Adapting K-Medians to Generate Normalized Cluster Centers

Benjamin J. Anderson, Deborah S. Gross, David R. Musicant
Anna M. Ritz, Thomas G. Smith, Leah E. Steinberg
Carleton College
andersbe@gmail.com, {dgross, dmusican, ritza, smitht, steinbel}@carleton.edu

Abstract

Many applications of clustering require the use of normalized data, such as text or mass spectra mining. The spherical K-means algorithm [6], an adaptation of the traditional K-means algorithm, is highly useful for data of this kind because it produces normalized cluster centers. The K-medians clustering algorithm is also an important clustering tool because of its well-known resistance to outliers. K-medians, however, is not trivially adapted to produce normalized cluster centers. We introduce a new algorithm (called MN), inspired by spherical K-means, that integrates with K-medians clustering to produce locally optimal normalized cluster centers. We then show theoretically and experimentally that MN produces clusters of significantly higher quality than one would obtain via a simple scaling of the cluster centers produced from traditional K-medians.

Keywords: K-medians, clustering, normalized data

1. Introduction

Clustering is one of the key techniques used in data mining. In this paper we focus on clustering *normalized* datasets. Data is often normalized before clustering in order to remove unimportant distinctions of scale. In clustering a corpus of text, for example, each document is often represented as a point where each dimension represents a word's frequency. When clustering data in this format, two documents of identical subject matter with different lengths should not be considered different. It therefore often makes sense to normalize text data before clustering.

Single-particle mass spectrometry [11] is another area where it is appropriate to normalize data. The chemical makeup of an aerosol particle is represented as two mass spectra, which are plots of signal intensity versus the mass-to-charge ratio (m/z) for positive and negative ions produced by fragmenting the components of the aerosol particle. Thus, the presence of a peak indicates the presence of one or more ions containing the m/z value indicated (see Figure 15 for examples). As with text, we are generally not interested in the absolute magnitude of these peaks, but in the relative size of one peak when compared with another.

Cluster centers are often used to represent prototypical points, so it is usually desirable to normalize cluster centers that arise from normalized data [5, 6]. The spherical K-means algorithm [6] is well-suited to this task.

For some applications it is more desirable to use 1-norm distance (also known as Manhattan distance, denoted here as $\|\cdot\|_1$) to measure the distance between points. This is because the cluster center that minimizes 1-norm distance to all points within that cluster is the median of that cluster, and using the median instead of the mean tends to be more robust to outliers. The K-medians algorithm [4, 9] is thus a powerful alternative.

When attempting to use K-medians on normalized data, however, unique challenges arise in finding normalized locally optimal cluster centers. We present the Manhattan Normalization (MN) algorithm, which when integrated with K-medians addresses these challenges.

There are two seemingly straightforward solutions to the problem of finding normalized cluster centers with K-medians. One could apply traditional K-medians and scale the cluster centers at the end, or alternatively one could use traditional K-medians and scale the cluster centers after each iteration. Both of these ideas are heuristically based and lack theoretical guarantees of convergence. We will show that MN integrated with K-medians has guaranteed convergence properties and performs better experimentally than either of these other approaches. We borrow many of our ideas from spherical K-means, but we address the additional difficulties introduced by the use of medians.

Section 2 of this paper describes relevant known clustering algorithms. Section 3 describes the challenges that arise in normalized K-medians clustering and our solutions to those challenges. Experimental results and analysis are in Section 4.

2. Clustering Review

2.1 Traditional K-means

The well-known K-means algorithm [8] is used when one wishes to find cluster centers that minimize the total of the squared 2-norm distance (also known as Euclidean distance, denoted here as $\|\cdot\|_2$) from each point to its closest cluster center. Since finding globally optimal cluster centers is an NP-hard problem, K-means may be used to find a local solution. In order to run K-means, one chooses the number of clusters to find and an initial set of cluster centers. There are many different approaches for choosing initial cluster centers [3], but all of our work here holds regardless of which technique is used. The K-means framework can then be described as follows:

K-means Algorithm. Let $x_1,x_2,...,x_n$ be a set of points. We wish to partition them into K disjoint clusters $\pi_1^*,\pi_2^*,...,\pi_K^*$ that minimize the objective function:

$$Q(\{\pi_j\}_{j=1}^K) = \sum_{j=1}^{K} \sum_{x\in\pi_j} \left\| x - c_j \right\|_2^2 \qquad (1)$$

For each cluster j, c_j is the center of each cluster, which is defined as the point for which $\sum_{x\in\pi_j} \left\| x - c_j \right\|_2^2$ is minimized. This has been proven to be accomplished by choosing c_j to be the centroid of cluster j, defined as

$$c_j = \frac{1}{n_j} \sum_{x\in\pi_j} x \qquad (2)$$

where n_j is the number of points in cluster j [8]. We therefore wish to find clusters that solve the following minimization problem:

$$\left\{\pi_j^*\right\}_{j=1}^K = \arg\min_{\left\{\pi_j\right\}_{j=1}^K} Q\left(\left\{\pi_j\right\}_{j=1}^K\right) \qquad (3)$$

To do this, start with an arbitrary partitioning of the data $\left\{\pi_j^{(0)}\right\}_{j=1}^K$. Define the cluster centers associated with this partitioning as $\left\{c_j^{(0)}\right\}_{j=1}^K$. Define the index of iteration $t = 1$. Then follow these steps:

1. For each point, find the cluster center with closest Euclidean distance. This yields a new partitioning

$$\pi_j^{(t)} = \left\{ x : \left\| x - c_j^{(t)} \right\|_2^2 \le \left\| x - c_i^{(t)} \right\|_2^2, 1 \le i \le K \right\}, \quad j = 1,\ldots K \qquad (4)$$

where ties between clusters are resolved by random assignment to one of the optimal centers. Note that this is guaranteed not to increase the objective Q, since each point is assigned to its closest center.

