ML Fundamentals

Instituto Balseiro

Instituto Balseiro
09/09/2022

Announcements

- Guía 03 – Unsupervised: se subirá durante el día de hoy/mañana
- Habrá una fecha adicional (la semana que viene?): Lunes o jueves, a confirmar, estén atentos al anuncio en github

How should we normalize the *test* partition? 🤔

- **Separately** from *training*?
  - But then each may be normalized differently
- **Jointly** with *training*?
  - But then some information goes from *test* to *train*

Data leakeage

- "*In general, with a multistep modeling procedure, cross-validation must be applied to the entire sequence of modeling steps*". Hastie et al. (2013) Section 7.10.2 The Wrong and Right Way to do validation, page 245

In particular: Don't use the whole dataset for:

- Feature selection
- Normalization
- Fine tuning (e..g. with grid search)

...All these should be inside the folds in cross-validation, or in the single validation partition

Lecture 7
Unsupervised

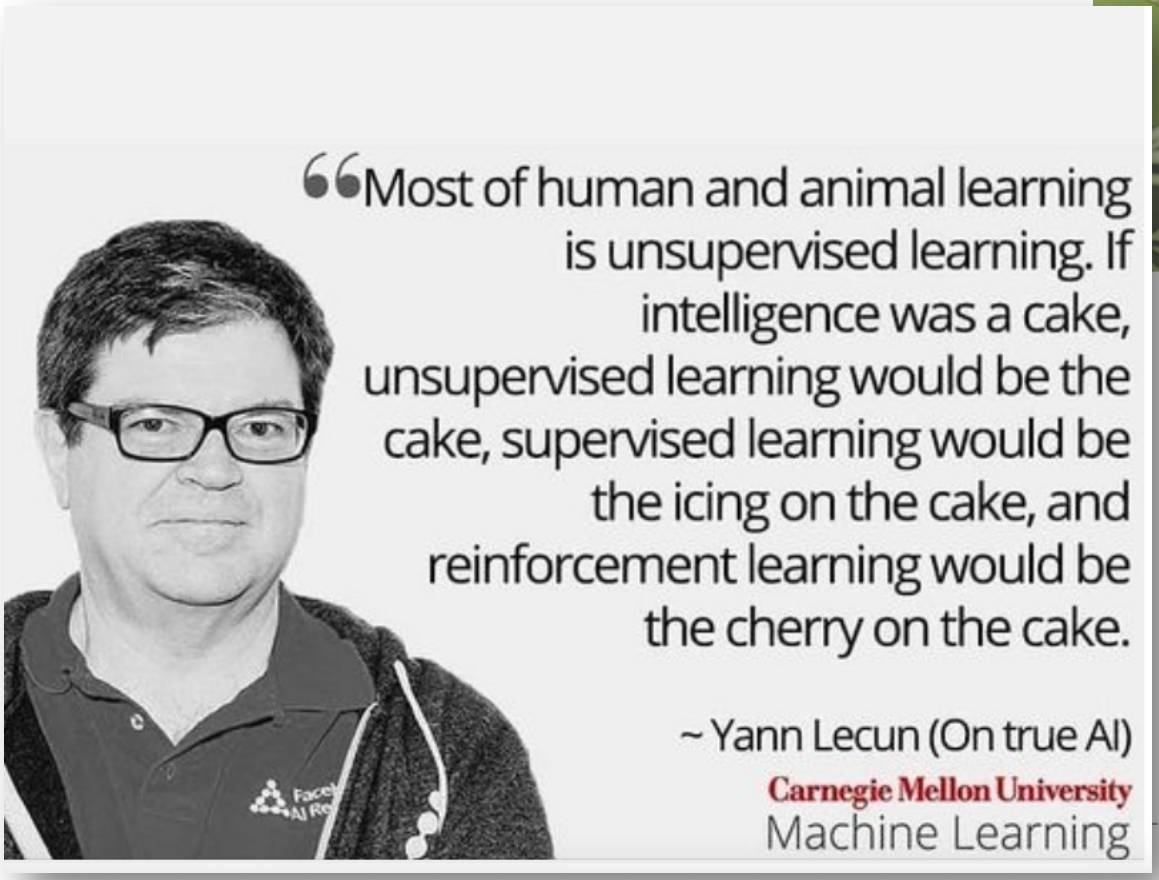
Underwater realm, Anuar Patjane

$$P(y|x) \text{ vs } P(x)$$


Laburen, neuronas!

LeCun's cake metaphor

https://www.youtube.com/watch?v=Ount2Y4qxQo#t=19m21s



ML Fundamentals – Lecture 7

- Unsupervised learning
- Clustering
  - k-means
  - Hierarchical methods
  - DBScan
- DR vs Representation vs Manifold learning
- Dimensionality reduction
  - PCA & kernel PCA
  - MDS

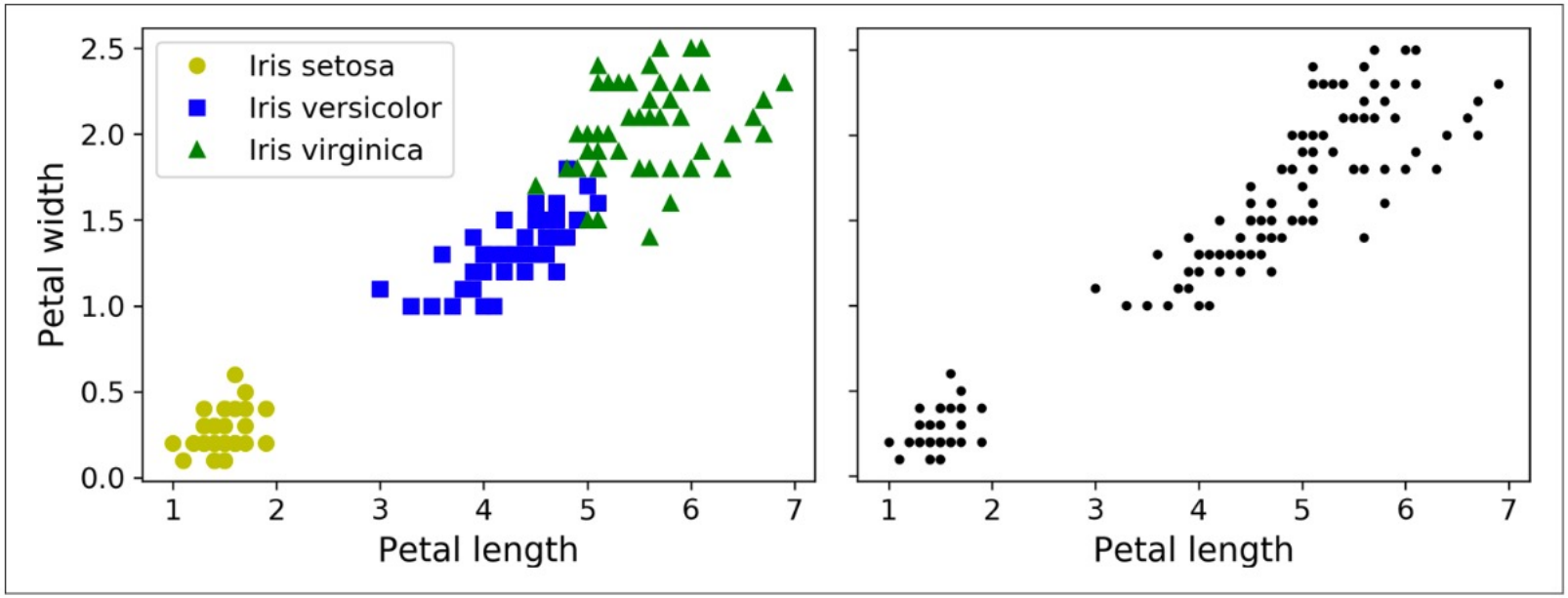
Unsupervised learning – use cases

- Clustering
- Anomaly detection (novelty, outliers)
- Density estimation
  - drawing samples from a distribution
- Dimensionality reduction
  - compression, denoising
- Semi-supervised learning
- Visualization
- Manifold learning

