CSE 591: Visual Analytics

Lecture 6: Data Transformations and Analysis

Klaus Mueller

Computer Science Department

Stony Brook University

High Dimensional Data

\# dimensions >> 3

Problems:

- hard to visualize
- massive storage
- hard to analyze (clustering and classification more efficient on low-D data)

Solution:

- reduce number of dimensions (but control loss)
- stretch N-D space somehow into 2D or 3D
- analyze (discover) structure, organize

We will discuss:

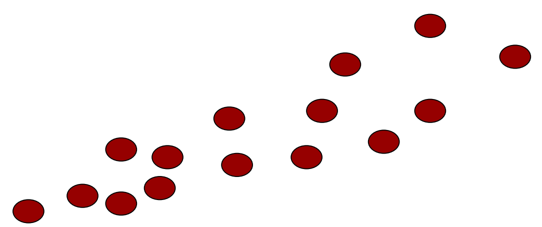
- principal component analysis (PCA) → reduce dimensions
- multi-dimensional scaling (MDS) → stretch space
- clustering → provide structure
- create hierarchies → provide structure
- self-organizing maps → provide structure

PCA: Algebraic Interpretation

Given m points in a n dimensional space, for large n, how does one project on to a low dimensional space while preserving broad trends in the data and allowing it to be visualized?

PCA: Algebraic Interpretation – 1D

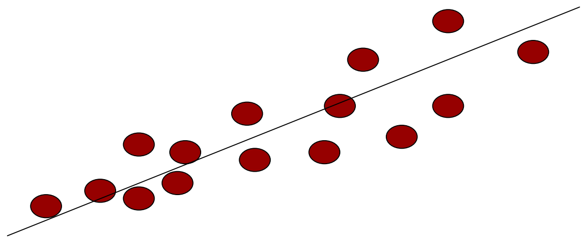
Given m points in a n dimensional space, for large n, how does one project on to a 1 dimensional space?



Choose a line that fits the data so the points are spread out well along the line

PCA: Algebraic Interpretation – 1D

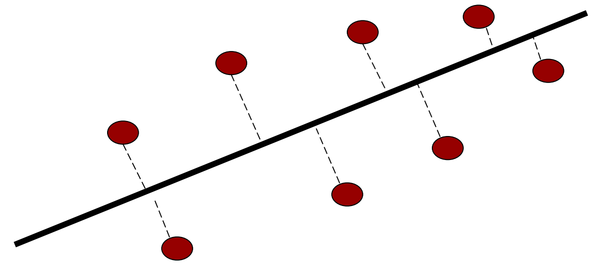
Given m points in a n dimensional space, for large n, how does one project on to a 1 dimensional space?



Choose a line that fits the data so the points are spread out well along the line

PCA: Algebraic Interpretation – 1D

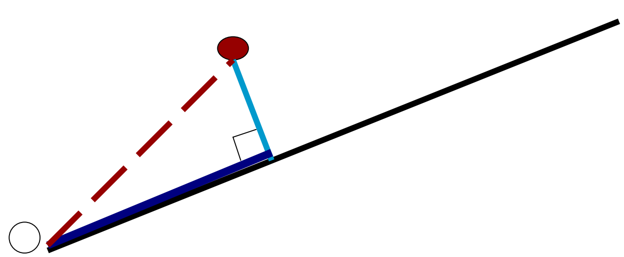
Formally, minimize sum of squares of distances to the line.



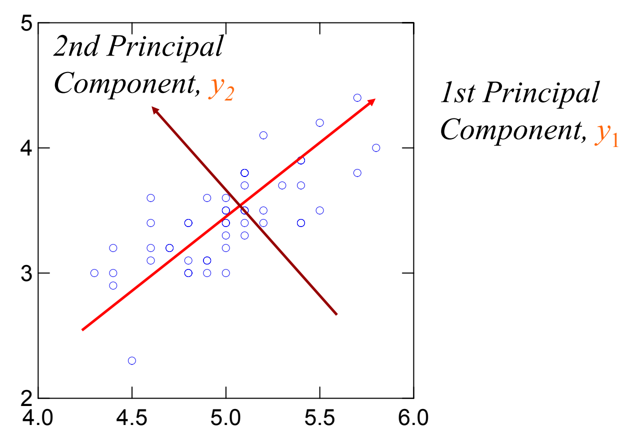
Why sum of squares? Because it allows fast minimization,

