CSE 590
Data Science Fundamentals

Data Preparation And reduction II

Klaus Mueller

Computer Science Department
Stony Brook University and SUNY Korea

Lecture	Topic	Projects
1	Intro, schedule, and logistics	
2	Data Science components and tasks	
3	Data types	Project #1 out
4	Introduction to R, statistics foundations	
5	Introduction to D3, visual analytics	
6	Data preparation and reduction	
7	Data preparation and reduction	Project #1 due
8	Similarity and distances	Project #2 out
9	Similarity and distances	
10	Cluster analysis	
11	Cluster analysis	
12	Pattern miming	Project #2 due
13	Pattern mining	
14	Outlier analysis	
15	Outlier analysis	Final Project proposal due
16	Classifiers	
17	Midterm	
18	Classifiers	
19	Optimization and model fitting	
20	Optimization and model fitting	
21	Causal modeling	
22	Streaming data	Final Project preliminary report due
23	Text data	
24	Time series data	
25	Graph data	
26	Scalability and data engineering	
27	Data journalism	
	Final project presentation	Final Project slides and final report due

Data Preparation Tasks

Data cleaning

- fill in missing values
- smooth noisy data
- identify or remove outliers
- resolve inconsistencies

Data reduction

- obtain reduced volume, but get same/similar analytical results
- data discretization (for numerical data)
- data aggregation (summarization)
- data transformation/normalization
- dimensionality reduction
- data compression/generalization

Dimensionality Reduction

By axis rotation

- determine a more efficient basis
- Principal Component Analysis (PCA)
- Singular value decomposition (SVD)
- Latent semantic analysis (LSA)

By type transformation

- determine a more efficient data type
- Fourier analysis and Wavelets for grids
- Multidimensional scaling (MSD) for graphs
- Locally Linear Embedding
- Isomap
- Self Organizing Maps (SOM)
- Linear Discriminant Analysis (LDA)

Principal Component Analysis (PCA)

Covariance Matrix

Analytical: $Cov(X,Y) = E[(X - \mu_x)(Y - \mu_y)]$

Samples: $\sigma_{xy} = \mathrm{cov}_{xy} = \sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})$

An n-D dataset has *n* variables $x_1, x_2, \ldots x_n$

- define pairwise covariance among all of these variables
- construct a covariance matrix

$$\Sigma = \mathrm{Cov}(X) = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_{nn} \end{bmatrix}$$

Correlation

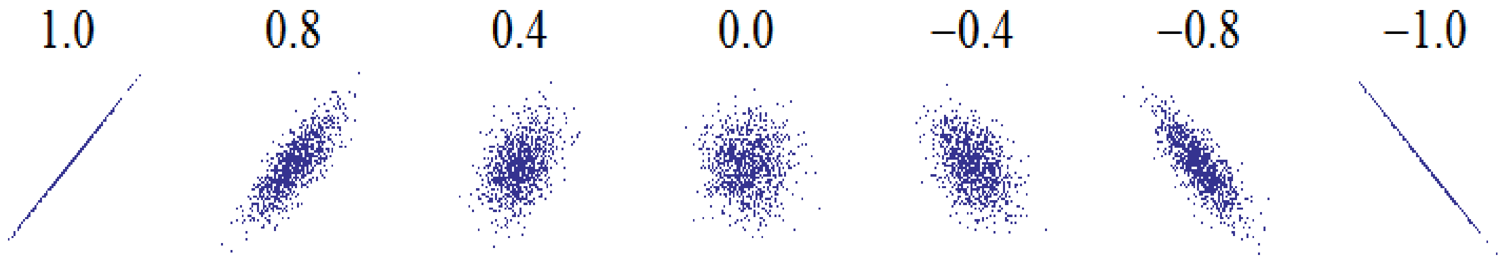
Pearson's correlation coefficient:

$$Corr(X,Y) = \frac{Cov(X,Y)}{\sigma_x \sigma_y} = \frac{E[(X-\mu_x)(Y-\mu_y)]}{\sigma_x \sigma_y}$$

Sample correlation (n observations):

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

Correlation rates between -1 and 1:

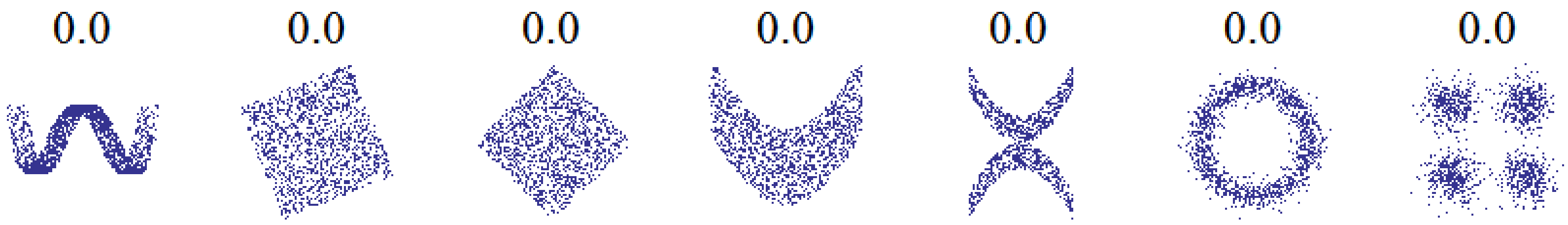


No Correlation

Correlation and regression are not reliable here

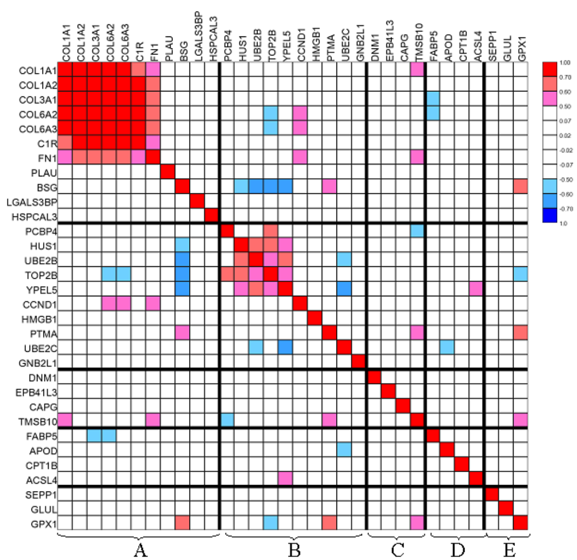
- defined for linear relationships
- visualization can help here

None of these point distributions have correlations:

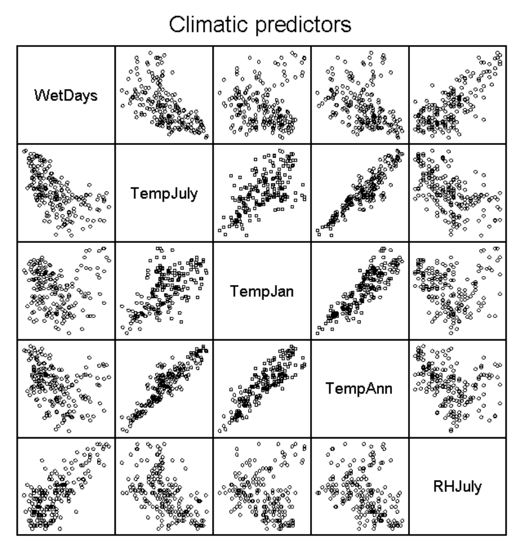


Correlation Matrix

	MO	FP	MP	IM	IC	FM	FE	FI	SPC	DSC	DST
MO	1.00										
FP	0.31[a]	1.00									
MP	0.32[a]	0.71[a]	1.00								
IM	0.36[a]	0.12[c]	0.14[c]	1.00							
IC	0.39[a]	0.18[b]	0.21[a]	0.62[a]	1.00						
FM	0.26[a]	0.21[a]	0.14[c]	0.30[a]	0.27[a]	1.00					
FE	0.47[a]	0.21[a]	0.18[b]	0.38[a]	0.28[a]	0.24[a]	1.00				
FI	0.53[a]	0.26[a]	0.22[a]	0.36[a]	0.37[a]	0.29[a]	0.47[a]	1.00			
SPC	0.32[a]	0.22[a]	0.31[a]	0.51[a]	0.47[a]	0.32[a]	0.37[a]	0.35[a]	1.00		
DSC	− 0.12[c]	0.03[c]	0.05[c]	0.17[b]	0.08[c]	0.18[b]	− 0.05[c]	0.06[c]	0.01[c]	1.00	
DST	− 0.02[c]	− 0.01[c]	0.05[c]	0.24[a]	0.14[c]	0.05[c]	− 0.05[c]	0.05[c]	0.05[c]	0.56[a]	1.00
DM	0.05[c]	0.144	0.136[c]	0.199[a]	0.169[b]	0.247[a]	0.08[c]	0.11[c]	0.14[c]	0.46[a]	0.71[a]



just value

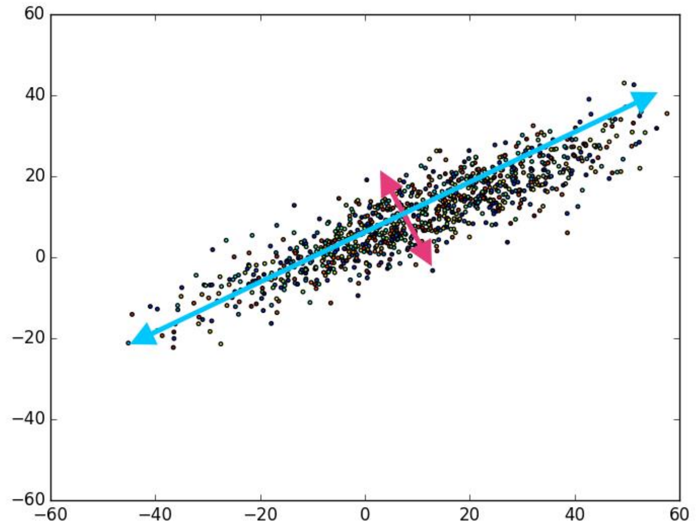


distribution (scatterplot matrix)

