Hindawi
Advances in Multimedia
Volume 2017, Article ID 3695323, 9 pages
https://doi.org/10.1155/2017/3695323

Research Article

A Novel DBSCAN Based on Binary Local Sensitive Hashing and Binary-KNN Representation

Qing He,[1,2] Hai Xia Gu,[1] Qin Wei,[2] and Xu Wang[1]

[1]*College of Big Data and Information Engineering, Guizhou University, Guiyang 550025, China*
[2]*Guizhou University, Guizhou Provincial Key Laboratory of Public Big Data, Guiyang, Guizhou 550025, China*

Correspondence should be addressed to Qin Wei; weiq@gzu.edu.cn

Received 17 August 2017; Accepted 12 November 2017; Published 7 December 2017

Academic Editor: Fumin Shen

We revisit the classic DBSCAN algorithm by proposing a series of strategies to improve its robustness to various densities and its efficiency. Unlike the original DBSCAN, we first use the binary local sensitive hashing (LSH) which enables faster region query for the k neighbors of a data point. The binary data representation method based on k neighborhood is then proposed to map the dataset into the Hamming space for faster cluster expansion. We define a core point based on binary influence space to enhance the robustness to various densities. Also, we propose a seed point selection method, which is based on influence space and k neighborhood similarity, to select some seed points instead of all the neighborhood during cluster expansion. Consequently, the number of region queries can be decreased. The experimental results show that the improved algorithm can greatly improve the clustering speed under the premise of ensuring better algorithm clustering accuracy, especially for large-scale datasets.

1. Introduction

Clustering studies [1, 2] play important roles in many fields including data mining and machine learning. The purpose of clustering is to partition the dataset into different subclasses in such a way that the similarities of objects in the same subclass are maximized and the similarities of objects between different subclasses are minimized. DBSCAN, a density-based clustering algorithm [3, 4], basic idea is to connect the adjacent areas with high density which exceeded the threshold. Different from k-means [5, 6], which are based on partitioned strategy, DASCAN does not need to initially set the number of data clusters; it is insensitive to noise and can identify clusters of any shape. However, it uses global parameters $\boldsymbol{\varepsilon}$ and *Minpts* to measure the density so that it does not perform well when the density and interclass distance distribution inconsistent. If the value of $\boldsymbol{\varepsilon}$ is higher, the data points in the cluster with relatively lower density will be defined as the boundary points, and the sparse cluster will be divided into several similar classes. On the contrary, if we give a lower $\boldsymbol{\varepsilon}$, it will combine those clusters who are closer with larger density. Consequently, DBSCAN only plays well in well-proportioned datasets. Also, to find the core points for cluster expansion, DBSCAN continuously queries the neighborhood so as to have a considerable I/O cost especially for the large-scale datasets. To overcome the flaws of DBSCAN, scholars have done a lot of research and put forward a lot of improvement methods. Zhu et al. [7] proposed the ReCon and ReScale approaches based on density-ratio to improve the limitation of DBSCAN in finding clusters of varying densities. But ReCon-DBSCAN use a density estimator (η) and ReScale-DBSCAN use two additional parameters (ε, η) which increase the reliance on parameters of the algorithm to compute density-ratio. ZHOU Shui-geng et al. proposed PDBSCAN and FDBSCAN algorithm to solve the defect of DBSCAN, respectively. PDBSCAN [8] divides the data space into several uniformed areas according to the histogram statistical analysis results of the data in one or more dimensions; then it uses different $\boldsymbol{\varepsilon}$ for different regions to solve the weakness for various densities. However, PDBSCAN uses human-computer interaction to achieve data partition; it does not make great difference in practical application, although the clustering quality is better. FDBSCAN [9] algorithm uses only a small number of representative points instead of

all in the neighborhood of a core point as seed points to expand the cluster. Unlike DBSCAN, FDBSCAN reduces the execution frequency of region query and decreases the I/O cost sharply. But it is at the expense of clustering accuracy, and the efficiency of region query has not been improved which means it still has room for improvement.

To address the issue related to DBSCAN, this paper proposes a fast clustering algorithm BLSH-DBSCAN based on collision-ordered LSH and binary k nearest neighbors. This algorithm makes the following contributions:

(i) Using the binary LSH to query the k nearest neighbors, it can improve the speed of region query greatly compared with traditional linear search.

(ii) It constructs a binary-KNN representation method which can map the data into the Hamming space for the next clustering operation and greatly improve the speed of clustering.

(iii) It introduces a core point distinguishing method based on the influence space and designs the solution of influence space in the binary dataset to boost the clustering speed. At the same time, due to the density sensitivity of influence space, this improved method has much better clustering quality and efficiency compared with the original DBSCAN.

(iv) It introduces a seed point selection method, based on influence space and the k neighborhood similarity, to select some seed points instead of all the neighborhood during cluster expansion. It can decrease the execution frequency of region query to realize faster clustering operation.

The rest of the paper is organized as follows.

In Section 2, we provide an explanation of locality sensitive hashing for region query and how to make the binary-KNN representation of a point. An improved density-based clustering algorithm is developed in Section 3. We introduce the influence space and its solving method in binary dataset; also, a seed point selection method is proposed. Section 4 reports the experimental results. Discussion and conclusions are provided in Section 5.

2. Binary Locality Sensitive Hashing and Binary-KNN Representation

2.1. About DBSCAN Algorithm. DBSCAN is a typical density-based spatial clustering algorithm. It has two important parameters $\boldsymbol{\varepsilon}$ and *Minpts*. $\boldsymbol{\varepsilon}$ defines the radius of the neighborhood of a data object, and *Minpts* defines the minimum number of data points contained in the neighborhood. DBSCAN gives the following definitions.

Suppose that we are given a dataset $\mathbf{D} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$.

Definition 1 (directly density reachable). If $\mathbf{p}$ is in the $\boldsymbol{\varepsilon}$ neighborhood of $\mathbf{q}$ and $\mathbf{q}$ is a core object, then object $\mathbf{p}$ is directly density reachable from $\mathbf{q}$.

