10.1109/ACCESS.2018.2810267, IEEE Access
IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Clustering Approach Based on Mini Batch Kmeans for Intrusion Detection System over Big Data

Kai Peng[1, 2], Member, IEEE, Victor C.M. Leung[3], Fellow, IEEE, Qingjia Huang[4]

[1]College of Engineering, Huaqiao University, Quanzhou, 362021, P.R. China
[2] Fujian Provincial Academic Engineering Research Centre in Industrial Intellectual Techniques and Systems, Quanzhou, 362021, P.R. China
[3]Department of Electrical and Computer Engineering, the University of British Columbia, Vancouver, V6T 1Z4, Canada
[4]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, 100093, P.R. China

Corresponding author: Kai Peng (**e-mail**: pkbupt@gmail.com).

This work is supported by Quanzhou Science and Technology Project (No. 2015Z115), the Scientific Research Foundation of Huaqiao University (No.14BS316). College Students Innovation and Entrepreneurship Project (201710385023), the Education Scientific Research Project for Middle-age and Young Teachers of Fujian Province (JZ160084). China Scholarship Council awards to Kai Peng for one year's research abroad at The University of British Columbia, Vancouver, Canada.

ABSTRACT Intrusion Detection System (IDS) provides an important basis for the network defense. Due to the development of the cloud computing and social network, massive amounts of data are generated, which inevitably brings much pressure to IDS. And therefore, it becomes crucial to efficiently divide the data into different classes over big data according to data features. Moreover, we can further determine whether one is normal behavior or not based on the classes information. Although the clustering approach based on Kmeans for IDS has been well studied, unfortunately directly using it in big data environment may suffer from inappropriateness. On the one hand, the efficiency of data clustering needs to be improved. On the other hand, differ from the classification, there is no unified evaluation indicator for clustering issue, and thus, it is necessary to study which indicator is more suitable for evaluating the clustering results of IDS. In this study, we propose a clustering method for IDS based on Mini Batch Kmeans combined with Principal Component Analysis. Firstly, a preprocessing method is proposed to digitize the strings and then the dataset is normalized so as to improve the clustering efficiency. Secondly, the Principal Component Analysis method is used to reduce the dimension of the processed dataset aiming to further improve the clustering efficiency, and then Mini Batch Kmeans method is used for data clustering. More specifically, we use Kmeans++ to initialize the centers of cluster in order to avoid the algorithm getting into the local optimum, in addition, we choose the Calsski Harabasz indicator so that the clustering result is more easily determined. Compared with the other methods, the experimental results and the time complexity analysis show that our proposed method is effective and efficient. Above all, our proposed clustering method can be used for IDS over big data environment.

INDEX TERMS IDS, Big Data, Clustering, Principal Component Analysis, Mini Batch Kmeans.

I. INTRODUCTION

Intrusion Detection System (IDS) can discover the malicious activities and irregularities in the network and provide an important basis for network defense [1]-[3]. Due to the development of the cloud computing, social network, as well as mobile cloud computing, IDS has become even more important than before [4]-[6]. Meanwhile, network traffic generated by various devices and hosts are exponentially rising [7]-[12]. In 2013, a total of 1000 EB traffic was generated globally [13]. It is estimated that worldwide network traffic will reach 8.6 ZB by 2018 according to Cisco [14]. This inevitably brings much burden to IDS. And thus, how to efficiently divide the data into different classes over big data in term of data features becomes much more and more important. Additionally, we can further determine which one is normal behavior or not based on the above

10.1109/ACCESS.2018.2810267, IEEE Access
IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

classes information. Data mining is an intelligent data analysis technique that finds useful knowledge from big data [15]-[16]. Data mining method for the first time was introduced to intrusion detection in 1998 [17], and then many researchers engaged in this issue. In general, research of IDS based on data mining is mainly focused on two aspects, thus clustering and classification. More specifically, there is no label for each data record in the initial dataset for clustering issue, and the object of the clustering algorithm is to put similar data records in the same class. That means the behavior of the packet will be marked as normal class or abnormal one according to the characteristics of the data, so as to achieve the purpose of clustering data. Differ from the former one, classification is the process of mining from a dataset which has been clustered. That means each data record has a label in the dataset. i.e., for a given new data record, the purpose of the classification algorithm is to determine which class the given data record belongs to. Both the two issues are different but are indispensable parts of IDS. In this study, we mainly concentrate on the research of the former one. Overall, the main clustering algorithms can be divided into the following categories [18], thus demarcation methods [19]-[23], hierarchical methods [24]-[25], density-based methods [26]-[28], as well as grid-based methods [29]-[30]. Moreover, we mainly focus on the application of Kmeans over IDS.

The literature [20] proposes a cascaded algorithm over Kmeans and C4.5 algorithms for anomaly detection in computer network. Since the Kmeans clustering results are more affected by the initial of clustering centers. The literature [21] proposes using improved IDS Kmeans algorithm to solve this issue. Unfortunately, they fail to consider the efficiency of clustering. The literature [22] proposes using the Apache Spark for IDS in big data environment. Although they consider the efficiency of clustering, unfortunately we cannot effectively determine which K-value is the best from the experimental results presented by them. And therefore, it is also crucial to choose the appropriate evaluation indicator. The literature [23] proposes using Kmeans clustering algorithm over Oracle Database 10 g and 1000 clusters are made. To the best of our knowledge, a good clustering algorithm does not mean that the more clusters, the better. In addition, the experiment in [23] is only presented over 10% KDDCUP99 dataset [31]. And thus, we cannot determine the performance of the algorithm over full dataset. Above all, both the initial of clustering K-value, the efficiency of clustering, as well as the evaluation of clustering results are important for IDS clustering method.

In order to address the above issue, based on Mini Batch Kmeans clustering algorithm [32], we propose an IDS clustering method named PMBKM (Mini Batch Kmeans with Principal Component Analysis) over Anaconda [33]. Firstly, a preprocessing method is proposed to digitize the strings and then the dataset is normalized for the purpose of improving the clustering efficiency. Secondly, Principal Component Analysis (PCA) [34] method is used to reduce dimensionality on the processed dataset in order to further improve the clustering efficiency for the following step. Thirdly, Mini Batch Kmeans is used for the data clustering. More specifically, we use Kmeans++ [35] to initialize the centers of cluster in order to avoid the algorithm getting into the local optimum, in addition, we choose the Calsski Harabasz (CH) indicator [36] so that the clustering results can be more easily determined. In addition, compared with Kmeans method (KM), Kmeans with PCA method (PKM), and Mini Batch Kmeans with PCA method (MBKM), as well as the literature [22] and [23], the experimental results and the time complexity analysis show that our proposed method is effective and efficient.

