Unsupervised Learning

Steven Van Vaerenbergh

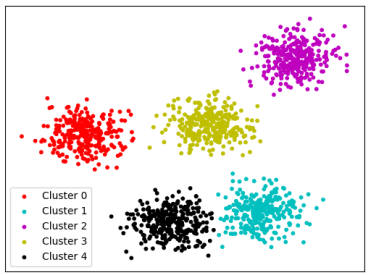
Universidad de Cantabria

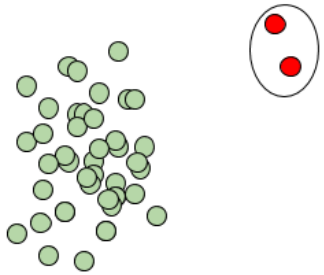
July 2021

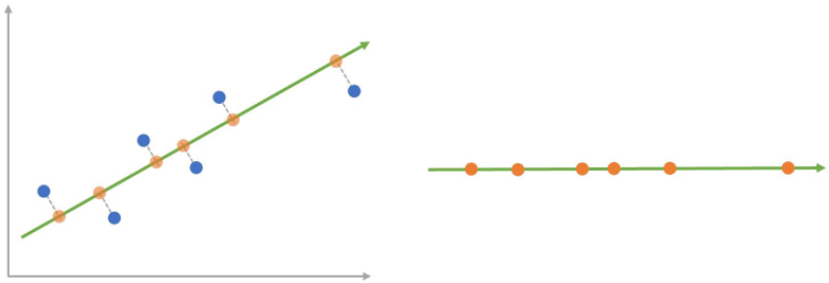
Unsupervised learning

Machine learning techniques that learn from unlabeled data.

- Clustering
- Outlier detection
- Dimensionality reduction
- ...







Data format

Table with rows and columns (Pandas dataframe):

	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

Unsupervised learning: input data without labeled responses.

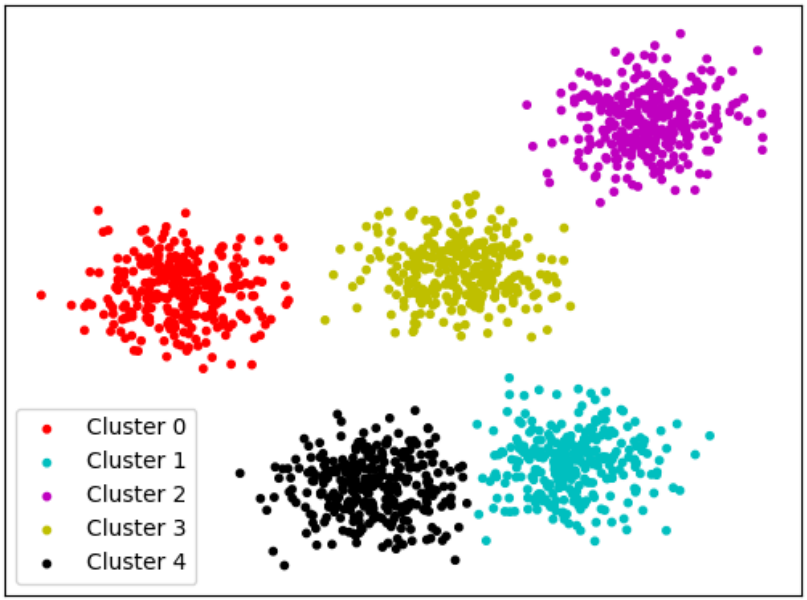
Software

- Python 3
- Algorithms included in scikit-learn

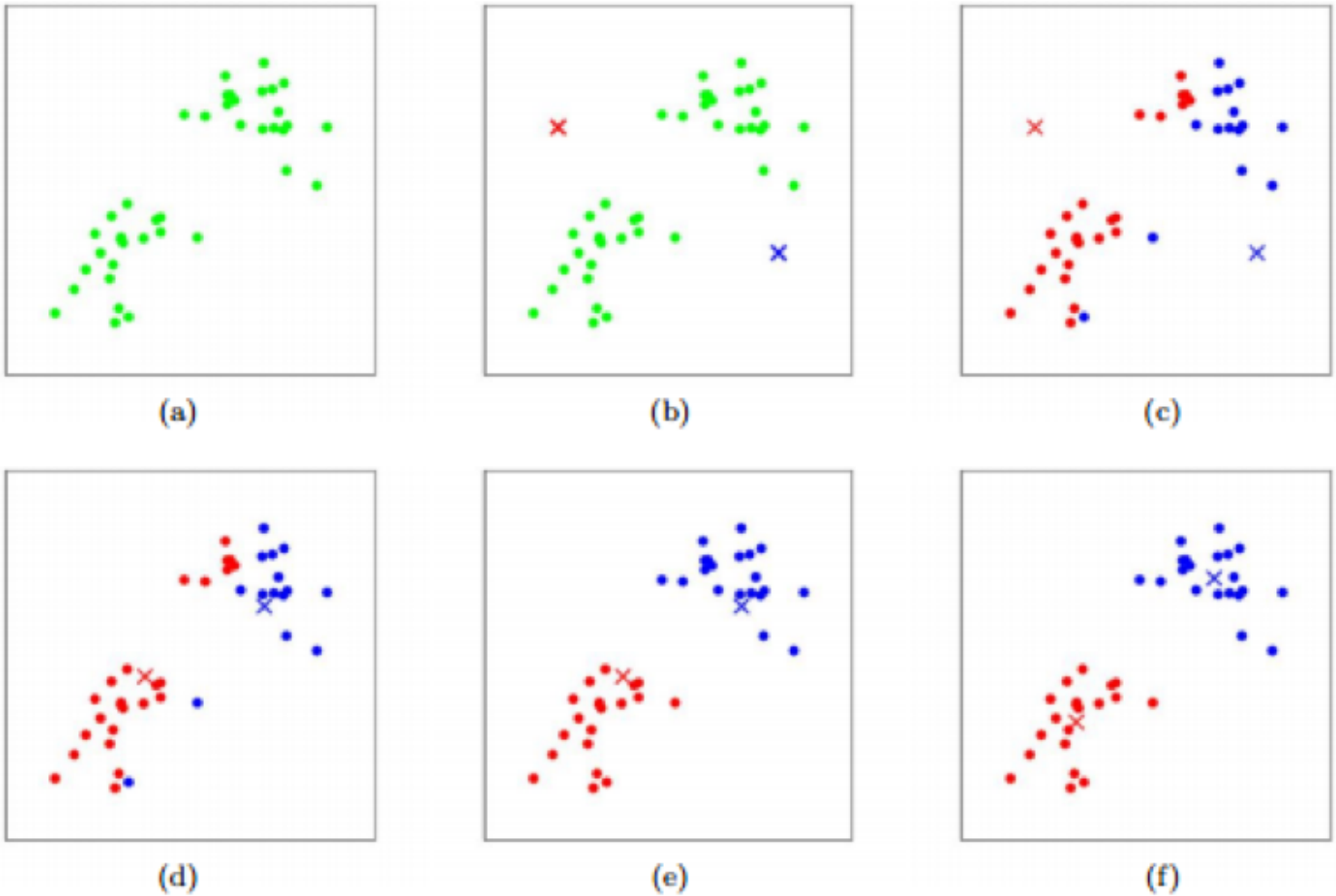
Part 1: Clustering

Clustering

Goal: to group the datapoints of a dataset into disjoint sets, such that points within one group are similar and the points in different groups are dissimilar, according to a given similarity or distance measure.



