathbf{W}(t)$, respectively. According to Eq.(12), $\mathbf{w}^i(t+1)$ takes a closed form solution as follows.

$$\mathbf{w}^{i*} = max\{\left|\mathbf{w}^i\right|-\rho_2,0\}\cdot sgn(\mathbf{w}^i) \quad (13)$$

Meanwhile, in order to accelerate the proximal gradient method in Eq.(8), we introduce an auxiliary variable $\mathbf{V}(t+1)$ as follows as follows.

$$\mathbf{V}(t+1) = \mathbf{W}(t)+\frac{\alpha(t)-1}{\alpha(t+1)}(\mathbf{W}(t+1)-\mathbf{W}(t)) \quad (14)$$

Algorithm 1: Pseudo code of solving Eq.(7).

Input: $\eta(0)=0.01$, $\alpha(1)=1$, $\gamma=0.002$, ρ_1, ρ_2;
Output: $\mathbf{W}$;

```
Initialize t = 1;
Initialize W(1) as a random diagonal matrix;
repeat
    while L(W(t)) > G_{η(t-1)}(π_{η(t-1)}(W(t)), W(t)) do
        Set η(t-1) = γη(t-1);
    end
    Set η(t) = η(t-1);
    Compute W(t+1) = arg min_W G_{η(t)}(W, V(t));
    Compute α(t+1) = (1+sqrt(1+4α(t)^2))/2;
    Compute Eq.(14);
until Eq.(7) converges;
```

where the coefficient $\alpha(t+1)$ is usually set as $\alpha(t+1) = \frac{1+\sqrt{1+4\alpha(t)^2}}{2}$.

Finally, we presnet the pseudo of our proposed optimization method in Algorithm 1 and its convergence in Theorem 1.

Theorem 1. *Let $\{\mathbf{W}(t)\}$ be the sequence generated by Algorithm 1, then for $\forall\ t\geq 1$, the following formula holds*

$$\vartheta(\mathbf{W}(t))-\vartheta(\mathbf{W}^*)\leq\frac{2\gamma L\|\mathbf{W}(1)-\mathbf{W}^*\|_{\mathrm{F}}^2}{(t+1)^2} \quad (15)$$

where $\gamma>0$ is a predefined constant, L is the Lipschitz constant of the gradient of $f(\mathbf{W})$ in Eq.(8), and $\mathbf{W}^* = \arg\min_{\mathbf{W}}\vartheta(\mathbf{W})$.

Theorem 1 shows that the convergence rate of the proposed accelerated proximal gradient method is $O(\frac{1}{t^2})$, where t is the count number of iterations in Algorithm 1.

3.5. S-kNN Algorithm

In our propose algorithm, firstly, we optimize Eq.(7) to obtain the correlation coefficient matrix $\mathbf{W}$, so as to obtain an optimal k value for each test sample. And then, we use the selected k to conduct kNN algorithm for different data mining tasks, such as classification, regression, and missing value imputation.

For regression and missing value imputation tasks, the bigger the correlation between a test sample and its nearest neighbors is, the larger the contribution of the nearest neighbors to the test sample is. Therefore, we propose to employ a weighted method for both the regression task and missing value imputation task (Zhu et al., 2013a). And we defined the weighted predictive value of jth test sample as follows.

$$\text{predictvalue\_weight} = \sum_{i=1}^{n}\left(\frac{w_{i,j}}{\sum_{i=1}^{n}w_{i,j}}\times\mathbf{y}_{train(i)}\right) \quad (16)$$

where n is the number of training samples, and $\mathbf{y}_{train(i)}$ stands for the true value of the ith training sample.

For classification applications, the proposed S-kNN algorithm uses k nearest neighbors of each test sample to predict its class label with the majority rule.

Algorithm 2: The pseudo of S-kNN algorithm.

Input: **X**, **Y**;
Output:
switch *task* **do**
  case *1*
    Class labels;
  end
  case *2*
    Predicted value;
  end
  case *3*
    Imputation value;
  end
endsw
Normalizing **X** and **Y** (When **Y** is class labels without normalization);
Optimizing Eq.(7) to obtain the optimal solution **W**;
Obtaining the optimal k value for test samples based on **W**;
switch *task* **do**
  case *1*
    Obtaining class labels via majority rule;
  end
  case *2*
    Obtaining prediction value via Eq.(16);
  end
  case *3*
    Obtaining imputation value via Eq.(16);
  end
endsw

Therefore, our model can be easily applied to data mining tasks, such as regression, missing value imputation and classification. We describe these computations with Algorithm 2 as follows.

