Computer Science and Information Systems 17(2):665–687 https://doi.org/10.2298/CSIS200406014X

A K-means algorithm based on characteristics of density applied to network intrusion detection*

Jing Xu[1], Dezhi Han[1], Kuan-Ching Li[2*], and Hai Jiang[3]

[1] Shanghai Maritime University, Shanghai, 201306, China
[2] Providence University, Taichung 43301, Taiwan
[3] Arkansas State University, Jonesboro, Arkansas 72467, USA

Abstract. K-means algorithms are a group of popular unsupervised algorithms widely used for cluster analysis. However, the results of traditional K-means clustering algorithms are greatly affected by the initial clustering center, with unstable accuracy and low speed, which makes the algorithm hard to meet the requirements for Big Data. In this paper, a modernized version of the K-means algorithm based on density to select the initial seed of clustering is proposed. Firstly, Kd-tree is used to divide the hyper-rectangle space, so those points close to each other are grouped into the same sub-tree during data pre-processing, and the generalized information is stored in the tree structure. Besides, an improved Kd-tree nearest neighbor search is used in the K-means algorithm to prune the search space and optimize the operation for speedup. The clustering results show that the clusters are stable and accurate when the numbers of clusters and iterations are constant. Experimental results in the network intrusion detection case show that the improved version of the K-means algorithms performs better in terms of detection rate and false rate.

Keywords: Network security; K-means; Kd-tree; Network intrusion detection.

1. Introduction

In recent years, with the rapid development of new technologies such as cloud platforms, Big Data, and artificial intelligence, Internet has become an indispensable driving force for economic development and social progress [1, 2]. The Internet has

*Corresponding author

changed people's living manners, and now it is gradually covering intelligent transportation, smart cities, and others, which is exceptionally convenient for people's living [3]. However, the exploding number of big data environments of networked information data has put tremendous pressure on data processing systems, and various network security threats have become more severe in anomalies such as viruses, hackers, network attacks, and Internet frauds plague various application users [4]. The way to detect abnormal traffic in the process of using the Internet is significant in preventing further harm [5]. As the name implies, anomaly detection is the detection of abnormal behaviors [6, 7]. Through collecting and analyzing some critical points of the computer networks [8, 9], we find out whether there is any violation of the security policy and sign of being attacked in the network.

In order to prevent network anomaly flow from harming Internet users and causing the loss of personal, company, enterprise, and government information, it is necessary to find a reasonable and effective network anomaly detection method [10]. In this paper, the K-means clustering algorithm is investigated, in which network behaviors are classified, and abnormal traffic is identified through clustering.

Clustering is a type of unsupervised learning [11]. That is, the points with the same property are grouped closer and closer in space. There are several types of clustering algorithms, and can be classified as: (1) Partitioning method: the main idea of the partitioning method is to give the number of initial seeds and the clustering center points, and then solve the problem iteratively according to the distance computation formula to retrieve a final partition, as in K-means, K-medoids [12,13], (2) Hierarchical method: the datasets are solved hierarchically, and the results are obtained by merging and iterating them to meet specific conditions as in HAC [14], CURE [15], (3) Density method: according to the comparison of the density within the range of regional data points against a certain threshold, the classification result can be achieved as in a mean shift clustering algorithm, DPC [16], DBSCAN [17], (4) The grid method: taking grid as the basic clustering unit, the dataset space is divided into a grid for clustering as in STING [18], WAVE-CLUSTER, and (5) Model method: the data matches the constructed model.

In the clustering algorithm of the partitioning method, K-means is a classical algorithm that is widely used in various fields: after analyzing the internal principles and characteristics of the algorithm, Heil, Jannis et al. [19] applied K-means algorithm to soil diffuse reflection spectrum classification; Gulnashin, Fatima et al. [20] studied and compared K-means algorithm in various aspects to classify documents; Means Kimberly et al. [21] applied K-means algorithm to the diagnosis of atrial fibrillation in the emergency department in the medical field. The working principle of a fast and straightforward K-means algorithm is to compute the average value of each cluster with a simple algorithm, and then divide the classes iteratively until all the clusters are covered. At present, K-means algorithm has become one of the crucial methods of machine learning and data mining technology. With the expansion of clustering applications and the rapid growth of different kinds of data, traditional clustering algorithms are difficult to meet people’s needs. Even more, investigations have focused on improving the traditional clustering algorithm to meet new development needs.

Aimed at the selection of the initial clustering center, this paper improves the K-means algorithm and applies the improved algorithm to detect abnormal network traffic. The main contributions of this paper can be summarized as follows:

• A K-means algorithm based on density to select the initial clustering center is proposed. Firstly, the Kd-tree structure is used to store the given dataset. Secondly, the initial clustering centers are selected using the data range information stored by the Kd-tree node structure so that the initial clustering centers are distributed in a region with a large data density, the relative distance between the initial clustering centers is reasonable, and the anti-noise ability is stronger. Third, the kd-tree nearest neighbor search algorithm is improved. The idea of this improved algorithm is used in the K-means algorithm. This advantage is used to prune the search space and optimize the operation for speedup. When the amount of data in the given data set is small, a clustering method using correlation nearest neighbor search is proposed for the situation of excessive pruning. Finally, the cluster partition is iteratively computed to retrieve the results, which show the stability of clustering results and accuracy under the condition where the numbers of clusters and iterations are constant.

• The algorithm proposed in this paper is superior to the traditional K-means algorithm and literature [27, 28] in detection rate and false rate in the network abnormal traffic detection scenario with a high data dimension.

The remainder of this paper is organized as follows. In the related work of Section 2, the Kd-tree data structure, and the principle of using Kd-tree to accelerate the K-means algorithm are introduced. The advantages and disadvantages of the K-means algorithm based on Kd-tree improvement are analyzed. In Section 3, the process of the traditional K-means algorithm is reviewed. An improved K-means algorithm based on density selection of initial seed is proposed. Section 4 shows the experimental details and gives the results and analyses. Finally, the conclusions and future work are provided in Section 5.

2. Related work

K-means algorithm, as a clustering method, has been used to detect anomalies in network environments. However, the K-means algorithm still has some defects. Many researchers start from data pre-processing before using clustering to improve the selection of the initial central point set for better accuracy of the K-means algorithm.

Canopy is an unsupervised pre-clustering algorithm proposed by McCallum [22] in 2000. Canopy algorithm only compares the distance between data in the same region and the central point each time. This algorithm reduces the number of comparisons, which reduces the execution time of clustering and improves the computational efficiency of the algorithm. Zhang et al. used a Canopy algorithm with density parameters as the pre-processing process of K-means [23]. The outputs are used as the cluster number and initial central point of the K-means algorithm, which improved the computational efficiency and accuracy of the algorithm. R-tree [24] and Kd-tree [25] are two improved methods based on indexes. The space points contained in each node of R-tree are included by a minimum enclosing rectangle (MBR). The core idea is to aggregate the points close to each other in the tree structure from bottom-up and cover them with a larger MBR in the upper layer until all space points are covered in the root node. Based on this idea, R-tree is widely used in spatial data indexing and partition optimization. Wei Wu et al. proposed a reverse k-nearest neighbor (RkNN) query using the R-tree structure [26]. The algorithm takes the query object as one of its K-nearest

neighbors and uses the new INCH algorithm to solve the evaluation RkNN query and its variant two-color RkNN query on two-dimensional location data.

