Convergence and Other Aspects of the k-modes Algorithm for Clustering Categorical data

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

March 25, 2004

Abstract

Three example variants of the k-modes algorithm are compared as tools to illustrate the effects of ties on convergence of any k-modes like algorithm. Two types of ties are discussed as well as their affect on the convergence of the 3 variants. These consequences of resolving these ties are shown to greatly affect speed of convergence and quality of results. Other aspects pertinent to the success of any k-modes method are discussed. These topics include: Cluster Death(empty clusters), the duality of the similarity/dissimilarity functions, starting with clusters or modes and restricting the domain of the mode vectors.

1 Introduction

Partitional Data clustering has typically focused on data sets which contain numerical values [**?**] using methods such as k-means and spherical k-means [1]. More recently, the focus has centered on clustering data with categorical, or, qualitative attributes. The k-modes algorithm [2]is one of iteratively partitions categorical data vectors into homogeneous groups called *clusters* based on a similarity measure. The term 'k-modes' was coined by Huang [3], who adapted the method from a clustering method that works on mixed categorical/numerical data [**?**] [**?**], given by Ralambondrainy [4].

The main focus of the paper is an examination of how stopping criteria and tie-breaking methods affect convergence of three example variants of k-modes. We have constructed these three variants with different tie-breaking policies in order to better demonstrate the effects of ties. Ways to prevent the phenomenon of Cluster Death, where a cluster has no data vectors associated with it, are also discussed for k-modes. 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. Our notation follows.

Let $\mathbb{I}$ denote a collection of categorical attributes. The attribute can be any type of object, so long as any two attributes in $\mathbb{I}$ can always be compared to determine whether they are the same or not. In practice, it is often the case

that observations or empirical data are ordered multiplets with attributes from ı. For convenience, we denote such a *categorical data vector* x as a column vector in the space $ı^m$ where m is the number of attributes comprising the multiplet. Given $\mathbb{P}^m$, let x denote a *categorical data vector*, where $x, x \in \mathbb{P}^m$, is a column vector of length m.

A k-modes-like algorithm partitions a given set X of data vectors into k sets or clusters $X_1, ..., X_k$, where k is the specified number of clusters. Each cluster has an associated mode(center) vector $\mathbf{c}_i$, $i = 1..k$. We will define the *similarity function* $s(x, y)$ between two categorical data vectors x and y, as

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

There are many ways to define similarity with categorical data[5]; we use the simple matching distance for this paper. We define the similarity of a vector y with a set, X, of n vectors as the sum of the similarities between y and each of the vectors in the cluster X.

$$s(X, y) := \sum_{i=1}^{n} s(x_i, y)\ , x_i \in\ X. \tag{2}$$

Huang[**?**] defines each *mode vector*, c_i as an m-vector which minimizes the dissimilarity with all vectors in the cluster X_i. In this paper, we equivalently define the mode vector as a vector that has maximum similarity with all vectors in the cluster.

$$c_i := \arg\max_{y}\ s(X_i, y). \tag{3}$$

Each component of the mode vector , c can be calculated by selecting the value that appears most frequently in each row of the vectors in X. For example, if

$$X = \left\{ \begin{array}{cccc} car & car & boat & bike \\ bike & plane & train & train \end{array} \right\},$$

then

$$c = \left[\begin{array}{c} car \\ train \end{array} \right].$$

We define the overall objective function Q of a given partition $X_1, \ldots X_k$ and a set of mode vectors $c_1, \ldots c_k$ as the sum of all matches between each mode vector and all of the data vectors in its corresponding cluster. This value also serves as a measure of coherence for all of the clusters in the data set.

$$Q(X_1..X_k, c_1..c_k) := \sum_{i=1}^{k}\ s(X_i, c_i). \tag{4}$$

Generally speaking, the objective of a k-modes algorithm is to maximize the values $Q(X_i, ..., X_k, c_1, ..., sc_k)$ among all possible $X_1, ..., X_k$ and $c_1..., c_k$.In this paper, we report our study showing that among the various existing k-modes algorithms, the tie breaker may make a difference in their convergent behavior.

