The Effects of Ties on Convergece in K-Modes Variants for Clustering Categorical Data

N. Orlowski, D. Schlorff,
J. Blevins, D. Cañas, M. T. Chu[*], R. E. Funderlic[†]
N.C. State University
Department of Computer Science

June 29, 2004

Abstract

Clustering methods for categorical data can easily have ties. These ties occur when an ambiguity arises in the process of executing an algorithm. This paper identifies two types of ties and studies their effect on the k-modes method for categorical data. Three variants of the k-modes algorithm, each of which handles tie breaking and stopping criterion differently, are compared. It is shown via simple yet subtly constructed examples that the convergence and quality of results can be greatly affected by how the ties are resolved. The notion of cluster death is also discussed.

1 Introduction

Employing traditional methods such as k-means and spherical k-means [2], partitional data clustering has typically focused on data sets which contain numerical values [7]. More recently, considerable efforts have been on clustering data with categorical or qualitative attributes. The k-modes algorithm [6] is an iterative scheme that partitions categorical data vectors into homogeneous groups, called *clusters*, based on a similarity measure. Apparently, the term "k-modes" was coined by Huang [5], who adapted the method from a clustering method that works on mixed categorical and numerical data [4, 6], originally conceived by Ralambondrainy [9].

The main focus of the paper is an examination of how stopping criteria and tie-breaking methods affect convergence of any k-modes variant. Two types of ties are introduced and three variants of k-modes using different tie-breaking

[*]Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205(Chu@math.ncsu.edu) NSF grants: CCR-6204157, DMS-0073056

[†]Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8205(ref@csc.ncsu.edu)

policies are constructed to demonstrate the effects of those ties. We also introduce ways to prevent the phenomenon of the so-called *cluster death* where a cluster has no associated data vectors. The purpose of this paper is not to present calculations on large data sets or to empirically evaluate a particular variant but to provide an in-depth understanding of the specific effect that ties have on convergence in any k-modes variant.

Let $\mathbb{P}$ denote a collection of categorical attributes. The attribute can be any type of object, so long as any two attributes in $\mathbb{P}$ can always be compared to determine whether or not they are the same. In practice, it is often the case that observations of empirical data are ordered n-tuples with attributes from $\mathbb{P}$. For convenience, we denote such a *categorical data vector* $\mathbf{x}$ as a column vector in the space $\mathbb{P}^m$ where m is the number of attributes comprising the m-tuple. A given subset X of n data vectors in $\mathbb{P}^m$ can therefore be considered as an $m \times n$ matrix with entries from $\mathbb{P}$ where the ordering of columns is immaterial

A k-means-like algorithm partitions a given set of data vectors in $\mathbb{P}^m$, X, into k sets or "clusters" $X_1, \ldots, X_k$, where k is the specified number of clusters. Each cluster X_i is a collection of columns of X and has an associated mode(center) vector $\mathbf{c}_i$, $i = 1, \ldots, k$, that will be defined later. In order to classify data into clusters, we need the notion of similarity. There are many ways to define similarity with categorical data [10]. We will define the *similarity function* $s(\mathbf{x}, \mathbf{y})$ between two categorical data vectors $\mathbf{x}$ and $\mathbf{y}$ as

$$s(\mathbf{x}, \mathbf{y}) := \#\text{ of matches between components of } \mathbf{x} \text{ and } \mathbf{y}. \tag{1}$$

Given any subset $Z \subset X$, we then define the *total similarity* $t(Z, \mathbf{y})$ of a vector $\mathbf{y}$ to the set Z as the sum of the similarities between $\mathbf{y}$ and each element in the set Z, that is,

$$t(Z, \mathbf{y}) := \sum_{\mathbf{z}_i \in Z} s(\mathbf{z}_i, \mathbf{y}). \tag{2}$$

Huang [6] defines the *mode vector* $\mathbf{c}_i$ of a cluster X_i as the vector in $\mathbb{P}^m$ which minimizes the dissimilarity with all vectors in the cluster X_i. In our context, we equivalently define the mode vector $\mathbf{c}_i$ of a cluster X_i as the vector that maximizes the similarity in the cluster, that is,

$$\mathbf{c}_i := \arg\max_{\mathbf{y} \in \mathbb{P}^m} t(X_i, \mathbf{y}). \tag{3}$$

This can be equivalently thought of as minimizing the dissimilarity between clusters. The relationship between similarity and dissimilarity can generally be related by

$$d(\mathbf{x}, \mathbf{y}) = m - s(\mathbf{x}, \mathbf{y}), \tag{4}$$

where m is the number of attributes in the categorical data vector and

$$d(\mathbf{x}, \mathbf{y}) := \#\text{ of mismatches between components of } \mathbf{x} \text{ and } \mathbf{y}. \tag{5}$$

Generally speaking, components of the mode vector $\mathbf{c}_i$ can be calculated by selecting the attribute that appears most frequently in each row of the vectors

in X_i. For example, if

$$X_i = \begin{bmatrix} car & car & boat & bike \\ bike & plane & train & train \end{bmatrix},$$

then

$$\mathbf{c}_i = \begin{bmatrix} car \\ train \end{bmatrix}$$

is the mode vector of X_i.