2. Compute the new set of cluster centers $\left\{c_j^{(t)}\right\}_{j=1}^K$ by computing the mean (centroid) of each cluster. Since the centroid is the point that minimizes the total distances from all points to it, this step is also guaranteed not to increase the objective Q.

3. If a stopping criterion is met, report $\left\{\pi_j^{(t)}\right\}_{j=1}^K$ as the final partitioning and $\left\{c_j^{(t)}\right\}_{j=1}^K$ as the final cluster centers. Otherwise, increment t by 1, and go to step 1 above. A variety of stopping conditions are available. One common condition is to stop when the difference between successive values of the objective Q is less than a small tolerance.

K-means is ultimately a local optimization algorithm for minimizing clustering error, where clustering error is defined as the total squared Euclidean distance from each point to its closest center. The objective Q never increases from one iteration to the next. Since K-means is applied to a finite number of points, the algorithm must therefore terminate.

2.2 Spherical K-means

Many applications for clustering normalized data require normalized cluster centers. The spherical K-means algorithm by Dhillon and Modha [6] addresses this need. The spherical K-means literature uses the cosine similarity metric. We find it more convenient for this paper to use the squared Euclidean distance metric for spherical K-means, but it is easily shown that for normalized data both of these metrics yield precisely the same results.

Spherical K-means produces cluster centers of magnitude 1 by normalizing the cluster centers after each iteration. In other words, an extra step is added to the traditional K-means algorithm as follows:

Spherical K-means algorithm. Start with a partitioning of the data as in the traditional K-means algorithm. Initialize $t = 1$.

1. For each point, find the closest cluster center as measured via squared Euclidean distance.

2. Compute the new set of cluster centers $\{c_j^{(t)}\}_{j=1}^{K}$ by computing the mean (centroid) of each cluster.

3. Normalize each cluster center by scaling it so that it has a magnitude of 1. In other words, redefine

$$c_j^{(t)} := \frac{c_j^{(t)}}{\| c_j^{(t)} \|_2}, j = 1, \ldots, K \quad (5)$$

4. Terminate if the stopping condition is met. Increment *t* and go to step 1 otherwise.

Note that step 3 may actually increase the clustering error following step 2, since step 2 finds the cluster centers that optimize cluster error regardless of cluster center magnitude. Step 3 modifies the optimal cluster centers so that they have a magnitude of 1 at the expense of increasing clustering error. However, Dhillon and Modha show [6] that if step 2 and step 3 are considered together as one operation, this procedure finds the optimal center of magnitude 1 for each cluster. The property that cluster centers have a magnitude of 1 is thus preserved from one iteration to the next, while the algorithm ensures that cluster error does not increase. Spherical K-means is therefore guaranteed to converge to a solution of locally optimal cluster centers, each with magnitude of 1.

2.3 K-medians

We now review the K-medians algorithm [4, 9], which is used when one wishes to minimize the total 1-norm distance from each point to its nearest cluster center. K-medians is quite similar to K-means, and its differences from K-means are defined as follows:

K-medians algorithm. Since we now work with 1-norm distance instead of squared Euclidean distance, our objective is stated as:

$$Q(\{\pi_j\}_{j=1}^{K}) = \sum_{j=1}^{K} \sum_{x \in \pi_j} \left\| x - c_j \right\|_1 \quad (6)$$

We start with a partitioning of the data as in K-means. Initialize $t = 1$.

1. For each point, find the closest cluster center as measured via 1-norm distance.

2. Compute the new set of cluster centers $\{c_j^{(t)}\}_{j=1}^{K}$ by computing the median of the cluster. In other words, for each dimension compute the median value for that dimension over all points in the cluster. We use the median because the median is the point that minimizes the total 1-norm distance from all points to it [4].

3. Terminate if the stopping condition is met. Increment *t* and go to step 1 otherwise.

In a similar fashion to K-means, steps 1 and 2 of K-medians are guaranteed not to increase the objective *Q*.

3. Normalized K-medians cluster centers

There are a variety of tradeoffs in choosing between K-medians and K-means. We consider the discussion of the choice of K-medians vs. K-means (K-medians is slower but more robust to outliers, etc.) outside the scope of this paper. Our purpose is to enable the appropriate use of K-medians on normalized data.

We mentioned in Section 1 some simple changes to K-medians that might seem to appropriately adapt it for obtaining normalized cluster centers from normalized data. We discuss these ideas, point out their flaws, and then move on to discuss our solution.

3.1 Simple Approaches

Simple approach #1: normalize by scaling at each iteration. A straightforward adaptation of spherical K-means is problematic. The concept seems to be easy enough: during each iteration, after new cluster centers have been determined, normalize them via scaling using the appropriate distance metric. In other words, redefine cluster centers as follows:

$$c_j^{(t)} := \frac{c_j^{(t)}}{\| c_j^{(t)} \|_1}, j = 1, \ldots K \quad (7)$$

There is a key problem with this approach. Unlike spherical K-means, scaled cluster centers *are not the points of magnitude 1 that minimize total 1-norm distance from each point to its cluster center*.

Example: Consider the 1-norm normalized points x_1 = (1/5, 1/5, 3/5), x_2 = (1/5, 1/5, 3/5), x_3 = (1/5, 3/5, 1/5), x_4 = (1/5, 3/5, 1/5), and x_5 = (3/5,1/5,1/5). The median of these points is (1/5, 1/5, 1/5). If we simply proportionately scale this median to have a magnitude of 1, we end up with a cluster center of (1/3, 1/3, 1/3). If we measure 1-norm distance from each point to this center, each point is a distance of 8/15 from this center, resulting in a total distance of 40/15. Suppose that we instead choose the normalized point (1/5, 3/5, 1/5) to be our cluster center. This center yields a total 1-norm distance of only 36/15. The scaled median is therefore clearly not the center that minimizes total 1-norm distance from all points to it.

