A Robust and Scalable Clustering Algorithm for Mixed Type Attributes in Large Database Environment

Tom Chiu

American Century Investments
4500 Main Street
Kansas City, MO 64111-7709
tom_chiu@americancentury.com

DongPing Fang, John Chen, Yao Wang, Christopher Jeris
SPSS Inc.
11th Fl., 233 S. Wacker Drive
Chicago, IL 60606-6307
{dfang, chen, ywang, cjeris}@spss.com

ABSTRACT

Clustering is a widely used technique in data mining applications to discover patterns in the underlying data. Most traditional clustering algorithms are limited to handling datasets that contain either continuous or categorical attributes. However, datasets with mixed types of attributes are common in real life data mining problems. In this paper, we propose a distance measure that enables clustering data with both continuous and categorical attributes. This distance measure is derived from a probabilistic model that the distance between two clusters is equivalent to the decrease in log-likelihood function as a result of merging. Calculation of this measure is memory efficient as it depends only on the merging cluster pair and not on all the other clusters. Zhang et al [8] proposed a clustering method named BIRCH that is especially suitable for very large datasets. We develop a clustering algorithm using our distance measure based on the framework of BIRCH. Similar to BIRCH, our algorithm first performs a pre-clustering step by scanning the entire dataset and storing the dense regions of data records in terms of summary statistics. A hierarchical clustering algorithm is then applied to cluster the dense regions. Apart from the ability of handling mixed type of attributes, our algorithm differs from BIRCH in that we add a procedure that enables the algorithm to automatically determine the appropriate number of clusters and a new strategy of assigning cluster membership to noisy data. For data with mixed type of attributes, our experimental results confirm that the algorithm not only generates better quality clusters than the traditional *k*-means algorithms, but also exhibits good scalability properties and is able to identify the underlying number of clusters in the data correctly. The algorithm is implemented in the commercial data mining tool Clementine 6.0 which supports the PMML standard of data mining model deployment.

Keywords

Mixed type of attributes, clustering, log-likelihood, number of clusters, noisy data.

KDD 01 San Francisco CA USA

1. INTRODUCTION

Clustering is widely used by retail and consumer product companies who need to learn more about their customers. Customers are grouped into clusters by their buying habits, gender, age, income level, etc., marketing promotion programs can then be tailored to customers in different clusters. It is common in real life that attributes are of mixed types; attributes such as age and number of years in school are of continuous scale while attributes such as occupation of household head and race are categorical with finite numbers of possible values. Most existing clustering algorithms are designed to find clusters based on an assumption that all attributes are either continuous or categorical. One of the most widely used clustering algorithms is the *k*-means algorithm [6]. The *k*-means clustering and its variants work reasonably well when all attributes are continuous. For a categorical attribute, each possible value is translated into a dummy attribute; the set of dummy attributes replaces the original categorical attribute. The dummy attributes are considered continuous and the *k*-means algorithm is applied as usual. Guha, Rastogi and Shim [4] illustrated that the Euclidean distance can be a poor measure of similarity under this situation by giving a simple example. Ganti, Gehrke & Ramakrishnan [3] introduced a clustering algorithm for categorical attributes. The algorithm requires only two scans and hence is fast and scalable. Moreover, it can find clusters in subsets of all attributes and can thus perform subspace clustering. Huang [5] proposed a new distance measure for categorical attributes based on the total mismatches of the categorical attributes of two data records in the *k*-modes algorithm. For mixed type of attributes, Huang used a weighted sum of Euclidean distance for continuous attributes and the proposed distance measure for categorical attributes. However, the weights have to be determined a priori. Improper weights may result in biased treatment of different attribute types. Guha, Rastogi and Shim [4] proposed a hierarchical clustering method named ROCK that uses a new concept called *link* to measure the similarity between a pair of data records with categorical attributes. Banfield and Raftery [1] introduced a model-based distance measure for data with continuous attributes. We extend the model-based distance measure and propose a new distance measure that enables clustering of data with mixed types of attributes. This measure is derived from a probability model and is equivalent to the decrease in log-likelihood resulting from merging two clusters. In our proposed

distance measure, only summary statistics are needed for distance calculation. Therefore, we remove the requirement that data fit into the main memory of a computer. Essentially, only one data pass is needed to obtain the clustering results. Another data pass can be used to obtain the cluster membership of each individual record. During cluster membership assignment, we employ a novel volume measure to identify outlier or noise data records.