Definition 2 (density reachable). A point $\mathbf{p}$ is density reachable from a point $\mathbf{q}$, if there is a chain of points $\mathbf{p_1}, \ldots, \mathbf{p_n}$, $\mathbf{p_1} = \mathbf{p}$, $\mathbf{p_n} = \mathbf{q}$, such that $\mathbf{p_{i+1}}$ is directly density reachable from $\mathbf{p_i}$ ($\mathbf{1} \leq \mathbf{i} \leq \mathbf{n}$, $\mathbf{p_i} \in \mathbf{D}$).

Definition 3 (density connected). A point $\mathbf{p}$ is density connected to a point $\mathbf{q}$, if there is a point $\mathbf{o} \in \mathbf{D}$ and both $\mathbf{p}$ and $\mathbf{q}$ are density reachable from $\mathbf{o}$.

Definition 4 (core point). In the ε-neighborhood of point $\mathbf{p}$, if the number of points which are directly density reachable from point $\mathbf{p}$ is greater than *Minpts*, then $\mathbf{p}$ is a core point.

Definition 5 (border point). If $\mathbf{p}$ is not a core point, but $\mathbf{p}$ is directly density reachable from a core point, then $\mathbf{p}$ is a border point.

Definition 6 (noise point). If point $\mathbf{p}$ is neither a core point nor a border point, then $\mathbf{p}$ is a noise point.

To find the cluster, DBSCAN starts with an arbitrary object $\mathbf{p}$ in $\mathbf{D}$ and then retrieves all points which are density reachable from $\mathbf{p}$ with respect to $\boldsymbol{\varepsilon}$ and *Minpts*. If $\mathbf{p}$ is a core point, then mark $\mathbf{p}$ and its ε-neighborhood as a new cluster. Then, DBSCAN continues to retrieve ε-neighborhood of other points in the cluster and adds ε-neighborhood of the core points to the current cluster until no new object can be added to the cluster. When all points have been divided into a cluster or been labeled as a noise point, clustering ends.

However, the neighborhood query needs to calculate the distance between the query object and all other objects by liner search and it has a huge I/O cost. To solve the problem, we propose the following improvements: to accelerate the region query, using the binary LSH rather than linear search to query the k nearest points, and to use the neighbor structure to represent data points which can map the high-dimensional datasets into Hamming space to expedite the clustering expansion.

2.2. Locality Sensitive Hashing. The LSH algorithm is usually for quick neighbor query. It involves two steps: index construction and object query. In index construction, through a set of hash functions, it projects similar data points into the same hash bucket with a higher probability. In object query, it uses a filter-and-refine framework to hash the data into the hash bucket through the same hash functions. All the data points in the hash buckets are adopted as candidates, which are used to calculate the similarity with the query object to find the k nearest neighbors.

Definition of LSH (see [10]). Let $\mathbf{d_1}, \mathbf{d_2}$ be two distances which satisfy the distance function. We call the hash function family $H(\mathbf{d_1}, \mathbf{d_2}, \mathbf{p_1}, \mathbf{p_2})$-sensitive when each function in $\mathbf{H}$ satisfies the following two conditions.

(1) If $\text{dist}(\mathbf{x}, \mathbf{y}) \leq \mathbf{d}_1$, $\mathbf{P}r[\mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{y})] \geq \mathbf{p}_1$.

(2) If $\text{dist}(\mathbf{x}, \mathbf{y}) \geq \mathbf{d}_2$, $\mathbf{P}r[\mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{y})] \leq \mathbf{p}_2$,

where $\mathbf{d}_1 < \mathbf{d}_2, \mathbf{p}_1 > \mathbf{p}_2 \in [0, 1]$.

LSH uses different hash function families for different distance functions. In this paper, binary hash function family

based on p-stable distribution which applies to the Euclidean space under the Lp norm is used. For each high-dimensional data point, the hash function family is [11]

$$h(\mathbf{x_i}) = \begin{cases} 1 & a\mathbf{x_i} > 0 \\ 0 & a\mathbf{x_i} \le 0, \end{cases} \quad (1)$$

where $\mathbf{a}$ is a random vector that follows the p-stable distribution and has the same dimension with $\mathbf{x_i}$.

The index structure of LSH can be summarized as the following two steps [12, 13]:

(i) Giving a set of hash functions $\mathbf{H}$ and a set of vectors $\mathbf{D}$, defining a new hash function family $\mathbf{G}$. For the vectors ($\mathbf{x_i}$) and the hash functions ($\mathbf{h}$) in $\mathbf{H}$ and $\mathbf{D}$, we select r hash functions to carry out the AND structure to construct functions $\mathbf{g}$.

$\mathbf{g}$ belongs to $\mathbf{G}$ and its detailedness is

$$g(\mathbf{x_i}) = \langle \mathbf{h}_1(\mathbf{x_i}), \mathbf{h}_2(\mathbf{x_i}), \ldots, \mathbf{h_r}(\mathbf{x_i}) \rangle. \quad (2)$$

(ii) Selecting an integer $\mathbf{b}$, then randomly selecting $\mathbf{b}$ hash tables ($\mathbf{g}_1(\mathbf{x_i}), \ldots, \mathbf{g_b}(\mathbf{x_i})$) from $\mathbf{G}$ to map the data points into the hash tables.

When using the binary LSH in object query, for each query object $\mathbf{q}$, it selects the same hash functions with those in index structure to calculate the conflicting bucket number of $\mathbf{q}$. It is clear that each set of functions ($\mathbf{g}$) can get a conflicting bucket number of $\mathbf{q}$, which is as $g(\mathbf{q}) = \langle \mathbf{h}_1(\mathbf{q}), \mathbf{h}_2(\mathbf{q}), \ldots, \mathbf{h_r}(\mathbf{q}) \rangle$.