Kd-tree is simply a multi-dimensional binary tree. The clustering iteration of the K-means algorithm can be improved according to the idea of the nearest neighbor search with the data initialized by Kd-tree. The pruning process is used to eliminate the distance computations and improve the efficiency of the K-means algorithm. Redmond et al. constructed Kd-tree and used its structural characteristics to obtain the density estimation of leaf buckets and used the max-min method to find the initial cluster seeds [27]. Kumar et al. improved the algorithm [28], from the viewpoint that density weight and distance weight are used to replace density estimation and distance estimation when selecting the initial seeds. Next, we discuss Kd-tree in detail and use the Kd-tree nearest neighbor search to accelerate the K-means algorithm.

2.1. Kd-tree

Kd-tree is a binary tree and a data structure for storing a finite set of points from an m-dimensional space. The points with similar partition-dimension can be stored in the same Kd-space. Given a dataset S, a Kd-tree can be constructed: the binary tree is built in a top-down manner, and each sub-tree is contained in a hyper-rectangle, which is containing the projection of maximum and minimum coordinate values of points in each dimension. Computing the variance in each dimension of the sub-tree to obtain the partition-dimension. The median value of partition-dimension is selected to balance both the left and right sub-trees. Then the hyperplane of the axial pair is used to divide the dataset into two subsets for two hyper-rectangles, which are used as the left and right sub-trees of the parent node. Iteration through the partition process continues until the leaf node or a specified number of leaf bucket nodes are divided. The schematic diagram of Kd-tree in two-dimensional space is shown in Fig. 1.

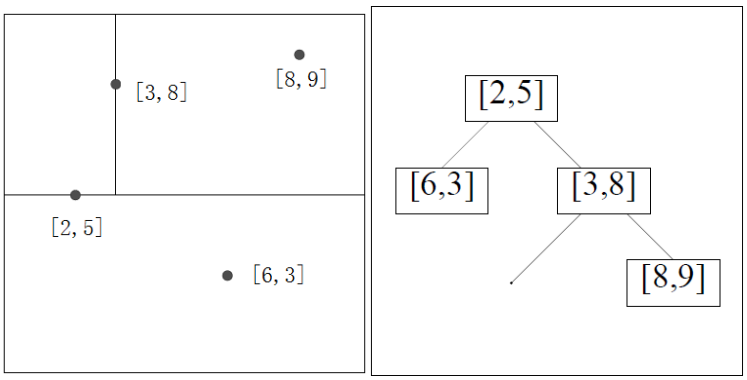

Fig. 1. Construction of Kd-tree with two-dimension data

2.2. Kd-tree nearest neighbor search

K-means algo The reason why Kd-tree can be used to reduce the number of nearest neighbor queries in K-means is that their nodes can represent a large number of points [29]. The node can store the super rectangle information of the dataset, including the Max and Min vectors. The process of finding the nearest neighbor of a node Q: Starting from the root node of dataset S, Kd-tree is accessed from top to the leaf node. The current node is denoted as "nearest neighbor point" and the minimum distance set as dis. A traceback program is used to find any points closer to Q in the branch of the access path. The evaluation is based on the fact that the hypersphere with Q as the center and dis as the radius intersects with the hyper-rectangle of the branch. The current "nearest neighbor" is updated if there is a data point that is closer to Q until you go back to the root node.

2.3. K-means accelerated by Kd-tree

This K-means algorithm with Kd-tree acceleration stores the data set in Kd-tree during data pre-processing, as displayed in Fig. 2. This algorithm starts using randomly generated cluster centers. Nodes in Kd-tree are accessed from the top down during each iteration. According to the spatial region information, the data of a specific point, and its left and right sub-trees are used to determine whether they belong to a specific central point. If it is not possible to do so, traverse the tree into the left and right sub-trees of this point until all data points are included in clustering.

The factors why Kd-tree can accelerate K-means [30] are because Kd-tree's nearest neighbor algorithm can prune Kd-tree during the traversal: firstly, search the minimum distance between each cluster center in the candidate cluster center set and the space area represented by the node object, which is obtained by Kd-tree's nearest neighbor query algorithm. Then, the maximum distance of the space area represented by the node object is denoted as Maxdis, and the maximum distance adopts the maximum value of the space range represented by the node. Finally, the cluster center of the smallest one whose nearest neighbor distance is greater than the maximum distance is cut off. A rectangle can represent the h in the two-dimensional space. The point in Fig. 2 represents the distribution of the subtree dataset of a particular data point, and the rectangle is the data range h.

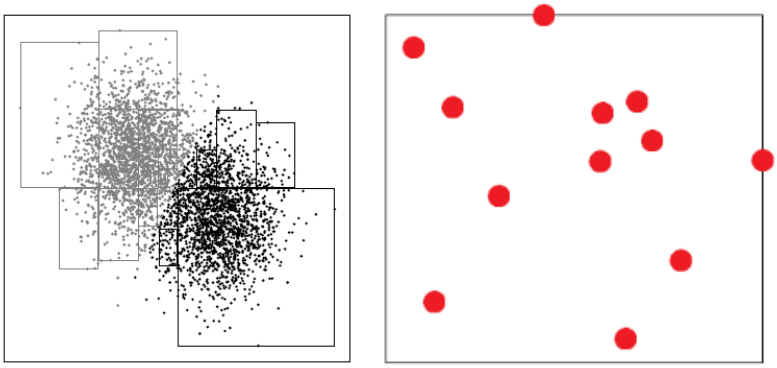

Fig. 2. Two-dimensional dataset rectangle partition

In this paper, the selection of the initial cluster centers is improved. Based on the pre-processing dataset with Kd-tree data structure, a method of correlation weighted distance computation and correlation weighted nearest neighbor search is proposed to improve the selection of clustering centers and iterative clustering computation, respectively. Based on the data structure of Kd-tree, the average density weight of the nodes in the leaf bucket is computed. A point with a maximum density as the first initial center point; then, the remaining k-1 initial central points are determined according to the product of the density weight of the buckets and the correlation weighted distance of the selected initial center. Finally, the improved method of correlation distance weighting based on Kd-tree acceleration is used for the iterative computation.

3. K-means algorithm

3.1. Traditional K-means algorithm

As mentioned above, the principle of the K-means algorithm is to obtain clustering by taking the initial central point as the prototype [31]. This algorithm belongs to the partitioning clustering algorithm. The letter K means the number of k used in the randomly selected initial cluster center. The Euclidean distance computation in the Cartesian coordinate system is used to classify the data into the closest cluster. Each data point belongs to a cluster. Each iteration recomputes the center point of each cluster, and the next iteration produces new clustering results. Repeat this step until the center point no longer changes or converges.