Principal Component Analysis

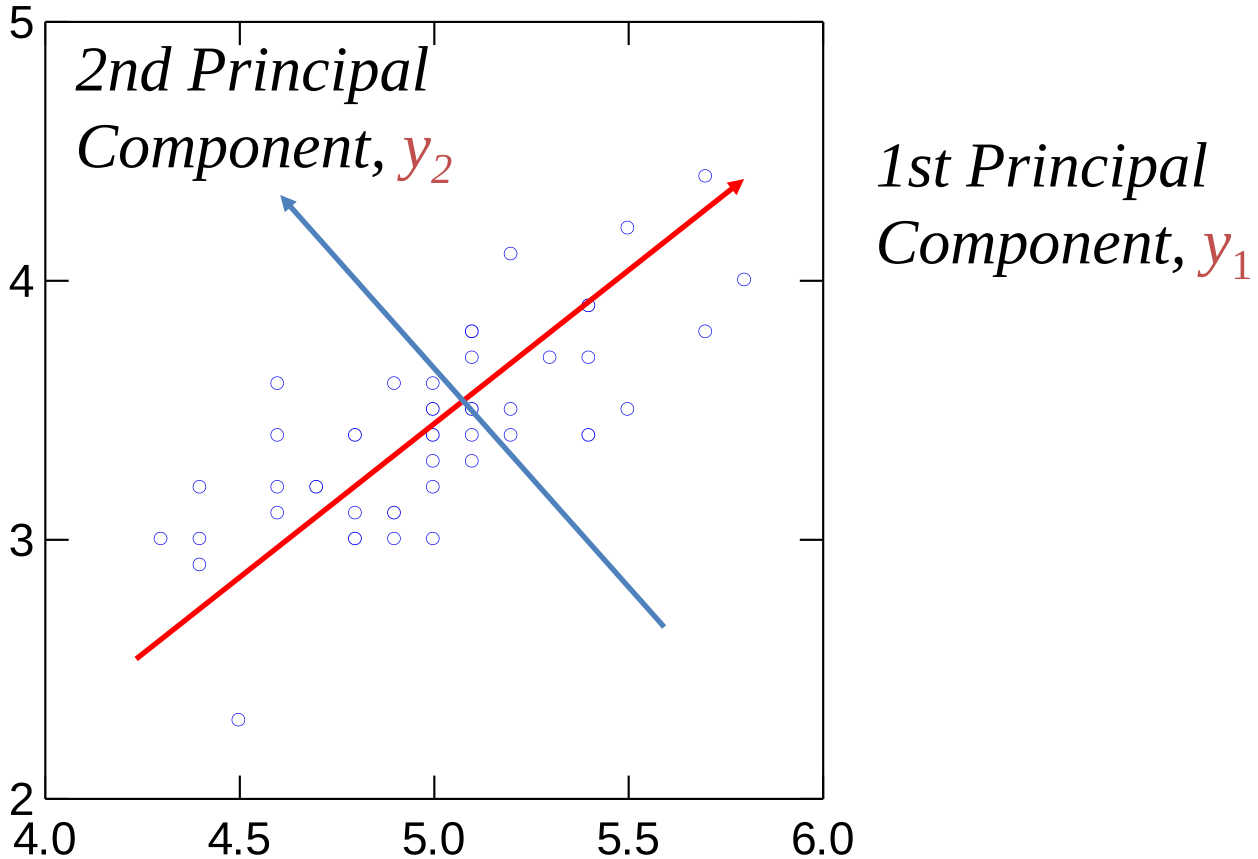
Ultimate goal:

- find a coordinate system that can represent the variance in the data with as few axes as possible



- rank these axes by the amount of variance (blue, red)
- drop the axes that have the least variance (red)

Principal Components



PCA – How To Do

Find the principal components by Eigen decomposition of

- covariance matrix Cov
- correlation matrix Corr
- lets call it C

- solve the Eigen value problem $(\mathbf{C} - \lambda_i \mathbf{I})\boldsymbol{x}_i = 0$

- do this via QR factorization or LU decomposition to get

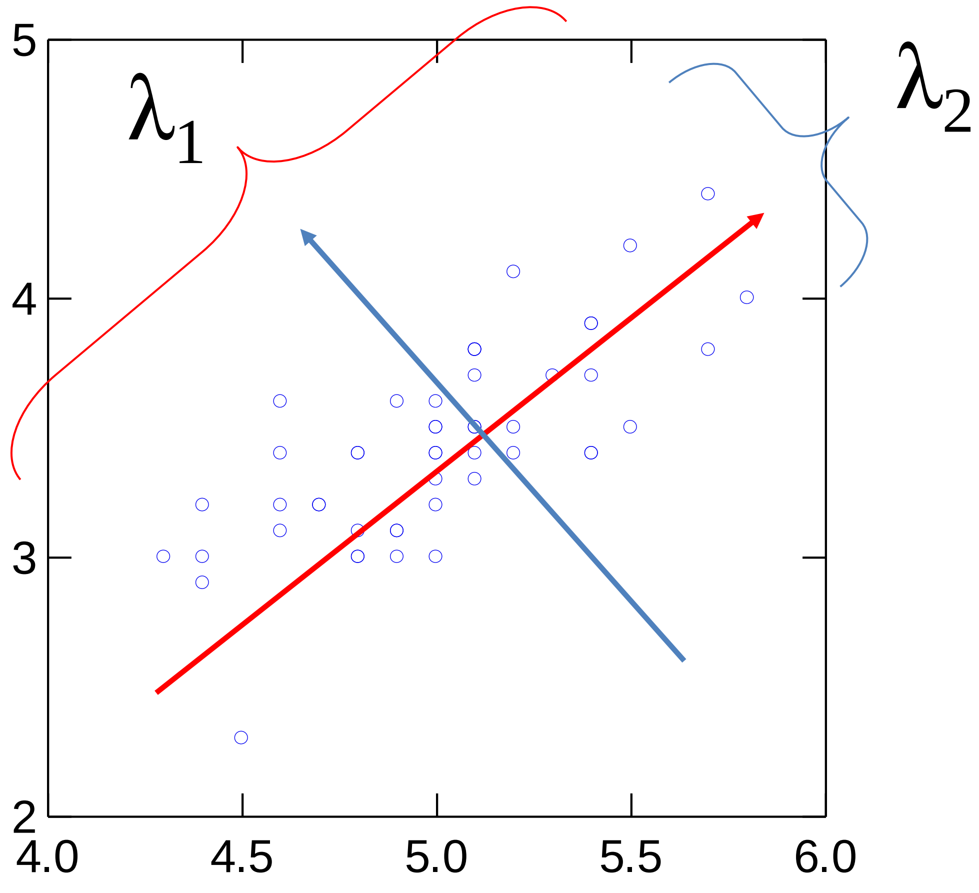
$$\boldsymbol{C} = Q \Lambda Q^{-1}$$

Q: matrix with Eigenvectors

Λ: diagonal matrix with Eigenvalues λ

- now order the Eigenvectors in terms of their Eigenvalues λ

Eigenvectors and values



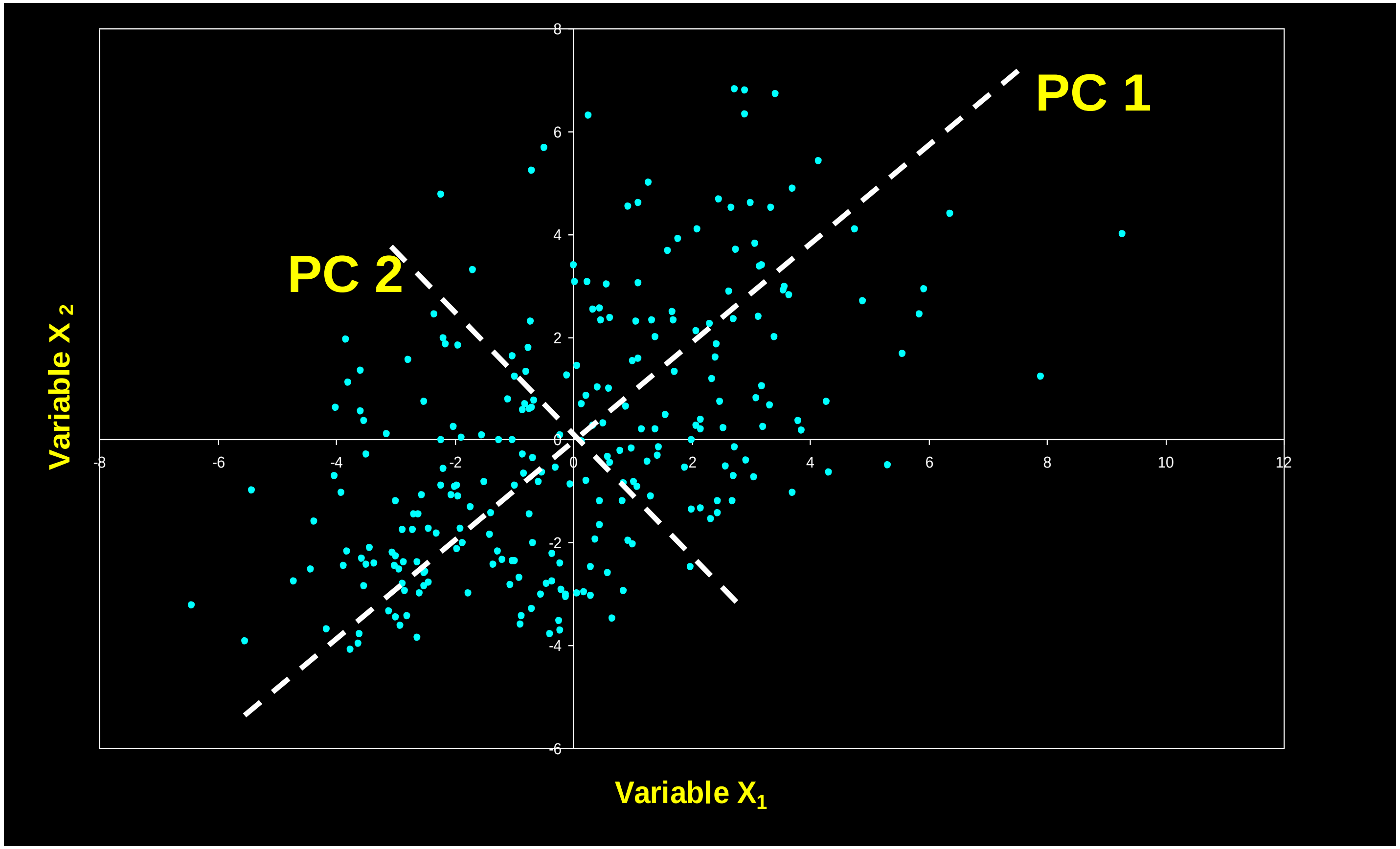
Covariance vs. Correlation

When to use what?