Clustering

Larch Stand on Bannock Mountain Scott Rinckenberger

Clusters as patterns



Patterns as similarity, distance, affinity

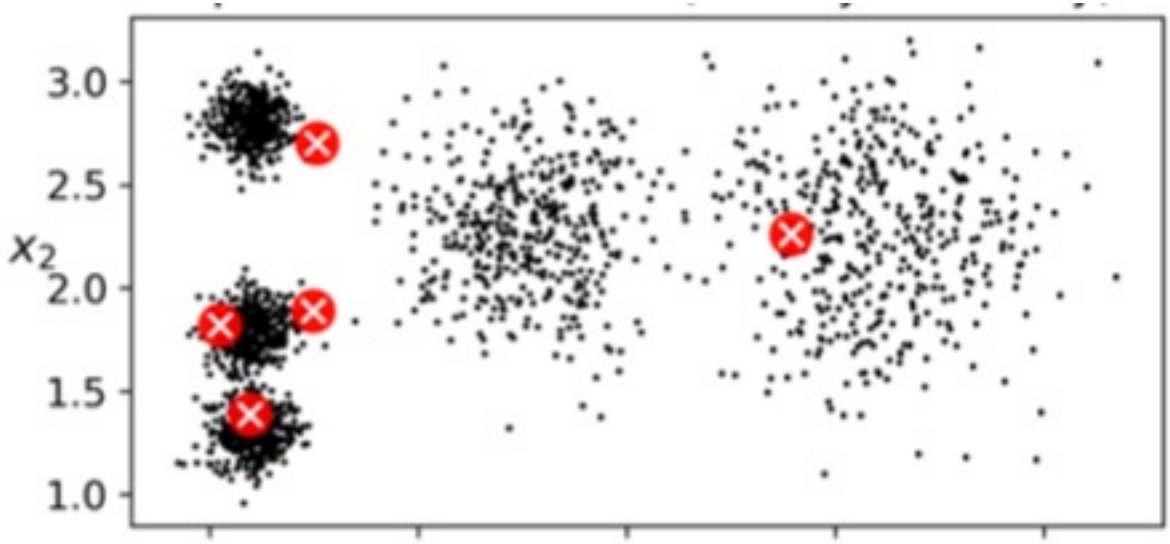
k-means

1)

- Compute centroids (implies distances)

2)

