Visualization

Data Analysis and Transformations

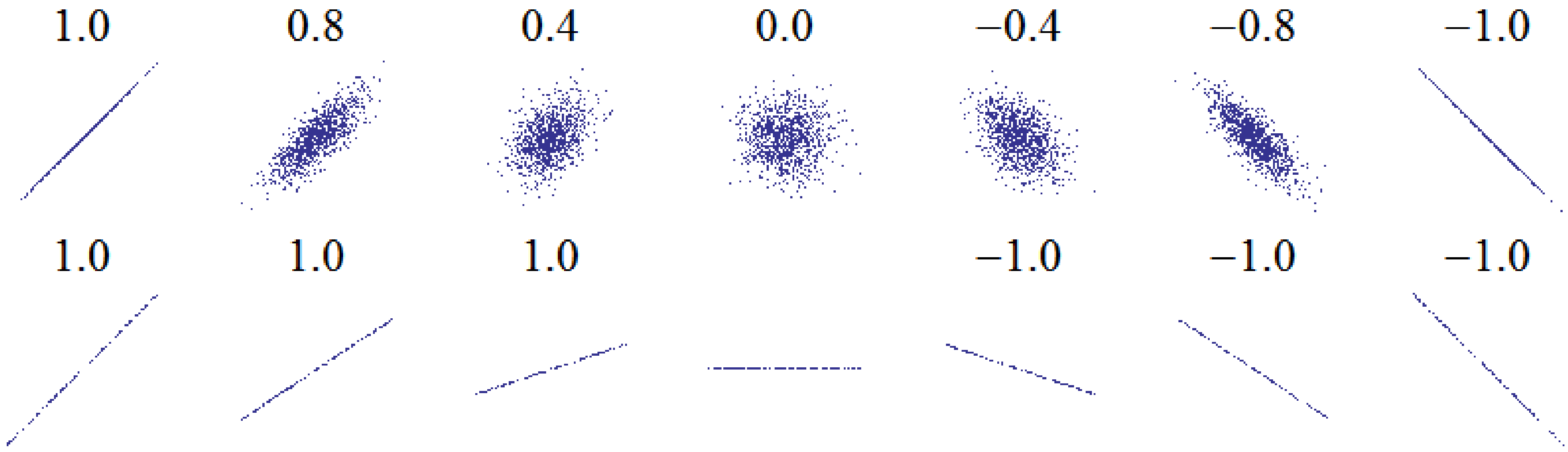
Klaus Mueller

Computer Science Department

Stony Brook University

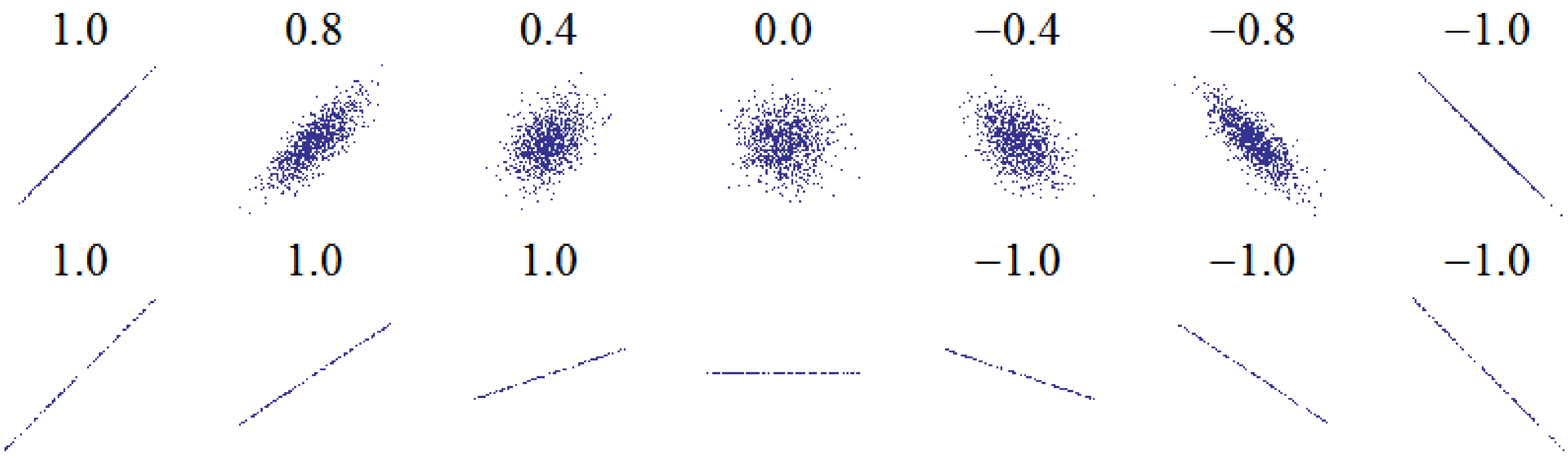
Relationships

What do these different measures show?



Relationships

What do these different measures show?



Top: correlation

- noisiness, direction, strength of relationship

Bottom: regression

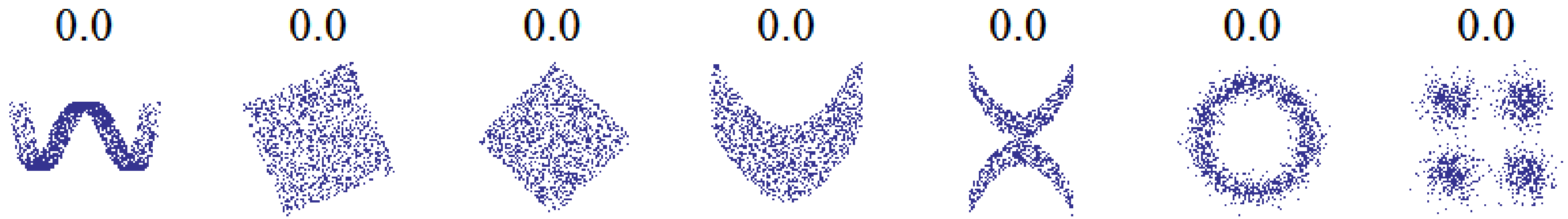
- slope, trend of relationship

These are complementary measures

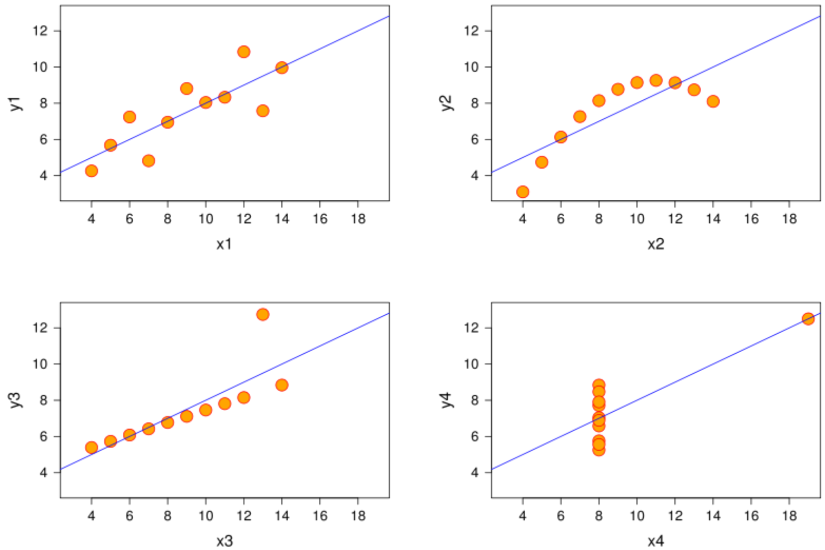
Linear vs. Non-Linear Relationships

Correlation and regression are not reliable here

- defined for linear relationships
- visualization can help here



- same goes for *outliers*
- recall Anscombe’s quartet



Correlation

Pearson's correlation coefficient

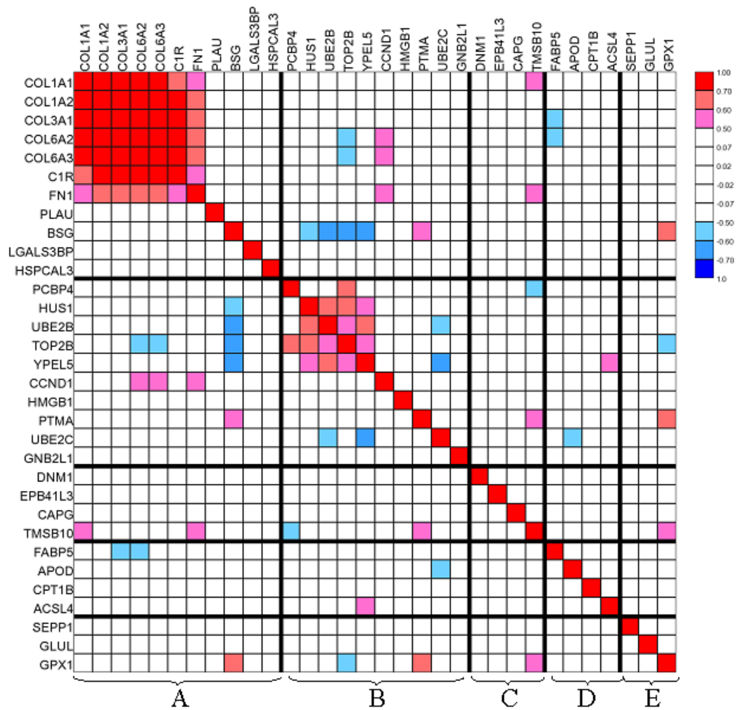
$$Corr(X,Y) = \frac{Cov(X,Y)}{\sigma_x \sigma_y} = \frac{E[(X-\mu_x)(Y-\mu_y)]}{\sigma_x \sigma_y}$$

Sample correlation (assume n observations):

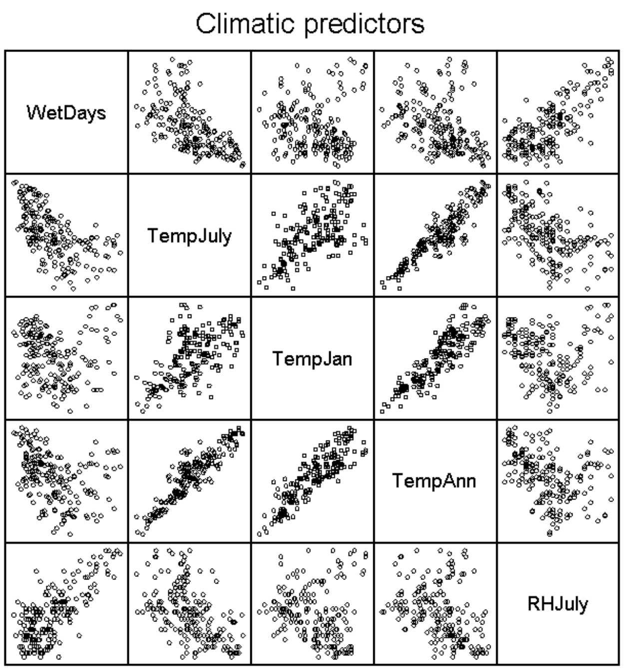
$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{x})^2}}$$

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)

Regression

Helps to understand how a dependent variable changes when any one of the independent variables is varied

- can be used for prediction and forecasting

Assumptions

- the errors are random and normally distributed,
- with mean = zero, and
- constant variance σ^2, independent and uniform
- the errors are independent of one another

Output:

- regression model : $y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i + \ldots + \varepsilon_i$
- get the coefficients by solving the least squares problem:

$$\frac{\partial}{\partial \beta} \sum_i (y_i - (\beta_0 + \beta_1 x_i + \beta_2 x_i \ldots))^2 = 0$$

- gives rise to a set of *normal equations* (one for each coefficient)

Goodness of Fit

Total sum of squares: $SST = \sum_{i=1}^{n} (Y_i - \overline{Y})^2 \qquad df_T = n - 1$

Regression sum of squares: $SSR = \sum_{i=1}^{n} (\hat{Y}_i - \overline{Y})^2 \qquad df_R = 1$

Error sum of squares: $SSE = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 \qquad df_E = n - 2$

Coefficient of determination: $r^2 = \frac{\text{explained variation}}{\text{total variation}} = \frac{SSR}{SST} \qquad 0 \le r^2 \le 1$

Coefficient r^2:

- proportion of variation in Y "explained" by the regression on X

There is much more on this

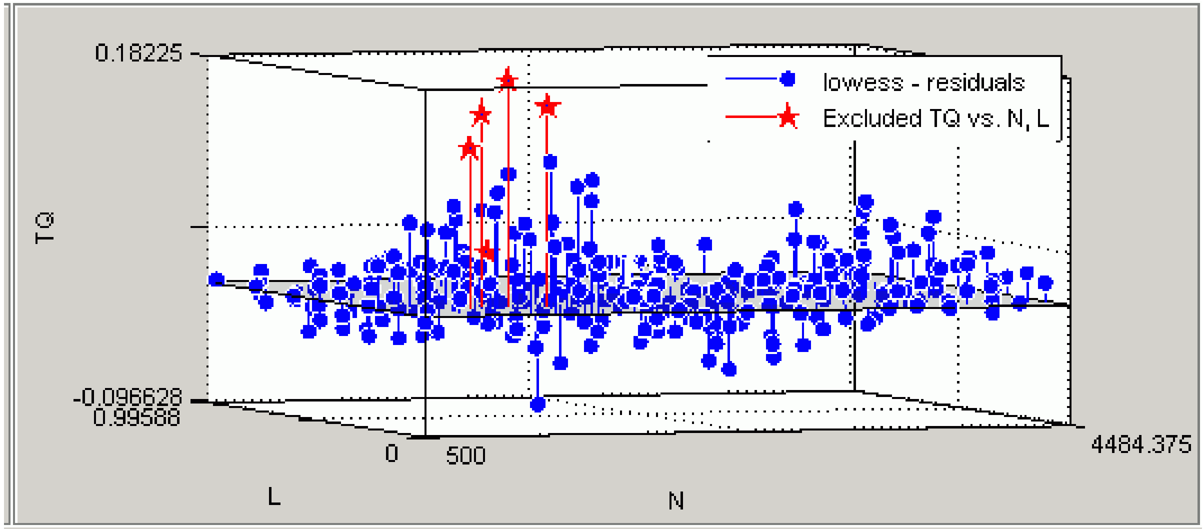
- confidence analysis, sensitivity analysis, F-test, ANOVA
- multivariate statistics → generalize all to matrix notation
- read a stats book (it's good for you ☺)

Residual Analysis

Check out the non-uniform errors

- where does the model not fit?
- are there outliers, and where?
- time to do some plotting
- time for visualization…

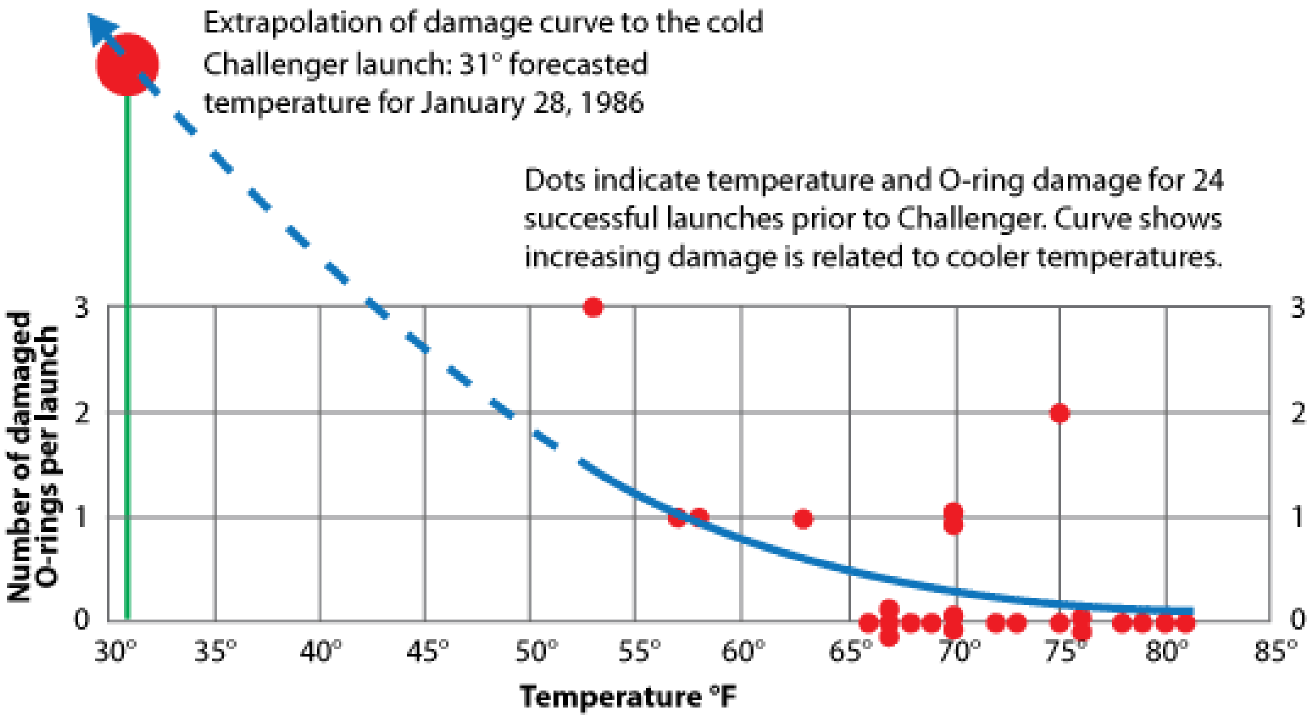
$$\text{plot:} \quad (y_i - (\beta_0 + \beta_1 x_i + \beta_2 x_i \ldots))$$



Visualization of Regression results

Visualization may also reveal trends

- extrapolations
- recall Challenger disaster plot



High Dimensional Data

dimensions >> 3

Problems:

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

Solution:

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

We will discuss:

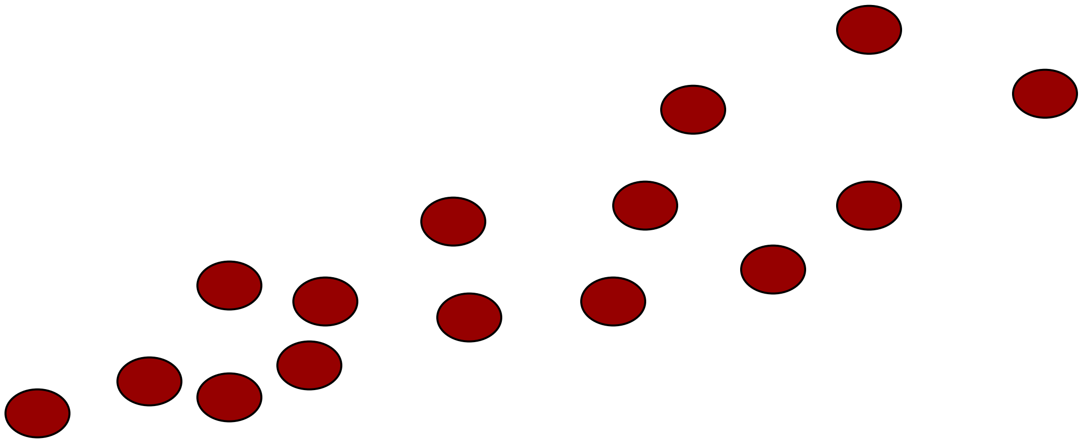
- principal component analysis (PCA) → reduce dimensions
- multi-dimensional scaling (MDS) → stretch space
- clustering → provide structure
- create hierarchies → provide structure
- self-organizing maps → provide structure
- and others

PCA: Algebraic Interpretation

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

PCA: Algebraic Interpretation – 1D

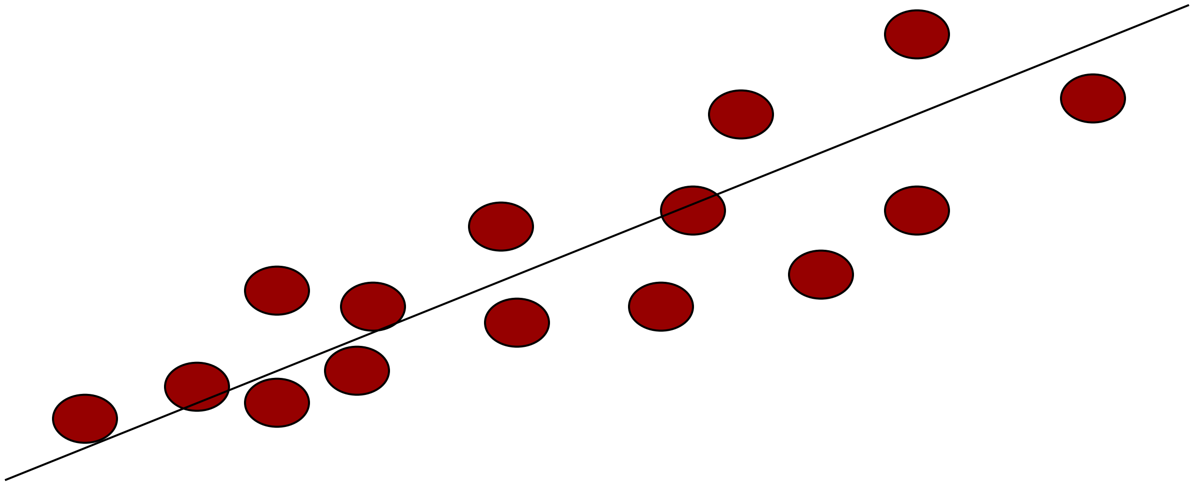
Given m points in a n dimensional space, for large n, how does one project onto a 1 dimensional space?



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

PCA: Algebraic Interpretation – 1D

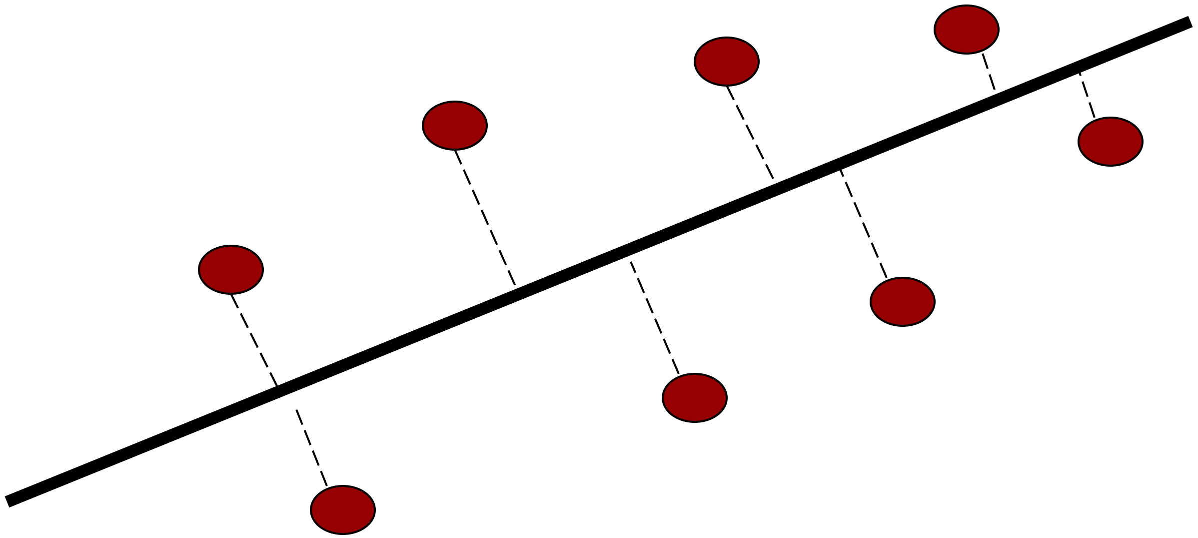
Given m points in a n dimensional space, for large n, how does one project onto a 1 dimensional space?