- use the covariance matrix when the variable scales are similar
- use the correlation matrix when the variables are on different scales
- the correlation matrix *standardizes* the data
- in general they give different results, especially when the scales are different

Example

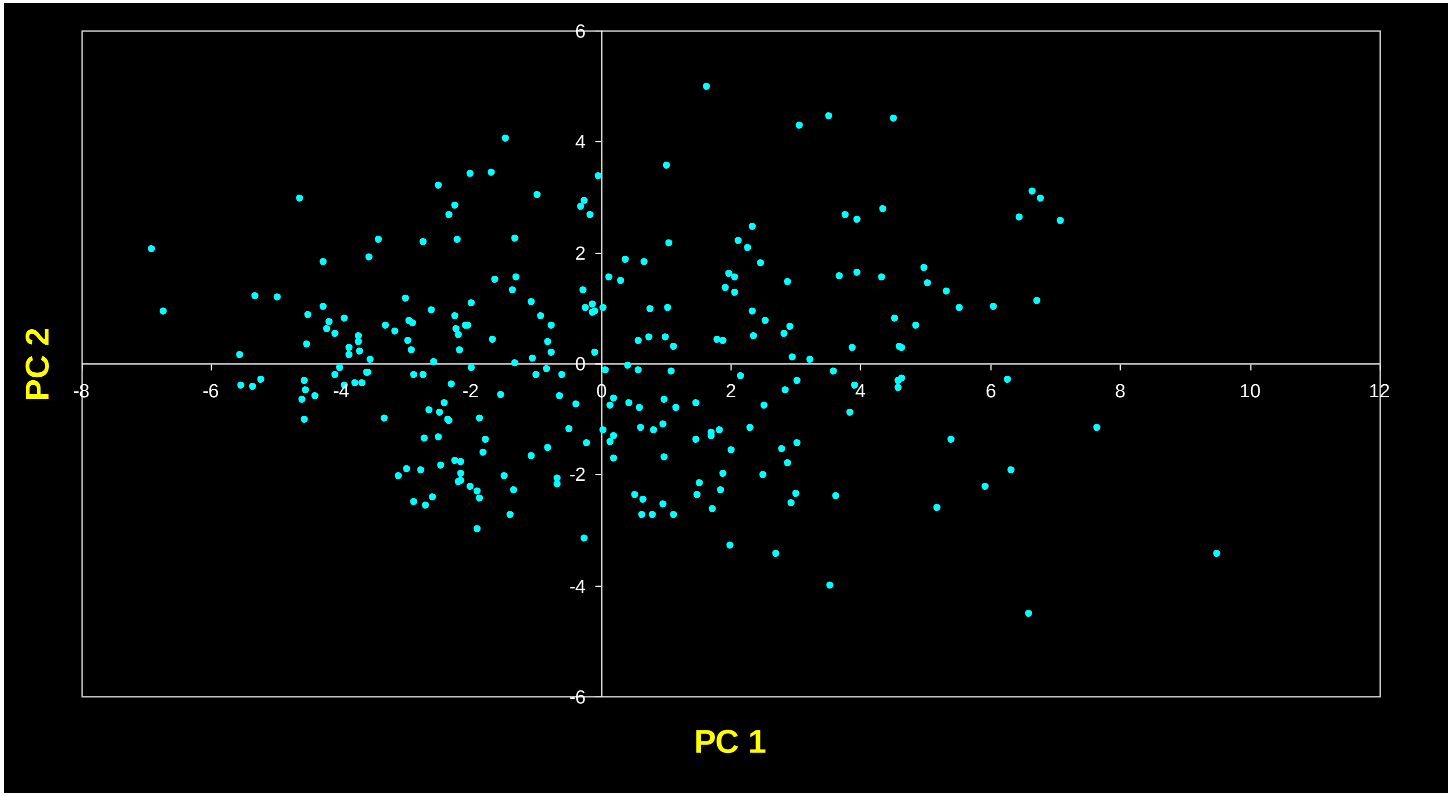
Before PCA



Example

$\lambda_1 = 9.8783$ $\lambda_2 = 3.0308$ Trace = 12.9091

- PC 1 displays ("explains") 9.8783/12.9091 = 76.5% of total variance

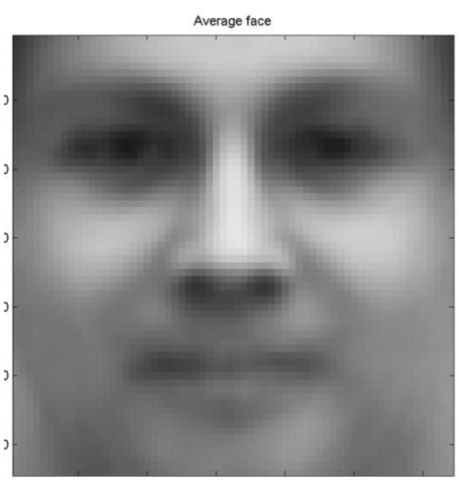


PCA Applied To Faces

Some familiar faces…

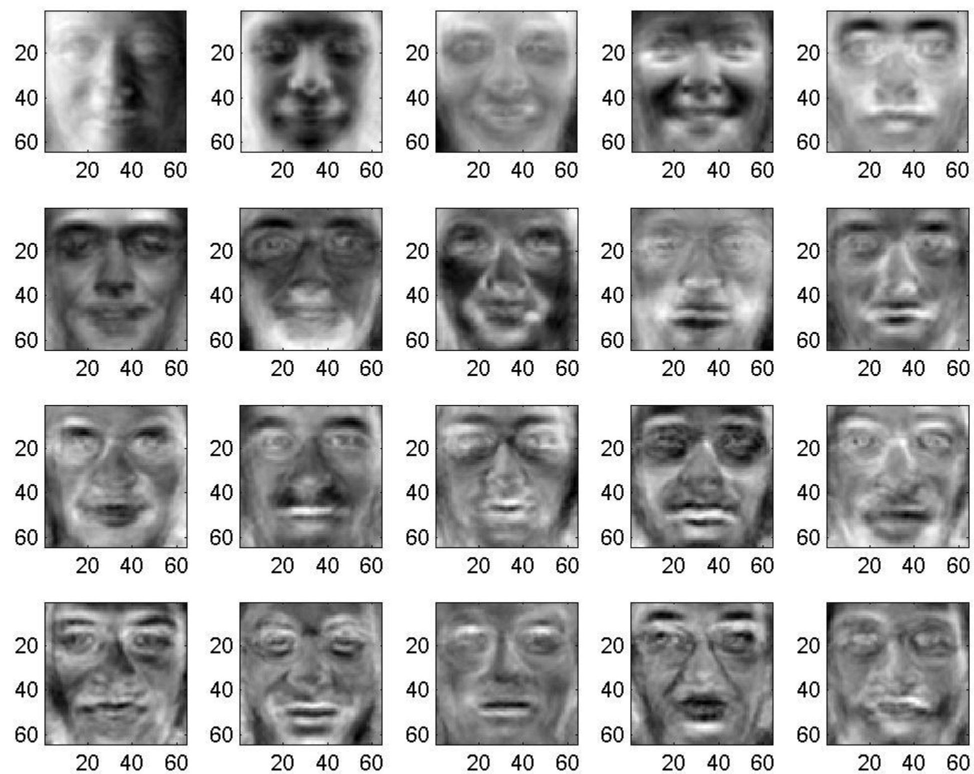
PCA Applied To Faces

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



Average Face

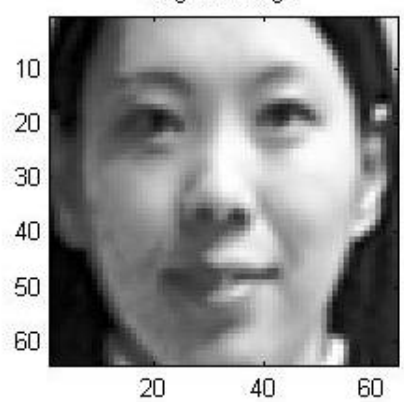
+

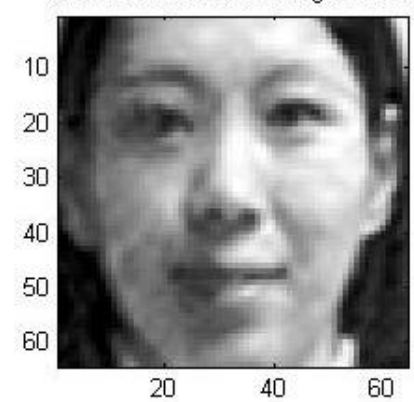


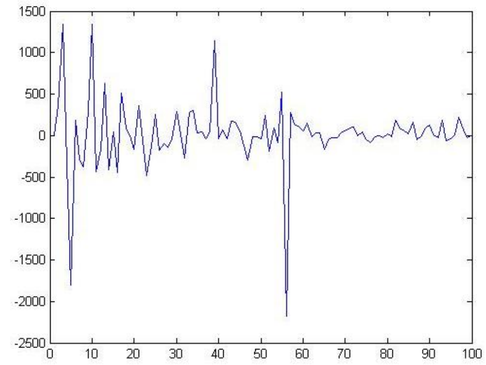
Eigenfaces

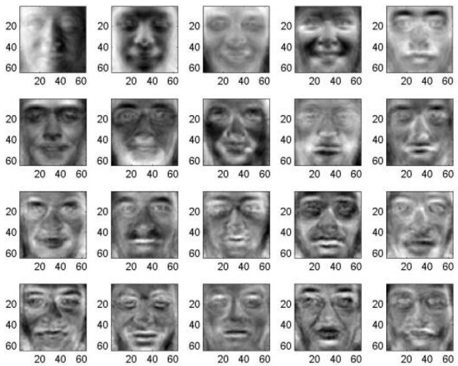
Reconstruction Using PCA

- 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









Singular Value Decomposition (SVD)

The same as PCA when the mean of each attribute is zero

SVD does not subtract the mean

- appropriate if values close to zero should not be influential
- PCA puts them at in the extreme negative side

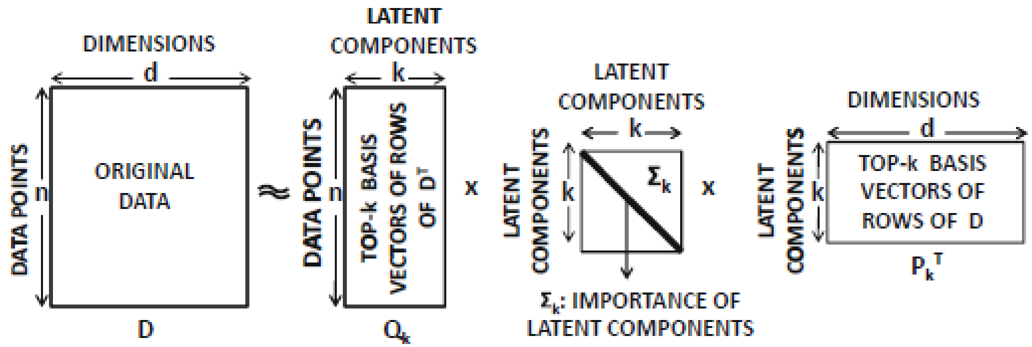
SVD often used for text analysis

- values close to zero are frequent and should not affect the analysis

Singular Value Decomposition (SVD)

Decomposes *C* into the matrix:

$$Q_k \Sigma_k P_k^T$$



q_i and p_i are two column vectors with significance σ_i

$$Q_k \Sigma_k P_k^T = \sum_{i=1}^{k} \overline{q_i} \sigma_i \overline{p_i}^T = \sum_{i=1}^{k} \sigma_i (\overline{q_i}\ \overline{p_i}^T)$$

Example: in a user-item ratings matrix we wish to determine:

- a reduced representation of the users
- a reduced representation of the items
