K-means+: An Autonomous Clustering Algorithm

Yu Guan[1, 2 *]*, Ali A. Ghorbani*[1]*, Nabil Belacel*[1,2]

[1]*Department of Computer Science, University of New Brunswick, Canada*
[2]*E-Health group, IIT, National Research Council of Canada*

Abstract

The traditional clustering algorithm, K-means, is famous for its simplicity and low time complexity. However, the usability of K-means is limited by its shortcoming that the clustering result is heavily dependent on the user-defined variants, i.e., the selection of the initial centroid seeds and the number of clusters (k). A new clustering algorithm, called K-means+, is proposed to extend K-means. The K-means+ algorithm can automatically determine a semi-optimal number of clusters according to the statistical nature of data; moreover, the initial centroid seeds are not critical to the clustering results. The experiment results on the Iris and the KDD-99 data illustrate the robustness of the K-means+ clustering algorithm, especially for a large amount of data in a high-dimensional space.

Keywords: Clustering; K-means; unsupervised learning

1. Introduction

Classification is the process of partitioning a group of existing objects into different classes in order to extract desired models for predicting the classes of new objects. In other words, it is a learning process that allows us to find models or rules for projecting data onto a number of classes. Many classification methods have

[*] Corresponding author. Tel.: +1-506-636-3772; fax: +1-506-636-4491.
E-mail address: yu.guan@nrc-cnrc.gc.ca (Y. Guan).

been proposed for knowledge discovery and pattern recognition. In general, classification approaches can be divided into two categories: the supervised learning methods and the unsupervised learning methods. The supervised learning methods, such as Multilayer Perceptron (*MLP*) [1], Support Vector Machine (*SVM*) [2], K-Nearest Neighbor [3] and Decision Tree [4], establish models by mapping inputs to the known outputs, i.e., mapping objects to the known classes; these models can be used for predicting the patterns of new objects.

Different from the supervised learning methods, the unsupervised learning methods, known as clustering methods, do not have the knowledge of the classes to which the objects can be mapped. The classes can be attained in the learning processes; consequently, models can be established by mapping examples to the classes. Clustering divides a collection of objects into different clusters according to their similarities. High inter-object similarity within each cluster and low inter-cluster similarity are desired for the clustering space. K-means is a very straightforward clustering algorithm [5]. It partitions a collection of objects into a number of clusters by assigning them to their closest clusters. The centroid of each cluster is the mean vector of the cluster members. The user usually needs to define the number of clusters (k). The similarity between two objects is usually measured using Euclidean distance.

K-means has been used as a popular clustering method due to its simplicity and high speed in clustering large data sets. However, K-means has two shortcomings: dependency on the initial state and degeneracy [6]. The initial state includes the selection of initial centroids and the value of k. Different selections of initial centroids often lead to different clustering results because the algorithms based on the mean squared-error often converge to local minima [7]. This is especially true if the initial centroids are not well separated. Usually, the better the initial centroids are separated, the better the clustering result can be obtained. Additionally, the number of clusters (*k*) is also critical to the clustering result, and obtaining an optimal *k* for a given data set is an *NP*-hard problem [6]. When the distribution of the data set is unknown, the optimal *k* is hard to attain. The second shortcoming, degeneracy, is that the clustering may end with some empty clusters, i.e., K-means stops with less *k* non-empty clusters. This clustering result is not what we expect since the classes of the empty clusters are meaningless for the classification.

Two possible methods can be applied for eliminating the degeneracy: (1) deleting the empty clusters, and (2) replacing the empty clusters with newly created non-empty clusters. The former solution reduces the number of clusters while the latter does not change it. The latter has to search for suitable objects to form a non-empty clusters to replace the empty ones; afterwards all the objects need to be re-assigned to their closest clusters until there is not empty cluster and all centroids are stable. This method is obviously more complicated and expensive than the former method. The H-means+ algorithm, an improved version of K-means, eliminates the degeneracy by using the latter method [8]. Whenever an empty cluster is created, H-means+ removes the *global furthest object* (*GFO*), and uses it to create a new cluster to replace the empty cluster. Afterwards, all the objects are reassigned to their closest centroids the same as the K-means algorithm does. This iteration of eliminating empty clusters and reassigning objects will continue until no empty clusters exist. Here, the *GFO* is one of the *local furthest objects* (*LFO*s). A *LFO* of a cluster is defined as the object that has the largest Euclidean distance from the centroid comparing to its siblings in the same cluster; therefore, for *k* clusters, there are *k LFO*:

Clusters:	$C_1, C_2, \ldots, C_i, \ldots, C_k$
local furthest objects:	$LFO_1, LFO_2, \ldots, LFO_i, \ldots, LFO_k$
Euclidean distances from LFO_i *to* C_i*:*	$d_1, d_2, \ldots, d_i, \ldots, d_k$

Let $d_m = \max(d_1, d_2, \ldots, d_i, \ldots, d_k)$, then *GFO is* LFO_m. Although, H-means+ can prevent the degeneracy by eliminating empty clusters, it still suffers from the shortcoming: dependency on the initial state.