A desirable feature in clustering is to determine the number of clusters automatically. Traditionally clustering algorithms do not address this issue directly; rather it is treated as a separate problem from clustering. We propose a new algorithm to automatically determine the best number of clusters. We show that our algorithm works very well with our clustering algorithm.

The widely used clustering algorithms such as *k*-means are *in-memory* algorithms; data being processed are kept in the main memory of a computer and scanned repeatedly. However, when the entire dataset cannot be loaded into the main memory at once, repeated disk paging causes dramatic slowdown. Therefore, it is important that the algorithm is scalable, meaning that the runtime of the algorithm increases linearly with the number of records in the dataset given fixed amount of main memory. In this paper, we use a modified two-phase clustering technique pioneered by Zhang, Ramakrishnan and Livny [8] in their BIRCH clustering method. BIRCH performs a pre-clustering step by scanning the entire dataset and storing the dense regions of data records in summary statistics called *cluster features* which are stored in memory in a data structure called *CF-tree*, a hierarchical clustering algorithm is then applied to cluster the set of dense regions. Since the set of dense regions is much smaller than the original dataset, the traditional hierarchical clustering can perform very efficiently. Euclidean or Manhattan distance measure is used in BIRCH, hence the method is suitable when all attributes are continuous. We use the summary statistics required by our proposed distance measure, build a CF-tree similar to that of BIRCH in the pre-clustering step, and apply our algorithm to automatically determine the best number of clusters during the final hierarchical clustering phase. As in BIRCH, the pre-clustering step is incremental in the sense that it only scans the database once and the model could be updated without passing the entire dataset again.

We show experimentally that our clustering algorithm exhibits linear scalability. We also compare the speed and accuracy with *k*-means algorithms and find that our algorithm gives up to 30% more accurate results compared to the best results from varies *k*-means implementations. Our algorithm is available commercially in the Clementine 6.0 data mining tool. The result of the clustering model can be deployed in the PMML data mining standard.

2. BACKGROUND

We begin by defining the log-likelihood function that is used to derive the distance measure in this paper.

2.1 Log-likelihood Function

Let $\{x_i: i = 1, \ldots, N_j\}$ be the set of K-dimensional data records in the j-th cluster $C_j, j = 1, \ldots, J$, and let $p(x|\theta_j)$ be the probability density function of x in cluster C_j, where θ_j is the vector of model parameters. The *classification likelihood function* is the joint probability density of all the data records $\{x_i: i = 1, \ldots, N_j, j = 1, \ldots, J\}$. Logarithm of the classification likelihood takes the form

$$l = \sum_{j=1}^{J}\sum_{i\in I_j} \log p(x_i|\theta_j) = \sum_{j=1}^{J} l_{C_j} \quad (1)$$

where $l_{C_j} = \sum_{i\in I_j} \log p(x_i|\theta_j)$ denote the contribution of cluster C_j to the log-likelihood and $I_j = \{i: x_i \in C_j\}$ is the index set of C_j. Given the set of data records x_i's, the *maximum likelihood* (m.l.) estimates $\hat{\theta}_j$ of θ_j maximizes the log-likelihood l. The value of l evaluated at $\theta_j = \hat{\theta}_j$ gives the highest probability (in logarithmic scale) that the data records x_i's are "most likely to occur" under the given probability model, that is, the given density function $p(x|\theta_j)$.