- *SVD* has the basis vectors for both of these reductions

Latent Semantic Analysis

Create an occurrence matrix (term-document matrix)

- words (terms *t*) are the rows
- paragraphs (documents *d*) are the columns
- uses the *term frequency–inverse document frequency (tf-idf)* metric
- *tf(t,d)* = simplest form is frequency of *t* in *d* = *f(t,d)*
- idf(*t*,*d*) $\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$
- *N* = number of docs = |*D*| , *D* is the corpus of documents
- idf is a measure of term rareness, it's 0 when term occurs in all of *D*
- important terms get a higher tf-idf

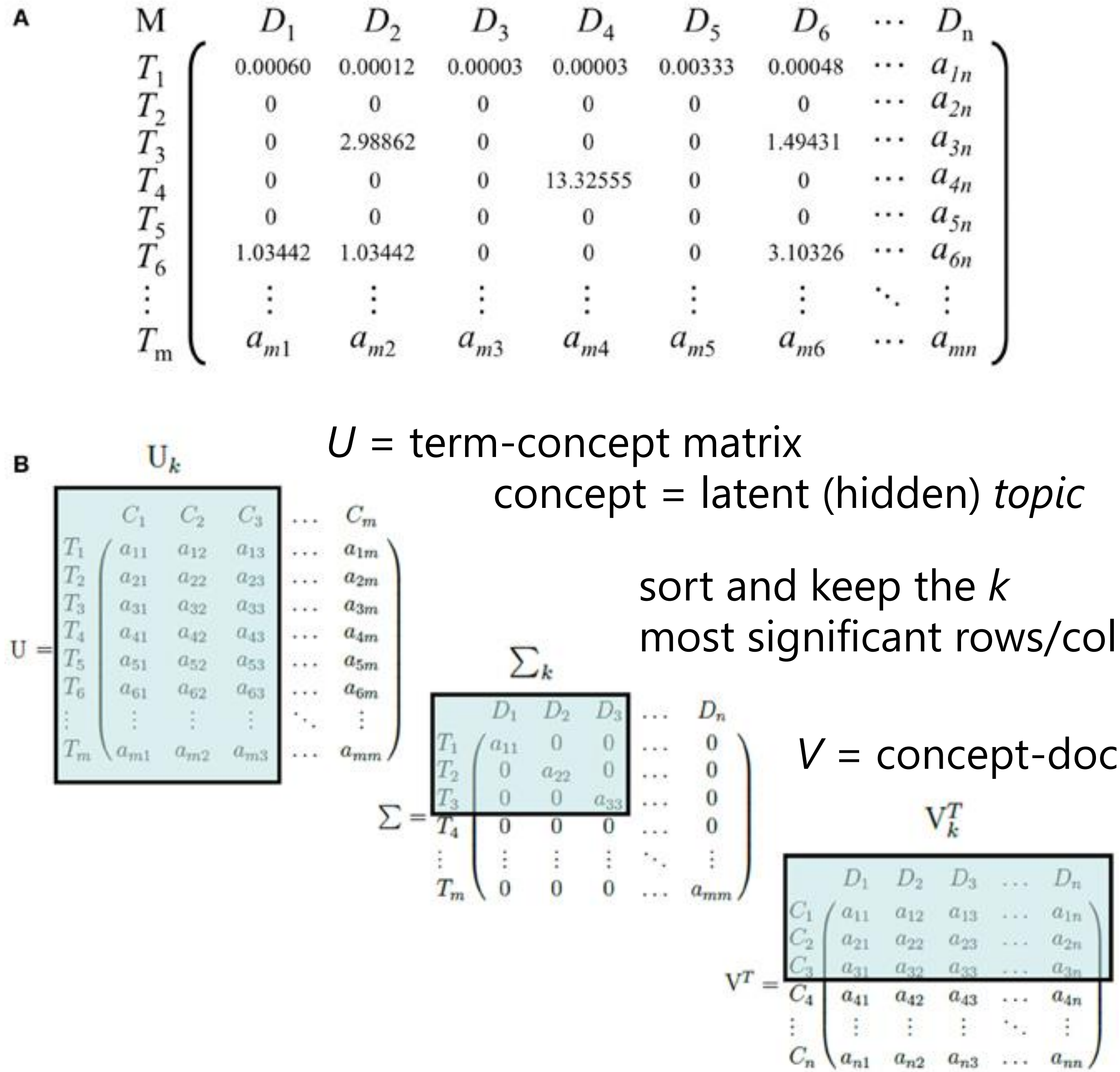
Use SVD to reduce the number of rows

- preserves similarity of columns

Co-Occurrence TF-IDT matrix

A

$$
\begin{array}{cc}
\mathrm{M} & \begin{array}{cccccccc} D_1 & D_2 & D_3 & D_4 & D_5 & D_6 & \cdots & D_{\mathrm{n}} \end{array} \\
\begin{array}{c} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ \vdots \\ T_{\mathrm{m}} \end{array} &
\begin{pmatrix}
0.00060 & 0.00012 & 0.00003 & 0.00003 & 0.00333 & 0.00048 & \cdots & a_{1n} \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & a_{2n} \\
0 & 2.98862 & 0 & 0 & 0 & 1.49431 & \cdots & a_{3n} \\
0 & 0 & 0 & 13.32555 & 0 & 0 & \cdots & a_{4n} \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & a_{5n} \\
1.03442 & 1.03442 & 0 & 0 & 0 & 3.10326 & \cdots & a_{6n} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
a_{m1} & a_{m2} & a_{m3} & a_{m4} & a_{m5} & a_{m6} & \cdots & a_{mn}
\end{pmatrix}
\end{array}
$$



U = term-concept matrix

concept = latent (hidden) *topic*

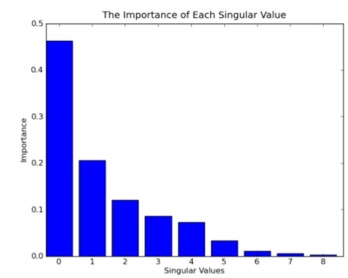
sort and keep the *k* most significant rows/columns

V = concept-document matrix

Visualizing The Concept Space

How many concepts to use when approximating the matrix?