It is natural to define the overall objective function of a given partition $X_1, \ldots, X_k$ of X and a set of mode vectors $\mathbf{c}_1, \ldots, \mathbf{c}_k$ as the sum

$$T(X_1, \ldots, X_k, \mathbf{c}_1, \ldots, \mathbf{c}_k) := \sum_{i=1}^{k} t(X_i, \mathbf{c}_i). \tag{6}$$

This value serves as a measurement of the *coherence*, or *total similarity* of the current clusters in the data set X.

Given a data set $X \subset \mathbb{P}^m$, the objective of a k-modes algorithm is to maximize the coherence $T(X_i, \ldots, X_k, \mathbf{c}_1, \ldots, \mathbf{c}_k)$ among all possible partitions $X_1, \ldots, X_k$ and their respective mode vectors $\mathbf{c}_1, \ldots, \mathbf{c}_k$. In this paper, we report our study showing that among the various existing k-modes algorithms, the tie breaking mechanism may make a difference in their convergent behavior.

An example application studied by Morgan [8] might elucidate the basic paradigm in a k-modes method.

Example 1.1 *Consider the scenario that a company sells electronic cabinets which contain m slots. When making an order for a cabinet, the customer specifies a type of circuit board to be inserted into each slot. Each order can be represented by a categorical data vector $\mathbf{x}$ whose components are selected from the categorical attributes in $\mathbb{P}$ of board types. For simplicity, we shall assume two different components with the same attribute are allowed. Assume that board insertion is a time-consuming process. The company desires to minimize their response time for business reasons. It is reasonable to analyze the past order history and keep a number of pre-configured models (mode vectors) in stock. The idea is that when an order comes in a pre-configured model can be modified to match the order exactly with as few board modifications as possible. Let $X \subset \mathbb{P}^m$ denote the collection of n past orders. By applying a k-modes algorithm to X, k mode vectors can be obtained which serve as pre-configured models to reduce total production time as much as possible.*

Similar ideas can be found in *market basket data analysis* [12] where each supermarket transaction is counted as a categorical data vector and clusters are found for the purposes of market research. These paradigms should provide a good vocabulary and be helpful in understanding the fundamental consequences of ties. Most of the k-modes variants proposed in the literature either terminate the iteration whenever a tie occurs or ignore the tie. It is our intention to study tie-breaking mechanisms over categorical data more carefully and thoroughly in

this paper. This paper is organized as follows. We begin in section 2 with the distinction between two different types of ties in any k-modes algorithm. How these ties are broken affects the convergence behavior of a k-modes variant. To demonstrate this idea, we introduce in section 3 three example variants of the k-modes method. The specific effects of ties and stopping criteria on these three k-modes variants are discussed in section 4 with examples. The notion of cluster death is described in section 5.

2 Ties in k-modes Methods

Working with vectors of floating point numbers does not provide much of an opportunity to encounter the ambiguity of ties because ties seldom occur in floating point arithmetic (There is, of course, the remote possibility of inexact ties caused by rounding errors, where .90005 = round(.900049)). With categorical/qualitative data, however, ties are not rare occurrences. How a particular k-modes variant deals with ties can affect the final results obtained. In any k-modes algorithm, there are two places where ties are possible — these are during mode calculation and while determining to which cluster a particular vector should be partitioned. In this section, we are merely introducing the notion of ties. For a better understanding of where ties fit into the greater k-modes, refer to section 3

2.1 Ties in Mode Calculation

In any k-modes variant a mode must be computed for a set of categorical data vectors. Due to the categorical nature of data in k-modes, there may be more than one optimum $\mathbf{c}$ for a given set. We call this situation a *mode tie*, or a *tie in the mode operation*. Note that the mean of a given set of values (if defined) is unique, while the mode is not necessarily unique.

Example 2.1 *Let the matrix*

$$X = \begin{bmatrix} car & car \\ boat & plane \end{bmatrix}$$

denote a given set of two categorical data vectors. Trivially, possible mode vectors would include

$$\begin{bmatrix} car \\ boat \end{bmatrix} \text{ and } \begin{bmatrix} car \\ plane \end{bmatrix}.$$

While "car" is the only choice for the first component, we can make no definite decision about whether to choose "boat" or "plane" for the second component of $\mathbf{c}$ *in the above example. Indeed, "boat" and "plane" are equally good choices for the second component.*

To expand on this point about mode ties, imagine the worst case where a cluster of n data vectors has p distinct attributes at each component and each

attribute occurs with the same frequency. In this situation, each mode vector could be obtained by breaking p ties in n ways and there would be n^p different equally optimal mode vectors.