Basic LSH adopts all data objects which have the same conflicting bucket number with the query object as the candidates, and then it compares the similarity between the candidates and the query object to find the k nearest neighbors of the query object.

However, in the k neighbor search of DBSCAN, there are more data points in the area with high density. That is to say, the computational complexity will be so large by comparing the similarity between query object and all the candidates that the query efficiency would not meet the requirements of large-scale datasets. It has been proved in [10] that the more similar the two objects are, the more times would they be mapped into the same hash bucket with the same scale of LSH operations. Therefore, this paper uses the conflict count sorting strategy proposed in [14], descending the candidates by the count number of the conflicts, to select the first k candidates as the neighbors of the query object.

2.3. Binary-KNN Representation. As we all know, the neighbor structure contains strong data class information. It is possible to effectively judge the similarity between the data objects through it [15, 16]. In this section, we propose a binary representation method based on k nearest neighbors. It expresses the neighbor structure in binary to represent the data points, which can map the complex high-dimensional dataset to Hamming space. It is obvious that the clustering in Hamming space will considerably decrease the run time of DBSCAN.

The details can be described as follows. For any data object ($\mathbf{x_i}, \mathbf{i} = \mathbf{1}, \ldots, \mathbf{n}$) in dataset $\mathbf{D}$ and its k neighborhood $\mathbf{NN_k}(\mathbf{x_i})$ ($\mathbf{NN_k}(\mathbf{x_i}) = \{\mathbf{n_{i1}}, \ldots, \mathbf{n_{ij}}, \ldots, \mathbf{n_{ik}}\}$, $\mathbf{n_{ij}}$ is the subscript of the jth neighbor of $\mathbf{x_i}$ which can be found by using the LSH proposed in Section 2.1; we definite its new expression as $\mathbf{x_i'}$ ($\mathbf{x_i'} = \{\mathbf{x'_{i1}}, \ldots, \mathbf{x'_{im}}, \ldots, \mathbf{x'_{in}}\}$). If and only if $\mathbf{m} = \mathbf{n_{ij}}$ *or* $\mathbf{i}$, $\mathbf{x'_{im}} = \mathbf{1}$; otherwise, its value is $\mathbf{0}$. With this, we get the binary-KNN representation of the object $\mathbf{x_i}$ which is as $\mathbf{x_i'}$ and thebinary-KNN representation dataset $\mathbf{D}(\mathbf{x_i'})$ of thedataset D at last.

Different from the original DBSCAN, this paper uses binary LSH rather than linear search to query the k-nearest neighbor and can improve the efficiency of the neighbor query. Also, it transforms the neighbor structure information of a data point into binary to represent the data, by which we can operate the clustering in Hamming space. It would be faster in Hamming space to divide the data points into core points, boundary points, and noise points so that the run time of cluster expansion can be decreased sharply.

3. Improved DBSCAN Clustering Algorithm

3.1. Influence Space in Binary Dataset. Density-based clustering is to find out the area where the density exceeds the threshold. In DBSCAN, it uses global parameters $\boldsymbol{\varepsilon}$ and *Minpts* to measure the density, which leads to a lower clustering quality for datasets with various densities. To improve its robustness for various densities, we introduce a core point distinguishing method which is based on the influence space and its solving method in binary-KNN dataset. Due to the local density sensitive feature of influence space, our method can improve the robustness of DBSCAN in datasets with various densities. Also, by applying the core point distinguishing method in binary Hamming space (Section 2.3), the efficiency will be further improved.

For further explanation, we give the following definitions.

Definition 7 (k-neighborhood-point set $\mathbf{NN_k}$). For $\mathbf{x_i} \in \mathbf{D}$, the k-neighborhood-point set is consisted of k nearest neighbors of $\mathbf{x_i}$, which is expressed as $\mathbf{NN_k}(\mathbf{x_i}) = \{\mathbf{n_{i1}}, \ldots, \mathbf{n_{ij}}, \ldots, \mathbf{n_{ik}}\}$ ($\mathbf{n_{ij}}$ is the subscript of the jth neighbor of $\mathbf{x_i}$).

Definition 8 (core point). For $\mathbf{x_i} \in \mathbf{D}$, if $\mathbf{x_i}$ is a core point, then it meets the following equation:

$$\left|\mathbf{IS}_k(\mathbf{x_i})\right| \ge \sigma\mathbf{k}. \quad (3)$$

Definition 9 (border point). For $\mathbf{x_i} \in \mathbf{D}$, if $\mathbf{x_i}$ is a border point, then it meets the following equation:

$$0 < \left|\mathbf{IS}_k(\mathbf{x_i})\right| < \sigma\mathbf{k}. \quad (4)$$

Definition 10 (noise point). For $\mathbf{x_i} \in \mathbf{D}$, if $\mathbf{x_i}$ is a noise point, then it meets the following equation:

$$\left|\mathbf{IS}_k(\mathbf{x_i})\right| = 0, \quad (5)$$

where $\mathbf{IS}(\mathbf{x_i})$ is the influence space of $\mathbf{x_i}$, which contains the data points in $\mathbf{NN_k}(\mathbf{x_i})$ whose k nearest neighbors also include $\mathbf{x_i}$. $|\mathbf{IS}(\mathbf{x_i'})|$ is the number of points in influence space. $\boldsymbol{\sigma}$ is the weight coefficient, and the general value is 2/3. k is the number of neighbors.

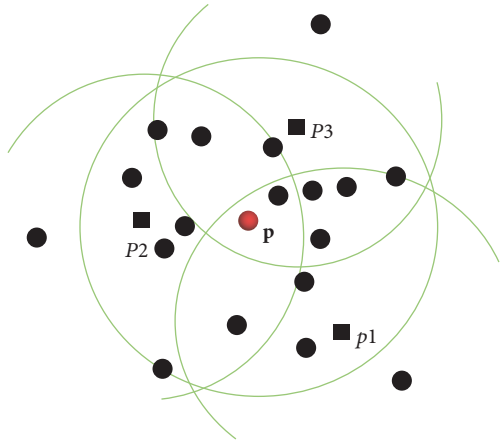

Figure 1: Seed points and k neighborhood in two-dimensional space; p is the core point and $p1$, $p2$, and $p3$ are the seed points.

