MACHINE LEARNING

Dimensionality Reduction

Olivier LÉZORAY

olivier.lezoray@unicaen.fr

https://lezoray.users.greyc.fr/

LEARNING OBJECTIVES

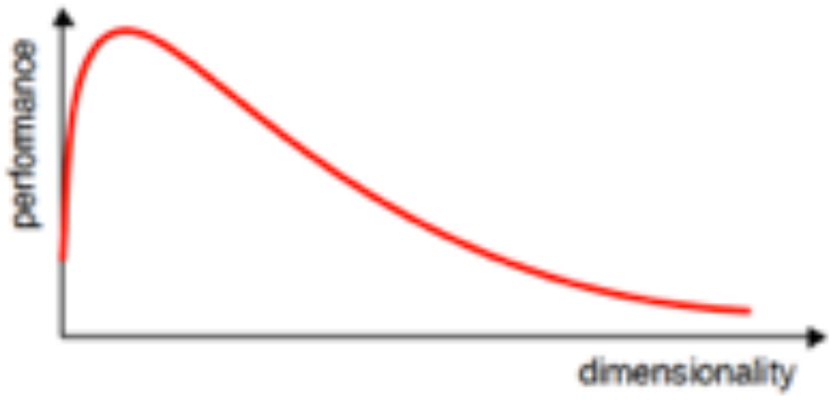
- Curse of dimensionality
- Dimensionality reduction
  - Linear methods
    - PCA
    - LDA
    - MDS
  - Non linear methods
    - ISOMAP
    - LLE
    - Laplacian Eigenmaps
    - Kernel PCA
    - t-SNE

LEARNING OBJECTIVES

- **Curse of dimensionality**
- Dimensionality reduction
  - Linear methods
    - PCA
    - LDA
    - MDS
  - Non linear methods
    - ISOMAP
    - LLE
    - Laplacian Eigenmaps
    - Kernel PCA
    - t-SNE

CURSE OF DIMENSIONALITY

- Many learning methods **don't scale well with large data**: it's the **curse of dimensionality**
  - Gaussian Mixture Models: $O(D^2)$
  - Nearest Neighbors: $O(N\ D)$
- Many learning methods **have decreasing performance for large data sets**.
- For a given dataset of examples, there is a maximum number of dimensions beyond which this degrades the performance of the algorithm.

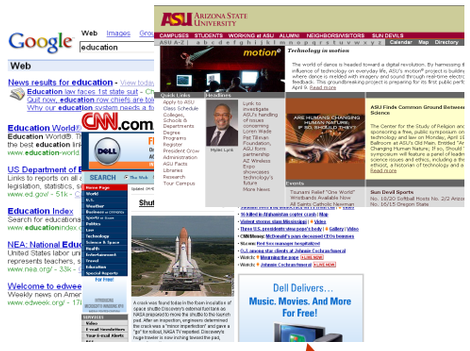


INTEREST OF DIMENSIONALITY REDUCTION

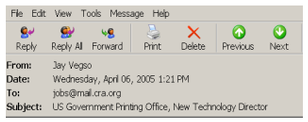
- Identify important attributes
- Statistical motivations (e.g., to remove noise)
- Reduction of the complexity of the learning algorithm
- Reduced complexity of the classifier: fewer parameters and avoid overfitting
- Simpler classifiers are more robust on smaller data volumes
- Motivations for visualization:
  - Visualize data in 2D or 3D
  - The intrinsic dimension of the data may in fact be small !

EXAMPLE : DOCUMENT CLASSIFICATION

Web pages



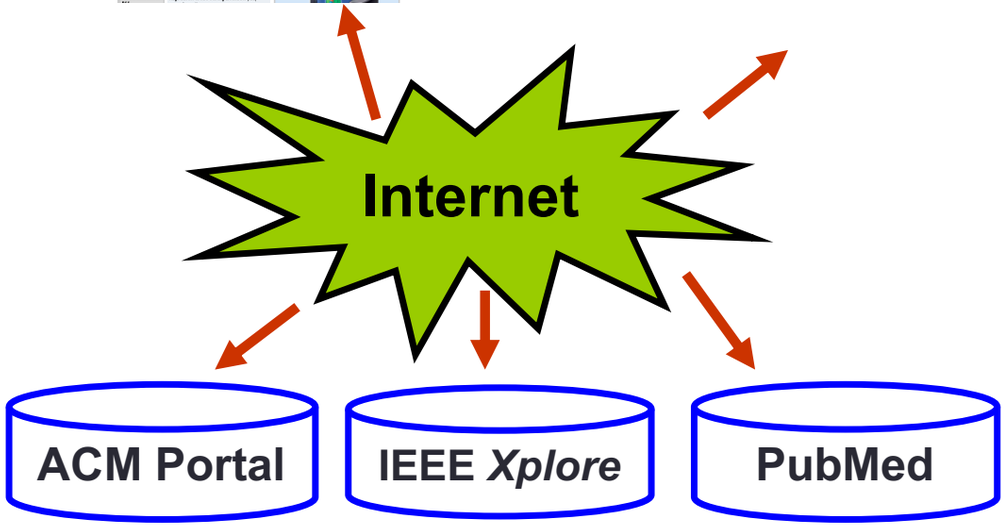
Emails



Documents

Terms

	T_1	T_2	.....	T_N	C
D_1	12	0	.....	6	Sports
D_2	3	10	.....	28	Travel
⋮	⋮			⋮	⋮
D_M	0	11	.....	16	Jobs



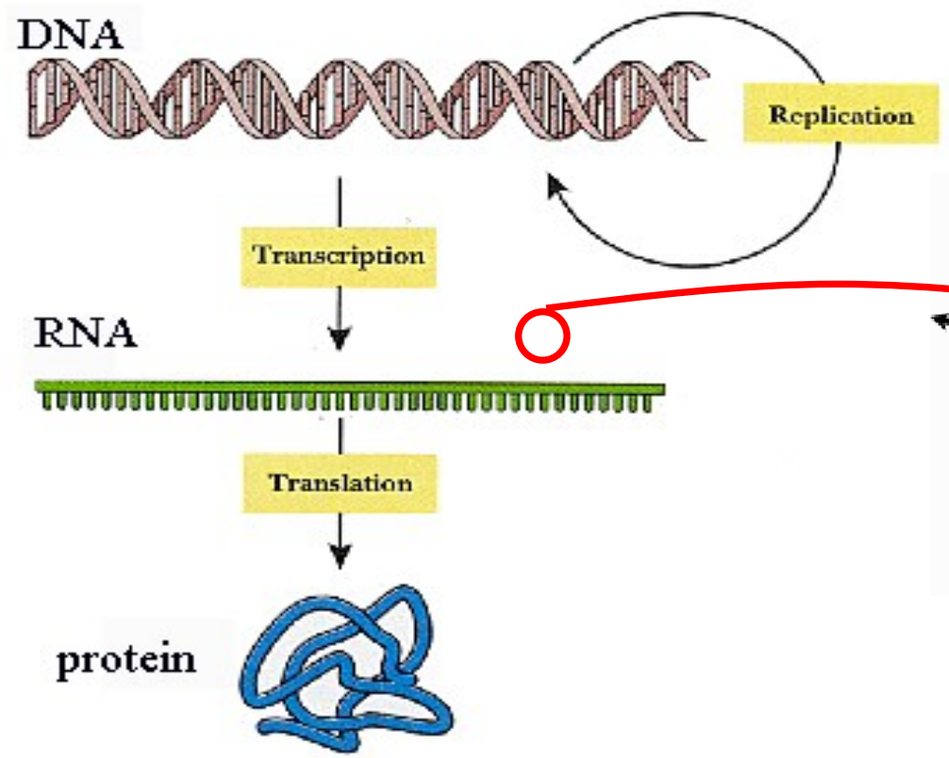
Digital Libraries

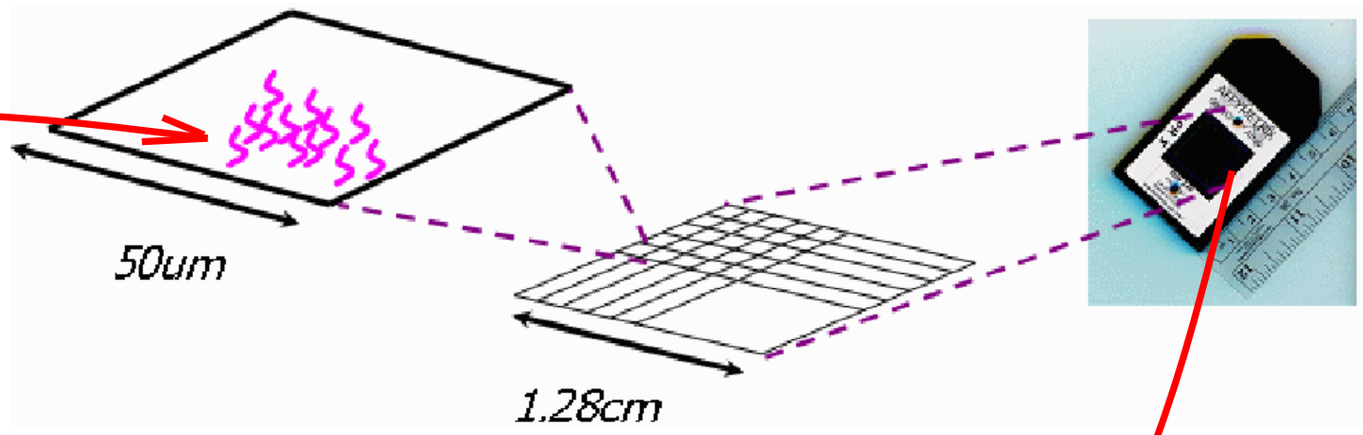
Goal: Classify unknown documents into categories

Challenge: thousands of descriptors

Solution: Dimension reduction

GENES AND DNA CHIPS





Goal: Classify new DNA chips according to diseases

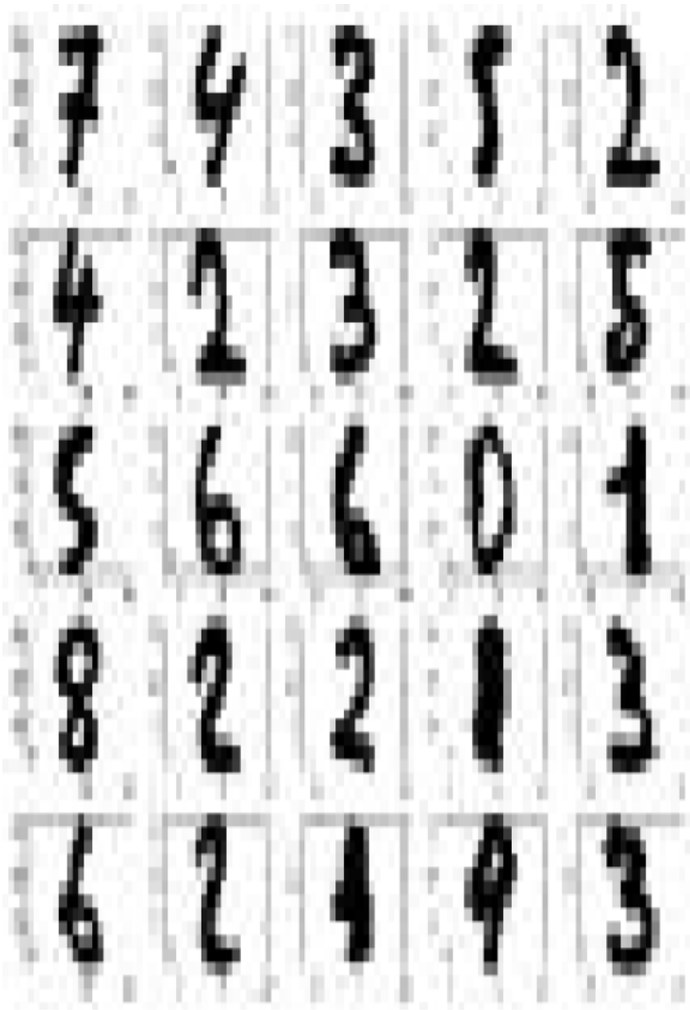
Challenge: thousands of genes, few instances

Solution: Dimension reduction

Sample \ Gene	M23197_at	U66497_at	M92287_at	. . .	Class
Sample 1	261	88	4778	. . .	ALL
Sample 2	101	74	2700	. . .	ALL
Sample 3	1450	34	498	. . .	AML
.	.	.	.	. . .	.
.	.	.	.	. . .	.
.	.	.	.	. . .	.

OTHER DATA OF HIGH DIMENSIONS

Faces



Digits

APPROACHES TO DIMENSION REDUCTION

- Dimension reduction can refer to two distinct processes
- **Feature selection**
  - Among the existing D attributes, we are trying to **select** the d most informative features with d<D

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{bmatrix}$$

- **Feature extraction**
  - The construction of a space of reduced dimension d by **extracting** new features
  - From the existing features, we want to create a new representation that is close to the original data, and whose first dimensions are the most informative.

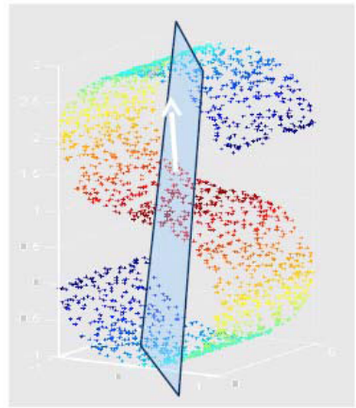
$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{bmatrix} = f\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \right)$$

LEARNING OBJECTIVES

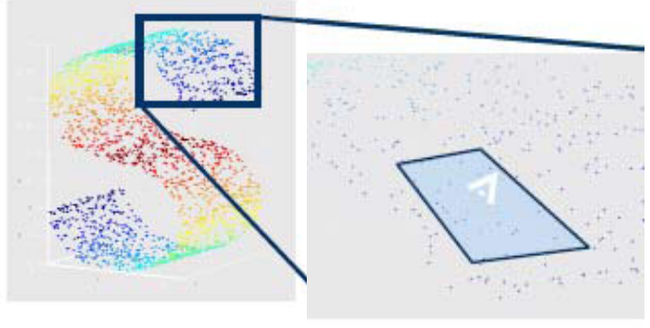
- Curse of dimensionality
- **Dimensionality reduction**
  - Linear methods
    - PCA
    - LDA
    - MDS
  - Non linear methods
    - ISOMAP
    - LLE
    - Laplacian Eigenmaps
    - Kernel PCA
    - t-SNE

FEATURE EXTRACTION METHODS

- **Objective:**
  - Detecting structures of low dimensions in a space of very large dimensions
- **Principle:**
  - Given data, look for a projection with d<D
  - The projection must preserve as much information as possible of the original space in $\mathbb{R}^D$



- **Linear methods**
  - Assume the data lies in a subspace
  - Principal Component Analysis, LDA, MDS
- **Non-linear based on matrix analysis**
  - The underlying small dimension is revealed by a decomposition into eigenvalues and eigenvectors.
  - Linked to the spectral theory on graphs
    - The matrices are created from graphs



LEARNING OBJECTIVES

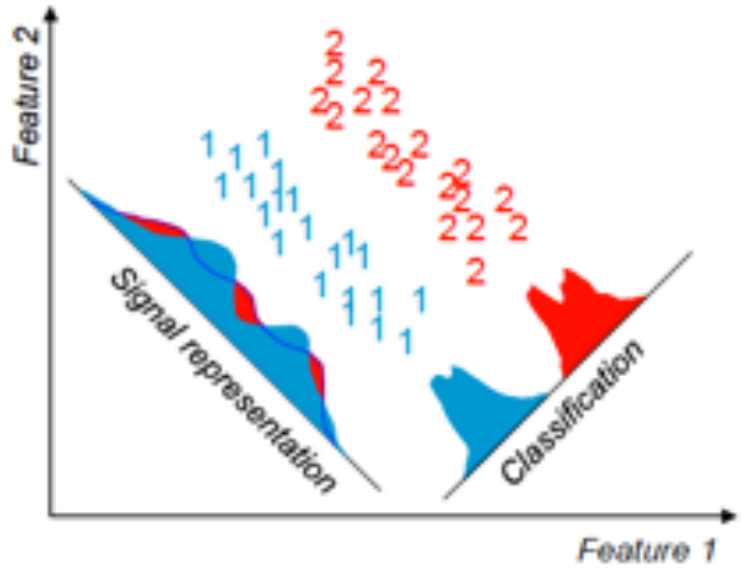
- Curse of dimensionality
- Dimensionality reduction
  - **Linear methods**
    - PCA
    - LDA
    - MDS
  - Non linear methods
    - ISOMAP
    - LLE
    - Laplacian Eigenmaps
    - Kernel PCA
    - t-SNE

LINEAR DIMENSIONALITY REDUCTION

- Many of the popular size reduction methods are linear.

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1D} \\ \vdots & \vdots & \vdots & \vdots \\ w_{d1} & w_{d2} & \cdots & w_{dD} \end{bmatrix} \left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \right)$$

- Finding the transformation $y=f(x)$ is guided by the minimization of an objective function
- Depending on the chosen criterion, linear methods can be distinguished according to an objective of
  - **Signal representation:** the goal of dimension reduction is to efficiently reduce the dimensionality of the representation
  - **Classification:** the goal is to increase the separability of classes in a reduced space size.
- Among the linear methods, we will distinguish:
  - Principal Component Analysis (PCA)
    - Signal representation objective
  - Linear Discriminate Analysis (LDA)
    - Classification objective (**is supervised**)



LEARNING OBJECTIVES

- Curse of dimensionality
- Dimensionality reduction
  - Linear methods
    - **PCA**
    - LDA
    - MDS
  - Non linear methods
    - ISOMAP
    - LLE
    - Laplacian Eigenmaps
    - Kernel PCA
    - t-SNE

PCA : MAXIMUM VARIANCE POINT OF VIEW

- PCA is a transformation that **preserves** as much as possible the **variance** of the initial data
- The projection of $\mathbf{x}$ is $\mathbf{z} = \mathbf{w}^T\mathbf{x}$
- Find $\mathbf{w}$ that maximizes $\mathrm{Var}(\mathbf{z})$

$$\begin{aligned}\mathrm{Var}(\mathbf{z}) &= \mathrm{Var}(\mathbf{w}^T\mathbf{x}) = E[(\mathbf{w}^T\mathbf{x} - E[\mathbf{w}^T\mathbf{x}])^2] = E[(\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\boldsymbol{\mu})^2] \\ &= E[(\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\boldsymbol{\mu})(\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\boldsymbol{\mu})^T] \\ &= E[\mathbf{w}^T(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T\mathbf{w}] \\ &= \mathbf{w}^T E[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T]\mathbf{w} = \mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w}\end{aligned}$$

- with the covariance matrix $\boldsymbol{\Sigma} = [(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T]$

PCA : MAXIMIZE VARIANCE

- PCA Maximizes $\mathrm{Var}(\mathbf{z}) \qquad s.t. \qquad \|\mathbf{w}\| = 1$

$$\max_{\mathbf{w}_1} \mathbf{w}_1^T \mathbf{\Sigma} \mathbf{w}_1 - \alpha(\mathbf{w}_1^T \mathbf{w}_1 - 1)$$

- which has a stationary point if $\mathbf{\Sigma}\mathbf{w}_1 = \alpha \mathbf{w}_1$ i.e. if $\mathbf{w}_1$ is an eigenvector of $\mathbf{\Sigma}$
- The Variance is then $\mathrm{Var}(\mathbf{z}) = \mathbf{w}_1^T \mathbf{\Sigma} \mathbf{w}_1 = \alpha$
- The variance is maximized if $\boldsymbol{\alpha}$ is the eigenvector with the largest eigenvalue.
- Second main component:

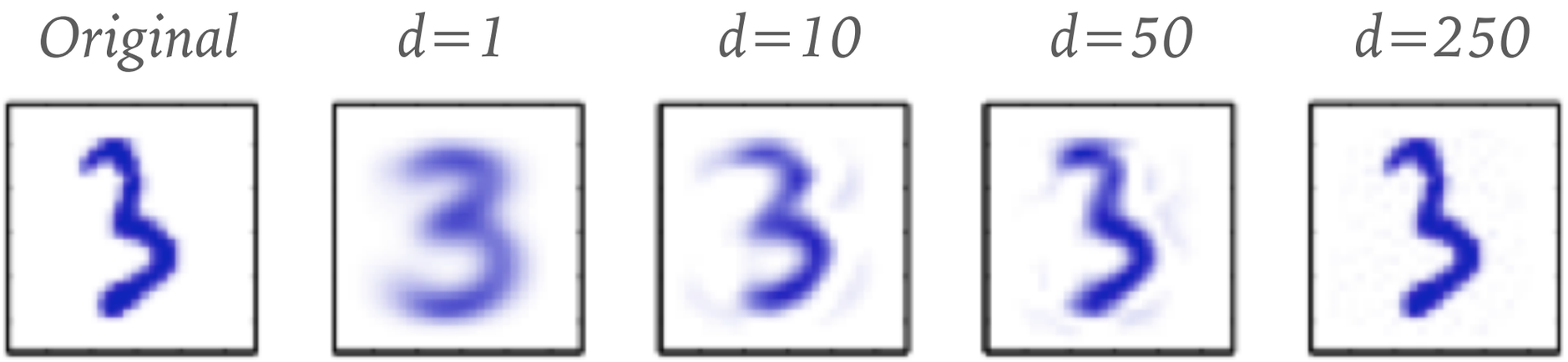
$$\mathrm{Var}(\mathbf{z}_2) \qquad s.t. \qquad \|\mathbf{w}_2\| = 1, \quad \mathbf{w}_2 \perp \mathbf{w}_1$$

$$\max_{\mathbf{w}_2} \mathbf{w}_2^T \mathbf{\Sigma} \mathbf{w}_2 - \alpha(\mathbf{w}_2^T \mathbf{w}_2 - 1) - \beta(\mathbf{w}_2^T \mathbf{w}_1 - 0)$$

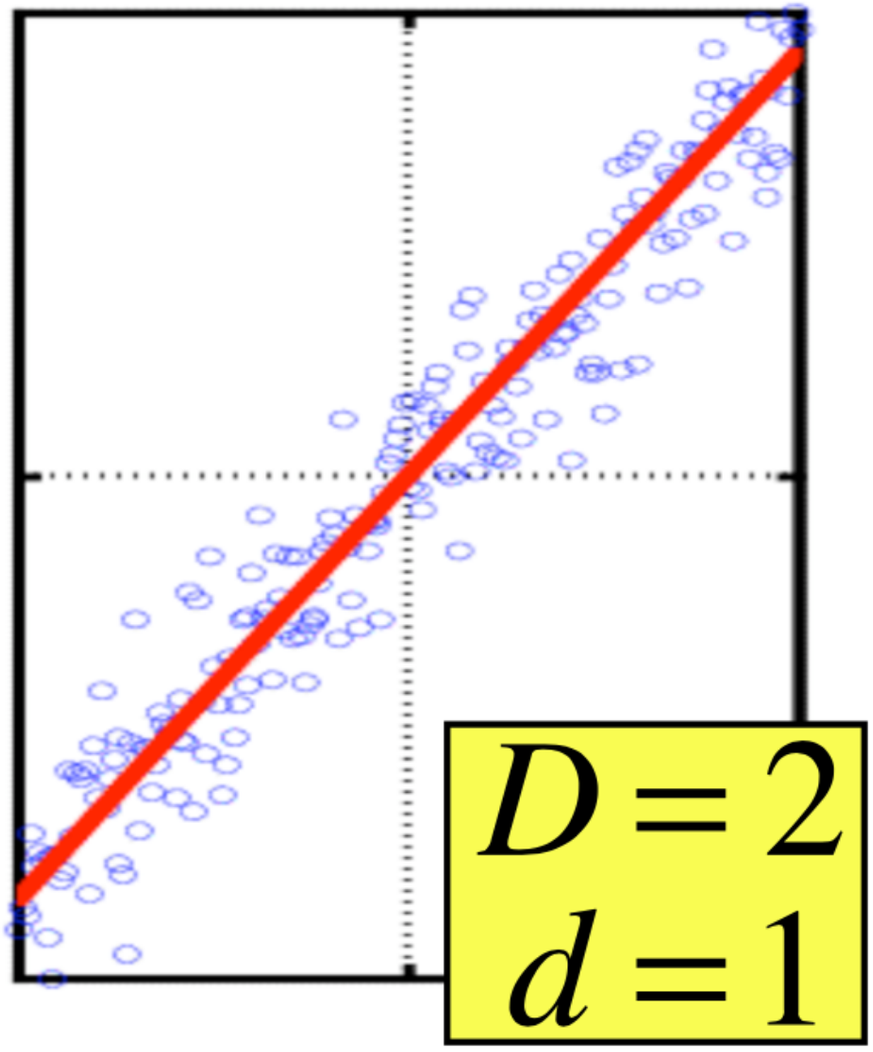
- i.e. $\mathbf{w}_2$ is another eigenvector of $\mathbf{\Sigma}$
- etc.

