CSE 564
Visualization & Visual Analytics

Cluster Analysis

Klaus Mueller

Computer Science Department
Stony Brook University and SUNY Korea

Lecture	Topic	Projects
1	Intro, schedule, and logistics	
2	Applications of visual analytics, basic tasks, data types	
3	Introduction to D3, basic vis techniques for non-spatial data	Project #1 out
4	Data assimilation and preparation	
5	Bias in visualization	
6	Data reduction and dimension reduction	
7	Visual perception and cognition	Project #1 due
8	Visual design and aesthetics	Project #2 out
9	Python/Flask hands-on	
10	Data mining techniques: clusters, text, patterns, classifiers	
11	Computer graphics and volume rendering	
12	Techniques to visualize spatial (3D) data	Project #2 due
13	Scientific and medical visualization	Project #3 out
14	Scientific and medical visualization	
15	Midterm #1	
16	High-dimensional data, dimensionality reduction	Project #3 due
17	Big data: data reduction, summarization	
18	Correlation and causal modeling	
19	Principles of interaction	
20	Visual analytics and the visual sense making process	Final project proposal due
21	Evaluation and user studies	
22	Visualization of time-varying and time-series data	
23	Visualization of streaming data	
24	Visualization of graph data	Final Project preliminary report due
25	Visualization of text data	
26	Midterm #2	
27	Data journalism	
	Final project presentations	Final Project slides and final report due

When to Use Cluster Analysis

Data summarization

- data reduction
- cluster centers, shapes, and statistics

Customer segmentation

- collaborative filtering

Social network analysis

- find similar groups of friends (communities)

Precursor to other analyses

- use as a preprocessing step for classification and outlier detection
- use it for sampling and data reduction

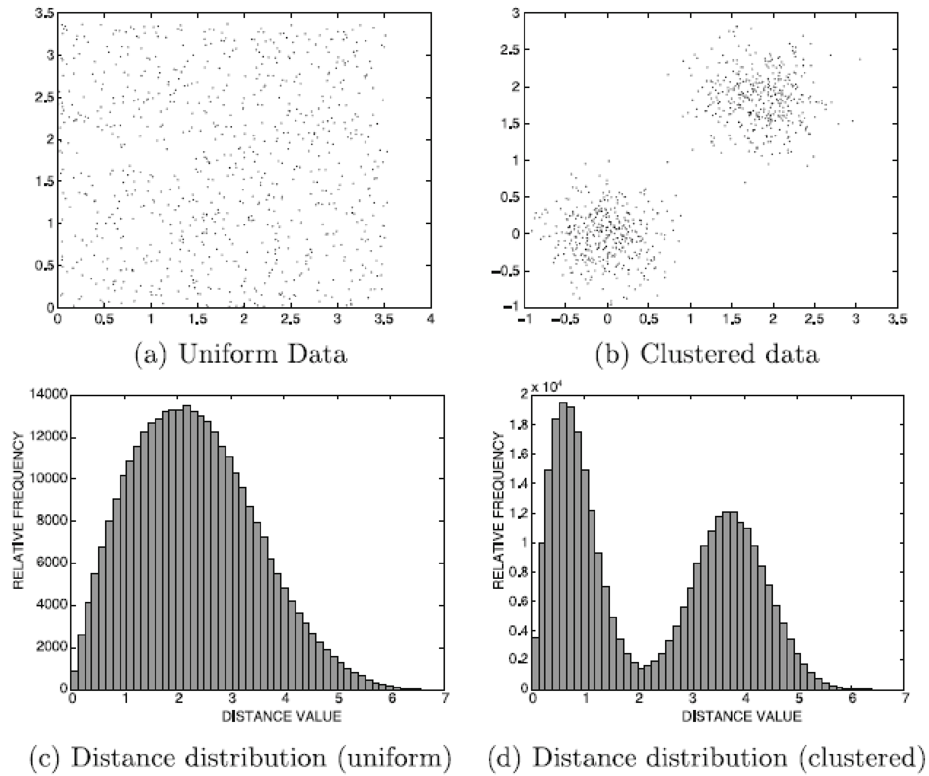
Attribute Selection

With 1,000s of attributes (dimensions) which ones are relevant and which one are not?

avoid

keep

histogram of pairwise distances in N-D space



(a) Uniform Data

(b) Clustered data

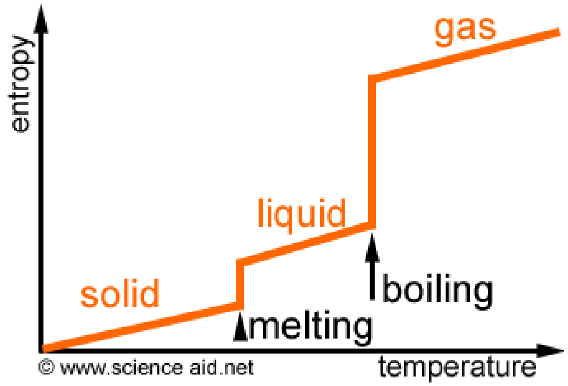
(c) Distance distribution (uniform)

(d) Distance distribution (clustered)

Attribute Selection

How to measure attribute "worthiness"

- use entropy



Entropy

- originates in thermodynamics
- measures lack of order or predictability

Entropy in statistics and information theory

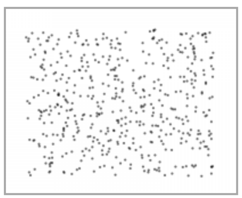
- has a value of 1 for uniform distributions (not predictable)
- knowing the value has a lot of information (high surprise)
- has a value of 0 for a constant signal (fully predicable)
- knowing the value has zero information (low surprise)

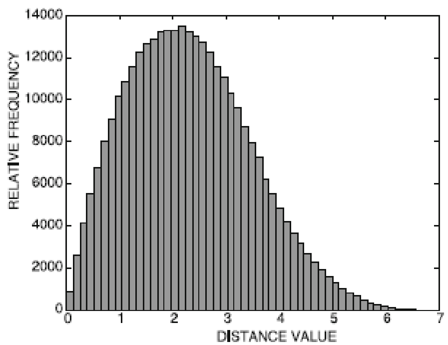
Entropy

Assume m bins, 1 ≤ i ≤ m: $$E = -\sum_{i=1}^{m}[p_i \log(p_i) + (1-p_i)\log(1-p_i)].$$

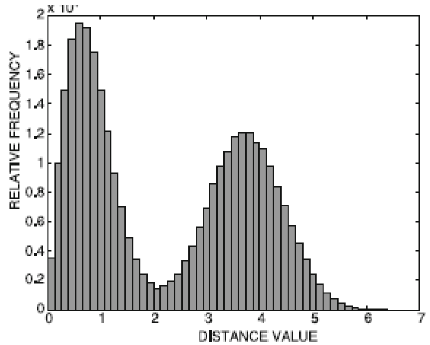
E high

E low

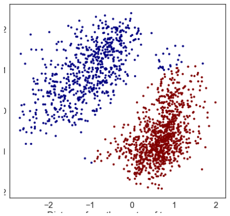


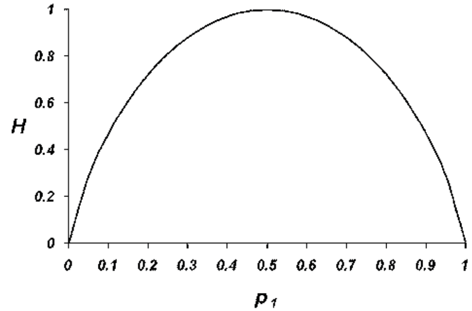


(c) Distance distribution (uniform)



(d) Distance distribution (clustered)



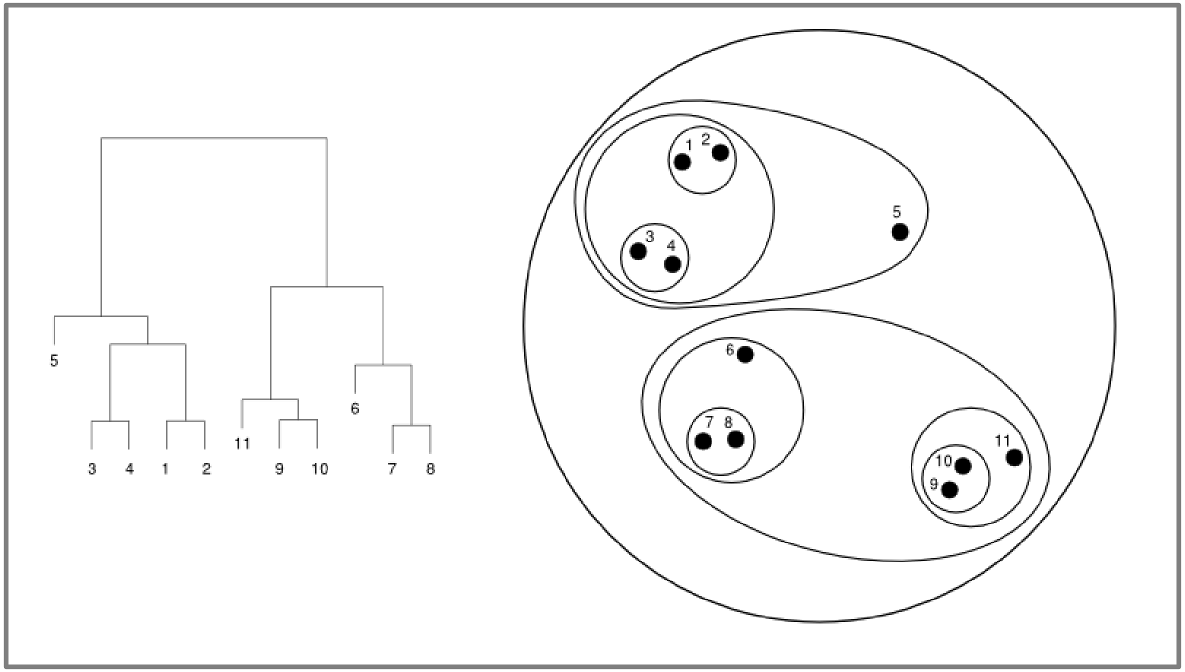


Binary source
(e.g. coin)

Algorithm:

- start with all attributes and compute distance entropy
- greedily eliminate attributes that reduce the entropy the most
- stop when entropy no longer reduces or even increases

Hierarchical Clustering

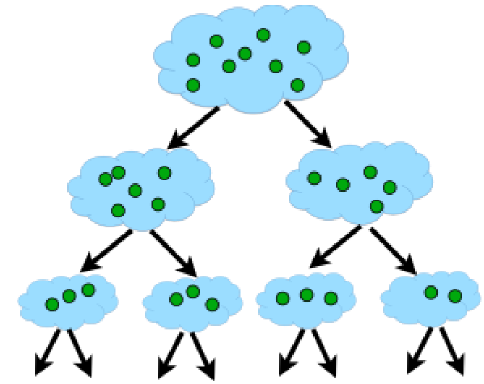


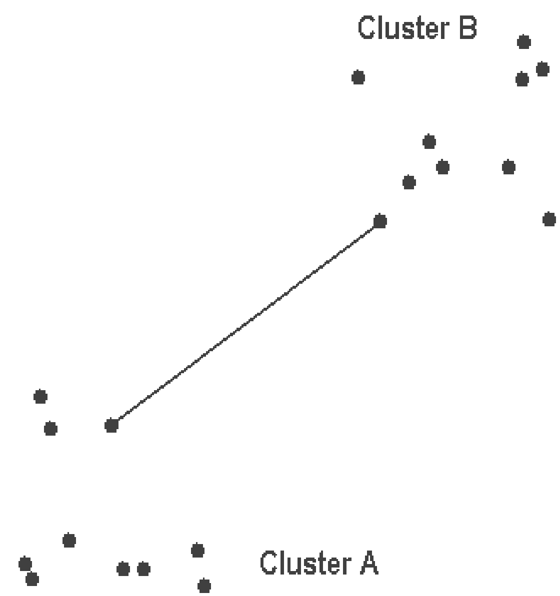
Two options for building the dendrogram on the left

- top down (divisive)
- bottom up (agglomerative)

Bottom-Up Agglomerative Methods

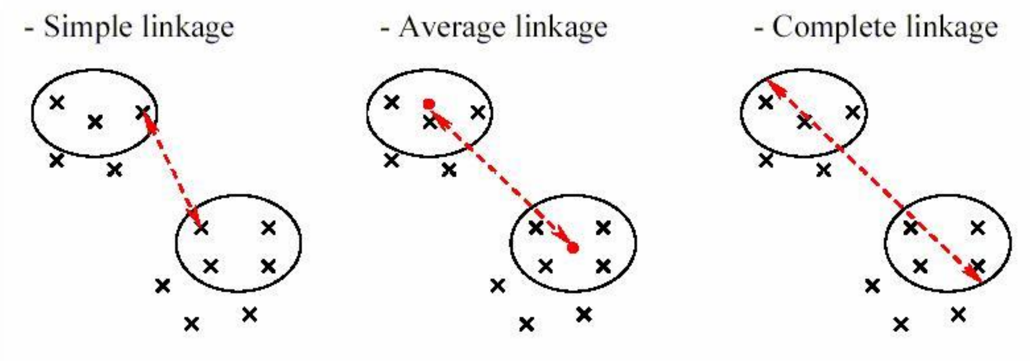
Algorithm *AgglomerativeMerge*(Data: $\mathcal{D}$)
begin
  Initialize $n \times n$ distance matrix M using $\mathcal{D}$;
  repeat
    Pick closest pair of clusters i and j using M;
    Merge clusters i and j;
    Delete rows/columns i and j from M and create
      a new row and column for newly merged cluster;
    Update the entries of new row and column of M;
  until termination criterion;
  return current merged cluster set;
end





How to merge?