The influence space $\mathbf{IS}(\mathbf{x_i})$ was first proposed by Jin et al. [17] for estimating the neighborhood density. Different from DBSCAN which is weak in various density, $\mathbf{IS}(\mathbf{x_i})$ is very sensitive to the change in density of local area. By using the influence space, it can improve the clustering quality obviously in the dataset with various densities.

Also, to calculate the $\mathbf{IS}(\mathbf{x_i})$ in binary dataset introduced in Section 2.2, we design a straightforward method as the following equation:

$$\mathbf{IS}_k(\mathbf{x_i}) = \mathbf{IS}_k\left(\mathbf{x_i'}\right) = \mathbf{x_i'} \cup \mathbf{x_i'}^{\mathrm{T}}, \tag{6}$$

where $\mathbf{x_i'}^{\mathrm{T}}$ takes the information of the data points whose $\mathbf{NN_k}$ contains $\mathbf{x_i}$.

$\mathbf{IS}(\mathbf{x_i})$ is just the intersection of $\mathbf{x_i'}$ and $\mathbf{x_i'}^{\mathrm{T}}$. Meanwhile, benefiting from the symmetry of influence space, acquiring $\mathbf{x_i'}^{\mathrm{T}}$ comes to be simple and fast. It first needs a transposition of $\mathbf{D}(\mathbf{x_i'})$, and then $\mathbf{x_i'}^{\mathrm{T}}$ is the vector where $\mathbf{x_i'}$ in $\mathbf{D}(\mathbf{x_i'})$. In general, only one step is needed in calculating the influence space. It greatly simplifies the query step and the algorithm efficiency is further improved.

3.2. Representative Objects Selection. To improve the efficiency of the algorithm, on the one hand, we need to improve the efficiency of the neighbor query which has been solved by LSH and binary-KNN representation in Section 2; on the other hand, we can also reduce the frequency of the neighbor query.

In the cluster expansion of DBSCAN, all points in the neighborhood are selected as the seeds for the next region query. However, our core point distinguishing the method proposed in Section 3.1 is based on influence space which contains the data points in one k neighborhood whose k neighborhood also includes the query object. For an object $\mathbf{p}$, it should be certain that there is overlap between $\mathbf{p}$'s $\mathbf{NN_k}$ and $\mathbf{NN_k}$ of its neighborhood points. When $\mathbf{p}$ is a core point, it is true that there are more points in its influence space. Theoretically, the more the points in influence space, the larger the overlapping area. There is even a case where the neighborhood of object $\mathbf{p}$ is completely covered by the neighborhood of its $\mathbf{NN_k}$ which is shown in Figure 1. Although the object p is a core object, if we choose all points in its neighborhood for the next cluster expansion, it will only increase the frequency of the neighbor query which is not conducive to the algorithm efficiency. Therefore, we need to select part of the data points rather than all neighbors of a core point as seed points for the clustering expansion.

In this section, we introduce a seed point selection method based on influence space and the similarity between each neighborhood. In a neighborhood of a core point, it ascends the points which are in the influence space by $\mathbf{NN_k}$ similarity and then only selects the first few points as the seed points for the next clustering expansion. For detailed explanation, we first give the definition of $\mathbf{NN_k}$ similarity.

Definition 11 ($\mathbf{NN_k}$ similarity). For $\mathbf{x_i}, \mathbf{x_j} \in \mathbf{D}$, $\mathbf{NN_k}$ similarity is as follows:

$$\mathrm{sim}\left(\mathbf{x_i}, \mathbf{x_j}\right) = \mathbf{NN_k}\left(\mathbf{x_i}\right) \cup \mathbf{NN_k}\left(\mathbf{x_j}\right). \tag{7}$$

It counts the same neighbors of the two objects in their neighborhood. In binary-KNN representation dataset, the $\mathbf{NN_k}$ similarity is as follows:

$$\mathrm{sim}\left(\mathbf{x_i}, \mathbf{x_j}\right) = \mathbf{x_i'} \cup \mathbf{x_j'}. \tag{8}$$

The detailed explanation for how to select the seed point in its $\mathbf{NN_k}$ of a core point is as the following three steps.

(1) Firstly, we ascend the points in the influence space of $\mathbf{p}$ by $\mathbf{NN_k}$ similarity.

(2) For the first point $\mathbf{q}$ in sorted influence space, estimate whether it is core point. If $\mathbf{q}$ is a core point, take it as the first seed point and then find the point in $\mathbf{p}$'s influence space who has the lowest $\mathbf{NN_k}$ similarity with the seed points which has been selected. Otherwise, no action should be taken.

(3) Repeat step (2) until all the required seed points are found or all points are processed.

Here we explain why we choose objects with the lowest $\mathbf{NN_k}$ similarity in the influence space. The selection of seed points is a trade-off between the core point and the overlap with the neighborhood. For a seed point, we want less overlap of the neighborhood with the current core object and higher probability of becoming a core object. At present, because theoverlap between neighborhoods of two objects is measured by $\mathbf{NN_k}$ similarity, it is obvious that the lower $\mathbf{NN_k}$ similarity can directly diminish the overlap of the neighborhood and further reduce the frequency of region query. Also, the points in the influence space of a core point are the ones whose neighborhood contains the core point in turn. They may have a higher probability of being a core point. Therefore, we design the selection method of seed points according to $\mathbf{NN_k}$ similarity and the influence space to reduce the times of neighborhood query. Consequently, the efficiency of the algorithm will be further improved.

3.3. Steps of BLSH-DBSCAN. The steps of the improved algorithm are as follows:

(1) Data input: dataset $\mathbf{D}(\mathbf{x_i}) = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$.