Without loss of generality, write $x_i = (x_{Ai1},\ldots,x_{AiK_A}, x_{Bi1},\ldots,x_{BiK_B})$ in cluster C_j, assume that x_{Aik} is independent normal distributed with mean μ_{jk} and variance σ_{jk}^2 and that x_{Bik} has L_k categories and the vector of categories of x_{Bik} is independent multinomial distributed with probability vector $(q_{jk1}, \ldots, q_{jkL_k})$ where q_{jkl} is the probability of category l, $0 \le q_{jkl} \le 1$ and $\sum_l q_{jkl} = 1$. The estimates of μ_{jk} and σ_{jk}^2 and q_{jkl}'s which maximize the log-likelihood function are the maximum likelihood estimates.

It can be shown that the maximum likelihood estimates of μ_{jk} and σ_{jk}^2 are the sample average $\hat{\mu}_{jk} = \sum_{i\in I_j} x_{Aik}/N_j$ and the sample variance $\hat{\sigma}_{jk}^2 = \sum_{i\in I_j}(x_{Aik} - \hat{\mu}_{jk})^2/N_j$ of the data records x_i in C_j. Moreover, let N_{jkl} be the number of data records in C_j whose k-th categorical attribute takes the l-th category, $l = 1, \ldots, L_k$, the maximum likelihood estimate of q_{jkl} is the proportion $\hat{q}_{jkl} = N_{jkl}/N_j$. Hence the maximum value of the log-likelihood function in (1) is $\hat{l}_{C_j} = \sum_{j=1}^{J}(\hat{l}_{AC_j} + \hat{l}_{BC_j})$ where $\hat{l}_{AC_j}$ and $\hat{l}_{BC_j}$ correspond to the contributions of continuous and categorical attributes respectively. It can be shown that

$$\hat{l}_{AC_j} = -\frac{1}{2}N_j[K_A\{\log(2\pi)+1\} + \sum_{k=1}^{K_A}\log(\hat{\sigma}_{jk}^2)] \quad (2)$$

and

$$\hat{l}_{BC_j} = -N\sum_{k=1}^{K_B}\hat{E}_{jk} \quad (3)$$

in which $\hat{E}_{jk} = -\sum_{l=1}^{L_k} \hat{q}_{jkl} \log \hat{q}_{jkl}$ is the entropy of the k-th categorical attribute in cluster C_j.

2.2 Distance Measure

We derive a distance measure between two clusters based on the decrease in log-likelihood as a result of merging them together. Suppose at present that the number of clusters is J and a cluster C_j is to be merged with another cluster C_s, $s \neq j$, $s, j = 1, \ldots, J$. The pair of merging clusters C_j and C_s is replaced by a new cluster $C_{<j,s>}$ that consists of data records originally belong to C_j or C_s. The value of the new log-likelihood $\hat{l}_{new}$ estimate is

$$\hat{l}_{new} = \sum_{j' \neq j,s}^{J} \hat{l}_{C_{j'}} + \hat{l}_{C_{<j,s>}} \tag{4}$$

where $\hat{l}_{C_{<j,s>}}$ is the value of contribution of merged cluster $C_{<j,s>}$. Since the values of contributions of all other clusters remain unchanged, the decrease in log-likelihood after merging is $\hat{l} - \hat{l}_{new} = \hat{l}_{C_j} + \hat{l}_{C_s} - \hat{l}_{C_{<j,s>}}$. After simplification, the decrease in log-likelihood can be written as

$$\hat{l} - \hat{l}_{new} = \xi_j + \xi_s - \xi_{<j,s>} \tag{5}$$

where

$$\xi_v = -N_v \{ \sum_{k=1}^{K_A} \frac{1}{2} \log(\hat{\sigma}_{vk}^2) + \sum_{k=1}^{K_B} \hat{E}_{vk} \} \tag{6}$$