Merge Criteria



Single (best-case) linkage

- distance = minimum distance between all $m_i \cdot m_j$ pairs of objects
- joins the closest pair

Complete (worst-case) linkage

- distance = maximum distance between all $m_i \cdot m_j$ pairs of objects
- joins the pair furthest apart

Group-average linkage

- distance = average distance between all object pairs in the groups

Other methods:

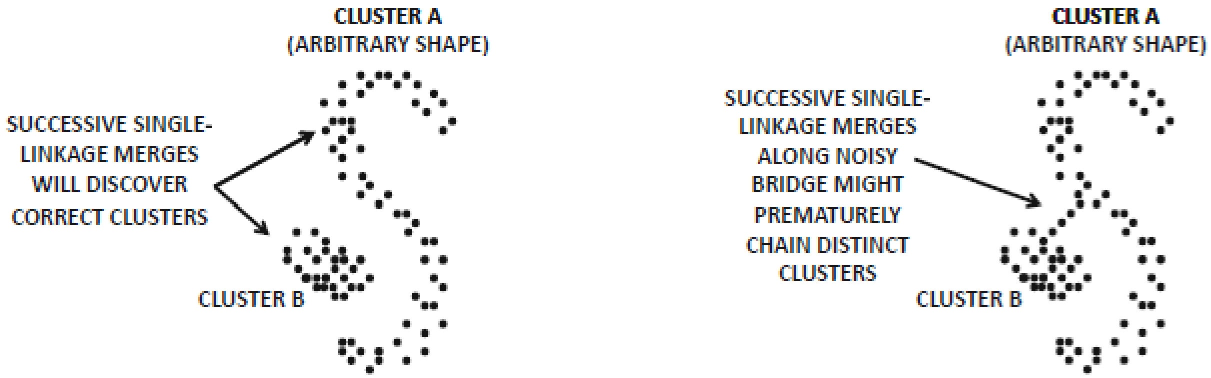
- closest centroid, variance-minimization, Ward's method

Comparison

Centroid-based methods tend to merge large clusters

Single linkage method can merge chains of closely related points to discover clusters of arbitrary shape

- but can also (inappropriately) merge two unrelated clusters, when the chaining is caused by noisy points between two clusters



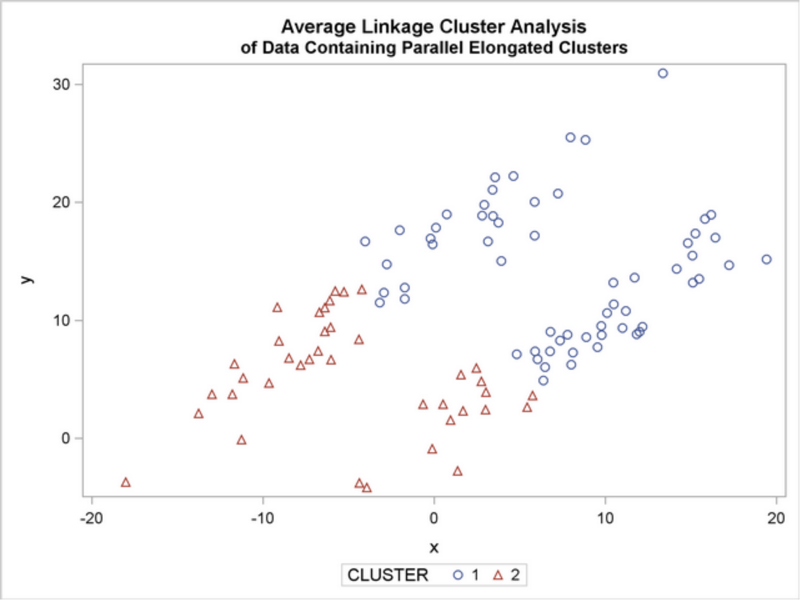
(a) Good case with no noise

(b) Bad case with noise

Comparison

Complete (worst-case) linkage method tends to create spherical clusters with similar diameter

- will break up the larger odd-shaped clusters into smaller spheres
- also gives too much importance to data points at the noisy fringes of a cluster

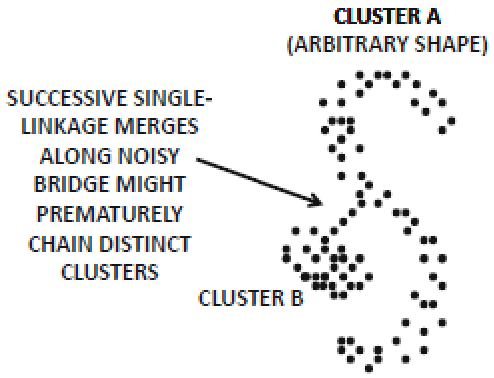


Comparison

The group average, variance, and Ward's methods are more robust to noise due to the use of multiple linkages in the distance computation

Hierarchical methods are sensitive to a small number of mistakes made during the merging process

- can be due to noise
- no way to undo these mistakes



(b) Bad case with noise

DBSCAN

Highly-cited density-based hierarchical clustering algorithm (Ester et al. 1996)

- clusters are defined as density-connected sets
- epsilon-distance neighbor criterion (Eps)

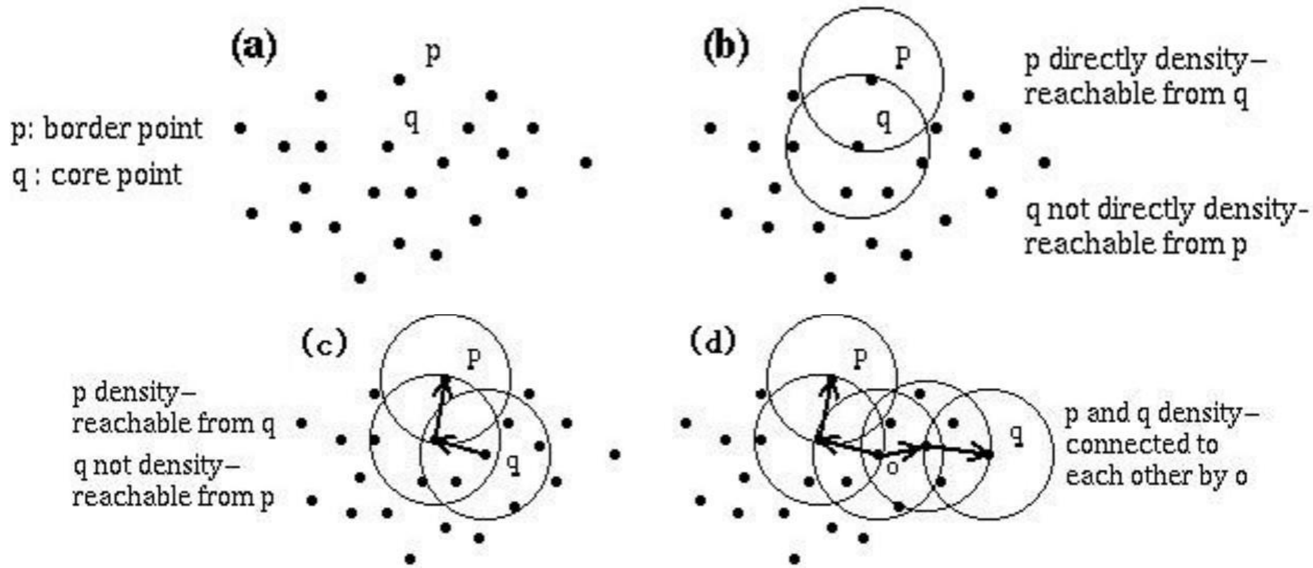
  $N_{Eps}(p) = \{q \in D \mid dist(p,q) \leq Eps\}$
- minimum point cluster membership and core point (MinPts)

  $|N_{Eps}(q)| \geq MinPts$
- notions of density-connected & density-reachable (direct, indirect)
- a point p is directly density-reachable from a point q wrt. Eps, MinPts if

  $p \in N_{Eps}(q)$ and

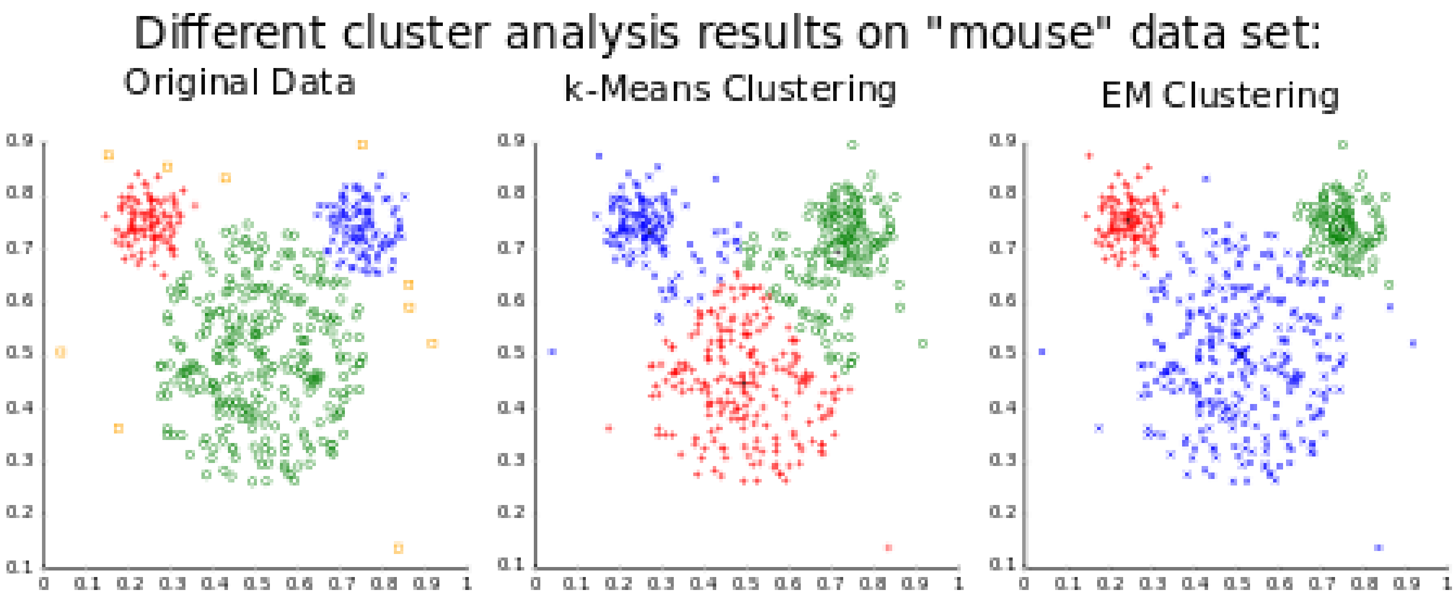
  $|N_{Eps}(q)| \geq MinPts$ (core point condition)

DBSCAN


(a)
p
q
p: border point
q : core point
(b)
p
q
p directly density–
reachable from q
q not directly density-
reachable from p
(c)
p
q
p density–
reachable from q
q not density–
reachable from p
(d)
p
q
o
p and q density–
connected to
each other by o

Probabilistic Extension to K-Means

First a comparison:



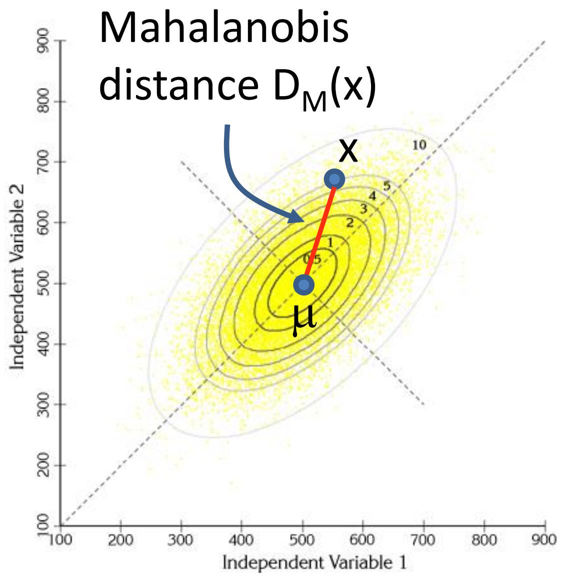
Mahalanobis Distance

The distance between a point X and a distribution D

- measures how many standard deviations X is away from the mean μ of D
- S is the covariance matrix of the distribution D
- the Mahanalobis distance D_M of a point x to a cluster center μ is

$$D_M(x) = \sqrt{(x-\mu)^T S^{-1} (x-\mu)}.$$

- x and μ are N-dimensional vectors
- S is the N×N covariance matrix
- the outcome $D_M(x)$ is a single-dimensional number



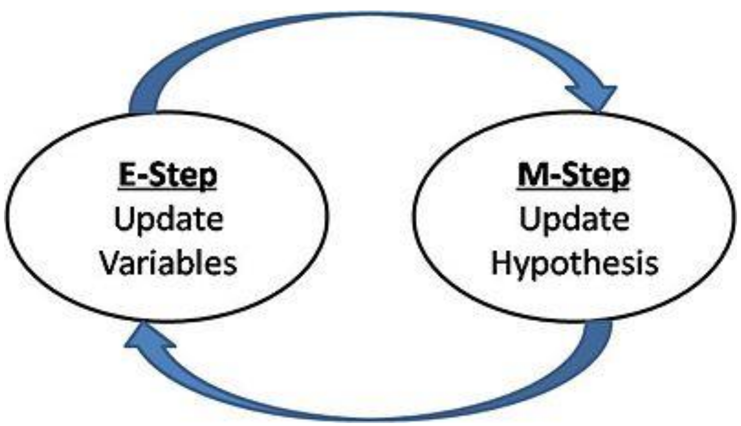
Probabilistic Clustering

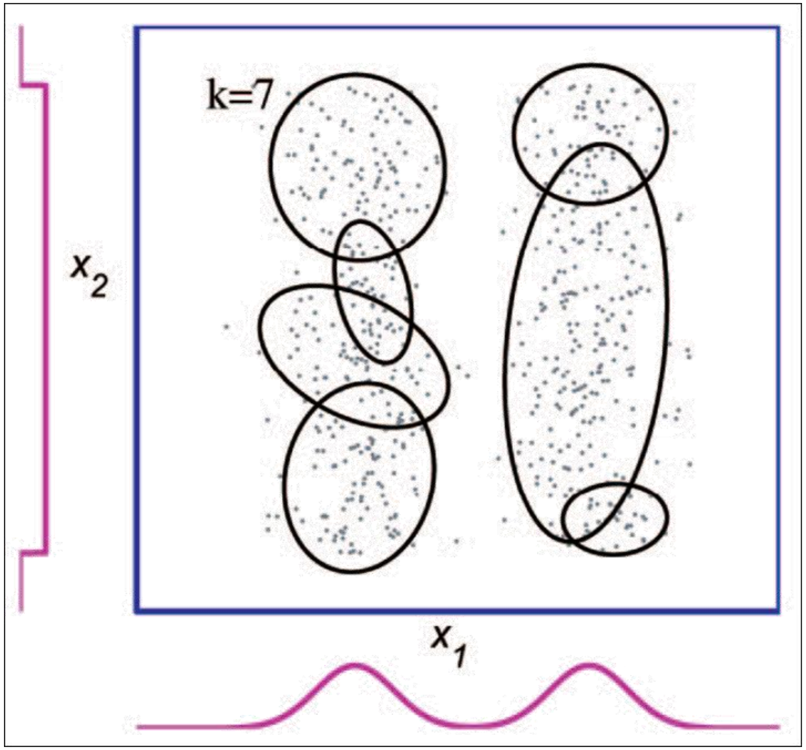
Is a better match for point distributions

- overlapping clusters are now possible
- better match with real world?
- Gaussian mixtures

Need a probabilistic algorithm

- Expectation-Maximization





EM Algorithm (Mixture Model)

probability that data point d_i is in class c_j
(= Mahanalobis distance of d_i to c_j)

- Initialize K cluster centers
- Iterate between two steps
  - **E**xpectation step: assign points to *m* clusters/classes

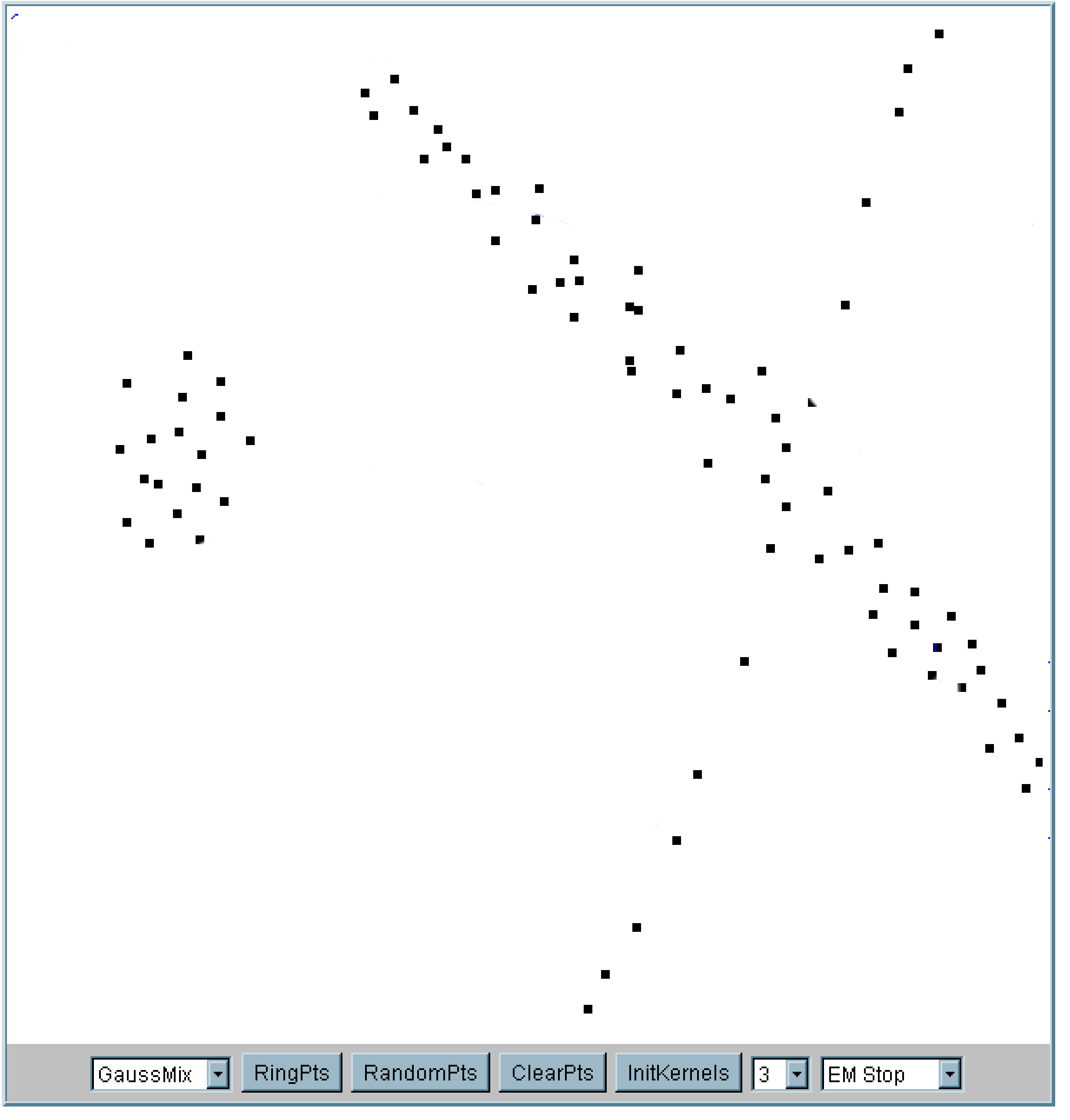
$$P(d_i \in c_k) = w_k \Pr(d_i \mid c_k) \Big/ \sum_j w_j \Pr(d_i \mid c_j)$$

$$w_k = \frac{\sum_i \Pr(d_i \in c_k)}{N}$$ = probability of class c_k

  - **M**aximation step: estimate model parameters

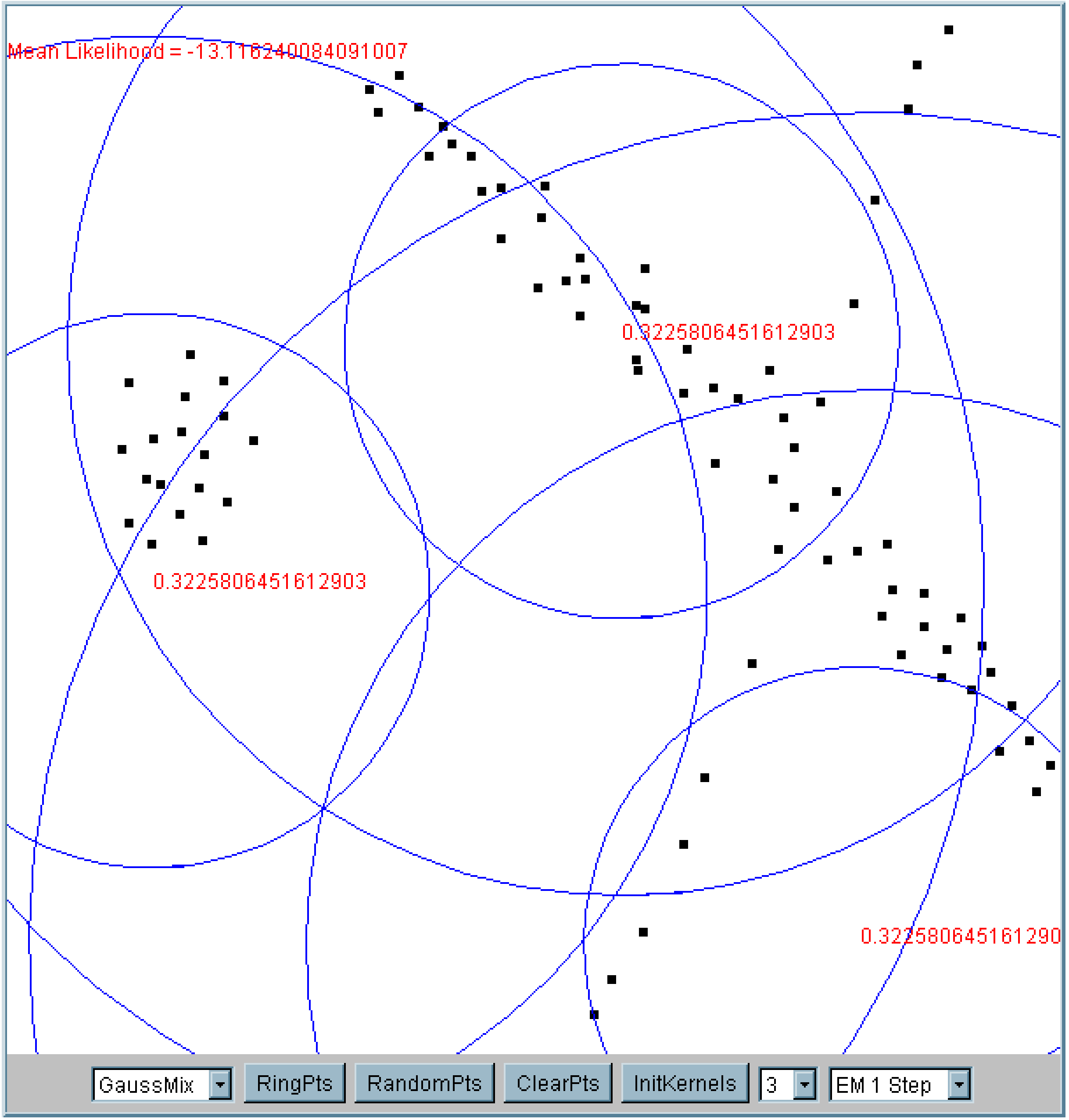
do similar also for covariance matrix S

$$\mu_k = \frac{1}{m} \sum_{i=1}^{m} \frac{d_i P(d_i \in c_k)}{\sum_k P(d_i \in c_j)}$$

