



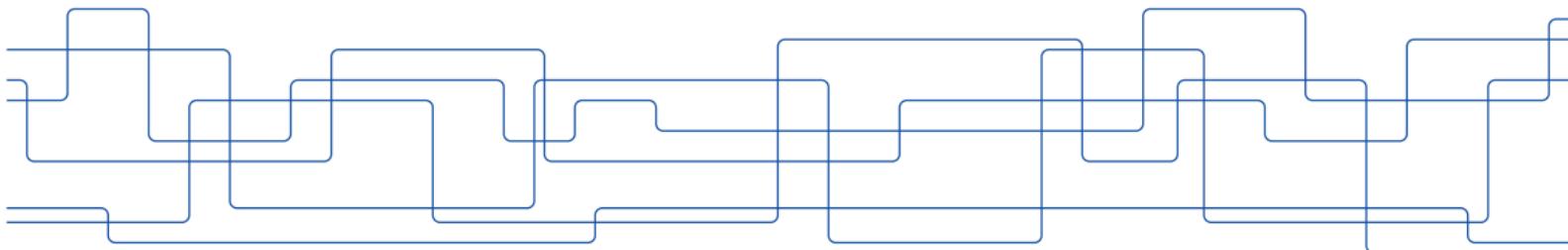
DD2434 Machine Learning, Advanced Course

Module 2 : high-dimensional data and dimensionality reduction

Aristides Gionis

argioni@kth.se

KTH Royal Institute of Technology



overview of module 2

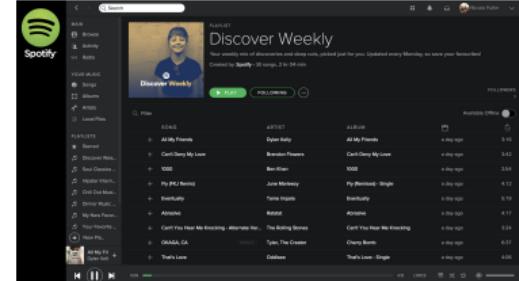
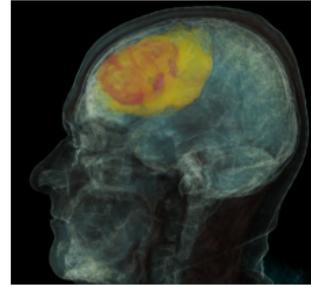
- ▶ high-dimensional data in machine learning
- ▶ basic principles of dimensionality reduction
- ▶ singular value decomposition (SVD) and eigen-decomposition
- ▶ principal component analysis (PCA)
- ▶ multidimensional scaling (MDS)
- ▶ isomap

reading material

- ▶ Lee and Verleysen. Nonlinear dimensionality reduction. Springer, 2007
 - chapters 1, 2, and 4
 - available online in the KTH-library website

machine learning today

- at the heart of an ongoing technological revolution and societal transformation



consider:

- autonomous vehicles
- healthcare (medical imaging, diagnostics, drug design, treatment, etc.)
- recommendation systems (netflix, amazon, etc.)
- searching for information in the web or social media
- intelligent interfaces (personal assistants, face recognition, speech recognition, etc.)
- ... and many more ...

data is at the center of machine learning

- ▶ data is collected everywhere nowadays
 - in human activities (mobile devices, traffic, sensors, surveillance data, etc.)
 - in science (climate, geological sensing, tracking of animals, biological data, etc.)
 - in industrial processes (monitoring of processes, products, materials)
- ▶ data can be used to
 - visualize and gain insights
 - acquire knowledge and make new discoveries
 - validate scientific hypotheses
 - build models, make predictions, optimize processes

data types

- ▶ many different types of data
 - relational data
 - text data
 - time series
 - sequential data
 - spatio-temporal data
 - graphs / networks

a very common data representation

- ▶ data is a set of **observations**; each observation consists of values over a set of **attributes**
 - observations are also called **points**, or **samples**
 - attributes are also called **dimensions**, or **variables**, or **features**
- ▶ such data is represented by a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$,
where n is the number of points and d the number of dimensions

$$\mathbf{Y} = \begin{pmatrix} y_{11} & y_{12} & \dots & y_{1d} \\ y_{21} & y_{22} & \dots & y_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n1} & y_{n2} & \dots & y_{nd} \end{pmatrix}$$

- ▶ sometimes attributes correspond to rows and observations to columns; we make it explicit
- ▶ very often data values are not numerical, e.g., categorical
 - it often pays off to transform them to numerical
- ▶ in some cases the data values are binary; such data are called transactional

examples of data that can be represented as a matrix

- ▶ text data : points = documents, dimensions = words

y_{ij} = {number of times that document i contains word j }

- ▶ movie-ratings data : points = users, dimensions = movies

y_{ij} = {the rating of user i for movie j }

- ▶ purchase data : points = customers, dimensions = products

$y_{ij} = 1$, if customer i has purchased product j , 0 otherwise (binary)

- ▶ stock-market time-series data : points = traded commodities, dimensions = time

y_{ij} = {the price of commodity i at time j }

(note that time imposes a natural order on the dimensions)

- ▶ social networks : points = individuals, dimensions = individuals

$y_{ij} = 1$, if individuals i and j are friends, 0 otherwise (binary)

(note that points and dimensions are the same set of individuals)

data dimensionality can be very large

- ▶ *dimensionality* = the number of dimensions of the data
- ▶ consider the previous examples
 - tens/hundreds of thousands, or millions, of words, movies, products, individuals, ...
- ▶ working with **high-dimensional data** can be **very challenging**
- ▶ **the curse of dimensionality**
 - phenomena/challenges that appear when working with high-dimensional data

the curse of dimensionality

a term that refers collectively to a number of different phenomena and challenges

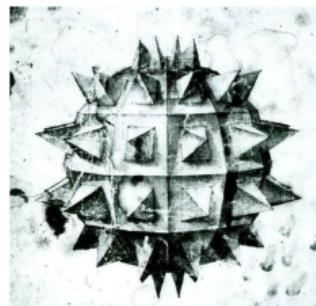
- ▶ the “host space” of the data increases exponentially with the dimension
 - as a result, high-dimensional data tend to be very sparse
 - also, optimization by exhaustive enumeration is not possible
- ▶ certain computational tasks become very challenging
 - e.g., for nearest-neighbor search, it is difficult to do better than exhaustive search
- ▶ a large number of dimensions may be irrelevant
 - e.g., may not have any predictive power in a classification task
- ▶ visualizing high-dimensional data is difficult
- ▶ the geometry of high-dimensional data becomes not intuitive

surprising properties of high-dimensional data

1. “spiky” (but convex) hypercubes

- ▶ volume of d -dimensional sphere: $V_s(r) = \pi^{d/2} r^d / \Gamma(1 + d/2)$
- ▶ volume of d -dimensional cube: $V_c(r) = (2r)^d$
- ▶ it follows

$$\lim_{d \rightarrow \infty} \frac{V_s(r)}{V_c(r)} = 0$$



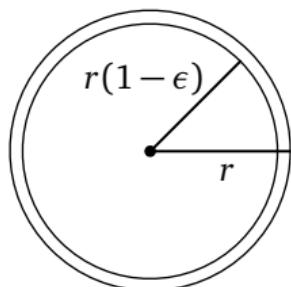
- ▶ an exponential number of cube corners “go out” from the sphere
these corners occupy all the available volume
- ▶ e.g., for $r = 1/2$, $V_c(1/2) = 1$, but $\lim_{d \rightarrow \infty} V_s(1/2) = 0$
so, the volume of the sphere vanishes, when dimensionality increases

surprising properties of high-dimensional data

2. volume of a thin spherical shell

- ▶ consider the relative hyper-volume of a thin spherical shell

$$\frac{V_s(r) - V_s((1-\epsilon)r)}{V_s(r)} = \frac{1^d - (1-\epsilon)^d}{1^d} \xrightarrow{d \rightarrow \infty} 1$$