Simple approach #2: normalize by (method of choice) when K-medians stabilizes. This approach is heuristic in nature, though it may yield positive results. If the normalization technique to be used is simple scaling, however, problems arise as discussed above. Like "Simple Approach #1," "Simple Approach #2"

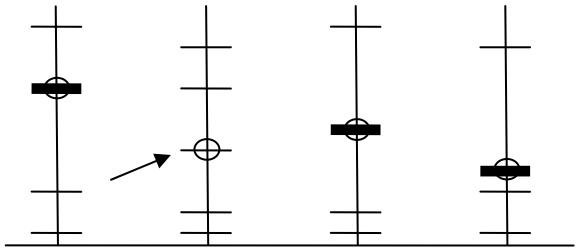


Figure 1: Graphical representation of our MN algorithm. Vertical lines are dimensions, horizontal lines are values (thick horizontal lines are the same value occurring twice), and circles are current cluster center locations. Sliding the second circle upward incurs less error than sliding any other circle would since more values are above it.

does not provide desirable theoretical guarantees that spherical K-means does. By performing regular K-medians at each iteration, the objective does not increase but the centers found at each iteration do not satisfy desired constraints (magnitude of 1). After K-medians stabilizes, when the cluster centers are normalized, the objective is likely to increase steeply.

3.2 Our Solution: The MN Algorithm

The structure of the spherical K-means algorithm is sound. In order to obtain cluster centers of magnitude 1, one should produce normalized cluster centers iteratively throughout the algorithm such that the objective error metric continues to decrease. "Simple approach #1" tried to address this, but was flawed. The missing link needed here is an algorithm to solve the following task: given a set of points assigned to a cluster, find a center of magnitude 1 (in a 1-norm sense) that minimizes the total 1-norm distance from all points to this center. Here is our algorithm for doing so.

Manhattan Normalization (MN) algorithm. Let $x_1,x_2,...,x_n$ be a set of points where $\|x_i\|_1=1$, $i=1,...,n$. We wish to find a point c, where $\|c\|_1=1$, that minimizes:

$$\sum_{i=1}^{n}\left\|x_i - c\right\|_1 = \sum_{i=1}^{n}\sum_{j=1}^{d}\left|x_{ij} - c_j\right| \qquad (8)$$

where d is the number of dimensions, c_j indicates the jth component of cluster center c, and x_{ij} represents the jth component of point x_i.

1. Initialize c to be the median of $x_1,x_2,...,x_n$. c minimizes the objective above, but it is not necessarily true that $\|c\|_1=1$. (If $\|c\|_1=1$, we terminate the algorithm.) Note that examples can be easily generated to show that $\|c\|_1$ can in fact be less than or greater than 1 (as was done in Section 3.1).

We also assume through the remainder of this algorithm and in the theorem that follows that $c_j \geq 0$, $j=1,...,d$. This is true for many applications of normalized data. If $c_j < 0$ for a particular j, transform this dimension by negating both c_j and all values x_{ij} ($i=1,...,n$). On completion of the algorithm, negate c_j again. This makes the rest of the algorithm easier to state (no special cases for negatives), yet has no effect on its correctness.

The algorithm is symmetric with respect to whether $\|c\|_1<1$ or $\|c\|_1>1$, so we describe the $\|c\|_1<1$ case and indicate the $\|c\|_1>1$ case in brackets.

2. For each dimension j, count the number of values x_{ij} that are strictly greater than [less than] c_j. Denote these counts as z_j , $j=1,...,d$.

3. Let m be the dimension for which z_j is maximal. If more than one dimension z_j has the same maximal value, choose one arbitrarily.

4. Redefine c_m to be the smallest [largest] value x_{im} ($i=1,...,n$) greater than [less than] c_m. If $c_m < 0$, set $c_m = 0$.

5. If $\|c\|_1=1$, terminate the algorithm. If $\|c\|_1<1$ $\left[\|c\|_1>1\right]$, go to step 2. Otherwise, redefine c_m as $c_m - \left(\|c\|_1 - 1\right)\left[c_m + \left(1 - \|c\|_1\right)\right]$ and terminate.

The idea behind the algorithm can be clarified via Figure 1. At each iteration, the dimension that we change is the one that has a maximal number of values in the direction that we want the center to move. To see this, consider sliding any of the circles in Figure 1 upward a small distance ε, which would have the effect of increasing $\|c\|_1$ by ε. Furthermore, the objective error metric (sum of 1-norm distances from all points to the cluster center) is increased by ε for each tick on or below the circle moved, and decreased by ε for each tick above the circle. Therefore, we slide the circle with the most ticks above it in order to increase $\|c\|_1$ while incurring as little error as possible. We repeat this entire process until $\|c\|_1 = 1$. With this intuition in mind, we prove the following theorem.

Theorem: Given a set of points $x_1,x_2,...,x_n$ where $\|x_i\|_1=1$, $i=1,...,n$, the MN algorithm finds a point c ($\|c\|_1=1$) that minimizes the total 1-norm error from all points to it. Formally, c is the solution to the following optimization problem:

$$\arg\min_{c} \sum_{i=1}^{n} \|x_i - c\|_1 = \sum_{i=1}^{n}\sum_{j=1}^{d} |x_{ij} - c_j| \quad s.t.\ \|c\|_1 = 1 \tag{9}$$

Proof: We know that the initial value of c as chosen in Step 1 of the algorithm minimizes the following alternative problem [4]:

$$\arg\min_{c} \sum_{i=1}^{n} \|x_i - c\|_1 \tag{10}$$

If $\|c\|_1=1$, then the theorem is trivially true.

