Dimensionality Reduction: Linear projective techniques

Minería de Datos (M1966)

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

Curso 2024-25

Maśter Universitario Oficial **Data Science**

con el apoyo del

Contents

Data format

Table with rows and columns:

columns
(features)

rows
(instances)

	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**.

columns
(features)

rows
(instances)

	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	~~25.0~~	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

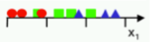
Why reduce dimensionality?

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

Why reduce dimensionality?

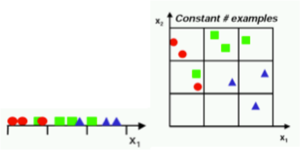
- The volume and the dimensionality of the data $\mathbf{x} \in \mathbb{R}^d$ in machine learning (ML) applications grows constantly.

- 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**

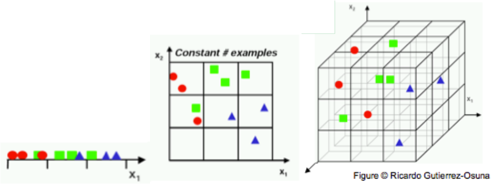
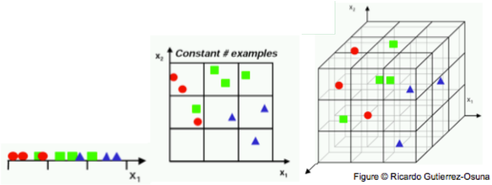


Figure © Ricardo Gutierrez-Osuna

Why reduce dimensionality?

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

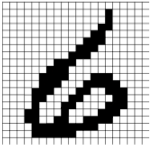
- ML techniques are not effective in high-dimensional spaces $\rightarrow$ **Curse of Dimensionality**



Examples: Datos biomédicos, datasets de IoT (p.ej. series temporales de sensores), etc.

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.

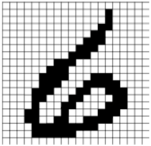


$\Longrightarrow$

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

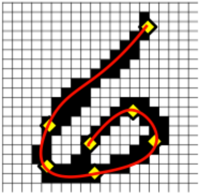
Why reduce dimensionality?

- 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**: Dimensionality reduction improves the convergence of regression algorithms. (It reduces multicollinearity.)

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.

PCA introductory 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	

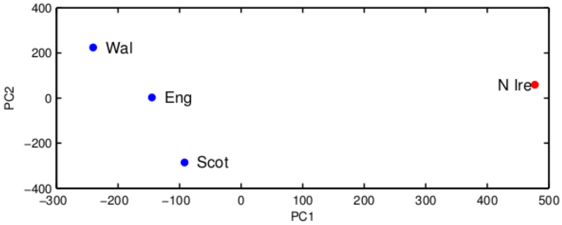
Are any countries similar? How do we visualize these data?

[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 $\mathbf{y}_i = \mathbf{P}_r^T \mathbf{x}_i$

Why does PCA look for the direction of maximum variance?

- PCA seeks a line (in 1D) or a subspace (in higher dimensions) onto which the data “spreads out” as much as possible when projected.
- In mathematical terms, projecting onto a direction $\mathbf{u}$ means taking $y_i = \mathbf{u}^T \mathbf{x}_i$. We want this set of y_i values to have the largest possible variance.
- If $\mathbf{u}$ a unit vector that aligns with the greatest spread of the data, then $\sum_{i=1}^{n} y_i^2$ is maximized. For any other vector $\mathbf{v}$ not in the same direction, the variance becomes smaller.

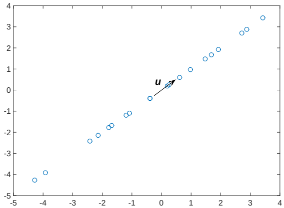
PCA: Introduction

Example: Direction of maximum variance

- Given a data set $\mathbf{x}_i \in \mathbb{R}^d$ ($i = 1, \ldots, n$) 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$\in \mathbb{R}^d$. In other words, all data points $\mathbf{x}_i$ lie on a line spanned by a unit vector $\mathbf{u}$.



