An Iterative Non-parametric Clustering Algorithm Based on Local Shrinking

Xiaogang Wang, Weiliang Qiu and Ruben H. Zamar
Department of Mathematics and Statistics, York University
Department of Statistics, University of British Columbia

Abstract

In this paper, we propose a new non-parametric clustering method based on local shrinking. Each data point is transformed in such a way that it moves a specific distance toward a cluster center. The direction and the associated size of each movement are determined by the median of its K-nearest neighbors. This process is repeated until a pre-defined convergence criterion is satisfied. The optimal value of the K is decided by optimizing index functions that measure the strengths of clusters. The number of clusters and the final partition are determined automatically without any input parameter except the stopping rule for convergence. Our performance studies have shown that this algorithm converges fast and achieves high accuracy.

Keywords: Automatic clustering; K-nearest neighbors; Local shrinking; Number of clusters; Strength of clusters.

1 Introduction

Clustering is the process of partitioning a set of objects into subsets based on some measure of similarity (or dissimilarity) between pairs of the objects. Cluster analysis has many applications in data mining in which large data sets, such as marketing data, need to be partitioned into much smaller and homogeneous groups. It also has wide implications to analyzing biological data. For example, given a set of gene expression data, a cluster of genes could suggest either the genes have a similar function in the cell or that they are regulated by the same transcription factors. For many clustering algorithms, such as K-means (MacQueen 1967; Hartigan and Wong 1979) and PAM (Kaufman and Rousseeuw 1990), the number of clusters or sub-populations needs to be specified by the user. The determination of the number of clusters is one of the most difficult problems in cluster analysis.

Most of the methods on estimating the number of clusters or sub-populations can be classified into several categories. The first category is to select the number of clusters by optimizing a certain measure of strength of the clusters (Tibshirani, Walther and Hastie 2000). This category embraces various methods of estimating the number of components of mixture of distributions (Fraley and Raftery 2002). The second class of methods first partition data into many small clusters, and then merge these small clusters until no clusters can be merged (Frigui and Krishnapuram 1999) .

Another strategy is to extract one cluster at a time (Zhung, Huang, Palaniappan and Lee 1996). Moreover, mode detection or bump hunting methods proposed by (Cheng and Hall 1998) can also be used to determine the number of clusters.

Recently, a new category for detecting the number of clusters has emerged. The main idea is to first iteratively move data points toward cluster centers, then use the number of convergent points as the number of clusters. The key issue there is how to move data points toward cluster centers. There are mainly two approaches to move data points toward cluster centers. The first approach, gravitational clustering (Wright 1977; Kundu 1999; Sato 2000; Wang and Rau 2001), can be interpreted from the point of view of field theory in physics. In these algorithms, each data point is considered as a particle of unit mass with zero velocity and it is gradually moving toward the cluster center due to the gravitation. The second approach, mean shift procedure studied by Fukunaga and Hostetler (1975) , Cheng (1995), and Comaniciu and Meer (1999, 2000, 2001, 2002) originates from the idea of kernel density estimation. In these algorithms, data points are transformed toward denser regions by using kernel functions.

The key idea of the mean-shift algorithm is the mean-shift vector which is defined as

$$m_{h,G}(\boldsymbol{y}) = \frac{\sum_{i=1}^{n} \boldsymbol{y}_i g\left(\left\|\frac{\boldsymbol{y}-\boldsymbol{y}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\boldsymbol{y}-\boldsymbol{y}_i}{h}\right\|^2\right)} - \boldsymbol{y} \tag{1}$$

where $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$ are n data points in p dimensional space $\mathcal{R}^p$, g is the profile of a kernel function G, h is the bandwidth, and $\boldsymbol{y}$ is any point in $\mathcal{R}^p$. If the kernel G has smoothed shadow, then

$$m_{h,G}(\boldsymbol{y}_0) = 0, \tag{2}$$

where $\boldsymbol{y}_0$ is a mode of the density estimate derived from $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$. The essential idea of the mean-shift algorithm is to find the mode of the density estimate by the following iteration formula:

$$\boldsymbol{y}^{(t+1)} = \frac{\sum_{i=1}^{n} \boldsymbol{y}_i g\left(\left\|\frac{\boldsymbol{y}^{(t)}-\boldsymbol{y}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\boldsymbol{y}^{(t)}-\boldsymbol{y}_i}{h}\right\|^2\right)}. \tag{3}$$

The partition can be derived once the modes are located.

Cheng (1995) generalized Fukunaga and Hostetler (1975)'s result to allow non-flat kernels, weighted data points, and *non-blurring process*. For non-blurring process, Comaniciu and Meer (1999) gave a rigorous proof that the sequence $\{\boldsymbol{y}^{(t)},\ t = 1, 2, \ldots\}$

converges if the kernel G is the Epanechnikov kernel. Comaniciu and Meer (2000) generalized this result to any kernel function which has a convex and monotonically decreasing profile. Comaniciu and Meer (2001) extended the results further to allow variable bandwidth $h(\boldsymbol{y}_i)$. It is also possible to consider another form of variable bandwidth which is a function of the estimation point $\boldsymbol{y}$ instead of data point $\boldsymbol{y}_i$. If $h(\boldsymbol{y})$ is the Euclidean distance between $\boldsymbol{y}$ and the K-th nearest neighbor of $\boldsymbol{y}$ among the $\boldsymbol{y}_i$'s, the non-parametric estimator is the K-nearest neighbor density estimator (Mack and Rosenblatt 1979) . Choi and Hall (1999) and Hall and Heckman (2000) studied data sharpening. Although the main purpose of data sharpening is to reduce the bias of the kernel estimate at the valleys and the peaks of the density function, it can also be used to estimate the number of clusters or modes.

