Dimensionality Reduction

Minería de Datos (M1966)

Steven Van Vaerenbergh
Depto. de Matemáticas, Estadística y Computación
Universidad de Cantabria

14 de diciembre de 2021

Maśter Universitario Oficial **Data Science**

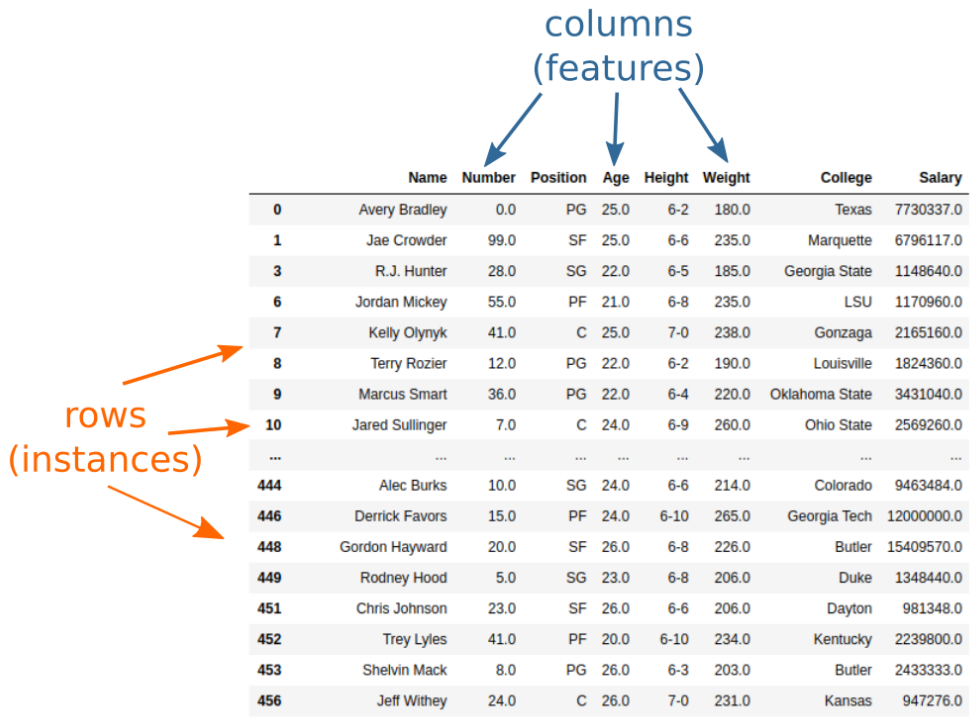
con el apoyo del

Contents

- **Part 1. Linear techniques**: PCA.
- **Part 2. Nonlinear techniques**: MDS, Isomap, LLE, t-SNE.

Data format

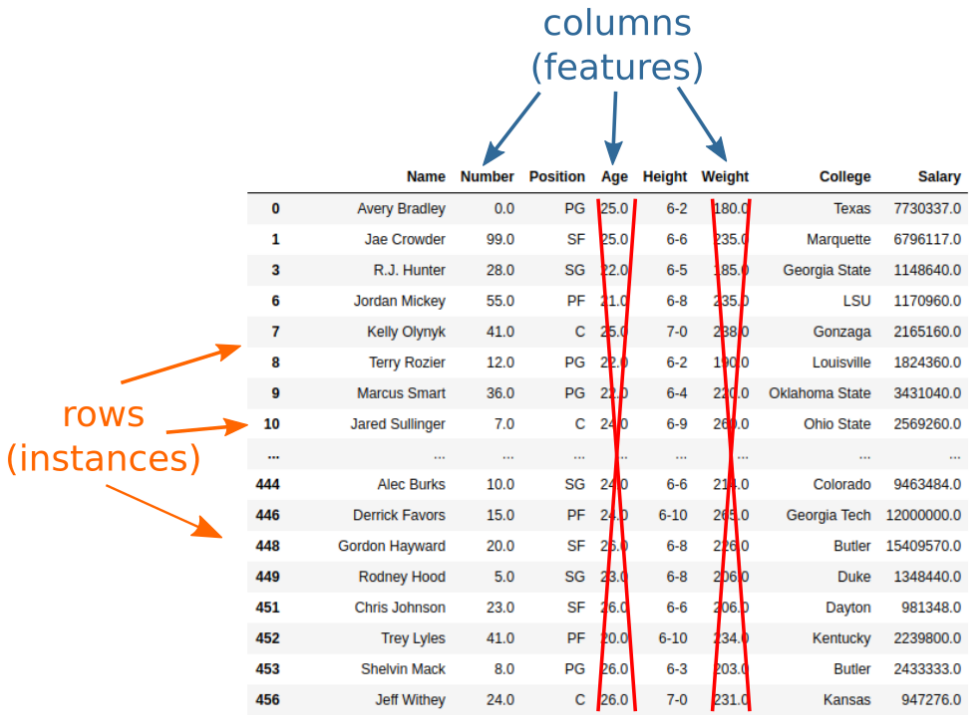
Table with rows and columns:



	Name	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
3	R.J. Hunter	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
6	Jordan Mickey	55.0	PF	21.0	6-8	235.0	LSU	1170960.0
7	Kelly Olynyk	41.0	C	25.0	7-0	238.0	Gonzaga	2165160.0
8	Terry Rozier	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
9	Marcus Smart	36.0	PG	22.0	6-4	220.0	Oklahoma State	3431040.0
10	Jared Sullinger	7.0	C	24.0	6-9	260.0	Ohio State	2569260.0
...	...	...	...	...	...	...	...	...
444	Alec Burks	10.0	SG	24.0	6-6	214.0	Colorado	9463484.0
446	Derrick Favors	15.0	PF	24.0	6-10	265.0	Georgia Tech	12000000.0
448	Gordon Hayward	20.0	SF	26.0	6-8	226.0	Butler	15409570.0
449	Rodney Hood	5.0	SG	23.0	6-8	206.0	Duke	1348440.0
451	Chris Johnson	23.0	SF	26.0	6-6	206.0	Dayton	981348.0
452	Trey Lyles	41.0	PF	20.0	6-10	234.0	Kentucky	2239800.0
453	Shelvin Mack	8.0	PG	26.0	6-3	203.0	Butler	2433333.0
456	Jeff Withey	24.0	C	26.0	7-0	231.0	Kansas	947276.0

Data format

Elimination of some features → **Feature Selection**.



	Name	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
3	R.J. Hunter	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
6	Jordan Mickey	55.0	PF	21.0	6-8	235.0	LSU	1170960.0
7	Kelly Olynyk	41.0	C	25.0	7-0	238.0	Gonzaga	2165160.0
8	Terry Rozier	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
9	Marcus Smart	36.0	PG	22.0	6-4	220.0	Oklahoma State	3431040.0
10	Jared Sullinger	7.0	C	24.0	6-9	260.0	Ohio State	2569260.0
...	...	...	...	...	...	...	...	...
444	Alec Burks	10.0	SG	24.0	6-6	214.0	Colorado	9463484.0
446	Derrick Favors	15.0	PF	24.0	6-10	265.0	Georgia Tech	12000000.0
448	Gordon Hayward	20.0	SF	26.0	6-8	226.0	Butler	15409570.0
449	Rodney Hood	5.0	SG	23.0	6-8	206.0	Duke	1348440.0
451	Chris Johnson	23.0	SF	26.0	6-6	206.0	Dayton	981348.0
452	Trey Lyles	41.0	PF	20.0	6-10	234.0	Kentucky	2239800.0
453	Shelvin Mack	8.0	PG	26.0	6-3	203.0	Butler	2433333.0
456	Jeff Withey	24.0	C	26.0	7-0	231.0	Kansas	947276.0

Data format

Converting the original d features to a smaller set of r **new** features $\rightarrow$ **Dimensionality Reduction**.

	a	b	c	d	e	f	g	h	i	j	k	l	m
0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	0.0	11.0	16.0	9.0
1	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	3.0	16.0	15.0	14.0
2	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	13.0	6.0	15.0	4.0
3	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	0.0	7.0	8.0	0.0
4	0.0	12.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	14.0	16.0	16.0	14.0
5	0.0	0.0	12.0	13.0	0.0	0.0	0.0	0.0	0.0	5.0	16.0	8.0	0.0
6	0.0	7.0	8.0	13.0	16.0	15.0	1.0	0.0	0.0	7.0	7.0	4.0	11.0
7	0.0	9.0	14.0	8.0	1.0	0.0	0.0	0.0	0.0	12.0	14.0	14.0	12.0
8	0.0	11.0	12.0	0.0	0.0	0.0	0.0	0.0	2.0	16.0	16.0	16.0	13.0
9	0.0	1.0	9.0	15.0	11.0	0.0	0.0	0.0	0.0	11.0	16.0	8.0	14.0
10	0.0	0.0	0.0	14.0	13.0	1.0	0.0	0.0	0.0	0.0	5.0	16.0	16.0
11	0.0	5.0	12.0	1.0	0.0	0.0	0.0	0.0	0.0	15.0	14.0	7.0	0.0
12	2.0	9.0	15.0	14.0	9.0	3.0	0.0	0.0	4.0	13.0	8.0	9.0	16.0
13	0.0	0.0	8.0	15.0	1.0	0.0	0.0	0.0	0.0	1.0	14.0	13.0	1.0
14	5.0	12.0	13.0	16.0	16.0	2.0	0.0	0.0	11.0	16.0	15.0	8.0	4.0
15	0.0	0.0	8.0	15.0	1.0	0.0	0.0	0.0	0.0	0.0	12.0	14.0	0.0
16	0.0	1.0	8.0	15.0	10.0	0.0	0.0	0.0	3.0	13.0	15.0	14.0	14.0
17	0.0	10.0	7.0	13.0	9.0	0.0	0.0	0.0	0.0	9.0	10.0	12.0	15.0
18	0.0	6.0	14.0	4.0	0.0	0.0	0.0	0.0	0.0	11.0	16.0	10.0	0.0

$\Longrightarrow$

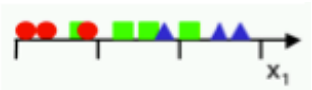
	p	q	r
0	6.714282	4.263453	-3.698543
1	10.808431	-3.605318	-5.503636
2	-5.706837	1.614261	4.903340
3	7.967174	13.080329	0.804771
4	-13.493403	-2.883923	-9.851133
5	0.428223	11.200599	4.443526
6	7.754553	-10.512425	8.192590
7	-8.213388	-1.709213	-4.202378
8	-15.188940	-2.985254	-7.907185
9	3.892792	-6.433339	0.217544
10	16.222026	-4.871032	-7.677208
11	-12.432569	7.402058	3.093797
12	-1.928984	-11.331136	2.937210
13	5.864160	11.300092	0.921414
14	-3.750759	-10.217692	14.785712
15	6.721941	12.206578	1.011207
16	2.723118	-6.903958	-1.785841
17	1.891926	-7.880966	-3.138556
18	-10.273747	8.266886	2.453369

Why reduce dimensionality?

- The volume and the dimensionality of the data $\mathbf{x} \in \mathbb{R}^d$ in machine learning (ML) applications grows constantly.

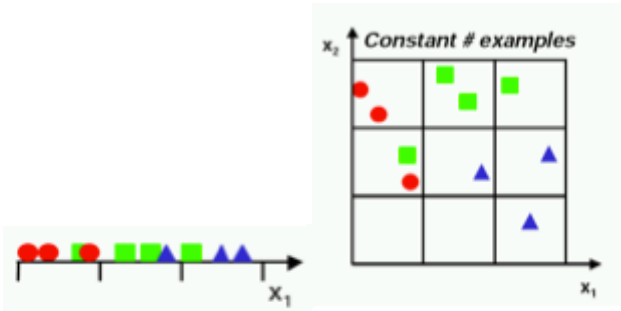
Why reduce dimensionality?

- The volume and the dimensionality of the data $\mathbf{x} \in \mathbb{R}^d$ in machine learning (ML) applications grows constantly.
- ML techniques are not effective in high-dimensional spaces $\rightarrow$ **Curse of Dimensionality**



Why reduce dimensionality?

- The volume and the dimensionality of the data $\mathbf{x} \in \mathbb{R}^d$ in machine learning (ML) applications grows constantly.
- ML techniques are not effective in high-dimensional spaces $\rightarrow$ **Curse of Dimensionality**



Why reduce dimensionality?

- The volume and the dimensionality of the data $\mathbf{x} \in \mathbb{R}^d$ in machine learning (ML) applications grows constantly.
- ML techniques are not effective in high-dimensional spaces $\rightarrow$ **Curse of Dimensionality**

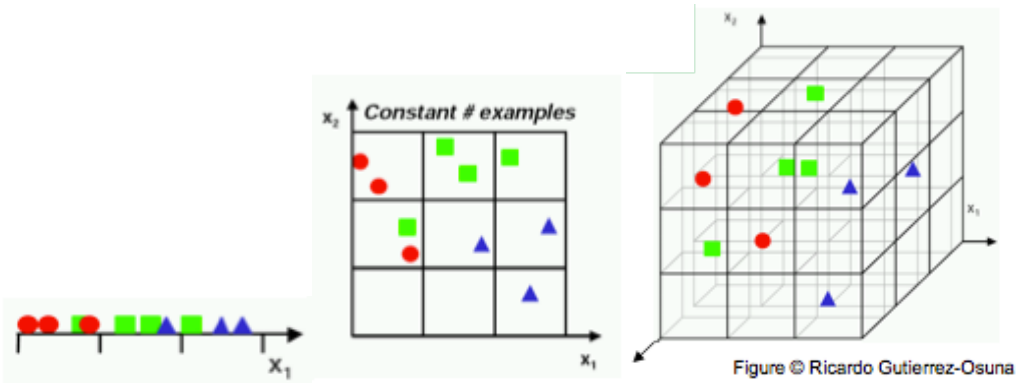
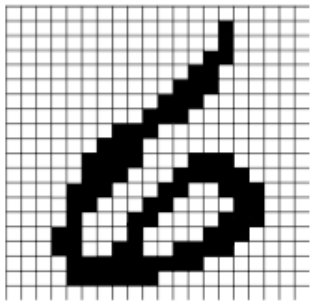


Figure © Ricardo Gutierrez-Osuna

Why reduce dimensionality?

