CSE/ISE 332
Introduction to Visualization

Data Reduction

Klaus Mueller

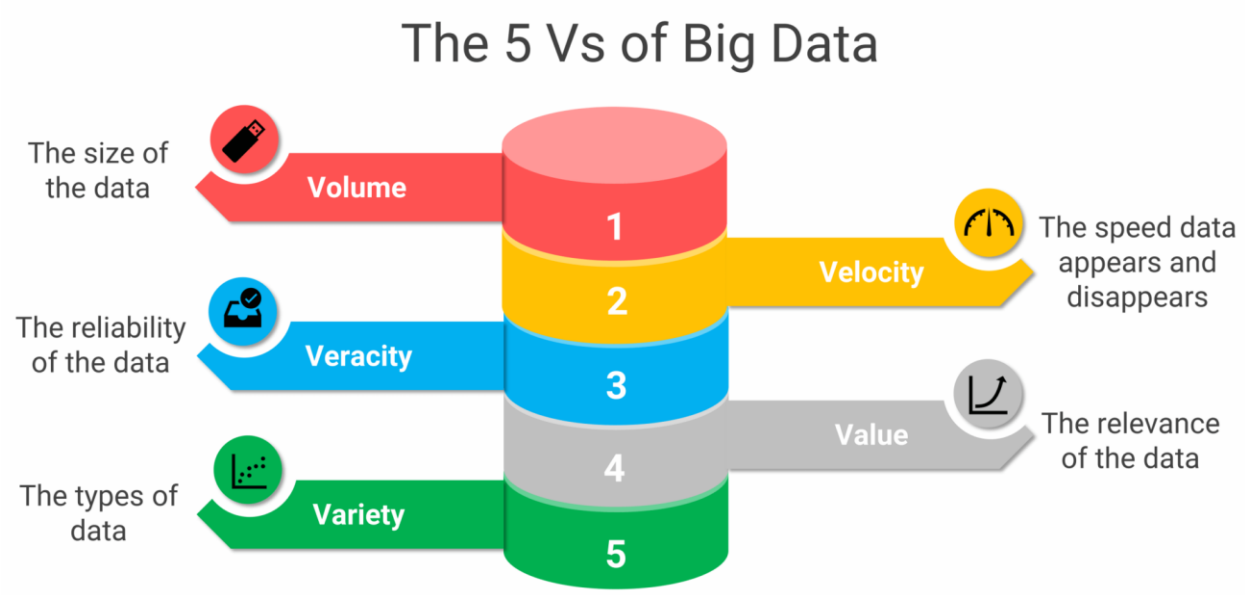
Computer Science Department
Stony Brook University

Lecture	Topic	Projects
1	Intro and logistics	
2	Basic visualizations and tasks, data types, examples, ethical considerations	
3	Data preparation (cleaning, imputation, data set integration)	Project #1 out
4	AI-assisted coding for VIS applications (design, debugging, refactoring)	
5	Big data and data reduction (distance/sim metrics, intro to clustering)	
6	High-D data: concept, subspaces, dimension reduction, PCA	
7	Cluster analysis: hierarchical, density, model, embedding, temporal	Project #2(a) out
8	Perception and cognition (human visual system, color, contrast)	
9	Visual design and aesthetics	
10	Visualization of multivariate and high-D data: linear methods, projections	
11	Vis. of multivariate and high-D data: non-linear methods, embeddings	Project #2(b) out
12	Visualization and AI: mutual support and capabilities (VIS4AI, AI4VIS)	
13	Principles of interaction: drive what is visualized, analyzed & how (HCI4VIS)	
14	Midterm recap	
15	Midterm	
16	Visual analytics, human-centered AI, mixed-initiative & collaborative VA	Capstone proposal due
17	Visualization system design & evaluation: nested model, study designs	
18	Visualization of hierarchical data	
19	Visualization of maps and data with geo-reference	
20	Vis. of time-varying, time-series, streaming data, progressive visualization	
21	Critiquing visualization: Tufte's views, integrity, uncertainty, responsibility	Capstone prelim report due
22	Design of effective infographics	
23	Foundations scientific & medical visualization, computer graphics	
24	Intro to volume rendering	
25	Scientific visualization	
26	Capstone project demos (4-5 minutes + 2 minutes Q&A)	All capstone materials due
27	Capstone project demos (4-5 minutes + 2 minutes Q&A)	
28	Capstone project demos (4-5 minutes + 2 minutes Q&A)	
Final	Final exam	

Big Data

Big data analytics

- Refers to the process of collecting, storing, analyzing, and extracting value from big data
- Hard to deal with big data unredacted
- Need to be sensitive/mindful how and what to redact



Data Reduction – How?

Reduce the number of data items (samples):

- random sampling
- stratified sampling

Reduce the number of attributes (dimensions):

- dimension reduction by transformation
- dimension reduction by elimination

Usually do both

Utmost goal

- keep the gist of the data
- only throw away what is redundant or superfluous
- it's a one way street – once it's gone, it's gone

Data Reduction

Sampling

- random
- stratified

Data summarization

- binning (already discussed)
- clustering (see a future lecture)
- dimension reduction (see next lecture)

Data Reduction – Why?

Because…

- need to reduce the data so they can be feasibly stored
- need to reduce the data so a mining algorithm can be feasibly run

What else could we do

- buy more storage
- buy more computers or faster ones
- develop more efficient algorithms (look beyond O-notation)

However, in practice, all of this is happening at the same time

- unfortunately, the growth of data and complexities is always faster
- and so, data reduction will always be important

Which Samples to Discard?

Good candidates are *redundant* data

- how many cans of ravioli will you buy?

Sampling Principles

Keep a representative number of samples:

- pick one of each
- or maybe a few more depending on importance

How to Pick?

You are faced with collections of many different data

- they are usually not nicely organized like this:

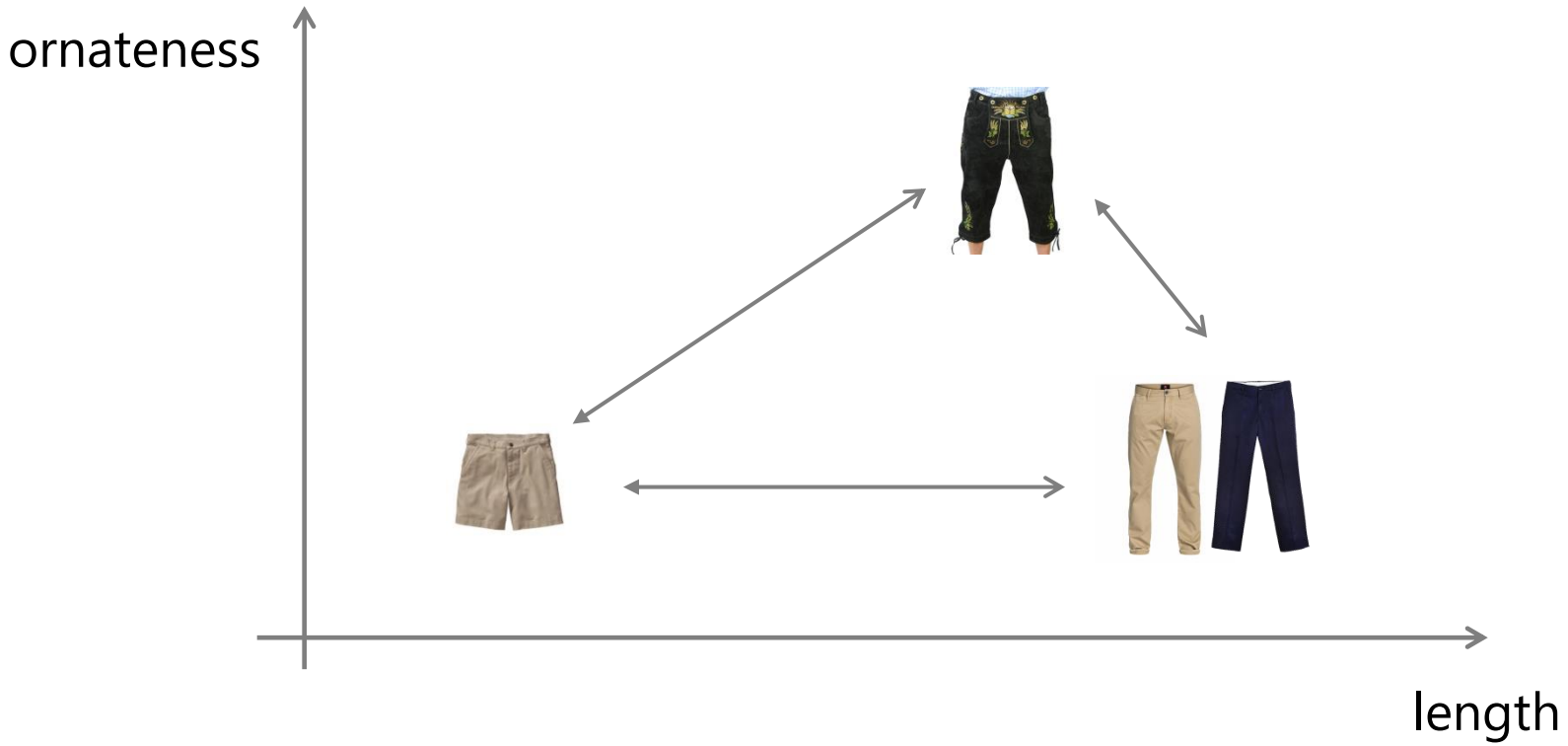
- but more like this:

Measure of Similarity

Are all of these items pants?