EXAMPLE

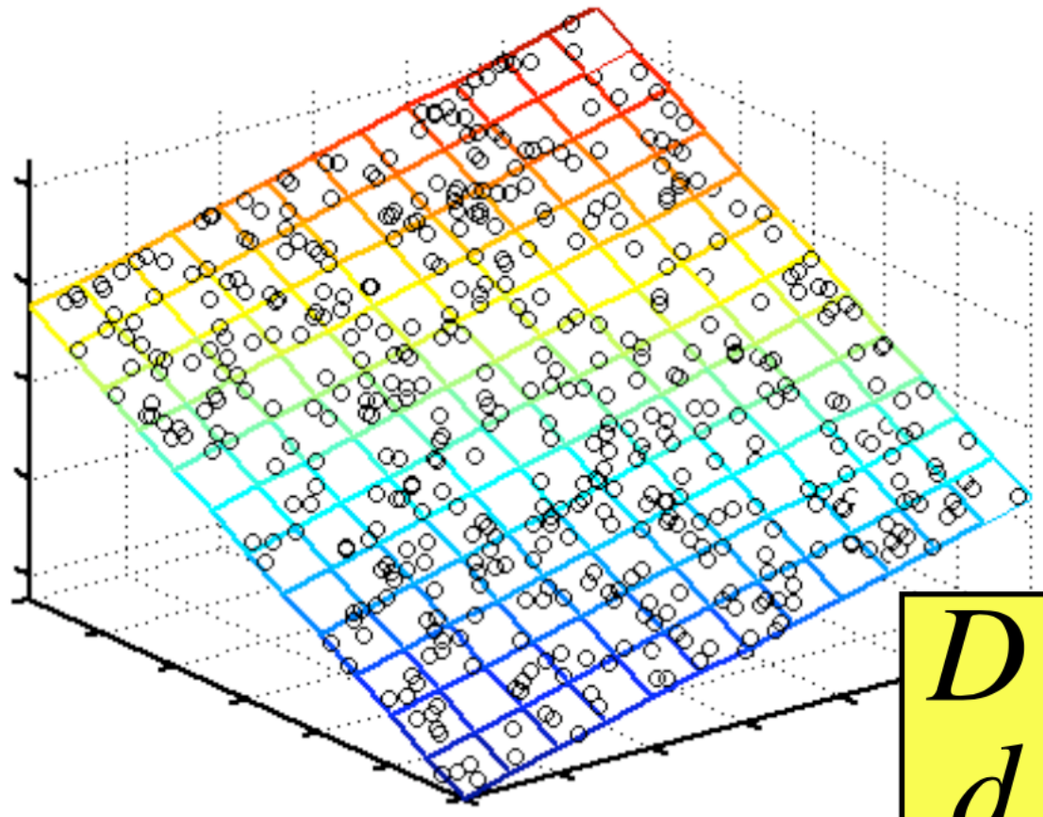
- We vary the number of eigenvectors used for the reconstruction of an image



TOY EXAMPLES

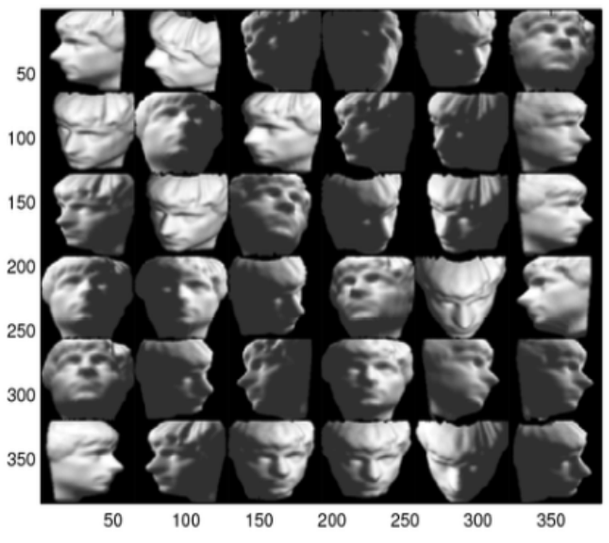


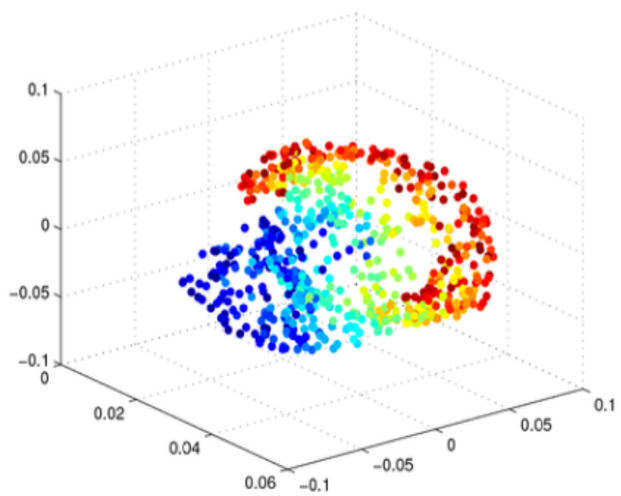
$D = 2$
$d = 1$



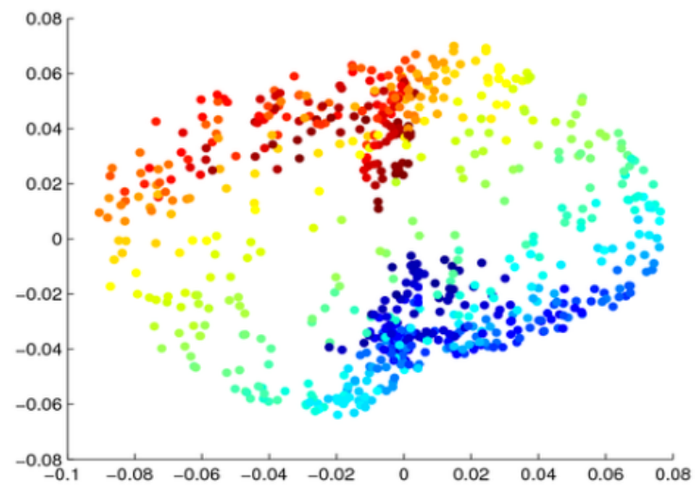
$D = 3$
$d = 2$

EXAMPLE ON REAL DATA





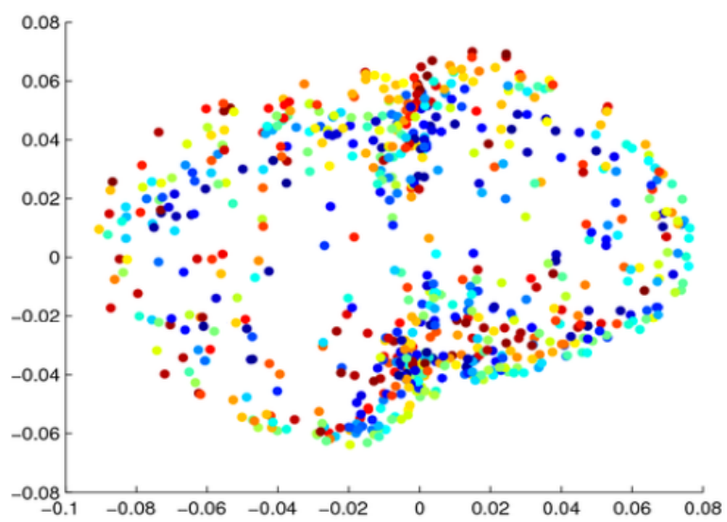
3D-proj: light



2D-proj: pose 1

Dimension: $64 * 64 = 4096$.
$N = 698$

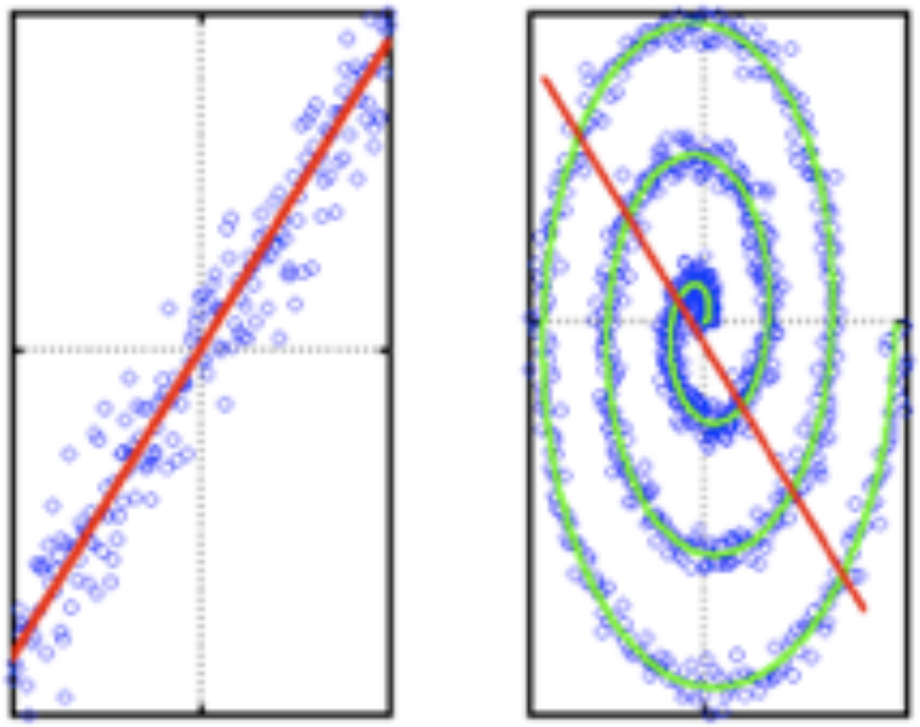
3 free parameters:
- left-right pose,
- up-down pose,
- light pose.



2D-proj: pose 2

PCA INTERPRETATION AND PROPERTIES

- Eigenvectors
  - Main axes of maximum variance
- Eigenvalues
  - Projected variance of inputs along the main axes
- Significant dimension
  - The number of high and non-negative eigenvalues
- **Pros**
  - Decomposition method
  - No parameters
  - Non iterative
  - No local minimum
- **Cons**
  - Limited to linear projections



LEARNING OBJECTIVES

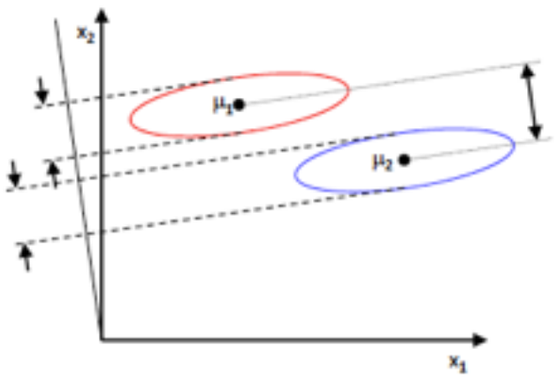
- Curse of dimensionality
- Dimensionality reduction
  - Linear methods
    - PCA
    - **LDA**
    - MDS
  - Non linear methods
    - ISOMAP
    - LLE
    - Laplacian Eigenmaps
    - Kernel PCA
    - t-SNE

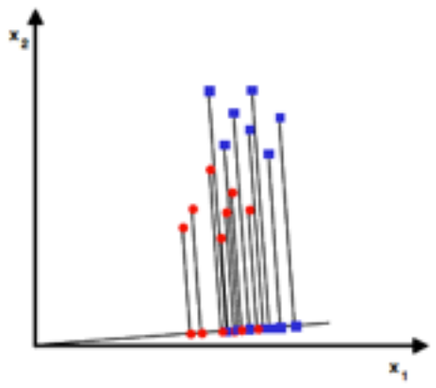
LINEAR DISCRIMINANT ANALYSIS – LDA

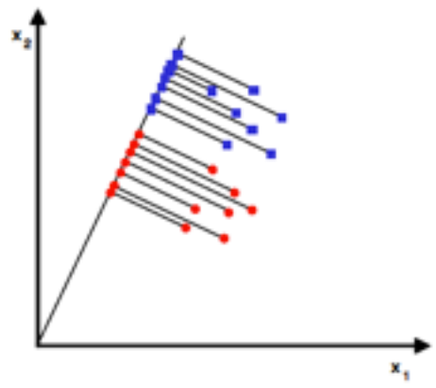
- LDA seeks to reduce dimensionality by making sure that classes are well separated.
- A separation measure is required:
  - We are looking for a projection where the examples of the same class have a close projection with the averages of the projections of the classes that are far away.
  - The Fisher LDA seeks to maximize the difference between projected averages normalized by a measure of intra-class dispersion (called scattering).

$$J(\mathbf{w}) = \frac{(\tilde{\boldsymbol{\mu}}_1 - \tilde{\boldsymbol{\mu}}_2)^2}{\tilde{s}_1^2 + \tilde{s}_2^2} \quad \textit{with} \quad \tilde{s}_i^2 = \sum_{\mathbf{y} \in C_i} (\mathbf{y} - \tilde{\boldsymbol{\mu}}_i)^2 \quad \textit{and} \quad \mathbf{y} = \mathbf{w}^T \mathbf{x}$$

$$\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x} \quad \textit{and} \quad \tilde{\boldsymbol{\mu}}_i = \frac{1}{|C_i|} \sum_{\mathbf{y} \in C_i} \mathbf{y} = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{w}^T \mathbf{x} = \mathbf{w}^T \boldsymbol{\mu}_i$$







FISHER LDA

- We reformulate $J(w)$ in function of w
- $\mathbf{S}_B$ is the Between-class dispersion matrix, and

$$(\tilde{\boldsymbol{\mu}}_1 - \tilde{\boldsymbol{\mu}}_2)^2 = (\mathbf{w}^T\boldsymbol{\mu}_1 - \mathbf{w}^T\boldsymbol{\mu}_2)^2 = \mathbf{w}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\mathbf{w} = \mathbf{w}^T\mathbf{S}_B\mathbf{w}$$

- $\mathbf{S}_W$ is the Within-class dispersion matrix, and one has

$$\mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2 \quad \text{with} \quad \mathbf{S}_i = \sum_{\mathbf{x}\in C_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^T$$

- Therefore

$$\tilde{s}_i^2 = \sum_{\mathbf{y}\in C_i} (\mathbf{y} - \tilde{\boldsymbol{\mu}}_i)^2 = \sum_{\mathbf{x}\in C_i} (\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\boldsymbol{\mu}_i)^2$$

$$= \sum_{\mathbf{x}\in C_i} \mathbf{w}^T(\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^T\mathbf{w} = \mathbf{w}^T\mathbf{S}_i\mathbf{w}$$

FISHER LDA

- Finally $\tilde{s}_1^2 + \tilde{s}_2^2 = \mathbf{w}^T \mathbf{S}_W \mathbf{w}$
- and one has

$$J(\mathbf{w}) = \frac{(\tilde{\boldsymbol{\mu}}_1 - \tilde{\boldsymbol{\mu}}_2)^2}{\tilde{s}_1^2 + \tilde{s}_2^2} = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

- The final solution is

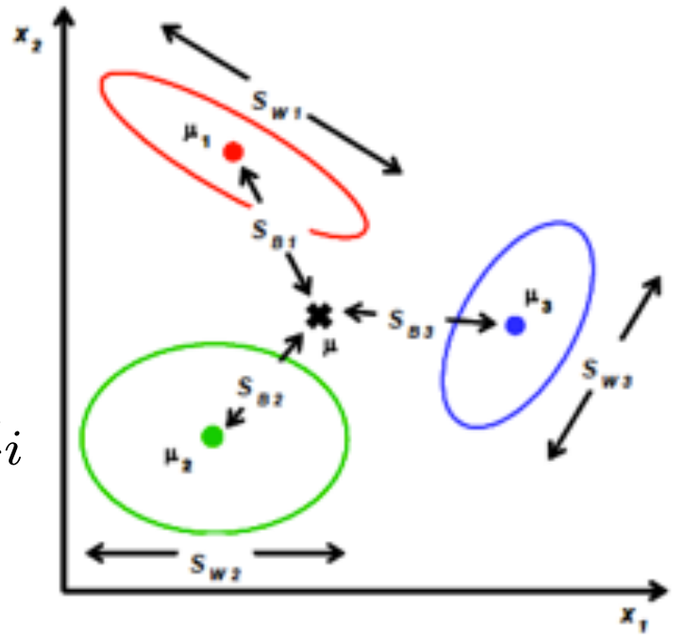
$$\mathbf{w}^* = \arg\max \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} = \mathbf{S}_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

- **However this formulation is valid only for two classes**

FISHER LDA FOR MORE THAN TWO CLASSES

- Instead of a 1D projection y, we seek (K-1) projections by (K-1) projection vectors arranged in a matrix $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_{K-1} \end{bmatrix}$
- Within-class scatter: $\mathbf{S}_W = \sum_{i=1}^{K} \mathbf{S}_i$

  with $\mathbf{S}_i = \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^T$ *and* $\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$



- Between-class scatter:

$$\mathbf{S}_B = \sum_{i=1}^{K} |C_i| (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \quad \textit{with} \quad \boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$$

- Find $\mathbf{W}^* = \begin{bmatrix} \mathbf{w}_1^* \\ \mathbf{w}_2^* \\ \vdots \\ \mathbf{w}_{K-1}^* \end{bmatrix} = \arg\max \frac{|\mathbf{W}^T \mathbf{S}_B \mathbf{W}|}{|\mathbf{W}^T \mathbf{S}_W \mathbf{W}|}$

The solution is given by

$$(\mathbf{S}_B - \lambda_i \mathbf{S}_W)\mathbf{w}_i^* = 0$$

Can be computed by the eigenvectors of $\mathbf{S}_W^{-1} \mathbf{S}_B$

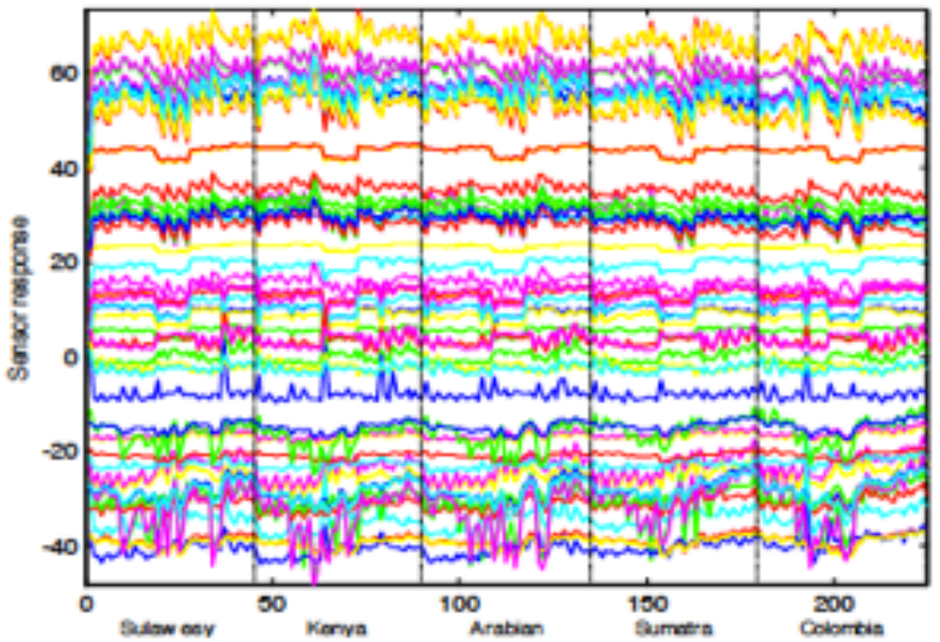
LDA VS PCA

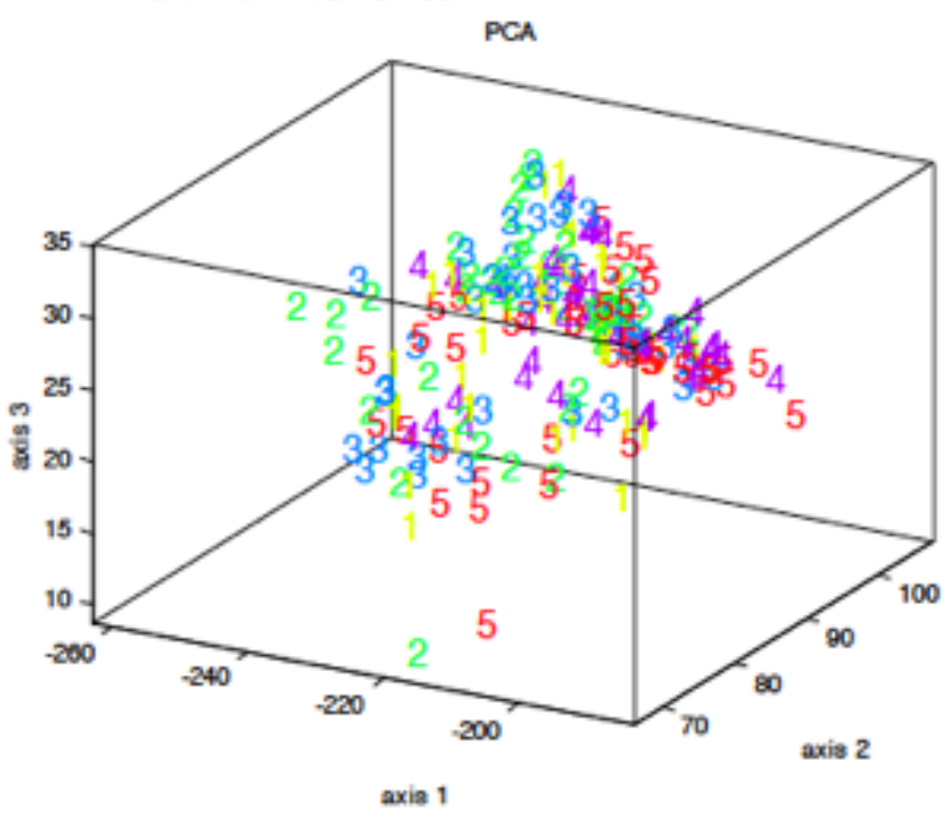
This example illustrates the performance of PCA and LDA on an odor recognition problem