In this paper, we propose an automatic clustering algorithm that determines the number of clusters and the partition without any input parameter except the convergence criterion. Our algorithm also shrinks data points towards cluster centers as in the mean-shift algorithm. This process is repeated until the specified criterion is satisfied. The final partition can then be obtained as all the data points have converged to the cluster centers. In our algorithm, however, the shrinking process is determined by K-nearest neighbor approach instead of kernel functions. As any method that utilizes the K-nearest neighbor approach, our algorithm is very sensitive to the local geometrical structure and highly adaptive especially in high-dimensional sparse sample space. This property would also lead to better clustering especially when the clusters are of irregular shapes. Fukunaga and Hostetler (1975) mentioned that their method could be generalized to K-nearest neighbor density estimator. Comaniciu and Meer (2001) also mentioned the possibility of using the K-nearest neighbor density estimate.

Since the shrinking in our algorithm is dictated by the K-nearest neighbor approach and the final partition is a function of the K, one would question how this value should be determined. To resolve this critical issue of choosing the value for K, our algorithm searches for the value of K that maximizes the index function such as the CH index (Calinski and Harabasz, 1974) and the Silhouette index (Kaufman and Rousseeuw, 1990). To be more specific our algorithm starts from small K and gradually increases the size of K until the measure of strength of clusters such as the CH index or the Silhouette index is optimized. The estimation of the number of clusters and the ultimate partition are then obtained simultaneously based on the optimal K. Our algorithm bears similarities to the second class of methods to determine the number of clusters as it starts from many small clusters first and then merge them together. However, the number of clusters is not determined by the user. Instead, it is obtained automatically through optimizing the measure of strength of clusters.

This paper is organized as follows. In next section, we present our clustering algorithm, CLUES. In Section 3, we describe the data sets used for performance study and discuss the results. The last section of the paper contains conclusion and

discussion for future works.

2 CLUES algorithm

CLUES (CLUstEring based on local Shrinking) algorithm consists of three major elements:
(1) Shrinking procedure;
(2) Partition procedure;
(3) Determination of the optimal K.
We will describe the shrinking procedure in the next subsection. We will then discuss the partition procedure in the following subsection. The measure of strength of clusters such as the CH index and the Silhouette index are discussed in Section 2.3. The determination of the K is discussed in Section 2.4.

2.1 Shrinking Procedure

The key idea of shrinking procedure resembles the gravitational clustering and the data sharpening procedure. As in the gravitational clustering, each data point can be viewed as a particle in a gravitational field with unit mass and zero velocity at the beginning. The local gravitational field would pull each data point into a denser region according some gravitational laws. This can also be called sharpening effect as the boundary of each cluster should be clearer after this step. Therefore, the minimum distance among clusters will increase as the procedure goes on. In the sparse section of the sample space, the magnitude of the movement can be relative large. In the dense region, however, the movement can be small since data points are quite close to each other already. Despite the choice of different gravitational laws, data points will converge to a few so called focal points which are indeed the cluster centers or the modes of a probability density function. Eventually, all data points within one cluster will converge arbitrarily close to their cluster centers. Once the cluster centers or the modes are identified, the partition procedure can then be applied.

In our algorithm, each data point will be "shrank" toward a denser region. However, the movement is determined by the median of its K nearest neighbors as the median is more robust than the mean. The iteration is defined as

$$\boldsymbol{y}_i^{[t+1]} = \text{median}_{\boldsymbol{y}_j^{[t]} \in \text{N}_K\left(\boldsymbol{y}_i^{[t]}\right)} \boldsymbol{y}_j^{[t]}, \tag{4}$$

where $\text{N}_K\left(\boldsymbol{y}_i^{[t]}\right)$ is the smallest sphere that contains K nearest neighbors of the data point $\boldsymbol{y}_i^{[t]}$.

The coordinates of each data point will then be updated at each iteration step. This procedure is terminated once the convergence is observed.

Let the $n \times p$ matrix Y be the collection of all the data points and $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n$ be rows of the matrix Y. They are data points in the p dimensional space $\mathcal{R}^p$. Let y_{ij} to be the j-th coordinates of $\boldsymbol{y}_i$. For any fixed K, ϵ and M, the algorithm is described as follows.

Step 1 *Initialize the iteration number* $t \leftarrow 1$. *Backup original data points* $Y^{[t]} \leftarrow Y$ *and* $Y^{[t+1]} \leftarrow Y$.

Step 2 *For each point* $\boldsymbol{y}_i^{[t]}$, *update coordinates:* $\boldsymbol{y}_i^{[t+1]} \leftarrow median_{\boldsymbol{y}_j^{[t]} \in \mathrm{N}_K\left(\boldsymbol{y}_i^{[t]}\right)} \boldsymbol{y}_j^{[t]}$.

Step 3 $d \leftarrow \max_{i=1}^{n} \max_{j=1}^{p} \left| y_{ij}^{[t]} - y_{ij}^{[t+1]} \right|$.

Step 4

S 4.1 If $d < \epsilon$ *or* $t > M$, *then go to Step 5.*

S 4.2 Otherwise, $y_{ij}^{[t]} \leftarrow y_{ij}^{[t+1]}$, *and then go to Step 2.*

Step 5 *Output* $\boldsymbol{y}_i^{[t+1]}, i = 1, \ldots, n$.

Note that the parameter ϵ is the stopping rule. It takes a small number, say, 10^{-4}. This parameter is required for every iterative algorithm in numerical analysis. As long as it is reasonably small, the algorithm will be satisfactory. The number M is the maximum number of iteration allowed to prevent infinitely many iterations due to possible unreasonable requirement for convergence.