This paper is organized as follows. First we will introduce our notation. Section 2 will discuss the possibility of ties in any k-modes algorithm. In Section 3, we will introduce our three example variants. In section 4 we will discuss the

specific effects that ties and stopping criteria have on convergence in each of the variants. The topic of section 7 is a numerical anomaly in the center and cluster variants when applied to numerical data. Section 8 will discuss the differences between starting by specifying clusters or centers. Section 9 is a discussion of the duality of matches and miss-matches in defining the similarity and dissimilarity functions. Section 10 is a look at the rather interesting phenomenon of cluster death. Finally, in section 12, we offer our concluding remarks.

An example of application by Morgan [6] might elucidate the basic paradigm in a k-modes method. We will outline it briefly. A company sells electronic cabinets that have a certain number of slots. When making an order for a cabinet, the customer specifies a type of circuit board to be inserted into each slot. Orders can be represented by a categorical data vector, x, whose m components represent slots. Each cabinet can be thought of as an m-vector whose elements represent the type of board in each slot There may be only one board inserted per slot, but there may be more than one board type in a particular cabinet(two different components with the same value are allowed).

Due to the testing involved, board insertion is a time-consuming process and the company wants to minimize their response time for business reasons. To do this, they decide to analyze the past order history and keep a number of pre-configured models (mode vectors) in stock. This way, when an order comes in, a pre-configured model can be modified to match the order exactly, with as few board insertions as possible by replacing the incorrect boards with ones that match the order (board removal time is negligible). of n customer orders. Let X denote a $m \times n$ matrix where each column represents one order. By applying a k-modes algorithm to X, the past order history, mode vectors can be obtained which serve as models to reduce production time as much as possible. Ultimately, each order is filled exactly to specification but the goal is to use k-modes to minimize the number of insertions required to fill each order.

This paradigm might be helpful in grasping our thoughts below and provide a convenient vocabulary. Similar ideas can be found in of Market Basket Data analysis [7][8], where each supermarket transaction is counted as a categorical data vector and clusters are found for the purposes of market research.

2 Ties in k-modes

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 possibility of inexact ties caused by rounding errors). With categorical/qualitative data, however, ties are not rare occurrences. How a particular k-modes method 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 comparing similarities between two pairs of objects.

2.1 Ties during Mode Calculation

In any k-modes method 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 c for a given set. We call this situation a *type-1* tie, or a *tie in the mode operation*. For example, consider the trivial set

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

of two categorical data vectors. Possible mode vectors include

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

We can make no definite decision about whether to choose 'boat' or 'plane' for the second component of c in the above example. Indeed, 'boat' and 'plane' are equally good choices for the second component, while 'car' is not.

To expand on this point about type 1 ties, imagine the worst case, where a cluster of n data vectors has m distinct components at each component. In this situation, each mode vector could be obtained by breaking m ties in n ways; there would be m^n different equally optimal mode vectors. Consider the following example where integers represent different categorical data entries. If

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

there would be 3^2 possible ways to create a mode vector, all of which would be equally optimal. Different similarity measures may include a scaling value, but the problem of ties is not completely alleviated, as ties are still far more frequent than when dealing with data in $\mathbb{R}^n$. There are non-random ways to resolve these ties,as the biasing of type 1 ties is a topic for further research. For the purposes of this paper, we will assume that type 1 ties are broken randomly; that a mode vector is randomly chosen from the set of equally optimal mode vectors.

2.2 Ties During Similarity Comparison

The second possibility of a tie comes when comparing similarities of pairs of vectors. This happens during the cluster re-assignment step of a k-modes algorithm. In this step, data vectors are clustered with the closest mode vector. Because a data vector is assigned to the cluster whose mode vector is most similar to that data vector, there is ambiguity if more than one modes are equally similar to that data vector. For example, if

$$x = \begin{bmatrix} car \\ boat \end{bmatrix},\ c_1 = \begin{bmatrix} car \\ plane \end{bmatrix} \text{ and } c_2 = \begin{bmatrix} car \\ train \end{bmatrix},$$

then

$$s(x, c_1) \ = \ s(x, c_2) \ = 1 \ match.$$

We call this situation a type-2 tie, or, a *tie during similarity comparison*. The clusters obtained are affected by how such ties are broken and thus, into which cluster x is placed. The effects that these two types of ties have on convergence is discussed in detail in section 4. Next, In the appendix, we have included a walk-thru of a sample data set with the three variants we will introduce.