In Algorithm 2, the input data is first normalized. And then, the dataset is divided into a set of training samples and a set of test samples for 10-fold cross validation. Thirdly, the correlation coefficient **W** between training samples and test samples is computed with Eq.(7), and the optimal solution **W** is obtained with the proposed optimization process. Consequently, we generate the most correlative training samples of a test sample, *i.e.,* top k candidates of nearest neighbors (training samples) of the test sample. Finally, we use the k nearest training samples to predict the test sample. This means that different test samples are predicted with different numbers of nearest neighbors. Note that the regression and missing value imputation tasks are carried out with Eq.(16), and data classification is with the majority rule.

4. Experimental analysis

The proposed S-kNN method was evaluated in the data mining tasks, such as classification, regression and missing value imputation, by compared with the state-of-the-art kNN algorithms. Note that the classification task includes binary classification and multi-class classification.

Table 1. Benchmark datasets

Dataset	Instances	Features	Type	Classes
Adult	1605	113	classification	2
Arcene	100	9920	classification	2
Australian	690	14	classification	2
Cleveland	214	13	classification	2
Derm	358	34	classification	2
Heart	270	13	classification	2
Ionosphere	350	34	classification	2
Sonar	208	60	classification	2
Satimage	620	36	classification	6
Seeds	210	7	classification	3
Bodyfat	252	14	regression	No
Concreteslump	103	10	regression	No
Mpg	398	8	regression	No
Triazines	186	60	regression	No
Wine-white	4898	11	regression	No
Abalone	4177	8	imputation	No
Eunite2001	336	16	imputation	No
Housing	506	13	imputation	No
Pyrim	74	28	imputation	No
YachtHydrodynamics(Yacht)	74	7	imputation	No

4.1. Experimental setting

In our experiments, standard kNN algorithm was regarded as first comparison algorithm, where the k value is setting to 5. The second compared algorithm is Eq.(7) with the setting of $\rho_1 = 0$, *i.e.,* via LASSO to learn different k values for test samples. We call this algorithm as L-kNN, with which we would like to show the importance of preserving the local structures of data (Kang and Cho, 2008; Tibshirani, 1996).

There are 20 datasets involved to validate the proposed algorithm, which were downloaded from UCI (Bache and Lichman, 2013), LIBSVM (Chang and Lin, 2011) and the literature[3]. These datasets are detailed in Table 1. We conduct experiments on 10 datasets for classification, 5 datasets for regression, and 5 datasets for missing value imputation, respectively. We coded all algorithms with MATLAB 7.1 in windows 7 system. We conducted experiments by 10-fold cross-validation method, and repeated the whole process 10 times to avoid the possible bias.

The classification accuracy is employed to measure the classification efficiency which is defined as follows.

$$Accuracy = \frac{n_{correct}}{n} \tag{17}$$

where n is the number of all samples, $n_{correct}$ is the number of correct classification samples. The higher accuracy the algorithm is, the better performance of classification it is.

The Root Mean Square Error (RMSE) (Zhu et al., 2013b) and correlation coefficient are employed to evaluate the performance of both regression analysis and missing value imputation. Note that there are not missing values in the original datasets, we randomly selected some independent values to be missed according to the literatures on missing value imputations.

The RMSE is defined as the square root of predicted value and the ground-truth. The formal formula is as follows.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{18}$$

[3] http://www.cc.gatech.edu/~lsong/code.html

where y_i indicates the ground-truth, $\widehat{y}_i$ indicates the predicted value. Obviously, the smaller the RMSE is, the better the performance of predictions is.

Correlation coefficient indicates the correlations between predictions and observations. The correlation coefficient is between +1 and -1, where 1 is perfect positive correlation, 0 is no correlation, and -1 is totally negative correlation. Generally, the larger the correlation coefficient is, the more accurate the prediction is.

4.2. Experimental Results

In this section, we evaluate the performance of the proposed S-*k*NN algorithm by compared with two algorithms on real datasets, in terms of three data mining tasks, classification, regression and missing value imputation. We evaluate both the regression and missing value imputation in the same subsection because they have the same prediction model.

4.2.1. Data classification

We summarize the classification accuracies of all algorithms in Table 2. We listed the results of each repeated average value which repeated the 10-fold cross-validation method process 10 times for all algorithms in Figures 3-12.