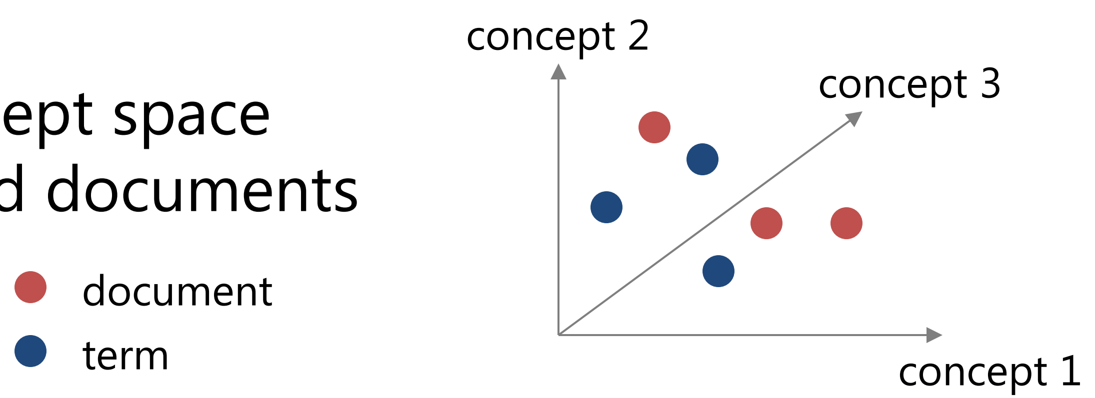
- if too few, important patterns are left out
- if too many, noise caused by random word choices will creep in
- can use the elbow method in the scree plot



Throw out the 1st dimension in U and V

- in U it is correlates with document length
- in V it correlates with the number of times a term was mentioned

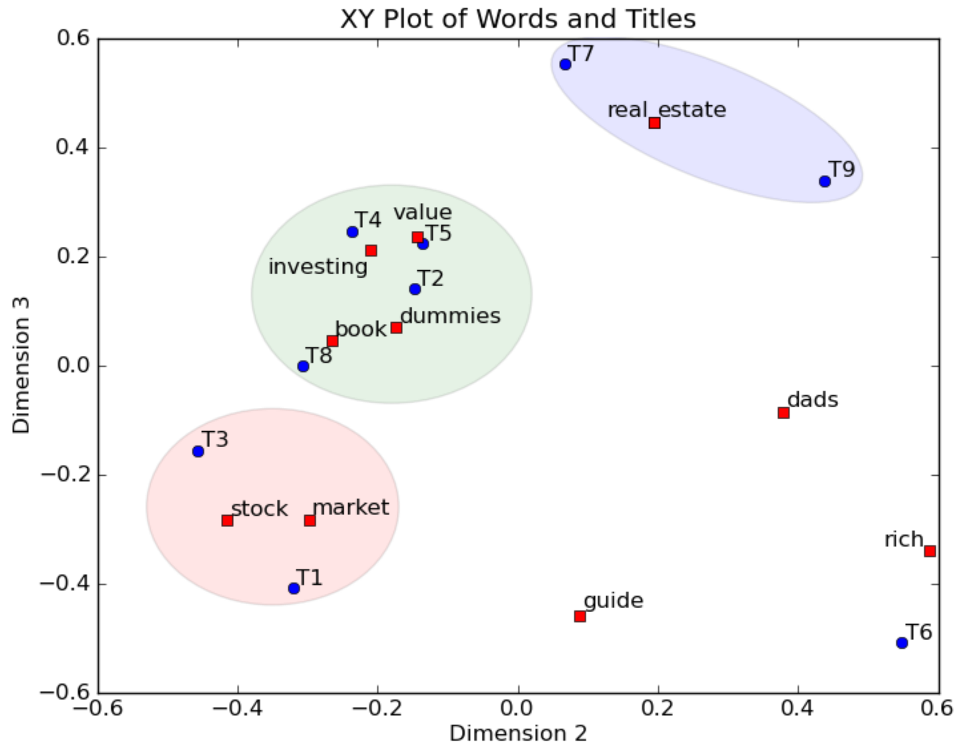
Now we have a k-D concept space shared by both terms and documents



Visualizing The Concept Space

Project the k-D concept space into 2D and visualize as a map

- can cluster the map
- the cluster of documents are then labeled by the terms
- provides map semantics



LSA Disadvantages

LSA assumes a Gaussian distribution and Frobenius norm

- this may not fit all problems

LSA cannot handle polysemy effectively

- need LDA (Latent Dirichlet Allocation) for this

LSA depends heavily on SVD

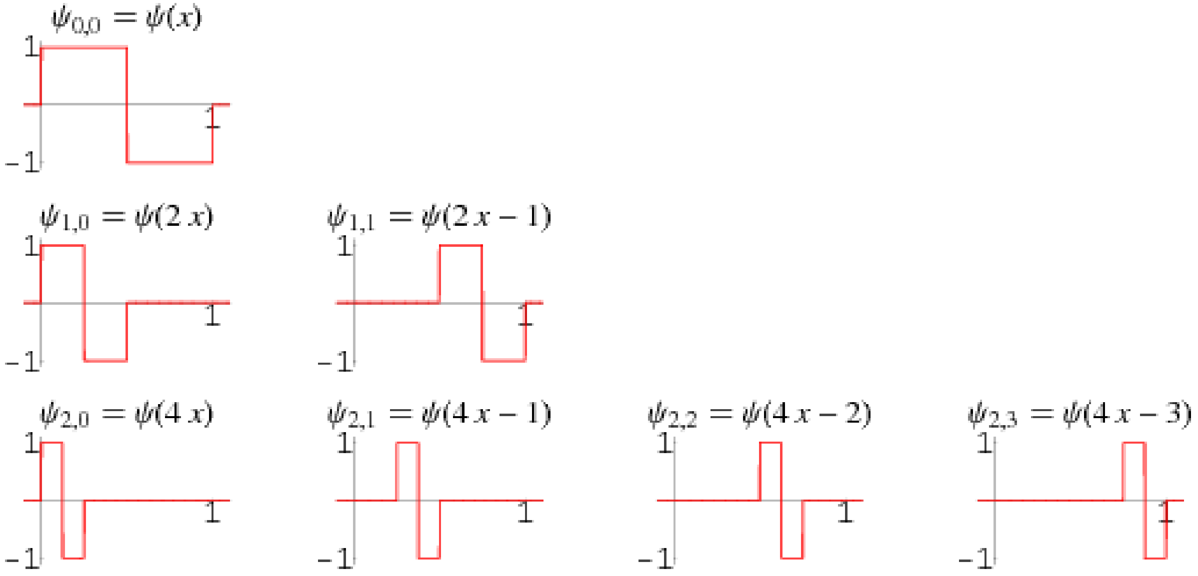
- computationally intensive
- hard to update as new documents appear
- but faster algorithms have emerged recently

Type Transformations

Haar Wavelets

A sequence of multi-scale square-shaped functions

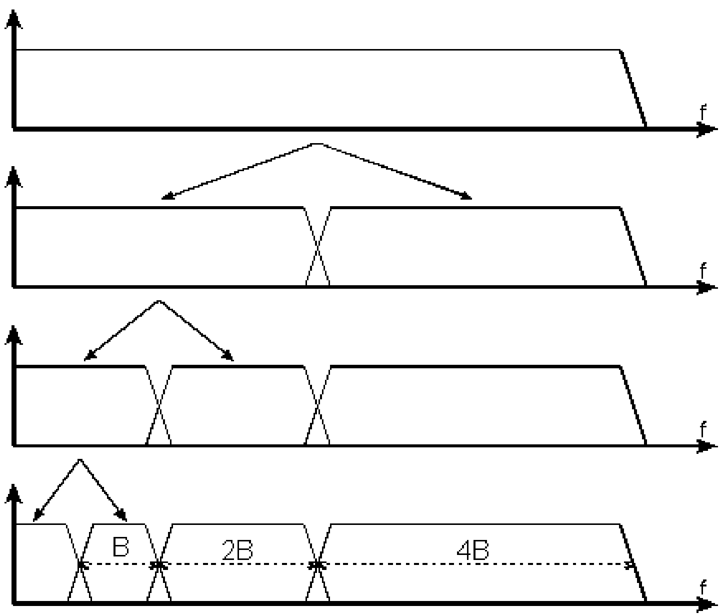
- together they form a wavelet family or basis
- each has half the size than the one before

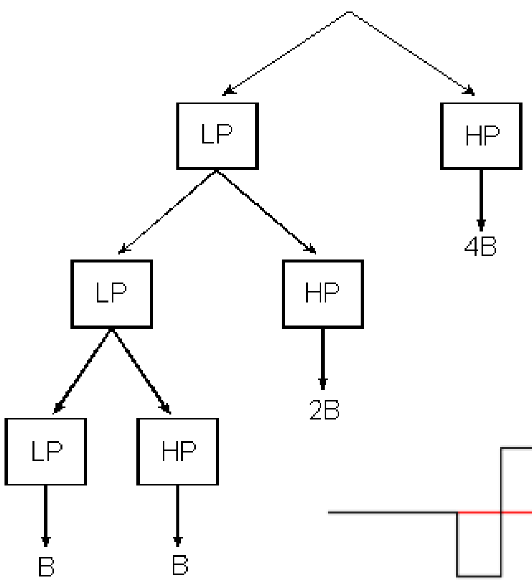


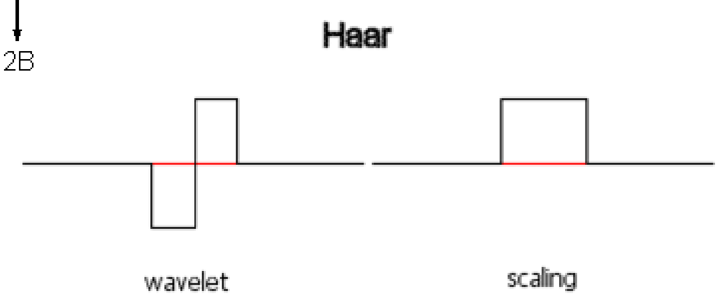
Discrete Wavelet Transform

Two basis function each level of scale

- wavelet = extract the detail at that level (HP)
- scaling = remove the detail and return what's left for the next level (LP)