so, when dimensionality increases, the thin shell contains almost all the volume

surprising properties of high-dimensional data

3. diagonal of a hypercube

- ▶ consider a diagonal of the hypercube $\mathbf{v} = (\pm 1, \dots, \pm 1)^T$
- ▶ and the i -th coordinate axis vector $\mathbf{e}_i = (0, \dots, 0, 1, 0, \dots, 0)^T$
- ▶ the cosine of their angle is

$$\cos\langle \mathbf{v}, \mathbf{e}_i \rangle = \frac{\mathbf{v}^T \mathbf{e}_i}{\|\mathbf{v}\| \|\mathbf{e}_i\|} = \frac{\pm 1}{\sqrt{d}} \xrightarrow{d \rightarrow \infty} 0$$

so, diagonals are nearly orthogonal to all coordinates axes

- ▶ we can also show : a random vector is nearly orthogonal to all coordinates axes

surprising properties of high-dimensional data

4. properties of the unit sphere

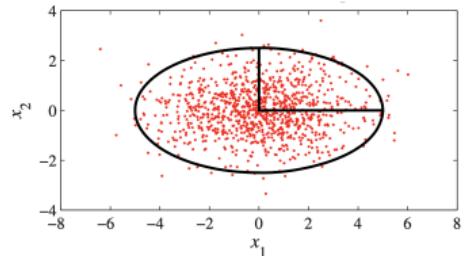
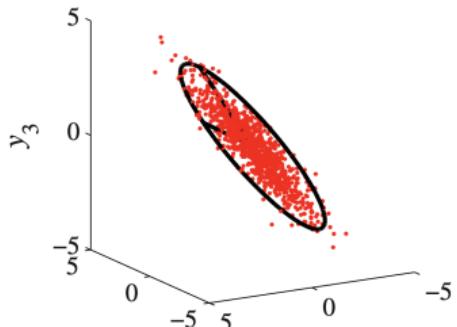
- ▶ most vectors lie on the surface of the sphere
- ▶ most vectors lie on the “equator”
- ▶ most vectors are nearly orthogonal
- ▶ most vectors have almost the same distance

how to cope with high-dimensional data?

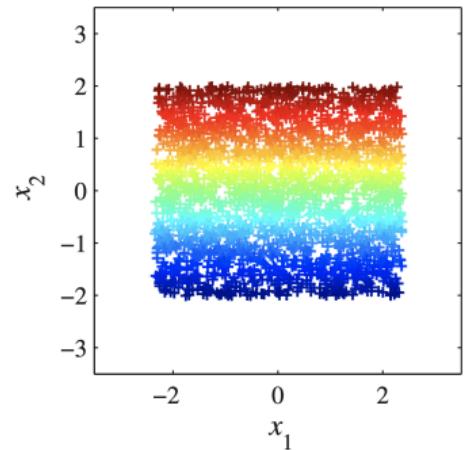
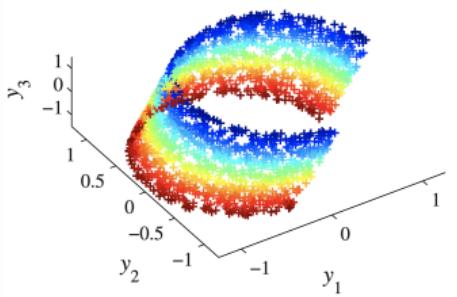
- ▶ quantify relevance of dimensions (variables), and possibly eliminate irrelevant dimensions
 - a.k.a., feature selection
 - the idea is applied mainly to supervised learning, where we can measure the correlation of variables (or subsets of variables) with the target
- ▶ explore correlations between dimensions
 - often pairs (or larger sets) of dimensions are correlated
 - e.g., a subset of dimensions can be explained by a single phenomenon which can be quantified by a latent dimension
 - in such a case we may want to keep only one of the correlated dimensions,
 - or we may want to transform the correlated dimensions into a new (smaller) set of dimensions, which reveal the latent dimensions
 - implementation of these ideas is known as **dimensionality reduction**

data may reside in lower-dimensional manifolds

linear
manifolds

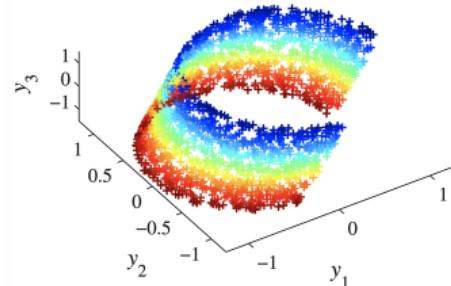
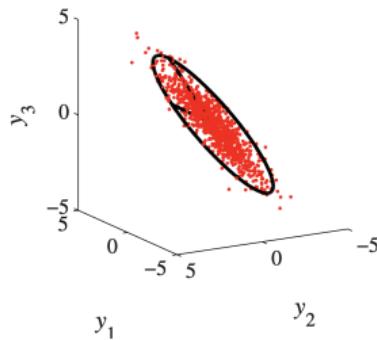


non-linear
manifolds



data may reside in lower-dimensional manifolds

- ▶ dimensionality reduction can be used to reveal hidden manifolds, and represent the data in the latent manifold dimensionality
- ▶ motivated by the idea that lower-dimensional manifolds can be either linear or non-linear, we have developed methods for **linear dimensionality reduction** or **non-linear dimensionality reduction**



objectives of dimensionality reduction

- ▶ dealing with high-dimensional data is challenging

thus, we need to develop methods to

- ▶ embed data in order to reduce their dimensionality
- ▶ embed data in order to recover (a potentiall smaller number of) latent variables
- ▶ estimate the number of latent variables

a high-level view of dimensionality reduction

- ▶ a data point is represented by a vector $\mathbf{y} = (y_1, \dots, y_d)^T \in \mathbb{R}^d$
the data is a set of data points $\mathcal{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\}$
equivalently, the data is represented by a matrix $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_n) \in \mathbb{R}^{d \times n}$
(notice that rows represent dimensions and columns represent points)
- ▶ the goal of dimensionality reduction is to map the data from \mathbb{R}^d to \mathbb{R}^k , with $k \ll d$
 - thus, we want to find a mapping

$$\text{cod} : \mathbb{R}^d \rightarrow \mathbb{R}^k, \quad \mathbf{y} \mapsto \mathbf{x} = \text{cod}(\mathbf{y})$$

- ▶ the inverse mapping should bring the mapped data close to the original data
 - to reason about this, we use the inverse mapping

$$\text{dec} : \mathbb{R}^k \rightarrow \mathbb{R}^d, \quad \mathbf{x} \mapsto \mathbf{y} = \text{dec}(\mathbf{x})$$

linear vs. non-linear dimensionality reduction

- ▶ a linear dimensionality reduction method uses linear mappings `cod` and `dec`
- ▶ a non-linear dimensionality reduction method uses non-linear mappings `cod` and `dec`

dimensionality reduction criterion

- ▶ to formalize the dimensionality-reduction problem, we need to define an optimization criterion, which captures our intuition about the quality of the dimensionality-reduction mapping
- ▶ a commonly-used criterion is **mean square error**

$$E_{\text{cod}, \text{dec}} = \mathbb{E}_{\mathbf{y}} [\|\mathbf{y} - \text{dec}(\text{cod}(\mathbf{y}))\|_2^2]$$

- ▶ other criteria exist, e.g.,
 - maximize the variance of the data on the embedded space
 - make the latent variables independent
 - ...

