Dimensionality Reduction

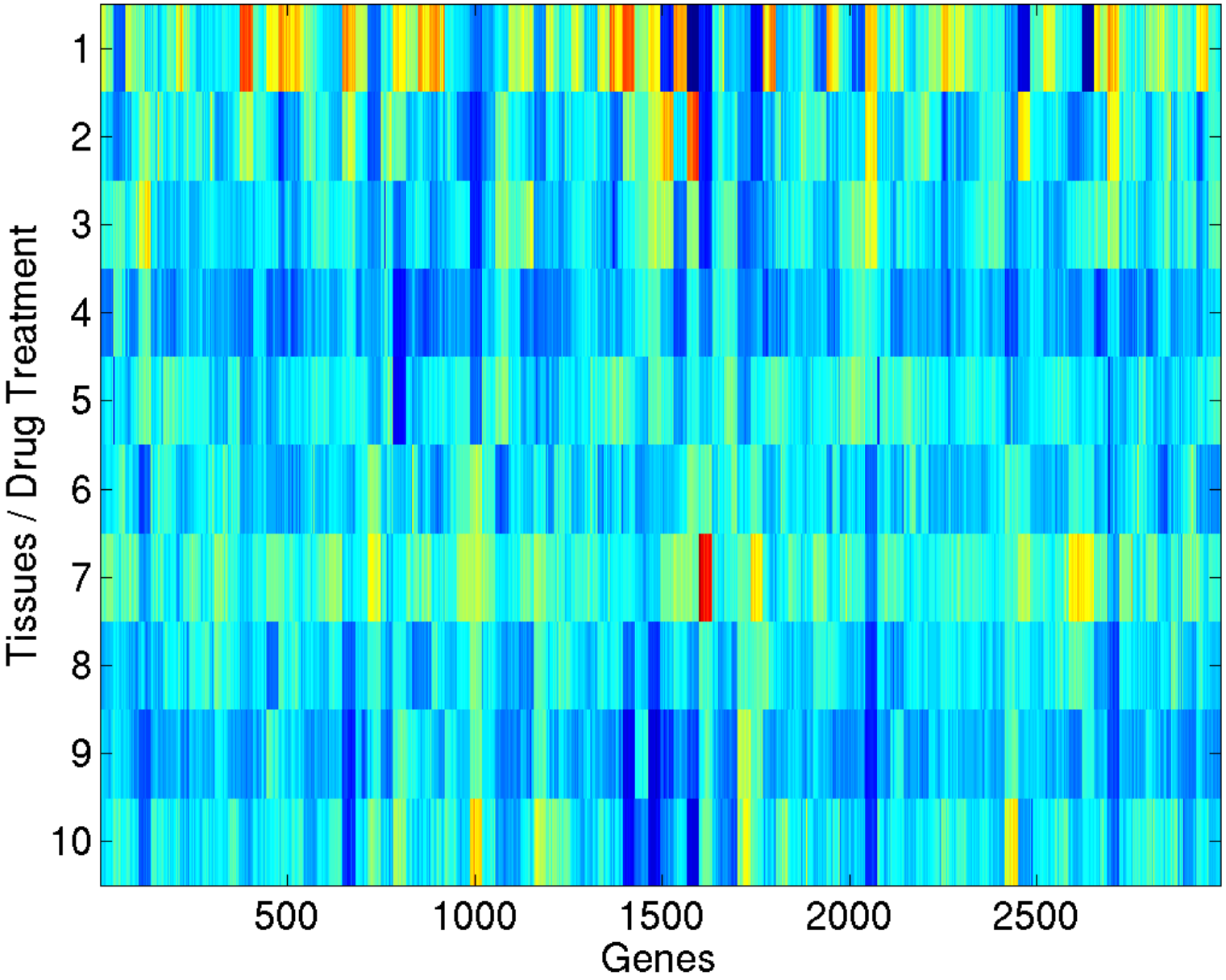
Maneesh Sahani
maneesh@gatsby.ucl.ac.uk

Gatsby Computational Neuroscience Unit, UCL

Apr/May 2016

High dimensional data

Example data: Gene Expression



Example data: Web Pages

Google Search: Unsupervised Learning http://www.google.com/search?q=Unsupervised+Learning&sourceid=fir...

Google Web Images Groups News Froogle more »
Unsupervised Learning Search Advanced Search Preferences

Web Results **1** - **10** of about **150,000** for **Unsupervised Learning**. (**0.27** seconds)

Mixture modelling, Clustering, Intrinsic classification **...**
Mixture Modelling page. Welcome to David Dowe's clustering, mixture modelling and **unsupervised learning** page. Mixture modelling (or **...**
www.csse.monash.edu.au/~dld/mixture.modelling.page.html - 26k - 4 Oct 2004 - Cached - Similar pages

ACL'99 Workshop -- **Unsupervised Learning** in Natural Language **...**
PROGRAM. ACL'99 Workshop **Unsupervised Learning** in Natural Language Processing. University of Maryland June 21, 1999. Endorsed by SIGNLL **...**
www.ai.sri.com/~kehler/unsup-acl-99.html - 5k - Cached - Similar pages

Unsupervised learning and Clustering
cgm.cs.mcgill.ca/~soss/cs644/projects/wijhe/ - 1k - Cached - Similar pages

NIPS*98 Workshop - Integrating Supervised and **Unsupervised ...**
NIPS*98 Workshop "Integrating Supervised and **Unsupervised Learning**" Friday, December 4, 1998. **...** 4:45-5:30, Theories of **Unsupervised Learning** and Missing Values. **...**
www-2.cs.cmu.edu/~mccallum/supunsup/ - 7k - Cached - Similar pages

NIPS Tutorial 1999
Probabilistic Models for **Unsupervised Learning** Tutorial presented at the 1999 NIPS Conference by Zoubin Ghahramani and Sam Roweis. **...**
www.gatsby.ucl.ac.uk/~zoubin/NIPStutorial.html - 4k - Cached - Similar pages

Gatsby Course: **Unsupervised Learning** : Homepage
Unsupervised Learning (Fall 2000). **...** Syllabus (resources page): 10/10 1 - Introduction to **Unsupervised Learning** Geoff project: (ps, pdf). **...**
www.gatsby.ucl.ac.uk/~quaid/course/ - 15k - Cached - Similar pages
[More results from www.gatsby.ucl.ac.uk]

[PDF] **Unsupervised Learning** of the Morphology of a Natural Language
File Format: PDF/Adobe Acrobat - View as HTML
Page 1. Page 2. Page 3. Page 4. Page 5. Page 6. Page 7. Page 8. Page 9. Page 10. Page 11. Page 12. Page 13. Page 14. Page 15. Page 16. Page 17. Page 18. Page 19 **...**
acl.ldc.upenn.edu/J/J01/J01-2001.pdf - Similar pages

Unsupervised Learning - The MIT Press
... From Bradford Books: **Unsupervised Learning** Foundations of Neural Computation Edited by Geoffrey Hinton and Terrence J. Sejnowski Since its founding in 1989 by **...**
mitpress.mit.edu/book-home.tcl?isbn=026258168X - 13k - Cached - Similar pages

[PS] **Unsupervised Learning** of Disambiguation Rules for Part of
File Format: Adobe PostScript - View as Text
Unsupervised Learning of Disambiguation Rules for Part of. Speech Tagging. Eric Brill. 1. **...** It is possible to use **unsupervised learning** to train stochastic. **...**
www.cs.jhu.edu/~brill/acl-wkshp.ps - Similar pages

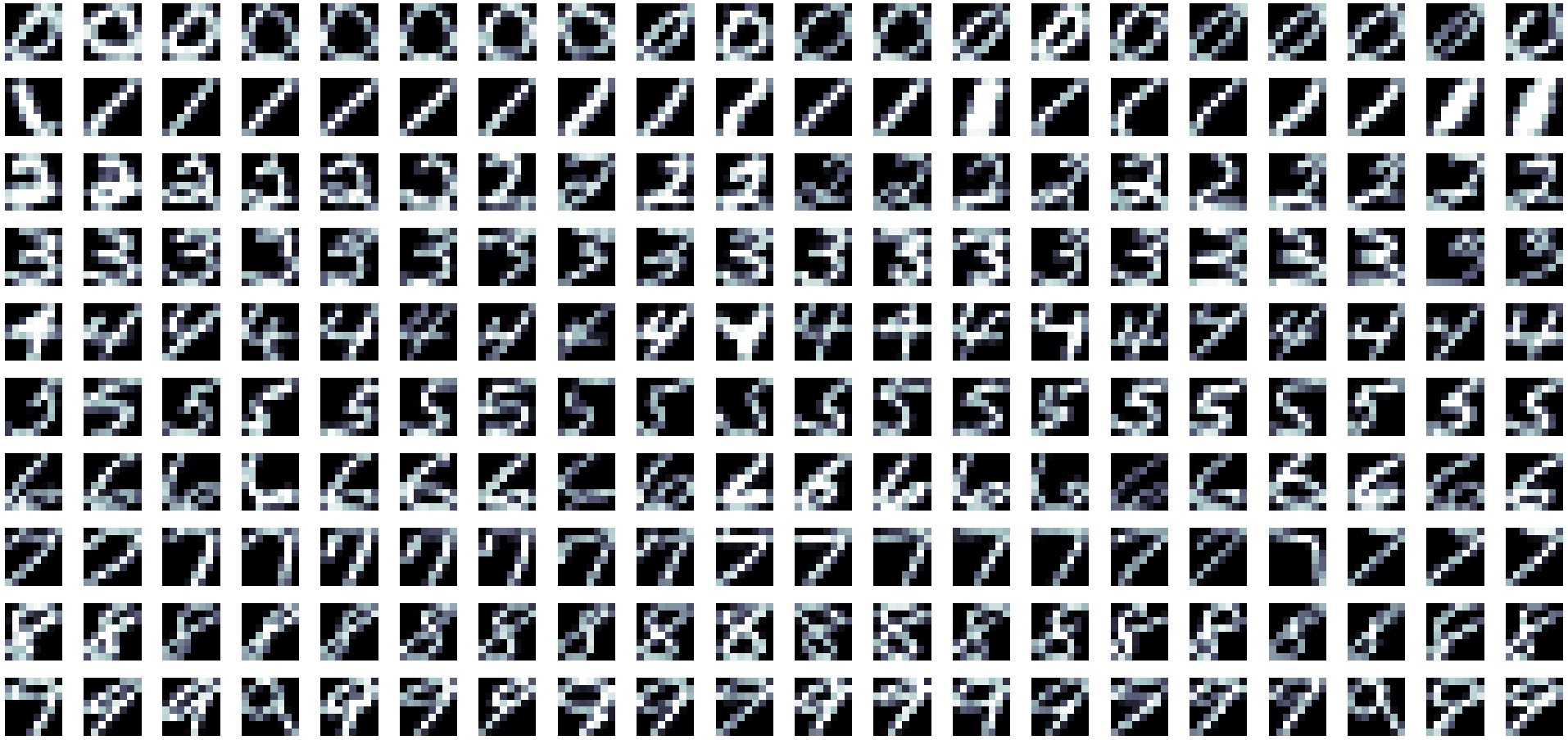
The **Unsupervised Learning** Group (ULG) at UT Austin
The **Unsupervised Learning** Group (ULG). What ? The **Unsupervised Learning** Group (ULG) is a group of graduate students from the Computer **...**
www.lans.ece.utexas.edu/ulg/ - 14k - Cached - Similar pages

Gooooooooooogle
Result Page: 1 2 3 4 5 6 7 8 9 10 **Next**

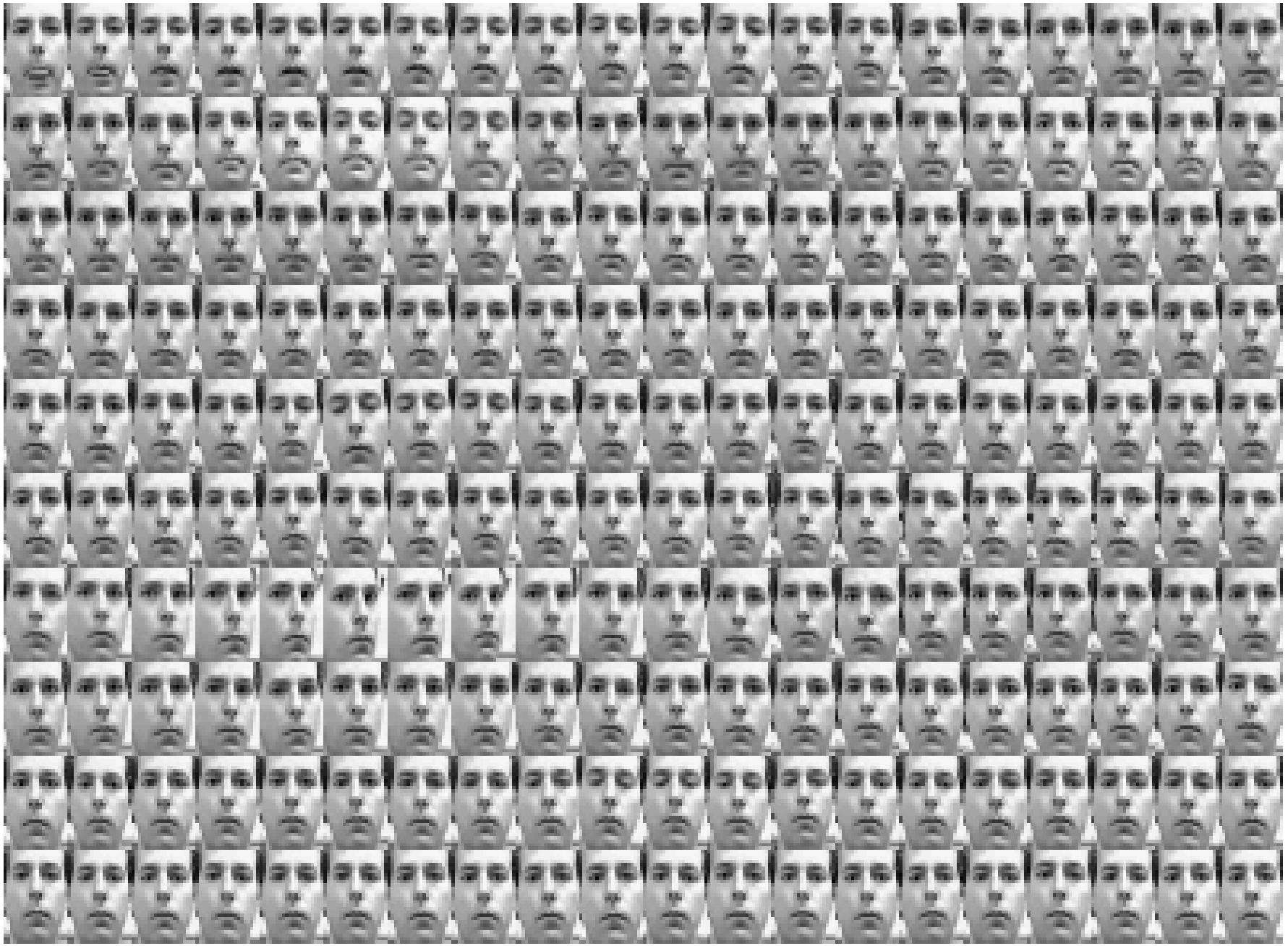
1 of 2 06/10/04 15:44

Tabulate word-frequency vectors

Example data: Images

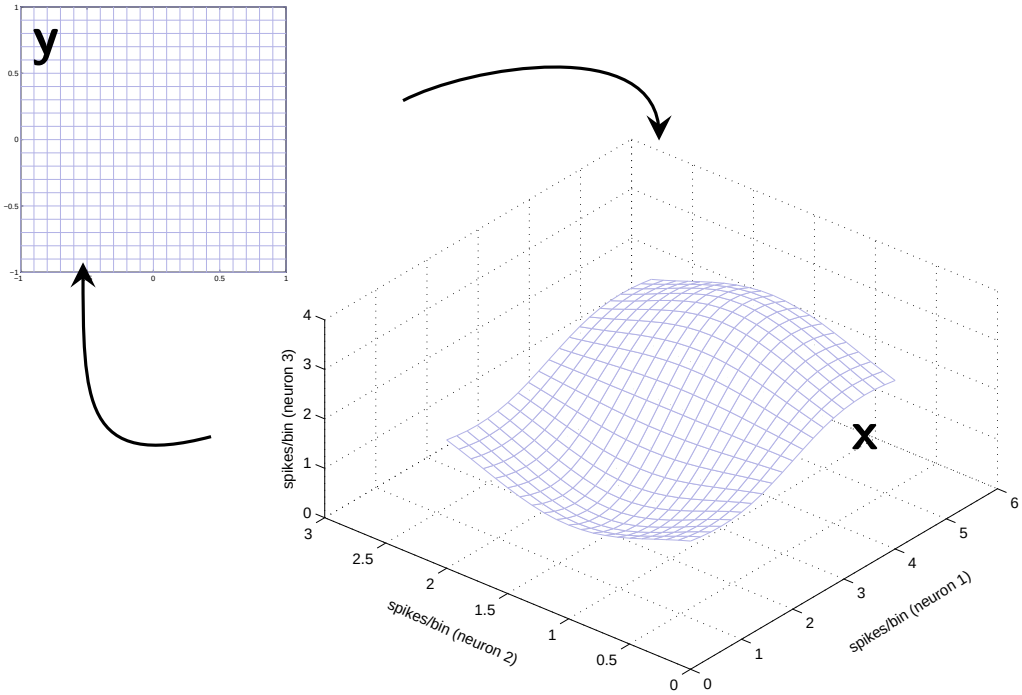


Example data: Images



High-dimensional data

These are all vectors ($\mathbf{x}_i \in \mathbb{R}^n, i = 1 \ldots m$) in a high-dimensional space. But not all possible vectors appear in the data set.



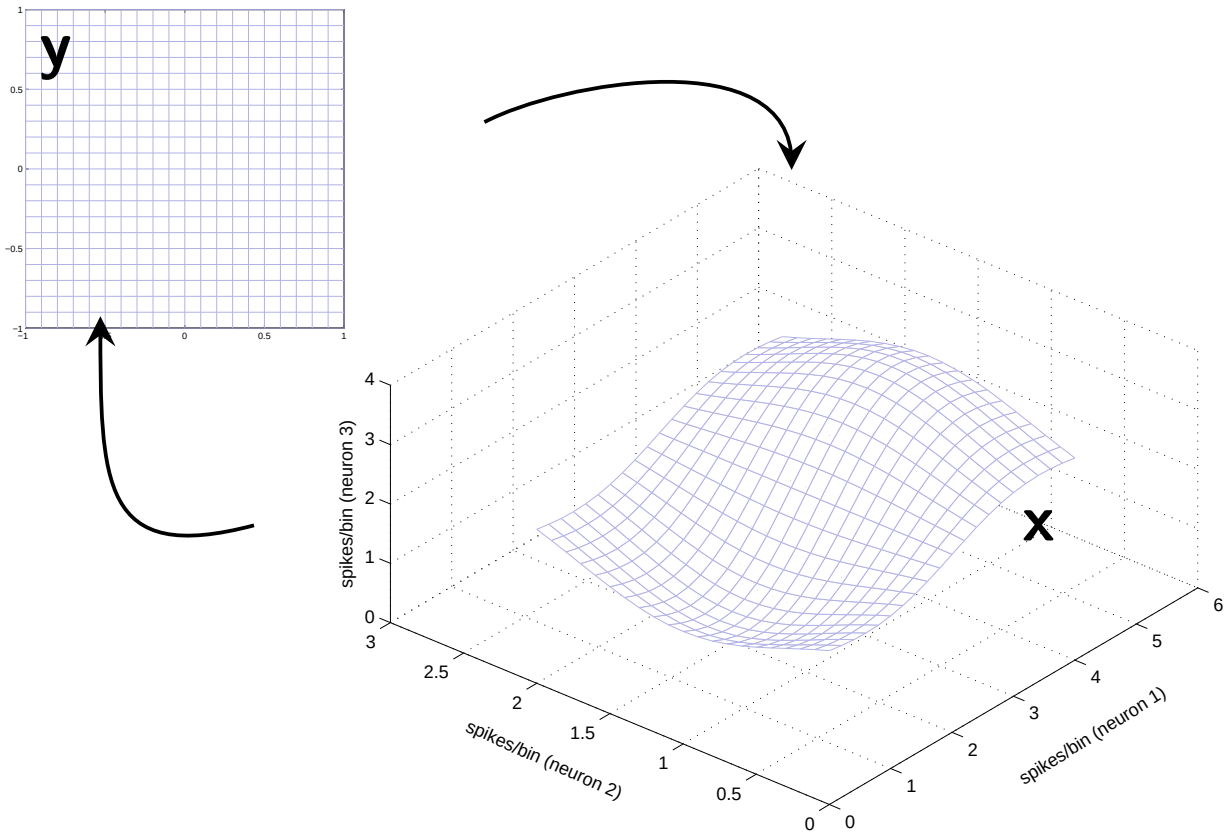
Data live on a low-dimensional manifold

- subset of possible values
- smoothly varying and dense
- may be parametrised by “latent variables.”

Dimensionality reduction

Goal: Find the manifold.

More precisely, find $\mathbf{y}_i \in \mathbb{R}^k, \ (k < n)$ so that $\mathbf{y}_i$ parametrises the location of $\mathbf{x}_i$ on the manifold.



The term embedding is used loosely for both $\mathbf{y} \rightarrow \mathbf{x}$ and $\mathbf{x} \rightarrow \mathbf{y}$.

Uses of dimensionality reduction

- Structure discovery
- Visualisation
- Pre-processing

Order of business

- Mathematical preliminaries — low rank factorisations
  - singular-value- and eigen-decompositions.
- Classical linear methods — just factorisation
  - PCA
  - MDS
- Nonlinear methods — pre-processing, then factorisation
  - Isomap
  - LLE
  - MVU
  - SNE (not actually a factorisation)

Some preliminaries

Some preliminaries

Data: $\mathbf{x}_i = \begin{bmatrix} x_{1i} \\ x_{2i} \\ \vdots \\ x_{ni} \end{bmatrix} \in \mathbb{R}^n$ can be collected into a matrix:

$$X = \underbrace{\left[\begin{array}{cccc} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_m \\ | & | & & | \end{array}\right]}_{m} \Bigg\} n$$

We will assume the data are **centred**: $\bar{\mathbf{x}} = \langle \mathbf{x} \rangle = \sum_{i=0}^{m} \mathbf{x}_i = \mathbf{0}$.