2.2 Partition Procedure

Once the data points are shrank to the cluster centers, we then perform the partition procedure to derive the membership function for the data points. To describe the partition procedure, we define two sets: S_U and S_R, where S_U contains the labels of the used data points and S_R represents the set of the labels of the remaining data points for this step. First, we choose an arbitrary label, say a, from the set of S_R. Then we find the closest point of $\boldsymbol{y}_a$, say $\boldsymbol{y}_b$. Denote the distance between $\boldsymbol{y}_a$ and $\boldsymbol{y}_b$ as d_b and push the label a into S_U. Then we replace $\boldsymbol{y}_a$ by $\boldsymbol{y}_b$ and repeat the procedure. The partition is then based on the order and the size of these distances, similar to the partition procedure used in OPTICS (Ankerst, Breunig, Kriegel, and Sander 1999) Thus, the distances are then put into an array with the order for the data points are based on how they entered the set S_U. Large jumps of the distances always suggest the beginning of a new cluster. A natural scientific question would be: how big the jump has to be in order to trigger the algorithm to begin recording another cluster? In order to avoid any subjective criterion, we employ a commonly used outlier detection procedure in Statistics to determine the threshold. To be more

specific, we first calculate the inter-quartile range (IQR) for the distances. The IQR is defined as the difference between 75% and 25% percentile. It is clear from the definition of the IQR that it offers a range such that 50% of the distances fall into. Then, the outlier detection procedure will keep track of all distances that are 1.5 IQR away from the average of all the distances. Note that small consecutive distances imply the existence of only one cluster. The lower bound for all metric function is zero and the majority of d_b's are close to zero after the shrinking procedure. Thus, the outlier detection procedure should be one-sided, *i.e.*, only the upper tail of the probability density function of d_b is of importance. We then find the exact location of these big jumps. All the data points lie between two consecutive jumps will be considered as members of the same cluster. The corresponding labels are then used to construct the partition. Detailed description of this procedure is given as follows.

Step 1 *Set the iteration number* $t = 1$. *Initialize* $S_U = \emptyset$ *and* $S_R = \{1, \ldots, n\}$.

Step 2 *Pick an arbitrary label from* S_R *and set it to be* a.

Step 3 *If* $S_R \neq \emptyset$, *then*

- *Find* b *s.t.* $b = arg\min_j d(\boldsymbol{y}_a, \boldsymbol{y}_j)$, i.e., $\boldsymbol{y}_b$ *is the nearest neighbor of* $\boldsymbol{y}_a$.
- $d_b \leftarrow d(\boldsymbol{y}_a, \boldsymbol{y}_b)$.
- $S_R \leftarrow S_R - \{b\}$ *and* $S_U = S_U + \{b\}$
- $t \leftarrow t + 1$ *and* $a \leftarrow b$.
- *Go to Step 3.*

Step 4 *Compute* $R = \sum_{i=1}^{n-1} d_i/(n-1) + 1.5IQR$.

Step 5 *Initialize the membership as* $C_i = 1$, $i = 1, \ldots, n$, *the number of clusters as* $g = 1$ *and the number of iteration as* $t = 1$.

Step 7 *If* $d_t > R$, *then* $g \leftarrow g + 1$.

Step 8 $C_{S_U[t+1]} \leftarrow g$, *where* $S_U[t+1]$ *is the* $(t+1)$*-th element of the set* S_U.

Step 9 *If* $t < n$, *then* $t \leftarrow t + 1$ *and go to Step 7. Otherwise, go to Step 10.*

Step 10 *Return* C_i, $i = 1, \ldots, n$.

The major difference between this partition procedure and the one used in OPTICS is the use of the outlier detection procedure. This outlier detection procedure is widely used in Statistics and Data Analysis. There are other alternatives available such as the calculation of t-score. If the size of each jump is quite significant, then any outlier detection procedure would be able to locate these jumps.

2.3 Measure of Strengths of Clusters

The clustering result obtained from the *partition procedure* depends on the result from the *shrinking procedure*. It is clear that the choice of the neighbors used is crucial to both steps as the second step is built on the first one. For example, if the number of neighbors used is set to n, the total number of data points, then there is one cluster. On the other hand, if the value for K is set to be 0, then there will be n clusters. Any other choice of K, $0 < K < n$ will result the number of clusters to be between 1 and n. In theory, the number of clusters is a monotone decreasing function of the value of K.

In order to make our algorithm parameter free, we propose to choose the value of K based on the "best" clustering result. The "best" clustering depends on the measure of strength of clusters. There are many measures of the strength of clusters. For example Milligan and Cooper (1985) compared 30 measures of the strengths of clusters for determining the number of clusters. Their simulation results show that the Calinski and Harabasz index (CH index) has the best performance. The CH index is defined as

$$CH(g) = \frac{B(g)/(g-1)}{W(g)/(n-g)} \tag{5}$$

where g is the number of clusters,

$$B(g) = \sum_{i=1}^{g} n_i(\bar{\boldsymbol{y}}^{(i)} - \bar{\boldsymbol{y}})(\bar{\boldsymbol{y}}^{(i)} - \bar{\boldsymbol{y}})^T \tag{6}$$

is the between-groups sum of squares and products matrix,

$$W(g) = \sum_{i=1}^{g} \sum_{j=1}^{n_i} (\boldsymbol{y}_j^{(i)} - \bar{\boldsymbol{y}}^{(i)})(\boldsymbol{y}_j^{(i)} - \bar{\boldsymbol{y}}^{(i)})^T \tag{7}$$

is the within-groups sum of squares and products matrix, n_i is the size of the i-th cluster, $\boldsymbol{y}_j^{(i)}$ is the j-th data point in the i-th cluster, $\bar{\boldsymbol{y}}^{(i)}$ is the mean vector of the i-th cluster, and $\bar{\boldsymbol{y}}$ is the overall mean of all data points.