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

PCA: Algebraic Interpretation – 1D

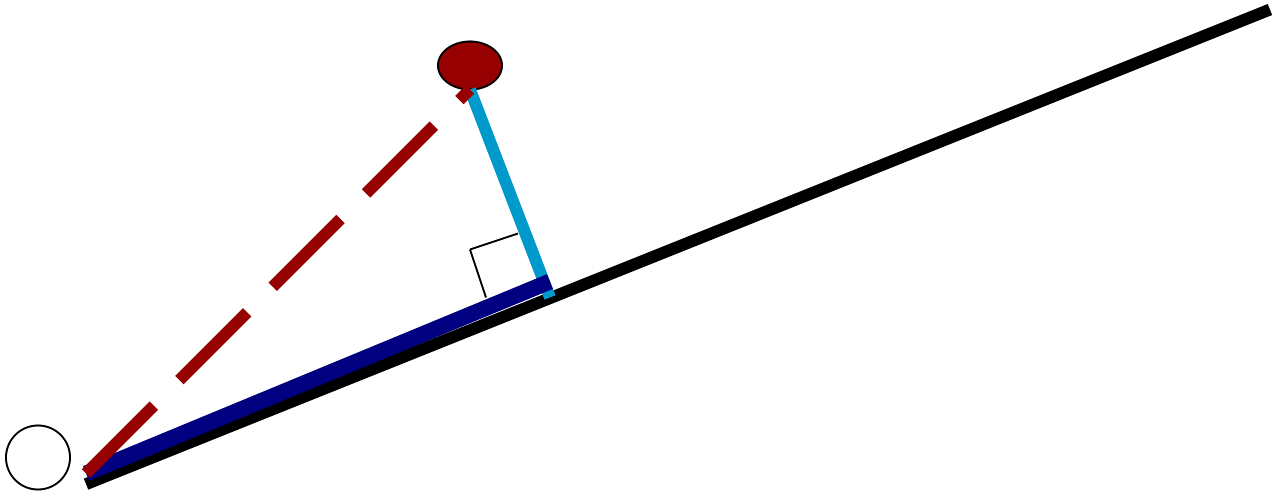
Formally, minimize sum of squares of distances to the line.



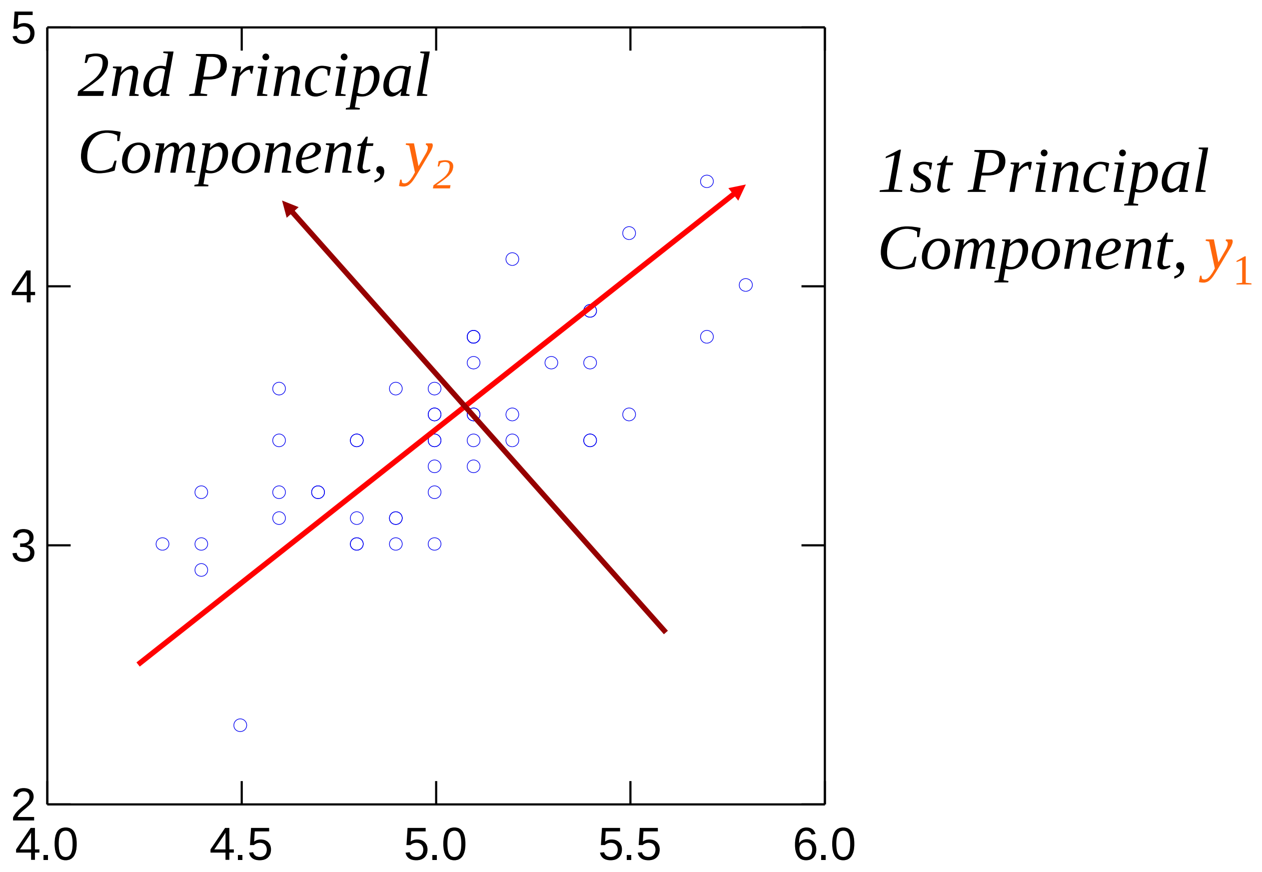
Why sum of squares? Because it allows fast minimization,

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.

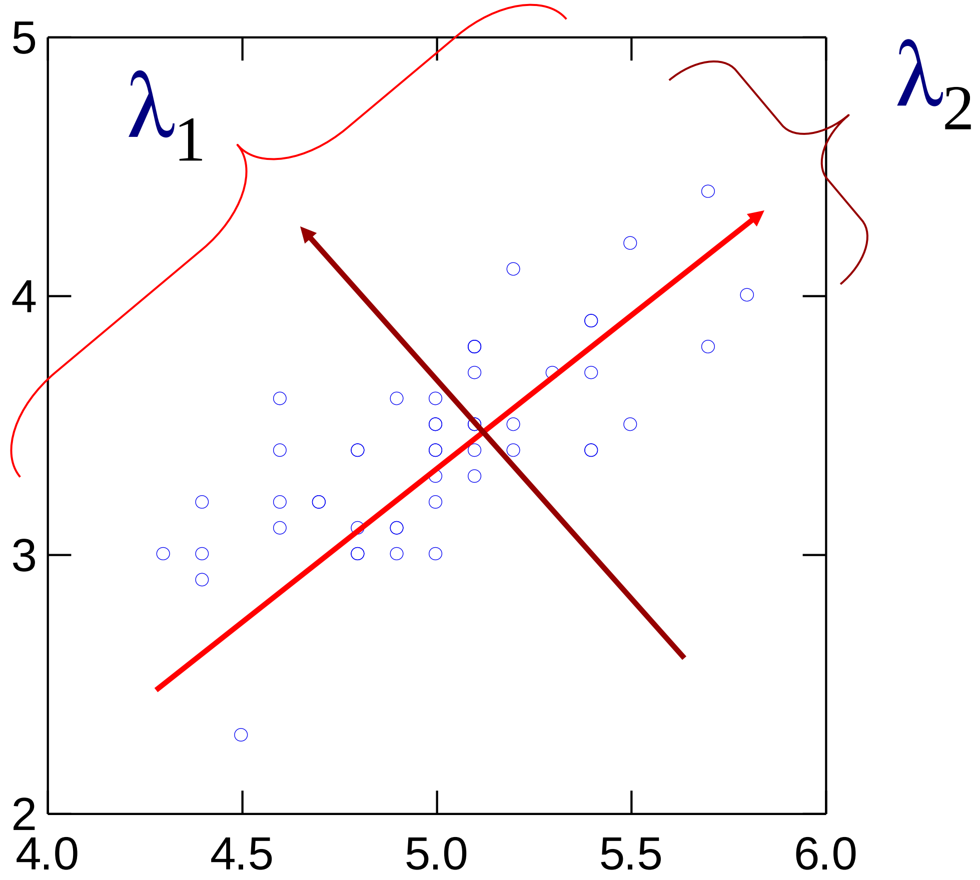


PCA Scores



5
4
3
2
4.0
4.5
5.0
5.5
6.0
2nd Principal Component, y_2
1st Principal Component, y_1