Leonid Portnoy proposed a new clustering algorithm based on K-means [9]. Similarly, Portnoy's algorithm also uses Euclidean distance to determine the similarity between objects. The value of *k* is automatically defined by Portnoy's algorithm. But this method requires the user to define the upper bound of the cluster widths (*W*) and the initial value of *k* before clustering. During the clustering, each object needs to find its closest centroid; if the distance between the object and its closest centroid is less than *W,* it joins the cluster; otherwise, it forms a new cluster, and the value of *k* increases. Note that, *k* increases monotonically, and its final value still heavily depends on its initial value as well as the value of *W*. A small *W* usually leads a large increment of *k,* and a large *W* leads a small or even no increment of *k*. When the distribution of a data set is unknown, a proper value of *W* is difficult to obtain.

In [10] a new genetic clustering heuristic, called CLUSTERING, is proposed. This heuristic integrates the genetic strategy into the nearest-neighbor algorithm. It is proposed to find a proper number as the number of clusters automatically. However, it still needs the user to specify two variables: the threshold, *d*, and the variable, *w*; and their values are critical to the clustering results. The number of generations specified by the user also affects the result significantly. On the other hand, the randomly generated initial strings at its initialization step and the randomly selected position for substring crossover make the performance of CLUSTERING show a random feature according to our experiments.

This paper introduces a new clustering method, named K-means+. It is developed based on the K-means algorithm. Different from the latter, K-means+ adjusts the value of *k* autonomously by exploiting the statistical nature of the data. In other words, K-means+ partitions the data into an appropriate number of clusters rather than an ad hoc fixed number of clusters; moreover, the initial clustering state is not critical to the final clustering results. K-means+ also eliminates the shortcoming of degeneracy by removing the empty clusters. Additionally, K-means+ uses multi-centered instead of mono-centered clusters to obtain better performances.

The rest of this paper is organized as follows. In Section 2, the K-means+ heuristic of clustering is introduced. Section 3 presents comparative computational results on the Iris [11] and the KDD-99 data [12]. The paper is concluded in Section 4.

2. Proposed Heuristic

Being developed from the K-means algorithm, K-means+ is an implementation of sum-square-error minimization as well. The main difference between the two algorithms is that the number of clusters in K-means+ is a self-defined variable instead of a user-defined constant. The user-defined *k* cannot always guarantee an appropriate partition of objects with an arbitrary distribution. An improper value of *k* usually leads to a poor clustering. One solution to this problem is to find an appropriate number of clusters by trying all the possible values of *k*. However, this approach suffers from a large time complexity as much as $O(mnkC_n^k)$, where n is the number of objects to be partitioned, and m is the number of iterations of the loop for stabilizing cluster centroids. Obviously, this approach is unpractical for a large data set. Our approach to this problem is to obtain a semi-optimal *k* according to the statistical properties of the data. For instance, if the granularities of clusters are too 'coarse' (i.e., the initial value of *k* is too small), we split the 'coarse' clusters to make them finer. On the other hand, if the clusters are too 'fine' (i.e., the initial value of *k* is too large), we merge some contiguous clusters to form larger clusters. Even without knowledge of the objects' distribution, the K-means+ algorithm can determine an appropriate value of *k* by splitting and merging clusters.

The K-means+ algorithm uses the Euclidean distance to calculate the similarity between two objects. In order to avoid some attributes (features) dominating other attributes in calculating Euclidean distance the data must be normalized. The initial value of *k* is chosen from the set $\{2, 3, \ldots, n\}$. First, these data are partitioned into *k* clusters in the same way as K-means does, then the K-means+ algorithm splits clusters by removing outliers from existing clusters to form new clusters. An outlier is an object that is far from the majority of the objects in a cluster. At the splitting stage, outliers are removed from their current clusters, and are assigned as new centroids. These new centroids may attract some objects from adjacent clusters to form new clusters. In this way, the coarse clusters are split into fine clusters, and the value of *k* is increased. During the process of splitting clusters, the module of eliminating degeneracy will be frequently revisited in case of empty clusters occur. If there are empty clusters after an iteration of clustering, K-means+ simply deletes them without creating new non-empty clusters to replace the empty ones, and thus avoids the time cost in the iterations of searching *GFO* and re-clustering.

After the splitting procedure, K-means+ may merge some adjacent clusters by linking them to form larger clusters. The centroids of linked clusters are kept intact after linking; therefore, the newly formed clusters are multi-centered, and they can be in arbitrary shapes, e.g. spatial chains. These multi-centered clusters are more appropriate than the mono-centered spherical clusters for classification. The detail of splitting and linking procedures are discussed below.

2.1 Splitting Clusters

An outlier is an object that is quite different from the majority of the objects in a cluster. When the Euclidean distance is used to measure the similarity between two objects, an outlier is an object that is far from the majority of the objects. One can find outliers by comparing the radii of the objects; that is, if the radius of an object is greater than a given threshold, it is deemed an outlier. This idea of determining outliers comes from the theory of robust regression and outlier detection [13]. As shown in Figure 1, a set of objects are partitioned into $k=3$ clusters. Let σ denote the standard deviation of cluster X. Each object is assigned to its nearest cluster. Object c represents the centroid of cluster X. Object p (the square object) is assigned to cluster X, since centroid c is the closest cluster that p can be assigned to. However, object p is far from the majority of the objects in the cluster and is probably a local outlier of cluster X. Let t_s denote a threshold of determining the outliers, and r represent the distance between object p and centroid c, i.e., $r = ||p, c||$. A point is deemed an outlier if $r > t_s$. The function of t_s of K-means+ is quite similar to that of W of Portnoy's method. The difference is that t_s is defined autonomously while W is user-defined.