- need a measure of similarity
- it's a distance measure in high-dimensional feature space

Feature Space



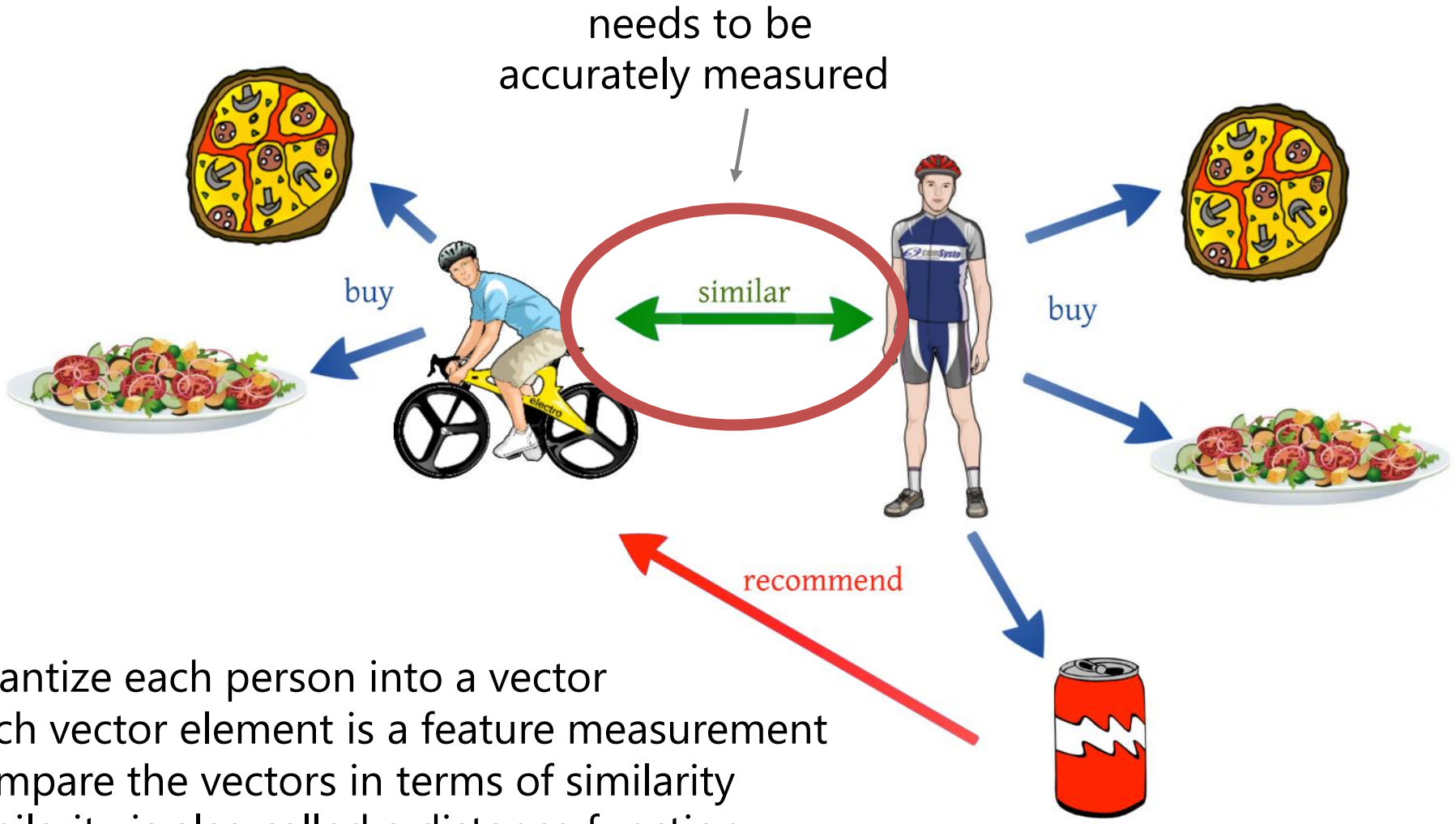
We did not consider color, texture, size, etc…

- this would have brought more differentiation (blue vs. tan pants)
- the more features, the better the differentiation

How Many Features Do We Need?

Measuring similarity can be difficult

Back To Similarity Functions



quantize each person into a vector
each vector element is a feature measurement
compare the vectors in terms of similarity
similarity is also called a distance function

Data Vectors

Pant:

<length, ornateness, color>

Food delivery customer:

<type-pizza, type-salad, type-drink>

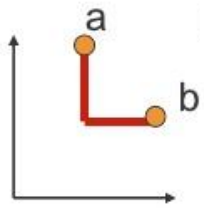
Examples:

- pants: <long, plain, tan>, <short, ornate, blue>, …
  expressed in numbers: <30”, 1, 2>, <15”, 2, 5>

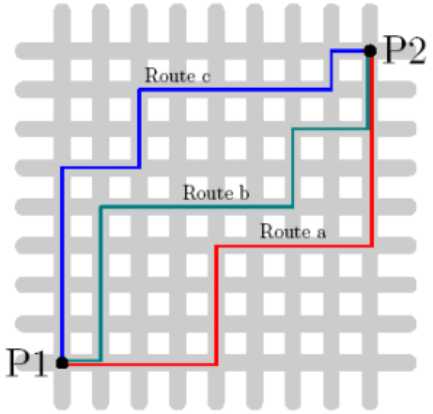
- food: <pepperoni, tossed, none>, <pepperoni, tossed, coke>, …
  expressed in numbers: <1, 1, 0>, <1, 1, 3>

Metric Distances

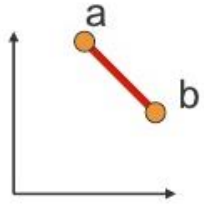
Manhattan distance



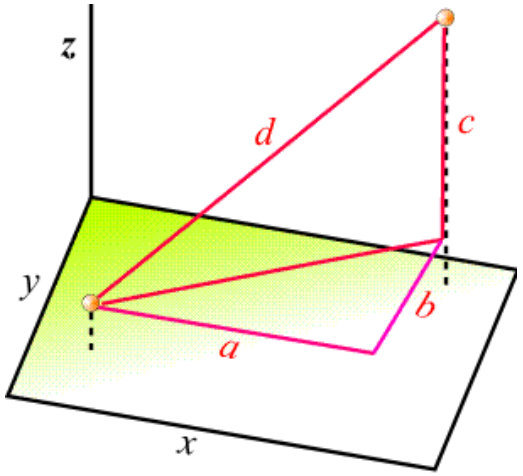
$$\text{dist(a,b)} = \|a - b\|_1 = \sum_i |a_i - b_i|$$



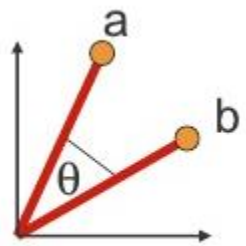
Euclidian distance



$$\text{dist(a,b)} = \|a - b\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$$



Cosine Similarity

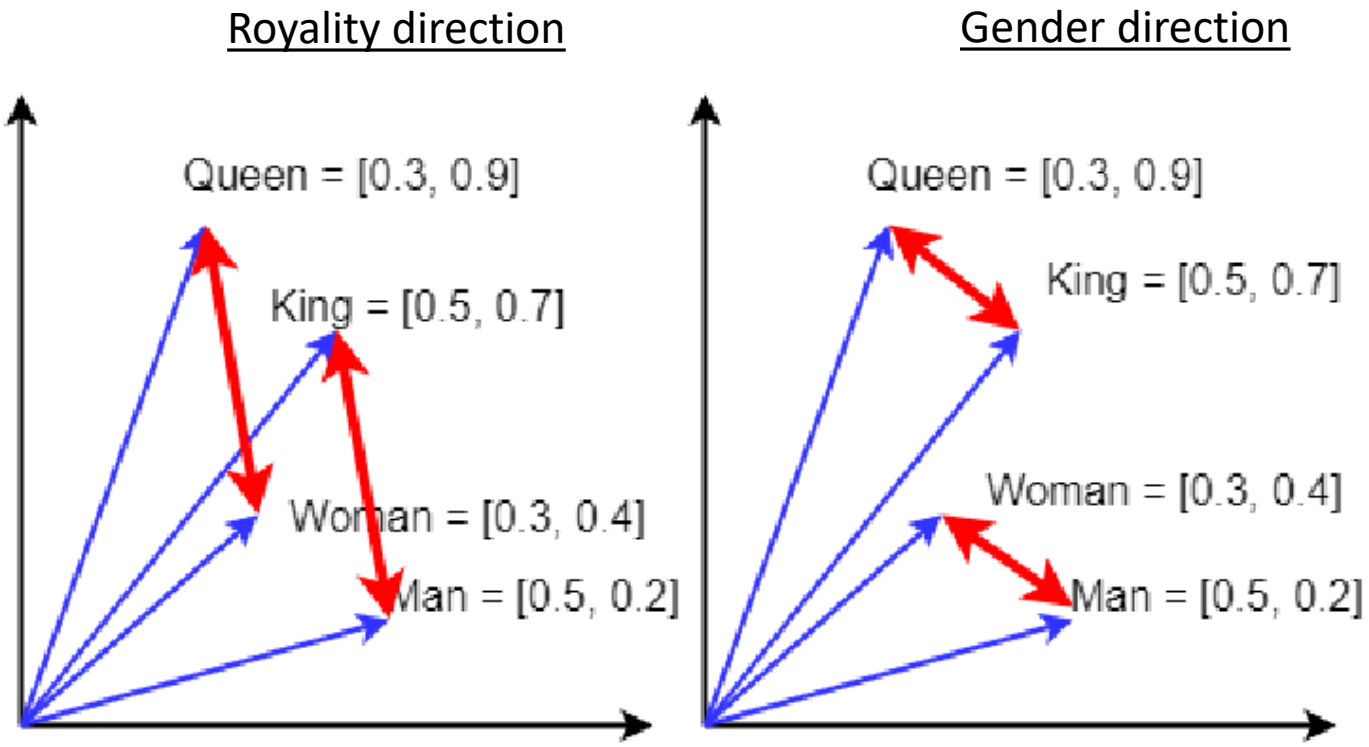


$$s(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} = \frac{\sum_{i=0}^{n-1} x_i y_i}{\sqrt{\sum_{i=0}^{n-1} (x_i)^2} \times \sqrt{\sum_{i=0}^{n-1} (y_i)^2}}$$