3 Variants of k-modes

In partitional cluster analysis there exists a class of algorithms whose members differ largely in the way the similarity (distance) between two data objects is measured[9]. These algorithms include k-means[10], Spherical k-means[1],k-modes and k-prototypes[11],[2] and others [4],[**?**]. each algorithm can be considered as a decedent of the general archetype of k-means-like algorithms in that each produces a partitioning of a data set, given X, k and a particular similarity function.

By modifying the similarity function of a particular algorithm to accommodate a problem's data type, any of the members of this class of algorithms can be modified to work on any data type. We will call algorithms modified in this way *variants* . For example, a variant of k-modes is an algorithm whose similarity function has been changed to matches/mismatches as in eqn(1). to accommodate categorical data, while a variant of k-means is an algorithm whose similarity function has been changed to the euclidean distance. The main reason for defining variants in this way is that different clustering algorithms have different stopping criteria and handle ties differently, and it is easier to refer to these different specifications as *variants*. We will examine three variants of k-modes in the following sections. We will also discuss an anomaly in a variant of k-means in section 5.

3.1 Cluster Variant

The first variant we will study is based on Huang's original k-modes algorithm. We acknowledge that Huang first published the k-modes algorithm but consider his specific implementation a variant from the general archetype of k-modes. We borrow from Huang's algorithm the type 2 tie-breaking policy and the choice to start the algorithm by specifying mode vectors. However, we did not borrow Huang's decision of when to re-calculate mode vectors. Unlike Huang's k-modes algorithm, which re-calculates mode vectors every time a vector is moved, this variant only calculate mode vectors once per iteration. We call this variant the *Cluster Variant* because it terminates when the clusters have not changed.

1. Begin with k initial mode vectors, one for each cluster.
$$c_1^{(t)}, c_2^{(t)}, ..., c_k^{(t)}.$$
2. Allocate each data vector in X to the cluster whose mode vector is most similar to that data vector according to equation (1) to obtain the partitioning
$$X = X_1^{(t)} \cup X_2^{(t)} \cup ... \cup X_k^{(t)}.$$
3. Update the modes of each cluster according to equation (3) to obtain
$$c_1^{(t+1)}, c_2^{(t+1)}, .., c_k^{(t+1)}.$$
4. Re-test the similarity of all data vectors with each mode vector. If a vector is found such that it is nearest to the mode of a cluster other than its current one, reallocate that vector to the closer cluster to obtain
$$X = X_1^{(t+1)} \cup X_2^{(t+1)} \cup ... \cup X_k^{(t+1)}.$$

5. Repeat at 2 until no object has changed clusters after full cycle of the whole data set, such that

$$X_i^{(t+1)} = X_i^{(t)}, i = 1..k.$$

3.2 Center Variant

The second algorithm, called the *Center Variant* is similar to the Cluster Variant but has a different stopping criterion. The Center Variant will stop when no center object has changed upon re-calculation. The first four steps of the Center Variant are identical to those in the Cluster Variant. The only difference comes in the fifth and final step. Both the Center and Cluster Variants break type-2 ties in the same way.

1. See Cluster Variant for steps 1-4
2.
3.
4.
5. Repeat at 2. until no center has changed upon re-calculation of all centers i.e.

$$c_i^{(t+1)} = c_i^{(t)}, i = 1..k$$

.

3.3 Objective Function Variant

The third variant of k-modes we will examine is the *Objective Function Variant*, based on Dhillon's spherical k-means algorithm[1]. By changing the usual domain of the data from R^m to the categorical/qualitative domain and changing the similarity measure from *cosine similarity* to number of matches as in equation (1), we can refer to this modified spherical k-means algorithm as the *Objective Function Variant*. It is important to note that the Objective Function Variant terminates when the absolute change in the objective function is below a certain threshold. The algorithm follows. Notice that unlike the Cluster and Center Variants, where each data vector stays in its cluster until a better cluster is found, the Objective Function Variant effectively dissolves the clusters during each iteration and re-allocates every data vector in X. However, the number of comparisons does not vary amongst the three variants.