Clustering algorithm 1: K-Means



1 parameter: number of clusters k.

K-means: algorithm

1. Random initialization of centroids $\boldsymbol{\mu}_j$, $(j = 1, \ldots, k)$, $\mathbb{R}^d$
2. **Assign patterns to clusters/centroids:** assign each pattern $\mathbf{x}_n$ to its closest centroid

$$\mathbf{x}_n \in \mathcal{C}_i, \quad i = \underset{j=1,\ldots,k}{\operatorname{argmin}} \|\mathbf{x}_n - \boldsymbol{\mu}_j\|_2^2$$

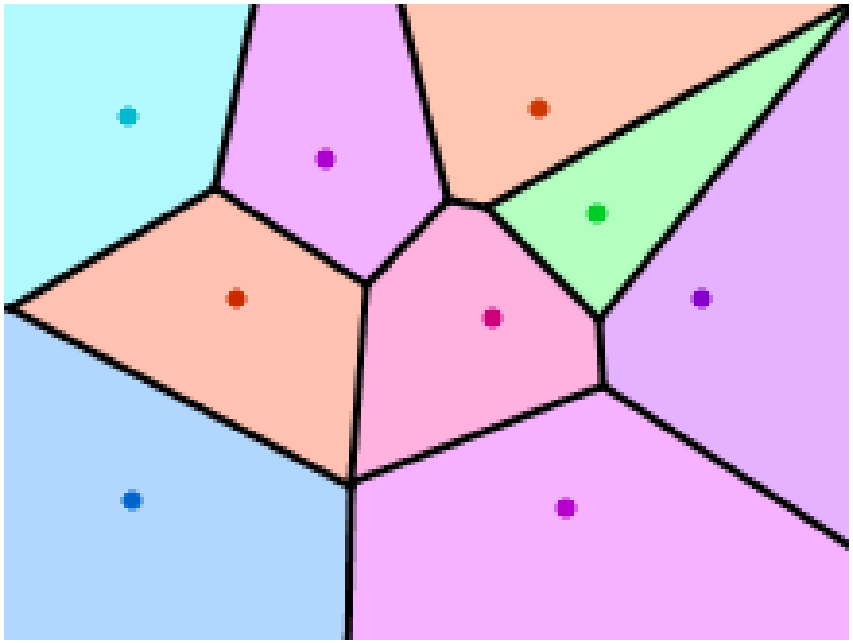
3. **Update centroids** as

$$\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{\mathbf{x}_n \in \mathcal{C}_i} \mathbf{x}_n$$

Repeat steps 2 and 3. Converges to a (local) minimum.

K-Means

- The most widely used clustering method
- Input: Data $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, and **number of clusters** k
- Output: k centroids $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k \in \mathbb{R}^d$
- Centroids divide the input space into k disjoint Voronoi regions (clusters)



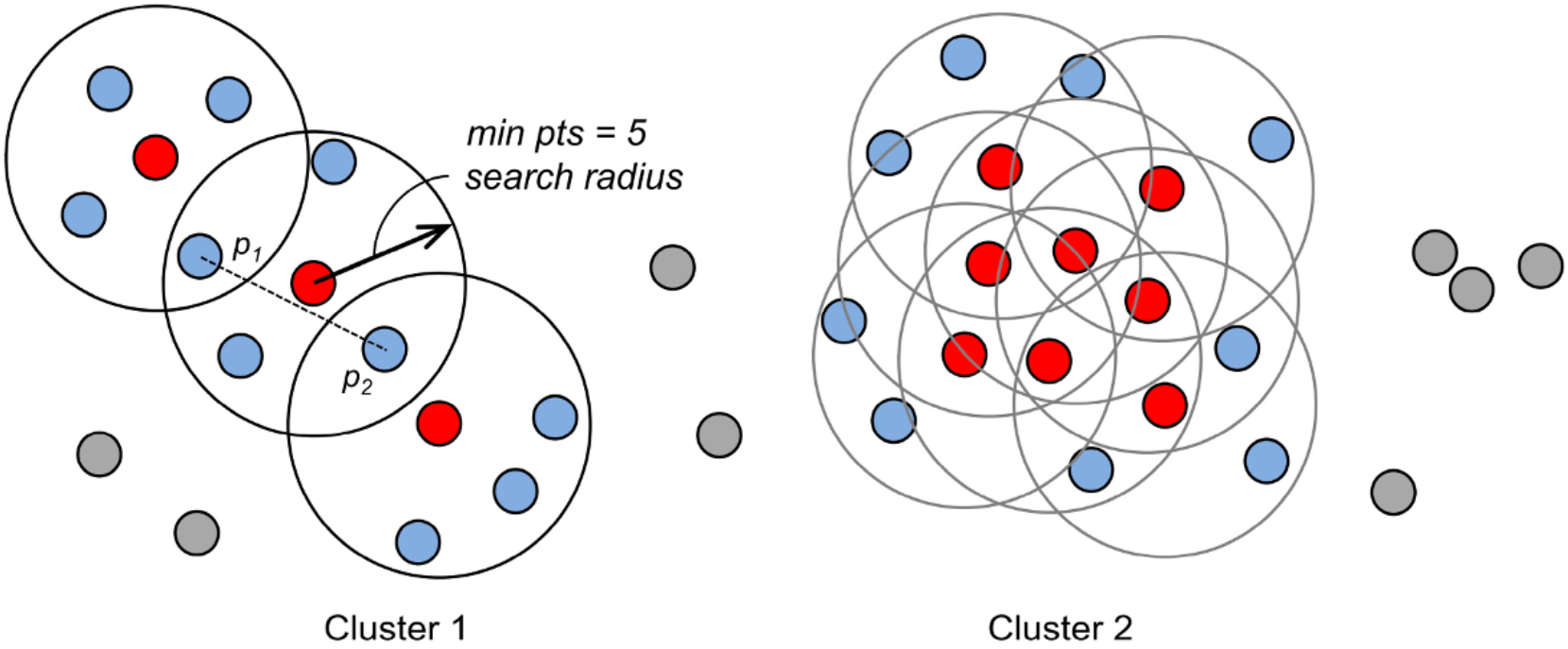
K-means: Additional details

- The clustering problem is to find the optimal centroids that minimize a distortion criterion

$$D(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k) = \sum_{j=1}^{k} \sum_{\mathbf{x}_n \in \mathcal{C}_j} \|\mathbf{x}_n - \boldsymbol{\mu}_j\|_2^2$$

- Each cluster $\mathcal{C}_j$, is defined by its corresponding centroid $\boldsymbol{\mu}_j$
- To solve the problem we have to:
  - Assign patterns to clusters $\mathbf{x}_n \rightarrow \mathcal{C}_j$
  - Estimate centroids $\boldsymbol{\mu}_j$
- There is no closed-form solution, so we have to resort to iterative algorithms