- The intrinsic dimension of the data of interest can be much less than the extrinsic data of the observation space or feature space.

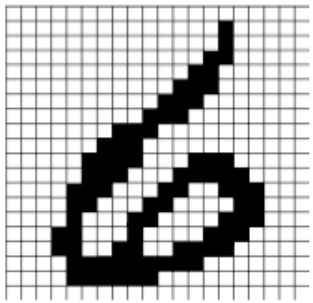


$\Longrightarrow$

The digit “6” can be represented by a small set of parameters.

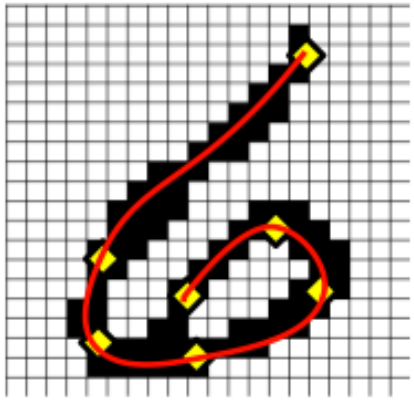
Why reduce dimensionality?

- The intrinsic dimension of the data of interest can be much less than the extrinsic data of the observation space or feature space.



$\Longrightarrow$

The digit “6” can be represented by a small set of parameters.



Other possible advantages of working in a lower dimensional data space:

- **Visualization**: Projection in 2D or 3D space.
- **Compression**: Reduction of the storage requirements while maintaining the possibility to recover the original data.
- **Noise reduction**: Projection onto a subspace or *manifold* in which the data of interest reside.
- **Convergence**: Lowering the dimension improves the convergence of regression algorithms.

Dimensionality reduction

Problem

Given n patterns or input vectors of dimension d, $\mathbf{x}_i \in \mathbb{R}^d$ ($i = 1, \ldots, n$) and a desired output space dimension, $r < d$, the problem consists in finding a transformation

$$\mathbf{x}_i \in \mathbb{R}^d \longrightarrow \mathbf{y}_i \in \mathbb{R}^r,$$

that preserves/optimizes some characteristic of the data (e.g. variance, distances, inter-class separation).

Classification of techniques

- **Linear techniques** (Projective): A linear transformation between the input space of dimension d and the output dimension space of dimension $r < d$

$$\mathbf{y}_i = \mathbf{P}^T \mathbf{x}_i, \qquad \mathbf{P} \in \mathbb{R}^{d \times r}$$

  - Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA).

Classification of techniques

- **Linear techniques** (Projective): A linear transformation between the input space of dimension d and the output dimension space of dimension $r < d$

$$\mathbf{y}_i = \mathbf{P}^T \mathbf{x}_i, \qquad \mathbf{P} \in \mathbb{R}^{d \times r}$$

  - Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA).
- **Non-linear techniques**: Techniques that aim to model/extract the nonlinear *manifold* on which the data reside.
  - Multidimensional Scaling (MDS), Isomap, Locally Linear Embedding (LLE), Stochastic Neighbor Embedding (SNE).
  - Kernel-based methods: Kernel PCA, etc.

17-dimensional example: UK food

Data from DEFRA[1]: Consumption in grams (per person, per week) of 17 different types of food-stuff measured and averaged in the four countries of the United Kingdom in 1997.

	Cheese	Carcass Meat	Other Meat	Fish	Fats and Oils	Sugars	Fresh potatoes	Fresh Veg	Other Veg	Processed potatoes	Processed Veg	Fresh Fruits	Cereals	Be
England	105	245	685	147	193	156	720	253	488	198	360	1102	1472	
Wales	103	227	803	160	235	175	874	265	570	203	365	1137	1582	
Scotland	103	242	750	122	184	147	566	171	418	220	337	957	1462	
N Ireland	66	267	568	93	209	139	1033	143	355	187	334	674	1494	

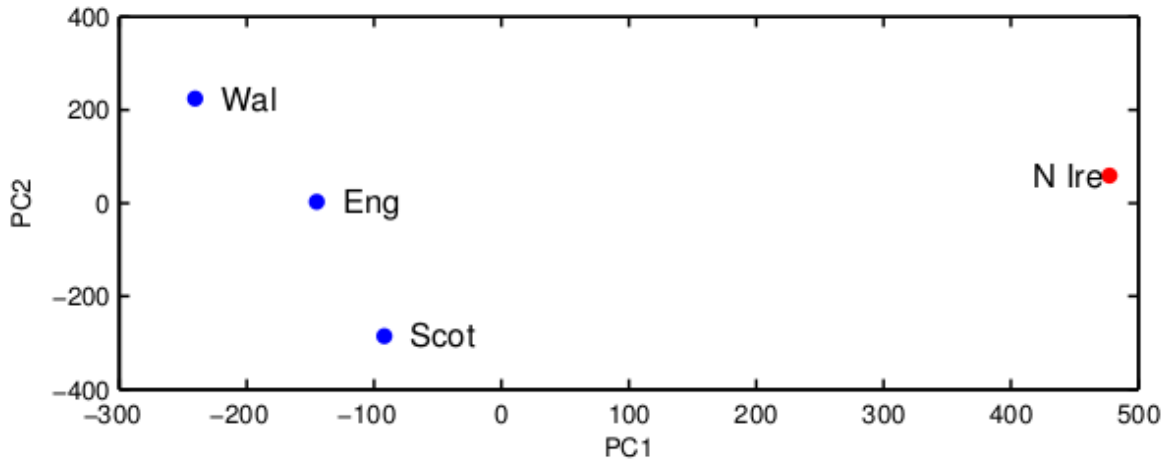
Are any countries similar? How do we visualize these data?

[1] UK's "Department for Environment, Food and Rural Affairs".

Example: UK food

	Cheese	Carcass Meat	Other Meat	Fish	Fats and Oils	Sugars	Fresh potatoes	Fresh Veg	Other Veg	Processed potatoes	Processed Veg	Fresh Fruits	Cereals	Be
England	105	245	685	147	193	156	720	253	488	198	360	1102	1472	
Wales	103	227	803	160	235	175	874	265	570	203	365	1137	1582	
Scotland	103	242	750	122	184	147	566	171	418	220	337	957	1462	
N Ireland	66	267	568	93	209	139	1033	143	355	187	334	674	1494	

Reduce 17 columns to 2, using PCA:



PCA: Introduction

PCA (Pearson 1901)

Principal Component Analysis (PCA) obtains a set of r orthogonal directions $\mathbf{P}_r = \begin{bmatrix} \mathbf{p}_1 & \dots & \mathbf{p}_r \end{bmatrix}$ that maximize the variance of the projected data $\mathbf{y}_i = \mathbf{P}_r^T \mathbf{x}_i$

PCA: Introduction

PCA (Pearson 1901)

Principal Component Analysis (PCA) obtains a set of r orthogonal directions $\mathbf{P}_r = \begin{bmatrix}\mathbf{p}_1 & \dots & \mathbf{p}_r\end{bmatrix}$ that maximize the variance of the projected data $\mathbf{y}_i = \mathbf{P}_r^T \mathbf{x}_i$

Example: Direction of maximum variance

- Given a data set $\mathbf{x}_i \in \mathbb{R}^d$ $(i = 1, \dots, n)$ with zero mean such that

$$\mathbf{x}_i = \theta_i \mathbf{u}, \qquad \sum_{i=1}^{n} \theta_i = 0$$

where $\mathbf{u}$ is a unit norm vector in$\in \mathbb{R}^d$

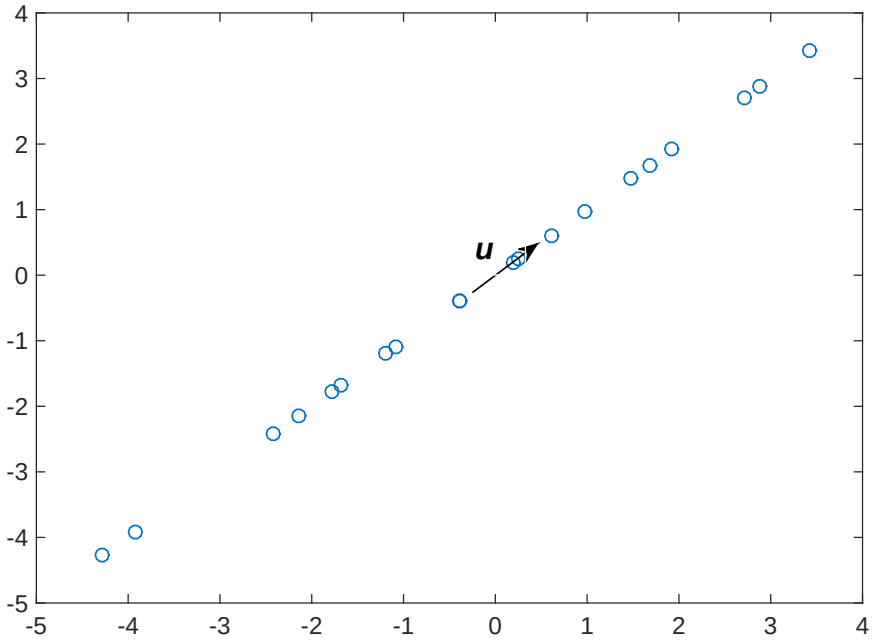
- The projection of $\mathbf{x}_i$ onto $\mathbf{u}$ is $y_i = \mathbf{u}^T \mathbf{x}_i = \theta_i$.
- The variance of the entire projected data set is

$$\sigma_u^2 = \frac{1}{n} \sum_{i=1}^{n} y_i^2 = \frac{1}{n} \sum_{i=1}^{n} \theta_i^2$$

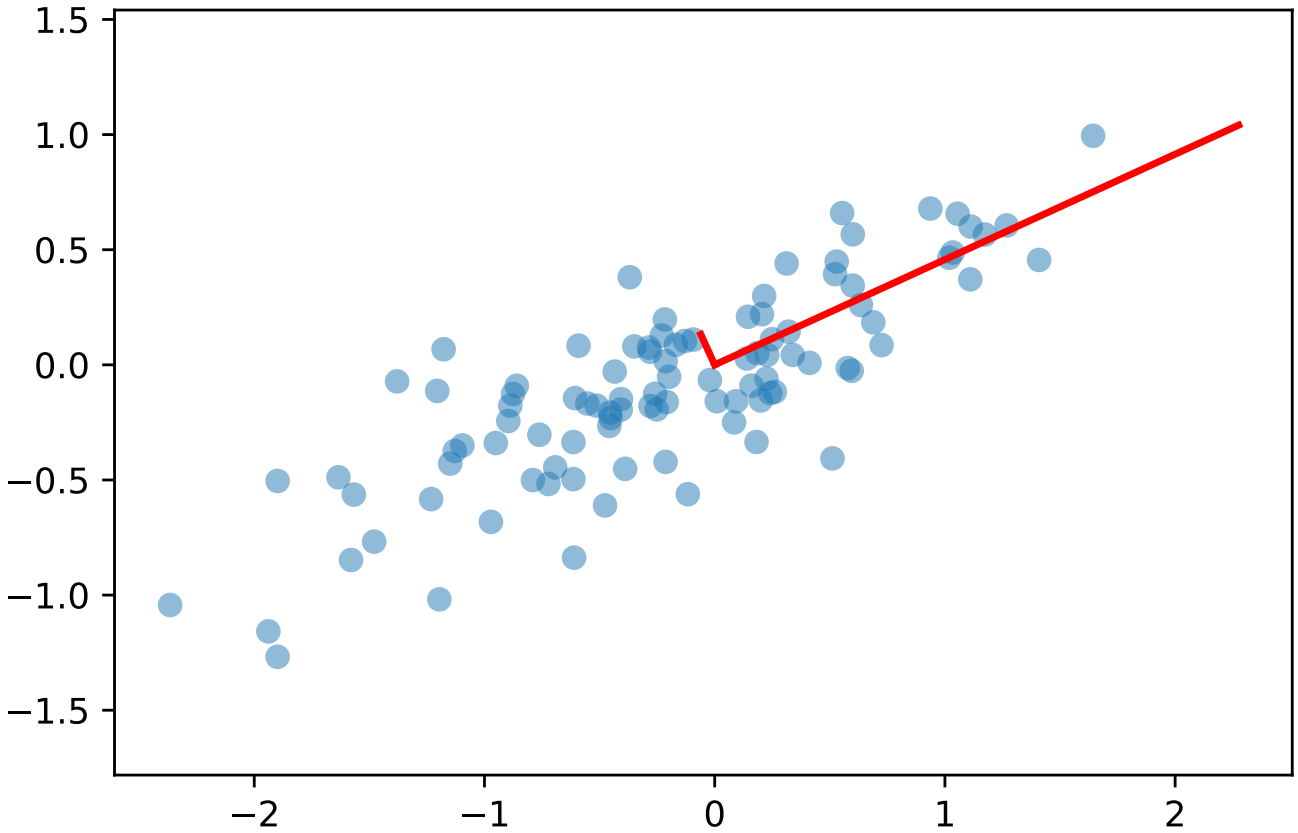
- If we project the $\mathbf{x}_i$ onto any other unit norm vector $\mathbf{v} \neq \mathbf{u}$, the variance will be smaller

$$\sigma_v^2 = \frac{1}{n} \sum_{i=1}^{n} \theta_i^2 \langle \mathbf{v}, \mathbf{u} \rangle^2 < \sigma_u^2$$