PCA: Algebraic Interpretation – 1D

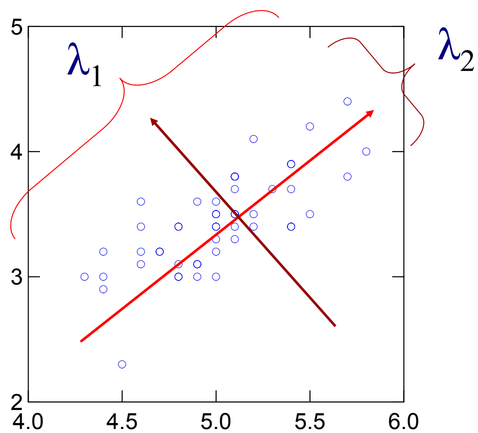
Minimizing sum of squares of distances to the line is the same as maximizing the sum of squares of the projections on that line, thanks to Pythagoras.



PCA Scores



PCA Eigenvalues



PCA: Solution

Also known to engineers as the Karhunen-Loéve Transform (KLT)

Rotate data points to align successive axes with directions of greatest variance

- subtract mean from data
- normalize variance along each direction, and reorder according to the variance magnitude from high to low
- normalized variance direction = principle component

Eigenvectors of system's Covariance Matrix ***C***

Permute eigenvectors so they are in descending order of eigenvalues

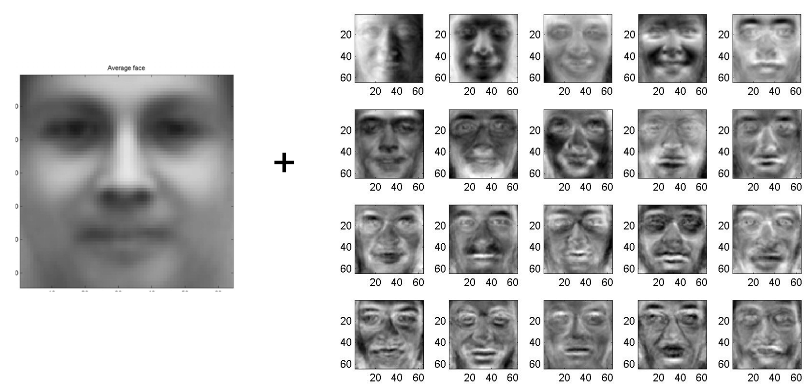
$$\mathbf{C} = \frac{1}{n-1}\sum_{i}^{n}(\bar{x}_i - \mu)(\bar{x}_i - \mu)^T \qquad (\mathbf{C} - \lambda_i \mathbf{I})\bar{e}_i = 0$$

PCA Applied to Faces

Some familiar faces…

PCA Applied to Faces

We can reconstruct each face as a linear combination of "basis" faces, or Eigenfaces [M. Turk and A. Pentland (1991)]



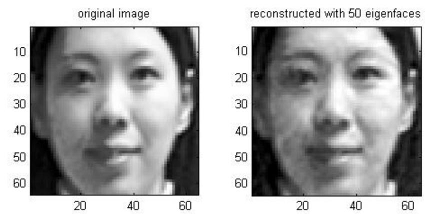
Reconstruction using PCA

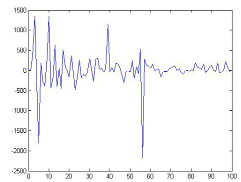
90% variance is captured by the first 50 eigenvectors

Reconstruct existing faces using only 50 basis images

We can also generate new faces by combining eigenvectors with different weights

f





Multidimensional Scaling (MDS)

Maps the distances between observations from N-D into a lower-D space (say 2D)

Attempts to ensure that differences between pairs of points in this reduced space match, as closely as possible, the true-ordered differences between the observations.