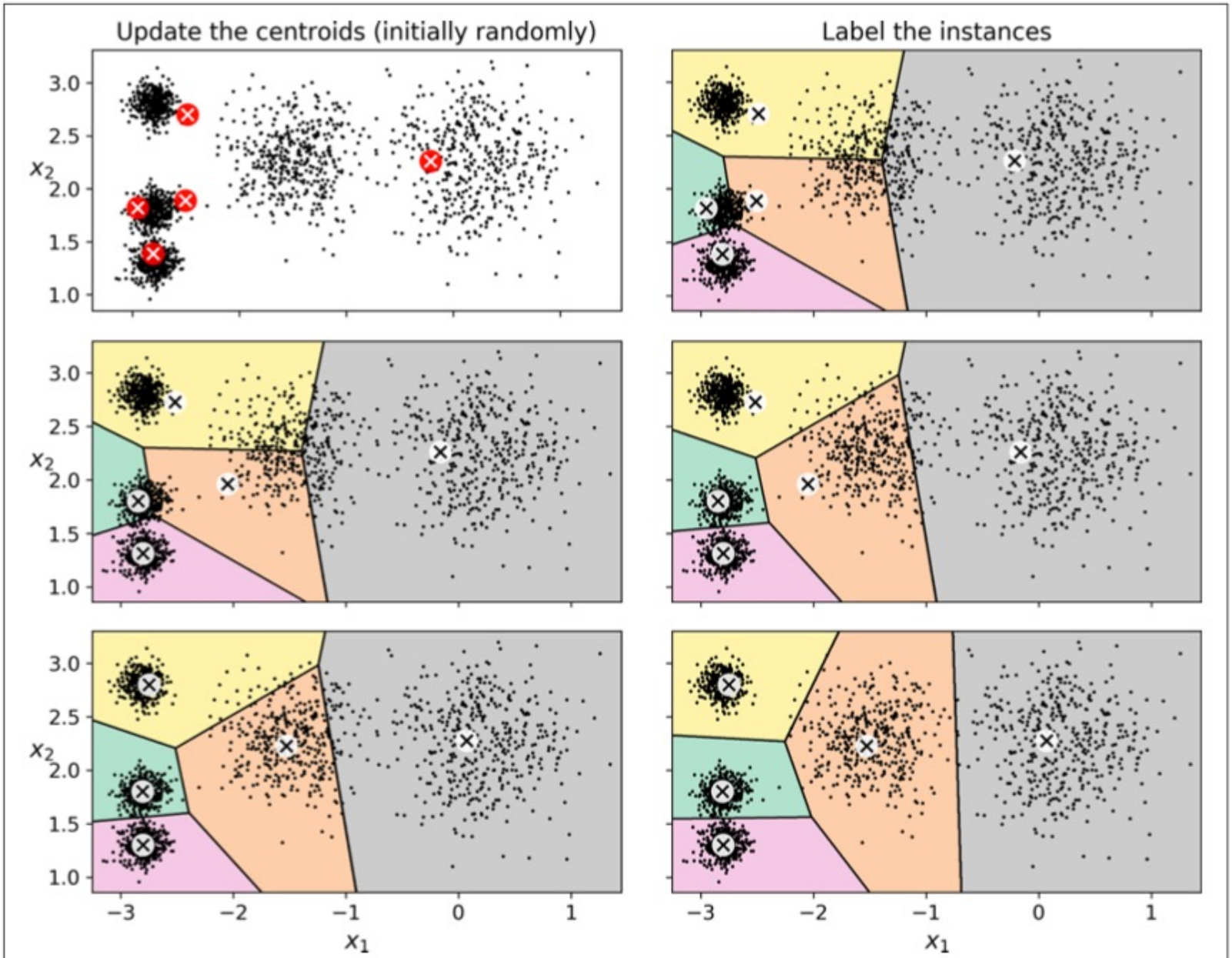
- Label instances



k-means

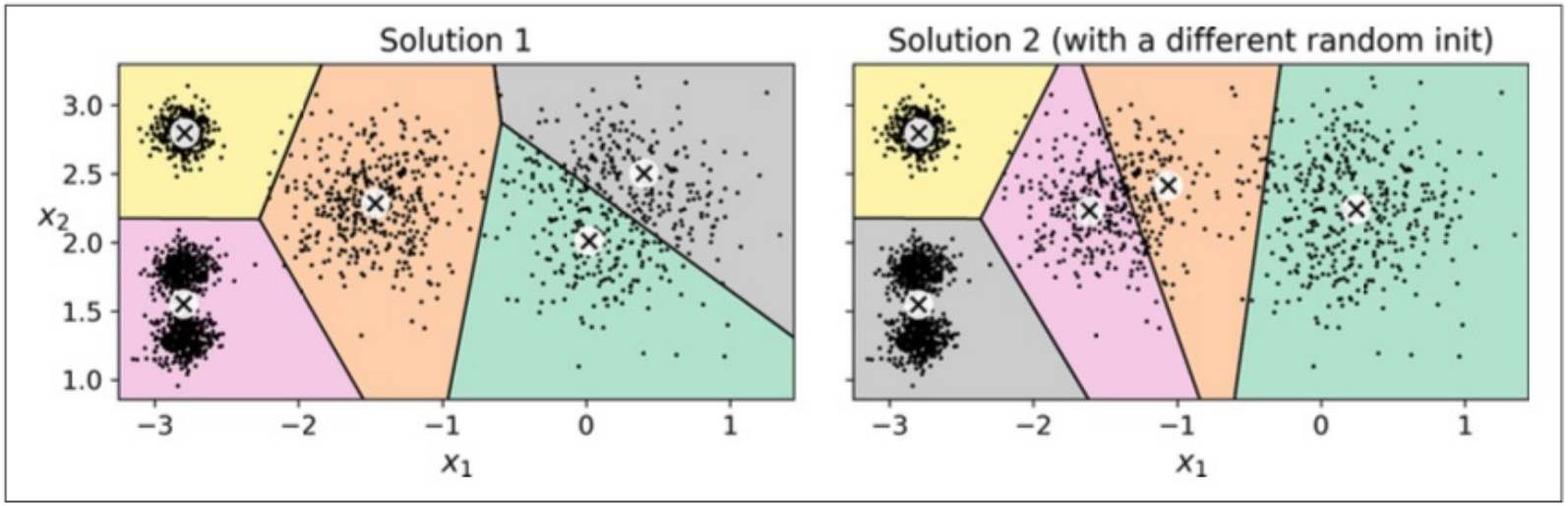
1)

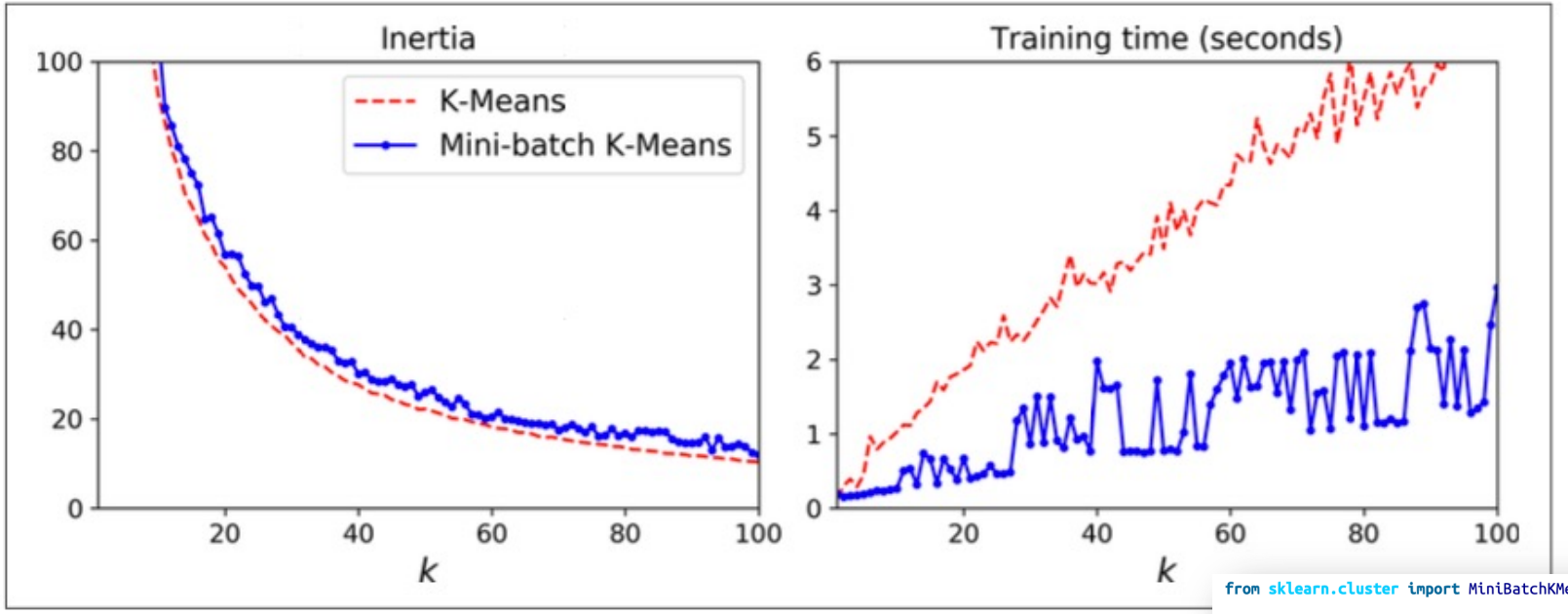
2)