for $v = s, j$ and $\xi_{<j,s>}$ is defined similarly. In view of the right hand side of (5), some degenerating situations need to be resolved before using it as a distance measure; these include the situation when some clusters are singleton and when the values of some attributes are constant. To deal with these degenerating situations, take the convention that $x\log(x) = 0$ if $x = 0$ and replace ξ 's with

$$\tilde{\xi}_v = -N_v \{ \frac{1}{2} \sum_{k=1}^{K_A} \log(\hat{\sigma}_{vk}^2 + \Delta_k) + \sum_{k=1}^{K_B} \hat{E}_{vk} \} \tag{7}$$

for $v = s, j$, and $\tilde{\xi}_{<j,s>}$ is defined similarly, where $\Delta_k > 0$ is a positive scalar. The *distance* $d(j, s)$ between two clusters C_j and C_s is defined as

$$d(j, s) = \tilde{\xi}_j + \tilde{\xi}_s - \tilde{\xi}_{<j,s>}. \tag{8}$$

This distance measure is non-negative and symmetric.

3. The Algorithm

Our algorithm consists of two steps. In the first step, all data records are scanned and the dense regions of the records are stored in summary statistics. In the second step, each dense region is treated as an individual point and a hierarchical clustering algorithm is applied to cluster the dense regions. Because the number of dense regions is far less than the total number of data records in the original dataset, the hierarchical clustering in the second step is very efficient.

3.1 Step One: Pre-Clustering

We follow the notations in BIRCH and call the collection of statistics that summarizes the characteristics of a dense region the *cluster feature*. In our algorithm, the cluster feature CF_j of a cluster C_j is

$$CF_j = \{N_j, s_{Aj}, s_{Aj}^2, N_{Bj}\}, \tag{9}$$

where N_j is the number of data records in C_j, s_{Aj} is the sum of continuous attributes of the N_j data records, s_{Aj}^2 is the sum of squared continuous attributes of the N_j data records, and $N_{Bj} = (N_{Bj1}, N_{Bj2}, \ldots, N_{BjK_B})$ is a $\sum_{k=1}^{K_B}(L_k - 1)$-dimensional vector where the k-th sub-vector is of $(L_k$-1) dimension, given by $N_{Bjk} = (N_{jk1}, \ldots, N_{jkL_k-1})$ in which N_{jkl} is the number of data records in C_j whose k-th categorical attribute takes the l-th category, $l = 1, \ldots, L_k$-1.

When two clusters C_j and C_s are said to merge, it simply means that the two corresponding sets of data points are gathered together to form a union. In this case, the $CF_{<j,s>}$ for the merged cluster $C_{<j,s>}$ can be calculated by simply adding the corresponding entries in CF_j and CF_s, that is,

$$CF_{<j,s>} = \{N_j + N_s, , s_{Aj} + s_{As}, \ s_{Aj}^2 + s_{As}^2, N_{Bj} + N_{Bs}\}. \tag{10}$$

This cluster feature is an efficient way of representation of data records because it stores much less than all the data records in the dense region and it is sufficient for calculating all the measures that are needed in the clustering algorithm.

In the pre-clustering step, data records are scanned sequentially from the dataset and the decision is made immediately whether the current record is to merge with any previously constructed dense region or to form a singleton by itself based on the distance criterion. During this data pass, a *CF-tree* is constructed to store the summary statistics of dense regions or singletons; it serves as a guidance structure to efficiently identify dense regions. Our pre-clustering step is implemented by constructing a modified CF-tree which is similar to that of BIRCH. When a data record is passing through a non-leaf node, it finds the closest entry in the node and travels to the next child node. The process continues recursively and the data record descends along the CF-tree and reaches a leaf-node. Upon reaching a leaf node, the data record finds the closest entry. The record is absorbed to its closest entry if the distance of the record and the closest entry is within a threshold value; otherwise it starts as a new leaf entry in the leaf node. If the CF-tree grows beyond the maximum size allowed, it is rebuilt by a larger threshold criterion. The new CF-tree is smaller and hence has more room for incoming records. The process continues until a complete data pass is finished.