Algorithm:

- compute the pair-wise Euclidian distance D_{ij}
- order these in terms of magnitude
- minimize energy function to get d_{ij} in lower-D space

$$E = \frac{\sum_{r=1}^{N} \sum_{s=1}^{r-1} \frac{(D_{rs} - d_{rs})^2}{D_{rs}}}{\sum_{r=1}^{N} \sum_{s=1}^{r-1} D_{rs}}$$

MDS: Specifics

Specify input as a dissimilarity matrix M, containing pairwise dissimilarities between N-dimensional data points

Finds the best D-dimensional linear parameterization compatible with M (down to rigid-body transform + possible reflection)

(in other words, output a projection of data in D-dimensional space where the pairwise distances match the original dissimilarities as faithfully as possible)

MDS is related to PCA when distances are Euclidian, but

- PCA provides low dimensional images of data points
- inadequacy of PCA: clustered structures may disappear

MDS project data points to low dimensional images AND

- respect constraints:
- keep informational content
- keep similarity / dissimilarity relationships

MDS: Applications

Dissimilarities can be metric or non-metric

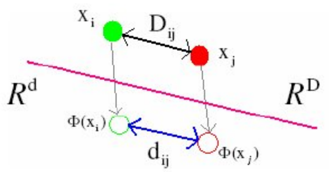
Useful when absolute measurements are unavailable

- uses relative measurements

Computation is invariant to dimensionality of data

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

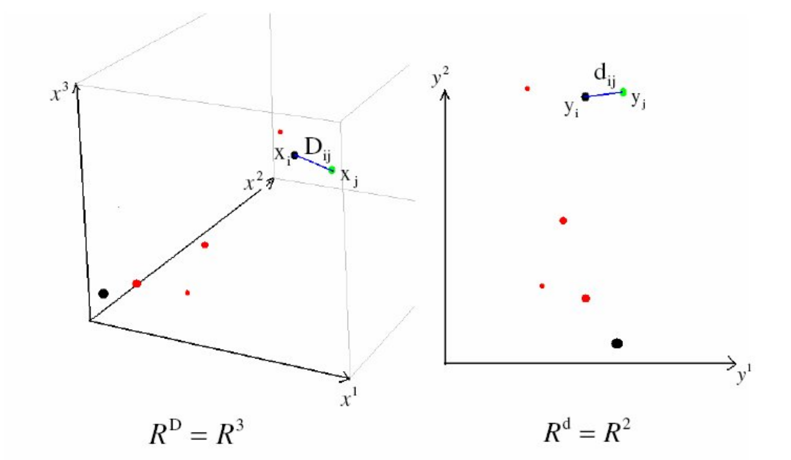
Strategy (of metric MDS):

- 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:

$$E[y_1,\ldots,y_n] = \frac{1}{\sum_{i<j} D_{ij}} \sum_{i<j} \frac{(d_{ij} - D_{ij})^2}{D_{ij}} = \frac{1}{\sum_{i<j} D_{ij}} \sum_{j} \sum_{i<j} \frac{(\| y_i - y_j \| - D_{ij})^2}{D_{ij}}$$

$$\nabla_{y_k} (E[y_1,\ldots,y_n])$$

MDS: Algorithm



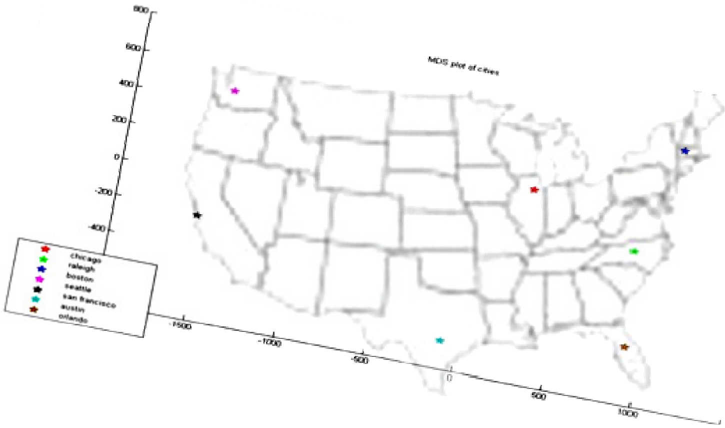
An Example: Map of the US

Suppose you know the distances between a bunch of 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

Distances calculated with geobytes.com/CityDistanceTool

Result of MDS



Actual Plot of Cities

Self-Organizing Maps (SOM)

Introduced by Teuvo Kohonen

- unsupervised learning and clustering algorithm
- has advantages compared to hierarchical clustering
- often realized as an artificial neural network

SOMs group the data

- they perform a nonlinear projection from N-dimensional input space onto two-dimensional visualization space
- they provide a useful topological arrangement of information objects in order to display clusters of similar objects in information space

SOM: Algorithm

Consists of a two-dimensional network of neurons, typically arranged on a regular lattice.