1. Begin by specifying initial clusters.

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

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

$$c_1^{(t)}, c_2^{(t)}, ...c_k^{(t)}.$$

3. Allocate each data vector from X to the cluster whose mode is most similar(breaking type-2 ties randomly) to obtain the new clusters,

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

4. Recompute the mode vector for each cluster to obtain new mode vectors,

$$c_1^{(t+1)}, c_2^{(t+1)}, ...c_k^{(t+1)}.$$

5. Repeat at 3 until the change in the objective function is less than a certain threshold, i.e.

$$|Q^{(t+1)} - Q^{(t)}| < \epsilon, \epsilon > 0.$$

The importance of choosing to compare these three variants of k-modes is that they have different convergence criteria, handle ties differently during data vector assignment and specify starting values differently. The next few sections will go into detail about each of these topics.

Variant	Stopping Criterion	Type-1 Tie Breaking	Type-2 Tie Breaking
Cluster	$X_i^{(t+1)} = X_i^{(t)}, i = 1..k$	random	stays in cluster
Center	$c_i^{(t+1)} = c_i^{(t)}, i = 1..k$	random	stays in cluster
Objective Function	$\|Q^{(t+1)} - Q^{(t)}\| < \epsilon$	random	random

4 The Effects of Ties and Stopping Criteria on Convergence

In this section we examine how the specific type 2 tie-breaking policies and the stopping criterion of each variant affect the convergence of each of the three variants.

We wish to highlight in this section that the quality of results and robustness of convergence for each these different variants are affected by the stopping criterion and how type 2 ties are handled.

4.1 Convergence of the Objective Function Variant

In the objective function 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 creates an interesting anomaly in the ascent of the objective function. If the stopping criterion were temporarily removed, 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. An example follows. In this example we use integers to describe the different objects. Given the data set

$$X = \left(\begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} \right)$$

one possible initial partitioning is

$$X_1 = \left(\begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} \right) \text{ and } X_2 = \left(\begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} \right).$$

Due to type 1 ties, it is possible that

$$c_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \text{ and } c_2 = \begin{bmatrix} 2 \\ 2 \end{bmatrix},$$

where

$$Q = 4.$$

With these new modes, we go on to the second iteration and repartition X, keeping in mind that because type 2 ties are randomly allocated, this partitioning is not unique given c_1 and c_2. It is possible that

$$X_1 = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \text{ and } X_2 = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix},$$

where the modes are

$$c_1 = \begin{pmatrix} 2 \\ 2 \end{pmatrix} \text{ and } c_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

and

$$Q = 4.$$

Notice that the objective function, Q, has not changed and under normal conditions, the algorithm would terminate. However, if we were to continue to the next iteration, it is possible that Q will increase if

$$X_1 = \begin{pmatrix} 2 & 2 \\ 1 & 1 \end{pmatrix} \text{ and } X_2 = \begin{pmatrix} 1 & 1 \\ 2 & 2 \end{pmatrix}.$$

This time, our mode vectors will certainly be

$$c_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \text{ and } c_2 = \begin{pmatrix} 1 \\ 2 \end{pmatrix},$$

where

$$Q = 8.$$

In this example, the Objective Function Variant would have stopped at the second iteration, but as we can see, it is possible to continue the algorithm in order to further maximize the objective function. What is really happening between the first and second iterations is that at least one data vector is swapping between clusters whose mode vectors are equally similar to that data vector. Dhillon showed that the objective function of this algorithm will converge, but acknowledges that the clusters may not. The idea of Dhillons's proof can be easily applied to prove the convergence of the Objective Function Variant.

Just because an iterative clustering procedure meets its stopping criteria does not guarantee that the algorithm has reached a local optimum[12]. The situation described in [12], is a type 2 tie and is resolved by resorting to fuzzy

clustering. This situation could also be solved by imparting some sort of type 2 tie-breaking policy.

In the Objective Function Variant, it is precisely because of random tie-breaking during re-clustering that 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 8. It is important to point out that the objective function is integer-valued. This makes it possible for the objective function to stay exactly constant from one iteration to the next such that

$$Q^{(t)} - Q^{(t+1)} \equiv 0.$$

To be sure, this example is not valid for normalized numerical vectors which Dhillon describes because ties are unlikely in floating point operations, but is presented in order to investigate the effect of the stopping criterion of the Objective Function Variant of k-modes.