Luis G. Moyano - Fundamentos de ML - Instituto Balseiro

Initial conditions strategies

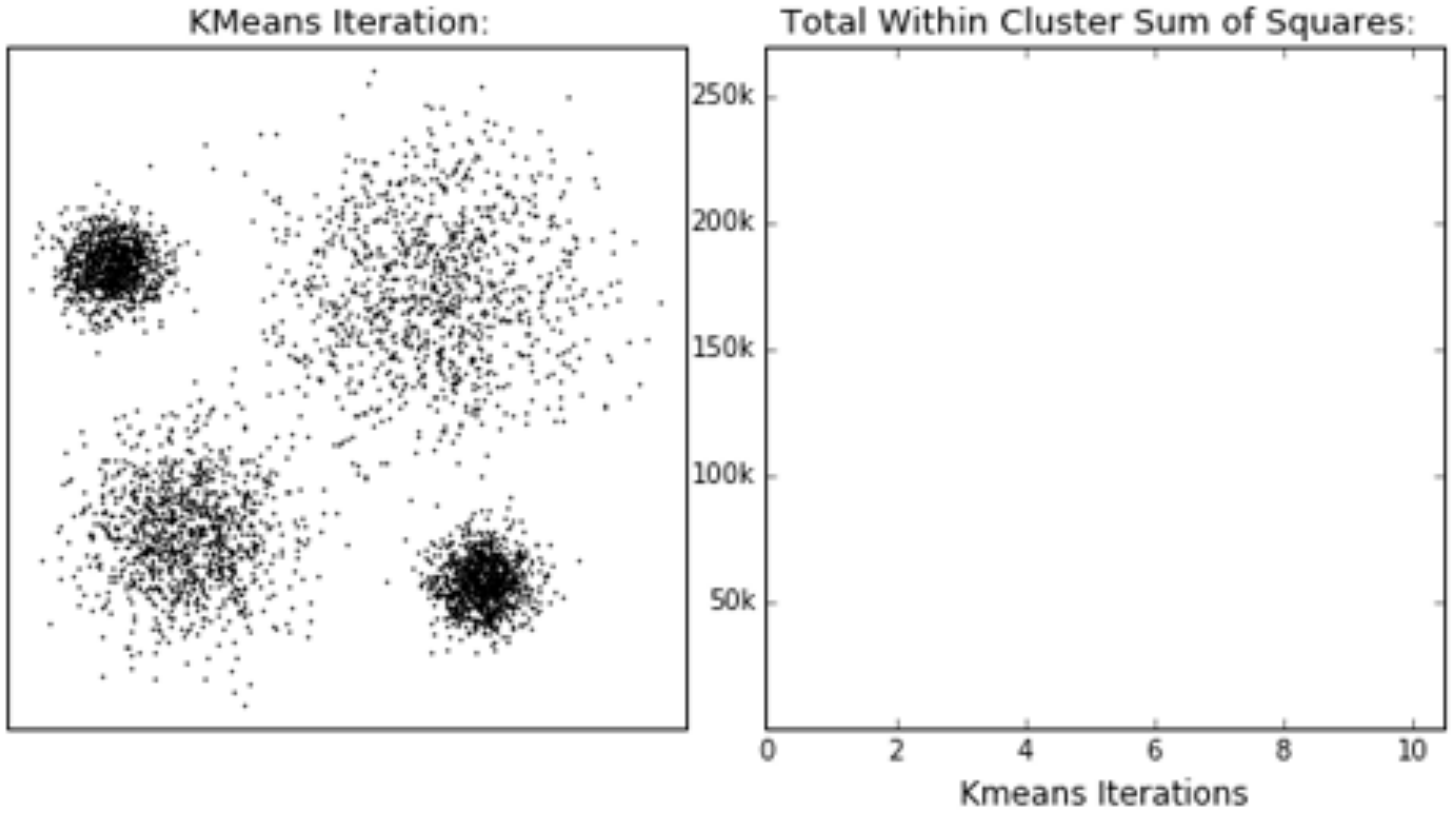





```
from sklearn.cluster import MiniBatchKMeans

minibatch_kmeans = MiniBatchKMeans(n_clusters=5)
minibatch_kmeans.fit(X)
```

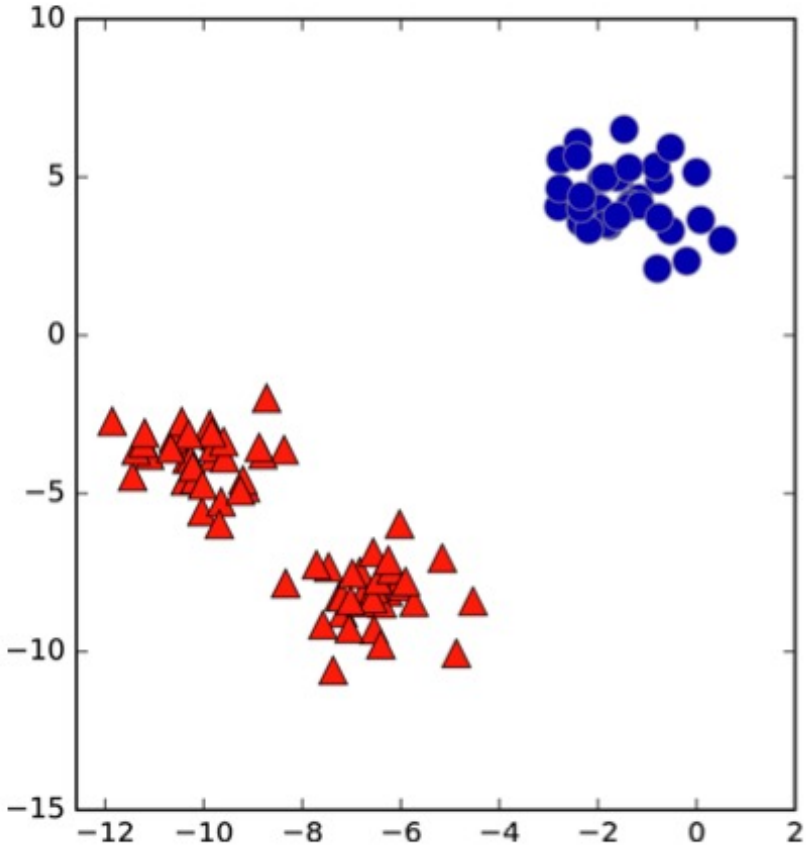
k-means

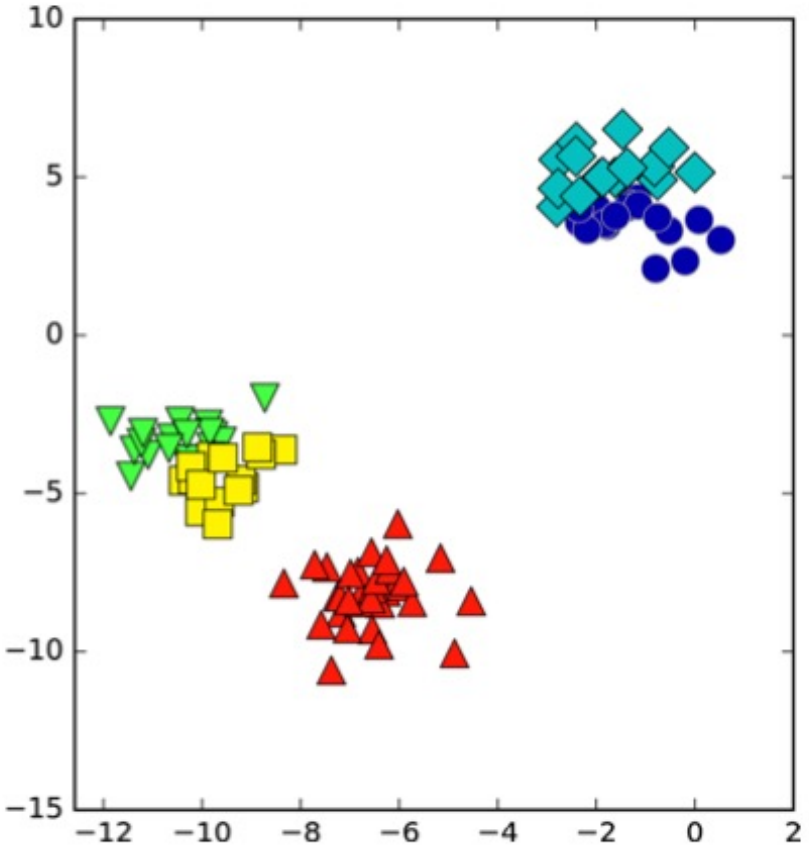



```
from sklearn.cluster import KMeans
k = 5
kmeans = KMeans(n_clusters=k)
y_pred = kmeans.fit_predict(X)
```