- each cell is associated with a single randomly initialized N-dimensional reference vector.

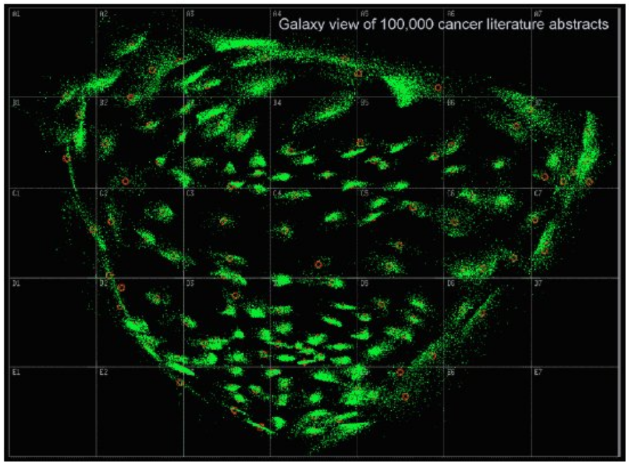
Training uses a set of input vectors several times:

- for each input vector search the map for the most similar reference vector, called the winning vector
- update the winning vector such that it more closely represents the input vector
- also adjust the reference vectors in the neighborhood around the winning vector in response to the actual input vector

After the training:

- reference vectors in adjacent cells represent input vectors which are close (i.e., similar) in information space

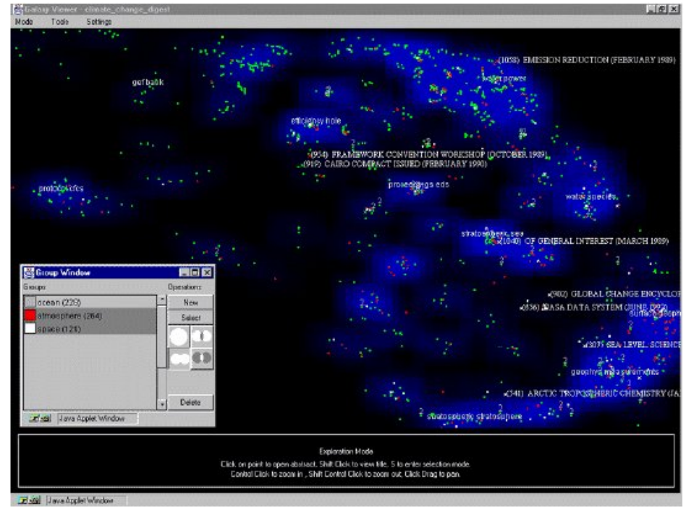
SOM Examples: Galaxies



Presentation of documents where similar ones cluster together

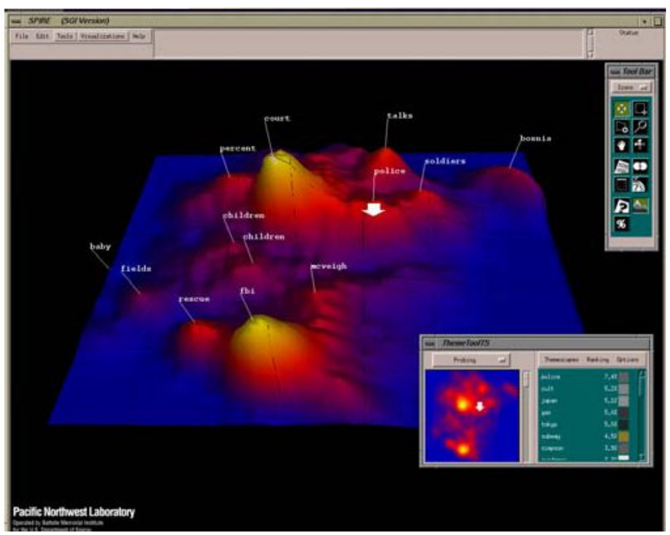
PNNL

SOM Examples: Webtheme



PNNL

SOM Examples: Themescape



PNNL

Uses 3D representation: height represents density or number of documents in region