Example 2.2 *If a set of data is represented by*

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix},$$

then there would be 3^2 *possible ways to create a mode vector, all of which would be equally optimal.*

There are non-random ways to resolve mode ties, but that will be a topic for further research. Some include using the last known unambiguous model. Following most of the k-means variants proposed in the literature, we will assume that mode ties are broken randomly, that is, a mode vector is randomly chosen from the set of equally optimal mode vectors.

2.2 Ties in Cluster Assignment

Each iteration in a k-modes method consists of two basic steps, mode calculation and cluster assignment. In the cluster assignment, data vectors are clustered on the basis of their "closeness" to the mode vectors. In general, a data vector is assigned to the cluster whose mode vector is most similar to that data vector. Ambiguity arises if that data vector is equally similar to more than one mode vector. As such, there is the possibility of a tie occurrence in the cluster assignment step.

Example 2.3 *If*

$$\mathbf{x} = \begin{bmatrix} car \\ boat \end{bmatrix}, \quad \mathbf{c}_1 = \begin{bmatrix} car \\ plane \end{bmatrix}, \quad \textit{and} \quad \mathbf{c}_2 = \begin{bmatrix} car \\ train \end{bmatrix},$$

then

$$s(\mathbf{x}, \mathbf{c}_1) = s(\mathbf{x}, \mathbf{c}_2) = \textit{1 match.}$$

We call this situation an *allocation tie*. The clusters obtained are affected by how such ties are broken and thus into which cluster $\mathbf{x}$ is assigned.

Depending on how these two types of ties are broken, the effects that these ties can have on convergence are significant. We shall detail our studies in section 4.

3 Variants of k-modes Methods

In partitional cluster analysis there exists a class of algorithms whose members differ largely in the way the similarity between two data objects is measured [1]. These algorithms include k-means [3], spherical k-means [2], k-modes and k-prototypes [4, 6] and many others [9, 8]. Each algorithm can be considered as

a descendant of the general archetype of k-means-like algorithms in that each produces a partitioning based on a given data set, a specified k, and a particular similarity function.

For the purposes of this paper, we define a variant of k-modes by specifying the stopping criterion and how ties are handled. We will examine three variants of the k-modes algorithm in this section and use them as examples to demonstrate the effect of ties later.

3.1 Cluster Variant

The first variant we will study is based on Huang's original k-modes variant. We borrow from Huang's algorithm the way his variant breaks allocation ties and the stopping criterion. However, we did not adopt Huang's decision of when to re-calculate mode vectors. Huang's k-modes variant re-calculates mode vectors every time a vector is moved. In contrast, our variant only calculates mode vectors once per full iteration. We call this variant the *cluster variant* because it terminates when all k clusters become invariant.

Algorithm 3.1 (**Cluster k-modes Method**) *Given a set X of categorical data vectors in $\mathbb{P}^m$ and a specified number k of desired clusters, do*

1. *Begin with an initial partition,*

$$X = X_1^{(0)} \cup X_2^{(0)} \cup \cdots \cup X_k^{(0)}.$$

2. *Update the modes of each cluster according to (3) to obtain*

$$\mathbf{c}_1^{(t)}, \mathbf{c}_2^{(t)}, \ldots, \mathbf{c}_k^{(t)}.$$

3. *Re-test the similarity of all data vectors from cluster to cluster with each mode vector in the following way. If a vector from $X_i^{(t)}$ is found to be strictly nearer to $\mathbf{c}_j^{(t)}$ than to the current $\mathbf{c}_i^{(t)}$, reallocate that vector to the cluster $X_j^{(t+1)}$ to obtain a new partition*

$$X = X_1^{(t+1)} \cup X_2^{(t+1)} \cup \cdots \cup X_k^{(t+1)}.$$

   Notice that ties here are biased so that the mode of a data vector's current cluster is preferred.

4. *Go back to Step 2 and repeat the iteration until no object has changed cluster assignment after full cycle of the whole data set, that is,*

$$X_i^{(t+1)} = X_i^{(t)}, \quad \textit{for all } i = 1, \ldots, k.$$

3.2 Center Variant

The second algorithm, called the *center variant*, differs from the cluster variant only in the stopping criterion.

