Generating Normalized Cluster Centers with KMedians

Introducing the Manhattan Normalization (MN) Algorithm

Benjamin J. Anderson (Microsoft)
Deborah S. Gross (Carleton College)
David R. Musicant (Carleton College)
Anna M. Ritz (Carleton College)
Thomas G. Smith (Carleton College)
Leah E. Steinberg (Carleton College)

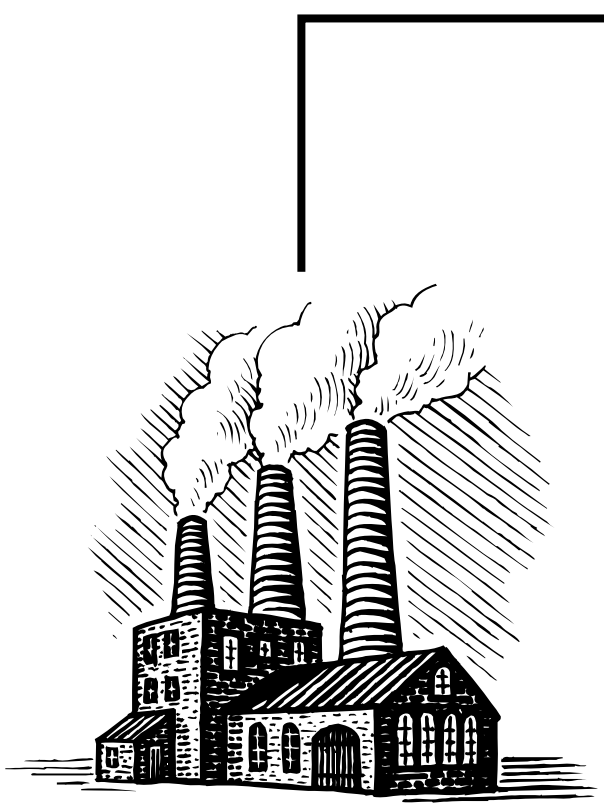
Northfield, MN

Contents

- Motivation & Background
- Problem with KMedians
- Our Solution: The MN Algorithm
- Experiments

Motivation

- Atmospheric particle research
- Aerosol Time-Of-Flight Mass Spectrometer

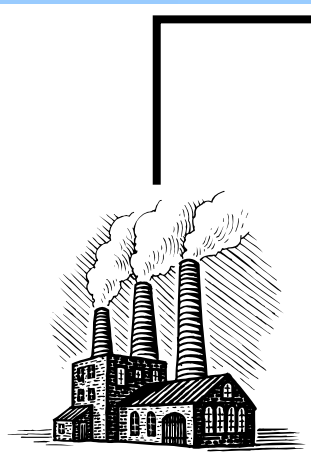


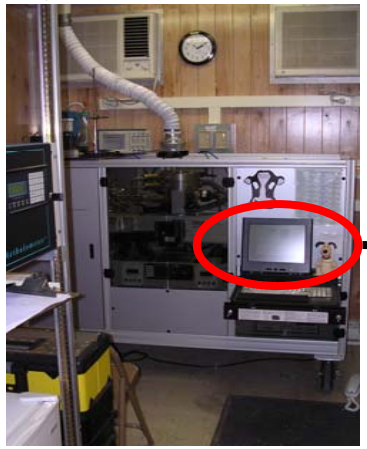
Smoke Particle

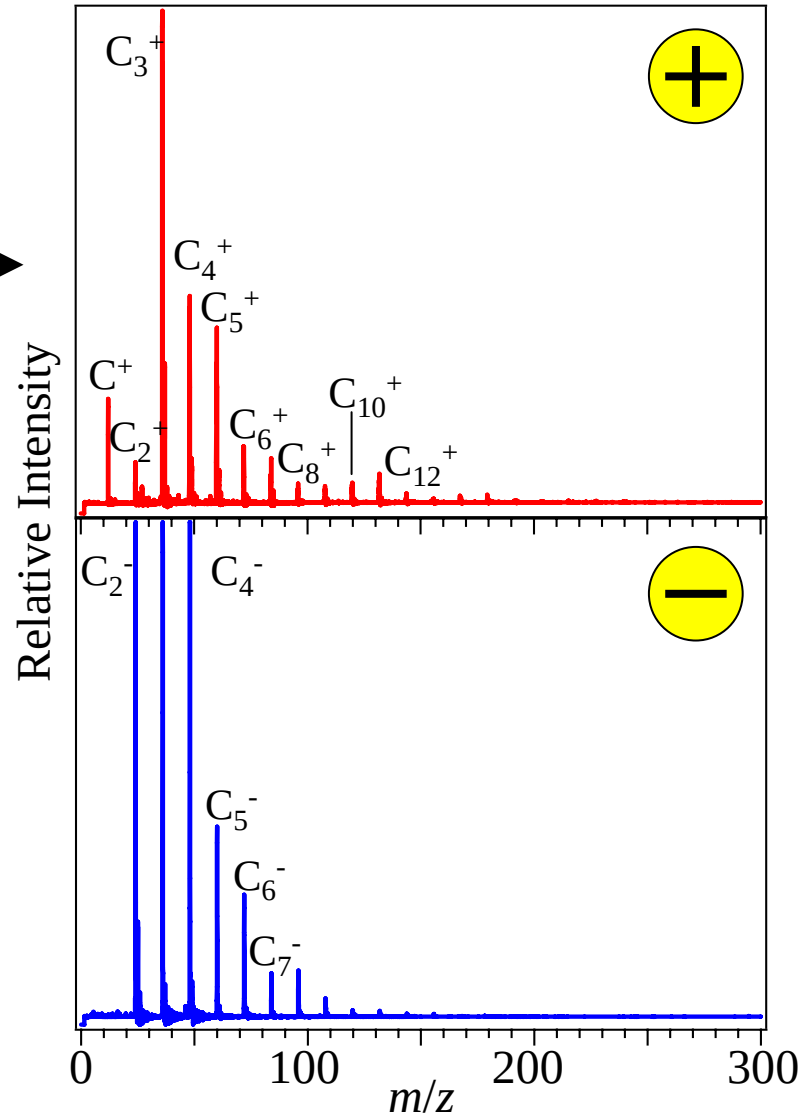
Motivation

- Particle Composition Analysis
- Why Cluster?
  - Large Datasets (1,000 particles per minute)
  - Learn about particle *types*
- Art2a: “Standard” clustering algorithm for chemists
- KMeans/KMedians untested