Clustering algorithm 2: DBSCAN



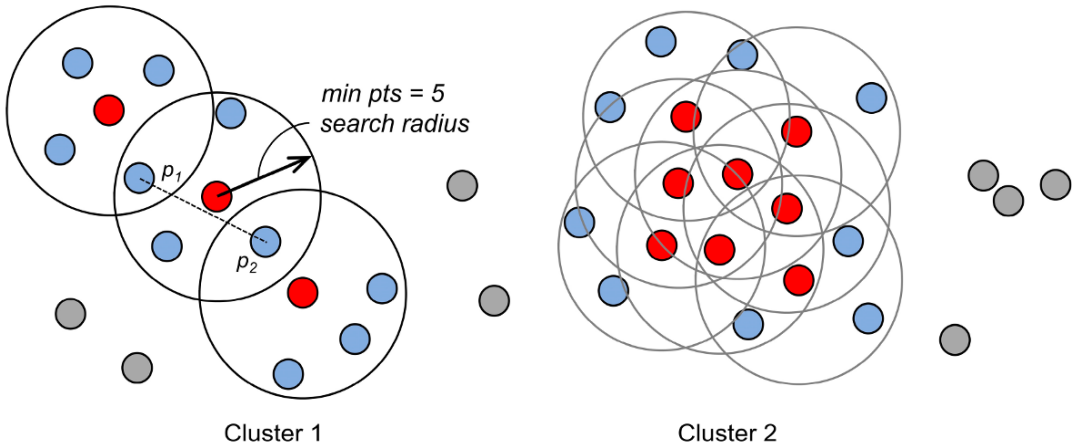
2 parameters: search radius ϵ, min pts.
3 types of points: core, border, noise.

Source: DiFrancesco, Bonneau & Hutchinson (2020). DOI: 10.3390/rs12111885

DBSCAN: Properties

"Density-Based Spatial Clustering of Applications with Noise"

- Core points: in areas of high density
- Clusters = areas of high density separated by low density
- Take a core point, find all of its neighbors that are core points, find all of their neighbors that are core points, etc.



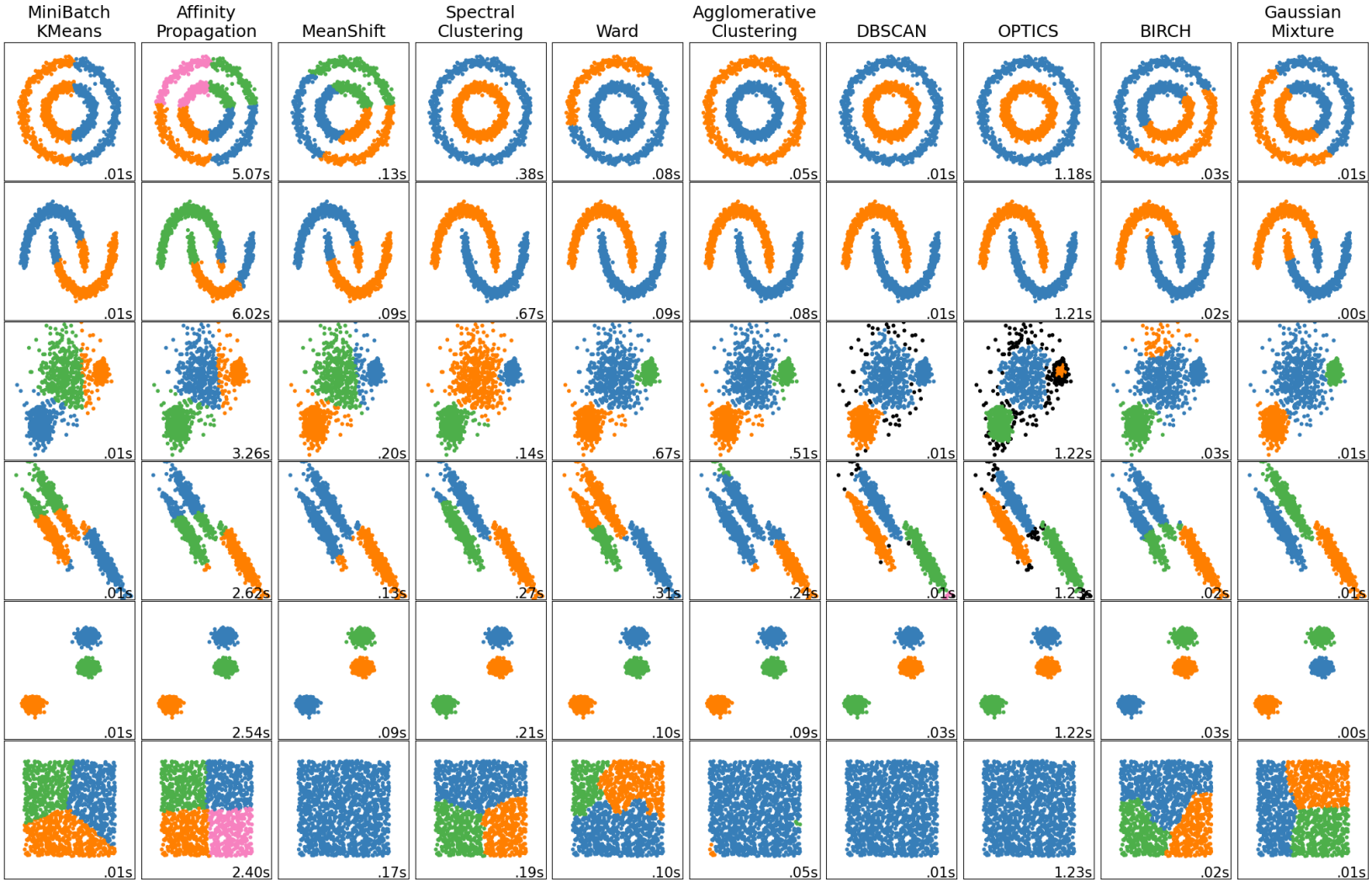
DBSCAN: Properties

- Clusters found by DBSCAN can be any shape.
- Number of clusters not required beforehand.
- Only 2 parameters, though choice is not always obvious.
- Deterministic under some circumstances.
- Difficulties when clusters have different densities.
- May be memory intensive, $\mathcal{O}(n^2)$.

Many more clustering algorithms

- Spectral clustering
- Hierarchical Clustering
- Affinity Propagation
- Mean Shift
- Gaussian Mixtures
- ...

Clustering algorithms: Comparison



Source: `https://scikit-learn.org/stable/modules/clustering.html`

Part 2: Dimensionality Reduction

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.