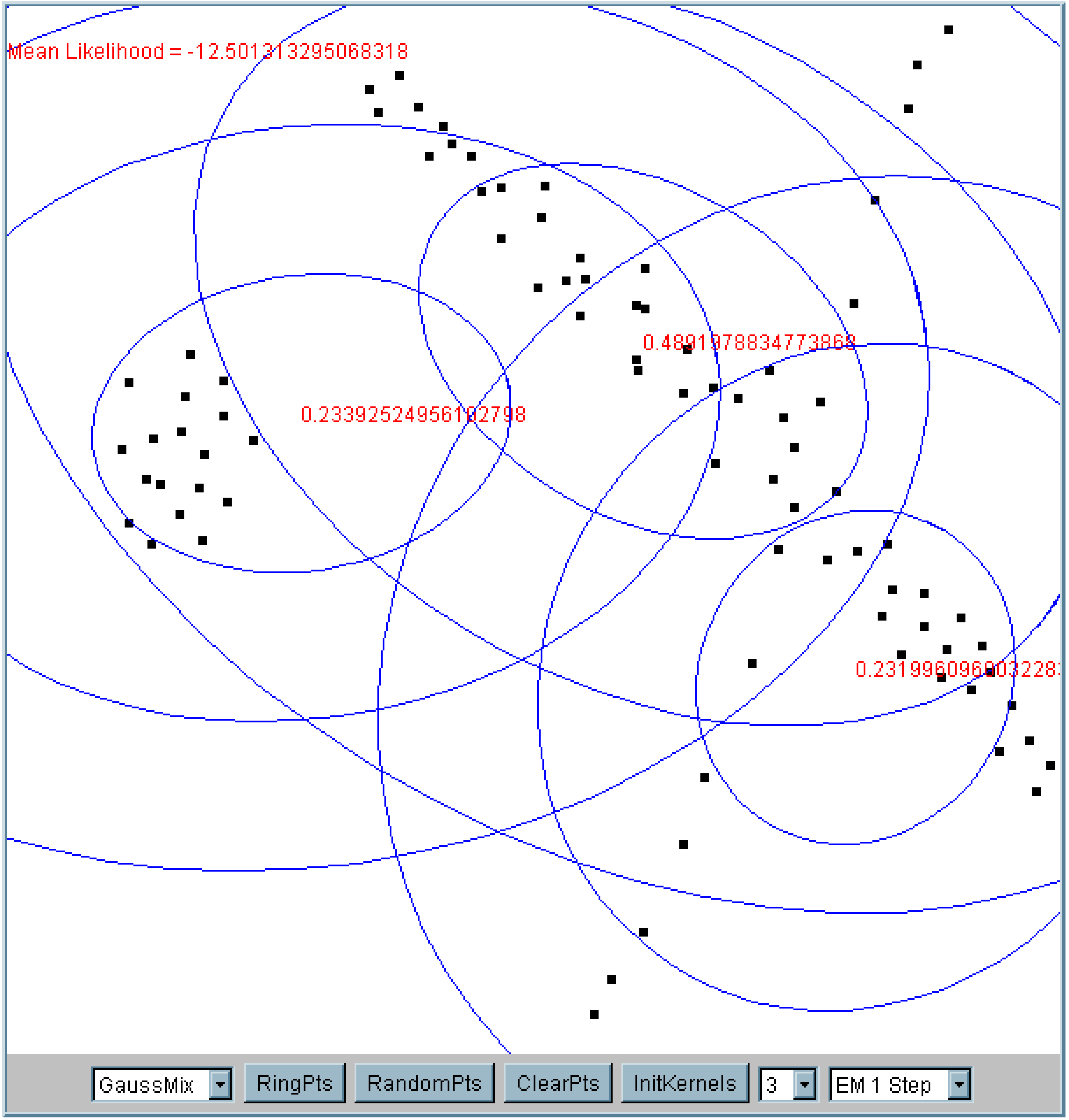

GaussMix
RingPts
RandomPts
ClearPts
InitKernels
3
EM Stop

Iteration 1

The cluster means are randomly assigned

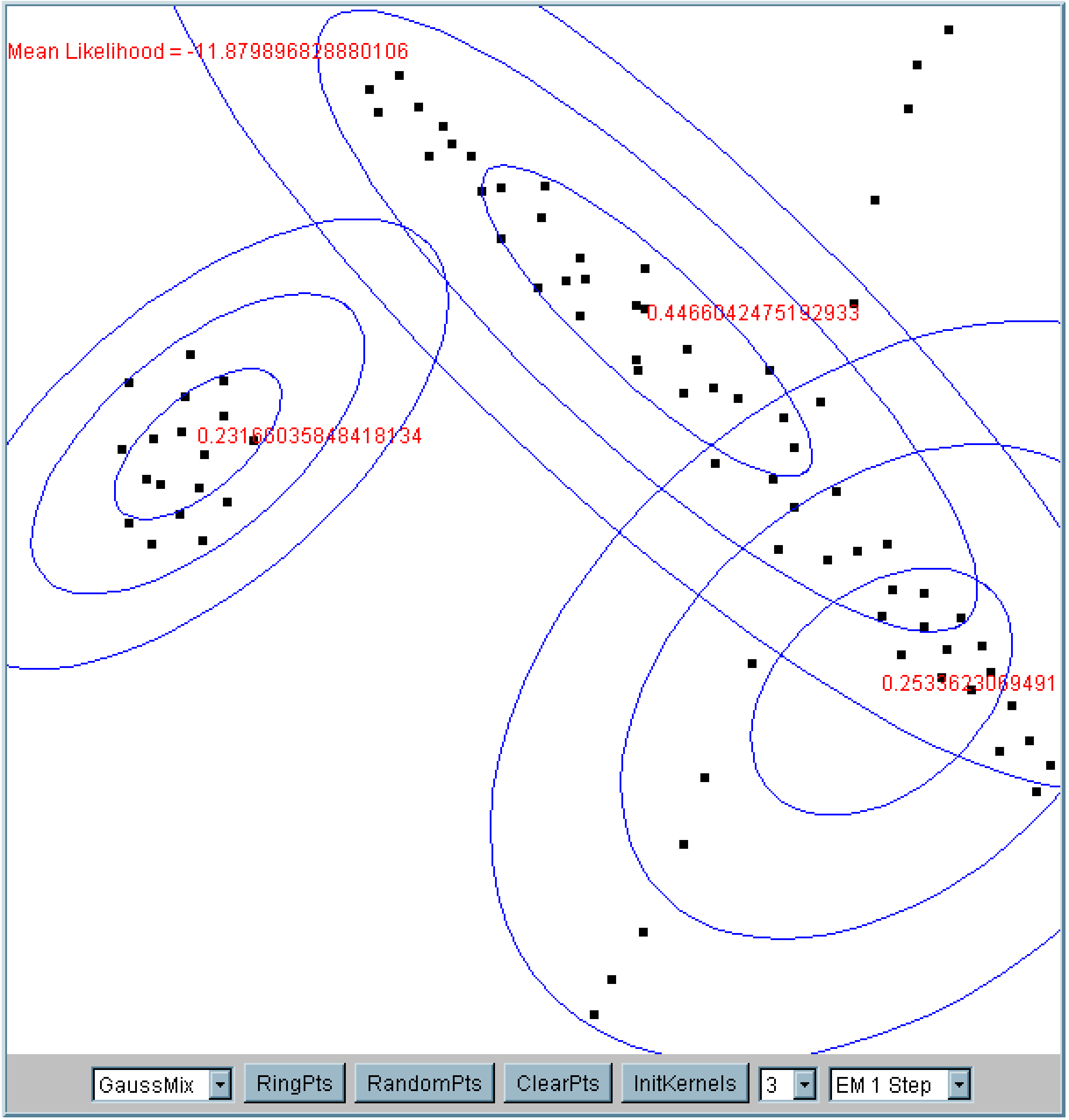


Iteration 2



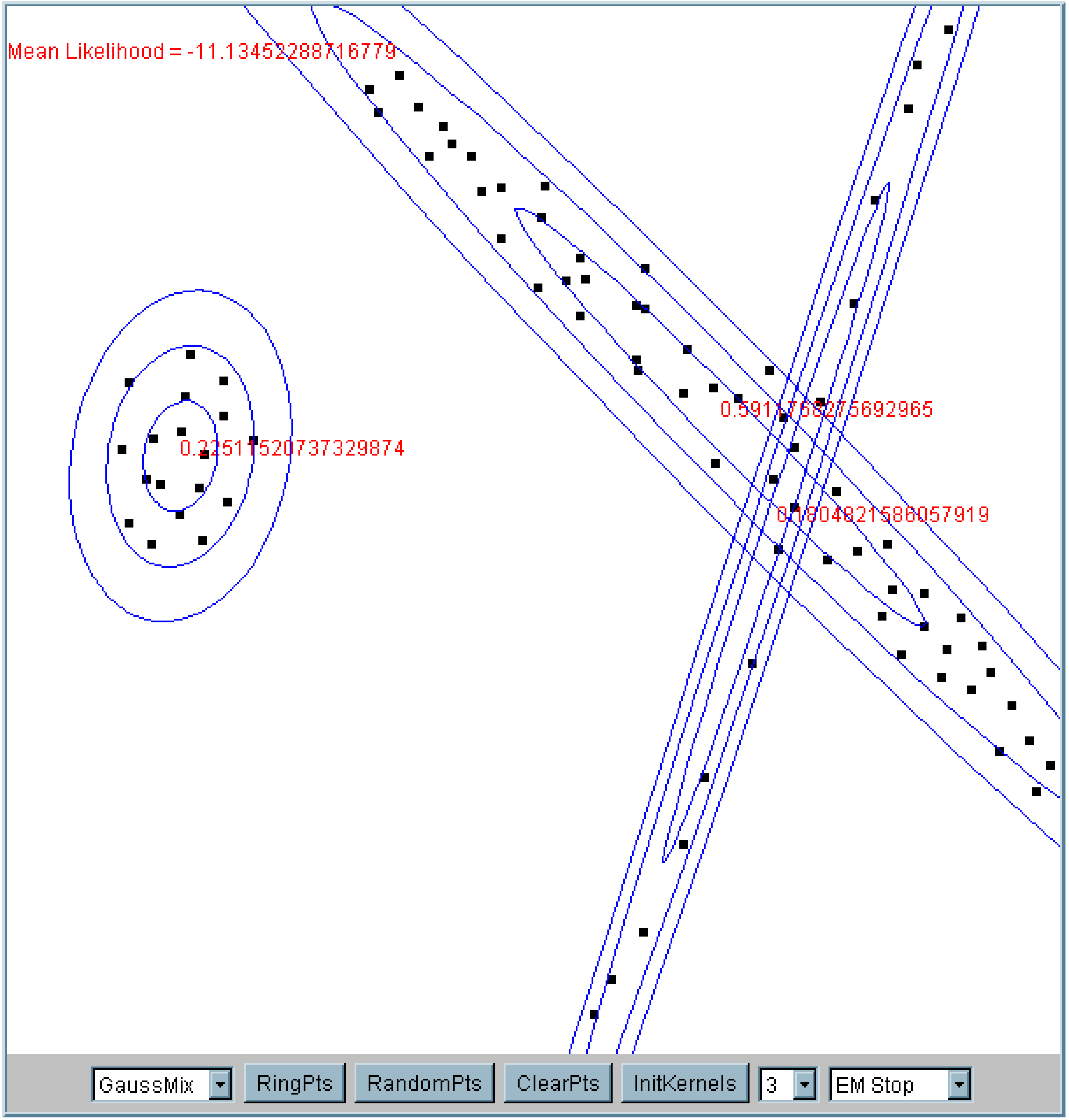
Mean Likelihood = -12.501313295068318
0.4891978834773868
0.2339252495610279
0.2319960960032283
GaussMix
RingPts
RandomPts
ClearPts
InitKernels
3
EM 1 Step