dimensionality reduction — general scheme

a dimensionality-reduction method has the following components

- ▶ a **model** to specify our data type and the mappings `cod` and `dec`
- ▶ a **criterion** to be optimized
- ▶ an **algorithm** to be used to optimize the criterion

the goal is to find optimal mappings `cod` and `dec`, within the specified model,
which optimize the desired criterion

categorization of dimensionality-reduction methods

- ▶ linear vs. non-linear model
- ▶ continuous vs. discrete model
- ▶ integrated vs. external estimation of the dimensionality
- ▶ layered vs. standalone embedding
- ▶ batch vs. online algorithm
- ▶ exact vs. approximate optimization

matrix decompositions

- ▶ we wish to **decompose** a matrix \mathbf{A} by writing it as a product of two or more matrices:

$$\mathbf{A}_{m \times n} = \mathbf{B}_{m \times k} \mathbf{C}_{k \times n} \quad \text{or} \quad \mathbf{A}_{m \times n} = \mathbf{B}_{m \times k} \mathbf{C}_{k \times r} \mathbf{D}_{r \times n}$$

ideally k and r are much smaller than m and n

- ▶ such a decomposition can yield insights about the nature of matrix \mathbf{A}
or, it can be useful to solve some problem at hand

the singular value decomposition (SVD)

- ▶ theorem : any $m \times n$ matrix \mathbf{A} , with $m \geq n$, can be factorized into

$$\mathbf{A} = \mathbf{U} \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} \mathbf{V}^T$$

where $\mathbf{U} \in \mathbb{R}^{m \times m}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$ are orthonormal (i.e., $\mathbf{U}^T \mathbf{U} = \mathbf{I}_{m \times m}$ and $\mathbf{V}^T \mathbf{V} = \mathbf{I}_{n \times n}$) and $\Sigma \in \mathbb{R}^{n \times n}$ is diagonal

$$\Sigma = \text{diag}(\sigma_1, \dots, \sigma_n), \quad \text{where } \sigma_1 \geq \dots \geq \sigma_n \geq 0$$

- ▶ “skinny” version of SVD : $\mathbf{A} = \mathbf{U}_1 \Sigma \mathbf{V}^T$, where $\mathbf{U}_1 \in \mathbb{R}^{m \times n}$

singular values and singular vectors

- ▶ the diagonal elements σ_j of Σ are the **singular values** of the matrix \mathbf{A}
- ▶ the columns of \mathbf{U} and \mathbf{V} are the **left singular vectors** and **right singular vectors**, resp.
- ▶ equivalent form of SVD

$$\mathbf{A} \mathbf{v}_j = \sigma_j \mathbf{u}_j \quad \text{and} \quad \mathbf{A}^T \mathbf{u}_j = \sigma_j \mathbf{v}_j$$

- ▶ outer-product form

$$\mathbf{A} = \mathbf{U}_1 \Sigma \mathbf{V}^T = (\mathbf{u}_1 \dots \mathbf{u}_n) \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{pmatrix} (\mathbf{v}_1 \dots \mathbf{v}_n)^T = \sum_{j=1}^n \sigma_j \mathbf{u}_j \mathbf{v}_j^T$$

which is a sum of rank-one matrices (each term $\sigma_j \mathbf{u}_j \mathbf{v}_j^T$ is a rank-one matrix)

matrix approximation using the singular-value decomposition

- ▶ theorem : let $\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$ be the singular-value decomposition of \mathbf{A}

let $\mathbf{U}_k = (\mathbf{u}_1 \dots \mathbf{u}_k)$, $\mathbf{V}_k = (\mathbf{v}_1 \dots \mathbf{v}_k)$, $\Sigma_k = \text{diag}(\sigma_1 \dots \sigma_k)$, and define

$$\mathbf{A}_k = \mathbf{U}_k \Sigma_k \mathbf{V}_k^T$$

then,

$$\min_{\text{rank}(\mathbf{B}) \leq k} \|\mathbf{A} - \mathbf{B}\|_2 = \|\mathbf{A} - \mathbf{A}_k\|_2 = \sigma_{k+1}$$

- ▶ in other words, \mathbf{A}_k is the best of rank- k approximation for the matrix \mathbf{A}
- ▶ useful for
 - compression
 - noise reduction
 - and as we will see, dimensionality-reduction

singular-value decomposition and matrix pseudo-inverse

- ▶ the Moore–Penrose inverse, or pseudo-inverse, of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, is a matrix $\mathbf{A}^+ \in \mathbb{R}^{n \times m}$ that generalizes the notion of inverse
- ▶ if \mathbf{A} has linearly independent columns (and thus, the matrix $\mathbf{A}^T \mathbf{A}$ is invertible), then \mathbf{A}^+ is computed by $\mathbf{A}^+ = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T$
in this case the pseudo-inverse is called left inverse and $\mathbf{A}^+ \mathbf{A} = \mathbf{I}$
- ▶ if $\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$ is the singular-value decomposition of \mathbf{A} , then

$$\mathbf{A}^+ = \mathbf{V} \Sigma^+ \mathbf{U}^T$$

where, Σ^+ is formed from Σ by taking the reciprocal of all the non-zero elements, leaving all the zeros in place, and making the matrix the right shape

eigen-decomposition

- ▶ let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a square matrix
- ▶ for a non-zero vector \mathbf{v} , the pair (λ, \mathbf{v}) is an eigenvalue-eigenvector pair, if

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

- ▶ an $n \times n$ matrix \mathbf{A} has n eigenvalue-eigenvector pairs
- ▶ the eigen-decomposition of matrix \mathbf{A} is

$$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^{-1}$$

where \mathbf{Q} is an $n \times n$ matrix whose i -th column is the eigenvector \mathbf{v}_i of \mathbf{A} , and Λ is a diagonal matrix whose diagonal elements are the eigenvalues, $\Lambda_{ii} = \lambda_i$

- ▶ a symmetric matrix has real eigenvalues and orthogonal eigenvectors ($\mathbf{Q}^{-1} = \mathbf{Q}^T$)
in addition, a symmetric positive semidefinite matrix has non-negative eigenvalues

singular value decomposition and eigen-decomposition

- ▶ from the singular value decomposition we can write

$$\mathbf{A}^T \mathbf{A} = \mathbf{V} \boldsymbol{\Sigma} \mathbf{U}^T \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^T = \mathbf{V} \boldsymbol{\Sigma}^2 \mathbf{V}^T$$

$$\mathbf{A} \mathbf{A}^T = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^T \mathbf{V} \boldsymbol{\Sigma} \mathbf{U}^T = \mathbf{U} \boldsymbol{\Sigma}^2 \mathbf{U}^T$$

- ▶ it follows :
 - the singular values of \mathbf{A} are the nonnegative square roots of the eigenvalues of $\mathbf{A}^T \mathbf{A}$
 - the columns of \mathbf{V} (right singular vectors of \mathbf{A}) are the eigenvectors of $\mathbf{A}^T \mathbf{A}$
 - the columns of \mathbf{U} (left singular vectors of \mathbf{A}) are the eigenvectors of $\mathbf{A} \mathbf{A}^T$

principal component analysis (PCA)

- ▶ one of the most common dimensionality-reduction techniques
- ▶ one of the simplest and most robust
- ▶ one of oldest methods, dating back to 1901 [Pearson, 1901]
- ▶ has been rediscovered many times
- ▶ also known as Karhunen-Loëve transformation, or Hotelling transformation

recall our general dimensionality-reduction scheme

we need to specify

- ▶ a **model** to define mappings between \mathbb{R}^d and \mathbb{R}^k
- ▶ a **criterion** to be optimized
- ▶ an **algorithm** to be used to optimize the criterion

recall our general dimensionality-reduction scheme

for PCA we use

- ▶ a **model** to define mappings between \mathbb{R}^d and \mathbb{R}^k
 - a linear mapping
- ▶ a **criterion** to be optimized
 - mean square error
- ▶ an **algorithm** to be used to optimize the criterion
 - singular-value decomposition

note: employing SVD is not a free choice, it is a consequence of the other two choices

data model of PCA

- ▶ a data point is represented by a vector $\mathbf{y} = (y_1, \dots, y_d)^T \in \mathbb{R}^d$
the data is a set of data points $\mathcal{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\}$
equivalently, the data is represented by a matrix $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_n) \in \mathbb{R}^{d \times n}$
(notice that rows represent dimensions and columns represent points)
- ▶ we assume that each vector $\mathbf{y} = (y_1, \dots, y_d)^T$ is the result of applying a linear transformation $\mathbf{W} \in \mathbb{R}^{d \times k}$ to a latent vector $\mathbf{x} = (x_1, \dots, x_k)^T$

