2012 International Conference on Network and Computational Intelligence (ICNCI 2012)
IPCSIT vol. 46 (2012) © (2012) IACSIT Press, Singapore

A Density-Based Method for Initializing the K-means Clustering Algorithm

Xuanyi Zhang, Qiang Shen [+], Haiyang Gao , Zhijun Zhao and Song Ci

Institute of Acoustics, Chinese Academy of Sciences

Abstract. The k-means algorithm is a commonly used technique in data mining. However, it is sensitive to the initial clustering centers. To reduce the sensitivity and get a better initialization solution, we present a density-based method for initializing the K-means clustering algorithm in this paper. The proposed algorithm employ the density of various areas in data space to choose the initial clustering centers. And the density is estimated by using a space-partitioning data structure called K-d tree. We test our algorithm with several real-world datasets and compare it with some other algorithms. Evaluation demonstrates that our algorithm has a more stable and better performance and especially its running time is considerably lower than other algorithms.

Keywords: Data mining, Clustering, K-means clustering, Initial cluster centers.

1. Introduction

Clustering algorithms are used to divide a given set of data points into several small groups (clusters). Cluster analysis has long played an important role in a wide variety of fields, such as statistics, machine learning, pattern recognition, decision support, knowledge discovery and bioinformatics.

To divide the data sets into several small clusters properly, a lot of algorithms have been proposed in the literature. K-means algorithm is one of the most commonly and widely used methodologies for researchers and is one of the top 10 data mining algorithms identified by the IEEE International Conference on Data Mining [1].

Although K-means has been widely used in many areas, it still has three main shortcomings:

(1) It assumes that the number of clusters K is previously known and this is not always true when it is used in a real application.

(2) It is especially sensitive to the initial positions of the cluster centers. Different initial partitions will result in the different final clusters with various qualities.

(3) It prefers clusters with the approximately equal size and does not work well with non-globular clusters.

K-means algorithm does not guarantee a unique clustering since there are different results caused by different initial centers. Actually, the problem of sensitivity caused by the initial cluster centers (namely, seed) exists for both the K-means algorithm and many other clustering algorithms, and their inherent behaviors lead to a local minima [2]. The K-means algorithm gives better result only when the initial partition is closed to the real situation [3]. To achieve a better initialization, many other techniques have been proposed. In this paper, we propose a density-based method to improve the quality of the initial cluster centers.

[+] Corresponding author. Tel.: + 13810724028.
E-mail address: shenq@hpnl.ac.cn.

The rest of this paper is organized as follows. Section 2 presents the related work of the initializing algorithm for the K-means clustering algorithm. Section 3 introduces the K-means algorithm briefly. The proposed center algorithm based on the density is discussed in details in section 4. Some experimental results and analysis for the algorithm are shown in the section 5 and some conclusions are drawn in the last section.

2. Related Work

The performance of the K-means algorithm depends on the positions of initial cluster centers. Due to the importance of the initialization for K-means, many solutions for choosing better initial cluster centers have been proposed in the literatures.

Tou and Gonzales introduce the Simple Cluster Seeking (SCS) method to initialize the K-means algorithm in [4]. Let the first seed equal the first instance in the database, i.e. $c_1 = x_1$. Calculate the distance between c_1 and the second point in the database, if it is greater than a threshold ρ then the point will be selected as the second seed c_2. Otherwise move to the next instance until the second seed c_2 is chosen. If $(j-1)$ seeds have been chosen, $(j = 3,...,n)$, calculate the distance between the following instance and all existing seeds. If all these distances are greater than the threshold ρ, the corresponding instance is selected as the j-th seed. The algorithm is finished until K seeds are chosen. The quality of this algorithm is dependent on the presentation order of the points in the database and the value of ρ.

McQueen introduces a learning strategy to initialize the K-means algorithm in [5]. The first K points in the database are chosen as the initial seeds. Then following the instance order, the rest data points in the database are assigned to the cluster with the nearest seed. After each assignment, update the center of that cluster to be equal to the mean of all points assigned to it. Repeat the process until all data points are assigned to a cluster.