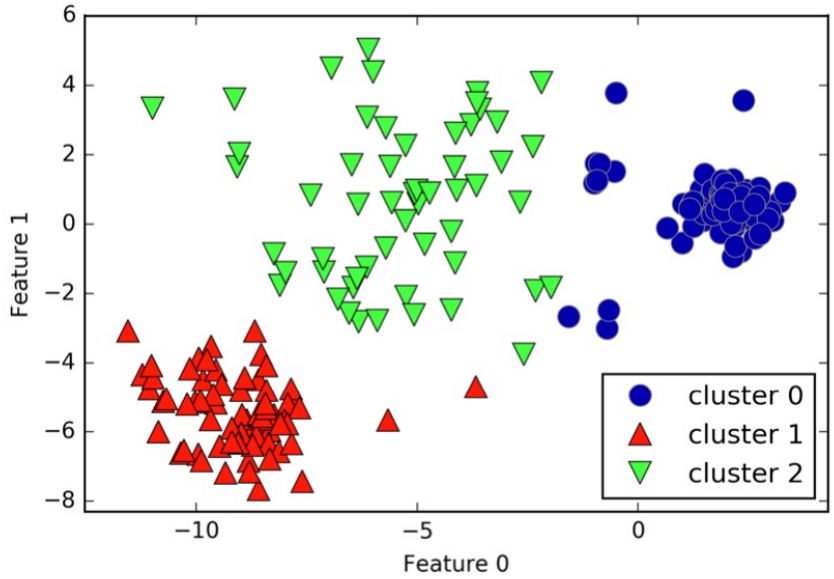
```
>>> y_pred
array([4, 0, 1, ..., 2, 1, 0], dtype=int32)
>>> y_pred is kmeans.labels_
True
>>> kmeans.cluster_centers_
array([[-2.80389616,  1.80117999],
       [ 0.20876306,  2.25551336],
       [-2.79290307,  2.79641063],
       [-1.46679593,  2.28585348],
       [-2.80037642,  1.30082566]])
```

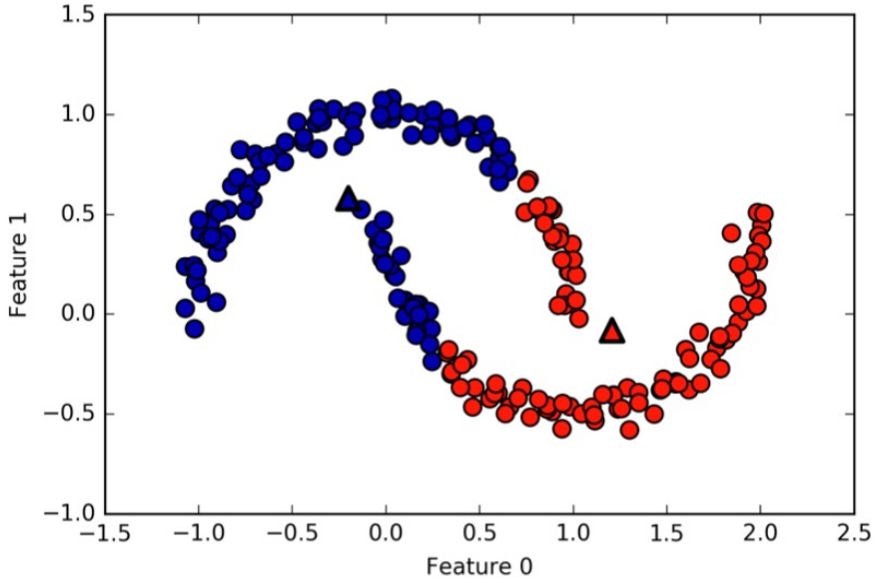
k is arbitrarily set

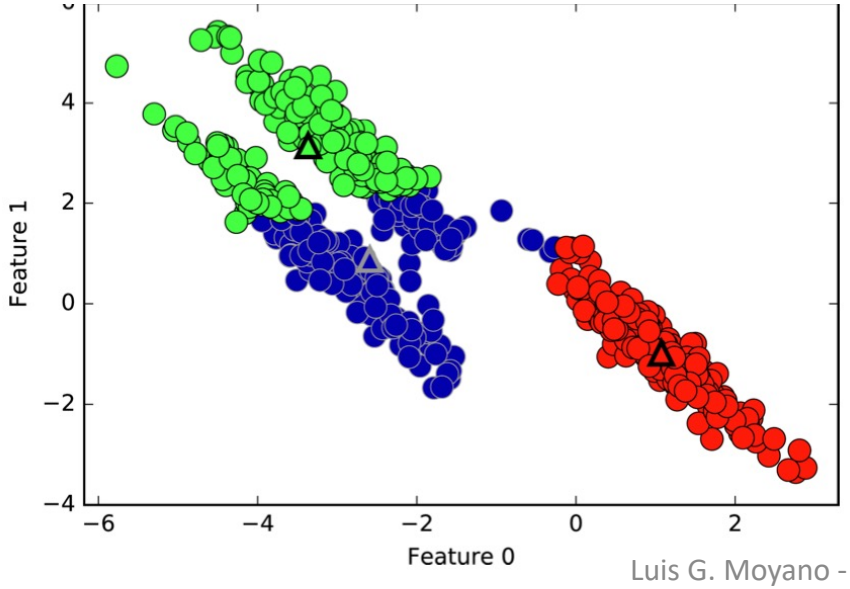


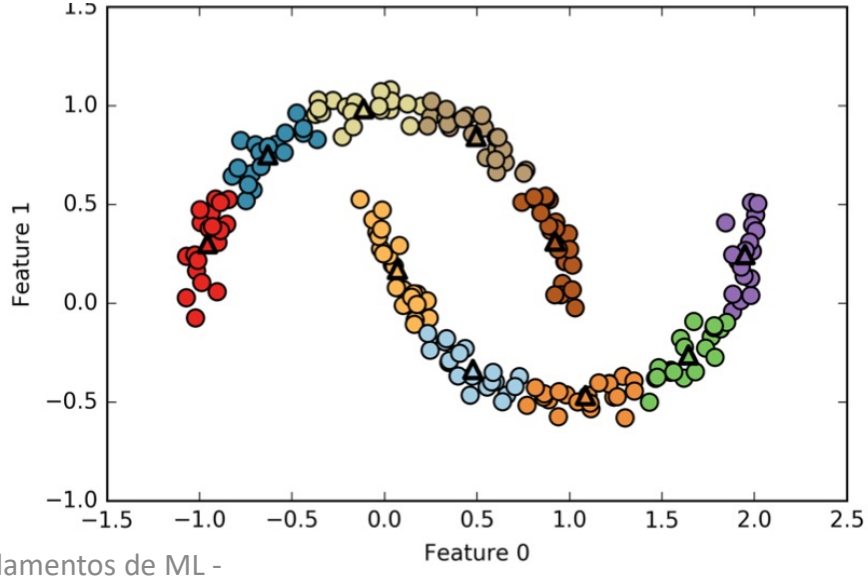


k-means weaknesses

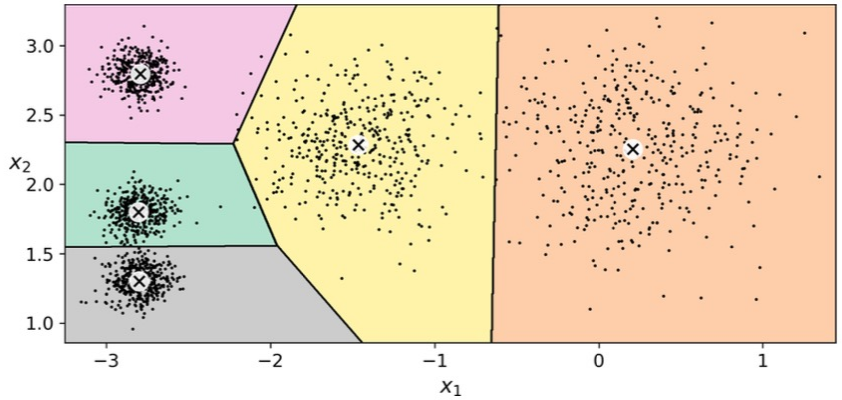








Hard vs soft clustering




```
>>> X_new = np.array([[0, 2], [3, 2], [-3, 3], [-3, 2.5]])
>>> kmeans.predict(X_new)
array([1, 1, 2, 2], dtype=int32)
```

```
>>> kmeans.transform(X_new)
array([[2.81093633, 0.32995317, 2.9042344 , 1.49439034, 2.88633901],
       [5.80730058, 2.80290755, 5.84739223, 4.4759332 , 5.84236351],
       [1.21475352, 3.29399768, 0.29040966, 1.69136631, 1.71086031],
       [0.72581411, 3.21806371, 0.36159148, 1.54808703, 1.21567622]])
```

