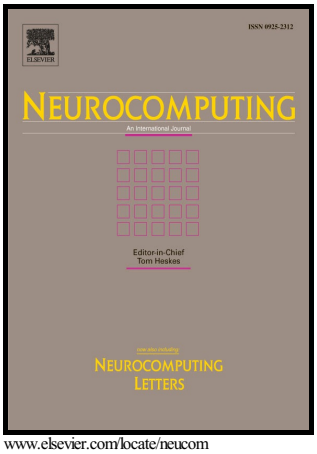



www.elsevier.com/locate/neucom

# Efficient *k*NN Classification Algorithm for Big Data

Shichao Zhang, Zhenyun Deng, Debo Cheng, Ming Zong, Xiaoshu Zhu

PII: S0925-2312(16)00113-2
DOI: http://dx.doi.org/10.1016/j.neucom.2015.08.112
Reference: NEUCOM16679

To appear in: *Neurocomputing*

Received date: 17 March 2015
Revised date: 1 July 2015
Accepted date: 7 August 2015

Cite this article as: Shichao Zhang, Zhenyun Deng, Debo Cheng, Ming Zong and Xiaoshu Zhu, Efficient *k*NN Classification Algorithm for Big Data *Neurocomputing,* http://dx.doi.org/10.1016/j.neucom.2015.08.112

# Efficient kNN Classification Algorithm for Big Data

Shichao Zhang*, Zhenyun Deng, Debo Cheng, Ming Zong, and Xiaoshu Zhu
Guangxi Key Lab of Multi-source Information Mining & Security,
Guangxi Normal University, Guilin, Guangxi, 541004, China
zhangsc@mailbox.gxnu.edu.cn

Abstract. K nearest neighbors (kNN) is an efficient lazy learning algorithm and has successfully been developed in real applications. It is very natural to scale the previous kNN method to the large scale datasets. In this paper, we propose to first conduct a k-means clustering to separate the whole dataset into several parts, each of which is then conducted kNN classification. We conduct sets of experiments on big data and medical imaging data. The experimental results show that the proposed kNN classification works well in terms of accuracy and efficiency.

Keywords: Classification; kNN; big data; data cluster; cluster center



## 1 Introduction

In the era of big data, it is the most important to efficiently learn from large scale in all kinds of real applications, such as classification and clustering. Therefore, it is very obvious to scale traditional classification algorithms, such as decision trees, support vector machine, Naive Bayes, neutral network, and k nearest neighbors (kNN), so that these methods can be easily used in big data. Due to the simplicity, easy-understand and relatively high performance of kNN, this paper focuses on scaling the kNN classification into the application of big data [13].

The previous kNN method first selects k nearest training samples for a test sample, and then predicts the test sample with the major class among k nearest training samples. However, kNN needs to compute the distance (or similarity) of all training samples for each test sample in the process of selecting k nearest neighbors [24][25]. The high cost (i.e., linear time complexity over the sample size) prohibits the

* Shichao Zhang: corresponding author.

traditional kNN method to be used in big data [21]. Obviously, conducting kNN algorithm in the memory of a PC should be an interesting issue.

Inspired by the recent progress on big data, this paper devised a new kNN method for dealing with big data. Specifically, we propose to first conduct a k-means clustering to separate the whole dataset into several parts [26]. And then we select the nearest cluster as the training samples and conduct the kNN classification. The time complexity of the proposed algorithm is linear to the sample size. The experimental results on real datasets including medical imaging datasets indicated that the proposed method achieved better performance than conventional kNN method in terms of both classification performance and time cost [15].

The rest of paper is organized as follows: in Section 2, we provide a brief review the previous fast kNN methods. We then propose the new kNN classification in Section 3. The experiment results are presented in Section 4. Finally, we give the conclusions and our future work in Section 5.

## 2 Related work

The research of kNN method has been becoming a hot research topic in data mining and machine learning since the algorithm was proposed in 1967. To apply for the traditional kNN method in big data, the previous literatures can be often categorized into two parts, i.e., fast finding the nearest samples [21] and selecting representatives samples (or removing some samples) to reduce the calculation of kNN [22]. For instance, Zhang proposed a Certainly Factor (CF) measure to deal with the unsuitability of skewed class distribution in kNN methods [14]. Li et al. proposed a density-based method for reducing the amount of training data [9]. Zhao et al. proposed a new algorithm based on the use of labeled samples and add the screening process condition, it making the new algorithm in time complexity have significantly decreased, and no significant effect on algorithm result [16]. These methods were mainly applied for fast search [17][18], dimension reduction [19][20], and improving the efficiency of the algorithms.