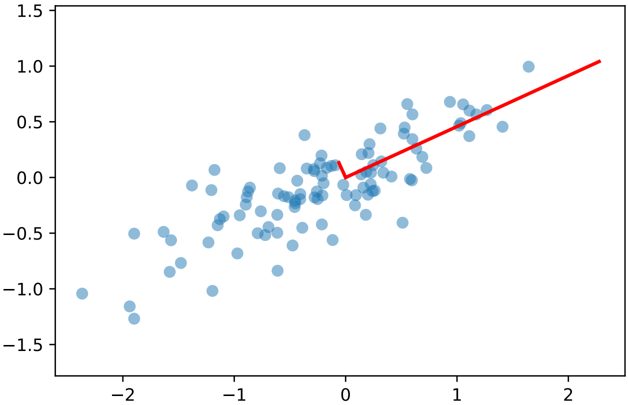
- The projection of $\mathbf{x}_i$ onto $\mathbf{u}$ is $y_i = \mathbf{u}^T\mathbf{x}_i = \theta_i$.
- The variance of the entire projected data set is

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

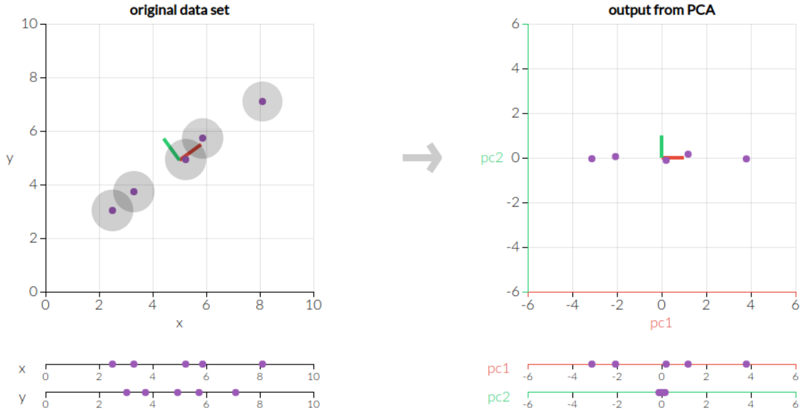
- If we project the $\mathbf{x}_i$ onto any other unit norm vector $\mathbf{v} \neq \mathbf{u}$, the variance will be smaller

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

Example: 2D data

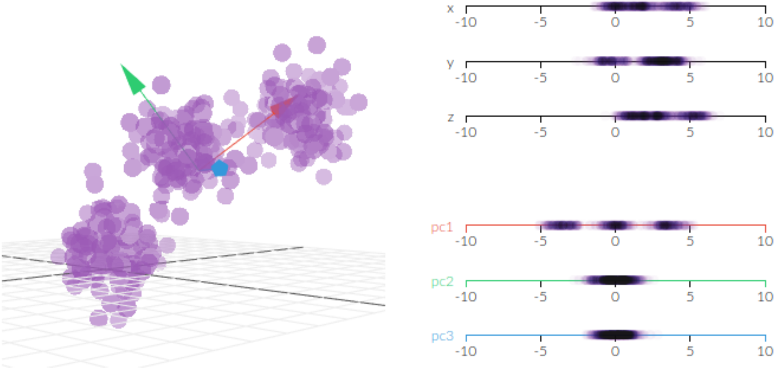


2D example



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

3D example



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

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

This measures how “spread out” the data is along the vector **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 $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_d^2)$ 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 $\hat{\mathbf{C}}_x$

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

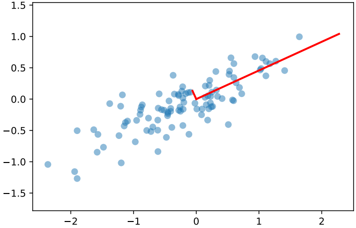
- The **first principal component** (1D projection) is

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

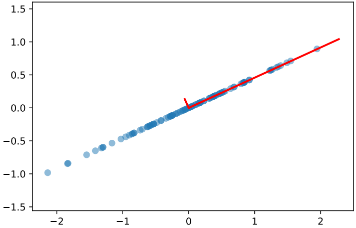
- The proportion of **variance retained** by y_i is $\frac{\sigma_1^2}{\sum_{i=1}^{d} \sigma_i^2}$

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 retained by y_i is now $\frac{\sum_{i=1}^{r} \sigma_i^2}{\sum_{i=1}^{d} \sigma_i^2}$.