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 = principal component

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

Permute eigenvectors ***x*** so they are in descending order of eigenvalues λ

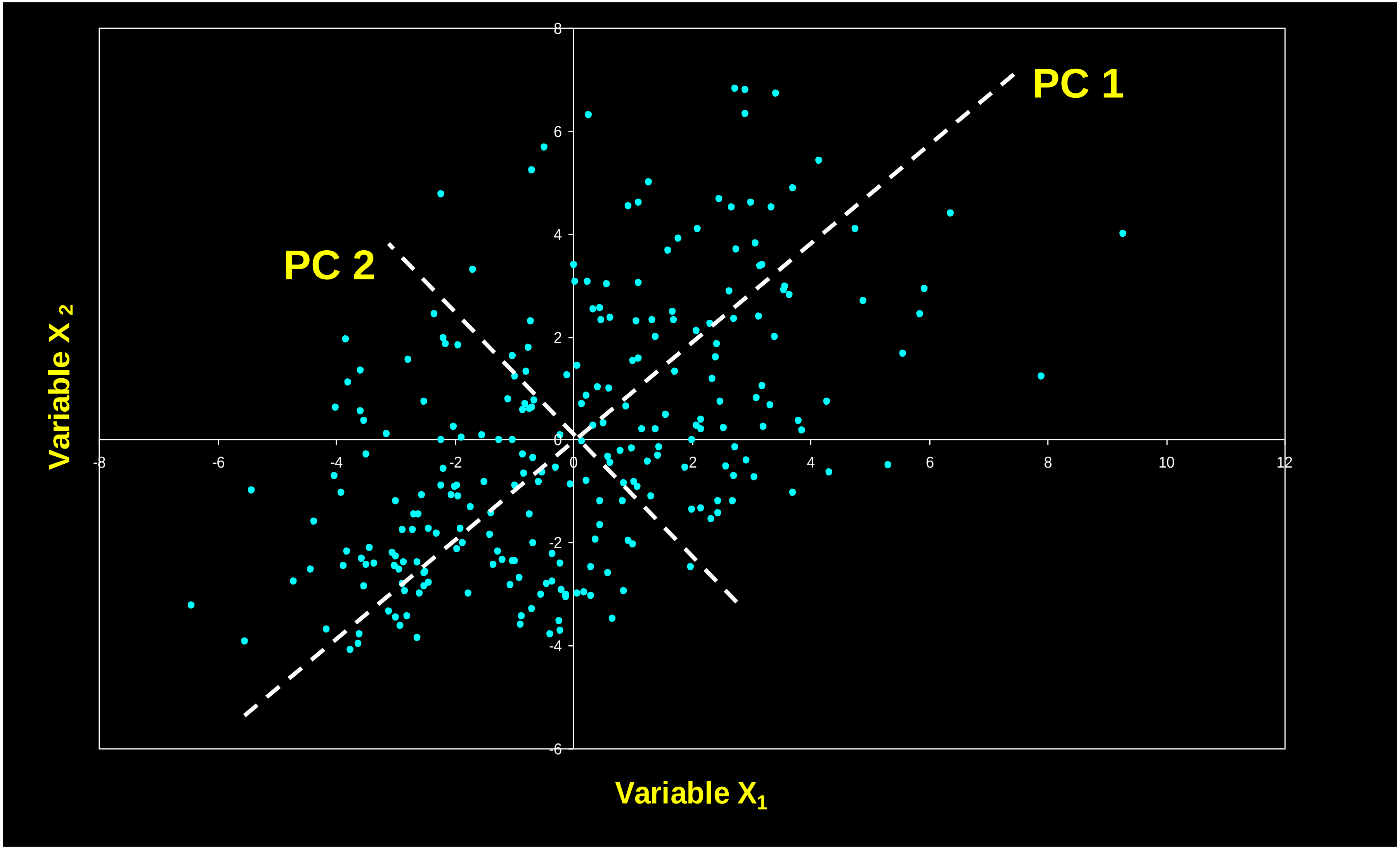
$$\mathbf{C} = \frac{1}{n-1}\sum_{i}^{n}(\overline{x}_i - \mu)(\overline{x}_i - \mu)^T \qquad (\mathbf{C} - \lambda_i \mathbf{I})\,\boldsymbol{x_i} = 0$$

Solve via QR factorization or LU decomposition to get $C = Q\Lambda Q^{-1}$

- Q: matrix with Eigenvectors, Λ diagonal matrix with Eigenvalues

Example

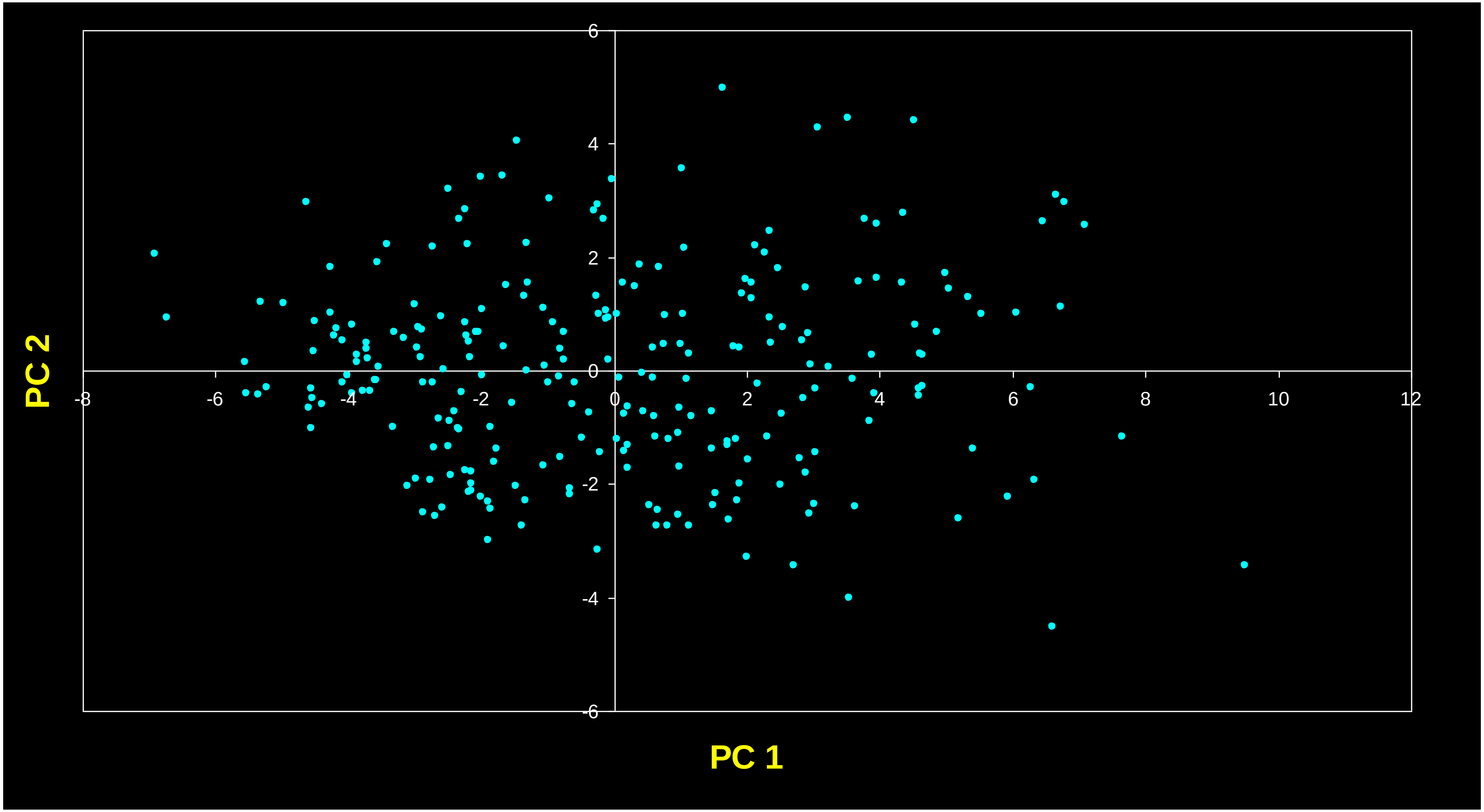
Before PCA



Example

$\lambda_1 = 9.8783 \quad \lambda_2 = 3.0308 \quad \text{Trace} = 12.9091$

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



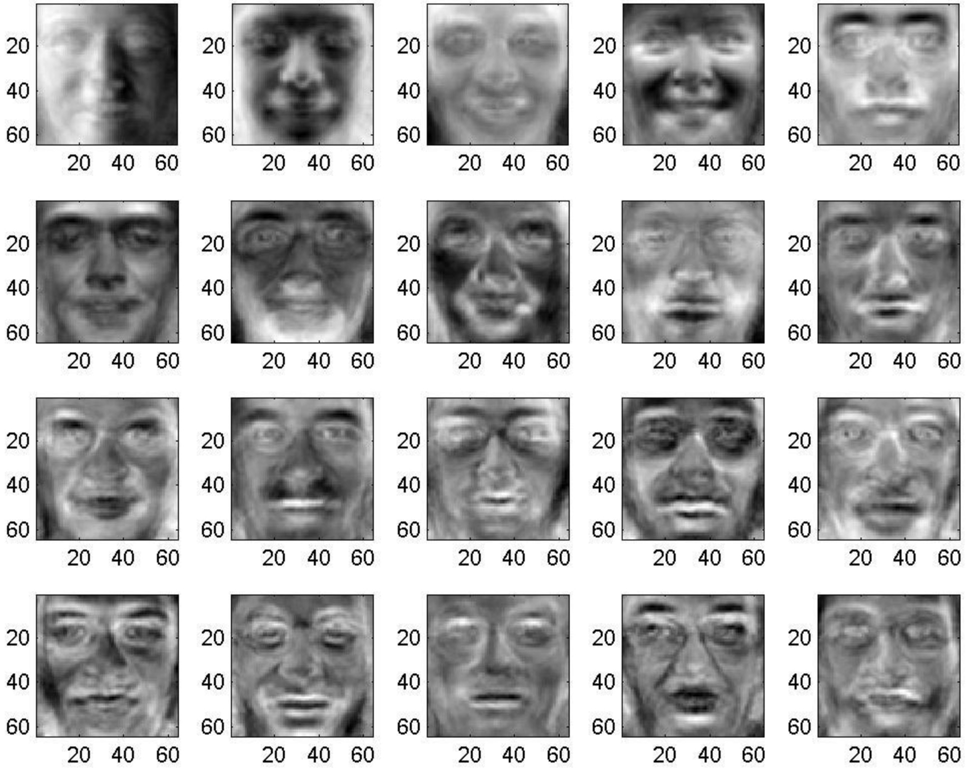
PCA Applied to Faces

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)]

+



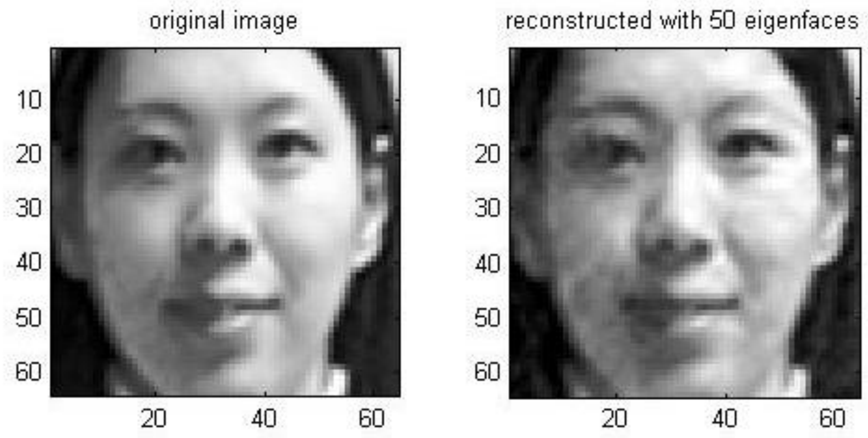
Reconstruction using PCA

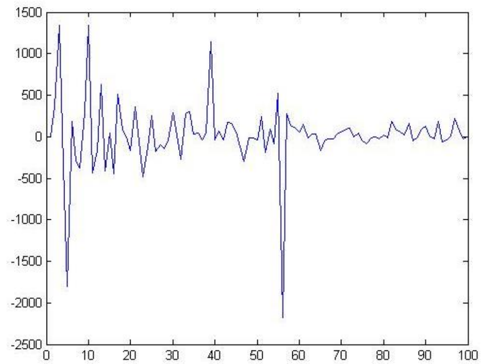
90% variance is captured by the first 50 eigenvectors

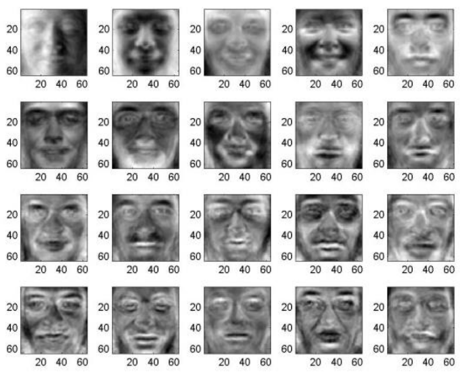
Reconstruct existing faces using only 50 basis images