In [6], Kaufman and Rousseeuw select the first seed as the most centrally located instance. Then they examine which instance of the rest points in the database can produce the greatest reduction in square-error distortion, and it will be chosen as the next seed. The next seed will be chosen in the same way. This process will continue until K seeds are chosen. The notable drawback of this algorithm is the considerable amount of computation. Given n input samples, at least $n \cdot (n-1)$ times of distance calculation are required. This algorithm will consume much more time than K-means itself when n is large [7].

Katsavounidis et al. utilize the sorted pairwise distances for initialization which has been termed as the KKZ algorithm in [8]. This algorithm chooses the vector with maximal norm as the first seed, i.e. $c_1 = x_i$ where $i = \mathrm{argmax}_m \left(\left|x_m\right|\right), \forall 1 \le m \le n$, and n denotes the number of all vectors. Calculate the distance between all vectors and the first seed, then the vector with the largest distance will be chosen as the second seed c_2. Generally, assume there are t, $(t = 2,3,...,k-1)$, seeds which have already been chosen, then compute the distance between any unchosen vectors and all existing seeds, and set $c_{t+1} = x_i$ where $i = \mathrm{argmax}_m \left(\mathrm{md}\left(x_m\right)\right)$. The definition of $\mathrm{md}\left(x_m\right)$ is as follows:

$$\mathrm{md}\left(x_m\right) = \min\left(\mathrm{dist}\left(x_m, c_j\right)\right), \forall 1 \le j \le t.$$

It denotes the distance between xm and its nearest center. This procedure ends until K seeds are chosen.

Bradley and Fayyad present a refinement algorithm for initializing the K-means algorithm in [9]. They firstly utilize K-means for j times to get j random sub-samples from the original data. The sub-samples are clustered with the proviso that empty clusters at termination will have their initial centers re-assigned and the sub-samples will be re-clustered. The sets $CM_i, i = 1,...,J$, which are these clustering solutions over the sub-samples, form the set CM. Then cluster CM via K-means initialized with CM_i produces a solution FM_i. The FM_i having minimal distortion over the set CM are chosen as the final centers.

Redmond and Heneghan introduce a method for initializing the K-means algorithm using k-d tree [10]. They divide the database into many buckets using k-d tree and set the bucket centers as the candidate seeds. Then the center of the bucket which has the maximum density is chosen as the first seed. The center which has the maximum value obtained by multiplying the distance between it and the first seed by its density is chosen as the second one. The third seed is chosen by computing the distance between each center and its nearest seed multiplied by the density of the bucket itself. Repeat this method until the number of seeds is up to K. Until now, the first initial solution has been finished. Then sort the centers by their density and run the method again on 80% of densest centers and get the second result. The final initial solution is the one which minimizes (3). Inspired by this algorithm, our algorithm uses a filtering method and the real density information to get a better performance, which is described in more detail in section 4.

3. K-means algorithm

In this paper, the numbers of dimensions, centers and instances are denoted by b, K, n, respectively. There are n points in the database, namely, $(x_1, x_2, ..., x_n)$, and each point x_i is associated with its d coordinates, namely, $(x_{i1}, x_{i2}, ..., x_{id})$.

K-means-like algorithms have been proposed by several researchers across different disciplines [1]. Therefore, to clarify the version we used in this paper, we will briefly introduce it. The K-means algorithm operates on a set of d -dimensional vectors, $X = \{i = 1, ..., n\}$, and aims to divide them into K clusters, i.e., $C_1, C_2, ..., C_K$. Firstly, it is initialized by randomly choosing K points from X as the initial cluster centers, i.e., $c_1, c_2, ..., c_k$. Then, for each of the remaining points, a point x_i is assigned to the cluster C_j while the distance between x_i and the cluster center c_j is the smallest than others. The distance function is as follows:

$$dist\left(x_i, c_j\right) = \sum_{m=1}^{d} \left(x_i m - c_j m\right) \tag{1}$$

In this way, the data points are divided into K clusters, $C_1, C_2, ..., C_K$. Then calculate the new c_j center of each cluster according to the following function:

$$c_j = \frac{\sum_{x_i \in C_j} x_i}{\left|C_j\right|} \tag{2}$$

Where x_i is a vector representing the i -th data point in the cluster C_j. Meanwhile, c_j is the geometric center of the C_j and $\left|C_j\right|$ denotes the number of data instances in C_j. Then update each cluster center with new center until there is no change regarding the cluster centers.