Iteration 5



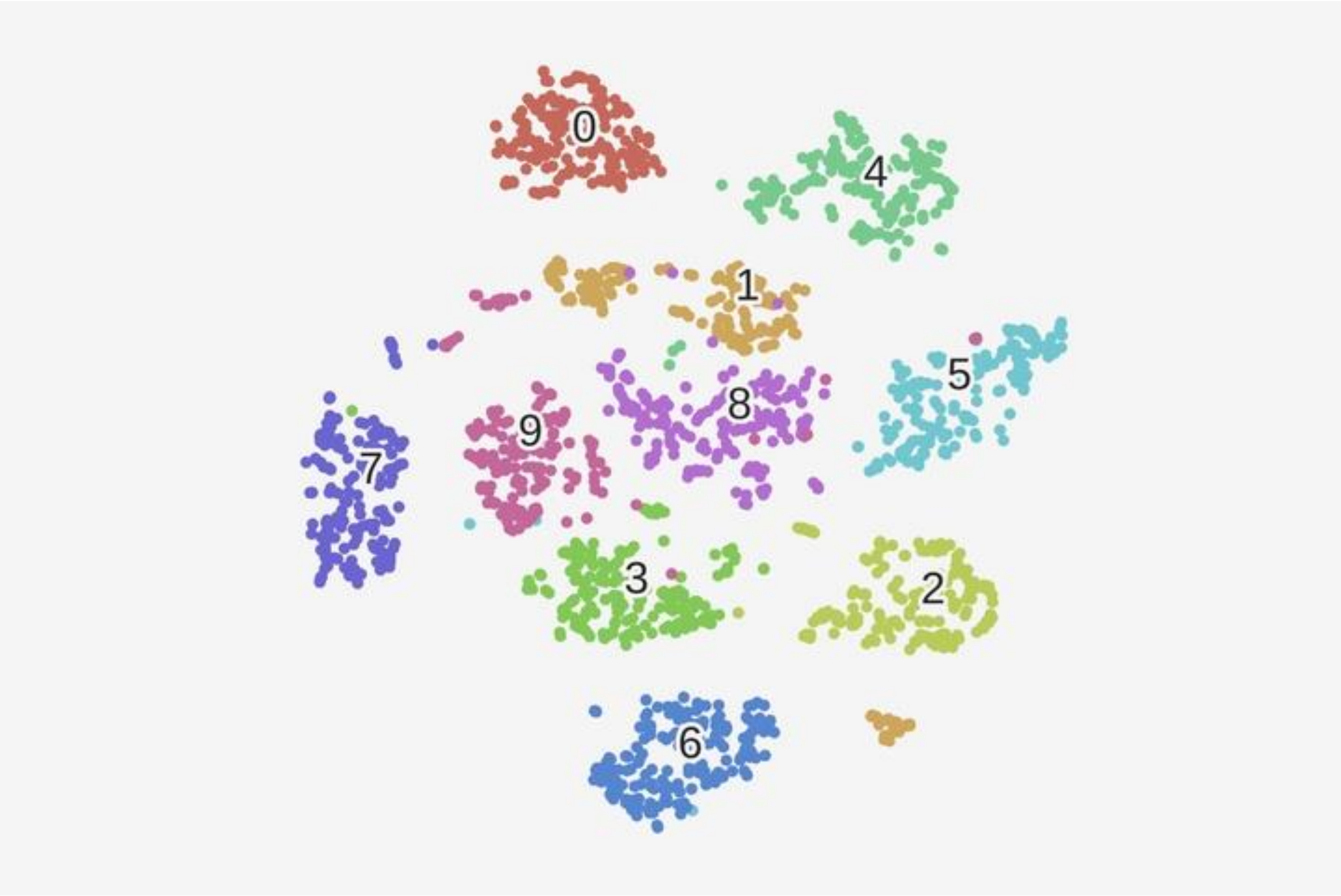
Mean Likelihood = -11.879896828880106
0.44660424751 92933
0.2316603584818134
0.2536230 69491
GaussMix
RingPts
RandomPts
ClearPts
InitKernels
3
EM 1 Step

Iteration 25



T-SNE

t-distributed stochastic neighbor embedding



T-SNE Distance Metric

Uses the following density-based (probabilistic) distance metric

$$p_{j|i} = \frac{\exp\left(-|x_i - x_j|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-|x_i - x_k|^2 / 2\sigma_i^2\right)}$$

Measures how (relatively) close x_j is from x_i, considering a Gaussian distribution around x_i with a given variance σ^2_i.

- this variance is different for every point
- t is chosen such that points in dense areas are given a smaller variance than points in sparse areas

T-SNE Implementation

Use a symmetrized version of the conditional similarity:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

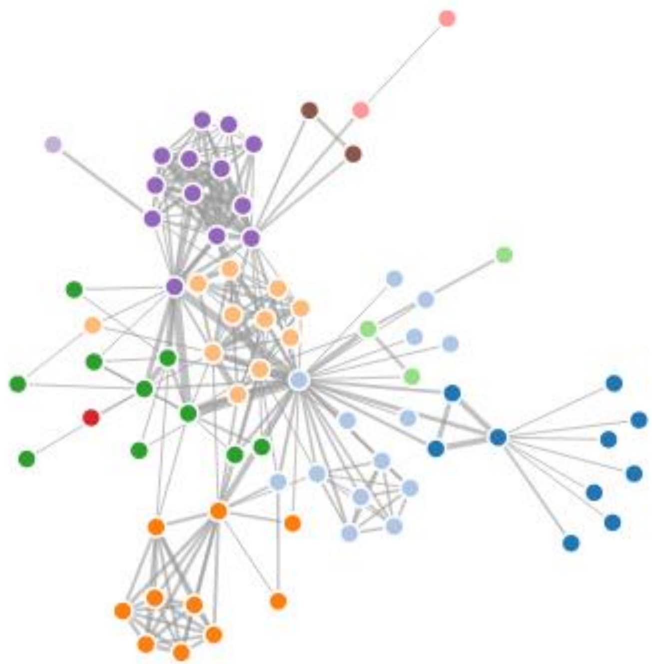
Similarity (distance) metric for mapped points:

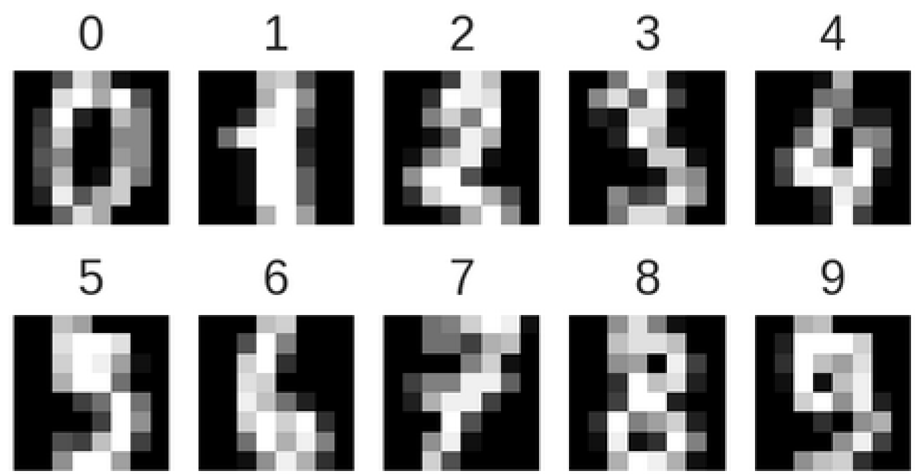
$$q_{ij} = \frac{f(|x_i - x_j|)}{\sum_{k \neq i} f(|x_i - x_k|)} \quad \text{with} \quad f(z) = \frac{1}{1+z^2}$$

This uses the t-student distribution with one degree of freedom, or Cauchy distribution, instead of a Gaussian distribution

Layout

Can use mass-spring system enforcing minimum of $|p_{ij} - q_{ij}|$





The classic *handwritten digits* datasets. It contains 1,797 images with 8∗8=64 pixels each.

Animated layout

More information

See this webpage

Time Series Data

Rectangular data set with a temporal component

	A	K	L	M	N
1	State	Burglary	Larceny-theft	Motor Vehicle Theft	Arson2
2	CALIFORNIA	2,616	6,298	3,344	71
3	MICHIGAN	1,049	979	154	72
4	MICHIGAN	5,638	8,451	5,828	296
5	TENNESSEE	5,604	12,141	1,373	177
6	MISSOURI	1,960	6,432	1,542	79
7	MARYLAND	3,372	8,761	1,936	137
8	ALABAMA	1,942	3,964	451	
9	OHIO	3,759	5,118	2,008	148
10	ILLINOIS	875	2,242	196	18
11	ARKANSAS	1,852	5,012	589	51
12	CALIFORNIA	2,212	4,450	1,100	43
13	WISCONSIN	2,819	7,622	1,719	120
14	GEORGIA	3,007	8,209	2,328	37

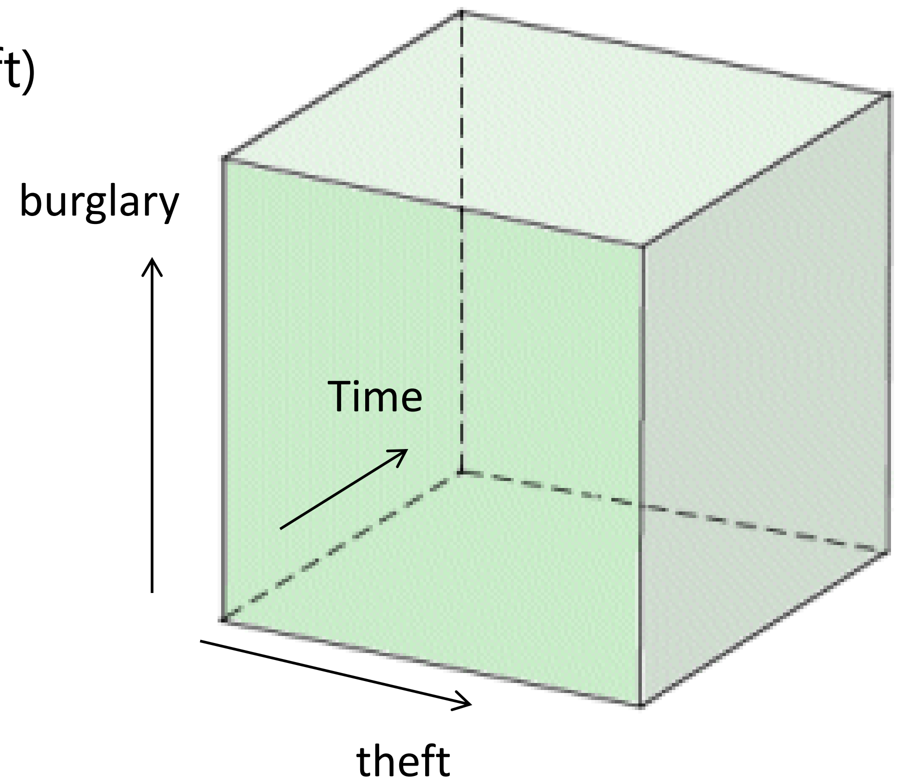
- assume you have these data for each year
- how to handle that, you might ask?

Time Cube

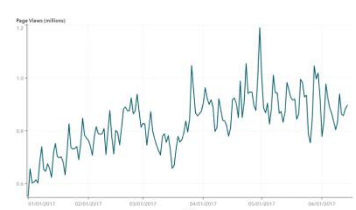
Assume for now we have

- two attributes (burglary, theft)
- both observed over time

Can visualize



- but each point is a time series!



Similarity Measures

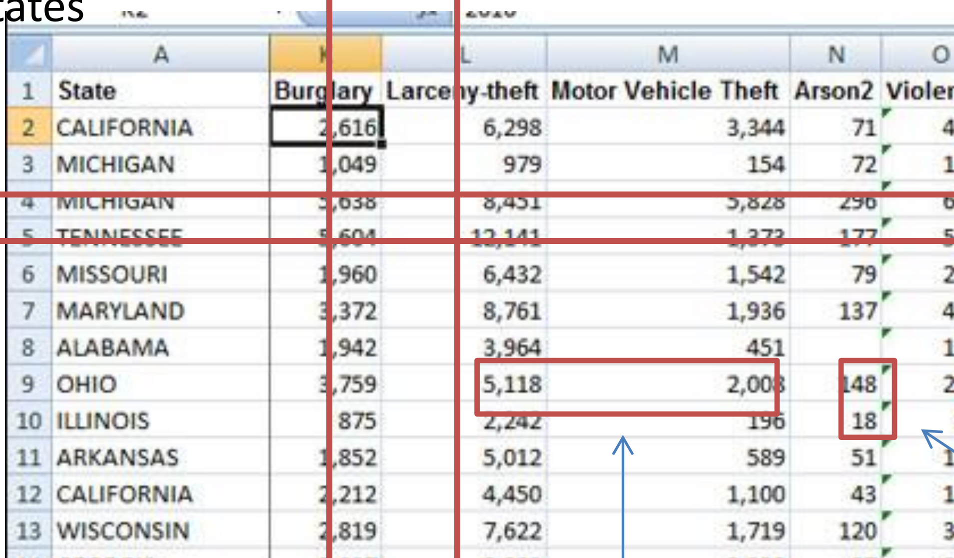
Needed it for clustering

- recall Euclidean, correlation, cosine distances

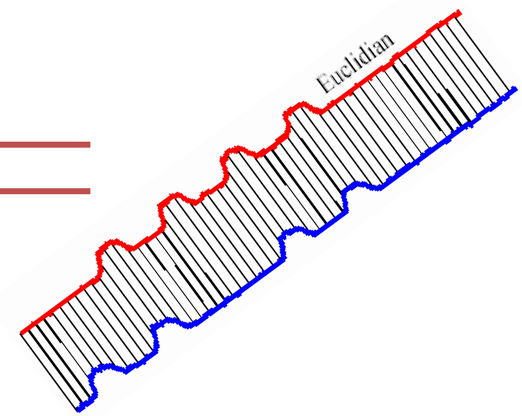
similarity of two states

similarity of two crimes

two time series



	A	K	L	M	N	O
1	State	Burglary	Larceny-theft	Motor Vehicle Theft	Arson2	Violen
2	CALIFORNIA	2,616	6,298	3,344	71	41
3	MICHIGAN	1,049	979	154	72	10
4	MICHIGAN	[illegible],638	8,451	5,828	296	6[illegible]
5	TENNESSEE	5,604	12,141	1,373	177	5[illegible]
6	MISSOURI	1,960	6,432	1,542	79	24
7	MARYLAND	3,372	8,761	1,936	137	43
8	ALABAMA	1,942	3,964	451		14
9	OHIO	3,759	5,118	2,008	148	26
10	ILLINOIS	875	2,242	196	18	9
11	ARKANSAS	1,852	5,012	589	51	1[illegible]
12	CALIFORNIA	2,212	4,450	1,100	43	18
13	WISCONSIN	2,819	7,622	1,719	120	37
14	GEORGIA	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