*** Figure 1 is about here***

Central Limit Theorem: Let $x_1, x_2, \ldots, x_n$ denote a set of n independent random variables with an arbitrary probability distribution $P(x_1, x_2, \ldots, x_n)$ with mean μ and a finite variance σ [14] . Then, the normal form variable as shown in Equation (1) has a limiting cumulative distribution function which approaches a normal distribution [14].

$$\bar{X} \equiv \frac{\sum_{i=1}^{n} x_i - n\mu}{\sigma\sqrt{n}} \tag{1}$$

"The normal approximation in the Central Limit theorem will be good if $n \geq 30$ regardless of the shape of the population. If $n < 30$, the approximation is good only if the shape of the population is not drastically different from a normal distribution" [15]. By the central limit theorem, we can infer that many arbitrary distributions are close to the normal distribution, i.e., the Gaussian distribution. The closer to the mean, the larger the population of objects can be observed. There is a very important proposition regarding the normal distribution known as *the Empirical Rule* [15].

Empirical Rule: *for any normal distribution*

- *about 68.26% of the objects will lie within one standard deviation of the mean;*
- *about 95.44% of the objects will lie within two standard deviations of the mean;*
- *about 99.73% of the objects will lie within three standard deviations of the mean;*
- *about 99.994% of the objects will lie within four standard deviations of the mean;*
- *about 99.99994% of the objects will lie within five standard deviations of the mean.*

Besides the Empirical Rule, Chebyshev's Theorem, which was discovered by the Russian mathematician P.L. Chebyshev, shows that the fraction of the measurements falling between any two values symmetric about the mean is related to the standard deviation [15].

Chebyshev's Theorem: *for* any *data distribution, at least* $(1 - 1/m^2)$ *of the objects in any data set will be within m standard deviations of the mean, where m is any integer greater than one* [14].

By Chebyshev's Theorem, we can see that at least 96% of objects (majority) in a cluster lie within the sphare of radius = 5σ (i.e., 5 standard deviations of the mean). Chebyshev's Theorem gives the lower bound of the percentage. We might assume that the objects of a cluster are approximately in a normal distribution by the

Central Limit Theorem. The Empirical Rule estimates that about 99.99994% of objects stay within the sphere of radius = 5σ for a normal distribution. Therefore, we can set the threshold $t_s = 5\sigma$ for splitting, i.e., the objects that stay beyond the five standard deviations of the cluster centroid can be deemed an outlier.

Once an outlier is found, it is removed from its current cluster and is assigned as the centroid of a new cluster. Then, all the data are partitioned again into k+1 clusters as illustrated in Figure 1.

The split procedure makes the cluster-granularities finer and the objects within the same cluster more similar to each other. On the contrary, if the initial k is too large, we may need to merge some close clusters to reduce the number of clusters.

2.2 Linking Clusters

In the merge procedure, we also need to set a threshold (t_m) for linking clusters. Using Chebyshev's theorem, we observe that when $m = \sqrt{2} \approx 1.414$, there are at least 50% of the objects within $m = 1.414$ standard deviations of the mean, which suggests that the objects in a cluster are approximately in a normal distribution. Let t_m denote the threshold of linking and defined as

$$t_m = m\,(\sigma_x+\sigma_y) = 1.414\,(\sigma_x+\sigma_y) \tag{2}$$

Let d represent the Euclidean distance between two cluster centroids. If $d < t_m$, as shown in Figure 2, some objects of cluster X are probably closer to the centroid of the cluster Y than some objects of cluster Y are. Thus, we may merge them into one cluster. In Figure 2, c_x and c_y represent the centroids of clusters X and Y *respectively*; and σ_x and σ_y denote their corresponding standard deviations.

There are two possible approaches to merge close clusters: fusing and linking. The first approach is to fuse two cluster centroids into a new centroid. The new centroid is set to the mean vector of the two ex-centroids. The standard deviation of the new cluster must be greater than the two former standard deviations. The threshold for merging this new cluster with its neighbors is enlarged. It probably leads to further merging until no neighbors are close enough to merge with.

There are two disadvantages of fusing clusters. Since each cluster can only have one centroid, the cluster can only be in spatial sphere-shape. However, the spatial spherical clusters may not properly reflect the arbitrary distribution of the real data. In the real world, objects can form clusters in arbitrary shapes, such as spatial concave or convex, or even a chain. The other disadvantage is that each merging may lead to expensive iterations for re-assigning all the objects to the updated centroids.

Figure 2 is about here

K-means+ uses the second approach of merging: linking close clusters. Their centroids will be kept intact and no new centroid is created. The merged cluster has multi centers. It is not required to re-assign data after linking. Another advantage is that the clusters can be in arbitrary shapes such as a chain.

3. Tests and Discussion

K-means+ is tested with the Iris data and the KDD-99 data. The results obtained by K-means+ are compared with those obtained by other well-known classification methods, such as K-means and Self-Organized Map (*SOM*) [16].