In order to evaluate the quality of each cluster center, the square-error critirion is used, as described as follows:

$$E = \sum_{j=1}^{K} \sum_{x_i \in C_j} \text{dist}\left(x_i, c_j\right)^2 \tag{3}$$

The function *dist* returns the Euclidean distance between two vectors as shown in (1). The smaller the value of E is, the better result K-means algorithm gets.

To sum up, K-means clustering algorithm aims to divide the n input data points $(x_1, x_2, ..., x_n)$ into K disjoint clusters $(C_1, C_2, ..., C_K)$ to minimize the sum of square-error, as shown in Algorithm 1.

Algorithm. 1: K-means algorithm

```
1: Procedure KMEANS($K$)
2:     Initialize $K$ centers $(c_1, c_2, ..., c_k)$ of $K$ clusters $(C_1, C_2, ..., C_K)$
3:     Repeat
4:         for $i \leftarrow 1, n$ do
5:             if $\text{dist}(x_i, c_j *) \le \text{dist}(x_i, c_j), \forall j \in \{1, ..., K\}$ then
6:                 Assign $x_i$ to the cluster $C_j *$
7:             end if
8:         end for
9:         for $j \leftarrow 1, K$ do
10:            Update the center of cluster $C_j$ to be the mean of all data points currently in cluster $C_j$
11:        end for
12:        Compute the square-error function: $E = \sum_{j=1}^{K} \sum_{x_i \in C_j} \text{dist}(x_i, c_j)^2$
13:    until the cluster membership no longer changes
14: end Procedure
```

4. The density based algorithm

The main purpose of our algorithm is to divide the data points into K clusters by considering the distribution features of data points, such as density and distance. To estimate the density distribution of the data space, a data structure called k-d tree is used. If we simply divide the data space into cubes by dividing each dimension into t divisions, there will be t^d cubes. And we must estimate the density of each cube by counting the number of points it contained. However, when t and d are large values, the number of t^d will become unacceptable. Hence, the density estimation method based on k-d tree is used instead.

The data structure, k-d tree, is a space-partitioning data structure for organizing points from a k-dimensional space. It is a binary tree. Each node in this tree represents all points contained in a hyper-rectangle h. The hyper-rectangle h is stored at the node as two d-length boundary vectors $V_{\max}$ and $V_{\min}$. These two vectors are defined by the maximum and minimum coordinate values of the data points for the node in each dimension. The node also stores the number and center of all points in h. Each non-leaf node has a split dimension M_{split} and a split value V_{split}. According to M_{split} and V_{split}, a node h can be split into two child nodes. And the left child (l) represents a hyper-rectangle (h_l) that contains all points in h which have their M_{split}-th coordinate smaller than V_{split}. And the right child is opposed to the left one. In our algorithm, M_{split} is the longest dimension of the node and V_{split} is the median value of the coordinates in M_{split}-th dimension. The root of the tree denotes the hyper-rectangle which encompasses all the point in the database. At the beginning, we will split the root node into two child nodes. This splitting process is recursively repeated on each node until a leaf node is created. A node is a leaf node only when it fulfills a certain condition, such as, the number of points included by the node is less than or equal to M_{number}, or its volume is no more than the M_{volume}, where M_{number} and M_{volume} denote the maximum number of points a node contains and the maximum volume of a node, respectively. We can assign predetermined numbers to M_{number} and M_{volume} in order to obtaining enough leaf nodes.

If when to stop splitting process is based on M_{volume}, each leaf node will have similar volume but various number of points. Alternatively, if it is decided by the value of M_{number}, each leaf node will have roughly the same number of points but different volumes. In either case, the densities for leaf nodes can reflect the actual area density distribution. Hence we can estimate the density of different regions in the data space simply and effectively. The estimation method based on the k-d tree only divides the data space into $\left(A_{volume} \div M_{volume}\right)$ or $\left(n \div M_{number}\right)$ cubes which is more smaller than t^d, where A_{volume} denotes the global volume of the entire database and n denotes the number of instances. Hence, this density estimation method can save a lot of time.