Suppose that $\|c\| < 1$. Steps 2 and 3 choose a dimension *m* of *c* to modify. Suppose that $c_m \geq 0$. Step 4 redefines the *m*th component of *c*. We denote here the new value of *c* as $\hat{c}$. We then observe that $\hat{c}$ must be a solution for the following optimization problem:

$$\arg\min_{c} \sum_{i=1}^{n} \|x_i - c\|_1 = \sum_{i=1}^{n}\sum_{j=1}^{d} |x_{ij} - c_j| \quad s.t.\ \|c\|_1 = \|\hat{c}\|_1 \tag{11}$$

To see this, recall that $\hat{c}$ was obtained by adding $\hat{c}_m - c_m$ to the *m*th component of c. This adds to the objective an additional error of $\hat{c}_m - c_m$ times the number of values x_{im}, i=1,...,n less than or equal to c_m. Likewise, it subtracts from the objective an error of $\hat{c}_m - c_m$ times the number of values x_{im}, i=1,...,n greater than c_m. Since m was chosen to be the dimension with maximal values greater than c_m, and since the total number of values in each dimension is the same, any increase at all in any other dimension of value less than or equal to $\hat{c}_m - c_m$ cannot result in a lesser increase of the objective.

This argument is easily adapted for the $\|c\| > 1$ and $c_m < 0$ cases through appropriate reversals (positive / negative, less than / greater than, etc.). We omit these other cases for space saving purposes.

Now that we have established the MN algorithm, we can integrate it with K-medians:

MN iterative K-medians algorithm. We start with a partitioning of the data. Initialize t = 1.

1. For each point, find the cluster center with closest 1-norm distance.
2. Compute the new set of cluster centers $\{c_j^{(t)}\}_{j=1}^{K}$ by computing the median of the cluster.
3. Normalize each cluster so that it has a 1-norm magnitude of 1 by using the MN algorithm.
4. Terminate if the stopping condition is met. Increment *t* and go to step 1 otherwise.

If one wishes to use K-medians on normalized data and yield normalized cluster centers, MN iterative K-medians will find cluster centers at each iteration with error no greater than that from the previous iteration.

3.3 MN Algorithm Performance Issues

K-medians is clearly not as fast as K-means due to its median computations. Since our MN algorithm occurs after a median is calculated, any pre-existing algorithm for calculating medians quickly can be used. The MN algorithm itself, in its worst case, could require an iteration for each value above the median in each dimension. This is yields an upper bound of *nd* iterations. At the beginning of the algorithm, there is an initial step where the number of values greater than or less than the initial center must be counted for each dimension. This also requires *nd* calculations (*d* dimensions, *n* points for each), but this is only necessary once. Each successive iteration then requires determination of which dimension has the greatest number of values above the current center. This could be handled via a priority queue where the priority is the number of values above the current center. With *d* entries in the priority queue, this would result in $\log_2 d$ calculations per iteration to update it. This yields a complexity of $O(nd \log d)$ each time that MN is used, i.e. at each major K-medians iteration.

For a sizeable dataset, then, MN forms a relatively negligible portion of calculation time. While there are many optimized approaches for calculating medians under certain circumstances, a straightforward modified quicksort algorithm has a complexity of $O(n \log n)$ on average. Since each major iteration of the K-medians algorithm requires that a median be calculated in each dimension within each cluster, this has a complexity of $O(nd \log n)$. (In the case where the *n* points are distributed evenly among the *K* clusters, we can refine this complexity to $O\left(dK \frac{n}{K} \log \frac{n}{K}\right)$. Since *K* is the number of clusters and is not expected to be dramatically large, this complexity can again be simplified to $O(nd \log n)$.) Therefore, for a large dataset where $n \gg d$, these $O(nd \log n)$ calculations required by the median algorithm will completely dominate the $O(nd \log d)$ calculations required by the MN algorithm.

It should also be pointed out that when the data is all non-negative, as our atmospheric data is, equation (8) can be formulated as a linear program (LP). An LP

solver could therefore be used as an alternative to our MN algorithm. To test this, we used CLP [7], a high quality open-source LP solver. Because the MN algorithm is designed to attack this problem directly and CLP is a generic LP solver, CLP took dramatically longer than MN.

Finally, we point out that our algorithm does not claim to have competitive running time with K-means. It is well known that K-means is considerably faster than K-medians. The purpose of this paper is to demonstrate that if one wishes to use K-medians because of its outlier-resistant properties, it can be so adapted by using the MN algorithm.

4. Experiments

In order to test the effectiveness of the MN algorithm in improving traditional K-medians, we compare four different versions of K-medians:

1. "MN at each iteration." This is our MN iterative K-medians algorithm as described in Section 3.2.
2. "MN only at the end." This is Simple Approach #2 (Section 3.1) with MN used as the normalization algorithm at the end.
3. "Scaling at each iteration." This is Simple Approach #1 (Section 3.1): at each iteration of the K-medians algorithm, 1-norm scaling is used to normalize the median.
4. "Scaling only at the end." This is "Simple Approach #2" with 1-norm scaling (Section 3.1) used as the normalization algorithm at the end.

We also compare these four algorithms with K-means as a sanity check. For all five techniques, we choose initial centers via the heuristic technique of choosing the first point as the first center, then choosing each successive center to be the point in the dataset whose distance to its closest center is greatest. We terminate iteration of the algorithm when the error metric changes by no more than 0.01. For the two approaches that normalize only at the end, note that the error undergoes a sudden increase after the last iteration.

We test our clustering algorithms against three different data sets. We first discuss results from text documents that are clustered by word frequency. We then move on to two mass spectral datasets, both representing aerosol particle data. The first of these two datasets consists of a synthetic dataset generated by adding noise to seven actual particle mass spectra. The second dataset corresponds to actual aerosol data taken from St. Louis in February of 2004.

For each of these three datasets, we report our experiments using a single value for K (number of clusters) for brevity. Our goal is to examine differences in clustering error and convergence properties, and we point out that the theoretical discussion above on the merits of these algorithms is independent of number of clusters. We typically choose K to be our best estimate as to the number of clusters actually in the data, since this enables an easy scan of the resulting clusters for correctness. Clustering error in these experiments is defined as the average Manhattan distance from each point to its closest center.