Atmospheric Particle Data



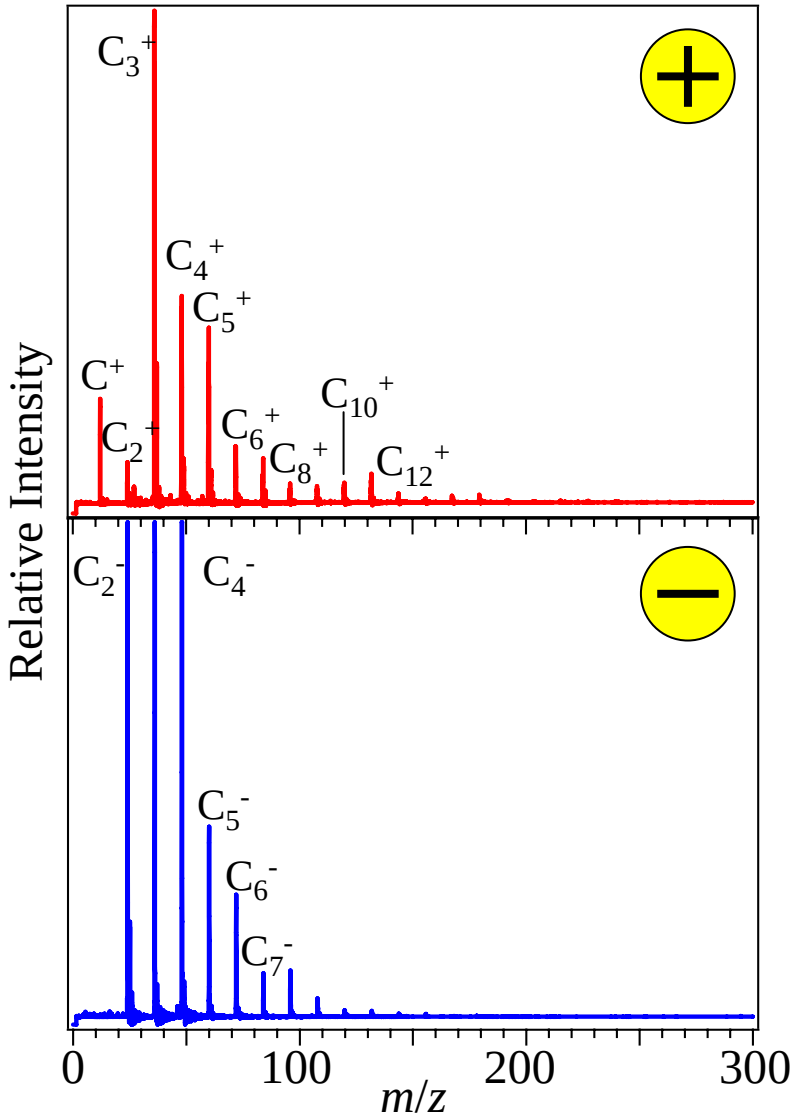




- Mass Spectrum
- Two parts (+ and -)
- Each x-axis value is a dimension

Normalized Data

- Normalize: Adjust the scale so the magnitude of the particle's vector equals 1
- Interested in values *relative* to each other
- Normalized data = normalized cluster centers

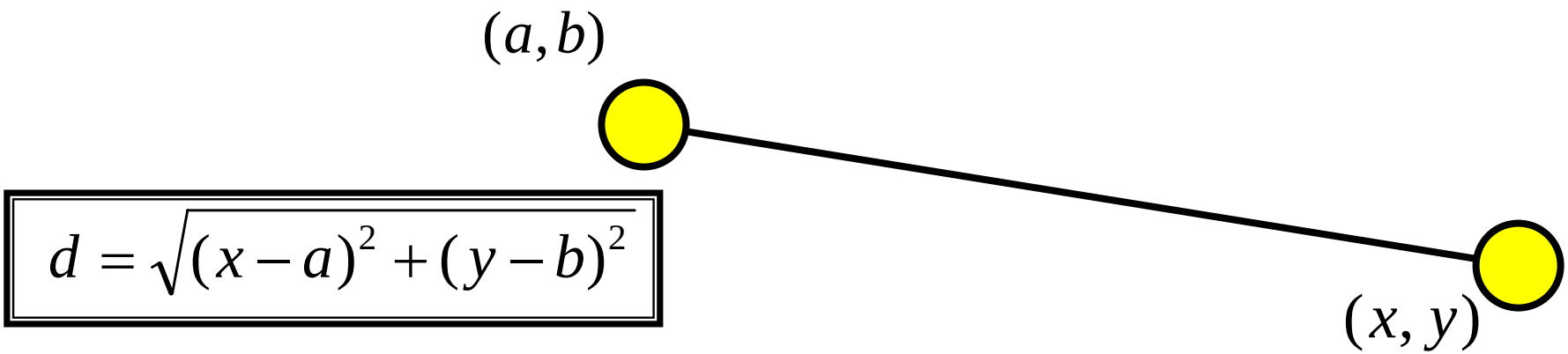


Standard data

- First consider non-normalized data
- Want to give chemists ability to use
  - KMeans
  - KMedians

KMeans : A Review

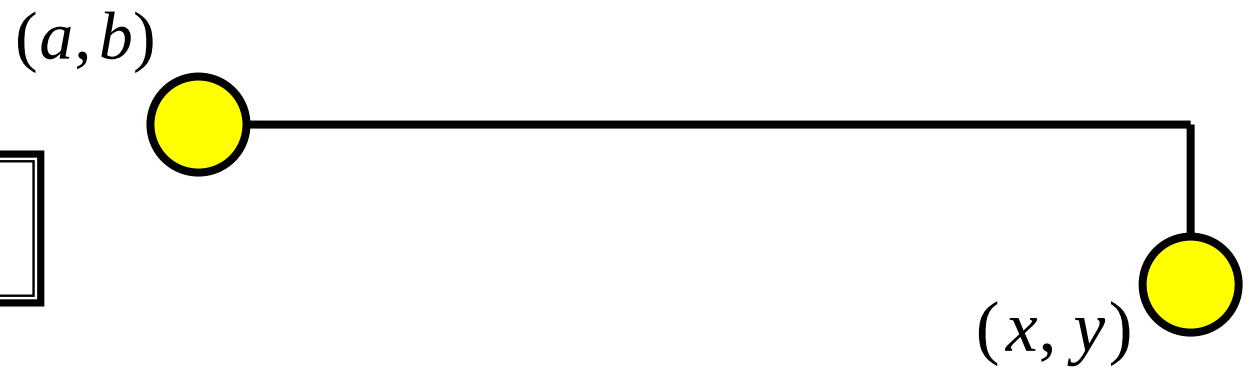
- Input parameter *k* = number of clusters algorithm will produce
- Cluster center = mean (points in cluster)
- The mean of a cluster minimizes the *Euclidean Squared* (*2-norm)* distance:



KMedians

- If you want to make algorithm more robust to outliers, you would want to use the median
- Cluster center = median (points in cluster)
- The median of a cluster minimizes the *Manhattan* (*1-norm*) distance (also known as *City Block*):

$$d = |x - a| + |y - b|$$

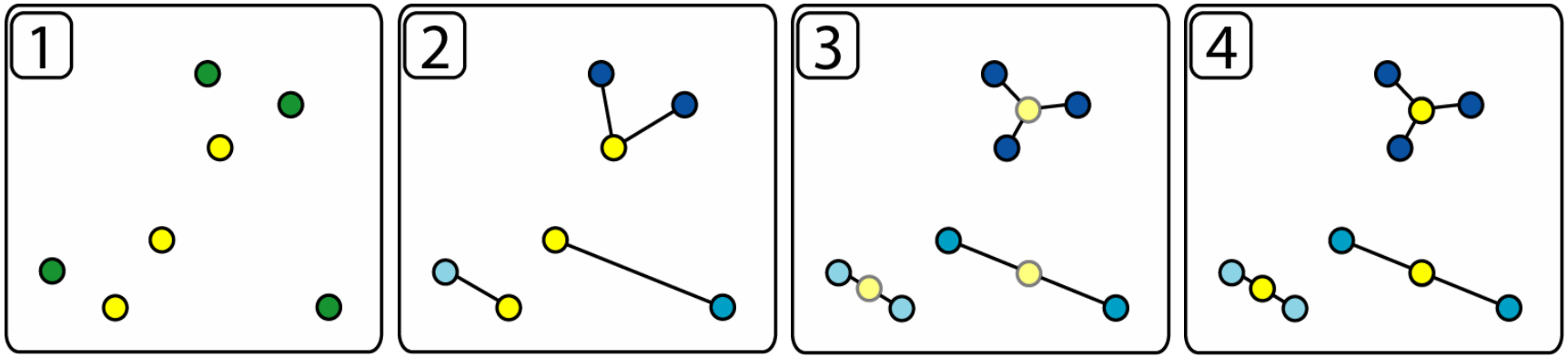


KMeans / KMedians Example

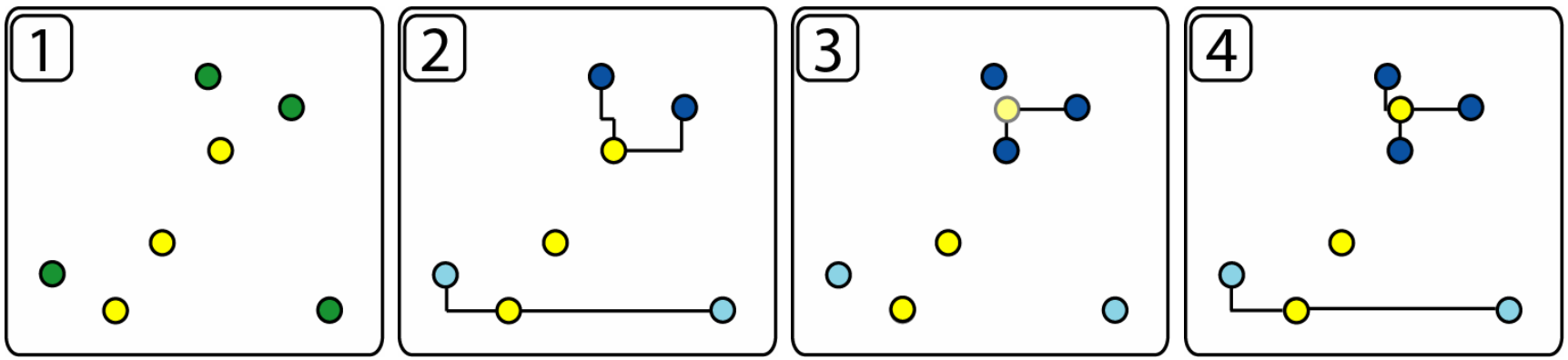
Need to know *k*, the number of clusters desired. *k* = 3.

