The Manhattan Normalization Algorithm

Anna Ritz '06
Carleton College

Clustering Background

Clustering is a technique used to identify trends in large amounts of data without any prior knowledge about what those trends might be. Each cluster contains a "cluster center" that best summarizes that group of points. One dataset might have many appropriate clustering solutions. Figure 1, for example, can be partitioned in multiple ways.

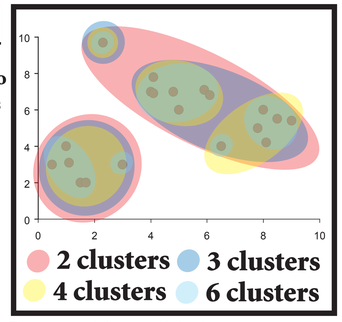


Figure 1: There might be many ways to cluster the same dataset, as shown by the clusters above.

K-Means & K-Medians Clustering Algorithms

0. Choose # of clusters desired, *k*.
1. Choose *k* arbitrary points as initial cluster centers.
2. Assign all other points to the closest cluster center.
   K-Means: Euclidean Squared; K-Medians: Manhattan (see below).
3. Re-evaluate the cluster center.
   K-Means: Use the mean; K-Medians: Use the median.
4. Repeat steps 2 & 3 until cluster centers are considered stable.

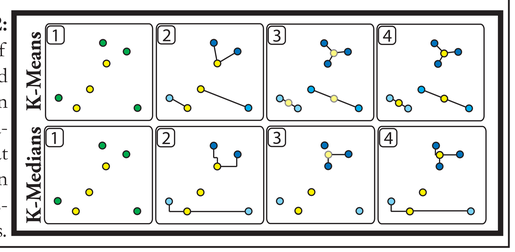


Figure 2: Steps of K-Means and K-Medians on the same dataset. Note that they result in different clusters.

Distance Metric Background

When clustering, you must decide how to measure the distance between points. Two distance metrics we explored are the Euclidean-Squared Distance Metric and Manhattan Distance Metric. K-Means uses the Euclidean Squared distance metric in conjunction with using the mean to re-evaluate clusters, and K-Medians uses the Manhattan distance metric in conjunction with using the median to evaluate cluster centers. Note that Figure 3 displays the Euclidean distance metric, not Euclidean-Squared, because Euclidean is easier to visualize.

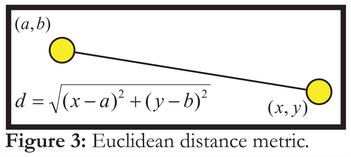


Figure 3: Euclidean distance metric.

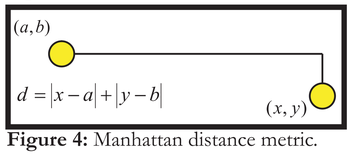


Figure 4: Manhattan distance metric.

Normalization: Why do we need to normalize?

Normalizing means to change a point so its magnitude equals 1. When we do this for all points, relative values along each dimension become the focus of analysis rather than the absolute values. In atmospheric particle research, the relative proportions of "peaks" are more important than the peaks themselves. Figure 5 shows an example particle where the significant data is the *relative* peak height. Normalized data poses a problem for clustering algorithms like K-Means and K-Medians.

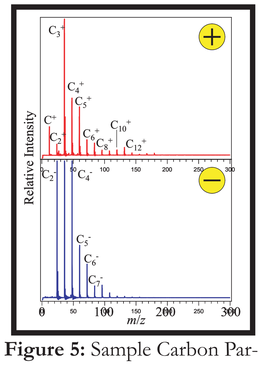


Figure 5: Sample Carbon Particle.

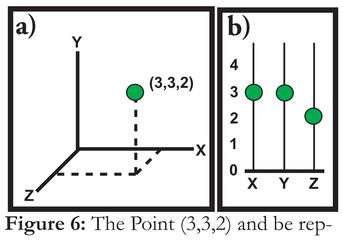


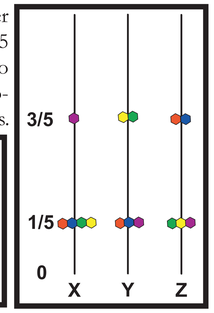
Figure 6: The Point (3,3,2) and be represented on a Cartesian system (**a**) or by isolating the dimensions (**b**).

Consider the Following Cluster...

The points in Figure 7 are normalized. We can visualize them by isolating the dimensions (see the example in Figure 6). Using K-Medians, we want to find the best cluster center. Note that this cluster center should be normalized because the data it is summarizing is normalized. We can also clump all points together, since we are trying to find a prototypical point; they will be grey from now on.

Figure 7: Consider the cluster below of 5 points, which can also be represented on isolated dimensions.