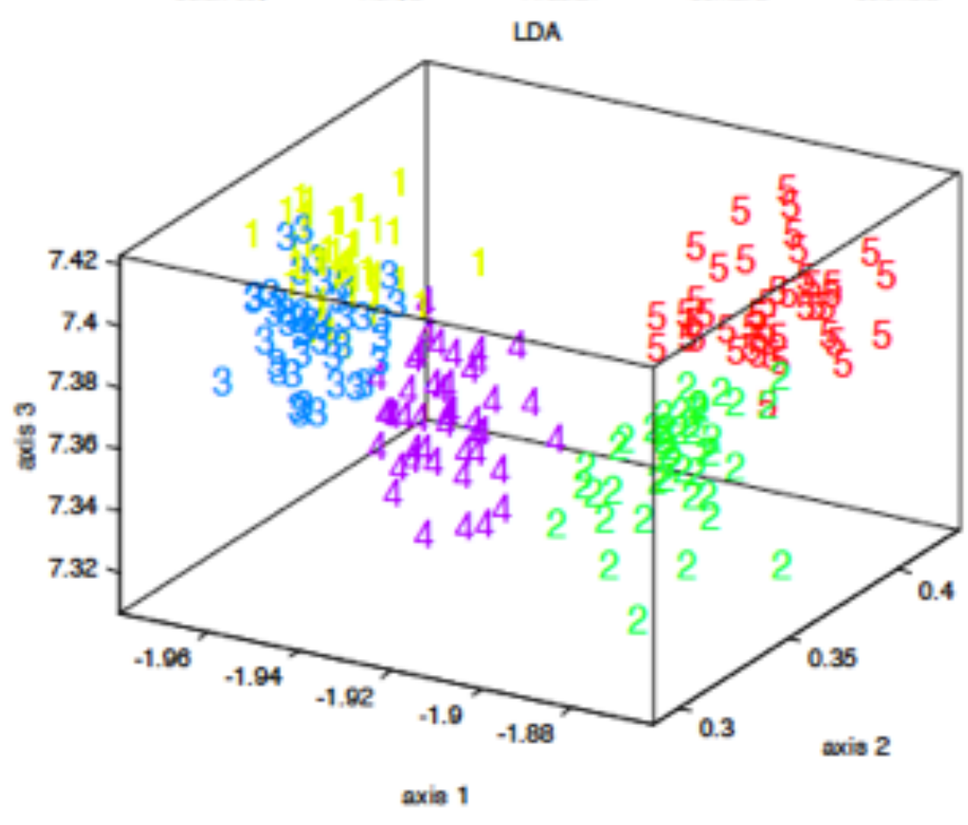
- Five types of coffee beans were presented to an array of gas sensors
- For each coffee type, 45 "sniffs" were performed and the response of the gas sensor array was processed in order to obtain a 60-dimensional feature vector

Results

- From the 3D scatter plots it is clear that LDA outperforms PCA in terms of class discrimination
- This is one example where the discriminatory information is not aligned with the direction of maximum variance







LEARNING OBJECTIVES

- Curse of dimensionality
- Dimensionality reduction
  - Linear methods
    - PCA
    - LDA
    - **MDS**
  - Non linear methods
    - ISOMAP
    - LLE
    - Laplacian Eigenmaps
    - Kernel PCA
    - t-SNE

METRIC MULTIDIMENSIONAL SCALING (MDS)

- This time we do not know the representations $\mathbf{x}_i \in \mathbb{R}^D$ of the examples $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$, we have only access to a a matrix $\boldsymbol{\Delta}$ of N(N-1)/2 distances between the examples.
- We have distances in the original space that can be expressed as a scalar product $\delta(\mathbf{x}_i, \mathbf{x}_j)^2 = (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T$
- We suppose that the original data are centered $\bar{\mathbf{x}} = 0$
- MDS seeks to find $\mathbf{y}_1, \cdots, \mathbf{y}_N \in \mathbb{R}^d$ such that

$$\|\mathbf{y}_i - \mathbf{y}_j\|_2 \approx \delta(\mathbf{x}_i, \mathbf{x}_j)$$

- We are going to work with scalar products instead of distances

USING SCALAR PRODUCTS

- We can consider the gram matrix $\mathbf{B} = \mathbf{\Delta}\mathbf{\Delta}^T$ of scalar products instead of the distance matrix $\mathbf{\Delta}$

- We have (since the original data are centered) :

$$\delta(\mathbf{x}_i, \mathbf{x}_j)^2 = (b_{ii} + b_{jj} - 2b_{ij}) \quad \textit{from} \quad \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 = \mathbf{x}_i^T\mathbf{x}_i + \mathbf{x}_j^T\mathbf{x}_j - 2\mathbf{x}_i^T\mathbf{x}_j$$

- From this, one can prove that

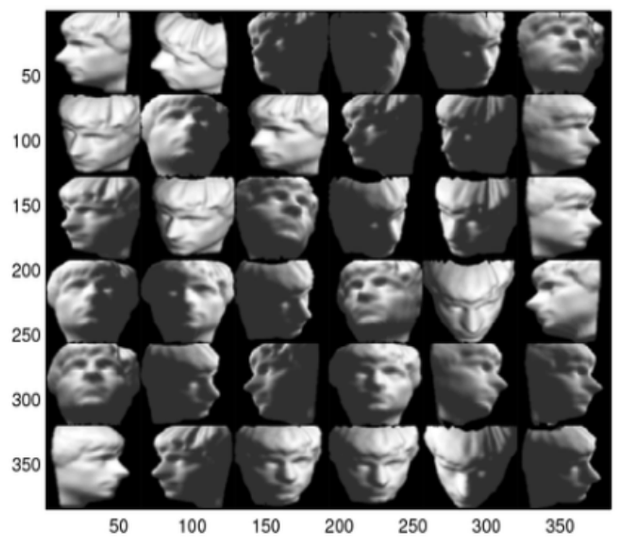
$$b_{ij} = -\frac{1}{2}\left(\delta_{ij}^2 - \frac{1}{n}\sum_{k=1}^{N}(\delta_{ik}^2 + \delta_{kj}^2) + \frac{1}{n^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\delta_{ij}^2\right)$$

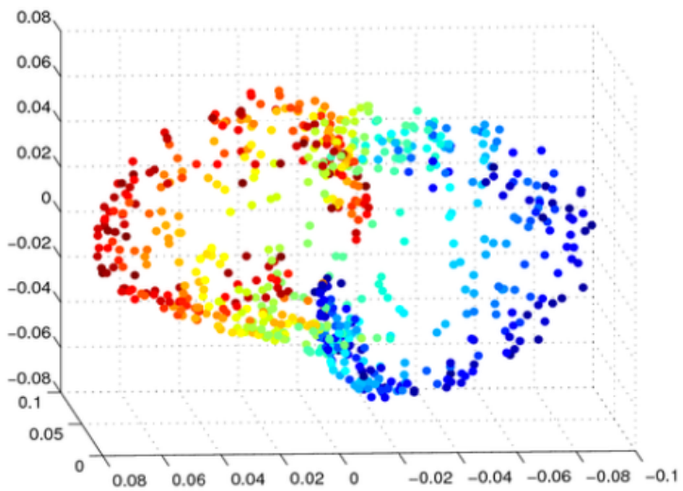
- Therefore we want to minimize

$$\sum_{i,j}(\mathbf{y}_i^T\mathbf{y}_j - b_{ij})^2$$

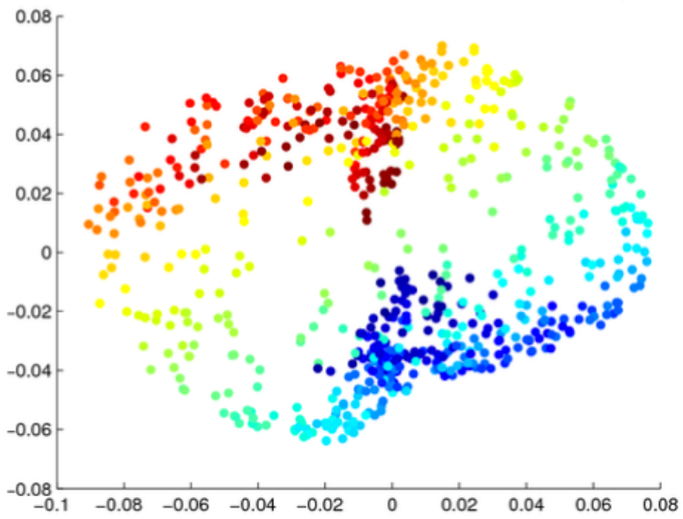
A solution is given by the eigendecomposition of B

MDS EXAMPLE





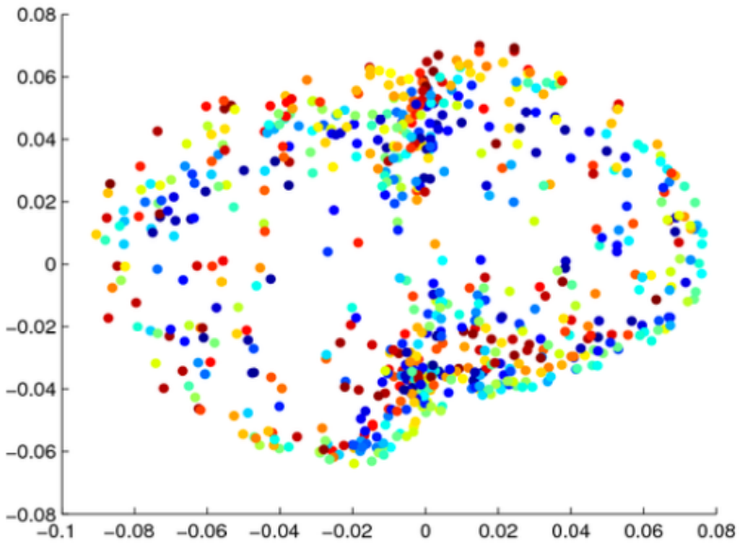
3D-proj: light



2D-proj: pose 1

Dimension: $64 * 64 = 4096$.
$N = 698$

3 free parameters:
- left-right pose,
- up-down pose,
- light pose.



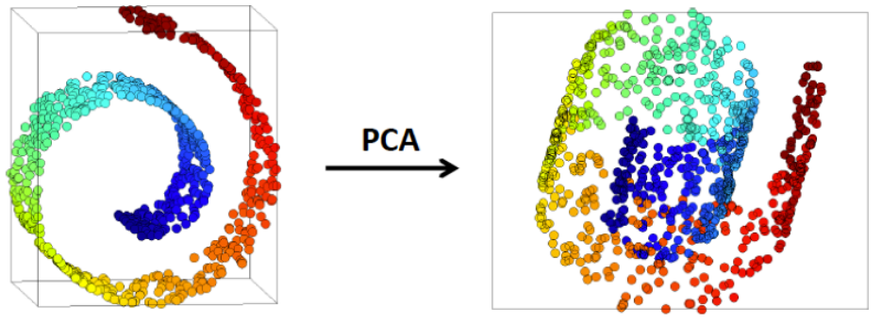
2D-proj: pose 2

LEARNING OBJECTIVES

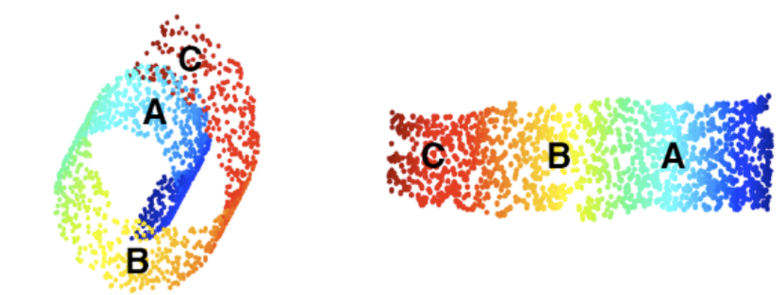
- Curse of dimensionality
- Dimensionality reduction
  - Linear methods
    - PCA
    - LDA
    - MDS
  - **Non linear methods**
    - ISOMAP
    - LLE
    - Laplacian Eigenmaps
    - Kernel PCA
    - t-SNE

TURNING NON LINEAR

- PCA/LDA perform a global transformation on the data (translation/ rotation/rescaling)
  - These methods assume that the data live in a linear subspace and become inefficient when the data is located around highly non linear manifolds



- Even for metric methods such as MDS, this does not necessarily preserve distances well relative to the underlying variety of data.

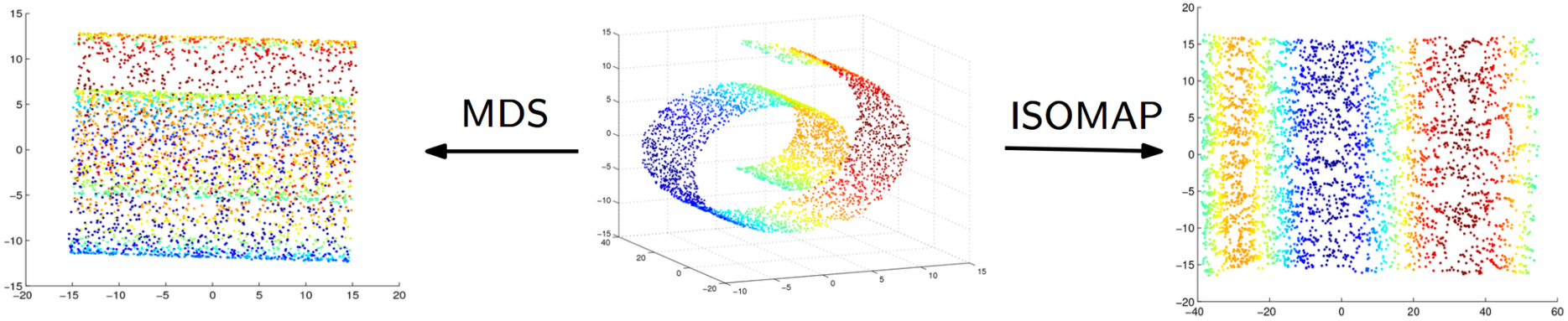


LEARNING OBJECTIVES

- Curse of dimensionality
- Dimensionality reduction
  - Linear methods
    - PCA
    - LDA
    - MDS
  - Non linear methods
    - **ISOMAP**
    - LLE
    - Laplacian Eigenmaps
    - Kernel PCA
    - t-SNE

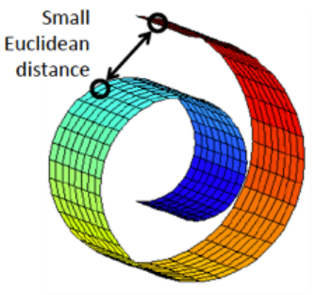
ISOMAP

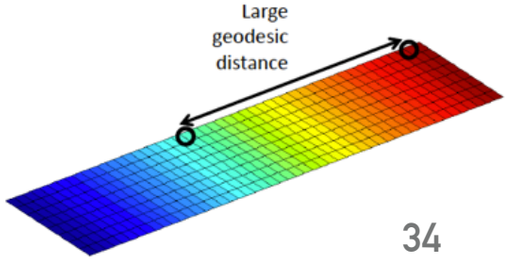
- Variant of MDS where the matrix of Euclidean distances between data points is replaced by the matrix of the geodesic distances between data points.



Idea :

- *find a transformation that preserves the geodesic distances measured on a graph of the initial data.*
- *For close examples, the Euclidean distance provides a good approximation of the geodesic distance.*
- *For remote examples, the geodesic distance can be approximated by a calculation of shortest paths.*

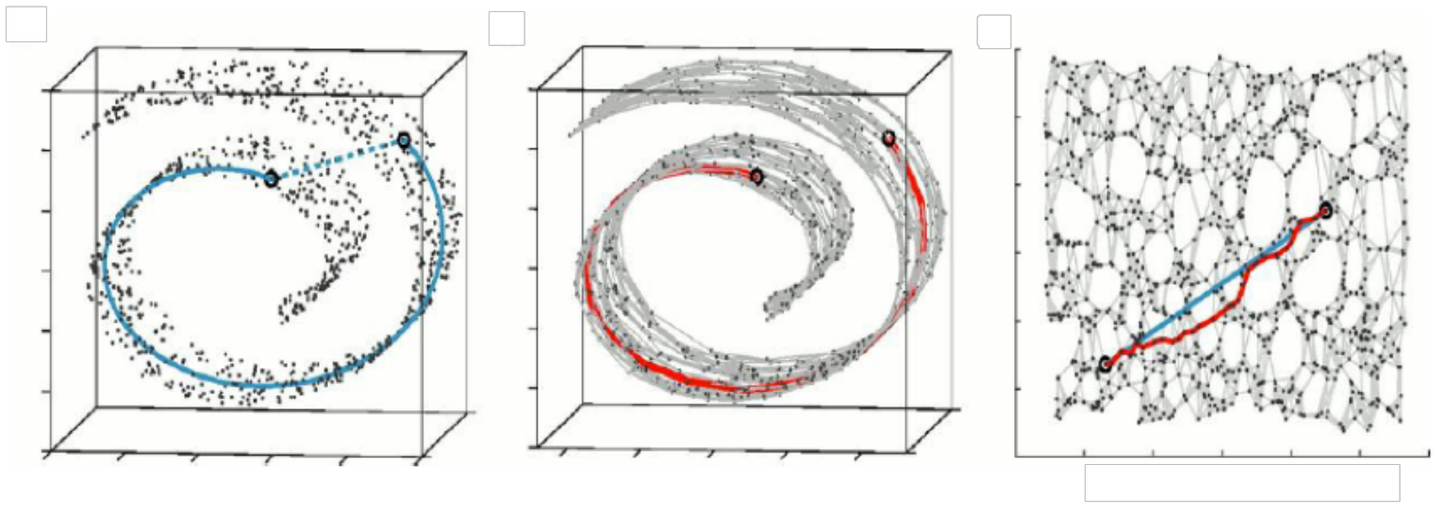




ISOMAP PRINCIPLE

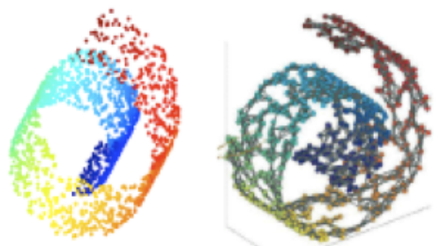
- Use geodesic distances instead of Euclidean distances in MDS
- Given a matrix of distances between points in the original space, find vectors such that $\|\mathbf{y}_i - \mathbf{y}_j\|_2 \approx g(\delta(\mathbf{x}_i, \mathbf{x}_j))$
- Three main steps

1. Find the closest neighbors for each example and built a graph
2. Find the shortest paths
3. Apply MDS



THE THREE STEPS

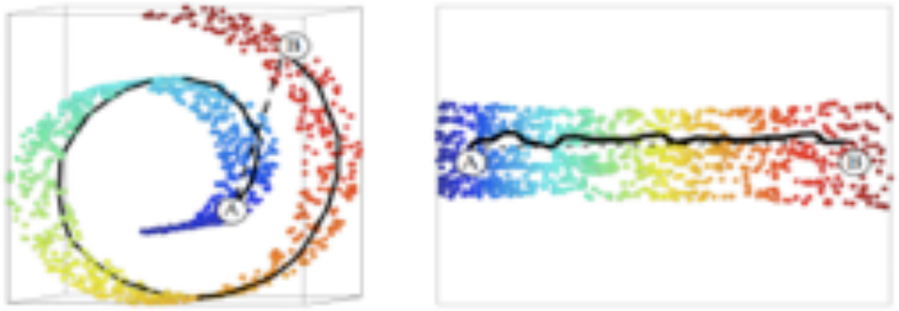
- First step
  - Consider a distance measure $\delta(\mathbf{x}_i, \mathbf{x}_j)$ in the original space.
  - Construct a k nearest neighbor graph weighted by **δ**, making sure that the graph is connected.
  - The graph is an approximation of the underlying variety.