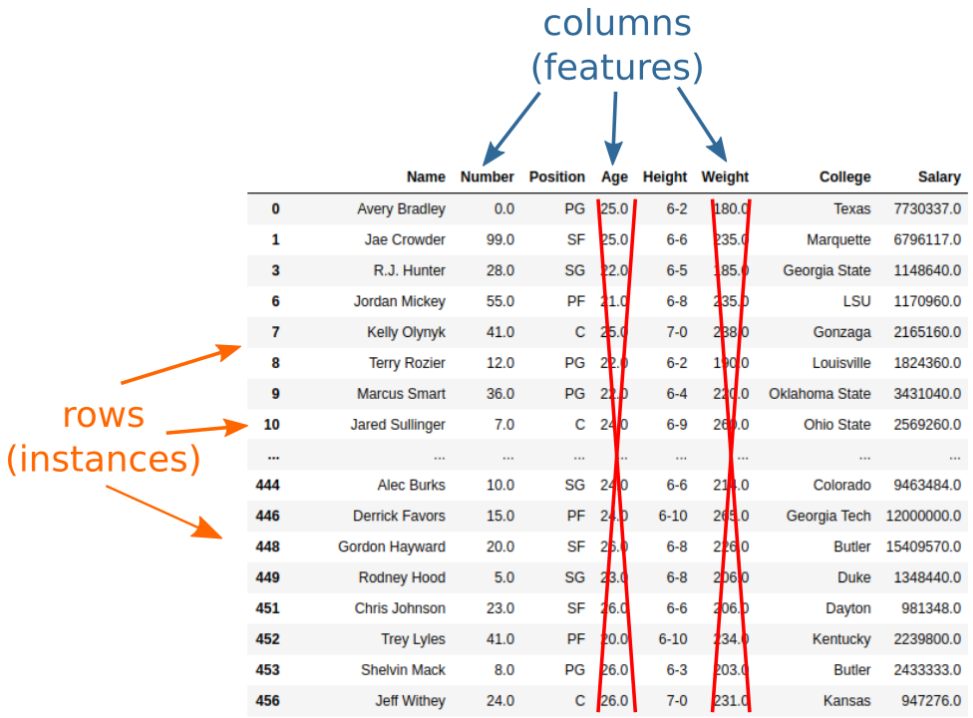
	Cheese	Carcass Meat	Other Meat	Fish	Fats and Oils	Sugars	Fresh potatoes	Fresh Veg	Other Veg	Processed potatoes	Processed Veg	Fresh Fruits	Cereals	Be
England	105	245	685	147	193	156	720	253	488	198	360	1102	1472	
Wales	103	227	803	160	235	175	874	265	570	203	365	1137	1582	
Scotland	103	242	750	122	184	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".

Data format

Selection / 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

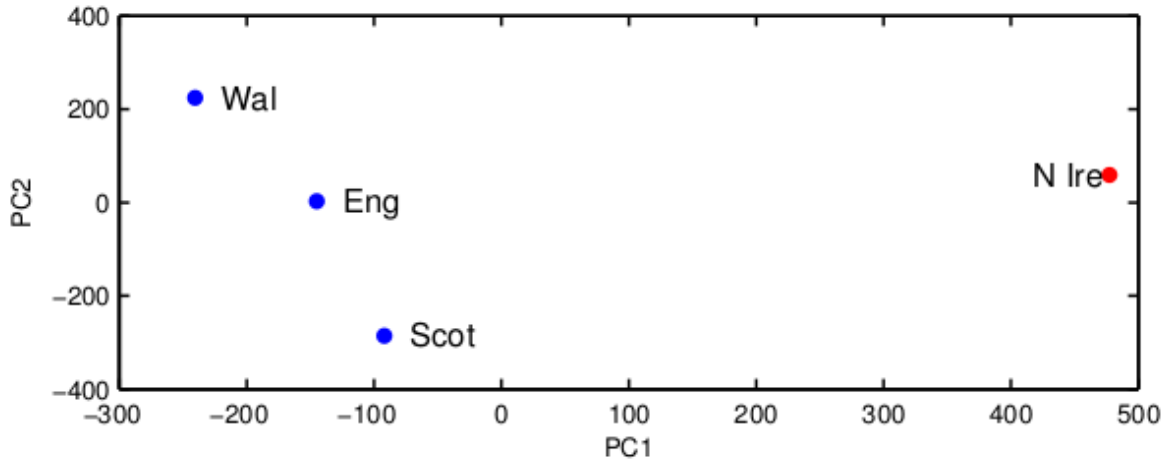
$\Longrightarrow$

	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

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 (preserves as much information as possible).



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

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

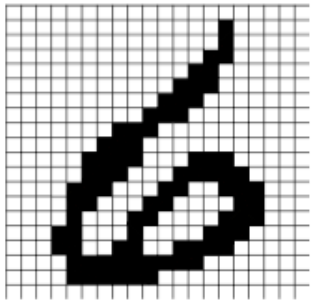
that preserves/optimizes some characteristic of the data (e.g. variance, correlation between data sets, inter-class separation).

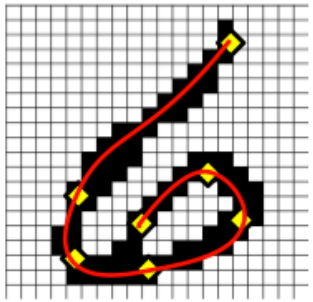
Why reduce dimensionality?

1. **Visualization**: Projection in 2D or 3D space.
2. **Compression**: Reduction of the storage requirements while maintaining the possibility to recover the original data.

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

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





Why reduce dimensionality?

3. **Noise reduction**: Projection onto a subspace or *manifold* in which the data of interest reside.
4. **Convergence**: Lowering the dimensionality improves the convergence of ML algorithms.

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

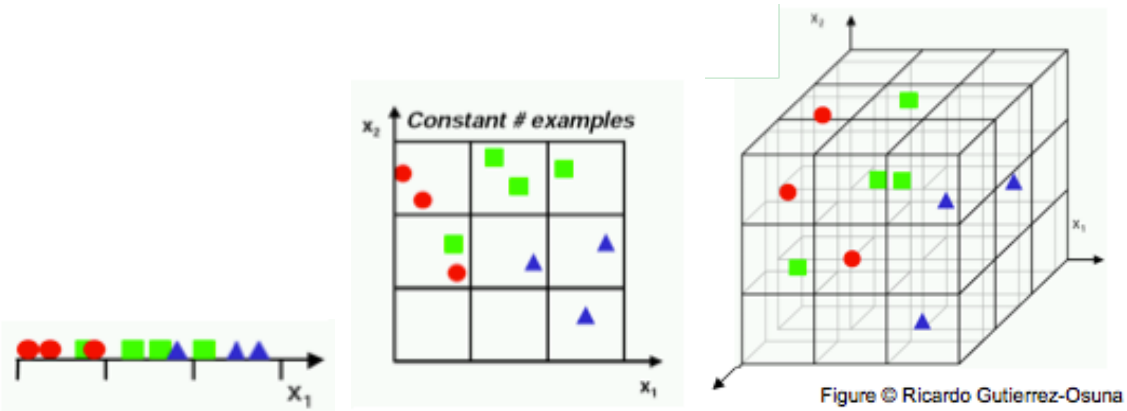
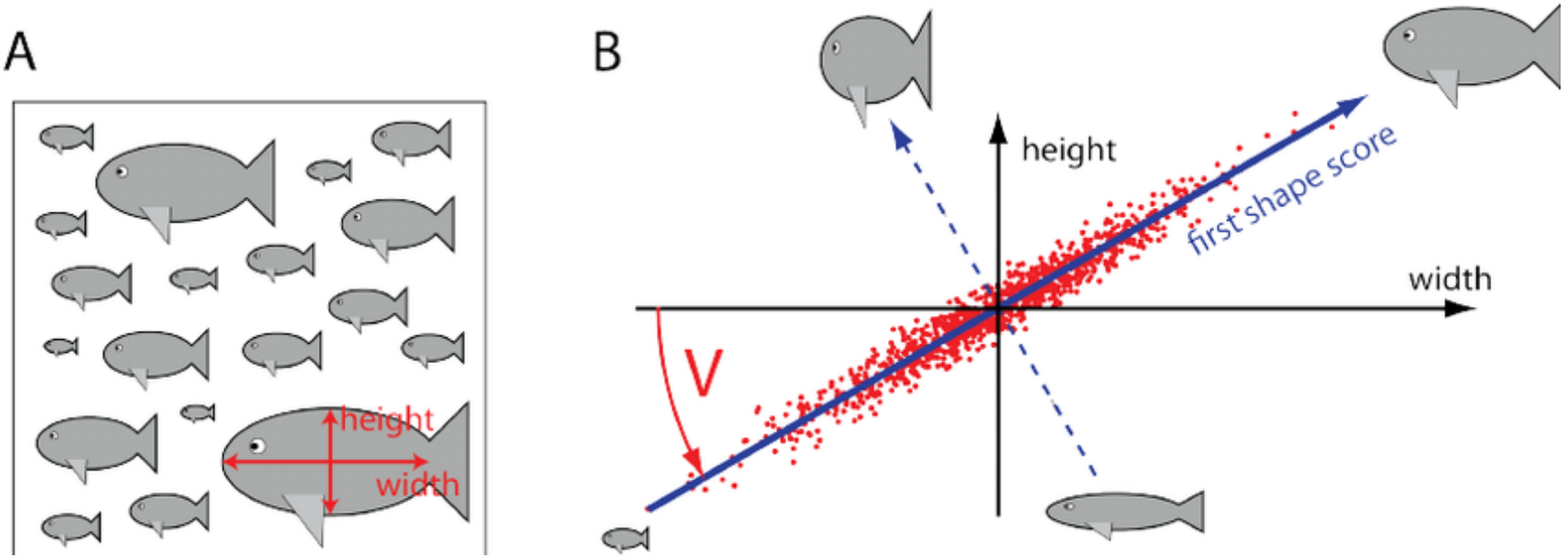