Two matrices of interest

The **data covariance** or **scatter** matrix:

$$S \;=\; \Big\langle (\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^{\mathsf{T}} \Big\rangle \;=\; \frac{1}{m}\sum_i \mathbf{x}_i \mathbf{x}_i^{\mathsf{T}} \;=\; \frac{1}{m} X X^{\mathsf{T}} \;\in\; \mathbb{R}^{n\times n}.$$

Two matrices of interest

The **data covariance** or **scatter** matrix:

$$S \;=\; \Big\langle (\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^\mathsf{T} \Big\rangle \;=\; \frac{1}{m}\sum_i \mathbf{x}_i \mathbf{x}_i^\mathsf{T} \;=\; \frac{1}{m} X X^\mathsf{T} \;\in\; \mathbb{R}^{n\times n}.$$

The **inner product** or **Gram** matrix:

$$G \;=\; \begin{bmatrix} \mathbf{x}_1^\mathsf{T}\mathbf{x}_1 & \mathbf{x}_1^\mathsf{T}\mathbf{x}_2 & \cdots & \mathbf{x}_1^\mathsf{T}\mathbf{x}_m \\ \mathbf{x}_2^\mathsf{T}\mathbf{x}_1 & \mathbf{x}_2^\mathsf{T}\mathbf{x}_2 & \cdots & \mathbf{x}_2^\mathsf{T}\mathbf{x}_m \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_m^\mathsf{T}\mathbf{x}_1 & \mathbf{x}_m^\mathsf{T}\mathbf{x}_2 & \cdots & \mathbf{x}_m^\mathsf{T}\mathbf{x}_m \end{bmatrix} \;=\; X^\mathsf{T} X \;\in\; \mathbb{R}^{m\times m}.$$

Two matrices of interest

The **data covariance** or **scatter** matrix:

$$S = \left\langle (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\mathsf{T} \right\rangle = \frac{1}{m} \sum_i \mathbf{x}_i \mathbf{x}_i^\mathsf{T} = \frac{1}{m} X X^\mathsf{T} \in \mathbb{R}^{n \times n}.$$

The **inner product** or **Gram** matrix:

$$G = \begin{bmatrix} \mathbf{x}_1^\mathsf{T}\mathbf{x}_1 & \mathbf{x}_1^\mathsf{T}\mathbf{x}_2 & \cdots & \mathbf{x}_1^\mathsf{T}\mathbf{x}_m \\ \mathbf{x}_2^\mathsf{T}\mathbf{x}_1 & \mathbf{x}_2^\mathsf{T}\mathbf{x}_2 & \cdots & \mathbf{x}_2^\mathsf{T}\mathbf{x}_m \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_m^\mathsf{T}\mathbf{x}_1 & \mathbf{x}_m^\mathsf{T}\mathbf{x}_2 & \cdots & \mathbf{x}_m^\mathsf{T}\mathbf{x}_m \end{bmatrix} = X^\mathsf{T} X \in \mathbb{R}^{m \times m}.$$

Both are real, symmetric and positive semi-definite.

Eigenvectors and Eigenvalues

Recall: **v** is an eigenvector, with scalar eigenvalue λ, of a matrix A if

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

Eigenvectors and Eigenvalues

Recall: **v** is an eigenvector, with scalar eigenvalue λ, of a matrix A if

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

v can have any norm, but we will define it to be unity (i.e., $\mathbf{v}^\mathsf{T}\mathbf{v} = 1$).

Eigenvectors and Eigenvalues

Recall: **v** is an eigenvector, with scalar eigenvalue λ, of a matrix A if

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

v can have any norm, but we will define it to be unity (i.e., $\mathbf{v}^\mathsf{T}\mathbf{v} = 1$).

For S ($n \times n$ real, symmetric, positive semi-definite):

Eigenvectors and Eigenvalues

Recall: **v** is an eigenvector, with scalar eigenvalue λ, of a matrix A if

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

v can have any norm, but we will define it to be unity (i.e., $\mathbf{v}^\mathsf{T}\mathbf{v} = 1$).

For S ($n \times n$ real, symmetric, positive semi-definite):

- In general there are n eigenvector-eigenvalue pairs $(\mathbf{v}_{(i)}, \lambda_{(i)})$.

Eigenvectors and Eigenvalues

Recall: **v** is an eigenvector, with scalar eigenvalue λ, of a matrix A if

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

v can have any norm, but we will define it to be unity (i.e., $\mathbf{v}^\mathsf{T}\mathbf{v} = 1$).

For S ($n \times n$ real, symmetric, positive semi-definite):

- In general there are n eigenvector-eigenvalue pairs $(\mathbf{v}_{(i)}, \lambda_{(i)})$.
- The n eigenvectors are orthogonal, forming an orthonormal basis.

$$\sum_i \mathbf{v}_{(i)}\mathbf{v}_{(i)}^\mathsf{T} = I.$$

Eigenvectors and Eigenvalues

Recall: **v** is an eigenvector, with scalar eigenvalue λ, of a matrix A if

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

v can have any norm, but we will define it to be unity (i.e., $\mathbf{v}^{\mathsf{T}}\mathbf{v} = 1$).

For S ($n \times n$ real, symmetric, positive semi-definite):

- In general there are n eigenvector-eigenvalue pairs $(\mathbf{v}_{(i)}, \lambda_{(i)})$.
- The n eigenvectors are orthogonal, forming an orthonormal basis.

$$\sum_i \mathbf{v}_{(i)}\mathbf{v}_{(i)}^{\mathsf{T}} = I.$$

- Any vector **u** can be written as

$$\mathbf{u} = \left(\sum_i \mathbf{v}_{(i)}\mathbf{v}_{(i)}^{\mathsf{T}}\right)\mathbf{u} = \sum_i (\mathbf{v}_{(i)}^{\mathsf{T}}\mathbf{u})\mathbf{v}_{(i)} = \sum_i u_{(i)}\mathbf{v}_{(i)}$$

Eigendecomposition

Define matrices

$$V = \begin{bmatrix} | & | & & | \\ \mathbf{v}_{(1)} & \mathbf{v}_{(2)} & \cdots & \mathbf{v}_{(n)} \\ | & | & & | \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_{(1)} & & & \\ & \lambda_{(2)} & & \\ & & \ddots & \\ & & & \lambda_{(n)} \end{bmatrix}$$

Eigendecomposition

Define matrices

$$V = \begin{bmatrix} | & | & & | \\ \mathbf{v}_{(1)} & \mathbf{v}_{(2)} & \cdots & \mathbf{v}_{(n)} \\ | & | & & | \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_{(1)} & & & \\ & \lambda_{(2)} & & \\ & & \ddots & \\ & & & \lambda_{(n)} \end{bmatrix}$$

Then we can write the eigenvector definition as:

$$SV = V\Lambda$$

Eigendecomposition

Define matrices

$$V = \begin{bmatrix} | & | & & | \\ \mathbf{v}_{(1)} & \mathbf{v}_{(2)} & \cdots & \mathbf{v}_{(n)} \\ | & | & & | \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_{(1)} & & & \\ & \lambda_{(2)} & & \\ & & \ddots & \\ & & & \lambda_{(n)} \end{bmatrix}$$

Then we can write the eigenvector definition as:

$$SV = V\Lambda$$

For symmetric S (i.e. orthonormal V):

$$SVV^{\mathsf{T}} = V\Lambda V^{\mathsf{T}}$$

Eigendecomposition

Define matrices

$$V = \begin{bmatrix} | & | & & | \\ \mathbf{v}_{(1)} & \mathbf{v}_{(2)} & \cdots & \mathbf{v}_{(n)} \\ | & | & & | \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_{(1)} & & & \\ & \lambda_{(2)} & & \\ & & \ddots & \\ & & & \lambda_{(n)} \end{bmatrix}$$

Then we can write the eigenvector definition as:

$$SV = V\Lambda$$

For symmetric S (i.e. orthonormal V):

$$S = SVV^{\mathsf{T}} = V\Lambda V^{\mathsf{T}}$$

Eigendecomposition

Define matrices

$$V = \begin{bmatrix} | & | & & | \\ \mathbf{v}_{(1)} & \mathbf{v}_{(2)} & \cdots & \mathbf{v}_{(n)} \\ | & | & & | \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_{(1)} & & & \\ & \lambda_{(2)} & & \\ & & \ddots & \\ & & & \lambda_{(n)} \end{bmatrix}$$

Then we can write the eigenvector definition as:

$$SV = V\Lambda$$

For symmetric S (i.e. orthonormal V):

$$S = SVV^{\mathsf{T}} = V\Lambda V^{\mathsf{T}} = \sum_i \lambda_{(i)} \mathbf{v}_{(i)} \mathbf{v}_{(i)}{}^{\mathsf{T}}$$

Eigendecomposition

Define matrices

$$V = \begin{bmatrix} | & | & & | \\ \mathbf{v}_{(1)} & \mathbf{v}_{(2)} & \cdots & \mathbf{v}_{(n)} \\ | & | & & | \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_{(1)} & & & \\ & \lambda_{(2)} & & \\ & & \ddots & \\ & & & \lambda_{(n)} \end{bmatrix}$$

Then we can write the eigenvector definition as:

$$SV = V\Lambda$$

For symmetric S (i.e. orthonormal V):

$$S = SVV^{\mathsf{T}} = V\Lambda V^{\mathsf{T}} = \sum_i \lambda_{(i)} \mathbf{v}_{(i)} \mathbf{v}_{(i)}^{\mathsf{T}}$$

This is called the eigendecomposition of the matrix.

Finding eigenvectors and eigenvalues

In theory

Just algebra.

$$S\mathbf{v} = \lambda\mathbf{v} \;\Rightarrow\; (S - \lambda I)\mathbf{v} = \mathbf{0}$$

Solve:

$$|S - \lambda I| = 0$$

(polynomial in λ) to find eigvals.

Solve:

$$(S - \lambda_{(i)} I)\mathbf{v} = 0$$

(linear system) to find eigvecs.

In practice

Use a linear algebra package.
In MATLAB

```
[V, L] = eig(S)
```

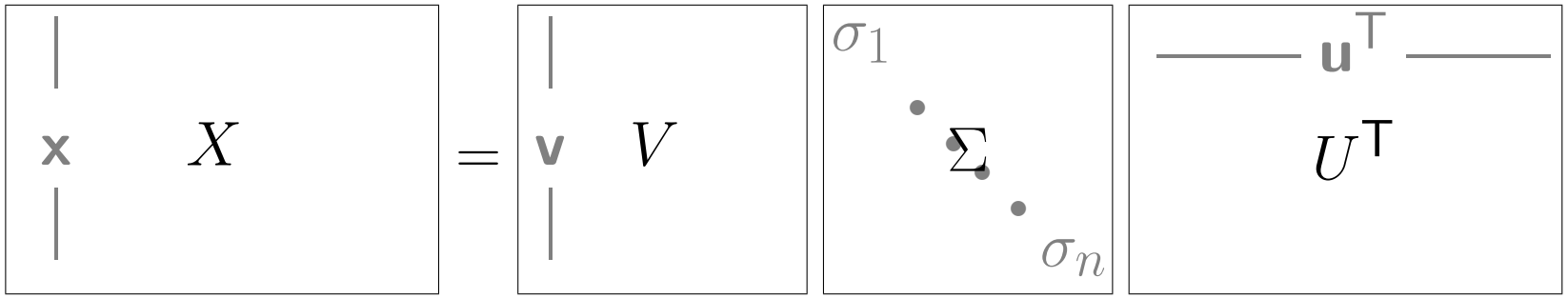
returns the matrices V (`V`) and Λ (`L`) defined before.

`eig` usually returns eigvals in increasing order, but don't count on it.

`eigs` can find largest or smallest k eigvals (and corresponding eigvecs).

The Singular Value Decomposition (SVD)

The SVD resembles an eigendecomposition, but is defined for rectangular matrices as well.



with

- $VV^{\mathsf{T}} = V^{\mathsf{T}}V = I$ an orthonormal basis $\{\mathbf{v}_i\}$ for the columns of X
- $U^{\mathsf{T}}U = I$ an orthonormal basis $\{\mathbf{u}_i\}$ for the rows of X
- Σ diagonal, and containing the singular values in decreasing order ($\sigma_1 > \sigma_2 > \cdots > \sigma_n$)

SVD and Eigendecompositions

We have:

$$S = \frac{1}{m} X X^{\mathsf{T}} = \frac{1}{m}(V \Sigma U^{\mathsf{T}})(U \Sigma V^{\mathsf{T}}) = V \left(\frac{1}{m} \Sigma^2 \right) V^{\mathsf{T}}$$

Comparing to the (unique, up to permutation) eigendecomposition of S

$\Rightarrow$ The eigenvectors of S are the left singular vectors of X.

$\Rightarrow$ The eigenvalues of S are given by $\lambda_{(i)} = \sigma_i^2 / m$.

Similarly

$$G = X^{\mathsf{T}} X = (U \Sigma V^{\mathsf{T}})(V \Sigma U^{\mathsf{T}}) = U \, \Sigma^2 \, U^{\mathsf{T}}$$

$\Rightarrow$ The eigenvectors of G are the right singular vectors of X.

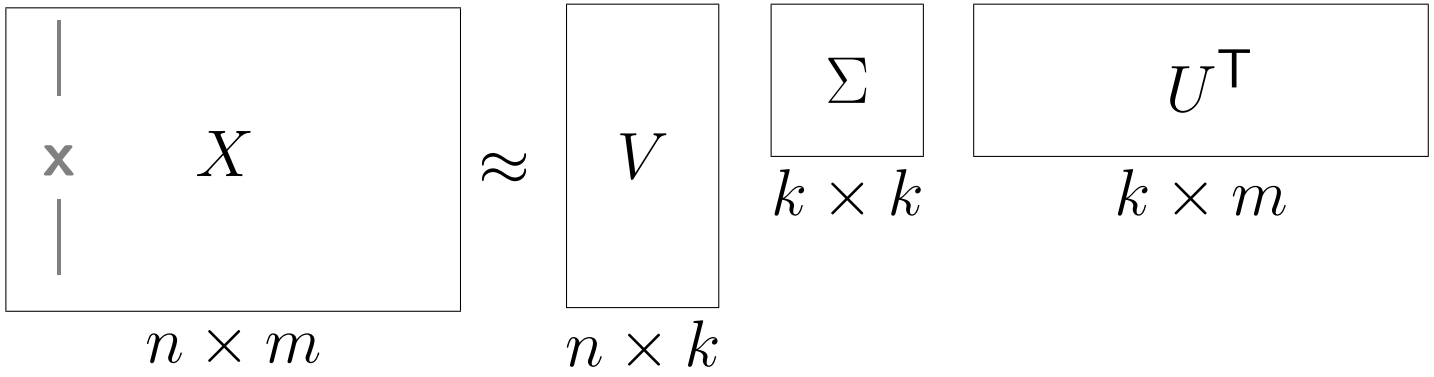
$\Rightarrow$ The eigenvalues of G are given by $\lambda_{(i)} = \sigma_i^2$.

(And the SVD of S or G equals the corresponding eigendecomposition.)

Approximating matrices

The SVD has an important property:

$\widetilde{X} = \sum_{i=1}^{k} \sigma_i \mathbf{v}_i \mathbf{u}_i^\mathsf{T}$ is the best rank-k least-squares-approximation to X.



That is, if we seek $\widetilde{V} \in \mathbb{R}^{n \times k}$ and $\widetilde{U} \in \mathbb{R}^{m \times k}$ such that $k < \min(m, n)$ and

$$\mathcal{E} = \sum_{ij} (x_{ij} - [\widetilde{V}\widetilde{U}^\mathsf{T}]_{ij})^2$$ is minimised,

then the answer is "proportional" to:

$$\widetilde{V} = [V]_{1:n,1:k} \, [\Sigma^{1/2}]_{1:k,1:k}; \quad \widetilde{U} = [U]_{1:m,1:k} \, [\Sigma^{1/2}]_{1:k,1:k}.$$

["Proportional" means we can right-multiply $\widetilde{V}$ by any non-singular $A \in \mathbb{R}^{k \times k}$ if we also right-multiply $\widetilde{U}$ by $(A^{-1})^\mathsf{T}$.]

Approximating matrices

The SVD has an important property:

$\widetilde{X} = \sum_{i=1}^{k} \sigma_i \mathbf{v}_i \mathbf{u}_i^\mathsf{T}$ is the best rank-k least-squares-approximation to X.

A corollary:

$$\mathcal{E} = \sum_{ij} (S_{ij} - [PP^\mathsf{T}]_{ij})^2$$

is minimised for $P \in \mathbb{R}^{n \times k}$ if

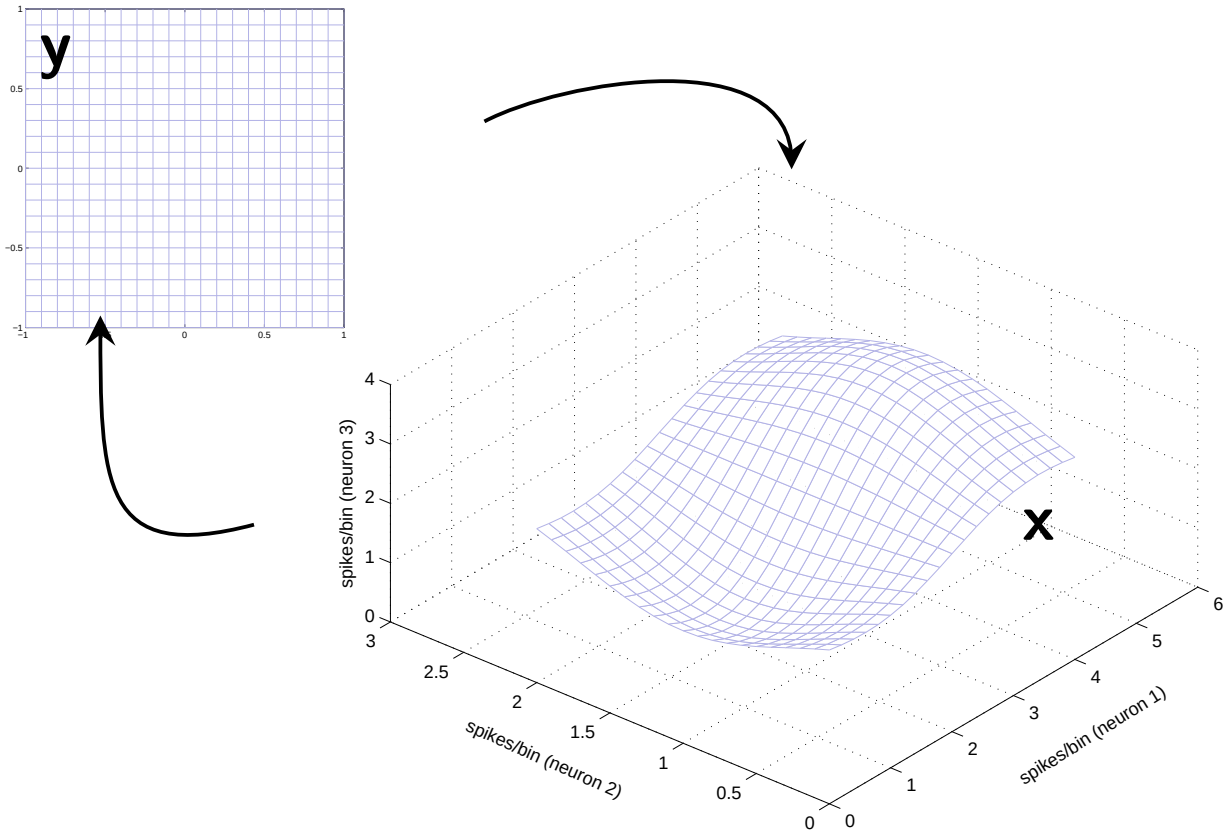
$$P = [V]_{1:n,1:k} [\Lambda^{1/2}]_{1:k,1:k}$$

Back to business

Dimensionality Reduction

Goal: Find the manifold.

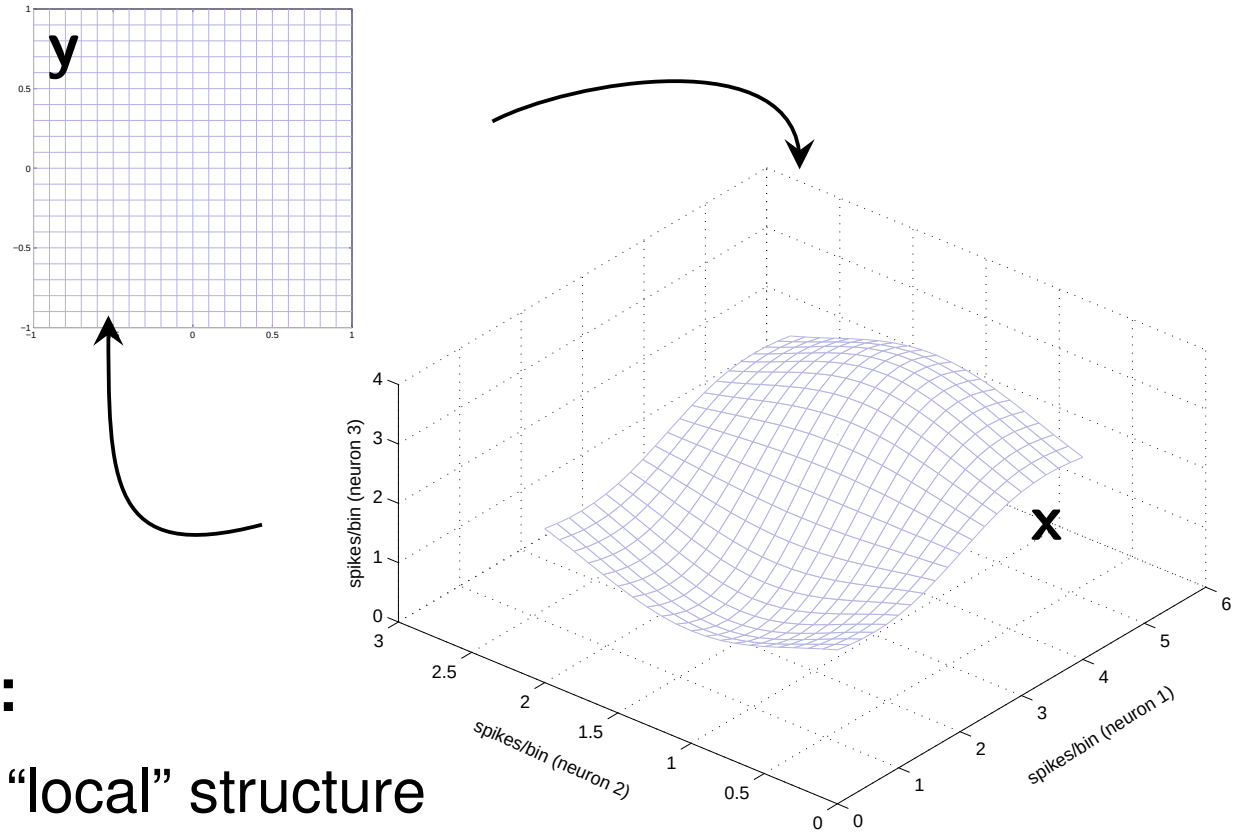
More precisely, find $\mathbf{y}_i \in \mathbb{R}^k, \ (k < n)$ so that $\mathbf{y}_i$ parameterises the location of $\mathbf{x}_i$ on the manifold.



Dimensionality Reduction

Goal: Find the manifold.

More precisely, find $\mathbf{y}_i \in \mathbb{R}^k, \ (k < n)$ so that $\mathbf{y}_i$ parameterises the location of $\mathbf{x}_i$ on the manifold.



Core ideas:

- preserve “local” structure
- preserve “information”

Linear (old) methods

A Linear Approach

Let $\mathbf{y}_i = P^\mathsf{T}\mathbf{x}_i$ for a projection matrix P^T.