4.2 Convergence of the Cluster Variant

In contrast to the Objective Function Variant, step 3 in the Cluster Variant effectively biases type 2 ties so that if an object is equally close to two centers, that object stays in its current cluster. The clusters of this variant has been proven to converge by Morgan[**?**].

The effect this has on the iterative path is that all else being equal (no type 1 ties and given the same starting conditions) the Cluster Variant will always iterate through the same path and produce the same final clusters. It may even be tempting to bias type 2 ties as in the Cluster Variant in a k-means implementation in order to ensure that clusters converge. This *Cluster Variant of k-means* also has its drawbacks as we will discuss in the section 7.

4.3 Convergence of the Center variant

The Center Variant handles type 2 ties in exactly the same way as the Cluster Variant. The only difference comes in the stopping criterion. The Clusters will still converge in the Center Variant, just as they did in the Cluster Variant, but because of the possibility of type 1 ties, the convergence of clusters does not imply the convergence of modes. As we saw in section 4.1, a mode vector is not necessarily unique to a given cluster and in section 4.2 we saw 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 cannot be proved. It is important to note, however, that such ties as in sections 4.1 and 4.2 rarely occur when dealing with numerical data. More is said about this in section 9. for the Center Variant. The next section discusses a numerical anomaly in the *Cluster Variant of k-means*.

5 An Anomaly in the Cluster Variant of k-means

One drawback to the method of random type 2 tie breaking in the Objective Function Variant of k-modes is that the clusters may not converge. As Dhillon[1]

points out, this drawback also occurs in the *Objective Function Variant of k-means*. Depending on the implementation goals, it may be more desireable to guarantee convergence of clusters rather than to obtain the a more optimum objective function value.

So, it might be tempting to create a *Cluster Variant of k-means*. In the same way that [**?**] proved convervence of the k-modes Cluster Variant, this new *Cluster Variant of k-means* can be proven to converge. However, roundoff error can cause an anomaly where a data vector indefinitely swaps between two clusters, thus never satsifying the stopping criterion. The following example operates on the principle that on a decimal computer, the average of two numbers may be larger than either number. A similar example can be shown on a binary computer using the idea that, for $b \geq 0$, it is possible that $round(a + b) = a$ if a is large enough.

Suppose we have a decimal computer with three digit precision that uses the IEEE round to nearest standard and we are operating on the data set

$$X = \{.165, .167, .168\},$$

and that the initial clusters are

$$X_1 = \{.165, .167\} \; and \; X_2 = \{.168\}.$$

We Obtain the centers(means) for each cluster by summing the elements and divide by the number of elements in the cluster. However we must keep in mind that we only have two digits of precision.

$$c_1 = \frac{round(.165 + .167)}{2} = \frac{round(.332)}{2} = \frac{.33}{2} = .165.$$

So, our initial means are

$$c_1 = \{.165\} \; and \; c_2 = \{.168\}.$$

The next iteration produces the clusters

$$X_1 = \{.165\} \; and \; X_2 = \{.167, .168\}.$$

Again we re-calulate the means.

$$c_2 = \frac{round(.167 + .168)}{2} = \frac{round(.335)}{2} = \frac{.34}{2} = .17.$$

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

$$X_1 = \{.165, .167\} \; and \; X_2 = \{.168\}.$$

In this example, at least one cluster will change every iteration, so convergence of the clusters can never be achieved.

6 Cluster Death

There is a possibility that upon data vector assignment, no vector is assigned to a particular cluster. The mode of this cluster is therefore indeterminant. We call this situation *Cluster Death*. As we will show, the type 2 tie-breaking policy and the decision of when to calculate modes, can allow or dissallow cluster death. Again, we will use the Objective Function and Cluster Variants of k-modes to describe how this is done.

6.1 Cluster Death in the Objective Function Variant

As discussed previously, the Objective Function Variant assigns type 2 ties randomly. The following example shows how random assignment can lead to an empty cluster. If

$$X_1 = \begin{pmatrix} 1 & 1 \\ 2 & 2 \end{pmatrix} \; and \; , X_2 = \begin{pmatrix} 1 & 1 \\ 2 & 2 \end{pmatrix},$$

then,

$$c_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \; and \; c_2 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

The next partitioning might be

$$x_1 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \end{pmatrix} \; and \; , x_2 = \begin{pmatrix} \quad \end{pmatrix}.$$

In the final iteration of this example, X_2 has an indeterminant mode vector, so the algorithm cannot continue. There are some simple ways of correcting this situation, such as choosing the last relevant mode, i.e. c_2, but they are workarounds at best, and the clusters above are nevertheless representative of the data set.