Our contributions in this study can be summarized as follows. Firstly, we propose using Mini Batch Kmeans with PCA (PMBKM) for IDS clustering, the experimental result shows that the clustering efficiency has been greatly improved. Secondly, we use Kmeans++ to initialize the cluster centers so as to effectively avoid the algorithm getting into local optimum and further ensure the effectiveness of clustering results. Last but not the least, we consider CH indicator for the evaluation of clustering results. The experimental results show that it is easy to determine which K-value is the best one. Last but not the least, not only the 10% dataset, but also the full dataset of KDDCUP99 are tested in this study. Above all, our proposed PMBKM can be used for IDS clustering over big data.

The rest of the paper is organized as follows. In Section II, the preliminaries are introduced. The main content of Mini Batch Kmeans method, Principal Component Analysis method, as well as the evaluation indicator is presented. Section III specifies Mini Batch Kmeans with Principal Component Analysis (PMBKM) for IDS. Each step in this system and relevant algorithms are described. The experiment and discussion are described in Section IV. Section V presents the related work. Finally, we conclude our work and describe the future work in Section VI.

II. Preliminaries

In this section, we mainly introduce the relevant theory. The Mini Batch Kmeans method is firstly introduced in section A, followed by the introduction of Principal Component Analysis in section B. Finally, the evaluation indicator is introduced in section C.

A. Mini Batch Kmeans method

For given dataset $T=\{x_1,x_2,...,x_p\}$, $x_i \in \mathbb{R}^{m*n}$

x_i represents a network record which is an n-dimensional real vector. m indicates the number of records contained in the dataset T . The object of clustering problem is to find the set C of cluster centers $c \in R^{m*n}$ so as to minimize over the dataset T of records $c \in R^{m*n}$ the following function.

$$\min \sum_{x \in T} \left\| f(C,x) - x \right\|^2 \qquad (1)$$

10.1109/ACCESS.2018.2810267, IEEE Access

Where, $f(C,x)$ returns the closest cluster center $c \in C$ to record x. $|C|=K$ and K is the number of clusters we want to find. To initialize the cluster centers, K records are randomly selected by using Kmeans++ [35], and the cluster centers C are set to be equal to the values of these ones. When the amount of data become very big, the convergence rate of original Kmeans will be dropped significantly. An improved Kmeans method named Mini Batch Kmeans [32] is proposed. Differ from Kmeans [37], this one does not use all the data records in the dataset each time, but select a subset of records randomly from the dataset, and therefore greatly reduces the clustering time, and overall reduces the convergence time.

B. Principal Component Analysis Method

When dataset has multiple features, while many of these features are related, and thus we can speed up the learning of clustering algorithms by downscaling the data. In this study, we use Principal Component Analysis (PCA) ([34], [38]-[39]) for the dimension reduction. The main idea of the PCA is to find a direction vector and project the original dataset onto it. The object of PCA is to minimize the mean square error of the projection. Besides the dimensionality reduction of the given dataset, another advantage is no parameter limitation of PCA. That means there is no need to set parameters manual during the calculation of PCA and the final result is data-related and user-independent. For given dataset T, the object of PCA is reducing n features to d dimensions. The corresponding formulas can be expressed as follows.

1) For each data x in T_{m*n}, calculate the mean of each column in T_{m*n}. The first step is mean centering, thus subtracting the column mean from each record and the processed data $TAdjust_{(m*n)}$ is obtained.

2) Calculate the covariance matrix of $TAdjust_{(m*n)}$ and denote as C.

Let

$$c_{ij} = Cov(x_i, x_j) = E\{[x_i - E(x_i)][x_j - E(x_j)]\} \quad (2)$$

Where $i, j = 1, 2, ..., n$

And then the covariance matrix can be expressed as follows.

$$C_{n*n} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix} \quad (3)$$

3) Calculate the eigenvalues and eigenvectors of the covariance matrix C.

4) The eigenvalues are sorted in descending order, the largest d of them are selected, and then d eigenvectors corresponding to the eigenvalues are used as column vectors to form an eigenvector matrix $EigenVectors_{n*d}$

5) Project the sample point onto the selected eigenvector and the $Finaldata_{m*d}$ is obtained.

$$FinalData_{m*d} = TAdjust_{m*n} * EigenVectors_{n*d} \quad (4)$$

C. Evaluation Indicator

It is essential to evaluate the clustering results. Both the clustering time and Calsski Harabasz are used in this study.

1) Clustering time t

The whole calculation time t of clustering algorithm. The shorter the better.

2) Calsski Harabasz (CH)

Differ from the classification in supervised learning, we cannot directly determine which result is right or not as there is no labels in the given dataset by using precision or recall. However, we can evaluate the clustering results based on the degree of in-cluster density and the degree of inter-cluster discretization. Both the Silhouette Coefficient and CH indicator [36] are the common indicators. Compared with the former one, the calculation process of the latter one is simple and has a low overhead. And therefore we choose this one as the clustering evaluation indicator. In addition, the higher CH score is, the better the clustering result will be. In other words, the covariance of the internal data of the category is as small as possible, and the covariance between the categories is as big as possible. CH can be obtained by formula (5) .

$$CH = \frac{SS_{Between}}{SS_{Within}} \times \frac{m-K}{K-1} \quad (5)$$

Where K is the number of clusters, and m is the total number of network records, SS_{Within} represents the overall within-cluster variance and $SS_{Between}$ represents the overall between-cluster variance.

III. Mini Batch Kmeans with Principal Component Analysis (PMBKM) for IDS

A. The overview of Mini Batch Kmeans with Principal Component Analysis (PMBKM)

In this section, IDS Clustering method is presented. The main steps of PMBKM are shown as follows. The PMBKM mainly consists of two steps, thus step1: Data preprocess, step 2: Mini Batch Kmeans with Principal Component Analysis for Intrusion Detection System.

B. Each step of PMBKM

Step1: Data preprocess

This step mainly contains the process of string and normalization. In this study, Euclidean distance is used in Mini Batch Kmeans and thus we should make sure that the data composed of numbers. If the strings are found in the dataset, we should digitize the string by using replace function. In addition, note that the range of numbers in the given dataset may not uniform. That means the columns with large numerical value will cause the role of columns with relatively small numerical value to be ignored, however those columns may play a very important role in

10.1109/ACCESS.2018.2810267, IEEE Access
IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

fact. And thus the normalization process should be performed before executing the clustering algorithm, and the object of normalization is make the characteristic data shrink in ranges of [0-1]. The **algorithm 1** firstly traverses the given dataset T and finds all the strings S in dataset T and obtain the corresponding columns c by using find function (Line 1 to Line 3), secondly, the replace function is called to replace S with random number m (Line 6 to Line 7). And then the processed dataset T' is obtained. In addition, the normalization result T_1 is obtained by using normalization function (Line 9). Obviously, T_1 will be the input for step 2.