$P \in \mathbb{R}^{n\times k}$ defines a linear mapping from data to manifold, and vice versa.

Linearity

- preserves local structure
- preserves global structure

Idea: look for projection that keeps data as spread out as possible $\Rightarrow$ most variance.

- preserves "information"

Principal Components Analysis (PCA)

Idea: look for projection that keeps data as spread out as possible $\Rightarrow$ most variance:

- Find direction of greatest variance – $\boldsymbol{\rho}_{(1)}$.

$$\boldsymbol{\rho}_{(1)} = \underset{\|\mathbf{u}\|=1}{\operatorname{argmax}} \sum_i (\mathbf{x}_i^\mathsf{T} \mathbf{u})^2$$

- Find direction orthogonal to $\boldsymbol{\rho}_{(1)}$ with greatest variance – $\boldsymbol{\rho}_{(2)}$

⋮

- Find direction orthogonal to $\{\boldsymbol{\rho}_{(1)}, \boldsymbol{\rho}_{(2)}, \ldots, \boldsymbol{\rho}_{(j-1)}\}$ with greatest variance – $\boldsymbol{\rho}_{(j)}$.
- Terminate when remaining variance drops below a threshold.

PCA and Eigenvectors

- The variance in eigendirection $\mathbf{v}_{(i)}$ is

PCA and Eigenvectors

- The variance in eigendirection $\mathbf{v}_{(i)}$ is

$$\left\langle (\mathbf{x}^{\mathsf{T}}\mathbf{v}_{(i)})^2 \right\rangle$$

PCA and Eigenvectors

- The variance in eigendirection $\mathbf{v}_{(i)}$ is

$$\left\langle (\mathbf{x}^{\mathsf{T}}\mathbf{v}_{(i)})^2 \right\rangle = \left\langle \mathbf{v}_{(i)}{}^{\mathsf{T}}\mathbf{x}\mathbf{x}^{\mathsf{T}}\mathbf{v}_{(i)} \right\rangle$$

PCA and Eigenvectors

- The variance in eigendirection $\mathbf{v}_{(i)}$ is

$$\left\langle (\mathbf{x}^{\mathsf{T}}\mathbf{v}_{(i)})^2 \right\rangle = \left\langle \mathbf{v}_{(i)}{}^{\mathsf{T}}\mathbf{x}\mathbf{x}^{\mathsf{T}}\mathbf{v}_{(i)} \right\rangle = \mathbf{v}_{(i)}{}^{\mathsf{T}} S \mathbf{v}_{(i)}$$

PCA and Eigenvectors

- The variance in eigendirection $\mathbf{v}_{(i)}$ is

$$\left\langle (\mathbf{x}^{\mathsf{T}}\mathbf{v}_{(i)})^2 \right\rangle = \left\langle \mathbf{v}_{(i)}{}^{\mathsf{T}}\mathbf{x}\mathbf{x}^{\mathsf{T}}\mathbf{v}_{(i)} \right\rangle = \mathbf{v}_{(i)}{}^{\mathsf{T}} S \mathbf{v}_{(i)} = \mathbf{v}_{(i)}{}^{\mathsf{T}} \lambda_{(i)} \mathbf{v}_{(i)} = \lambda_{(i)}$$

PCA and Eigenvectors

- The variance in eigendirection $\mathbf{v}_{(i)}$ is

$$\left\langle (\mathbf{x}^{\mathsf{T}}\mathbf{v}_{(i)})^2 \right\rangle = \left\langle \mathbf{v}_{(i)}{}^{\mathsf{T}}\mathbf{x}\mathbf{x}^{\mathsf{T}}\mathbf{v}_{(i)} \right\rangle = \mathbf{v}_{(i)}{}^{\mathsf{T}} S \mathbf{v}_{(i)} = \mathbf{v}_{(i)}{}^{\mathsf{T}} \lambda_{(i)} \mathbf{v}_{(i)} = \lambda_{(i)}$$

- The variance in an arbitrary direction $\mathbf{u}$ is

PCA and Eigenvectors

- The variance in eigendirection $\mathbf{v}_{(i)}$ is

$$\left\langle (\mathbf{x}^{\mathsf{T}}\mathbf{v}_{(i)})^2 \right\rangle = \left\langle \mathbf{v}_{(i)}{}^{\mathsf{T}}\mathbf{x}\mathbf{x}^{\mathsf{T}}\mathbf{v}_{(i)} \right\rangle = \mathbf{v}_{(i)}{}^{\mathsf{T}} S \mathbf{v}_{(i)} = \mathbf{v}_{(i)}{}^{\mathsf{T}} \lambda_{(i)} \mathbf{v}_{(i)} = \lambda_{(i)}$$

- The variance in an arbitrary direction $\mathbf{u}$ is

$$\left\langle (\mathbf{x}^{\mathsf{T}}\mathbf{u})^2 \right\rangle$$

PCA and Eigenvectors

- The variance in eigendirection $\mathbf{v}_{(i)}$ is

$$\left\langle (\mathbf{x}^{\mathsf{T}}\mathbf{v}_{(i)})^2 \right\rangle = \left\langle \mathbf{v}_{(i)}{}^{\mathsf{T}}\mathbf{x}\mathbf{x}^{\mathsf{T}}\mathbf{v}_{(i)} \right\rangle = \mathbf{v}_{(i)}{}^{\mathsf{T}} S \mathbf{v}_{(i)} = \mathbf{v}_{(i)}{}^{\mathsf{T}} \lambda_{(i)} \mathbf{v}_{(i)} = \lambda_{(i)}$$

- The variance in an arbitrary direction $\mathbf{u}$ is

$$\left\langle (\mathbf{x}^{\mathsf{T}}\mathbf{u})^2 \right\rangle = \left\langle \left(\mathbf{x}^{\mathsf{T}} \left(\sum_i u_{(i)} \mathbf{v}_{(i)} \right) \right)^2 \right\rangle$$

PCA and Eigenvectors

- The variance in eigendirection $\mathbf{v}_{(i)}$ is

$$\left\langle (\mathbf{x}^{\mathsf{T}}\mathbf{v}_{(i)})^2 \right\rangle = \left\langle \mathbf{v}_{(i)}{}^{\mathsf{T}}\mathbf{x}\mathbf{x}^{\mathsf{T}}\mathbf{v}_{(i)} \right\rangle = \mathbf{v}_{(i)}{}^{\mathsf{T}} S \mathbf{v}_{(i)} = \mathbf{v}_{(i)}{}^{\mathsf{T}} \lambda_{(i)} \mathbf{v}_{(i)} = \lambda_{(i)}$$

- The variance in an arbitrary direction $\mathbf{u}$ is

$$\left\langle (\mathbf{x}^{\mathsf{T}}\mathbf{u})^2 \right\rangle = \left\langle \left(\mathbf{x}^{\mathsf{T}} \left(\sum_i u_{(i)} \mathbf{v}_{(i)} \right) \right)^2 \right\rangle = \sum_{ij} u_{(i)} \mathbf{v}_{(i)}{}^{\mathsf{T}} S \mathbf{v}_{(j)} u_{(j)}$$

PCA and Eigenvectors

- The variance in eigendirection $\mathbf{v}_{(i)}$ is

$$\left\langle (\mathbf{x}^{\mathsf{T}}\mathbf{v}_{(i)})^2 \right\rangle = \left\langle \mathbf{v}_{(i)}{}^{\mathsf{T}}\mathbf{x}\mathbf{x}^{\mathsf{T}}\mathbf{v}_{(i)} \right\rangle = \mathbf{v}_{(i)}{}^{\mathsf{T}} S \mathbf{v}_{(i)} = \mathbf{v}_{(i)}{}^{\mathsf{T}} \lambda_{(i)} \mathbf{v}_{(i)} = \lambda_{(i)}$$

- The variance in an arbitrary direction $\mathbf{u}$ is

$$\left\langle (\mathbf{x}^{\mathsf{T}}\mathbf{u})^2 \right\rangle = \left\langle \left(\mathbf{x}^{\mathsf{T}} \left(\sum_i u_{(i)} \mathbf{v}_{(i)} \right) \right)^2 \right\rangle = \sum_{ij} u_{(i)} \mathbf{v}_{(i)}{}^{\mathsf{T}} S \mathbf{v}_{(j)} u_{(j)}$$

$$= \sum_{ij} u_{(i)} \lambda_{(j)} u_{(j)} \mathbf{v}_{(i)}{}^{\mathsf{T}} \mathbf{v}_{(j)}$$

PCA and Eigenvectors

- The variance in eigendirection $\mathbf{v}_{(i)}$ is

$$\left\langle (\mathbf{x}^{\mathsf{T}}\mathbf{v}_{(i)})^2 \right\rangle = \left\langle \mathbf{v}_{(i)}{}^{\mathsf{T}}\mathbf{x}\mathbf{x}^{\mathsf{T}}\mathbf{v}_{(i)} \right\rangle = \mathbf{v}_{(i)}{}^{\mathsf{T}} S \mathbf{v}_{(i)} = \mathbf{v}_{(i)}{}^{\mathsf{T}} \lambda_{(i)} \mathbf{v}_{(i)} = \lambda_{(i)}$$

- The variance in an arbitrary direction $\mathbf{u}$ is

$$\left\langle (\mathbf{x}^{\mathsf{T}}\mathbf{u})^2 \right\rangle = \left\langle \left(\mathbf{x}^{\mathsf{T}} \left(\sum_i u_{(i)} \mathbf{v}_{(i)} \right) \right)^2 \right\rangle = \sum_{ij} u_{(i)} \mathbf{v}_{(i)}{}^{\mathsf{T}} S \mathbf{v}_{(j)} u_{(j)}$$

$$= \sum_{ij} u_{(i)} \lambda_{(j)} u_{(j)} \mathbf{v}_{(i)}{}^{\mathsf{T}} \mathbf{v}_{(j)} = \sum_i u_{(i)}^2 \lambda_{(i)}$$

PCA and Eigenvectors

- The variance in eigendirection $\mathbf{v}_{(i)}$ is

$$\left\langle (\mathbf{x}^\mathsf{T}\mathbf{v}_{(i)})^2 \right\rangle = \left\langle \mathbf{v}_{(i)}{}^\mathsf{T}\mathbf{x}\mathbf{x}^\mathsf{T}\mathbf{v}_{(i)} \right\rangle = \mathbf{v}_{(i)}{}^\mathsf{T} S \mathbf{v}_{(i)} = \mathbf{v}_{(i)}{}^\mathsf{T} \lambda_{(i)} \mathbf{v}_{(i)} = \lambda_{(i)}$$

- The variance in an arbitrary direction $\mathbf{u}$ is

$$\left\langle (\mathbf{x}^\mathsf{T}\mathbf{u})^2 \right\rangle = \left\langle \left(\mathbf{x}^\mathsf{T} \left(\sum_i u_{(i)} \mathbf{v}_{(i)} \right) \right)^2 \right\rangle = \sum_{ij} u_{(i)} \mathbf{v}_{(i)}{}^\mathsf{T} S \mathbf{v}_{(j)} u_{(j)}$$

$$= \sum_{ij} u_{(i)} \lambda_{(j)} u_{(j)} \mathbf{v}_{(i)}{}^\mathsf{T} \mathbf{v}_{(j)} = \sum_i u_{(i)}^2 \lambda_{(i)}$$

- If $\mathbf{u}^\mathsf{T}\mathbf{u} = 1$, then $\sum_i u_{(i)}^2 = 1$ and so $\operatorname{argmax}_{\|\mathbf{u}\|=1} \left\langle (\mathbf{x}^\mathsf{T}\mathbf{u})^2 \right\rangle = \mathbf{v}_{(\text{max})}$

PCA and Eigenvectors

- The variance in eigendirection $\mathbf{v}_{(i)}$ is

$$\left\langle (\mathbf{x}^\mathsf{T}\mathbf{v}_{(i)})^2 \right\rangle = \left\langle \mathbf{v}_{(i)}{}^\mathsf{T}\mathbf{x}\mathbf{x}^\mathsf{T}\mathbf{v}_{(i)} \right\rangle = \mathbf{v}_{(i)}{}^\mathsf{T} S \mathbf{v}_{(i)} = \mathbf{v}_{(i)}{}^\mathsf{T} \lambda_{(i)} \mathbf{v}_{(i)} = \lambda_{(i)}$$

- The variance in an arbitrary direction $\mathbf{u}$ is

$$\left\langle (\mathbf{x}^\mathsf{T}\mathbf{u})^2 \right\rangle = \left\langle \left(\mathbf{x}^\mathsf{T} \left(\sum_i u_{(i)} \mathbf{v}_{(i)} \right) \right)^2 \right\rangle = \sum_{ij} u_{(i)} \mathbf{v}_{(i)}{}^\mathsf{T} S \mathbf{v}_{(j)} u_{(j)}$$

$$= \sum_{ij} u_{(i)} \lambda_{(j)} u_{(j)} \mathbf{v}_{(i)}{}^\mathsf{T} \mathbf{v}_{(j)} = \sum_i u_{(i)}^2 \lambda_{(i)}$$

- If $\mathbf{u}^\mathsf{T}\mathbf{u} = 1$, then $\sum_i u_{(i)}^2 = 1$ and so $\operatorname{argmax}_{\|\mathbf{u}\|=1} \left\langle (\mathbf{x}^\mathsf{T}\mathbf{u})^2 \right\rangle = \mathbf{v}_{(\text{max})}$

The direction of greatest variance is the eigenvector corresponding to the largest eigenvalue.

PCA and Eigenvectors

- The variance in eigendirection $\mathbf{v}_{(i)}$ is

$$\left\langle (\mathbf{x}^\mathsf{T}\mathbf{v}_{(i)})^2 \right\rangle = \left\langle \mathbf{v}_{(i)}{}^\mathsf{T}\mathbf{x}\mathbf{x}^\mathsf{T}\mathbf{v}_{(i)} \right\rangle = \mathbf{v}_{(i)}{}^\mathsf{T} S \mathbf{v}_{(i)} = \mathbf{v}_{(i)}{}^\mathsf{T} \lambda_{(i)} \mathbf{v}_{(i)} = \lambda_{(i)}$$

- The variance in an arbitrary direction $\mathbf{u}$ is

$$\left\langle (\mathbf{x}^\mathsf{T}\mathbf{u})^2 \right\rangle = \left\langle \left(\mathbf{x}^\mathsf{T} \left(\sum_i u_{(i)} \mathbf{v}_{(i)} \right) \right)^2 \right\rangle = \sum_{ij} u_{(i)} \mathbf{v}_{(i)}{}^\mathsf{T} S \mathbf{v}_{(j)} u_{(j)}$$

$$= \sum_{ij} u_{(i)} \lambda_{(j)} u_{(j)} \mathbf{v}_{(i)}{}^\mathsf{T} \mathbf{v}_{(j)} = \sum_i u_{(i)}^2 \lambda_{(i)}$$

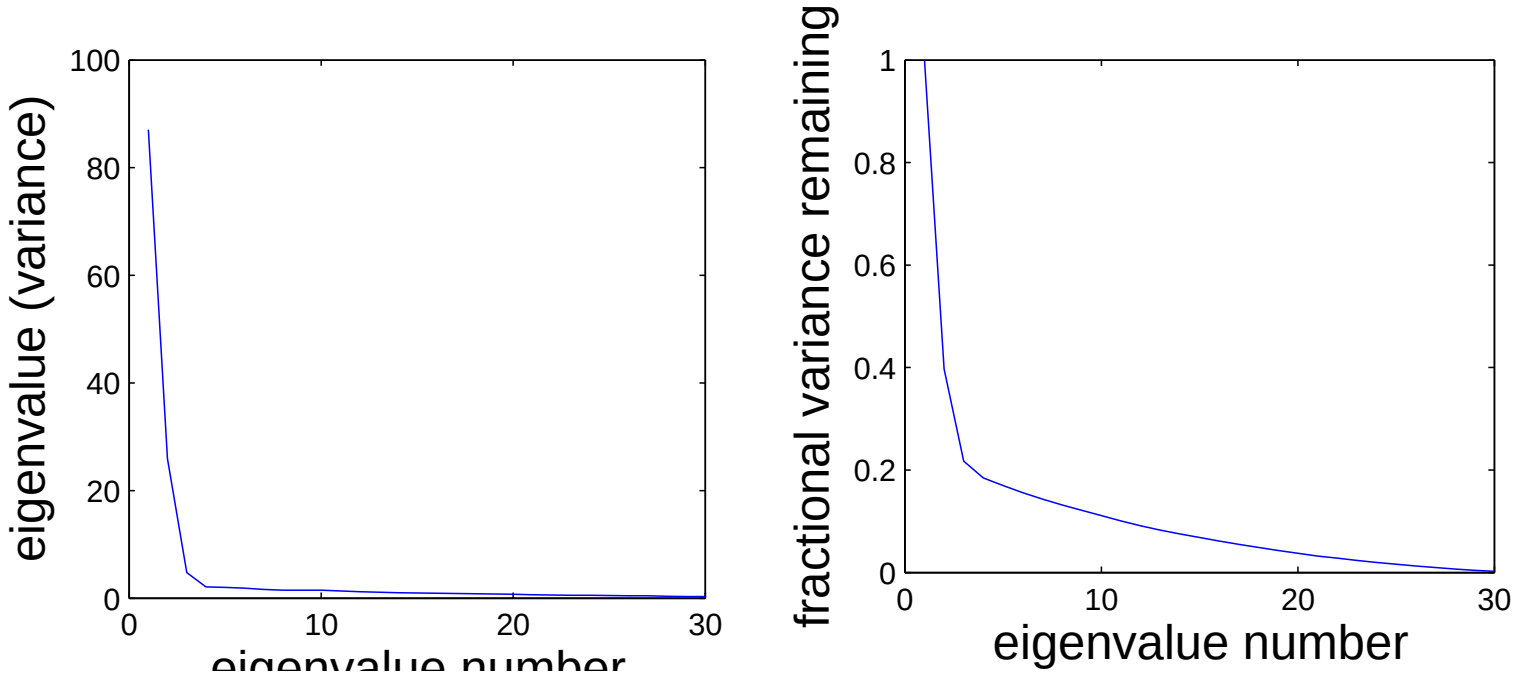
- If $\mathbf{u}^\mathsf{T}\mathbf{u} = 1$, then $\sum_i u_{(i)}^2 = 1$ and so $\operatorname{argmax}_{\|\mathbf{u}\|=1} \left\langle (\mathbf{x}^\mathsf{T}\mathbf{u})^2 \right\rangle = \mathbf{v}_{(\text{max})}$

The direction of greatest variance is the eigenvector corresponding to the largest eigenvalue.

- In general, the PCs are exactly the eigenvectors of the empirical covariance matrix, ordered by decreasing eigenvalue.

PCA and Eigenvectors

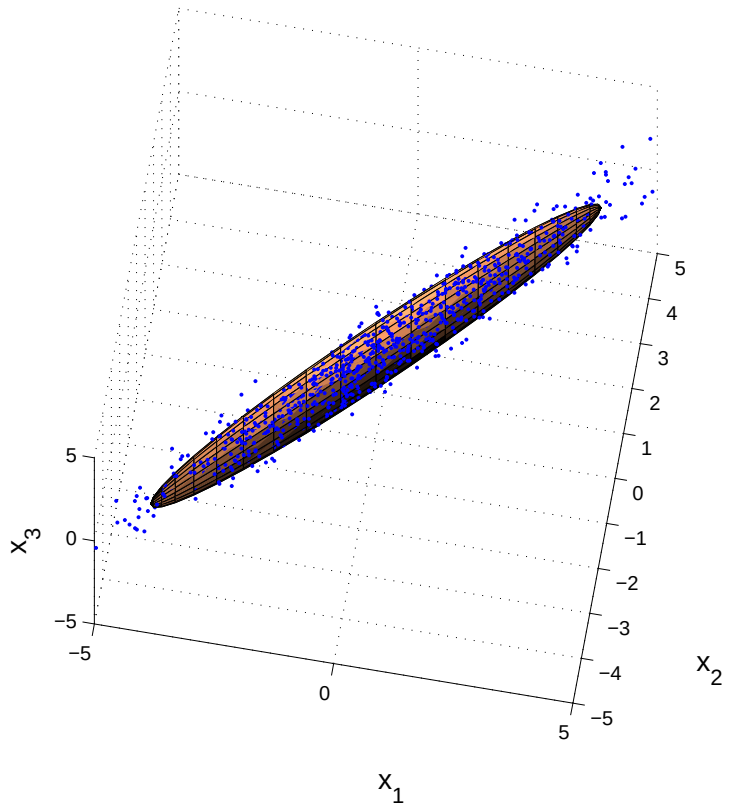
The eigenspectrum shows how the variance is distributed across dimensions



This can be used to estimate the “right” dimensionality.

PCA manifold or hyperplane

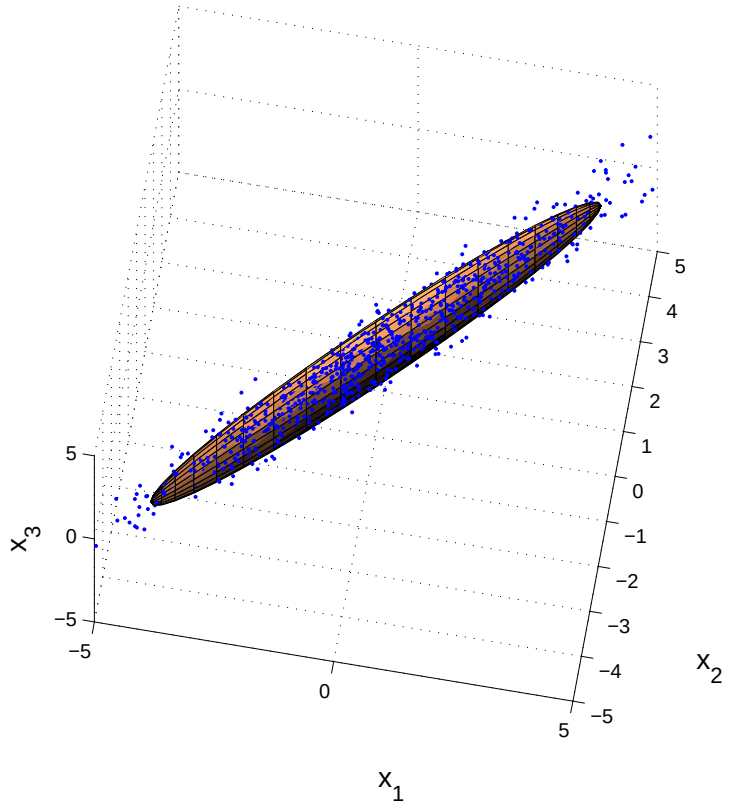
The k principal components define a k-dimensional linear manifold.