3.1.1 Outlier-Handling Option

An optional outlier-handling step is implemented in our algorithm in the process of building the CF-tree. Outliers are considered as data records that do not fit well into any cluster. We consider data records in a leaf entry as outliers if the number

of records in the entry is less than a certain fraction of the size of the largest leaf entry in the CF-tree. This is different from BIRCH in which data records in a leaf entry are outliers if the size of the leaf node is less than a fraction of the average. We found from our experimental study that this definition is quite efficient in capturing outliers in a noisy data situation.

3.2 Step Two: Clustering

After the CF-tree is built in Step one, a collection of dense regions is identified and is stored in the leaf nodes of the tree. Since the number of dense regions is usually far less than the number of data records in the dataset and the summary statistics stored in the cluster features are sufficient for calculating the distance and related criterion, most clustering algorithms can be applied to cluster the dense regions very efficiently. In our algorithm, a hierarchical clustering algorithm using the log-likelihood based distance measure is used in this step.

3.3 Automatic Determination of Number of Clusters

When the number of clusters is not pre-specified, an additional step is added in our algorithm to determine the most appropriate number of clusters automatically in the final result. The basic existing strategy is to calculate criterion statistics for models with 1 cluster, 2 clusters, 3 clusters, etc. The number of clusters that results in the optimal criterion is the most appropriate number of clusters. A number of criterion functions can be found in the statistics literature. Banfield and Raftery [1] proposed a criterion called the Approximate Weight of Evidence (AWE) for hierarchical clustering based on the likelihood information in the data. AWE works reasonably well when the dataset does not have too much noise, however it performs poorly when the data is noisy. Fraley and Raftery [2] proposed to use the Bayes Information Criterion (BIC) [7] as a criterion statistic for determination of the appropriate number of cluster in their clustering method based on EM algorithm. In our study, we found that none of these methods gives completely satisfactory result for determining the number of clusters in our algorithm. Therefore, we developed a two-phase procedure to determine the number of clusters automatically, in which each phase is based on a different criterion. The first phase is to detect a coarse estimate of the number of clusters in the data based on Bayesian Information Criterion. BIC is a likelihood criterion penalized by the model complexity, which is measured by the number of parameters in the model. Models with small BIC are often considered good models. Usually as the number of clusters increases, BIC decreases first and then increases. In clustering step of our algorithm hierarchical clustering algorithm is applied to cluster the set of dense regions from pre-clustering step. At each merge, decrease in BIC is calculated. A coarse estimate of the number of clusters is obtained as the number of clusters at which the decrease in BIC starts to diminish when the number of clusters increases. In phase two, the ratio change in distance at each merge is used as the determination criterion. Merging starts from the set of clusters resulted from phase one, and an estimate of the number of cluster is obtained at the step where a big jump of the ratio change is observed. The rationale behind phase two is that a big jump in ratio change of the distance usually occurs when we start to merge two clusters that should not be merged.

We found these combined criteria of phase one and two work accurately in our study. The coarse estimate in phase one almost always overestimates the true number of clusters, which is adjusted by the distance-based criterion in phase two to bring the estimate down to the true number of clusters. Phase two needs the coarse estimate in phase one as well because it gives a good estimate of maximum number of clusters for phase two to start the calculation.

4. Cluster Membership Assignment

During the cluster membership assignment, each data record is assigned to a cluster. If the outlier-handling is not applied during the algorithm, the data record will be assigned to the closest cluster according to the distance measure. If the outlier-handling is used, two types of dense regions will appear at the end of pre-clustering step of the algorithm. The first type is the dense region gathered from the leaf nodes of the CF-tree. Data records in dense regions of this type are close to each other by the clustering nature of the algorithm. The other type is the collection of outliers at the end of pre-clustering step. Data records in this type of "dense region" do not fit well into any one of the first type dense regions and are thus put aside. In clustering step, the first type of dense regions is further clustered to the desired number of clusters. Hence the final result of the algorithm consists of the resulting clusters from clustering step and a noise cluster that is a collection of outliers from pre-clustering step.