Extension to r principal components

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

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

- 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 retained by y_i is now $\frac{\sum_{i=1}^r \sigma_i^2}{\sum_{i=1}^d \sigma_i^2}$.
- We often choose r so that we retain, e.g., 90 % or 95 % of total variance.
- We can approximate the original data via $\hat{\mathbf{X}} = \mathbf{Y}\mathbf{U}_r^T$.

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

$$
\begin{aligned}
E\left[\mathbf{y}_i \mathbf{y}_i^T\right] \; &= E\left[\mathbf{U}_r^T \mathbf{x}_i \mathbf{x}_i^T \mathbf{U}_r\right] = \mathbf{U}_r^T E\left[\mathbf{x}_i \mathbf{x}_i^T\right] \mathbf{U}_r = \\
&\mathbf{U}_r^T E\left[\mathbf{x}_i \mathbf{x}_i^T\right] \mathbf{U}_r = \mathbf{U}_r^T \mathbf{U} \boldsymbol{\Sigma} \mathbf{U}^T \mathbf{U}_r = \\
&\boldsymbol{\Sigma}_r = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_r^2)
\end{aligned}
$$

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

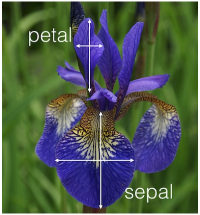
$$\underset{\mathbf{M}}{\text{minimize}} \sum_{i=i}^{n} ||\mathbf{x}_i - \tilde{\mathbf{x}}_i||^2 = ||\mathbf{X} - \mathbf{X}\mathbf{M}\mathbf{M}^T||_F^2 \qquad s.t. \quad \mathbf{M}^T\mathbf{M} = \mathbf{I}_r$$

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

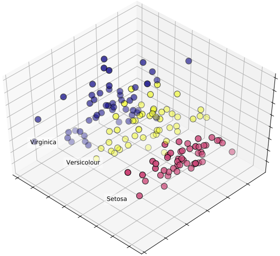
Example: PCA dimensionality reduction

Iris data set:

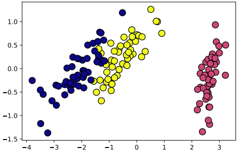
- 3 classes;
- 150 data;
- 4 features.



3 principal components:



2 principal components:

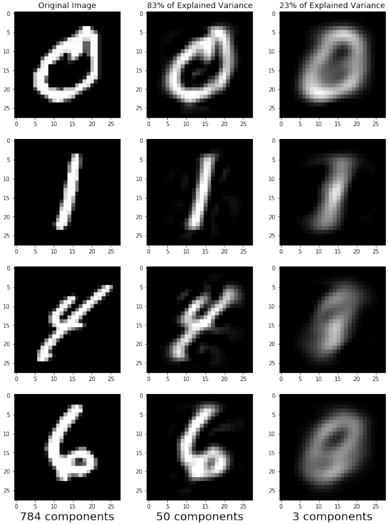


Example: Compression/reconstruction with PCA

MNIST dataset:

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

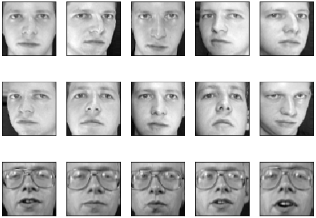
Reconstructions:



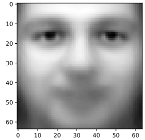
Example: Compression/reconstruction with PCA

Olivetti Faces dataset:

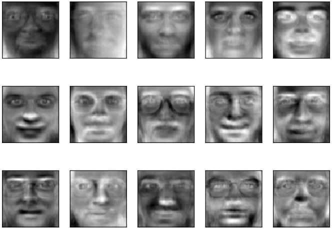
- 40 classes;
- 400 images;
- 4096 pixels.



