Dimensionality Reduction and Manifold Learning

Tu Bao Ho
Japan Advance Institute of Science and Technology
John von Neumann Institute, VNU-HCM

Introduction

- The background
  - The data is high-dimensional
  - The desire of projecting those data onto a lower-dimensional subspace without losing importance information regarding some characteristic of the original variables

- The methods
  - Accomplishing the reduction of dimensionality through variable selection, referred to as feature selection.
  - Accomplishing the reduction of dimensionality by creating a reduced set of linear or nonlinear transformations of the input variables, referred to as feature extraction.

Linear techniques

- Transformations $\Re^p \to \Re^k$, $(x_1, \dots, x_p) \mapsto (s_1, \dots, s_k)$, $k \ll p$, and result in each of the $k \leq p$ components of the new variable being a linear combination of the original variables:

$$s_i = w_{i1}x_1 + \cdots + w_{ip}x_p \text{ for } i = 1, \dots, k \text{ or}$$

$$\mathbf{s} = \mathbf{W}\mathbf{x}$$

where $\mathbf{W}_{k\times p}$ is the linear transformation weight matrix. Expressing the same relationship as

$$\mathbf{x} = \mathbf{A}\mathbf{s}$$

with $\mathbf{A}_{p\times k}$, new variables **s** are also the *hidden* or the *latent* variables.

- In terms of an $n \times p$ observation matrix $\mathbf{X}$, we have

$$S_{ij} = w_{i1}X_{1j} + \cdots + w_{ip}X_{pj} \text{ for } i = 1, \dots, k; \text{j} = 1, \dots, \text{n}$$

where j indicates the jth realization, or, equivalently,

$$\mathbf{S}_{k\times n} = \mathbf{W}_{k\times p}\mathbf{X}_{p\times n}, \qquad \mathbf{X}_{p\times n} = \mathbf{A}_{p\times k}\boldsymbol{S}_{k\times n}$$

Content

Principal component analysis (PCA)

- PC is also known as the *singular value decomposition* (SVD), the Karhunen-Loève transform, the Hotelling transform (1933), and the *empirical orthogonal function* (EOF) method.
- PCA reduces the dimension by finding a few orthogonal linear combinations (the PCs) of the original variables with the largest variance (PCA giảm số chiều bằng cách tìm ra một số nhỏ các tổ hợp tuyến tính trực giao (PC) của các biến gốc với phương sai lớn nhất)
- The first PC, s_1, is the linear combination with the largest variance $\boldsymbol{s}_1 = \mathbf{x}^\tau \mathbf{w}_1$, where coefficient vector $\mathbf{w}_1 = \left(w_{11}, \dots, w_{1p}\right)^\tau$ solves

$$\mathbf{w}_1 = \underset{\|\mathbf{w}=1\|}{\arg\max} \operatorname{Var}\{\mathbf{x}^\tau \mathbf{w}\}$$

- The second PC , s_2, is the linear combination with the second largest variance and orthogonal to the first PC, and so on. There are as many PCs as the number of the original variables.

Principal component analysis

- The first several PCs may explain most of the variance, so that the rest can be disregarded with minimal loss of information.
- As variance depends on the variable scale, first standardize each variable to have mean zero and standard deviation one. Assuming a standardized data with the empirical covariance matrix

$$\mathbf{\Sigma}_{p\times p} = \frac{1}{n}\mathbf{X}\mathbf{X}^{\tau}$$

- We can use the spectral decomposition theorem to write $\mathbf{\Sigma}$ as

$$\mathbf{\Sigma} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\tau}$$

where $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \dots, \lambda_p)$ is the diagonal matrix of the ordered eigenvalues $\lambda_1 \le \cdots \le \lambda_p$, and $\mathbf{U}$ is a $p \times p$ orthogonal matrix containing the eigenvectors.

Principal component analysis

- *Property 1*. The subspace spanned by the first *k* eigenvectors has the smallest mean square deviation (độ lệch trung bình bình phương nhỏ nhất) from X among all subspaces of dimension *k* (Mardia et al., 1995).

- *Property 2*. The total variation in the eigenvalue decomposition is equal to the sum of the eigenvalues of the covariance matrix, (sai khác toàn bộ trong phân tích gía trị riêng bằng tổng các vector riêng của ma trận phương sai)

$$\sum_{i=1}^{p} Var(PC_i) = \sum_{i=1}^{p} \lambda_i = \sum_{i=1}^{p} \mathrm{trace}(\boldsymbol{\Sigma})$$

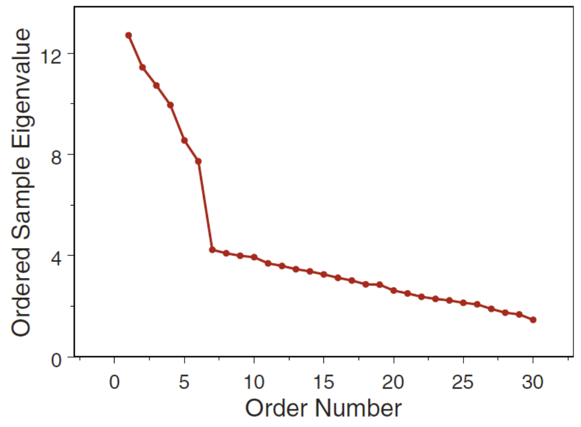
and the fraction

$$\sum_{i=1}^{k} \lambda_i \Big/ \mathrm{trace}(\boldsymbol{\Sigma})$$

gives the cumulative proportion of the variance explained by the first *k* PCs (tỷ lệ tích lũy của biến đổi tính theo *k* PC đầu tiên) .

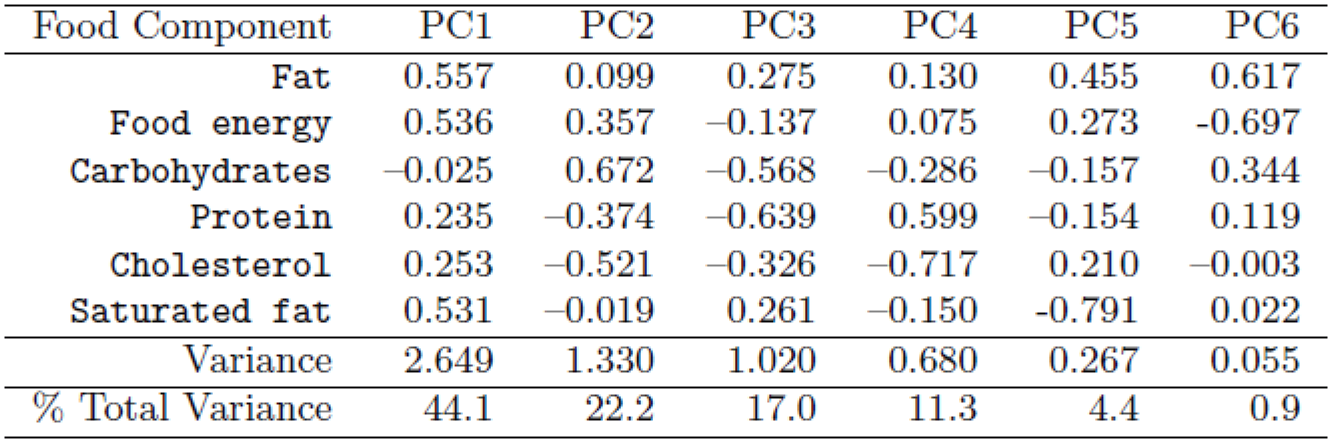
Principal component analysis

- By plotting the cumulative proportions in as a function of k, one can select the appropriate number of PCs to keep in order to explain a given percentage of the overall variation.
- An alternative way to reduce the dimension of a dataset using PCA: Instead of using the PCs as the new variables, this method uses the information in the PCs to find important variables in the original dataset.
- Another way: The number of PCs to keep is determined by first fixing a threshold λ_0, then only keeping the eigenvectors (at least four) such that their corresponding eigenvalues are greater than λ_0 (Jolliffe, 1972, 1973).