As shown in Table 2, we found that the proposed S-*k*NN algorithm outperformed the comparison algorithms, L-*k*NN and standard *k*NN. Specifically, in terms of the classification accuracy, the proposed S-*k*NN algorithm averagely improves 4.47% and 22.38% accuracies more than the L-*k*NN algorithm and standard *k*NN method, respectively. In addition, Figures 3-12 have showed that the S-*k*NN algorithm had the highest accuracy in each of iteration. From the results on the Satimage (a multi-class dataset), the performance of the standard *k*NN algorithm is poor. However, the proposed algorithm performs steadily results for multi-class classification applications.

As demonstrated in the above, the S-*k*NN approach performs better than the L-*k*NN algorithm. This is because we utilized the LPP regularization term into the S-*k*NN method for preserving the local structure of data. In particular, the S-*k*NN method has averagely improved by 5.47% accuracy more than the L-*k*NN. And on Heart dataset, the S-*k*NN algorithm improved by 11.48% accuracy more than the L-*k*NN method.

Both the S-*k*NN and L-*k*NN outperformed the *k*NN algorithm. It is the reason that the both methods using different *k* values in *k*NN algorithm and lead to better classification performance than standard *k*NN method that is with a fixed *k* value for all test samples.

4.2.2. Regression results & Imputation results

We summarize the regression RMSE results of regression and missing value imputation in Tables 3 and 4 respectively, and the correlation coefficient in Tables 5 and 6, respectively. We also depict the RMSE results of repeated the 10-fold cross-validation method process 10 times for all algorithms in Figures 13-22, and the correlation coefficient results of all algorithms in each of iterations in Figures 23-32.

From Tables 3 and 4, in terms of RMSE, we can find that the proposed S-*k*NN achieved the best performance of regression and missing value imputation, followed by L-*k*NN and