3.1 Performance Measures

Unsupervised learning methods, such as clustering algorithms, normally do not use the data labels for classification. However, the labels can be used for evaluating the performance of partition result; that is, a cluster can be labeled according to the majority data inside. For example, if the data with label "*X*" has the largest population in a cluster, the cluster will be labeled "*X*". During the test, each datum will be assigned to the closest cluster, and identified with the same label of the cluster.

Confusion matrix is a common measure of the performance of a classification method. It contains the information of the actual and predicted classification results [17]. For comparing and analyzing the performances of classification methods, accuracy (*AC*) is often used as the primary indicator. The accuracy is the proportion of the total number of correct classification, and is defined as,

$$AC = \frac{number\ \ of\ \ correctly\ \ labeled\ \ instances}{total\ \ number\ \ of\ \ instances} \tag{3}$$

AC is not sufficient to evaluate the classifier's performance when the number of instances of one class is overwhelmingly greater than the other [18]. For example, there are 10 000 instances, 9 990 of which are negative and 10 of which are positive. If all of them are classified as negative, the accuracy is 99.9% even though all of the positive instances are misclassified. For binary classifiers, true positive rate (*TP*), and false positive rate (*FP*), are also used to reinforce the accuracy [18]. True positive rate is the proportion of positive instances that were correctly classified, and it is defined as,

$$TP = \frac{number\ of\ correctly\ labeled\ \ positive\ instances}{total\ \ number\ of\ \ \ positive\ instances} \tag{4}$$

The false positive rate is the proportion of negative instances that are incorrectly classified as positive, and it is defined as,

$$FP = \frac{number\ of\ \ mis\text{-}labeled\ \ negative\ instances}{number\ of\ \ negative\ instances} \tag{5}$$

The confusion matrix is used to calculate accuracy, true positive rate and false positive rate.

3.2 Tests with the Iris data

The Iris data, which is created by R.A. Fisher, is a well-known dataset for classification. It has been used for testing many classification methods. This dataset contains 3 classes: Setosa, Versicolor, and Virginica [19]. Each class has 50 instances and refers to a type of Iris flower. Figure 3 illustrates the Iris data distribution in 3-dimensional space. There are totally C_4^3 = 4 combinations of three attributes of the Iris data. The graphs show that the Setosa class can be linearly separated from the Versicolor and Virginica classes, and the latter two classes are overlapping so that they are not linearly separable.

*** Figure 3 is about here***

Two-fold cross-validation is used for evaluating the classification methods. The Iris data are divided into two halves. One half is used for training data while another half for testing. Each of the both data sets has 25 Setosa data, 25 Versicolor data and 25 Virginica data. After the training data are partitioned into clusters, each cluster is identified according to the majority of the data inside. For example, if the Setosa data in a cluster has the largest population, the cluster is identified *Setosa*. During the test, each datum was assigned to its closest centroid, and identified with the same label of the closest cluster.

We run K-means with different values of k and different initial centroids, and obtained different clustering results. The best one when $k = 3$ is shown in Table 1.

The *SOM* toolbox from Matlab 6.1 is used to classify the Iris data. The configuration of *SOM* used in our simulation is:

$$net = newsom(PR,[1\ 3],'hextop', 'linkdist', 0.9, 1000, 0.05,1) \quad (6)$$

It means that the network has 1 layer, which has 1 × 3 dimension; the function is 'hextop', which calculates the neuron positions for layers whose neurons are arranged in a multi-dimensional hexagonal pattern. The distance function is 'linkdist', which is a layer distance function used to find the distances between the layer's neurons. The learning rate is 0.9. The number of the ordering phase steps is 1000. The tuning phase learning rate is 0.05. The tuning phase neighborhood distance = 1. The confusion matrix of the simulation is shown in Table 1. Hereinafter, *Se* denotes Setosa; *Ve* denotes Versicolor; and *Vi* denotes Virginica.

Attempting with different initial number of clusters from set {2, 3,…, 75}, K-means+ classified the testing data after the training with the training data, and produced 74 confusion matrices. All of them are quite similar to each other, and most of them are the same as Table 1. Even the initial centroids are randomly selected from Iris data; the results are still the same. Obviously, the initial number of clusters and the initial centroids are no longer critical to the clustering result. That is, K-means+ does not have the shortcomings of dependency on the initial state for the classification of the Iris data.

	Predictions with clustering algorithms								
	K-means (k=3)			*SOM* (1 × 3)			K-means+		
	Se	*Ve*	*Vi*	*Se*	*Ve*	*Vi*	*Se*	*Ve*	*Vi*
Se	25	0	0	25	0	0	25	0	0
Ve	1	23	1	0	24	1	0	0	25
Vi	0	5	20	0	2	23	0	0	25

Table 1: Confusion Matrices for the Iris data

Since the Versicolor and Virginica classes are not linearly separable, without the knowledge of the classes, K-means+ always identifies them as one class. Actually, no classification methods based on the unsupervised learning to date can separate the Versicolor and Virginica classes well without the knowledge of their classes.

3.3 Tests with KDD-99 data

The log data are the footprints of activities on computers and networks. Intrusion log data are usually different from normal log data. Therefore, clustering methods may be used to distinguish them by partitioning intrusion data and normal data into different clusters. We assess the applicability of K-means+ in intrusion detection using KDD-99 dataset [12].