New representation or Feature engineering

k-means as one-hot decomposition

original

reconstruction

50×37-pixel grayscale images
100-means reconstruction

Clustering for Semi-Supervised Learning

```
n_labeled = 50
log_reg = LogisticRegression()
log_reg.fit(X_train[:n_labeled], y_train[:n_labeled])
```

```
>>> log_reg.score(X_test, y_test)
0.8333333333333334
```

MNist dataset (70k labeled images)
Test accuracy = 0.97

Train logistic reg. with 50 random points
Test accuracy = 0.83

```
k = 50
kmeans = KMeans(n_clusters=k)
X_digits_dist = kmeans.fit_transform(X_train)
representative_digit_idx = np.argmin(X_digits_dist, axis=0)
X_representative_digits = X_train[representative_digit_idx]
```

Compute 50 clusters & show centroids:

```
y_representative_digits = np.array([4, 8, 0, 6, 8, 3, ..., 7, 6, 2, 3, 1, 1])
```

Manually label 50 instances

```
>>> log_reg = LogisticRegression()
>>> log_reg.fit(X_representative_digits, y_representative_digits)
>>> log_reg.score(X_test, y_test)
0.9222222222222223
```

Train with representative intances
Test accuracy = 0.92

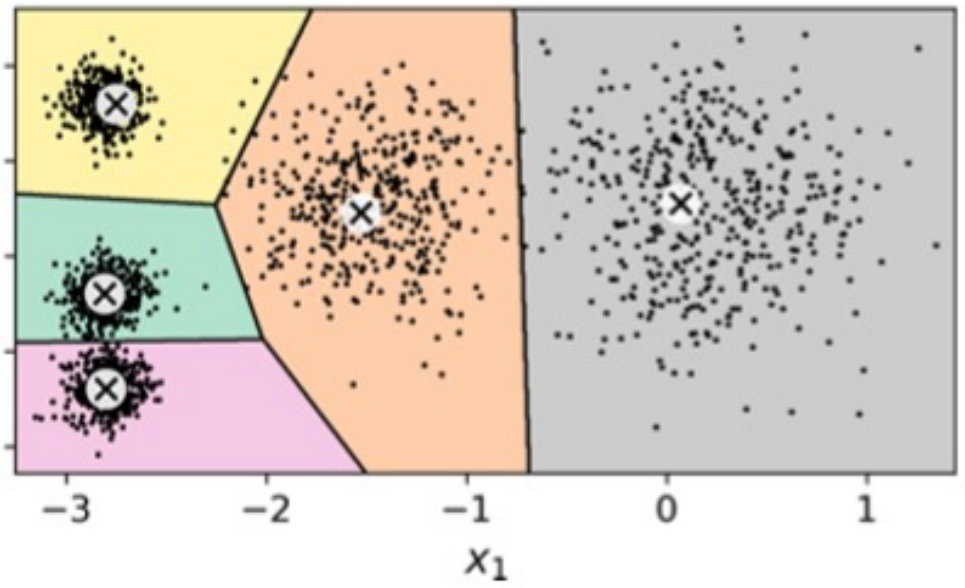
```
y_train_propagated = np.empty(len(X_train), dtype=np.int32)
for i in range(k):
    y_train_propagated[kmeans.labels_==i] = y_representative_digits[i]
```

```
>>> log_reg = LogisticRegression()
>>> log_reg.fit(X_train, y_train_propagated)
>>> log_reg.score(X_test, y_test)
0.9333333333333333
```

Propagate labels to all cluster members, train and predict
Test accuracy= 0.93 (50 vs 70k)

k-means pros & cons

- fast
- scales
- interpretable



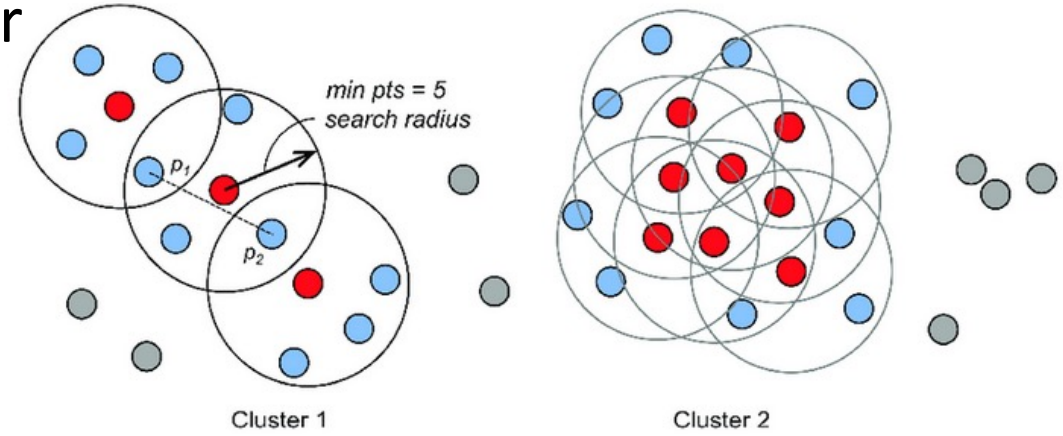
- k fixed beforehand
- all features equally important
- equally-sized clusters