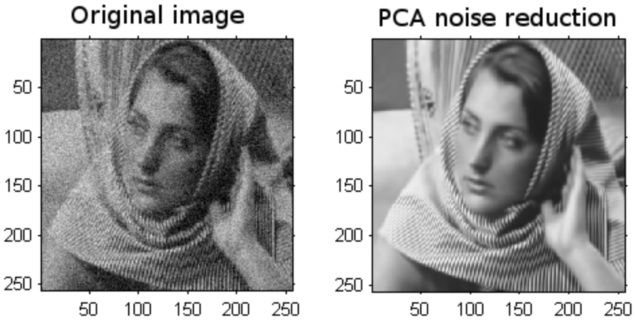
Average (“Mean face”):



Principal directions (“eigenfaces”):



Example: Noise reduction with PCA

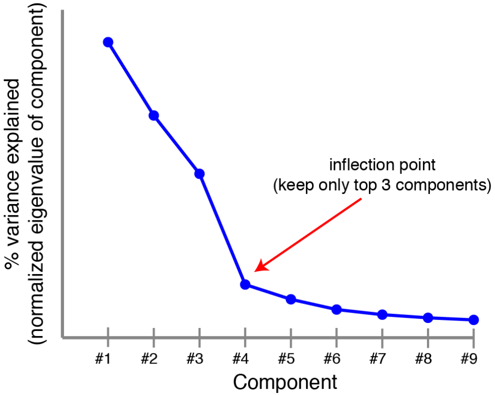


Using patches of 7x7 pixels.

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

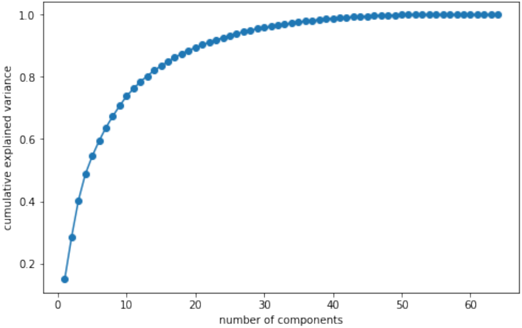
Order estimation 1: Scree plot

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.

Probabilistic PCA (PPCA)

- So far we have not considered any **probabilistic model** for the input data $\mathbf{x}_i$
- Assuming a probabilistic model can help in estimating the most appropriate order or dimension r
- PPCA (Tipping& Bishop 1999) considers the following generative model of the data

$$\mathbf{x}_i = \mathbf{W}\mathbf{f} + \mathbf{e}$$

with

- $\mathbf{f} \sim N(0, \mathbf{I}_r)$ a vector of r uncorrelated Gaussian factors (iid)
- $\mathbf{W} \in \mathcal{R}^{d \times r}$ a deterministic weight matrix
- $\mathbf{e} \sim N(0, \sigma^2 \mathbf{I}_d)$ a vector of Gaussian iid errors, independent of the factors $\mathbf{f}$

- With the above model the observations are also Gaussian $\mathbf{x}_i \sim N(0, \mathbf{C}_x)$ with covariance matrix

$$\mathbf{C}_x = \mathbf{W}\mathbf{W}^T + \sigma^2 \mathbf{I}_d$$

- From a set of n data we estimate the sample covariance as $\hat{\mathbf{C}}_x = \mathbf{U}\Sigma\mathbf{U}^T$
- The maximum likelihood (ML) estimates of the parameters of the PPCA model are as follows

$$\begin{aligned} \hat{\mathbf{W}} &= \mathbf{U}_r \left[\Sigma_r - \hat{\sigma}^2 \mathbf{I}_r\right]^{1/2} \mathbf{Q} \\ \hat{\sigma}^2 &= \frac{1}{d-r} \sum_{i=r+1}^{d} \sigma_i^2 \end{aligned}$$

where $\mathbf{Q}$ is an arbitrary orthogonal matrix
- The factors (principal components) can be estimated as follows

$$\hat{\mathbf{f}}_i = \hat{\mathbf{W}}^T \left(\hat{\mathbf{W}}\hat{\mathbf{W}}^T + \hat{\sigma}^2 \mathbf{I}_d\right)^{-1} \mathbf{x}_i$$

Factor Analysis (FA)

- Spearman 1904
- Model similar to that assumed in PPCA

$$\mathbf{x}_i = \mathbf{W}\mathbf{f} + \mathbf{e}$$

with the only difference being that the noisess $\mathbf{e} \sim N(0, \mathbf{E})$ can now have different variances in each dimension.

$$\mathbf{E} = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_d^2)$$

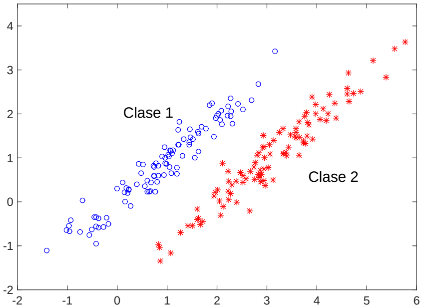
- Iterative algorithms are available to obtain ML estimates of the model parameters.