- Second step
  - Estimate the geodesic distance matrix $\mathbf{D}_G$ between all pairs of examples by the Dijkstra algorithm
- Third step
  - Finding the projection that best preserves the estimated geodesic distances
  - Perform MDS on the geodesic distance matrix $\mathbf{D}_G$

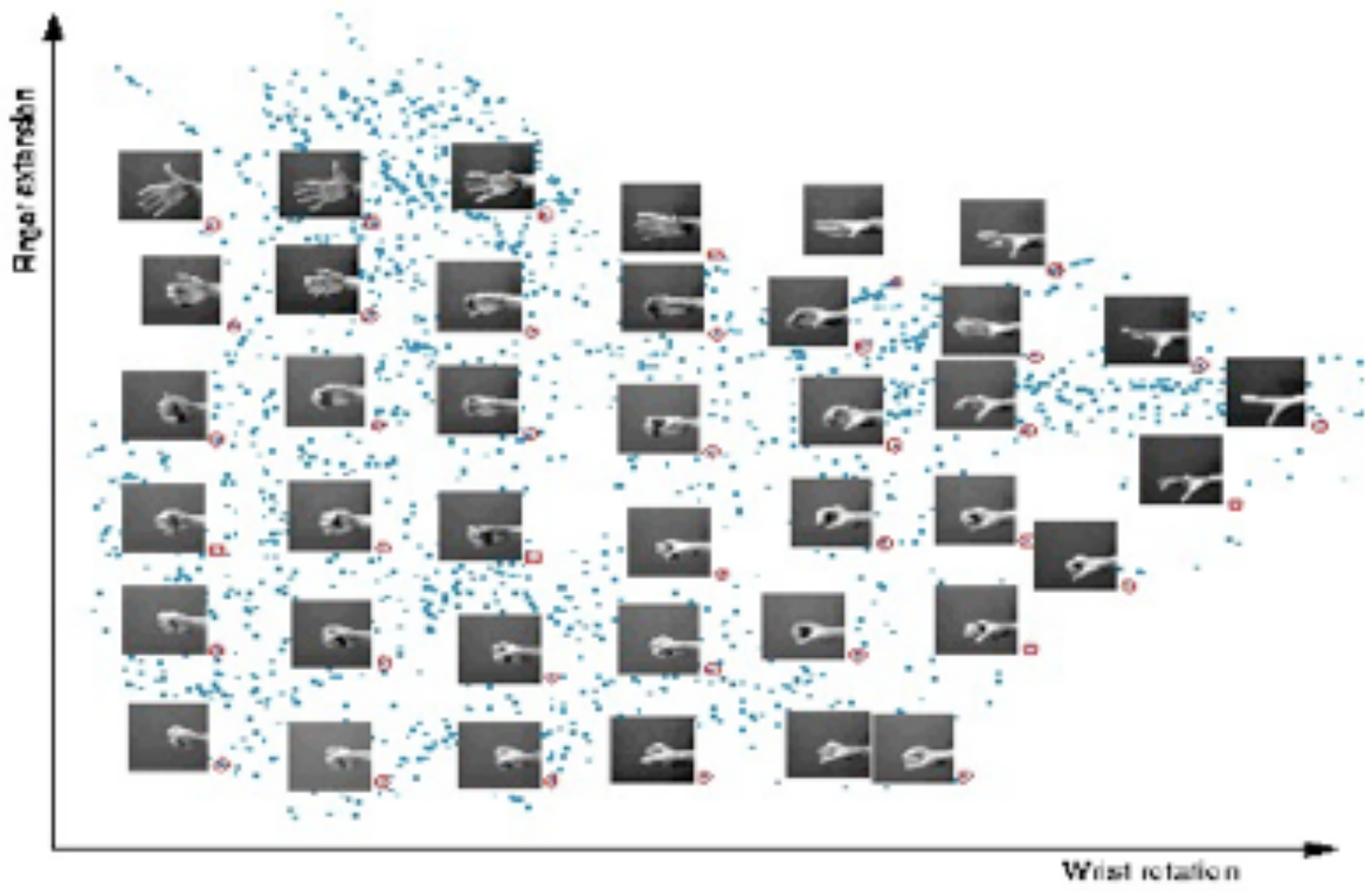
EXAMPLES

- **Swiss roll**



$n = 1024$
$k = 12$

- **Wrist images**



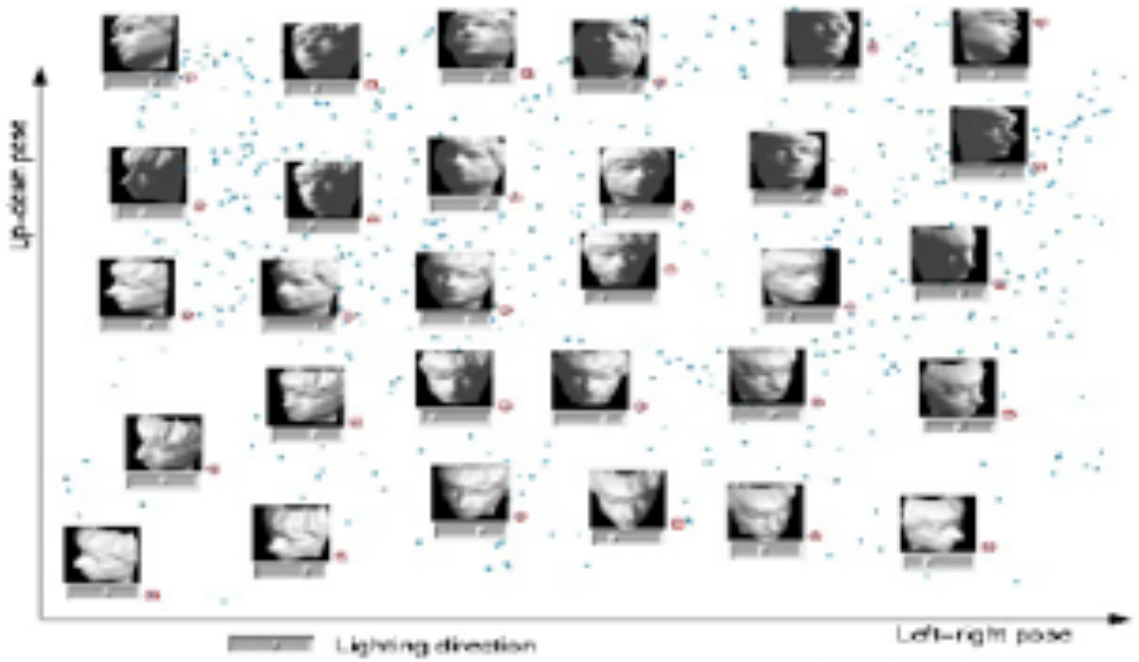
$n = 2000$
$k = 6$
$D = 64^2$

EXAMPLES

Examples

- **Face images**

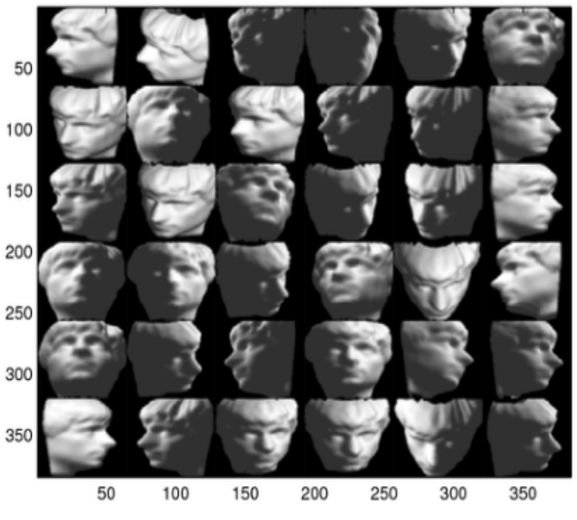
$n = 698$
$k = 6$



- **Digit images**

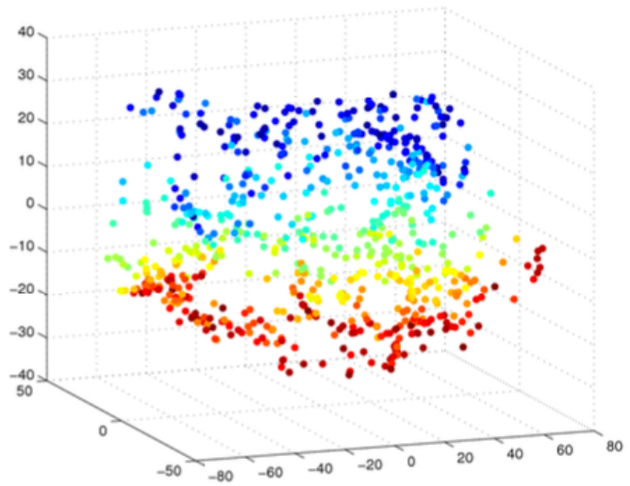
$n = 1000$
$r = 4.2$
$D = 20^2$

EXAMPLE

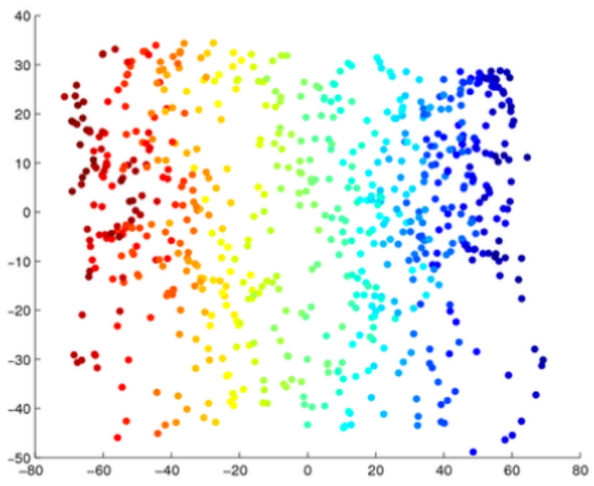


Dimension: $64 * 64 = 4096$.
$N = 698$

3 free parameters:
- left-right pose,
- up-down pose,
- light pose.

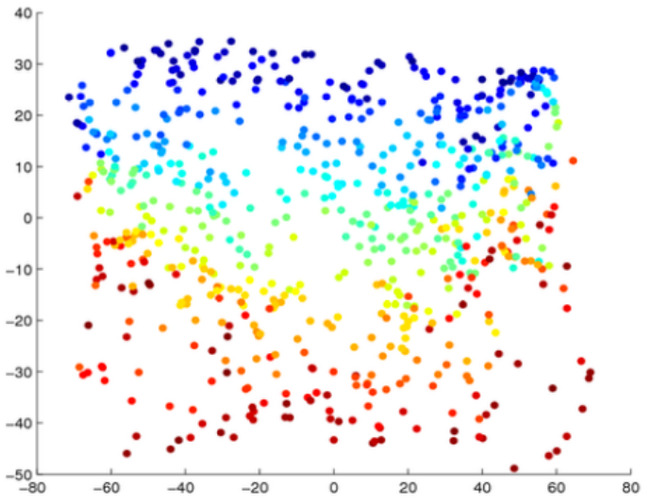


3D-proj: light



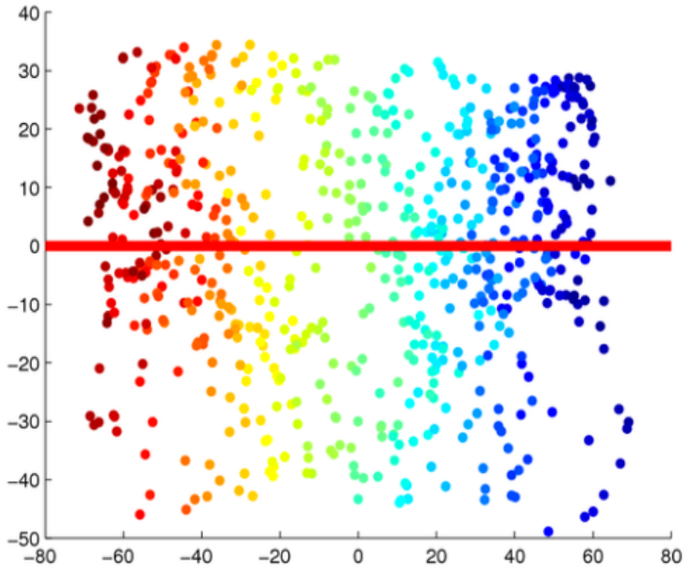
2D-proj: pose 1

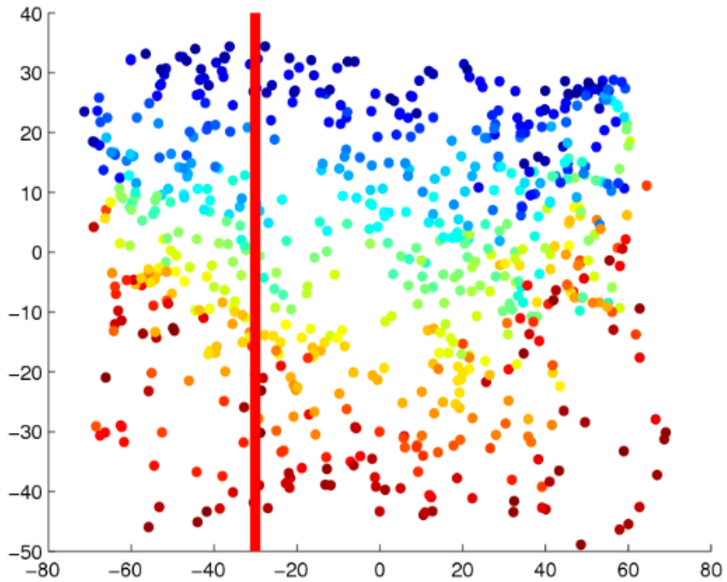
$k = 6$ NN

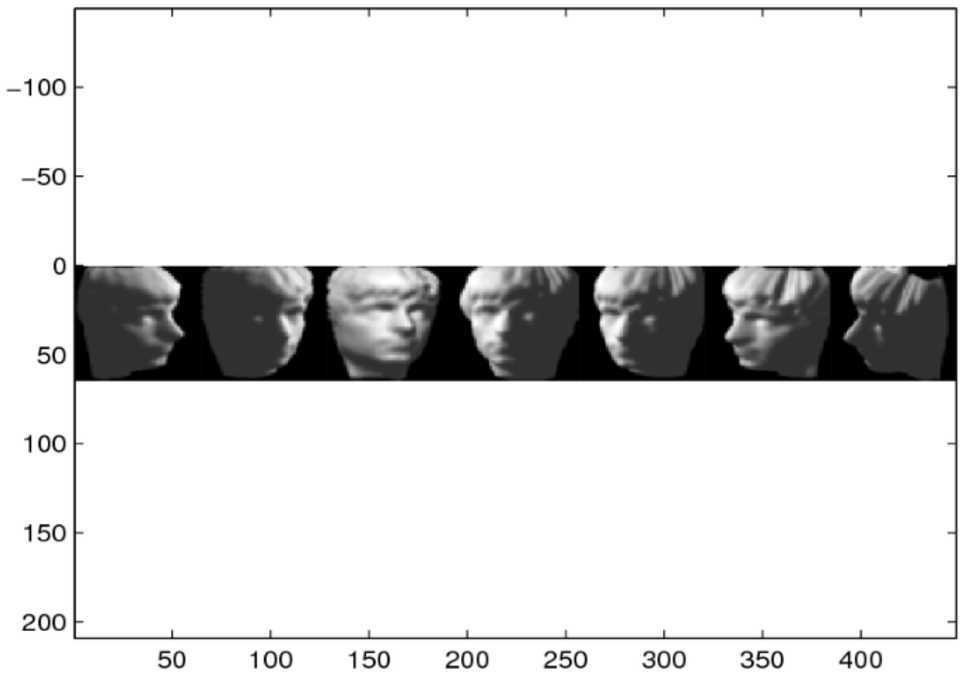


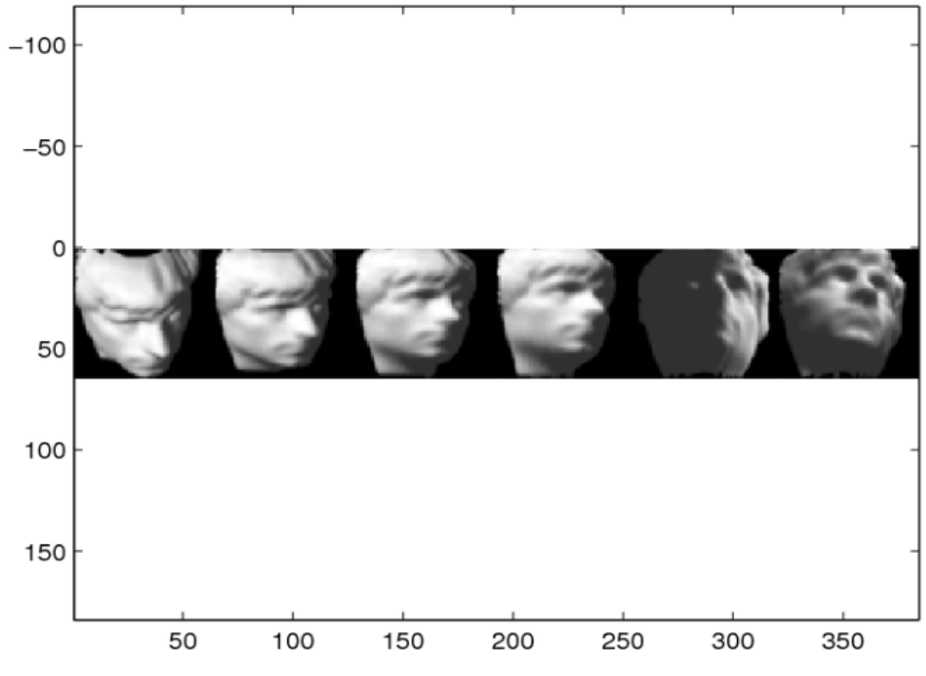
2D-proj: pose 2

EXAMPLE


40
30
20
10
0
−10
−20
−30
−40
−50
−80
−60
−40
−20
0
20
40
60
80


40
30
20
10
0
−10
−20
−30
−40
−50
−80
−60
−40
−20
0
20
40
60
80


−100
−50
0
50
100
150
200
50
100
150
200
250
300
350
400


−100
−50
0
50
100
150
50
100
150
200
250
300
350

ISOMAP PROPERTIES

- **Pros:**
- ISOMAP is guaranteed to discover the true dimensionality and geometry of a certain class of Euclidean varieties.
- This guarantee comes from the fact that when the number of examples increases the geodesic distance measured on the graph is a good estimate of the true geodesic distance
- **Cons:**
- ISOMAP is very sensitive to short circuits.
- Computation of eigenvalues on a large Gram Matrix is computationally demanding

LEARNING OBJECTIVES

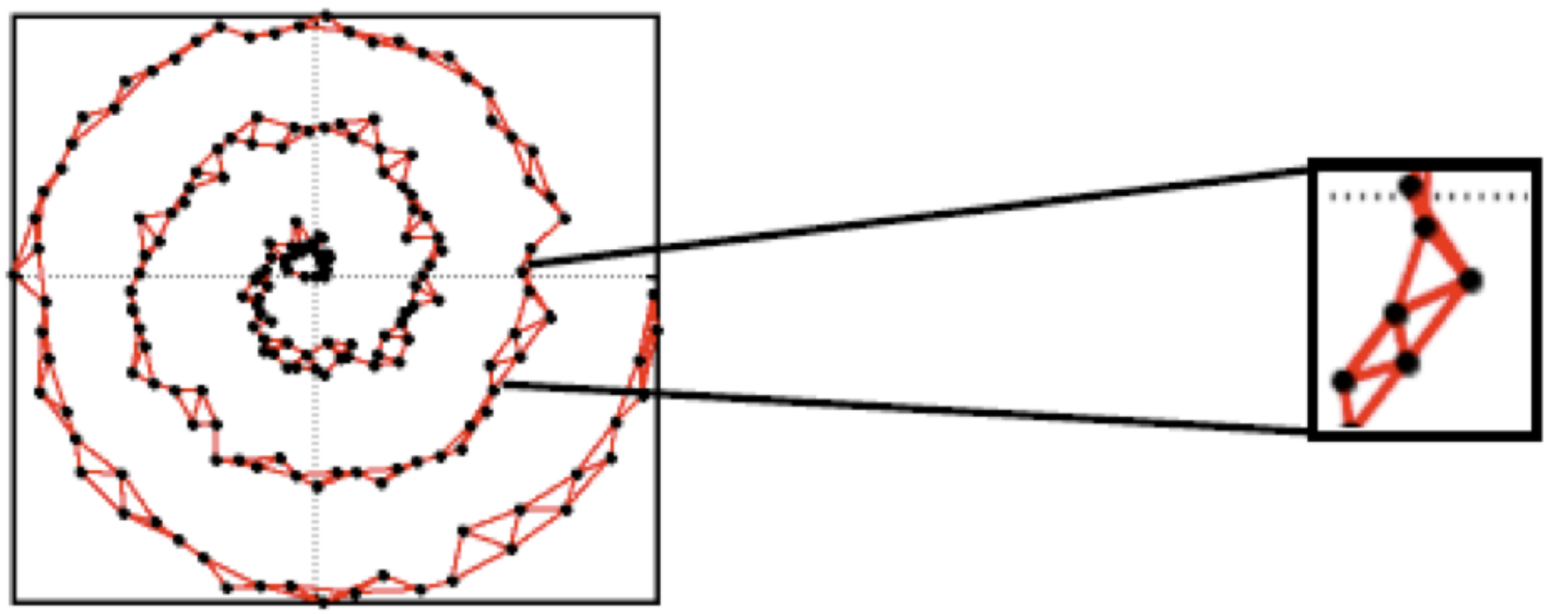
- Curse of dimensionality
- Dimensionality reduction
  - Linear methods
    - PCA
    - LDA
    - MDS
  - Non linear methods
    - ISOMAP
    - **LLE**
    - Laplacian Eigenmaps
    - Kernel PCA
    - t-SNE

LOCALLY LINEAR EMBEDDING: LLE

- LLE uses a different strategy than ISOMAP to find the overall structure of the underlying variety.
- LLE exploits distances measured on **local linear neighborhoods**
- **Intuition**
  - The examples are assumed to come from a sampling of the variety.
  - If sufficient data are available (i.e., the variety is well sampled), it can be assumed that each example and its neighbors live on a locally linear patch of the variety.
- LLE solves the problem in two steps
  - Find a linear combination approximating each example of the input space
  - Find coordinates in a small space that are compatible with the linear approximation

LOCAL CHARACTERIZATION OF THE VARIETY

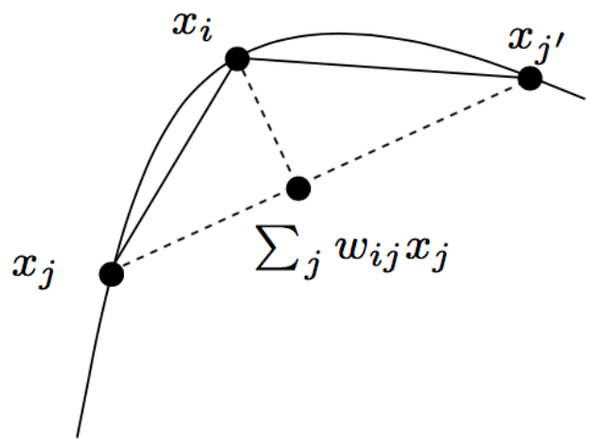
- The neighborhood graph G must be connected
- Local neighborhoods (called patches) should locally reflect the data structure
- The weights characterize the local structure of each neighborhood: calculated by linear reconstruction of the inputs from the neighbors.