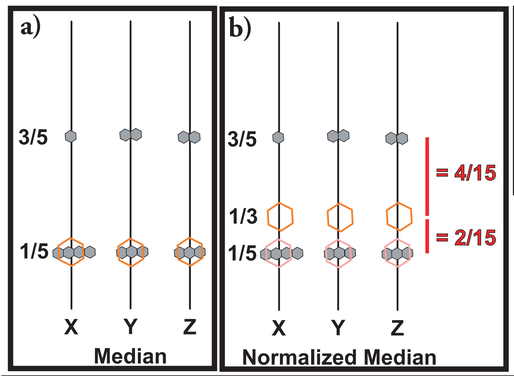
(1/5,1/5,3/5)
(1/5,1/5,3/5)
(1/5,3/5,1/5)
(1/5,3/5,1/5)
(3/5,1/5,1/5)



Q: Does Scaled Normalization Produce the Best Center?

Scaled Normalization is the simple solution to K-Means*. It is proven to produce the best cluster center for that algorithm. Does it work for K-Medians?

1. Find the median of the cluster. If it is normalized, you have your center.
2. If it is unnormalized, scale it so the cluster center magnitude equals 1.
3. Calculate the error by adding up the distance from every grey value to the cluster center value along that dimension (Figure 8).



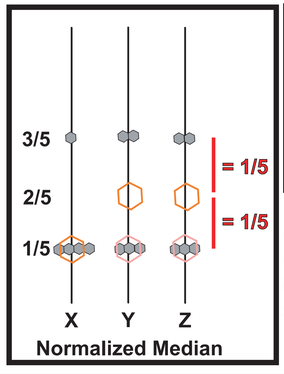
Total Error:
5(4/15)
+ 10(2/15)

= 40/15
= 2.667

Figure 8: The cluster center unnormalized at (1/5,1/5,1/5) (**a**), and normalized to (1/3,1/3,1/3) (**b**). Error shown above.

A: Nope! An Arbitrary Center is Better!

Consider the center (1/5,2/5,2/5). It is normalized, and it has a lower error than the center achieved with Scaled Normalization (Figure 9). This means that the arbitrary center is a better summary of the points, showing that there is at least one center that is better than the Scaled Normalization approach.



Total Error:
1(2/5)
+ 10(1/5)

= 12/5
= 2.4

Figure 9: The normalized cluster center (1/5,2/5,2/5) has a lower error than scaled normalization. Error shown above.

MN Algorithm's Goal

Given a cluster of points, the MN Algorithm finds a cluster center that has the lowest error possible for K-Medians.

The crux of the algorithm lies in the basic concept of sliding points along a dimension (Figures 10 & 11). As you slide a cluster center along a dimension, the amount of error incurred depends on the number of points above and below it. Error decreases as you slide the cluster center towards the pole with more points.

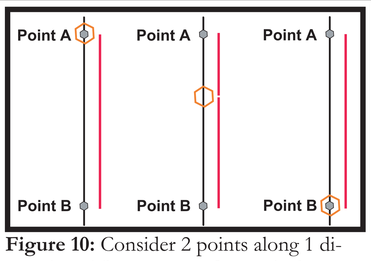


Figure 10: Consider 2 points along 1 dimension. The amount of error is the same regardless of cluster center location.

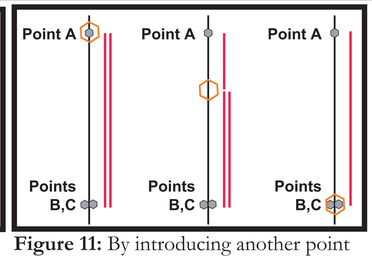
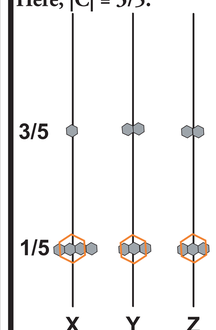


Figure 11: By introducing another point (point C), the error decreases as you slide the cluster center towards points B and C.

The MN Algorithm, Explained

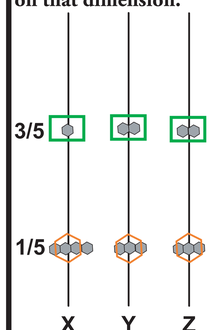
The following example is a walk-through of the MN Algorithm. There are some subtleties (for example, the magnitude of the center can be greater than 1 in step 1, and all the inequalities will reversed), yet every cluster can be simplified to an example such as this.