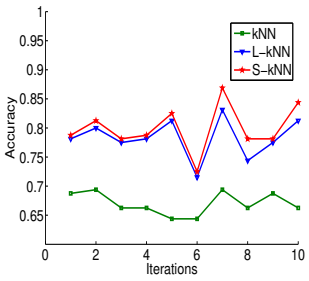

Fig. 3. Adult

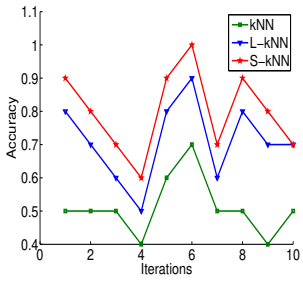

Fig. 4. Arcene

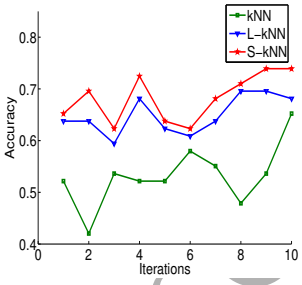

Fig. 5. Australian

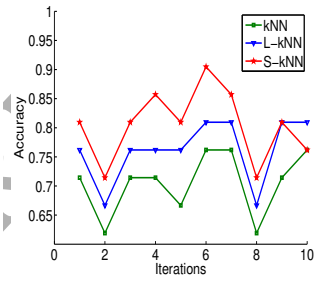

Fig. 6. Cleveland

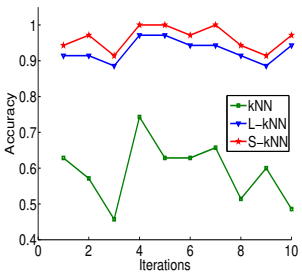

Fig. 7. Derm

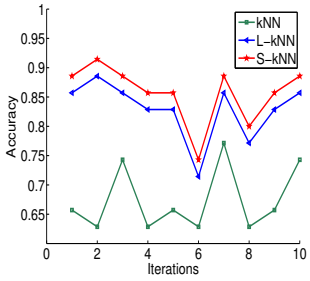

Fig. 8. Ionosphere

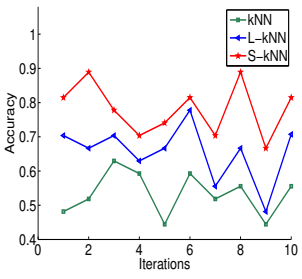

Fig. 9. Heart

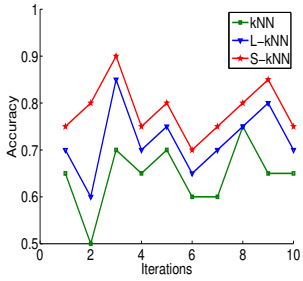

Fig. 10. Sonar

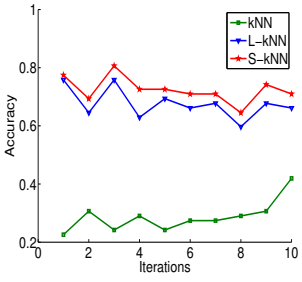

Fig. 11. Satimage

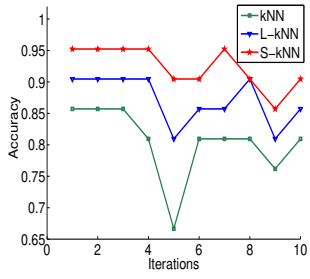

Fig. 12. Seeds

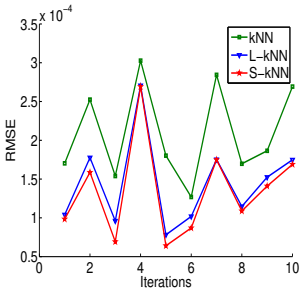

Fig. 13. Bodyfat

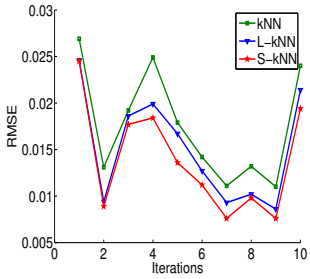

Fig. 14. Concretes-lump

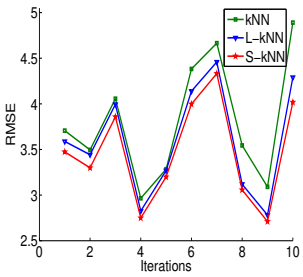

Fig. 15. Mpg

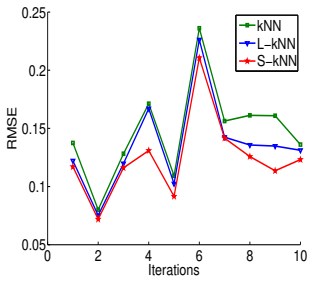

Fig. 16. Triazines

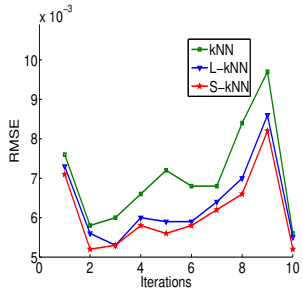

Fig. 17. Wine-white

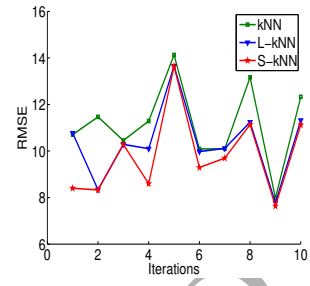

Fig. 18. Abalone

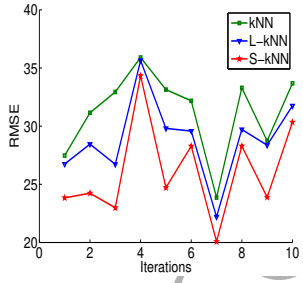

Fig. 19. Eunite2001

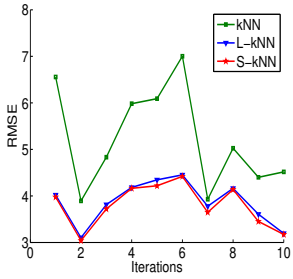

Fig. 20. Housing

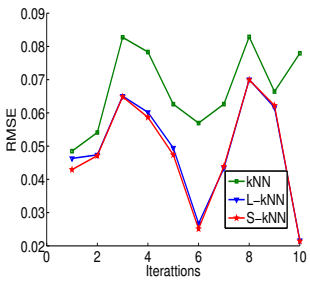

Fig. 21. Pyrim

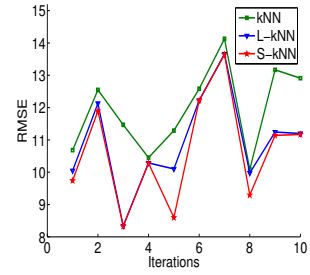

Fig. 22. Yacht

Table 2. Comparison of classification accuracy(mean±std)