4.1 Text Documents

Our text dataset was generated from a set of 505 sonnets from 5 different authors (Rossetti, Spenser, Browning, Sidney, and Shakespeare), where each sonnet was represented as a single point by counting word frequencies within that text [2]. We pre-processed our data with the Porter stemming algorithm [10]. We clustered with $K = 5$ for all algorithms.

We compare the clustering error from our four K-medians variants in Figure 2. Both of the iterative normalization methods have the same general curve, but MN iterative normalization has lower error than scaled iterative normalization. The scaled iterative normalization error increases and decreases slightly as it stabilizes, while MN iterative always decreases as explained in Section 3.2. For the two cases where normalization is performed only at the end, the errors are identical until the very end of the algorithm because these two clustering methods are exactly the same until the last pass. Notice, however, that after normalization scaling gives a higher error than MN.

Figure 2 also shows that for this dataset, normalizing with either technique at the end yields only slightly worse errors than normalization at each iteration. This suggests that one can use this shortcut of normalizing only at the end if a slight degradation in cluster accuracy and a rapid increase in clustering error at the end are acceptable.

Figures 5-9 (grouped together at the end of the paper) represent the breakdown of each cluster by author for the four K-medians algorithms and the spherical K-means algorithm. Since we know the author of each sonnet, we can then measure how homogenous each cluster is with respect to the five authors.

Figures 7 and 8, which represent the clusters generated from the two algorithms that use scaled normalization, show fairly poor results. Cluster 2 in Figure 7 and cluster 1 in Figure 8 each contain the bulk of the sonnets by all authors. MN iterative (Figure 5), on the other hand, does a much better job clustering by

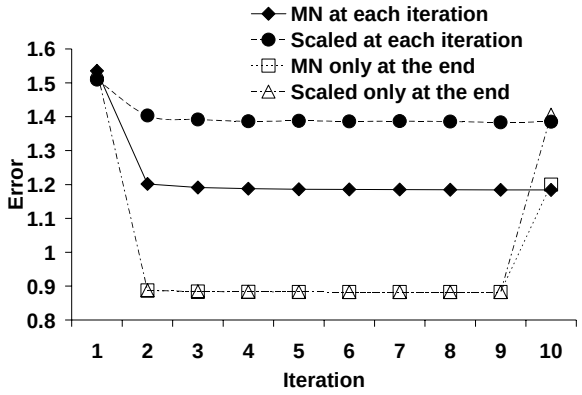


Figure 2: Error per iteration for the four K-medians algorithms on the text data. Note that the algorithms involving scaling consistently perform worse than the algorithms using MN. The scale for the vertical axis has been chosen to render the results as clearly as possible.

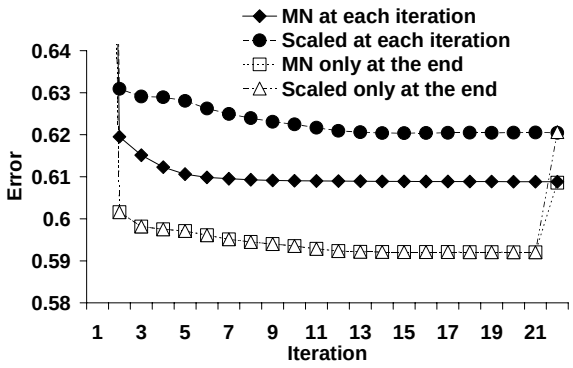


Figure 3: Error per iteration for the four K-medians algorithms on the synthetic particles.

author: clusters 1, 3, 4, and 5 are mostly homogenous. These results are considerably better than scaled normalization. Additionally, though MN at the end has worse clusters than iterative MN (Figure 6 only has three decent clusters; 3, 4, and 5), it still performs better than scaled normalization.

Finally, when comparing iterative MN with spherical K-means (Figure 9), we see that it performs at least as well. Note that we are unable to directly compare the clustering error from spherical K-means to the error from the K-medians algorithms in Figure 2 because the distance metrics are different.

4.2 Synthetic Particles

We generated a synthetic dataset of mass spectra by starting with spectra from seven actual aerosol particles. Based on these seven particles, we created 2000 artificial particles by adding noise to the spectra of the seven particles [1]. Since there are seven known particles, we clustered using $K = 7$.

Looking at the error versus the number of passes, we see the same trends as in the text data (Figure 3). MN outperforms scaled normalization when performed either iteratively or at the end.

There is one important difference to note between Figures 2 and 3. The final errors for iterative MN and MN at the end are identical in Figure 3. (Technically, iterative MN is 2.46×10^{-4} higher than MN at the end, but this is negligible due to possible rounding errors.) In Figure 2, however, iterative MN has visibly lower error. This again shows that our MN algorithm is a considerably better normalization technique than simple scaling, and the user can choose between using it at each iteration or at the end depending on needs.

The improvement that MN shows over simple scaling is not as dramatic in Figure 3 as it is in Figure 2. Note that we have chosen a scale for the vertical axis to render the results as clearly as possible. Although we have shown theoretically that MN will always perform better than (or equal to) simple scaling, how much better MN performs depends on the particular dataset. It is apparently the case that for this synthetic dataset, simple scaling is less error-prone than it is for our sonnets dataset. Nonetheless, in both examples, MN performs better. This is to be expected, since MN has been proven to be locally optimal.

Figures 10-13 (grouped together at the end of the paper) show the cluster distributions from these four algorithms. All four graphs show that each cluster is mostly homogeneous. Figure 14 shows the results from spherical K-means clustering on this data. Three of the clusters in this graph (clusters 1, 3, and 5) show considerably less homogeneity.

4.3 St. Louis Dataset

We finally present results from clustering a dataset of mass spectra corresponding to 2966 particles collected during February 2004 in East St. Louis, Illinois at the EPA SuperSite location. We used K=9 which seemed to provide easily interpretable results. Figure 4 shows error versus the number of passes for the four K-medians algorithms, indicating behavior consistent with the previous datasets.