(2) Index and query: construct the index of all data points by using the binary LSH within collision order; then

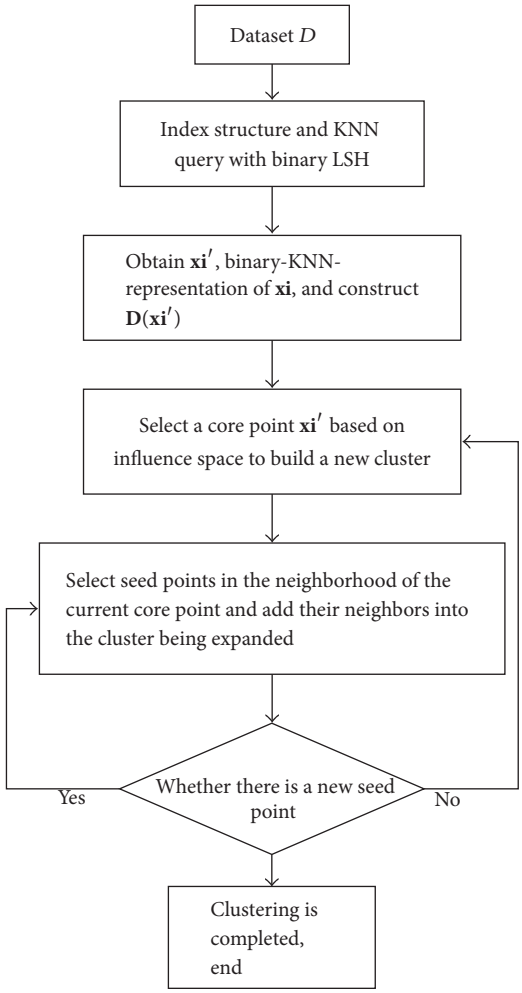

Figure 2: Flowchart of the improved algorithm.

query the k neighbors of each point, saved as $\mathbf{NN_k}(\mathbf{x_i}) = \{\mathbf{n_{i1}}, \ldots, \mathbf{n_{ij}}, \ldots, \mathbf{n_{ik}}\}$.

(3) Binary-KNN representation: according to the KNN information, represent the data point $\mathbf{x_i}$ as the $\mathbf{x_i'} = \{\mathbf{x_{i1}'}, \ldots, \mathbf{x_{im}'}, \ldots, \mathbf{x_{in}'}\}$, when $\mathbf{m} = \mathbf{n_{ij}}$, $\mathbf{x_{im}} = \mathbf{1}$; otherwise, the value is $\mathbf{0}$. At last convert the dataset $\mathbf{D}(\mathbf{x_i})$ into $\mathbf{D}(\mathbf{x_i'}) = \{\mathbf{x_1'}, \mathbf{x_2'}, \ldots, \mathbf{x_n'}\}$.

(4) Calculate $\mathbf{IS}(\mathbf{x_i'})$ of the data point $\mathbf{x_i'}$ which has not been divided into a cluster or marked as noise point, when $|\mathbf{IS}(\mathbf{x_i'})| \geq \boldsymbol{\sigma}\mathbf{k}$, establish a new cluster C, and put $\mathbf{NN_k}$ of $\mathbf{x_i'}$ in cluster C.

(5) Select seed points in $\mathbf{NN_k}(\mathbf{x_i'})$ and add their k neighbors in cluster C.

(6) Repeat step (5), select seed points in the k neighborhood of the current processed core point, and then add their k neighbors to C until there are no new points that can be added.

(7) Repeat steps (4), (5), and (6), until all points have been divided into some cluster or marked as noise point.

The flowchart is as Figure 2.

Table 1: Test datasets summary.

Dataset	Objects	Dimensions
Synthetic dataset	5000	5
Synthetic dataset	10000	5
	...	
Synthetic dataset	60000	5

4. Experiments and Performance Evaluation

To evaluate our approaches, we demonstrate the superiority of BLSH-DBSCAN in three aspects, the query time of neighborhood, clustering speed, and clustering quality in both synthetic datasets and real-world datasets. All the experiments are implemented in MATLAB under windows operating system. In the following experiments, we compare our BLSH-DBSCAN with both DBSCAN and IS-DBSCAN. The reason why we choose DBSCAN for comparison is that DBSCAN is the original algorithm; it makes great sense for illustrating the effectiveness of the improvement method by comparing the clustering quality and speed with each other. The reason we compare the BLSH-DBSCAN with the algorithm IS-DBSCAN is that they all use influence space to enhance the robustness in the various-density dataset. Unlike IS-DBSCAN, our BLSH-DBSCAN operates the clustering in Hamming space and adopts a seed point selection strategy to reduce the frequency of neighborhood query. Comparing the two algorithms can further illustrate the effectiveness of our improved strategy.

4.1. Query Time Comparison. Unlike the DBSCAN which queries the neighborhood by linear search, our BLSH-DBSCAN uses the binary LSH to query the k nearest neighbors of each point. To compare the query time of each other, the experiments are performed in the synthetic datasets described in Table 1. Our synthetic datasets vary from 5000 to 60000 points, which are generated based on multiple-Gaussian distribution. We record the 10-time query time of both linear search and LSH in the datasets in Table 1 and then calculate their averages.

Figure 3 is a chart compared with MATLAB which shows the average run time of both linear search and LSH in datasets with different scale. It can be seen from Figure 3 that it is only in the dataset with small scale that the linear search has a weak advantage. The larger the scale of the dataset, the bigger the gap between their run time. For a dataset with 60000 points, the query time of linear search is almost 200 seconds, but of binary LSH, it is stable in a few seconds and the change is not very obvious. This means that it is practicable to apply the binary LSH for neighbor query to improve the efficiency of the algorithm.

4.2. Clustering Quality Comparison. The BLSH-DBSCAN has introduced the binary influence space to improve the clustering quality in the datasets with various densities. To illustrate the positive influence of our improve strategy on the clustering quality, we carry out the experiments in several synthetic datasets which are introduced in Table 2.

Table 2: Overview of the synthetic datasets used in experiments.