The algorithm process is as follows:

Let the input dataset be $S = \{x_1, x_2, \ldots, x_n\}$, among them $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ where $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ is a data object with p dimensional characteristics.

1) k initial clustering centers are randomly selected from the sample S.

2) The Euclidean distance $d(x_i, c_i)$ between data points x_i and the clustering center c_i is computed. Through partitioning, data points were classified to the nearest clustering center, and k partitioning classes $L = \{l_1, l_2, \ldots, l_k\}$ were obtained.

3) The center c_i of each cluster is computed and the result is used as the new cluster center.

4) Repeat steps 2 and 3 until the maximum number of iterations is reached, or the cluster difference is less than a threshold.

The output are k clusters.

K-means is a classical algorithm to solve clustering problems with the features of simplicity and fast speed. Scalability and efficiency of the algorithm for processing large datasets are maintained. When a cluster is close to the Gaussian distribution, the clustering effect is better.

However, in the K-means algorithm, an initial partition shall be determined according to the initial clustering center. Different initial centers determine the level of clustering accuracy, so that the algorithm is sensitive to the initial values. If the initially selected clustering center has a substantial deviation from the actual clustering center, the clustering effect will be deplorable. The random selection of the center of the initial cluster is not stable. If the cluster contains abnormal points, such as noises and sensitive points of isolated data, the mean deviation will be severe. The Euclidean distances from all sample points to each cluster center need to be computed in each iteration. As the number of iterations increases in the clustering center, the execution speed of the algorithm cannot meet the requirements of Big Data.

3.2. Improved K-means algorithm for selecting the initial center based on density

At this point, a K-means algorithm based on Kd-tree is proposed to select the initial center according to density. To overcome K-means algorithm's unstable clustering results and slow execution, this paper presents a method of selecting the initial clustering centers. Meanwhile, the distance weighting is used in Kd-tree nearest neighbor search and the initial centers at the weighted distance, which avoids the problem of using Euclidean distance to classify samples with similar distances but low correlation into the same category. However, samples with low correlation will cause problems due to the deviation in the same class.

Symbol definitions are shown in Table 1. The algorithm details are as follows:

Table 1. Notations

Symbol	Description
k	Number of cluster center
V	Volume of leaf node
q	Number of leaf buckets

l	Number of leaf nodes in one leaf bucket
c	Cluster center
m	Mean of leaf node
d	Dimensionality of dataset
r	Distance weighting coefficient
ε	Density estimate of leaf node
ρ	Density weight of leaf node
g	Distance estimate of leaf node
β	Correlation distance weight of leaf node

Pearson correlation coefficient [32] is used to weight the Kd-tree nearest neighbor search process and the distance computation used at the initial center point.

$$P = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{x_i-\bar{x}}{s_x}\right)\left(\frac{y_i-\bar{y}}{s_y}\right). \tag{1}$$

Formula definition: The Pearson correlation coefficient P of two continuous variables $(\mathbf{x}, \mathbf{y})$ is equal to the product of their covariance $\mathrm{cov}(x, y)$ divided by their respective standard deviations. The value of the Pearson correlation coefficient is between -1.0 and 1.0. Variables close to 0 are considered not correlated, while those close to 1 or -1 are considered strongly correlated.

The distance weighting coefficient is computed as follows:

$$r = 1.0 - P + 0.001. \tag{2}$$

The value 0.001 is set so that the value of the correlation weighting coefficient is not 0.

Data have been standardized and normalized during data preprocessing.

Initial centers selection method uses correlation weighting as follows:

The Kd-tree structure is iteratively established. When the number of data sets is to a certain number, it is no longer divided. This structure is named leaf bucket. As the data is split into a specified number of child nodes, it is no longer divided and stored as a leaf bucket. As with other nodes, the information on the hyper-rectangle space of the dataset is stored on the parent node. The number of leaf buckets is q, and the number of leaf nodes in the leaf bucket is l. First, the average value of the data points in the leaf bucket is computed. Assume that the average value of data point objects in the leaf bucket stored on node x_i is m_i, that is:

$$m_i = \frac{\sum_{i=1}^{l} x_i}{l}. \tag{3}$$

The Max and min vectors stored in the Kd-tree are used to compute the volume of each leaf bucket:

$$V_i = \prod_{i=1}^{d}\left(x_{i\max} - x_{i\min}\right). \tag{4}$$

The density estimate $\varepsilon_i = N_i / V_i$ is computed; where N_i represents the number of data objects in the leaf bucket; the density weight is computed as:

$$\rho_i = \frac{\varepsilon_{sum}}{\varepsilon_{sum} - \varepsilon_i}. \tag{5}$$

where ε_{sum} is the sum of density estimate with $\varepsilon_{sum} = \sum_{i=1}^{q} \varepsilon_i$. The average value of the leaf bucket with the most considerable density weight is selected as the first center point C_1. Then, the correlation weighted distance between the remaining leaf buckets and the center points C_i are computed. The higher the correlation weighted distance, the more significant the difference between the two points and the less the correlation will be.

$$g_i = \arg\min_{k=1...q}[d(m_i - c_k)\mathrm{r}]. \tag{6}$$

The Eq. (6) represents the minimum value of the correlation weighted distance between each initial central point and the average value of the leaf bucket. By computing $g_{sum} = \sum_{i=1}^{q} g_i$, the distance weights are as follows:

$$\beta_i = \frac{g_{sum}}{g_{sum} - g_i}. \tag{7}$$

$$c_k = \{\mathrm{m}_i \mid z = \arg\max_{i=1...q}(\rho_i \beta_i)\}. \tag{8}$$

The candidate central point is determined while the density of leaf bucket is large and it is far away from the initial central point. The easiest way to do so is to select it as the next initial central point c_{k}. However, there is a case where if a leaf bucket itself has a low density but is far from the existing initial central point, the outlier is easily misselected as the initial center, and such a point is called an abnormal point. When running the algorithm, a scheme to remove noise points before the next selection of the center algorithm is executed.

$$N\mathrm{oise} = \beta_i / \rho_i. \tag{9}$$

Prior to the execution of each algorithm, the leaf bucket with the maximum noise value of 10% is discarded to obtain a new candidate central point set. The algorithm is as follows.

Table 2. Description of selecting the initial center algorithm

Algorithm description
Input: n-dimensional data set $S = \{x_1, x_2, \ldots, x_n\}$;Cluster data quantity k Output: k clustering centers begin Step1:Establish a Kd-tree.