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

- ▶ additionally, we assume that the columns of \mathbf{W} are orthonormal, i.e., $\mathbf{W}^T\mathbf{W} = \mathbf{I}_{k \times k}$
that is, \mathbf{W} is full rank and its columns are a basis of the space that it spans
(but $\mathbf{W}\mathbf{W}^T$ is not necessarily equal to $\mathbf{I}_{d \times d}$)

data centering

- we assume that the observed points \mathbf{y} and the latent points \mathbf{x} are “centered”

$$\mathbb{E}_{\mathbf{y}}[\mathbf{y}] = \mathbf{0}_d \quad \text{and} \quad \mathbb{E}_{\mathbf{x}}[\mathbf{x}] = \mathbf{0}_k$$

this can be achieved by removing the expectation $\mathbb{E}_{\mathbf{y}}[\mathbf{y}]$ from each point \mathbf{y}_i

$$\mathbf{y}_i \leftarrow \mathbf{y}_i - \mathbb{E}_{\mathbf{y}}[\mathbf{y}]$$

in practice, the expectation $\mathbb{E}_{\mathbf{y}}[\mathbf{y}]$ is computed by the sample mean

$$\mathbb{E}_{\mathbf{y}}[\mathbf{y}] \approx \frac{1}{n} \sum_{i=1}^n \mathbf{y}_i = \frac{1}{n} \mathbf{Y} \mathbf{1}_n$$

- data centering in matrix notation

$$\mathbf{Y} \leftarrow \mathbf{Y} - \frac{1}{n} \mathbf{Y} \mathbf{1}_n \mathbf{1}_n^T$$

PCA linear mapping

- ▶ the linear transformation $\mathbf{y} = \mathbf{W}\mathbf{x}$ provides the mapping dec from \mathbb{R}^k to \mathbb{R}^d

$$\text{dec} : \mathbb{R}^k \rightarrow \mathbb{R}^d, \quad \mathbf{x} \mapsto \mathbf{y} = \text{dec}(\mathbf{x}) = \mathbf{W}\mathbf{x}$$

- ▶ for the mapping from \mathbb{R}^d to \mathbb{R}^k we use the pseudo inverse $\mathbf{W}^+ = (\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T = \mathbf{W}^T$

$$\text{cod} : \mathbb{R}^d \rightarrow \mathbb{R}^k, \quad \mathbf{y} \mapsto \mathbf{x} = \text{cod}(\mathbf{y}) = \mathbf{W}^+\mathbf{y} = \mathbf{W}^T\mathbf{y}$$

PCA optimization criterion

- ▶ mean square error = reconstruction error

$$E_{\mathbf{W}} = \mathbb{E}_{\mathbf{y}} [\|\mathbf{y} - \text{dec}(\text{cod}(\mathbf{y}))\|_2^2] = \mathbb{E}_{\mathbf{y}} [\|\mathbf{y} - \mathbf{W}\mathbf{W}^T \mathbf{y}\|_2^2]$$

recall from previous slide that $\text{cod}(\mathbf{y}) = \mathbf{W}^T \mathbf{y}$ and $\text{dec}(\mathbf{x}) = \mathbf{W} \mathbf{x}$

recall that while $\mathbf{W}^T \mathbf{W} = \mathbf{I}_{k \times k}$, matrix $\mathbf{W} \mathbf{W}^T$ is not necessarily equal to $\mathbf{I}_{d \times d}$

optimizing the PCA reconstruction-error criterion

$$\begin{aligned}E_{\mathbf{W}} &= \mathbb{E}_{\mathbf{y}} \left[\|\mathbf{y} - \mathbf{W} \mathbf{W}^T \mathbf{y}\|_2^2 \right] \\&= \mathbb{E}_{\mathbf{y}} \left[(\mathbf{y} - \mathbf{W} \mathbf{W}^T \mathbf{y})^T (\mathbf{y} - \mathbf{W} \mathbf{W}^T \mathbf{y}) \right] \\&= \mathbb{E}_{\mathbf{y}} \left[\mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{W} \mathbf{W}^T \mathbf{y} + \mathbf{y}^T \mathbf{W} \mathbf{W}^T \mathbf{W} \mathbf{W}^T \mathbf{y} \right] \\&= \mathbb{E}_{\mathbf{y}} \left[\mathbf{y}^T \mathbf{y} - \mathbf{y}^T \mathbf{W} \mathbf{W}^T \mathbf{y} \right] \\&= \mathbb{E}_{\mathbf{y}} \left[\mathbf{y}^T \mathbf{y} \right] - \mathbb{E}_{\mathbf{y}} \left[\mathbf{y}^T \mathbf{W} \mathbf{W}^T \mathbf{y} \right]\end{aligned}$$

$\mathbb{E}_{\mathbf{y}} [\mathbf{y}^T \mathbf{y}]$ is constant, so minimizing $E_{\mathbf{W}}$ is equivalent to maximizing $\mathbb{E}_{\mathbf{y}} [\mathbf{y}^T \mathbf{W} \mathbf{W}^T \mathbf{y}]$

- ▶ we use again the sample mean

$$\begin{aligned}\mathbb{E}_{\mathbf{y}} \left[\mathbf{y}^T \mathbf{W} \mathbf{W}^T \mathbf{y} \right] &= \frac{1}{n} \sum_{i=1}^n \mathbf{y}_i^T \mathbf{W} \mathbf{W}^T \mathbf{y}_i \\&= \frac{1}{n} \text{tr}(\mathbf{Y}^T \mathbf{W} \mathbf{W}^T \mathbf{Y})\end{aligned}$$

optimizing the PCA reconstruction-error criterion

- ▶ we want to maximize $\mathbb{E}_y[y^T \mathbf{W} \mathbf{W}^T y] = \frac{1}{n} \text{tr}(\mathbf{Y}^T \mathbf{W} \mathbf{W}^T \mathbf{Y})$
- ▶ consider the singular value decomposition of the data matrix $\mathbf{Y} = \mathbf{U} \Sigma \mathbf{V}^T$

$$\text{tr}(\mathbf{Y}^T \mathbf{W} \mathbf{W}^T \mathbf{Y}) = \text{tr}(\mathbf{V} \Sigma \mathbf{U}^T \mathbf{W} \mathbf{W}^T \mathbf{U} \Sigma \mathbf{V}^T)$$

- ▶ we can observe that the above expression reaches its maximum when the k columns of \mathbf{W} are colinear with the columns of \mathbf{U} that are associated with the k largest singular values in Σ , i.e.,

$$\mathbf{W} = \mathbf{U}_k$$

- ▶ finally, we obtain the projection of point \mathbf{y} to the latent point $\mathbf{x} = \mathbf{W}^T \mathbf{y} = \mathbf{U}_k^T \mathbf{y}$

putting everything together

the PCA method

- ▶ input data $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_n) \in \mathbb{R}^{d \times n}$
- ▶ data centering $\mathbf{Y} \leftarrow \mathbf{Y} - \frac{1}{n} \mathbf{Y}\mathbf{1}_n\mathbf{1}_n^T$
- ▶ singular value decomposition $\mathbf{Y} = \mathbf{U}\Sigma\mathbf{V}^T$
- ▶ take \mathbf{U}_k to be the k columns of \mathbf{U} that correspond to the k largest singular values
- ▶ map data to k -dimensional space by $\mathbf{X} = \mathbf{U}_k^T \mathbf{Y}$

explained variance and reconstruction error

- ▶ SVD gives a nice connection of explained variance and reconstruction error through singular values
- ▶ the variance of the data explained via the first k principal components is

$$V_{\mathbf{W}} = \sum_{i=1}^k \sigma_i^2$$

note that the variance of the whole data is $V_{\mathbf{Y}} = \sum_{i=1}^d \sigma_i^2$