Cosine Similarity

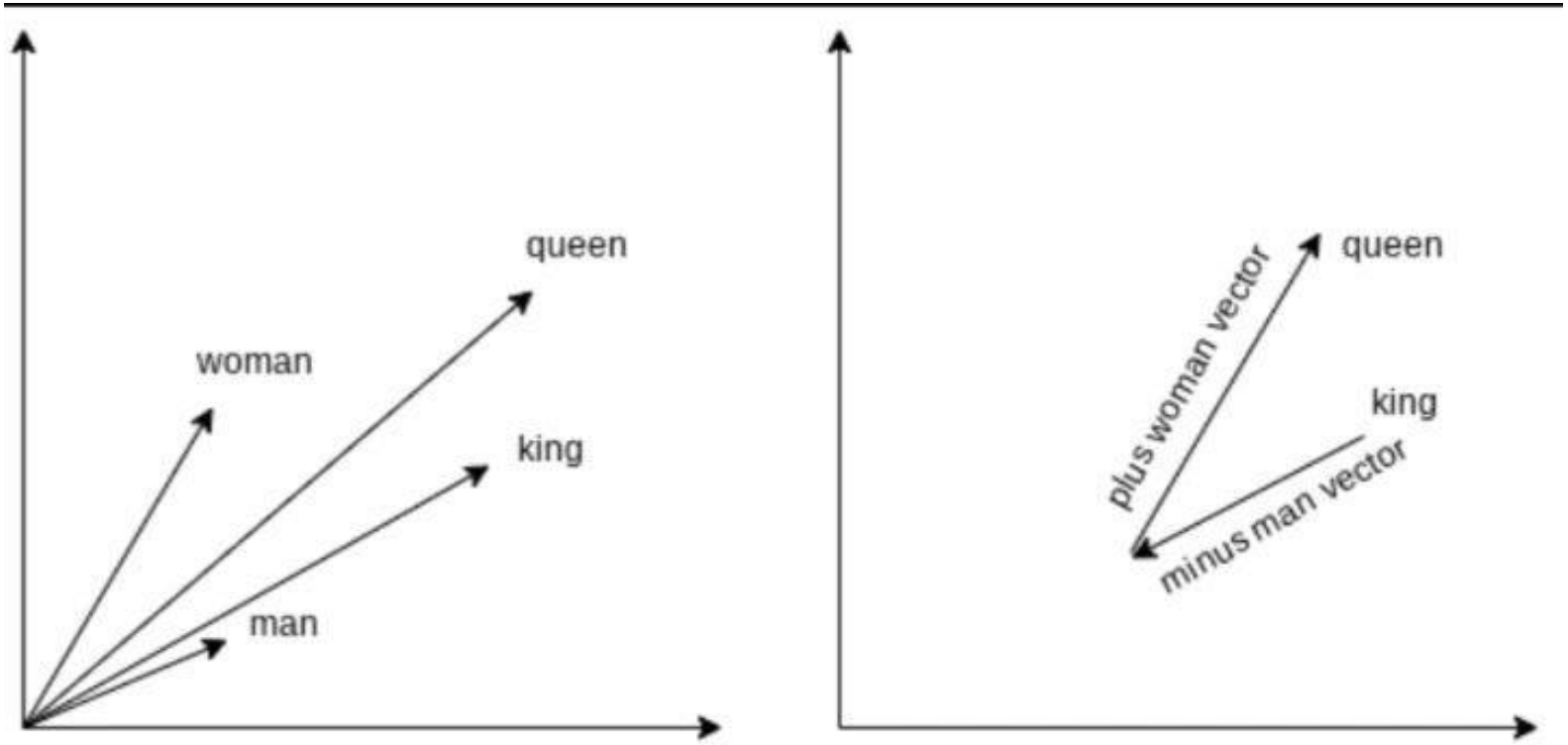
Cosine similarity asks:

- Do these two vectors point in the same direction from the origin?
- Use case: word embeddings → high-D directions encode semantics



Word Embedding Algebra

Flattened high-D space visualized:



Correlation

Pearson's Correlation = correlation similarity

$$r = r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

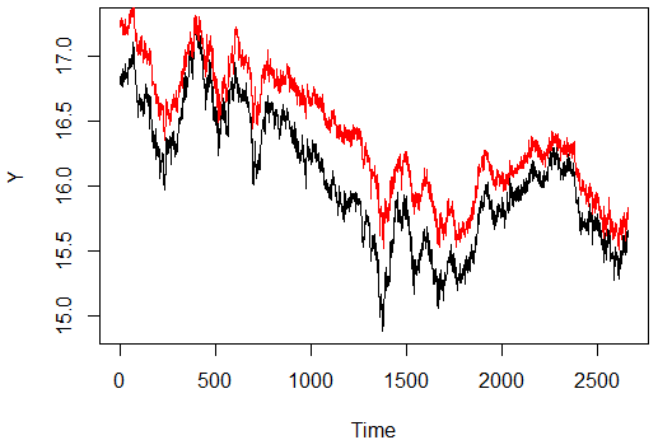
mean across all attribute values for data points x, y
or
mean across all time points of time series x(t) and y(t)

Correlation

Correlation asks:

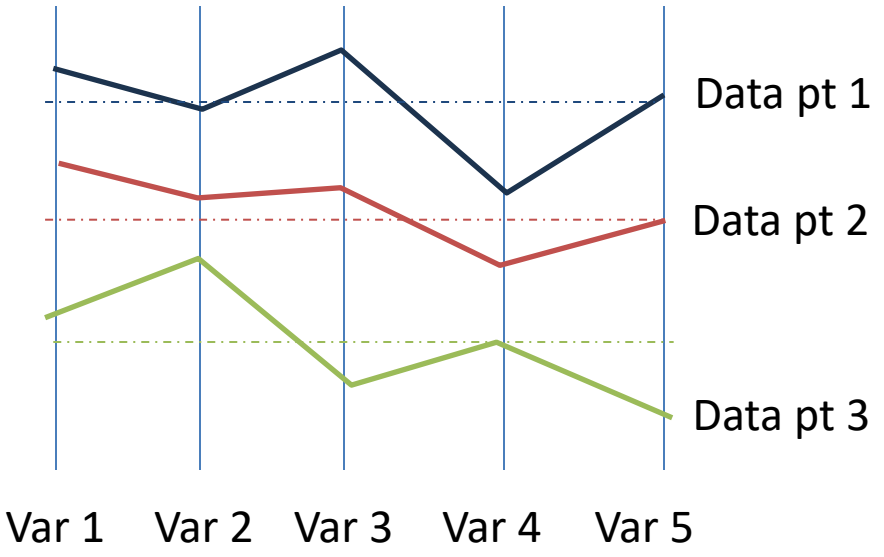
- Do these two data points (or two variables) vary together around their own typical (average) values?

Time series



The two time series are highly correlated

Parallel Coordinates

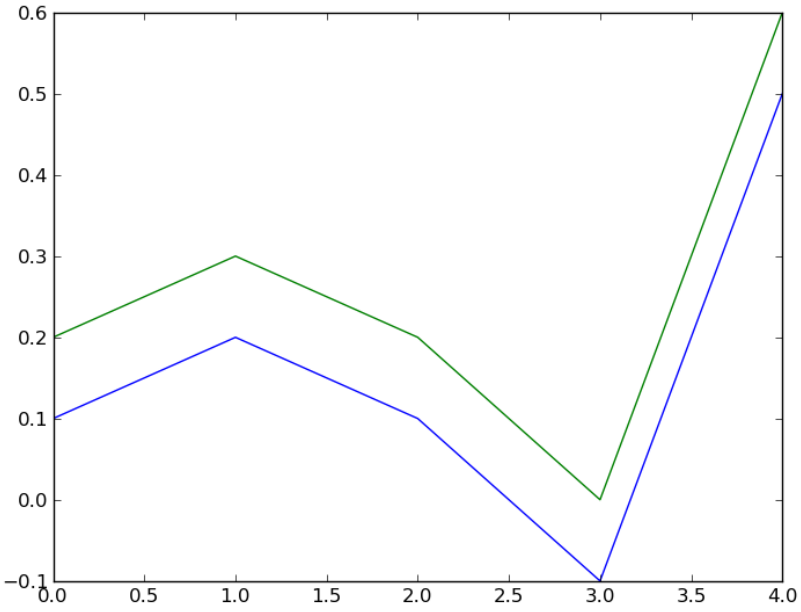


pt 1 and pt 2 are highly positively correlated
pt 3's correlation with pt 1 & pt 2 is moderately negative

Correlation vs. Cosine Distance

Correlation distance is invariant to addition of a constant

- subtracts out by construction
- green and blue curve have correlation of 1
- but cosine similarity is < 1
- correlated vectors just vary in the same way
- cosine similarity is stricter



green = blue + 0.1

Both correlation and cosine similarity are invariant to multiplication with a constant

- invariant to scaling

Distance vs Similarity

Distance

- Measures how far apart things are
- Large value → very different; Small value → very similar
- Has units or scale
- Often satisfies metric properties
- Examples: Euclidean, Manhattan, Jaccard

Similarity

- Measures how alike things are
- Large value → very similar; Small (or negative) value → dissimilar
- Often bounded (e.g., 0 to 1 or −1 to 1)
- Often easier to interpret
- Examples: Cosine similarity, Correlation