Algorithm 3.2 (**Center k-modes Method**) *Given a set X of categorical data vectors in $\mathbb{P}^m$ and a specified number k of desired clusters, do*

1. *Begin with an initial partition,*

$$X = X_1^{(0)} \cup X_2^{(0)} \cup \cdots \cup X_k^{(0)}.$$

2. *Update the modes of each cluster according to (3) to obtain*

$$\mathbf{c}_1^{(t)}, \mathbf{c}_2^{(t)}, \ldots, \mathbf{c}_k^{(t)}.$$

3. *Re-test the similarity of all data vectors from cluster to cluster with each mode vector in the following way. If a vector from $X_i^{(t)}$ is found to be strictly nearer to $\mathbf{c}_j^{(t)}$ than to the current $\mathbf{c}_i^{(t)}$, reallocate that vector to the cluster $X_j^{(t+1)}$ to obtain a new partition*

$$X = X_1^{(t+1)} \cup X_2^{(t+1)} \cup \cdots \cup X_k^{(t+1)}.$$

   As in the Cluster Variant, allocation ties here are biased so that the mode of a data vector's current cluster is preferred.

4. *Go back to Step 2 and repeat the iteration until no mode vector has changed upon re-calculation of centers after full cycle of the whole data set, that is,*

$$\mathbf{c}_i^{(t+1)} = \mathbf{c}_i^{(t)}, \quad \textit{for all } i = 1, \ldots, k.$$

Because of the biased ties that occur in the Cluster and Center variants, it is possible to have a secondary type of allocation tie. This secondary type of tie occurs when a data vector, x, is in the cluster whose mode is c_3, but is closer to two different modes, namely c_1 and c_2. The normal decision to bias x towards its current cluster cannot be applied here as c_1 and c_2 are strictly better choices than c_3. So, as with the mode ties, for the purposes of this paper we will assume such ties are broken randomly.

3.3 Coherence Variant

The third variant of the k-modes algorithm is analogous to Dhillon's spherical k-means algorithm discussed in [2], except that we are dealing with categorical data and that the similarity is defined by the number of matches as in (5). The resulting scheme is referred to as the *coherence variant*.

Algorithm 3.3 (**Coherence k-modes Method**) *Given a set X of categorical data vectors in $\mathbb{P}^m$, a specified number k of desired clusters, and tolerance $\epsilon > 0$, do*

1. *Begin by specifying initial clusters.*

$$X = X_1^{(0)} \cup X_2^{(0)} \cup \cdots \cup X_k^{(0)}.$$

2. *Compute mode vector for each cluster according to (3) to obtain a mode vector for each cluster,*

$$\mathbf{c}_1^{(t)}, \mathbf{c}_2^{(t)}, \ldots, \mathbf{c}_k^{(t)}.$$

3. *Allocate each data vector from X to the cluster whose mode is most similar to obtain the new clusters,*

$$X = X_1^{(t+1)} \cup X_2^{(t+1)} \cup \cdots \cup X_k^{(t+1)}.$$

Break allocation ties, if any, randomly.

4. *Go back to Step 2 and repeat the iteration until the change in the objective function is less than a certain threshold, that is,*

$$|T(X_1^{(t+1)}, \ldots, X_k^{(t+1)}, \mathbf{c}_1^{(t+1)}, \ldots, \mathbf{c}_k^{(t+1)}) - T(X_1^{(t)}, \ldots, X_k^{(t)}, \mathbf{c}_1^{(t)}, \ldots, \mathbf{c}_k^{(t)})| < \epsilon, \quad (7)$$

for $\epsilon \geq 0$ [2].

It is important to note that, unlike the cluster or the center variants where each data vector stays in its cluster until a better cluster is found, the coherence variant effectively dissolves the clusters during each iteration and re-allocates *every* data vector in X. Additionally, the coherence variant terminates when the absolute change in the objective function is below a certain threshold.

We choose to compare these three variants of the k-modes method because they have different convergence criteria and handle ties differently during data vector assignment. The variants are summarized in Table 1. We shall examine the different behavior in the next few sections.

We stress that in the k-means-like methods designed for continuum data, unique centers(means) can be obtained from clusters and vice versa in exact arithmetic. This is not the case in k-modes methods, especially where there are ties. Given k mode vectors, there is not necessarily a unique partitioning corresponding to each mode vector. For this reason, we start each variant with prescribed initial clusters.

We have provided figure 1 so that the reader can develop a further understanding of the three variants. To use this figure, choose a variant and iterate per the algorithm corresponding to that variant, starting at node (a). All options for all variants are displayed in the figure. As you trace your route, the places where ties occur and the consequences of how they are resolved will become evident. The first 5 nodes (a-e) might be visited no matter what variant is being used. However, in order to get to step (f), the allocation tie in (e) must be broken randomly, i.e., the coherence variant must be used.