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

f







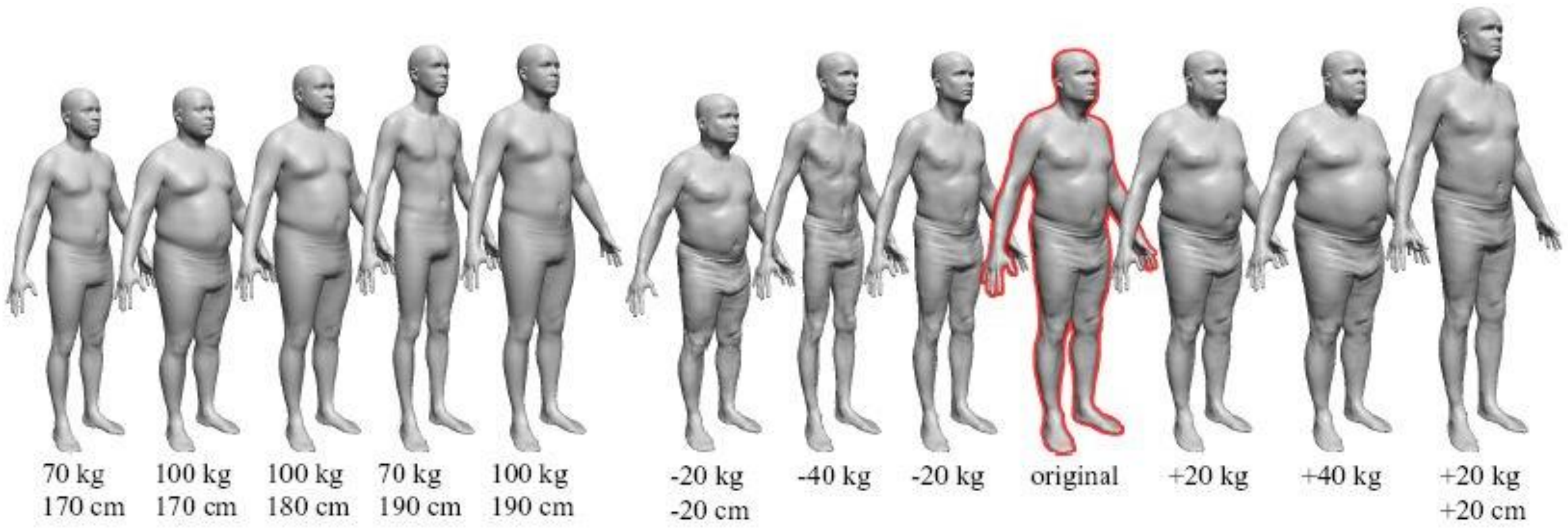
PCA Applied to Human Body Shapes

Similar concepts can also be used for human body shapes

- see Allen, Curless, Popovic, "The Space of Human Body Shapes", SIGGRAPH 2003.
- interpolation in PCA space allows generation of plausible new body shapes

Store additional data (age, weight, height, etc.) with each body

- learn the derivative function: Δ data → Δ body
- use this derivative function to predict Δ data → Δ given body



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

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

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 projects 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

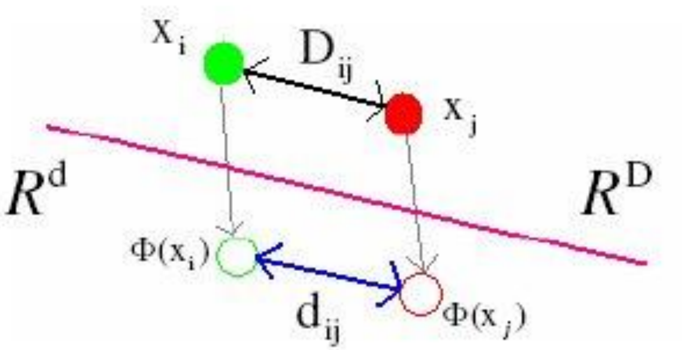
Useful when absolute measurements are unavailable

- 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

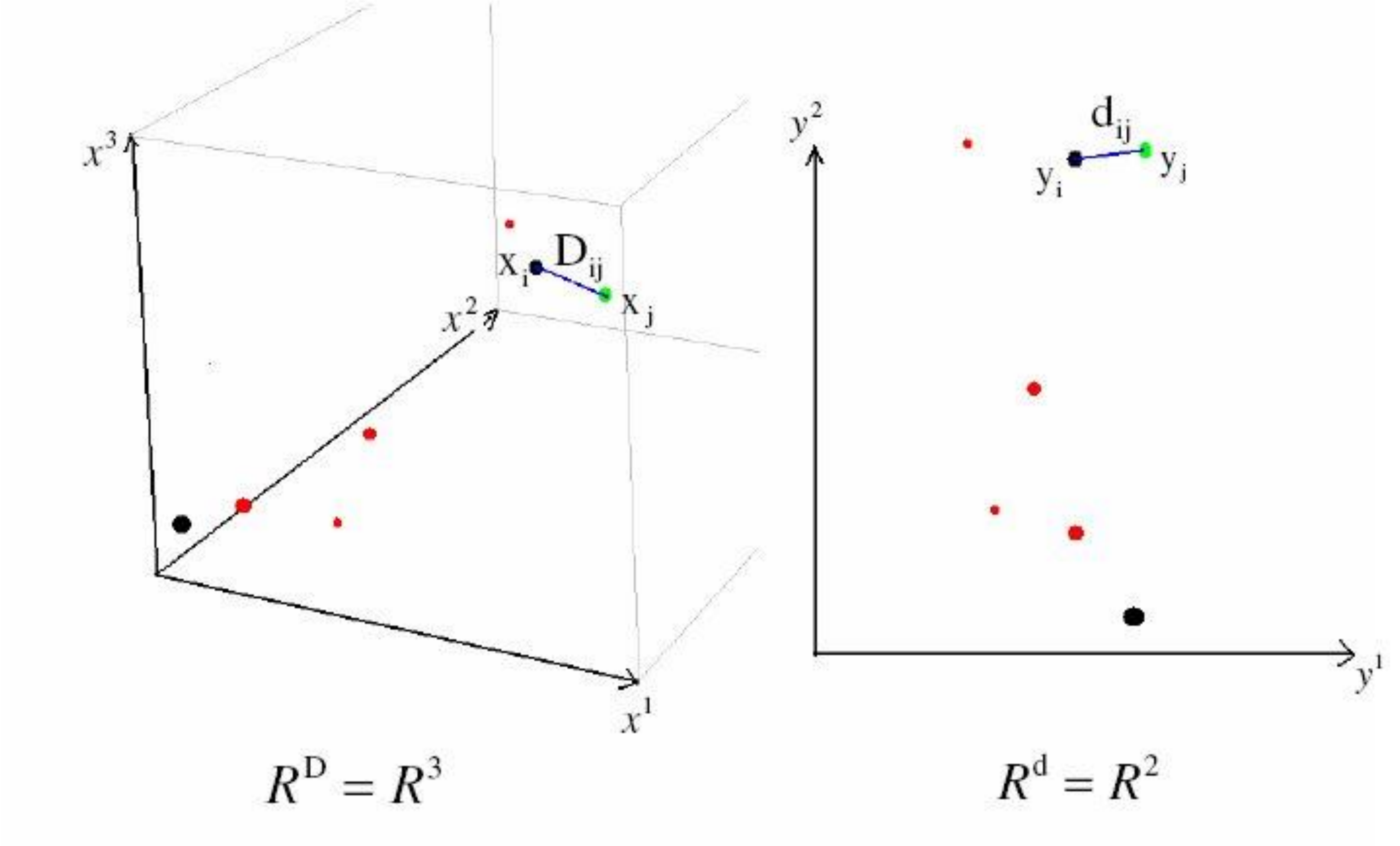
Strategy (of metric MDS):

- 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:

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

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

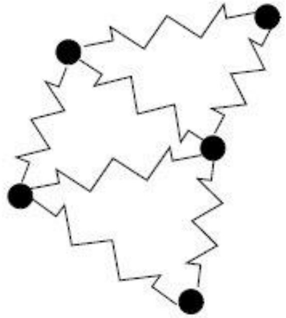
MDS: Algorithm



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



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

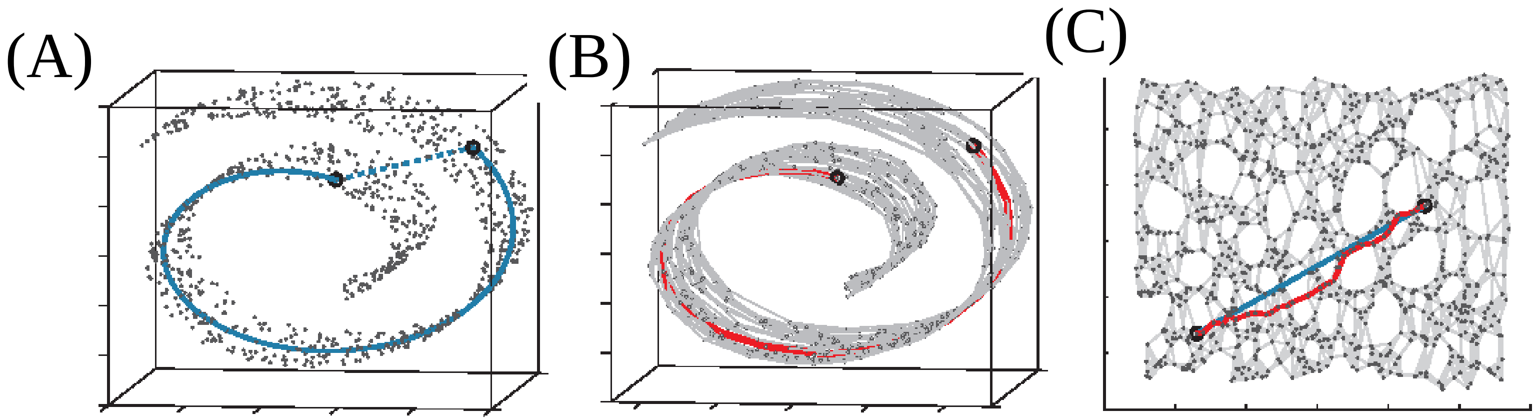
Distances calculated with geobytes.com/CityDistanceTool

Result of MDS

Actual Plot of Cities

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

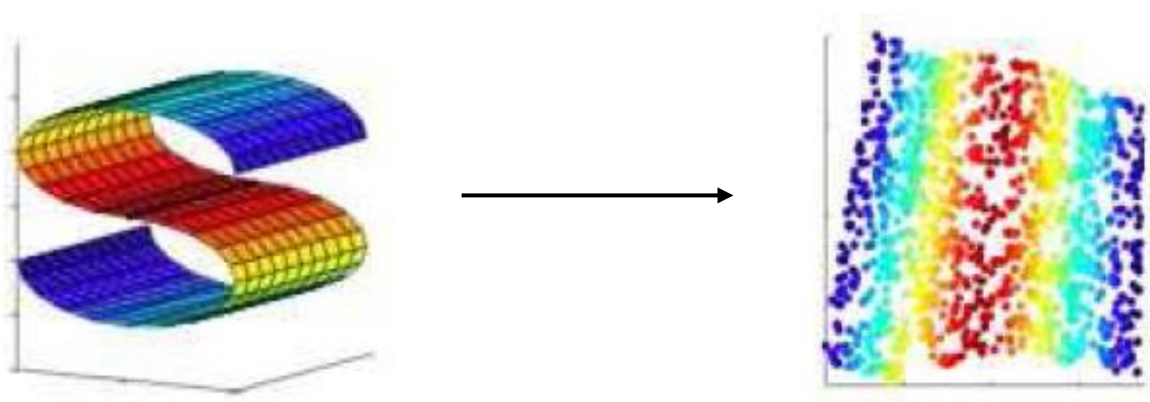
- construct neighborhood graph G → (B)
- for each pair of points in G compute the shortest path distances → geodesic distances
- fill similarity matrix with these geodesic distances
- embed (layout) in low-D (2D) with MDS → (C)