Principal component analysis
Example

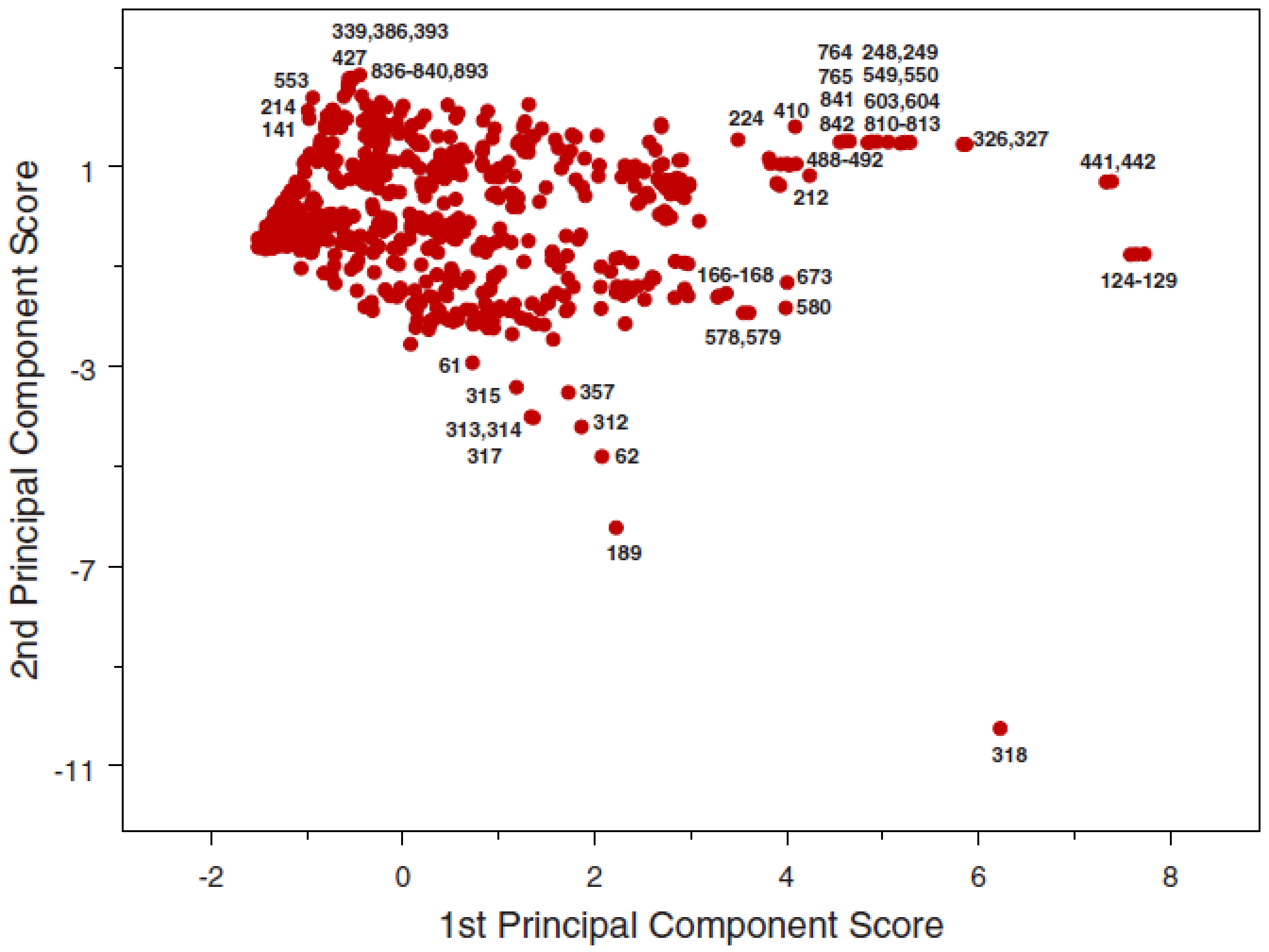
- Nutritional data from 961 food items. The nutritional components of each food item are given by seven variables: fat (grams), food energy (calories), carbohydrates (grams), protein (grams), cholesterol (milligrams), weight (grams), and saturated fat (grams).
- PCA of the transformed data yields six principal components ordered by decreasing variances. The first three principal components, PC1, PC2, and PC3, which account for more than 83% of the total variance



Food Component	PC1	PC2	PC3	PC4	PC5	PC6
Fat	0.557	0.099	0.275	0.130	0.455	0.617
Food energy	0.536	0.357	–0.137	0.075	0.273	-0.697
Carbohydrates	–0.025	0.672	–0.568	–0.286	–0.157	0.344
Protein	0.235	–0.374	–0.639	0.599	–0.154	0.119
Cholesterol	0.253	–0.521	–0.326	–0.717	0.210	–0.003
Saturated fat	0.531	–0.019	0.261	–0.150	-0.791	0.022
Variance	2.649	1.330	1.020	0.680	0.267	0.055
% Total Variance	44.1	22.2	17.0	11.3	4.4	0.9

Principal component analysis

Example



Independent component analysis (ICA)

- A method that seeks linear projections, not necessarily orthogonal to each other, but nearly statistically independent as possible.
- The random variables $\mathbf{x} = \{x_1, \dots, x_p\}$ are uncorrelated, if for $\forall\ i \neq j$, $1 \leq i, j \leq p$, we have $Cov(x_i, x_j) = 0$. Independence requires that the multivariate probability density function factorizes,

$$f(x_1, \dots, x_p) = f_1(x_1) \dots f_p(x_p).$$

independence $\Rightarrow$ *uncorrelated, but uncorrelated* $\nRightarrow$ *independence.*

- The noise-free ICA model for the p-dimensional random vector $\mathbf{x}$ seeks to estimate the components of the k-dimensional vector $\boldsymbol{s}$ and the $p \times k$ full column rank mixing matrix $\mathbf{A}$

$$(x_1, \dots, x_p)^\tau = \mathbf{A}_{p\times k}(s_1, \dots, s_k)^\tau$$

such that the components of s are as independent as possible, according to some definition of independence.

Independent component analysis (ICA)

- The noisy ICA contains an additive random noise component,

$$(x_1, \dots, x_p)^\tau = \mathbf{A}_{p \times k}(s_1, \dots, s_k)^\tau + (u_1, \dots, u_p)^\tau$$

  Estimation of such models is still an open research issue.

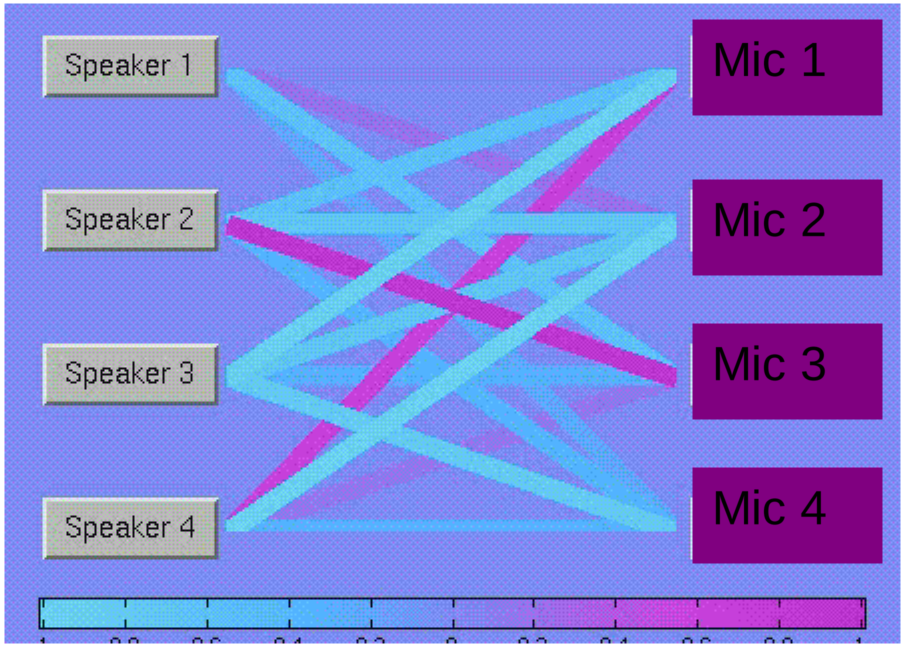
- In contrast with PCA, the goal of ICA is not necessarily dimension reduction. To find $k < p$ independent components, one needs to first reduce the dimension of the original data p to k, by a method such as PCA.

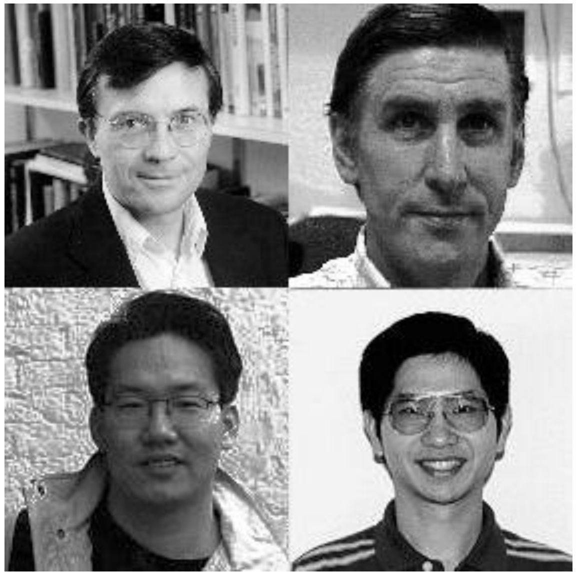
- No order among the PCs of ICA. ICA can be considered a generalization of the PCA and the PP (project pursuit) concepts.