We have no knowledge of what the "correct" clusters are for this dataset. We did, however, examine the cluster centers that resulted from these four K-medians algorithms (author Prof. Gross is an atmospheric scientist accustomed to examining such plots). We observed that the MN iterative K-medians algorithm picked up more peaks due to negative ions

than any of the other three K-medians algorithms did. This set of cluster centers is therefore more likely to be correct than the other three sets. The MN iterative K-medians algorithm also showed considerably more peaks at locations with high *m*/*z* values (mass-to-charge ratios) than any of the other algorithms did, which also added to its credibility. Interestingly, the only other one of these four algorithms to show a cluster center with peaks at high *m*/*z* values was the scaled iterative algorithm. We present two sample cluster centers to illustrate these findings (Figure 15).

5. Conclusions and Future Work

We have presented the MN algorithm, which allows successful use of K-medians on normalized data when normalized cluster centers are desired. This algorithm can be used during each iteration of K-medians or simply at the end of the algorithm, depending on whether one wants to emphasize non-increasing clustering error or speed. We provided theoretical and experimental evidence that our algorithm is correct. Finally, to demonstrate experimentally the viability of our technique, we provided some brief comparisons with spherical K-means. On our datasets, K-medians combined with MN yields clusters that are comparable to or better than those produced by spherical K-means.

Most of the scalability issues for our algorithm are limited by the state of the art in scaling traditional K-medians. There is nonetheless room for future work in considering how to scale our algorithm, particularly when the number of dimensions is high.

Acknowledgements. This research is supported by NSF ITR grant IIS-0326328 and by Carleton College.

6. References

[1] B. J. Anderson, D. R. Musicant, A. M. Ritz, A. Ault, D. S. Gross, M. Yuen, and M. Gaelli, "User-Friendly Clustering for Atmospheric Data Analysis," Carleton College, Northfield, MN, Technical Report 05-a, 2005.

[2] E. Blomquist, *Sonnet Central*, 2003, www.sonnets.org.

[3] P. S. Bradley and U. M. Fayyad, "Refining Initial Points for K-Means Clustering," presented at Proc. 15th International Conf. on Machine Learning, 1998.

[4] P. S. Bradley, O. L. Mangasarian, and W. N. Street, "Clustering via Concave Minimization," in *Advances in Neural Information Processing Systems*, vol. 9, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds. Cambridge, MA: MIT Press, 1997, pp. 368-374.

[5] I. S. Dhillon, Y. Guan, and J. Fan, "Efficient Clustering of Very Large Document Collections," in *Data Mining for Scientific and Engineering Applications*, 2001, pp. 357-381.

[6] I. S. Dhillon and D. S. Modha, "Concept Decompositions for Large Sparse Text Data using Clustering," *Machine Learning*, vol. 42, pp. 143-175, 2001.

[7] J. Forrest, D. d. l. Nuez, and R. Lougee-Heimer, "CLP: COIN Linear Program Code," 1.02.02 ed, 2005.

[8] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*: Springer, 2001.

[9] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*: Prentice-Hall, 1981.

[10] M. F. Porter, "An Algorithm for Suffix Stripping," *Program*, vol. 14, pp. 130-137, 1980.

[11] D. T. Suess and K. A. Prather, "Mass Spectrometry of Aerosols," *Chemical Reviews*, vol. 99, pp. 3007-3035, 1999.

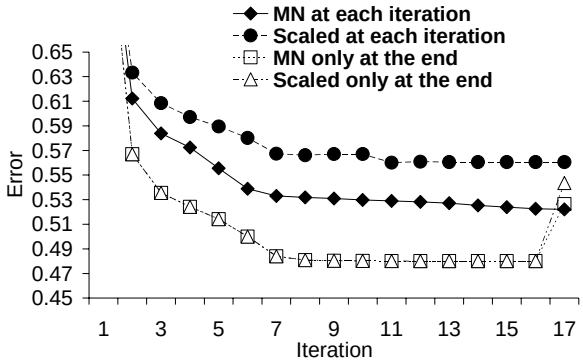


Figure 4: Error per iteration for the four K-medians algorithms on the St. Louis data.