Algorithm 1. Data Preprocess Method

Input: Dataset T
Output: processed Dataset T_1
1. **for** $n \leftarrow 1$ to N **do**
2. **for** $m \leftarrow 1$ to N **do**
3. (S, c) $\leftarrow$find (T)
4. **end for**
5. **end for**
6. **for** $n \leftarrow 1$ to c **do**
7. T' (S) $\leftarrow$replace (m)
8. **end**
9. $T_1 \leftarrow$normalization (T')
10. **return** T_1

Step 2: Mini Batch Kmeans with Principal Component Analysis for Intrusion Detection System

Algorithm 2 illustrates the process of PMBKM. The dimension reduction data T_1' is obtained by calling PCA function (**Algorithm 3**) (Line 1). And then PMBKM obtains cluster centers k by calling Kmeansplusplus function (**Algorithm 4**) (Line 2). And then it randomly selects batch sizes and denoted as b and then form a collection of M (Line 7). In addition, it calculates the distance between each point in M and K centers and then assigns each record to the closest center (Line 8 to Line 10). And then, the center of each cluster is it recalculated. The per-center learning rates is used for fast convergence, in the manners of [32] and [40] (Line 12 to Line 15). Finally, K clusters are formed and CH is returned by calling score function (Line 19).

Algorithm 2. (PMBKM)

Input: Given dataset T_1, K (number of clusters),
Given mini batch size b, iterations t
Maximumiter (The maximum number of iterations t)
Output: *CH*
1. $T_1' \leftarrow$PCA (T_1)
2. C $\leftarrow$Kmeansplusplus ( )
3. // Initialize each $c \in R^{m*n}$ by calling Kmeansplusplus procedure
4. v $\leftarrow$0
5. t $\leftarrow$0
6. **repeat**
7. M $\leftarrow$b instances selected randomly from T_1
8. **for** x $\in$ M **do**
9. d[x] $\leftarrow$f (C, x) //Cache the center closest to x
10. **end for**
11. **for** x $\in$ M **do**
12. c $\leftarrow$d[x] // Get cached center for this x
13. v[c] $\leftarrow$v[c] $+1$ // Per-center counts are updated
14. $\eta \leftarrow \frac{1}{\text{v[c]}}$ // Per-center learning rate is obtained
15. $c \leftarrow (1-\eta)c + \eta x$ // Take gradient steps
16. **end for**
17. **t++**
18. **until** t <= *MaxIters*
19. *CH* $\leftarrow$score(T_1)
20. **return** *CH*

Algorithm 3 is implemented according to the formula (2) to (4) in **section** II.C. The PCA method mainly contain four steps. Firstly, the processed data TAdjust is obtained by calling Adjust function (Line 2). And then the covariance matrix C is obtained by using covariancematrix function (Line 3). In addition, eigenvector matrix EigenVectors is obtain by calling EigenVectors function. Finally the Dimension reduction data FinalData is obtained and returned (Line 4 to Line 5).

Algorithm 3. PCA

Input: T, Number of principal components d
Output: *FinalData*
1. **procedure** PCA(T,d)
2. T*Adjust* $\leftarrow$Adjust(T)
3. C $\leftarrow$ *covariancematrix*(T*Adjust*)
4. *EigenVectors* $\leftarrow$EigenVectors(C, d)
5. *FinalData* $\leftarrow$ T*Adjust* * *EigenVectors*
6. **return** *FinalData*
7. **end procedure**

The Kmeansplusplus procedure of **Algorithm 4** firstly chooses a record randomly from the input dataset which names K_1 as the first cluster center (Line 2). Secondly, it computes the distance between the records x_i in T_1 and the selected cluster center K_1 by calling Euclidean_dis function, a new record K_2 will be selected as the second cluster center. Data records with larger distance values are selected with a higher probability. We use Probability function to implement

10.1109/ACCESS.2018.2810267, IEEE Access

this. Repeat this operation until the K centers are selected (Line 5 to Line 10).

Algorithm 4. Kmeansplusplus

Input: Given dataset T_1, K (number of clusters),

Output: K_j //cluster centers

1. **procedure** Kmeansplusplus (T_1, K)
2. $K_1 \leftarrow$random(T_1)
3. $j \leftarrow 2$
4. **repeat**
5. **for** all $i \in T_1$ **do**
6. D (i, K_1) $\leftarrow$Euclidean_dis ( )
7. $K_j \leftarrow$ Probability (Distance_max)
8. $j++$
9. **end for**
10. **until** $j = K$
11. **return** K_j
12. **end procedure**

IV. Experiment and Discussion

A. Experiment Environment

In this section, our proposed IDS method PMBKM is tested over KDDCUP99 dataset [31]. The experiment is implemented by Python on a windows 10 Operating System, which the processor is Inter Core i7 2.7GHZ, the RAM is 16GB, and the main software platform is Eclipse and Anaconda 2.7 SCIkit-learn. KDDCUP99 dataset is a 9-week network connection data collected from a simulated LAN of US Air Force. The dataset contains two types of files, one is 10% dataset named as KDDCUP.Data.10.percent.correceted and the other is the full dataset named as kddcup.data.corrected. Each data record contains forty one fixed feature attributes. Actually, there are two official documents, the first one has been labeled and the other one has no labels. We will choose the latter one for our clustering experiments.

B. The comparison experiments design

In step 2, four kinds of clustering methods are implemented and compared. Thus, Mini Batch Kmeans with PCA (PMBKM) (**Algorithm 2**), Kmeans (KM), and Kmeans method with PCA (PKM), as well as Mini Batch Kmeans (MBKM). We show a basic description of the other three methods. The process of KM is shown as follows. This algorithm firstly selects K centers from the given dataset T_1 by calling Kmeansplusplus. The procedure of Kmeansplusplus is the same as the function in Kmeansplusplus in **Algorithm 4**. And then it calculates the distance between each point in T_1 and K centers and then assigns each point to the nearest center. Thirdly, recalculates the center of each cluster. Finally, k clusters will be formed and CH is returned by calling score function. As the main idea of PKM is the same as KM. And therefore we only describe the difference between KM and PKM. The main difference is that PKM uses PCA function firstly (**Algorithm 3**). The remaining process is the same as the former one. Meanwhile, as the main idea of MPKM is the same as PMBKM. And therefore we only describe the difference between algorithm MBKM and algorithm PMBKM. The main difference is that MBKM does not use PCA function (**Algorithm 3**).

C. Experimental Results and Discussion

Selecting a different value of K, we mainly conduct two groups of experiments. The first group is that K= {10, 20, 30, 40, 50, 60, 70, 80, 90,100} and the other one is that K= {2, 3, 4, 5, 6, 7, 8, 9}. The above four methods are compared from the perspective of clustering time t and CH. The experiments are presented over both 10% dataset and full dataset.

(1)The first group (K= {10, 20, 30, 40, 50, 60, 70, 80, 90,100}**)**

We present the experiment over 10% dataset and full dataset.