- ICA is applied to many different problems, including exploratory data analysis, blind source separation, blind deconvolution, and feature extraction.

Independent component analysis (ICA)

Perform ICA





Terry

Scott

Te-Won

Tzyy-Ping

Play Mixtures

Play Components

Factor analysis (FA)

- A linear method, based on the second-order data summaries. Factor analysis assumes that the measured variables depend on some unknown, and often unmeasurable, common factors.
- Typical examples include variables defined as various test scores of individuals, as such scores are thought to be related to a common “intelligence" factor.
- The goal of FA is to uncover such relations, and thus can be used to reduce the dimension of datasets following the factor model.
- The zero-mean p-dimensional random vector $x_{p\times 1}$ with covariance matrix $\boldsymbol{\Sigma}$ satisfies the k-factor model

$$\mathbf{x} = \boldsymbol{\Lambda}\mathbf{f} + \mathbf{u}$$

where $\boldsymbol{\Lambda}_{p\times k}$ is a matrix of constants, and $\mathbf{f}_{k\times 1}$ and $\boldsymbol{\Lambda}_{p\times k}$ and $\mathbf{u}_{p\times 1}$ are the random common factors and specific factors, respectively.

Canonical variate analysis
(*CVA*, Hotelling, 1936)

- A method for studying linear relationships between two vector variates,
$$\mathbf{X} = (X_1, \dots, X_r)^\tau \text{ and } \mathbf{Y} = (Y_1, \dots, Y_s)^\tau$$
which have different dimensions.

- CVA seeks to replace the two sets of correlated variables, $\mathbf{X}$ and $\mathbf{Y}$, by t pairs of new variables,
$$(\xi_i, \omega_i), i = 1, \dots, t; \quad t \le min(r, s)$$
where
$$\xi_i = \boldsymbol{g}_j^\tau \mathbf{X} = \sum_{k=1}^{r} g_{kj} X_k \,, \qquad \omega_i = \boldsymbol{h}_j^\tau \mathbf{Y} = \sum_{k=1}^{s} h_{kj} Y_k$$

Canonical variate and correlation analysis

Least-squares optimality of CVA

- The task

$$(X_1, \dots, X_r, Y_1, \dots, Y_s)^\tau \rightarrow \left((\xi_1, \omega_1), \dots, (\xi_t, \omega_t)\right)^\tau$$

Linear projections by $(t \times r)$-matrix **G** and $(t \times s)$-matrix **H** with $\left(1 \le t \le min(r, s)\right)$:

$$\boldsymbol{\xi} = \mathbf{GX}, \qquad \boldsymbol{\omega} = \mathbf{HY},$$

Least-square error criterion, to find $\boldsymbol{\nu}$, **G**, and **H** so that

$$\mathbf{HY} \approx \boldsymbol{\nu} + \mathbf{GX}$$

to minimize

$$\mathrm{E}\{(\mathbf{HY} - \boldsymbol{\nu} + \mathbf{GX})(\mathbf{HY} - \boldsymbol{\nu} + \mathbf{GX})^\tau\}$$

which measure how well we can reconstruct **X** and **Y** from pairs of (ξ_i, ω_i)

- The goal is

To choose the best $\boldsymbol{\nu}$, **G**, and **H** in the least-square sense.

Projection Pursuit
Phép chiếu đuổi

- The motivation

  The desire to discover "interesting" low-dimensional (typically, one- or two-dimensional) linear projections of high-dimensional data
- The origin

  The desire to expose specific non-Gaussian features (variously referred to as "local concentration," "clusters of distinct groups," "clumpiness," or "clottedness") of the data.
- The strategy
  1. Set up a *projection index* $\mathfrak{I}$ to judge the merit of a particular one or two-dimensional (or sometimes three-dimensional) projection of a given set of multivariate data.
  2. Use an *optimization algorithm* to find the global and local extrema of that projection index over all m-dimensional projections of the data.

Projection Pursuit
Projection Indexes

- Projection indexes should be chosen to possess certain computational and analytical properties, especially that of affine invariance (location and scale invariance).
- A special case of PP occurs when the projection index is the *variance*. Maximizing the variance reduces PP to PCA, and the resulting projections are the leading principal components of **X**.
- Maximizing the variance is equivalent to minimizing the corresponding Gaussian log-likelihood; in other words, the projection is most interesting (in a variance sense) when **X** is least likely to be Gaussian. Typical PP:
  - Cumulant-based index
  - Polynomial-based indexes
  - Shannon negentropy
  - Optimizing the projection index

Introduction

- The linear projection methods can be extremely useful in discovering low-dimensional structure when the data actually lie in a linear (or approximately linear) lower-dimensional subspace M (called a manifold) of input space $\Re^r$.
- What can we do if we know or suspect that the data actually lie on a low dimensional nonlinear manifold, whose structure and dimensionality are both assumed unknown?
- Dimensionality reduction → Problem of nonlinear manifold learning.
- When a linear representation of the data is unsatisfactory, we turn to specialized methods designed to recover nonlinear structure. Even so, we may not always be successful.
- *Key ideas*: Generalizing linear multivariate methods. Note that, these equivalences in the linear case do not always transfer to the nonlinear case. (tổng quát hóa các pp tuyến tính đa biến dù không luôn thành công).

Polynomial PCA

- How should we generalize PCA to the nonlinear case? One possibility is to transform the set of input variables using a quadratic, cubic, or higher degree polynomial, and then apply linear PCA.
- Focused on the *smallest* few eigenvalues for nonlinear dimensionality reduction.
- Quadratic PCA: the r-vector $\mathbf{X}$ is transformed into an extended r-vector $\mathbf{X}$, where $r = 2r + r(r - 1)/2$, $\mathbf{X} = (X_1, X_2) \rightarrow \mathbf{X}' = (X_1, X_2, X_1^2, X_2^2, X_1X_2)$
- Some problems inevitably arise when using quadratic PCA.
  - First, the variables in $\mathbf{X}$ will not be uniformly scaled, especially for large r, and so a standardization of all r variables may be desirable.
  - Second, the size of the extended vector $\mathbf{X}$ for quadratic PCA increases quickly with increasing r: when $r = 10$, $r = 65$, and when $r = 20$, $r = 230$.

Principal curves and surfaces

- Suppose $\mathbf{X}$ is a continuous random r-vector having density $p_{\mathbf{X}}$, zero mean, and finite second moments. Suppose further that the data observed on $\mathbf{X}$ lie close to a smooth nonlinear manifold of low dimension.
- A principal curve (Hastie, 1984; Hastie and Stuetzle, 1989) is a smooth one-dimensional parameterized curve $\mathbf{f}$ that passes through the "middle" of the data, regardless of whether the "middle" is a straight line or a nonlinear curve.
- A principal surface is a generalization of principal curve to a smooth two-(or higher-) dimensional curve.
- We use an analogue of least-squares optimality as the defining characteristic: we determine the principal curve or surface by minimizing the average of the squared distances between the data points and their projections onto that curve.

Content

Introduction

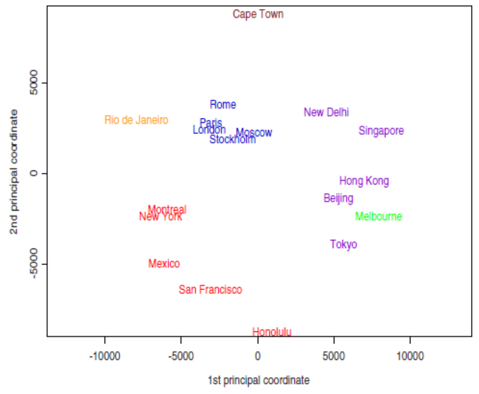
- A map including cities and towns → table where each cell shows the degree of "closeness" (or proximity, sự gần gũi) of a row city to a column city. Proximity could have different meanings: straight-line distance or as shortest traveling distance.
- Consider entities as objects, products, a nations, a stimulus, etc. and we can talk about proximity of any two entities as measures of association (e.g., absolute value of a correlation coefficient), confusion frequency (i.e., to what extend one entity is confused with another in an identification exercise), or measure of how alike (or how different), etc.
- Multidimensional scaling (MDS): [tái dựng bản đồ gốc nhiều chiều] given a table of proximities of entities, reconstruct the original map of entities as closely as possible.
- MDS is a family of different algorithms, each designed to arrive at an optimal low-dimensional configuration for a particular type of proximity data.

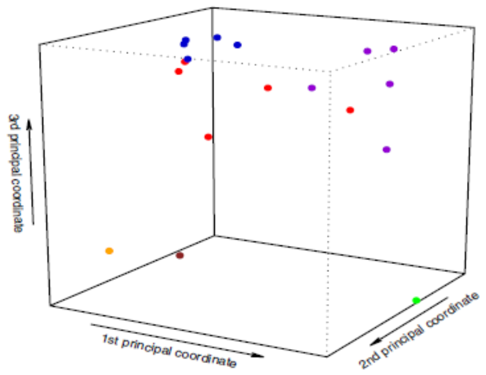
Introduction

Problem: Re-create the map that yielded the table of airline distances.