Step2:Compute the density weight ρ_i of each leaf bucket using Eq. (5).
Step3:The average value of leaf bucket with the largest ρ_i value is selected from the candidate center data set as the first initial center C_1.
Step4:for j=2 to q
Compute the distance weight β_i of each leaf bucket in the candidate center dataset, and use Eq. (7) to compute the z value. Select the ith central point according to Eq. (8).
end for
Step5:Eq. (9) is used to compute the noise value $Noise$ of each leaf bucket in the candidate center dataset. Discard the top 10% of noise points to generate a new candidate center dataset. Repeat step 4,until i=k.
Output: $(C_1, C_2, \ldots, C_k)$
end

3.3. Relevance weighted nearest neighbor search

Kd-tree can accelerate the K-means algorithm, which is introduced in Section 2. This section introduces the relevance weighted nearest neighbor search. Based on the nearest neighbor search idea, data points are allocated to the nearest central point, which avoids the redundant computation of distance in the iterative process of the traditional K-means algorithm; this process saves a lot of time while improving the efficiency of the algorithm. In the improved nearest neighbor search, a correlation weighting is also used in the distance computation, and the dataset is classified into the central point with the strong correlation in the steps requiring a distance computation to make the clustering effect more accurate.

The dataset is traversed from the root node to the leaf node. Let this point be the node to calculate the weighted distance between the leaf node and the candidate central point. The node is classified into the cluster with the smallest distance, and then the backtracking operation is performed. The correlation weighted distance of each current node space area information h and the candidate central point is calculated. If the spatial region information can be used to judge that the data in the sub-tree of this node belong to the central point c, the computation of the distance from the data in the sub-tree of this node to the central point c can be reduced. If it can be judged that none of the data in h belongs to a particular central point c, the computation of the distance from the node data in the h to the central point c can be reduced. When it is determined that all the data in this node sub-tree of this node belong to c, the statistical information stored in the h can be directly classified into the central point c.

Determine the node center in the h, including the following steps:

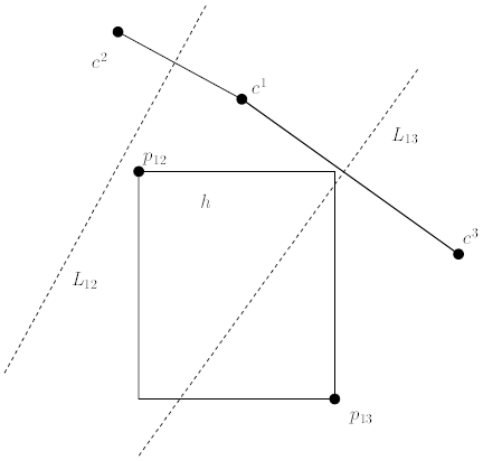

Fig. 3. Domination with respect to a hyper-rectangle

Step 1: determine if there is a minimum distance.

The point with the smallest distance by computing the distance value of $d(ci,h)$ from the point of dataset C of the candidate centers to the h of the current node is found. If there is one minimum distance, this c_i may be the central point of the Kd-tree or its sub-Kd-tree represented by the hyper-rectangle h. If there is more than one minimum distance, the central point of the Kd-tree or its sub-Kd-tree represented by the hyper-rectangle h does not exist and h is divided into left and right subtrees for search. Through the above analysis and computation, some inferior central points can be excluded in the process of subtree traversals.

Step 2: determine whether c_i is the central point.

In step one, if the result of the operation is that there is a minimum distance, then this c_i may be the central point of h, and further judgment is needed. Take Fig. 3 as an example. To determine whether c_1 is superior to c_3:orientation $v=(c_3-c_1)$. If the p is picked as a member of h, the principle is to maximize the inner product of two vectors. As shown in Fig. 3, L_{12} is the decision line between centers c_1 and c_2. L_{13} is the decision line between c_1 and c_3. Vector v's positive and negative dimensions are $(+,-)$. The selection principle of any point p is as large as possible on the X-axis and as small as possible on Y-axis. In this case, p_{12} is the extreme point in h in the direction c_1 c_2, and p_{13} is the extreme point in h in the direction c_3 c_1. The inner product of the two selected vectors is the maximum. Since p_{13} is picked, whose Euclidean distance in a two-dimensional space is closer to c_3, so c_1 is not better than c_3. If c_i is superior to other points, it can be judged that c_i is the central point of Kd-

tree or its sub-tree represented by h. Otherwise, c_i is not the central point of Kd-tree or its children represented by h; Although c_i is not the central point of Kd-tree or its sub-tree represented by h, c_i is better than c_2, which is excluded from the set of candidate points from the central point of the sub-tree of h.

Table 3. Description of the clustering algorithm

Algorithm description

```
Input: Kd-tree of dataset S;Initial cluster center C
improvedKmeans(S,C):
begin
      Start from the S's root node Kd-tree.root.
      if point u is the leaf node
           The correlation weighted distance of the point was
           computed and the point was classified into the
           clustering center with the minimum distance.
           Return.
      else
          for each C compute d(ci,node.h)*r
            if there are multiple minimum d
                  Recursively invoke the improved kmeans
                  algorithm.
                  Call improvedKmeans(node.lchild,C).
                  Call improvedKmeans(node.rchild,C).
                  Return.
            else
                  Set the minimum value to ct.
                  Compute whether there is a center superior
                  to ct in C.
                  if(ct better then ci)
                      Add ci to Cout.
      if there is only one advantage
           This point and its child nodes were classified
           into ct.
      Cnew = C-Cout.
      Recursively call the improved K-means and pass in the
      filtered clustering center.
      Call improvedKmeans(node.lchild,Cnew).

      Call improvedKmeans(node.rchild,Cnew).
      Return.
Output: k cluster centers
end
```

For small datasets, a clustering method using correlation nearest neighbor search is designed, and the algorithm steps are presented in Table 4. In this method, the Kd-tree structure is performed to find the nearest neighbor.

Table 4. Description of the clustering algorithm

Algorithm description

Input: N-dimensional dataset $S=\{x_1, x_2, \ldots, x_n\}$;Cluster data quantity k
begin
for i=1 to len(S)
for j=1 to k
The relevance weighted nearest neighbor is used to search for the nearest point x in S to center[j].Mark it as category j.
end for
Remove x from S.
end for
Output: k cluster centers
end

3.4. Algorithm analysis

Using Kd-tree leaf bucket structure to compute the density poses some problems. Since the density weight and distance weight dimensions are different, the data density calculations of different datasets are different, and the distance varies greatly. Then, the actual effect of the algorithm could be biased. Therefore, it is necessary to do some processing on these two values to eliminate such bias. Firstly, the influence of density weight is that leaf buckets with low density are underestimated. When selecting the initial center, more leaf buckets with high density are preferred. Thus, most leaf buckets with high density are selected as the initial center. The results ignored some smaller clusters in the actual situation. A method is proposed to solve this problem by using the rank value of density estimation instead of the actual density estimation [27].

