Neurocomputing 195 (2016) 143–148

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Efficient *k*NN classification algorithm for big data

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

Guangxi Key Lab of Multi-Source Information Mining & Security,Guangxi Normal University, Guilin, Guangxi 541004, China

ARTICLE INFO

Article history:
Received 17 March 2015
Received in revised form
1 July 2015
Accepted 7 August 2015
Available online 5 February 2016

Keywords:
Classification
*k*NN
Big data
Data cluster
Cluster center

ABSTRACT

K nearest neighbors (*k*NN) is an efficient lazy learning algorithm and has successfully been developed in real applications. It is natural to scale the *k*NN 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 *k*NN classification. We conduct sets of experiments on big data and medical imaging data. The experimental results show that the proposed *k*NN classification works well in terms of accuracy and efficiency.

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 (*k*NN), so that these methods can be easily used in big data. Due to the simplicity, easy-understand and relatively high performance of *k*NN, this paper focuses on scaling the *k*NN classification into the application of big data [13].

The previous *k*NN 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, *k*NN 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 traditional *k*NN method to be used in big data [21]. Obviously, conducting *k*NN 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 *k*NN 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 *k*NN 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 *k*NN 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 *k*NN methods. We then propose the new *k*NN 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 *k*NN 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 *k*NN 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 *k*NN [22]. For instance, Zhang proposed a Certainly Factor (CF) measure to deal with the unsuitability of skewed class distribution in *k*NN 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.

*k*NN 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

* Corresponding authors.
E-mail address: zhangsc@mailbox.gxnu.edu.cn (S. Zhang).

http://dx.doi.org/10.1016/j.neucom.2015.08.112
0925-2312/

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 ($\ll n$) 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 $\mathbf{Z}$. 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 $\mathbf{W}$ which is projection matrix of $\mathbf{X}$ at the landmark matrix $\mathbf{Z}$ [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 $\mathbf{Z}$, and $\mathbf{Z}_{<i>}$ denote a submatrix of $\mathbf{Z}$ composed of r nearest landmarks of $\mathbf{x}_i$. Here we need $O(pmn)$ to construct $\mathbf{W}$. $K_h(\cdot)$ is a kernel function with a bandwidths h. The Gaussian kernel $K_h(\mathbf{x}_i,\mathbf{z}_j)=\exp\left(-\|\mathbf{x}_i-\mathbf{z}_j\|^2/2h^2\right)$ 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 $\mathbf{D}$ is the row sum of $\mathbf{W}$. Note that each column of $\mathbf{W}$ sums up to 1 and thus the degree matrix of $\mathbf{G}$ is $\mathbf{I}$.

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 $\mathbf{U}$ within $O(p^3)$, linear to the sample size. $\mathbf{V}$ can be compute as:

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

The overall time complexity of $\mathbf{V}$ is $O(p^3+p^2n)$, which is a significant reduction from $O(n^3)$ by considering $p\ll n$. Each row of $\mathbf{V}$ 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 $\mathbf{Z}$ between data points and landmark samples, with the affinity calculated according to Eq. (1);
3 Compute the first k eigenvectors of $\mathbf{W}\mathbf{W}^T$, denoted by $\mathbf{U}=[\mathbf{u}_1,\mathbf{u}_2,...,\mathbf{u}_k]$;
4 Compute $\mathbf{V}=[\mathbf{v}_1,\mathbf{v}_2,...,\mathbf{v}_k]$ according to Eq. (4);
5 Each row of $\mathbf{V}$ is a data point and apply k-means to get the clusters.

3.2. Testing process

Suppose 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.