Manifold Learning: Locally Linear Embedding (LLE)

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

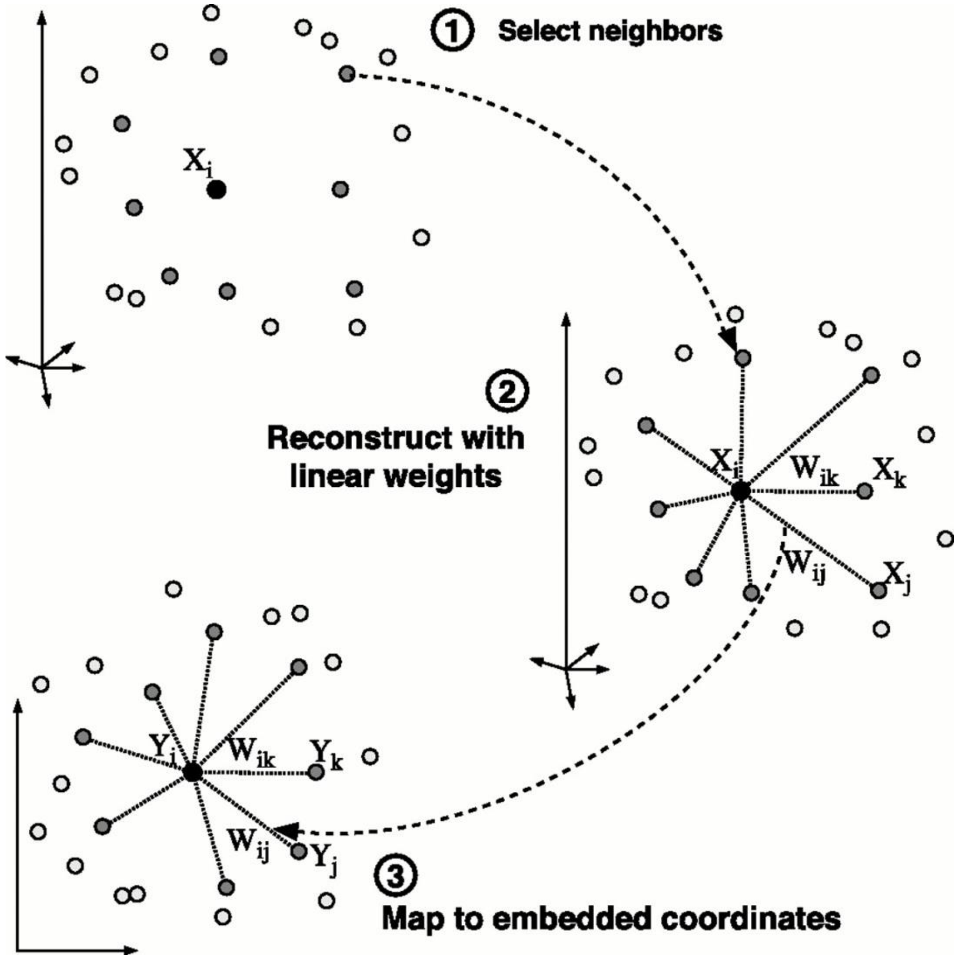
Based on simple geometric intuitions.

- 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



from: "Nonlinear Dimensionality Reduction by Locally Linear Embedding"
S. Roweis, L. Saul

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

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

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

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

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

- 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.

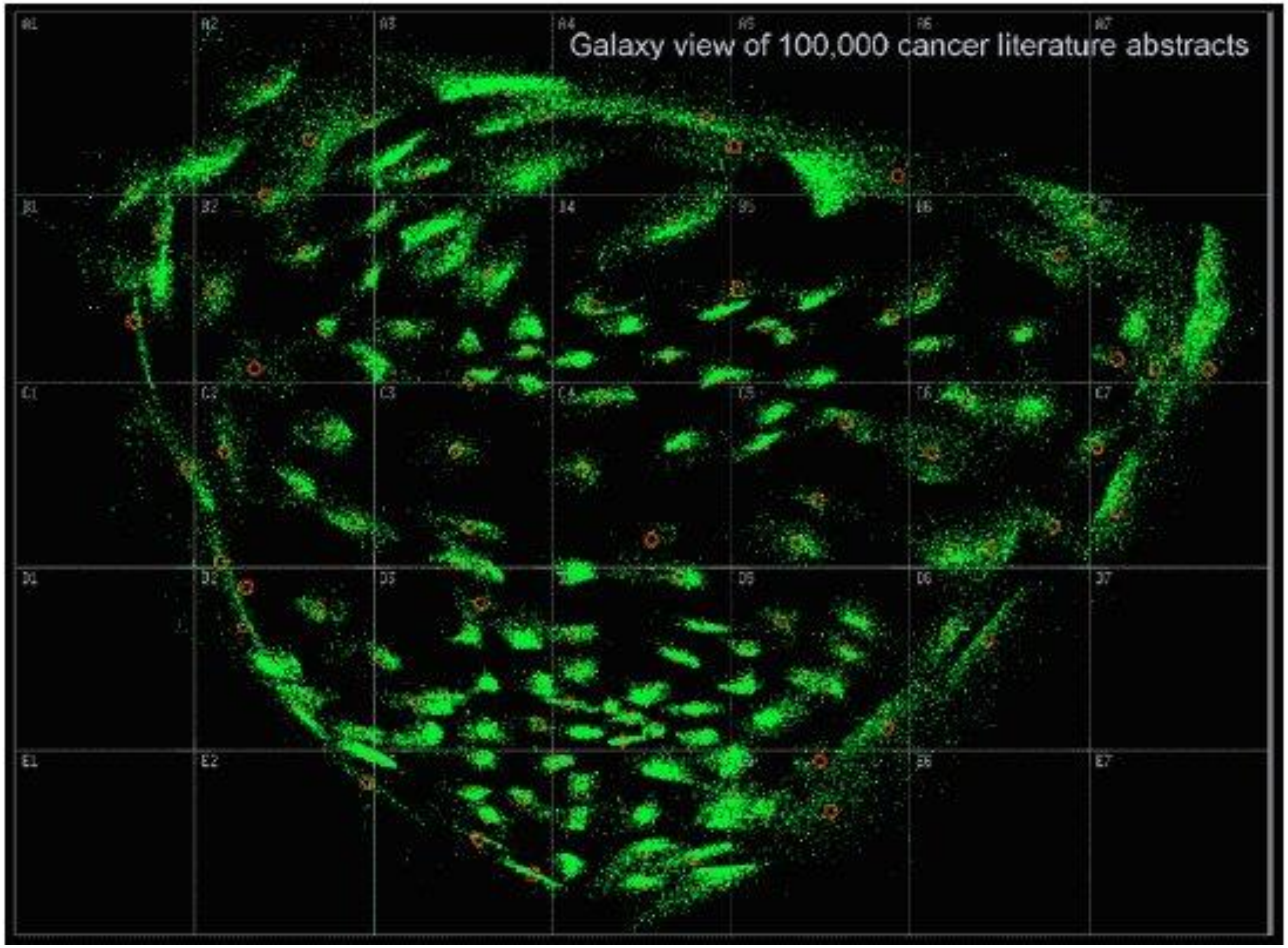
Training uses a set of input vectors several times:

- 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

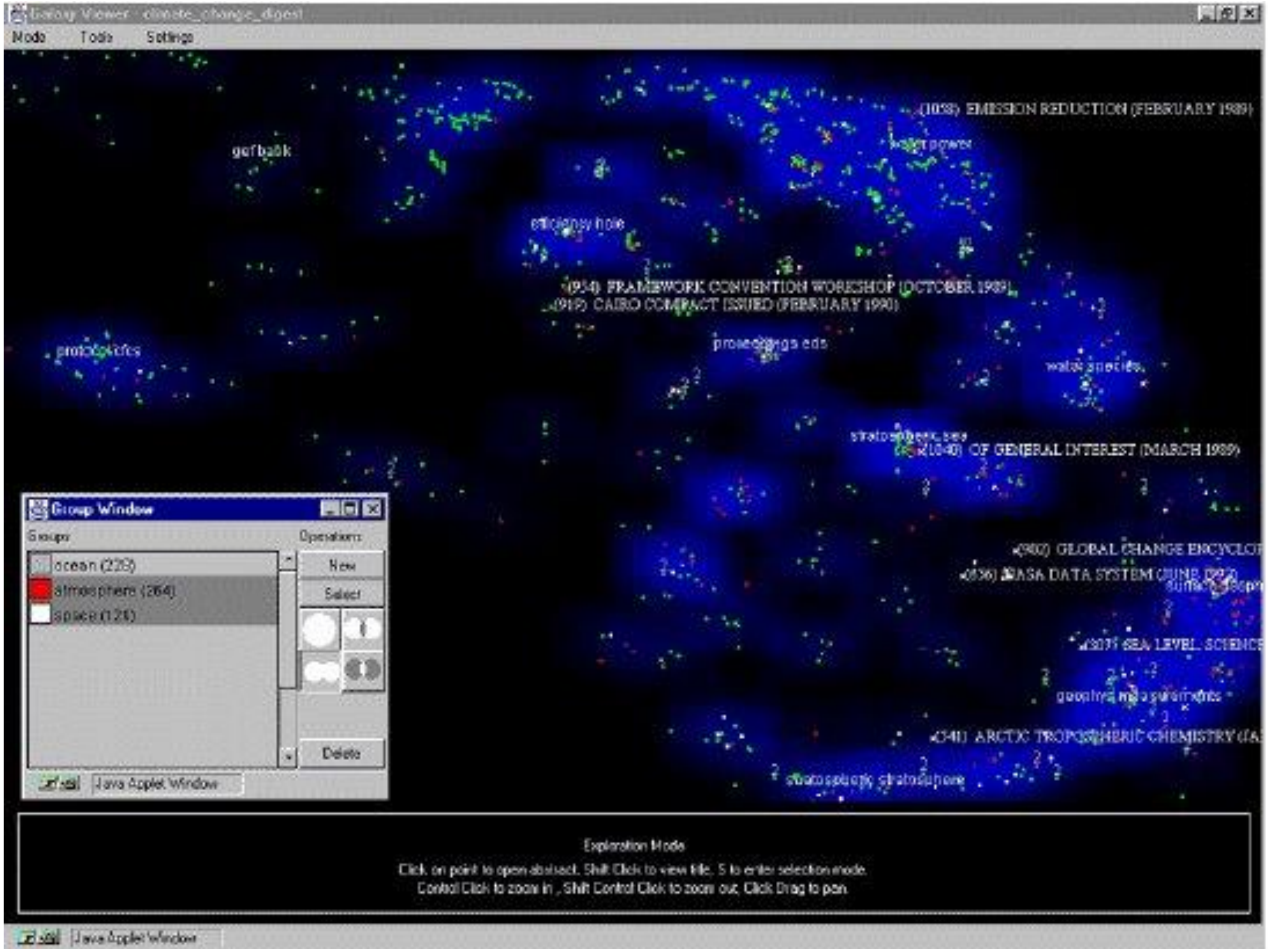
SOM Examples: Galaxies



Presentation of documents where similar ones cluster together

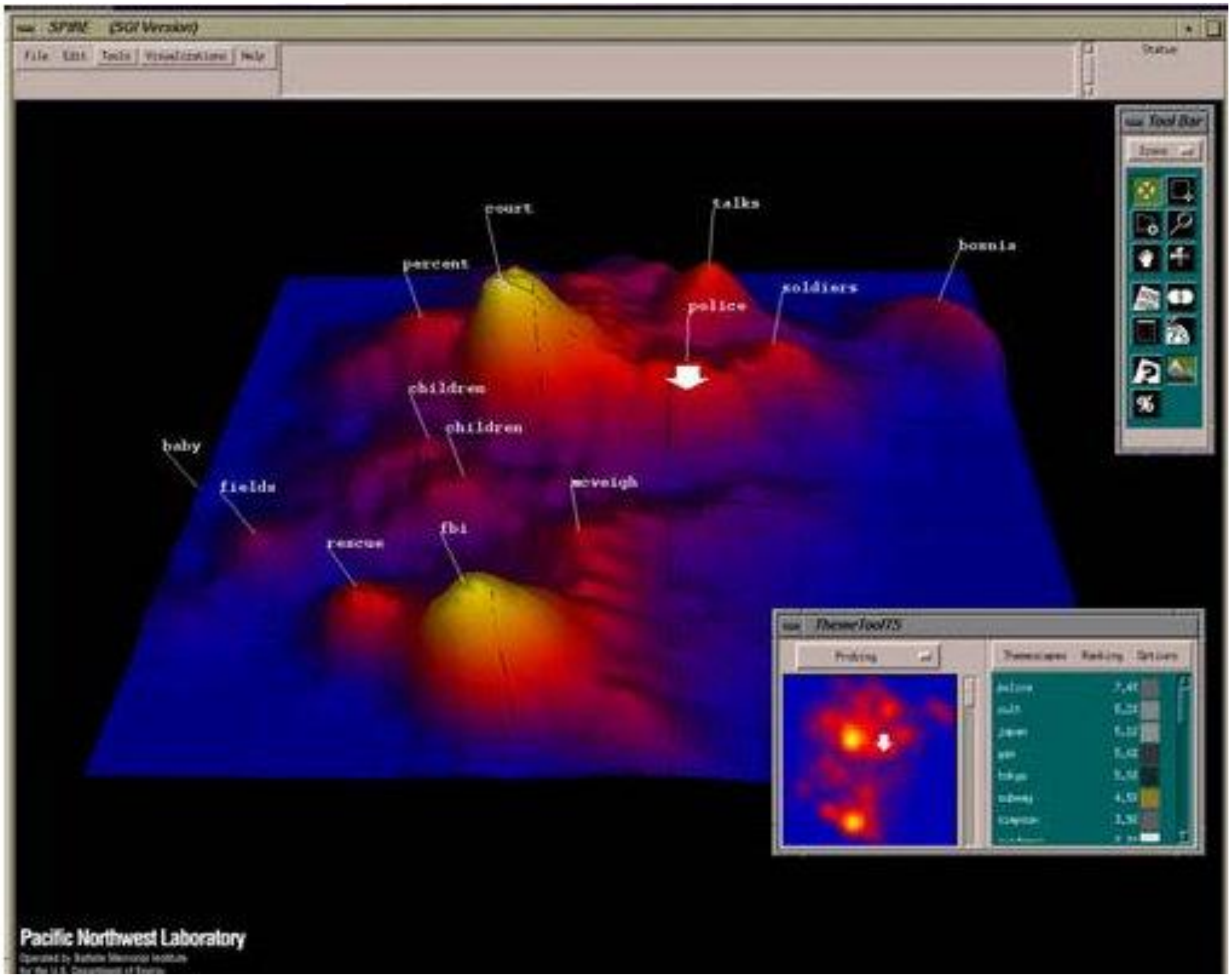
PNNL

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)

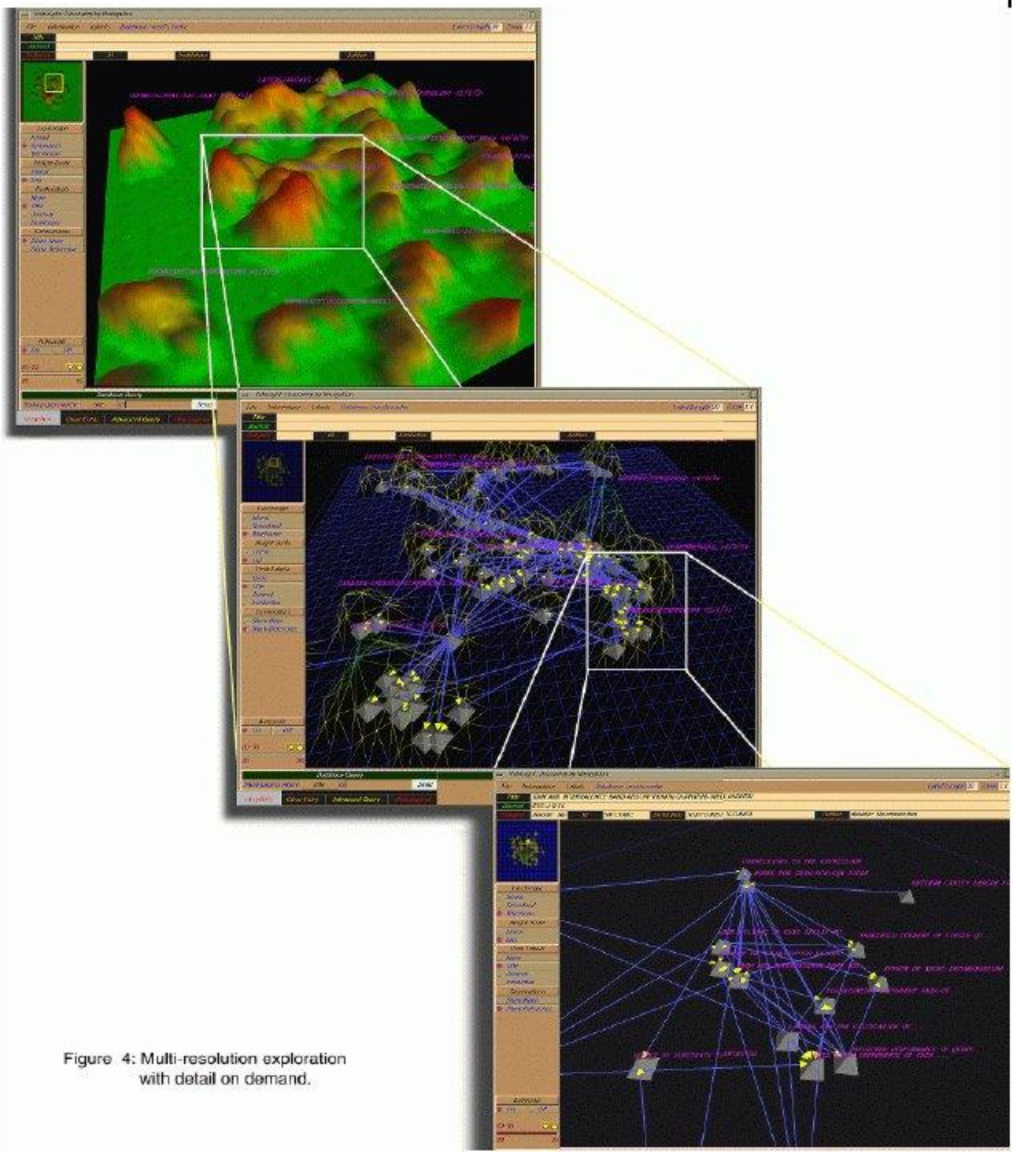
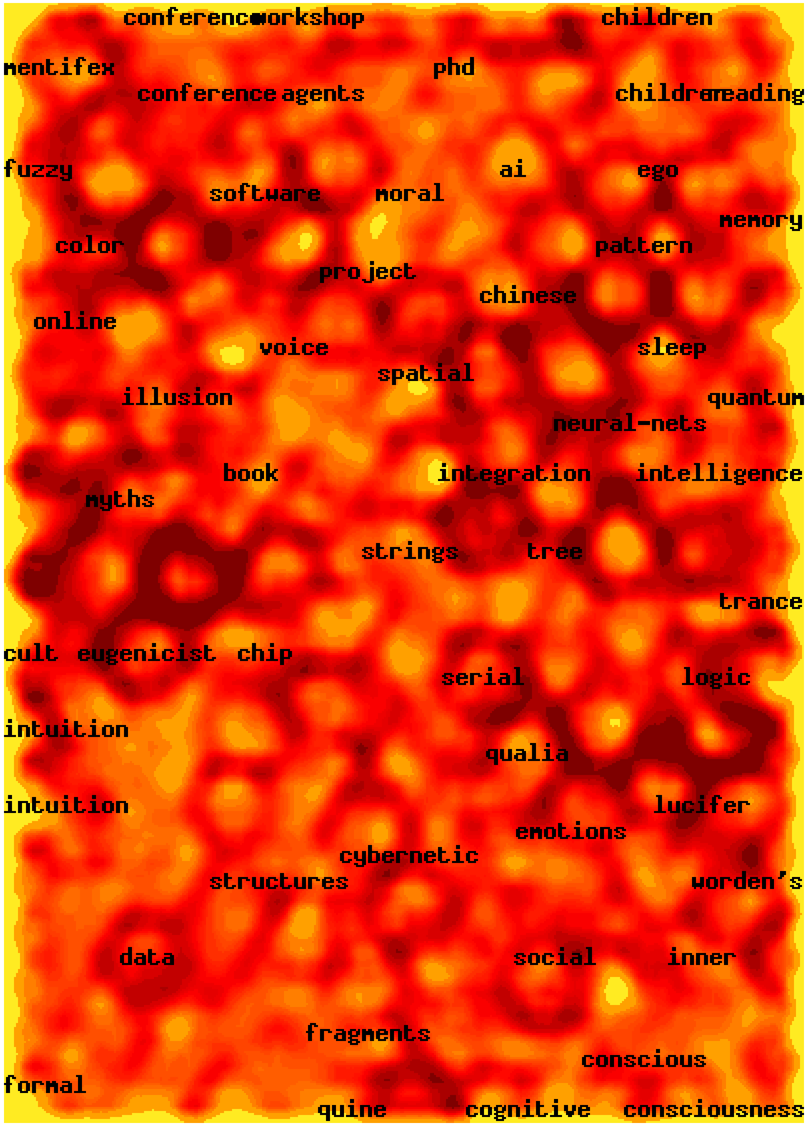
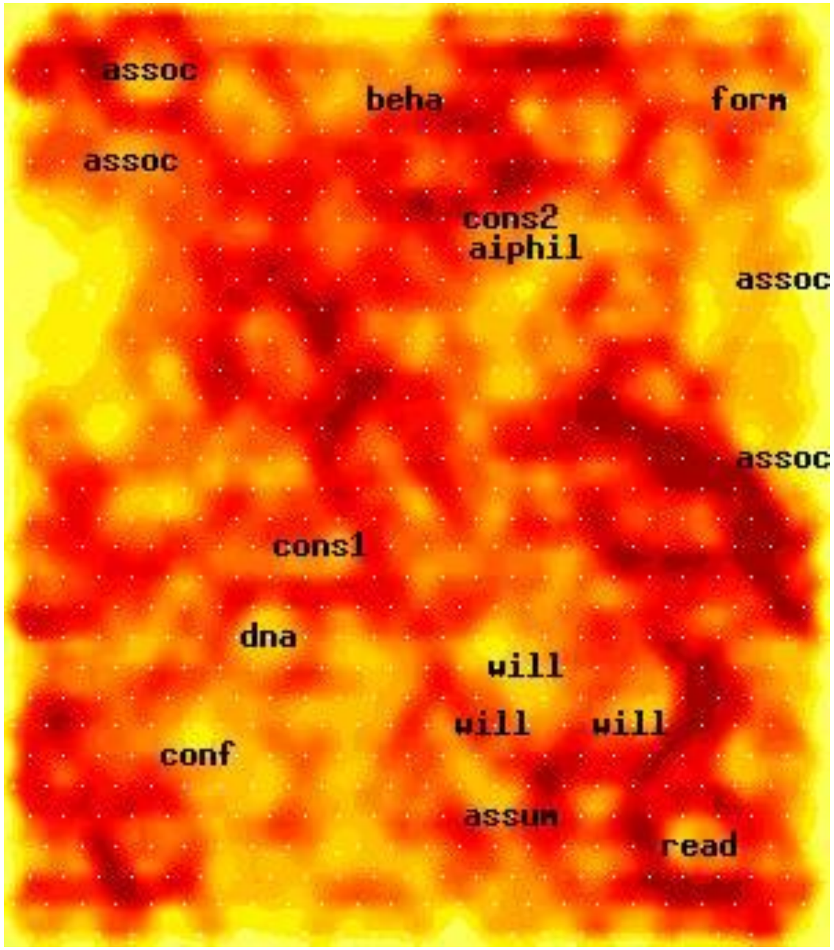


Figure 4: Multi-resolution exploration with detail on demand.