kNN algorithm computes the distance between each training sample and test samples in the dataset and then returns k closest samples. Its time complexity is linearly and is guaranteed to find exact k nearest neighbors. However, the computational complexity of the linear search method is proportional to the size of the training dataset for each test sample, it is O(nd), where n is the size of the training

dataset and d is the dimensionality [11][12]. This complexity is expensive for big data. Since the kNN method is not training process, we propose to introduce a new training process for kNN, which blocks training dataset by a clustering algorithm with linear complexity. During the testing process, for each test sample, we find the k nearest cluster centers and conduct a clustering for each test sample, and then construct a new classification model base on each cluster [21]. In particularly, the samples within a cluster has high similarity. Thus, comparing to the traditional kNN method, the proposed algorithm not only reduces the time complexity of kNN, but also does not add significantly effect on classification accuracy [22].

## 3 Method

In this section, we introduce two processes in our algorithm, namely training process and testing process. The training process is designed to select a nearest cluster for each test sample as its new training dataset, and the testing process is used to classify each test sample by kNN algorithm within its nearest cluster.

### 3.1 Training process

Clustering is one of the fundamental technique in data mining because I can be used for database segmentation and data compression and can also be employed for data preprocessing of data mining. Clustering is designed to group a set of samples in such a way that samples in the same group (cluster) are more similar to each other than to those in other groups. That is, samples is with high similarity within a cluster and low similarity between clusters.

The recent clustering methods can be parted into the following categories: density-based clustering, grid-based clustering, partitioning clustering and hierarchical clustering, respectively. Even though the previous clustering methods showed good performance, but they are limited in its applicability to big data due to theirs high computational complexity. To address this, this paper considers to employ a clustering algorithm satisfying the following two advantages: low complexity; scales linearly [1][5], respectively. To this end, we used the Landmark-based Spectral Clustering (LSC) [4] in this paper. The rationale of LSC is to select p (<<n) representative sample as landmarks and represents the original samples as the linear

combinations of these landmarks. Different from that traditional spectral clustering method use the entire samples to represent each sample, the LSC significant reduces the complexity of affinity matrix. At the same time, the complexity of solving the eigenvalue down to scales linearly [24].

LSC tries to compress the original samples by finding a set of basis vectors and the representation with respect to the bases for each sample, i.e., searching for p representative samples. In this way, we have two simple and effective methods to select landmark sample from original sample, such as random sampling and k-means-based method. Random sampling randomly selects samples as landmark samples while the k-means-based method first conducts clustering on all samples several times (no need to convergence) and then uses the cluster centers as the landmark samples. In this paper, we repeat k-means algorithm 10 times and then use the cluster centers as the landmark samples.

First, we treat every sample as a basis vector to construct landmark matrix Z. Note that LSC uses p landmarks to represent each sample $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] \in R^{m \times n}$. Thus, we need to find the matrix W which is projection matrix of X at the landmark matrix Z [10]. The projection function can be defined as follows:

$$w_{ji} = \frac{K_h(\mathbf{x}_i, \mathbf{z}_j)}{\sum_{j' \in Z_{<i>}} K_h(\mathbf{x}_i, \mathbf{z}_{j'})}, \quad j \in Z_{<i>}. \tag{1}$$

Where $\mathbf{z}_i$ is j-th column vector of Z, and $\mathbf{Z}_{<i>}$ denote a sub-matrix of Z composed of r nearest landmarks of $\mathbf{x}_i$. Here we need O(pmn) to construct W. $K_h(\cdot)$ is a kernel function with a bandwidths h. The Gaussian kernel $K_h(\mathbf{x}_i, \mathbf{z}_j) = \exp(-\left\|\mathbf{x}_i - \mathbf{z}_j\right\|^2 / 2h^2)$ is one of the most commonly used. Then we conduct spectral analysis on landmark-based graph and compute the graph matrix as:

$$\mathbf{G} = \hat{\mathbf{W}}^T \hat{\mathbf{W}}. \tag{2}$$

which has a very efficient eigen-decomposition. In this method, we choose $\hat{\mathbf{W}} = \mathbf{D}^{-1/2}\mathbf{W}$ where D is the row sum of W. Note that each column of W sums up to 1 and thus the degree matrix of G is I.