Converting one to the other

- Formula depends on how bounded the space is

$$s = 1 - d$$

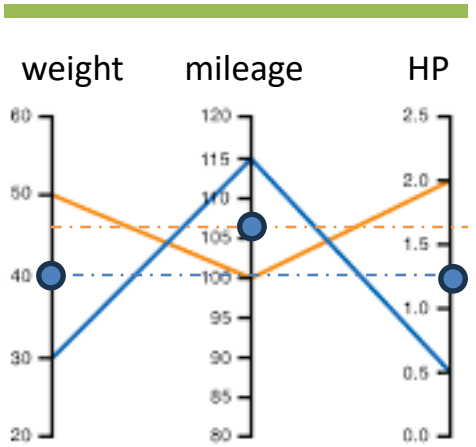
$$s = \frac{1}{1+d}$$

$$s = e^{-d}$$

Variables vs. Data Items

Distances can compare two attributes or two data items

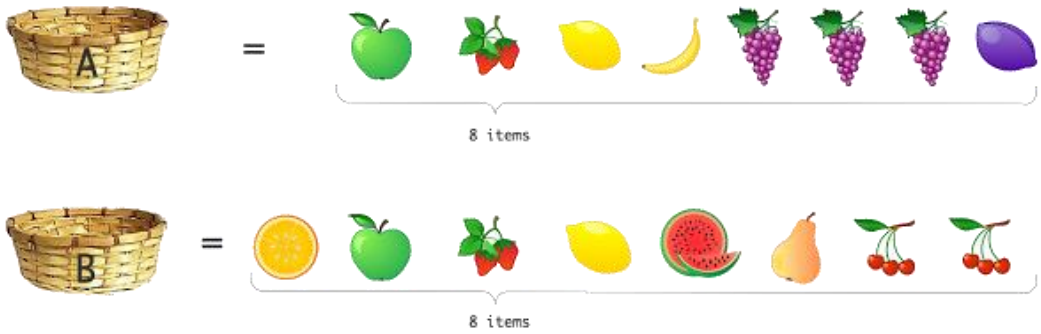
- For example, for correlation distance
- **Attributes:** 1. compute mean (dotted lines) and std dev for all data item values for each of the two attributes. 2. compute correlation; for example: correlation of weight and mileage (which is negative)
- **Data items**: 1. compute mean (filled disks) and std dev of all attribute values for each of the two data items; 2. compute correlation; for example, the two cars; blue and orange one (which is also negative)

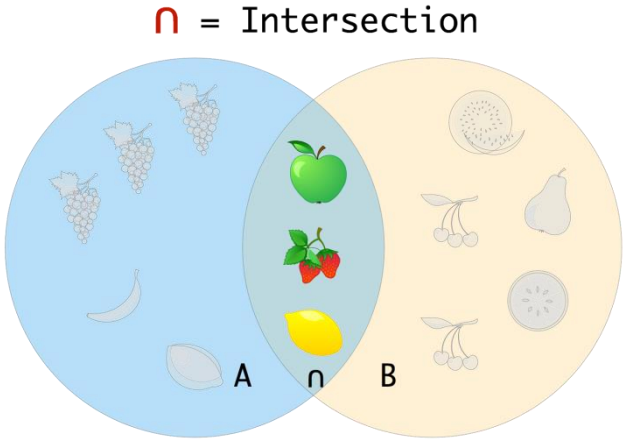


Data			
	Variable A	Variable B	Variable C
Item 1	50	100	2.0
Item 2	30	115	0.5

Jaccard Distance

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$





What's the Jaccard similarity of the two baskets A and B?

Gower's Distance

When you have numerical and categorical data

- Distance metrics typically only applicable to either
- Gower's distance combines these
- Works for numeric features, categorical features, binary features

Definition

$$d(i,j) = \frac{\sum_k w_k \, d_k(i,j)}{\sum_k w_k}$$

- k is the feature index
- i and j are the two data points with mixed features
- w_k is the feature weight, usually set to 1

$$d_k(i,j) = \frac{|x_{ik} - x_{jk}|}{\max(x_k) - \min(x_k)}$$

numerical feature

$$d_k(i,j) = \begin{cases} 0, & \text{if } c_{ik} = c_{jk} \\ 1, & \text{if } c_{ik} \neq c_{jk} \end{cases}$$

categorical feature

$$d_k(i,j) = \begin{cases} 0, & \text{if } x_{ik} = x_{jk} \\ 1, & \text{if } x_{ik} \neq x_{jk} \end{cases}$$

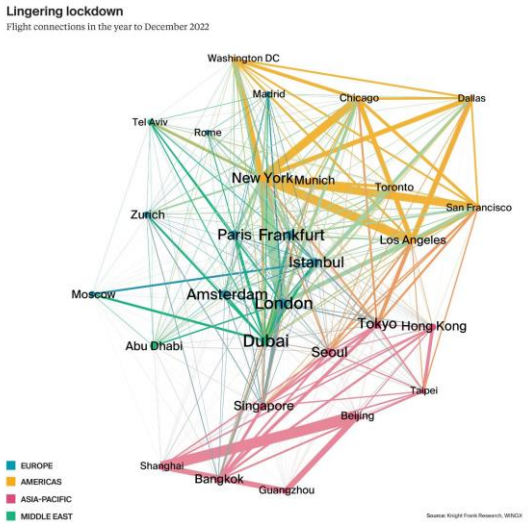
binary feature

Non-Geometric Distances

Transport-Based Distances

- Highway / Road-Network Distance
- Distance = shortest path on the road graph

- Flight Distance (Air Connectivity)
- Distance = number of hops, travel time, or ticket price



- Cost-Based Distance (Prices, Effort, Friction)
- Distance = cost (money, time, stress, carbon)

Example: Airfare Price Distance Map

Figure 1.8 Airlines' view of the United States.
Maps can be scaled to units other than distance. In this case, airline fares are used instead of miles or other linear units.
(Map copyright by the author.)

Non-Geometric Distances: Social Distance

Social Connectedness Index (SCI) from Facebook

- Two places are socially close if people in those places are friends with each other more often than expected

For two regions i and j: (zip codes, cities, counties)

- Count Facebook friendship links between users in i and users in j
- Normalize by population (so big places don't dominate)
- This gives social connectedness
- Distance is then defined as something like:

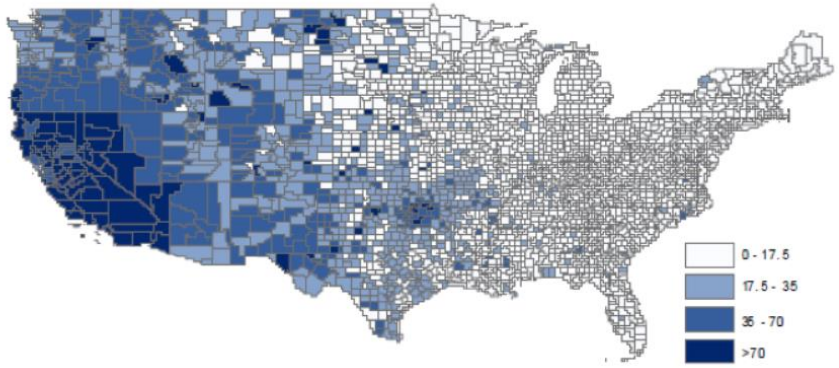
$$\text{social distance} = \frac{1}{\text{connectedness}} \quad \text{or} \quad -\log(\text{connectedness})$$

- More friendship → smaller distance
- Less friendship → larger distance

Social Connectedness Index (SCI): Examples

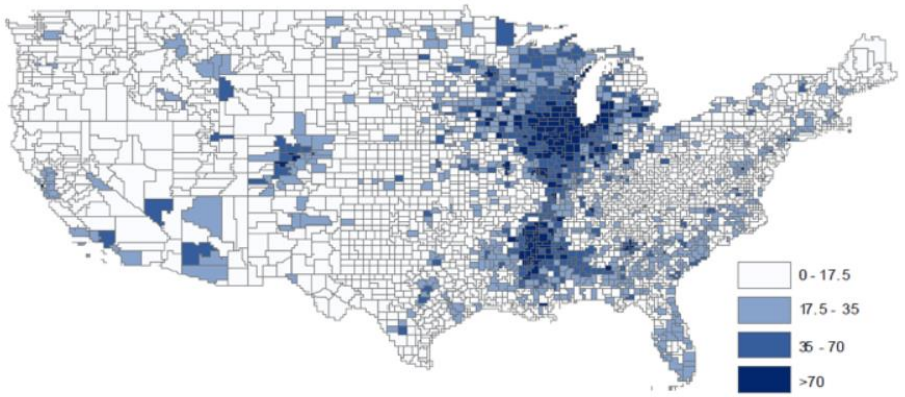
Examples (from Wong et al.)

Panel A: Kern County, CA (Bakersfield)



Kern County, CA has strong social connectedness to Oklahoma and Arkansas (and throughout California). This is likely related to past migration patterns: Kern County was a major destination for migrants fleeing the Dust Bowl in the 1930s, and half of the residents of the San Joaquin Valley in Kern County have ancestors who migrated from there.

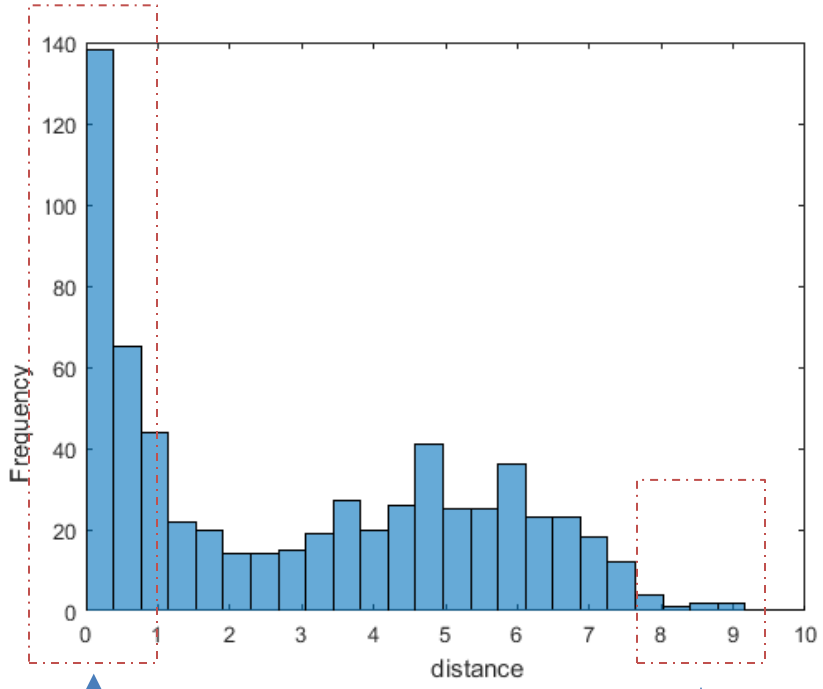
Panel B: Cook County, IL (Chicago)



Cook County, IL, the home of the city of Chicago, has strong connections to the South (in addition to connections throughout the Midwest). This pattern is likely explained by the 'Great Migration' of southern African Americans to Northern and Midwestern cities throughout the earlier and middle parts of the 20th century.