Kaufman and Rousseeuw (1990) proposed the Silhouette index to measure the strengths of clusters. The silhouette index is defined as

$$\bar{s} = \frac{1}{n} \sum_{i=1}^{n} s_i \tag{8}$$

where

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)} \tag{9}$$

and a_i is the average distance of the data point $\boldsymbol{y}_i$ to other points in the cluster A where $\boldsymbol{y}_i$ belongs to, i.e.

$$a_i = \frac{1}{(n_A - 1)} \sum_{j \in A, j \neq i} d(\boldsymbol{y}_i, \boldsymbol{y}_j),$$

and b_i is the average distance to points in the nearest neighbor cluster besides its own. Define

$d(i, C)$ = average dissimilarity of the data point $\boldsymbol{y}_i$ to all data points in Cluster C.

Then

$$b_i = \min_{C \neq A} d(i, C).$$

This index s_i can take values from -1 to 1. When the index is zero, then the data point $\boldsymbol{y}_i$ has equal distance to its cluster and its nearest neighbor cluster. If the index is positive, then the data point $\boldsymbol{y}_i$ is closer to its cluster than other clusters. If the index is negative, then the data point $\boldsymbol{y}_i$ is wrongly assigned to the current cluster. Thus, if all data points are correctly assigned, then average of s_i's should be close to 1.

We have used both the CH index and the Silhouette index functions to measure the strengths of clusters. For three of the data sets used, the results are quite similar. The results do differ in two data sets. We will discuss this issue further in Section 3.

2.4 Determination of the Optimal K

In order to find the optimal K, we could calculate the value of the index function for K from 2 to n-1. The optimal K is then chosen as the value that gives the optima of the index function. Although this idea works in theory, the corresponding algorithm will be very slow. There are many unnecessary computations required by this naive approach. One obvious improvement is to terminate the search when the number of clusters has been reduced to 2. The logic is that the number of clusters will be either 2 or 1 if the value of K is increased further. Either of the two cases will not cause the improvement of the index functions. This strategy will result significant saving of computing time if small value of K can identify the true underlying pattern correctly. For example, if the clusters are all well separated and very tight, then small K would result in good partition. Moderate increase of the value of K would result merges of these tight clusters hence decreasing the measure of strength of clusters. Thus, there is no need to search through all the possible values of the K as many of them will not bring any benefit to the improvement of the measure of strength of clusters. For large data sets with un-usual patterns, this strategy would not save much of computing

time. Given the fact that the K-nearest neighbor is used to ensure good quality of partition and the fact that the algorithm might needs to search through many values of K, the speed of the proposed algorithm is an issue. For example, if the total number of data point is 1000 and the search eventually stops at $K = 500$, then our algorithm will not be very efficient. To speed up this process, we introduce the speed factor α, $0 < \alpha < 1$. The initial starting point of K and the increment is set to be αn. For example, if there are 1000 data points and α is set to be 0.05, then the initial search will start at $\alpha n = 50$. The next search will be at 100. Thus, the algorithm will only search the sampling point determined by the speed factor α. By tuning the size of the α, the search can go much faster than the default setting in which the initial value of K and the increment are both set to be 1, *i.e.* $\alpha = 1/n$.

By introducing the speed factor, the algorithm will converge faster. However, the speed might be obtained at the cost of the losing accuracy. To rectify this potential problem, a refinement procedure can be applied once the rough estimate of the optimal K is obtained. This refinement procedure is essentially the same as the search procedure except the increment is now set to be 1. For example, if there are 980 data points and the optimal K is 257. Assume that by using the speed factor $\alpha = 0.05$, our algorithm delivers 250. The refinement procedure will then search from $201(= 250 - \alpha n)$ to $299(= 250 + \alpha n)$ in order to find the true optimal value for K. Experiences have shown that the choice of K does not affect the cluster result as long as it lies in the neighborhood of the optimal K obtained from the first pass. In the above setting, the clustering result could remain the same for the values of K ranging from 250 to 300. Therefore the refinement procedure is optional.

To further increase the power and the speed of the search of the optimal K, the shrinking procedure for the next value of the K is built on the shrinking result by using the previous K. The transformed coordinates for every data point obtained from previous shrinking are then used as the input for the next shrinking procedure. Simulation studies have shown that this approach is much more efficient when compared with the naive approach. More importantly, it is much more accurate than the result obtained by the naive approach. Intuitively, by using the result from previous shrinking and increase the value of K is equivalent to merging small clusters to form bigger ones. However, the merge is governed again by the rule of K nearest neighbors. No threshold is needed for any possible merge. The process is automatic except that the convergence criterion is specified by the user.

In this algorithm, we also implemented outlier detection procedure. Ideally, each data point should go through the outlier detection procedure as introduced before. In reality, however, this proves to be impractical as the clustering patterns are changing for every iteration. We are then forced to think about other alternative. Outliers, in principle, behave differently than the majority of the data points. Thus, the number of outliers should be very small. Based on this observation, we could put a threshold on the minimum size of a cluster. This threshold is also set to be αn.

Denote the number of data points by n, the dimensionality by p, the number of clusters by g, the partition by C, the outlier threshold by T, and the increment of the number of nearest neighbors by δ. The CLUES algorithm is defined as follows:

Step 1. *Initializing.* $\alpha \leftarrow 0.05$, $K \leftarrow \alpha n$, $\delta \leftarrow \alpha n$, $T \leftarrow \alpha n$, $t \leftarrow 0$ *and* $Y^{(t)}_{n\times p} \leftarrow Y_{n\times p}$, *and* $s_{\max} = -100$, *where* $s_{\max}$ *will record the maximum value of the strength of clusters.*

Step 2. $t \leftarrow t+1$. *Perform the* shrinking procedure *and set* $Y^{(t)}_{n\times p} \leftarrow Y^{(t-1)}_{n\times p}$.

Step 3. *Perform the* partition procedure *and obtain a partition* $C^{(t)}$ *based on* $Y^{(t)}_{n\times p}$.

Step 4. *If* $g^{(t)} > 1$, *then*

S 4.1. *Obtain the value of the chosen index* $s^{(t)}$.