Let the Singular value Decomposition (SVD) of $\hat{\mathbf{W}}$ is as follows:

$$\hat{\mathbf{W}} = \mathbf{U}\Sigma\mathbf{V}^T . \tag{3}$$

where $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k] \in R^{p \times p}$ is called as the left singular vectors of the first k eigenvectors of $\hat{\mathbf{Z}}\hat{\mathbf{Z}}^T$, $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_k] \in R^{n \times p}$ is called as right singular vectors of the k eigenvectors of $\hat{\mathbf{Z}}^T\hat{\mathbf{Z}}$. We compute U within $O(p^3)$, linear to the sample size. V can be compute as:

$$\mathbf{V}^T = \Sigma^{-1}\mathbf{U}^T\hat{\mathbf{W}} . \tag{4}$$

The overall time complexity of V is $O(p^3+p^2n)$, which is a significant reduction from $O(n^3)$ by considering $p<<n$. Each row of V is a sample and apply k-means to get the clusters. Due to the time complexity from $O(n^3)$ to $O(n)$, the LSC algorithm substantially reduces computational time. Thus, the proposed algorithm with low-complexity is very suitable to be applied in the domain of big data.

Finally, the pseudo of LSC is presented in Algorithm 1.

Algorithm 1: The pseudo of LSC

Input: n data points $x_1, x_2, \cdots\cdots, x_n \in R^m$; Cluster number k;

Output: k clusters;

1 Produce p landmark points using the k-means method;
2 Construct a landmark matrix Z between data points and landmark samples, with the affinity calculated according to Eq.(1);
3 Compute the first k eigenvectors of $WW^T$, denoted by $U=[u_1, u_2, ..., u_k]$;
4 Compute $V=[v_1, v_2, ..., v_k]$ according to Eq.(4);
5 Each row of V is a data point and apply k-means to get the clusters.

### 3.2 Testing process

Supposing that we already produce k clusters and cluster centers by LSC algorithm, we then find out the nearest cluster center for each test sample and use the

corresponding cluster as the new training dataset for each test sample. Due to conducting the clusters with high similarity within a cluster, we apply kNN to classify test samples in the new training dataset. In this way, the proposed algorithm still guarantees relatively high classification accuracy. We list the pseudo of our proposed method in Algorithm 2.

Algorithm 2: The pseudo of LC-kNN algorithm.

Input: Training dataset, test samples Y;

Output: Class label;

1 Produce m cluster centers using LSC algorithm, denoted by $C_1,C_2,C_3,.....,C_m$;
2 Compute the distance $D(y,C_i)$ between test sample y and each cluster center, denoted by $D(y,C_i)$, i=1,2,...,m;
3 Compute the nearest cluster center $C_i$ to y, $C_i=\min\{D(y,C_i)\}$, i=1,2,...,m;
4 Using the corresponding cluster of $C_i$ as the training dataset, denoted by $NewX_i$;
5 Apple to kNN algorithm to predict y in the training dataset.

From algorithm 2, the number of $NewX_i$ is much smaller than the size of the training dataset. When the size of m is large, LC-kNN is easy to reduce the calculation of kNN and improves classification quality. However, with the increase of m, the overhead of clustering will increase and the classification accuracy will be lower at the same time. Therefore, in order to avoid this situation, the number of cluster m needs to be set in a reasonable value.

In general, the larger the value of m, the higher the classification efficiency, which lead to a higher classification accuracy. However, if the training dataset distribution is relatively concentrated, which will lead to low accuracy. In addition, if m is small, i.e., m=1, that is standard kNN algorithm, which prohibits the kNN to be used in big data. Thus we set m in a suitable range. Assuming that the proposed algorithm need M memory space, the memory of PC is $M_1$ and the smallest class of training sample is $n_0$. To this end, the value range of m is expressed as follows:

$$\frac{M}{M_1} < m < n_0 \,. \tag{5}$$

Note that we conduct a k-means clustering to separate the whole dataset into several parts, each of which we conduct kNN classification, the selection of k value is important in the later process. For example, Lall mentioned that the suitable setting of

k value should satisfy $k = \sqrt{n}$ for the datasets with sample size larger than 100 [8]. However, such a setting has been proved to not suitable for all cases of datasets. Considering that the samples has a strong correlation in sub-cluster, so the k value not be setting too large. Thus, the selection of k value should be set as small as possible in the case of high classification accuracy.

## 4 Experiments

As mentioned in the previous sections, our proposed algorithm is an extension of kNN. Thus, in order to show the effectiveness of LC-kNN algorithm, we took the kNN as the baseline and made a comparison between kNN, LC-kNN and RC-kNN (random clustering kNN), and several real datasets from UCI [2], medical imaging datasets [6][7], and LIBSVM [3] datasets.