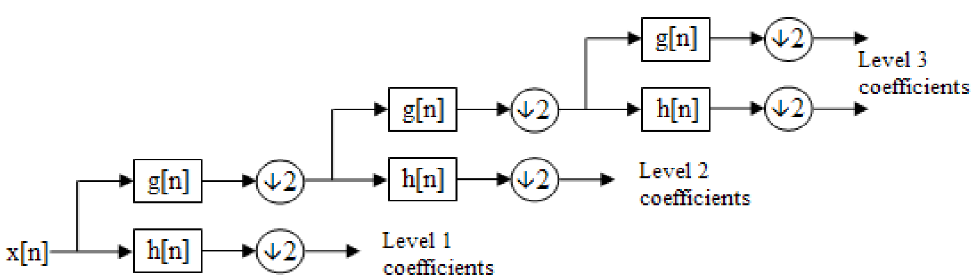


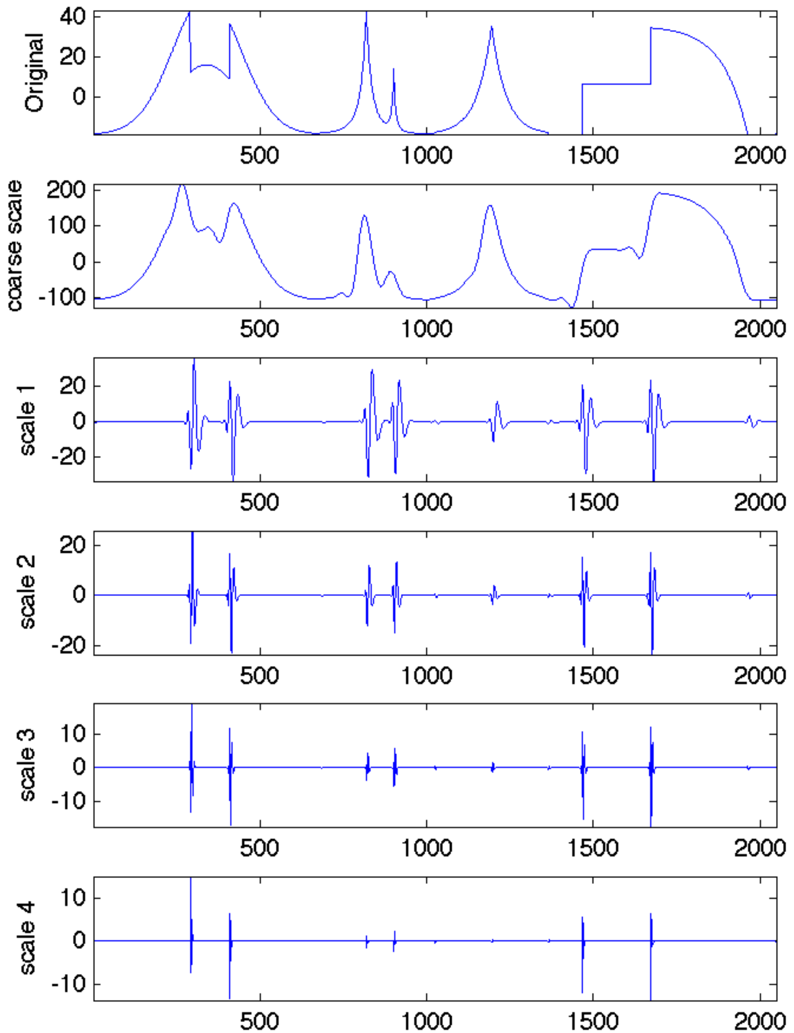


Wavelet Compression

Goal

- decompose the signal into wavelet coefficients
- eliminate the coefficients with magnitude < threshold
- keep the others
- the higher the threshold the more the compression





Works in Higher Dimensions

(a) (b)

2D case

$$output^{(1)} = \begin{array}{|c|c|}\hline LL^{(1)} & LH^{(1)} \\ \hline HL^{(1)} & HH^{(1)} \\ \hline\end{array}$$

Multidimensional Scaling (MDS)

Wavelets are for regular grids

MDS is for irregular structures

- 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

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

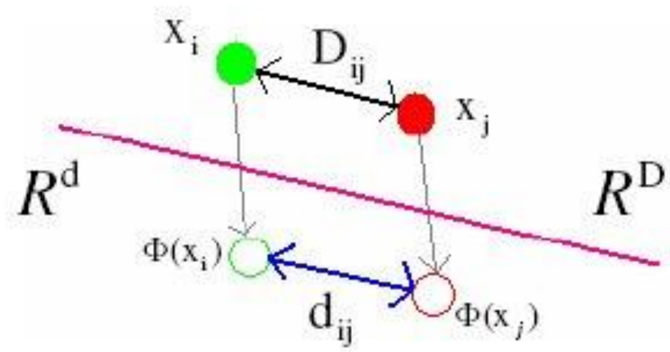
	Chicago	Raleigh	Boston	Seattle	S.F.	Austin	Orlando
Chicago	0						
Raleigh	641	0					
Boston	851	608	0				
Seattle	1733	2363	2488	0			
S.F.	1855	2406	2696	684	0		
Austin	972	1167	1691	1764	1495	0	
Orlando	994	520	1105	2565	2458	1015	0

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: $D_{ij} = \| x_i - x_j \|_D \qquad d_{ij} = \| y_i - y_j \|_d$
  - Claim: $D_{ij} \equiv d_{ij} \qquad \forall i, j \in [1, n]$



- 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:

MDS Algorithm

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

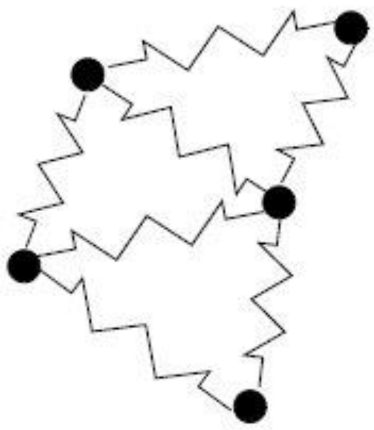
- iterative procedure to find a good configuration of image points
  - 1) Initialization
    → Begin with some (arbitrary) initial configuration
  - 2) Alter the image points and try to find a configuration of points that minimizes the following sum-of-squares error function:

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

Force-Directed Algorithm

Spring-like system

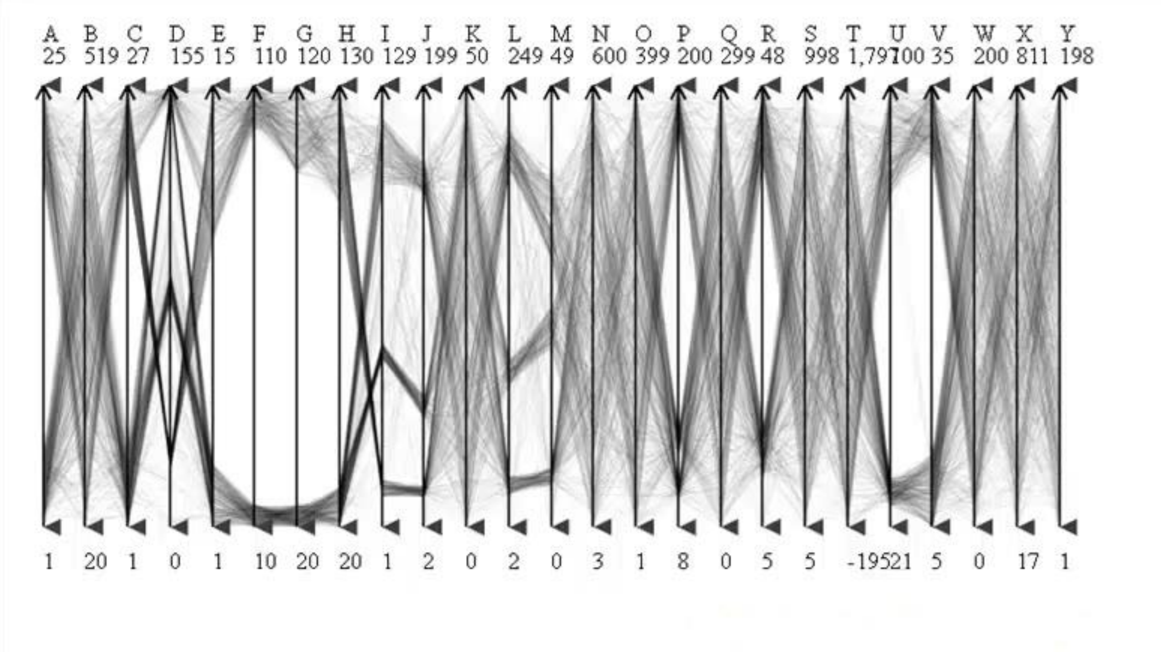
- insert springs within each node
- the length of the spring encodes the desired node distance
- start at an initial configuration
- iteratively move nodes until an energy minimum is reached