FIRST STEP

- The local geometry of the patches is modeled by weights that allow to reconstruct each example by a linear combination of its neighbors.
- Reconstruction errors are measured by an error function.

$$E(\mathbf{W}) = \sum_{i=1}^{N} \|\mathbf{x}_i - \sum_{j=1}^{N} w_{ij}\mathbf{x}_j\|_2^2 = \sum_{i=1}^{N} \|\mathbf{x}_i - \sum_{j\sim i} w_{ij}\mathbf{x}_j\|_2^2$$



- The w_{ij} weights measure the contribution of the j[th] example for the reconstruction of the i[th] example.
- w_{ij} weights are subject to two constraints
  - Each example is reconstructed from its neighbors
  - The rows of the matrix sum to one : $\sum_{j\sim i} w_{ij} = 1$
- These constraints ensure that, for each example, the weights are invariant by translation, rotation, rescaling, and so on.
- Weights can be obtained by optimization of a least squares criterion.

SECOND STEP

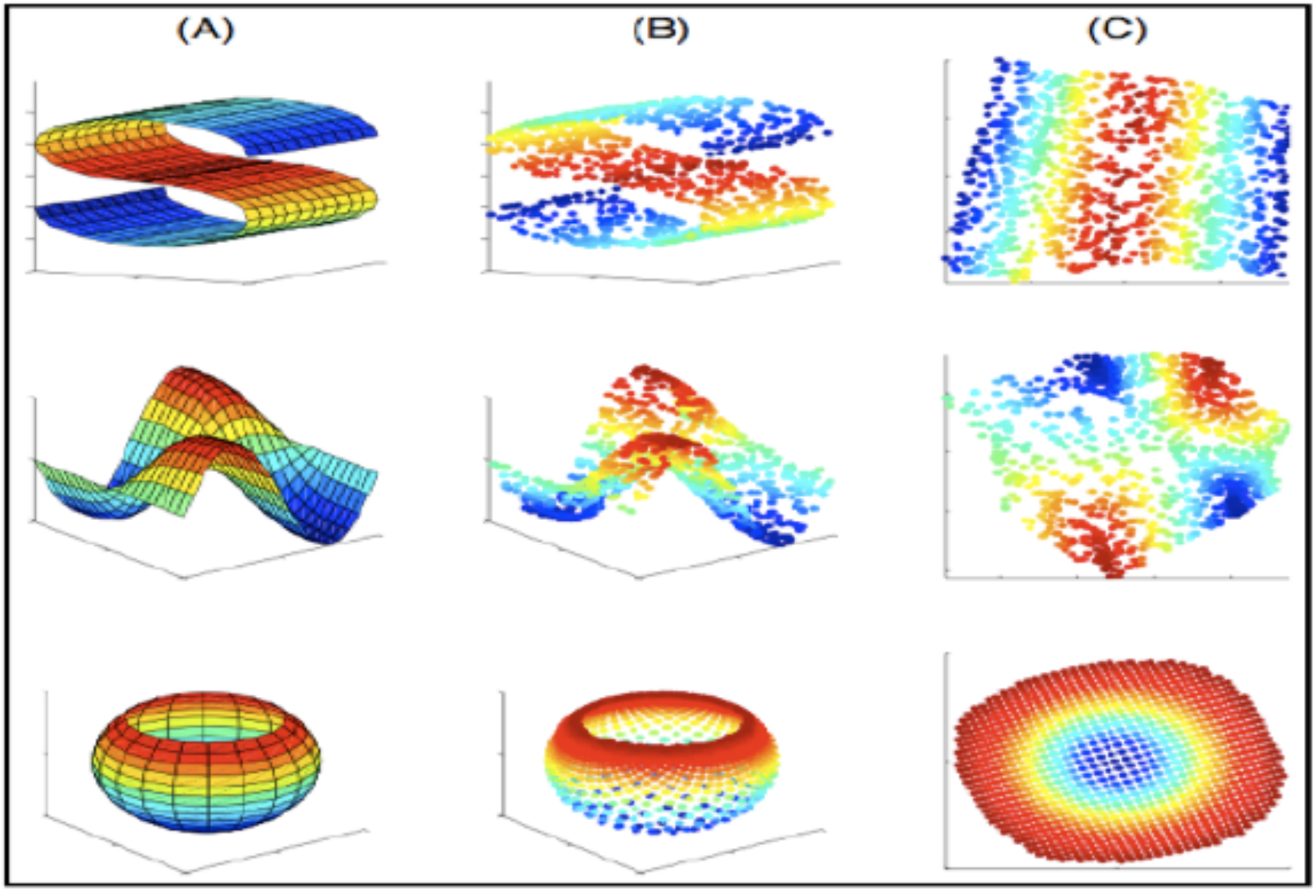
- The reconstruction weights reflect the properties of the local geometry, they must represent well the local patches of the variety.
- So we are looking for a projection Y of dimension d which minimizes the following cost function

$$E(\mathbf{Y}) = \sum_{i=1}^{N} \|\mathbf{y}_i - \sum_{j=1}^{N} w_{ij}\mathbf{y}_j\|_2^2$$

- This cost function is similar to the previous one, but this time the w_{ij} weights are fixed and we are looking for the $\mathbf{Y}_i$
- In order for this optimization problem to be well posed, we add two constraints
  - Since the $\mathbf{Y}_i$ can be translated without modifying the cost function, they are centered: $\bar{\mathbf{Y}} = 0$
  - The projection vectors must have a unitary covariance matrix $\sum_{i=1}^{N} \frac{1}{N}\mathbf{Y}_i\mathbf{Y}_i^T = \mathbf{I}$
- This minimization problem can be solved by retaining the d smallest eigenvectors of the matrix

$$(\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W})$$

EXAMPLES



INFLUENCE OF THE GRAPH CONSTRUCTION

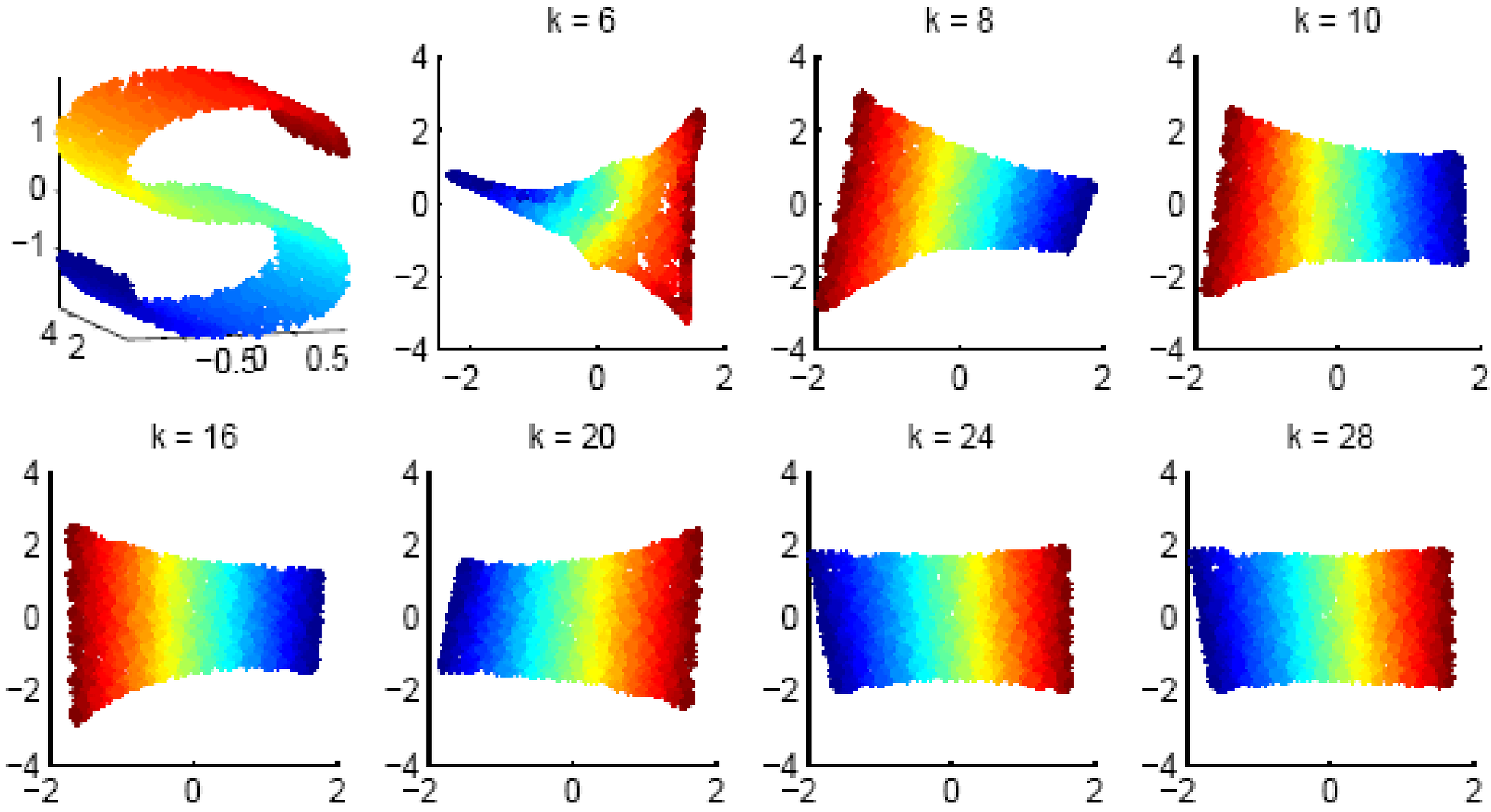


FIG. 5. *S-curve (top left) and computed 2D coordinates by LLE with various neighborhood size k.*

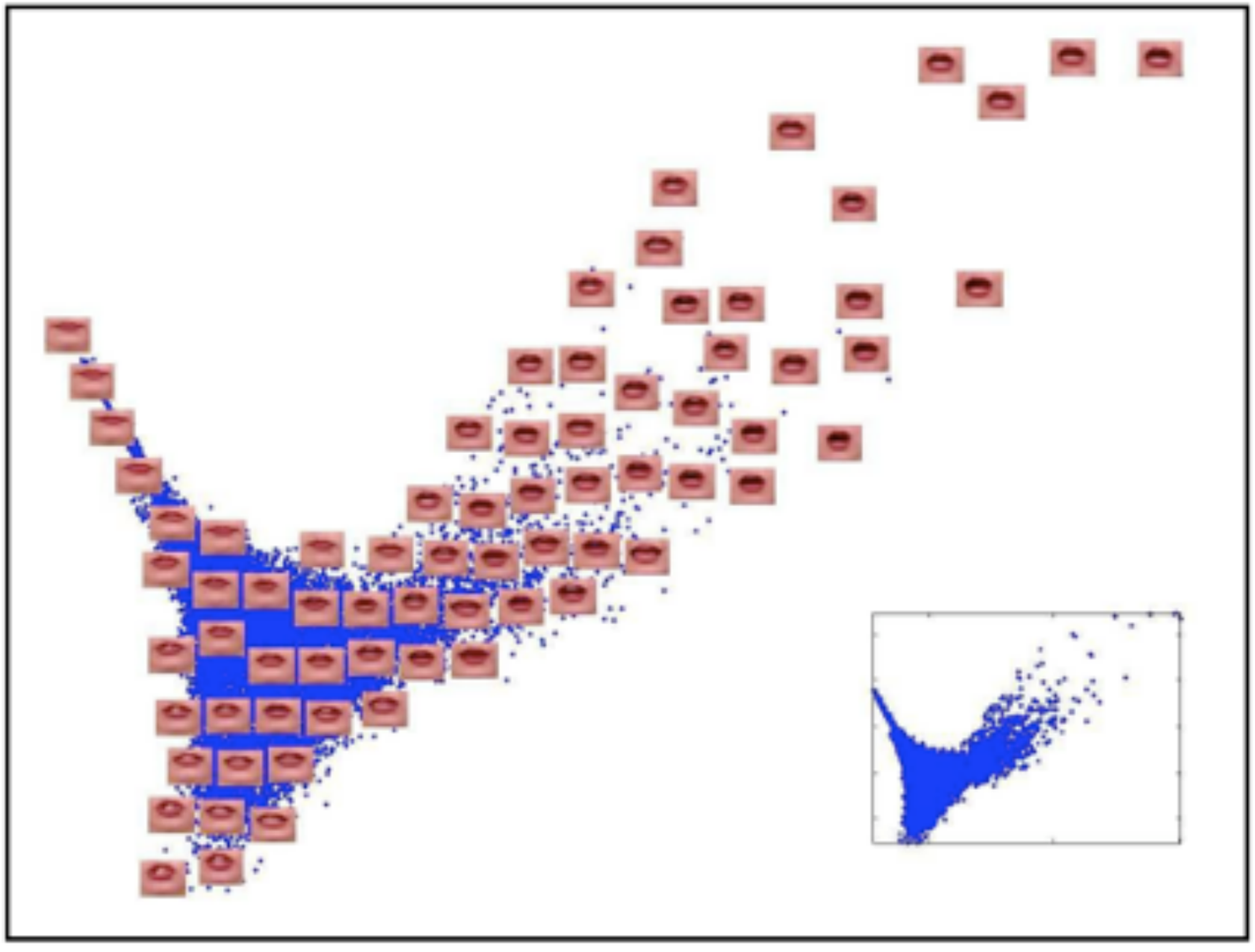
Lips

N=15960
images

K=24
neighbors

D=65664
pixels

d=2
(shown)

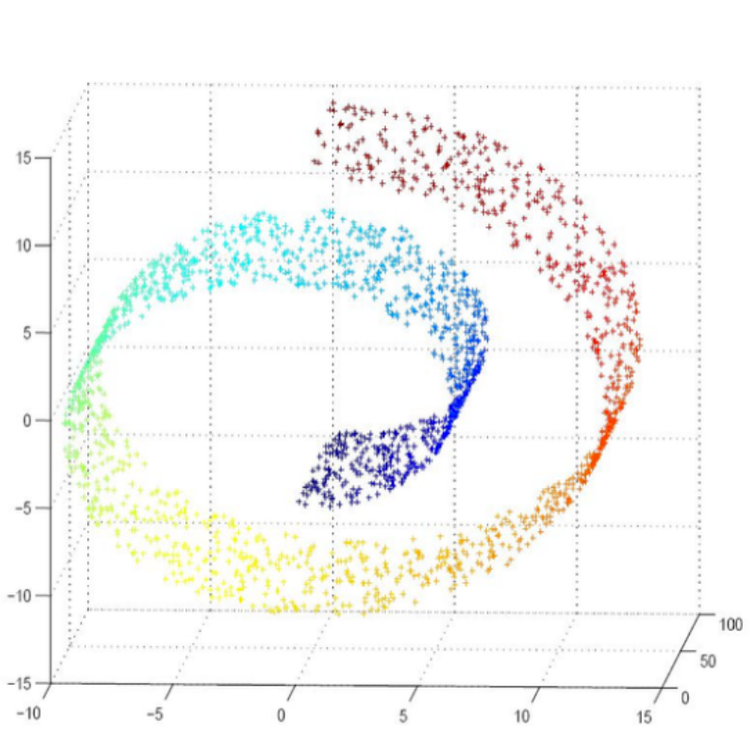


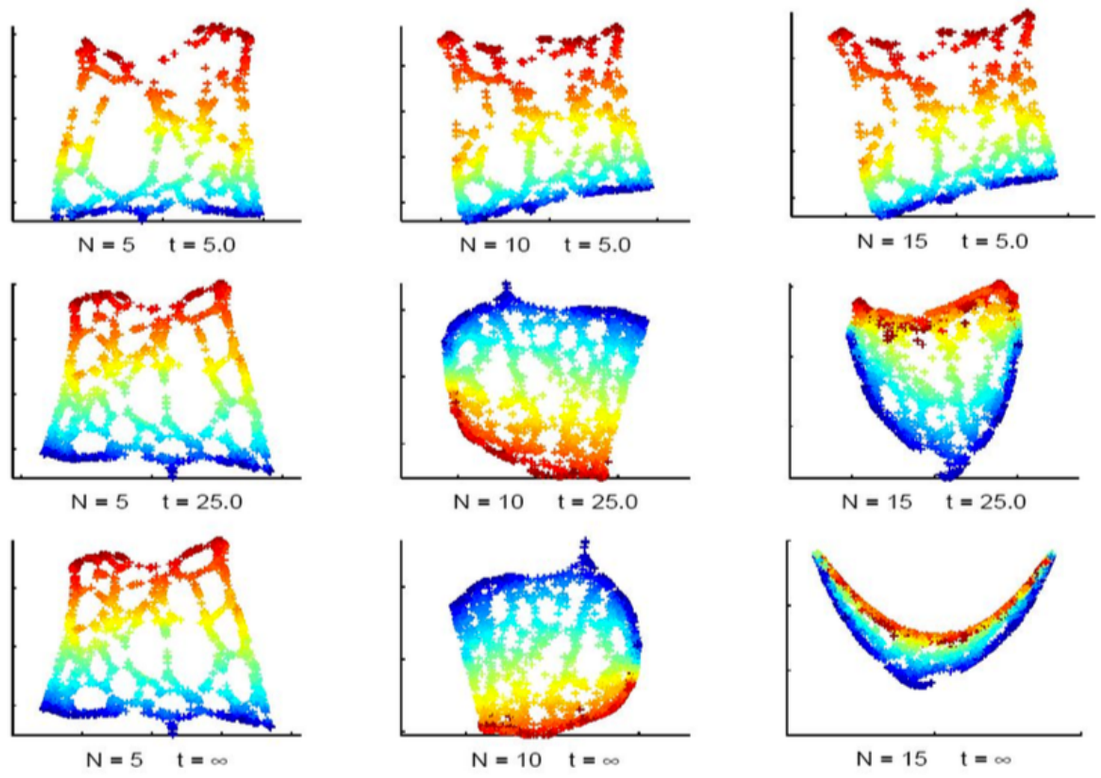
LEARNING OBJECTIVES

- Curse of dimensionality
- Dimensionality reduction
  - Linear methods
    - PCA
    - LDA
    - MDS
  - Non linear methods
    - ISOMAP
    - LLE
    - **Laplacian Eigenmaps**
    - Kernel PCA
    - t-SNE

LAPLACIAN EIGENMAPS

- Laplacian Eigenmaps intend to embed the data $\mathcal{X}$ in a d-dimensional in such a way that close/similar points in $\mathcal{X}$ remain close in the low dimensional space.
- Proximity (or similarity) is coded by the graph weights.





LAPLACIAN EIGENMAPS

- Overview of the method
  - Build a k-NN neighborhood graph
  - Assign weights w_{ij} to edges with a Gaussian kernel

$$w_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{t^2}}$$

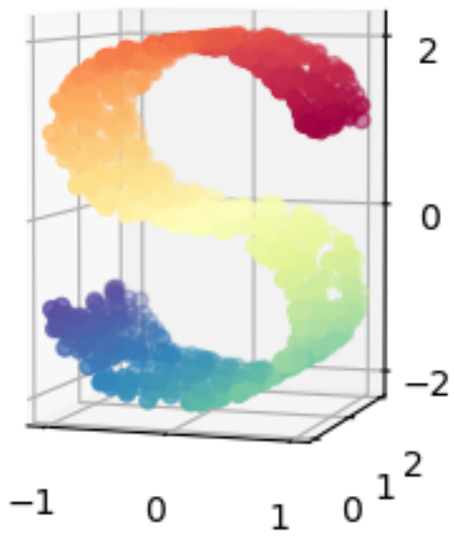
  - Find $\mathbf{Y} = \{\mathbf{y}_1, \cdots, \mathbf{y}_N\} \in \mathbb{R}^d$ that minimizes

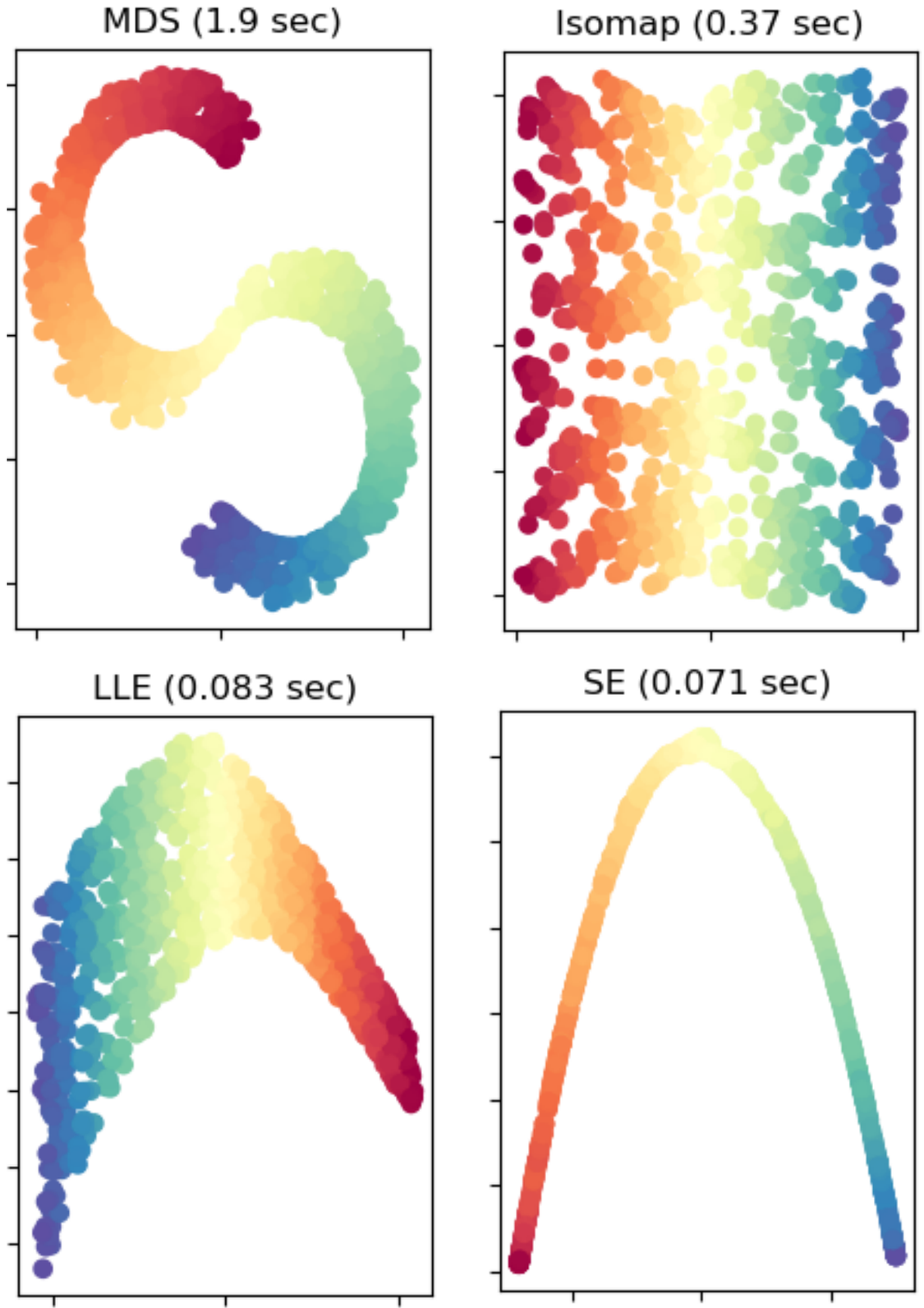
$$E(\mathbf{Y}) = \sum_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|_2^2 w_{ij}$$

with a scaling factor to avoid obvious solutions $\mathbf{Y}^T\mathbf{D}\mathbf{Y} = 1$