SOM Examples: Websom



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



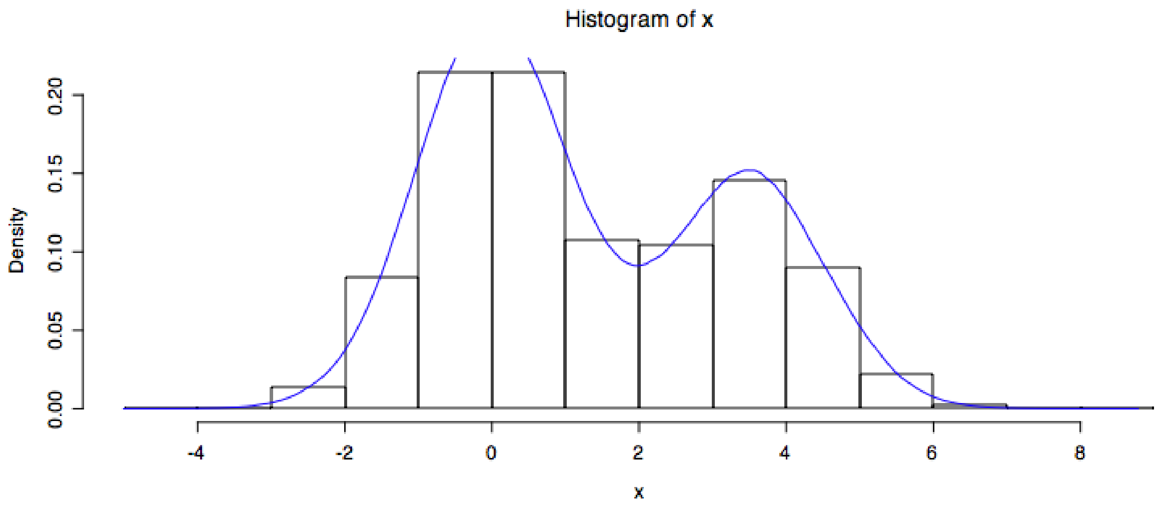
Non-Parametric Statistics

Distribution free

- does not rely on assumptions that the data are drawn from a given probability distribution (such as a normal distribution)

Often used tools:

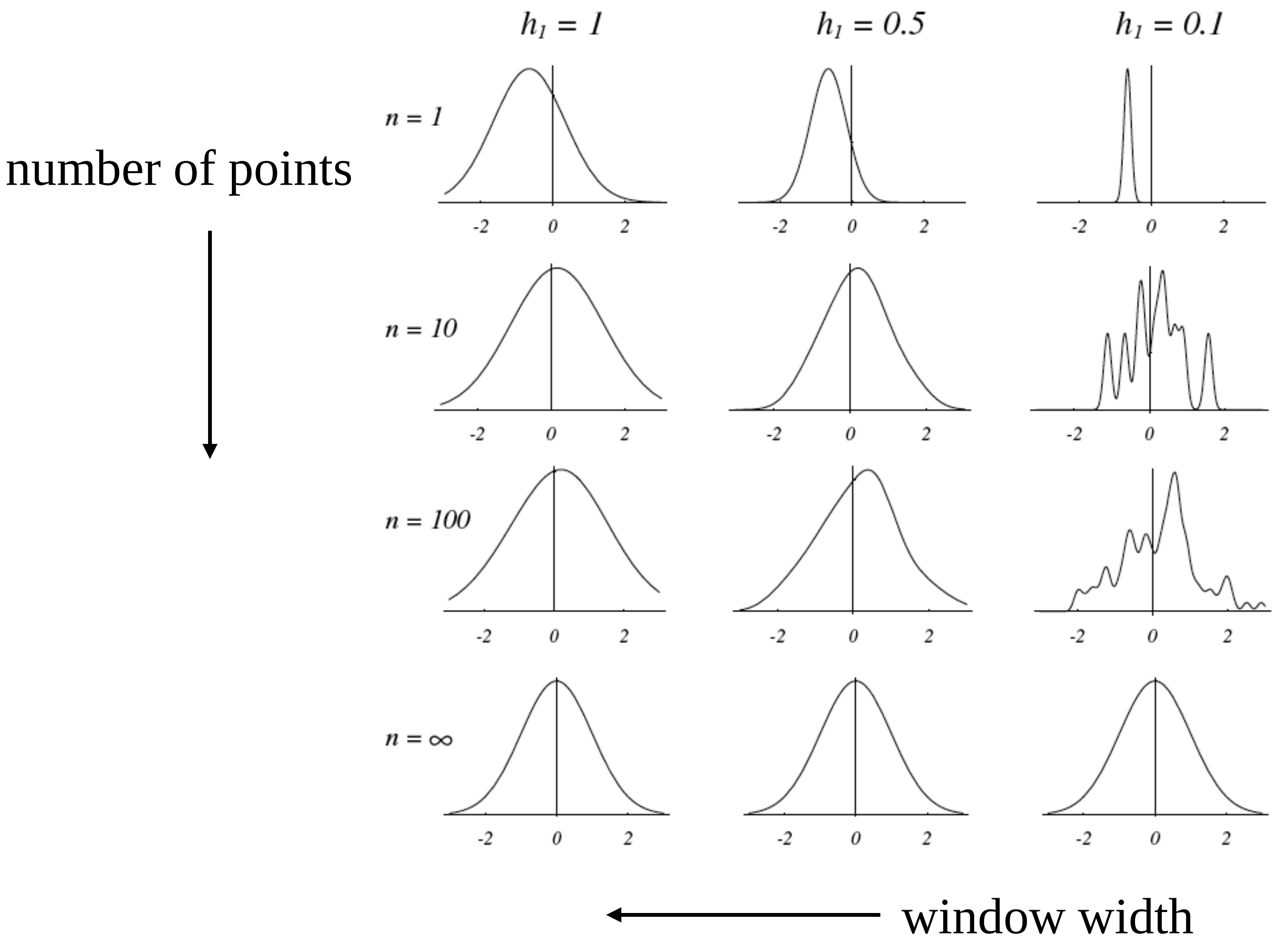
- histograms (partitions space into bins)
- kernel density estimation (better than histograms → continuous)
- regression based on kernels, splines, wavelets, etc.
- data envelope analysis



Parzen Window

Estimates density from discrete observations

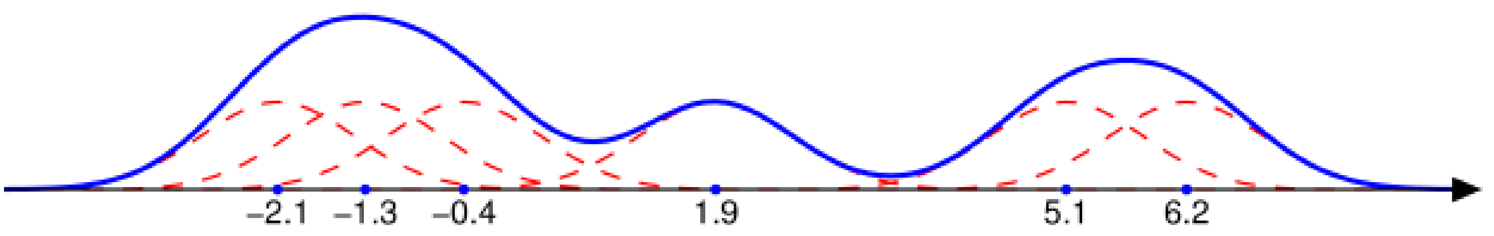
- smooth (blur) with a smooth kernel function (such as a Gaussian)



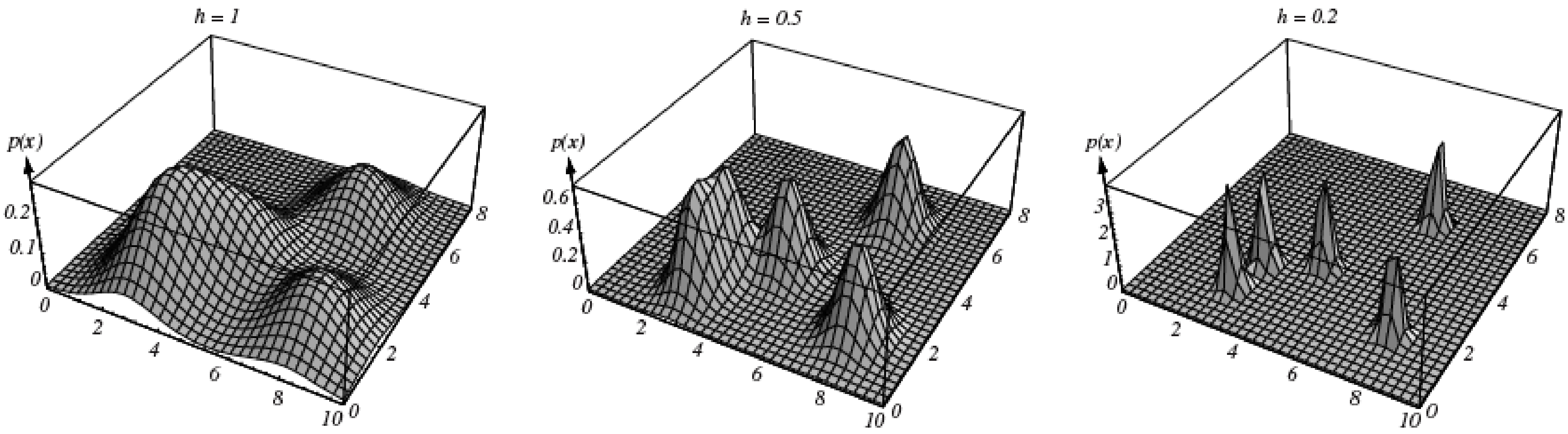
Parzen Window

Think of every data point as a Gaussian kernel

- superposition creates density “humps”



- varying the kernel size yields multi-scale data decompositions



from Duda, Hart, Stork: Pattern Classification

Analogous to Human Vision

Gaussian standard deviation doubles for each image