The presence of the noise cluster in the result complicates the process of the cluster membership assignment; data records in the noise cluster do not fit well into any of the other non-noise clusters and hence the underlying model assumption of these records does not hold theoretically, which implies that the distance measure we derived from the current model assumption is less appropriate when it is used to measure the closeness of a data record to the noise cluster. To assign a newly coming data record to the appropriate cluster, we take an alternative approach based on the likelihood information theory. Knowing that the current model assumption does not hold well for data records in the noise cluster, we model these data records by considering a more relaxed distribution assumption on them. In our algorithm, we consider records in the noise cluster follow a uniform distribution. In the process of membership assignment, both the increases in likelihood of assigning the record to the noise cluster and to the closest non-noise cluster are first observed, the record is then assigned to the cluster that leads to the larger increase in the likelihood value.

5. Experimental Results

In this section, we present the experimental evaluation of our algorithm and compare its accuracy and performance with *k*-means clustering algorithm.

5.1 Generating Synthetic Data

In our experimental study, we have generated datasets of different number of records, clusters, continuous attributes and categorical attributes. Continuous attributes in a cluster are generated from multivariate normal distribution. The means and covariance matrices of different clusters are used to control the separation of continuous attributes of the clusters. We do not

assume independence among the attributes, although it is assumed in the distance measure development as discussed in section 2. Our simulation results show that the independent assumption is not critical. To generate categorical attributes in a cluster, we first choose a cluster center and then sample all attribute category combinations such that the center occurs most frequently. For any pair of clusters, the number of mismatched attributes of the two centers is used to control the separation of categorical attribute part of the clusters. When mixed type attributes are present, categorical attributes are assumed independent of each other, and are assumed independent of all continuous attributes.

5.2 Determination of Number of Clusters and Cluster Membership

We have generated over one thousand datasets to cover wide range of number of clusters, cluster sizes, and number of attributes. The number of attributes goes up to 640, number of cases up to 5 million, number of clusters up to 30, and cluster size up to 1 million. For about 98% of the datasets we generated, our algorithm is able to find the correct number of clusters. For the rest of the 2% of the datasets, we examined the data and found the clusters in the data are literally indistinguishable due to too much overlap. For all the datasets used in this paper, our algorithm correctly identifies the number of clusters.

The running time used in clustering with and without automatic determination of number of clusters is studied. It shows our method of determining the number of clusters is not only accurate, but also quick. The extra time it takes over using a pre-specified number of clusters is at most 3 seconds for all the datasets used in this paper. All the running times shown in later figures include the time of determining the number of clusters.

Since data are synthetically generated, we know which cluster each data record belongs to. By comparing the true cluster membership with the membership assigned by the algorithm, we can obtain the number of data records that are wrongly assigned and get a clustering error rate. The clustering error rate is an important index to evaluate the performance. For all the datasets that we tried and the correct numbers of clusters are determined, the clustering error rates are lower than 5%, and mostly under 1%. We also used the clustering error rate to compare the performance of our method to that of k-means, as shown later.

5.3 Scalability

Scalability is a desired property for algorithms dealing with large datasets. We test the scalability of our algorithm by increasing number of data records and number of attributes. All simulations are run on a PC with a 733 Mhz Pentinum III processor and 256MB RAM. The main memory is set to be 64 MB. The left panel of Figure 1 shows the relation between run time and number of cases, other factors fixed, on two different scenarios. All datasets have 5 clusters of equal size. For datasets with mixed type attributes, there are 5 continuous attributes and 5 categorical attributes with 2, 3, 5, 8, and 12 categories respectively. Other datasets have 5 continuous attributes only.