S 4.2. *If* $t = 1$, *then* $s_{\max} \leftarrow s^{(t)}$, $g^* \leftarrow g^{(t)}$, *and* $C^* \leftarrow C^{(t)}$.

S 4.3. *If* $\min_i n_i \geq T$ *and* $s^{(t)} > s_{\max}$, *then* $s_{\max} \leftarrow s^{(t)}$, $g^* \leftarrow g^{(t)}$, $C^* \leftarrow C^{(t)}$. n_i *is the size of the* i*-th cluster.*

S 4.4. *If* $\min_i n_i \geq T$ *and* $g^* = 2$, *then go to Step 7.*

S 4.5. *If* $\min_i n_i \geq T$ *and* $g^{(t)} = 2$, *then go to Step 7.*

S 4.6. $K \leftarrow K + \delta$. *Go to Step 2.*

Step 5. *Otherwise if* $g^{(t)} = 1$ *and* $t = 1$, *then* $g^* \leftarrow 1$, $C^* \leftarrow \{1, \ldots, 1\}$, *and go to Step 7.*

Step 6. *Otherwise if* $g^{(t)} = 1$ *and* $t > 1$, *go to Step 7.*

Step 7. *Output the optimal number of cluster* g^* *and the final partition* C^*.

Since index functions measure the strength of clustering, the optimum of the index function is achieved by the true partition. If the partition is identical or reasonable close to the true underlying pattern, then the optimum of the index function is reached. Therefore the optimum of the index function is unique. We remark that the optimal K might not be unique.

3 Performance Analysis

3.1 Recover rate index

Once the partition is derived, one wants to know how close the partition is if compared with the true partition. One such measure is the misclassification rate of the number

of data points. However, for most clustering algorithms, the labeling of the partition may be different from that of the true partition. For example, the label of "Cluster 1" in the partition you get may be the label of "Cluster 6" in the true partition. It is time-consuming and not easy to re-label the partition to make it consistent with the initial one. Moreover, in order to determine the correct number of clusters, it is crucial to compare partitions with different number of clusters. Thus, it is desirable to use index functions that do not depend on the initial labeling. There are many such indices to measure the agreement between two partitions. Milligan and Cooper (1986) compared 5 such kind of indices: *Rand index*, *Hubert and Arabie's adjusted Rand index*, *Morey and Agresti's adjusted Rand index*, *Fowlkes and Mallows index*, and *Jaccard index*. We have used all five index functions to assess the accuracy of our algorithm although Milligan and Cooper (1986)'s simulation study show that the Hubert and Arabie's adjusted Rand index has the best performance.

3.2 Analysis of Four Famous Data Sets

We consider four widely studied data sets. They are:

- Ruspini data set (Ruspini 1970). It has 4 well-separated clusters in the two-dimensional space.

- The Maronna data set (Maronna and Jacovkis 1974) . It has 4 slightly overlapped clusters in the two-dimensional space. Each cluster contains 50 data points. The Maronna data is a simulated data set. The data are drawn from 4 bivariate normal distributions with identity covariance matrix and mean vectors

$$\boldsymbol{\mu}_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \ \boldsymbol{\mu}_2 = \begin{pmatrix} 4 \\ 0 \end{pmatrix}, \ \boldsymbol{\mu}_3 = \begin{pmatrix} 1 \\ 6 \end{pmatrix}, \ \boldsymbol{\mu}_4 = \begin{pmatrix} 5 \\ 7 \end{pmatrix}.$$

- The Iris data set(Fisher 1936) . The Iris data set has four variables and three clusters. The first two variables ("Sepal.Length" and "Sepal.Width") are noisy variables. Two clusters in the Iris data set are overlapped.

- The Vowel data set (Hastie, Tibshirani and Friedman 2001) . In the Vowel data set, there are 11 different words, each containing a vowel sound. The 11 different vowel sounds, which are measured by 10 variables, correspond to 11 clusters. There are 528 training observations and 462 test observations. Here we use the 528 training observations to check the performance of our algorithm and refer it as the Vowel data set. Figure 1 shows the 11 clusters in the optimal two-dimensional subspace (Hastie et al. 2001, pp. 84-95). The 11 clusters are heavily overlapped.

We apply our algorithm to these four data sets and compare the clustering result with that obtained by K-means and PAM. We use the K-means and PAM procedure in Splus 6.

Table 1 lists the results obtained by the CLUES algorithm with the CH index and the Silhouette index. Table 1 also lists the results obtained by PAM and K-means. The second column represents the true number of clusters. The third column represents the estimated number of clusters by CLUES. For PAM and K-means algorithm, the numbers in parenthesis are the values used as the input parameter for these two algorithms since both of them need the user to provide the number of clusters in advance.

For the Ruspini data set, not only does the CLUES algorithm estimate the number of clusters correctly but also it has achieved a perfect partition according to the values of the 5 recover rate indices. If we provide the correct number of clusters for PAM and K-means, they will achieve perfect partition as well. However, this very important parameter has to be given to both K-means and PAM. Otherwise, these two algorithms will not be able to partition perfectly as it is evident from the table.

For the Maronna data set, the optimal partition produced by the CH index and the Silhouette index are marginally better than the K-means and PAM if they are provided with the right number of clusters. However, the results from the CLUES algorithm are significantly better than that from K-means and PAM with wrongly specified number of clusters.

For the Iris data, the number of clusters was correctly identified by using Silhouette index but not by the CH index. In fact, the partition and the estimation of number of clusters are not satisfactory by using CH index. This could be cause by the fact that there are two noisy variables in this data set. The partition by Silhouette index is still uniformly better than the partition by PAM and K-means even when these two algorithms are provided by the right number of clusters.