There are many important observations to be made here:

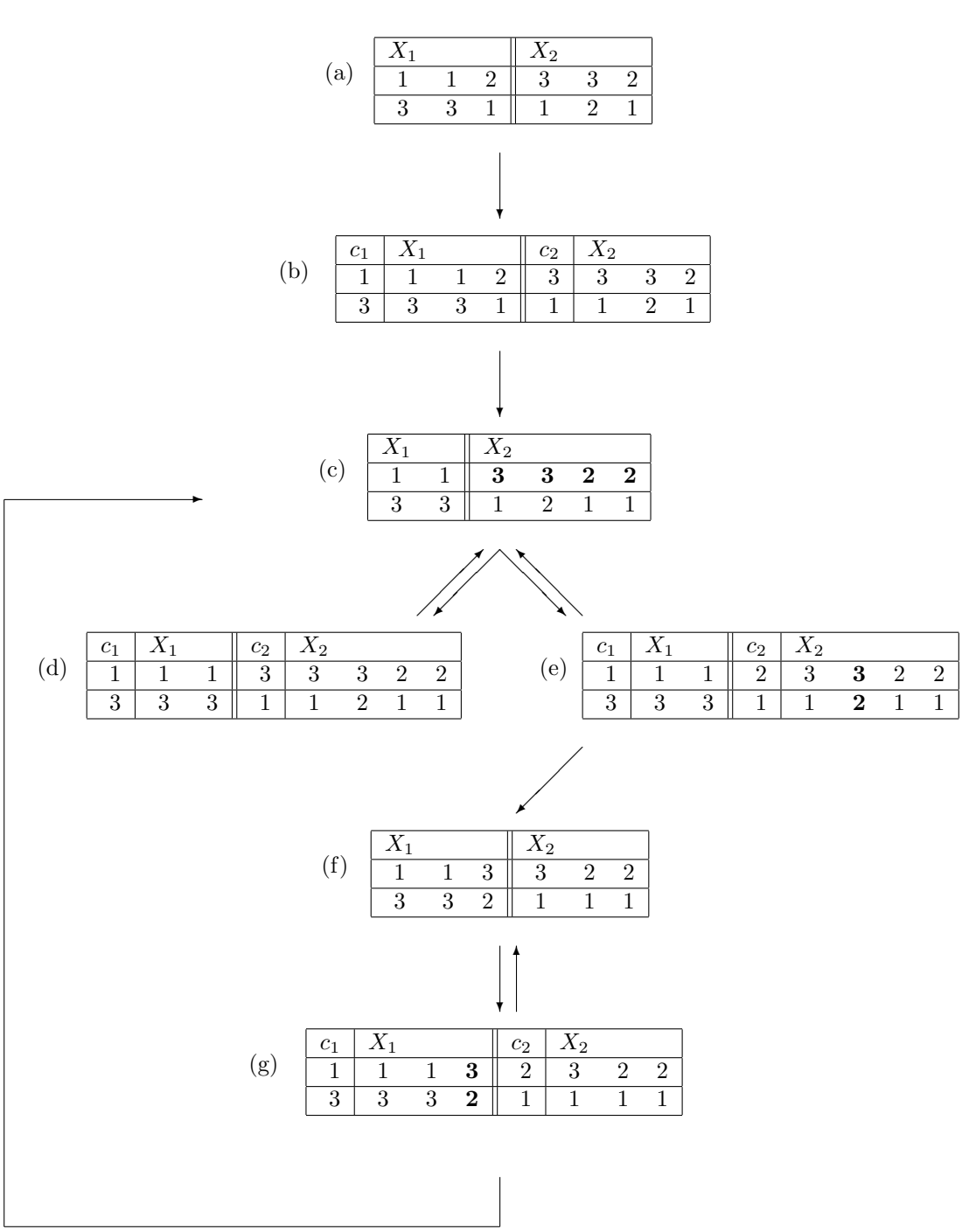

Figure 1: Sample run of a generic k-modes variant (ties are in bold)

Variant	Stopping Criterion	Mode Ties	Allocation Ties
Cluster	$X_i^{(t+1)} = X_i^{(t)}, i = 1, \ldots, k$	random	biased
Center	$\mathbf{c}_i^{(t+1)} = \mathbf{c}_i^{(t)}, i = 1, \ldots, k$	random	biased
Coherence	$\lvert T^{(t+1)} - T^{(t)} \rvert < \epsilon$	random	random

Table 1: Comparison of k-modes Variants.

1. At (c), the first component of the mode vector must be selected randomly from $\{3, 2\}$.
2. At (d), the only option is to return to (c) so the mode tie remains.
3. The step from (c) to (e) occurs if the tie in (c) is randomly selected to be $\{3\}$.
4. From (e), we can repartition two ways. If we use the Cluster/Center variants with biased allocation ties, we must go back to (c). If we allow random allocation ties, we might advance to (f).
5. If random allocation ties are allowed, we can go from (g) to (c).
6. The objective function can be calculated only when mode vectors are defined. Notice that the objective function is the same at (e), (d), and (g).