Dataset	*k*NN	L-*k*NN	S-*k*NN
Adult	0.6700±0.0004	0.7828±0.0012	**0.7994±0.0016**
Arcene	0.5100±0.0077	0.7100±0.0143	**0.8000±0.0156**
Australian	0.5319±0.0036	0.6493±0.0013	**0.6826±0.0021**
Cleveland	0.7048±0.0029	0.7619±0.0030	**0.8048±0.0038**
Derm	0.6057±0.0061	0.9343±0.0007	**0.9714±0.0009**
Ionosphere	0.6743±0.0031	0.8286±0.0025	**0.8571±0.0025**
Heart	0.5381±0.0031	0.6741±0.0033	**0.7889±0.0043**
Sonar	0.6450±0.0047	0.7200±0.0051	**0.7850±0.0034**
Satimage	0.2871±0.0029	0.6758±0.0026	**0.7424±0.0019**
Seeds	0.8048±0.0033	0.8714±0.0015	**0.9238±0.0011**
MEAN	0.5972±0.0200	0.7608±0.0088	**0.8155±0.0070**

Table 3. Regression performances in terms of rmse(mean±std)

Dataset	*k*NN	L-*k*NN	S-*k*NN
Bodyfat	2.1e-05±3.8e-09	1.4e-05±3.3e-09	**1.3e-05±3.9e-09**
Concreteslump	0.0176±4.1e-04	0.0151±3.2e-04	**0.0139±3.1e-04**
Mpg	3.8080±0.4417	3.5909±0.3662	**3.4693±0.3158**
Triazines	0.1477±0.0017	0.1357±0.0016	**0.1242±0.0013**
Wine-white	0.0071±1.6e-06	0.0064±1.0e-06	**0.0061±9.3e-07**

*k*NN. Figures. 13-22 also showed that the S-*k*NN algorithm had higher prediction performance than the two compared algorithms in each of iteration. For the correlation coefficient, Figures 23-32 demonstrated the results as similar to that in Figures 13-22.

As we have seen, in the evaluation of RMSE, the proposed S-*k*NN has averagely has averagely reduced the classification error by 0.0734 and 0.0269 less than the L-*k*NN and standard *k*NN, respectively. In particular, the S-*k*NN algorithm made the most improvement on Mpg dataset, *i.e.,* reduced 0.1216 and 0.3387 less than the L-*k*NN and standard *k*NN. There is a similar performance in the evaluation of S-*k*NN missing value imputation with the RMSE.

In terms of correlation coefficient on five datasets, the proposed S-*k*NN has averagely increases by 3.66% more than L-*k*NN, and by 10.42% more than standard *k*NN. Moreover, the proposed method achieved the maximal increment on Triazines dataset, *i.e.,* 8.8% more than the L-*k*NN and 22.85% more than standard *k*NN.

Like the above experiments for evaluating S-*k*NN classification, the evaluations for S-*k*NN regression and missing value imputation have also demonstrated two improvements as follows.

- The proposed S-*k*NN outperformed L-*k*NN duo to that the preservation of local structures of data is well considered in the S-*k*NN algorithms.
- Both the S-*k*NN and L-*k*NN outperformed standard *k*NN because they learn different optimal k values for different

Table 4. Imputation performances in terms of rmse(mean±std)

Dataset	*k*NN	L-*k*NN	S-*k*NN
Abalone	2.3894±0.1019	2.1261±0.1360	**2.0850±0.1207**
Eunite2001	31.2345±12.691	28.8972±12.253	**26.0969±17.403**
Housing	5.2228±1.2315	3.8666±0.2108	**3.7948±0.2175**
Pyrim	0.0673±0.0002	0.0492±0.0003	**0.0484±0.0003**
Yacht	10.5379±3.0492	10.1437±2.6463	**9.4637±3.2310**

Table 5. Regression performances in terms of correlation coefficient(mean±std)

Dataset	*k*NN	L-*k*NN	S-*k*NN
Bodyfat	0.9846±8.1e-05	0.9918±4.4e-05	**0.9930±4.3e-05**
Concreteslump	0.6606±0.0312	0.7719±0.0249	**0.8194±0.0195**
Mpg	0.8865±0.0019	0.8978±0.0014	**0.9076±0.0011**
Triazines	0.4256±0.0148	0.5661±0.0123	**0.6541±0.0106**
Wine-white	0.9041±4.9e-04	0.9260±2.7e-04	**0.9307±1.9e-04**

Table 6. Imputation performances in terms of correlation coefficient(mean±std)

Dataset	*k*NN	L-*k*NN	S-*k*NN
Abalone	0.6499±0.0008	0.7282±0.0024	**0.7376±0.0021**
Eunite2001	0.8239±0.0048	0.8493±0.0036	**0.8741±0.0013**
Housing	0.8285±0.0004	0.9142±0.0004	**0.9197±0.0003**
Pyrim	0.8283±0.0079	0.9140±0.0025	**0.9201±0.0022**
Yacht	0.7200±0.0044	0.7948±0.0039	**0.8162±0.0034**

test samples.