SOM / MDS Example: VxInsight (Sandia)

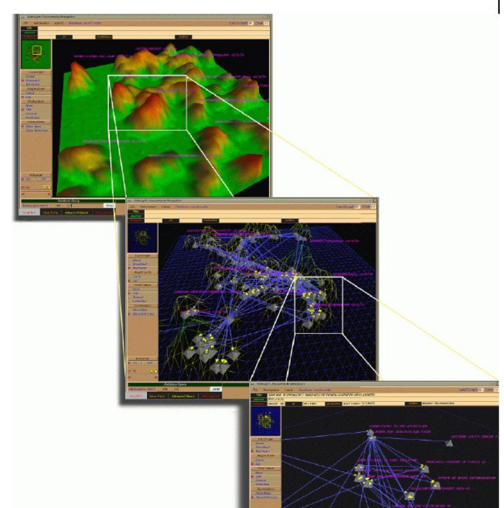
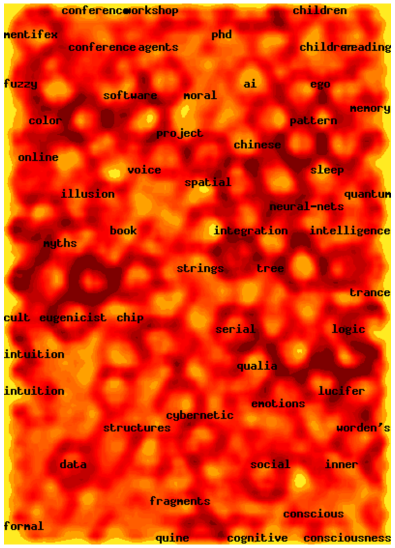
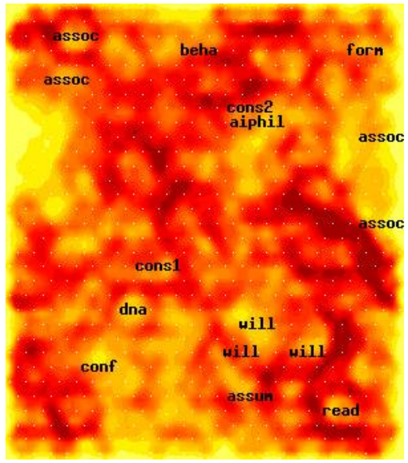


Figure 4: Multi-resolution exploration with detail on demand.

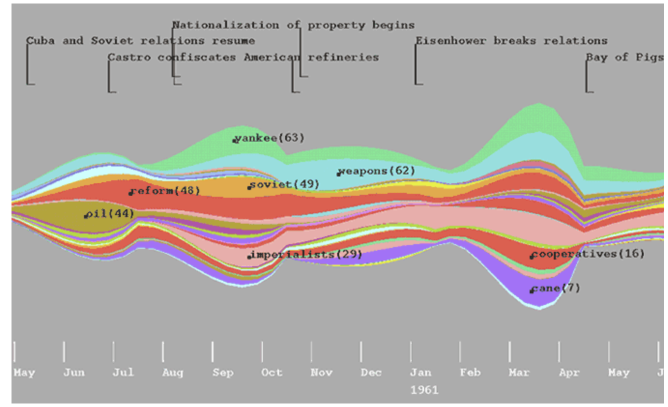
SOM Examples: Websom





Self-organizing map of Net newsgroups and postings (websom.hut.fi)

Theme River



Data as a stream along time

PNNL

Force-Directed Methods

Force-directed methods can remove remaining occlusions/overlaps in the 2D projection space:

- forces are used to position clusters according to distance (and variance) in N-space
- insert springs within each node
- the length of the spring encodes the desired node distance
- starting at an initial configuration, iteratively move nodes until an energy minimum is reached