DBScan

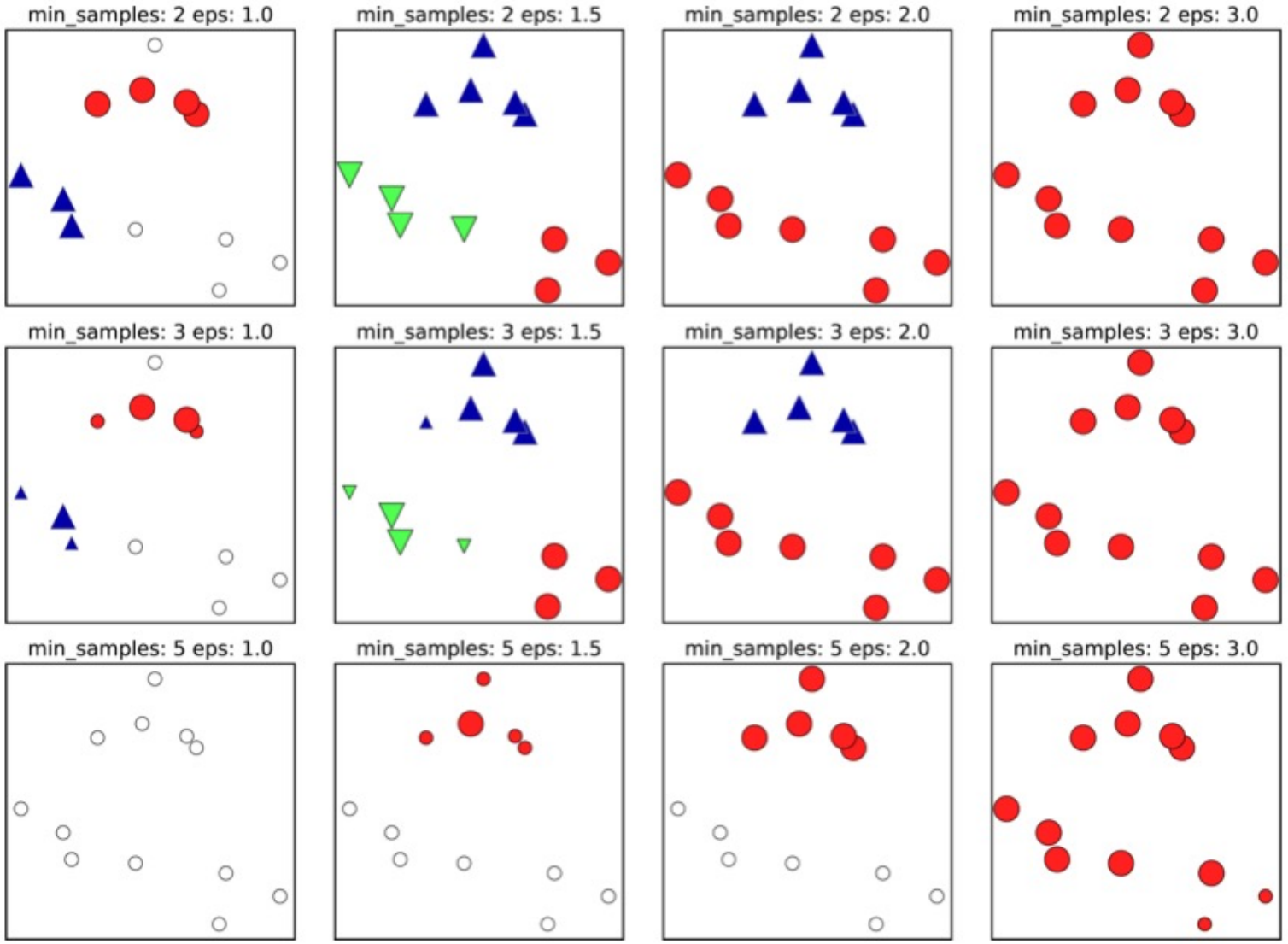
"Density- based spatial clustering of applications with noise"

- High density areas separated by low density areas
- No preset cluster number
- Possibility of outliers
- Parameters:
  - *min_samples*
  - *eps*
- Algorithm:
  - for a given datum, if there are at least *min_samples* data points within a distance of *eps*, the point is considered *core sample*
  - Iterate recursively

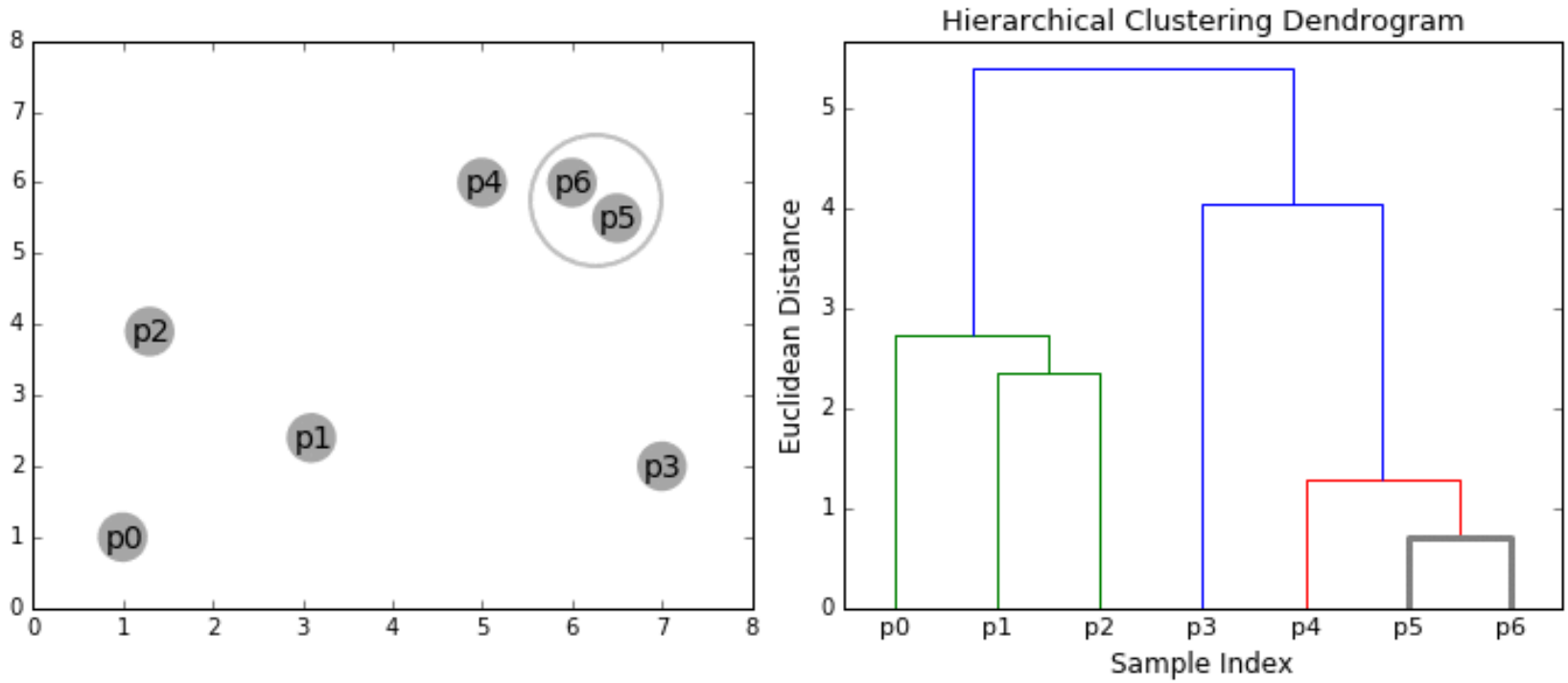
  A cluster is a set of core data points and their non-core neighbours (in the fringes)
  - There may be outliers, i.e. non-core data points at a distance larger than eps