To be sure, this is a simplistic case of cluster death where all objects are the same, but it can occur in other situations with the Objective Function Variant.

6.2 Cluster Death in the Cluster Variant

Cluster Death is also possible in the Cluster Variant due to type 1 ties. Suppose we are given the following partition of X,

$$X_1 = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, X_2 = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \; and \; X_3 = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}.$$

The mode vectors might be

$$c_1 = \begin{pmatrix} 1 \\ 3 \end{pmatrix}, c_2 = \begin{pmatrix} 2 \\ 4 \end{pmatrix} \; and \; c_3 = \begin{pmatrix} 2 \\ 3 \end{pmatrix}.$$

The subsequent iteration would produce the clusters,

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

The fact that Cluster Death can occur in the Cluster Variant should not imply that it can in Huang's k-modes algorithm. In his algorithm, the modes are updated each time a data vector changes clusters, not simply at each iteration. This modification prevents cluster death from occuring 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 type 2 tie-breaking.

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

Cluster Death can be guarded against in any k-modes algorithm if these two steps are taken. However, this does not include the case of cluster death during initial assignment, which we will discuss in the next section.

7 Starting Points : Clusters or Mode Vectors? it matters in k-modes

In algorithms designed for numerical data, such as k-means and spherical k-means, unique centers(means) can be obtained from clusters and vice versa.

This is due to the relatively precise nature of a floating point machine. In k-means, if the center(mean) vectors are specified, only one partitioning can be obtained if there is no round-off error. This is not the case in k-modes. both types of ties do not allow for the symmetry of clusters and centers as in k-means. If the mode vectors are specified in k-modes, that does not necessarily mean that a unique partitioning can be obtained.

Because of ties in k-modes, The decision of whether to start by specifying clusters (as in the Objective Function Variant) or modes (as in the Center and Cluster Variants) is not as straight-forward. However, it seems that starting with clusters provides a more stable initial assignment. By starting with clusters, the danger of cluster death is avoided for the initial assignment as each cluster is clearly defined at the start.

If the starting positions are specified by modes and one mode is not at least 'as similar' to a data vector as another mode, no data vectors will be assigned to that cluster and cluster death will certainly result.

There are, however, ways to guarantee that cluster death does not occur if we start with mode vectors. One could specify each mode vector to be an instance of a data vector in the set X. This method will eliminate the possibility of cluster death during the initial assignment. This method is not perfect, however, as the optimum mode for a given cluster may not be identical to any vector in that cluster.

There is no hard and fast rule for chosing clusters or modes to start with, but it is our intuition that starting with clusters, as in the Objective Function Variant, will provide the most stable initial assignment. It is important to point out that the situations of cluster death we have referred to in this section are only relevant for the initial cluster assignment. After that, The stability and quality of the clusters are determined by the processes of the algorithm.

8 Matches vs. Mismatches, a Discussion of the Similarity/Dissimilarity Function

We have introduced the notion of similarity and we wish now to investigate the notion of dissimilarity. A general k-modes method maximizes

$$\sum_{i=1}^{k} s(X_i, c_i). \tag{5}$$

This maximization can equivalently be thought of as minimizing

$$\sum_{i=1}^{k} d(X_i, c_i) \tag{6}$$

where

$$d(x, y) = \#mismatches.$$

The data set

$$X = \{car, car, boat, plane\}$$

has a center vector, c, and it is the same with regard to maximization or minimization:

$$c = \arg\min_y \ d(X, y) \ = \ car \ = \ \arg\max_y \ s(X, y).$$

In general

$$d(x, y) = m - s(x, y), \tag{7}$$

where m is the number of components in vectors x and y. The truth of Equation (7) is the key to retaining the duality of similarity and dissimilarity functions, however (7) is not true in all cases. The two exceptions we will discuss here are *Unknown Entries* and *Blank Entries*.