Suppose there are 1024 leaf buckets. Compute the density weight of the leaf bucket and arrange it in descending order. The node with the highest density weight is assigned a density level of 1024, followed by 1023, 1022, until the lowest level is 1. The method reduces the influence of the leaf bucket with high-density weight on the center selection to a certain extent. But the density rank method does not reflect the actual density of the node, ignoring the variation of the density weight between the leaf buckets. It is assumed that after the sorting, two adjacent leaf buckets with a significant difference in density weights and two other smaller leaf buckets with different density weights are sorted; thus, the span between ranks is 1 and the degree of discrimination tums to be smaller. When the gap between the leaf bucket density values of the data is small, the gap is enlarged. A hierarchical classification strategy with a large density estimation span is proposed [28], reflecting the gap between the estimated values of the leaf bucket density. It can distinguish the density estimation value more effectively, though this method is random. Besides, there is a drawback that the buckets with the value of low-density are neglected. A mapping method that utilizes a function to slow down density growth is performed to increase the weight of the leaf bucket with a lower density value,

and maintain the leaf bucket with a higher density value. The final density level is divided into points that are not integers in the number of leaf buckets.

3.5. Performance analysis

The time consumption of the proposed algorithm mainly includes the establishment of Kd-tree and clustering, which also contains the initial central point and the partition. The time complexity of constructing a modified Kd-tree is $O(dn\log n)$ [28]. The time complexity of density weight computation in the initial cluster center is $O(dq)$, the same as in [27, 28]. The average value of the leaf bucket is the distance weight and its time complexity is $O(dql)$. The time complexity of computing the distance weight is $O(qk)$, so the time of the 2 to k initial clustering center algorithm is $\frac{1}{2}qk^2-\frac{5}{2}qk+3q$. The partition time is the time spent on browsing the tree multiplied by the pruning rate. If the pruning rate is α, the partition time is $O((1\text{-}\alpha)\log n),(0\le\alpha<1)$.

The clustering time complexity of traditional K-means is $O(kdn)$, and the clustering complexity is $T(n)=O((1-\alpha)\log n)$, same as in [27, 28]. Among them, the computation formula of the pruning rate is $\alpha = Amount\ of\ pruned\ data/Total\ data$. The initial center selected in the improved algorithm is of higher accuracy, so increasing the pruning rate reduces the computation of clustering times in the iteration and thus reduces the time complexity of clustering.

4. Experiment and analysis

4.1. Experimental environment

All the experiments are executed as single-threaded on ThinkStation P920 workstation, configured with a 12-core Intel(R) Xeon(R) Silver 4116CPU 2.10 GHz and 32G RAM. All application programs are compiled and executed using Python3.5 in Pycharm environment running on Ubuntu16.04 OS.

4.2. Experimental data set

The datasets selected for the experiment are Iris and Wine, two classic data sets of UCI Repository, and Nsl-kdd datasets. Iris dataset is a common standard data set for cluster

analysis. It has 150 data and 4 features, and the last one is the data label. The features include the length of the sepals and petals of the iris, and the classification of this plant can be determined through the evaluation of the features. The wine dataset is often used for cluster analysis. There are 178 pieces of data and 13 different numeric features. All the experimental data in this investigation have been pre-processed for standardization.

The Kdd cup 99 [33] is the dataset of the Kdd cup network intrusion theme contest in 1999. Nsl-kdd [34] is an enhancement to the KDD Cup 99 dataset that does not contain redundant and duplicate records in its own set of tests, making it suitable for experiments and tests. Nsl-kdd is the same as the KDD Cup 99 dataset, containing 23 attack behaviors and 42 attribute features.

4.3. Experimental results and analysis

Experiment 1

In order to verify the advantages of the initial center selection of the algorithm in this investigation, traditional K-means clustering and Kd-tree accelerated clustering were used respectively after the completion of the center selection.

Principal component analysis (PCA) is a method to simplify datasets and reduce the data dimension [35]. The main idea of PCA is to reallocate the original d-dimensional features to the k-dimension, which is a brand new orthogonal feature, also known as the principal component [36]. PCA sequentially searches for a set of orthogonal coordinate axes in the original linear space, taking maximum variance as the standard, and projecting the transformed data onto a new set of coordinate axes. The choice of new axes is closely related to the data itself.

The results of the initial cluster centers are shown in Fig. 4. The Iris dataset was projected onto the two-dimensional plane using PCA dimensionality reduction. After being standardized, the data of the Wine dataset is projected onto a two-dimensional plane using the PCA method.

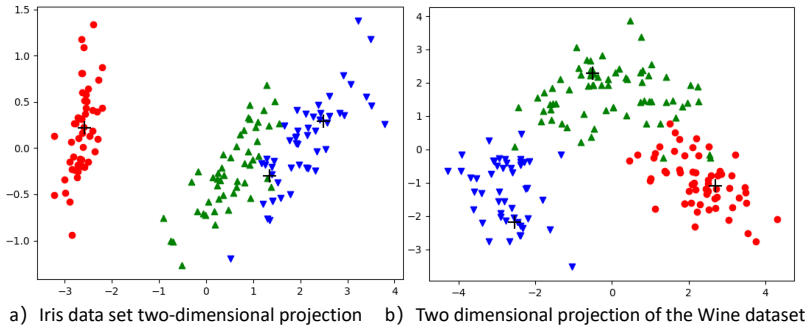

Fig. 4. Data centralized cardiac projection

Three shapes represent different categories of data. The initial central points of clustering are distributed in three different classes.

The Iris and Wine datasets contain a small amount of data. As a result, the traditional Kd-tree accelerated K-means clustering algorithm has no tangible benefit in reducing the execution time, and the high pruning rate will affect the clustering accuracy.

For these two datasets, the algorithm of Table 4 is used. In order to verify the improvement of the initial data center selection method and the correlation weighted nearest neighbor search clustering method, four types of initial cluster center selection methods and three combinations of clustering methods were used. In this experiment, the numbers of nodes 6, 7, and 8 are selected in the leaf buckets. Experimental clustering results in the Iris dataset are presented in Table 5.

Table 5. Iris data set experimental results

The initial center	Clustering method	k	1	2	3	means
K-means	K-means	3	0.593	0.740	0.613	0.649
		4	0.653	0.906	0.827	0.795
		5	0.740	0.760	0.820	0.773
Literature [27]	K-means	3	0.866	0.866	0.866	0.866
		4	0.866	0.866	0.866	0.866
		5	0.900	0.900	0.900	0.900
	Literature [27]	3	0.706	0.706	0.706	0.706
		4	0.740	0.740	0.740	0.740
		5	0.767	0.767	0.767	0.767
Literature [28]	K-means	3	0.887	0.887	0.887	0.887
		4	0.893	0.893	0.893	0.893
		5	0.913	0.913	0.913	0.913
	Literature [28]	3	0.740	0.740	0.740	0.740
		4	0.787	0.787	0.787	0.787
		5	0.806	0.806	0.806	0.806
Proposed	K-means	3	0.900	0.900	0.900	0.900
		4	0.900	0.900	0.900	0.900
		5	0.913	0.913	0.913	0.913
	Proposed	3	0.953	0.953	0.953	0.953
		4	0.940	0.940	0.940	0.940
		5	0.967	0.967	0.967	0.967

The experimental results of the Iris dataset show that the clustering results of the K-means algorithm are unstable. The accuracy of each experiment is uncertain, but the overall accuracy is low. The clustering results of the algorithm proposed in [19, 20] using the traditional K-means algorithm are better than those using the traditional K-means algorithm to select the initial clustering center randomly. However, when using the Kd-tree accelerated K-means algorithm to achieve clustering, there exists a problem that the clustering results are inaccurate due to the high pruning rate.