Python exercises

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

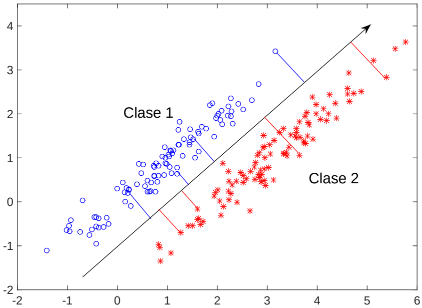
Linear Discriminant Analysis (LDA)

- In (supervised) classification problems, the directions of maximum variance (PCA) do not always give features that allow separation between classes.
- What is the direction of maximum variance in this binary classification problem?

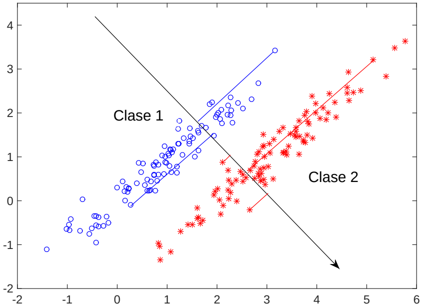


Linear Discriminant Analysis (LDA)

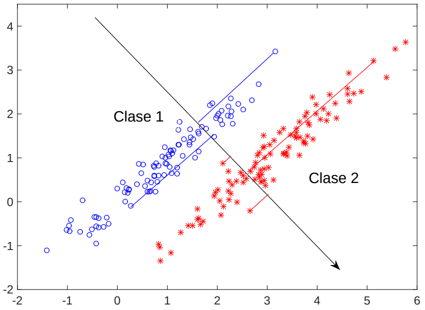
- In (supervised) classification problems, the directions of maximum variance (PCA) do not always give features that allow separation between classes.
- What is the direction of maximum variance in this binary classification problem?



- A better direction would be

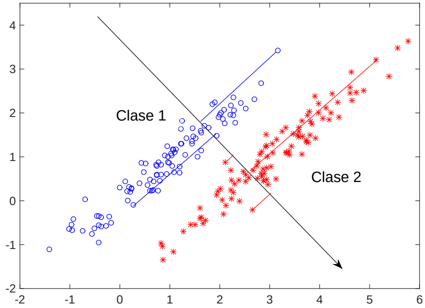


- A better direction would be



- What is the criterion for finding a projection $y = \mathbf{w}^T\mathbf{x}$ such that the classes are maximally separated?

- A better direction would be



- What is the criterion for finding a projection $y = \mathbf{w}^T \mathbf{x}$ such that the classes are maximally separated?

$$J(\mathbf{w}) = \frac{(m_1 - m_2)^2}{\sigma_1^2 + \sigma_2^2}$$

Problem formulation

LDA or Fisher's Linear Discriminant (Fisher 1936)

LDA obtains the projector $\mathbf{w}$ (of unit norm) that maximizes the function

$$J(\mathbf{w}) = \frac{(m_1 - m_2)^2}{\sigma_1^2 + \sigma_2^2}$$

where (m_i, σ_i^2) are the sample mean and variance of the projected data of the class i

- Assume a **supervised classification problem** binary with d-dimensional patterns $\mathbf{x}_1, \ldots, \mathbf{x}_n$
- Set of patterns of each class: $\mathcal{X}_1$ y $\mathcal{X}_2$
- $|\mathcal{X}_1| = n_1$ and $|\mathcal{X}_2| = n_2$ with $n_1 + n_2 = n$

- We apply a linear projector $\mathbf{w}$ to the data of the two classes

$$y = \mathbf{w}^T \mathbf{x} \in \mathcal{R}$$

- The set of 1D (projected) patterns of each class we denote as $\mathcal{Y}_1$ and $\mathcal{Y}_2$
- The sample mean of the classes in the input and output spaces are

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{i \in \mathcal{X}_i} \mathbf{x}_i, \quad \text{y} \quad m_i = \frac{1}{n_i} \sum_{i \in \mathcal{X}_i} \mathbf{w}^T \mathbf{x}_i = \frac{1}{n_i} \sum_{i \in \mathcal{Y}_i} y_i$$