8.1 Unknown Entries

The first issue that creates an asymmetry in the similarity and dissimilarity functions is that of unknown entries, or missing data. Many times, the data set under analysis has indeterminant or 'missing' data, stemming from network errors to real-world data gathering uncertainty. Some These data provide problems for the similarity and dissimilarity functions. For instance, if $*$ represents a missing data entry and

$$x = \begin{pmatrix} 1 \\ * \end{pmatrix}, \ y = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \ and \ z = \begin{pmatrix} 3 \\ * \end{pmatrix},$$

what is the value of s(x,y) or s(x,z)? Should s(x,z) be different than $s(x, y)$? Some solve the problem by ignoring missing values [13] One proposal we have to answer these questions in order to maintain the integrity of (7) is to consider the missing element as another 'board type', or as another element in the domain $\mathbb{P}^m$. This way,

$$s(x, y) = 1 \ match$$

and

$$s(x, z) = 1 \ match,$$

and equation (7) holds true because we have merely added an element to the categorical domain $\mathbb{P}$.

In this case, the similarity and dissimilarity functions can stay essentially the same and the modes can be found just as in equation (3). For example, the mode of the set

$$X = \{1, 1, *, *, 1, 1\}$$

would be $\{1\}$. This shows that, although not all the values in X are determined, the majority that are determined have the value $\{1\}$. Similary, if

$$X = \{1, *, *, *, 1, 1\}$$

the mode would be $\{*\}$, which would reflect that the majority of the values in X are uncertain. This schema works to separate those objects which have '*'s in a particular entry and those that don't just as, under normal circumstances, k-modes works to separate those objects that have '2's in a particular entry and those that don't. In this way, the uncertainties and the certainties in the data set will become more separated. Missing data is difficult to deal with because of its uncertain nature. We do not wish to suggest that this is the best way to deal with missing data in k-modes but, instead, wish to emphasize that this is merely one way to equilibrate the similarity and dissimilarity functions to make equation (7) true in this special case.

8.2 Blank Entries

The second exception to equation (7) we studied is the notion of blank entries. Our particular notion of blank entries resides in the domain of Morgan's paradigm which was introduced in section 3, but the following ideas presented about wildcards could be extended to a similar application. Suppose a customer did not wish to specify a board for each slot in her order; that she wanted to leave a slot blank. The similarity and dissimilarity functions must be modified in order for them to remain duals. We must also keep in mind that, in Morgan's paradigm, board replacement is the only time-consuming step(removal time is negligible). In order to accomodate this fact, we introduce the notion of a 'wildcard' entry, denoted by 'w'. In the paradigm of customer orders for electronic cabinets, if a customer ordered a cabinet with a 'blank' slot, he is essentially saying "No matter what comes in that slot, remove it". In other words, any board in that slot should count as a match because insertions are the *only* time consuming operation. For example, if

$$x = \begin{pmatrix} 1 \\ w \end{pmatrix} c_1 = \begin{pmatrix} 1 \\ 4 \end{pmatrix}, \text{ and } c_2 = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$$

c_1 would be just as good a model to modify as c_2 so as to match the order, x, because in either case, no board insertions are necessary; only a removal is required.

A wildcard will always match whatever component it is compared with; it will always produce a *match* for the similarity function and a *mismatch* for the dissimilarity function. If n is the maximum of the number of wilcards in x and y, we can add n to the number of matches and subtract n from the number of mis-matches.

$$d(x, y) - n = m - s(x, y) + n \tag{8}$$

The maximum number of wildcards from x and y is used because comparing two wildcards to each other produces the same results as comparing one wild card with a component that is not a wildcard. In this way, we can reformulate equation (7) in order to retain the duality of similarity and dissimilarity in the face of bank entries.

9 Restricting the Domain of Mode Vectors

Some categorical clustering algorithms[14] suggest limiting the domain of each componentof c, such that the values $c_j, j = 1..m$ are restricted to the domain $\mathbb{P}_j, j = 1..m$.