The method proposed in this search for selecting the initial center has higher accuracy than the traditional K-means algorithm for randomly selecting the clustering center after using the traditional K-means clustering method; though, the clustering effect is better than the algorithm in [27, 28]. The clustering results are also more stable, as the accuracy of the new method of clustering of relevance concerning the nearest neighbor is considerably improved compared to the traditional K-means algorithm. Experiments show that the improved K-means algorithm in this investigation can achieve a better clustering effect. For the same dataset, the same initial clustering center is obtained by setting the same number of leaf buckets l and the same number of

clusters k. Under the condition of constant clustering number and iteration number, the clustering results are stable and accurate.

The Wine dataset contains 13 attributes, and the data with different dimensions among the attributes have considerable differences, which is not favorable to the clustering algorithm. The normalization is used to preprocess the data and transform it into dimensionless scalars to improve the accuracy of the algorithm. Like Iris dataset, four clustering center initialization methods and the combination of three clustering methods are used to verify the accuracy and reliability of the algorithm in this investigation. The numbers of leaf buckets selected in the experiment are 4, 6 and 9. Experimental clustering results in the Wine dataset are presented in Table 6.

Table 6. Experimental results of Wine data set

The initial center	Clustering method	k	1	2	3	means
K-means	K-means	3	0.579	0.557	0.562	0.553
		4	0.512	0.601	0.579	0.564
		5	0.713	0.663	0.612	0.663
Literature [27]	K-means	3	0.657	0.657	0.657	0.657
		4	0.674	0.674	0.674	0.674
		5	0.719	0.719	0.719	0.719
	Literature [27]	3	0.545	0.545	0.545	0.545
		4	0.624	0.624	0.624	0.624
		5	0.679	0.679	0.679	0.679
Literature [28]	K-means	3	0.740	0.740	0.740	0.740
		4	0.767	0.767	0.767	0.767
		5	0.787	0.787	0.787	0.787
	Literature [28]	3	0.567	0.567	0.567	0.567
		4	0.668	0.668	0.668	0.668
		5	0.680	0.680	0.680	0.68
Proposed	K-means	3	0.740	0.740	0.740	0.74
		4	0.781	0.781	0.781	0.781
		5	0.803	0.803	0.803	0.803
	Proposed	3	0.781	0.781	0.781	0.781
		4	0.826	0.826	0.826	0.826
		5	0.871	0.871	0.871	0.871

It can be seen from the experimental results of the Wine dataset in Table 5 that the clustering results of the K-means algorithm are still unstable. The experimental accuracy of the algorithm in this investigation on the Wine dataset is not as high as that of Iris, but the clustering results are relatively more stable than the results obtained by the K-means algorithm. The accuracy of the proposed algorithm in selecting the initial clustering centers is verified.

Experiment 2

In experiment 2, the improved algorithm for network intrusion detection is used to verify the stability and accuracy of the improved algorithm in datasets with large data volumes and high dimensions. The Nsl-kdd dataset contains 125,973 pieces of data with rich characteristic attributes, which provides rich information for research and application, but also increases the workload. The problem can be simplified by analyzing the correlation between variables to reduce the dimension of data. Large

characteristic quantity can reduce the analysis index and minimize the information loss contained in the original index. Taking into consideration that the transformation of closely related variables into fewer new variables, the experimental efficiency is improved.

PCA method is used for dimensionality reduction. As depicted in Fig. 5, different dimension K values in the data and the results obtained by selecting different dimensions are shown that, as K increases, the curve flattens out, and choosing K=20 explains 93% of the variation of the initial variable. Therefore, 20 main components were chosen to be conserved and the characteristics of the Nsl-kdd dataset were reduced to the characteristics of the 20-dimensional vectors.

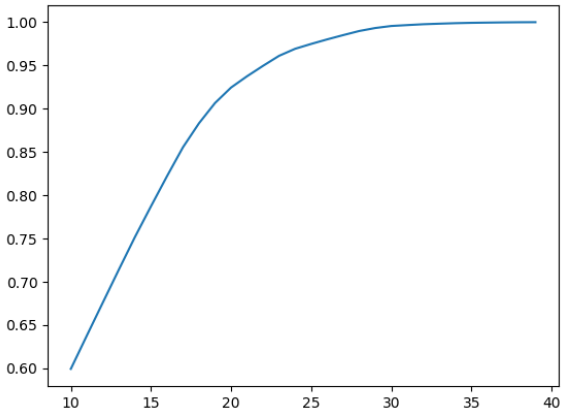

Fig. 5. Data dimension reduction

Experiments were carried out under different clustering numbers k and the number of data l in the leaf bucket. The parameter of the leaf bucket did not exist in the experiment of the traditional K-means algorithm. K-means with the same K value corresponds to several randomized experiments. The results were analyzed, and the detection rate (DR) and false alarm rate (FR) are compared. The detection rate represents the proportion of abnormal records detected correctly in all abnormal records, and the false alarm rate represents the proportion of normal records misdetected as abnormal records in all normal records. In the test of computing running time and computation, several groups of experiments are defined with different K values, in which the iteration times of each algorithm are 10. The experimental results are presented in Table 7.

Table 7. Intrusion detection experiment accuracy results

k	l	d	DR				FR			
			K-means	[27]	[28]	Proposed	K-means	[27]	[28]	Proposed
40	24	20	0.4781	0.8129	0.8545	0.8688	0.1161	0.0619	0.0640	0.0594
50	24	20	0.5173	0.8254	0.8178	0.8713	0.1323	0.0685	0.0788	0.0625
60	24	20	0.5268	0.8052	0.8313	0.8775	0.1355	0.0642	0.0628	0.0633
40	25	20	0.5091	0.8471	0.8664	0.8981	0.1432	0.0721	0.0699	0.0705
50	25	20	0.5314	0.8336	0.8683	0.8913	0.1574	0.0874	0.0767	0.0697

60	25	20	0.5581	0.8594	0.8922	0.9021	0.1711	0.0925	0.0794	0.0720
40	11	20	0.4237	0.7923	0.8339	0.8885	0.1324	0.0674	0.0626	0.0631
50	11	20	0.6140	0.8316	0.8241	0.8091	0.2156	0.0748	0.0683	0.0522
60	11	20	0.5947	0.8191	0.8611	0.8758	0.1978	0.0884	0.0899	0.0823

Table 8. Results of computational efficiency of intrusion detection experiment

k	d	Runtime(s)				Compute distance			
		K-means	[27]	[28]	Proposed	K-means	[27]	[28]	Proposed
40	20	360	189	116	113.	50389200	27646772	23874631	23302891
45	20	407	212	146	139	56687850	29297394	26781201	24547793
50	20	439	228	152	150	62986500	36538875	30100369	28947137
55	20	472	239	163	164	69285150	39909562	33674610	31897414
60	20	496	251	173	177	75583800	40242634	34816779	33471059

Experimental results show that the DR of the algorithm in this investigation is improved, and the FR is low compared to the traditional k-means algorithm and references [19, 20] under different clustering numbers. The algorithm model has a better intrusion detection effect of abnormal traffic.