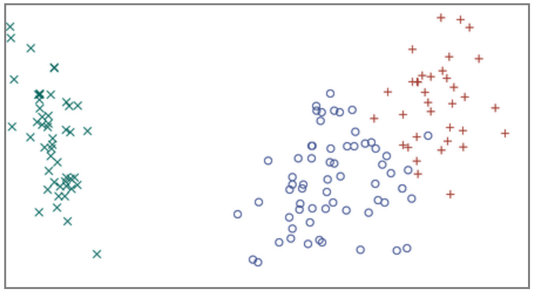


Time

similarity of two crimes in a given state over time

similarity of two states for a given crime over time

Clustering



What can be clustered with these measures?

- crimes (averaged over time)
- states (averaged over time)
- crimes in a given state (taking time series into account)
- states for a given crime (taking time series into account)

Can we get more inclusive?

- cluster crimes but including the time series characteristics
- cluster states but including the time series characteristics

Capture more information about their time series when you compare two data points

- compute the similarity of two crimes by summing the times-series similarities for each state
- compute the similarity of two states by summing the times-series similarities for each crime

In Practice

Time-series aware similarity (distance) S_{tsa} for a pair of states

for a given pair of states i, j

for each crime c

compute the time series *similarity* → sim_t(c)

sum all sim_t(c) → $S_{tsa}(i,j)$

similarity could be some measure of correlation of the two time series vectors

S_{tsa} for a pair of crimes

for each pair of crimes i, j

for each state s

compute the time series *similarity* → sim_t(s)

sum all sim_t(s) → $S_{tsa}(i,j)$

If the time series are aligned for all states then the S_{tsa} will be high and the two crimes have very similar time behaviors nationwide.

Some Thoughts

The time series might not be aligned

- one crime might cause another
- can apply dynamic time warping (see next)

You may (also) have a geospatial component in your data

- can use them as a regular attribute (encoded by an ID)
- can you make them more continuous and linearly ordered?
- use a space filling curve (see next)

You may want to just keep time instances as separate entities

- that will work too
- then you might discover clusters that are sensitive to time
- or you can see how the years relate to another along a trajectory
- as a general rule, when you visualize multivariate data first decide what you will put into the rectangular data matrix (samples, attributes)

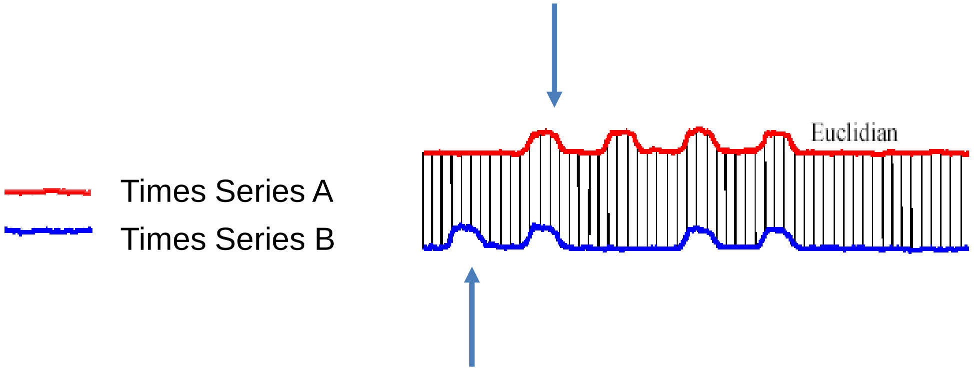
L_p Norm and its Shortcomings

Standard pairwise distance

$$Dist(\overline{X}, \overline{Y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

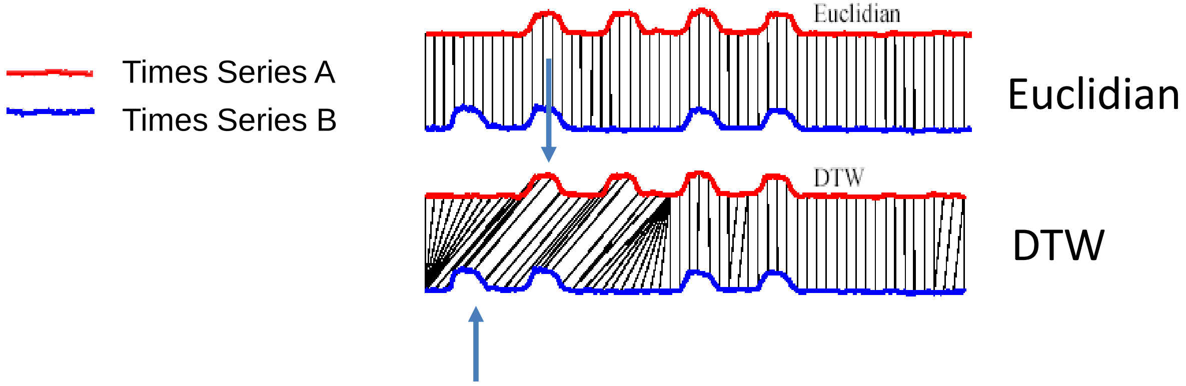
Shortcomings:

- designed for time series of equal length
- cannot address distortions on the temporal (contextual) attributes



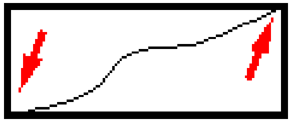
Dynamic Time Warping Distance

Can better accommodate local mismatches

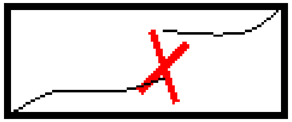


Three constraints

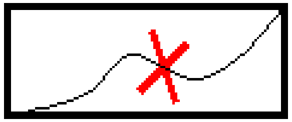
- no skipping of beginning or ends of either sequence