The right panel of Figure 1 shows the relation between run time and number of attributes, other factors fixed. All datasets have 7 clusters, each cluster having 1200 data records. All categorical attributes have 12 categories. The strong linearity indicates that our algorithm is highly scalable. Similar linearity is also shown when data with mixed type attributes are used. We would like to point out that all the running times include the time of determining the number of clusters. It is interesting to notice that categorical attributes need longer processing time than continuous attributes. It is because that the numbers of data records in every category of categorical attributes have to be recorded and kept track. Certainly the extra time used by categorical attributes depends on how many categories there are.

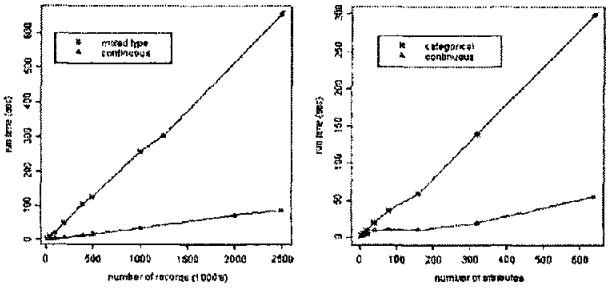

Figure 1. Relation between run time and number of records and number of attributes

5.4 Comparisons with k-means

We compared our algorithm to k-means algorithm implemented in Clementine 6.0, SAS 8e and SPSS 10.1. We decided to choose several k-means implementations because the performance of k-means varies greatly with its implementation.

We simulated 9 datasets, all of which have two continuous attributes and 5 clusters of equal size. The number of data records goes from 50,000 to 5 million. For all these datasets, our algorithm can automatically pick 5 in the range of 1 to 25 as the best number of clusters. However, we have to supply the number of clusters, 5, to k-means algorithm, since it cannot determine the number of clusters automatically. Automatically determine the best number of clusters is a big advantage of our algorithm over k-means. In the following, we compare the two algorithms in more detail from two aspects: speed and accuracy.

Table 1. Comparisons between our algorithm and k-means in speed and accuracy

Data records	Time (s) Our algorithm	Best of time*(s) k-means**	Error rate (%) Our algorithm	Best error rate*(%) k-means
50k	2	4	0.04	29.96
100k	4	2	0.07	0.04
200k	7	16	0.09	30.05
400k	15	19	0.03	1.1
500k	18	28	0.35	29.94
1000k	35	26	1.28	1.24
2000k	73	46	0.07	0.1
2500k	87	39	0.11	0.11
5000k	187	642	0.12	29.93

*: The least time and the lowest error rate from the 3 implementations of k-means algorithms.

**: The convergence criteria used is maximum 50 iterations or convergence rate = 0.0001.

The running time used by our algorithm shows a linear pattern similar to what is shown in Figure 1, it mostly depends on the number of data records in the scenario. For *k*-means, it shows much variation. Table 1 lists the time used by our algorithm and the best time used by *k*-means in the three implementations. We would like to emphasize that the running time used by our algorithm includes time to search for the optimal number of clusters from 1 to 25. Should one want to do a similar search via *k*-means using certain criterion, one would have to run it 25 times and the run time would be about 25 times longer!

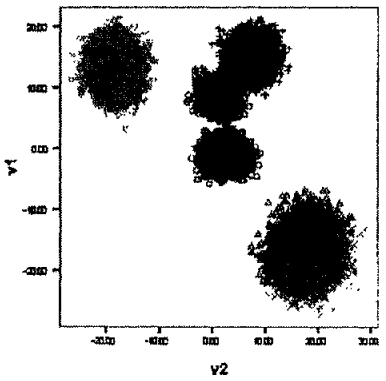

Figure 2. SAS *k*-means result for 5 clusters

As for the clustering error rate, our algorithm gets consistent rates of less than 1.3%, as shown in Table 1. For *k*-means, there is much variation in the implementations. Table 1 shows the best error rates from *k*-means among the three implementations. For 4 out of the 9 datasets we simulated, it cannot identify the five true clusters correctly. To look at the results more closely, we plot the assigned cluster membership for one dataset with 50,000 data records, with which all three implementations have difficulty dealing. Figure 2 shows the result from SAS *k*-means, which is close to those from Clementine and SPSS. Though the 5 clusters are well separated, *k*-means ends up with relatively large clustering error rate. The lower right cluster is split into 2 clusters, while the middle 2 clusters are treated as one. On the other hand, our algorithm identifies these 5 clusters with a clustering error rate as low as 0.04%. Figure 3 plots the results.