Figure © Ricardo Gutierrez-Osuna

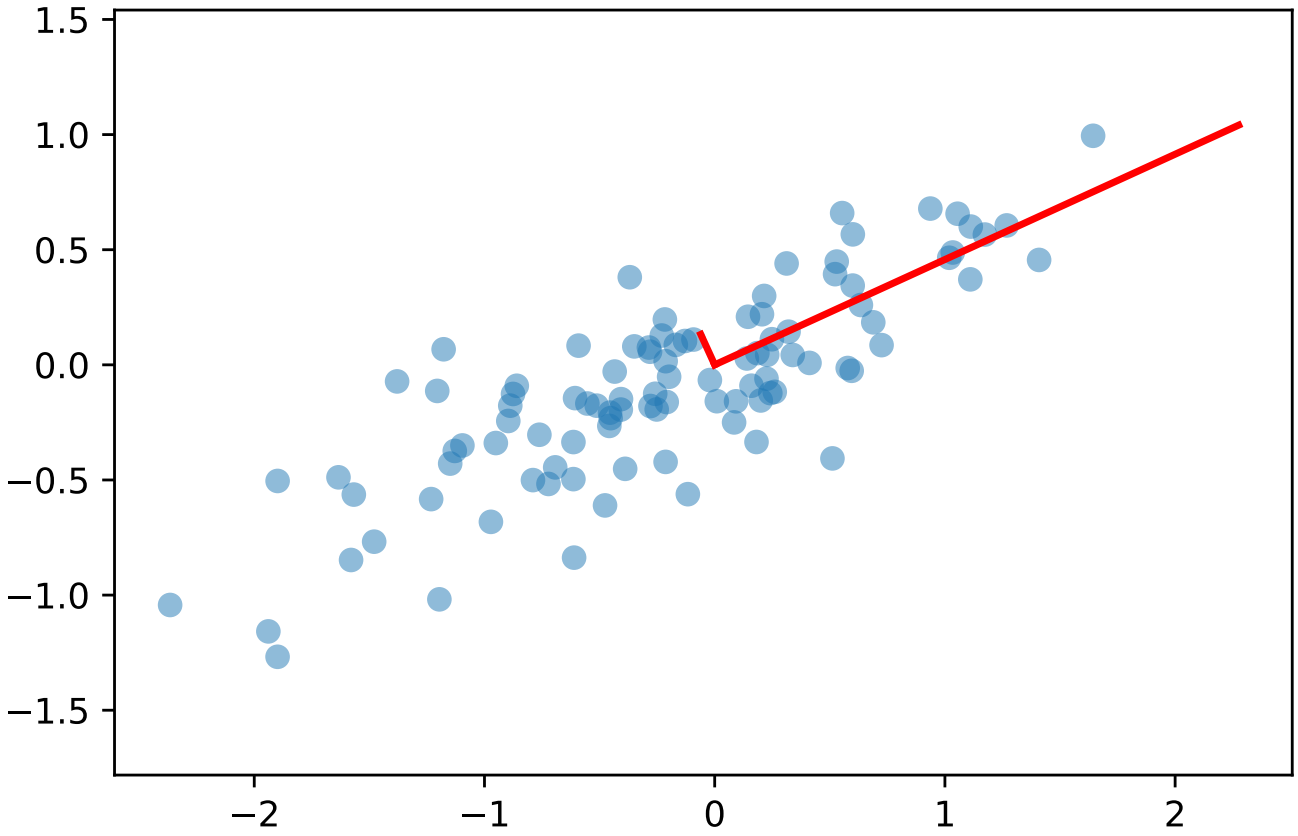
Dimensionality reduction algorithm 1: PCA

PCA: “Principal Component Analysis”



Source: Werner and Friedrich (2014). DOI: 10.1371/journal.pone.0113083

Example: 2D data



PCA

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 $\mathbf{y}_i = \mathbf{P}_r \mathbf{x}_i$

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 $\mathbf{m}_x = \frac{1}{n}\sum_i \mathbf{x}_i$)

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

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

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

- 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**:

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

where $\mathbf{\Sigma} = \text{diag}(\sigma_1^2, \ldots, \sigma_d^2)$ 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 $\hat{\mathbf{C}}_x$

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

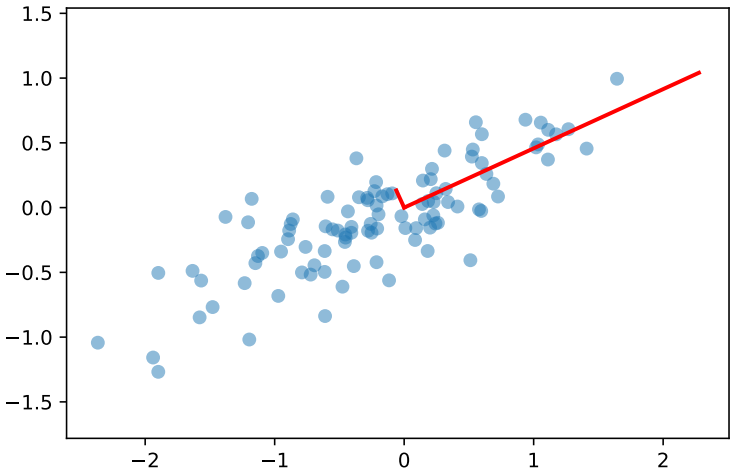
- The **first principal component** (1D projection) is