- The distance between the averages is

$$(m_1 - m_2)^2 = \left(\mathbf{w}^T \mathbf{m}_1 - \mathbf{w}^T \mathbf{m}_2\right)^2 = \mathbf{w}^T (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T \mathbf{w}$$

- Defining the $d \times d$ **between-class scatter matrix**

$$\mathbf{S}_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T,$$

the distance is $(m_1 - m_2)^2 = \mathbf{w}^T \mathbf{S}_B \mathbf{w}$

- The sum of variances in the projected space can be written as

$$\sigma_1^2 + \sigma_2^2 = \frac{1}{n} \mathbf{w}^T \mathbf{S}_W \mathbf{w}$$

where $\mathbf{S}_W$ (**within-class scatter matrix**)

$$\mathbf{S}_W = \left(\sum_{i \in \mathcal{X}_1} (\mathbf{x}_i - \mathbf{m}_1)(\mathbf{x}_i - \mathbf{m}_1)^T + \sum_{i \in \mathcal{X}_2} (\mathbf{x}_i - \mathbf{m}_2)(\mathbf{x}_i - \mathbf{m}_2)^T \right)$$

- Based on the dispersion matrices, the function to be maximized in LDA can be written as follows

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

  which is a generalized Rayleigh quotient.
- Generalized eigenvalue problem

$$\mathbf{S}_B \mathbf{w} = \lambda \mathbf{S}_W \mathbf{w}$$

- The solution for $\mathbf{w}$ is the eigenvector corresponding to the maximum eigenvalue $\mathbf{S}_W^{-1}\mathbf{S}_B$
- In the binary problem under consideration $\mathbf{S}_B$ is of rank one and the solution of the LDA projector is

$$\mathbf{w} = \mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$$

Extensions

- We can consider more projections by taking more eigenvectors (in this case the projectors are not orthogonal).
- The extension to the multi-class case with C classes is immediate by defining the inter-class scattering matrix as

$$\mathbf{S}_B = \sum_{j=1}^{C} n_j (\mathbf{m}_j - \mathbf{m})(\mathbf{m}_j - \mathbf{m})^T$$

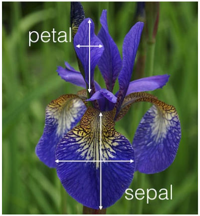
where $\mathbf{m}$ is the sample mean of all data, and the intra-class dispersion matrix is

$$\mathbf{S}_W = \sum_{j=1}^{C} \sum_{\mathbf{x} \in \mathcal{X}_j} (\mathbf{x} - \mathbf{m}_j)(\mathbf{x} - \mathbf{m}_j)^T$$

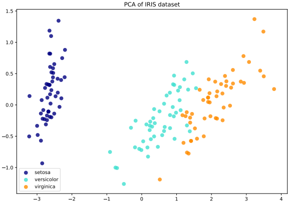
Example: PCA vs LDA

Iris data set:

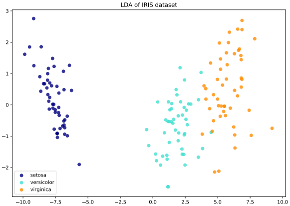
- 3 classes;
- 150 data;
- 4 features.



PCA projection„
2 main components:



LDA projection:



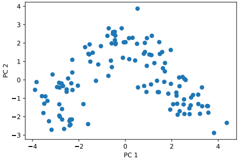
Ejemplo: LDA in supervised classification

Wine dataset:

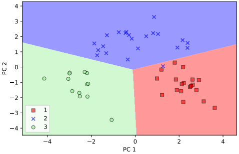
- 3 classes;
- 178 data;
- 14 features.

	Class label	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	OD280/OD315 of diluted wines	Proline
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735

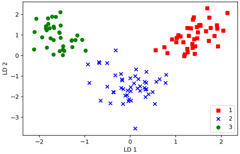
PCA projection, 2 princ. comp.:



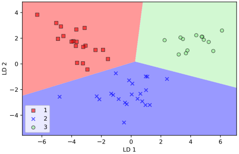
Log. regression (boundaries):



LDA projection, 2 components:



Log. regression (boundaries):



Conclusions

- Dimensionality reduction is a fundamental preprocessing step for many ML techniques.
- Related to the selection/extraction of features.

Linear dimensionality reduction techniques:

- **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.

References

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