Suppose we have already divided the data space into t leaf nodes using k-d tree, then we should calculate the density of each leaf node. Let V_i denote the volume of leaf node $L_i\left(1 \le i \le t\right)$. The density of each leaf node can be calculated as follows:

$$\rho_i = N_i / V_i \tag{4}$$

In (4), N_i denotes the number of points contained in the leaf node L_i. The volume of leaf node is calculated according to the following equation:

$$V = \prod_{i=1}^{d}\left(x_{V_{\text{maxi}}} - x_{V_{\text{mini}}}\right) \tag{5}$$

The $x_{V_{\text{maxi}}}$ and $x_{V_{\text{mini}}}$ denote the i-th coordinate value of $V_{\max}$ and $V_{\min}$, separately. If $\left(x_{V_{\text{maxi}}} - x_{V_{\text{mini}}}\right) = 0$, $1 \le j \le d$, it will be replaced by the geometric mean of the nonzero dimensions. To choose the initial centers properly, we associate the density of each leaf node, ρ_i, with a point $\left(s_i\right)$ in the node. We choose s_i to be the mean of the data points contained within the leaf node. Hence there is a list of t points $\left(s_1, ..., s_t\right)$ and the corresponding densities of data are represented as $\left(\rho_1, ..., \rho_t\right)$. Then K centers will be chosen from the t points, and those centers are separated by a reasonable distance and have a large density.

The first seed c_1 is chosen to be the point with largest density:

$$c_1 = s_{\text{argmaxi}}\left(\rho_i\right) \tag{6}$$

The second candidate seed is chosen as the point m_i with the maximum value of f_i (7). The value of f_i is the product of the density of s_i and the distance between the first seed and s_i , as shown in (7).

$$f_i = \text{dist}\left(c_1, s_i\right) \cdot \rho_i \tag{7}$$

This idea is that the further away a point is from an existing seed and the larger density it has, the more likely it will be chosen as a seed. There are some shortcomings in choosing seeds only based on the product of the distance and the density. In [10], Redmond and Heneghan propose an improvement by using the rank of the estimated density instead of the actual value. But this rank doesn't reflect the real density situation information. Furthermore, those seeds chosen by their method are not isolated enough and the placements of several seed locations are still near the denser clusters [11]. Hence, we propose the filtering method to improve the result. The filtering method is used to determine the candidate m_i whether to be the second seed or not. Firstly, we introduce a definition used in this paper, and the definition is similar to another clustering algorithm call DBSCAN [12].

Definition 1: (density-reachable) A point s_i is density-reachable from a point s_j if each point s_k between them can satisfy the following condition: $\rho_k \geq \left[\min\left(\rho_i, \rho_j\right) \cdot W_{weight}\right]$. And s_k between s_i and s_j means:

$$\forall 1 \leq h \leq d, \min\left(x_{s_{ih}}, x_{s_{jh}}\right) \leq x_{s_{kh}} \leq \max\left(x_{s_{ih}}, x_{s_{jh}}\right)$$

Where $x_{s_{kh}}$ denotes the h-th coordinate value of s_k. When m_i is density-reachable from c_1, we filter m_i and choose another candidate, otherwise, m_i will be chosen as the second seed.

If h seeds have been chosen, the point m_i will be chosen as the next seed c_{h+1}, while m_i the maximum value of (8) and is not density-reachable from existing seeds. In this step, f_i has been changed to the following description:

$$f_i = \left(\min_{k=1,\dots,h}\left(\text{dist}\left(c_k, m_i\right)\right)\right) \cdot \rho_i \tag{8}$$

When K seeds have been chosen, the algorithm will be terminated automatically. Our algorithm is summarized in Algorithm 2.