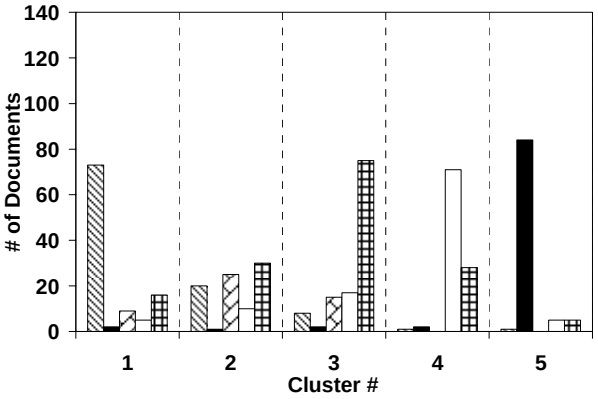


Figure 5: K-medians with MN at each iteration

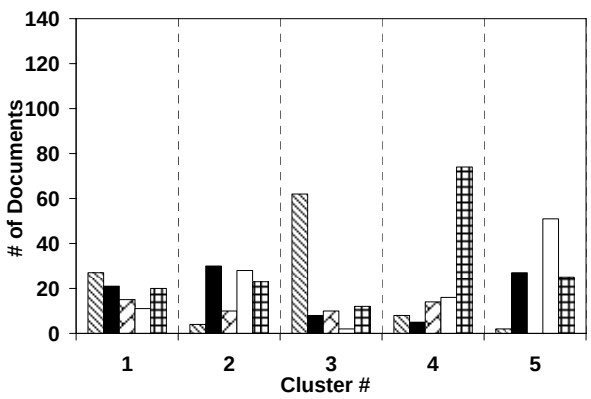


Figure 6: K-medians with MN only at the end

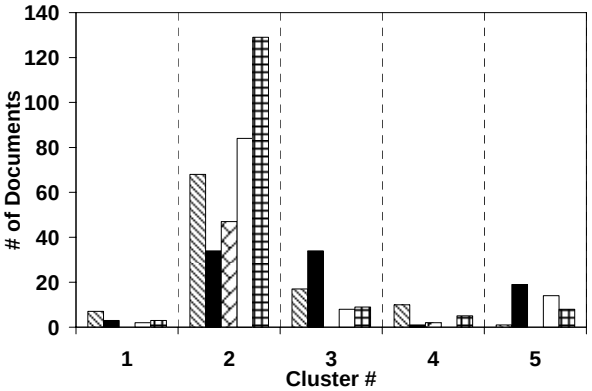


Figure 7: K-medians scaled at each iteration

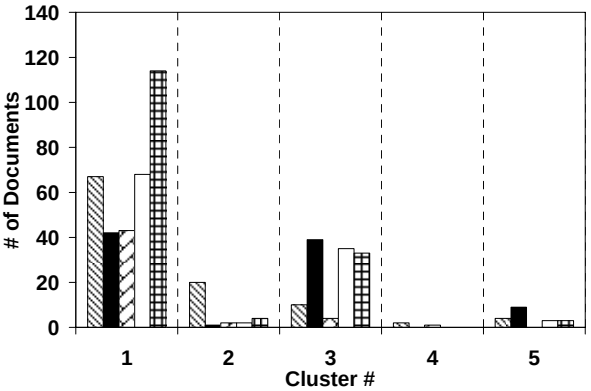


Figure 8: K-medians scaled only at the end

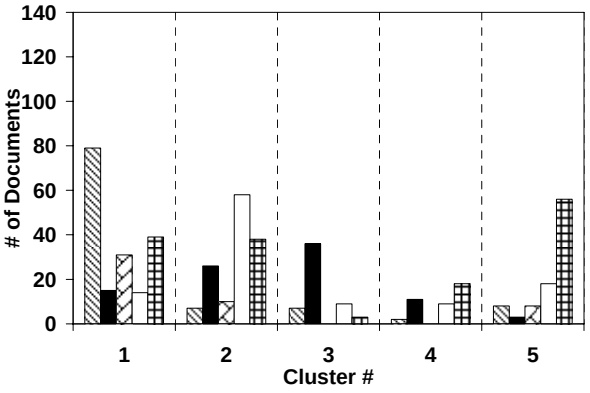
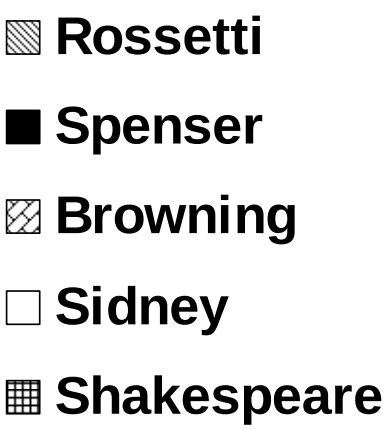