1) 10% dataset

As shown in Figure 1, MBKM and PMBKM are time guaranteed. With the increasing of K, the clustering time is acceptable and stable. Additionally, the clustering time of PMBKM is much shorter than MBKM. That means the formal one is more suitable for big data environment.

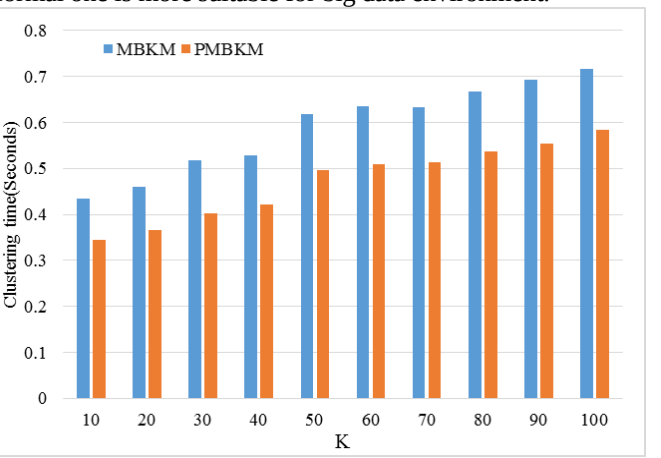

Figure 1. Clustering time comparison of MBKM and PMBKM over 10% dataset

As shown in Figure 2, CH of MBKM has been ignorant compared with PMBKM, meanwhile, PMBKM obtains the best clustering results when K equals 40. Next, we test the experiment over full dataset.

10.1109/ACCESS.2018.2810267, IEEE Access

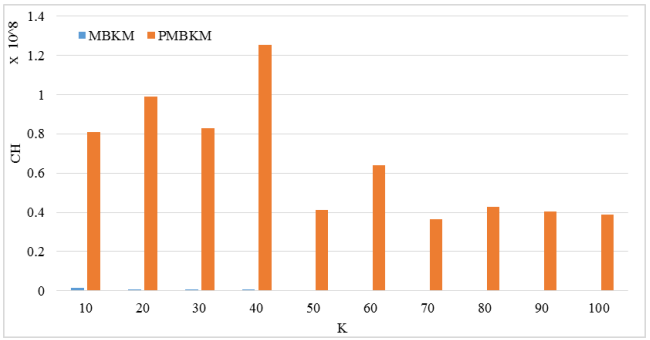

Figure 2. CH comparison of MBKM and PMBKM over 10% dataset

2) Full dataset

As shown in Figure 3, the clustering time over full dataset is the same as the situation over 10% dataset. Both MBKM and PMBKM are effective and the latter one is much better regardless of the value of K .

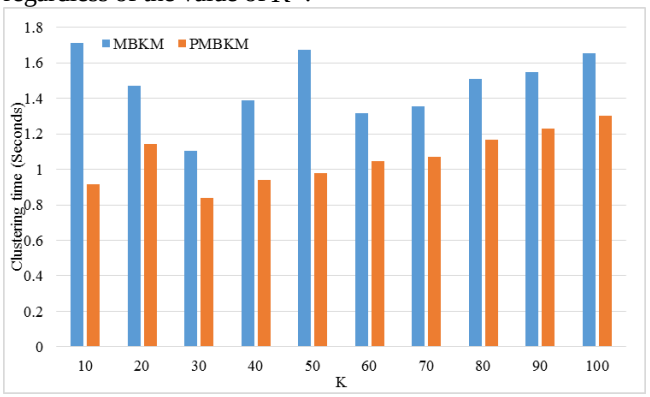

Figure 3. Clustering time comparison of MBKM and PMBKM over Full dataset

As shown in Figure 4, *CH* of MBKM has been ignorant compared with PMBKM, meanwhile, PMBKM obtains the best clustering results when K equals 30.

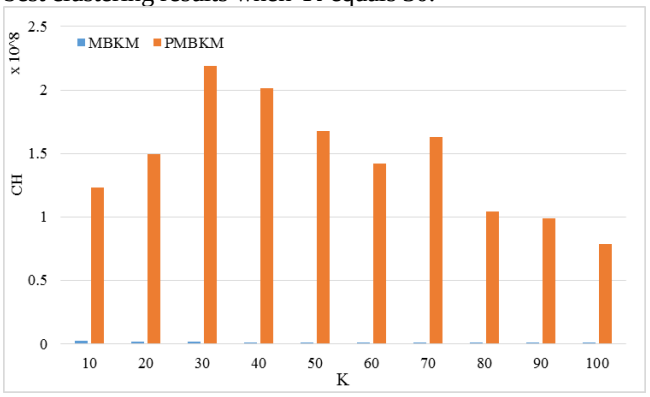

Figure 4. CH comparison of MBKM and PMBKM over full dataset

3) The experimental results comparison between 10% dataset and full dataset

We analyze the clustering time and *CH* for the proposed method PMBKM in both 10% dataset and full dataset. As shown in Figure 5, the clustering time increases with the increase of dataset, but the change is stable. At the same time, with the change of K , the time also has a certain change, but the change is stable.

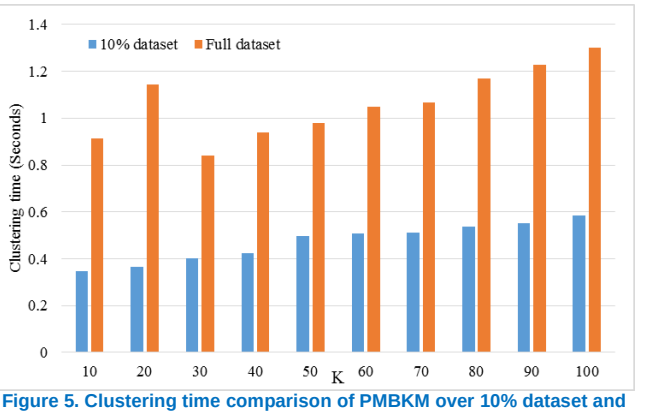

Figure 5. Clustering time comparison of PMBKM over 10% dataset and full dataset

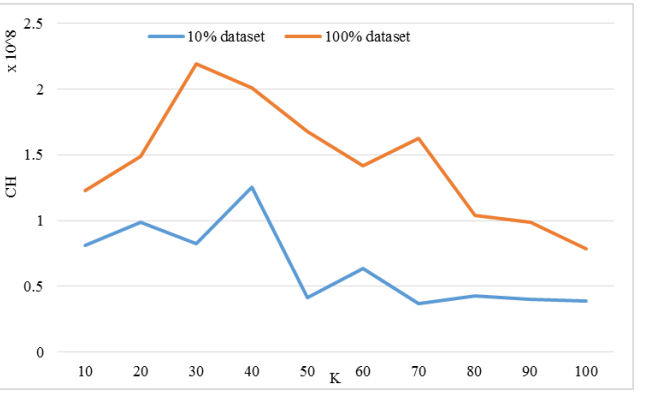

Figure 6. CH comparison of PMBKM over 10% dataset and full dataset

Next, we analyze *CH* of PMBKM method in both cases of 10% dataset and full dataset. As shown in Figure 6, the trend of the curve is similar, it starts to rise and then drop. The difference is that PMBKM obtains the best clustering result when K equals 40 over 10% dataset while K equals 30 over full dataset. That means our method is effective and can be used for different scenarios.