  - Since $E(\mathbf{Y}) = 2\mathbf{Y}^T\mathbf{L}\mathbf{Y}$ **the solution is the same than for the balanced cut graph clustering : eigenvectors of the Laplacian !**

EXAMPLE





LEARNING OBJECTIVES

- Curse of dimensionality
- Dimensionality reduction
  - Linear methods
    - PCA
    - LDA
    - MDS
  - Non linear methods
    - ISOMAP
    - LLE
    - Laplacian Eigenmaps
    - **Kernel PCA**
    - t-SNE

KERNEL PCA

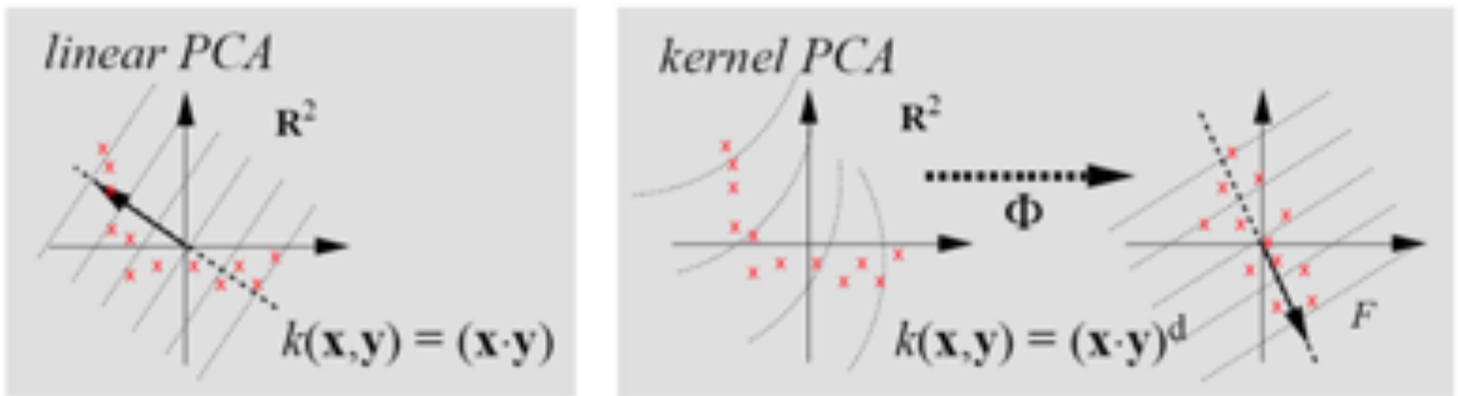
- We have seen that PCA allows to define a linear projection of the data.
- The dataset examples $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ are centered: $\bar{\mathbf{x}} = 0$
- We compute the covariance matrix

$$\mathbf{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T$$

- Compute eigenvectors and eigenvalues: $\mathbf{\Sigma v} = \lambda \mathbf{v}$
- The final projection is obtained by retaining only the d largest eigenvectors

$$\mathbf{y} = [\mathbf{v}_1\mathbf{v}_2 \cdots \mathbf{v}_d]\mathbf{x}$$

- The Kernel (Nonlinear) PCA consists in determining the projection in an implicit attribute space obtained by a kernel



KERNEL PCA

- To define the PCA Kernel
  - We need to project the data into a projection space of large dimension F $\Phi : \mathbb{R}^D \rightarrow F; \mathbf{x} \rightarrow \mathbf{X}$
  - We need to define the covariance matrix in this space

$$\mathbf{\Sigma}_F = \frac{1}{N} \sum_{i=1}^{N} \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^T$$

- Data are assumed to be centered $\overline{\phi(\mathbf{x})} = 0$
- The decomposition of the matrix is then computed

$$\mathbf{\Sigma}_F \mathbf{v} = \lambda \mathbf{v}$$

KERNEL PCA

- As we saw for the PCA, the eigenvectors can be expressed as linear combinations of the training data $\mathbf{\Sigma}_F \mathbf{v} = \left(\frac{1}{N} \sum_{i=1}^{N} \phi(\mathbf{x}_i) \phi(\mathbf{x}_i)^T \right) \mathbf{v} = \lambda \mathbf{v}$

$$\mathbf{v} = \left(\frac{1}{\lambda N} \sum_{i=1}^{N} \phi(\mathbf{x}_i) \phi(\mathbf{x}_i)^T \right) \mathbf{v} = \sum_{i=1}^{N} \frac{\phi(\mathbf{x}_i)^T \mathbf{v}}{\lambda N} \phi(\mathbf{x}_i) = \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i)$$

- We multiply by $\phi(\mathbf{x}_k)$ on both sides of $\mathbf{\Sigma}_F \mathbf{v} = \lambda \mathbf{v}$

$$\lambda [\phi(\mathbf{x}_k) \mathbf{v}] = [\phi(\mathbf{x}_k) \mathbf{\Sigma}_F \mathbf{v}]$$

- which, combining with the previous expression

$$\lambda \left[\phi(\mathbf{x}_k) \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i) \right] = \phi(\mathbf{x}_k) \left[\frac{1}{N} \sum_{j=1}^{N} \phi(\mathbf{x}_j) \phi(\mathbf{x}_j)^T \right] \left[\sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i) \right]$$

- and regrouping terms, yields

$$\lambda \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_k) \phi(\mathbf{x}_i) = \frac{1}{N} \sum_{i=1}^{N} \alpha_i \left(\phi(\mathbf{x}_k) \sum_{j=1}^{N} \phi(\mathbf{x}_j) \right) (\phi(\mathbf{x}_j) \phi(\mathbf{x}_i))$$

KERNEL PCA

- Defining an *N* x *N* matrix *K* as $\mathbf{K}_{ij} = (\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j))$
- The previous expression becomes $N\lambda\mathbf{K}\boldsymbol{\alpha} = \mathbf{K}^2\boldsymbol{\alpha}$
- Which can be solved through the eigenvalue problem

$$N\lambda\boldsymbol{\alpha} = \mathbf{K}\boldsymbol{\alpha}$$

- **Normalization:**
- To ensure that eigenvectors *V* are orthonormal, we then scale the eigenvectors α $(\mathbf{v}_i^k \mathbf{v}_i^k) = 1 \implies \left(\sum_{i=1}^{N} \alpha_i^k \phi(\mathbf{x}_i)\right)\left(\sum_{j=1}^{N} \alpha_j^k \phi(\mathbf{x}_j)\right) = 1$

$$\sum_{i,j=1}^{N} \alpha_i^k \alpha_j^k \phi(\mathbf{x}_i)\phi(\mathbf{x}_j) = 1 \implies \sum_{i,j=1}^{N} \alpha_i^k \alpha_j^k \mathbf{K}_{ij} = 1 \implies (\boldsymbol{\alpha}^k \mathbf{K} \boldsymbol{\alpha}^k) = 1$$

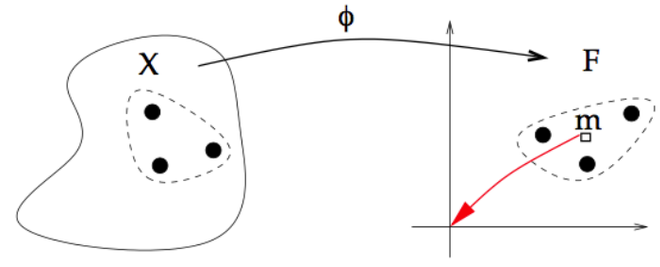
- which, since α are the eigenvectors of *K*, yields

$$\lambda_k(\boldsymbol{\alpha}^k \boldsymbol{\alpha}^k) = 1$$

CENTERING IN THE HIGH DIMENSIONAL SPACE F

- **Earlier we assumed that the data was centered in F**

$$\tilde{\phi}(\mathbf{x}_i) = \phi(\mathbf{x}_i) - \frac{1}{N}\sum_{i=1}^{N}\phi(\mathbf{x}_i)$$



- So the covariance matrix in this centered space is $\tilde{\mathbf{K}}_{ij} = \left(\tilde{\phi}(\mathbf{x}_i)\cdot\tilde{\phi}(\mathbf{x}_j)\right)$
- And the eigenvalue problem that we need to solve is $\tilde{\lambda}\tilde{\boldsymbol{\alpha}} = \tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}}$
- Merging the first expression into the second one

$$\begin{aligned}\tilde{\mathbf{K}}_{ij} &= \left[\left(\phi(\mathbf{x}_i) - \frac{1}{N}\sum_{k=1}^{N}\phi(\mathbf{x}_k)\right)\left(\phi(\mathbf{x}_j) - \frac{1}{N}\sum_{l=1}^{N}\phi(\mathbf{x}_l)\right)\right] \\ &= \mathbf{K}_{ij} - \frac{1}{N}\sum_{k=1}^{N}\mathbf{1}_{ik}\mathbf{K}_{ik} - \frac{1}{N}\sum_{l=1}^{N}\mathbf{1}_{lj}\mathbf{K}_{lj} + \frac{1}{N^2}\sum_{k=1}^{N}\mathbf{1}_{ik}\mathbf{K}_{kl}\mathbf{1}_{lj} \\ &= [\mathbf{K} - \mathbf{1}_N\mathbf{K} - \mathbf{K}\mathbf{1}_N + \mathbf{1}_N\mathbf{K}\mathbf{1}_N]_{ij}\end{aligned}$$

with $\mathbf{1}_{ij} = 1 \quad \forall i,j$ *and* $(\mathbf{1}_N)_{ij} = \frac{1}{N}$