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

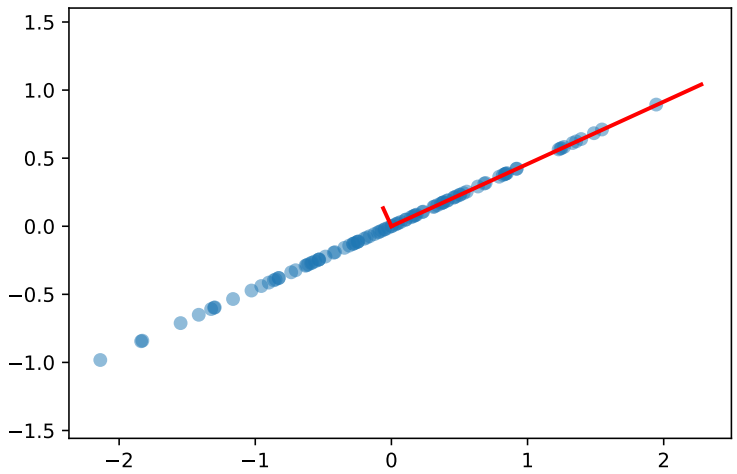
- The proportion of **variance "explained"** by y_i is $\frac{\sigma_1^2}{\sum_{i=1}^{d} \sigma_i^2}$

Example: PCA on 2D data

Data (blue) and principal directions (red).



Projection onto the first principal direction.



Extension to r principal components

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

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

- Data with reduced dimensionality (principal components):

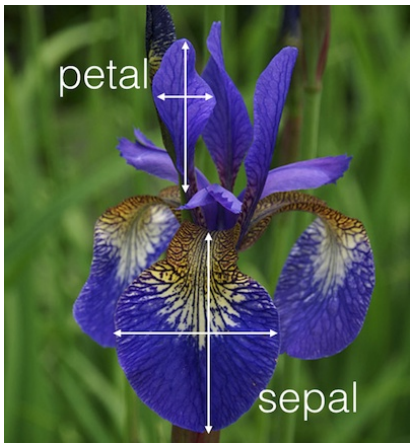
$$\mathbf{y}_i = \mathbf{U}_r^T \mathbf{x}_i \in \mathbb{R}^r \Rightarrow \mathbf{Y} = \mathbf{U}_r^T \mathbf{X}$$

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

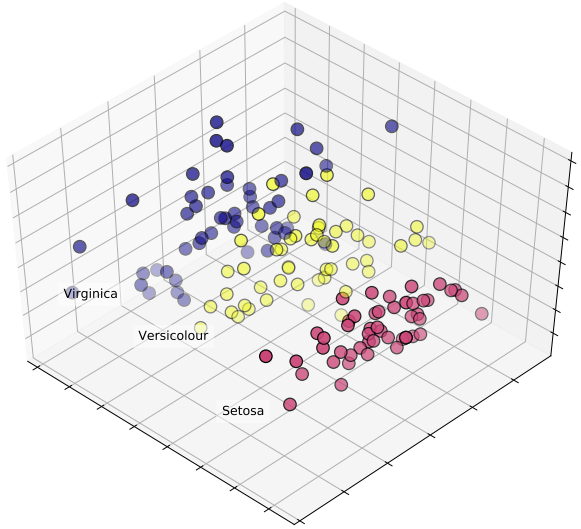
Example: PCA for visualization

Iris dataset:

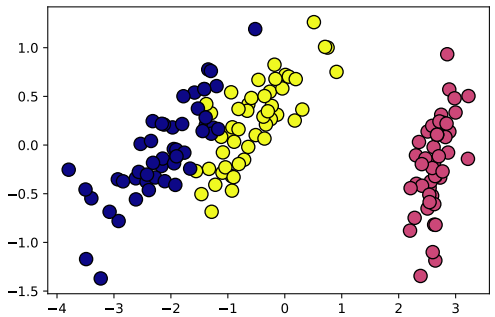
- 3 classes
- 150 data
- 4 features



3 principal components:



2 principal components:

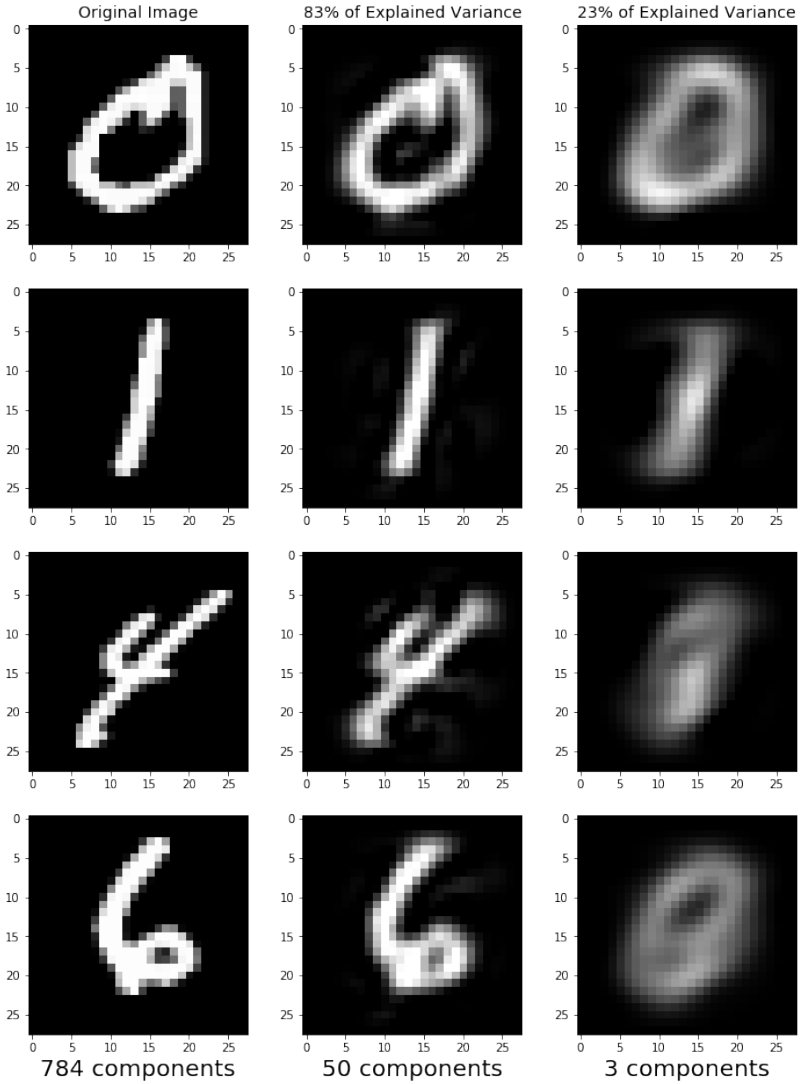


Example: PCA for compression/reconstruction

MNIST dataset:

- 10 classes
- 70000 images
- 784 pixels

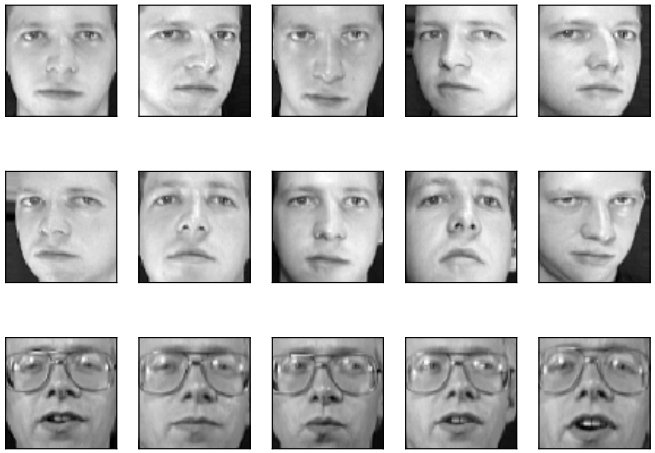
Reconstructions:



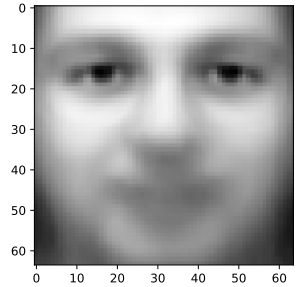
Example: PCA for compression/reconstruction

Olivetti Faces dataset:

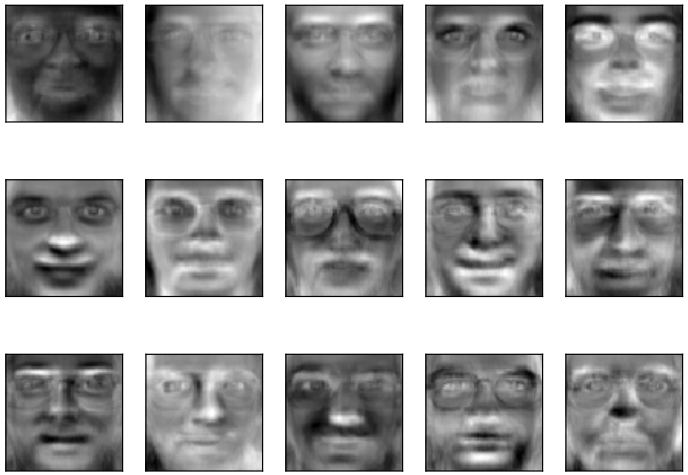
- 40 classes
- 400 images
- 4096 pixels



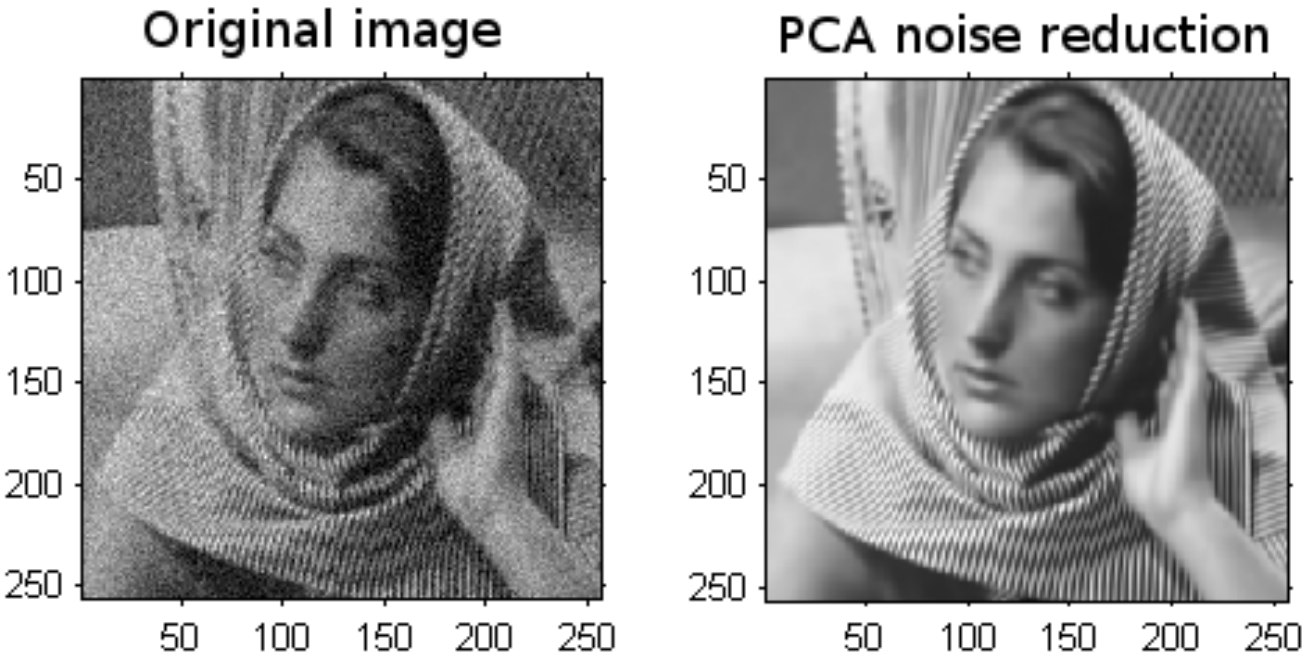
Average (“Mean face”):



Principal directions (“eigenfaces”):



Example: PCA for noise reduction

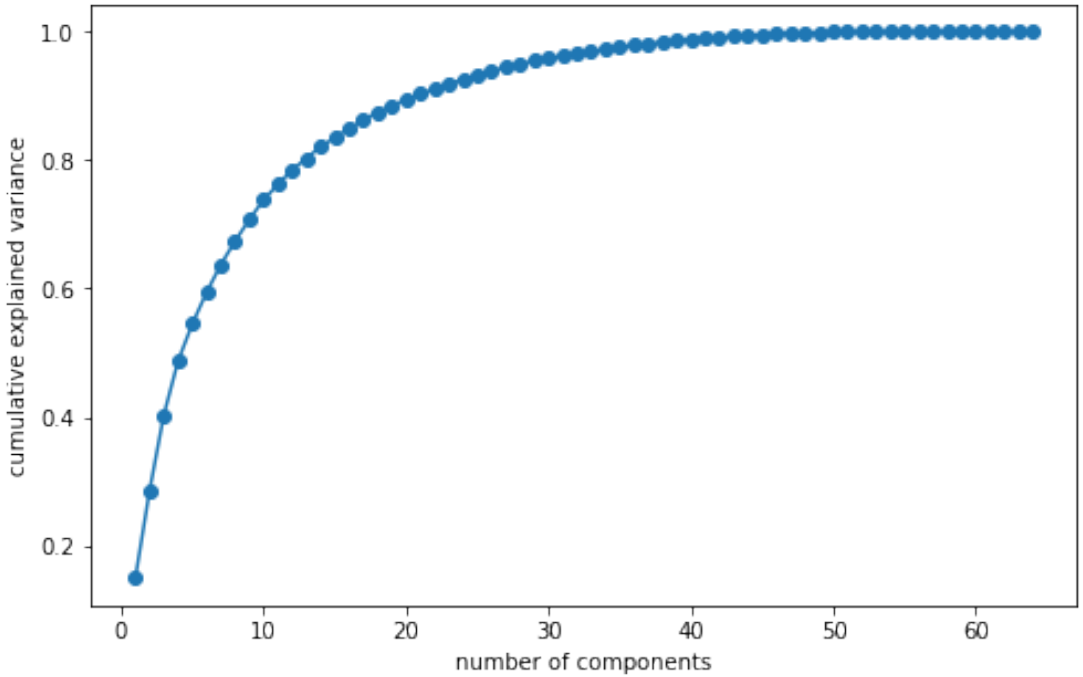


Using patches of 7x7 pixels.