Figure 9: Spherical K-means



I. Text Documents

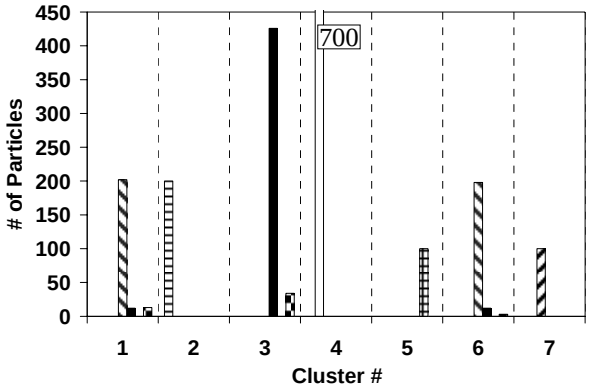


Figure 10: K-medians with MN at each iteration

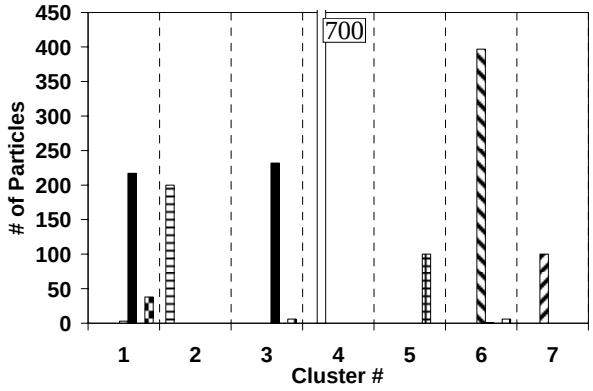


Figure 11: K-medians with MN only at the end

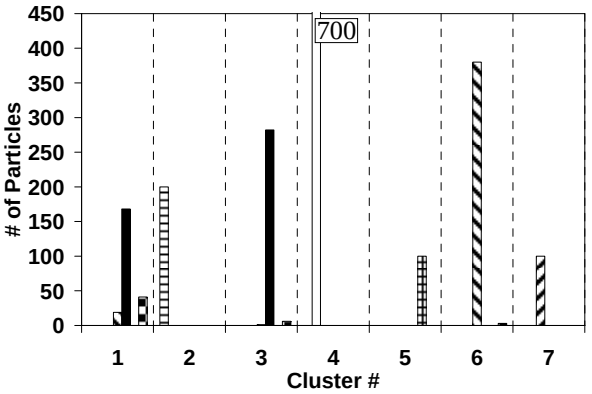


Figure 12: K-medians scaled at each iteration

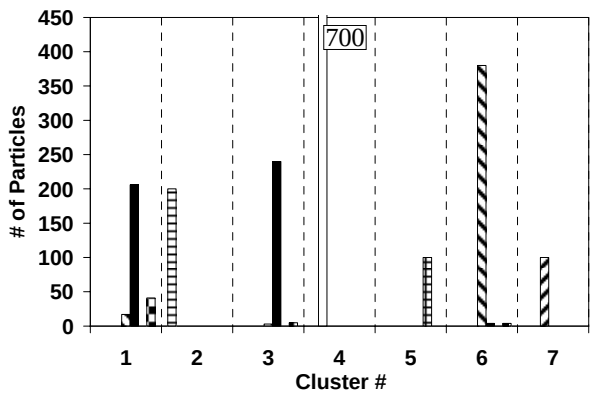


Figure 13: K-medians scaled only at the end

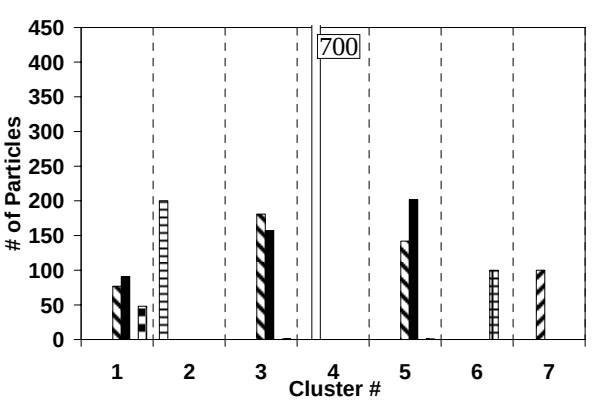
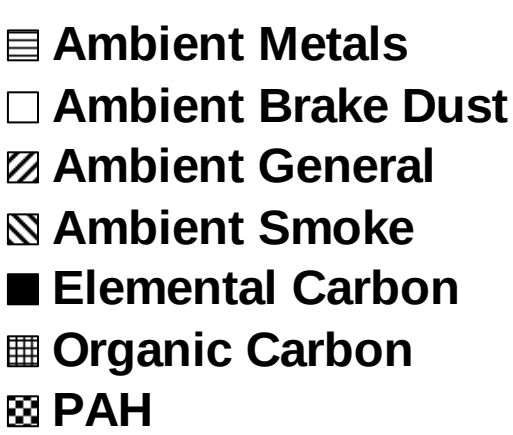


Figure 14: Spherical K-means



II. Synthetic Particles

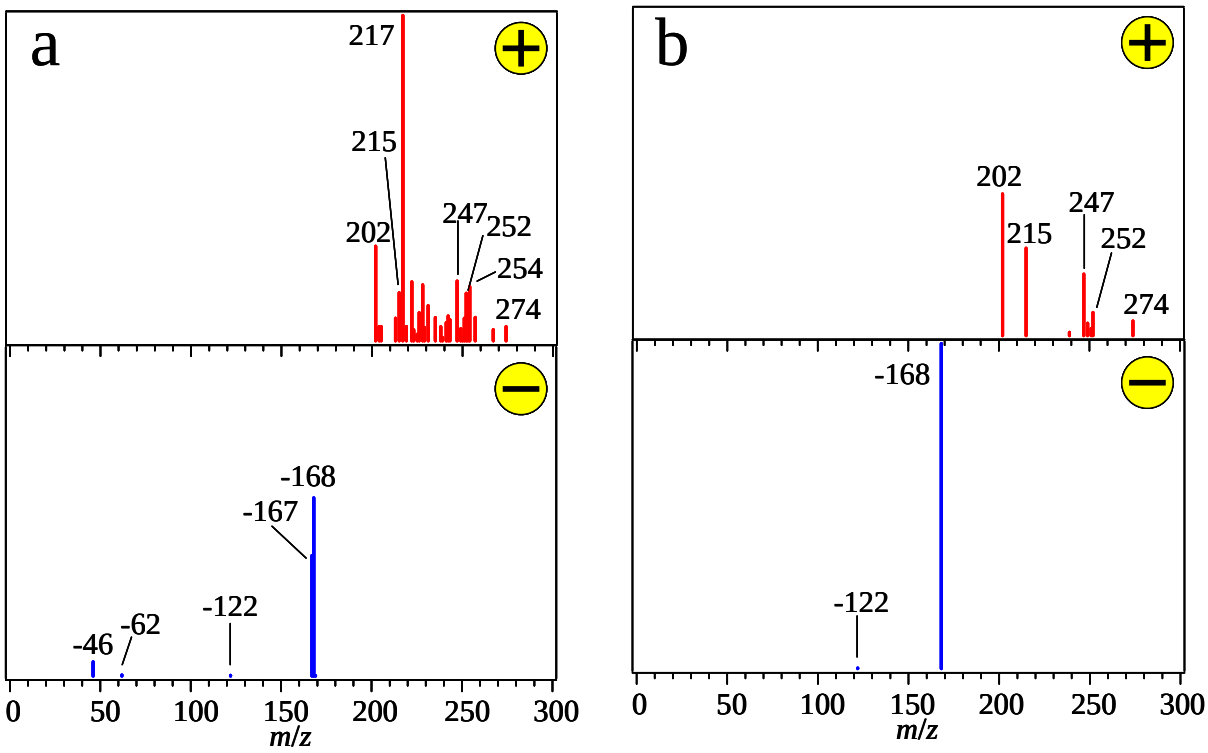


Figure 15: St. Louis cluster samples. Corresponding aerosol particle mass spectra cluster centers from K-medians with MN at each iteration (a) and K-medians scaled at each iteration (b). Of the four K-medians algorithms, only these two produced cluster centers such as these with peaks in the high *m/z* range. K-medians with MN at each iteration shows more of these peaks. Additionally, K-medians with MN at each iteration produces more cluster centers with peaks in negative spectra. These characteristics are more representative of this data.

III. St. Louis Data