	Beijing	Cape Town	Hong Kong	Honolulu	London	Melbourne
Cape Town	12947					
Hong Kong	1972	11867				
Honolulu	8171	18562	8945			
London	8160	9635	9646	11653		
Melbourne	9093	10338	7392	8862	16902	
Mexico	12478	13703	14155	6098	8947	13557
Montreal	10490	12744	12462	7915	5240	16730
Moscow	5809	10101	7158	11342	2506	14418
New Delhi	3788	9284	3770	11930	6724	10192
New York	11012	12551	12984	7996	5586	16671
Paris	8236	9307	9650	11988	341	16793
Rio de Janeiro	17325	6075	17710	13343	9254	13227
Rome	8144	8417	9300	12936	1434	15987
San Francisco	9524	16487	11121	3857	8640	12644
Singapore	4465	9671	2575	10824	10860	6050
Stockholm	6725	10334	8243	11059	1436	15593
Tokyo	2104	14737	2893	6208	9585	8159

Airline distances (km) between 18 cities.
Source: Atlas of the World, Revised 6th Edition, National Geographic Society, 1995, p. 131





Two and three dimensional map of 18 world cities using the classical scaling algorithm on airline distances between those cities. The colors reflect the different continents: Asia (purple), North America (red), South America (orange), Europe (blue), Africa (brown), and Australasia (green).

Examples of MDS applications

- **Marketing:** Derive "product maps" of consumer choice and product preference (e.g., automobiles, beer) so that relationships between products can be discerned
- **Ecology:** Provide "environmental impact maps" of pollution (e.g., oil spills, sewage pollution, drilling-mud dispersal) on local communities of animals, marine species, and insects.
- **Molecular Biology:** Reconstruct the spatial structures of molecules (e.g., amino acids) using biomolecular conformation (3D structure). Interpret their interrelations, similarities, and differences. Construct a 3D "protein map" as a global view of the protein structure universe.
- **Social Networks:** Develop "telephone-call graphs," where the vertices are telephone numbers and the edges correspond to calls between them. Recognize instances of credit card fraud and network intrusion detection.

Proximity matrices

- The focus on pairwise comparisons of entities is fundamental to MDS.
- The “closeness” of two entities is measured by a proximity measure, defined in a number of different ways.
  - A *continuous measure* of how physically close one entity is to another or a *subjective judgment* recorded on an ordinal scale, but where the scale is well-calibrated as to be considered continuous.
  - In perception study, proximity is not quantitative but a subjective rating of similarity (or dissimilarity) recorded on a pair of entities.
- “Closeness” of one entity to another could be measured by a small or large value. Importance is a *monotonic relationship* between the “closeness” of two entities.

Proximity matrices

- Collection of n entities. Let δ_{ij} represent the dissimilarity of the ith entity to the jth entity. Consider m dissimilarities, $m = \binom{n}{2} = \frac{1}{2}n(n-1)$, and $(m \times m)$ proximity matrix

$$\boldsymbol{\Delta} = (\delta_{ij}) \tag{1}$$

- The proximity matrix is usually displayed as a lower-triangular array of nonnegative entries, with the understanding that the diagonal entries are all zeroes and the matrix is symmetric: for all $i, j = 1, \ldots, n$,

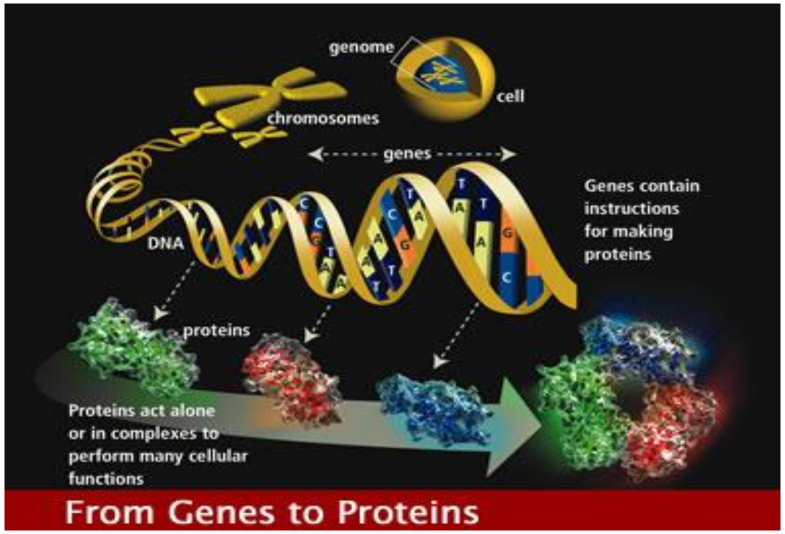
$$\delta_{ij} \geq 0, \delta_{ii} = 0, \delta_{ji} = \delta_{ij}$$

- In order for a dissimilarity measure to be regarded as a metric distance, we require that δ_{ij} satisfy the triangle inequality,

$$\delta_{ij} \leq \delta_{ik} + \delta_{kj} \quad \text{for all } k$$

Comparing protein sequences
Optimal sequence alignment

- About 100,000 different proteins in the human body.
- Proteins carry out important bodily functions: supporting cell structure, protecting against infection from bacteria and viruses, aiding movement, transporting materials (hemoglobin for oxygen), regulating control (enzymes, hormones, metabolism, insulin) of the body.
- "protein map": how existing protein families relate to one another, structurally and functionally.
- Might be able to predict the functions of newly discovered proteins from their spatial locations and proximities to other proteins, where we would expect neighboring proteins to have very similar biochemical properties.


genome
cell
chromosomes
genes
Genes contain instructions for making proteins
DNA
proteins
Proteins act alone or in complexes to perform many cellular functions
From Genes to Proteins

Comparing protein sequences
Optimal sequence alignment

- Key idea in computing the proximity of two proteins is amino acids can be *altered by random mutations (đột biến)* over a long period of evolution.
- Mutations can take various forms: deletion or insertion of amino acids, … For an evolving organism to survive, structure/functionality of the most important segments of its protein would have to be *preserved.*
- Compute a similarity value between two sequences that have different lengths and *different amino acid distributions*.
- *Trick*: Align the two sequences so that as many letters in one sequence can be “matched” with the corresponding letters in the other sequence. Several methods for *sequence alignment*:
  - *Global alignment* aligns all the letters in entire sequences assuming that the two sequences are very similar from beginning to end;
  - *Local alignment* assumes that the two sequences are highly similar only over short segments of letters.

Comparing protein sequences

Optimal sequence alignment

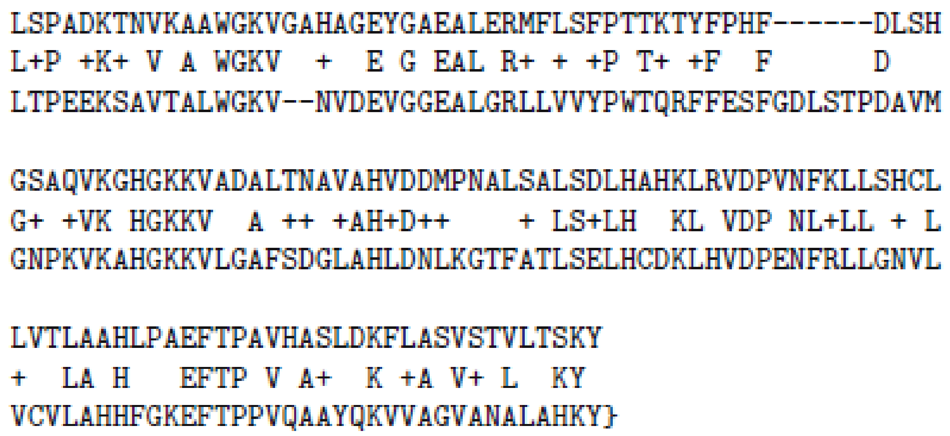
- Alignment methods use *dynamic programming* algorithms as the primary tool. BLAST and FASTA are popular tools for huge databases.
- An “optimal” alignment if maximizing an alignment score. For example, an *alignment score* is the sum of a number of terms, such as *identity* (high positive), *substitution* (positive, negative or 0).
- *Substitution score*: “cost” of replacing one amino acid (aa). Scores for all 210 possible aa pairs are collected to form a (20 × 20) *substitution matrix.* One popular is BLOSUM62 (BLOcks Substitution Matrix), assuming no more than 62% of letters in sequences are identical (Henikoff, 1996).
- A “gap” (*indel*), an empty space (“-”), penalizes an *insertion* or a *deletion* of an aa. Two types of gap penalties, starting a gap and extending the gap.
- *The alignment score s is the sum of the identity and substitution scores, minus the gap score.*

Comparing protein sequences

Example: Two hemoglobin chains

- Given *n* proteins, let s_{ij} be the alignment score between the *i*th and *j*th protein. We have $\delta_{ij} = s_{max} - s_{ij}$, where $s_{\max}$ is the largest alignment score among all pairs. The proximity matrix is then given by $\boldsymbol{\Delta} = (\delta_{ij})$.
- Compare the *hemoglobin alpha chain* protein HBA HUMAN having length 141 with the related *hemoglobin beta chain* protein HBB HUMAN having length 146.
- We would obtain different optimal alignments and alignment scores.