In a word, according to the results on three learning tasks, we can make the following conclusion: First, it might be reasonable to use varied k values in kNN algorithm in real applications. Second, the optimal k values should be learnt from the data, *i.e.,* the data-driven k value in kNN algorithm.

5. Conclusion

In this work, we have proposed a novel kNN algorithm, called S-kNN approach, by replacing the fixed k value for all test samples with learning different k values for different test samples according to the distribution of the data. It is an example-driven k-parameter computation. The key is the reconstruction of correlation between test samples and training samples, which is a sparse coefficient matrix. With this sparse correlation, we can obtain the optimal k value for a test sample. For efficiency, the LPP regularization is adopted to keep the local structures of the data. The experiments on 20 real datasets have demonstrated that the proposed S-kNN method is efficiency and promising, compared with the state-of-the-art kNN methods.

6. Acknowledgements

This work was supported in part by the China Key Research Program (Grant No: 2016YFB1000905), the China 973 Program (Grant No: 2013CB329404), the China 1000-Plan National Distinguished Professorship, the Nation Natural Science Foundation of China (Grants No: 61573270, 61672177, and 61363009), National Association of public funds, the Guangxi Natural Science Foundation (Grant No: 2015GXNSFCB139011), the Guangxi High Institutions Program of Introducing 100 High-Level Overseas Talents, the Guangxi Collaborative Innovation Center of Multi-Source Information Integration and Intelligent Processing, the Research Fund of Guangxi Key Lab of MIMS (16-A-01-01 and 16-A-01-02), and the Guangxi Bagui Teams for Innovation and Research.