### 4.1 Experimental results of LC-kNN and RC-kNN with different value of m

The value of m is very important to the LC-kNN algorithm, because it directly affects the final performance and real applications. Thus, in order to select appropriate m for bringing great effect to LC-kNN, a group of experiments were conducted on datasets by choosing different values of m for LC-kNN and RC-kNN. Specifically, the LC-kNN and RC-kNN in this group experiments were carried out on the datasets with m=10, 15, 20, 25 and 30, respectively. The comparison of classification performance and time cost of two algorithms (i.e., RC-kNN and LC-kNN, respectively) with different values of m are shown in Table 1~9.

Table 1. Classification accuracy (mean±standard deviation) and Time cost (seconds: mean± standard deviation) on USPS dataset at different values of m.

| m | Criterion | RC-kNN | LC-kNN |
|---|---|---|---|
| 10 | Accuracy | 0.9027±1.6498e-005 | 0.9355±7.1306e-006 |
| | Time | 3.5589±0.0107 | 3.7605±0.0242 |
| 15 | Accuracy | 0.8964±5.1803e-005 | 0.9338±4.1625e-006 |
| | time | 2.4857±0.0032 | 2.7260±0.0077 |
| 20 | Accuracy | 0.8770±7.4889e-005 | 0.9300±4.9238e-006 |
| | time | 2.3202±0.0010 | 2.5157±0.01928 |

| | | | |
|---|---|---|---|
| 25 | Accuracy | 0.8793±4.9917e-005 | 0.9284±1.0637e-005 |
| | time | 1.8586±0.0008 | 1.9971±0.0042 |
| 30 | Accuracy | 0.8607±4.6629e-005 | 0.9275±1.1596e-005 |
| | time | 1.6441±0.0002 | 1.9249±0.0023 |

Table 2. Classification accuracy (mean±standard deviation) and Time cost (seconds: mean± standard deviation) on MNIST dataset at different values of m.

| m | Criterion | RC-kNN | LC-kNN |
|---|---|---|---|
| 10 | Accuracy | 0.7221±4.8878e-005 | 0.8389±3.1656e-005 |
| | time | 2.9369±0.0508 | 3.5504±0.0927 |
| 15 | Accuracy | 0.6840±2.3333e-004 | 0.8364±2.3136e-005 |
| | time | 2.8905±0.0456 | 3.1222±0.01397 |
| 20 | Accuracy | 0.6657±2.4739e-004 | 0.8353±3.3233e-005 |
| | time | 2.0564±0.0011 | 2.1490±0.0065 |
| 25 | Accuracy | 0.6478±2.2689e-004 | 0.8338±8.7844e-005 |
| | time | 1.8240±0.0020 | 2.1148±0.0094 |
| 30 | Accuracy | 0.6396±6.9156e-005 | 0.8313±3.8678e-005 |
| | time | 1.5457±0.0002 | 1.7274±0.0011 |

Table 3. Classification accuracy (mean±standard deviation) and Time cost (seconds: mean± standard deviation) on GISETTE dataset at different values of m.

| m | Criterion | RC-kNN | LC-kNN |
|---|---|---|---|
| 10 | Accuracy | 0.9311±5.0989e-005 | 0.9526±1.4511e-005 |
| | time | 23.3933±0.9677 | 28.5940±3.2405 |
| 15 | Accuracy | 0.9252±1.0573e-004 | 0.9494±1.3378e-005 |
| | time | 18.0106±0.2434 | 23.1904±1.0894 |
| 20 | Accuracy | 0.9166±2.8267e-005 | 0.9411±5.4699e-004 |
| | time | 12.7685±0.0966 | 16.2759±0.8880 |
| 25 | Accuracy | 0.9150±7.0000e-005 | 0.9321±6.4810e-004 |
| | time | 9.9201±0.3696 | 13.8645±1.5093 |
| 30 | Accuracy | 0.9079±1.0366e-004 | 0.9192±5.3796e-004 |
| | time | 8.4064±0.0784 | 11.3922±0.0658 |

Table 4. Classification accuracy (mean±standard deviation) and Time cost (seconds: mean± standard deviation) on LETTER dataset at different values of m.

| m | Criterion | RC-kNN | LC-kNN |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 10 | Accuracy | 0.7892±3.8822e-005 | 0.9495±1.0760e-006 |
| | time | 3.2391±0.0015 | 3.2994±0.0010 |
| 15 | Accuracy | 0.7932±3.7106e-005 | 0.9469±5.5751e-006 |
| | time | 3.3808±0.0435 | 3.4334±0.0585 |
| 20 | Accuracy | 0.6815±1.3812e-004 | 0.9451±1.9756e-006 |
| | time | 3.0938±5.8392e-004 | 3.1285±3.1243e-004 |
| 25 | Accuracy | 0.7279±5.6480e-005 | 0.9423±5.2818e-006 |
| | time | 3.3950±0.0018 | 3.4813±0.0054 |
| 30 | Accuracy | 0.6214±9.8480e-005 | 0.9403±3.9204e-006 |
| | time | 3.0889±1.3000e-003 | 3.1168±3.8514e-004 |