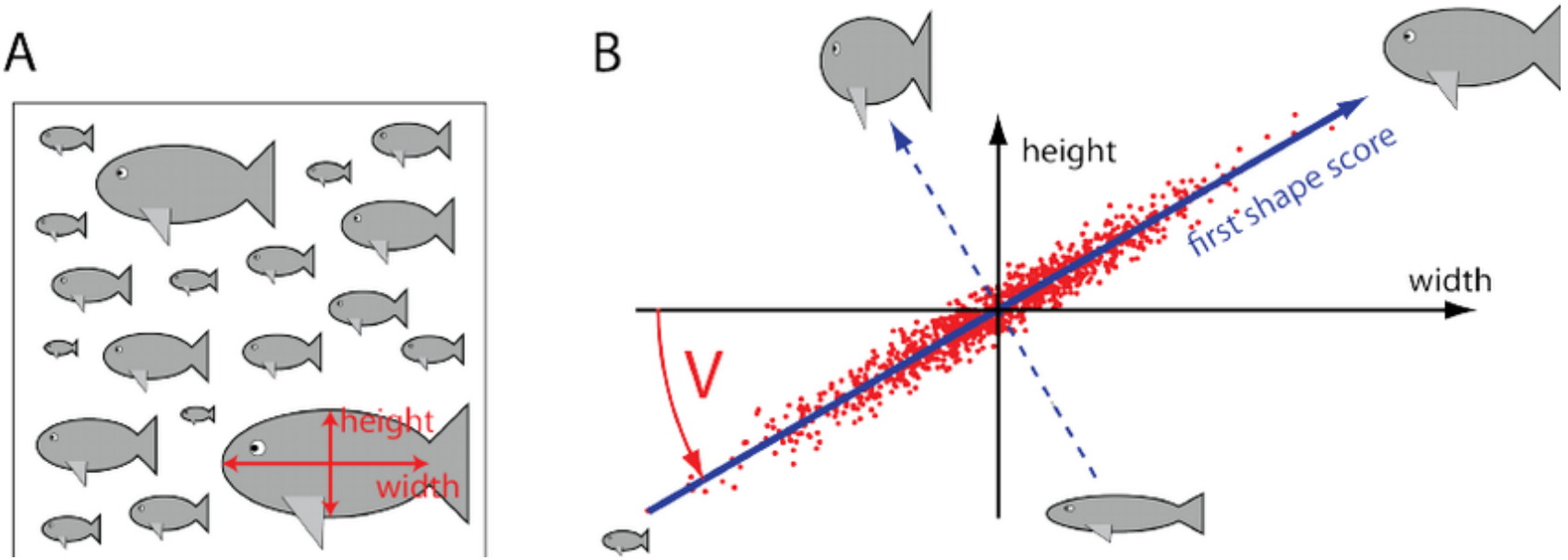
Obviously, all data from the previous example lie on a straight line:



Example: 2D data

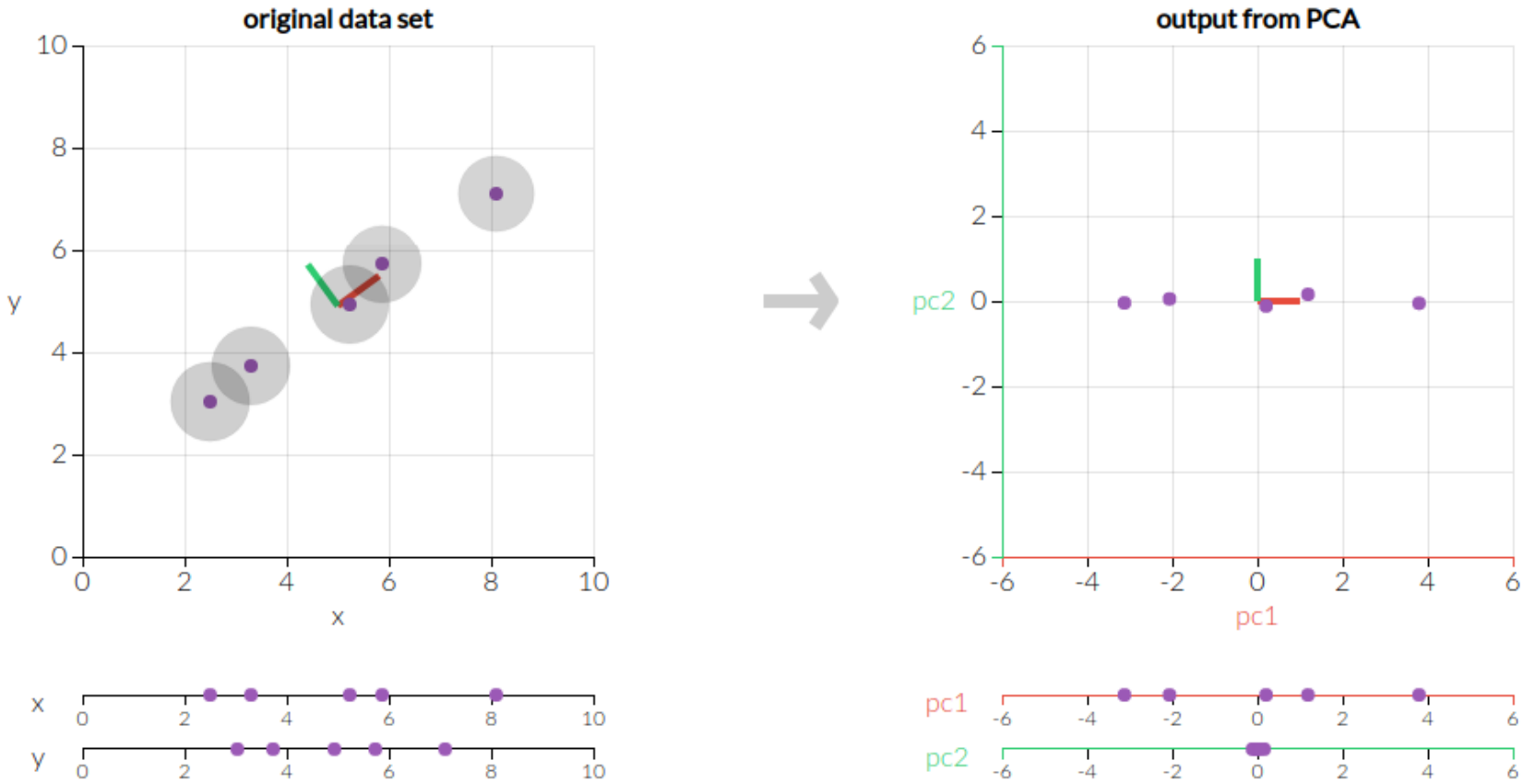


2D example



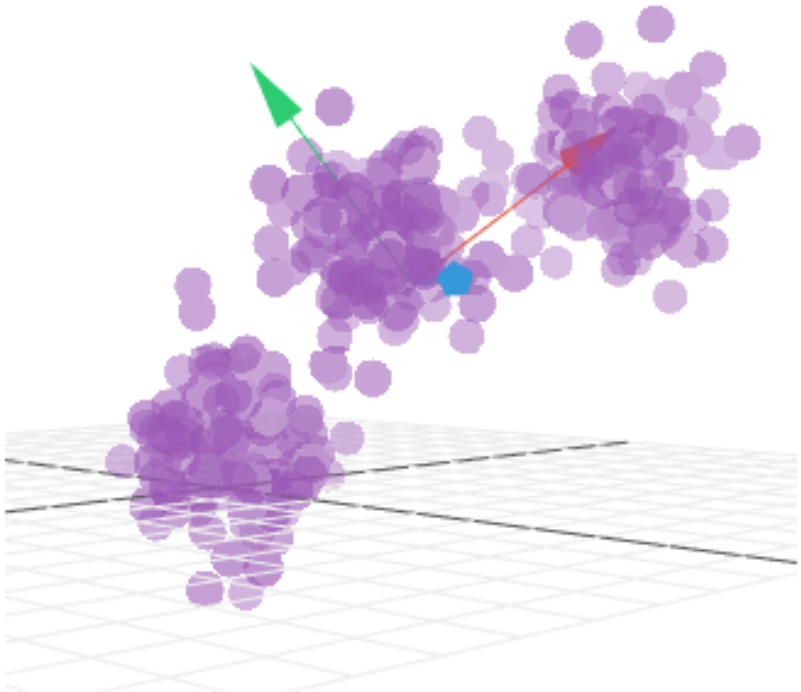
Source: Werner and Friedrich, 2014.

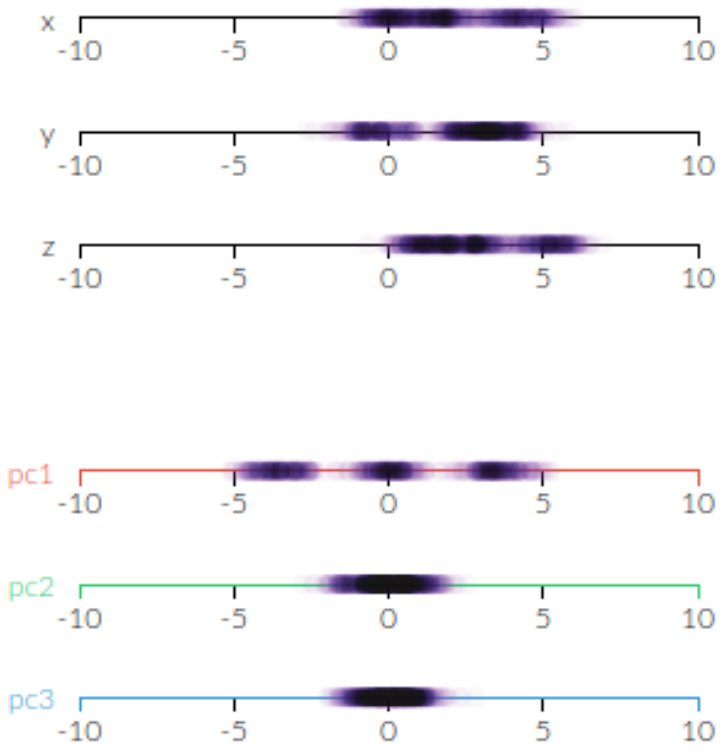
2D example



Source and interactive example: `http://setosa.io/ev/principal-component-analysis/`

3D example




x
-10
-5
0
5
10
y
-10
-5
0
5
10
z
-10
-5
0
5
10
pc1
-10
-5
0
5
10
pc2
-10
-5
0
5
10
pc3
-10
-5
0
5
10

PCA theory (1/3)

- Given a data set $\mathbf{x}_i \in \mathbb{R}^d$ $(i = 1, \ldots, n)$ with zero mean (if the mean is not zero, we subtract the sample mean $\mathbf{m}_x = \frac{1}{n}\sum_i \mathbf{x}_i$)

$$\mathbf{X} = \left[\mathbf{x}_1, \ldots, \mathbf{x}_n\right]^T \in \mathbb{R}^{n \times d}$$

- The **sample covariance** matrix (dimensions $d \times d$) is

$$\hat{\mathbf{C}}_x = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T = \frac{1}{n}\mathbf{X}^T\mathbf{X}$$

- If we choose a direction $\mathbf{v} \in \mathbb{R}^d$ such that $||\mathbf{v}||^2 = 1$, the variance of the data projected onto this direction is

$$\sigma_v^2 = \mathbf{v}^T \hat{\mathbf{C}}_x \mathbf{v}$$

PCA theory (2/3)

- **Decompose $\hat{\mathbf{C}}_x$ into eigenvectors and eigenvalues**:

$$\hat{\mathbf{C}}_x = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^T$$

where $\boldsymbol{\Sigma} = \text{diag}(\sigma_1^2, \ldots, \sigma_d^2)$ with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_d$

- Since $\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \ldots & \mathbf{u}_d \end{bmatrix}$ forms a basis of $\mathbb{R}^d$, $\mathbf{v}$ can be expanded as

$$\mathbf{v} = \sum_{i=1}^{n} \alpha_i \mathbf{u}_i, \quad \text{where} \quad \sum_{i=1}^{n} \alpha_i^2 = 1$$

and the variance of the projection is

$$\sigma_v^2 = \sum_i \alpha_i \sigma_i^2$$

PCA theory (3/3)

- The projection of maximum variance is found by solving

$$\underset{\alpha_i}{\text{maximize}} \sum_i \alpha_i \sigma_i^2 \qquad s.t. \quad \sum_{i=1}^{d} \alpha_i^2 = 1$$

  whose solution is $\alpha_1^* = 1, \alpha_2^* = \ldots = \alpha_d^* = 0$
- The **direction that maximizes the variance is the principal eigenvector** of $\hat{\mathbf{C}}_x$

$$\mathbf{v} = \mathbf{u}_1$$

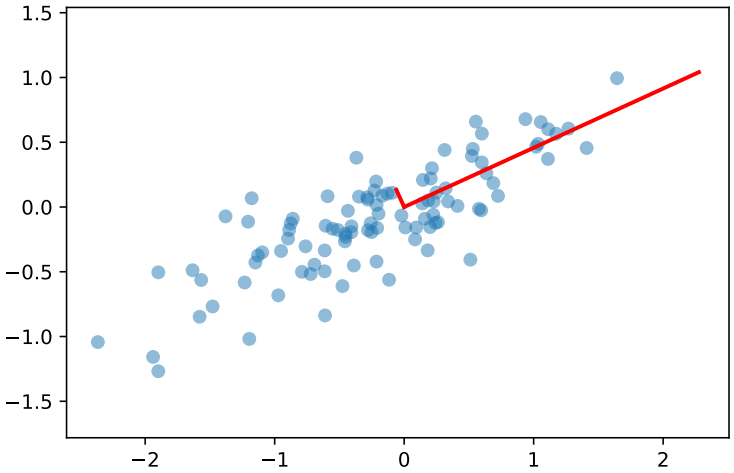
- The **first principal component** (1D projection) is

$$y_i = \mathbf{u}_1^T \mathbf{x}_i$$

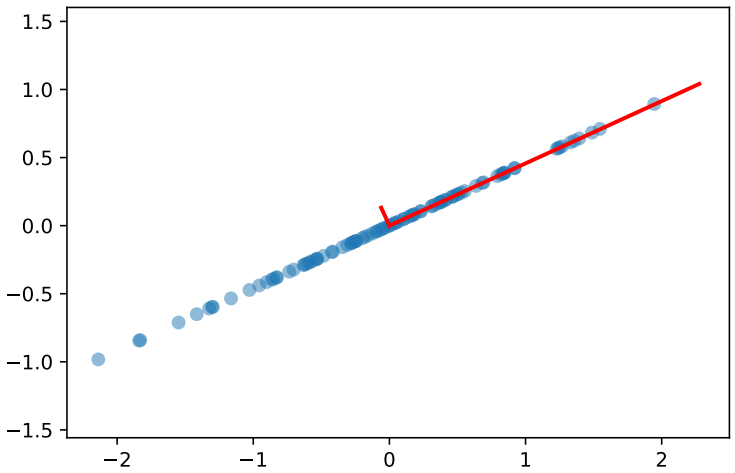
- The proportion of **variance "explained"** by y_i is $\frac{\sigma_1^2}{\sum_{i=1}^{d} \sigma_i^2}$

Example: PCA on 2D data

Data (blue) and principal directions (red).