PCA manifold or hyperplane

The k principal components define a k-dimensional linear manifold.

manifold coordinates: $\mathbf{y} = P^{\mathsf{T}}\mathbf{x} \quad P = \left[\boldsymbol{\rho}_{(1)}\boldsymbol{\rho}_{(2)}\ldots\boldsymbol{\rho}_{(k)}\right] \quad \left[\mathbf{y} \in \mathbb{R}^k\right]$

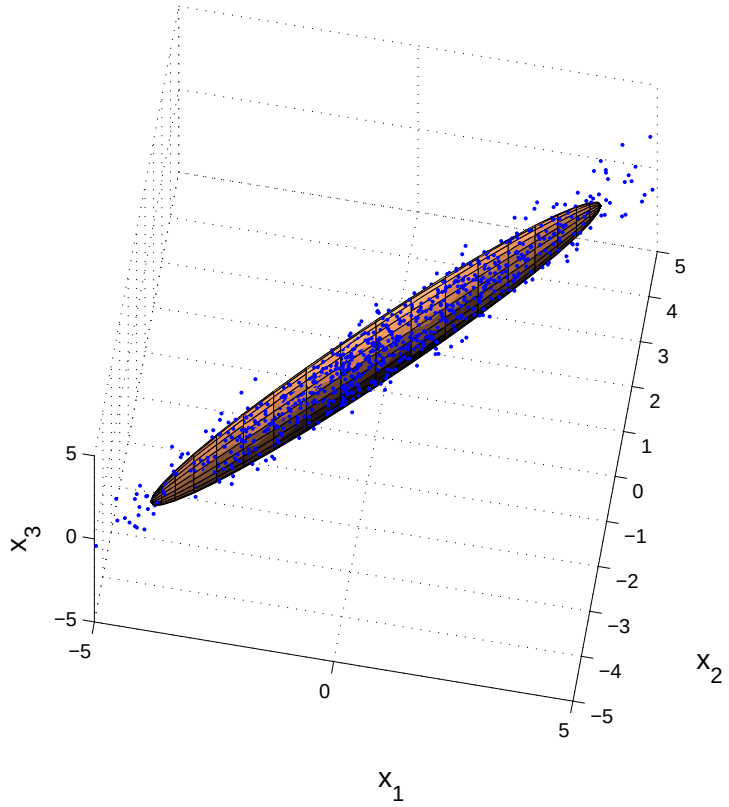


PCA manifold or hyperplane

The k principal components define a k-dimensional linear manifold.

manifold coordinates: $\mathbf{y} = P^{\mathsf{T}}\mathbf{x} \quad P = \left[\boldsymbol{\rho}_{(1)}\boldsymbol{\rho}_{(2)}\cdots\boldsymbol{\rho}_{(k)}\right] \quad \left[\mathbf{y} \in \mathbb{R}^k\right]$

hyperplane projection: $\hat{\mathbf{x}} = P\mathbf{y} = \sum_{\kappa=1}^{k} y_\kappa \boldsymbol{\rho}_{(\kappa)} = PP^{\mathsf{T}}\mathbf{x} \quad \left[\hat{\mathbf{x}} \in \mathbb{R}^n\right]$

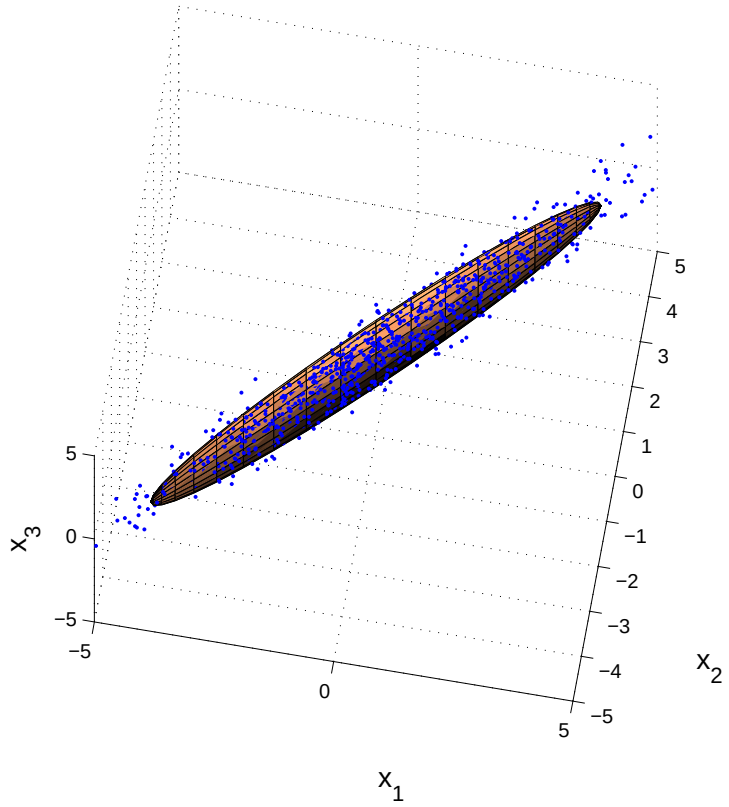


PCA manifold or hyperplane

The k principal components define a k-dimensional linear manifold.

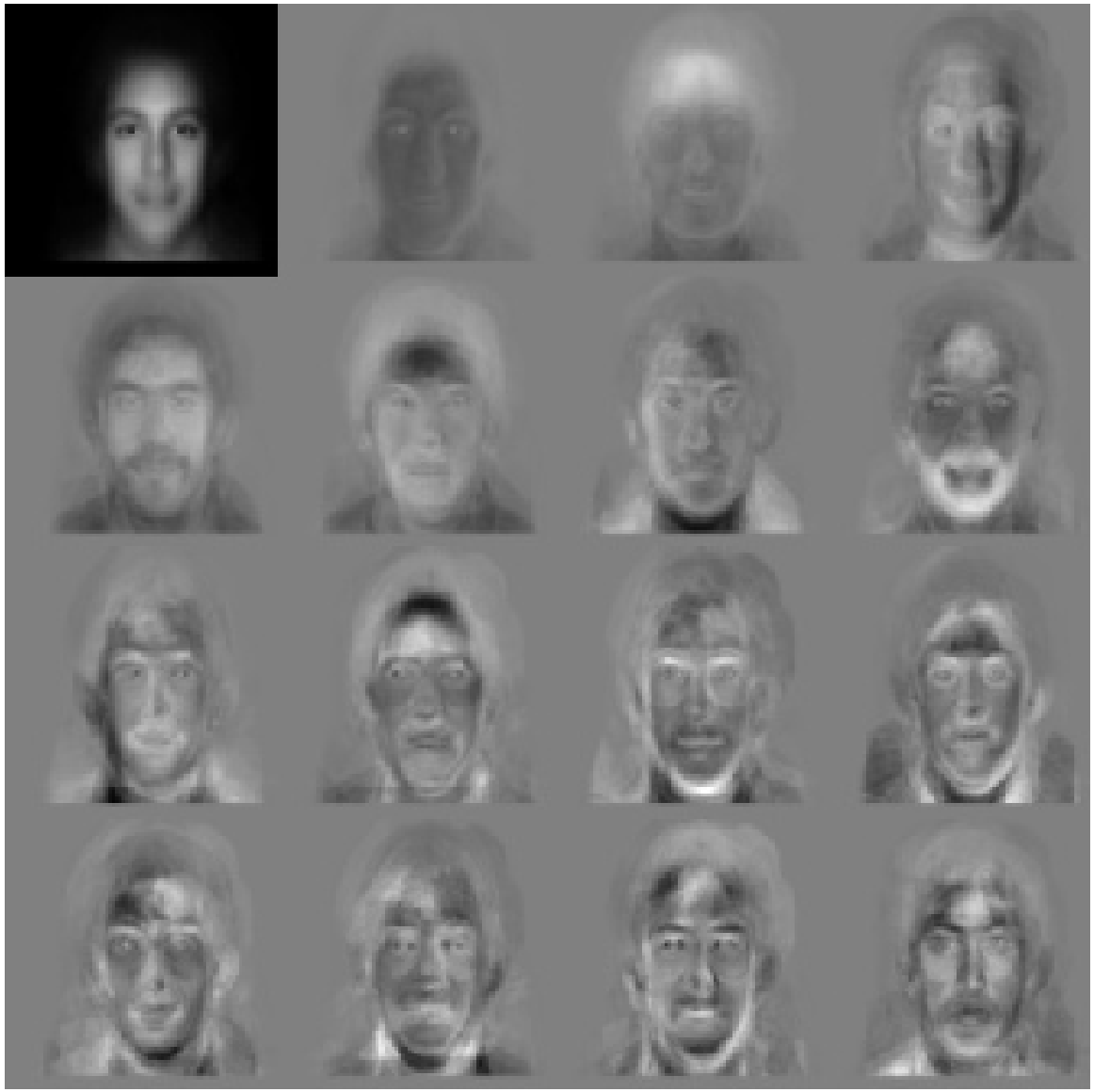
manifold coordinates: $\mathbf{y} = P^{\mathsf{T}}\mathbf{x} \quad P = \left[\boldsymbol{\rho}_{(1)}\boldsymbol{\rho}_{(2)}\cdots\boldsymbol{\rho}_{(k)}\right] \quad \left[\mathbf{y} \in \mathbb{R}^k\right]$

hyperplane projection: $\hat{\mathbf{x}} = P\mathbf{y} = \sum_{\kappa=1}^{k} y_\kappa \boldsymbol{\rho}_{(\kappa)} = PP^{\mathsf{T}}\mathbf{x} \quad \left[\hat{\mathbf{x}} \in \mathbb{R}^n\right]$



The projection can be used for lossy compression, denoising, . . .

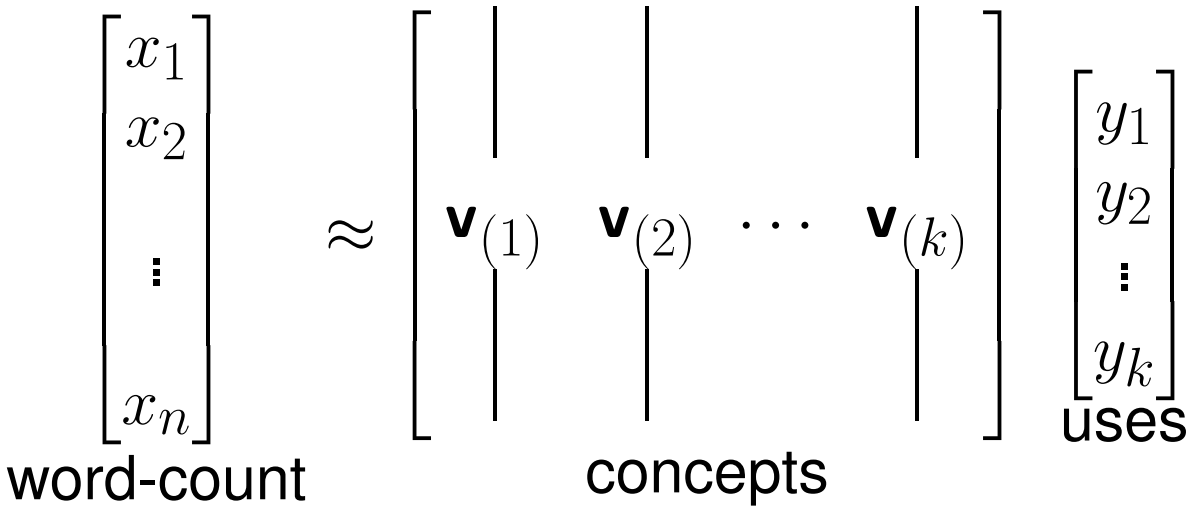
Example of PCA: Eigenfaces



from www-white.media.mit.edu/vismod/demos/facerec/basic.html

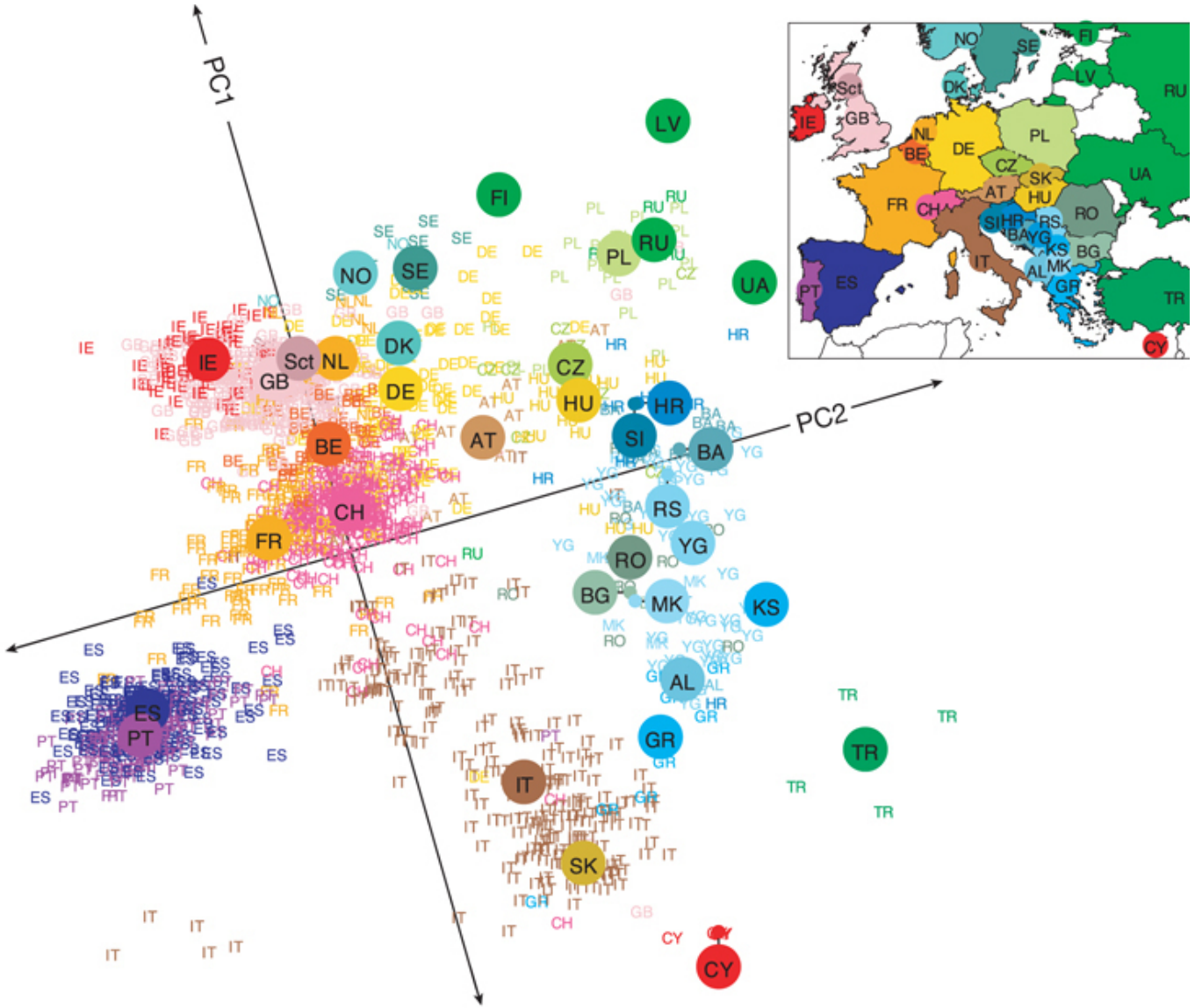
Example of PCA: Latent Semantic Analysis

PCA applied to documents (word-count vectors)



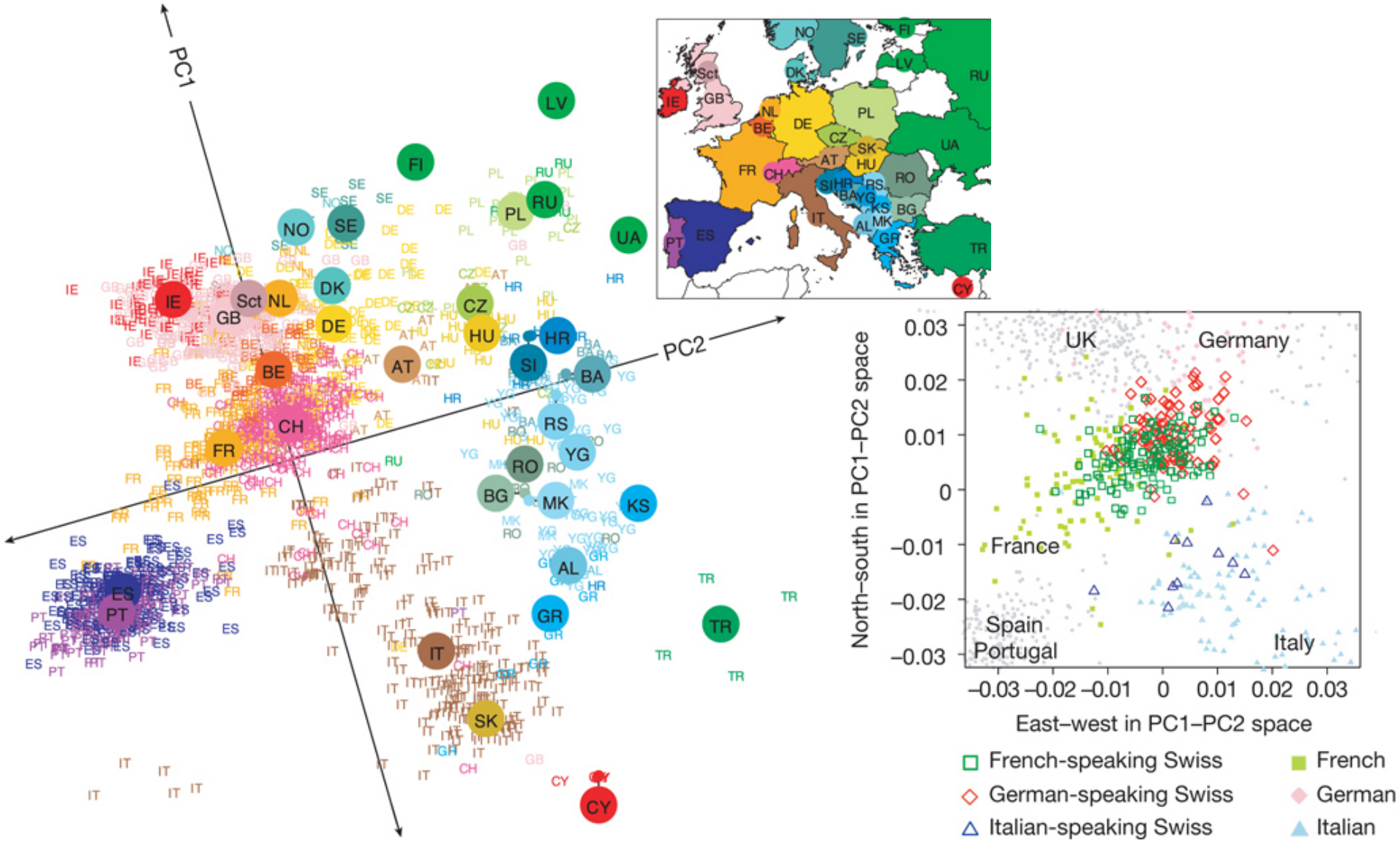
Has been used to mark essays!! Also used for retrieval (called Latent Semantic Indexing (LSI) – reportedly Google uses something like this).

Example of PCA: Genetic variation within Europe



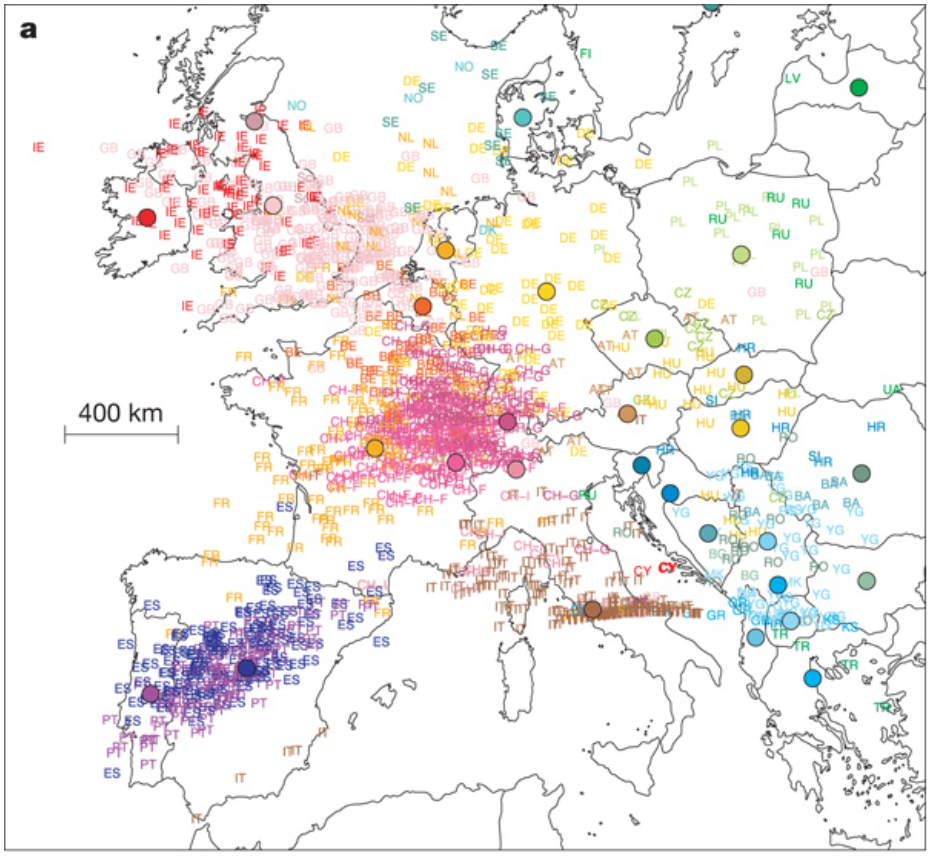
Novembre et al. (2008) Nature 456:98-101

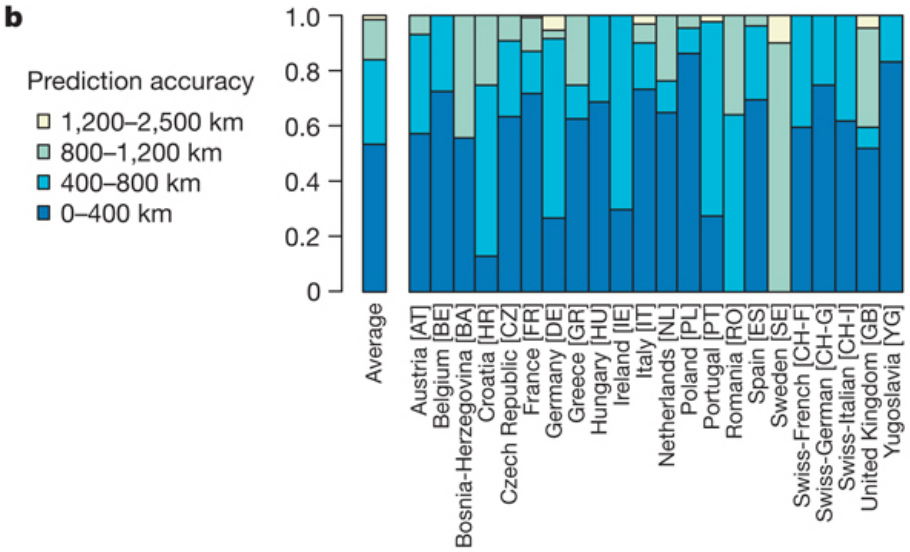
Example of PCA: Genetic variation within Europe



Novembre et al. (2008) Nature 456:98-101

Example of PCA: Genetic variation within Europe





Novembre et al. (2008) Nature 456:98-101

Variance partitioning

PCA can be seen to partition the data variance into an in-manifold (signal) and out-of-manifold (noise) part.