- ▶ on the other hand, the reconstruction error is

$$E_{\mathbf{W}} = \sum_{i=k+1}^d \sigma_i^2$$

and thus, the reconstruction error is 0, when $k = d$ and $\mathbf{W} = \mathbf{U}$

selecting the number of latent variables (k)

- ▶ selecting the number of latent variables is a mix of science and art
- ▶ we can use the concept of explained variance

1. rule of thumb : select k so as to explain at least 85% of the data variance

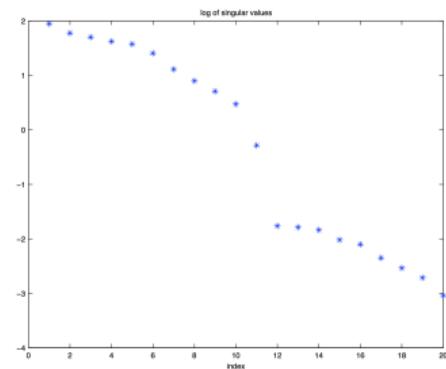
- select the smallest k for which

$$\frac{V_W}{V_Y} = \frac{\sum_{i=1}^k \sigma_i^2}{\sum_{i=1}^d \sigma_i^2} \geq 0.85$$

(or is it 95%?)

2. plot the singular values and look for a “gap”

- it is often more revealing when plotting the singular values in log scale



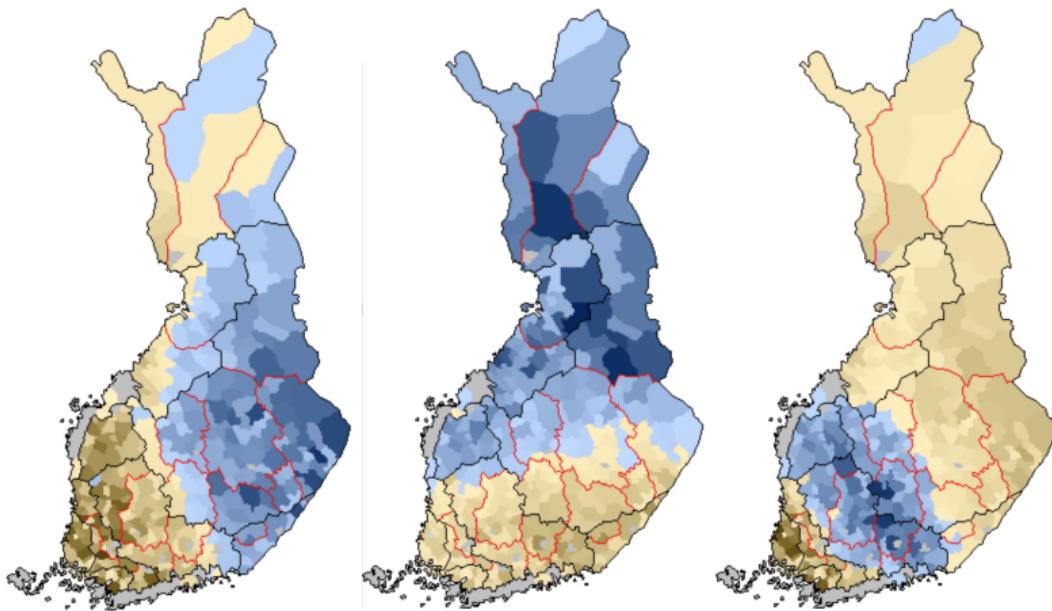
example : spatial / linguistic data analysis

- ▶ data : 9000 dialect words, 500 counties
 - points = words, dimensions = counties
 - data matrix \mathbf{Y} , so that $y_{ij} = 1$ if word i appears in county j , and $y_{ij} = 0$ otherwise
- ▶ apply PCA to this data
- ▶ obtain principal component matrix $\mathbf{W} \in \mathbb{R}^{d \times k}$
 - the i -th column of \mathbf{W} (i -th principal component), $i = 1, \dots, k$ indicates the counties that explain significant part of the variation left in the data

example credited to Saara Hyvönen

example : spatial / linguistic data analysis

visualizing the first three components



geographical structure of principal components is apparent

note : PCA knows nothing about geography

conclusion and discussion

- ▶ PCA is a simple and robust dimensionality-reduction method
 - a few lines of code in python, Matlab, R
 - popular and widely used
- ▶ closed-form optimal solution via SVD
- ▶ it comes with some rules of thumb to select the number of latent variables
- ▶ “layered” method : adds optimal principal components one by one
- ▶ main assumption, which can be a limitation in some cases : linear model

distance preservation as a means for dimensionality reduction

- ▶ in some cases, data can lie in a non-linear manifold inside a high-dimensional space
 - manifold not known, but we may be able to compute distances on the manifold
- ▶ in other cases, data points have no vector representation, but we can compute distances
 - e.g., string edit distance
- ▶ in both cases, we can compute distances between pairs of data points
- ▶ we want to embed points in a geometric space so that distances are preserved
 - distance between the embedded image of two points \approx original distance
- ▶ if distances are preserved in the low-dimensional embedding, intuitively, the embedding has captured the structure of the manifold, and the dimensionality reduction is successful

distance functions

- ▶ given a space \mathcal{Y} , a distance d is a function $d : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$
- ▶ if a distance function d satisfies the following properties

$d(\mathbf{a}, \mathbf{b}) \geq 0$, for all $\mathbf{a}, \mathbf{b} \in \mathcal{Y}$ non-negativity

$d(\mathbf{a}, \mathbf{b}) = 0$ if and only if $\mathbf{a} = \mathbf{b}$ isolation

$d(\mathbf{a}, \mathbf{b}) = d(\mathbf{b}, \mathbf{a})$, for all $\mathbf{a}, \mathbf{b} \in \mathcal{Y}$ symmetry

$d(\mathbf{a}, \mathbf{b}) \leq d(\mathbf{a}, \mathbf{c}) + d(\mathbf{c}, \mathbf{b})$, for all $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathcal{Y}$ triangle inequality

we say that the distance d is a **metric**

we also say that (\mathcal{Y}, d) (the space \mathcal{Y} equipped with distance d) is a **metric space**

Minkowski distances

- ▶ the Minkowski distances is a family of distance functions in the vector space \mathbb{R}^d
- ▶ for any two points (vectors) $\mathbf{a} = (a_1, \dots, a_d)$ and $\mathbf{b} = (b_1, \dots, b_d)$ in \mathbb{R}^d the Minkowski distance L_p between \mathbf{a} and \mathbf{b} is defined as

$$d(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|_p = \left(\sum_{i=1}^d |a_i - b_i|^p \right)^{\frac{1}{p}}$$

- ▶ the following special cases of L_p are most commonly used
 - $p = 1$: city-block or Manhattan distance
 - $p = 2$: Euclidean distance
 - $p = \infty$: maximum distance
- ▶ the Minkowski distance is a metric, for all $p \geq 1$

similarity functions

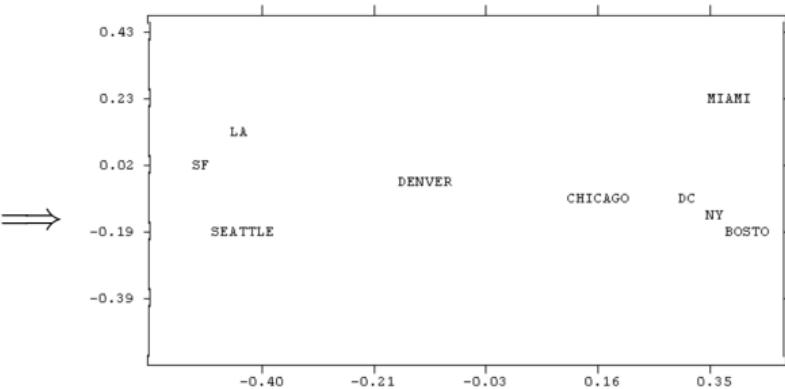
- ▶ distances are measures of dissimilarity
- ▶ often we work with similarity functions, which are measures of similarity
- ▶ similarity functions map pairs of points to reals, i.e., $\text{sim} : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$
- ▶ commonly-used similarity functions are the dot-product, the cosine similarity, the Jaccard coefficient, and more ...
- ▶ similarity functions often take values in the range $[0, 1]$, where
 - value 1 indicates that two points are identical, and
 - value 0 indicates that they are totally dissimilar
- ▶ on the other hand, for distance functions
 - value 0 indicates that two points are identical, and
 - a large value indicates that they are totally dissimilar
 - sometimes we normalize distances to be in the interval $[0, 1]$

multidimensional scaling (MDS)

- ▶ intuitively : given a matrix of pair-wise distances, can be map the data in a low-dimensional space so that distances are preserved?