Dataset	Objects	Dimension	Label	Characteristics of datasets
Dataset 1	1502	2	2	Different shape
Dataset 2	1419	2	7	Different shape, size, and density

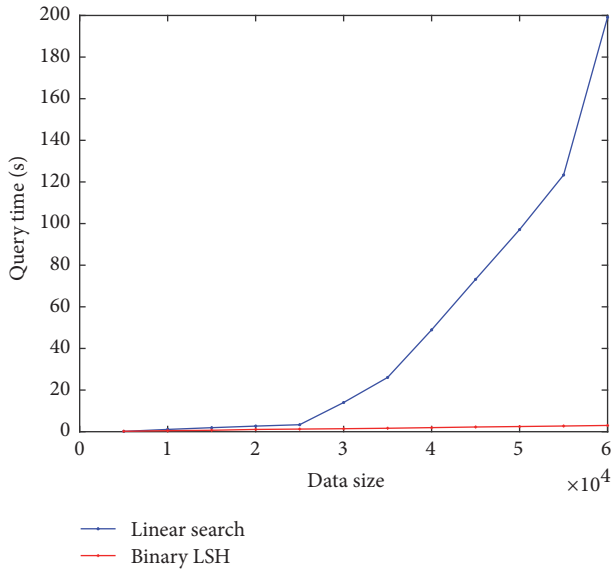

Figure 3: Comparison of query time.

Since DBSCAN uses the global parameters $\boldsymbol{\varepsilon}$ and *Minpts*, in the following experiments, we constantly adjust $\boldsymbol{\varepsilon}$ and the *Minpts* accordingly to obtain the best clustering results. BLSH-DBSCAN uses the global parameter k, and we continue to adjust the value of k and ultimately find the best clustering results. The best results are shown in Figure 4, the first row is the best cluster results in dataset 1, and the second row shows the best cluster results of dataset 2.

From the first row of Figure 4, we can see that the improved algorithm in this paper shows the advantages of DBSCAN which can identify clusters of any shapes as dataset 1. The second row of Figure 4 shows the best clustering results of two algorithms for dataset 2 which is a dataset with different shapes, sizes, and densities. From the results of the clustering, we can see that DBSCAN incorrectly merges the two data clusters that are close to each other into one. It is because that DBSCAN uses global parameters ε and Minpts to measure the density which is not suitable for dataset 2 who has clusters with different sizes and densities. If we set a higher ε, the data points in the cluster with relatively lower density will be defined as the boundary points which leads the sparse cluster to be divided into several similar classes. If we give a lower ε, it will combine those clusters who are closer to each other with larger density. Our BLSH-DBSCAN can accurately identify each cluster of dataset 2. It further shows the fact that the density measurement strategy based on the influence space can improve the quality of DBSCAN in dataset with various densities.

Table 3: Overview of the UCI datasets used in experiments.

Dataset	Object	Dimension	Label
Iris	150	4	3
Contraceptive Method Choice	1437	9	3
Letter Recognition	20000	16	26

Table 4: Correct rate comparison in different UCI datasets.

	DBSCAN	ISB-DBSCAN	BLSH-DBSCAN
Iris	69.33%	88%	89.33%
Contraceptive Method Choice	42.70%	44.89%	42.70%
Letter Recognition	50.2%	64.45%	64.425%

4.3. Performance on Real-World Datasets. In this section, we will compare the clustering efficiency and clustering accuracy in real-world datasets. We use run time to represent the clustering efficiency and the clustering correct rate to represent the clustering accuracy. Because the points in experimental datasets are all classified, the correct rate in this section is obtained by comparing the clustering results of the algorithm with the original label of the data points.

BLSH-DBSCAN adopts several strategies to enhance the clustering speed. It uses binary LSH which is a fast neighborhood query algorithm to speed up the region query. It adopts the binary-KNN representation method to map the clustering operation in Hamming space. It also selects few seed points instead of all neighbors for cluster expansion to decrease the frequency of region query. These methods all have improved the clustering efficiency to some extent. In order to illustrate the efficiency of these methods, we select several datasets from UCI datasets and then compare the run time and clustering accuracy of BLSH-DBSCAN, DBSCAN, and IS-DBSCAN. Table 3 shows the details of the experimental datasets. These datasets are different in scale, dimension, and number of clusters so that the comparison would be convictive.

Table 4 and Figure 5 show, respectively, the clustering correct rate and run time of DBSCAN, IS-DBSCAN, and BLSH-DBSCAN for UCI dataset Iris, Contraceptive Method Choice, and Letter Recognition. It needs to be explained that the test results of the three algorithms on different datasets are the best test results by adjusting the parameters.

Correct rates shown in Table 4 are the best clustering results by adjusting the parameters of the three algorithms. Run time shown in Figure 5 is the average of the 10 with the same parameters and the parameters are those when the correct rate is the highest.

From Table 4, we can see that the clustering accuracy of IS-DBSCAN and BLSH-DBSCAN in the three datasets is superior to that of the traditional DBSCAN, which further proves the superiority of the core point distinguishing method which is based on the influence space. From Figure 4, it is obvious that there is a big gap between the run times of