1. Choose *k* arbitrary points as initial cluster centers.
2. Assign all other points to closest cluster center.
3. Re-evaluate the cluster center.
4. Repeat steps 2 & 3 until clusters are considered stable.

KMeans



KMedians



Setting up the Problem

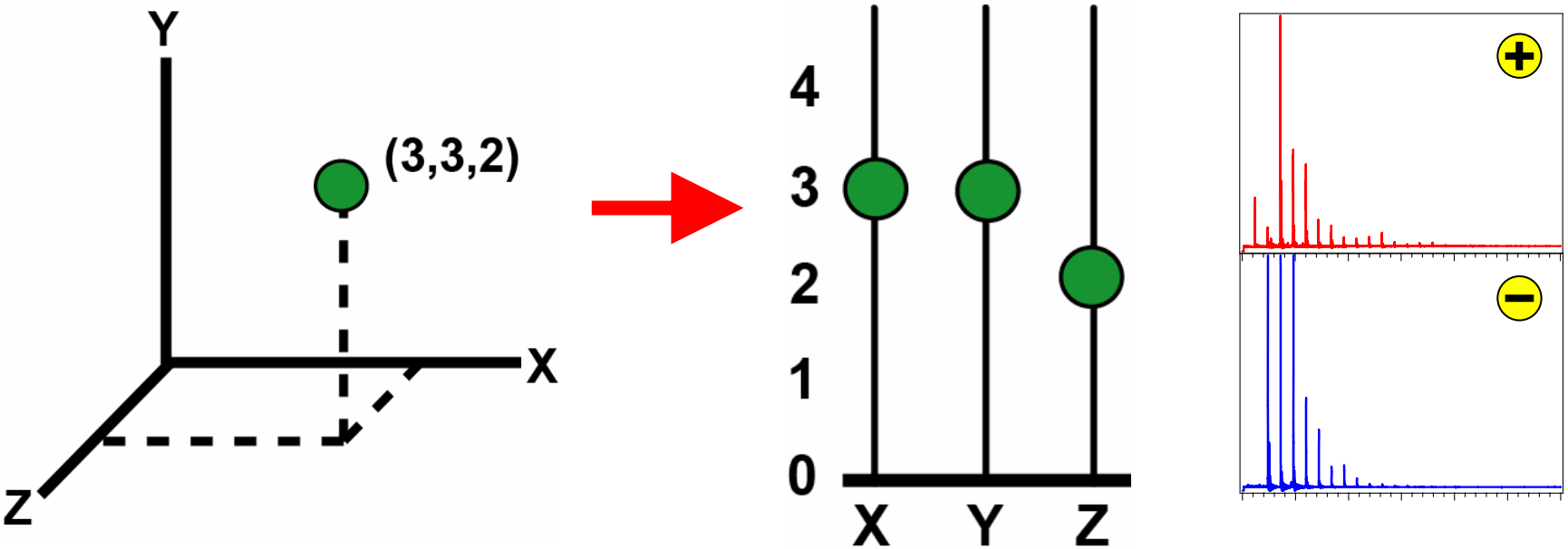
	Standard Data	Normalized Data
Euclidean Squared	KMeans	“Spherical KMeans” (Dhillon & Modha, 2001)
Manhattan	KMedians	??

Contents

- Motivation & Background
- Problem with KMedians
- Our Solution: The MN Algorithm
- Experiments

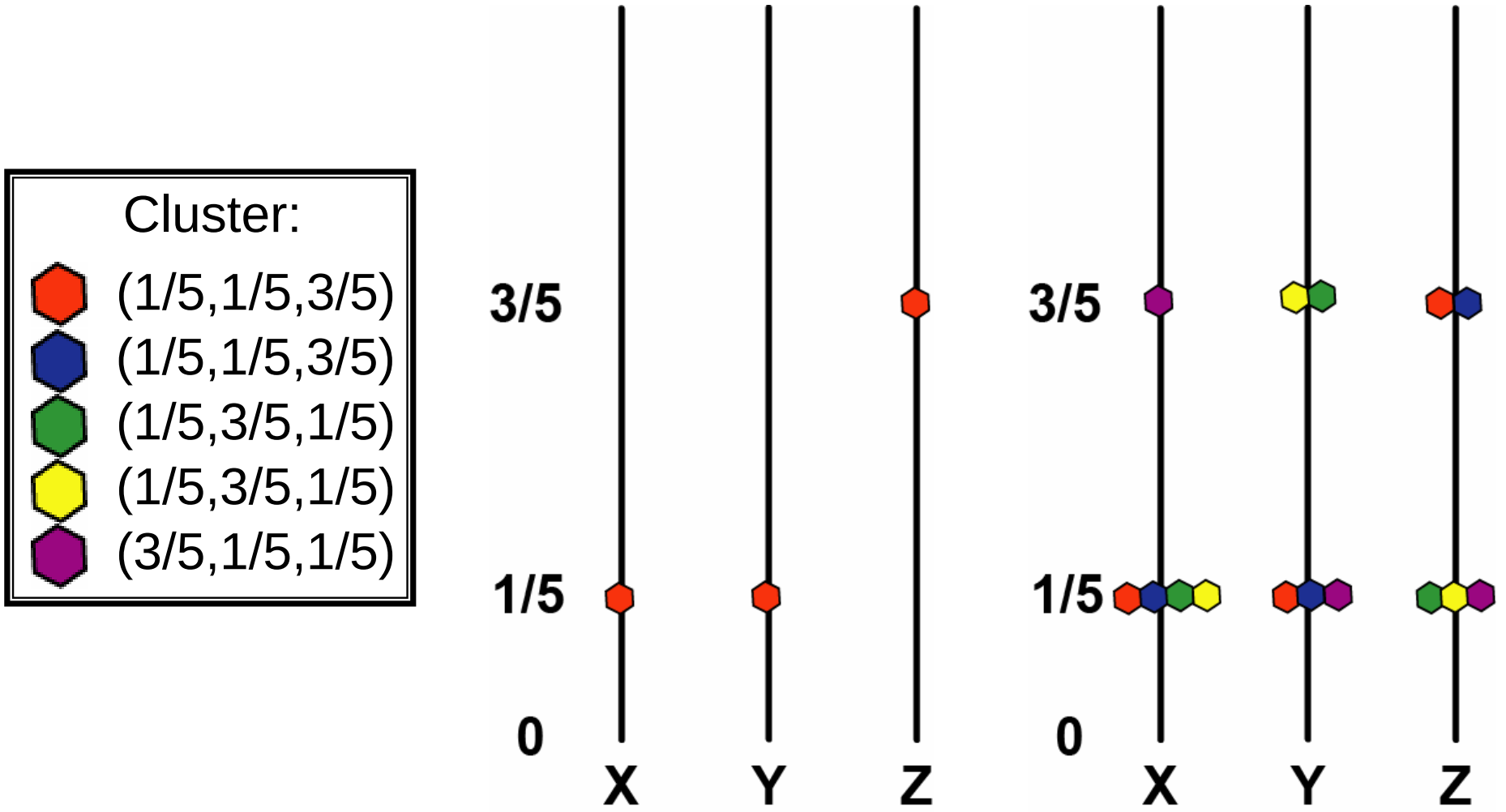
Setting up the Problem

- **Observation:** Manhattan distance is computed independently along each axis
- A different way of viewing dimensions
  - **Example:** Plot point (3,3,2)