Possible termination criteria:
Depending on which variant is chosen, its stopping criterion will be met if the following nodes are visited in order. This list is not necessarily comprehensive.

- Cluster Variant:

  (c) (d) (c)

  (c) (e) (c)

  (f) (g) (f)

- Center Variant

  (d) (c) (d)

  (e) (c) (e)

- Coherence Variant

  (c) (d) (c)

  (c) (e) (c)

  (c) (e) (c)

  (e) (g) (c)

4 The Effects on Convergence

We now begin to examine how the specific allocation tie resolution policies and the stopping criterion of each variant affect the convergence of each of the three variants. We wish to highlight the effect on the quality of results and robustness of convergence by simple examples. Despite its simplicity, the concern addressed in this section seems to have been overlooked in the literature.

4.1 Convergence of the Coherence Variant

In the coherence variant, if a data vector is equally close to more than one mode vector, that data vector is allocated to a cluster selected *randomly* from those whose modes are equally close to the object. This randomness can create an interesting anomaly in the ascent of the objective function.

Observation 4.1 *It is possible for the objective function to remain constant from one iteration to the next during the course of the algorithm, only to have an increase on the third iteration.*

A simple example can best illustrate the scenario. We caution readers not to underestimate the simplicity of the following example, as ties are not rare occurrences in categorical data.

Example 4.1 *Consider the data set*

$$X = \begin{bmatrix} 1 & 2 & 1 & 2 \\ 2 & 1 & 2 & 1 \end{bmatrix}$$

where we use integers to describe the different categorical objects. One possible initial partitioning is

$$X_1^{(0)} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \quad \textit{and} \quad X_2^{(0)} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}.$$

Clearly there are mode ties in both $X_1^{(0)}$ and $X_2^{(0)}$. It is possible that the mode vectors

$$\mathbf{c}_1^{(0)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \textit{and} \quad \mathbf{c}_2^{(0)} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

are selected, leading to

$$T(X_1^{(0)}, X_2^{(0)}, \mathbf{c}_1^{(0)}, \mathbf{c}_2^{(0)}) = 4.$$

With these initial mode vectors, we go on to Step 3 in Algorithm 3.3 and repartition X. Keeping in mind that because allocation ties are randomly allocated, this partitioning is not unique given $\mathbf{c}_1^{(0)}$ and $\mathbf{c}_2^{(0)}$. It is possible that

$$X_1^{(1)} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \quad \textit{and} \quad X_2^{(1)} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

are selected and, by the random selection on the mode ties again, the mode vectors are chosen to be

$$\mathbf{c}_1^{(1)} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \quad \textit{and} \quad \mathbf{c}_2^{(1)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

It follows that

$$T(X_1^{(1)}, X_2^{(1)}, \mathbf{c}_1^{(1)}, \mathbf{c}_2^{(1)}) = 4.$$

Because the values of the objective function have not changed, convergence would be assumed using Algorithm 3.3. However, suppose we were to continue with another iteration. It is possible to increase the value of the objective function substantially and, hence, obtain a better clustering. Indeed, it is entirely possible that by random selection in breaking the allocation tie in Step 3 we end with the partition

$$X_1^{(2)} = \begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix} \quad \textit{and} \quad X_2^{(2)} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}.$$

This time, the new mode vectors will certainly be

$$\mathbf{c}_1^{(2)} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad \textit{and} \quad \mathbf{c}_2^{(2)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

It follows that

$$T(X_1^{(2)}, X_2^{(2)}, \mathbf{c}_1^{(2)}, \mathbf{c}_2^{(2)}) = 8,$$

the maximal achievable objective value.

In the above example, the coherence variant would have stopped after the first iteration by the stopping criterion. However, as we have demonstrated, if we continue the iteration, we can further maximize the objective function. What is really happening between the first and second iterations is that at least one data vector swaps between clusters whose mode vectors are equally similar to that data vector. This swap is advantageous to the next iteration.

Dhillon has shown that Algorithm 3.3 applied to non-categorical data will converge, but acknowledges that the clusters may not [2]. The idea of Dhillon's proof can be easily applied to prove the convergence of the coherence variant. On the other hand, we have shown above that just because an iterative clustering procedure meets its stopping criteria does not guarantee that the algorithm has reached a local optimum. Another example can be found in [11]. The situation described in [] is an allocation tie and is resolved by resorting to what is called fuzzy clustering, but that situation might also be solved by imparting some sort of allocation tie resolution policy.

In the coherence variant, it is precisely because of random tie-breaking during re-clustering that the anomaly in the previous example occurs. There are other ramifications of random tie breaking, including what we call *Cluster Death* which is discussed in section 5.