Table 5. Classification accuracy (mean±standard deviation) and Time cost (seconds: mean± standard deviation) on PENDIGITS dataset at different values of m.

| m | Criterion | RC-kNN | LC-kNN |
|---|---|---|---|
| 10 | Accuracy | 0.9452±3.5382e-005 | 0.9721±4.7991e-006 |
| | time | 2.3380±0.0041 | 2.4056±0.0101 |
| 15 | Accuracy | 0.9316±1.0341e-004 | 0.9711±6.0196e-006 |
| | time | 2.5451±0.0011 | 2.5709±0.0089 |
| 20 | Accuracy | 0.9163±1.5515e-004 | 0.9700±2.5390e-006 |
| | time | 2.2233±6.4795e-005 | 2.2554±2.1569e-004 |
| 25 | Accuracy | 0.9216±1.5677e-004 | 0.9687±3.5642e-006 |
| | time | 2.5270±0.0056 | 2.5468±0.0083 |
| 30 | Accuracy | 0.9088±1.8409e-004 | 0.9683±1.5809e-006 |
| | time | 2.1805±7.4785e-005 | 2.2022±8.9611e-005 |

Table 6. Classification accuracy (mean±standard deviation) and Time (seconds: mean± standard deviation) on SATIMAGE dataset at different values of m.

| m | Criterion | RC-kNN | LC-kNN |
|---|---|---|---|
| 10 | Accuracy | 0.8603±8.9122e-005 | 0.8883±8.1139e-006 |
| | time | 1.2868±6.5495e-005 | 1.3027±1.1429e-004 |
| 15 | Accuracy | 0.7917±3.1680e-005 | 0.9468±3.7244e-006 |
| | time | 3.8583±0.0332 | 3.9337±0.0152 |
| 20 | Accuracy | 0.8418±8.8847e-005 | 0.8884±6.8028e-006 |
| | time | 1.2292±3.2277e-005 | 1.2463±4.3126e-005 |
| 25 | Accuracy | 0.7283±1.1039e-004 | 0.9421±8.8449e-006 |
| | time | 3.5062±0.0061 | 3.6287±0.0052 |

| | | | |
|---|---|---|---|
| 30 | Accuracy | 0.8312±3.5146e-004 | 0.8878±4.9556e-006 |
| | time | 1.2225±1.8176e-005 | 1.2396±1.7711e-005 |

Table 7. Classification accuracy (mean±standard deviation) and Time cost (seconds: mean ±standard deviation) on ADNC dataset at different values of m.

| m | Criterion | RC-kNN | LC-kNN |
|---|---|---|---|
| 10 | Accuracy | 0.7024±0.0010 | 0.7667±0.0019 |
| | time | 0.0310±2.7390e-004 | 0.0333±1.9201e-005 |
| 15 | Accuracy | 0.6857±0.0014 | 0.7500±0.0013 |
| | time | 0.0308±7.9360e-006 | 0.0325±3.3635e-005 |
| 20 | Accuracy | 0.6619±0.0029 | 0.7143±0.0028 |
| | time | 0.0295±3.0063e-006 | 0.0313±6.8908e-006 |
| 25 | Accuracy | 0.6548±0.0039 | 0.7071±0.0038 |
| | time | 0.0281±4.8219e-007 | 0.0286±7.0881e-007 |
| 30 | Accuracy | 0.6619±0.0015 | 0.7190±0.0031 |
| | time | 0.0306±1.2641e-005 | 0.0326±5.3013e-006 |

Table 8. Classification accuracy (mean±standard deviation) and Time cost (seconds: mean ±standard deviation) on psMCI dataset at different values of m.

| m | Criterion | RC-kNN | LC-kNN |
|---|---|---|---|
| 10 | Accuracy | 0.4792±0.0082 | 0.5833±0.0089 |
| | time | 0.0168±4.5147e-007 | 0.0171±1.4130e-006 |
| 15 | Accuracy | 0.5125±0.0019 | 0.6042±0.0040 |
| | time | 0.0167±9.4536e-007 | 0.0173±5.6957e-007 |
| 20 | Accuracy | 0.5458±0.0060 | 0.6500±0.0035 |
| | time | 0.0170±1.2416e-006 | 0.0188±1.1793e-005 |
| 25 | Accuracy | 0.5333±0.0053 | 0.6417±0.0035 |
| | time | 0.0163±4.9822e-007 | 0.0286±7.0881e-007 |
| 30 | Accuracy | 0.5000±0.0042 | 0.6125±0.0019 |
| | time | 0.0166±8.6385e-007 | 0.0250±2.2840e-004 |