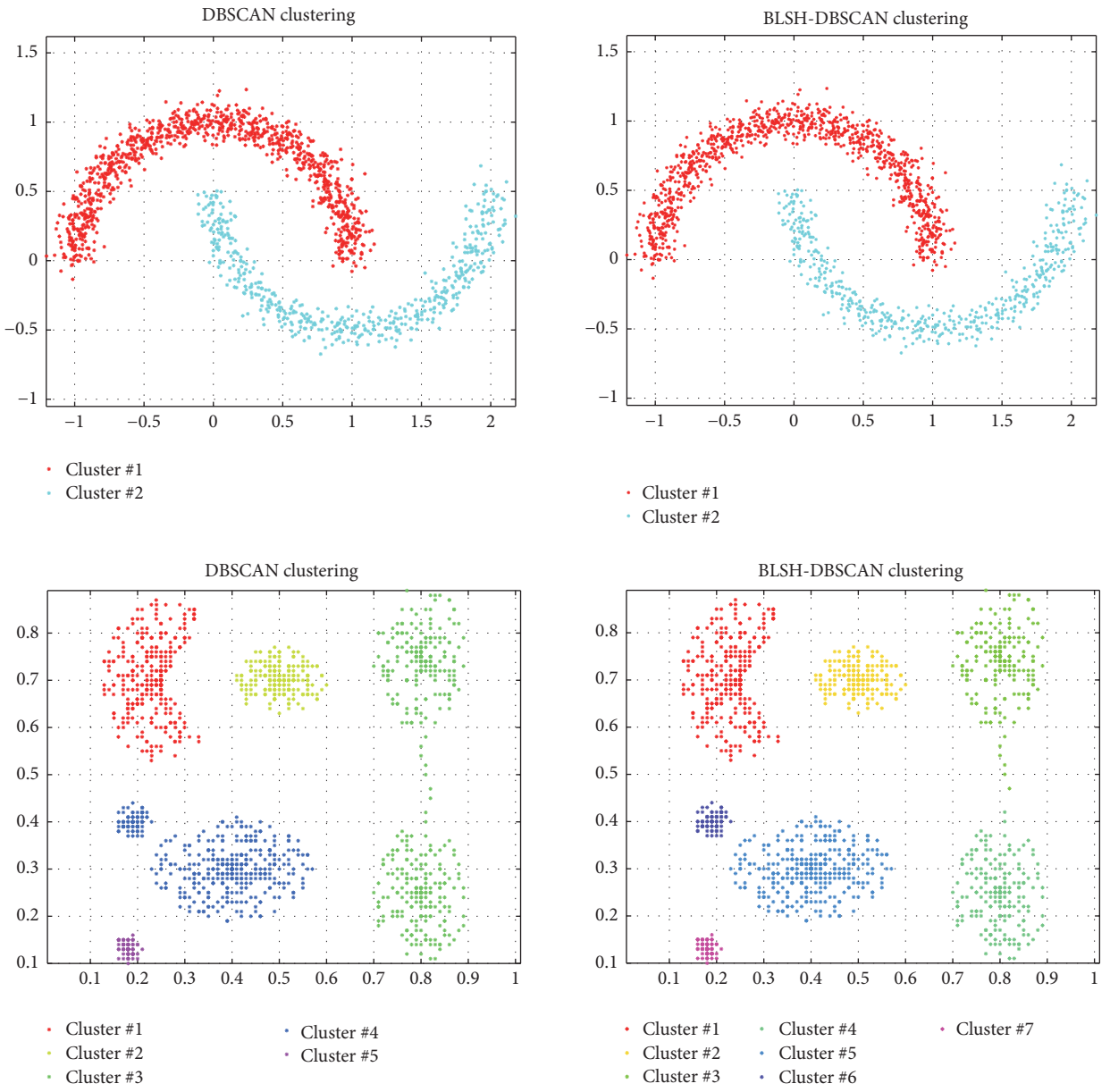

Figure 4: Best cluster results in different datasets.

the three algorithms in different datasets. The run time of IS-DBSCAN is always longer than that of DBSCAN. It is because that IS-DBSCAN uses the influence space for distinguishing the core point and the calculation is more complex. Therefore, IS-DBSCAN takes longer run time than DBSCAN. However, our improved algorithm has no advantage of run time on the dataset Iris. That is because Iris is a typical dataset with small scale in which, as we can see from Figure 2, the query time of LSH is much longer than the linear search. In Contraceptive Method Choice dataset, since its scale is not large, the query time of LSH is still slightly longer than the linear query. In addition, from the second figure in Figure 5, we can see that the run time of BLSH-DBSCAN is basically the same with DBSCAN and much less than the IS-DBSCAN. It indirectly illustrates the positive influence of our improvement strategy including binary influence space and seed point selection method on decreasing the run time of the algorithm. In the large-scale Letter Recognition dataset, the advantage is so obvious that the run time of BLSH-DBSCAN is much shorter than that of the other two algorithms. It shows that our improved algorithm can take full advantage of the LSH, binary influence, and seed point selection.

In conclusion, in the small-scale dataset, BLSH-DBSCAN can greatly improve the clustering accuracy just as the IS-DBSCAN. In the large-scale dataset, compared with DBSCAN, it can get a higher accuracy and efficiency; compared with IS-DBSCAN, it can decrease the run time sharply while maintaining the same level of accuracy.