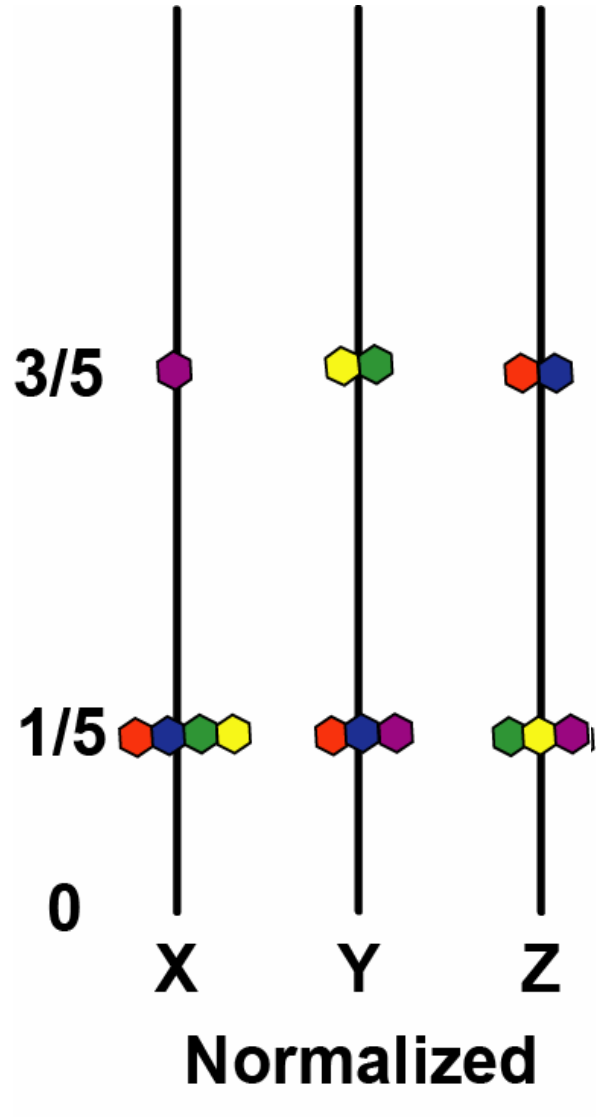


Example

GOAL: to find an optimal *normalized* cluster center

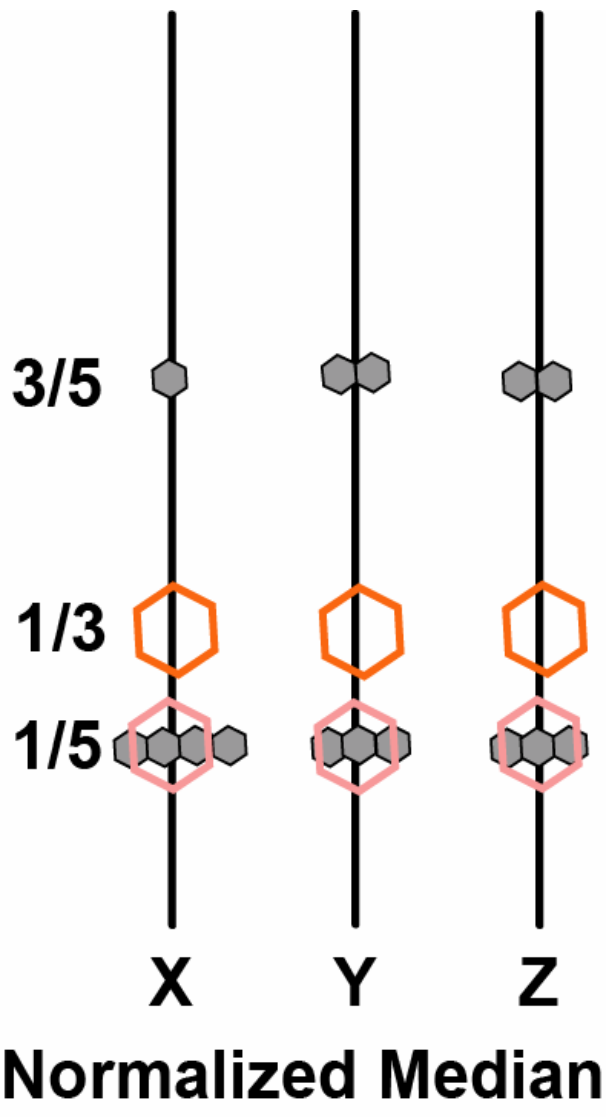


Example

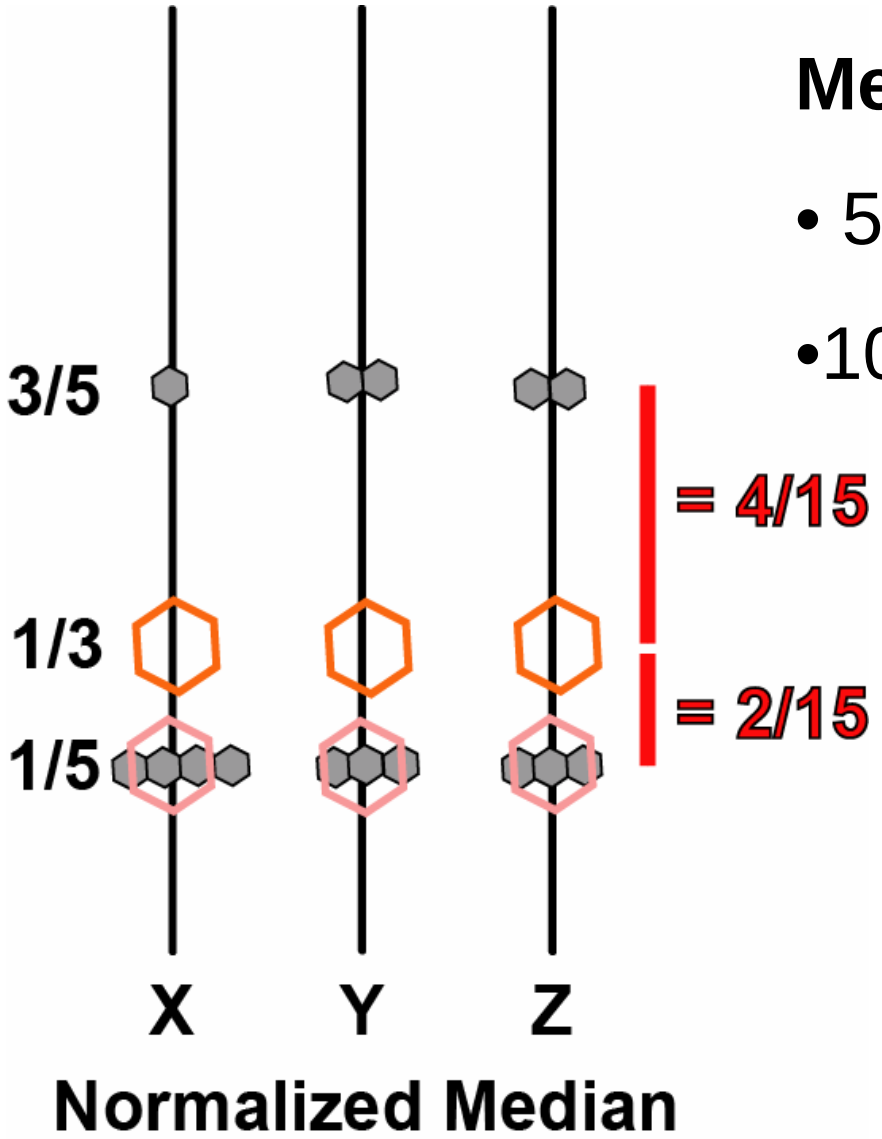


Median:
(1/5,1/5,1/5)

Norm. Median:
(1/3,1/3,1/3)



Example



Measure the Error:

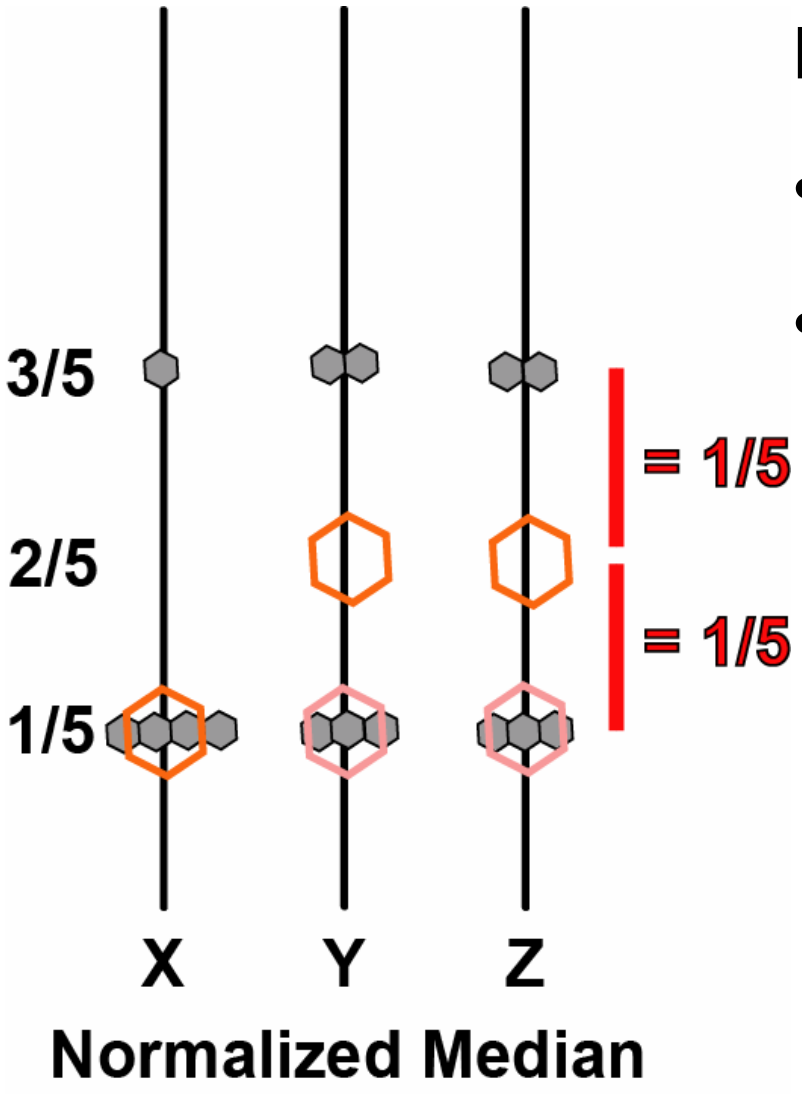
- 5  have an error of 4/15
- 10  have an error of 2/15

Total Error:

5(4/15)
+ 10(2/15)

= 40/15
= 2.667

Example



Measure the Error:

- 1  has an error of 2/5
- 10  have an error of 1/5

Total Error:

1(2/5)
+ 10(1/5)

= 12/5
= 2.4

Setting up the Problem

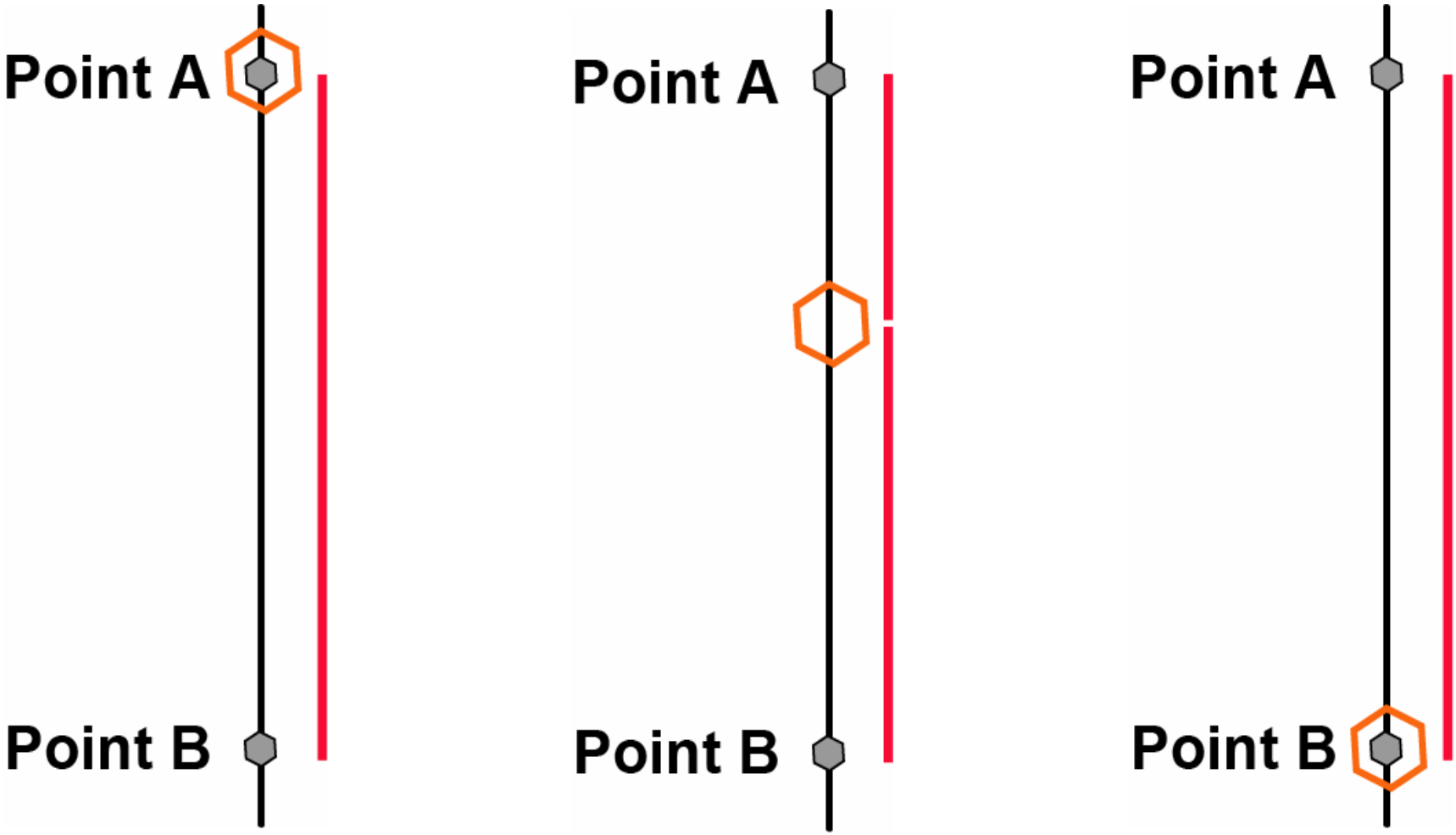
	Standard Data	Normalized Data
Euclidean Squared	KMeans	"Spherical KMeans" (Dhillon & Modha, 2001)
Manhattan	KMedians	**"Manhattan Normalization"**

Contents

- Motivation & Background
- Problem with KMedians
- Our Solution: The MN Algorithm
- Experiments