Table 9. Classification accuracy (mean±standard deviation) and Time cost (seconds: mean ±standard deviation) on MCINC dataset at different values of m.

| m | Criterion | RC-kNN | LC-kNN |
|---|---|---|---|
| 10 | Accuracy | 0.5476±0.0049 | 0.6159±0.0045 |
| | time | 0.0468±2.8577e-005 | 0.0506±5.9543e-005 |

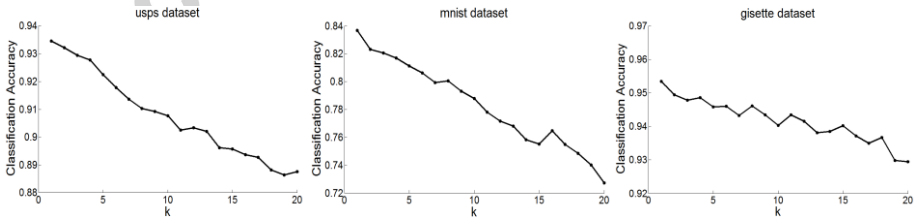
| | | | |
|---|---|---|---|
| 15 | Accuracy | 0.5397±0.0032 | 0.5633±0.0336 |
| | time | 0.0505±2.7585e-005 | 0.0538±4.6465e-005 |
| 20 | Accuracy | 0.5458±0.0060 | 0.6500±0.0035 |
| | time | 0.0452±4.9222e-006 | 0.0480±1.1608e-005 |
| 25 | Accuracy | 0.5222±0.0015 | 0.5746±0.0008 |
| | time | 0.0489±1.5349e-005 | 0.0540±4.5819e-005 |
| 30 | Accuracy | 0.5016±0.0245 | 0.5984±0.0021 |
| | time | 0.0490±2.6160e-005 | 0.0536±4.5625e-005 |

From Table 1~9, we found that the proposed two algorithms needed less time with the larger number of clusters m, while more time cost for the smaller number of m. For example, when m value is 10, i.e., we separated the whole dataset into 10 parts. We then conducted kNN classification for each part, and obtained about one-tenth time cost of the method conducting in the whole dataset.

Besides, considering that the proposed two algorithms are the extension of kNN, their classification accuracies should as close as possible kNN. We found that two algorithms have high classification accuracy with the lager number of clusters m, while these two algorithms have low classification accuracy for the small number of m. In addition, from Table 10, we found that two algorithms were closer to kNN classification accuracy with m =10. Thus, we set m=10 in the next section.

## 4.2 To determine parameter k

In this section, in order to select the appropriate k value, a group of experiments were conducted on nine datasets with m=1 to 20, each point represent the mean of 10 results in the figures.



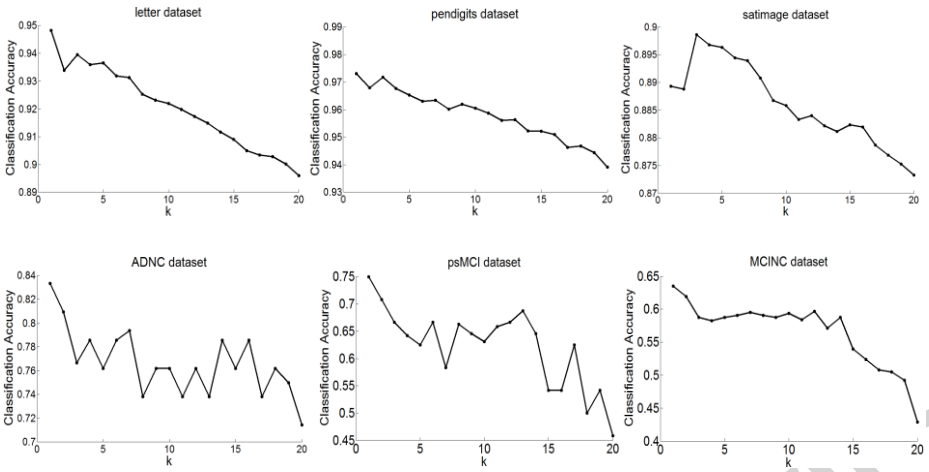



Fig. 1. Classification accuracy of LC-kNN on nine dataset with different k

From Fig. 1, with the increase of k value, the overall of classification accuracy decreases. That is because that the smaller the training dataset is, the more the classification accuracy is. Hence, the difference between the samples is significant and the classification accuracy will reduce. We can make a conclusion that we should choose a suitable k value in this case. According to the previous analysis, k value should be set as small as possible in the case of the higher accuracy of the proposed algorithm. From Fig. 1, the classification accuracy was 0.8986 with k=3, which was higher than k=1 on satimage dataset. But the LC-kNN performance has higher than kNN with k=1 in Table 10. Thus we set k=1 in the next experiment.