Variance partitioning

PCA can be seen to partition the data variance into an in-manifold (signal) and out-of-manifold (noise) part.

- all out-of-manifold dimensions are weighted equally
- real noise processes are more likely to be fully isotropic

Variance partitioning

PCA can be seen to partition the data variance into an in-manifold (signal) and out-of-manifold (noise) part.

- all out-of-manifold dimensions are weighted equally
- real noise processes are more likely to be fully isotropic

Two extensions (that we won't discuss further)

Variance partitioning

PCA can be seen to partition the data variance into an in-manifold (signal) and out-of-manifold (noise) part.

- all out-of-manifold dimensions are weighted equally
- real noise processes are more likely to be fully isotropic

Two extensions (that we won't discuss further)

- Include isotropic Gaussian noise, and estimate its scale: probabilistic Principal Components Analysis (pPCA).

Variance partitioning

PCA can be seen to partition the data variance into an in-manifold (signal) and out-of-manifold (noise) part.

- all out-of-manifold dimensions are weighted equally
- real noise processes are more likely to be fully isotropic

Two extensions (that we won't discuss further)

- Include isotropic Gaussian noise, and estimate its scale: probabilistic Principal Components Analysis (pPCA).
  - "Projection" displaced toward mean to compensate for noise.

Variance partitioning

PCA can be seen to partition the data variance into an in-manifold (signal) and out-of-manifold (noise) part.

- all out-of-manifold dimensions are weighted equally
- real noise processes are more likely to be fully isotropic

Two extensions (that we won't discuss further)

- Include isotropic Gaussian noise, and estimate its scale: probabilistic Principal Components Analysis (pPCA).
  - "Projection" displaced toward mean to compensate for noise.
  - Easier to combine into hierarchical models.

Variance partitioning

PCA can be seen to partition the data variance into an in-manifold (signal) and out-of-manifold (noise) part.

- all out-of-manifold dimensions are weighted equally
- real noise processes are more likely to be fully isotropic

Two extensions (that we won't discuss further)

- Include isotropic Gaussian noise, and estimate its scale: probabilistic Principal Components Analysis (pPCA).
  - "Projection" displaced toward mean to compensate for noise.
  - Easier to combine into hierarchical models.
- Allow independent Gaussian noise along each (measured) dimension: Factor Analysis (FA).

Variance partitioning

PCA can be seen to partition the data variance into an in-manifold (signal) and out-of-manifold (noise) part.

- all out-of-manifold dimensions are weighted equally
- real noise processes are more likely to be fully isotropic

Two extensions (that we won't discuss further)

- Include isotropic Gaussian noise, and estimate its scale: probabilistic Principal Components Analysis (pPCA).
  - "Projection" displaced toward mean to compensate for noise.
  - Easier to combine into hierarchical models.
- Allow independent Gaussian noise along each (measured) dimension: Factor Analysis (FA).
  - Sensible when measured quantities are not in comparable units.

Variance partitioning

PCA can be seen to partition the data variance into an in-manifold (signal) and out-of-manifold (noise) part.

- all out-of-manifold dimensions are weighted equally
- real noise processes are more likely to be fully isotropic

Two extensions (that we won't discuss further)

- Include isotropic Gaussian noise, and estimate its scale: probabilistic Principal Components Analysis (pPCA).
  - "Projection" displaced toward mean to compensate for noise.
  - Easier to combine into hierarchical models.
- Allow independent Gaussian noise along each (measured) dimension: Factor Analysis (FA).
  - Sensible when measured quantities are not in comparable units.
  - Note: PCA is often applied with rescaled measurements of equal variance $\Rightarrow$ sum of signal and noise variance isotropic, but noise variance is still unequal. Still, it's better than nothing.

Another view of PCA: Minimizing Error

We can implement the "preserve information" criterion more directly.

Idea: Find $P \in \mathbb{R}^{n \times k}$ and $\mathbf{y}_i \in \mathbb{R}^k$ so that reconstruction error

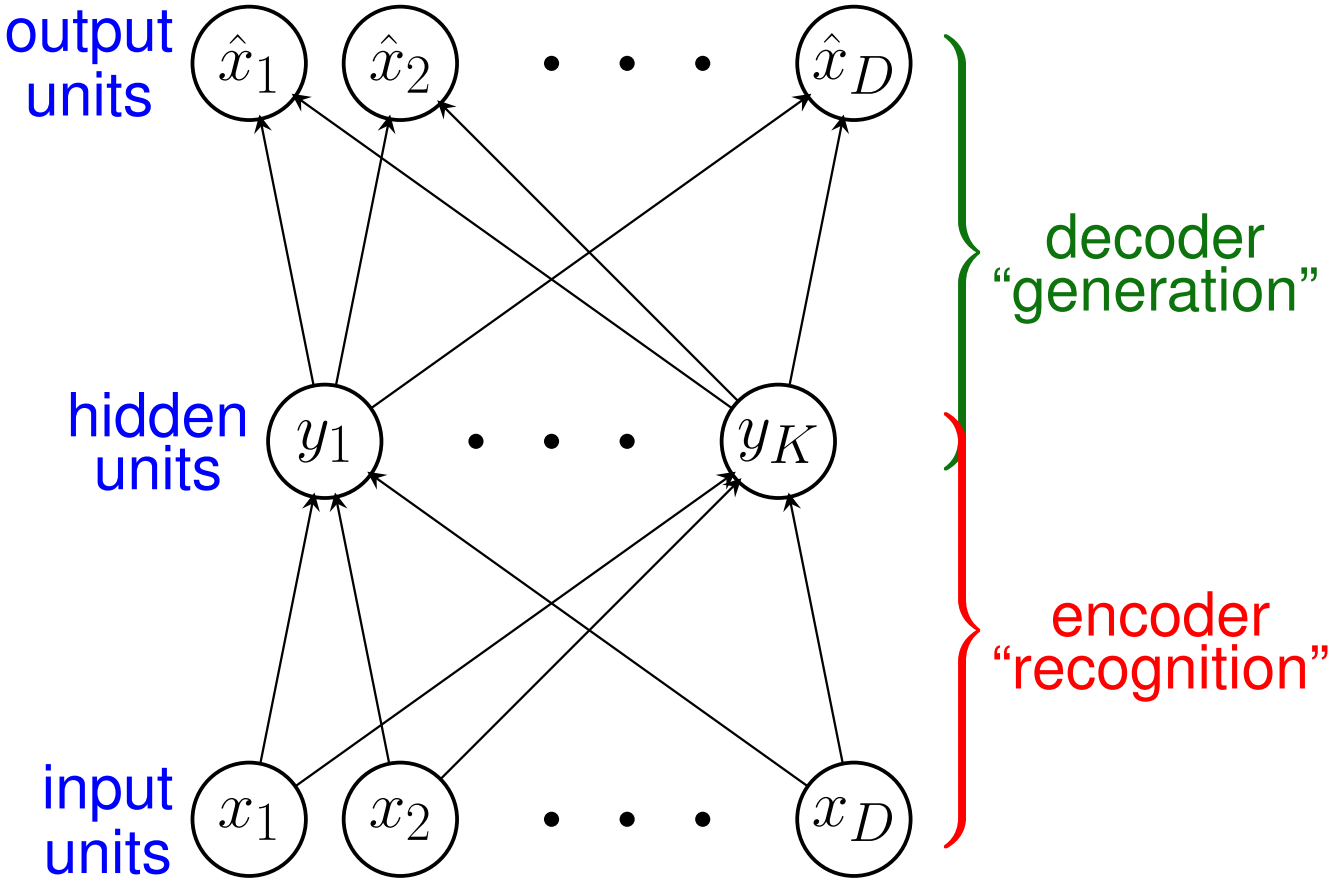
$$\mathcal{E} = \sum_i \|\mathbf{x}_i - P\mathbf{y}_i\|^2 = \sum_{ij} (X_{ij} - [PY]_{ij})^2$$

is minimised.

Our discussion of SVD approximation tells us that P must be:

- "proportional" to the first k left singular vectors of X
- that is, span the same space as the first k eigenvectors of S.

From Supervised Learning to PCA



A linear autoencoder neural network trained to minimise squared error learns to perform PCA (Baldi & Hornik, 1989).

Digression: other (linear) factor or component models

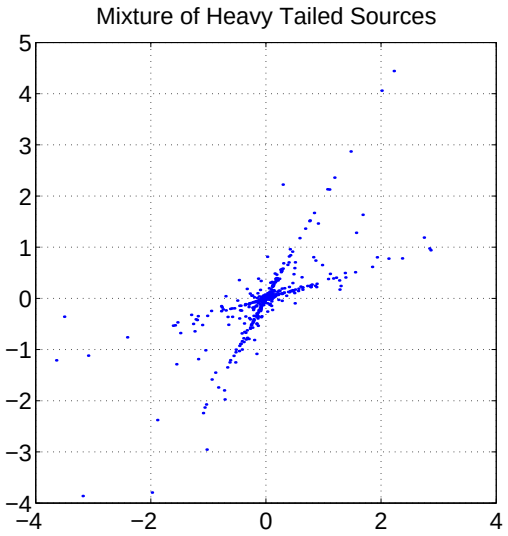
Data points are reconstructed by a linear combination of "factors":

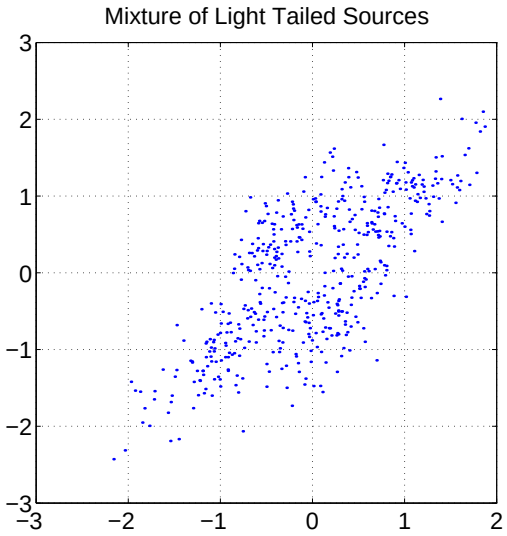
$$\hat{\mathbf{x}} = P(\mathbf{y}) = \sum_{\kappa=1}^{K} y_k \boldsymbol{\rho}_{(k)}$$

- PCA finds both a manifold and "principal components" that are orthogonal, and capture successive maxima of the variance.
- Any coordinates y_k and $y_{k'}$ are also uncorrelated over the data set.

It may sometimes be valuable to find a different coordinate system (or basis) in the manifold, or indeed in the original space.

Digression: Independent Components Analysis (ICA)





These distributions are generated by linearly combining (or mixing) two *non-Gaussian* sources.

- Not low-dimensional, but still well explained by linear factors.
- Factors not ordered in variance, and not orthogonal.
- How to find them?

Idea: coordinates along the basis vectors will be independent, sparse, or maximally non-Gaussian.

Square, Noiseless Causal ICA

- The special case of $K = D$, and zero observation noise is easy.

  $\mathbf{x} = \Lambda \mathbf{y}$ which implies $\mathbf{y} = W\mathbf{x}$ where $W = \Lambda^{-1}$

  W is an "unmixing" matrix.

- The likelihood can be obtained by transforming the density of $\mathbf{y}$ to that of $\mathbf{x}$. If $F : \mathbf{y} \mapsto \mathbf{x}$ is a differentiable bijection, and if $d\mathbf{y}$ is a small neighbourhood around $\mathbf{y}$, then

$$P_x(\mathbf{x})d\mathbf{x} = P_y(\mathbf{y})d\mathbf{y} = P_y(F^{-1}(\mathbf{x}))\left|\frac{d\mathbf{y}}{d\mathbf{x}}\right| d\mathbf{x} = P_y(F^{-1}(\mathbf{x}))\left|\nabla F^{-1}\right| d\mathbf{x}$$

- This gives (for parameter W):

$$P(\mathbf{x}|W) = |W| \prod_k P_y(\underbrace{[W\mathbf{x}]_k}_{y_k})$$

  where p_y is marginal probability distribution of factors.

- Often called "infomax" ICA (Comon, Bell & Sejnowski)

Finding the parameters in infomax ICA

Not a spectral algorithm.

- Log likelihood of data:

$$\log P(\mathbf{x}) = \log |W| + \sum_i \log P_y(W_i \mathbf{x})$$

- Learning by gradient ascent:

$$\Delta W \propto \nabla W = W^{-T} + g(\mathbf{y})\mathbf{x}^{\mathsf{T}} \qquad g(y) = \frac{\partial \log P_y(y)}{\partial y}$$

- Better approach: natural gradient

$$\Delta W \propto \nabla W(W^{\mathsf{T}}W) = W + g(\mathbf{y})\mathbf{y}^{\mathsf{T}}W$$

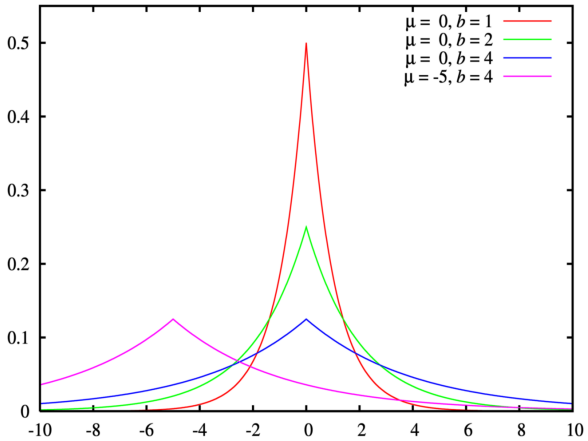
(see MacKay 1996).

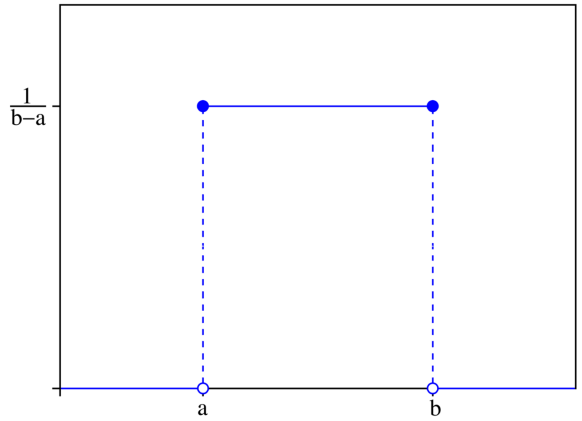
Kurtosis

(Excess) kurtosis measures how "peaky" or "heavy-tailed" a distribution:

$K = \frac{E((x-\mu)^4)}{E((x-\mu)^2)^2} - 3$, where $\mu = E(x)$ is the mean of x.

Gaussian distributions have zero kurtosis.

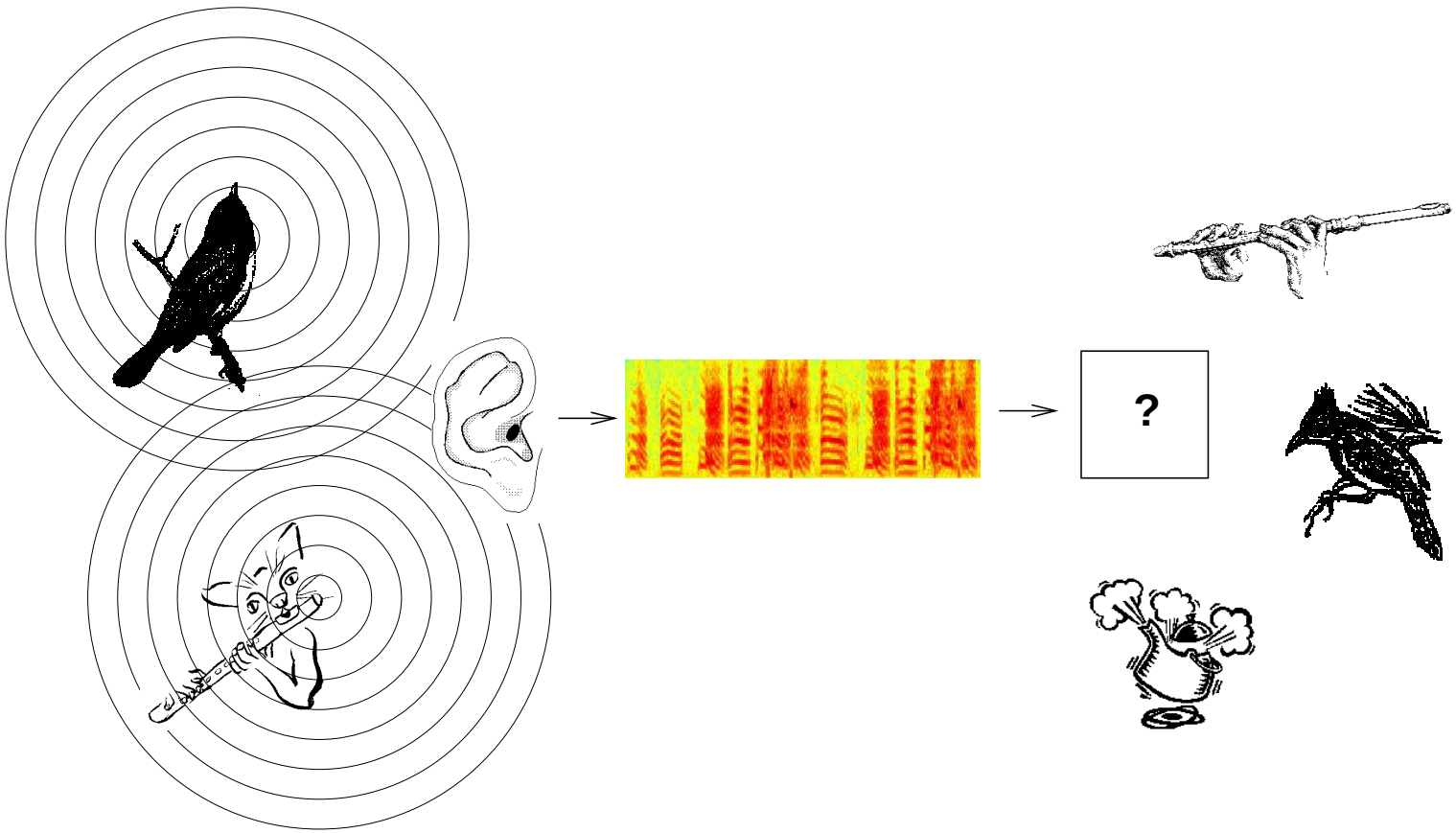




Heavy tailed: $K > 0$ (leptokurtic). Light tailed: $K < 0$ (platykurtic).

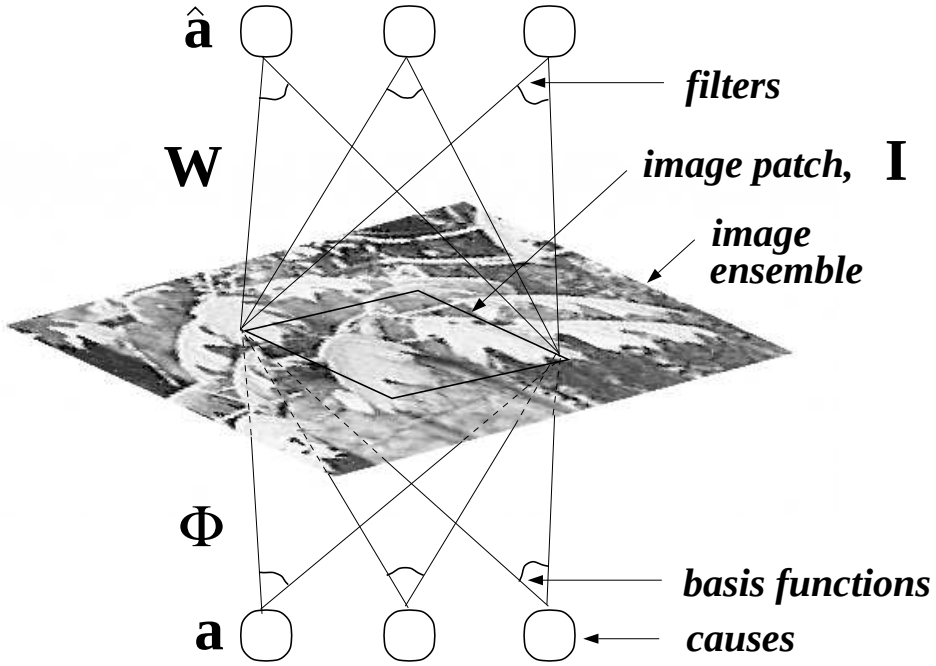
- Some ICA algorithms are essentially kurtosis pursuit approaches. Possibly fewer assumptions about generating distributions.
- Can find fewer ("undercomplete") or more ("overcomplete") factors than observed dimensions.
- Related to "sparse coding" or "sparse dictionary learning".

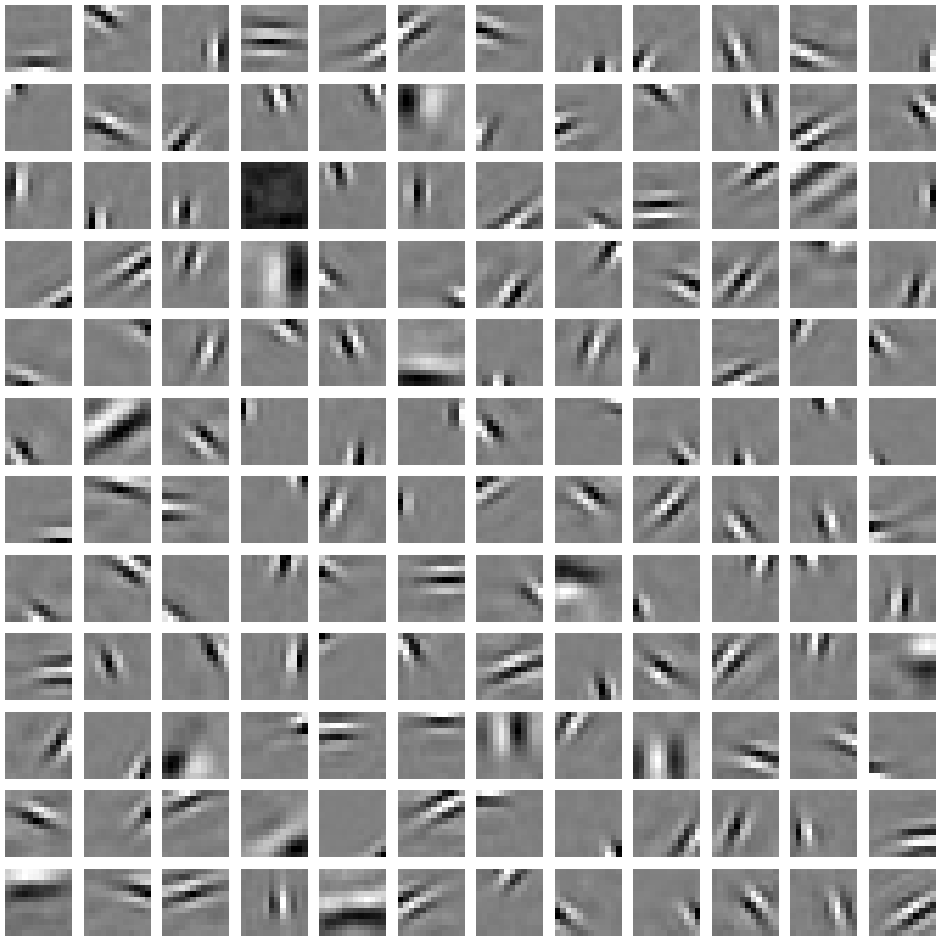
Blind Source Separation



- ICA solution to blind source separation assumes no dependence across time; still works fine much of the time.
- Many other algorithms: DCA, SOBI, JADE, . . .