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

How to choose the number of components r?

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.

Dimensionality reduction algorithm 2: t-SNE

Based on 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.

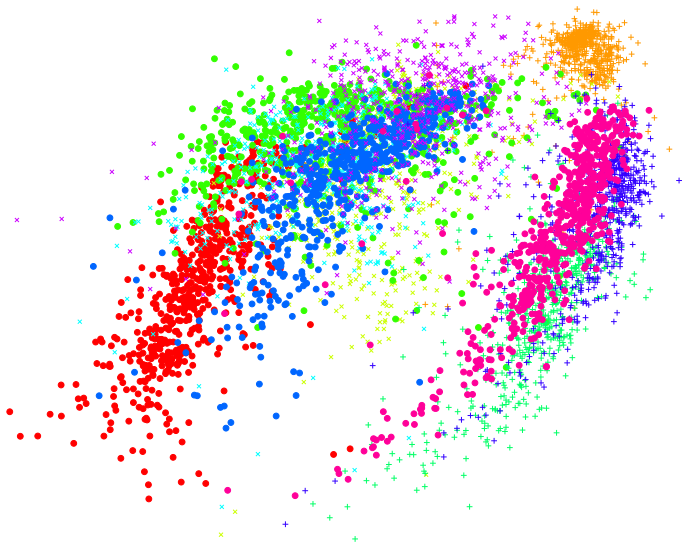
t-distributed SNE (t-SNE)

- Extension of SNE (van der Maaten & Hinton, 2008), 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.
- “How to Use t-SNE Effectively”
  `https://distill.pub/2016/misread-tsne`

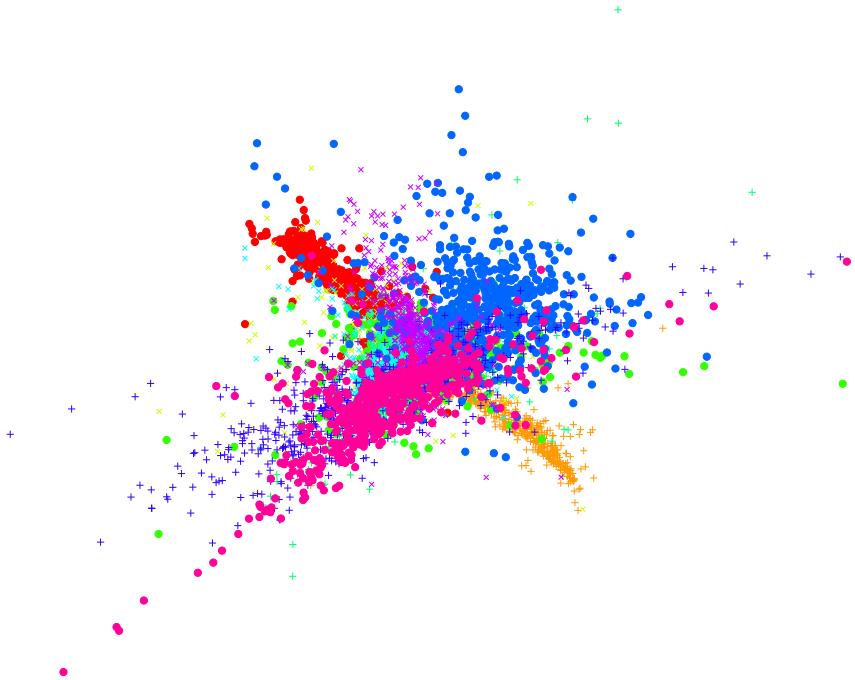
Other dimensionality reduction algorithms

- MDS: Multidimensional Scaling
- Isomap
- LLE: Locally Linear Embedding
- UMAP: Uniform Manifold Approximation and Projection (no incluido en scikit-learn)
- ...

Example: Visualizations of MNIST

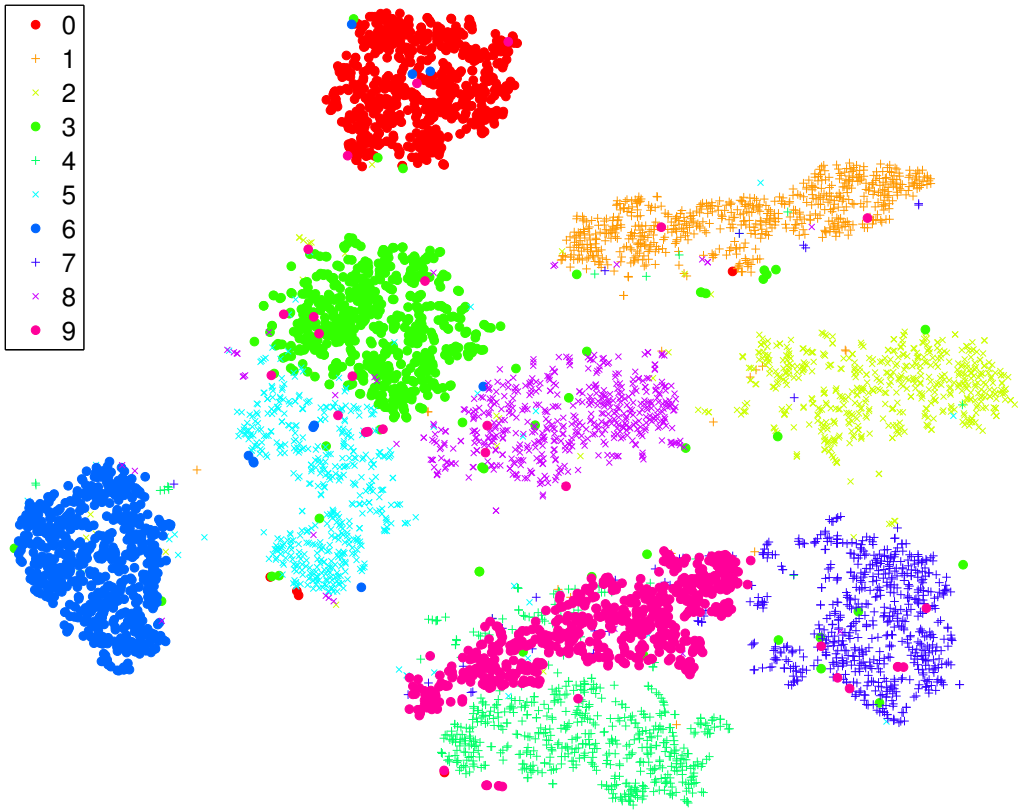


Isomap



LLE

Example: Visualizations of MNIST



t-SNE