### 4.3 Performance comparison of kNN, RC-kNN and LC-kNN

In this experiment, according to previous analysis, we set m=10, k=1. Then, we use classification accuracy and running time as the evaluations for the classification task [23]. The shorter time and higher accuracy the algorithm is, the better the performance is. We report our experimental results in Table 10.

Table 10. Classification Accuracy and Time Cost of three algorithm on nine dataset

| Dataset | RC-kNN | | LC-kNN | | kNN | |
|---|---|---|---|---|---|---|
| | Accuracy | time | Accuracy | time | Accuracy | time |
| USPS | 0.9027 | 3.5589 | 0.9355 | 3.7605 | 0.9482 | 32.8764 |

| | | | | | | |
|---|---|---|---|---|---|---|
| MNIST | 0.7221 | 2.9369 | 0.8389 | 3.5504 | 0.8635 | 24.1575 |
| GISETTE | 0.9311 | 23.3933 | 0.9526 | 28.594 | 0.9660 | 217.3327 |
| LETTER | 0.7892 | 3.2391 | 0.9495 | 3.2994 | 0.9518 | 19.8246 |
| PENDIGITS | 0.9452 | 2.3380 | 0.9721 | 2.4056 | 0.9780 | 7.2982 |
| SATIMAGE | 0.8603 | 1.2868 | 0.8883 | 1.3027 | 0.9065 | 3.5525 |
| ADNC | 0.7024 | 0.0310 | 0.7667 | 0.0333 | 0.7857 | 0.0355 |
| psMCI | 0.4792 | 0.0168 | 0.5833 | 0.0171 | 0.6042 | 0.0176 |
| MCINC | 0.5476 | 0.0468 | 0.6159 | 0.0506 | 0.6413 | 0.0575 |

From Table 10, we can observe that the proposed RC-kNN and LC-kNN improved by 7~9 times than the kNN (close to the number of cluster), in terms of time cost. In the evaluation of classification accuracy, the LC-kNN and RC-kNN is lower by 1%~2.6% and 3.3%~14% than kNN. Therefore, according to the experimental results, we may the conclusion that the LC-kNN works well in terms of classification accuracy and time.

## 5 Conclusions and Future Work

In this work, we have proposed an efficient kNN classification to conduct a k-means clustering to separate the whole dataset into several parts. We then conducted kNN classification for each part. To do this, we parted the conventional kNN method into two processes, namely training process and testing process. Moreover, we analyzed the suitable value for the parameters, such as m and k. Furthermore, we took the kNN as the baseline and conducted a groups of compared experiments among kNN, LC-kNN and RC-kNN. The experimental results showed that the proposed kNN classification worked well in terms of accuracy and efficiency, and it is appropriate to deal with big data.



Acknowledgements

The authors are supported by the China 863 Program (Grant No: 2012AA011005), the China 973 Program (Grant No: 2013CB329404), the Natural Science Foundation of China (Grants No: 61170131, 61450001 and 61263035), the Guangxi Natural Science Foundation for Teams of Innovation and Research (Grant No: 2012GXNSFGA06000

4), the funding of Guangxi 100 Plan, and the Guangxi "Bagui" Teams for Innovation and Research, Innovation Project of Guangxi Graduate Education (Grant NO: YCSZ2 015095, YCSZ2015096).

.