As presented in Table 8, the center of the cluster converges and the pruning rate of the Kd-tree search space tends to be stable. Experimental results of the Nsl-kdd dataset show that: with the increase of the number of iterations, the algorithm proposed in this study is superior to the traditional K-means algorithm in terms of efficiency and the distance computation in each iteration process is lower than the traditional k-means algorithm. The accuracy of selecting the initial center is improved, and the distance computation in the iteration process can also be reduced. The reliability of the algorithm in this paper is proven when applied in the field of network intrusion detection.

Experment 3

This experiment intends to verify the advantages of the algorithm in the case of a large volume of high dimensional data. NumPy library in Python is used to generate 20 uniformly distributed random cluster central sets C, where the dimension is 20. Make_blobs from the Sklearn library is then performed to generate random data for the cluster model. 20,000 ~ 300,000 samples are generated each time, with 20 features for each sample and a total of 20 clusters, as presented in Table 9. The test made a comparison between clustering time and tree building time.

Table 9 Random data sets

Data sets	Leaf buckets	Clustering number	Data set size /103	Dimension
S1	18	20	20	20
S2	13	20	30	20

S3	18	20	40	20
S4	12	20	55	20
S5	17	20	80	20
S6	10	20	100	20
S7	16	20	150	20
S8	10	20	200	20
S9	14	20	250	20
S10	17	20	300	20

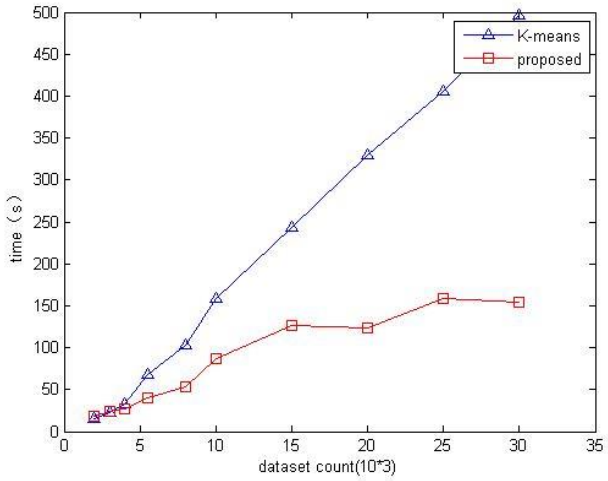

Fig. 6. Experimental results

As displayed in Fig. 6, the K-means algorithm shows that the use of Kd-tree acceleration for datasets with a small amount of data is not significant. There may happen that the efficiency is even lower than the traditional K-means algorithm. The main reason is that there is extra computation when building Kd-tree, and the access overhead also brings time consumption. As the amount of data increases, the effectiveness of the algorithm in this paper is more prominent.

5. Conclusions and Future Work

K-means algorithm is one of the most classic data mining algorithms. Based on the K-means algorithm of unsupervised learning method, aiming at solving the problems of unstable cluster effect caused by the random selection of initial clustering central points and by a low operation efficiency when the data volume is large, a k-means algorithm of selecting clustering center based on density is proposed.

In the improved algorithm, Kd-tree is used as a data pre-processing method to store data, and leaf buckets are performed to organize leaf nodes when building Kd-tree. The initial cluster center was selected by computing the density estimation and density weight of leaf buckets, as well as the correlation weighted distance estimation and distance weight between leaf buckets. In the process of clustering, the relevance

weighted nearest neighbor search is used to prune the search space, to improve the accuracy of clustering and the overall rate of the algorithm operation, and to guarantee the reliability of it. Meanwhile, high efficiency and accuracy are more suitable for detecting abnormal network traffic with large amounts of data under complex and changeable conditions.

From the experiment 1, the results show that the proposed algorithm has higher accuracy. The network intrusion detection results in experiment 2 show that the proposed algorithm has higher accuracy and a lower false alarm rate, where the initial clustering center is more accurate and has a stronger anti-noise ability. Results of experiment 3 show that the proposed algorithm is faster and more user-friendly to datasets containing a large amount of data, and can adapt to the efficiency requirements of computing in the era of Big Data.

As future research, the density-based K-means algorithm can be improved through parallelization, so thus, the speed of execution can be further improved in large datasets. An example of such a parallel environment is Multi-GPU parallel systems [28, 29], where thousands of different types of cores are available in a single computing environment, and the exploitation of the parallelism of the K-means algorithm is crucial for high performance.

References

1. Jiang, H., Chen, Y., Qiao, Z., Li, K. C., Ro, W. W., & Gaudiot, J. L. (2014). Accelerating MapReduce framework on multi-GPU systems. Cluster Computing, 17(2), 293-301.
2. Jiang, H., Chen, Y., Qiao, Z., Weng, T. H., & Li, K. C. (2015). Scaling up MapReduce-based Big Data Processing on Multi-GPU systems. Cluster Computing, 18(1), 369-383.
3. Cui, M., Han, D., & Wang, J. (2019). An Efficient and Safe Road Condition Monitoring Authentication Scheme Based on Fog Computing. IEEE Internet of Things Journal, 6(5), 9076-9084. https://doi.org/10.1109/JIOT.2019.2927497
4. Lin, C. H., Xiao-Dong, L. I., Jin, J., Xue-Biao, Y., & Jun, W. U. (2013). DNS traffic anomaly detection based on W-Kmeans algorithm. Computer Engineering & Design.
5. Zhang, W., Han, D., Li, K.-C., & Massetto, F. I. (2020). Wireless sensor network intrusion detection system based on MK-ELM. Soft Computing(2).
6. Liang, W., Fan, Y., Li, K.-C., Zhang, D., & Gaudiot, J.-L. (2020). Secure Data Storage and Recovery in Industrial Blockchain Network Environments. IEEE Transactions on Industrial Informatics, PP(99), 1-1.
7. Liang, W., Li, K.-C., Long, J., Kui, X., & Zomaya, A. Y. (2019). An Industrial Network Intrusion Detection Algorithm based on Multi-Feature Data Clustering Optimization Model. IEEE Transactions on Industrial Informatics, PP(99), 1-1.
8. Han, D., Pan, N., & Li, K. (2020). A Traceable and Revocable Ciphertext-policy Attribute-based Encryption Scheme Based on Privacy Protection. IEEE Transactions on Dependable and Secure Computing, 1-1. https://doi.org/10.1109/TDSC.2020.2977646
9. Han, D., Yu, Y., Li, K.-C., & Mello, R. F. d. (2020). Enhancing the Sensor Node Localization Algorithm Based on Improved DV-Hop and DE Algorithms in Wireless Sensor Networks. Sensors, 20(2), 343.
10. Li, L., Chen, X., Jiang, H., Li, Z., & Li, K. (2016, 30 May-1 June 2016). P-CP-ABE: Parallelizing Ciphertext-Policy Attribute-Based Encryption for clouds.The 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD).