The Vowel data set is quite challenging as the clusters are heavily overlapped. Our algorithm could not identify the correct number of clusters. However, the final partition by CLUES is reasonable (See Figure 2) and better than the partition by PAM and K-means even though they are provided with the correct number of clusters. Given the true number of clusters, PAM got slightly larger value of the Rand index than that by CLUES. For all the 6 methods, the values of the Rand index are much larger than those of the other 4 indices. It could be due to chance agreement (The Rand index does not correct for the chance agreement (Morey and Agresti 1984; Hubert and Arabie 1985). The values of the other four indices show that the partition obtained by CLUES is closer to the true partition than the one obtained by PAM is. The raw data points and the partition obtained by CLUES are plotted on Figure 1 and 2.

3.3 Further Analysis

Since both the CH index and the Silhouette index are employed by the CLUES algorithm, a decision must be made if these two give different partitions. The fact that CLUES with the CH index did not perform well for the Iris data set also warrants further analysis. To challenge the proposed CLUES algorithm, we simulated the following data set with 5 clusters. There are 4 clusters in symmetric positions while one cluster sits in the middle region. We name this data set as the "Broken Ring" data set. The clusters are reasonably separated but 4 of them have slightly un-usual shapes. The data points are plotted in Figure 3.

As shown in Figure 3, the CLUES with Silhouette does a perfect job. It not only gives the correct number of clusters but it also achieves the perfect partition. That is, there is no classification error. However, the partition done by the CH index failed once again. This time, the estimated number is 22 which is far from the truth. True clusters are cut into smaller portions. Given the true partition, the partition by using CH is not very reasonable. By examining the formula for the CH index, it can be seen that the denominator is the within group sum of squares. Because the clusters are of un-usual shapes, the use of the CH index would demand that many clusters be formed to reduce the within group sum of squares. Therefore, the CH index will be idea for clusters with spherical shapes. This also partially explains why the algorithm by the CH index did not do well in the Iris data set. Given the fact that the CLUES with silhouette out-performs the CLUES with CH for the data sets we tried, the result by the Silhouette index seems to be more reliable than that from the CH index.

It is also interesting to compare CLUES with PAM and K-means by using the Broken Ring data set. From Table 2, it can be seen that PAM and K-means could not get a perfect partition even if the true number of clusters is provided. Figure 4 shows that PAM mis-classified some data points on the boundary of the middle clusters. Since PAM and K-means always try to approximate clusters with spherical shapes or normal density functions, it is not surprising that they would mis-classify some points on the boundary of the middle cluster. If the correct number of clusters is not provided, the K-means and PAM do not do well when compared with CLUES with the Silhouette index.

4 Conclusion and Future Work

In this paper, we propose a non-parametric algorithm based on local shrinking on estimating the number of clusters and partition of data points without any input parameter except the convergence criterion. This overcomes the shortcoming of many traditional clustering algorithms that demand the exact knowledge of the number of clusters. Even when the true number of clusters is provided to PAM and K-means,

the CLUES algorithm with Silhouette index still out-performs the aforementioned two clustering algorithms for all the data sets we used. The improvement would be quite significant when dealing with un-usual shaped clusters. The analysis of the Broken Ring data set shows that the CLUES algorithm can achieve remarkable result to cluster un-usual patterns.

In this paper, we use the CH index and the Silhouette index. The Silhouette index seems to do better than the CH index. There are many other good measures of the strengths of clusters such as the Gap statistic proposed by Tibshirani et al. (2000) and many indices listed in Milligan and Cooper (1985). We will try these indices to select good indices. In our future work, we will use this algorithm to detect outliers by examining the distance they travel to the cluster center. We will also make this algorithm scalable so that it can handle large disk-resident data sets.

References

Ankerst, M., Breunig, M.M., Kriegel, H.P., and Sander, J. (1999), "OPTICS: Ordering Points to Identify the Clustering Structure," *International Conference On Management of Data*, 49–60.

Calinski, R. B. and Harabasz, J. (1974), "A dendrite method for cluster analysis," *Communications in Statistics*, 3, 1–27.

Cheng, M. Y. and Hall, P. (1998), "Calibrating the excess mass and dip tests of modality," *Journal of the Royal Statistical Society, Ser. B*, 60, 579–589.

Cheng, Y. (1995), "Mean Shift, Mode Seeking, and Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17, 790–799.

Choi, E. and Hall, P. (1999), "Data sharpening as a prelude to density estimation," *Biometrika*, 86, 941–947.

Comaniciu, D. and Meer, P. (1999), "Mean Shift Analysis and Applications," in *Proceedings of the Seventh International Conference on Computer Vision*, pp. 1197–1203.

— (2000), "Real-Time Tracking of Non-Rigid Objects Using Mean Shift," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00)*, vol. 2, pp. 142–149.

— (2001), "The Variable Bandwidth Mean Shift and Data-Driven Scale Selection," in *Proceedings of the Eighth International Conference on Computer Vision*, vol. 1, pp. 438–445.

— (2002), "Mean Shift: a robust approach toward feature space analysis," *IEEE transactions on pattern analysis and machine intelligence*, 24, 603–619.

Fisher, R. A. (1936), "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, 7, 179–188.

Fraley, C. and Raftery, A. E. (2002), "Model-based clustering, discriminant analysis, and density estimation," *Journal of the American Statistical Association*, 97, 611–631.

Frigui, H. and Krishnapuram, R. (1999), "A robust competitive clustering algorithm with applications in computer vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21, 450–465.

Fukunaga, K. and Hostetler, L. D. (1975), "The estimation of the Gradient of a density function, with applications in pattern recognition," *IEEE Transactions on Information Theory*, 21, 32–40.

Hall, P. and Heckman, N. (2000), "Estimating and depicting the structure of a distribution of random functions," Tech. Rep. 197, Dept. of Statistics, University of British Columbia.

Hartigan, J. A. and Wong, M. A. (1979), "A K-means clustering algorithm," *Applied Statistics*, 28, 100–108.

Hastie, T., Tibshirani, R., and Friedman, J. (2001), *The elements of statistical learning: data mining, inference, and prediction*, Springer-Verlag.