1. Initialize cluster center C to be the median.
If |C| = 1, we are done.
Here, |C| = 3/5.



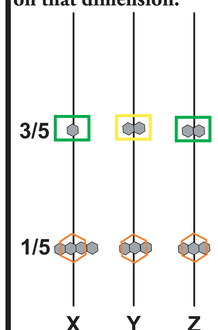
C = (1/5,1/5,1/5)
|C| = 3/5

2. For each dimension, find the number of values strictly greater than the center value on that dimension.



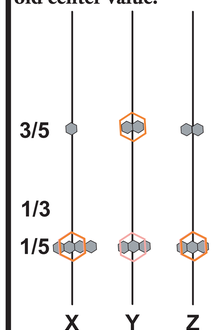
= 1 Value	= 2 Values	= 2 Values

3. Choose the dimension that has the most values stricly greater than the center value on that dimension.



Y & Z both have 2 vals.
Pick one arbitrarily.

4. Redefine the center value on that dimension to be the smallest value greater than the old center value.



C = (1/5,3/5,1/5)
|C| = 1

5. Terminating Cond.
If |C| = 1, we are done.
If |C| < 1, repeat steps 2 and 3.
If |C| > 1, adjust last defined value so |C| = 1.

Here, |C| = 1, so we are done.

Error Comparison

Scaled Normalization:
2.667
Arbitrary Center:
2.4
MN Algorithm:
2.0

K-Medians Experiment 1: Relative Word Frequency

We have introduced the MN algorithm; now we need to see if it performs well on actual datasets. One way to do this is to have a dataset where we can predict what the resulting clusters should look like. We developed a dataset of thousands of sonnets written by 5 different authors. For each sonnet, we counted the *relative* word frequency (so the data is normalized). Our assumption is that similar sonnets will have the same author; thus, each author should be a cluster.

We produced graphs (Figure 12) to show the breakdown of each cluster. For the 5 clusters shown, groups consisting of primarily one author are better than groups with significant amounts of different authors. Here, the MN algorithm was able to distinguish authors better than the Scaled Normalization algorithm for K-Medians.

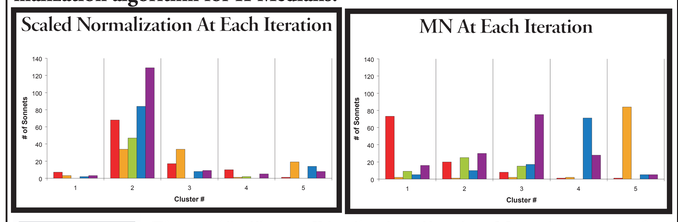


Figure 12: Cluster Distribution Graphs. Note that most of the authors in the Scaled Normalization graph ended up in Cluster 2, while the MN graph shows groups of Rossetti sonnets, Shakespeare sonnets, Sidney sonnets, and Spenser sonnets in Clusters 1,3,4, and 5 respectively.

K-Medians Experiment 2: Aerosol Data

Our research mainly involves atmospheric particle, or *aerosol*, data. Figure 5 is an example of an aerosol. Considering each peak as a dimension, we can cluster a normalized aerosol dataset. We clustered with the Scaled Normalization and the MN algorithms; here, however, we do not know what the clusters should look like, as in Experiment 1. Instead of Cluster Distribution Graphs, we rely on error measurements (the distance from all points to their closest cluster center). We assume that the lower the error, the better the clusters.

Figure 13 shows the amount of error per iteration for an aerosol dataset collected in St. Louis in 2002. As the number of iterations increase, the error decreases, meaning that the clusters are becoming a better fit for the data. In this graph, the MN algorithm's error is consistently lower than the Scaled Normalization's error, demonstrating MN's optimal performance for finding cluster centers.

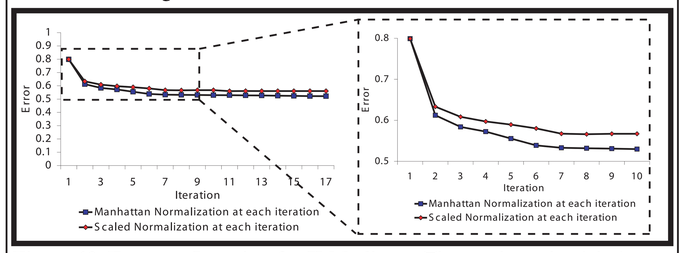


Figure 13: Error vs. Iteration for K-Medians. The graph on the right is an enlargement of the one on the left. Each algorithm ran for 17 iterations, and they both started with an error of approximately 0.8. Note that the MN algorithm is consistently lower than the Scaled Normalization algorithm, though they follow the same curve and eventually level off.

Acknowledgements

First and foremost, this research has been a team effort. Special thanks to the members of my research group: Ben Anderson '05, Leah Steinberg '07, and Thomas Smith '07. Special thanks to our advisor, Professor Dave Musicant, as well. The Chemistry research team at Carleton College advised by Professor Deborah Gross has also helped us tremendously with our aerosol analysis. This research is part of an NSF grant shared by UW-Madison Computer Science and Chemistry research groups advised by Professors Raghu Ramakrishnan and James Schauer, respectively.

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