Distance Histograms

Measure distances of all point pairs and store in a histogram



Very similar data points - micro clusters
Data reduction opportunity

Outliers – highlight or remove

Let's Use These Distances

Organizing the Shelf

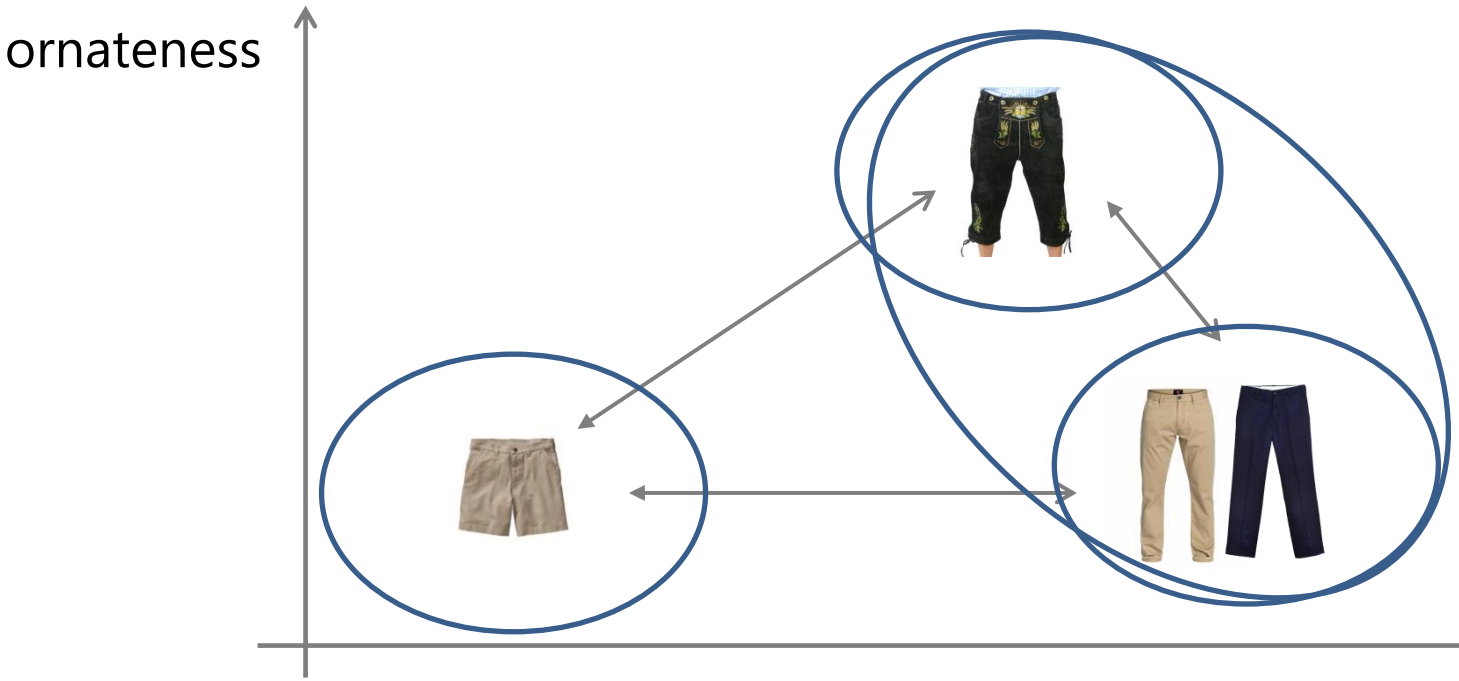
This process is called *clustering*

- and in contrast to a real store, we can make the computer do it for us

What is Clustering?

Note:

- in data mining similarity and distance are the same thing
- so we will use these terms interchangeably



Clustering = grouping of similar items (as determined by the distance function)

What is a Good Cluster?

A cluster is a group of objects that are similar

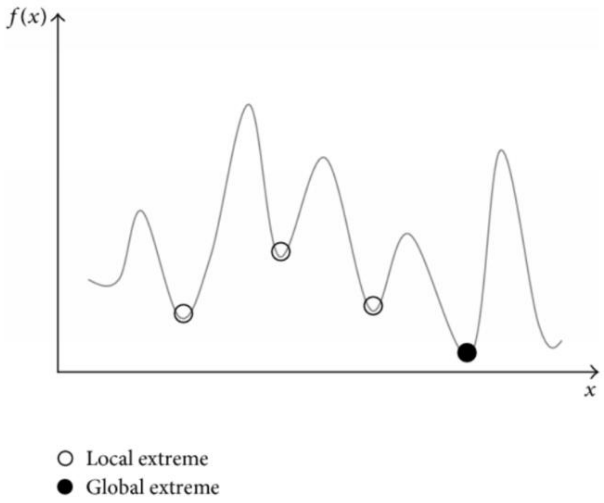
- and dissimilar from other groups of objects at the same time

We need an objective function to capture this mathematically

- the computer will evaluate this function within an algorithm
- one such function is the mean-squared error (MSE)
- and the objective is to minimize the MSE

It's not that easy in practice

- there is only one global minimum
- but often there are many local minima
- need to find the global minimum



Objective – Minimize Squared Error

number of clusters

number of cases

centroid for cluster *j*

case *i*

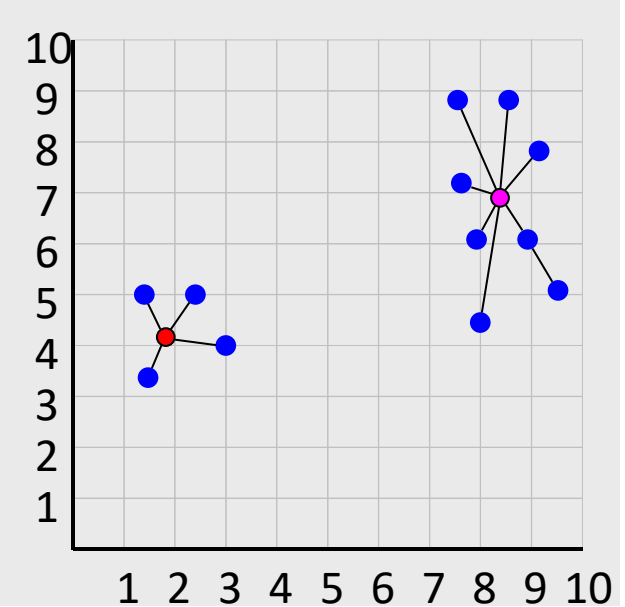
objective function ←

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$

Distance function

In this case

- n=12 (blue points)
- k=2 (red points, the computed centroids)
- distance metric used: Euclidian
- minimization seems to be achieved



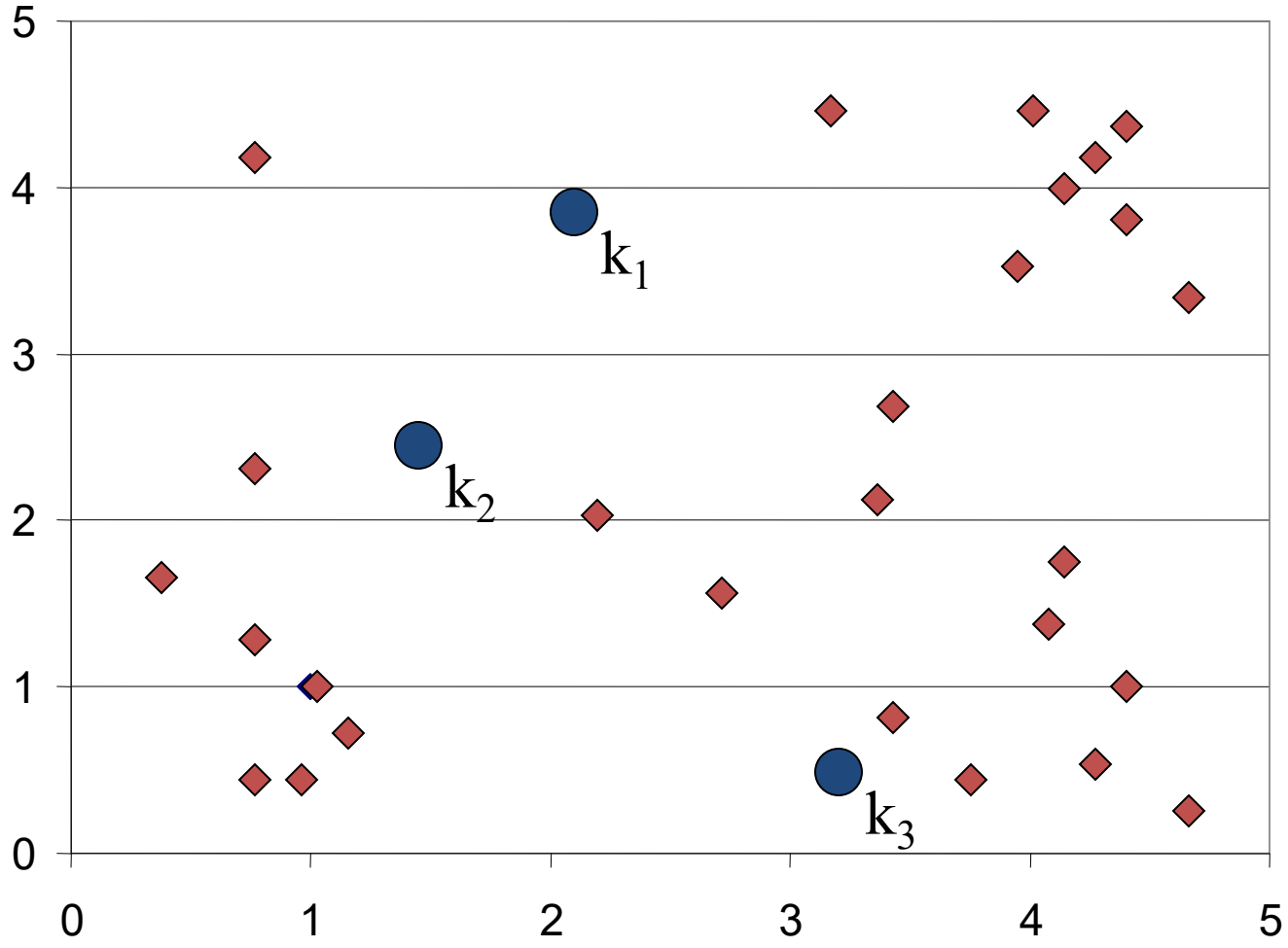
The K-Means Clustering Algorithm

1. Decide on a value for k
2. Initialize the k cluster centers (randomly, if necessary)
3. Decide the class memberships of the N objects by assigning them to the nearest cluster center
4. Re-estimate the k cluster centers, by assuming the memberships found above are correct
5. If none of the N objects changed membership in the last iteration, exit. Otherwise goto 3