ICA or sparse coding of natural images





Olshausen & Field (1996)

Bell & Sejnowski (1997)

Yet another view of PCA: matching inner products

We have viewed PCA as an approximation to the scatter matrix S or to X. We obtain similar results if we approximate the Gram matrix:

$$\text{minimise} \quad \mathcal{E} = \sum_{ij} (G_{ij} - \mathbf{y}_i \cdot \mathbf{y}_j)^2$$

for $\mathbf{y} \in \mathbb{R}^k$.

That is, look for a k-dimensional embedding in which dot products (which depend on lengths, and angles) are preserved as well as possible.

We will see that this is also equivalent to preserving distances between points.

Yet another view of PCA: matching inner products

Consider the eigendecomposition of G:

$$G = U\Lambda U^\mathsf{T} \quad \text{arranged so} \quad \lambda_1 \geq \cdots \geq \lambda_m \geq 0$$

The best rank-k approximation $G \approx Y^\mathsf{T} Y$ is given by:

$$\begin{aligned} Y^\mathsf{T} &= [U]_{1:m,1:k}[\Lambda^{1/2}]_{1:k,1:k}; \\ &= [U\Lambda^{1/2}]_{1:m,1:k} \end{aligned}$$

$$Y = [\Lambda^{1/2}U^\mathsf{T}]_{1:k,1:m}$$

$$\begin{bmatrix} \text{———} & \sqrt{\lambda_1}\,\mathbf{u}_1^\mathsf{T} & \text{———} \\ \text{———} & \sqrt{\lambda_2}\,\mathbf{u}_2^\mathsf{T} & \text{———} \\ & \vdots & \\ \text{———} & \sqrt{\lambda_k}\,\mathbf{u}_k^\mathsf{T} & \text{———} \\ & \vdots & \\ \text{———} & \sqrt{\lambda_m}\,\mathbf{u}_m^\mathsf{T} & \text{———} \end{bmatrix}$$

Yet another view of PCA: matching inner products

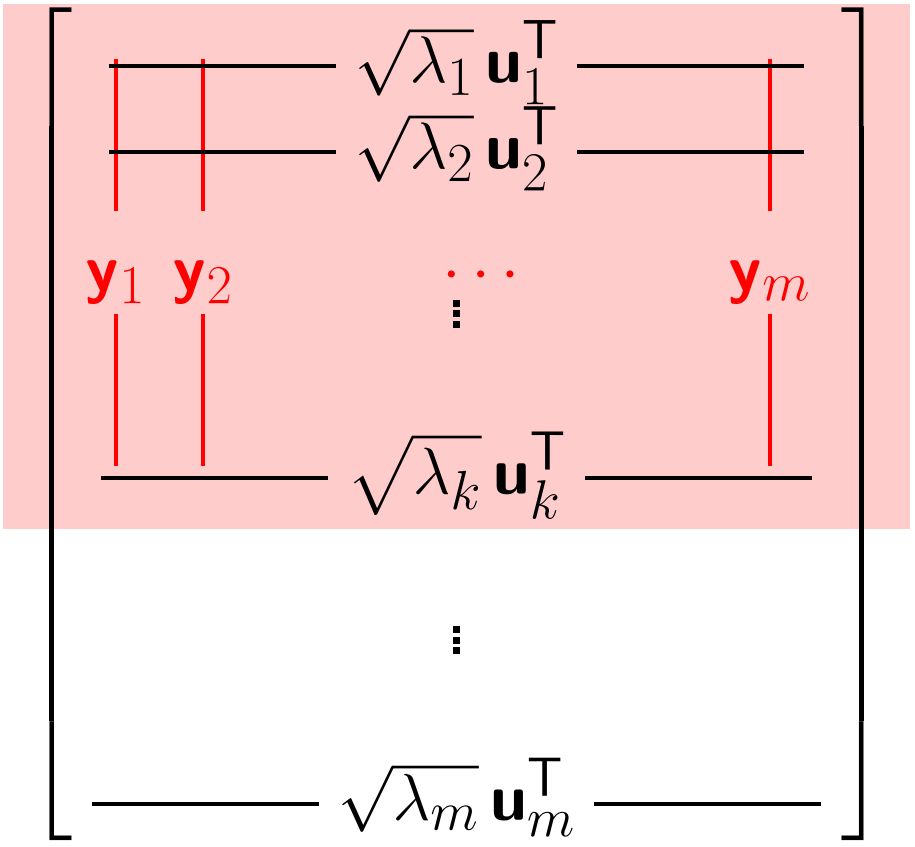
Consider the eigendecomposition of G:

$$G = U\Lambda U^\mathsf{T} \quad \text{arranged so} \quad \lambda_1 \geq \cdots \geq \lambda_m \geq 0$$

The best rank-k approximation $G \approx Y^\mathsf{T} Y$ is given by:

$$\begin{aligned} Y^\mathsf{T} &= [U]_{1:m,1:k}[\Lambda^{1/2}]_{1:k,1:k}; \\ &= [U\Lambda^{1/2}]_{1:m,1:k} \end{aligned}$$

$$Y = [\Lambda^{1/2}U^\mathsf{T}]_{1:k,1:m}$$



Multidimensional Scaling

Suppose all we were given were distances or symmetric "dissimilarities" Δ_{ij}.

$$\Delta = \begin{bmatrix} 0 & \Delta_{12} & \Delta_{13} & \Delta_{14} \\ \Delta_{12} & 0 & \Delta_{23} & \Delta_{24} \\ \Delta_{13} & \Delta_{23} & 0 & \Delta_{34} \\ \Delta_{14} & \Delta_{24} & \Delta_{34} & 0 \end{bmatrix}$$

Goal: Find vectors $\mathbf{y}_i$ such that $\|\mathbf{y}_i - \mathbf{y}_j\| \approx \Delta_{ij}$.

This is called **Multidimensional Scaling (MDS)**.

Metric MDS

Assume the dissimilarities represent Euclidean distances between points in some high-D space.

$$\Delta_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\| \text{ with } \sum_i \mathbf{x}_i = \mathbf{0}.$$

We have:

$$\begin{aligned}
\Delta_{ij}^2 &= \|\mathbf{x}_i\|^2 + \|\mathbf{x}_j\|^2 - 2\mathbf{x}_i \cdot \mathbf{x}_j \\
\sum_k \Delta_{ik}^2 &= m\|\mathbf{x}_i\|^2 + \sum_k \|\mathbf{x}_k\|^2 - \mathbf{0} \\
\sum_k \Delta_{kj}^2 &= \sum_k \|\mathbf{x}_k\|^2 + m\|\mathbf{x}_j\|^2 - \mathbf{0} \\
\sum_{kl} \Delta_{kl}^2 &= 2m \sum_k \|\mathbf{x}_k\|^2
\end{aligned}$$

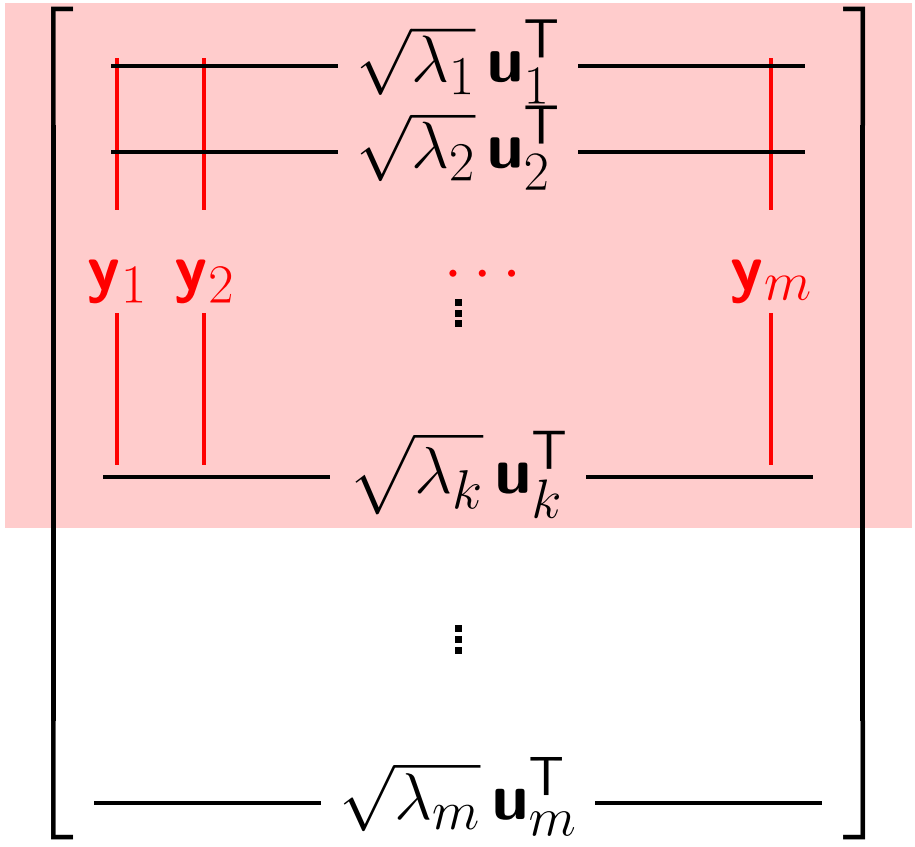
$$\Rightarrow G_{ij} = \mathbf{x}_i \cdot \mathbf{x}_j = \frac{1}{2}\left(\frac{1}{m}\sum_k (\Delta_{ik}^2 + \Delta_{kj}^2) - \frac{1}{m^2}\sum_{kl} \Delta_{kl}^2 - \Delta_{ij}^2\right)$$

Metric MDS and eigenvalues

We will actually minimize the error in the dot products:

$$\mathcal{E} = \sum_{ij} (G_{ij} - \mathbf{y}_i \cdot \mathbf{y}_j)^2$$

As in PCA, this is given by the top slice of the eigenvector matrix.



Interpreting MDS

$$G = \frac{1}{2}\left(\frac{1}{m}(\Delta^2 1 + 1\Delta^2) - \Delta^2 - \frac{1}{m^2}1^\mathsf{T}\Delta^2 1\right)$$

$$G = U\Lambda U^\mathsf{T}; \qquad Y = [\Lambda^{1/2}U^\mathsf{T}]_{1:k,1:m}$$

(1 is a matrix of ones.)

- **Eigenvectors.** Ordered, scaled and truncated to yield low-dimensional embedded points $\mathbf{y}_i$.
- **Eigenvalues.** Measure how much each dimension contributes to dot products.
- **Estimated dimensionality.** Number of significant (nonnegative – negative possible if Δ_{ij} are not metric) eigenvalues.

MDS for Scotch Whisky

From: Multidimensional Scaling, 2nd Ed, TF Cox, MAA Cox

Features:

Table 6.2 *Nose characteristics of nineteen whiskies.*

	Characteristics											
Whisky	1	2	3	4	5	6	7	8	9	10	11	12
Glenburgie	1	1			1							
Strathisla	1											
Balblair						1	1					
Clynelish	1							1				
Royal Brackla		1				1	1	1				
Teaninich	1					1	1					
Glen Garioch						1	1					
Glenturret				1		1			1			
Oban						1		1				
Bladnoch	1		1						1			
Littlemill	1		1	1								
Ardbeg						1		1				
Bowmore		1					1	1				
Lagavulin							1	1			1	
Laphroaig							1	1		1		
Highland Park				1			1					
Isle of Jura								1				1
Tobermory	1		1			1						
Bushmills						1	1				1	

Key: (1)-fruit, (2)-floral, (3)-light, (4)-delicate, (5)-fragrant, (6)-sweetness, (7)-smoke, (8)-peaty, (9)-aromatic, (10)-medicinal, (11)-sherry, (12)-tart

Table 6.3 *Taste characteristics of nineteen whiskies.*

	Characteristics													
Whisky	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Glenburgie	1			1										
Strathisla		1										1	1	
Balblair												1	1	
Clynelish		1												1
Royal Brackla							1	1				1	1	
Teaninich					1								1	
Glen Garioch										1				
Glenturret														1
Oban												1		
Bladnoch									1			1		
Littlemill				1										
Ardbeg						1				1				
Bowmore					1					1				
Lagavulin						1		1	1					
Laphroaig										1		1		
Highland Park					1		1				1			
Isle of Jura			1										1	
Tobermory				1							1	1		
Bushmills									1					

Key: (1)-delicate, (2)-fruit, (3)-floral, (4)-light, (5)-medium bodied, (6)-full bodied, (7)-dry, (8)-sherry, (9)-smooth, (10)-peaty, (11)-smoke, (12)-sweetness, (13)-lingering, (14)-full

MDS for Scotch Whisky

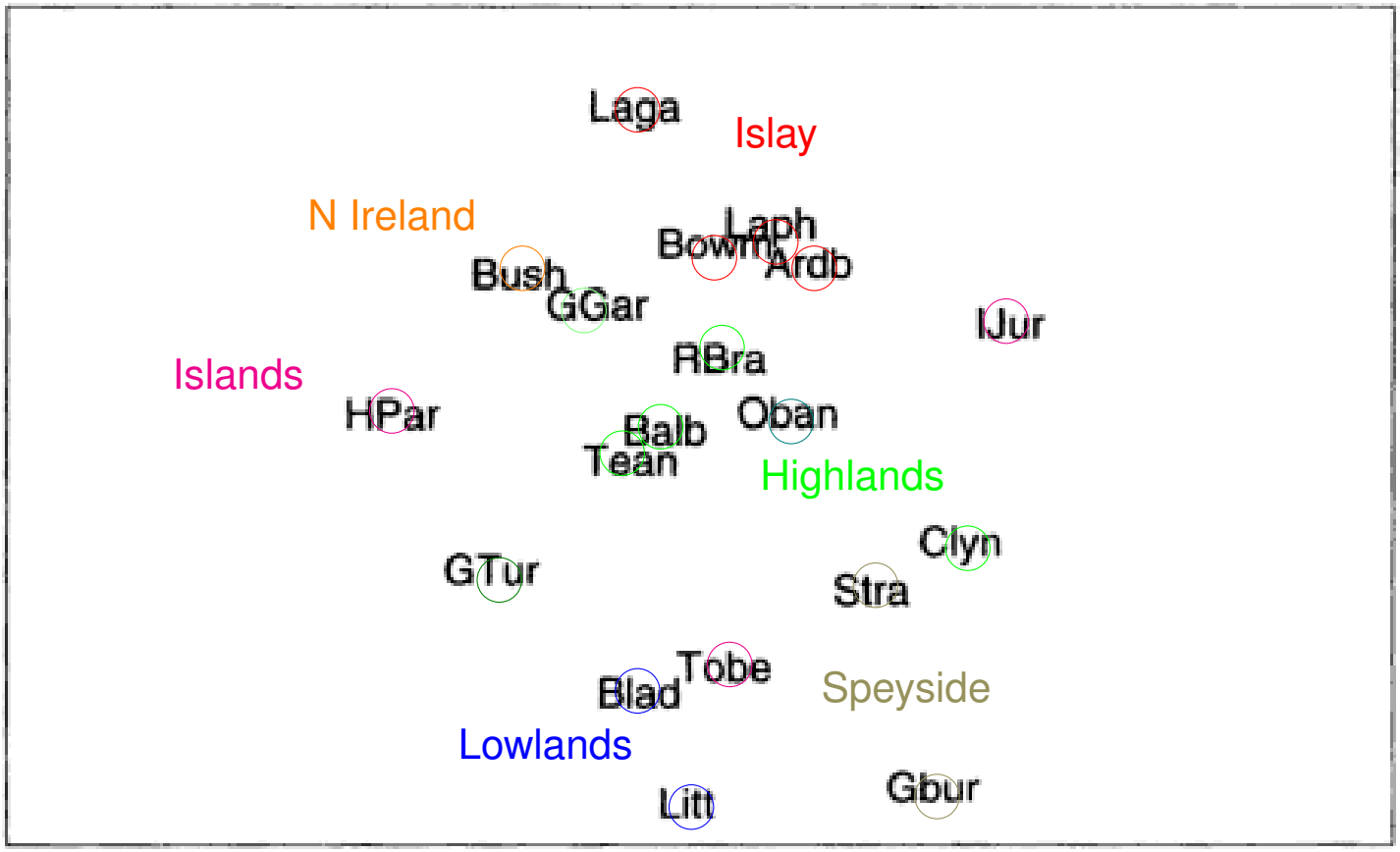


Figure 6.2 *Nonmetric MDS of the whisky data*

MDS and PCA

Dual matrices:

$S = \frac{1}{m} X X^\mathsf{T}$	scatter matrix	$(n \times n)$
$G = X^\mathsf{T} X$	Gram matrix	$(m \times m)$

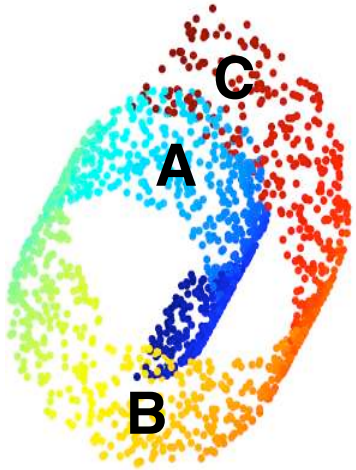
- **Same eigenvalues** up to a constant factor.
- **Equivalent on metric data**, but MDS can run on non-metric dissimilarities.
- **Computational cost** is different.
  - PCA: $O((m+k)n^2)$
  - MDS: $O((n+k)m^2)$

Non-linear extensions to PCA and MDS

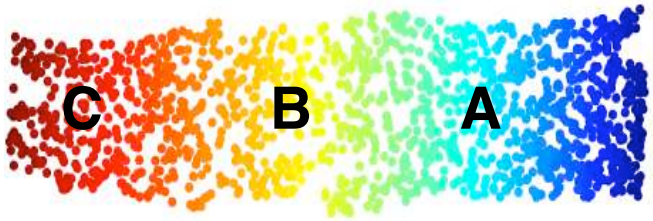
- Non-linear autoencoder (e.g. multilayer neural network)
- Gaussian Process Latent Variable Models (beyond our scope today)
- Kernel methods (replace inner products by kernel evaluations)
- Distance rescaling $\Delta_{ij} \rightarrow g(\Delta_{ij})$ (even if this violates metric rules).

But

Rank ordering of Euclidean distances is NOT preserved in "manifold learning".



$d(A,C) < d(A,B)$

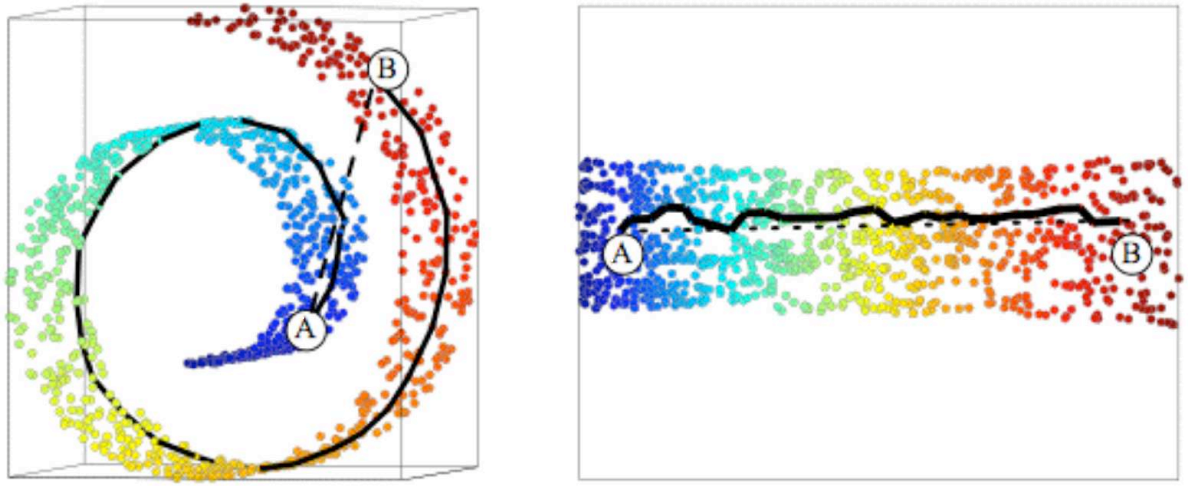


$d(A,C) > d(A,B)$

Nonlinear (newer) methods

Isomap

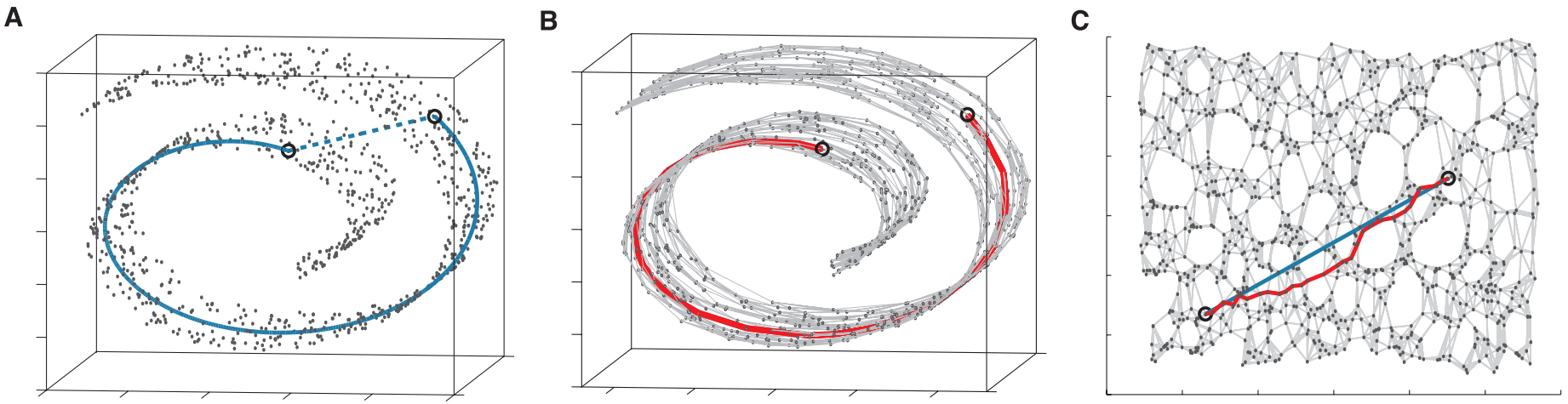
Idea: try to trace distance along the manifold. Use geodesic instead of (transformed) Euclidean distances in MDS.