It is important to realize that the nature of categorical data makes it possible for the objective function to stay exactly constant from one iteration to the next. For continuum data such as normalized data vectors described by Dhillon [2], it is far less likely that the objective function will stay exactly constant from one iteration to the next.

4.2 Convergence of the Cluster Variant

In contrast to the coherence variant, Step 3 in the cluster variant effectively *biases* allocation ties such that if an object is equally close to two centers, that object stays in its current cluster. The clusters of this variant have been proven to converge by Morgan [8]. The effect of the tie-breaking mechanism specified in Step 3 of Algorithm 3.1 on the iterative path is that all else being equal, that is, no mode ties and given the same starting conditions, the cluster variant will always iterate through the same path and produce the same final clusters.

We pointed out earlier that one drawback to the coherence k-modes variant is that the clusters may not converge. The same behavior has been observed in the context of k-means [2] applied to continuum data. As it may be more desirable in practice to guarantee convergence of clusters rather than to obtain an optimal objective value, it is tempting to bias allocation ties in a k-means implementation for continuum data as in the cluster variant for categorical data. In exact arithmetic, the convergence of such a scheme can be proved in exactly the same way as that done in [8]. However, we hasten to point out that the finite precision of a computer can bring up another concern.

Observation 4.2 *Roundoff error can cause an anomaly where a data vector indefinitely swaps between two clusters, thus never satisfying the stopping criterion.*

Again, we use a simple example to illustrate our point.

Example 4.2 *Suppose we have a computer with two digit precision to operate on the data set*

$$X = [.65, .67, .68]\,.$$

Assume that the initial clusters are

$$X_1^{(0)} = [.65, .67] \quad \textit{and} \quad X_2^{(0)} = [.68]\,,$$

and that the machine does rounding.

*The new centers(means) for each cluster are obtained by averaging the elements within the cluster. However we must keep in mind that we only have **two** digits of precision, so*

$$\mathbf{c}_1^{(0)} = \frac{round(.65 + .67)}{2} = \frac{round(1.32)}{2} = \frac{1.3}{2} = .65 \quad \textit{and} \quad \mathbf{c}_2^{(0)} = .68.$$

The next iteration produces the clusters

$$X_1^{(1)} = [.65] \quad \textit{and} \quad X_2^{(1)} = [.67, .68]\,.$$

Again we re-calculate the means and obtain

$$\mathbf{c}_1^{(1)} = .65 \quad \textit{and} \quad \mathbf{c}_2^{(1)} = \frac{round(.67 + .68)}{2} = \frac{round(1.35)}{2} = \frac{1.4}{2} = .7.$$

With these new means, we re-cluster the data set, only to realize that we are back to where we started.

$$X_1^{(2)} = [.65, .67] \quad \textit{and} \quad X_2^{(2)} = [.68]\,.$$

In this example, at least one cluster will change at every iteration, so convergence of the clusters will never be achieved. A more complicated example can be given for a binary computer with standard rounding using the principle that a very small number plus a very large number will return the larger one in finite binary arithmetic.

4.3 Convergence of the Center variant

Recall that the center variant handles allocation ties in exactly the same way as the cluster variant. The only difference comes in the stopping criterion. The sequence of clusters will still converge in the center variant in the same way as in the cluster variant.

Observation 4.3 *Because of the possibility of mode ties, the convergence of clusters does not necessarily imply the convergence of modes.*

We have demonstrated in section 4.1 that a mode vector is not necessarily unique to a given cluster and that a cluster is not unique to a particular mode vector. This presents a problem in the convergence of the modes of this variant. Because it is often possible to calculate an arbitrary number of modes with the same set X and never have two consecutive equal modes, convergence of modes cannot be proved.

5 Cluster Death

In all three variants of the k-modes method, it is essential to re-assign data vectors into clusters at every iteration. Note that k, the number of clusters, is predetermined. During the process, however, the iteration may break down due to a situation which we refer to as the *cluster death*.

Observation 5.1 *There is a possibility that upon data vector assignment, no vector is assigned to a particular cluster. The mode of this cluster is therefore indeterminate.*

In this section we want to show that the way in which allocation ties are broken and the decision of when to calculate modes can allow or disallow cluster death. Again, we will use the coherence and the cluster k-modes variants to describe how this is done.

We first consider the cluster death in the coherence variant. Recall that the coherence variant assigns allocation ties randomly. The following example shows how random assignment can lead to an empty cluster. Again, we caution not to underestimate the importance of this simple example.