Projection onto the first principal direction.



- Python code in `example_1_pca_toy.ipynb`

Extension to r principal components

- Since the eigenvectors form an orthogonal basis and the eigenvalues are in descending order, the r PCA directions are

$$\mathbf{U}_r = \begin{bmatrix} \mathbf{u}_1 & \dots & \mathbf{u}_r \end{bmatrix} \in \mathbb{R}^{d \times r}$$

- The data with reduced dimension are (principal components)

$$\mathbf{y}_i = \mathbf{U}_r^T \mathbf{x}_i \in \mathbb{R}^r \Rightarrow \mathbf{Y} = \mathbf{X}\mathbf{U}_r$$

- The proportion of variance "explained" by y_i is now $prop = \frac{\sum_{i=1}^{r} \sigma_i^2}{\sum_{i=1}^{d} \sigma_i^2}$

Python examples

- 3D example: `example_2_pca_3D.ipynb`
- Exercise 1: `exercise_1_iris_visualization_PCA.ipynb`

PCA decorrelates the projected data

The r principal components of the vector $\mathbf{y}_i = \mathbf{U}_r^T \mathbf{x}_i$ are uncorrelated

$$E\left[\mathbf{y}_i \mathbf{y}_i^T\right] = E\left[\mathbf{U}_r^T \mathbf{x}_i \mathbf{x}_i^T \mathbf{U}_r\right] = \mathbf{U}_r^T E\left[\mathbf{x}_i \mathbf{x}_i^T\right] \mathbf{U}_r =$$
$$\mathbf{U}_r^T E\left[\mathbf{x}_i \mathbf{x}_i^T\right] \mathbf{U}_r = \mathbf{U}_r^T \mathbf{U} \Sigma \mathbf{U}^T \mathbf{U}_r =$$
$$\Sigma_r = \text{diag}(\sigma_1^2, \ldots, \sigma_r^2)$$

PCA: Minimum MSE reconstruction

- The projected data is $\mathbf{y}_i = \mathbf{U}_r^T \mathbf{x}_i$
- If we want to reconstruct the original d-dimensional data from $\mathbf{y}_i$ we calculate

$$\tilde{\mathbf{x}}_i = \mathbf{U}_r \mathbf{y}_i = \mathbf{U}_r \mathbf{U}_r^T \mathbf{x}_i = \mathbf{P}_r \mathbf{x}_i, \quad \textbf{or} \quad \tilde{\mathbf{X}} = \mathbf{X}\mathbf{U}_r \mathbf{U}_r^T$$

  where $\mathbf{P}_r = \mathbf{U}_r \mathbf{U}_r^T$ is a projection matrix in the subspace generated by the r eigenvectors of $\hat{\mathbf{C}}_x$
- PCA solves the following problem

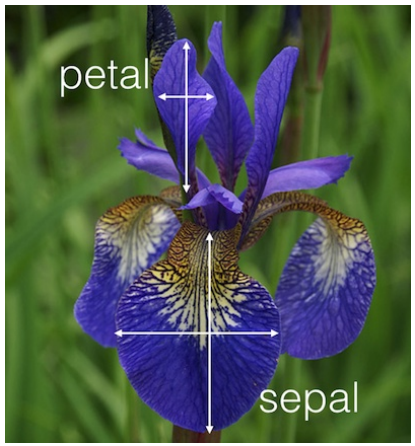
$$\underset{\mathbf{M}}{\text{minimize}} \sum_{i=i}^{n} ||\mathbf{x}_i - \tilde{\mathbf{x}}_i||^2 = ||\mathbf{X} - \mathbf{X}\mathbf{M}\mathbf{M}^T||_F^2 \qquad s.t. \quad \mathbf{M}^T\mathbf{M} = \mathbf{I}_r$$

  whose solution is $\mathbf{M} = \mathbf{U}_r$!!

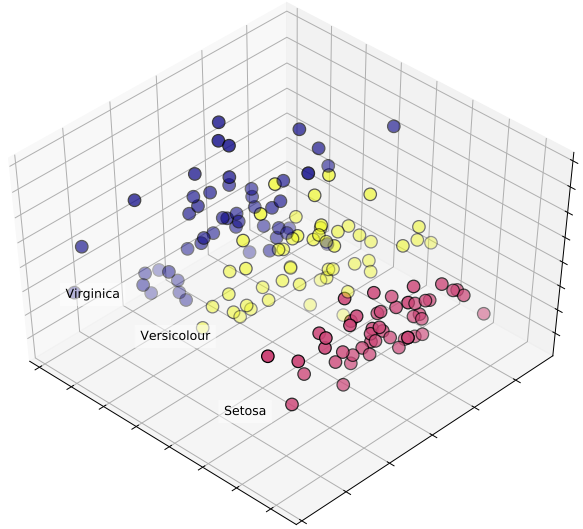
Example: PCA dimensionality reduction

Iris dataset:

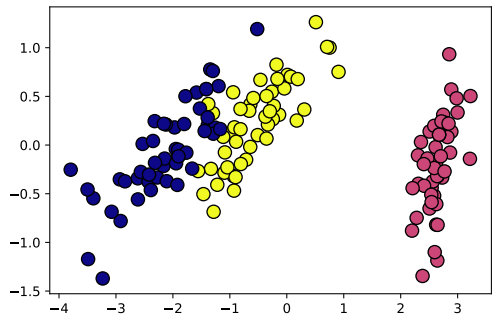
- 3 classes;
- 150 data;
- 4 features.



3 principal components:



2 principal components:

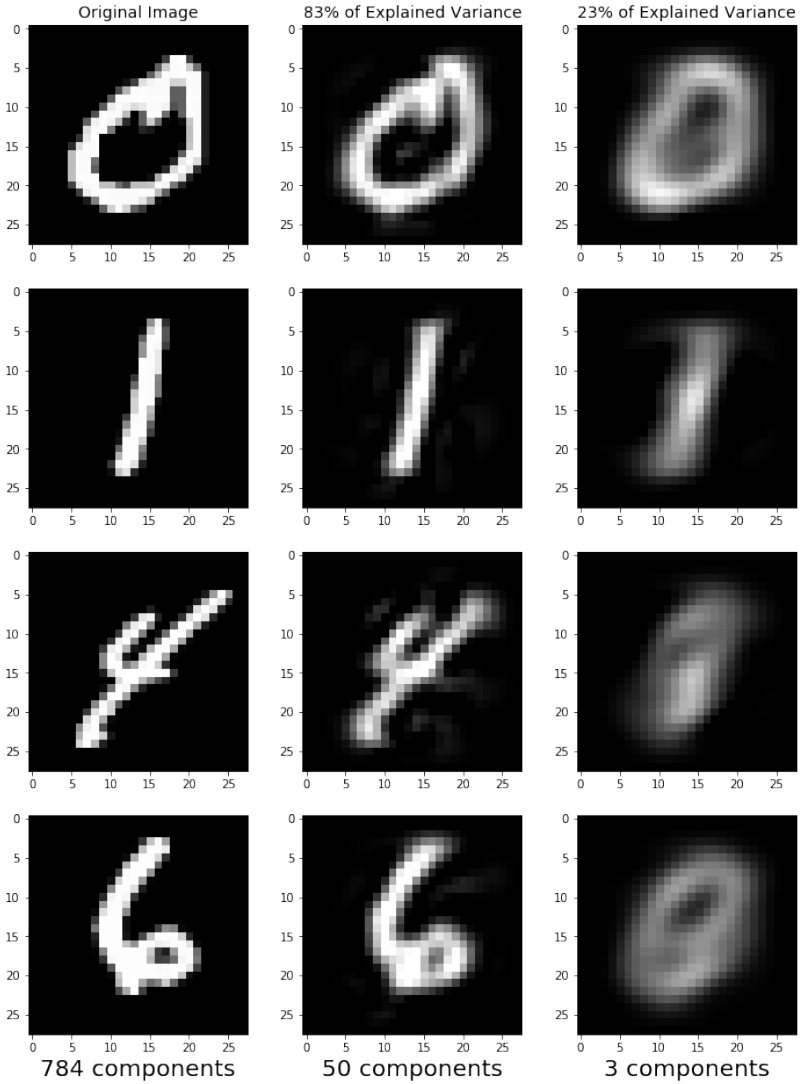


Example: Compression/reconstruction with PCA

MNIST dataset:

- 10 classes;
- 70000 images;
- 784 pixels.

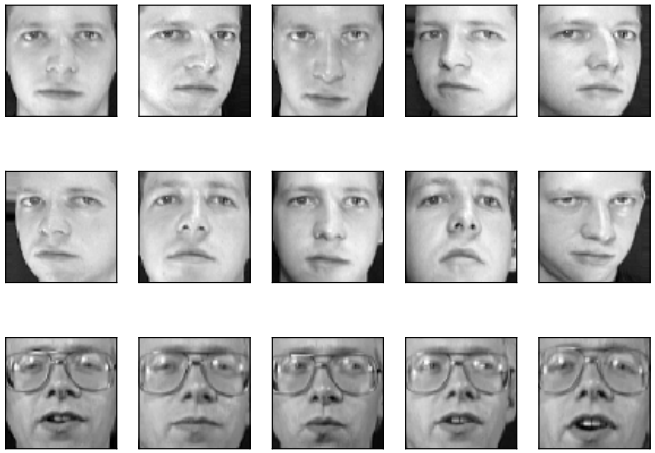
Reconstructions:



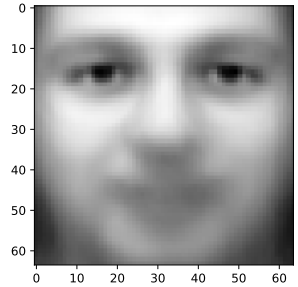
Example: Compression/reconstruction with PCA

Olivetti Faces dataset:

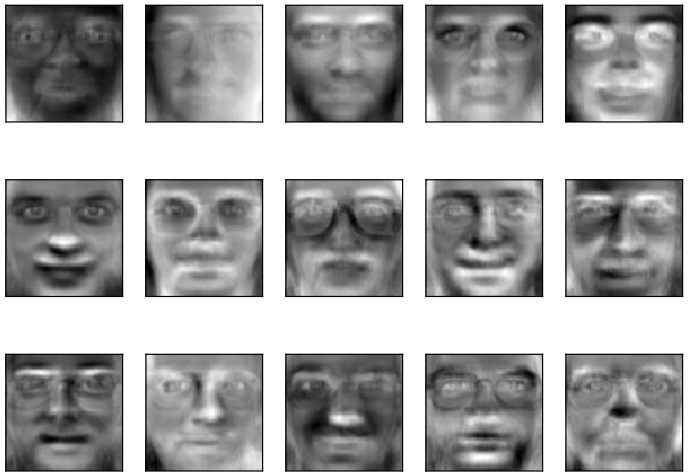
- 40 classes;
- 400 images;
- 4096 pixels.



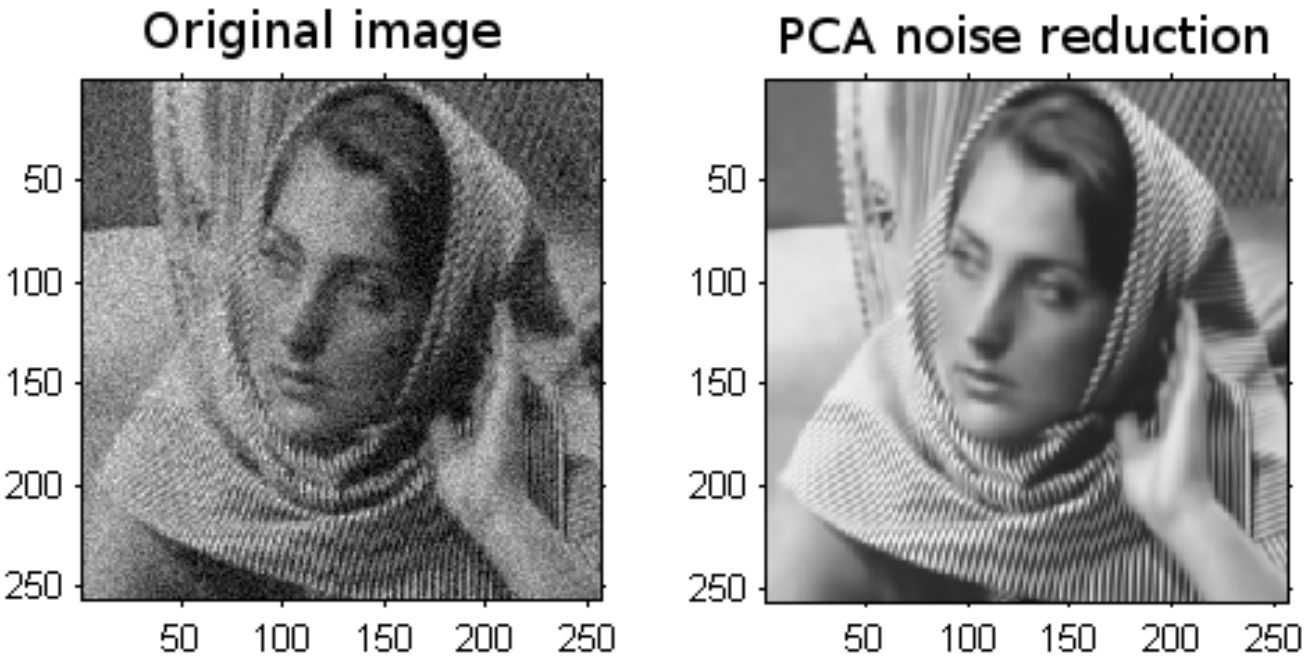
Average ("Mean face"):



Principal directions ("eigenfaces"):