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
|---|---------|------|------|------|------|------|------|------|------|------|
| | BOST | NY | DC | MIAM | CHIC | SEAT | SF | LA | DENV | |
| 1 | BOSTON | 0 | 206 | 429 | 1504 | 963 | 2976 | 3095 | 2979 | 1949 |
| 2 | NY | 206 | 0 | 233 | 1308 | 802 | 2815 | 2934 | 2786 | 1771 |
| 3 | DC | 429 | 233 | 0 | 1075 | 671 | 2684 | 2799 | 2631 | 1616 |
| 4 | MIAMI | 1504 | 1308 | 1075 | 0 | 1329 | 3273 | 3053 | 2687 | 2037 |
| 5 | CHICAGO | 963 | 802 | 671 | 1329 | 0 | 2013 | 2142 | 2054 | 996 |
| 6 | SEATTLE | 2976 | 2815 | 2684 | 3273 | 2013 | 0 | 808 | 1131 | 1307 |
| 7 | SF | 3095 | 2934 | 2799 | 3053 | 2142 | 808 | 0 | 379 | 1235 |
| 8 | LA | 2979 | 2786 | 2631 | 2687 | 2054 | 1131 | 379 | 0 | 1059 |
| 9 | DENVER | 1949 | 1771 | 1616 | 2037 | 996 | 1307 | 1235 | 1059 | 0 |

distance matrix



mapping to 2D

example credited to Stephen Borgatti, 1997

color perception

[Ekman, 1954]

- ▶ study color perception in human vision
- ▶ consider 14 colors differing only in their hue
- ▶ 31 people rate differences between all pairs of colors, on a five-point scale
- ▶ average ratings and convert to distances
 - 0 indicates that two colors are identical, 1 indicates that they are totally different

color perception

pairwise distance matrix

| | 434 | 445 | 465 | 472 | 490 | 504 | 537 | 555 | 584 | 600 | 610 | 628 | 651 |
|-----|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 445 | 0.14 | | | | | | | | | | | | |
| 465 | 0.58 | 0.50 | | | | | | | | | | | |
| 472 | 0.58 | 0.56 | 0.19 | | | | | | | | | | |
| 490 | 0.82 | 0.78 | 0.53 | 0.46 | | | | | | | | | |
| 504 | 0.94 | 0.91 | 0.83 | 0.75 | 0.39 | | | | | | | | |
| 537 | 0.93 | 0.93 | 0.90 | 0.90 | 0.69 | 0.38 | | | | | | | |
| 555 | 0.96 | 0.93 | 0.92 | 0.91 | 0.74 | 0.55 | 0.27 | | | | | | |
| 584 | 0.98 | 0.98 | 0.98 | 0.98 | 0.93 | 0.86 | 0.78 | 0.67 | | | | | |
| 600 | 0.93 | 0.96 | 0.99 | 0.99 | 0.98 | 0.92 | 0.86 | 0.81 | 0.42 | | | | |
| 610 | 0.91 | 0.93 | 0.98 | 1.00 | 0.98 | 0.98 | 0.95 | 0.96 | 0.63 | 0.26 | | | |
| 628 | 0.88 | 0.89 | 0.99 | 0.99 | 0.99 | 0.98 | 0.98 | 0.97 | 0.73 | 0.50 | 0.24 | | |
| 651 | 0.87 | 0.87 | 0.95 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.80 | 0.59 | 0.38 | 0.15 | |
| 674 | 0.84 | 0.86 | 0.97 | 0.96 | 1.00 | 0.99 | 1.00 | 0.98 | 0.77 | 0.72 | 0.45 | 0.32 | 0.24 |

color perception

MDS reproduces the well-known 2-dimensional color circle

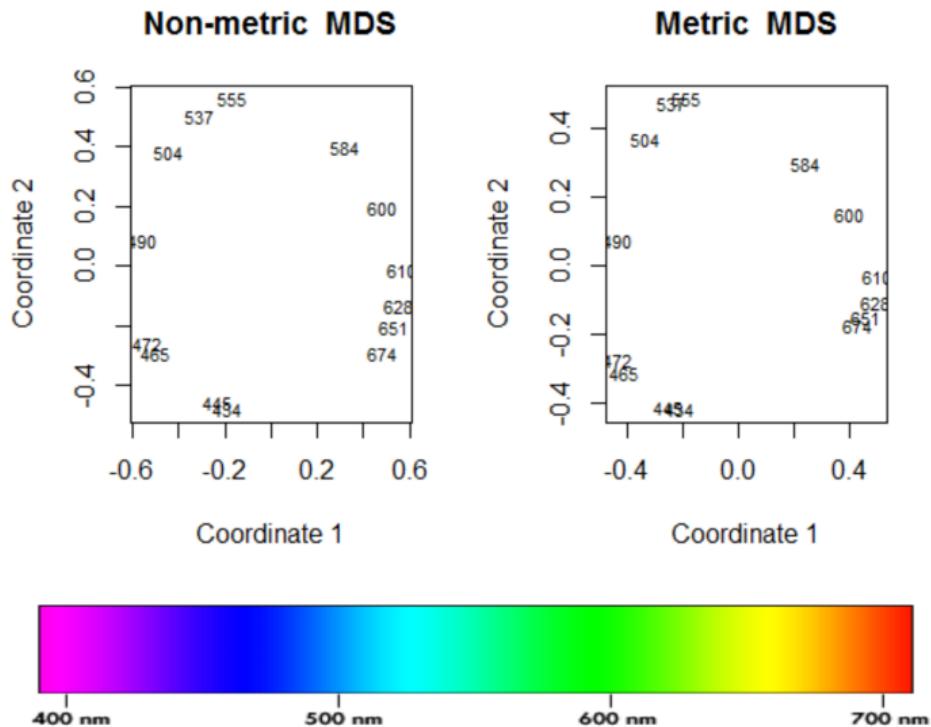
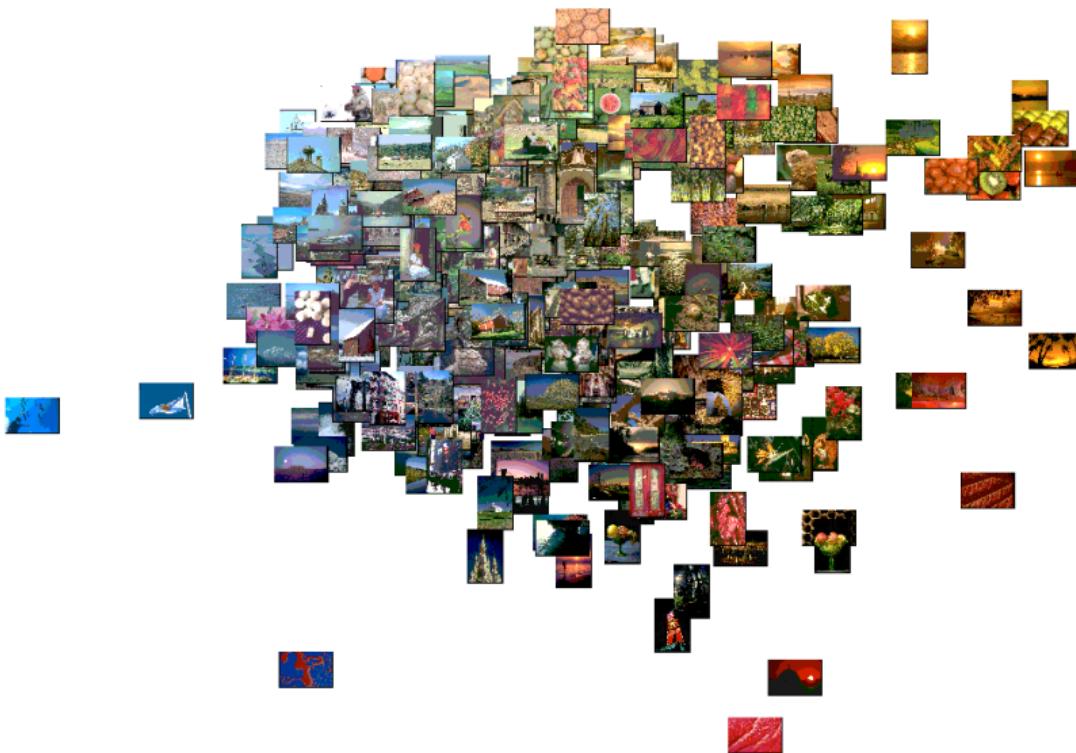