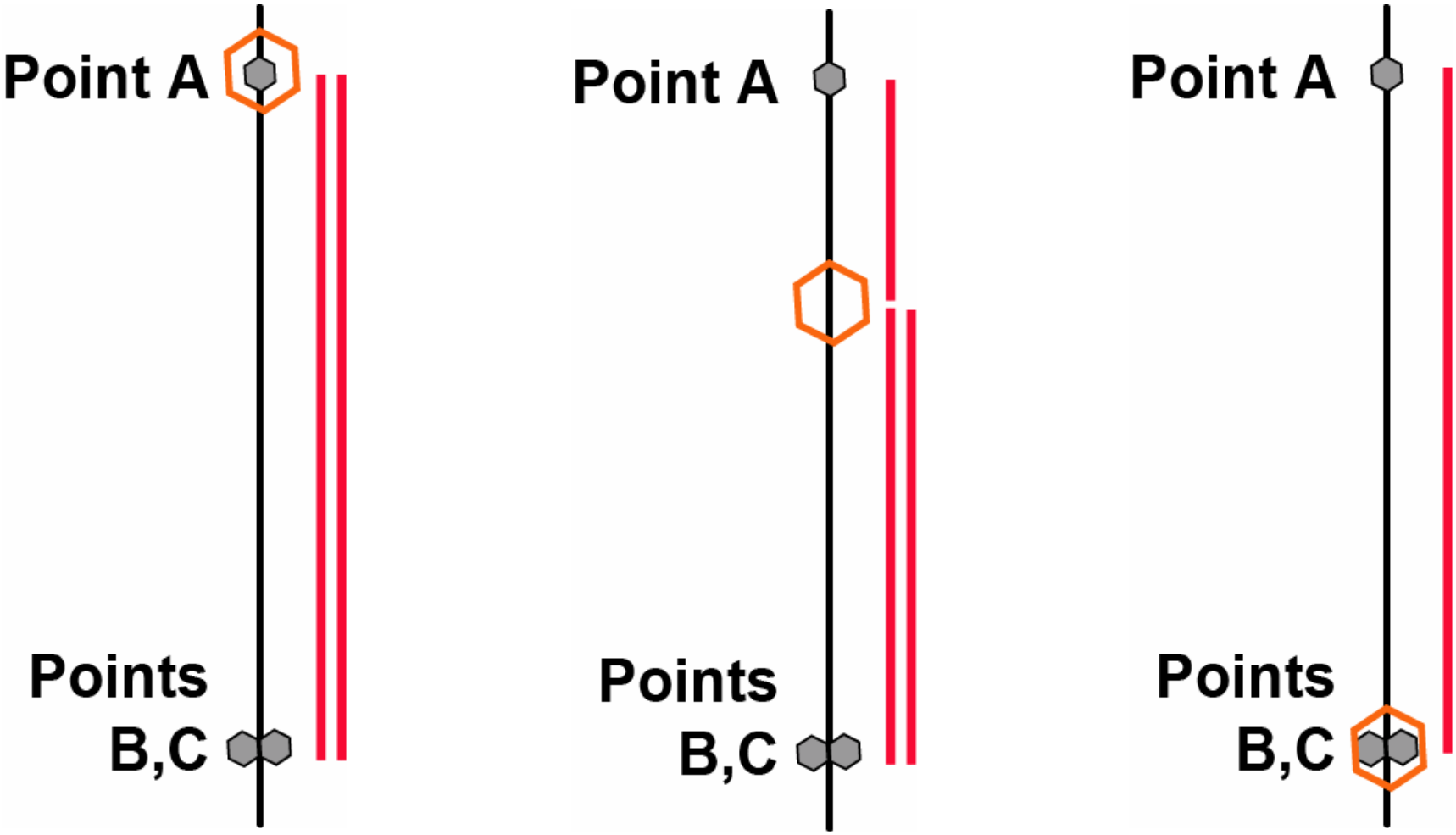
The Crux of the MN Algorithm

Consider 2 points along 1 dimension



The Crux of the MN Algorithm

Add another point (Point C)



The MN Algorithm

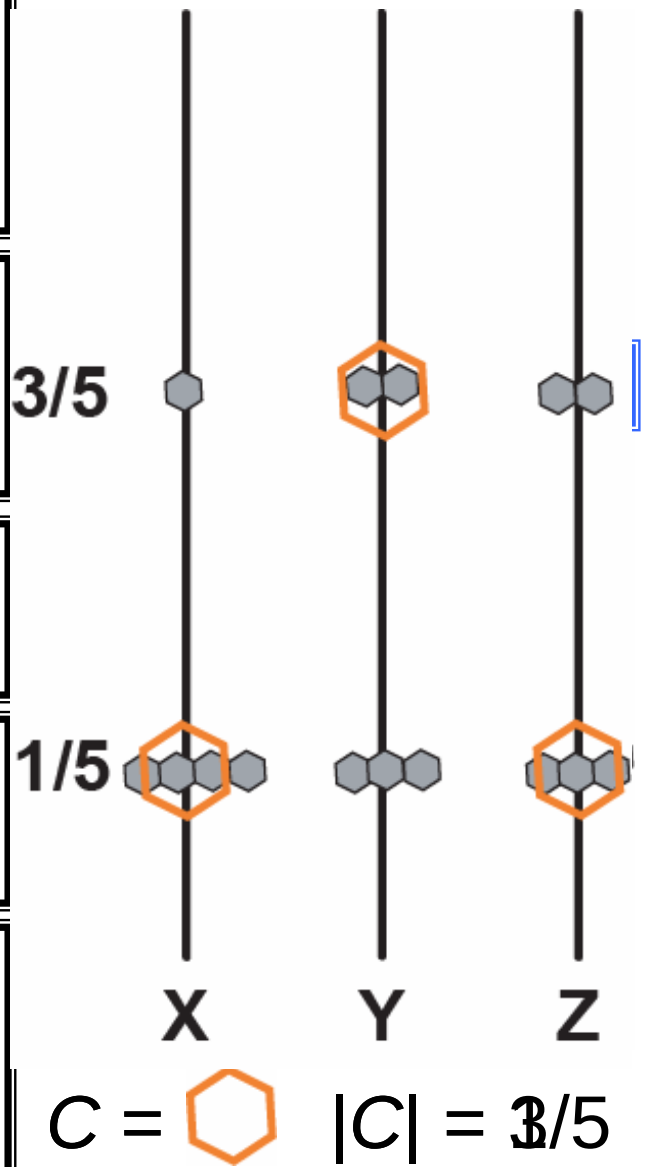
1. Initialize cluster center C to be the median of the cluster. If $|C| = 1$, we are done.

2. Find the number of values that are strictly greater than C_m for each dimension.

3. Find the dimension that has the most values above C_m.

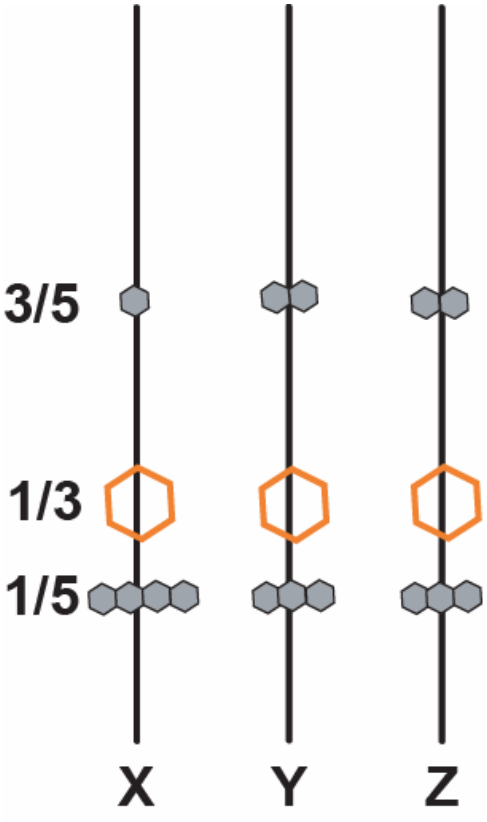
4. Redefine C_m to be the smallest value greater than the old C_m.

5. If $|C| = 1$, we are done.
   - If $|C| < 1$, go to step 2.
   - If $|C| > 1$, redefine C_m so $|C| = 1$.

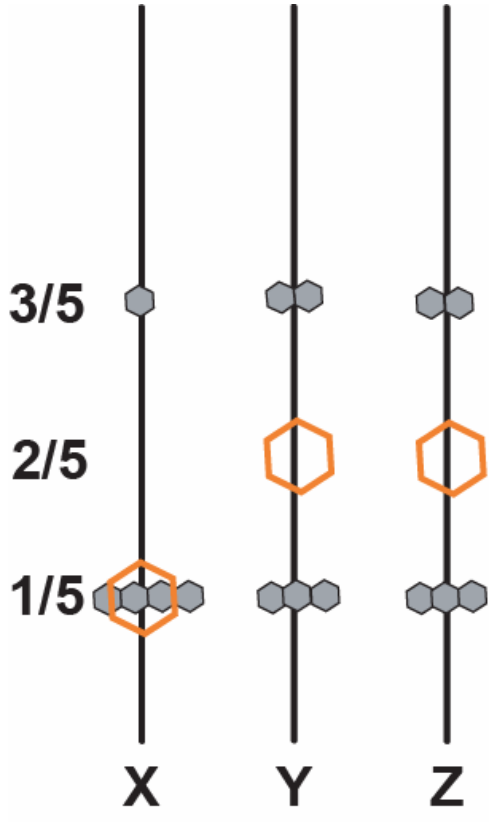


Compare with Scaled Norm.