The last slide and the next 8 slides contain figures courtesy of Eamonn Keogh, UC Riverside

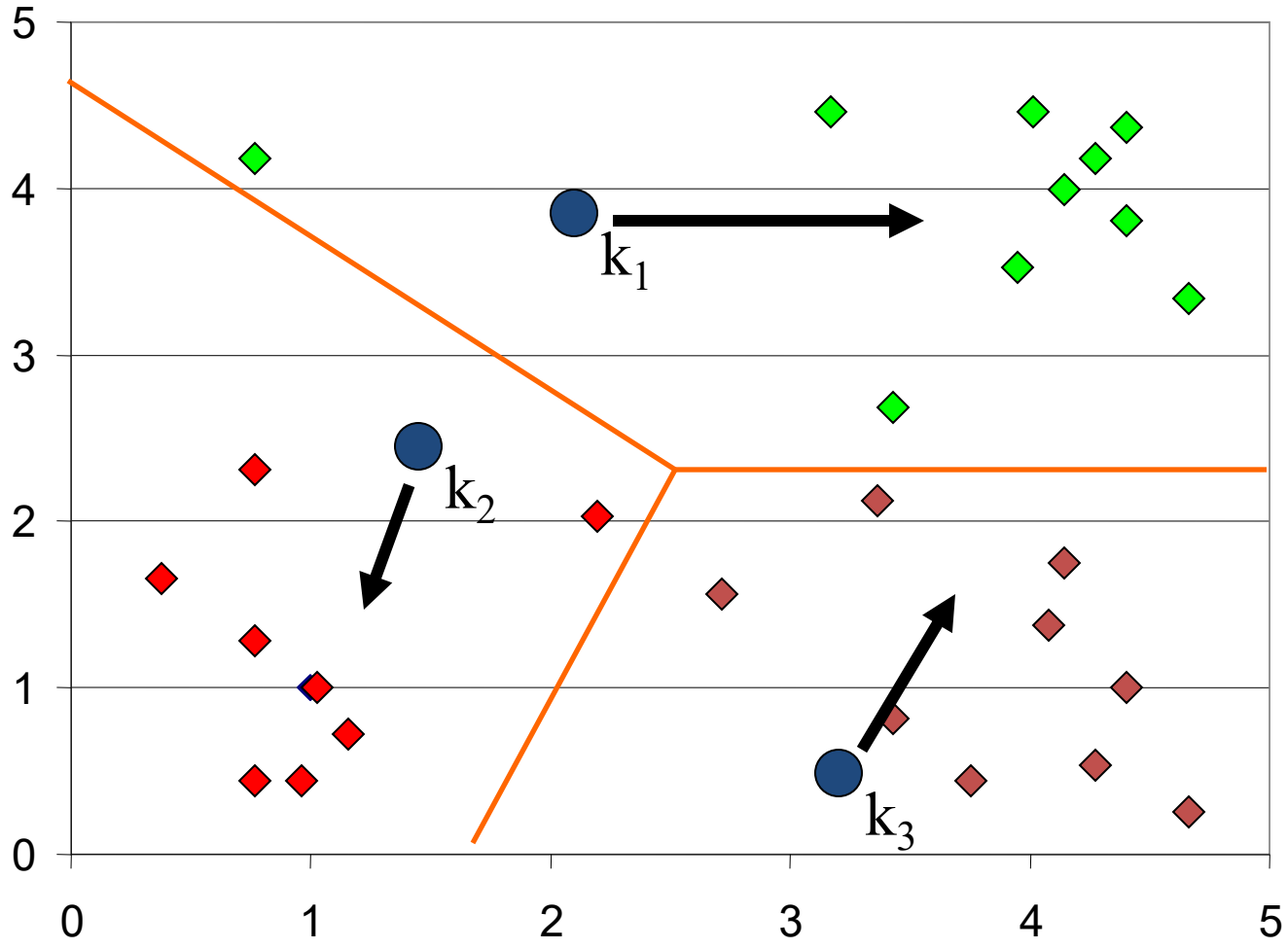
K-means Clustering: Step 1

Algorithm: k-means, Distance Metric: Euclidean Distance



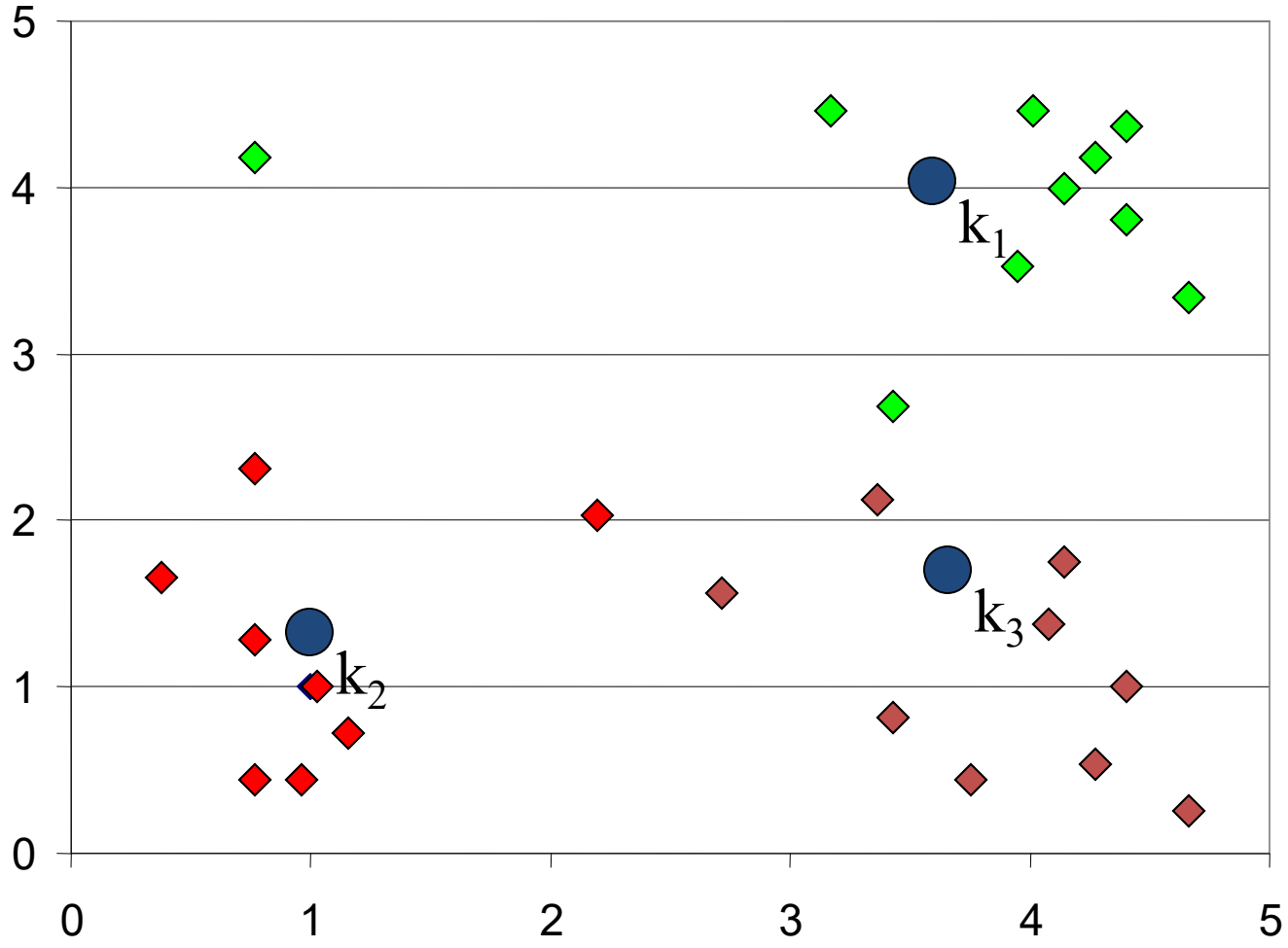
K-means Clustering: Step 2

Algorithm: k-means, Distance Metric: Euclidean Distance



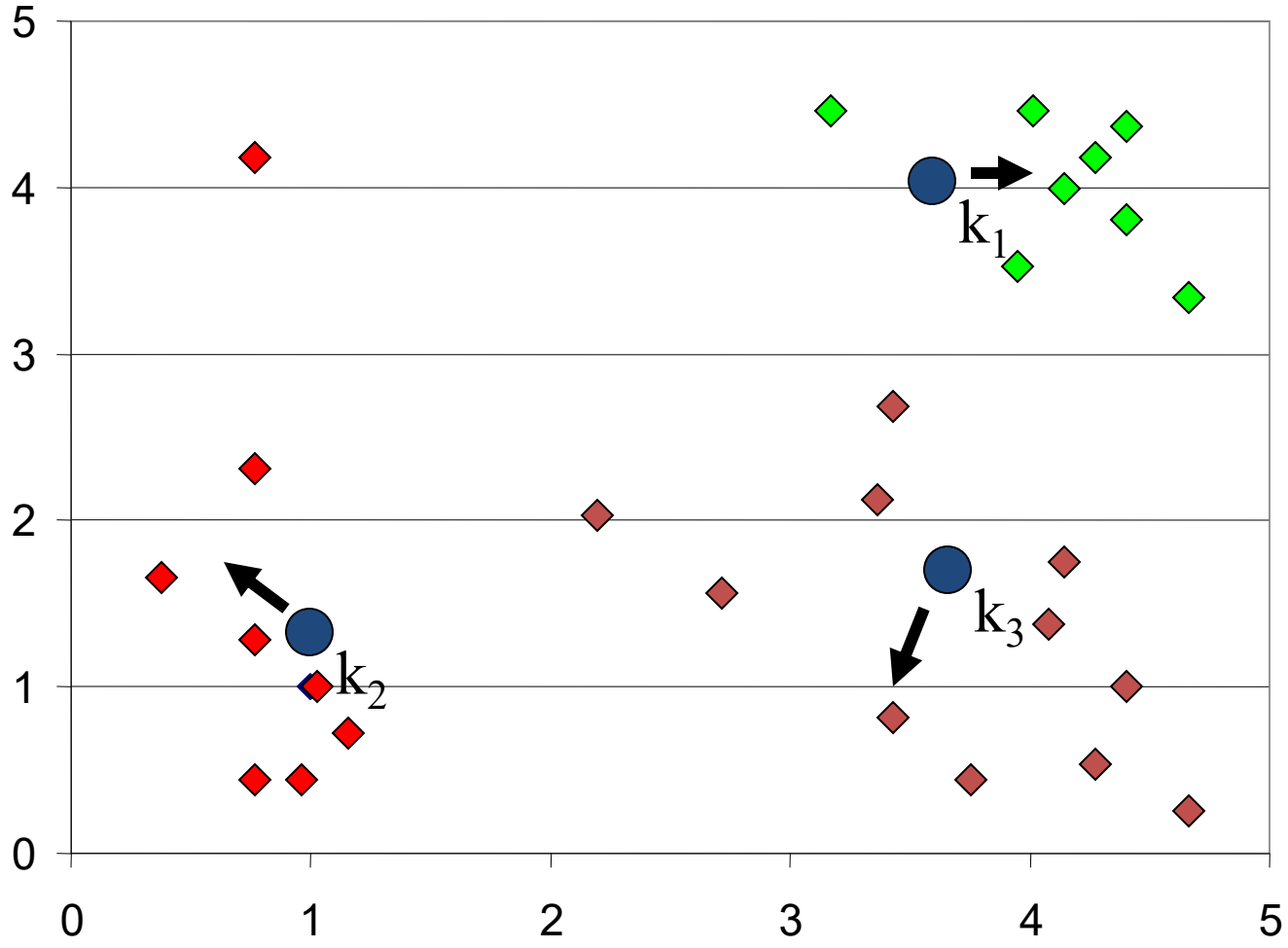