86 positive substitution scores (the 25 “+”s and the 61 identities). The alignment score is *s* = 259.



```
LSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF------DLSH
L+P +K+ V A WGKV  +  E G EAL R+ + +P T+ +F  F      D
LTPEEKSAVTALWGKV--NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVM

GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCL
G+ +VK HGKKV  A ++ +AH+D++    + LS+LH  KL VDP NL+LL + L
GNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVL

LVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKY
+  LA H   EFTP V A+  K +A V+ L  KY
VCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKY}
```

String matching
Edit distance

- In pattern matching, we study the problem of finding a given pattern within a body of text. If a pattern is a single string, the problem is called string matching, used extensively in text-processing applications.
- A popular numerical measure of the similarity between two strings is edit distance (also called *Levenshtein distance*).
- The usual definition of edit distance is the fewest number of editing operations (*insertions, deletions, substitutions*) which would be needed to transform one string into the other.
- An *insertion* inserts a letter into the sequence, a *deletion* deletes a letter from the sequence, and a *substitution* replaces one letter in the sequence by another letter. Identities (or matches) are not counted in the distance measure. Each editing operation can be assigned a cost.
- Used to construct a *phylogenetic tree* — a diagram laying out a possible evolutionary history — of a single protein.

Classical scaling and distance geometry

- Suppose we are given n points $\mathbf{X}_1, \ldots, \mathbf{X}_n \in \Re^r$. From these points, we compute an $(n \times n)$ proximity matrix $\mathbf{\Delta} = (\delta_{ij})$ of dissimilarities, where

$$\delta_{ij} = \|X_i - X_j\| = \{\textstyle\sum_{k=1}^{r}(X_{ik} - X_{jk})^2\}^{1/2} \qquad (2)$$

- Many kinds of distance can be considered: the *Minkowski* or L_p *distance* is given by

$$\delta_{ij} = \{\textstyle\sum_{k=1}^{r}|X_{ik} - X_{jk}|^p\}^{1/\mathrm{p}}$$

- $p = 1$: *city-block* or *Manhattan distance,*

- $p = 2$: Euclidean distance.

Classical scaling and distance geometry

From dissimilarities to principal coordinates

- From (2), $\sigma_{ij}^2 = \|X_i\|^2 + \|X_j\|^2 - 2X_i^\tau X_j$. Let $b_{ij} = X_i^\tau X_j = -\frac{1}{2}(\delta_{ij}^2 - \delta_{i0}^2 - \delta_{j0}^2)$ where $\delta_{i0}^2 = \|X_i\|^2$. We get $b_{ij} = a_{ij} - a_{i.} - a_{.j} + a_{ii}$

  where $a_{ij} = -\frac{1}{2}\delta_{ij}^2$, $a_{i.} = n^{-1}\sum_j a_{ij}^2$, $a_{.j} = n^{-1}\sum_i a_{ij}^2$, $a_{..} = n^{-2}\sum_i\sum_j a_{ij}^2$.
- If setting $\mathbf{A} = (a_{ij})$ the matrix of squared dissimilarities and $\mathbf{B} = (b_{ij})$, we have $\mathbf{B} = \mathbf{HAH}$ where $\mathbf{H} = \mathbf{I}_\mathrm{n} - \mathrm{n}^{-1}\mathbf{J}_\mathrm{n}$, $\mathbf{J}_n$ is square matrix of ones.
- Wish to find a *t* dimensional representation, $\mathbf{Y}_1, \ldots, \mathbf{Y}_\mathrm{n} \in \Re^\mathrm{t}$ (referred to as principal coordinates), of those r-dimensional points (with t < r), such that the interpoint distances in t-space "match" those in r-space.
- When dissimilarities are defined as Euclidean interpoint distances, this type of "classical" MDS is equivalent to PCA in that the principal coordinates are identical to the scores of the first *t* principal components of the $\{\mathbf{X}_\mathrm{i}\}$.

Classical scaling and distance geometry

From dissimilarities to principal coordinates

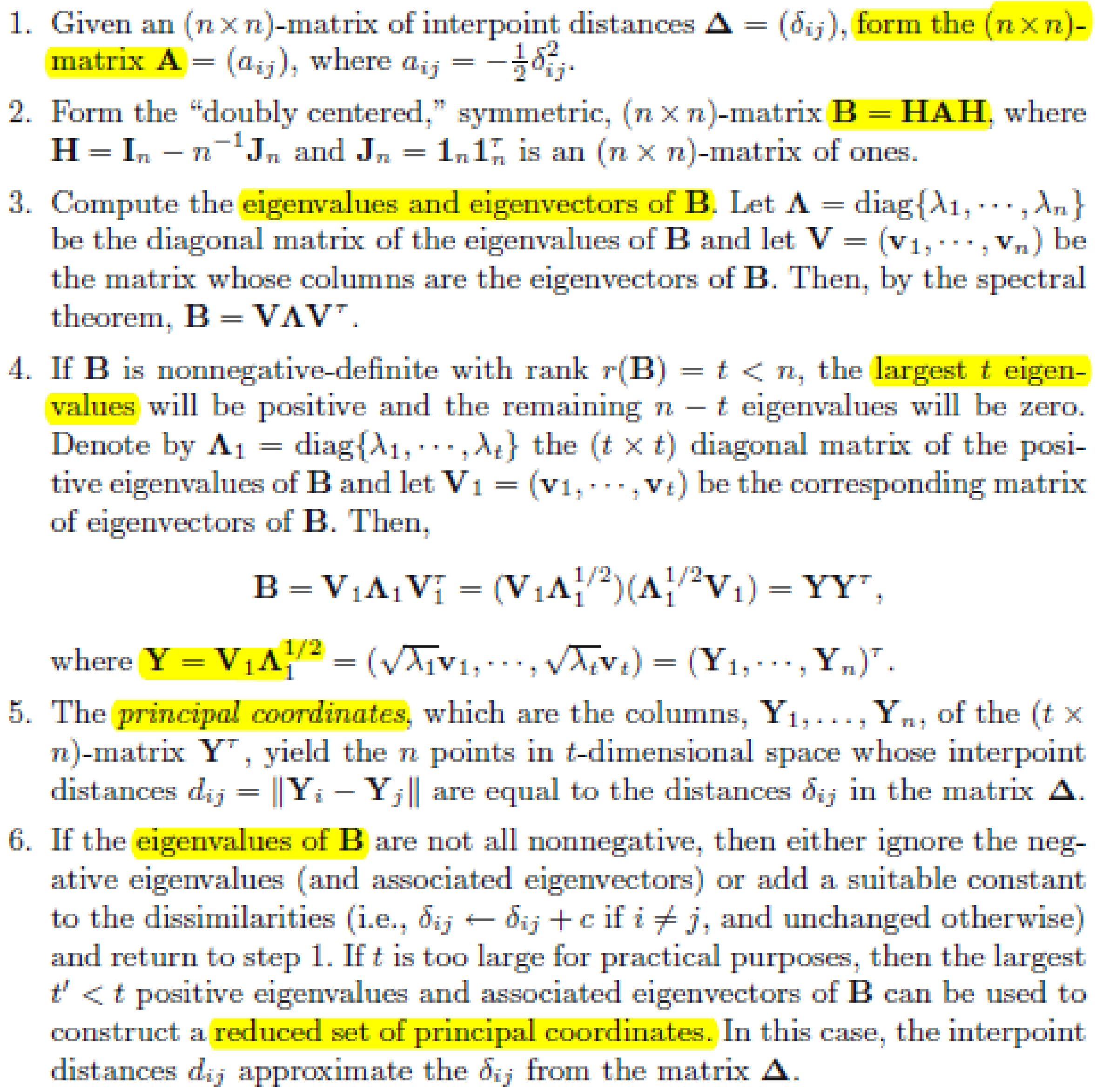
- Typically, in classical scaling (Torgerson, 1952, 1958) we are not given $\{\boldsymbol{X}_i\} \in \Re^t$; instead, we are given only the dissimilarities $\{\delta_{ij}\}$ through the $(n \times n)$ proximity matrix $\mathbf{\Delta}$. Using $\mathbf{\Delta}$, we form $\mathbf{A}$, and then $\mathbf{B}$.
- Motivation for classical scaling comes from a least-squares argument similar to the one employed for PCA.
- The classical scaling algorithm is based upon an eigendecomposition of the matrix $\mathbf{B}$. This eigendecomposition produces $\mathbf{Y}_1, \ldots, \mathbf{Y}_n \in \Re^t$, $t < r$, a configuration whose Euclidean interpoint distances,

$$d_{ij}^2 = \left\|Y_i - Y_j\right\|^2 = \left(Y_i - Y_j\right)^\tau (Y_i - Y_j)$$

- The solution of the classical scaling problem is not unique. A common orthogonal transformation of the points in the configuration found by classical scaling yields a different solution of the classical scaling problem.