The KDD-99 dataset was used for The Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99, the fifth International Conference on Knowledge Discovery and Data Mining [12]. The competition task was to build a network intrusion detector. This database was acquired from the 1998 DARPA intrusion detection evaluation program. An environment was set up to acquire raw TCP/IP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN, which was operated as if it was a true environment, but blasted with multiple attacks. There are totally 4 898 431 connections recorded, of which 3 925 650 are attacks. For each TCP/IP connection, 41 various quantitative and qualitative features were extracted [12].

There are total 42 features of each datum. The first three qualitative features are protocol_type, service and flag. Currently, only three protocol types (tcp, udp or icmp) are used. In KDD-99 data, there are 70 different services (such as, http or smtp) and 11 flags (such as, *SF* or *S*2). We map these three qualitative features into quantitative features so as to calculate the similarities of instances. There are also some other qualitative features, such as root_shell ("1" if root shell is obtained; "0" otherwise), logged_in ("1" if successfully logged in; "0" otherwise), land ("1" if connection is from/to the same host/port; "0" otherwise). They are also used as quantitative features here because they are in the form of an integer. The rest of the features except the last one are positive quantitative features, such as src_bytes (number of data bytes from source to destination), urgent (number of urgent packets) and serror_rate (percentage of connections that have "*SYN*" errors). They can be used directly to calculate the similarity of instances. The last feature is the label, which indicates the identification of the instance. If the instance is a normal instance, the label is “normal”; otherwise it is a string of an attack type.

From the KDD-99 dataset, which contains 4 898 431 labeled data, we randomly select 101 000 data for training. Among them, 100 000 are normal and 1 000 are intrusive. From the KDD-99 dataset, we also randomly select 200 000 normal data and 200 000 intrusive data for test. If a datum has been selected for training, it will not be selected for test.

K-means partitioned the training data into a given number of clusters, and labeled each cluster according to the label of the majority data of the cluster. For example, if intrusion data form the largest population in a cluster, the cluster is labeled 'intrusive'; otherwise, it is labeled 'normal'. During the simulation with the testing data, each datum was assigned to its closest centroid, and identified with the same label of the closest centroid. The simulation accuracy of K-means varies with the value of *k*. The best accuracy is obtained when *k* equals to 53.

The *SOM* toolbox from Matlab 6.1 is used to classify the KDD-99 data. The configuration of *SOM* used in our simulation is:

net = newsom(minmax, [2 2, 2 2],'hextop', 'linkdist', 0.9, 1000, 0.05,1) (7)

It means that the network has 2 layers. Each layer has 2×2 dimension; the function is 'hextop'. The distance function is 'linkdist'. The learning rate is 0.9. The number of the ordering phase steps is 1 000. The tuning phase learning rate is 0.05. The tuning phase neighborhood distance is 1.

With different initial values of *k*, K-means+ finally partitioned the 101 000 training data into 53 multi-centred clusters. During the simulation, each test datum is assigned to its closest cluster, and identified according to the label of the cluster.

As illustrated in Table 2, K-means+ has attained *AC* = 96.38%, *TP* = 99.98% and *FP* = 7.22%. This performance is better than the performances of K-means and *SOM*. Almost all the normal data have been correctly classified by K-means+.

	Predictions with clustering algorithms					
	K-means+		**K-means**		***SOM***	
	Normal	*Intrusive*	*Normal*	*Intrusive*	*Normal*	*Intrusive*
Normal	199952	48	191202	8798	171021	28979
Intrusion	14438	185562	82168	117832	36168	163832
Accuracy	96.38%		77.26%		83.71%	
TP	99.98%		95.60%		85.51%	
FP	7.22%		41.08%		18.08%	

Table 2: Simulation results with 400,000 KDD-99 data after the training

K-means+ not only has a better performance than the other clustering methods, but also has a better usability compared to the other classification methods. The user does not need to worry about the initial value of *k* since it rarely affects the result of the classification. For K-means, the user has to choose a value of *k* before clustering. However, the real data are often in an arbitrary distribution and the appropriate value of *k* is hard to obtain. The performances of *SOM* are heavily dependent on the selected topologies and corresponding parameters. To find a proper topology and parameters, the user must have enough knowledge of the complicated topology of the network.

With different initial values of *k* and randomly selected initial cluster centroids, K-means+ partitioned a dataset consisting of 5 000 normal data and 5 000 intrusion data. These data are randomly selected from KDD-99 dataset. Figure 4 illustrates the relationship curve of the initial number of clusters and the final number of clusters. The final number of clusters is almost steadily at 20 when the initial number of clusters varies from 2 to 100. Figure 5 shows the curves of accuracy, true positive rate and false positive rate vs. the initial number of clusters. All of the three curves are nearly horizontal lines, it means that the initial state rarely affect the clustering result.

Moreover, the CPU time of these tests with different initial number of clusters is quite similar to each other. It means that the initial number of clusters also rarely affects the time complexity of K-means+. Therefore, K-means+ does not have the shortcomings of dependency on the initial state.

*** figures 4 and 5 are about here***

The K-means+ algorithm, which is implemented in Java, is run on a personal computer of Dell Dimension 2300 with a Celeron CUP 1.80 GHz and a RAM of 256MB. There are two primary parameters in studying the time complexity of K-means+: the number of initial clusters (*k*) and the size of data (*n*).