image-retrieval interface with MDS



example from Rubner et al., 1997

2D MDS map of images



example from Rubner et al., 1997

multidimensional scaling (MDS)

a family of related techniques

- ▶ **scaling** refers to constructing a configuration of points in a target metric space from interpoint distance information
- ▶ **MDS** is scaling when the target metric space is the Euclidean space
- ▶ **classical MDS** : the basic MDS formulation
- ▶ **metric MDS** : a generalization of classical MDS, with interpoint distance information, and optimizing a general stress function
- ▶ **non-metric MDS** : a generalization of classical MDS, with rank information

classical MDS with a similarity matrix

- ▶ consider dataset $\mathcal{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\}$, represented as matrix $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_n) \in \mathbb{R}^{d \times n}$
- ▶ assume that **data is not known**, instead **pairwise similarities** are known
- ▶ we know the similarity matrix \mathbf{S} , such that $[\mathbf{S}]_{ij} = s_{ij} = \mathbf{y}_i^T \mathbf{y}_j$ (dot-product similarity)
then, $\mathbf{S} = \mathbf{Y}^T \mathbf{Y}$
- ▶ as with PCA, for point i we assume a latent vector \mathbf{x}_i , so that $\mathbf{y}_i = \mathbf{W} \mathbf{x}_i$,
where \mathbf{W} defines a linear transformation
 - we have $\mathbf{Y} = \mathbf{W} \mathbf{X}$
 - we assume again that the columns of \mathbf{W} are orthonormal, i.e., $\mathbf{W}^T \mathbf{W} = \mathbf{I}_{k \times k}$
- ▶ we have
$$\mathbf{S} = \mathbf{Y}^T \mathbf{Y} = (\mathbf{W} \mathbf{X})^T (\mathbf{W} \mathbf{X}) = \mathbf{X}^T (\mathbf{W}^T \mathbf{W}) \mathbf{X} = \mathbf{X}^T \mathbf{X}$$
- and note again that both \mathbf{Y} and \mathbf{X} are unknown
- ▶ the matrix \mathbf{S} is called the **Gram matrix**

classical MDS with a similarity matrix

- ▶ recap : given similarity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$, we want to find latent matrix $\mathbf{X} \in \mathbb{R}^{k \times n}$, such that $\mathbf{S} = \mathbf{X}^T \mathbf{X}$
- ▶ the similarity matrix \mathbf{S} is symmetric positive semi-definite, so its eigen-decomposition is

$$\mathbf{S} = \mathbf{U} \Lambda \mathbf{U}^{-1} = \mathbf{U} \Lambda \mathbf{U}^T = (\Lambda^{1/2} \mathbf{U}^T)^T (\Lambda^{1/2} \mathbf{U}^T)$$

the non-negative eigenvalues of \mathbf{S} are sorted in descending order in the diagonal of Λ

- ▶ it follows that we can take the $k \times n$ matrix \mathbf{X} to be

$$\mathbf{X} = \mathbf{I}_{k \times n} \Lambda^{1/2} \mathbf{U}^T$$

- ▶ the matrix \mathbf{X} provides a k embedding for each data point
(point i is taken as the i -th column of \mathbf{X})

classical MDS with a distance matrix

- ▶ in many cases, instead of the similarity matrix \mathbf{S} , where $[\mathbf{S}]_{ij} = s_{ij} = \mathbf{y}_i^T \mathbf{y}_j$
we are given as input the pairwise distance matrix \mathbf{D} , where $[\mathbf{D}]_{ij} = d_{ij} = \|\mathbf{y}_i - \mathbf{y}_j\|_2$
- ▶ we want to perform MDS on the distance matrix \mathbf{D}
- ▶ we can use the fact

$$d_{ij}^2 = s_{ii} + s_{jj} - 2s_{ij} \Rightarrow s_{ij} = -\frac{1}{2}(d_{ij}^2 - s_{ii} - s_{jj})$$

but note that $s_{ii} = \mathbf{y}_i^T \mathbf{y}_i$ and $s_{jj} = \mathbf{y}_j^T \mathbf{y}_j$ are unknown

- ▶ however, we can estimate s_{ij} by a “double centering” trick:
subtract from each entry of \mathbf{D} the mean of the corresponding row and the mean of the corresponding column, and add back the mean of all entries
 - ▶ in matrix form
- $$\mathbf{S} = -\frac{1}{2} \left(\mathbf{D} - \frac{1}{n} \mathbf{D} \mathbf{1}_n \mathbf{1}_n^T - \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^T \mathbf{D} + \frac{1}{n^2} \mathbf{1}_n \mathbf{1}_n^T \mathbf{D} \mathbf{1}_n \mathbf{1}_n^T \right)$$
- ▶ so, we can compute \mathbf{S} from \mathbf{D} , and apply MDS with similarities

classical MDS — putting everything together

1. if the similarity matrix \mathbf{S} is available go to step 4
2. if the data matrix \mathbf{Y} is available, compute $\mathbf{S} = \mathbf{Y}^T \mathbf{Y}$, and go to step 4
3. if the distance matrix \mathbf{D} is available, compute \mathbf{S} by the “double centering” trick
4. compute the eigen-decomposition $\mathbf{S} = \mathbf{U} \Lambda \mathbf{U}^T$
5. a k -dimensional representation of the data is obtained by $\mathbf{X} = \mathbf{I}_{k \times n} \Lambda^{1/2} \mathbf{U}^T$

metric MDS

- ▶ a generalization of classical MDS, where we ask to optimize the stress function (reconstruction error)

$$E_{\text{mMDS}} = \sum_{i < j} w_{ij}(d_{ij} - \hat{d}_{ij})^2$$

where d_{ij} are the input distances, and \hat{d}_{ij} the distances of the reconstructed points