Example: Noise reduction with PCA

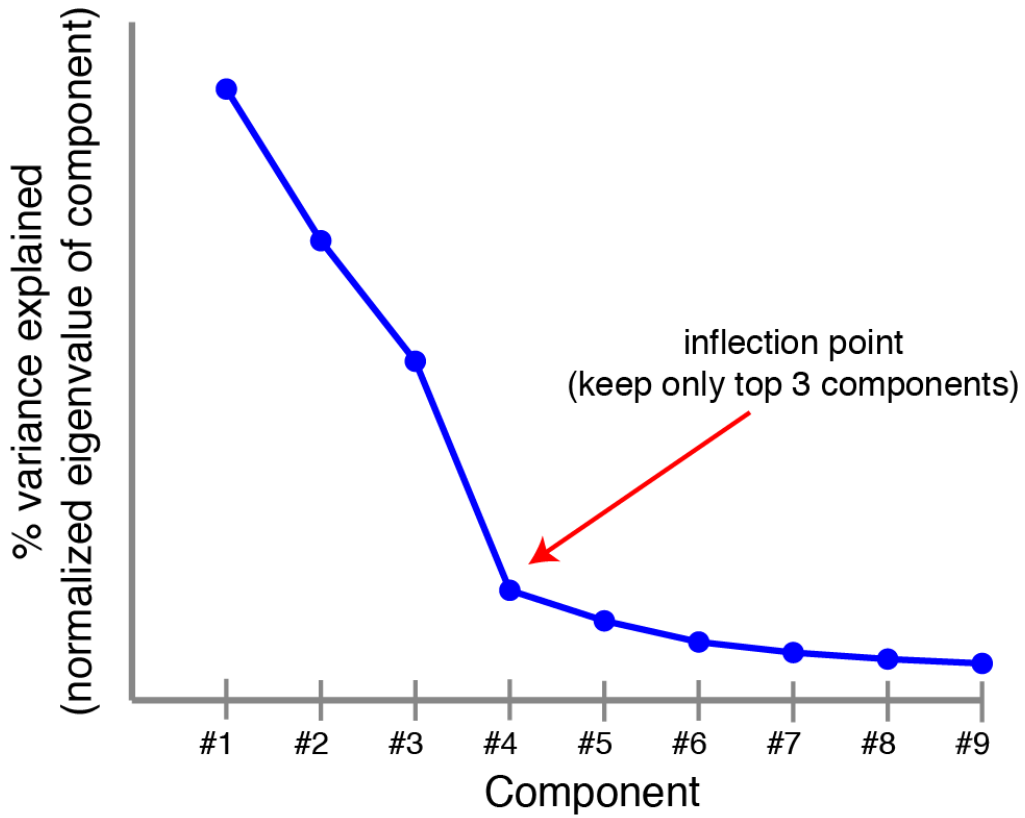


Using patches of 7x7 pixels.

Source: Shah & Bhalgat (2015). https://github.com/meetps/CS-663

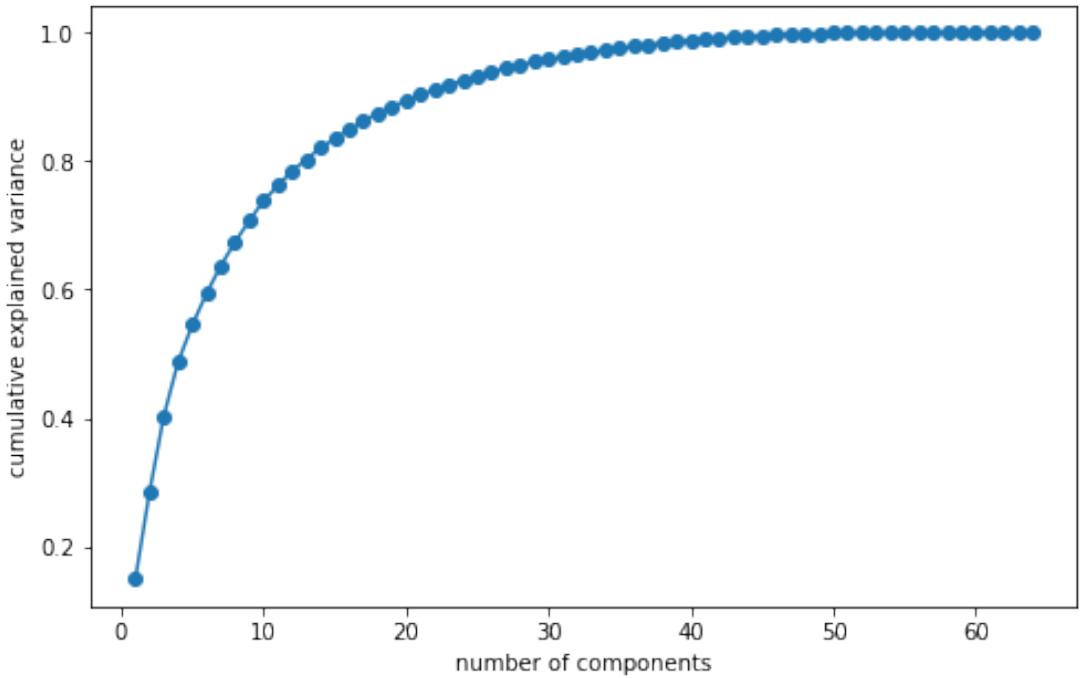
Order estimation 1: Scree plot

Plot of the variance explained by each successive principal component:



Order estimation 2: Cumulative explained variance

Cumulative plot of the variance explained by the principal components:



Given a target percentage of explained variance, this plot shows the number of principal components required.

Order estimation techniques

- Apart from establishing a threshold on the percentage of explained variance, there exist many other methods in the literature to estimate the optimal number of principal components of the model.
- On of the most popular methods is the **Minimum Description Length** (MDL) criterion [Rissanen, 1978].
- MDL selects the rank r that minimizes

$$MDL(r) = n\left[\log\left(\prod_{i=1}^{r}\sigma_i^2\right) + (d-r)\log\left(\frac{1}{d-r}\sum_{i=r+1}^{d}\sigma_i^2\right)\right] + \frac{r(2d-r)+1}{2}\log(n)$$

where n is the total number of observations and d is the dimension of the observation vectors.

PCA takeaways

- Dimensionality reduction is a fundamental stage in most machine learning pipelines.
- Related to feature selection/extraction.
- **PCA** is the most widely used linear dimensionality reduction technique.
  - Preprocessing in regression/classification problems; compression/storage of information, etc.
- Many related techniques. E.g. Linear Discriminant Analysis (**LDA**) which takes into account the class labels.

Python exercises

- Exercise 2:
  `exercise_2_data_compression_PCA.ipynb`
- Exercise 3:
  `exercise_3_digits_mapping_PCA.ipynb`

Nonlinear dimensionality reduction

- Some data have **structure** that cannot be revealed by linear projections.
- PCA will fall short.
- Example: How to make a 2D map of MNIST digits?

- We need **nonlinear dimensionality reduction** techniques.

Multidimensional Scaling (MDS)

- Torgerson, 1952
- Originally proposed in behavioral sciences for visualization
- Technique for **nonlinear dimensionality reduction**
- Data may be of any kind (not just scalar), as long as we can define a **dissimilarity measure** on them:
  - E.g. respondents are asked to rate similarities between product pairs.
- MDS searches for a mapping of the data to a low-dimensional space such that the pairwise dissimilarities become **squared distances**.

Multidimensional Scaling

1. Define a pairwise dissimilarity measure on the data

$$d_{ij}^* = diss(\mathbf{x}_i, \mathbf{x}_j)$$

   We want to map the data $\mathbf{x}_i \in \mathbb{R}^d \longrightarrow \mathbf{y}_i \in \mathbb{R}^r$

2. Denote the Euclidean distance on the mapped data as

$$d_{ij} = \|\mathbf{y}_i - \mathbf{y}_j\|$$

3. MDS searches for the mapping that minimizes a cost

$$C = \sum_{i \neq j} (d_{ij}^* - d_{ij})^2$$

Classical MDS

Classical MDS assumes d^* to be Euclidean distances.

1. Matrix of squared dissimilarities $\boldsymbol{D}_{ij}^{(2)} = (d_{ij}^*)^2$.
2. Apply double centering:

$$\boldsymbol{B} = -\frac{1}{2}\boldsymbol{J}\boldsymbol{D}^{(2)}\boldsymbol{J}$$

   where $\boldsymbol{J}$ is the centering matrix $\boldsymbol{J} = \boldsymbol{I} - \mathbf{1}\mathbf{1}^\top/n$.
   Note: $\boldsymbol{B} = \boldsymbol{X}\boldsymbol{X}^\top$.

3. Extract the r largest eigenvectors of $\boldsymbol{B}$ and their corresponding eigenvalues; place them in $\boldsymbol{E}_r$ and Λ_r.
4. MDS solution: $\boldsymbol{Y} = \boldsymbol{E}_r\Lambda_r^{1/2}$.

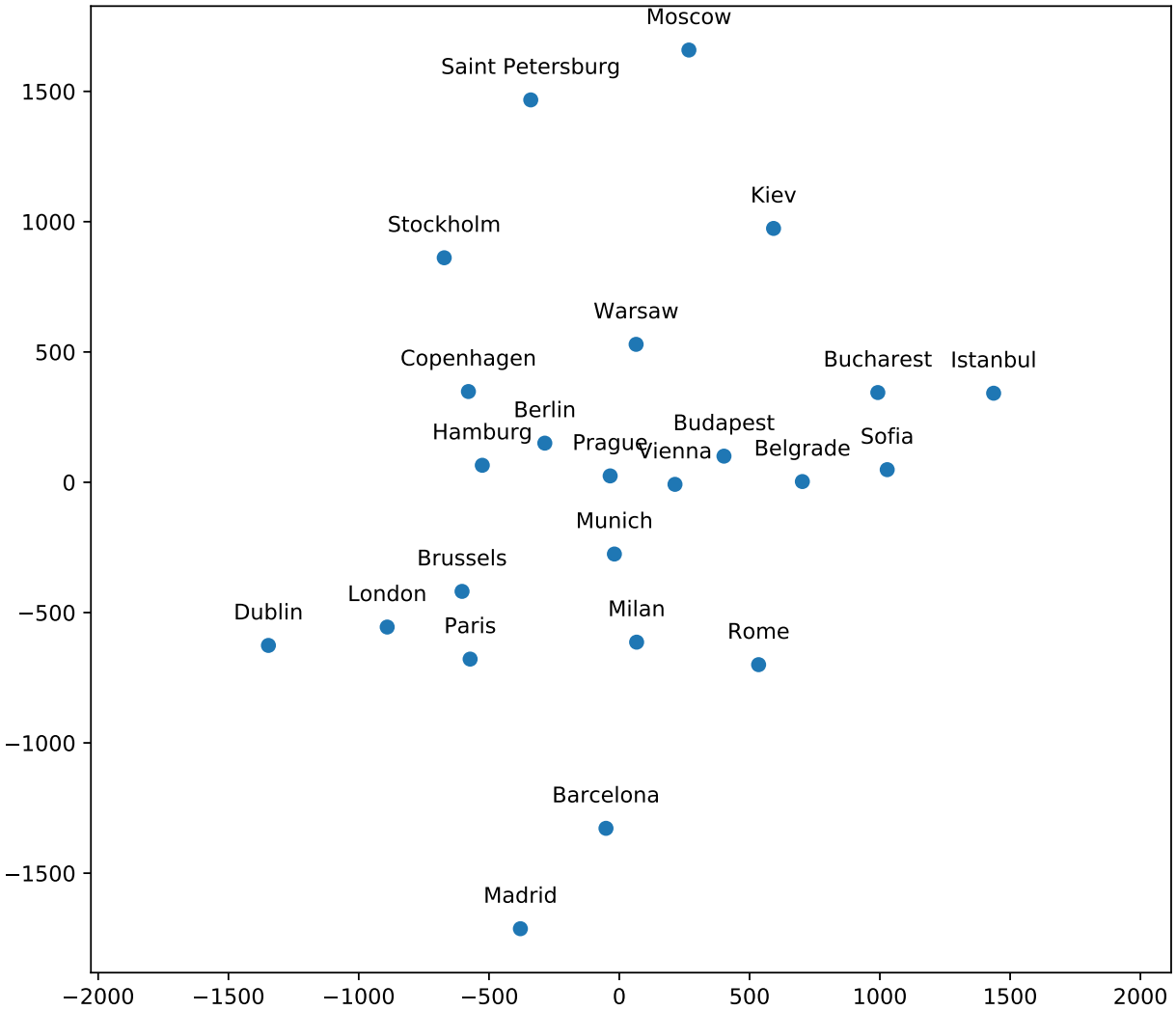
MDS example: European cities

We are given a list of distances between cities. We will use MDS to create a map.

City	Barcelona	Belgrade	Berlin	Brussels	Bucharest	...
Barcelona	0	1528.13	1497.61	1062.89	1968.42	...
Belgrade	1528.13	0	999.25	1372.59	447.34	...
Berlin	1497.61	999.25	0	651.62	1293.4	...
Brussels	1062.89	1372.59	651.62	0	1769.69	...
Bucharest	1968.42	447.34	1293.4	1769.69	0	...
⋮	⋮	⋮	⋮	⋮	⋮	⋱

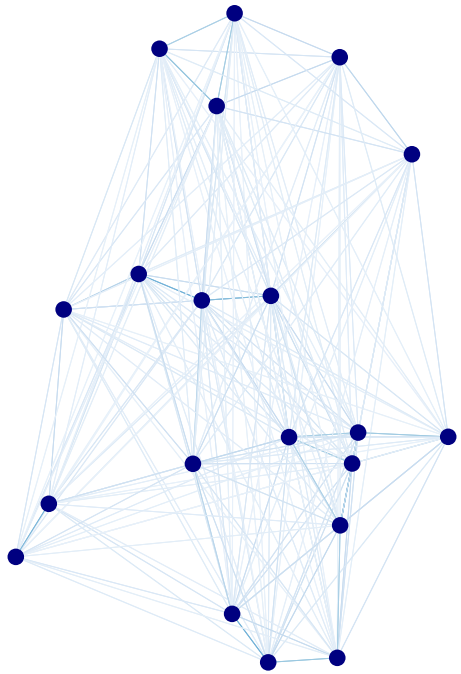
- Python code in `example_3_mds_cities.ipynb`

MDS example: European cities



Physical intuition of MDS

1. Randomly place each object on a 2D map;
2. Connect each pair $(\mathbf{x}_i, \mathbf{x}_j)$ with a spring with the length of the dissimilarity d^*_{ij};
3. Let physics take its course.