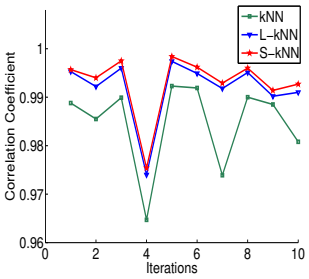

Fig. 23. Bodyfat

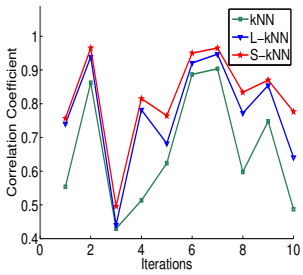

Fig. 24. Concretes-lump

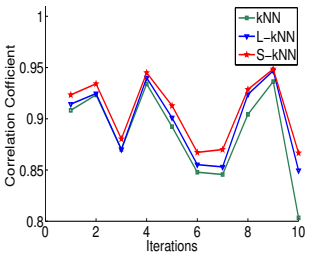

Fig. 25. Mpg

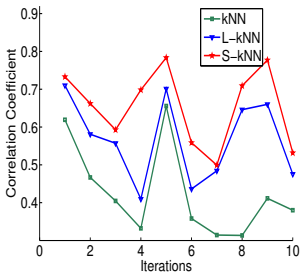

Fig. 26. Triazines

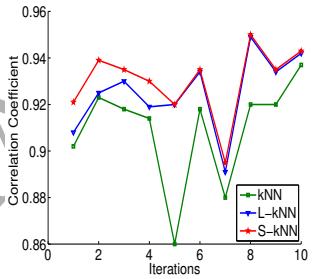

Fig. 27. Wine-white

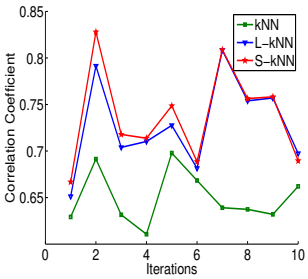

Fig. 28. Abalone

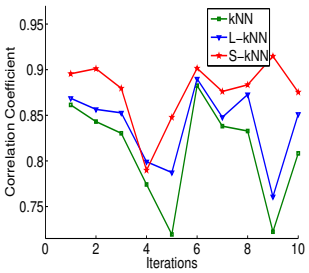

Fig. 29. Eunite2001

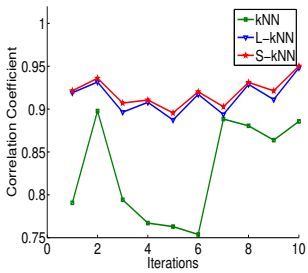

Fig. 30. Housing

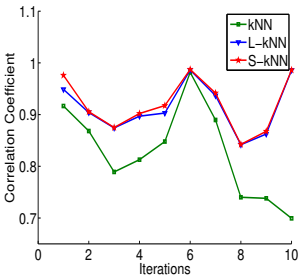

Fig. 31. Pyrim

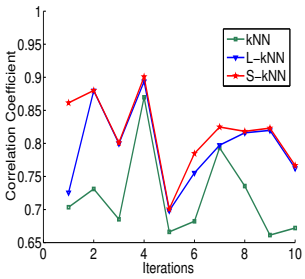

Fig. 32. Yacht

References

Arefi, M., Taheri, S.M., 2015. Least-squares regression based on atanassov's intuitionistic fuzzy inputs–outputs and atanassov's intuitionistic fuzzy parameters. IEEE Transactions on Fuzzy Systems 23, 1142–1154.

Bache, K., Lichman, M., 2013. UCI machine learning repository. URL: `http://archive.ics.uci.edu/ml`.

Belkin, M., Niyogi, P., 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering, in: International Conference on Neural Information Processing Systems: Natural and Synthetic, pp. 585–591.

Burba, F., Ferraty, F., Vieu, P., 2009. k-nearest neighbour method in functional nonparametric regression. Journal of Nonparametric Statistics 21, 453–469.

Chang, C.C., Lin, C.J., 2011. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chen, J., Shao, J., 2001. Jackknife variance estimation for nearest-neighbor imputation. Journal of the American Statistical Association 96, 260–269.

Cheng, D., Zhang, S., Deng, Z., Zhu, Y., Zong, M., 2014. knn algorithm with data-driven k value, in: ADMA, pp. 499–512.

Cover, T., Hart, P., 1967. Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13, 21–27.

Devroye, L., et al., 1981. On the almost everywhere convergence of nonparametric regression function estimates. The Annals of Statistics 9, 1310–1319.

Ferraty, F., Vieu, P., 2006. Nonparametric functional data analysis: theory and practice. Springer Series in Statistics, Springer.

Ghosh, A.K., 2006. On optimum choice of k in nearest neighbor classification. Computational Statistics & Data Analysis 50, 3113–3123.

Goldberger, J., Roweis, S.T., Hinton, G.E., Salakhutdinov, R., 2004. Neighbourhood components analysis, in: NIPS, pp. 513–520.

Goyal, R., Chandra, P., Singh, Y., 2014. Suitability of knn regression in the development of interaction based software fault prediction models. Ieri Procedia 6, 15–21.

He, X., Niyogi, P., 2003. Locality preserving projections, in: NIPS, pp. 153–160.

Hoef, Jay M, V., Hailemariam, T., 2013. A comparison of the spatial linear model to nearest neighbor (k-nn) methods for forestry applications. Plos One 8, e59129.

Hron, K., Templ, M., Filzmoser, P., 2010. Imputation of missing values for compositional data using classical and robust methods. Computational Statistics & Data Analysis 54, 3095–3107.

Hu, C., Jain, G., Zhang, P., Schmidt, C., Gomadam, P., Gorka, T., 2014. Data-driven method based on particle swarm optimization and k-nearest neighbor regression for estimating capacity of lithium-ion battery. Applied Energy 129, 49–55.

Hu, R., Zhu, X., Cheng, D., He, W., Yan, Y., Song, J., Zhang, S., 2017. Graph self-representation method for unsupervised feature selection. Neurocomputing 220, 130–137.

Jamshidi, Y., Kaburlasos, V.G., 2014. gsainknn: A gsa optimized, lattice computing knn classifier. Engineering Applications of Artificial Intelligence 35, 277–285.

Kang, P., Cho, S., 2008. Locally linear reconstruction for instance-based learning. Pattern Recognition 41, 3507–3518.

Lall, U., Sharma, A., 1996. A nearest neighbor bootstrap for resampling hydrologic time series. Water Resources Research 32, 679–693.

Liu, H., Zhang, S., Zhao, J., Zhao, X., Mo, Y., 2010. A new classification algorithm using mutual nearest neighbors, in: GCC, pp. 52–57.

Mack, Y.P., 1981. Local properties of k-nn regression estimates. SIAM Journal on Algebraic Discrete Methods 2, 311–323.

Meesad, P., Hengpraprohm, K., 2008. Combination of knn-based feature selection and knnbased missing-value imputation of microarray data, in: ICICIC, pp. 341–341.

Ni, J., Qiao, F., Li, L., Di Wu, Q., 2012. A memetic pso based knn regression method for cycle time prediction in a wafer fab, in: WCICA, pp. 474–478.