- preserves local structure
- estimates “global” structure
- preserves information (MDS)

Stages of Isomap

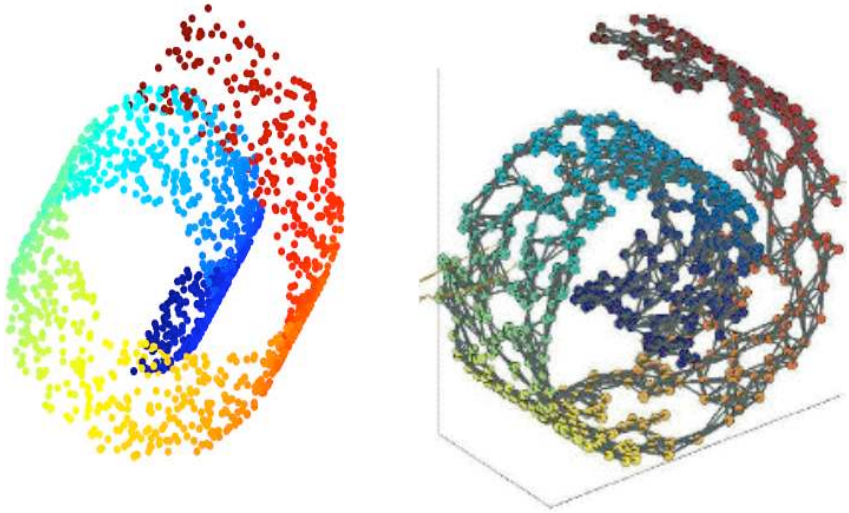
1. Identify neighbourhoods around each point (local points, assumed to be local on the manifold). Euclidean distances are preserved within a neighbourhood.
2. For points outside the neighbourhood, estimate distances by hopping between points within neighbourhoods.
3. Embed using MDS.



Step 1: Neighbourhood graph

First we construct a graph linking each point to its neighbours.

- vertices represent input points
- undirected edges connect neighbours (weight = Euclidean distance)



Forms a discretised approximation to the submanifold, assuming:

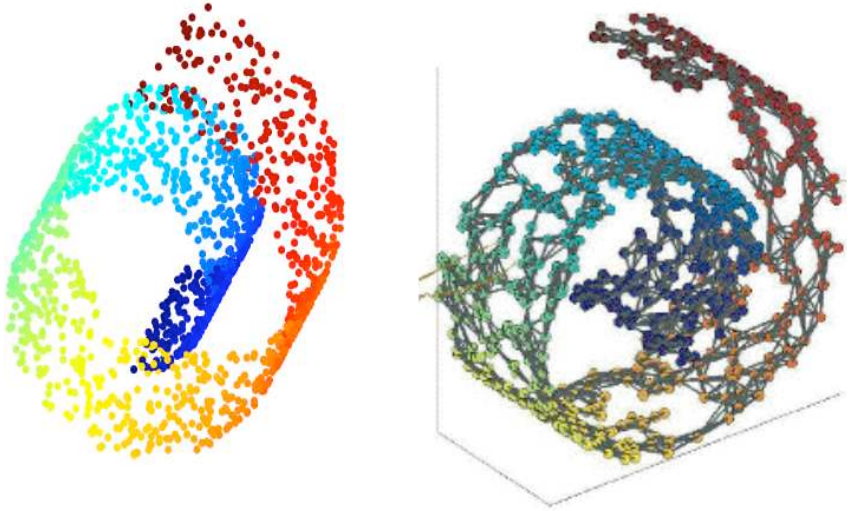
- Graph has single connected component.
- Graph neighborhoods reflect manifold neighborhoods. No “short cuts”.

Defining the neighbourhood is critical: k-nearest neighbours, inputs within a ball of radius r, prior knowledge.

Step 2: Geodesics

Estimate distances by shortest path in graph.

$$\Delta_{ij} = \min_{\text{path}(\mathbf{x}_i,\mathbf{x}_j)} \left\{ \sum_{e_i \in \text{path}(\mathbf{x}_i,\mathbf{x}_j)} \delta_i \right\}$$

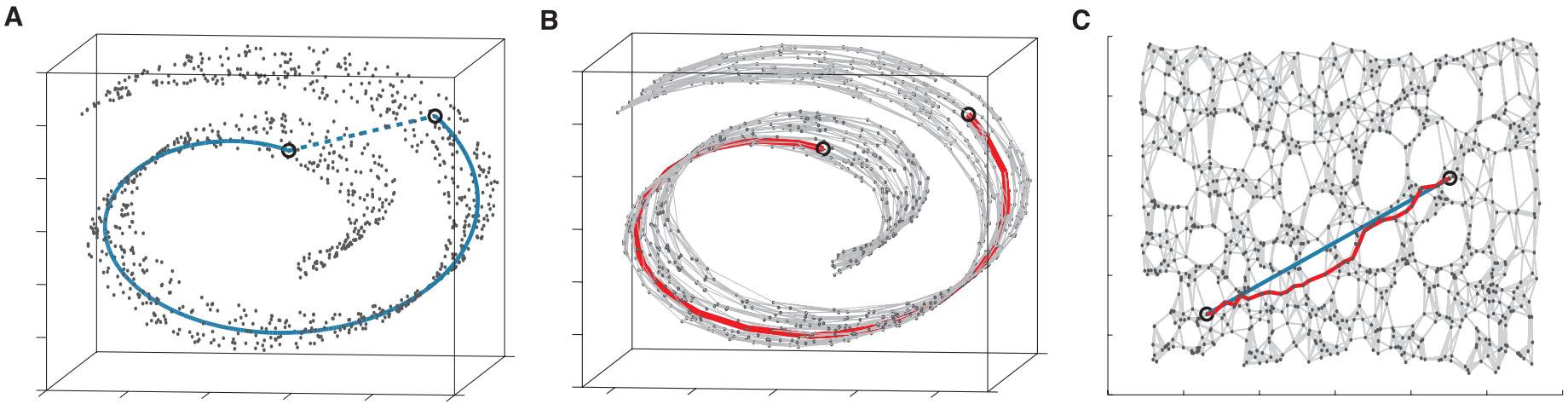


- Standard graph problem. Solved by Dijkstra's algorithm (and others).
- Better estimates for denser sampling.
- Short cuts very dangerous ("average" path distance?) .

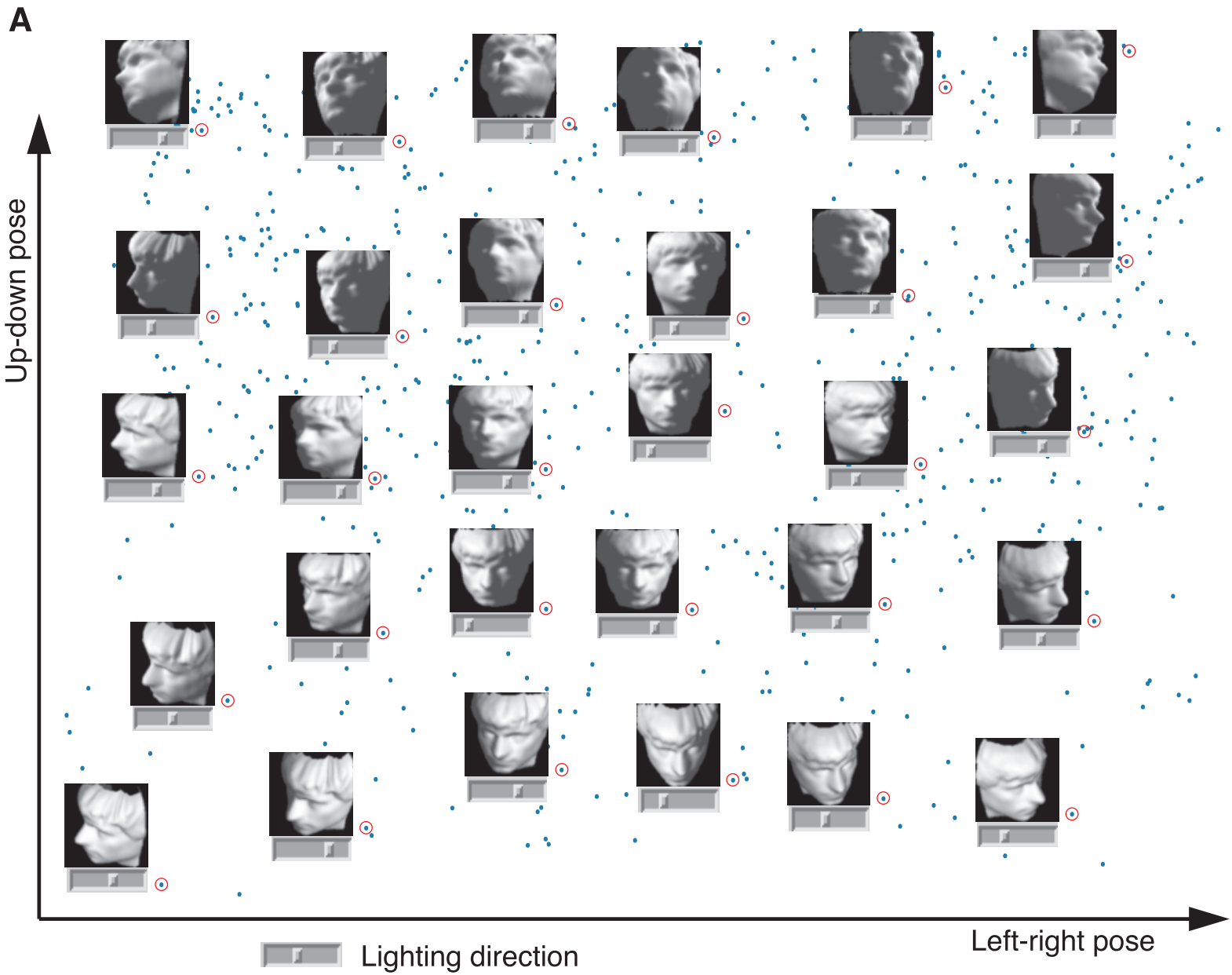
Step 3: Embed

Embed using metric MDS (path distances obey the triangle inequality)

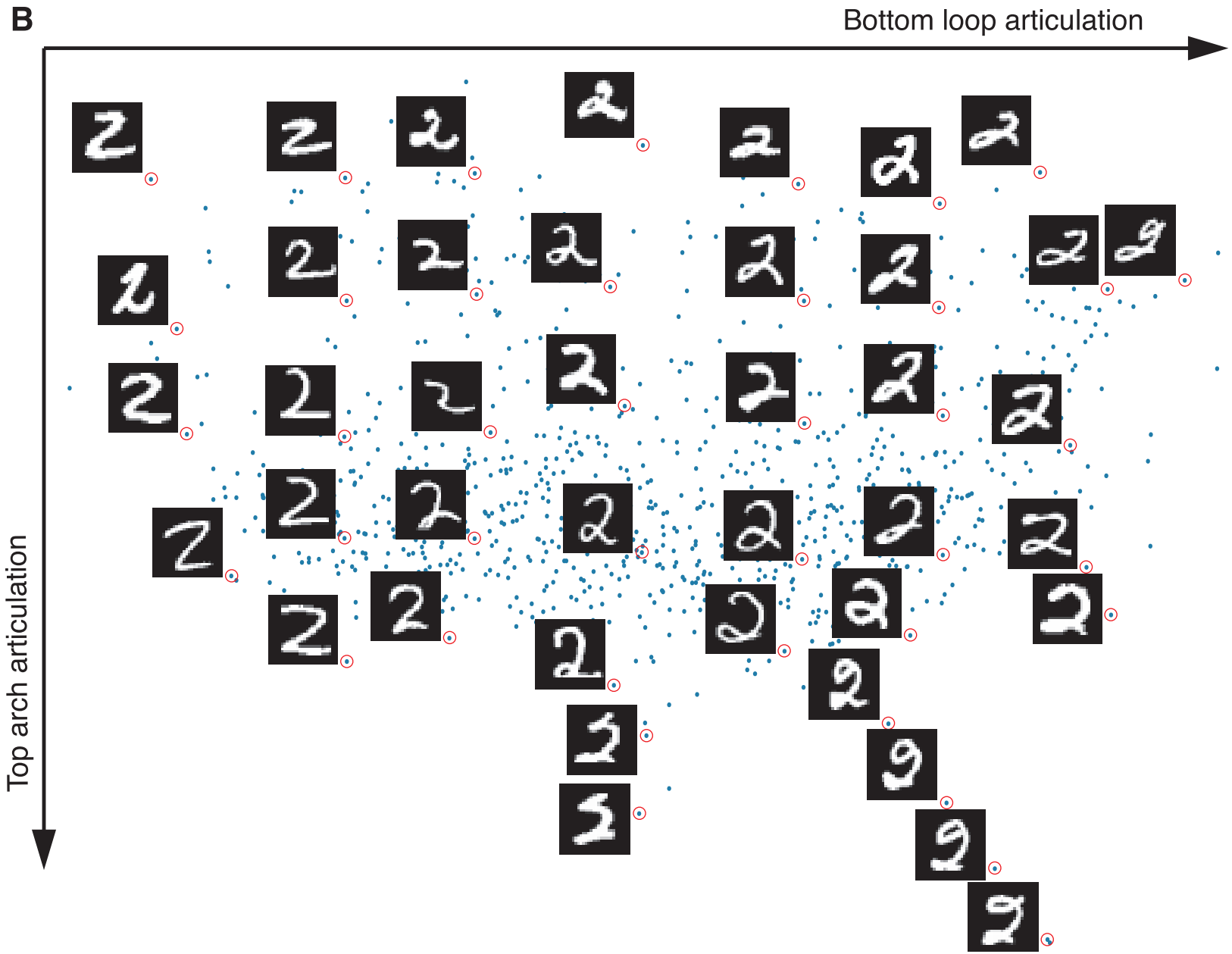
- Eigenvectors of Gram matrix yield low-dimensional embedding.
- Number of significant eigenvalues estimates dimensionality.



Isomap example 1



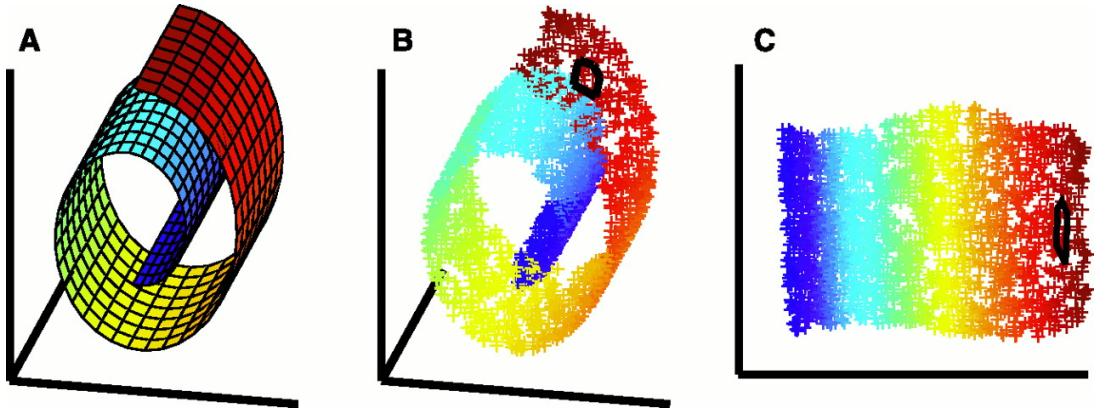
Isomap example 2



Locally Linear Embedding (LLE)

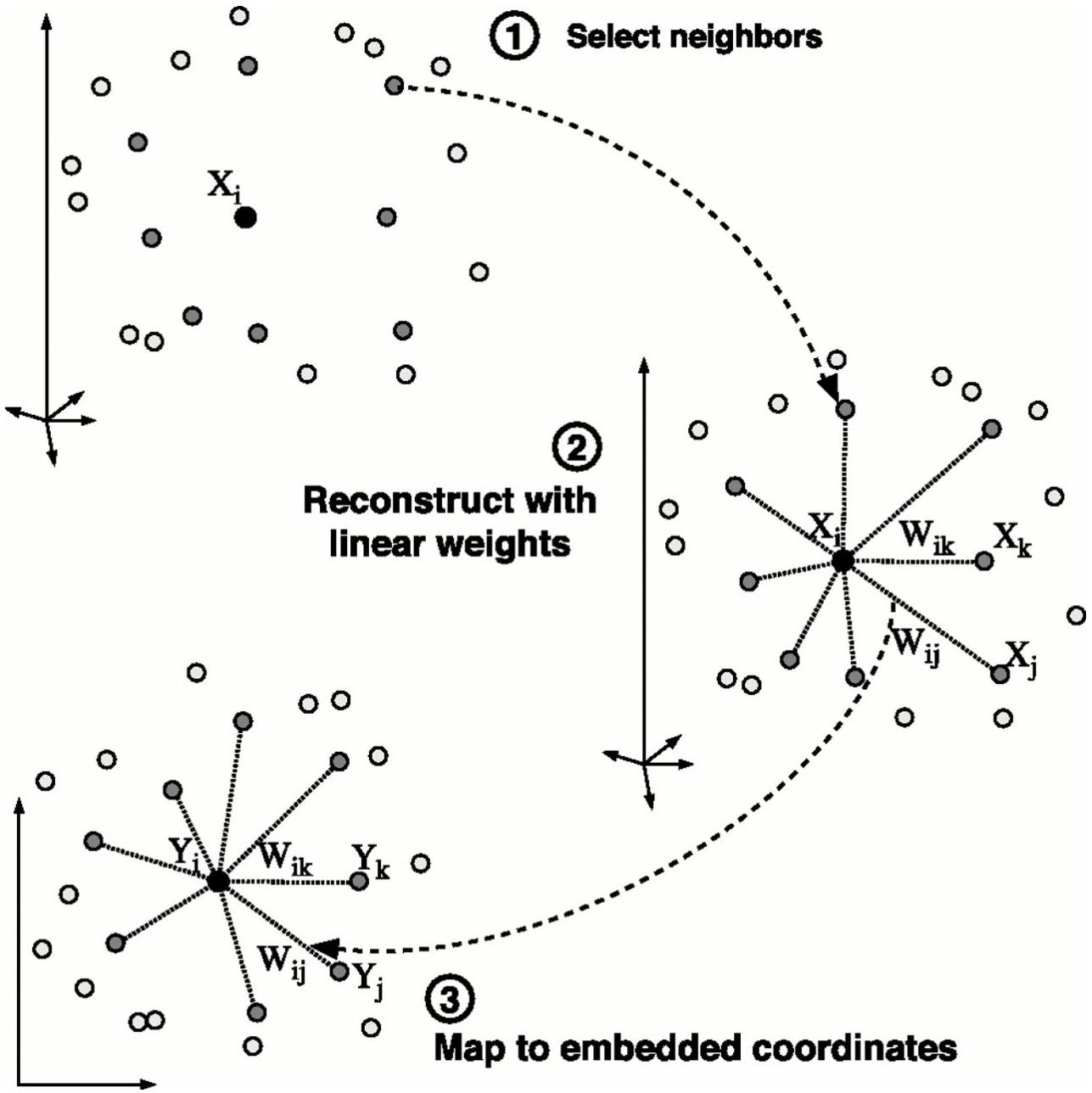
MDS and isomap preserve local and global (estimated, for isomap) distances. PCA preserves local and global structure.

Idea: estimate local (linear) structure of manifold. Preserve this as well as possible.



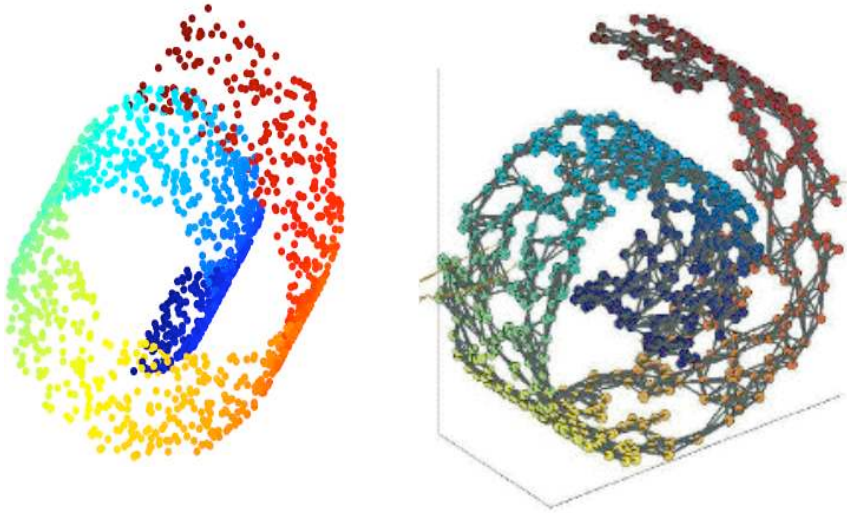
- preserves local structure (not just distance)
- not explicitly global
- preserves only local information

Stages of LLE



Step 1: Neighbourhoods

Just as in isomap, we first define neighbouring points for each input. Equivalent to the isomap graph, but we won't need the graph structure.



Forms a discretised approximation to the submanifold, assuming:

- Graph has single connected component — although will "work" if not.
- Neighborhoods reflect manifold neighborhoods. No "short cuts".

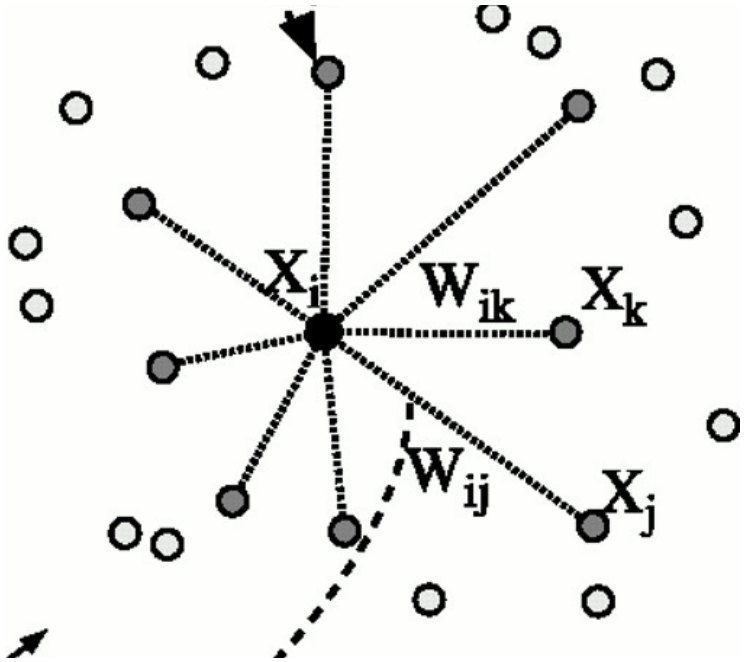
Defining the neighbourhood is critical: k-nearest neighbours, inputs within a ball of radius r, prior knowledge.