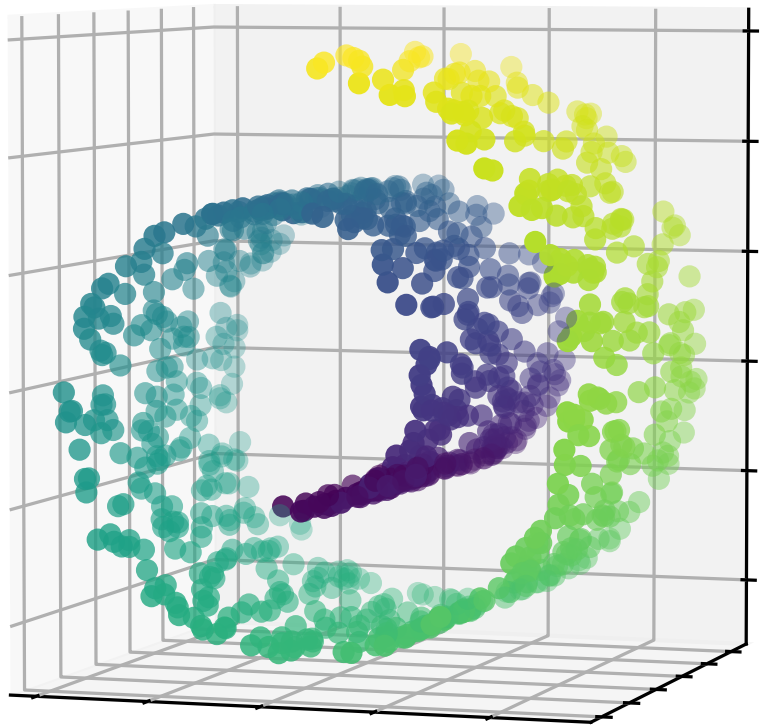


Landmark MDS

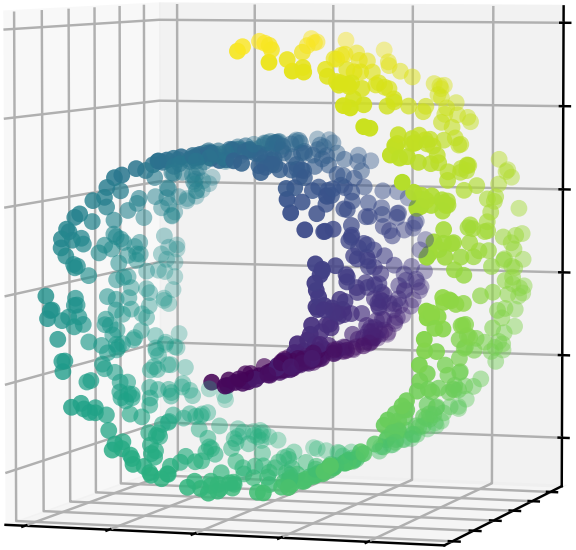
- MDS requires the eigendecomposition of the dissimilarity matrix, which is computationally expensive.
- Speedup: "Landmark MDS"
  1. Choose q data points ("landmarks"), with $r < q \ll m$;
  2. Perform MDS on landmarks;
  3. Map the remaining points to $\mathbb{R}^r$ using only their distances to landmarks.
- Landmark MDS combines MDS with the Nyström algorithm for matrix decomposition.

Manifold Modeling

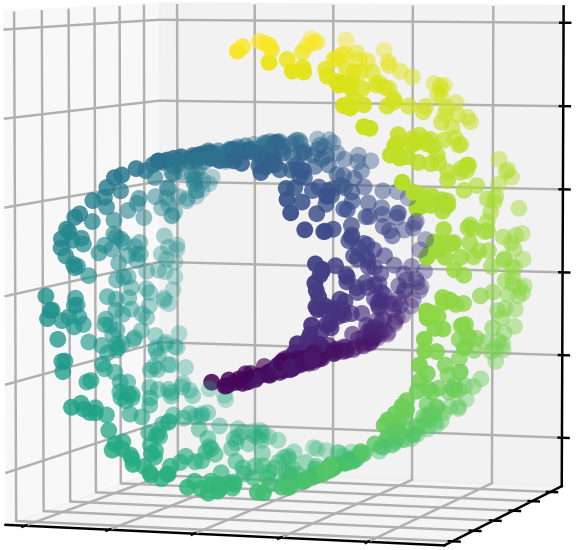
- All previous methods aim to preserve the global distance.
- What if we want to preserve **local** distance?
- Example: “Swiss roll” data:

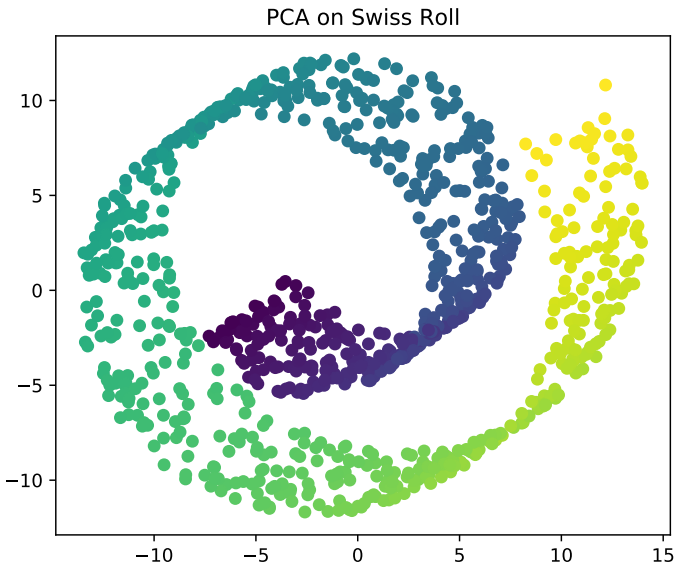


PCA on Swiss Roll data



PCA on Swiss Roll data

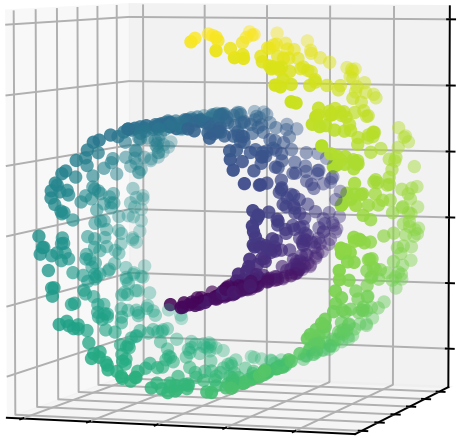




Dimensionality Reduction through Manifold Modeling

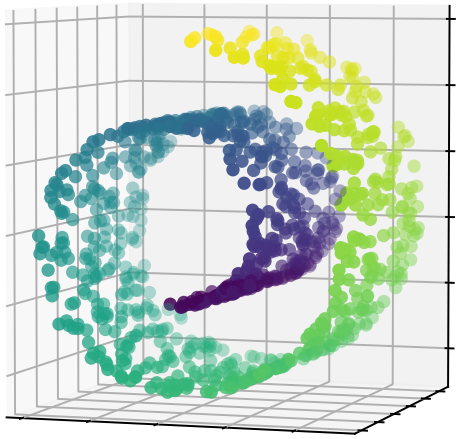
Manifold: a space that locally resembles Euclidean space.
Main idea: Find a low-dimensional representation that preserves **local properties**.

- "**local**" $\rightarrow$ with regard to neighbors.
- "**properties** $\rightarrow$ distances, linear relationships, neighborhood probabilities, etc.



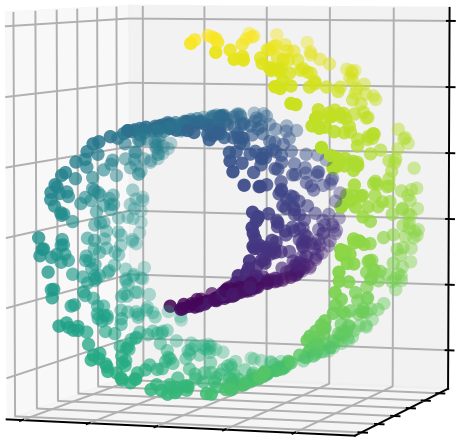
Manifold modeling

- In order to preserve the local distance we need to **model the manifold** on which the data lies.
- Example: “Swiss roll” data:

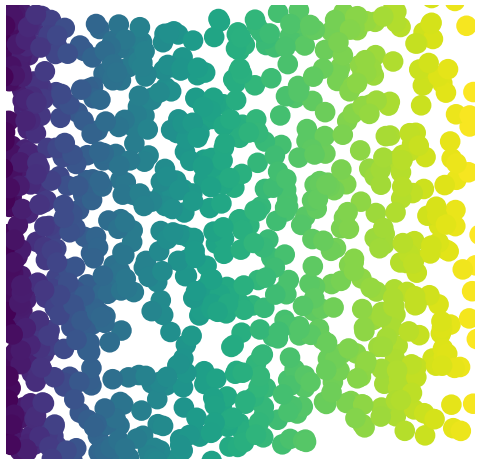


Manifold modeling

- In order to preserve the local distance we need to **model the manifold** on which the data lies.
- Example: “Swiss roll” data:



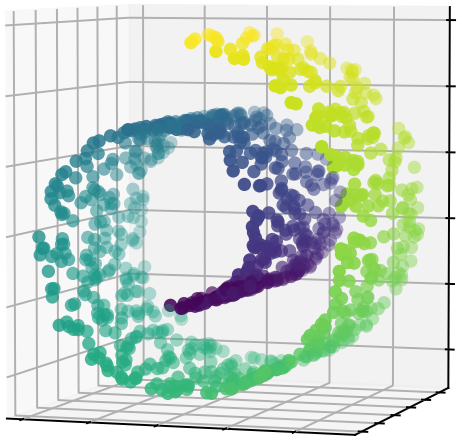
→



- Local distance on the manifold: **“geodesic distance”**.

Manifold modeling

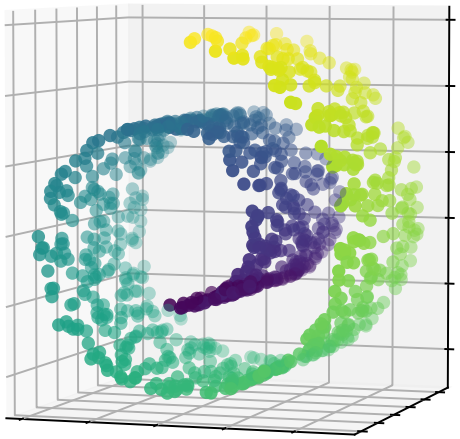
- Swiss roll data: Data on a 2D surface in 3D space.
- PCA and MDS extract a low-dimensional representation of the data but they do not explicitly model the manifold.
- PCA and MDS will fail to discover this 2D structure.



- 2 methods that do model the manifold: Isomap and LLE.

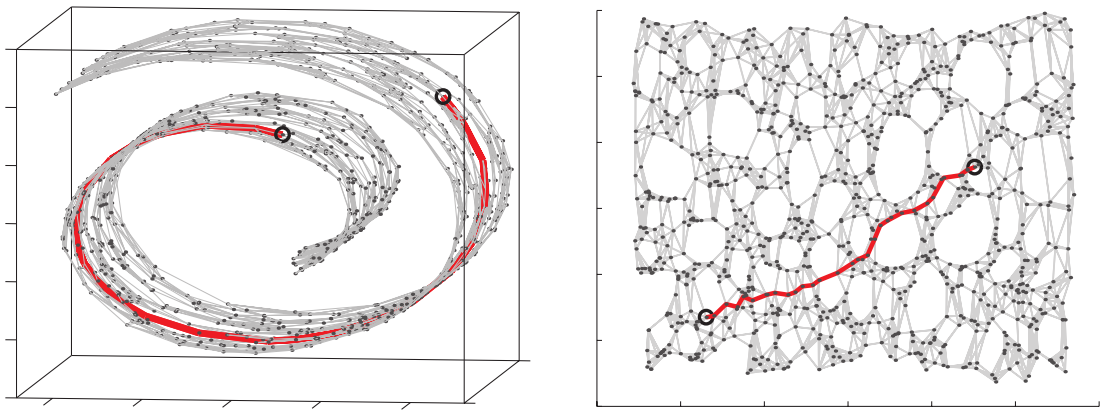
Isometric feature map (Isomap)

- Tenenbaum et al., 2000;
- Idea: Instead of true distance between two points use the **distance along the manifold**;
- This distance can be very large even if the points are close in $\mathbb{R}^d$;



Isomap: Steps

1. Construct a **graph** whose nodes are the data points, where a pair of nodes are adjacent only if the two points are close in $\mathbb{R}^d$.
2. Approximate the **geodesic distance** along the manifold between any two points as the **shortest path** in the graph;
3. Finally use **MDS** to extract the low-dimensional representation from the resulting matrix of squared distances.



Dijkstra's algorithm for shortest path calculation

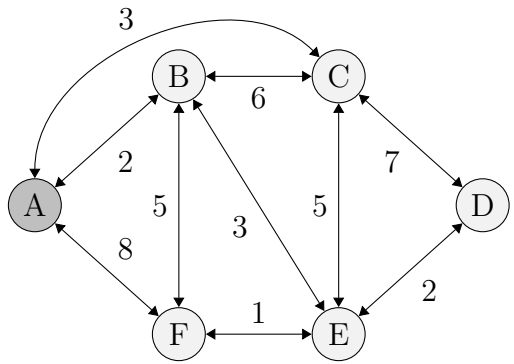
Algorithm 1 Dijkstra's Shortest Path Algorithm.