Qin, Y., Zhang, S., et al., 2007. Semi-parametric optimization for missing data imputation. Applied Intelligence 27, 79–88.

Qin, Z., Wang, A.T., Zhang, C., Zhang, S., 2013. Cost-sensitive classification with k-nearest neighbors, in: International Conference on Knowledge Science, Engineering and Management, Springer. pp. 112–131.

Saini, I., Singh, D., Khosla, A., 2013. Qrs detection using k-nearest neighbor algorithm (knn) and evaluation on standard ecg databases. Journal of Advanced Research 4, 331–344.

Tibshirani, R., 1996. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological) , 267–288.

Varmuza, K., Filzmoser, P., Hilchenbach, M., Krger, H., Siln, J., 2014. Knn classification evaluated by repeated double cross validation: Recognition of minerals relevant for comet dust. Chemometrics & Intelligent Laboratory Systems 138, 64–71.

Weinberger, K.Q., Saul, L.K., 2009. Distance metric learning for large margin nearest neighbor classification. Journal of Machine Learning Research 10, 207–244.

Wu, X., Kumar, V., et al., 2008. Top 10 algorithms in data mining. Knowledge and Information Systems 14, 1–37.

Yager, R.R., Petry, F.E., 2014. Hypermatching: Similarity matching with extreme values. IEEE Transactions on Fuzzy Systems 22, 949–957.

Zhang, C., Zhu, X., et al., 2007. GBKII: an imputation method for missing values, in: PAKDD, pp. 1080–1087.

Zhang, S., 2011. Shell-neighbor method and its application in missing data imputation. Applied Intelligence 35, 123–133.

Zhang, S., Jin, Z., Zhu, X., 2011. Missing data imputation by utilizing information within incomplete instances. Journal of Systems and Software 84, 452–459.

Zhang, S., Li, X., Zong, M., Zhu, X., Cheng, D., 2017a. Learning k for knn classification. ACM Transactions on Intelligent Systems and Technology 8, 43.

Zhang, S., Li, X., Zong, M., Zhu, X., Wang, R., 2017b. Efficient knn classification with different numbers of nearest neighbors. IEEE Transactions on Neural Networks and Learning Systems , 10.1109/TNNLS.2017.2673241.

Zhang, S., Qin, Y., Zhu, X., Zhang, J., Zhang, C., 2006. Optimized parameters for missing data imputation, in: PRICAI 2006: Trends in Artificial Intelligence. Springer, pp. 1010–1016.

Zhang, S., Wu, X., Zhu, M., 2010. Efficient missing data imputation for supervised learning, in: ICCI, pp. 672–679.

Zhou, Z.H., Li, M., 2005. Semi-supervised regression with co-training, in: IJCAI, pp. 908–916.

Zhu, X., Huang, Z., Shen, H.T., Zhao, X., 2013a. Linear cross-modal hashing for efficient multimedia search, in: ACM Multimedia, pp. 143–152.

Zhu, X., Huang, Z., et al., 2013b. Video-to-shot tag propagation by graph sparse group lasso. IEEE Transactions on Multimedia 15, 633–646.

Zhu, X., Li, X., Zhang, S., 2016a. Block-row sparse multiview multilabel learning for image classification. IEEE transactions on cybernetics 46, 450–461.

Zhu, X., Li, X., Zhang, S., Ju, C., Wu, X., 2017a. Robust joint graph sparse coding for unsupervised spectral feature selection. IEEE transactions on neural networks and learning systems 28, 1263–1275.

Zhu, X., Li, X., Zhang, S., Xu, Z., Yu, L., Wang, C., 2017b. Graph pca hashing for similarity search. IEEE Transactions on Multimedia .

Zhu, X., Suk, H., Wang, L., Lee, S., Shen, D., 2017c. A novel relational regularization feature selection method for joint regression and classification in AD diagnosis. Medical Image Analysis 38, 205–214.

Zhu, X., Suk, H.I., Huang, H., Shen, D., 2017d. Low-rank graph-regularized structured sparse regression for identifying genetic biomarkers. IEEE Transactions on Big Data , 10.1109/TBDATA.2017.2735991.

Zhu, X., Suk, H.I., Lee, S.W., Shen, D., 2016b. Subspace regularized sparse multitask learning for multiclass neurodegenerative disease identification. IEEE Transactions on Biomedical Engineering 63, 607–618.

Zhu, X., Zhang, L., Huang, Z., 2014. A sparse embedding and least variance encoding approach to hashing. IEEE Transactions on Image Processing 23, 3737–3750.