Classical scaling

The classical scaling algorithm



1. Given an $(n \times n)$-matrix of interpoint distances $\mathbf{\Delta} = (\delta_{ij})$, form the $(n \times n)$-matrix $\mathbf{A} = (a_{ij})$, where $a_{ij} = -\frac{1}{2}\delta_{ij}^2$.
2. Form the "doubly centered," symmetric, $(n \times n)$-matrix $\mathbf{B} = \mathbf{HAH}$, where $\mathbf{H} = \mathbf{I}_n - n^{-1}\mathbf{J}_n$ and $\mathbf{J}_n = \mathbf{1}_n\mathbf{1}_n^\tau$ is an $(n \times n)$-matrix of ones.
3. Compute the eigenvalues and eigenvectors of $\mathbf{B}$. Let $\mathbf{\Lambda} = \mathrm{diag}\{\lambda_1, \cdots, \lambda_n\}$ be the diagonal matrix of the eigenvalues of $\mathbf{B}$ and let $\mathbf{V} = (\mathbf{v}_1, \cdots, \mathbf{v}_n)$ be the matrix whose columns are the eigenvectors of $\mathbf{B}$. Then, by the spectral theorem, $\mathbf{B} = \mathbf{V\Lambda V}^\tau$.
4. If $\mathbf{B}$ is nonnegative-definite with rank $r(\mathbf{B}) = t < n$, the largest t eigenvalues will be positive and the remaining $n - t$ eigenvalues will be zero. Denote by $\mathbf{\Lambda}_1 = \mathrm{diag}\{\lambda_1, \cdots, \lambda_t\}$ the $(t \times t)$ diagonal matrix of the positive eigenvalues of $\mathbf{B}$ and let $\mathbf{V}_1 = (\mathbf{v}_1, \cdots, \mathbf{v}_t)$ be the corresponding matrix of eigenvectors of $\mathbf{B}$. Then,
$$\mathbf{B} = \mathbf{V}_1\mathbf{\Lambda}_1\mathbf{V}_1^\tau = (\mathbf{V}_1\mathbf{\Lambda}_1^{1/2})(\mathbf{\Lambda}_1^{1/2}\mathbf{V}_1) = \mathbf{YY}^\tau,$$
where $\mathbf{Y} = \mathbf{V}_1\mathbf{\Lambda}_1^{1/2} = (\sqrt{\lambda_1}\mathbf{v}_1, \cdots, \sqrt{\lambda_t}\mathbf{v}_t) = (\mathbf{Y}_1, \cdots, \mathbf{Y}_n)^\tau$.
5. The *principal coordinates*, which are the columns, $\mathbf{Y}_1, \ldots, \mathbf{Y}_n$, of the $(t \times n)$-matrix $\mathbf{Y}^\tau$, yield the n points in t-dimensional space whose interpoint distances $d_{ij} = \|\mathbf{Y}_i - \mathbf{Y}_j\|$ are equal to the distances δ_{ij} in the matrix $\mathbf{\Delta}$.
6. If the eigenvalues of $\mathbf{B}$ are not all nonnegative, then either ignore the negative eigenvalues (and associated eigenvectors) or add a suitable constant to the dissimilarities (i.e., $\delta_{ij} \leftarrow \delta_{ij} + c$ if $i \neq j$, and unchanged otherwise) and return to step 1. If t is too large for practical purposes, then the largest $t' < t$ positive eigenvalues and associated eigenvectors of $\mathbf{B}$ can be used to construct a reduced set of principal coordinates. In this case, the interpoint distances d_{ij} approximate the δ_{ij} from the matrix $\mathbf{\Delta}$.

Classical scaling and distance geometry

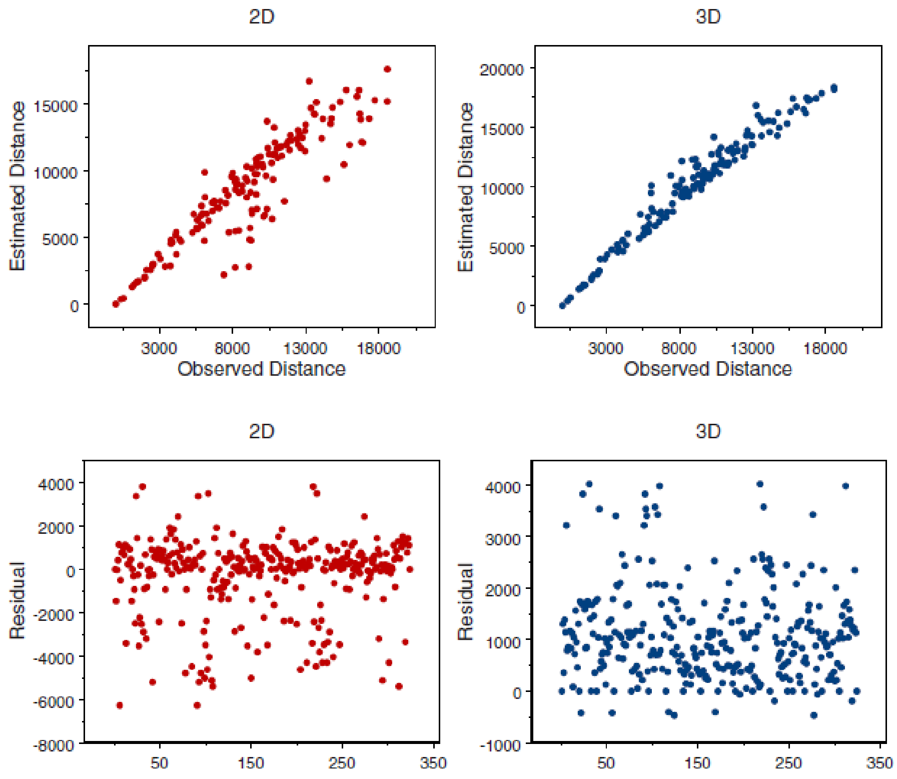
Airlines distances

Eigenvalues of **B** *and the eigenvectors corresponding to the first three largest eigenvalues (in red) for the airline distances example.*

	Eigenvalues	Eigenvectors		
1	471582511	0.245	–0.072	0.183
2	316824787	0.003	0.502	-0.347
3	253943687	0.323	–0.017	0.103
4	–98466163	0.044	–0.487	-0.080
5	–74912121	–0.145	0.144	0.205
6	–47505097	0.366	–0.128	–0.569
7	31736348	–0.281	–0.275	–0.174
8	–7508328	–0.272	–0.115	0.094
9	4338497	–0.010	0.134	0.202
10	1747583	0.209	0.195	0.110
11	–1498641	–0.292	–0.117	0.061
12	145113	–0.141	0.163	0.196
13	–102966	–0.364	0.172	–0.473
14	60477	–0.104	0.220	0.163
15	–6334	–0.140	–0.356	–0.009
16	–1362	0.375	0.139	–0.054
17	100	–0.074	0.112	0.215
18	0	0.260	–0.214	0.173

First three principal coordinates of the 18 cities in the airline distances example.