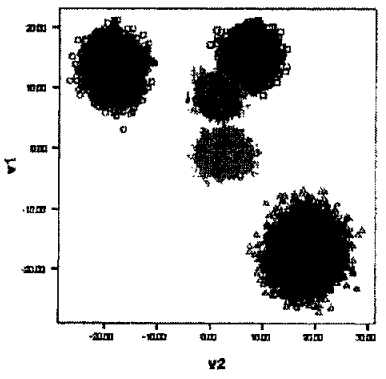

Figure 3. Result from our algorithm

5.5 Noise and Outlier Handling

One feature of our algorithm is its ability of handling noise and outliers. Background noise and outliers can be identified and screened out in our algorithm. To illustrate, we generate a dataset with 2 continuous attributes, and 2 clusters, each having 112 data records. We then randomly generated 1000 noises that are uniformly distributed around. The data is shown in the left panel of Figure 4. If we turn on the noise handling option, our algorithm can successfully identify the two clusters. 900 out of the 1000 noises will be screened out, other 100 that are close to the two clusters will be included into the 2 identified clusters. The result is shown in the right panel of Figure 4.

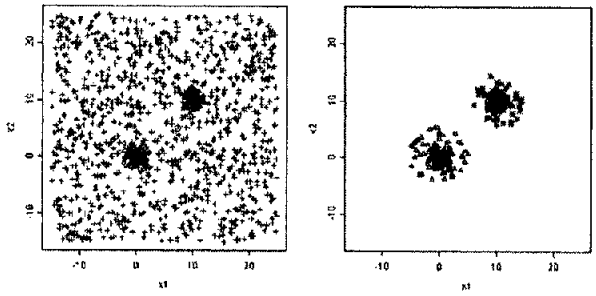

Figure 4. Results on noise handling

6. Conclusion

In this paper, we present a highly scalable and incremental clustering algorithm that handles mixed types of attributes, determines the best number of clusters automatically, and identifies outlier or noise data records. Experimental results show linear scalability and over 95% accuracy in identifying the number of clusters and cluster membership assignment.

7. REFERENCES

[1] Banfield, J. D. and Raftery, A. E. (1993). Model-based Gaussian and Non-Gaussian Clustering. Biometrics, 49, p. 803-821.

[2] Fraley, C., and Raftery, A. E. (1998). How Many Clusters? Which Clustering Method? Answers via Model-based Cluster Analysis. Computer Journal, 4, p. 578-588.

[3] Ganti, V., Gehrke, J., and Ramakrishnan, R. (1999). CACTUS – Clustering Categorical Data Using Summaries. In Proceedings of 1999 SIGKDD Conference. p. 73-82.

[4] Guha, S., Rastogi, R., and Shim, K. (1999). ROCK: A Robust Clustering Algorithm for Categorical Attributes. URL: http://www.bell-labs.com/project/serendip/.

[5] Huang, Z. (1998). Extensions to the K-means Algorithm for Clustering Large Datasets with Categorical Values. Data Mining and Knowledge Discovery, 2, p. 283-304.

[6] MacQueen, J. B. (1967). Some Methods for Classification and Analysis of Multivariate Observations. In Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, p. 281-297.

[7] Schwarz, G. (1978). Estimating the dimension of a model. The Annual of Statistics, 6, p. 461-464.

[8] Zhang, T., Ramakrishnan, R., and Livny M. (1997). BIRCH: A New Data Clustering Algorithm and its Applications. Data Mining and Knowledge Discovery 1 (2).