Hubert, L. and Arabie, P. (1985), "Comparing partitions," *Journal of Classification*, 2, 193–218.

Kaufman, L. and Rousseeuw, P. J. (1990), *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley, New York.

Kundu, S. (1999), "Gravitational clustering: a new approach based on the spatial distribution of the points," *Pattern Recognition*, 32, 1149–1160.

Mack, Y. P. and Rosenblatt, M. (1979), "Multivariate k-Nearest Neighbor Density Estimates," *Journal of Multivariate Analysis*, 9, 1–15.

MacQueen, J. B. (1967), "Some Methods for Classification and Analysis of Multivariate Observations," in *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, 1*, Berkeley, Calif: University of California Press, pp. 281–297.

Maronna, R. and Jacovkis, P. M. (1974), "Multivariate Clustering Procedures with Variable Metrics," *Biometrics*, 30, 499–505.

Milligan, G. W. and Cooper, M. C. (1985), "An Examination of procedures for determining the number of clusters in a data set," *Psychometrika*, 50, 159–179.

— (1986), "A study of the comparability of external criteria for hierarchical cluster analysis," *Multivariate Behavioral Research*, 21, 441–458.

Morey, L. C. and Agresti, A. (1984), "The measurement of classification agreement: an adjustment to the Rand statistic for chance agreement," *Educational and Psychological Measurement*, 44, 33–37.

Ruspini, E. H. (1970), "Numerical Methods for Fuzzy Clustering," *Information Science*, 2, 319–350.

Sato, Y. (2000), "An Autonomous Clustering Technique," in *Data analysis, classification, and related methods*, eds. Kiers, A. L. Henk, Rasson, Jean-Paul, Groenen, Patrick J. E., and Schader, Martin, Springer.

Tibshirani, R., Walther, G., and Hastie, T. (2000), "Estimating the number of clusters in a dataset via the gap statistic." Tech. Rep. 208, Dept. of Statistics, Stanford Univ.

Wang, J. H. and Rau, J. D. (2001), "VQ-Agglomeration: A Novel Approach to Clustering," *IEE Proceedings-Vision, Image and Signal Processing*, 148, 36–44.

Wright, W. E. (1977), "Gravitational Clustering," *Pattern Recognition*, 9, 151–166.

Zhung, X., Huang, Y., Palaniappan, K., and Lee, J. S. (1996), "Gaussian mixture modeling, decomposition and applications," *IEEE Transactions on Signal Process*, 5, 1293–1302.

Table 1: Results for the 4 widely used data sets

Data Set	g	g	method	HA	MA	Rand	FM	Jaccard
Ruspini	4	4	CLUES-C	1	1	1	1	1
	4	4	CLUES-S	1	1	1	1	1
	4	(3)	PAM	0.74	0.75	0.89	0.83	0.69
	4	(4)	PAM	1	1	1	1	1
	4	(5)	PAM	0.96	0.96	0.98	0.96	0.93
	4	(3)	K-means	0.68	0.69	0.86	0.80	0.64
	4	(4)	K-means	1	1	1	1	1
	4	(5)	K-means	0.91	0.91	0.97	0.93	0.87
Maronna	4	4	CLUES-C	0.93	0.94	0.98	0.95	0.91
	4	4	CLUES-S	0.93	0.95	0.98	0.95	0.91
	4	(3)	PAM	0.68	0.68	0.86	0.79	0.63
	4	(4)	PAM	0.92	0.92	0.97	0.94	0.89
	4	(5)	PAM	0.84	0.84	0.94	0.88	0.78
	4	(3)	K-means	0.68	0.68	0.86	0.79	0.63
	4	(4)	K-means	0.92	0.92	0.97	0.94	0.89
	4	(5)	K-means	0.80	0.80	0.93	0.85	0.73
Iris	3	2	CLUES-C	0.57	0.57	0.78	0.77	0.60
	3	3	CLUES-S	0.75	0.75	0.89	0.83	0.71
	3	(2)	PAM	0.56	0.56	0.77	0.76	0.59
	3	(3)	PAM	0.73	0.73	0.88	0.82	0.70
	3	(4)	PAM	0.65	0.65	0.85	0.76	0.60
	3	(2)	K-means	0.54	0.54	0.76	0.75	0.57
	3	(3)	K-means	0.73	0.73	0.88	0.82	0.70
	3	(4)	K-means	0.61	0.62	0.84	0.73	0.57
Vowel	11	9	CLUES-C	0.42	0.43	0.89	0.48	0.31
	11	9	CLUES-S	0.42	0.43	0.89	0.48	0.31
	11	(11)	PAM	0.38	0.39	0.90	0.44	0.28
	11	(11)	K-means	0.34	0.35	0.88	0.41	0.25
	11	(9)	PAM	0.41	0.42	0.89	0.47	0.31
	11	(9)	K-means	0.36	0.37	0.87	0.44	0.27

g: true number of clusters; $\hat{g}$: estimated or specified number of clusters.
HA index: Hubert and Arabie's adjusted Rand index; MA index: Morey and Agresti's adjusted Rand index; FM index: Fowlkes and Mallows index.