Input: nodes $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and the distances $d_{i,j}$ between pairs.
for all i **do**
  Initialize effective dist. $\bar{d}_{i,i} = 0$ and $\bar{d}_{i,j} = \infty$, for all $j \neq i$.
  Set node i as current node c.
  Mark al nodes as "unvisited".
  repeat
    for all unvisited nodes n neighboring current node **do**
      if $\bar{d}_{i,n} > \bar{d}_{i,c} + d_{c,n}$ **then**
        Update effective distance: $\bar{d}_{i,n} = \bar{d}_{i,c} + d_{c,n}$.
      end if
    end for
    Mark current node as visited.
    Next current = closest unvisited.
  until all nodes are visited.
end for
Output: effective distances $\bar{d}_{i,j}$.

Dijkstra's algorithm for shortest path calculation

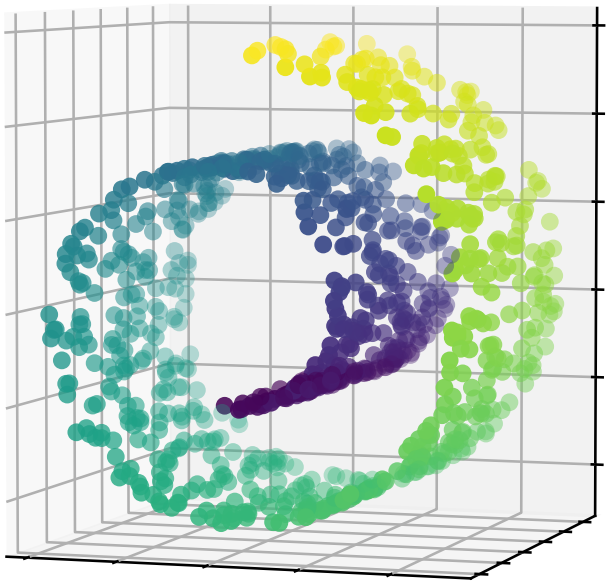
Algorithm 2 Dijkstra's Shortest Path Algorithm.

Input: nodes $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and the distances $d_{i,j}$ between pairs.
for all i **do**
  Initialize effective dist. $\bar{d}_{i,i} = 0$ and $\bar{d}_{i,j} = \infty$, for all $j \neq i$.
  Set node i as current node c.
  Mark al nodes as "unvisited".
  repeat
    for all unvisited nodes n neighboring current node **do**
      if $\bar{d}_{i,n} > \bar{d}_{i,c} + d_{c,n}$ **then**
        Update effective distance: $\bar{d}_{i,n} = \bar{d}_{i,c} + d_{c,n}$.
      end if
    end for
    Mark current node as visited.
    Next current = closest unvisited.
  until all nodes are visited.
end for
Output: effective distances $\bar{d}_{i,j}$.



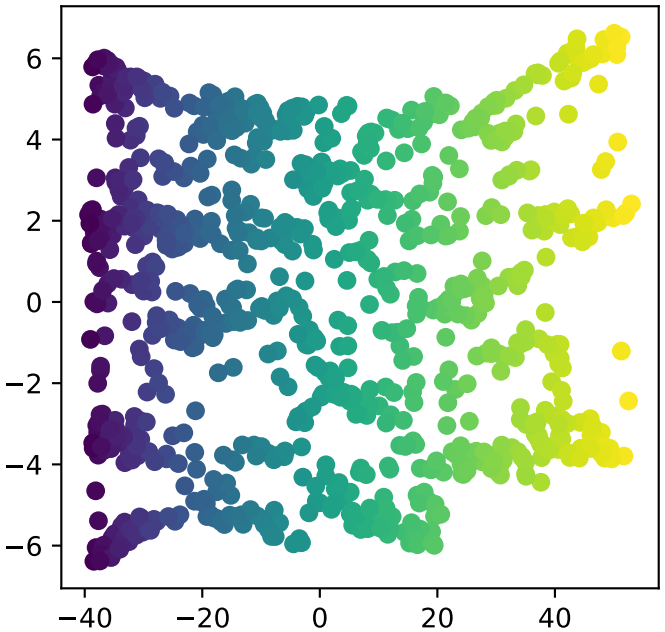
Example: Isomap on Swiss roll data

Data in $\mathbb{R}^d$



$\rightarrow$

Data in $\mathbb{R}^2$



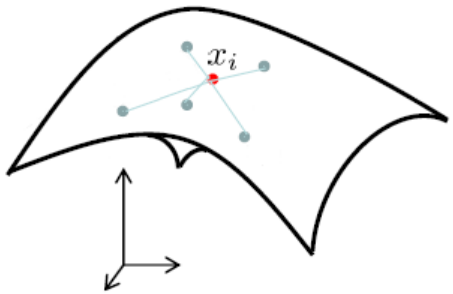
- Python code in `example_4_isomap_swiss_roll.ipynb`

Additional properties of Isomap

- Isomap includes an eigendecomposition, which is **computationally expensive**, $\mathcal{O}(N)^3$.
- Speedup: "Landmark Isomap", based on Landmark MDS and calculates the geodesic distances only to the landmarks.
- Isomap does not provide a direct functional form for the mapping $\mathcal{I} : \mathbb{R}^d \to \mathbb{R}^r$ that can simply be applied to new data. This further raises computational complexity.

Locally Linear Embedding (LLE)

- Roweis & Saul, 2004.
- Motivation: on a **local** scale the manifold can be approximated by a **linear** subspace.
- Idea: Model the manifold as a **union of linear patches**.
- Approximate each point $\mathbf{x}_i$ as a linear combination of its **neighbors**: $\mathbf{x}_i \approx \sum_{j \in \mathcal{N}(i)} W_{ij} \mathbf{x}_j$



LLE: steps

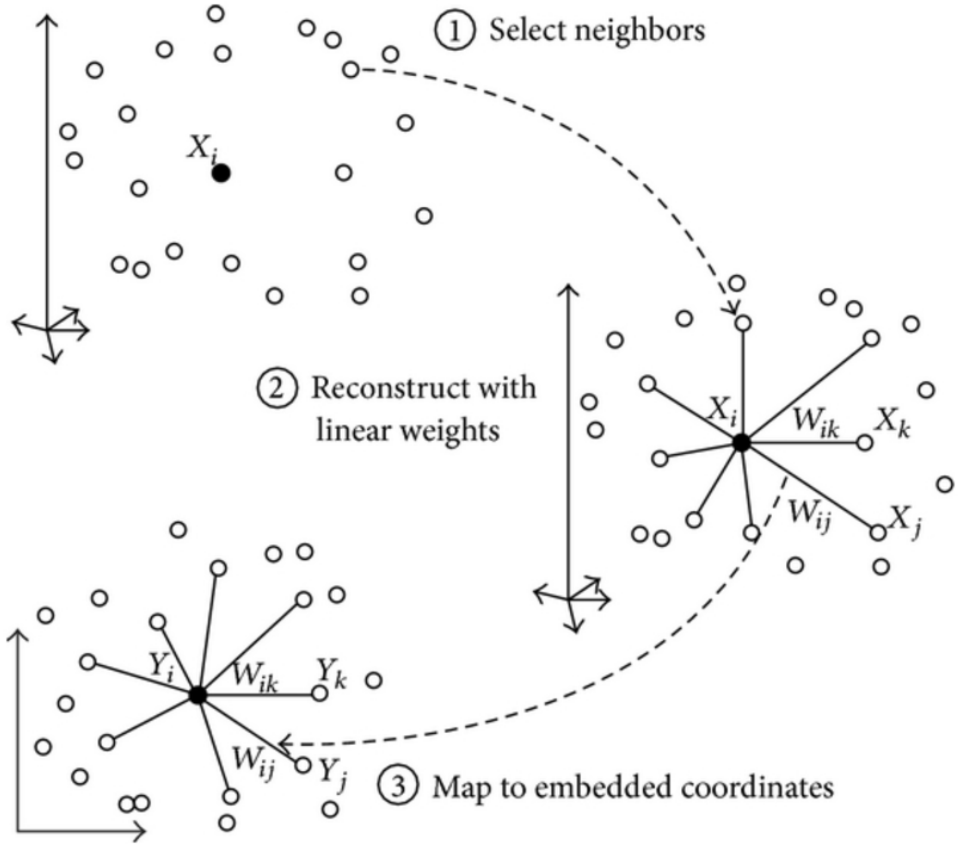
1. Index the nearest neighbors of each $\mathbf{x}_i \in \mathbb{R}^d$ as $\mathcal{N}(i)$.
2. Find the W that minimizes the reconstruction error

$$\sum_i \|\mathbf{x}_i - \sum_{j \in \mathcal{N}(i)} W_{ij}\mathbf{x}_j\|^2$$

3. Find a set of $\mathbf{y}_i \in \mathbb{R}^r$ by minimizing

$$\sum_i \|\mathbf{y}_i - \sum_{j\mathcal{N}(i)} W_{ij}\mathbf{y}_j\|^2$$

LLE steps: graphically



Python exercise

- Exercise 4: `exercise_4_swiss_roll_reduction_nonlinear.ipynb`

Extensions: Modified LLE

- Regularization problem in LLE:
  - Local W matrices become **rank-deficient** when the number of **neighbors** is **greater than** the number of input **dimensions**.
  - Therefore, LLE applies a regularization coefficient.
  - But regularization distorts the geometry of the manifold.

Extensions: Modified LLE

- Regularization problem in LLE:
  - Local W matrices become **rank-deficient** when the number of **neighbors** is **greater than** the number of input **dimensions**.
  - Therefore, LLE applies a regularization coefficient.
  - But regularization distorts the geometry of the manifold.
- "Modified" LLE solves this problem by using multiple weight vectors in each neighborhood.
- Several other extensions: Hessian LLE, LTSA, etc.
- Note: LLE provides a functional form for the mapping $\mathcal{I} : \mathbb{R}^d \rightarrow \mathbb{R}^r$

Advanced techniques

- Popular state-of-the-art techniques:
  - t-SNE (2008)
  - UMAP (2018)
- Capable operating on complex data.
- Popular in the data science community.

Stochastic Neighbor Embedding (SNE)

- Hinton & Roweis, 2003.
- Constructs a **probability distribution of the potential neighbors** of all $\mathbf{x}_i$ by placing a Gaussian at each location.
- Similarly, constructs a probability distribution over all $\mathbf{y}_i \in \mathbb{R}^r$.
- SNE uses gradient descent to **minimize the Kullback-Leibler divergence** between both distributions.
- Non-convex optimization problem; SNE uses several heuristics.

t-distributed SNE (t-SNE)

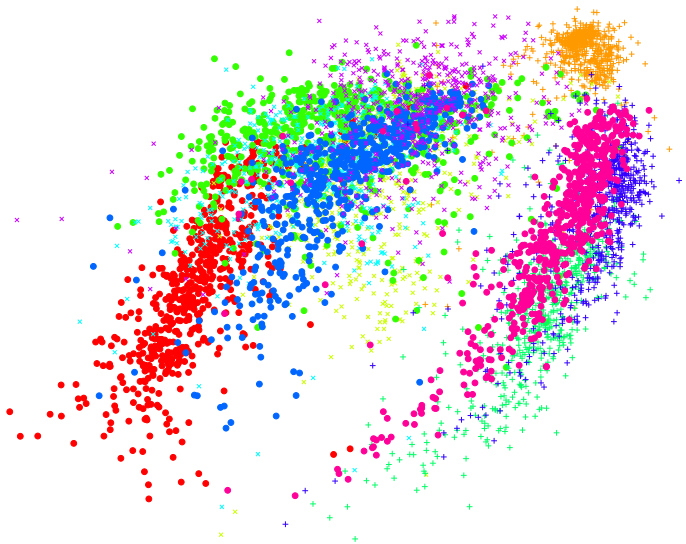
t-SNE:

- A recent extension of SNE (van der Maaten & Hinton, 2008).
- Based on a simpler cost function than SNE and uses Student t-distributions rather than Gaussians.
- Better results than SNE and faster (converges earlier).
- Parameter “perplexity”: balance between local and global aspects.
- Very **flexible** algorithm, but **hard to interpret** and finetune.