## References

1. A. Andrew, M. Jordan, Y. Weiss. On spectral clustering: Analysis and and algorithm. In Advance in Neural Information Processing Systems 14. MIT Press.
2. K. Bache and M. Lichman. UCI machine learning repository, 2013.
3. C.-C. Chang and C.-J Lin. LIBSVM: A library for support vector machines. ACM Trans-on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
4. X. Chen, D. Cai. Large Scale Spectral Clustering with Landmark-Based Representation. AAAI, 313-318, 2011.
5. M. Filippone, F. Camestra, F. Masulli, et al. A survey of kernel and spectral methods for clustering. Pattern Recognition, 41:176-190, 2007.
6. Y. Gao, M. Wang, Z. Zha, J. Shen, X. Li, X. Wu. Visual-Textual Joint Relevance Learning for Tag-Based Social Image Search, IEEE Transactions on Image Processing, 22(1):363-376, 2013.
7. Y. Gao, M. Wang, D. Tao, R. Ji, Q. Dai. "3D Object Retrieval and Recognition with Hypergraph Analysis", IEEE Transactions on Image Processing, 21(9): 4290-4303, 2012.
8. U. Lall and A. Sharma. A nearest neighbor bootstrap for resampling hydrologic time series. Water Resources Research, 32(3):679–693, 1996.
9. R. Li, Y. Hu. A density-based method for reducing the amount of training data in kNN text classification. Journal of Computer Research and Development. 41(4):539-545, 2004
10. W. Liu, J. He, S. Chang. Large graph construction for scalable semi-supervised learning. In Proceedings of the 27th International Conference on Machine Learning, 2010.
11. X. Wu, C. Zhang, S. Zhang. Database classification for multi-database mining. Information System, 30(1):71-88, 2005.
12. X. Wu, S. Zhang. Synthesizing High-Frequency Rules from Different Data Sources. IEEE Transactions on Knowledge and Data Engineering, 15(2):353-367, 2003.

13. X. Wu, C. Zhang, S. Zhang. Efficient mining of both positive and negative association rules. ACM Transactions on Information Systems, 22(3):381-405, 2004.
14. S. Zhang. KNN-CF approach: Incorporating certainty factor to knn classification. IEEE Intelligent Informatics Bulletin, 11(1):24–33, 2010.
15. Y. Zhao, S. Zhang. Generalized dimension-reduction framework for recent-biased time series analysis. IEEE Transactions on Knowledge and Data Engineering, 18(2):231-244, 2006.
16. D. Zhao, Wei. Zou, G. Sun. A Fast image Classification Algorithm using Support vector Machine. In ICCTD, 2010.
17. X. Zhu, Z. Huang, H. Cheng, J. Cui, H. Shen. Sparse hashing for fast multimedia search. ACM Transaction on Information Systems, 31(2):9, 2013.
18. X. Zhu, Z. Huang, H. Shen, X. Zhao. Linear cross-modal hashing for efficient multimedia search. In ACM Multimedia, pages 143-152, 2013.
19. X. Zhu, Z. Huang, H. Shen, J. Cheng, C. Xu. Dimensionality reduction by mixed kernel canonical correlation analysis. Pattern Recognition, 45(8):3003-3016, 2012.
20. X. Zhu, Z. Huang, Y. Yang, H. Shen, C. Xu, J. Luo. Self-taught dimensionality reduction on the high-dimensional small sized data. Pattern Recognition, 46(1):215-229, 2013.
21. X. Zhu, L. Zhang and Z. Huang, A Sparse Embedding and Least Variance Encoding Approach to Hashing, IEEE Transactions on Image Processing, 2014.
22. X. Zhu, S. Zhang, Z. Jin, Z. Zhang, Z. Xu: Missing Value Estimation for Mixed-Attribute Datasets, IEEE Transactions on Knowledge and Data Engineering, 23(1):110-121, 2011.
23. X. Zhu, Z. Huang, J. Cui, H. T. Shen. "Video-to-Shot Tag Propagation by Graph Sparse Group Lasso". IEEE Transactions on Multimedia, 15(3): 633-646, 2013.
24. X. Zhu, H.-I Suk, D. Shen. A novel matrix-similarity based loss function for joint regression and classification in ad diagnosis. NeuroImage, 2014.
25. X. Zhu, H.-I Suk, D. Shen. Matrix-similarity based loss function and feature selection for alzheimer's disease diagnosis. In CVPR, pages 3089-3096, 2014.
26. X. Zhu, X. Li, and S. Zhang, Block-Row Sparse Multiview Multilabel Learning for Image Classification, IEEE Transactions on Cybernetics, accepted, 2015.

Shichao Zhang is a Distinguished Professor and the director of Institute of School of Computer Science and Information Technology at the Guangxi Normal University, Guilin, China. He holds a Ph.D. degree in Computer Science from Deakin University, Australia. His research interests include data analysis and smart pattern discovery. He has published over 50 international journal papers and over 60 international conference papers. He has won over 10 nation-class grants, such as the China NSF, China 863 Program, China 973 Program, and Australia Large ARC. He is an Editor-in-Chief for International Journal of Information Quality and Computing, and is served as an associate editor for IEEE Transactions on Knowledge and Data Engineering, Knowledge and Information Systems, and IEEE Intelligent Informatics Bulletin.