- ▶ when $w_{ij} = 1/d_{ij}$, the method is called [Sammon's nonlinear mapping](#)
 - the intuition is to give less importance to errors made on large distances
- ▶ for this MDS variant we cannot derive a closed-form optimal solution, instead we search for a solution via a general optimization method, such as gradient descent

non-metric MDS

- ▶ in many cases ordinal information is given instead of quantitative distances
 - e.g., in a psychology study people may be able to rank a set of concepts, or indicate relative similarity between groups of objects, without being able to assign quantitative scores
- ▶ non-metric MDS asks to optimize a stress function of the form

$$E_{\text{nmMDS}} = \left(\sum_{i < j} w_{ij} (f(\delta_{ij}) - \hat{d}_{ij})^2 \right)^{\frac{1}{2}}$$

where δ_{ij} are the ordinal proximities

f is monotone transformation of proximities, such that $f(\delta_{ij}) \approx d_{ij}$

d_{ij} is the Euclidean distance between the unknown data points \mathbf{y}_i and \mathbf{y}_j , and \hat{d}_{ij} the distances of the reconstructed points

- ▶ as with metric MDS, we rely on general optimization methods, such as gradient descent

conclusion and discussion

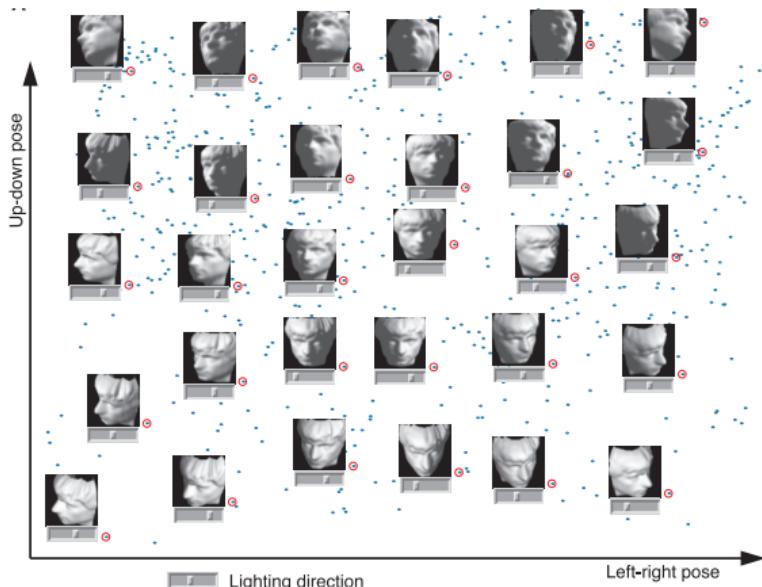
- ▶ like PCA, classical MDS is a simple and robust dimensionality-reduction method
- ▶ it also assumes a linear model
- ▶ optimal solution can be derived and efficiently computed via SVD
- ▶ MDS is more flexible than PCA, as it only assumes a similarity or distance matrix, and it is applicable when no vector representation of data is available
- ▶ on the other hand, MDS is computationally more intensive when working with matrices of size $n \times n$
- ▶ metric and non-metric MDS are more general methods but they come with the cost of having to solve a more difficult optimization problem

nonlinear dimensionality reduction — motivation

many high-dimensional points lie in lower-dimensional manifolds

example

- consider a set of images, of resolution 128×128 , depicting an object from different positions, and different lighting
- an image can be considered as a point in a 16384-dimensional space
- arguably, there are 3 degrees of freedom (specifying position and lighting)
- thus, the set of images lies in a 3-dimensional nonlinear manifold within a 16384-dimensional space



example from Tenenbaum et al., 2000

nonlinear dimensionality reduction

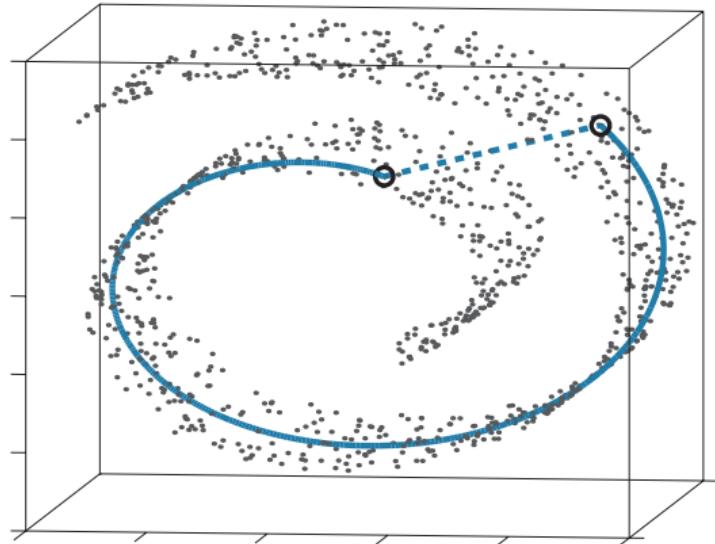
objective

given high-dimensional points $\{\mathbf{y}_i\}$, possibly lying on a non-linear manifold,
reconstruct low-dimensional coordinates of the data $\{\mathbf{x}_i\}$, which describe where
the points lie on the manifold

isomap

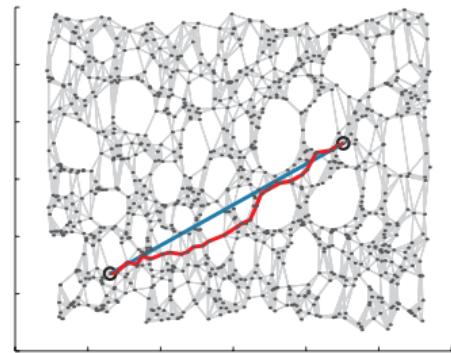
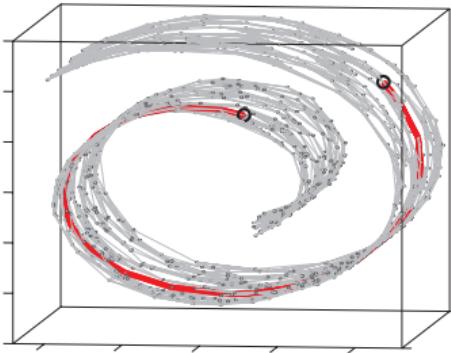
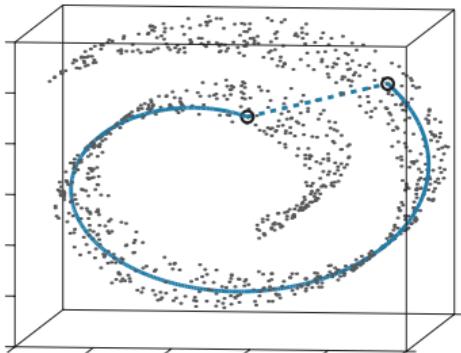
intuition

- Euclidean distance in the original space may be a poor measure of dissimilarity between points
- instead use **geodesic distance**, distance between points on the manifold
- geodesic distance captures more accurately captures the neighborhood relationships that should be preserved



example from Tenenbaum et al., 2000

calculating geodesic distance



- ▶ without knowing the manifold, calculating geodesic distance is impossible
- ▶ for nearby points, geodesic distance \approx Euclidean distance
- ▶ for faraway points, approximate geodesic distance by a sequence “short hops” between neighboring points

isomap algorithm

given data $\mathcal{Y} = \{\mathbf{y}_i\}$

1. compute distances $\delta_{ij} = \|\mathbf{y}_i - \mathbf{y}_j\|^2$ in the original space
2. construct a graph G , where vertices represent points, and each point is connected to its p nearest points, according to distances δ_{ij}
3. for each pair of points i and j compute the shortest path distance d_{ij} from i to j on the graph G (e.g., using the Floyd-Warshall algorithm)
4. use MDS on the shortest-path distances $\{d_{ij}\}$ to compute the low-dimensional embedding $\{\mathbf{x}_i\}$

conclusion and discussion

- ▶ isomap is an algorithm designed to recover low-dimensional non-linear manifolds within high-dimensional spaces
- ▶ it works by constructing a graph, which approximates geodesic distances on the manifold, and then uses MDS
- ▶ the method can be unstable, for example
 - sensitive to noise
 - sensitive to number of nearest neighbors to construct the graph G
 - if the graph G is disconnected, the algorithm will fail