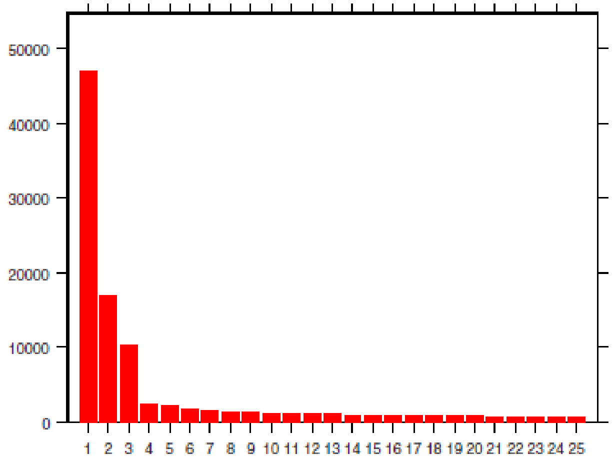
	Principal Coordinates		
City	1st	2nd	3rd
Beijing	5315.24	–1272.90	2920.75
Cape Town	57.63	8935.14	–5522.26
Hong Kong	7010.90	–306.52	1645.53
Honolulu	962.86	–8677.05	–1270.47
London	–3157.53	2557.96	3268.11
Melbourne	7948.29	–2283.67	–9062.28
Mexico	–6108.97	–4896.64	–2778.04
Montreal	–5912.57	–2039.70	1495.92
Moscow	–220.84	2377.27	3221.22
New Delhi	4528.94	3474.33	1751.50
New York	–6341.02	–2078.66	972.39
Paris	–3058.30	2910.08	3118.95
Rio de Janeiro	–7905.60	3067.34	–7537.69
Rome	–2262.26	3916.47	2595.85
San Francisco	–3041.92	–6341.23	–142.88
Singapore	8139.01	2470.83	–867.84
Stockholm	–1610.37	1997.61	3429.67
Tokyo	5656.51	–3810.66	2761.56



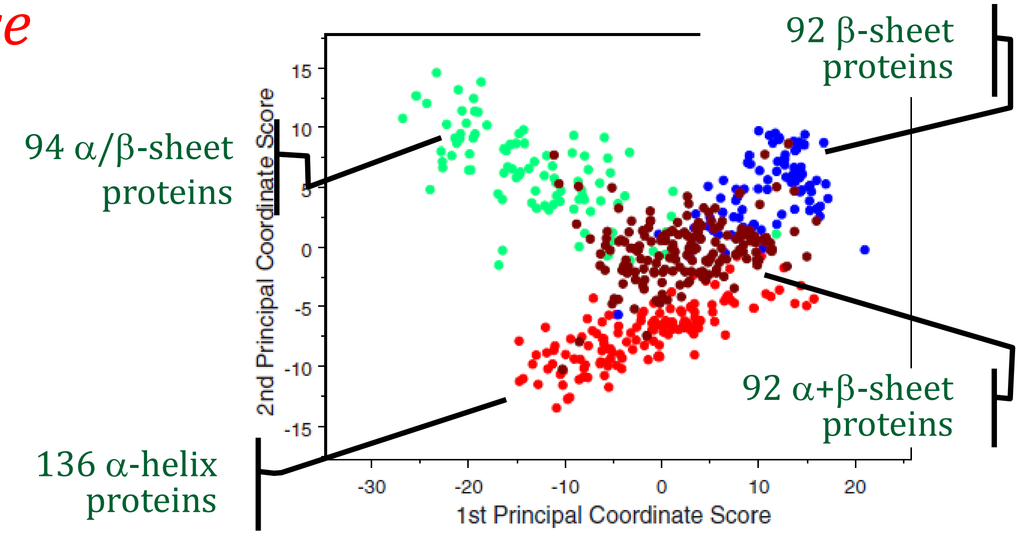
Estimated and observed airline distances. The lefnels show the 2D solution and the right panels show the 3D solution. The top panels show the estimated distances plotted against the observed distances, and the bottom panels show the residuals from the the fit (residual = estimated distance – observed distance) plotted against sequence number.

Classical scaling and distance geometry

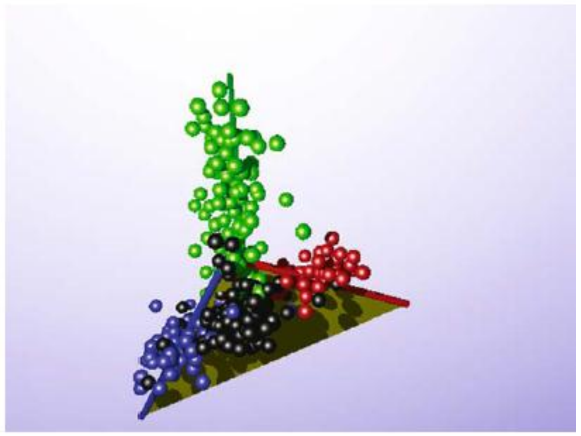
Mapping the protein universe



The first 25 ordered eigenvalues of **B** obtained from the classical scaling algorithm on 498 proteins.



Two-dimensional map of four protein classes using the classical scaling algorithm on 498 proteins.



A three-dimensional map of four protein classes using the classical scaling algorithm on 498 proteins.

- 498 proteins → classical scaling algorithm → largest 25 eigenvalues of **B** → first three eigenvalues are dominant → 3D configuration is probably most appropriate.
- 2D map and 3D map

Distance scaling

- Given *n* items (or entities) and their dissimilarity matrix s, $\boldsymbol{\Delta} = (\delta_{ij})$
- Classical scaling problem is to find a configuration of points in a lower-dimensional space such that the interpoint distances $\{d_{ij}\}$ satisfy $d_{ij} \approx \delta_{ij}$.
- In distance scaling, this relationship is relaxed; we wish to find a suitable configuration for which $d_{ij} \approx f(\delta_{ij})$ where f is some monotonic function.
- The function f transforms the dissimilarities into distances. The use of "metric" or "nonmetric" distance scaling depends upon the nature of the dissimilarities.
- If the dissimilarities are *quantitative we use metric distance scaling,* whereas if the dissimilarities are *qualitative we use nonmetric distance scaling*. In the MDS literature, metric distance scaling is traditionally called metric MDS, nonmetric distance scaling is called nonmetric MDS.

Summary

- **Introduction**: Given a table of proximities, reconstruct the original map as closely as possible and the space dimension.
- **Proximity Matrices**: $\boldsymbol{\Delta} = (\delta_{ij})$ measures "closeness" of pairwise objects, metric (triangle inequality)
- Comparing Protein Sequences
- **String Matching**: Edit distance
- **Classical Scaling**: require $d_{ij} \approx \delta_{ij}$
- **Distance Scaling:** $d_{ij} \approx f(\delta_{ij})$ where f is monotonic
  - **Metric Distance Scaling:** metric distance scaling
  - **Nonmetric Distance Scaling:** nonmetric distance scaling

Nonlinear manifold learning

Manifold?

- An ant at a picnic: The ant crawls all over the picnic items, as diminutive size, the ant sees everything on a very small scale as flat and featureless.
- A manifold can be thought of in similar terms, as a topological space that locally looks flat and featureless and behaves like Euclidean space.
- A manifold also satisfies certain topological conditions. A submanifold is just a manifold lying inside another manifold of higher dimension.

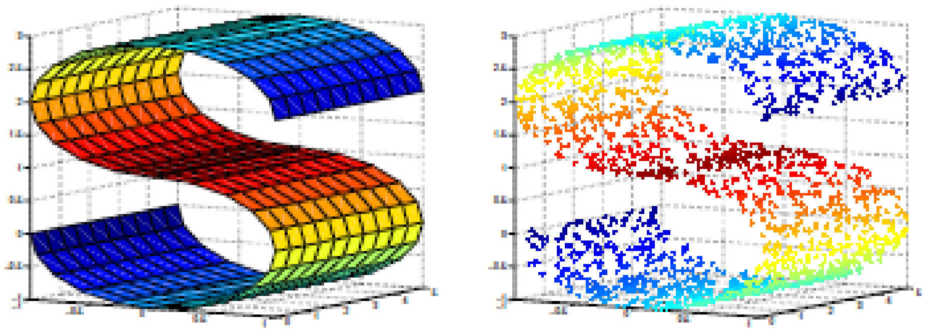


FIGURE 16.6. *Left panel: The S-curve, a two-dimensional S-shaped manifold embedded in three-dimensional space. Right panel: 2,000 data points randomly generated to lie on the surface of the S-shaped manifold.*

Nonlinear manifold learning

- Many exciting new algorithmic techniques: Isomap, Local Linear Embedding, Laplacian Eigenmap, and Hessian Eigenmap. They aim to *recover the full low-dimensional representation of an unknown nonlinear manifold M.*
- Having different philosophies for recovering nonlinear manifolds, each methods consists of a three-step approach.
  1) Incorporating *neighborhood information* from each data point to construct a weighted graph having the data points as vertices.
  2) Taking the weighted neighborhood graph and *transforming* it into suitable input for the embedding step.
  3) Computing an ($n \times n$)-eigenequation (*embedding step*).
- Manifold learning involves concepts from differential geometry: What means a *manifold* and what means to be *embedded* in a higher-dimensional space?