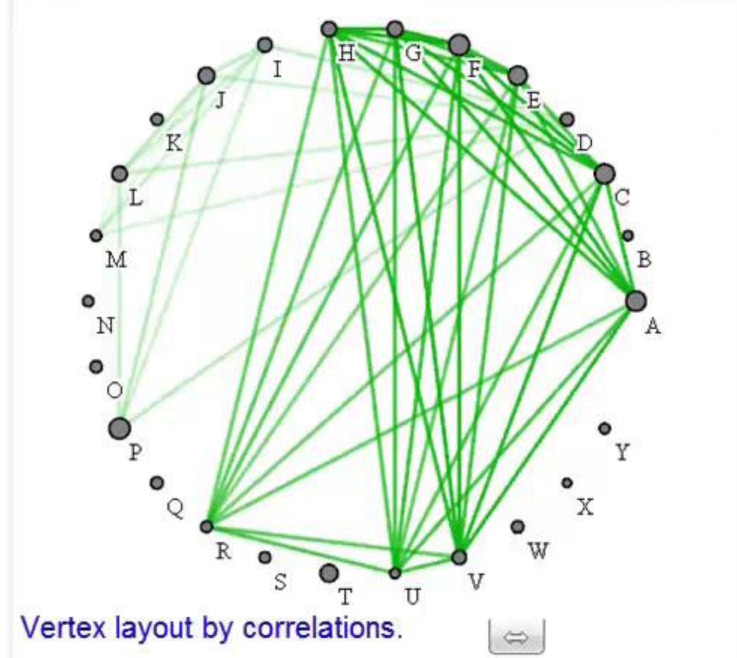


Force-Directed Algorithm

Spring-like system

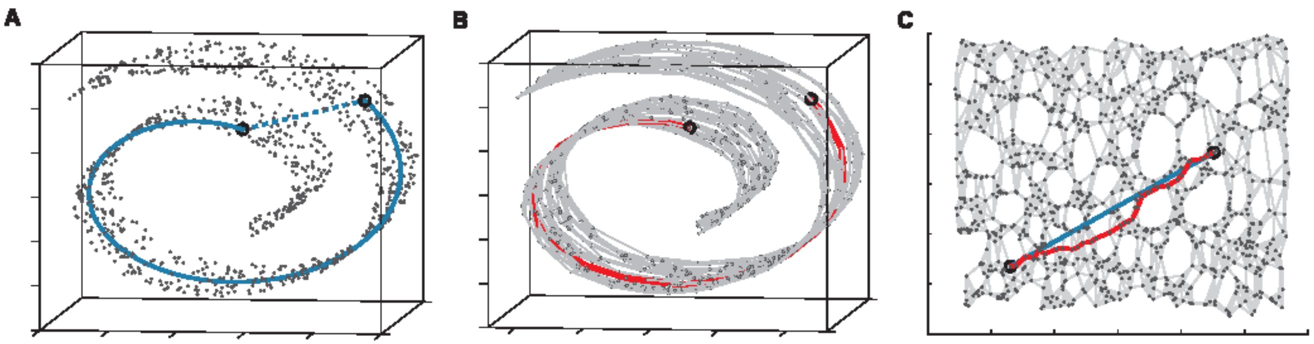
- insert springs within each node
- the length of the spring encodes the desired node distance
- start at an initial configuration
- iteratively move nodes until an energy minimum is reached





Manifold Learning: Isomap

by: J. Tenenbaum, V. de Silva, J. Langford, Science, 2000



Tries to unwrap a high-dimensional surface (A) → manifold

- noisy points could be averaged first and projected onto the manifold

Algorithm

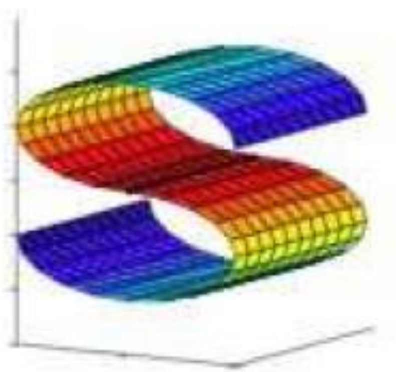
- construct neighborhood graph G → (B)
- for each pair of points in G compute the shortest path distances → geodesic distances
- fill similarity matrix with these geodesic distances
- embed (layout) in low-D (2D) with MDS → (C)

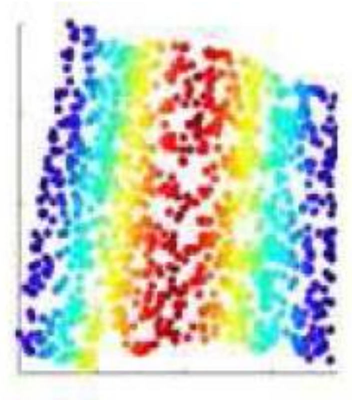
Manifold Learning: Locally Linear Embedding (LLE)

by: S. Roweis, L. Saul, Science, 2000

Based on simple geometric intuitions.

- suppose the data consist of N real-valued vectors X_i, each of dimensionality D
- each data point and its neighbors are expected to lie on or close to a locally linear patch of the manifold

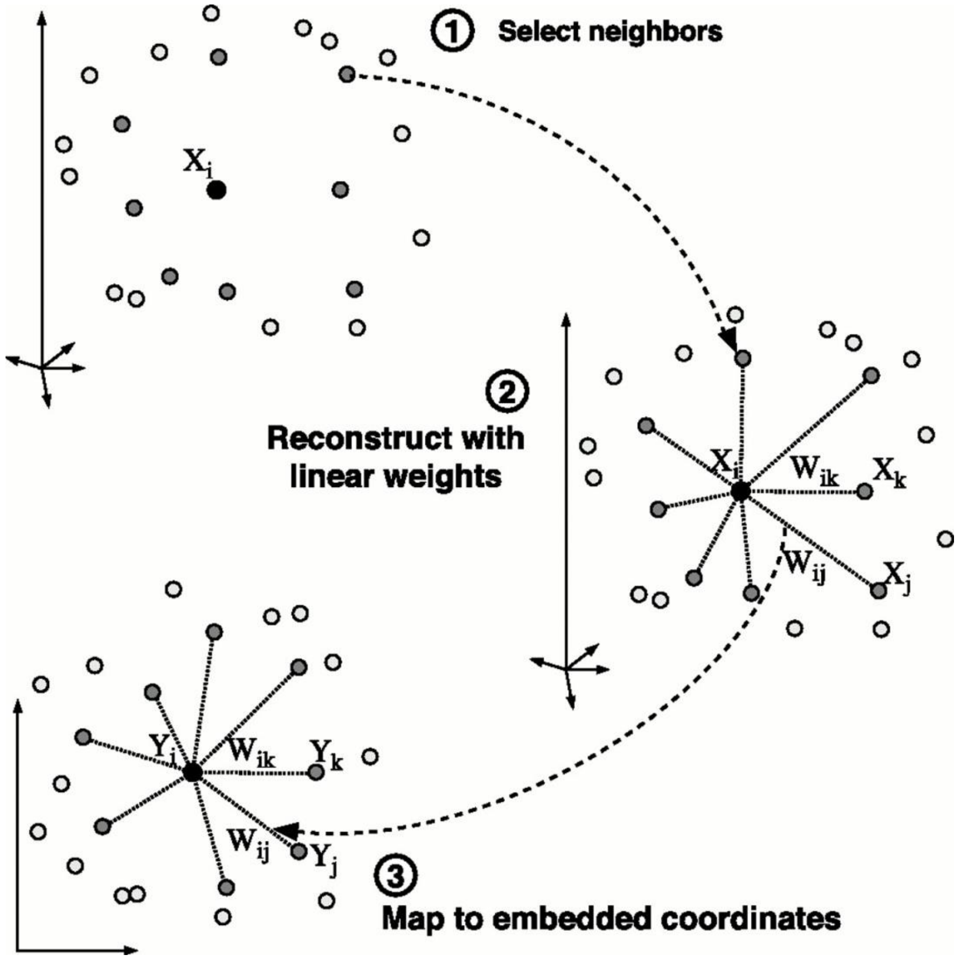




High dimensional Manifold

Low dimensional Manifold

LLE Overview



LLE Details

Steps:

- assign K neighbors to each data point $\vec{X}_i$
- compute the weights W_{ij} that best linearly reconstruct the data point from its K neighbors, solving the constrained least-squares problem

$$\acute{\varepsilon}(W) = \sum_i \left| \vec{X}_i - \sum_j W_{ij} \vec{X}_j \right|^2$$

- compute the low-dimensional embedding vectors $\vec{Y}_i$ best reconstructed by W_{ij}