- **The centered kernel matrix can be computed from the uncentered one**

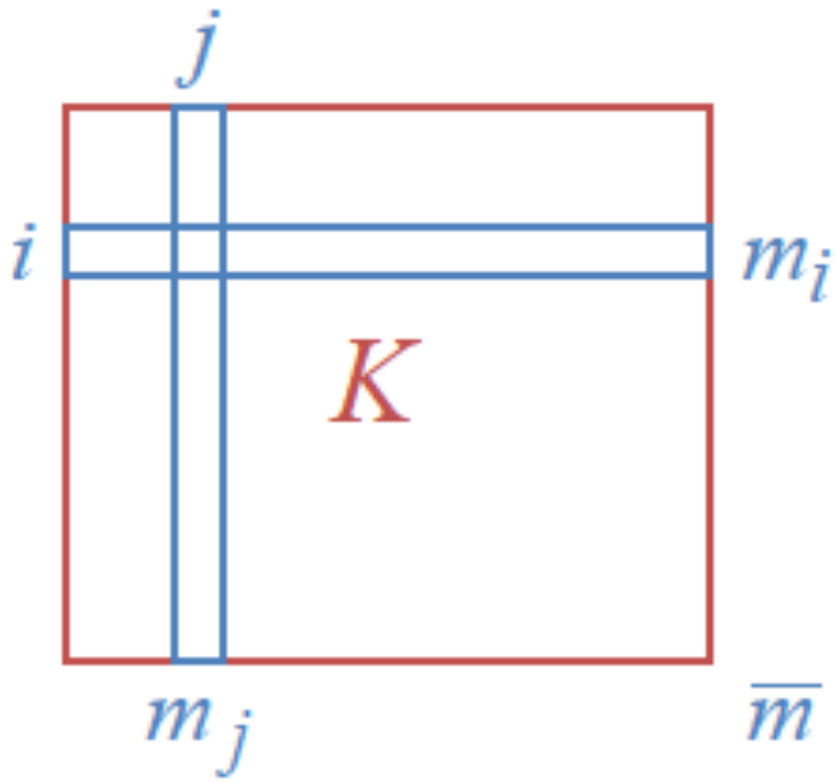
INTERPRETATION

$$\tilde{K}_{ij} = K_{ij} - m_i - m_j + \overline{m}$$

mean of i^{th} row

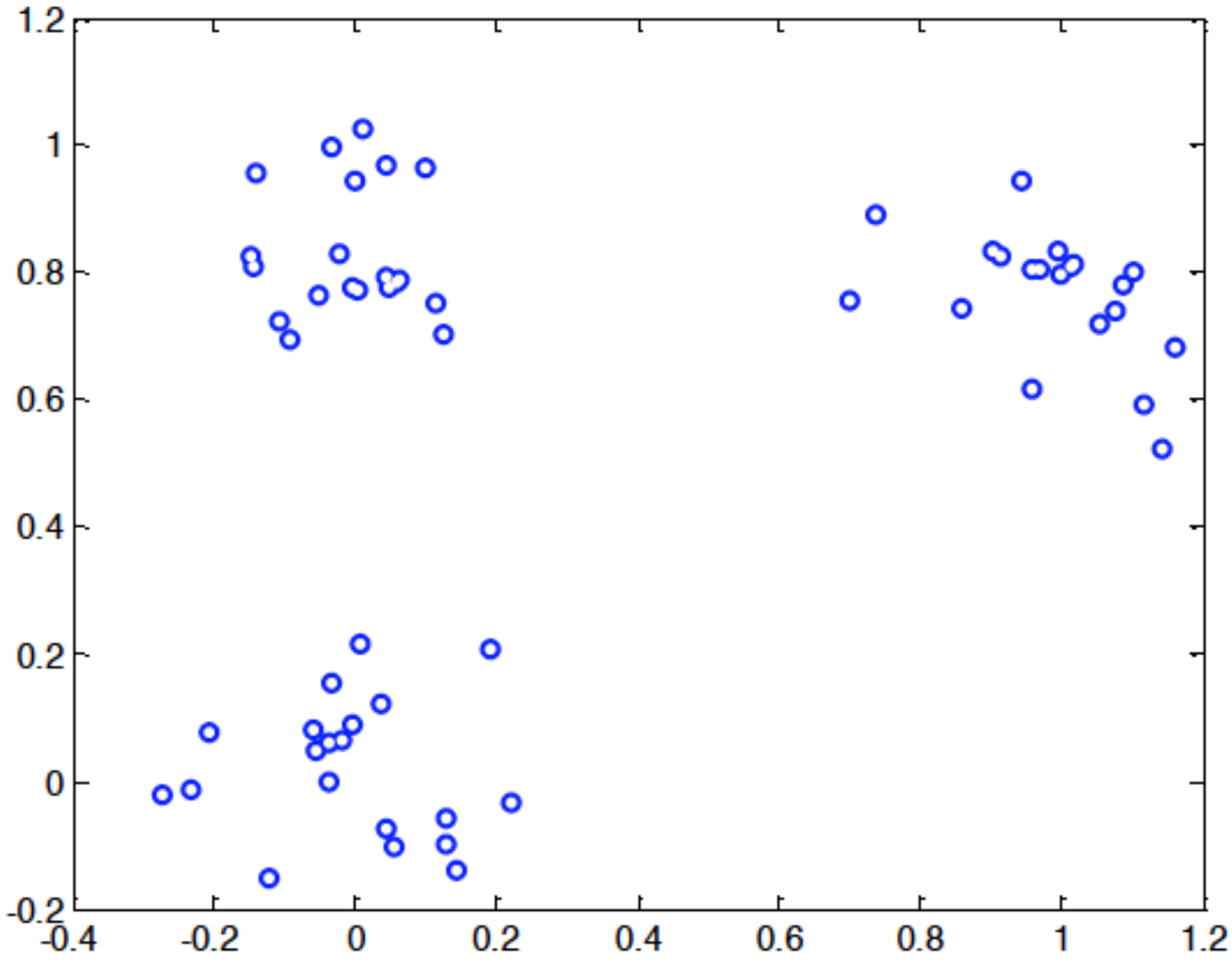
mean of j^{th} col

mean of all entries in K

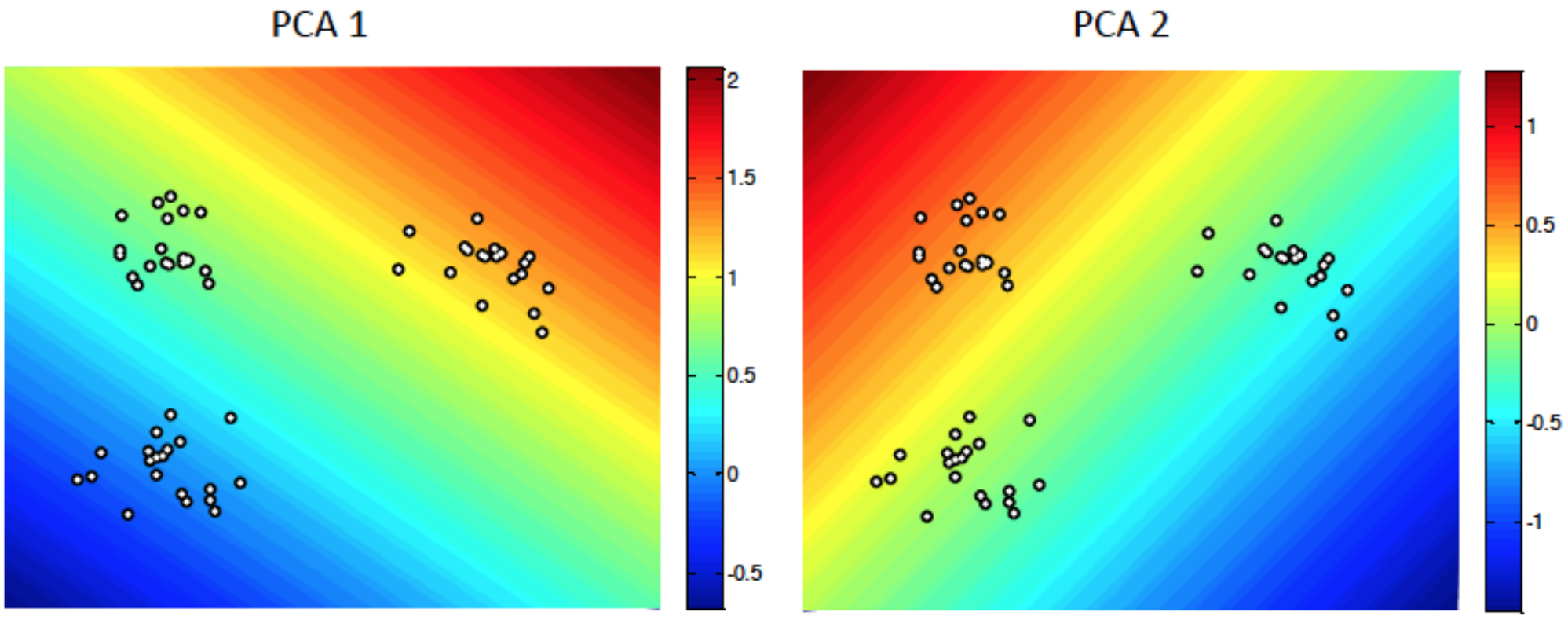


KERNEL PCA EXAMPLE

- Simple dataset with three modes, 20 samples per mode

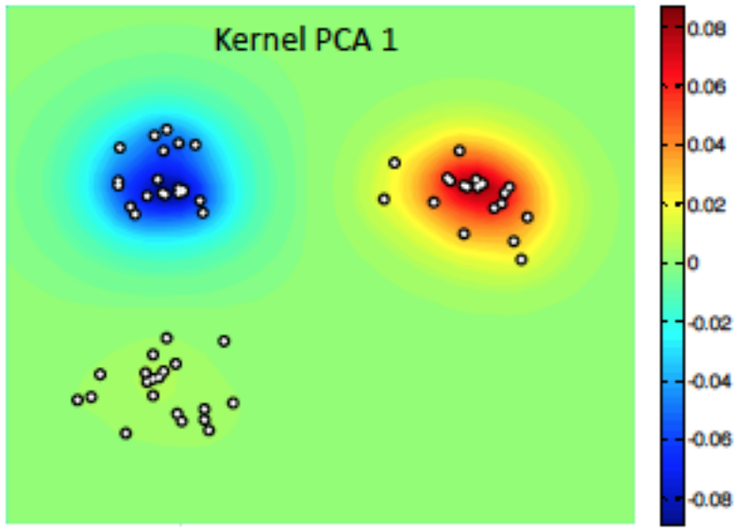


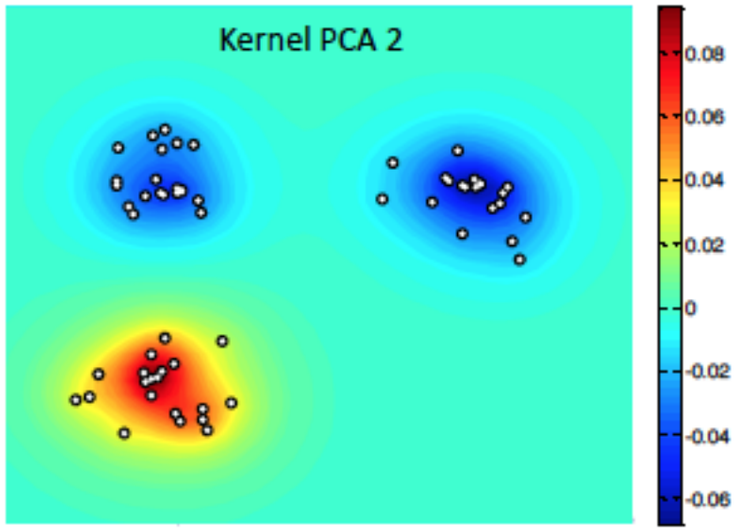
PCA RESULT

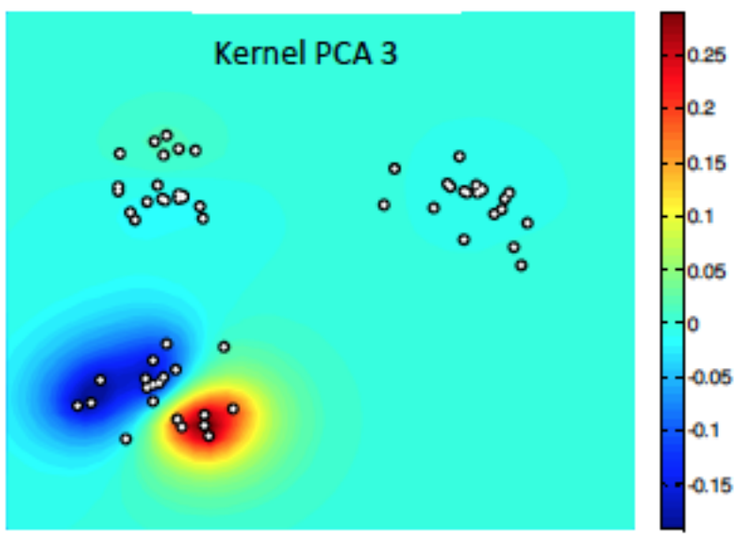


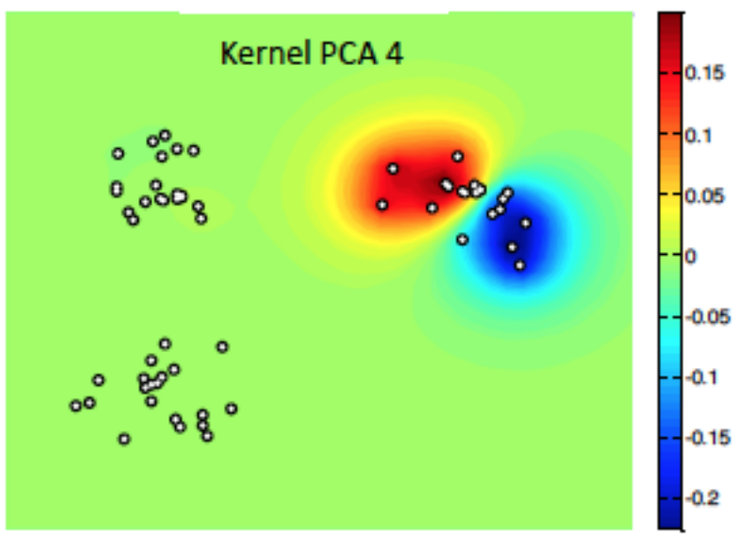
KPCA RESULT

- with a Gaussian kernel

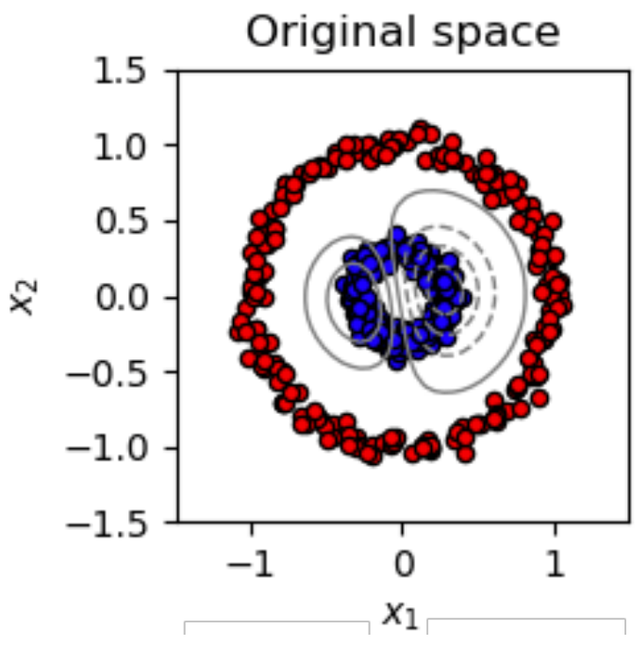


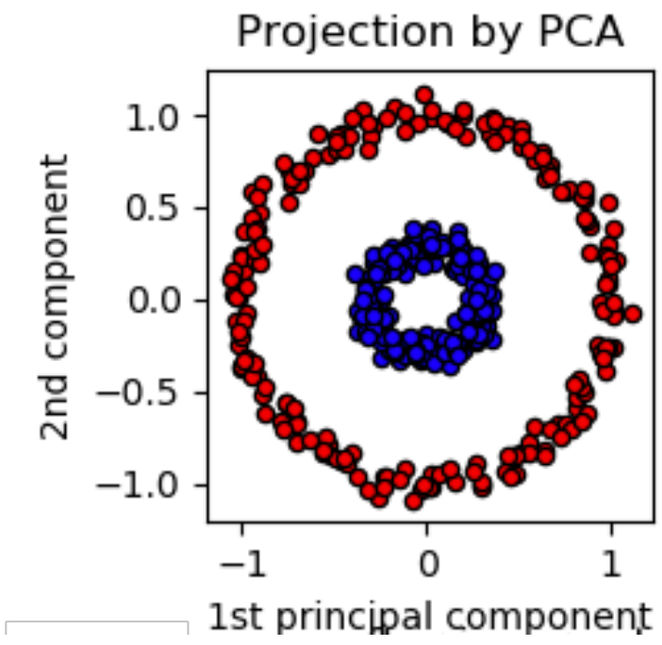


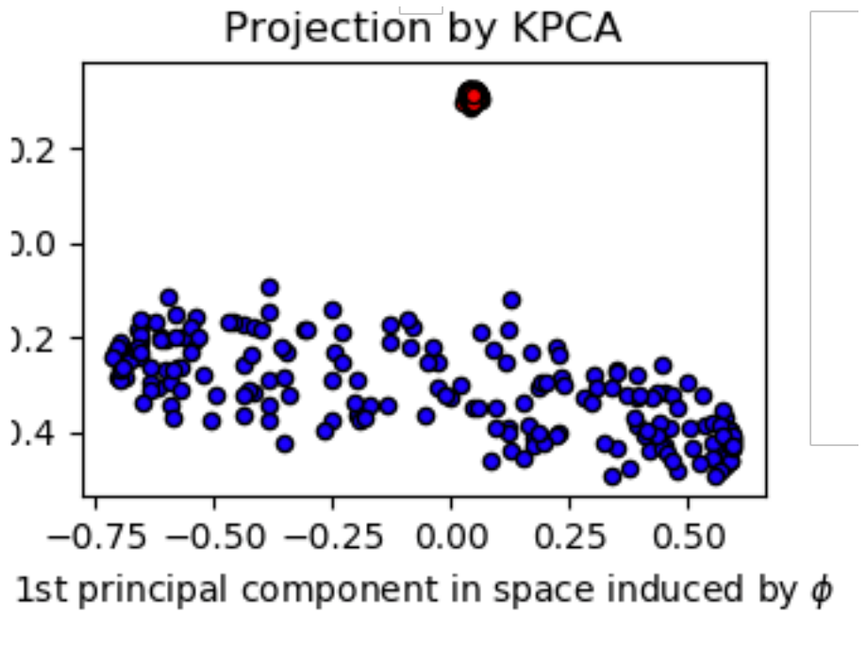




KPCA VS PCA


Original space
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1
0
1
x_1
x_2


Projection by PCA
1.0
0.5
0.0
−0.5
−1.0
−1
0
1
1st principal component
2nd component


Projection by KPCA
−0.75
−0.50
−0.25
0.00
0.25
0.50
1st principal component in space induced by ϕ

LET'S KERNELIZE THE LDA !

- For classical LDA we had to maximize $J(\mathbf{w}) = \dfrac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$
- A kerneled version is $J(\mathbf{w}) = \dfrac{\mathbf{w}^T \mathbf{S}_B^\phi \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W^\phi \mathbf{w}}$

$$\textit{with} \quad \mathbf{S}_W^\phi = \sum_{i=1}^{2} \sum_{\mathbf{x}_i \in C_i} (\phi(\mathbf{x}) - \boldsymbol{\mu}_i^\phi)(\phi(\mathbf{x}) - \boldsymbol{\mu}_i^\phi)^T \quad \textit{and} \quad \mathbf{S}_B^\phi = (\boldsymbol{\mu}_1^\phi - \boldsymbol{\mu}_2^\phi)(\boldsymbol{\mu}_1^\phi - \boldsymbol{\mu}_2^\phi)^T$$

$$\textit{and} \quad \boldsymbol{\mu}_i^\phi = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \phi(\mathbf{x})$$

- The eigenvector is a linear combination of the examples

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i)$$

- Then

$$\mathbf{w}^T \boldsymbol{\mu}_i^\phi = \left(\sum_{j=1}^{N} \alpha_j \phi(\mathbf{x}_j) \right)^T \left(\frac{1}{|C_i|} \sum_{\mathbf{x}_k \in C_i} \phi(\mathbf{x}_k) \right) = \frac{1}{|C_i|} \sum_{j=1}^{N} \sum_{\mathbf{x}_k \in C_i} \alpha_j \mathbf{K}(\mathbf{x}_j, \mathbf{x}_k) = \boldsymbol{\alpha}^T \mathbf{M}_i$$

$$\textit{with} \quad (\mathbf{M}_i)_j = \frac{1}{|C_i|} \sum_{\mathbf{x}_k \in C_i} \mathbf{K}(\mathbf{x}_j, \mathbf{x}_k)$$

LET'S KERNELIZE THE LDA !

- Putting all together

$$\mathbf{w}^T \mathbf{S}_B^\phi \mathbf{w} = \boldsymbol{\alpha}^T \mathbf{M} \boldsymbol{\alpha} \quad \textit{with} \quad \mathbf{M} = (\mathbf{M}_1 - \mathbf{M}_2)(\mathbf{M}_1 - \mathbf{M}_2)^T$$

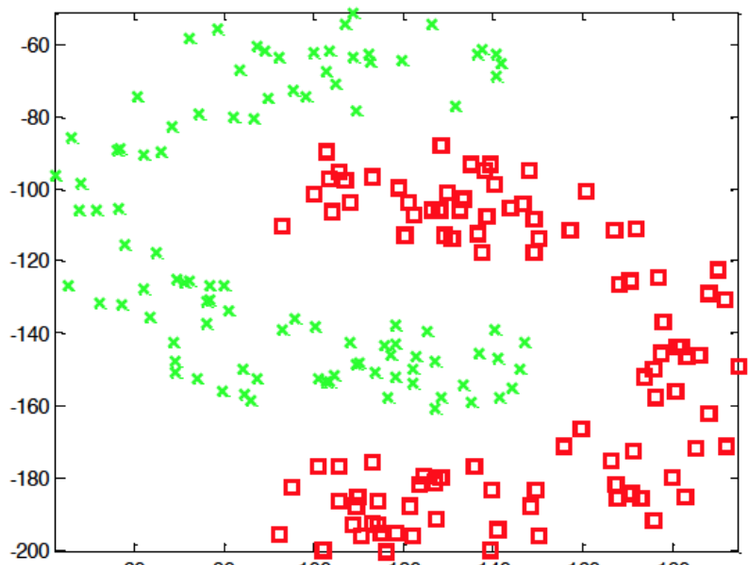
$$\mathbf{w}^T \mathbf{S}_W^\phi \mathbf{w} = \boldsymbol{\alpha}^T \mathbf{N} \boldsymbol{\alpha} \quad \textit{with} \quad \mathbf{N} = \sum_{j=1}^{2} \mathbf{K}_j (\mathbf{I} - \mathbf{1}_{|C_j|} \mathbf{K}_j^T)$$

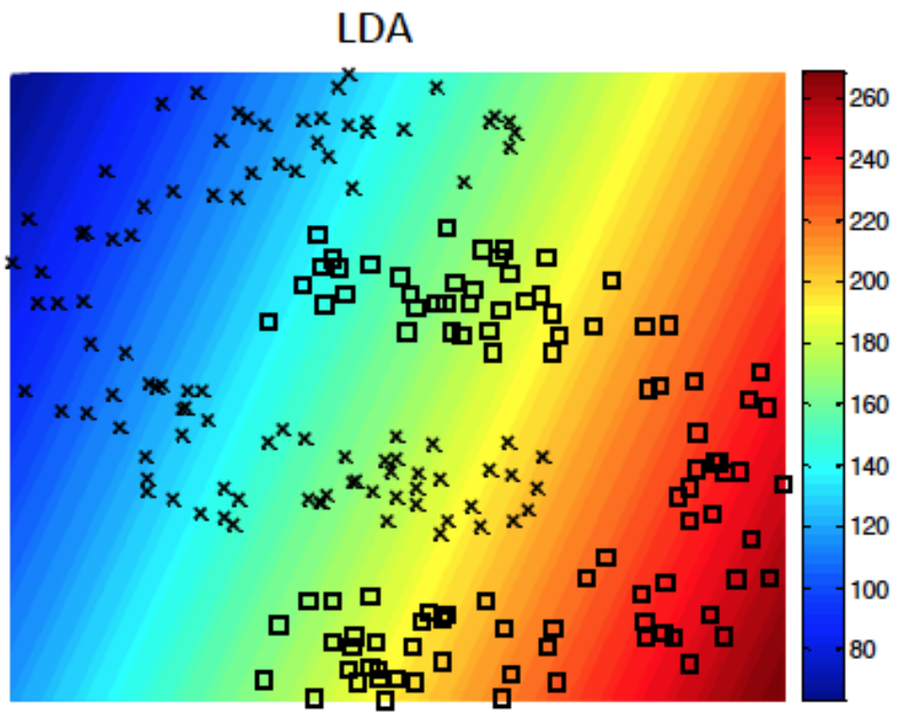
$$\textit{and} \qquad (\mathbf{K}_j)_{nm} = K(\mathbf{x}_n, \mathbf{x}_m) \quad \text{with } \mathbf{x}_m \in C_j$$

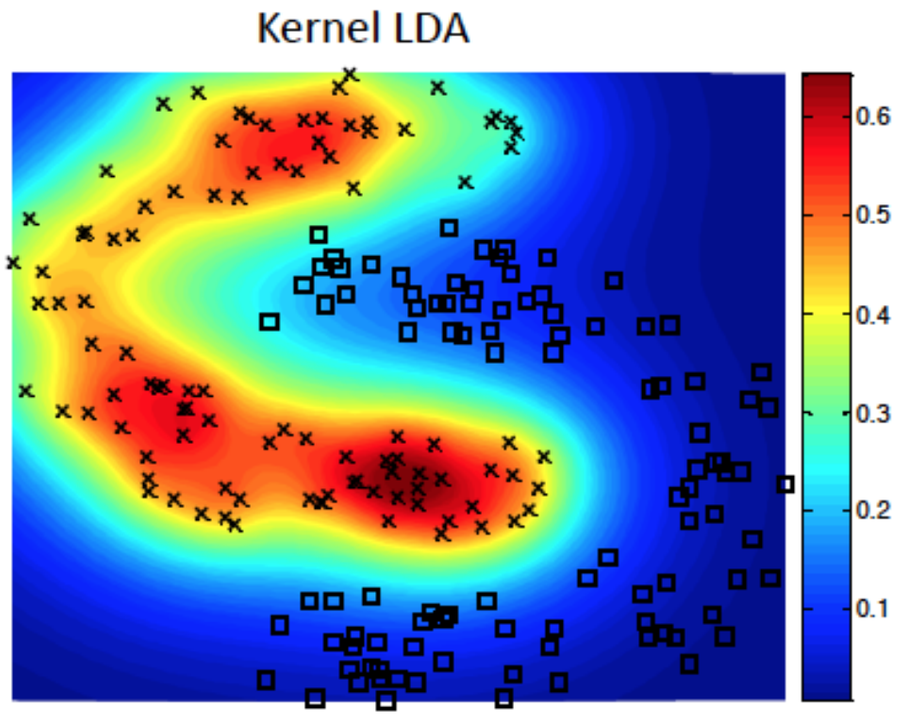
$$\textit{which gives} \qquad J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B^\phi \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W^\phi \mathbf{w}} = \frac{\boldsymbol{\alpha}^T \mathbf{M} \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T \mathbf{N} \boldsymbol{\alpha}}$$

The solution is given by the leading eigenvector of $\mathbf{N}^{-1}\mathbf{M}$

KERNEL LDA EXAMPLE







LEARNING OBJECTIVES

- Curse of dimensionality
- Dimensionality reduction
  - Linear methods
    - PCA
    - LDA
    - MDS
  - Non linear methods
    - ISOMAP
    - LLE
    - Laplacian Eigenmaps
    - Kernel PCA
    - **t-SNE**

T-SNE

- It approximates the distribution of the pairwise distances in the original space by a t-distribution (Student's)
- Converts distances to similarities that can be interpreted as probabilities

$$p_{j|i} = \frac{e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}}}{\sum_{k \neq i} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_k\|^2}{2\sigma_i^2}}} \qquad q_{j|i} = \frac{e^{-\frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{2\sigma_i^2}}}{\sum_{k \neq i} e^{-\frac{\|\mathbf{y}_i - \mathbf{y}_k\|^2}{2\sigma_i^2}}}$$

$$p_{i|i} = 0, q_{i|i} = 0$$

DISTANCE BETWEEN DISTRIBUTIONS

- $P_i = \{p_{1|i}, p_{2|i}, \cdots, p_{N|i}\}$ and $Q_i = \{q_{1|i}, q_{2|i}, \cdots, q_{N|i}\}$ are the distributions on the neighbors of $\mathbf{x}_i$
- The Kullback-Leibler Divergence (KL) compares the two distributions

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

- that we want to minimize (derivative to 0)

$$\frac{dC}{dy_i} = 2 \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

T-SNE

- T-SNE simplifies the problem by symmetrizing and simplifying the computations

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N} \qquad q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j|^2)^{-1}}{\sum_{k \neq i}(1 + \|\mathbf{y}_i - \mathbf{y}_k|^2)^{-1}}$$

- The cost becomes $C = KL(P||Q)$

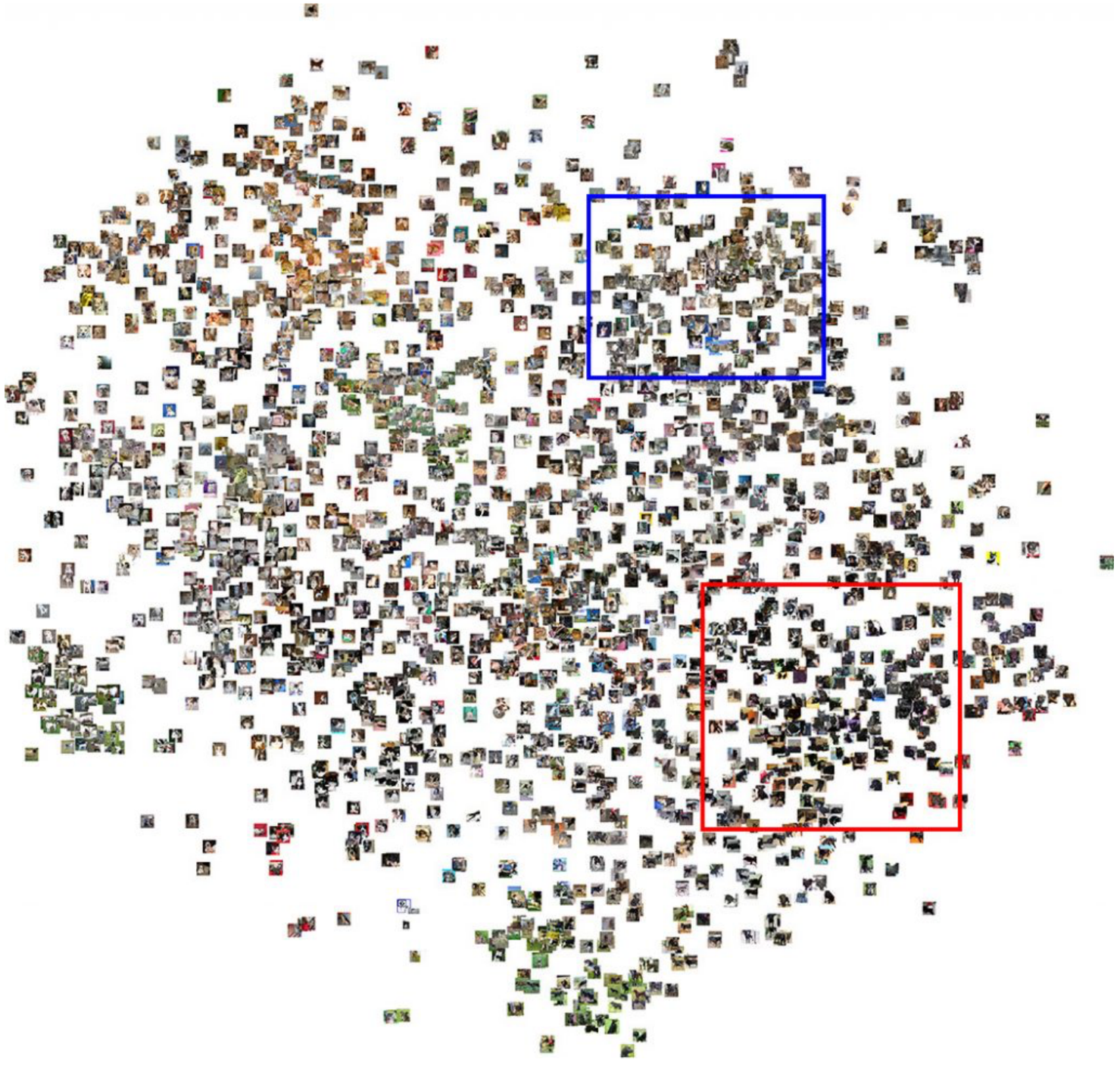
$$\frac{dC}{dy_i} = 4 \sum_j (p_{iji} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1}$$