Overall, from the perspective of clustering time and *CH* , we can conclude that our proposed method PMBKM is effective and can be used for big data environment.

(2) **The second group (**K= {2, 3, 4, 5, 6, 7, 8, 9 }**)**

Because the K value is relatively small may be more sensitive to the center value. We added a second group of experiment to test the validity of our method. Thus, test the situation that K= {2, 3, 4, 5, 6, 7, 8, 9 }.

1) 10% dataset

As shown in Figure 7, both MBKM and PMBKM are time guaranteed. And the latter one is much better regardless of the value of K . In addition, the clustering time is very stable, and thus PMBKM is very efficient.

10.1109/ACCESS.2018.2810267, IEEE Access
IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

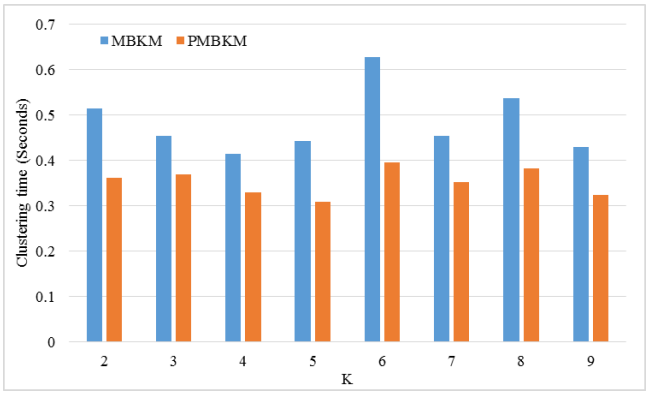

Figure 7. Clustering time comparison of MBKM and PMBKM over 10% dataset

As shown in Figure 8, *CH* of MBKM has been ignorant compared with PMBKM. In addition, PMBKM is increasing with the change of K value. And PMBKM obtains the best clustering result when K equals 9. And thus our method is still effective when the value of K is quite small over 10% dataset.

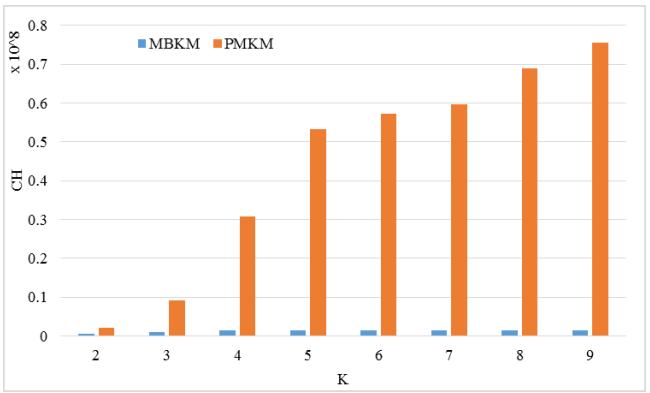

Figure 8. CH comparison of MBKM and PMBKM over 10% dataset

2) Full dataset

As shown in Figure 9, both MBKM and PMBKM are time guaranteed. And the latter one is much better regardless of the value of K. In addition, the clustering time is very stable, and thus PMBKM is very efficient.

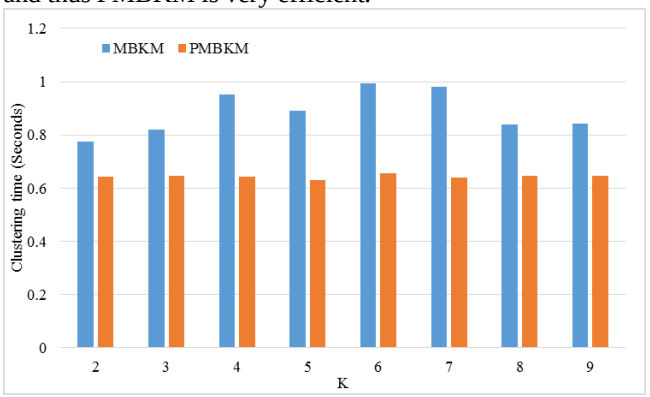

Figure 9. Clustering time comparison of MBKM and PMBKM over Full dataset

As shown in Figure 10, *CH* of MBKM has been ignorant compared with PMBKM. In addition, PMBKM is increasing with the change of K-value. And PMBKM obtains the best clustering result when K equals 9. And thus our method is still effective when the value of K is quite small over full dataset.

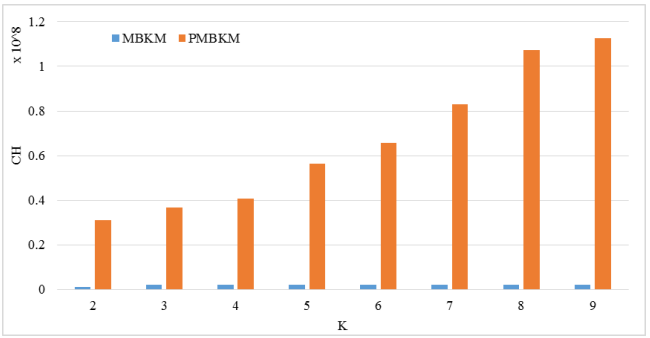

Figure 10. CH comparison of MBKM and PMBKM over full dataset

3) The experimental result comparison between 10% dataset and full dataset

We analyze the clustering time t and *CH* for the proposed method PMBKM in both 10% dataset and full dataset. As shown in Figure 11, the clustering time is stable in both two situations. And thus, PMBKM is effective with the increasing of dataset.

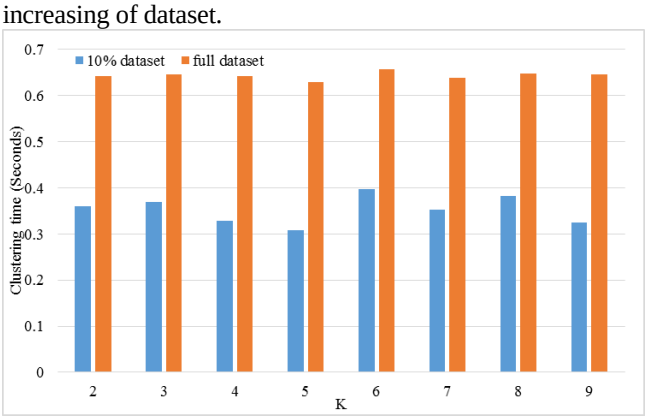

Figure 11. Clustering time comparison of PMBKM over 10% dataset and full dataset

Next, we analyze *CH* of this method in both cases of 10% dataset and full dataset. As shown in Figure 12, the trend of the curve is similar, *CH* is increasing with the change of K. PMBKM obtains the best clustering result when K equals 9 over both two kinds of dataset. Above all, from the perspective of calculation time t and *CH*, we can conclude that our proposed method PMBKM is efficient and effective.