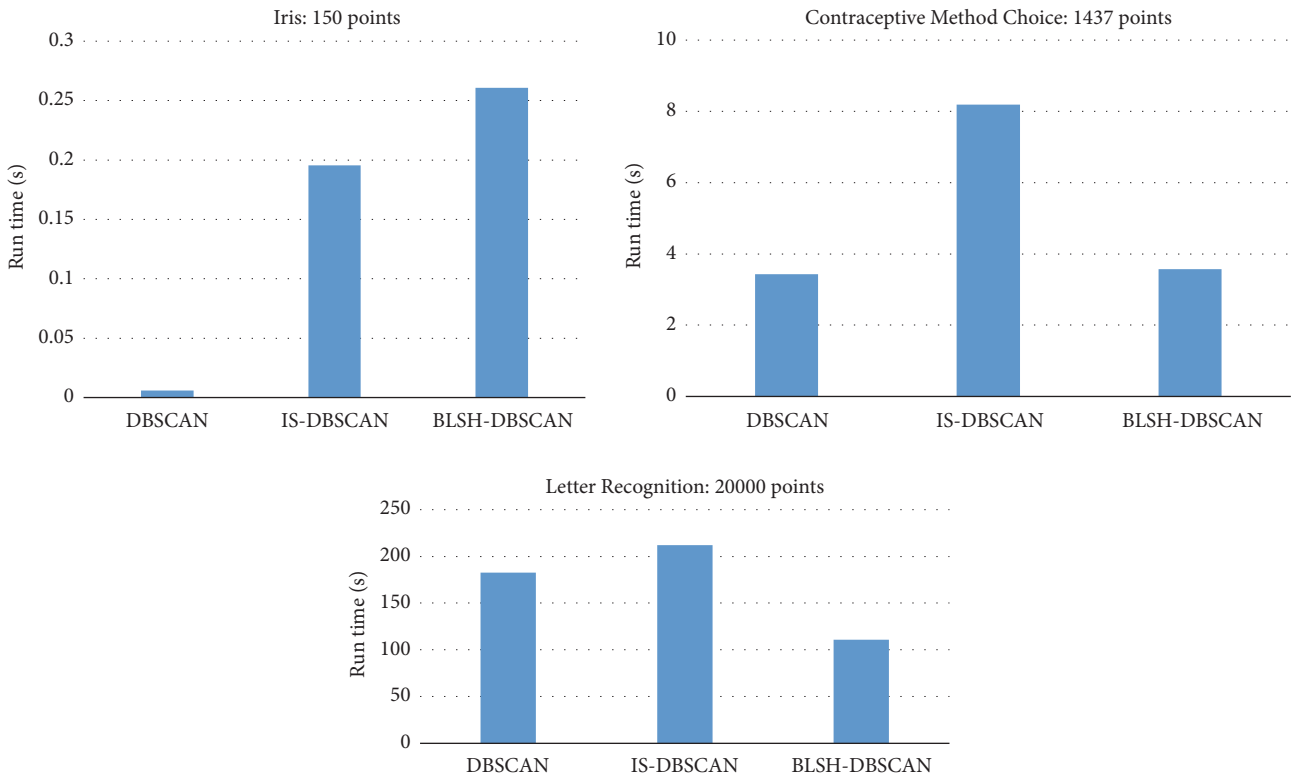

Figure 5: Comparison of run time of the DBSCAN, IS-DBSCAN, and BLSH-DBSCAN.

5. Conclusion

In this paper, an improved DBSCAN algorithm is proposed to improve the robustness for various densities and clustering efficiency of the algorithm. The improved strategy includes using binary LSH instead of linear search for region query; designing a binary representation method based on k neighborhood to map the clustering into Hamming space; and introducing a seed point selection method based on influence space and $\mathbf{NN_k}$ similarity for clustering expansion. By comparing the improved algorithm with DBSCAN and its improved variant, it shows that our improved algorithm has a higher clustering accuracy in small-scale datasets and has considerable advantage in both clustering accuracy and efficiency in large-scale dataset. Therefore, our improved algorithm is especially suitable for large-scale datasets for faster and more accurate clustering results.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by National Key Basic Research Program of China (973 Program) (Project 2015CB856001) and by Guizhou Provincial Key Laboratory of Public Big Data (2017BDKFJJ002 and 2017BDKFJJ004); this research is also funded by project of Guizhou Provincial Education Department (KY[2016]124) and project of Department of Science and Technology of Guizhou Province (LH[2014]7628).

References

[1] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[2] R. Xu and D. C. W. Ii, "IEEE, survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, 2005.

[3] M. Ester, H. P. Kriegel, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 226–231, AAAI Press, 1996.

[4] Y. He, H. Tan, W. Luo et al., "MR-DBSCAN: an efficient parallel density-based clustering algorithm using MapReduce," in *Proceedings of the IEEE International Conference on Parallel and Distributed Systems*, vol. 42, pp. 473–480, IEEE, 2012.

[5] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithms: analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, 2002.

[6] J. A. Hartigan and M. A. Wong, "A k-means clustering algorithm," *Applied Statistics*, vol. 28, no. 1, pp. 100–108, 1979.

[7] Y. Zhu, K. M. Ting, and M. J. Carman, "Density-ratio based clustering for discovering clusters with varying densities," *Pattern Recognition*, vol. 60, pp. 983–997, 2016.

[8] S. Zhou, A. Zhou, and J. Cao, "A data-partitioning-based DBSCAN algorithm," *Journal of Computer Research & Development*, vol. 37, no. 10, pp. 1153–1159, 2000.

[9] S. Zhou, A. Zhou, W. Jin, and Y. Fan, "Fdbscan: a fast dbscan algorithm fdbscan," *Journal of Software*, vol. 15, no. 6, pp. 735–744, 2000.

[10] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the 13th Annual ACM Symposium on Theory of Computing*, pp. 604–613, ACM Press, 1998.

[11] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the 20th Annual Symposium on Computational Geometry (SCG '04)*, vol. 34, pp. 253–262, ACM, June 2004.

[12] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-probe LSH: efficient indexing for high-dimensional similarity search," in *Proceedings of the 33rd International Conference on Very Large Data Bases*, pp. 950–961, VLDB Endowment, September 2007.

[13] F. Shen, C. Shen, W. Liu, and H. T. Shen, "Supervised discrete hashing," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15)*, pp. 37–45, June 2015.

[14] W. W. Wang, Y. F. Chen, J. B. Qian, and H. H. Chen, "LSH-based algorithm for k nearest neighbor search on big data," *Acta Electronica Sinica*, vol. 44, no. 4, pp. 906–912, 2016 (Chinese).

[15] J. Wu, "Balance support vector machines locally using the structural similarity Kernel," in *Proceedings of the Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, vol. 6634, pp. 112–123, Springer, 2011.

[16] F. Shen, X. Zhou, Y. Yang, J. Song, H. T. Shen, and D. Tao, "A fast optimization method for general binary code learning," *IEEE Transactions on Image Processing*, vol. 25, no. 12, pp. 5610–5621, 2016.

[17] W. Jin, A. K. H. Tung, J. Han, and W. Wang, "Ranking outliers using symmetric neighborhood relationship," in *PAKDD 2006: Advances in Knowledge Discovery and Data Mining*, vol. 3918 of *Lecture Notes in Computer Science*, pp. 577–593, Springer, 2006.