K-means Clustering: Step 3

Algorithm: k-means, Distance Metric: Euclidean Distance



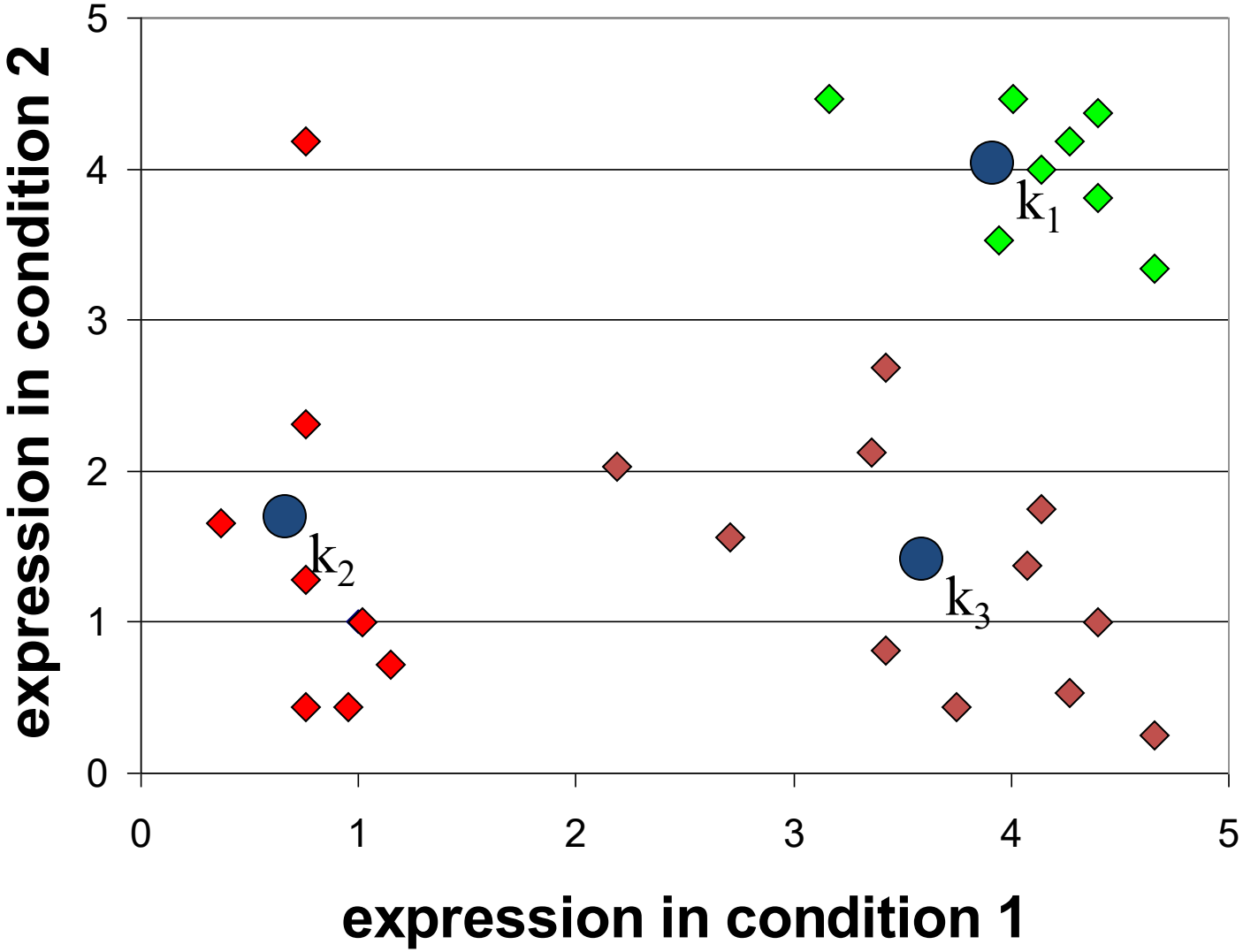
K-means Clustering: Step 4

Algorithm: k-means, Distance Metric: Euclidean Distance



K-means Clustering: Step 5

Algorithm: k-means, Distance Metric: Euclidean Distance



K-Means Algorithm – Comments

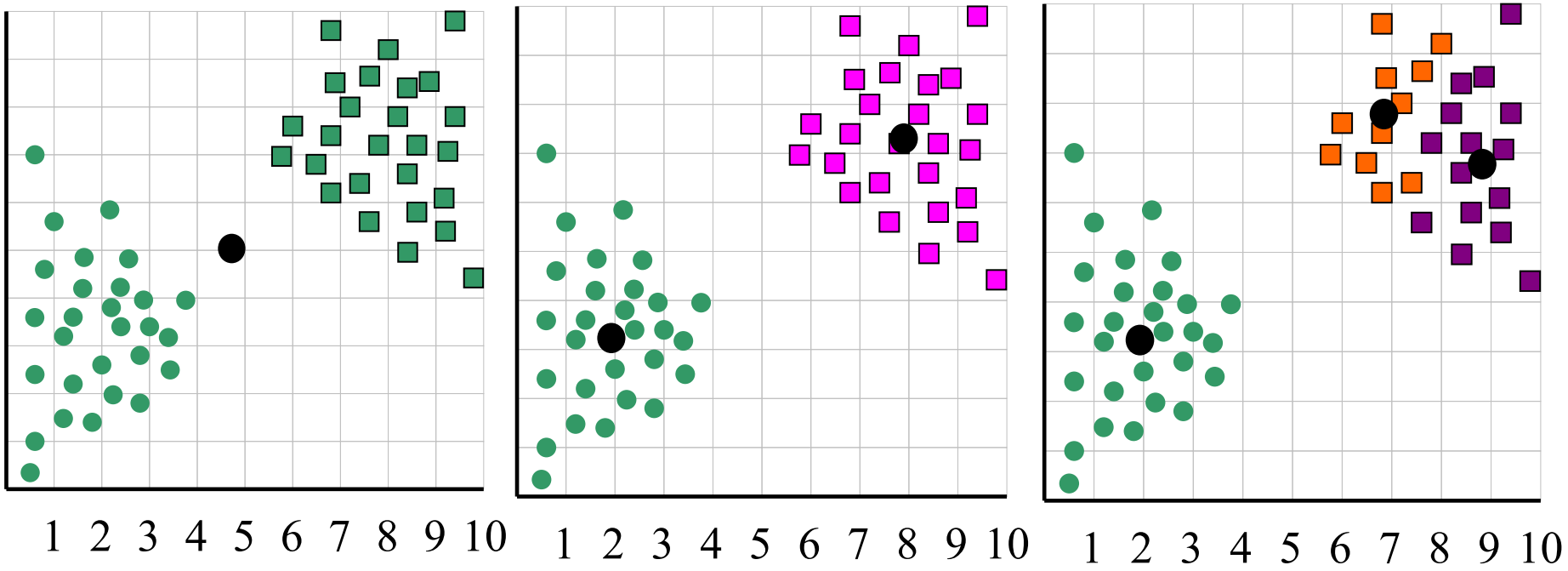
Strengths:

- *relatively efficient*: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, $k, t << n$.
- simple to code

Weaknesses:

- need to specify k in advance which is often unknown
- find the best k by trying many different ones and picking the one with the lowest error
- often terminates at a *local optimum*
- the *global optimum* may be found by trying many times and using the best result

How Can We Find the Best K?



k=1, MSE=873.0

k=2, MSE=173.1

k=3, MSE=133.6

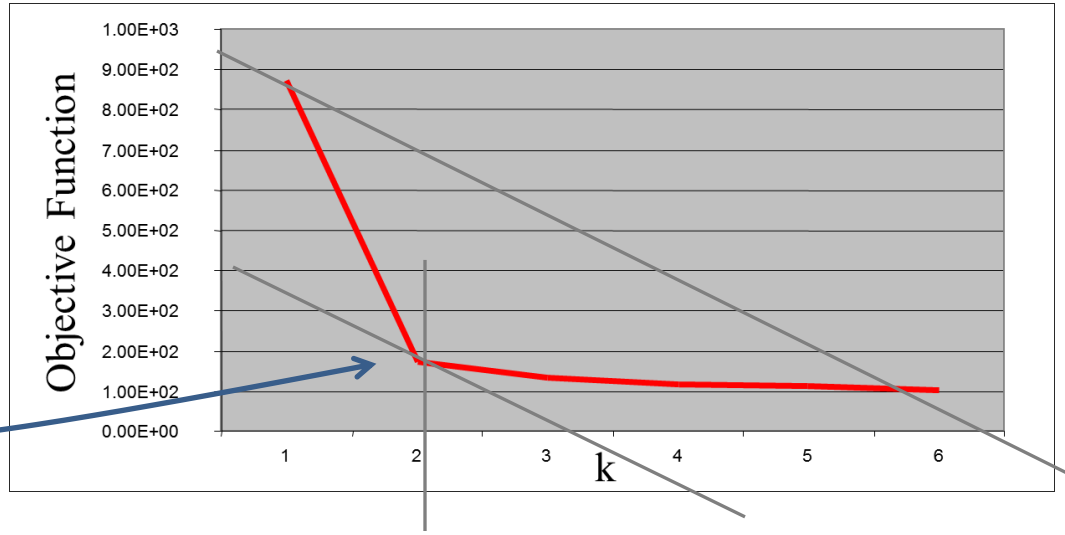
How About K=2?

Is there a principled way we can know when to stop looking?

Yes…

- we can plot the objective function values for k equals 1 to 6…
- then check for a flattening of the curve

tangent at k=2



- the abrupt change at k = 2 is highly suggestive of two clusters
- this technique is known as “knee finding” or “elbow finding”

Drawbacks of K-Means

Need to choose the right k
- Choice of K can determine data interpretation

Assumes spherical, equally sized clusters
- Fails for elongated, curved, or varying-density clusters

Sensitive to outliers
- Means are not robust; outliers pull centroids

Limited to Euclidean-like distances
- Mean is undefined for categorical, social, or arbitrary distances

Hard cluster assignments only
- No uncertainty or overlap at boundaries

Sensitive to feature scaling
- Requires careful normalization

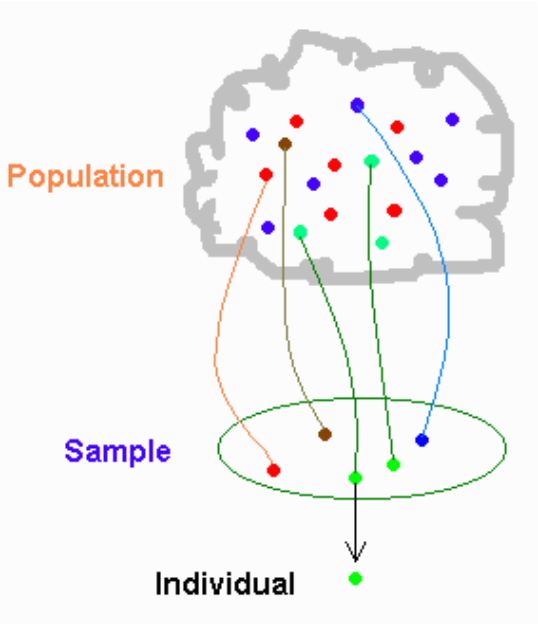
Degrades in high dimensions
- Distance concentration reduces cluster meaning.

Data Reduction

Back to Data reduction

What is sampling?

- pick a representative subset of the data
- discard the remaining data
- pick as many you can afford to keep
- recall: once it's gone, it's gone
- be smart about it



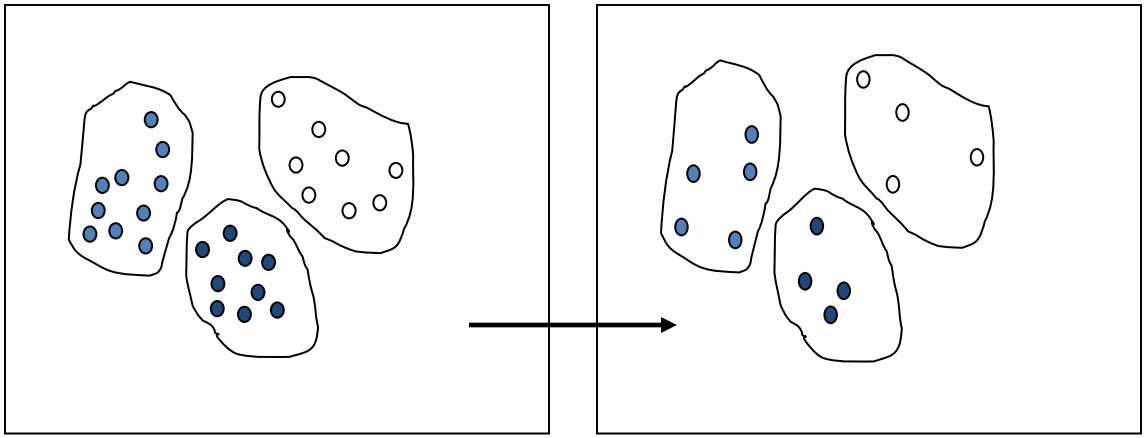
Simplest: random sampling

- pick sample points at random
- will work if the points are distributed uniformly
- this is usually not the case
- outliers will likely be missed
- so the sample will not be representative

Better: Adaptive Sampling

Pick the samples according to some knowledge of the data distribution

- cluster the data (outliers will form clusters as well)
- these clusters are also called *strata* (hence, stratified sampling)
- the size of each cluster represents its percentage in the population
- guides the number of samples – bigger clusters get more samples



sampling rate ~ cluster size

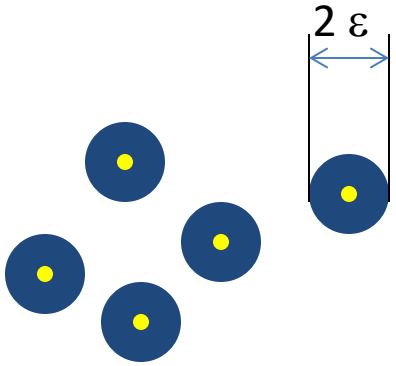
Redundancy Sampling

Eliminate redundant attributes

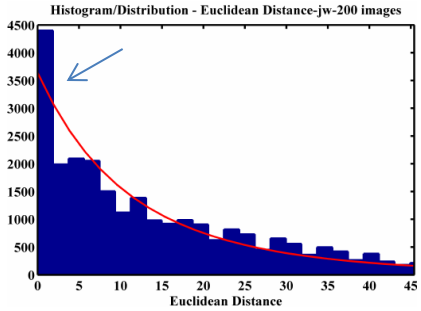
- eliminate highly correlated attributes
  - km vs. miles
  - a + b + c = d → can possibly eliminate 'c' (or 'a' or 'b')

Eliminate redundant data

- cluster the data with small ranges ε
- only keep the cluster centroids
- store size of clusters along to keep importance



- question: how do we find a good ε?
- answer: compute histogram of distances
- choose a reasonable threshold from the left



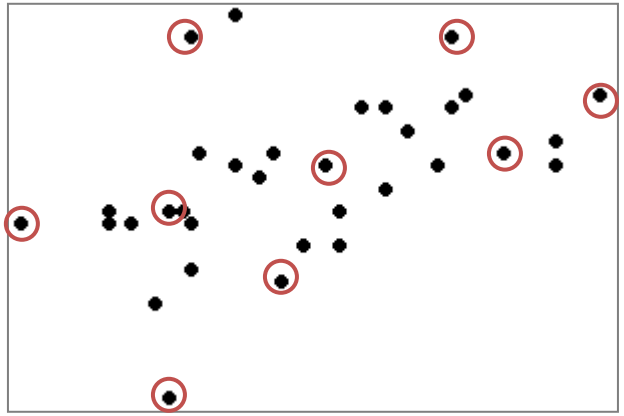
Sampling of Well-Scattered Points

Used in the CURE high-dimensional clustering algorithm

- S. Guha, R. Rajeev, and K. Shim. "CURE: an efficient clustering algorithm for large databases." *ACM SIGMOD*, 27(2): 73-84, 1998

Algorithm

- initialize the point set *S* to empty
- pick the point farthest from the mean as the first point for *S*
- then iteratively pick points that are furthest from the points in S collected so far



Complexity is $O(m \cdot n^2)$

- *n* is the total number of points, *m* is the number of desired points
- can find arbitrarily shaped clusters and preserve outliers, too
- need some good data structures to run efficiently: kd-tree, heap