Step 2: Local weights

Estimate local weights to minimize error

$$\Phi(W) = \sum_i \left\| \mathbf{x}_i - \sum_{j \in \mathrm{Ne}(i)} W_{ij} \mathbf{x}_j \right\|^2$$

$$\sum_{j \in \mathrm{Ne}(i)} W_{ij} = 1$$

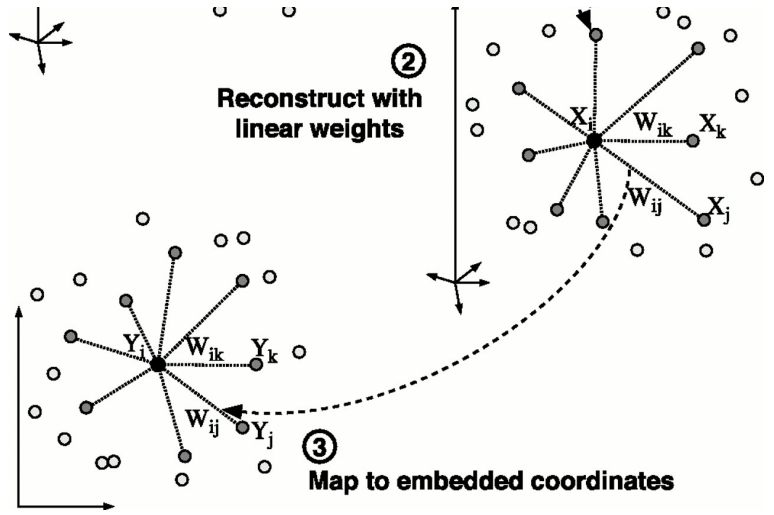


- Linear regression – under- or over-constrained depending on $|\mathrm{Ne}(i)|$.
- Local structure – optimal weights are invariant to rotation, translation and scaling.
- Short cuts less dangerous (one in many).

Step 3: Embed

Minimise reconstruction errors in **y**-space under the same weights:

$$\psi(Y) = \sum_i \left\| \mathbf{y}_i - \sum_{j \in \mathsf{Ne}(i)} W_{ij} \mathbf{y}_j \right\|^2$$



subject to:

$$\sum_i \mathbf{y}_i = \mathbf{0}; \qquad \sum_i \mathbf{y}_i \mathbf{y}_i^\mathsf{T} = mI$$

We can re-write the cost function in quadratic form:

$$\psi(Y) = \sum_{ij} \Psi_{ij} [Y^\mathsf{T} Y]_{ij} \text{ with } \Psi = (I - W)^\mathsf{T} (I - W)$$

Minimise by setting Y to equal the bottom $2 \ldots k+1$ eigenvectors of Ψ.
(Bottom eigenvector always $\mathbf{1}$ – discard due to centering constraint)

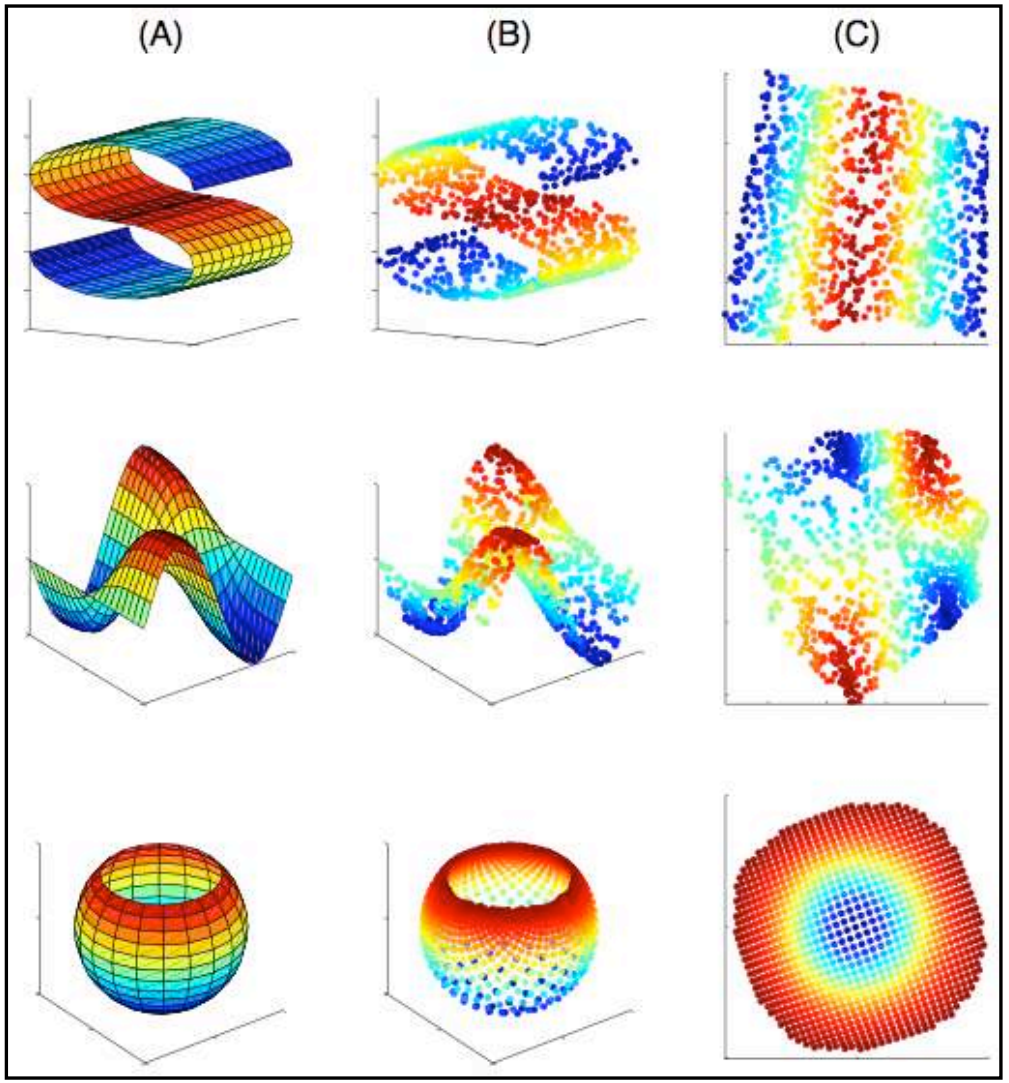
LLE example 1

Surfaces

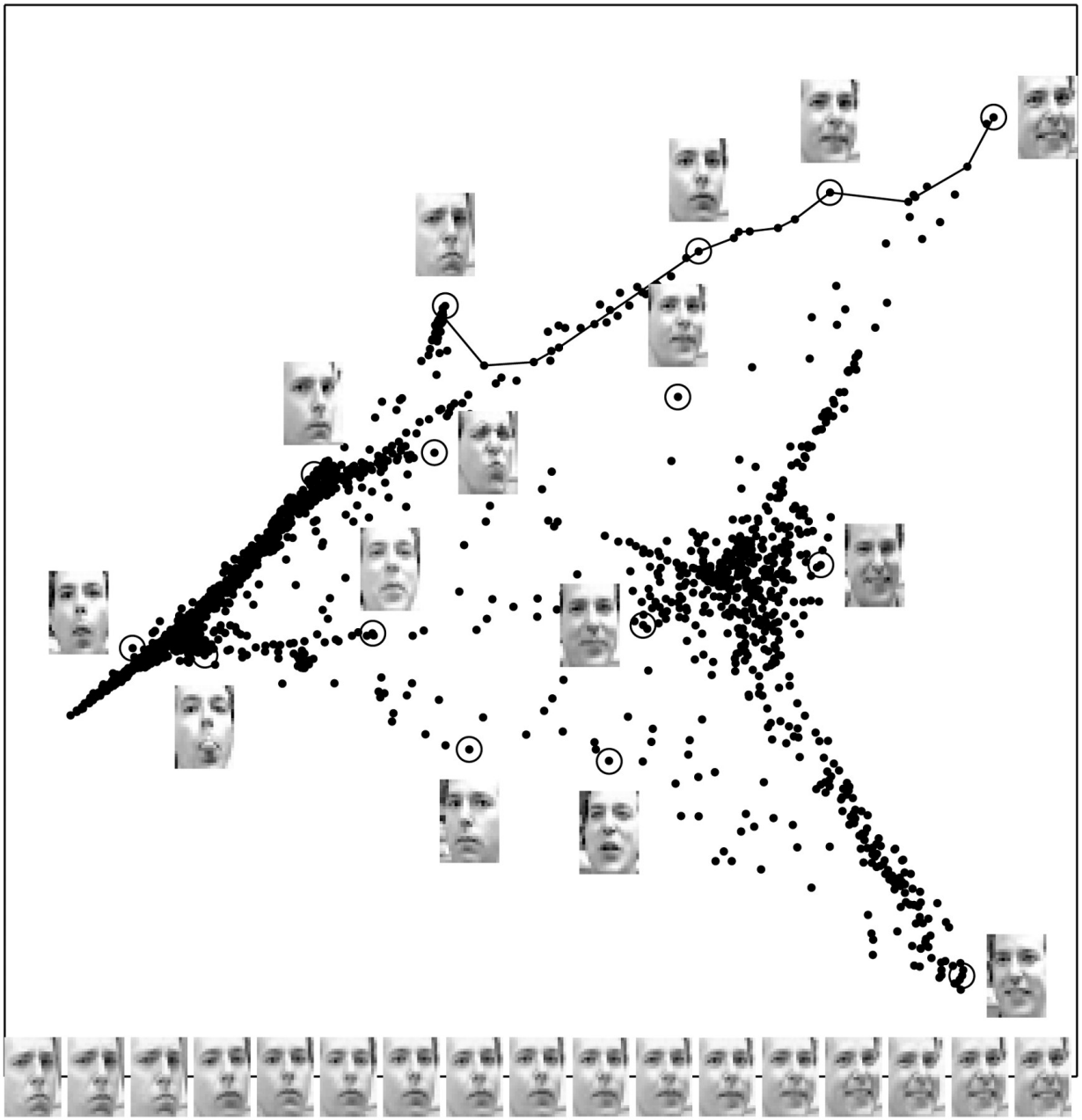
N=1000
inputs

k=8
nearest
neighbors

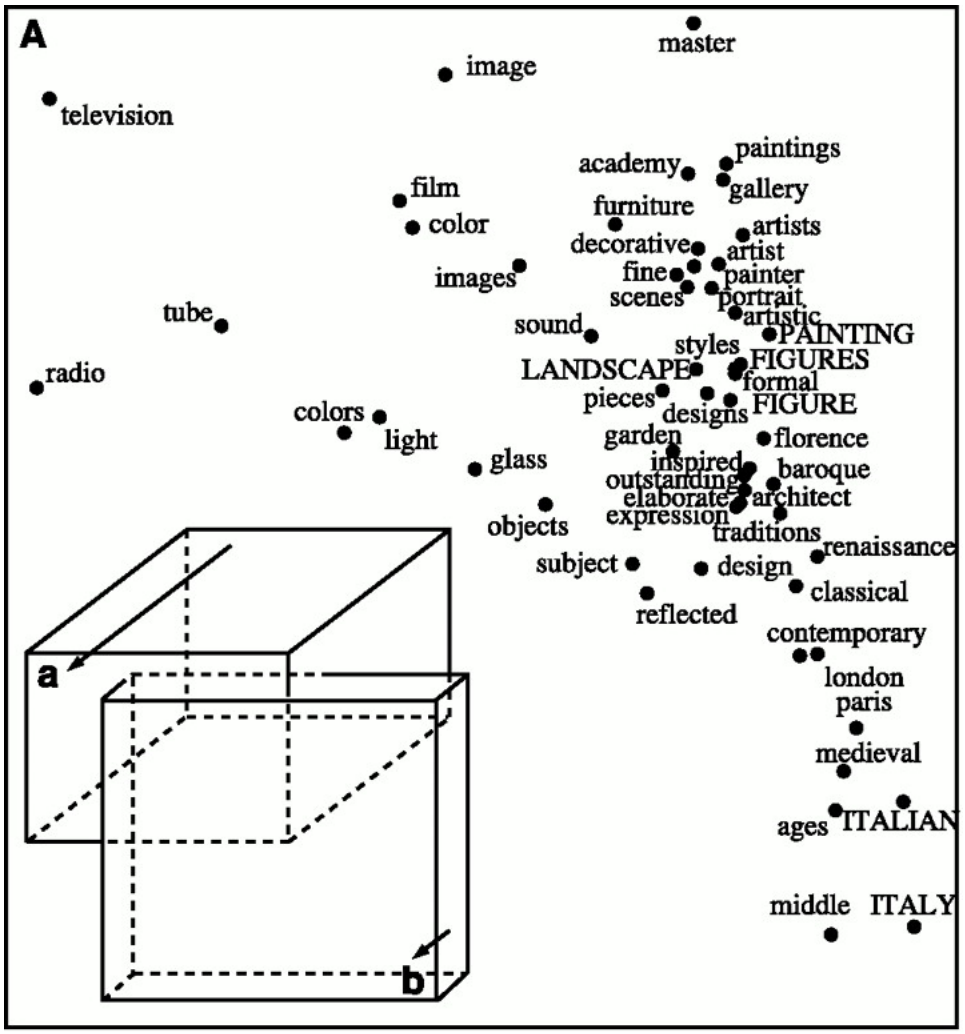
D=3
d=2
dimensions

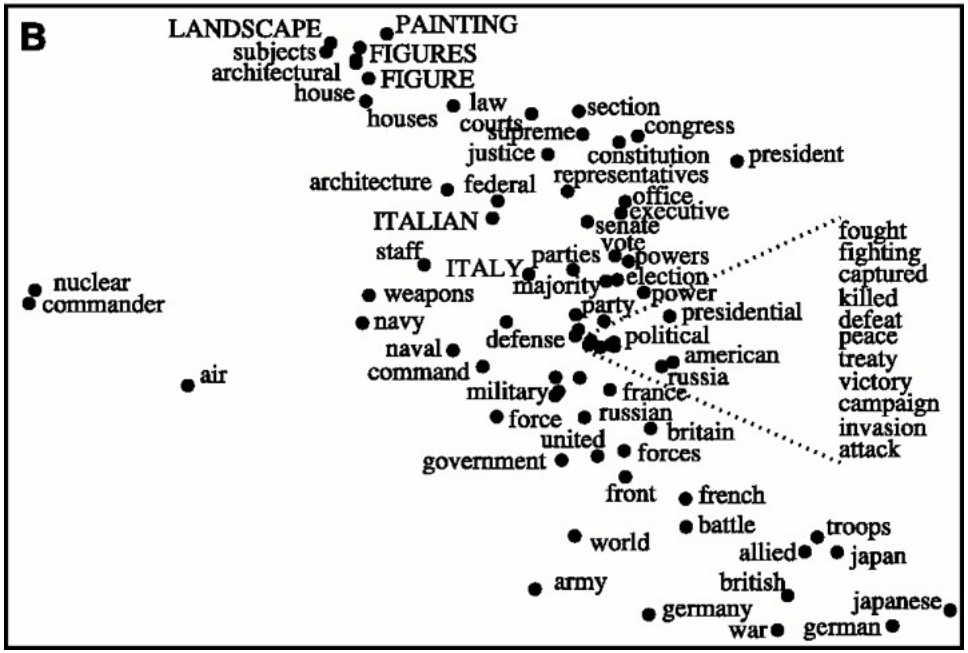


LLE example 2



LLE example 3





LLE and Isomap

Many similarities

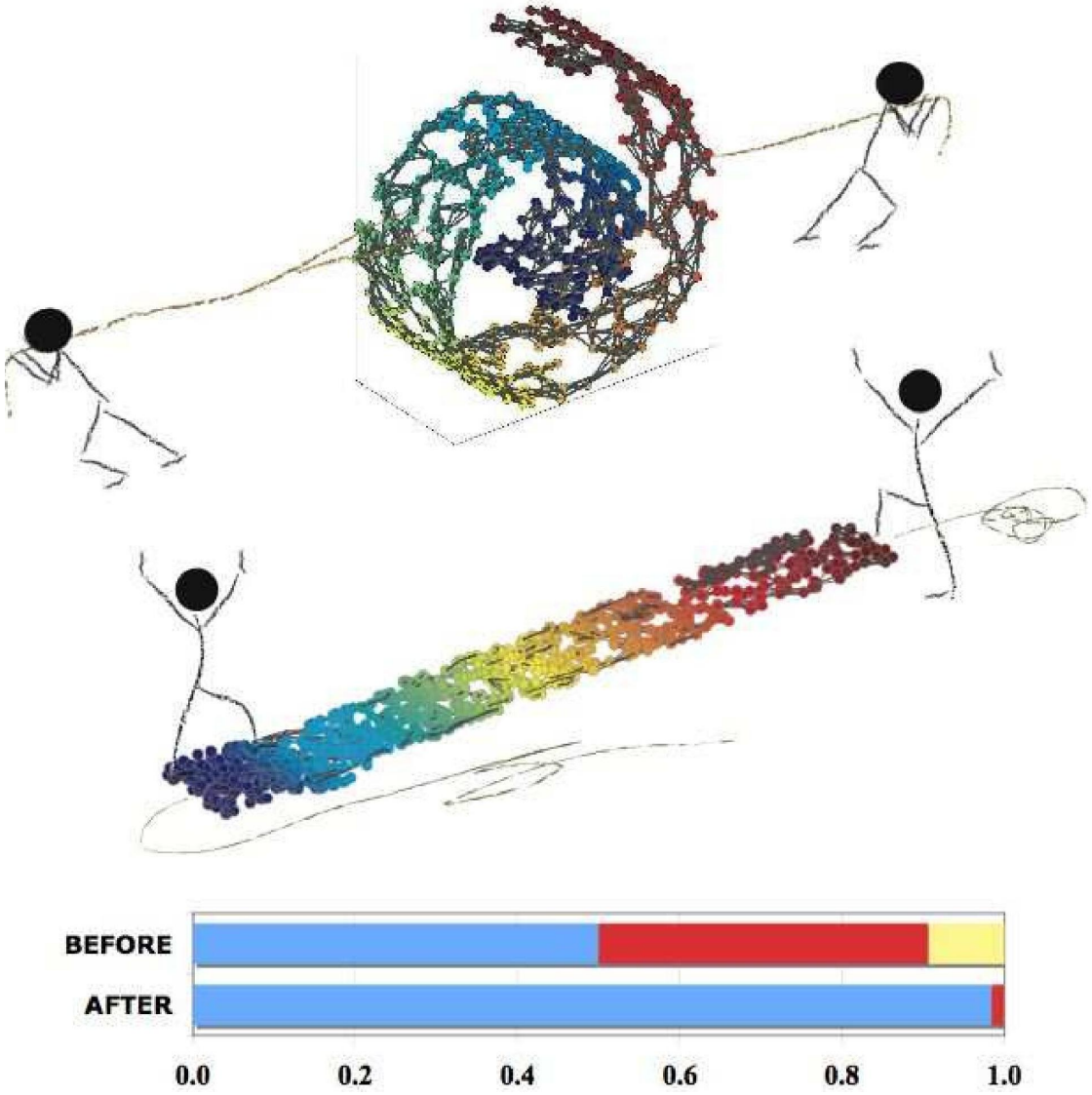
- Graph-based, spectral methods
- No local optima

Essential differences

- LLE does not estimate dimensionality
- Isomap can be shown to be consistent; no theoretical guarantees for LLE.
- LLE diagonalises a sparse matrix – more efficient than isomap.
- Local weights vs. local & global distances.

Maximum Variance Unfolding

Unfold neighbourhood graph preserving local structure.



Maximum Variance Unfolding

Unfold neighbourhood graph preserving local structure.

1. Build the neighbourhood graph.
2. Find $\{\mathbf{y}_i\} \subset \mathbb{R}^n$ (points in **high-D** space) with maximum variance, preserving local distances. Let $K_{ij} = \mathbf{y}_i^\mathsf{T}\mathbf{y}_j$. Then:

   Maximise Tr $[K]$ subject to:

   $$\sum_{ij} K_{ij} = 0 \qquad \text{(centered)}$$
   $$K \succeq 0 \qquad \text{(positive definite)}$$
   $$\underbrace{K_{ii} - 2K_{ij} + K_{jj}}_{\|\mathbf{y}_i - \mathbf{y}_j\|^2} = \|\mathbf{x}_i - \mathbf{x}_j\|^2 \text{ for } j \in \text{Ne}(i) \qquad \text{(locally metric)}$$

   This is a **semi-definite program**: convex optimisation with unique solution.
3. Embed $\mathbf{y}_i$ in $\mathbb{R}^k$ using linear methods (PCA/MDS).

Stochastic Neighbour Embedding

Softer "probabilistic" notions of neighbourhood and consistency.

High-D "transition" probabilities:

$$p_{j|i} = \frac{e^{-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}}{\sum_{k\neq i} e^{-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_k\|^2/\sigma^2}} \quad \text{for } j \neq i, \qquad p_{i|i} = 0$$

Find $\{\mathbf{y}_i\} \subset \mathbb{R}^k$ to:

$$\text{minimise } \sum_{ij} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \qquad \text{with } q_{j|i} = \frac{e^{-\frac{1}{2}\|\mathbf{y}_i - \mathbf{y}_j\|^2}}{\sum_{k\neq i} e^{-\frac{1}{2}\|\mathbf{y}_i - \mathbf{y}_k\|^2}}.$$

Nonconvex optimisation is initialisation dependent.

Scale σ plays a similar role to neighbourhood definition:

- Fixed σ: resembles a fixed-radius ball.
- Choose σ_i to maintain consistent entropy in $p_{j|i}$ of $\log_2 k$: similar to k-nearest neighbours.

SNE variants

- Symmetrise probabilities ($p_{ij} = p_{ji}$)

$$p_{ij} = \frac{e^{-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}}{\sum_{k \neq l} e^{-\frac{1}{2}\|\mathbf{x}_l - \mathbf{x}_k\|^2/\sigma^2}} \quad \text{for } j \neq i$$

  Define q_{ij} analagously, optimise joint KL.

- Heavy-tailed embedding distributions allow embedding to lower dimensions than true manifold:

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}$$

  Student-t distribution defines "t-SNE".

Focus is on visualisation, rather than manifold discovery.

Further reading

- Isomap. Tenenbaum, de Silva & Langford, Science, **290**(5500):2319–23 (2000).
- LLE. Roweis & Saul, Science, **290**(5500):2323–6 (2000).
- Laplacian Eigenmaps. Belkin & Niyogi, Neural Comput **23**(6):1373–96 (2003).
- Hessian LLE. Donoho & Grimes, PNAS **100**(10): 5591–6 (2003).
- Maximum variance unfolding. Weinberger & Saul, Int J Comput Vis **70**(1):77–90 (2006).
- Conformal eigenmaps. Sha & Saul ICML **22**:785–92 (2005).
- SNE Hinton & Roweis, NIPS, 2002; t-SNE van der Maaten & Hinton, JMLR, 9:2579–2605, 2008.

More at:

`http://www.gatsby.ucl.ac.uk/~maneesh/dimred/`

See also:

`http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html`