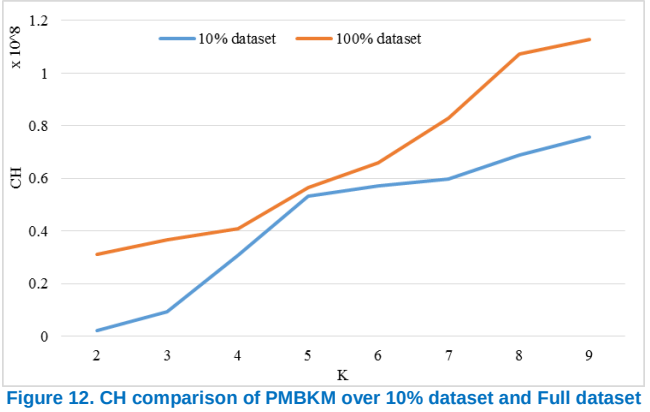

Figure 12. CH comparison of PMBKM over 10% dataset and Full dataset

10.1109/ACCESS.2018.2810267, IEEE Access
IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

4) Experiment comparison of KM, PKM, MBKM and PMBKM over 10% dataset

The four methods, KM, PKM, MBKM, as well as PMBKM are compared. Because the calculation time overhead of KM and PKM algorithms is very large, we only compare the experimental results over 10% dataset. As shown in Figure 13, from the perspective of clustering time, our proposed PMBKM is the best. As shown in Figure 14, *CH* of PKM is the best, followed by PMBKM. Since PMBKM randomly selects some samples (200 samples in this study) not all the samples for calculation, the value of *CH* is generally lower than MBKM. However, since the value of *CH* is increasing, we still can conclude which K is the best one. Thus the best clustering result of PMBKM is that K equals 30. In addition, PMBKM obtain the same *CH* as PKM when K equals 30. Above all, our proposed PMBKM is effective.

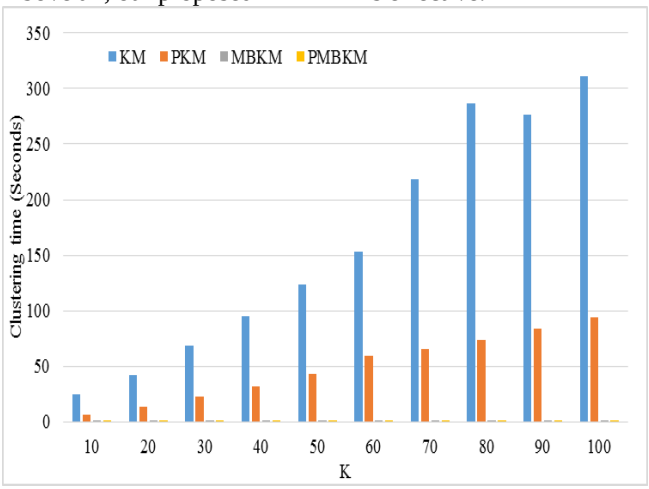

Figure 13. Clustering time comparison of KM, PKM, MBKM and PMBKM over 10% dataset

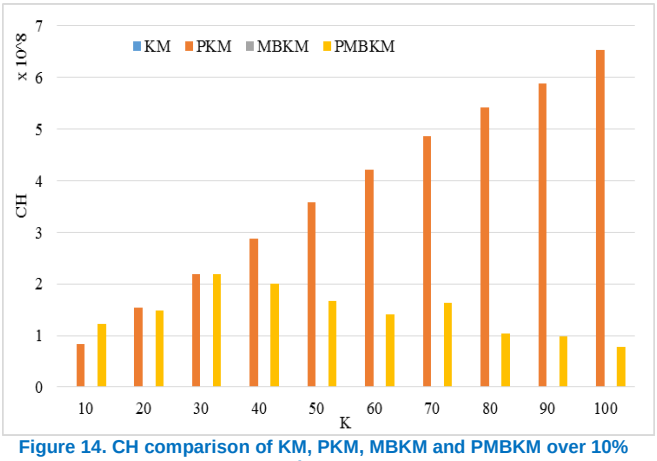

Figure 14. CH comparison of KM, PKM, MBKM and PMBKM over 10% dataset

5) Discussion

Time complexity analysis of each Method (PMBKM, KM, PKM, MBKM)

The PMBKM combines all three algorithms of Kmeans++, PCA, and Mini Batch Kmeans. As the time complexity of Kmeans++ is $O(m)$, the time complexity of PCA is $O(n*n*d)$ and the time complexity of Mini Batch Kmeans is $O(b*t*K)$. And thus the time complexity of PMBKM is $O(m+n*n*d+b*t*\mathrm{K})$. The time complexity of MBKM is $O(\mathrm{m}+b*t*K)$. As shown in the Section III, m is the number of data records. n is the dimension of dataset and d is the number of principal components. b is the given mini batch size and t is the number of iteration. K is the number of clusters. Obviously, m is the smallest value among the three cost, while the value of $(b*t*K)$ is larger than $(n*n*d)$. Hence, the complexity of PMBKM and MBKM is in the same level. Based on the same principle, the time complexity of KM is $O(\mathrm{m}+m*t*K)$, the time complexity of PKM is $O(m+n*n*d+\mathrm{m}*t*\mathrm{K})$. As a result of b is much smaller than m , and thus both MBKM and PMBKM obtain the low cost. Furthermore, PMBKM is much better than MBKM as a result of the clustering efficiency is improved after PCA process. Because the PMBKM only needs to take into account of d features instead of n features in clustering process. Moreover, both the 10% dataset and full dataset of KDDCUP99 are tested in our study, the experimental results show that the performance of PMBKM has been stable with the increasing of data number. So we can conclude that our propose method PMBKM is efficient and can be used for big data.

Discussion with relevant researches

In addition, the authors in [22] propose using the Apache Spark for IDS in big data environments, both the 10% dataset and the full dataset are tested. Although they consider and improve the efficiency of clustering, unfortunately we cannot effectively determine which K-value clustering works best from the experimental results presented by them. Meanwhile, the clustering time in this study is much lower than that of theirs. Next, we compare PMBKM with the method in [23]. We mainly discuss the situation when K equals 1000 over both 10% dataset and the full dataset. As shown in Figure 14, both MBKM and PMBKM are time efficiency. This also proves that our proposed method PMBKM can be used in different scenarios, regardless of which value K chooses. However, we hold the opinion that there is no need to choose so big value of K. For one thing, it does not mean that the bigger the clustering value is, the larger the *CH* will be. In addition, as shown in Figure 6, PMBKM obtains the best clustering results when K equals 30. As shown in Figure 6 and Figure 16, CH at K equals 1000 is basically the same as K equals 30, however, the time overhead greatly increases when K equals 1000. For another, the clustering result is the reference for classification, too many clusters will reduce the accuracy of the classification.