$$\Phi(Y) = \sum_i \left| \vec{Y} - \sum_j W_{ij} \vec{Y}_j \right|^2$$

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

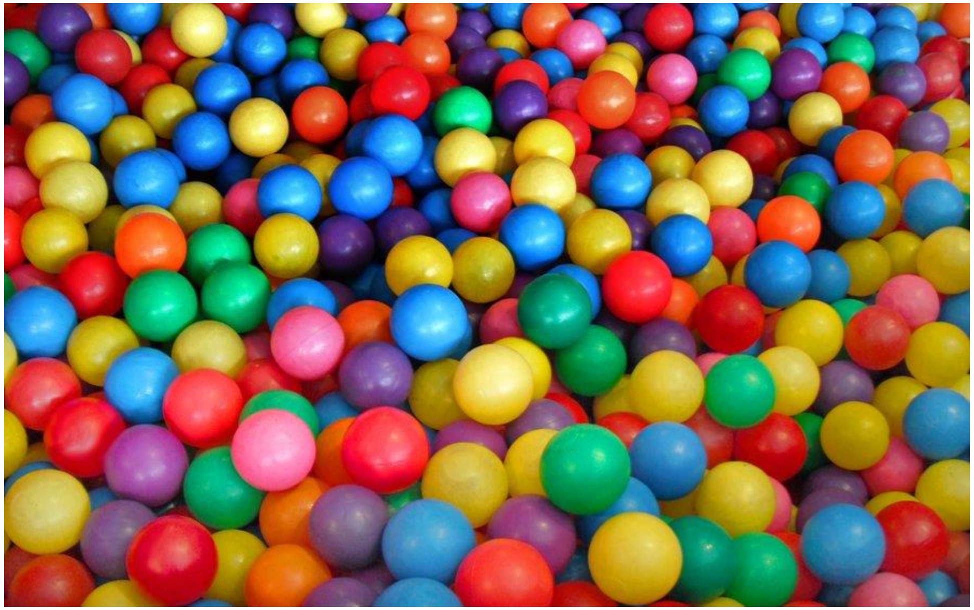
SOMs group the data

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

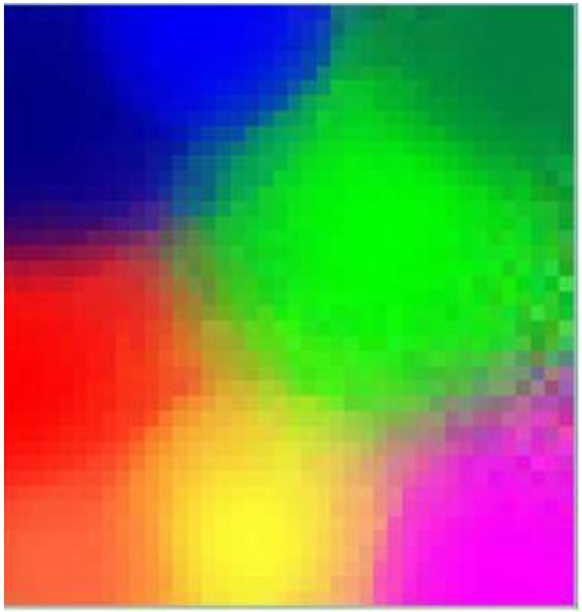
SOM Example

Map a dataset of 3D color vectors into a 2D plane

- assume you have an image with 5 colors
- want to see how many there are of each
- compute a SOM of the color vectors



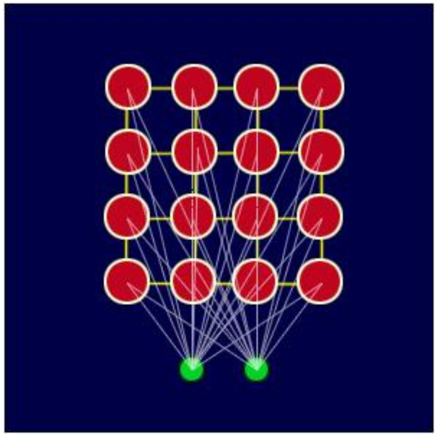
SOM



SOM Algorithm

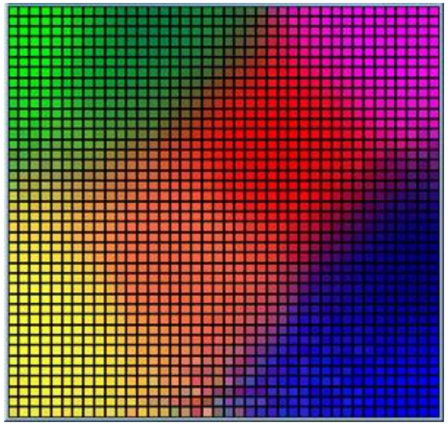
Create array and connect all elements to the N input vector dimensions

- shown here: 2D vector with 4×4 elements
- initialize weights

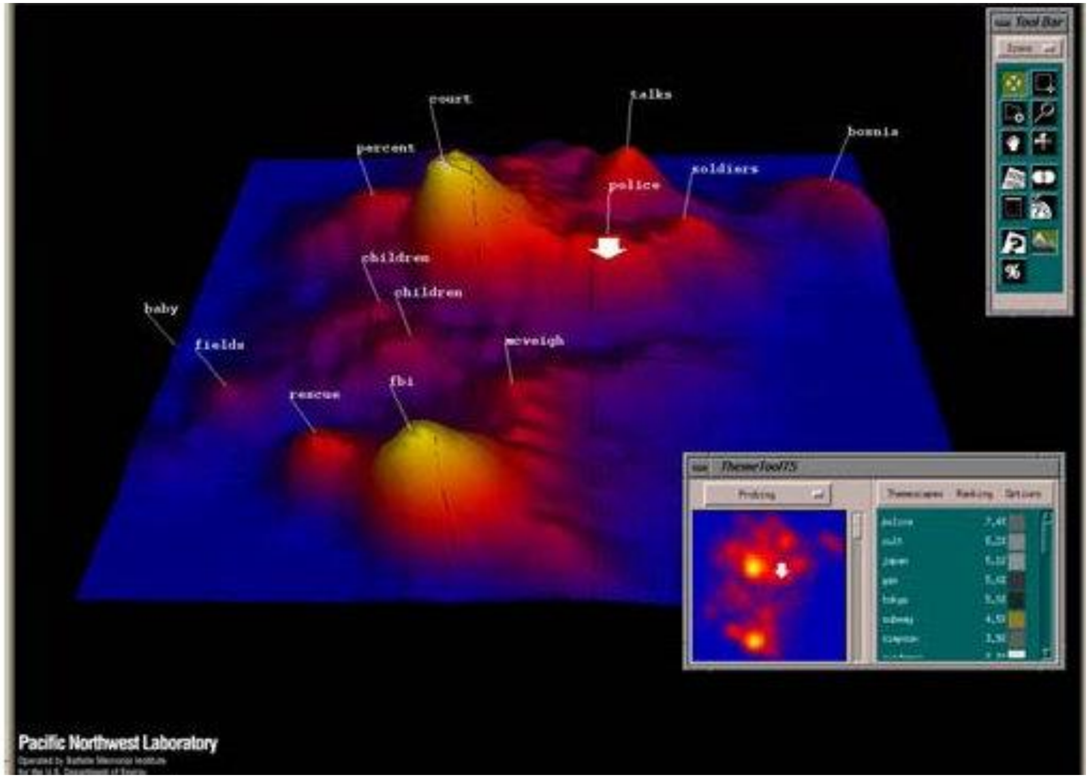


For each input vector chosen at random

- find node with weights most like the input vector
- call that node the Best Matching Unit (BMU)
- find nodes within neighborhood radius *r* of BMU
  - initially *r* is chosen as the radius of the lattice
  - diminishes at each time step
- adjust the weights of the neighboring nodes to make them more like the input vector
  - the closer a node is to the BMU, the more its weights get altered



SOM Example: ThemeScape



Height represents density or number of documents in the region

Invented at Pacific Northwest National Lab (PNNL)

SOM / MDS Example: VxInsight (Sandia)

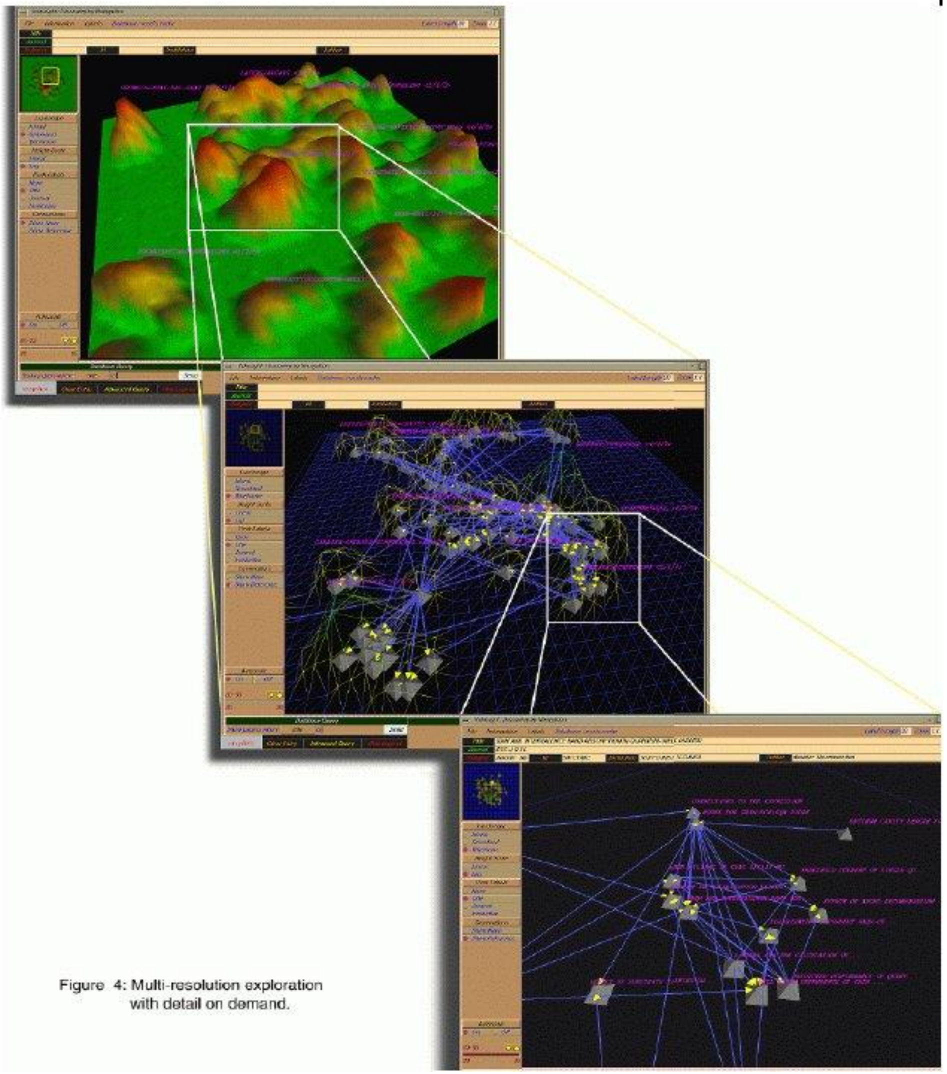


Figure 4: Multi-resolution exploration with detail on demand.

Linear Discriminant Analysis (LDA)

LDA was proposed by Ronald Fisher in 1936

See separate slides

- by Ricardo Gutierrez-Osuna (Texas A&M University)