Algorithm. 2: Our algorithm

```
1: Procedure INITIALIZATION_ALGORITHM ( $K$ )
2:     Create a k-d tree for the database
3:     for $i \leftarrow 1, t$ do
4:             Calculate the density $\rho_j$ of the leaf node $L_i$
5:             Calculate the mean $m_j$ of the leaf node $L_i$
6:     end for
7:     $\rho_{c_1} \leftarrow 0$
8:     for $i \leftarrow 1, t$ do
9:             if $\rho_{c_1} < \rho_{m_i}$ then
10:                    $c_1 \leftarrow m_i$
11:            end if
12:    end for
13:    $h < 1$
14:    repeat
15:            for $i \leftarrow 1, t$ do
16:                    if $m \notin \{c_1, ..., c_h\}$ then
17:                            $f_i \leftarrow \left(\min_{k \leftarrow 1,\dots,h}\left(\text{dist}\left(c_k, m_i\right)\right)\right) \cdot \rho_i$
18:                            $\triangleright$ $f_{\text{maxidx}}$ is initialized to be the minimum
19:                            if $f_i > f_{\text{manidx}}$ then
20:                                    $\text{maxidx} \leftarrow i$
21:                            end if
22:                    end if
```

```
23:             if $m_{\text{maxidx}}$ is not density-reachable from $\{c_1,\ldots,c_h\}$ then
24:                 $c_h \leftarrow m_{\text{maxidx}}$
25:                 $h \leftarrow h+1$
26:                 break
27:             end if
28:         end for
29:     until $h = K$
30: end Procedure
```

5. Experimental result

To evaluate the practical applicability of our algorithm, its performance is tested on a number of real-world data sets, such as the pen-based recognition of digit handwritten database and the statlog data set.

The digit handwritten database (TDHD) is created by collecting 250 samples from 44 writers [13]. The database creaters use a WACOM PL-100V pressure sensitive tablet with an integrated LCD display and a cordless stylus. The input and display areas are located in the same place. Attached to the serial port of an Intel 486 based PC, it allows us to collect handwritten samples. The tablet sends x and y coordinates and pressure values of the pen at the fixed time intervals (sampling rate) of 100 milliseconds. There are overall 10,992 instances of 16 attributes in the database and 10 clusters.

The statlog data set (TSDS) is an image segmentation database [13]. It consists of 2310 instances and 19 attributes. The instances are drawn randomly from a database of 7 outdoor images which are segmented to create a classification for every pixel. So there are 7 clusters. Our experimental computer has the following configurations: Windows 7 OS with Pentium Dual-Core CPU 3.20GHz and 2G RAM, and the algorithm was implemented by the Java language.

In [2], J.M. Pena et al. compare four initialization methods: RANDOM Approach, Forgy Approach [14], Macqueen Approach [5] and Kaufman Approach [6]. And they conclude that RANDOM and Kaufman's initialization methods outperform the other two methods compared in their paper with respecting to the effectiveness and the robustness of the K-means algorithm when these two initialization methods are used. The RANDOM method is to partition the database into a partition of K clusters at random.

In [15], Ranjan Maitra et al. compare eleven initialization methods and the initialization methods proposed by [16], [9] and [17] are found to be the top performers both in terms of minimizing the sum-of-squares and in the best recovery of the true groupings.

Hence we choose the RANDOM method, Bradley and Fayyads method [9] for comparing. And we also compare Redmond's method and the traditional K-means initialization method with ours. The results of square-error function are shown in Table 1 and the results of running time are shown in Table 2. We run the traditional K-means initialization method for 25 times, and 'KM(mean)' and 'KM(best)' denote the mean and the best one of the 25 results, individually. We run Bradley and Fayyads method [9] for 15 times and 'Bradley(mean)' in tables means the mean of results and 'Bradley(best)' means the best result of the 15 results. 'Redmond' denotes Redmond's method and 'TDBA' (The Density-Based Algorithm) denotes the algorithm proposed in this paper.

For statlog data set, Bradley's algorithm gets the best square-error result. But before we get the result, the method has been performed fifteen times and this value is only 3% smaller than that of our method. For the most important, the running time of Bradley's algorithm is 13.6 times of ours. For the digit handwritten database, our algorithm not only costs less running time but also obtains almost the best value of square-error function.