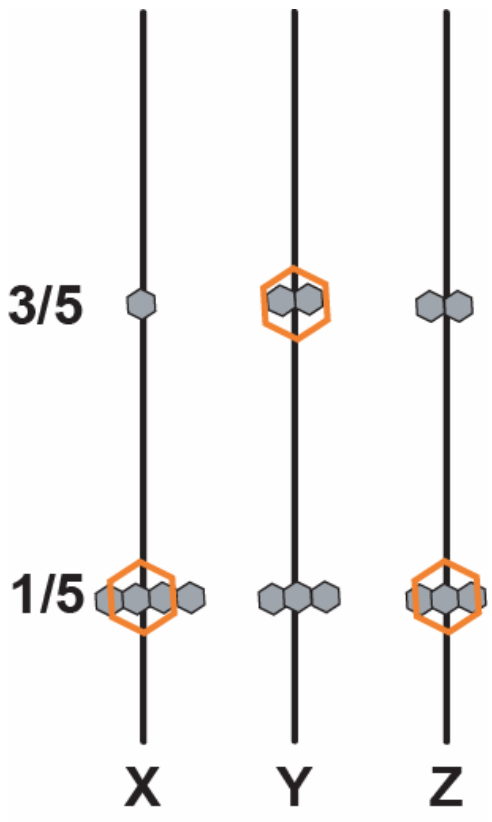
Scaled
Cluster Center



Chosen
Cluster Center



Manhattan Normalized
Cluster Center



Error = 2.667

Error = 2.4

Error = 2.0

Theorem

Given a set of points $x_1, x_2, \ldots, x_n$ where $||x_i||1=1$, $i=1,\ldots,n$, the MN algorithm finds a point c ($||c||1=1$) that minimizes the total 1-norm error from all points to it.

Contents

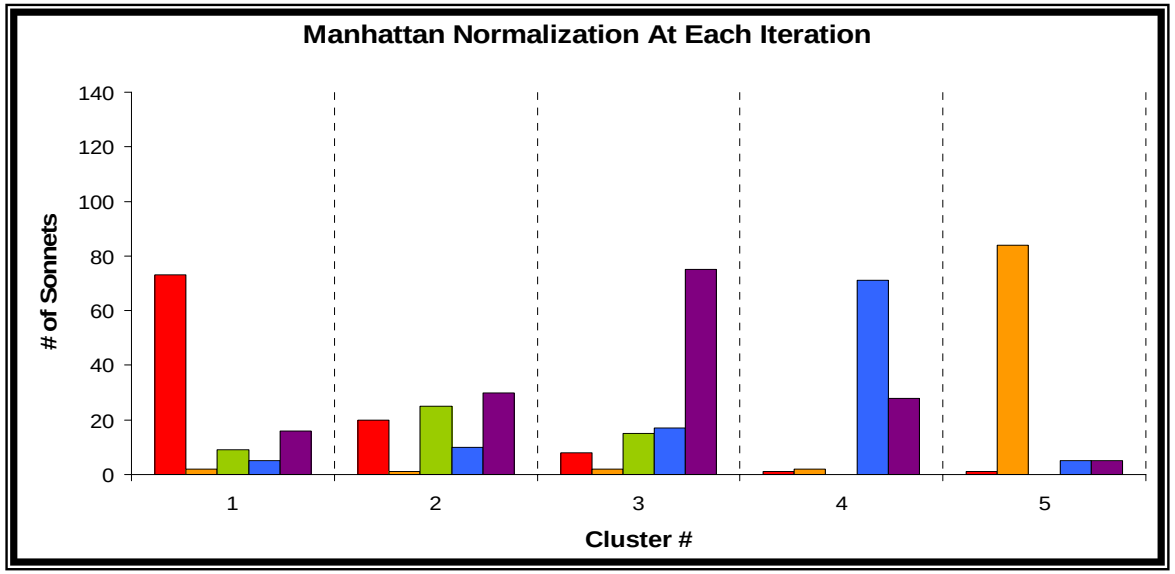
- Motivation & Background
- Problem with KMedians
- Our Solution: The MN Algorithm
- Experiments

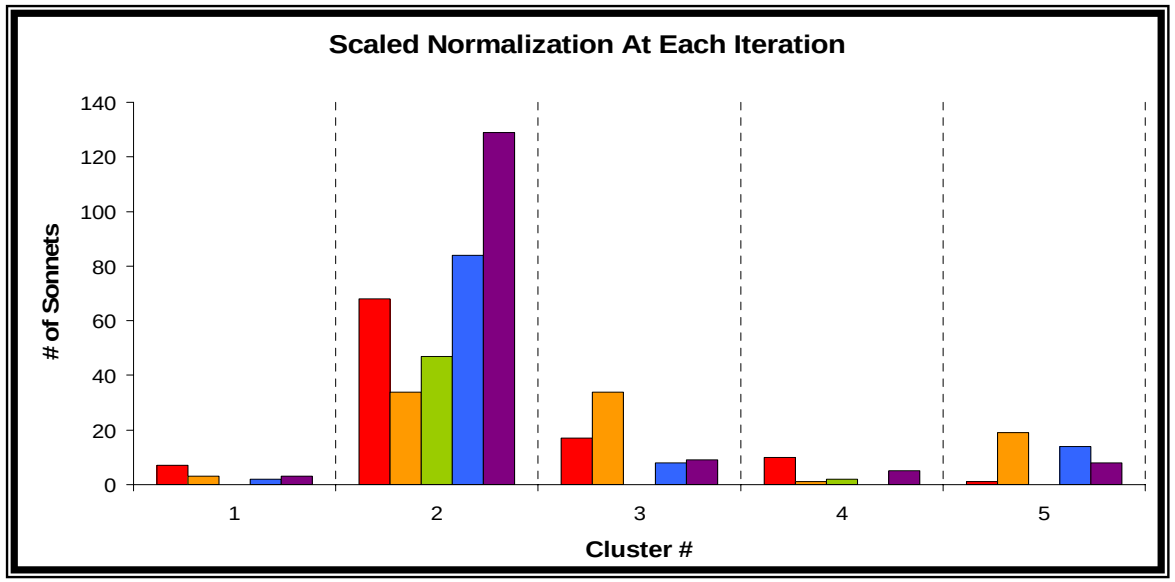
Experiment 1: Word Frequency

- 5 different authors, hundreds of sonnets
- *Relative* word frequency
- Assumption - Similar sonnets have the same author
- We know how the clusters should look
- Tests how “good” the cluster results are

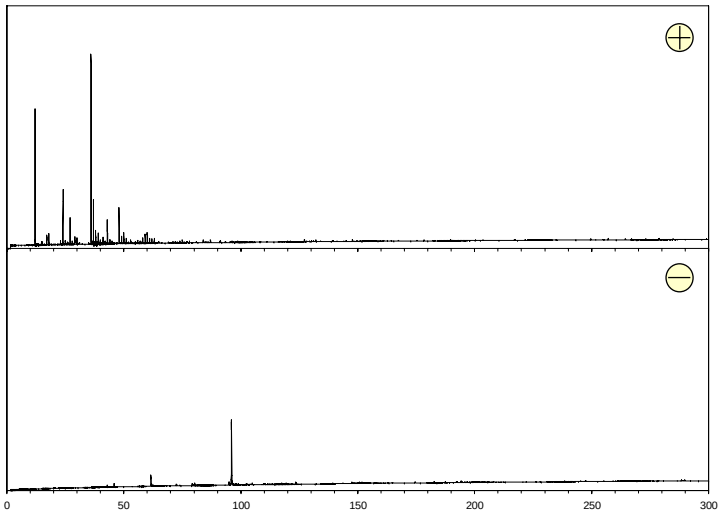
Experiment 1: Word Frequency

Cluster Distribution Graphs

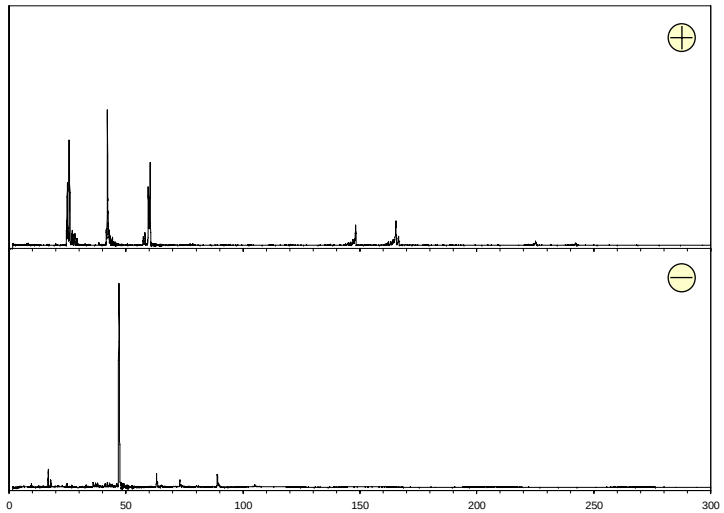




Experiment 2: Aerosols



Smoke (St. Louis)