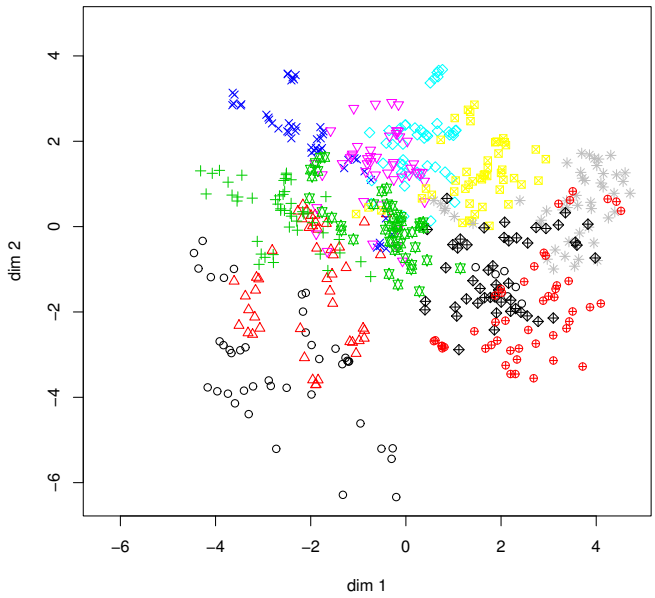

Figure 1: True partition of Vowel in the optimal 2-D subspace.

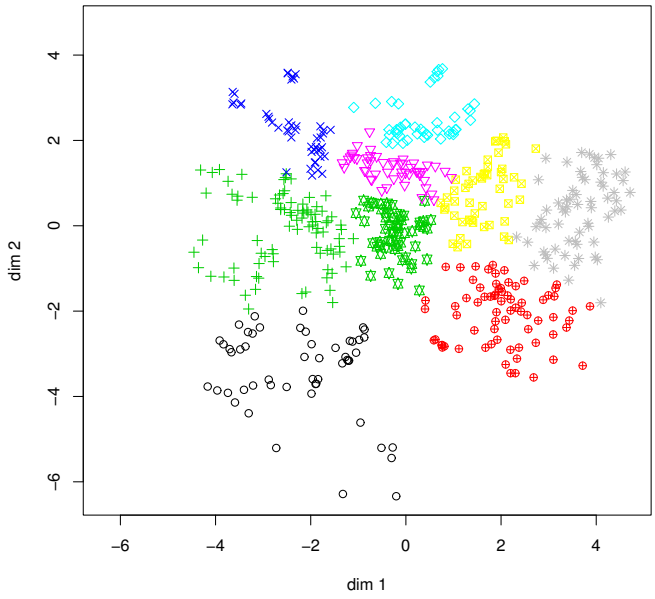

Figure 2: Optimal Partition of the Vowel data set.

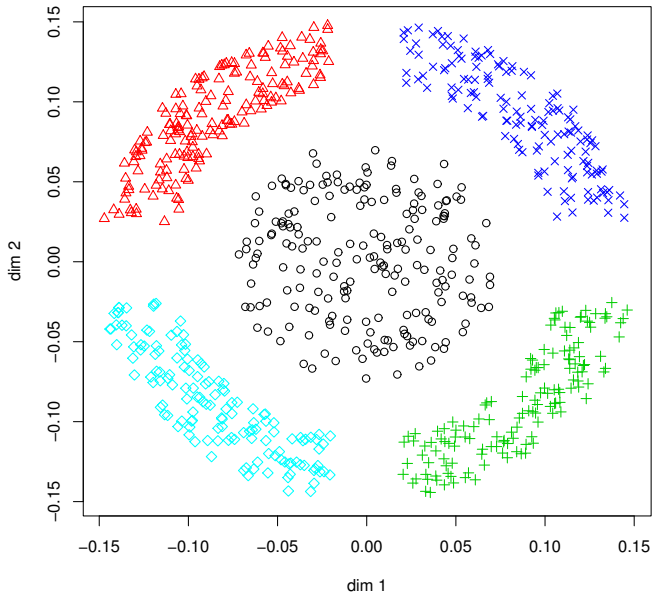

Figure 3: True partition and the partition by CLUES-Silhouette

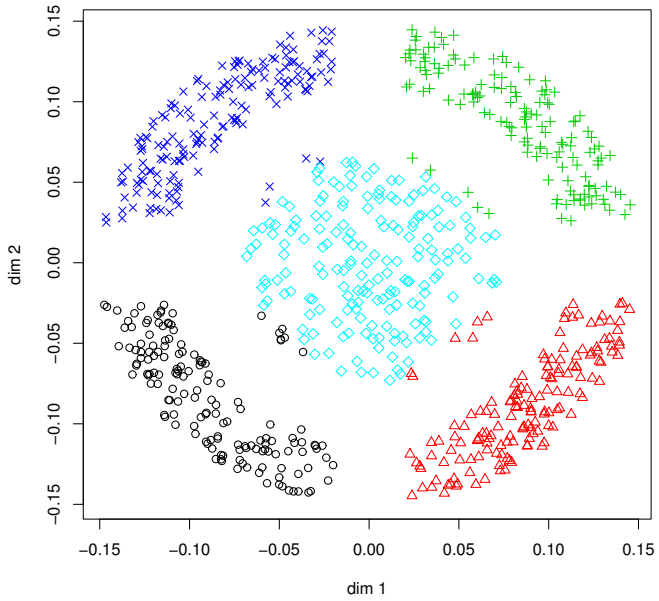

Figure 4: Partition of the Broken Ring data set by PAM

Table 2: Results for the Broken Ring data set

data set	g	$\hat{g}$	method	HA	MA	Rand	FM	Jaccard
Broken Ring	5	22	CLUES-C	0.34	0.35	0.85	0.50	0.25
	5	5	CLUES-S	1	1	1	1	1
	5	(4)	PAM	0.60	0.60	0.86	0.69	0.53
	5	(5)	PAM	0.89	0.89	0.96	0.91	0.84
	5	(6)	PAM	0.83	0.83	0.95	0.97	0.76
	5	(4)	K-means	0.60	0.60	0.85	0.69	0.52
	5	(5)	K-means	0.88	0.88	0.96	0.90	0.83
	5	(6)	K-means	0.80	0.80	0.93	0.84	0.72

g: true number of clusters; $\hat{g}$: estimated or specified number of clusters.
HA index: Hubert and Arabie's adjusted Rand index; MA index: Morey and Agresti's adjusted Rand index; FM index: Fowlkes and Mallows index.