The traditional K-means initialization method is widely used in many fields as it costs less runtime and it can be implemented easily. However, it takes more time and gets higher value of square-error function than ours.

We have compared our method with other four initialization techniques to obtain a better experimental result. We conclude that our initialization method for the K-means algorithm is fast and efficient.

Table. 1: Results of square-error function

	Bradley (best)	Bradley (mean)	KM (mean)	KM (best)	Redmond	RANDOM	TDBA
TDHD	49372752.877	49990985.956	51943668.455	49365672.589	49302206.233	163398346.501	49301514.883
TSDS	13427996.537	14104189.381	14194951.172	13929188.088	14432455.351	51896440.561	13950010.091

Table. 2: Results of running time

	Bradley(best)	Bradley (mean)	KM (mean)	KM (best)	Redmond	RANDOM	TDBA
TDHD	6911	7531.727	3491.7	4663	4927	117.3	3743
TSDS	6039	6486.75	3906.3	3216	565	68	443

6. Conclusion

In this paper, an initialization algorithm for K-means clustering to obtain the initial value is proposed. This algorithm use k-d tree to partition the given database into many parts and estimate the density of each part and choose K initialization seeds according to the density and distance information.

We conduct a performance evaluation on real-world data sets and the results of these experiments demonstrate that our algorithm is more effective for discovering clusters than other well-known algorithms.

Future work could consider the following issues. Firstly, the number of point in a node should be predetermined. Secondly, the adaptability of our algorithm to the high dimensional data set should be investigated.

7. Acknowledgements

This work is supported by National S&T Major Special Project under Grant 2011ZX03005-002, 2012ZX03005007-002, China Postdoctoral Science Foundation funded project under Grant 20100470568 and K. C. Wong education foundation, Hong Kong.

8. References

[1] X. Wu, V. Kumar et al.. Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, vol. 14, pp. 1–37, Dec. 2007.

[2] J. M. Pena, J. A. Lozano, and P. Larranaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, vol. 20, pp. 1027–1040, 1999.

[3] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[4] J. T. Tou and R. C. Gonzalez. *Pattern recognition principles*. Image Rochester NY, p. 377, 1974.

[5] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1. 1967, pp. 281–297.

[6] L. Kaufman and P. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. Wiley, 2005.

[7] J. He, M. Lan, C.-L. Tan, S.-Y. Sung, and H.-B. Low. Initialization of cluster refinement algorithms: a review and comparative study. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 1, Jul. 2004, pp. 297–302.

[8] I. Katsavounidis, C. C. Jay Kuo, and Z. Zhang. A new initialization technique for generalized lloyd iteration. *IEEE Signal Processing Letters*, vol. 1, pp. 144–146, Oct. 1994.

[9] P. S. Bradley and U. M. Fayyad. Refining initial points for k-means clustering. In *Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 1998, pp. 91–99.

[10] S. Redmond and C. Heneghan. A method for initialising the k-means clustering algorithm using kd-trees. *Pattern Recognition Letters*, vol. 28, pp. 965–973, 2007.

[11] W. Y.n, LI C.s.. New initialization method for cluster center. *Control Theory Applications*, pp. 1435–1440, 2010.

[12] M. Ester, H. peter Kriegel, J. S, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of 2nd International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996, pp. 226–231.

[13] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[14] E. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, vol. 21, pp. 768–780, 1965.

[15] A. D. Peterson, A. P. Ghosh, and R. Maitra. A systematic evaluation of different methods for initializing the k-means clustering algorithm. *Knowledge Creation Diffusion Utilization*, pp. 1–11, 2010.

[16] G. W. Milligan and P. D. Isaac. The validation of four ultrametric clustering algorithms. *Pattern Recognition*, vol. 12, pp. 41–50, 1980.

[17] B. Mirkin. *Clustering for Data Mining: A Data Recovery Approach (Chapman & Hall/CRC Computer Science & Data Analysis).* 1st ed. Chapman and Hall/CRC, Apr. 2005.