Example 5.1 *If*

$$X_1 \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix} \quad \textit{and} \quad , X_2 \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix},$$

then,

$$\mathbf{c}_1 \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \textit{and} \quad \mathbf{c}_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

The next partitioning, since assigned randomly might be

$$X_1 \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \end{bmatrix} \quad \textit{and} \quad X_2 \begin{bmatrix} \quad \end{bmatrix}.$$

In the final iteration of this example, X_2 has an indeterminate mode vector, so continuation of the algorithm is less meaningful. What cluster death most likely suggests is that the predetermined k is too large. There are some simple ways of correcting this situation, such as retaining the last relevant mode or generating a random mode vector. However, be aware that the cluster death indicated above represents the data set more truly. We also wish to suggest that there are benefits to cluster death as it may aid in improving the selection of the k value.

Cluster Death is also possible in the cluster variant due to mode ties.

Example 5.2 *Suppose we are given the following partition of X,*

$$X_1 = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad X_2 = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad \textit{and} \quad X_3 = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}.$$

The mode vectors might be

$$\mathbf{c}_1 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \quad \mathbf{c}_2 = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, \quad \textit{and} \quad \mathbf{c}_3 = \begin{bmatrix} 2 \\ 3 \end{bmatrix},$$

respectively. The subsequent iteration would produce the clusters,

$$X_1 = \begin{bmatrix} 1 & 1 & 1 \\ 3 & 3 & 3 \end{bmatrix}, \quad X_2 = \begin{bmatrix} 2 & 2 & 2 \\ 4 & 4 & 4 \end{bmatrix}, \quad \textit{and} \quad X_3 = \begin{bmatrix} \quad \end{bmatrix}.$$

We point out that in Huang's k-modes algorithm, the modes are updated each time a data vector changes clusters, not simply at each iteration. This modification prevents cluster death from occurring as it guarantees that a cluster with only one data vector has a mode that is identical to that data vector. There are two main aspects of Huang's k-modes algorithm that make it immune to cluster death:

1. Biased allocation tie resolution.

2. Recalculating the modes of the destination and origin clusters each time a data vector is re-allocated.

Cluster death can be prevented in any k-modes algorithm if these two steps are taken.

6 Conclusion

The mode calculation and the vector assignment are the two major components in most k-modes variants. In both operations the issue of ties arises frequently when dealing with categorical data. Using three variants of the k-modes algorithm, each of which handles ties and stopping criteria differently, we demonstrate the effect of the tie-breaking policy on the behavior of the underlying method. Convergence for the cluster variant and the coherence variant has been known in the literature. However, we illustrate in some interesting details the effect including premature termination, cluster death, and infinite cycle.

7 Acknowledgments

Thanks to the following for their correspondence and assistance in writing this paper:

- I. Dhillon
- S. Morgan, Y. Fahti
- Z. Huang
- Anonymous Reviewers

Special thanks to the National Science Foundation for their generous grant.

References

[1] D. Cañas, D., M. Chu, and R. Funderlic: 2004, 'Unmasking the k-Means Clustering Method(Generalizations and Unification of Clustering Methods)'. *Preprint*.

[2] Dhillon, I. and D. Modha: 2001, 'Concept Decompositions for Large Sparse Text Data Using Clustering'. *Machine Learning* **42**, 143–175.

[3] Dubes, R. and A. Jain: 1975, 'Clustering Techniques: The users Dilemma'. *Pattern Recognition* **8**.

[4] Huang, Z.: 1997a, 'Clustering Large Data Sets with Mixed Numeric and Categorical Values'. In: *Proceedings of the First Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp.21-34, 1997.*

[5] Huang, Z.: 1997b, 'A fast clustering algorithm to cluster very large categorical data sets in data mining'. *SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (SIGMOD-DMKD'97), Tucson, Arizona.*

[6] Huang, Z.: 1998, 'Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values'. *Data Mining and Knowledge Discovery* **2**(3), 283–304.

[7] Jain, A., M. Murty, and P. Flynn: 1999, 'Data clustering: a review'. *ACM Computing Surveys* **31**(3), 264–323.

[8] Morgan, S. and Y. Fathi: 2001, 'Algorithms for the Model Configuration Problem'. *IIE Transactions* **36**, 169–180.

[9] Ralambondrainy, H.: 1995, 'A conceptual version of the K-means algorithm'. *Pattern Recognition Letters* **16**.

[10] Ryu, T. and C. Eick: 1998, 'A Unified Similarity Measure for Attributes with Set or Bag of Values for Database Clustering'. *Sixth International Workshop on Rough Sets, Data Mining and Granular Computing (RSDM-GrC'98).*

[11] S. Selim, S. and M. Ismail: 1984, 'K-means Type Algorithms: A Generalized Convergence Theorem and Characterzation of Local Optimality'. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**.

[12] Wang, K., C. Xu, and B. Liu: Kansas, Missouri,1999, 'Clustering Transactions Using Large Items'. In: *Proc. ACM CIKM.* pp. 483–490.