This makes sense in the Paradigm given in [**?**] as it is possible that certain boards are never ordered for a certain slot and will thus never appear in the model. Analytically, such restriction is unecessary because those values that are to be excluded are not taken into account during the optimization of (3). However, in an implementation of k-modes, decreasing the size of the domain of the modes vectors might reduce the time and space required to compute those modes. Restricting the domain of

10 Conclusion

We discussed the two types of ties involved in any k-modes method; ties in the mode operation and ties during vector assignment. Ties in the mode operation are difficult to resolve and if resolved randomly, represent an exponential solution space. There are some choices in dealing with the second type of ties, and those choices affect the convergence of any k-modes method. The Cluster Variant has been proven to converge by Morgan and the Objective Function Variant has been proven to converge by Dhillon. We showed that the two types of ties in k-modes prevent the Center Variant from guaranteeing convergence. We Introduced a numerical anomaly in the Cluster Variant of k-means that prevents is from meeting its stopping criterion. We introduced three variants and showed how the tie-breaking policies can affect convergence and prevent or allow Cluster Death. We have also shown that, as stated by Ismail, et. al.,[12] the fact that a particular k-modes algorithm has stopped does not guarantee that is has converged to a local minimum.

less to converge.

Appendix A:

The following examples illustrate a side-by-side comparison of the Objective Function, Center and Cluster variants. Each variant is given the initial conditions (specified below) and each arrow denotes either the calculation of the mode vectors or reassignment of the data vectors.

Places where ties occur are highlighted in bold. The iterations are displayed as follows; each cluster and mode vector pair is separated by a $\|$ symbol so that each iteration has the form

$$\left\{ \begin{array}{cccc|cccc} X_1 & . & c_1 & | & X_2 & . & c_2 \end{array} \right\}$$

We begin each variant with the same partition, namely,

$$X = \left\{ \begin{array}{cccc|cccc} 1 & 2 & | & 2 & 2 \\ 2 & 2 & | & 1 & 1 \end{array} \right\}.$$

Figure 1. shows the decision tree of the Cluster variant. In this small example,the type 1 tie can be broken two ways, but the Cluster variant stops quickly as neither X_1 nor X_2 change from the first iteration to the next.

Figure 2. shows the decision tree of the Center Variant. Here, the Cluster Variant may never stop because at each point there is a type 1 tie to be handled. The Center Variant will only converge if this type 1 ties is broken the same way twice, giving identical mode vectors for X_1. Also note that the partitions do not change in this variant.

3. Figure 3. shows the decision tree for the Objective Function Variant. In this variant, both type 1 and type 2 ties are randomly broken, so the number of possible partitions is greater. All possible decisions for tie-breaking are shown. There are certainly more possibilities for different partitions in this variant, but it is important to notice that the objective function (total number of matches) does not change, so at any one of these iterations, this method may be declared convergent. It is interesting that so many equal optimums can be obtained from so simple example due to both types of ties in k-modes.

References

[1] I. S. DHILLON. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42:143–175, 2001.

[2] Z.HAUNG. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, 1998.

[3] Z. HUANG. A fast clustering algorithm to cluster very large categorical data sets in data mining. In *Research Issues on Data Mining and Knowledge Discovery*, pages 0–, 1997.

[4] H. RALAMBONDRAINY. A conceptual version of the k-means algorithm. *Pattern Recognition Letters volume=*.

[5] T. Ryu and C. F. Eick. A unified measure for attributes with set or bag of values. 1998.

[6] S. D. MORGAN. and Y. FATHI.

[7] C. Xu K. Wang and B. Liu. Clustering transactions using large items. In *CIKM*, pages 483–490, 1999.

[8] C. M. PROCOPIUC C. C. AGGARWAL and P. S. YU. Finding localized associations in market basket data. *Knowledge and Data Engineering*, 14(1):51–62, 2002.

[9] D. Ca NAS and R. E. FUNDERLIC. Unification of k-center clustering methods based on similarity measures. 2003.

[10] R. DUBES and A. K. JAIN. Clustering techniques: The users dilemma. 8.

[11] Z. HUANG. 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.*

[12] Shorki. Selim and M.A.Ismail. K-means type algorithms: A generalized convergence theorem and characterzation of local optimality. 6.

[13] S GUHA, R. RASTOGI, and K. SHIM. ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366, 2000.

[14] Venkatesh Ganti, Johannes Gehrke, and Raghu Ramakrishnan. CACTUS - clustering categorical data using summaries. In *Knowledge Discovery and Data Mining*, pages 73–83, 1999.