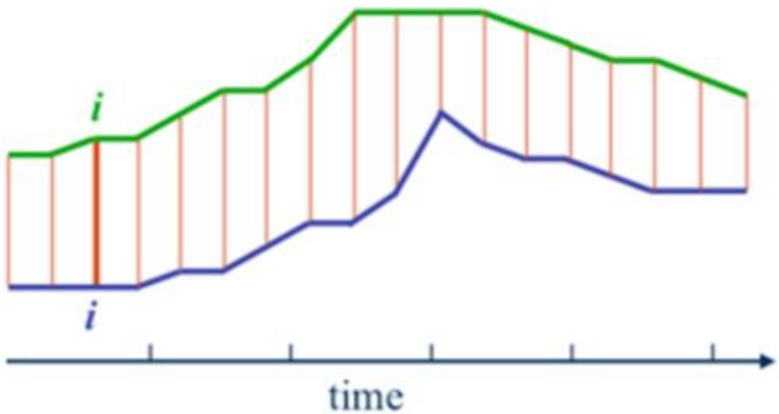
- continuity – no jumps



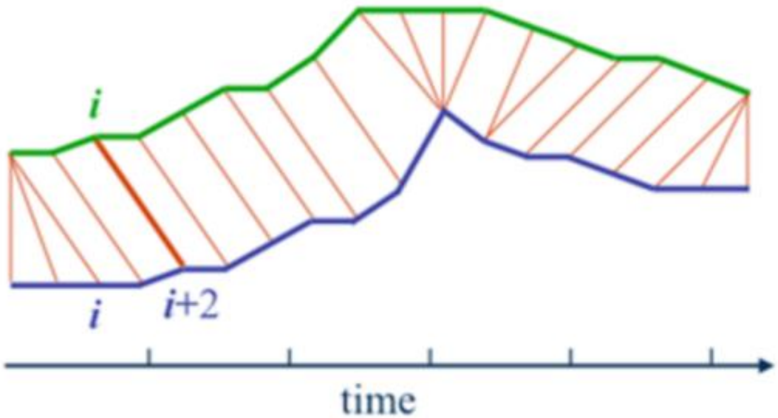
- monotonicity – can't go back in time



DTW – Find The Minimum Cost Path

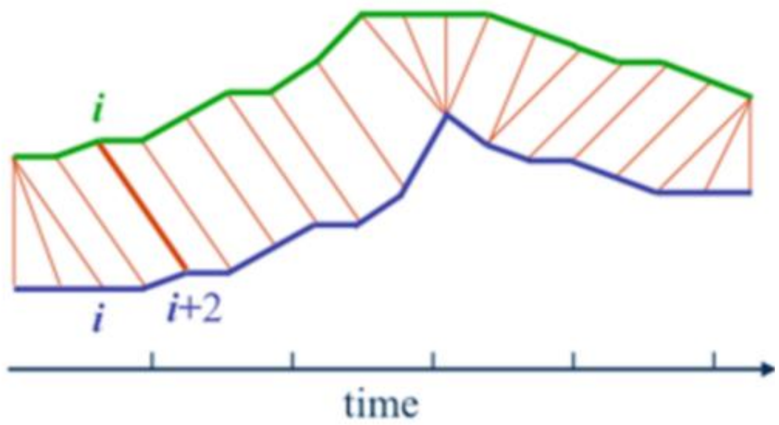


Euclidian



DTW

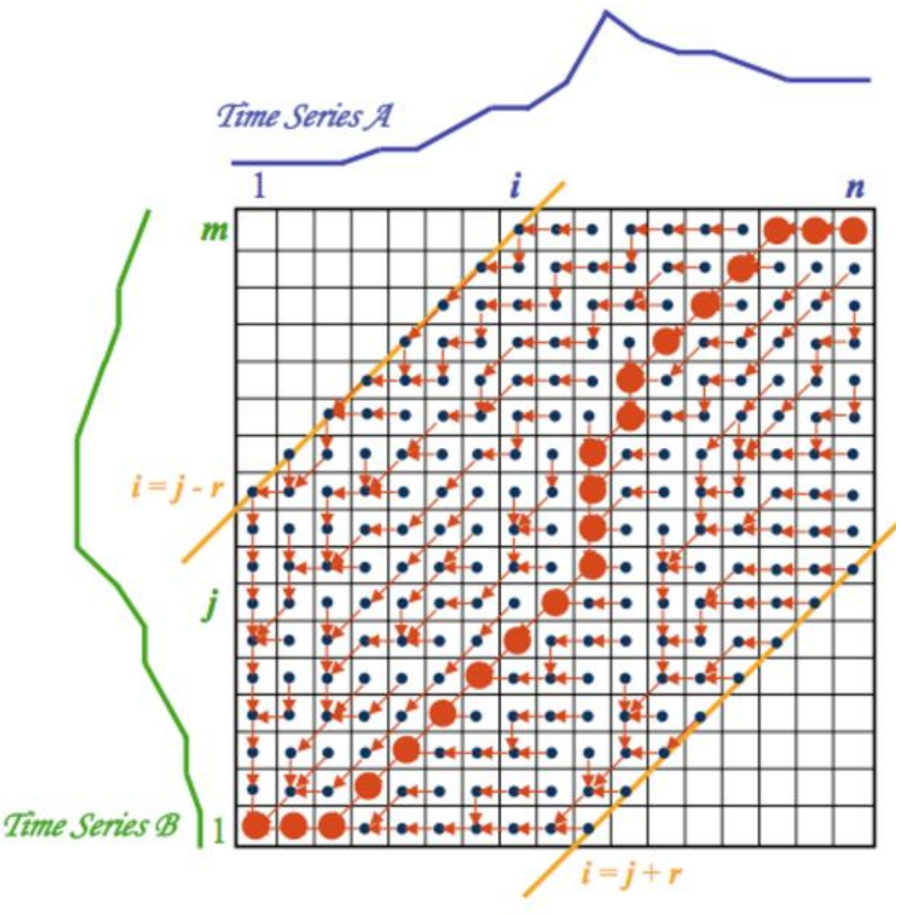
DTW – Find The Minimum Cost Path



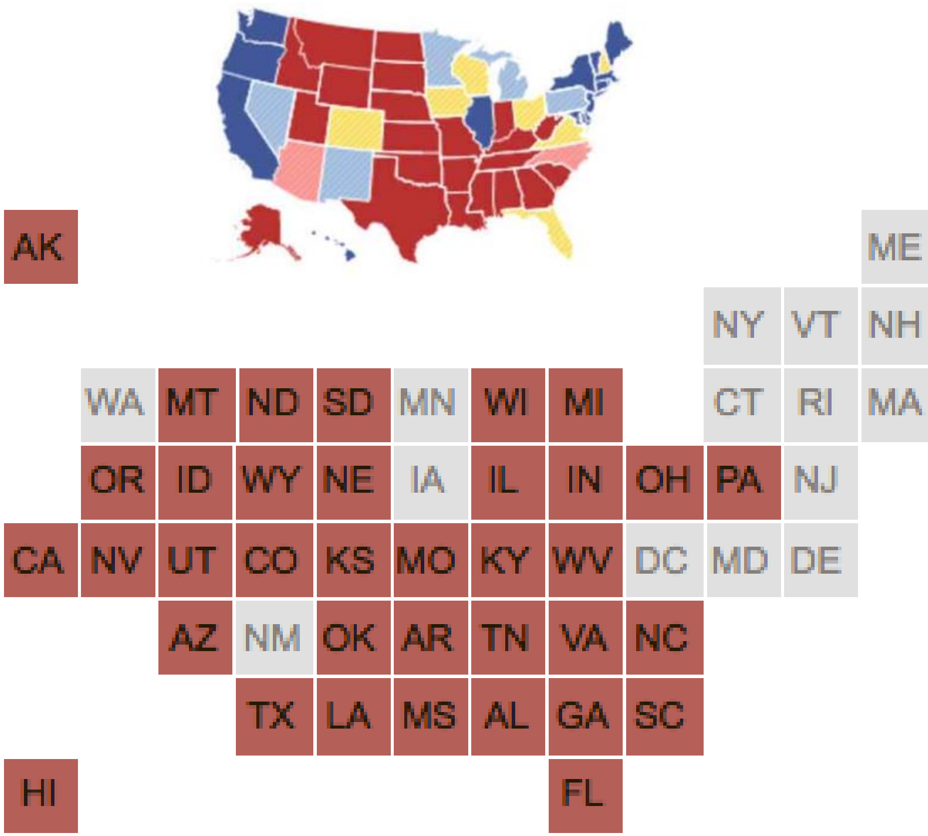
DTW

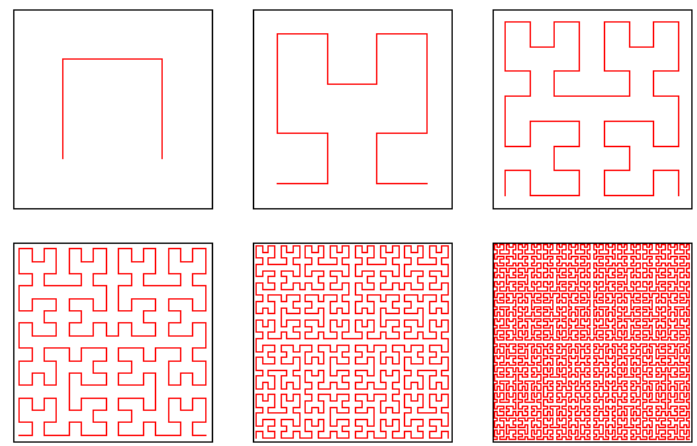
Compute using dynamic programming

Available in python



Linearizing Map Locations



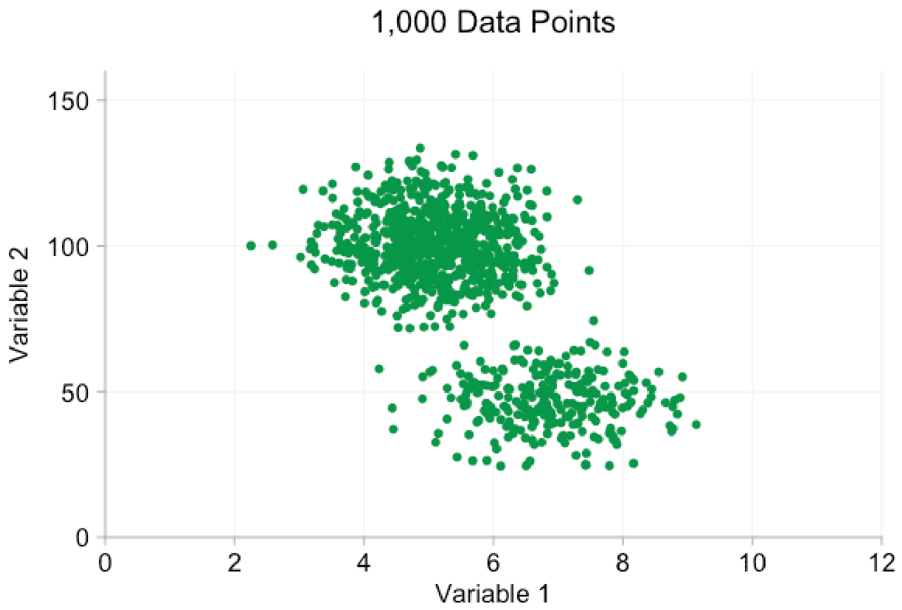


Convert a geographical map into a grid map

Linearize using a space-filling curve (Hilbert curve)

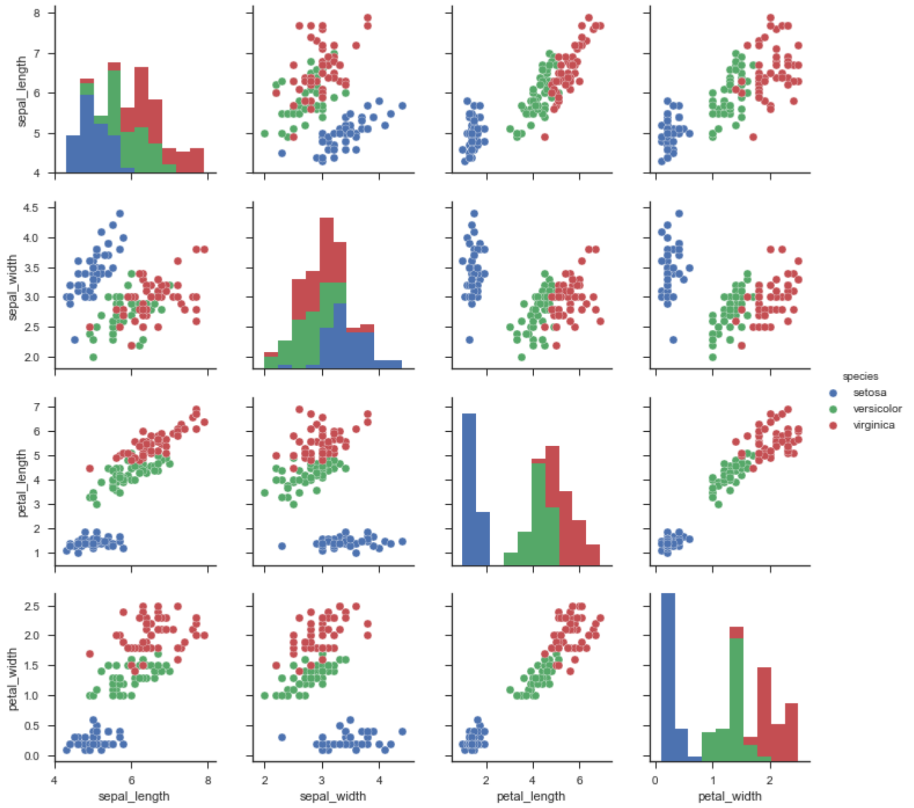
Scatterplots

Plot two variables



But what if you have more than two variables?

Scatterplot Matrix



Problem:

- Multivariate relationships are scattered across the tiles

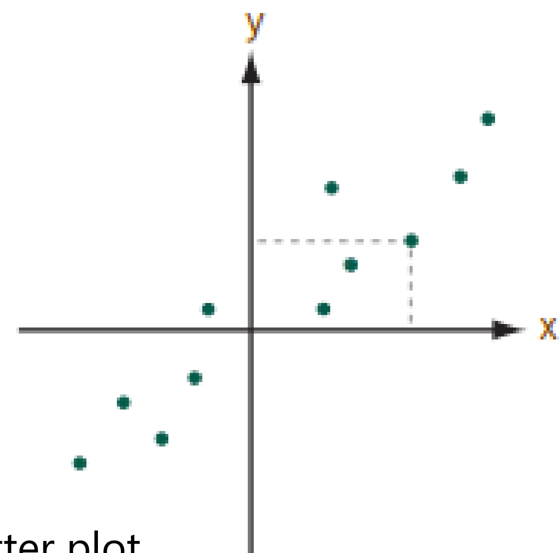
BIPLOTS

Plots data points and dimension axes into a single visualization

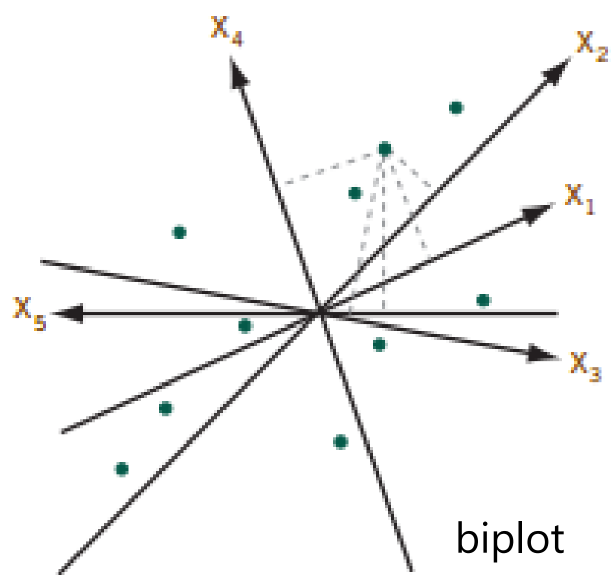
- uses first two PCA vectors as the basis to project into
- find plot coordinates [x] [y]

  for data points: [PCA_1 · data vector] [PCA_2 · data vector]

  for dimension axes: [PCA_1[dimension]] [PCA_2[dimension]]