Nonlinear manifold learning

- If a topological manifold M is continuously differentiable to any order (i.e., $\mathcal{M} \in C^{\infty}$), we call it a *smooth* (or *differentiable*) *manifold*.
- *Riemannian manifold* $(\mathcal{M}, d^{\mathcal{M}})$ is smooth manifold $\mathcal{M}$ with a metric $d^{\mathcal{M}}$. We take $d^{\mathcal{M}}$ to be a manifold metric defined by

$$d^{\mathcal{M}}(\mathbf{y}, \mathbf{y}') = \inf_{c}\{\mathcal{L}(c) | \text{c is a curve in } \mathcal{M} \text{ which joins } \mathbf{y} \text{ and } \mathbf{y}'\},$$

  where **y**, **y'** $\in \mathcal{M}$ and $\mathcal{L}(c)$ is the arc-length of the curve c. Thus, $d^{\mathcal{M}}$ finds the shortest curve (or *geodesic, đo đạc*) between any two points on $\mathcal{M}$, and the arc-length of that curve is the *geodesic distance* between the points.
- **Data assumption**: Finitely many data points, $\{\boldsymbol{y}_i\}$, are randomly sampled from a smooth t-dimensional manifold $\mathcal{M}$ with metric given by geodesic distance.

Nonlinear manifold learning
ISOMAP

- The *isometric feature mapping* (or ISOMAP) algorithm (Tenenbaum et al., 2000) assumes that the smooth manifold M is a convex region of $\Re^t$ $(t \ll r)$ and that the embedding $\psi : \mathcal{M} \longrightarrow \mathcal{X}$ is an isometry.
- This assumption has two key ingredients:
  - *Isometry:* The geodesic distance is invariant under the map ψ. For any pair of points on the manifold, $\mathbf{y}, \mathbf{y}' \in \mathcal{M}$, the geodesic distance between those points equals the Euclidean distance between their corresponding coordinates, $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$; i.e.,
  $$d^{\mathcal{M}}(y, y') = \|x - x'\|\mathcal{X}$$
  where $\mathbf{y} = \varphi(\mathbf{x})$ and $\mathbf{y}' = \varphi(\mathbf{x}')$.
  - *Convexity:* The manifold $\mathcal{M}$ is a convex subset of $\Re^t$.
- ISOMAP regards $\mathcal{M}$ as a convex region that may have been distorted in any of a number of ways (e.g., by folding or twisting), e.g., Swiss roll.

isometry is a distance-preserving map between metric spaces.

Nonlinear manifold learning

Three steps of ISOMAP

1. *Neighborhood graph.*
   - Fix either an integer K or an $\epsilon > 0$. Calculate the distances,

     $$d_{ij}^{\mathcal{M}} = d^{\mathcal{X}}(x_i, x_j) = \|x_i - x_j\|\mathcal{X}$$

     between all pairs of data points $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}, i, j = 1, 2, \ldots, n$. *Determine which data points are "neighbors" on the manifold M by connecting each point either to its K nearest neighbors or to all points lying within a ball of radius ϵ of that point.* Choice of K or ϵ controls neighborhood size and also the success of ISOMAP.

   - We obtain *weighted neighborhood graph* $\mathcal{G} = \mathcal{G}(\mathcal{V}, \mathcal{E})$, where the set of *vertices* $\mathcal{V} = \{x_1, \ldots, x_n\}$ are the input data points, and the set of *edges* $\mathcal{E} = \{e_{ij}\}$ indicate neighborhood relationships between the points.

Nonlinear manifold learning

Three steps of ISOMAP

2. *Compute graph distances*

- Estimate the unknown true *geodesic distances*, $\{d_{ij}^{\mathcal{M}}\}$, between pairs of points in $\mathcal{M}$ by *graph distances*, $\{d_{ij}^{\mathcal{G}}\}$, with respect to the graph $\mathcal{G}$.

- *The graph distances are the shortest path distances between all pairs of points in the graph $\mathcal{G}$.* Points that are not neighbors of each other are connected by a sequence of neighbor-to-neighbor links.

- An efficient algorithm for computing the shortest path between every pair of vertices in a graph is *Floyd's algorithm* (Floyd, 1962), which works best with dense graphs (graphs with many edges).

Nonlinear manifold learning

Three steps of ISOMAP

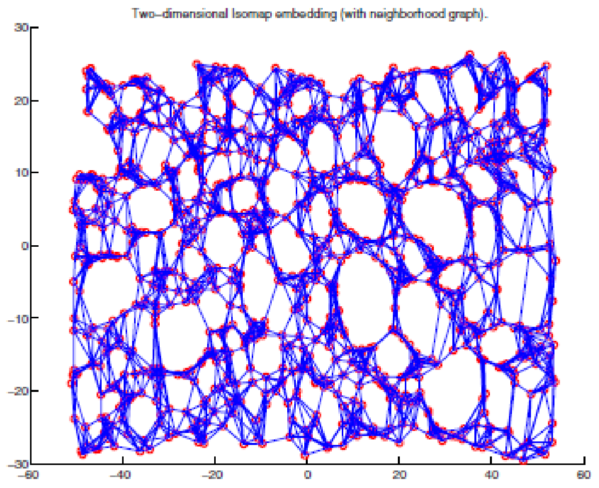
3. Embedding via multidimensional scaling

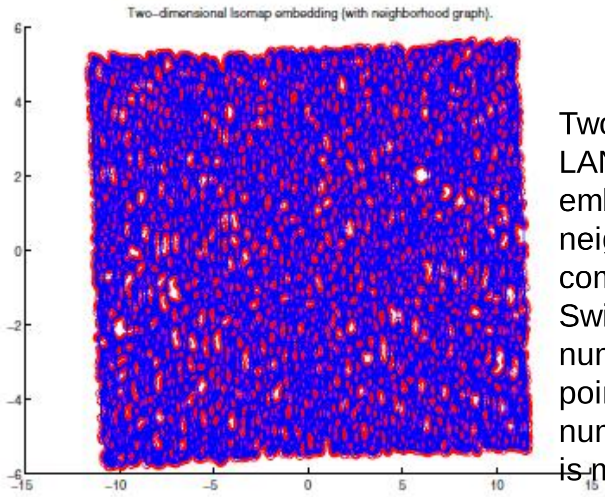
- Let $\boldsymbol{D}^{\mathcal{G}} = \left(d_{ij}^{\mathcal{G}}\right)$ be the symmetric $(n \times n)$-matrix of graph distances. Apply "classical" MDS to $\boldsymbol{D}^{\mathcal{G}}$ to give the reconstructed data points in a t-dimensional feature space $\mathcal{Y}$, so that the geodesic distances on M between data points is preserved as much as possible:
  - Form the "doubly centered," symmetric, $(n \times n)$-matrix of squared graph distances.
  - The embedding vectors $\{\hat{y}_i\}$ are chosen to minimize $\left\|\boldsymbol{A}_n^{\mathcal{G}} - \boldsymbol{A}_n^{\mathcal{Y}}\right\|$ where $\mathbf{A}_n^{\mathcal{G}} = -\frac{1}{2}\mathbf{H}\mathbf{S}^{\mathcal{G}}\mathbf{H}$ and $\mathbf{A}_n^{\mathcal{Y}} = -\frac{1}{2}\mathbf{H}\left(\left[d_{ij}^{\mathcal{Y}}\right]\right)^2 \mathbf{H}$ and $d_{ij}^{\mathcal{Y}} = \left\|y_i - y_j\right\|$.
  - The graph $\mathcal{G}$ is embedded into $\mathcal{Y}$ by the $(t \times n)$-matrix

$$\widehat{\mathbf{Y}} = (\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_n) = \left(\sqrt{\lambda_1}v_1, \dots, \sqrt{\lambda_t}v_t\right)^{\tau}$$

Nonlinear manifold learning

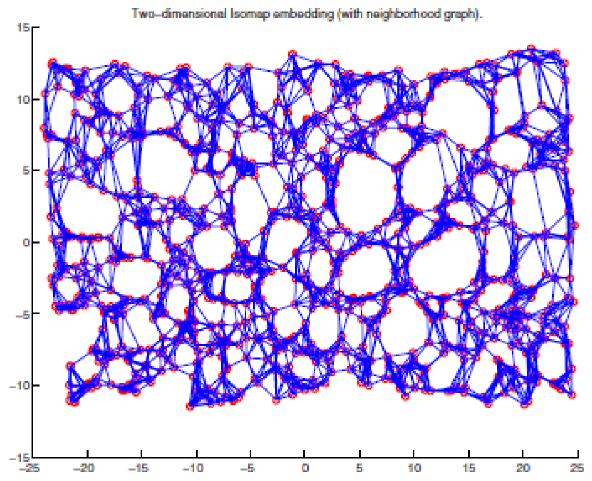
Examples





Two-dimensional LANDMARK ISOMAP embedding, with neighborhood graph, of the complete set of n = 20,000 Swiss-Roll data points. The number of neighborhood points is K = 7, and the number of landmark points is m = 50.

Two-dimensional ISOMAP embedding, with neighborhood graph, of the n = 1,000 Swiss roll data points. The number of neighborhood points is K = 7.



Two-dimensional LANDMARK ISOMAP embedding, with neighborhood graph, of the n = 1,000 Swiss roll data points. The number of neighborhood points is K = 7 and the number of landmark points is m = 50.