- It is solved by gradient descent

$$\mathbf{y}^t = \mathbf{y}^{t-1} + \nu(t) \frac{dC}{d\mathbf{y}}$$

T-SNE EXAMPLES

- On the Image Net Google dataset



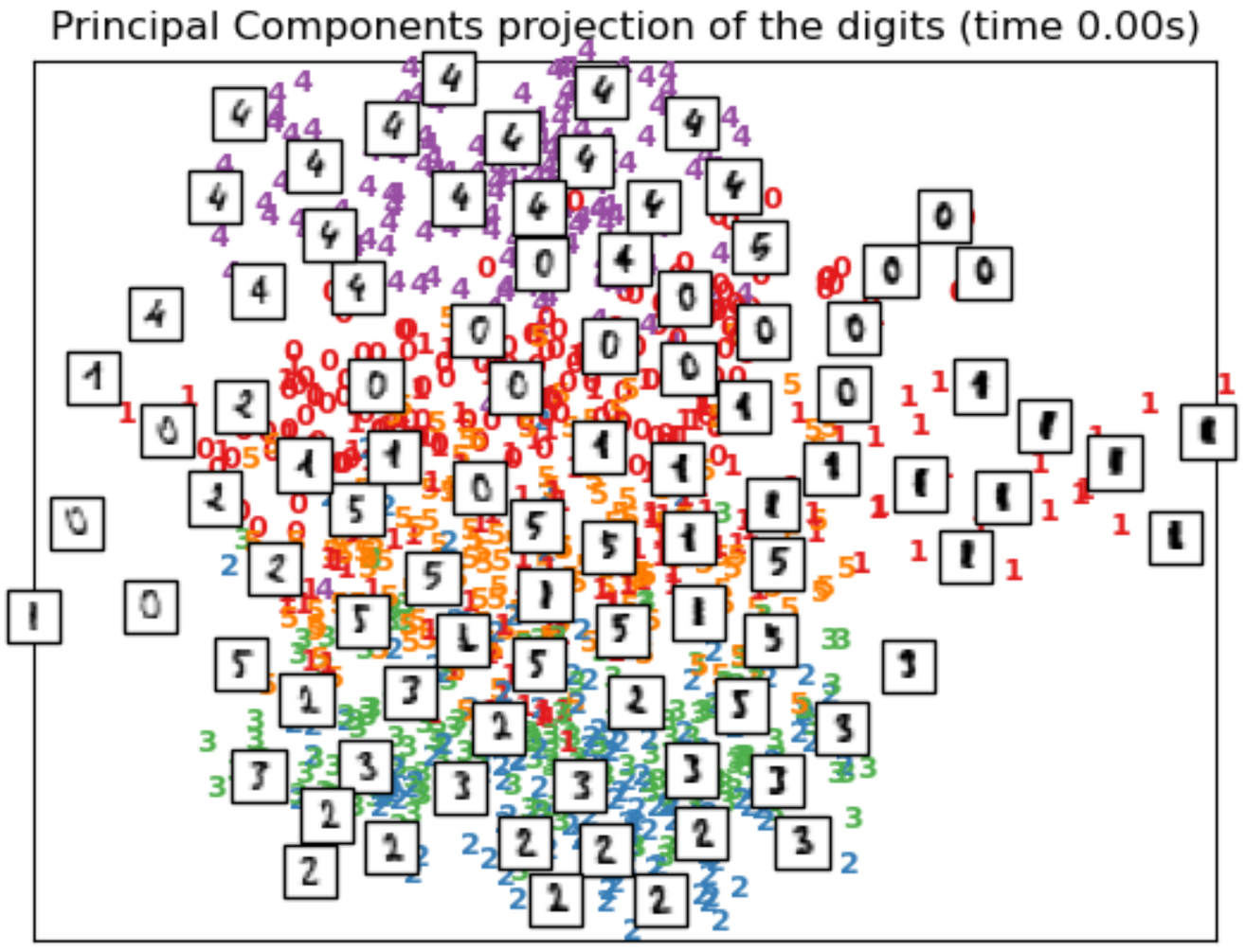
Gray cats

Black cats

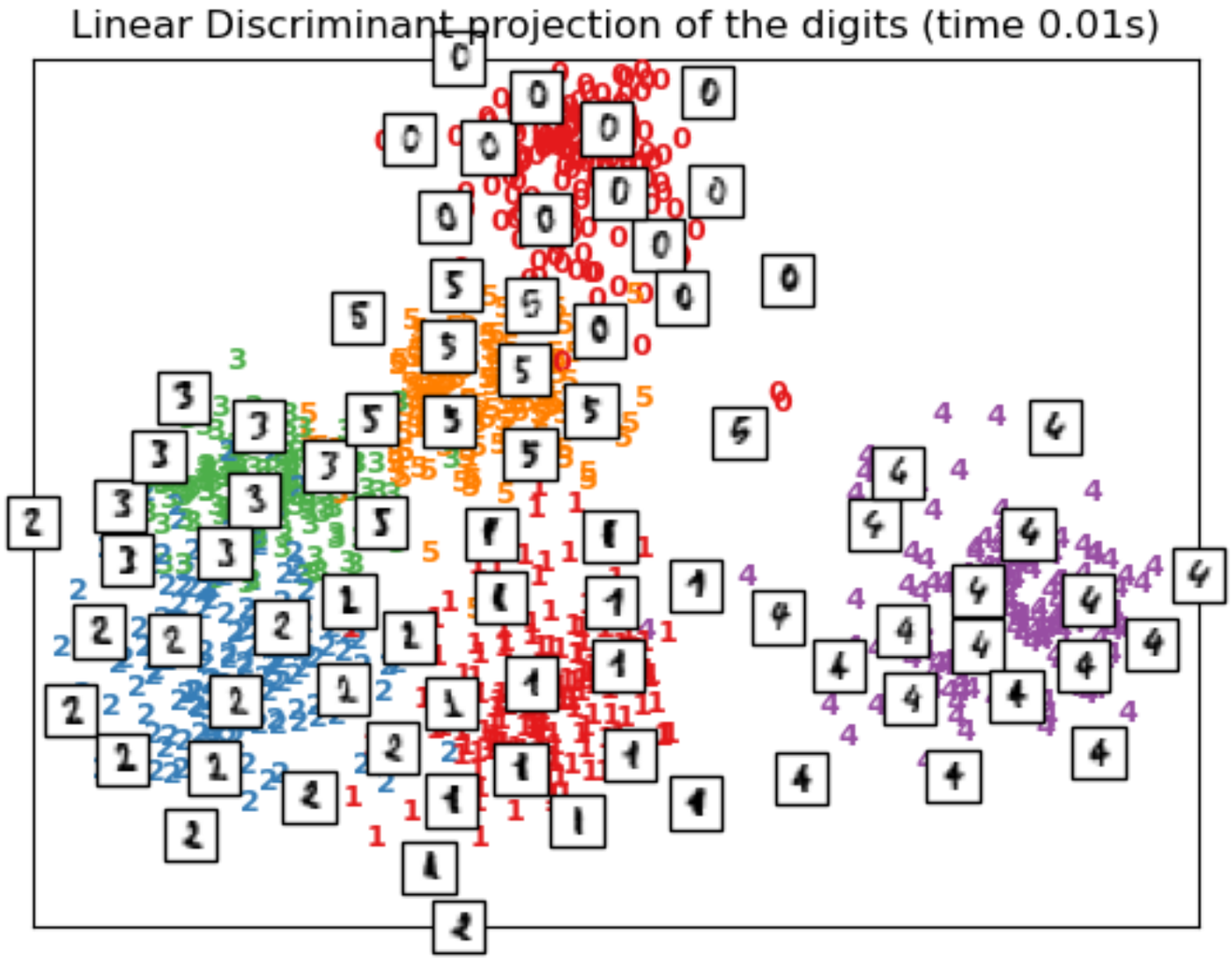
FINAL COMPARISONS FROM SCIKIT-LEARN

A selection from the 64-dimensional digits dataset

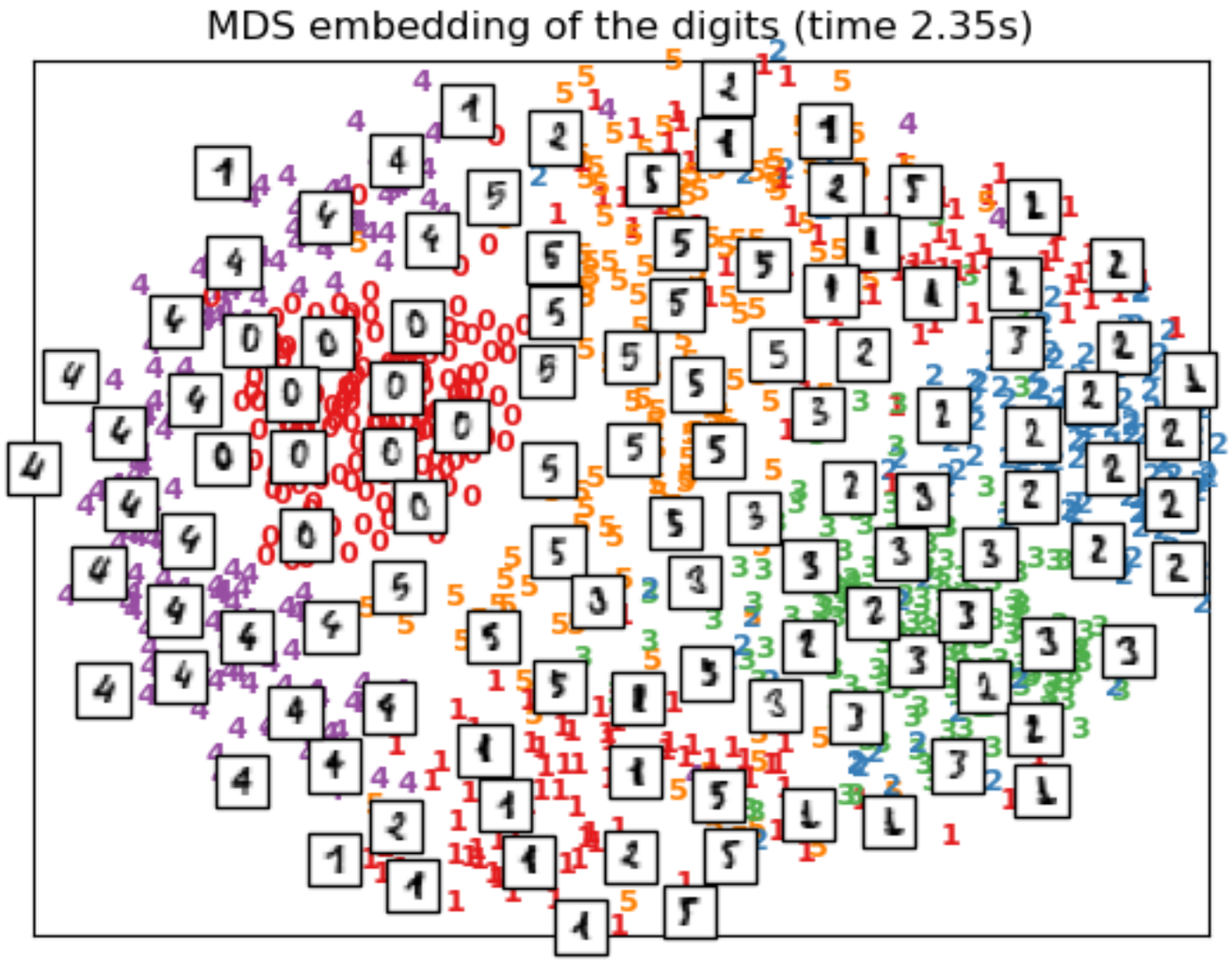
PCA



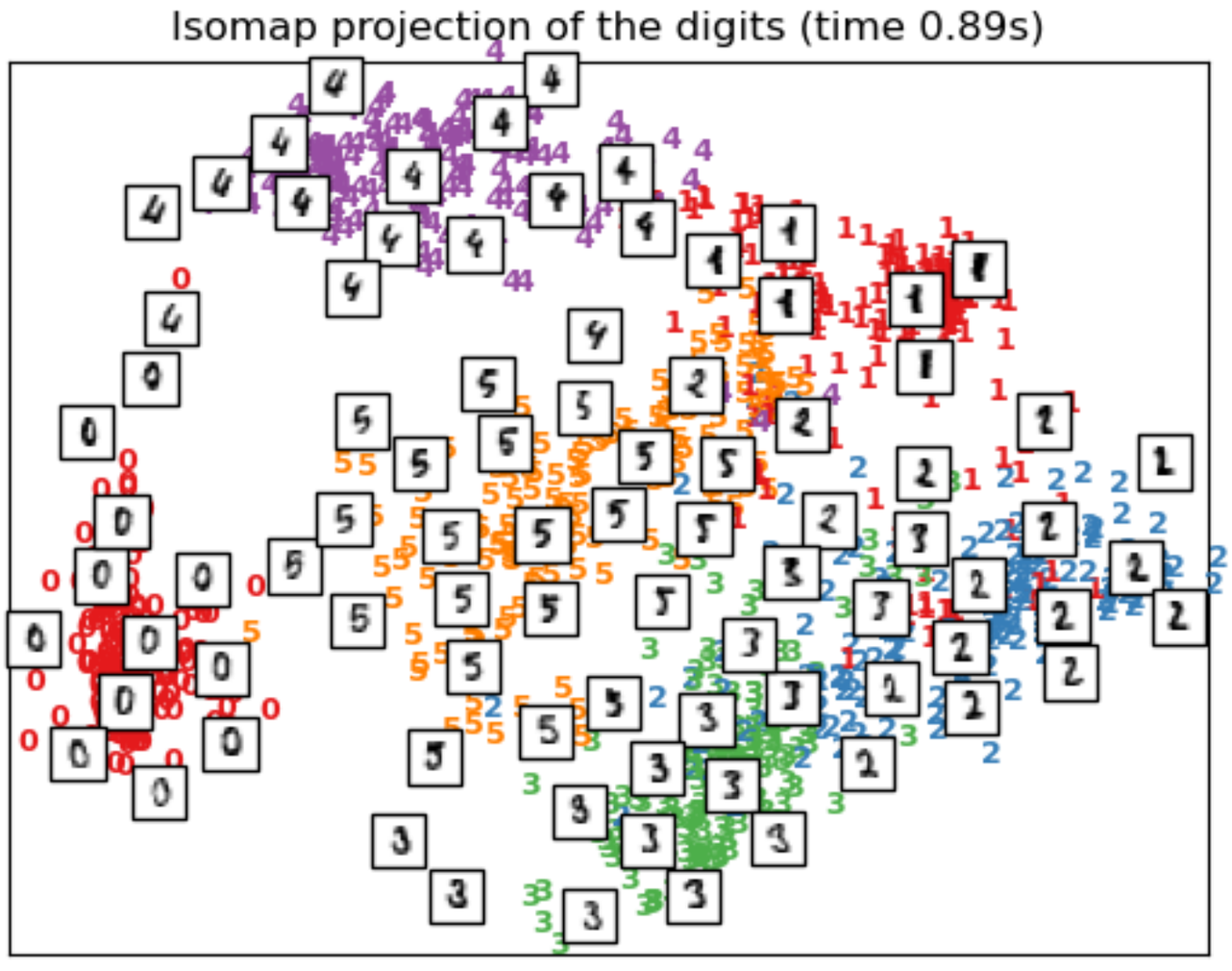
LDA



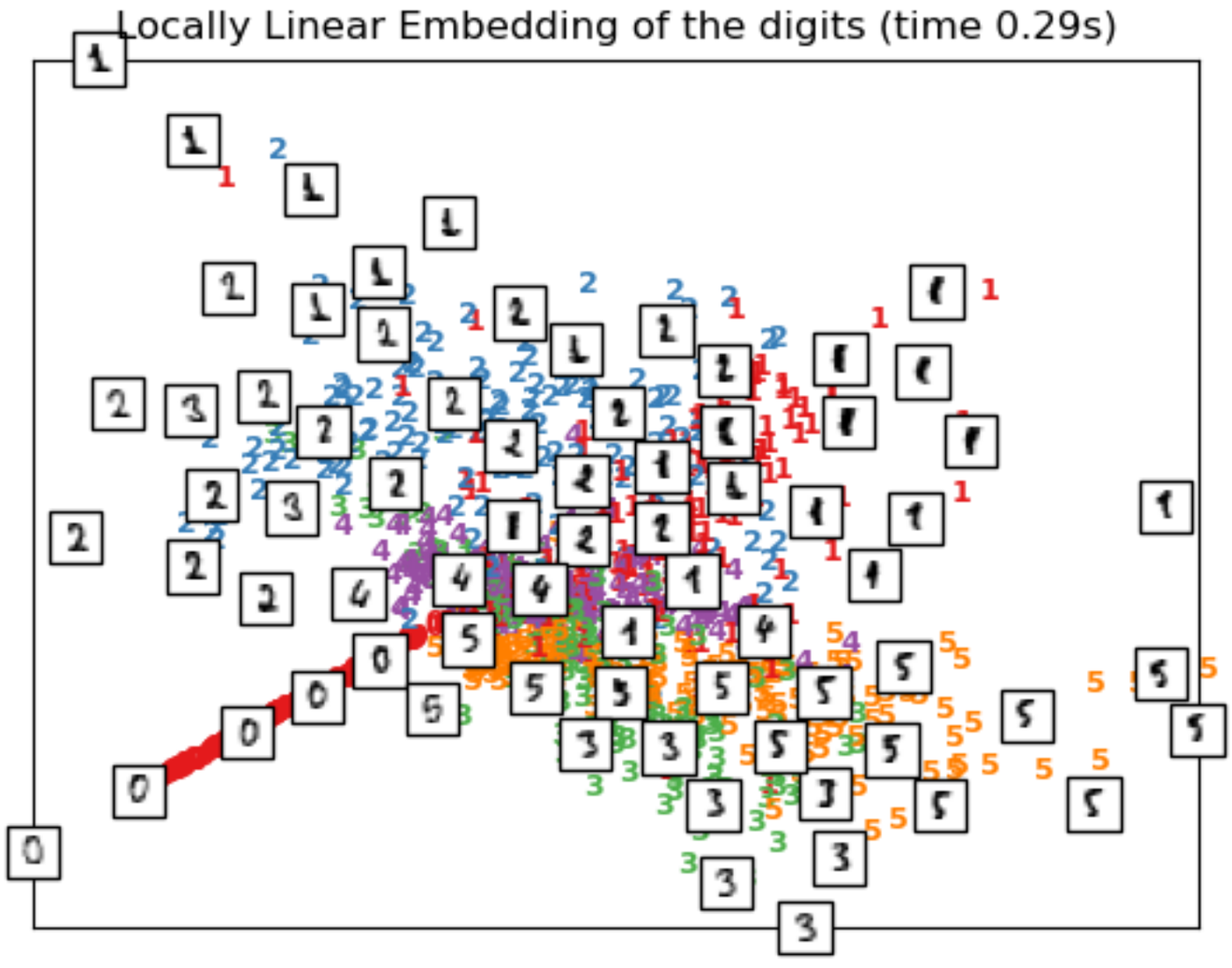
MDS



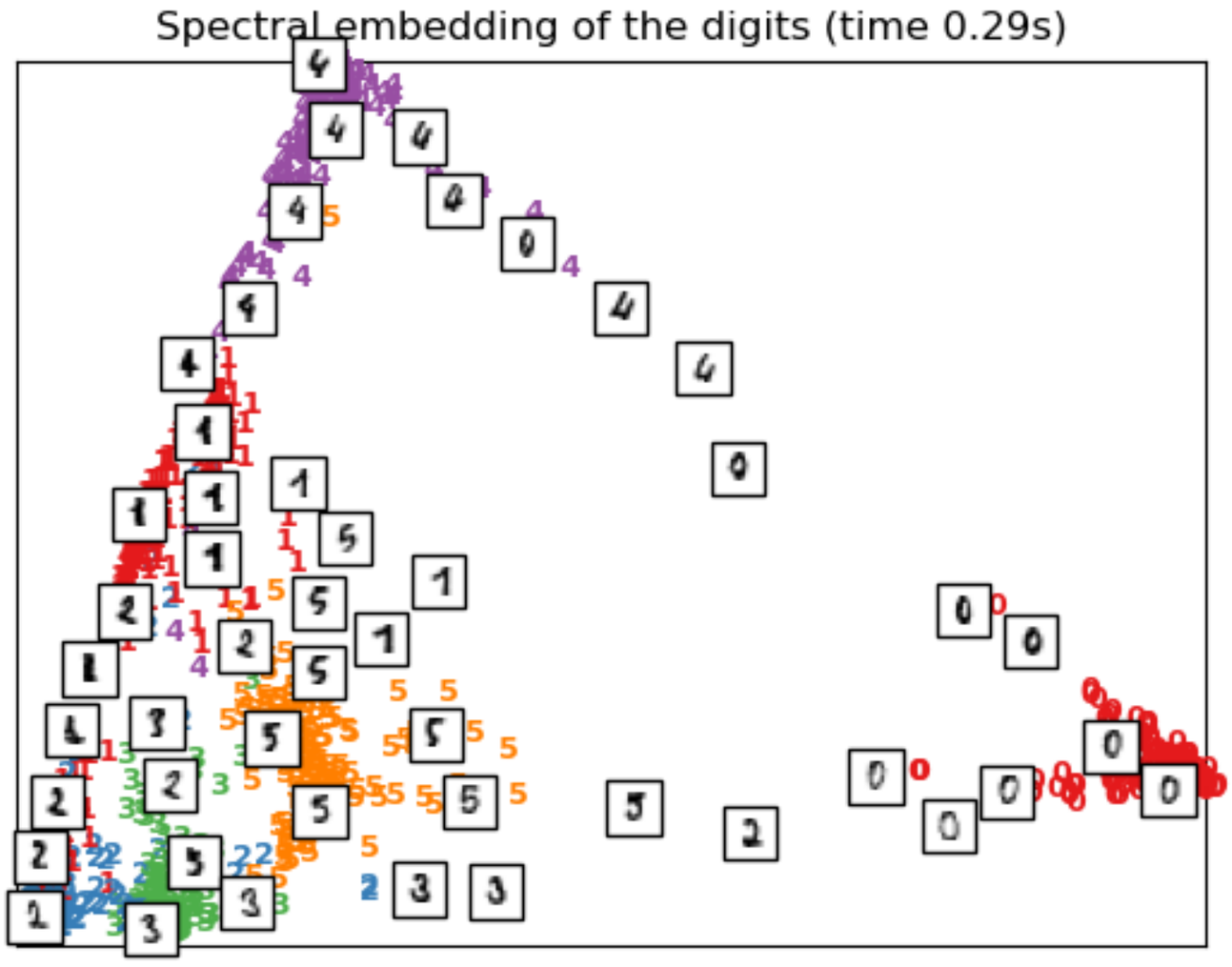
ISOMAP



LLE



LE



T-SNE