11. Wei, M., Su, J., Jin, J., & Wang, L. (2014). Research on Intrusion Detection System Based on BP Neural Network ((pp. 657-663). https://doi.org/10.1007/978-3-642-40618-8_85
12. Huang, Z. Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. Data Mining & Knowledge Discovery, 2(3), 283-304.
13. Jiang, H., Chen, Y., Qiao, Z., Weng, T.-H., & Li, K.-C. Scaling up MapReduce-based Big Data Processing on Multi-GPU systems. Cluster Computing, 18(1), 369-383.
14. Aliahmadipour, L., & Eslami, E. (2016). GHFHC: Generalized Hesitant Fuzzy Hierarchical Clustering Algorithm. International Journal of Intelligent Systems, 31(9), n/a-n/a.
15. Yanfeng, Peng, Di, & Jian. Research of Performance of Distributed Platforms Based on Clustering Algorithm.
16. Rodriguez, A., & Laio, A. Clustering by fast search and find of density peaks. Science, 344(6191), 1492-1496.
17. Viswanath, P., & Pinkesh, R. (2006, 20-24 Aug. 2006). l-DBSCAN: A Fast Hybrid Density Based Clustering Method. 18th International Conference on Pattern Recognition (ICPR'06).
18. Wang, W. (1997). STING: A Statistical Information Grid Approach to Spatial Data Mining. Proc. of the 23rd Very Large Database Conf., 1997.
19. Heil, J., Haring, V., Marschner, B., & Stumpe, B. (2019). Advantages of fuzzy k-means over k-means clustering in the classification of diffuse reflectance soil spectra: A case study with West African soils [Article]. Geoderma, 337, 11-21. https://doi.org/10.1016/j.geoderma.2018.09.004
20. Gulnashin, F., Sharma, I., & Sharma, H. A New Deterministic Method of Initializing Spherical K-means for Document Clustering.
21. Kimberly, M., Amanda, G., & Tammy, N. Intravenous Continuous Infusion vs. Oral Immediate-release Diltiazem for Acute Heart Rate Control. Western Journal of Emergency Medicine, 19(2), 417-422.
22. Mccallum, A., Nigam, K., & Ungar, L. H. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining.
23. Zhang, G., Zhang, C., & Zhang, H. Improved K-means Algorithm Based on Density Canopy. Knowledge-Based Systems, S0950705118300479.
24. Guttman, A. (1984). A Dynamic Index Structure for Spatial Searching. Acm Sigmod Record, 14(2), 47-57.
25. Bentley, J. (1975). Multidimensional Binary Search Trees Used for Associative Searching. Communications of the ACM, Septembar, 1975. vol. 18: pp. 509-517: ill. includes bibliography., 18.
26. Wu, W., Yang, F., Chan, C. Y., & Tan, K. L. (2008). Continuous Reverse k-Nearest-Neighbor Monitoring. (Ed.),^(Eds.). The 9th International Conference on Mobile Data Management, 2008 (MDM '08).
27. Redmond, S. J., & Heneghan, C. A method for initializing the K-means clustering algorithm using kd-trees. Pattern Recognition Letters, 28(8), 965-973.
28. Kumar, K. M., & Reddy, A. R. M. (2017). An Efficient k -Means Clustering Filtering Algorithm Using Density Based Initial Cluster Centers. Information Sciences, 418.
29. Bris, R. L., & Paul, F. An automatic method to create flow lines for determination of glacier length: A pilot study with Alaskan glaciers. Computers & Geosciences, 52, 234-245.
30. Pelleg, D., & Moore, A. (1999). Accelerating Exact k-means Algorithms with Geometric Reasoning. https://doi.org/10.1145/312129.312248
31. Hartigan, J. A., & Wong, M. A. (1979). A K-means Clustering Algorithm: Algorithm AS 136 ((Vol. 28, pp. 100-108). https://doi.org/10.2307/2346830
32. Wang, X., & Bai, Y. (2016). A Modified MinMax -Means Algorithm Based on PSO. 2016(7), 4606384.
33. Pfahringer, B. (2000). Winning the KDD99 Classification Cup: Bagged Boosting. 1(2), 65-66.

34. Deshmukh, D. H., Ghorpade, T., & Padiya, P. (2015). Improving classification using preprocessing and machine learning algorithms on NSL-KDD dataset. (Ed.),^(Eds.).
35. Roweis, S. (1999). Em algorithms for pca and spca. Advances in Neural Information Processing Systems, 10.
36. Tian, Q., Han, D., Li, K.-C., Liu, X., Duan, L., & Castiglione, A. (2020). An intrusion detection approach based on improved deep belief network. Applied Intelligence. https://doi.org/10.1007/s10489-020-01694-4

Jing Xu is pursuing a master's degree in software engineering at the School of Information Engineering, Shanghai Maritime University, China. Her research interests include Big Data storage and intelligent decision-making, key technologies for DoS attack defense, and wireless sensor network security.

Dezhi Han received the PhD degree from the Huazhong University of Science and Technology. He is currently a professor of computer science and engineering with Shanghai Maritime University, China, and his research interests include network security, cloud computing, mobile networking, wireless communication, and cloud security.

Kuan-Ching Li is a Distinguished Professor in the Dept of Computer Science and Information Engineering (CSIE) at Providence University, Taiwan, where he also serves as the Director of the High-Performance Computing and Networking Center. He is a recipient of awards and funding support from several agencies and industrial companies, as he also received distinguished chair professorships from universities in several countries. He published more than 300 scientific papers and articles and is co-author or co-editor of more than 25 books published by leading publishers. His research interests include parallel and distributed computing, Big Data, and emerging technologies. Professor Li is a Fellow of the IET and a senior member of the IEEE.

Hai Jiang is a Professor in the Department of Computer Science at Arkansas State University, USA. His current research interests include Parallel & Distributed Systems, Cloud Computing, Big Data, and Cryptography. He has published more than 150 research papers in major international journals and conference proceedings. He has served as a U.S. National Science Foundation proposal review panelist and a U.S. DoE (Department of Energy) Smart Grid Investment Grant (SGIG) reviewer multiple times. He is a professional member of ACM and IEEE computer society, also a representative of U.S. NSF XSEDE (Extreme Science and Engineering Discovery Environment) Campus Champion for Arkansas State University.

Received: April 06, 2020; Accepted: May 18, 2020