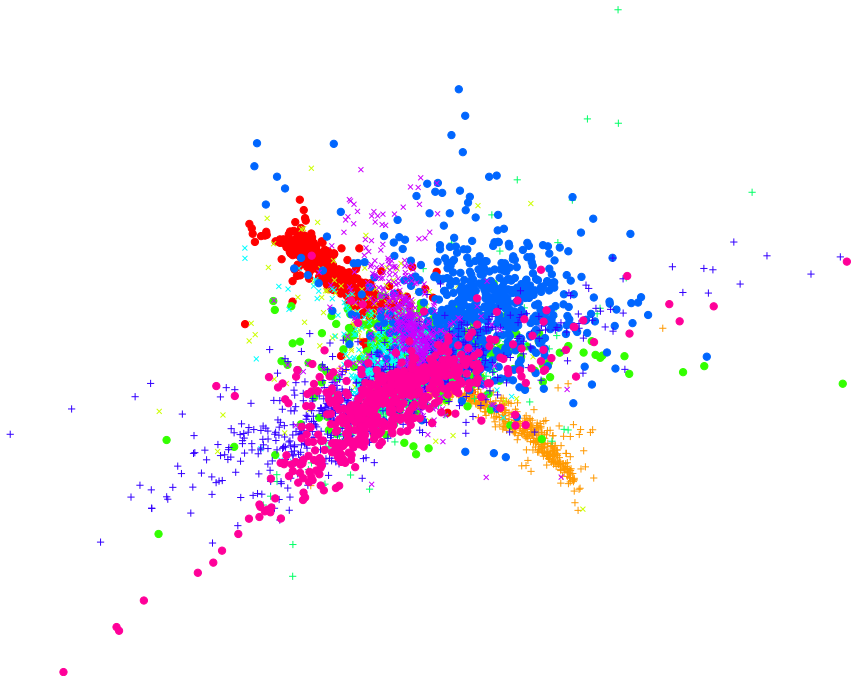
Python exercise

- Exercise 5:
  `exercise_5_digits_mapping_nonlinear.ipynb`

Example: Visualizations of MNIST

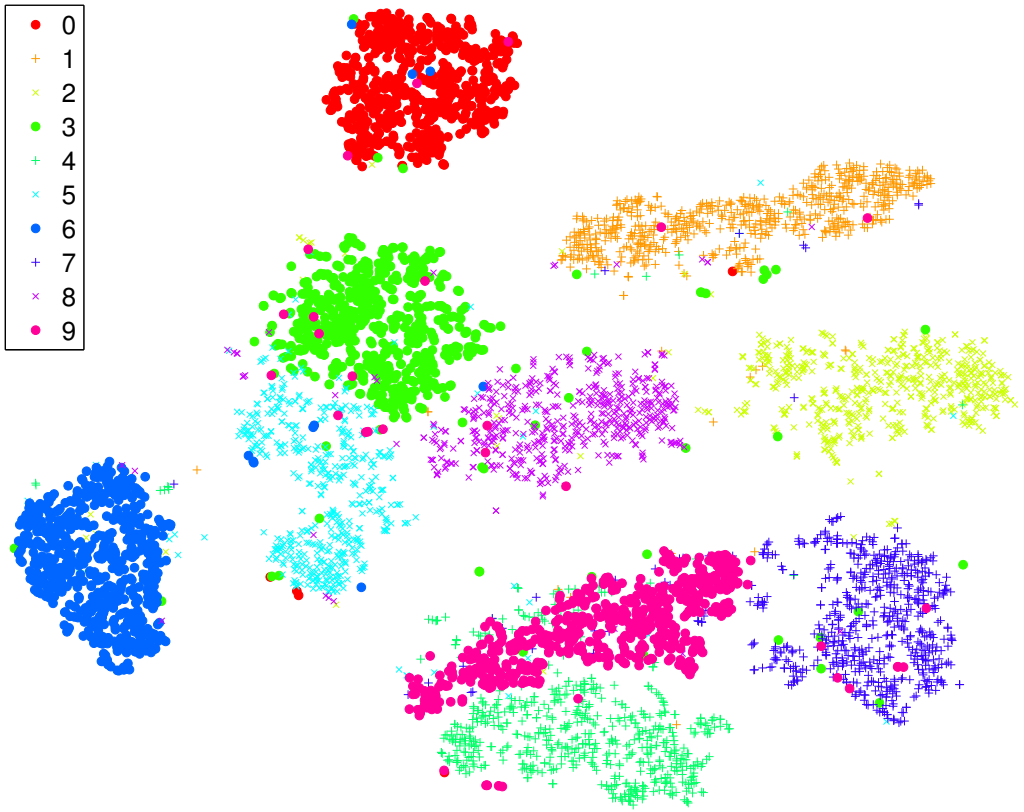


Isomap



LLE

Example: Visualizations of MNIST

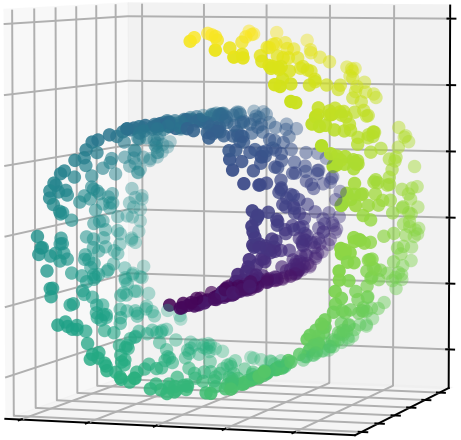


t-SNE

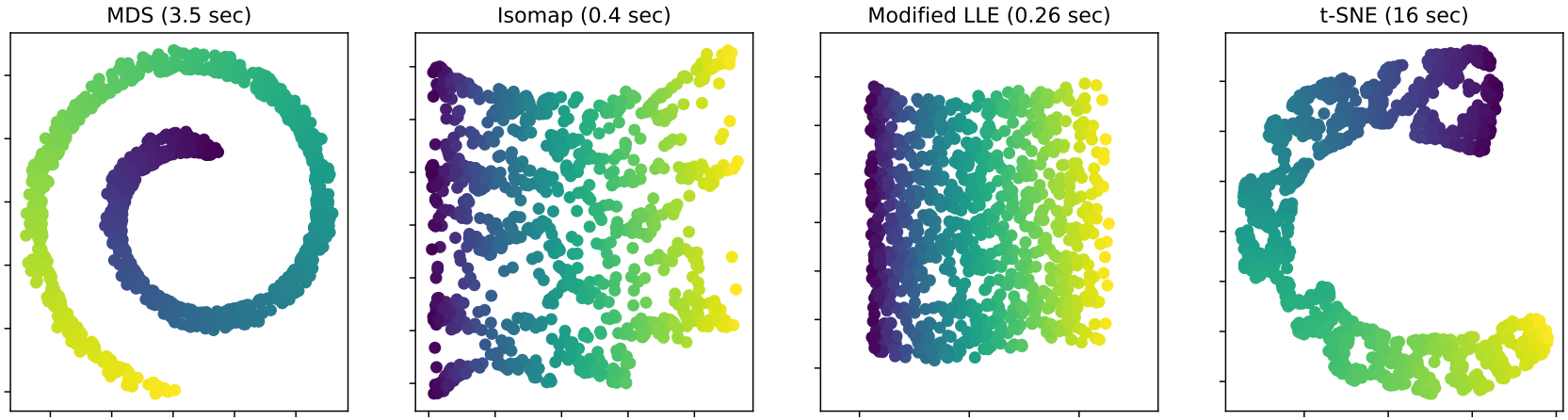
Uniform Manifold Approximation and Projection (UMAP)

- McInnes & Healy, 2018.
- Novel manifold learning technique for dimension reduction.
- Theoretical framework based in Riemannian geometry and algebraic topology.
- Results comparable to t-SNE, but faster.
- Python code in `example_5_umap_digits.ipynb`

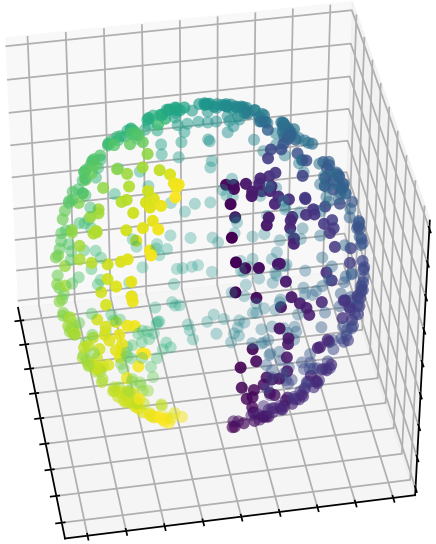
Example: Swiss roll dataset



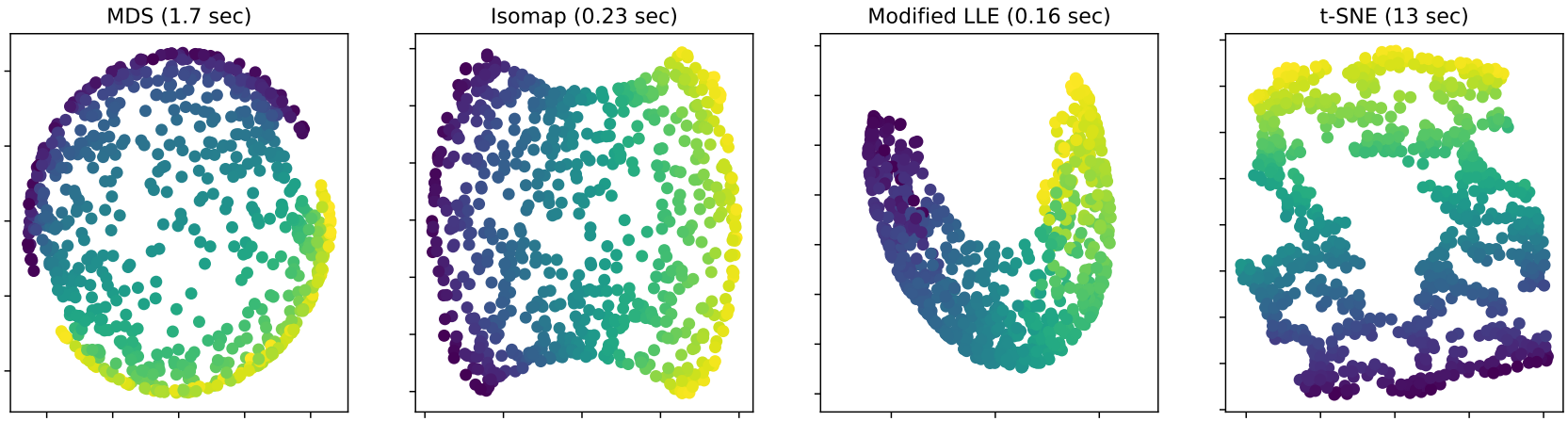
- S-curve dataset.
- Manifold Learning with 1000 points, 10 neighbors.



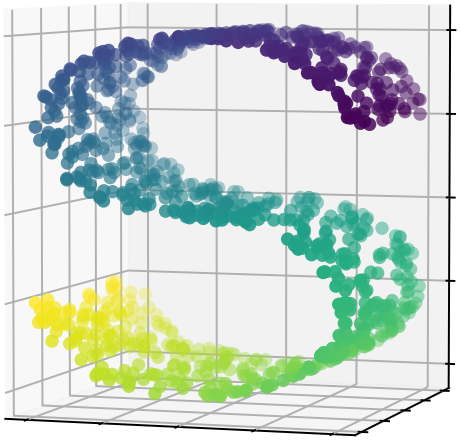
Example: Spherical dataset



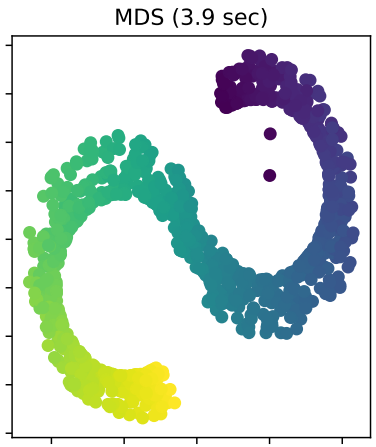
- Spherical data: poles and one slice removed.
- Manifold Learning with 1000 points, 10 neighbors.

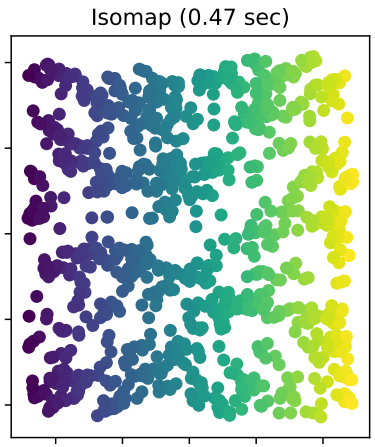


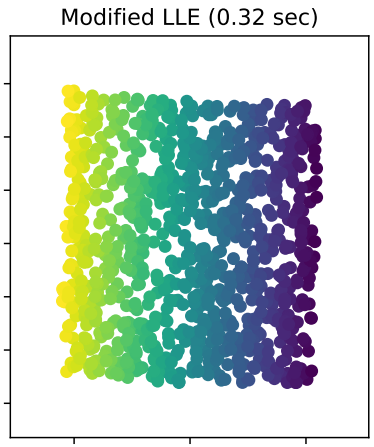
Example: S-curve dataset

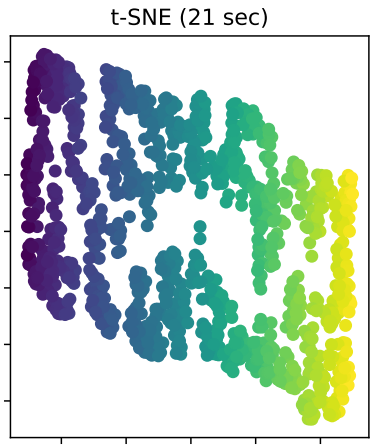


- S-curve dataset.
- Manifold Learning with 1000 points, 10 neighbors.









Interactive examples

- Isomap: `https://plot.ly/~empet/14345.embed`
- t-SNE: `https://distill.pub/2016/misread-tsne/`

Comparison: Objective

What does each technique try to preserve?

- PCA: linear structure
- MDS: global geometry
- Isomap: geodesic distances (globally)
- LLE: local translations, rotations, and scalings
- t-SNE: topology (neighborhood structure)

Comparison: Computational cost

Execution times on 1000 points of toy data:

- PCA: 0.1 s
- MDS: 3 s
- Isomap: 0.2 s (Landmark Isomap)
- LLE: 0.2 s (Modified LLE)
- t-SNE: 15 s

Comparison: Restrictions

- PCA: Unable to discover nonlinear structure.
- MDS: Requires selection of a meaningful distance.
- Isomap: Topological instability (affected by noise).
- LLE: Difficulties to treat manifolds with holes (also the case for Isomap); Requires dense manifold.
- t-SNE: Slow; hard to interpret; different convergences.

Comparison: Guidelines

- PCA: To decorrelate data; to discover linear structure.
- MDS: We are given only a distance matrix.
- Isomap: Data is on a well-connected manifold.
- LLE: Manifold is approx. linear on a local scale.
- t-SNE: To visualize complex real-world data.

Other nonlinear dimensionality reduction techniques

- **Kernel PCA**: nonlinear transformation of data into high-dimensional feature space, then apply PCA in that space.
- Linear Discriminant Analysis (**LDA**): Supervised dimensionality reduction that maximizes class separation.

Conclusions

- Reducing the input data dimensionality is a fundamental preprocessing step for many ML techniques.
- Related to the selection/extraction of features.
- Linear dimensionality reduction techniques:
  - **PCA**: pre-processing for regression/classification techniques; compression/storage of information.
  - **LDA**: pre-processing in problems of supervised classification.

Conclusions

- Nonlinear dimensionality reduction techniques allow to **reveal structure** that linear methods cannot.
- Higher **computational cost**.
- Often used for **visualization**, data exploration.
- Wide range of techniques: **MDS**, **Isomap**, **LLE**, **t-SNE**, **UMAP**, etc.
- Related: Kernel-based dimensionality reduction techniques (KPCA, KCCA, etc.).

References

- Christopher J. C. Burges (2010), “Dimension Reduction: A Guided Tour”, Foundations and Trends in Machine Learning: Vol. 2: No. 4, pp 275-365.