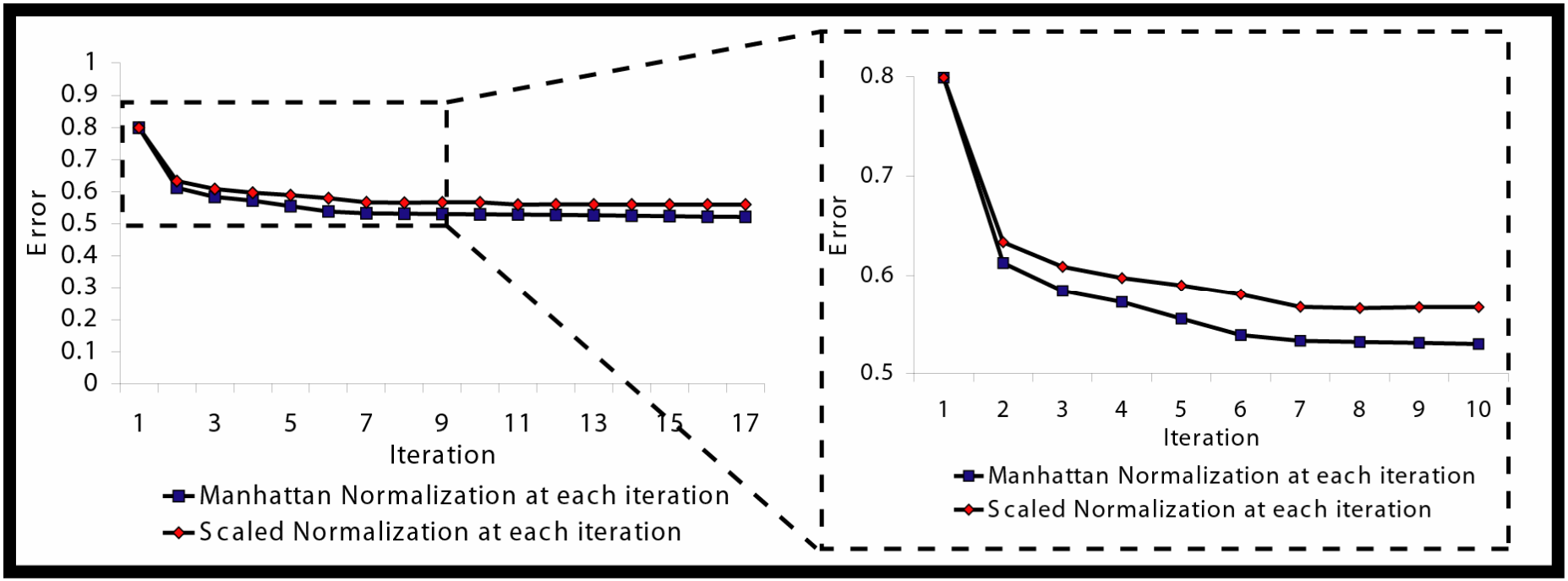


Brake Dust (Atlanta)

- 2004 dataset; roughly 3,000 particles
- Cannot predict clustering results
- Rely on error to judge algorithm
  - Error: The sum of the distance of all points in a cluster to the cluster center.

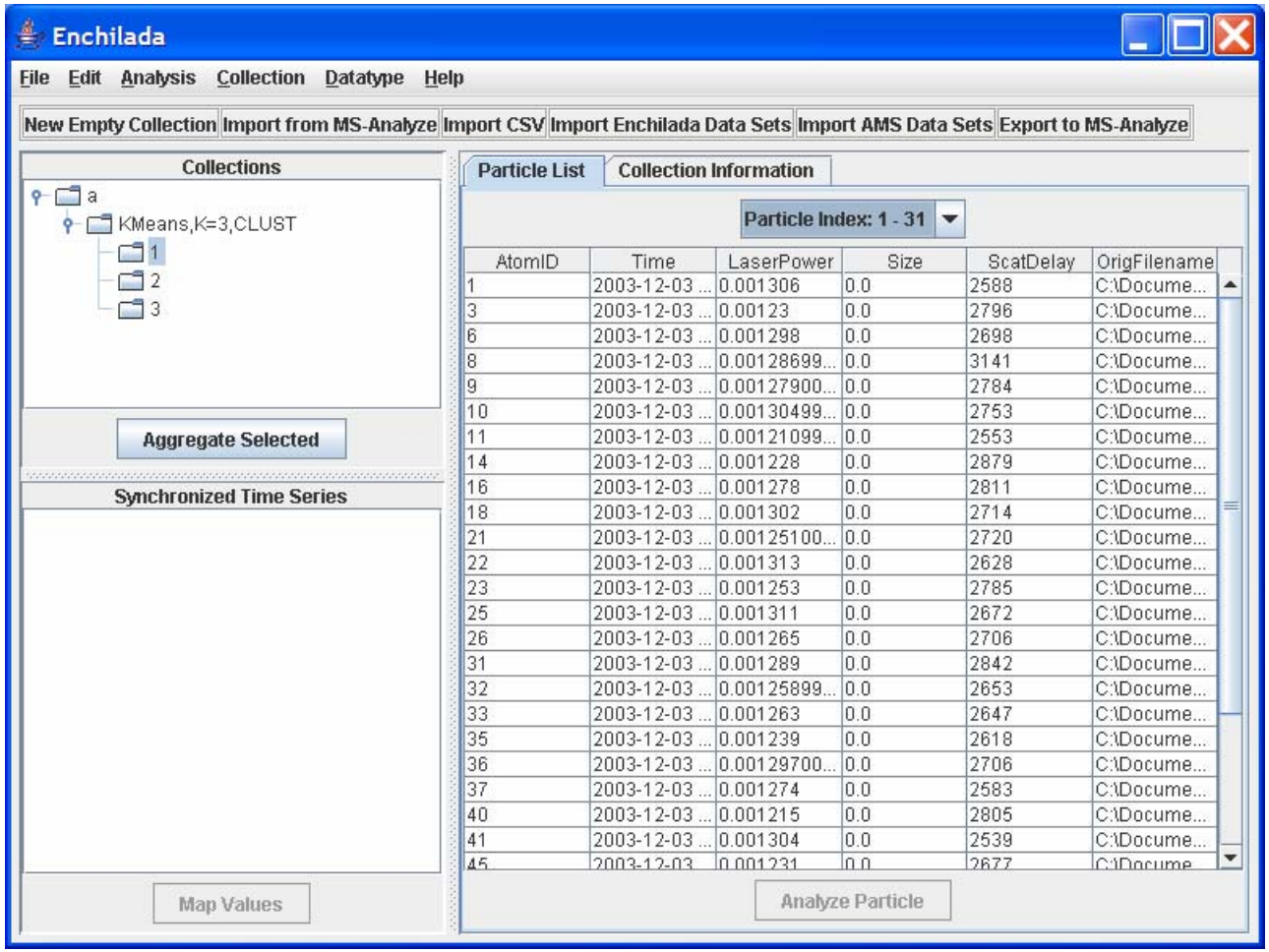
Experiment 2: Aerosols

Error vs. Iteration for KMedians



- 17 iterations, error decreases with each iteration
- MN consistently lower than Scaled Normalization for this dataset

Further Study



Acknowledgements

Thanks to the following groups:

The Carleton CS research group led by Professor David Musicant:

- Ben Anderson ’05
- Thomas Smith ’07
- Leah Steinberg ’07

The Carleton Chemistry research group led by Professor Deborah Gross:

- Melanie Yuen ‘06
- John Choiniere ‘07
- Katie Barton ‘07

The UW-Madison graduate research team led by Professor Raghu Ramakrishnan