10.1109/ACCESS.2018.2810267, IEEE Access

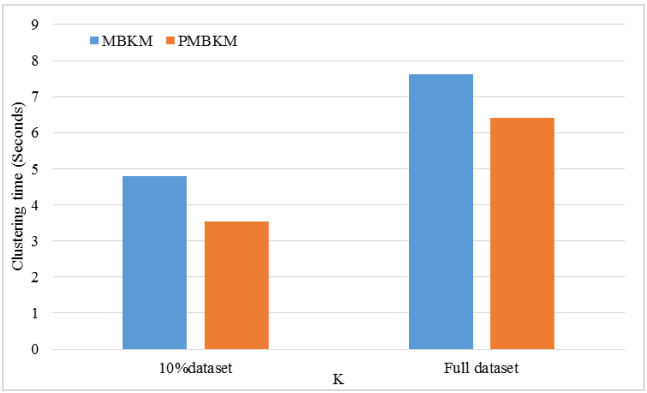

Figure 15. Clustering time comparison of MBKM and PMBKM over 10% dataset and full dataset

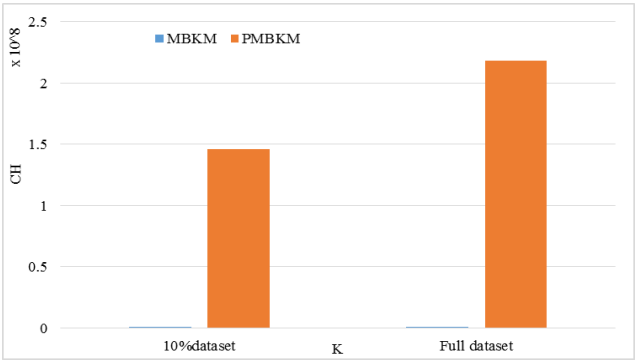

Figure 16. CH comparison of MBKM and PMBKM over 10% dataset and Full dataset

V. Related work

IDS based on data mining is mainly concentrated in two aspects, thus clustering and classification. The initial dataset of the clustering algorithm has no labels while the classification algorithm has labels. In this study, we mainly focus on the clustering research. The clustering algorithms can be divided into the following categories, thus partitioning methods, hierarchical methods, density-based methods, grid-based methods, as well as model-based methods. Clustering algorithms based on partitioning mainly include Kmeans algorithm and Kmedoids algorithm [19]. For a given dataset, the N objects contained in the dataset need to be divided into K clusters, each cluster represents a division. The partitioning methods mainly use similarity measurement function such as Euclidean distance function and angle cosine function. Differ from the partitioning methods, Hierarchical clustering method does not need to preset the number of clusters. The typical hierarchical clustering algorithms mainly contain Birch algorithm [24] and Cure algorithm [25]. In addition, there are density-based clustering algorithms which include Denclue [26], DBSCAN [27] as well as OPTICS [28]. Moreover, there are grid-based clustering algorithms, such as WaveCluster [29] and Clique [30]. These algorithms firstly divide the data space into a limited number of grids and record the data point information contained in each grid, and then present the clustering operation. On the whole, the above clustering methods are designed for solving different issues and there is no universal clustering method for solving all the problems. In this study, we mainly engage in the application of Kmeans for IDS. Muniyand et al. [20] proposed using "Kmeans + C4.5" for classifying anomalous and normal activities in a computer network. Firstly, the given training examples are partitioned into K clusters by using Kmeans. And each cluster represented a density region of normal or abnormal instances. And then the decision tree was established by using C4.5 decision tree algorithm for the attacks detection. Moreover, as a result of the convergence of Kmeans is highly depending on the initial center point. In order to overcome this drawback, Liu et al. [21] proposed an improved Kmeans clustering algorithm. Unfortunately, the above researches did not pay attention to the clustering efficiency. Harifi et al. [22] proposed using Kmeans over Apache Spark Machine Learning Library, and three benchmark datasets, such as Forest Cover Type, KDDCUP99 and Internet Advertisements are tested for their experiments. Although they consider the efficiency of clustering, unfortunately we cannot effectively decide which K-value clustering works best from the experimental results presented by them. Siddiqui et al. [23]proposed using Kmeans clustering algorithm over ODM and 1000 clusters are built to segment the given data records. To the best of our knowledge, a good clustering result does not mean that the more clusters are, the better the clustering results will be. In addition, the experiment in [23] is only presented over 10% KDDCUP99 dataset. Consequently, we cannot judge the performance of the algorithm over full dataset. Differ from the above researches, both the clustering efficiency and results evaluation are taken into account in this study. The Mini Batch Kmeans method is used for improving the efficiency of clustering. The Kmeans++ method is used to initialize the cluster centers so as to avoid the algorithm getting into the local optimum. Additionally, the CH indicator is chosen to evaluate the result so that it is easier to determine which K in the clustering result is best.

VI. CONCLUSION

In this study, we proposed a clustering method based on Mini Batch Kmeans with PCA (PMBKM) for Intrusion Detection System. Taking IDS classic dataset KDDCUP99 for example, both 10% dataset and full dataset are tested. Firstly, we preprocess the given dataset and then the PCA method is used to reduce the dimension so as to improve the clustering efficiency. Additionally, the Mini Batch Kmeans method is used for the clustering of the processed dataset. Compared with Kmeans (KM), Kmeans with PCA (PKM), as well as Mini Batch Kmeans (MBKM), the experimental results show that our proposed PMBKM is effective and efficient. Above all, PMBKM can be used for intrusion detection system over big data environment. In our future work, we will engage in the research of clustering method over fog computing.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publicati
10.1109/ACCESS.2018.2810267, IEEE Access

REFERENCES

[1] J. P. Anderson, "Computer security threat monitoring and surveillance, "James P. Anderson Company, Fort Washington, Pennsylvania, *Tech. Rep*, vol. 17, Apr.15, 1980.

[2] D. E. Denning. "An intrusion-detection model," *IEEE Trans. Softw. Eng.*, vol.2, pp.222-232, 1987.

[3] A. Milenkoski et al., "Evaluating Computer Intrusion Detection Systems: A Survey of Common Practices," *ACM Comput. Surv.*, vol. 48, no.1, pp.1-41, Sep.29, 2015.

[4] K. Peng et al., "Link Importance Evaluation of Data Center Network Based on Maximum Flow," *J. Internet Technol.*, vol.18, no.1, pp. 23-31, Jan.1, 2017.