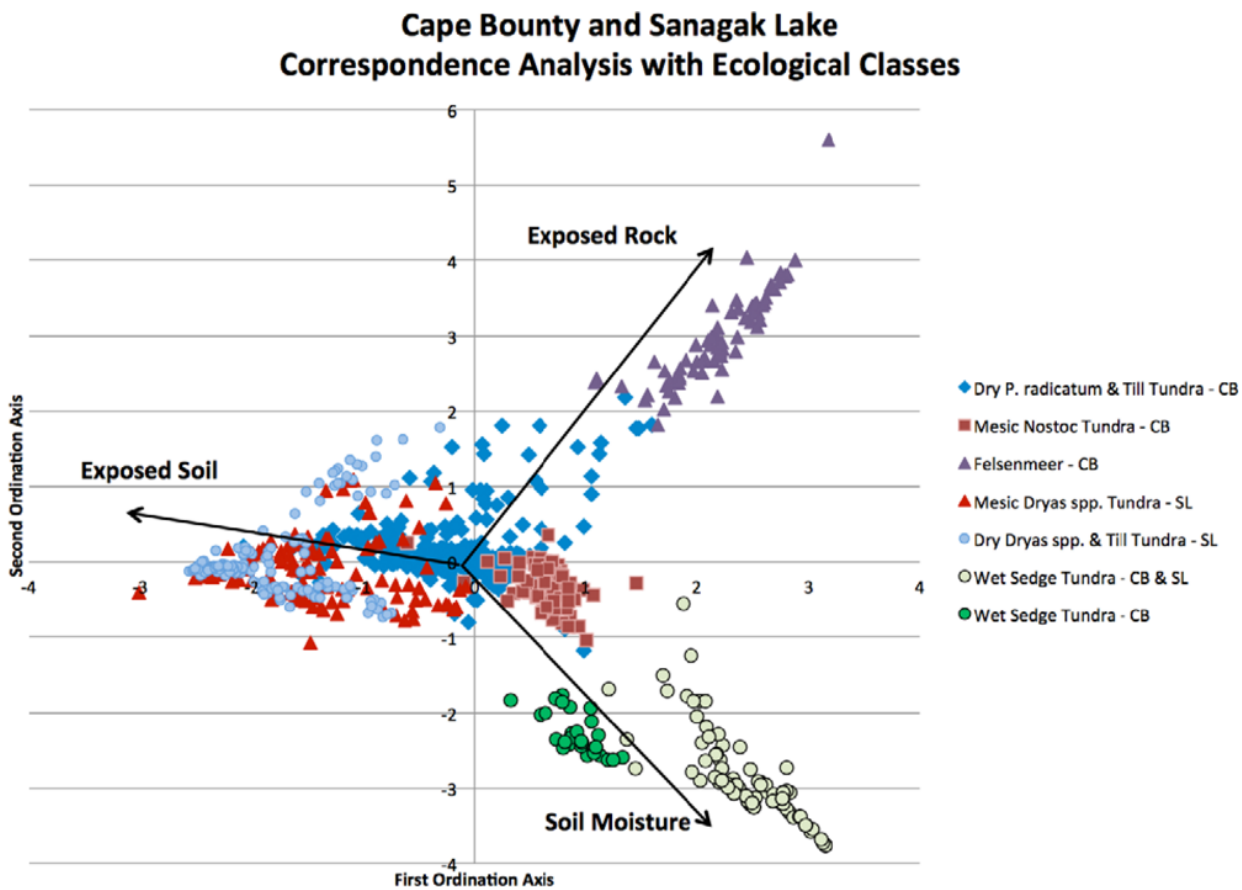
scatter plot



biplot

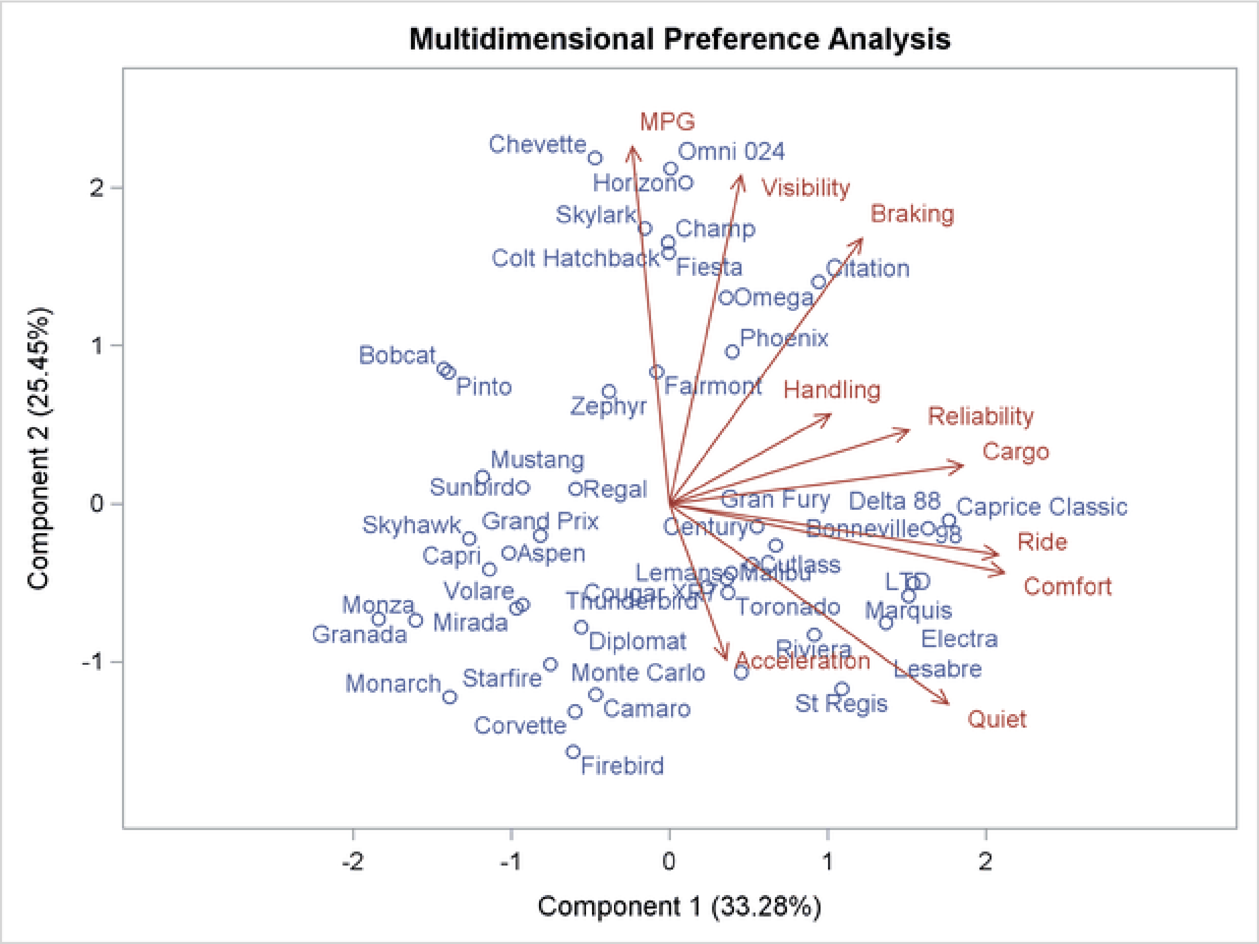
Biplots in Practice

See data distributions into the context of their attributes



BIPLOTS IN PRACTICE

See data points into the context of their attributes



Biplots – A Word of caution

Do be aware that the projections may not be fully accurate

- you are projecting N-D into 2D by a linear transformation
- if there are more than 2 significant PCA vectors then some variability will be lost and won't be visualized
- remote data points might project into nearby plot locations suggesting false relationships
- leads to projection ambiguity

Multidimensional Scaling (MDS)

MDS preserves similarity relationships, prevents ambiguity

- scattered points in high-dimensions (N-D)
- adjacency matrices

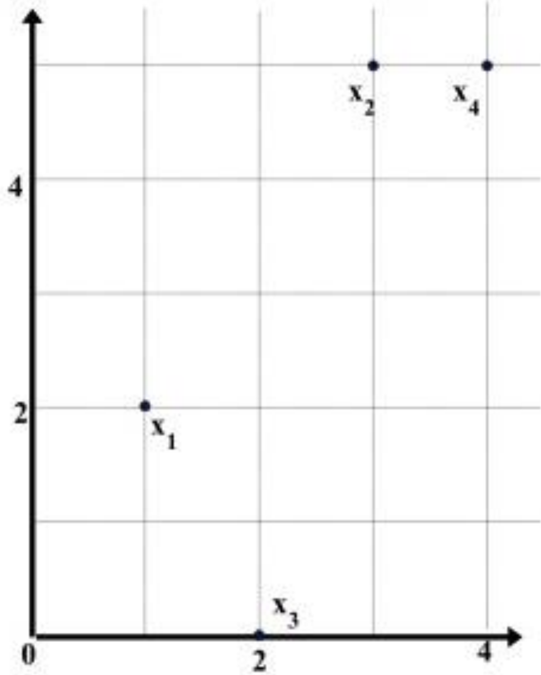
Maps the distances between observations from N-D into low-D (say 2D)

- attempts to ensure that differences between pairs of points in this reduced space match as closely as possible

The input to MDS is a distance (similarity) matrix

- actually, you use the *dissimilarity* matrix because you want similar points mapped closely
- dissimilar point pairs will have greater values and map father apart

The Dissimilarity Matrix



Data Matrix

point	attribute1	attribute2
x1	1	2
x2	3	5
x3	2	0
x4	4	5

Dissimilarity Matrix

(with Euclidean Distance)

	x1	*x2*	*x3*	*x4*
x1	0			
x2	3.61	0		
x3	2.24	5.1	0	
x4	4.24	1	5.39	0

Distance Matrix

MDS turns a distance matrix into a network or point cloud

- correlation, cosine, Euclidian, and so on

Suppose you know a matrix of distances among cities

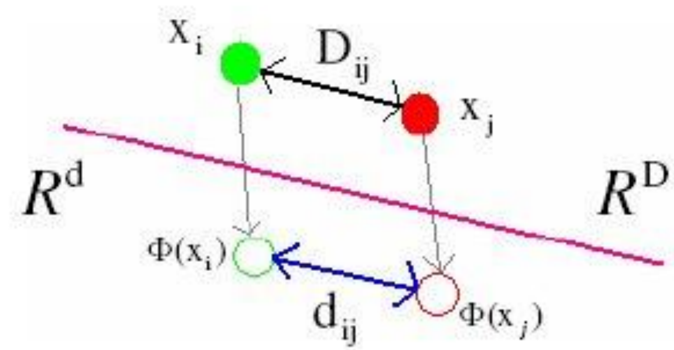
	Chicago	Raleigh	Boston	Seattle	S.F.	Austin	Orlando
Chicago	0						
Raleigh	641	0					
Boston	851	608	0				
Seattle	1733	2363	2488	0			
S.F.	1855	2406	2696	684	0		
Austin	972	1167	1691	1764	1495	0	
Orlando	994	520	1105	2565	2458	1015	0

Result of MDS

Compare with real Map

MDS ALGORITHM

- Task:
  - Find that configuration of image points whose pairwise distances are most similar to the original inter-point distances !!!
- Formally:
  - Define: $D_{ij} = \| x_i - x_j \|_D$ $\quad d_{ij} = \| y_i - y_j \|_d$
  - Claim: $D_{ij} \equiv d_{ij} \quad \forall i, j \in [1, n]$
- In general: an exact solution is not possible !!!
- Inter Point distances → invariance features



MDS Algorithm

<u>Strategy (of metric MDS):</u>

- iterative procedure to find a good configuration of image points
    - 1) Initialization
      → Begin with some (arbitrary) initial configuration
    - 2) Alter the image points and try to find a configuration of points that minimizes the following sum-of-squares error function:

MDS Algorithm

Strategy (of metric MDS):

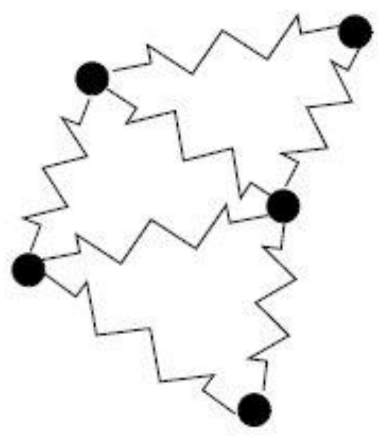
- iterative procedure to find a good configuration of image points
    - 1) Initialization
      → Begin with some (arbitrary) initial configuration
    - 2) Alter the image points and try to find a configuration of points that minimizes the following sum-of-squares error function:

$$E = \sum_{i<j}^{N} \left(D_{ij} - d_{ij}\right)^2$$

Force-Directed Algorithm

Spring-like system

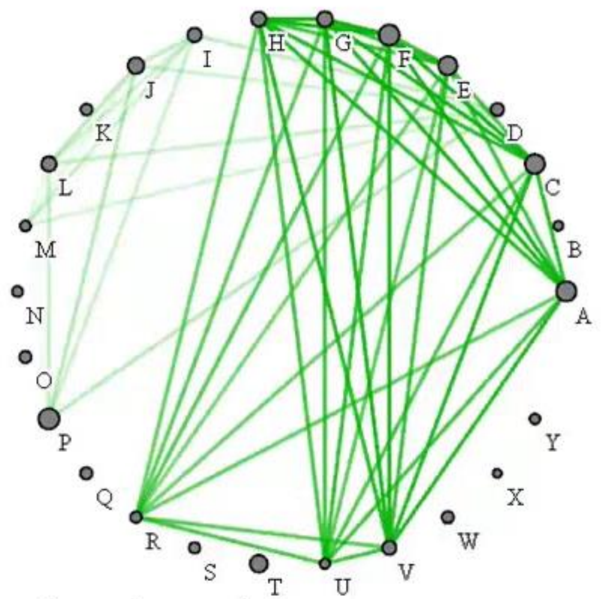
- insert springs within each node
- the length of the spring encodes the desired node distance
- start at an initial configuration
- iteratively move nodes until an energy minimum is reached



Force-Directed Algorithm

Spring-like system

- insert springs within each node
- the length of the spring encodes the desired node distance
- start at an initial configuration
- iteratively move nodes until an energy minimum is reached



Vertex layout by correlations.

Uses of MDS

Distance (similarity) metric

- Euclidian distance (best for data)
- Cosine distance (best for data)
- |1-correlation| distance (best for attributes)
- use 1-correlation to move correlated attribute points closer
- use | | if you do not care about positive or negative correlations

MDS Examples