The curve in Figure 6 shows that the CPU time of K-means+ does not vary much with different initial *k* when *n* is a constant. Therefore, *k* will not be considered as a parameter of the time complexity of K-means+. When *k* is a constant and *n* is a variable, K-means+ classifies a number of subsets of KDD-99 data. The relationship of CPU time and *n* is shown in Figure 7. The curve is the corresponding fitting polynomial. The least-square approximating polynomial of degree 2 is:

$$Y = 3.76 \times 10 -3\, X^{2} + 18.57X -3.3632 \tag{8}$$

The axis Y is the CPU time of K-means+, and axis X is the number of data for clustering. The coefficient of X^3 in the least-square approximating polynomial of degree 3 is 5.16×10^{-7}, which is too small to consider by comparing with other coefficients; thus, the quadratic fitting polynomial is used to express the relationship between the CPU time. Thus, the time complexity of K-means+ is approximately $O(n^2)$.

In order to avoid the memory overflow when the training data set is very large, K-means+ has to use I/O frequently to read in the training data one by one instead of reading all of them into memory at once. However, the I/O is very expensive time-wisely; it is probably the most expensive portion of K-means+.

4. Discussion and Concluding Remarks

We have introduced a new clustering method based on K-means. The number of clusters of K-means+ is a self-defined variable. To the best of our knowledge, it is the first time that a method using the standard deviation of clusters for splitting and linking clusters according to the statistic nature of data. K-means+ eliminates two shortcomings of K-means: degeneracy and dependency on the initial state.

Empirically, the initial cluster number is no longer critical to the clustering result of K-means+; therefore, it can partition data into an appropriate number of clusters without knowing their distribution. This is the primary advantage of K-means+ over K-means.

As an unsupervised classification method, K-means+ does not need pre-define any parameter or topology. It is simple yet very powerful method that can be implemented easily. The comparative analysis of the test results shows that K-means+ is a robust method for solving classification problems. For intrusion detection, security administrators can use K-means+ to filter out a large amount of normal data before searching the database for intrusions [20]. Thus, the workload of security administrators can be significantly reduced. K-means+ can also be used for classification in many other fields. For medical application, K-means+ can be used to group diseases by their symptoms, this could help to find effective treatments. For e-commerce, K-means+ can be used to group customers according to the customer profile in order to find consumer need and target their advertising advertisements efforts more effectively than sending a bulk of spam to everyone.

Further developments of K-means+ include the following research directions: (i) integrating fuzzy clustering techniques into K-means+ to improve its performance [21]; (ii) building parallel versions of this heuristic to reduce the time complexity; (iii) combining the K-means+ algorithm and meta-heuristics, such as Tabu Search [22] and genetic strategy [23], for solving very large instances; and, (iv) applying enhanced procedure to more real world problems in pattern recognition, medical diagnosis, data mining and e-business.

Acknowledgements

The first author and Dr. Nabil Belacel acknowledge the funding provided by NRC, Saint John, in support of this work. Dr. Ghorbani acknowledges the partial support from the National Science and Engineering Research Council of Canada (NSERC) through grant RGPIN 2277441.

References

[1] R.P. Lippman, An Introduction to Computing with Neural Nets, IEEE ASSP Magazine, April (1987) 4-22.

[2] C. Corte, V. Vapnik, Support Vector Machines, Machine Learning 20 (1995) 273-297.

[3] Cover T, Hart P.G. Nearest neighbor pattern classification, IEEE Trans. Inf. theory IT-13 (1967) 21-27.

[4] J. Quinlan, Induction of Decision Trees, Machine Learning 1 (1986) 81-106.

[5] J. MacQueen, Some methods for classification and analysis of multivariate observations, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol.2, 1967, pp.281-297.

[6] H. Spath, Clustering Analysis Algorithms for Data Reduction and Classification of Objects, Ellis Horwood, Chichester, 1980.

[7] A.K. Jain, R.C. Dubes, Algorithms for cluster Data, Prentice Hall, Inc., 1988.

[8] P. Hansen, N. Mladenović. J-Means: a new local search heuristic for Minimum sum-of-squares clustering, Pattern Recognition 34 (2002) 405-413.

[9] L. Portnoy, E. Eskin, S. J. Stolfo, Intrusion Detection with Unlabeled data Using Clustering, Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001), 2001.

[10] Y. Lin, C. Shiueng, A Genetic Approach to the Automatic Clustering Problem, Pattern Recognition 34 (2001) 415 – 424.

[11] R.A. Fisher, The use of multiple measurements in taxonomic problems, Ann. Eugenics Part II, VII (1936) 179-188.

[12] KDD Cup 1999 Data, University of California, Irvine, October, 1999, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, July, 2002.

[13] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers, New York, 2001.

[14] H.R. Gibson, Elementary Statistics, Wm. C. Brown Publishers, Dubuque, Iowa, 1994.

[15] R.E. Walpole, Elementary Statistical Concepts, second ed., Macmillan, New York, 1983.

[16] T. Kohonen, Self-Organization and Associative Memory, third ed., Springer, 1989.