[5] X. L. Xu et al., "Data placement for privacy-aware applications over big data in hybrid clouds," *Secur. Commun. Netw.*,2017,https://doi.org/10.1155/2017/2376484.

[6] L. Y. Qi et al., "Structural Balance Theory-based E-commerce Recommendation over Big Rating Data," *IEEE Trans. Big Data.*, DOI:10.1109/TBDATA.2016.2602849.

[7] H. Zhou et al., "Predicting Temporal Social Contact Patterns for Data Forwarding in Opportunistic Mobile Networks," *IEEE Trans. Veh. Technol.*, vol.66, no.11, pp. 10372-10383, Nov, 2017.

[8] S. G. Wang et al., "Service Composition in Cyber-Physical-Social Systems," *IEEE Trans. Emerg. Top. Comput.*, 2017, no.99, pp.1-1.doi: 10.1109/TETC.2017.2675479

[9] L. Y. Qi et al., "A Context-aware Service Evaluation Approach over Big Data for Cloud Applications," *IEEE Trans. Cloud Comput.*, DOI: 10.1109/TCC.2015.2511764.

[10] H. Zhou, H. Wang, X. Li, and V. Leung. "A survey on Mobile Data Offloading Technologies," *IEEE Access.*, vol. pp, no. 99, pp. 1-11, Jan. 2018.

[11] R.L Deng, P. Zhuang, and H, Liang, "CCPA: Coordinated Cyber-Physical Attacks and Countermeasures in Smart Grid," *IEEE Trans. Smart Grid.*, vol.8, no.5, pp.2420-2430, Sept.2017.

[12] H. Zhou et al., "Analysis of Event-driven Warning Message Propagation in Vehicular Ad Hoc Networks," *Ad Hoc Netw.*, vo.55, pp.87-96, Feb.2017.

[13] M. Wall, "Big Data: Are you ready for blast-off," *BBC Business News.* Mar. 2014.

[14] CSICO, "Forecast and Methodology. 2013-2018," *Cisco Systems*, San Jose, CA, USA, 2013.

[15] X. W. Chen, X. Lin, "Big data deep learning: challenges and perspectives," *IEEE Access*, vol.2, pp.514-525, May.16, 2014.

[16] X. Wu et al., "Data mining with big data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97-107, Jan. 2014.

[17] W. Lee, S. J. Stolfo, "Data mining approaches for intrusion detection," *in 7th USENIX. Secur. Symp.*, pp. 79-93, Jan.26, 1998.

[18] J. G. Sun, J. Liu, and L. Y. Zhao, "Clustering algorithms research," *J. Software*, vol.19, no.1, pp.48-61, Jan.2008.

[19] H. S. Park, C. H. Jun, "A simple and fast algorithm for K-medoids clustering," *Expert Syst. Appl.*, vol.36, no.2, pp.3336-3341, Mar.31, 2009.

[20] A. P. Muniyandi, R. Rajeswari, R. Rajaram, "Network anomaly detection by cascading k-Means clustering and C4. 5 decision tree algorithm," *Procedia Eng.*, vol.30, pp.174-182, Dec.31, 2012.

[21] H. C. Liu, X. N. Hou, Z. Yang, "Design of Intrusion Detection System Based on Improved Kmeans Algorithm," *Comput. Technol. Dev.*, vol.26, no.1, Jan.2016.

[22] S. Harifi, E. Byagowi, and M. Khalilian, "Comparative Study of Apache Spark MLlib Clustering Algorithms," *In Proc. Int. Conf. Data Min. Big Data*, pp.61-73, Jul.27-Aug.1, 2017.

[23] M. K. Siddiqui, S. Naahid, "Analysis of KDDCUP 99 dataset using clustering based data mining," *Int. J. Data Theory Appl.*, vol.6, no.5, pp. 23-34, June.2013.

[24] S. Madan, K. J. Dana, "Modified balanced iterative reducing and clustering using hierarchies (m-BIRCH) for visual clustering," *Pattern Anal. Appl.*, vol.19, no.4, pp.1023-1040, Nov.1, 2016.

[25] S. Guha, R. Rastogi, and K. Shim, "CURE: an efficient clustering algorithm for large databases," *ACM SIGMOD. Rec.*, vol. 27, no. 2, pp. 73-84, Jun.1, 1998.

[26] A. Hinneburg, D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," *in 4th. Int. Conf. Knowl. Discovery. Data Min.*, vol. 98, pp. 58-65. Aug.27, 1998.

[27] M. Ester et al, "A density-based algorithm for discovering clusters in large spatial databases with noise," *in 2nd. Int. Conf. Knowl. Discovery. Data Min.*, vol. 96, no. 34, pp.226-231, Aug.2, 1996.

[28] M. Ankerst et al., "OPTICS: ordering points to identify the clustering structure," *ACM SIGMOD. Rec.*, vol.28, no.2, pp.49-60, Jun.1, 1999.

[29] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "Wavecluster: A multi-resolution clustering approach for very large spatial databases," *In 24th. Int. Conf. Very Large Databases.*, vol. 98, pp. 428-439, Aug.24, 1998.

[30] D. Medeiros, "Process mining based on clustering: A quest for precision," *In Proc. Int. Conf. Bus. Process Manage.*, vol. 4928, pp. 17-29, Sep.24, 2007.

[31] KDDCUP99, "http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html".

[32] D. Sculley, "Web-scale k-means clustering," *In Proc. 19th Int. Conf. World. Wide. Web.*, Apr.2010, pp.1177-1178.

[33] Scikit-learn. http://scikit-learn.org/stable/index.html

[34] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions,"

10.1109/ACCESS.2018.2810267, IEEE Access

http://resolver.caltech.edu/CaltechAUTHORS:20111012-111324407

[35] Kmeans++,"https://en.wikipedia.org/wiki/K-means%2B%2B".

[36] T. Calinski and J. Harabasz, "A dendrite method for cluster analysis," *Commun. Stat.-Theory Methods.*, vol.3, no.1, pp. 1-27, Jan.1, 1974.

[37] D. Arthur, S. Vassilvitskii, "How slow is the k-means method?," *In Proc. 22nd. Annu. Symp. Comput. Geom.*, pp.144-153, Jun.5-7, 2006.

[38] M. E. Tipping, C.M. Bishop, "Mixtures of probabilistic principal component analyzers", *Neural Comput.*, vol.11, no.2, pp.443-82, Feb.15, 1999.

[39] P. G. Martinsson, V. Rokhlin, M. Tygert, "A randomized algorithm for the decomposition of matrices", *Appl. Comput. Harmon. Anal.*, vol.30, no.1, pp.47-68 Jan.1, 2011.

[40] L. Bottou, Y. Bengio, "Convergence properties of the k-means algorithms", *Adv. Neural. Inf. Process. Syst.*, pp. 585-592, 1995.