[17] F. J. Provost, R. Kohavi. Guest Editors' Introduction: On Applied Research in Machine Learning. Machine Learning 30 (1998) 2-3.

[18] R.S. Michalski, I. Bratko, M. Kubat, Machine Learning and Data Mining, John Wiley & Sons, 1998.

[19] R. Duda, P.E. Peter, D.G. Stork, Pattern Classification, second ed., John Wiley & Sons, 2001.

[20] Y. Guan, A.A. Ghorbani, N. Belacel, Automated Clustering Method for Intrusion Detection, Proceedings of Symposium on Advances in Artificial Intelligence, AI' 2003, Halifax, Canada, 2003, pp.616-617.

[21] N. Belacel, P. Hansen, N. Mladenović, Fuzzy J-Means: a new heuristic for fuzzy clustering, Pattern Recognition 35 (2002) 2193-2200.

[22] F. Glover, Future paths for Integer Programming and Links to Artificial Intelligence, Computers and Operations Research 5 (1986) 533-549.

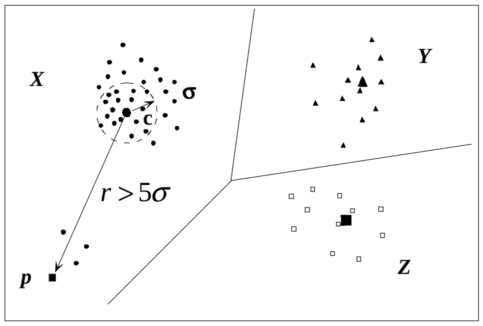
[23] C.A. Murthy, N. Chowdhury, In search of optimal clusters using genetic algorithms, Pattern Recognition Letters 17 (1996) 825-832.

About the Author −Yu Guan received his B.S. degree in 1989 from Mechanical Engineering Department of Tianjin University and M.S. degree in 2003 from Faculty of Computer Science of University of New Brunswick, Fredericton, Canada. He started his doctoral study September 2003 in the same department of University of New Brunswick. He is also working as a research support of the National Research Council of Canada. His current research interests include pattern recognition, data mining and Multiagent systems.

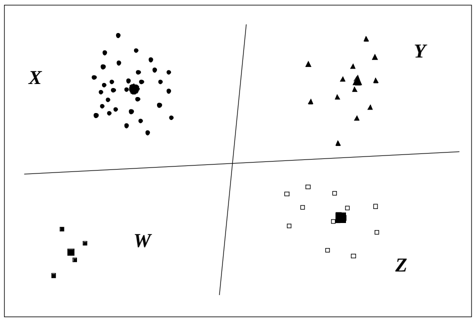
About the Author −Ali Ghorbani has held a variety of positions in academia for the past 25 years. Currently, he is a Professor of Computer Science at the University of New Brunswick, Fredericton, Canada. His current research focus is Web intelligence: adaptive hypermedia, adaptive web systems and autonomous web page synthesis); Multiagent systems: agent-based information systems, trust and reputation in agent societies; and, the application of AI to network security. He has authored more than 75 research papers. Please visit his website (www.cs.unb.ca/~ghorbani) for details.

About the Author −Nabil Belacel is a research officer with the National Research Council of Canada, Institute for Information Technology - e-Business and Adjunct professor in the department of Mathematical Sciences, at the University of New Brunswick-Saint John and Faculty of Computer Science at the University of New Brunswick-Fredericton. He obtained his Bachelor in Operations Research from Algiers University 1991 and Ph.D. degree in Operations Research from Brussels University in 1999. After graduation, Nabil worked as a postdoctoral researcher at GERAD: "Groupe d'études et de recherche en analyse des décisions", Montreal, and as a scientific consultant for the Visual Decision Company, Montreal. Nabil's research interests include operations research, artificial intelligence, fuzzy sets, classification, data mining, clinical decision support systems and bioinformatics. He is well published in a number of these different areas.

Illustrations:



(a) Clusters before splitting



(b) Clusters after splitting

Figure 1: An Example of Splitting Clusters

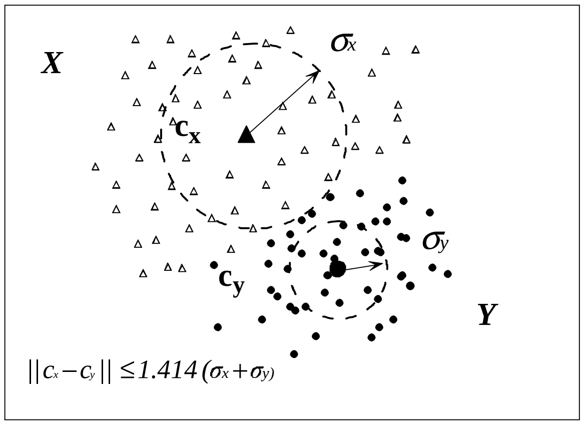


Figure 2: Merging clusters

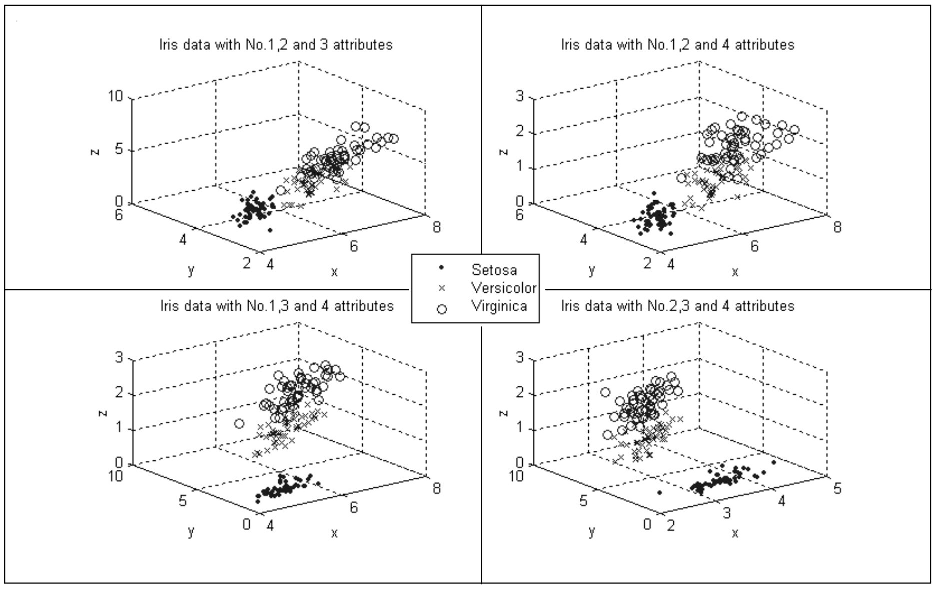


Figure 3: Visualization of the Iris data

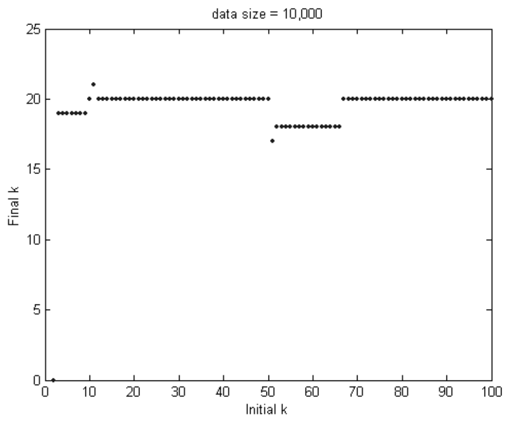


Figure 4: Final number of clusters vs. initial number of clusters

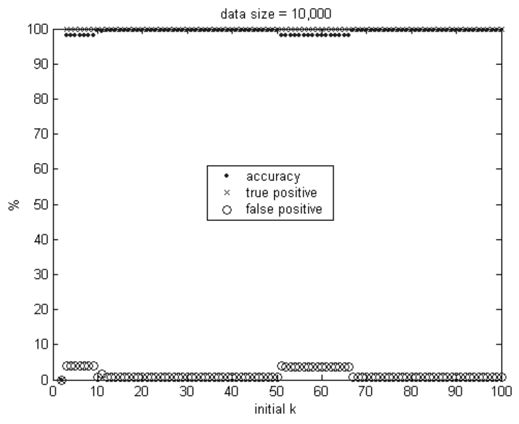


Figure 5: Performance vs. initial number of clusters

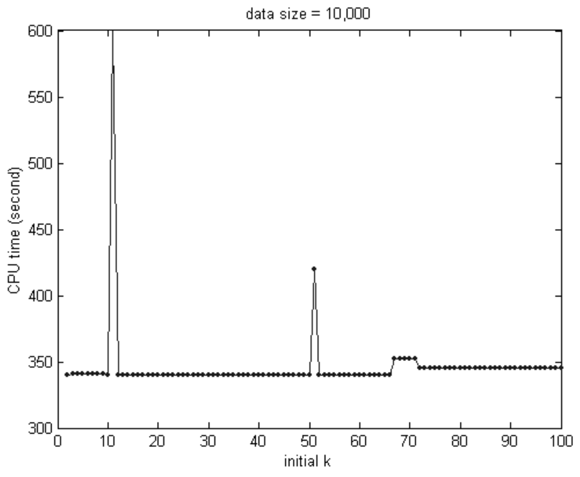


Figure 6: Initial number of clusters vs. CPU time

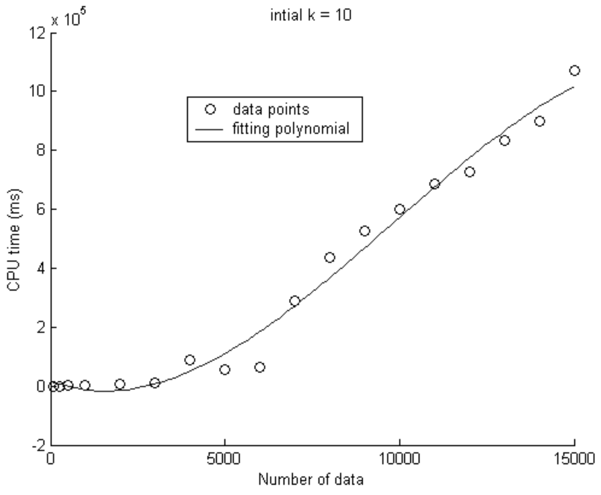


Figure 7: CPU Time vs. the number of data