DBScan

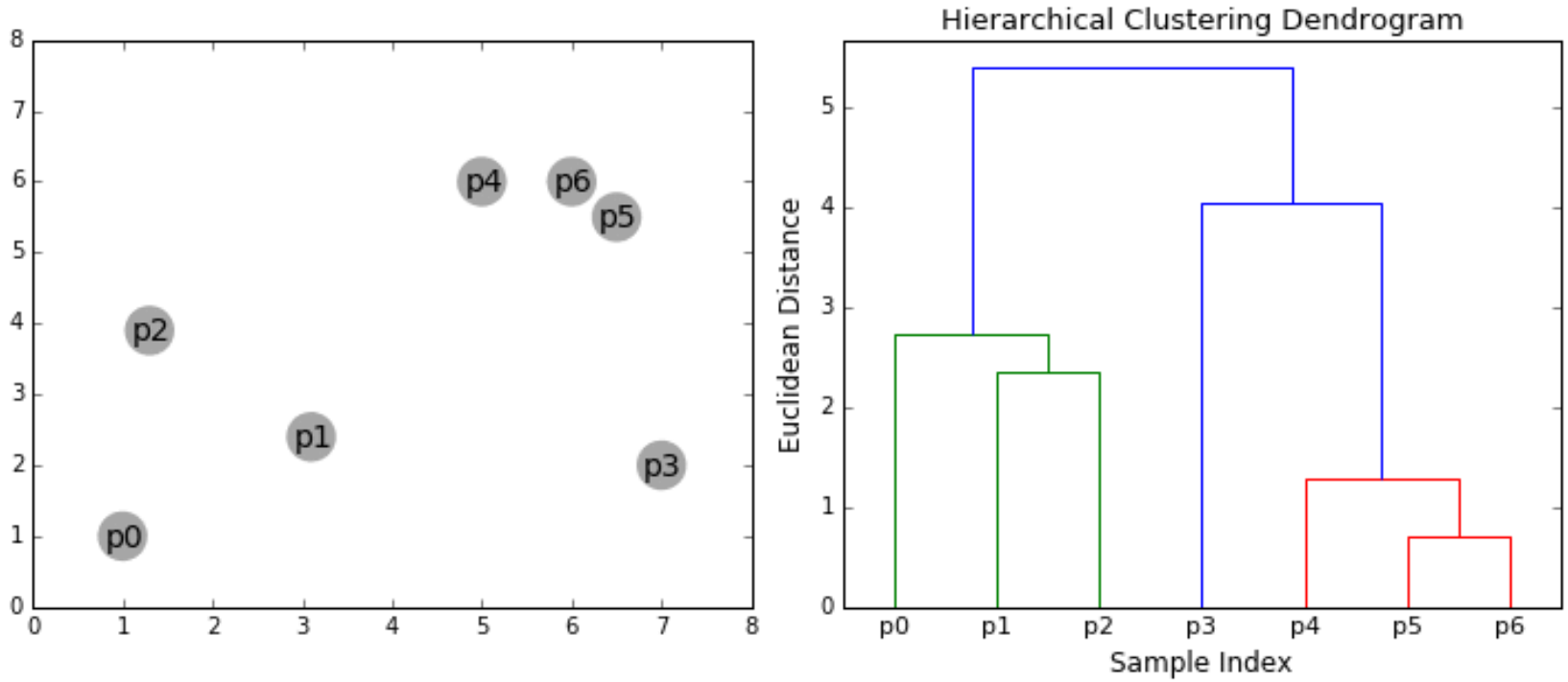

min_samples: 2 eps: 1.0
min_samples: 2 eps: 1.5
min_samples: 2 eps: 2.0
min_samples: 2 eps: 3.0
min_samples: 3 eps: 1.0
min_samples: 3 eps: 1.5
min_samples: 3 eps: 2.0
min_samples: 3 eps: 3.0
min_samples: 5 eps: 1.0
min_samples: 5 eps: 1.5
min_samples: 5 eps: 2.0
min_samples: 5 eps: 3.0

Hierarchical agglomerative clustering



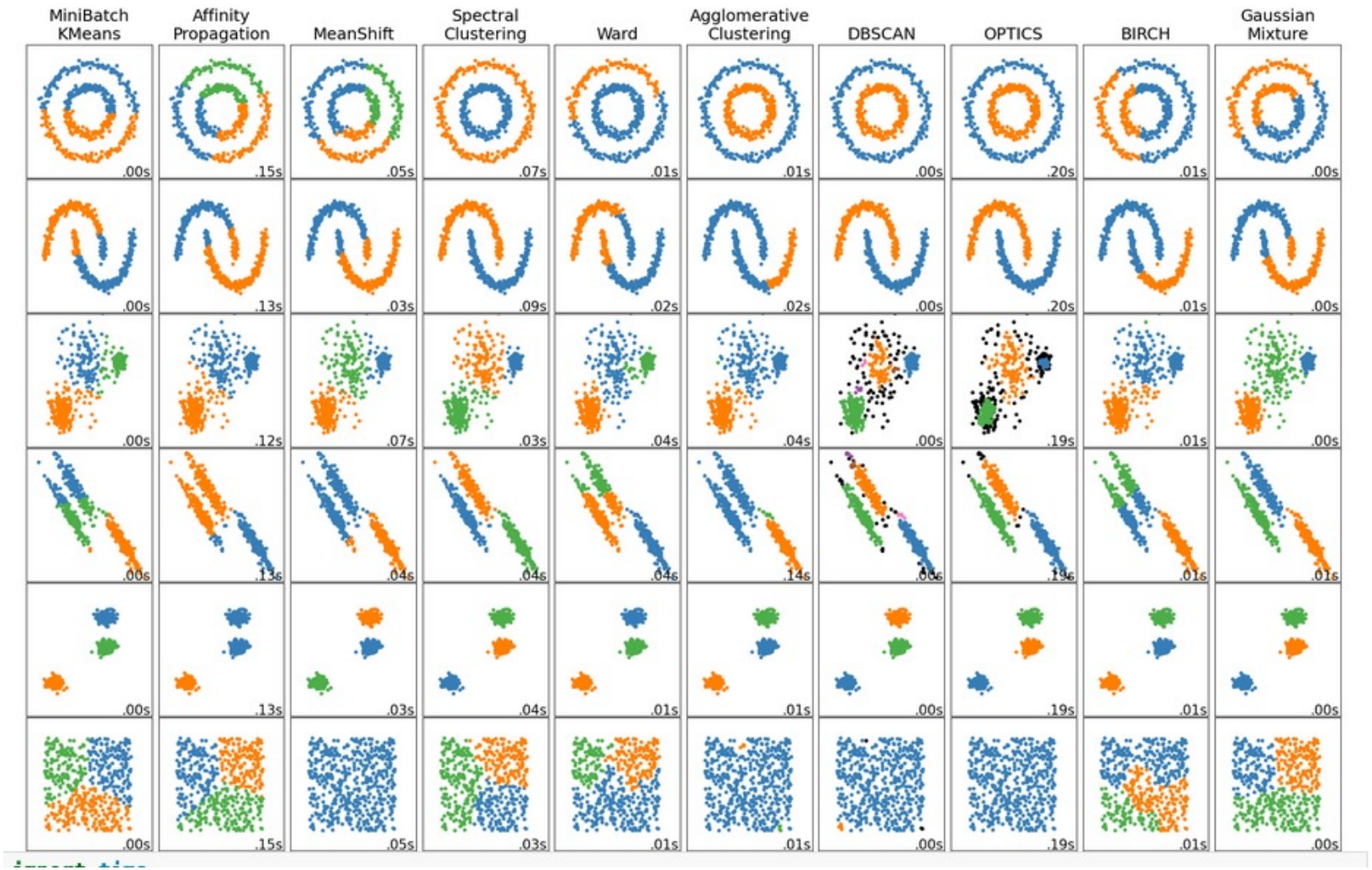
Credit: **David Sheehan** – Clustering with Scikit with GIFs
https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/

Hierarchical agglomerative clustering



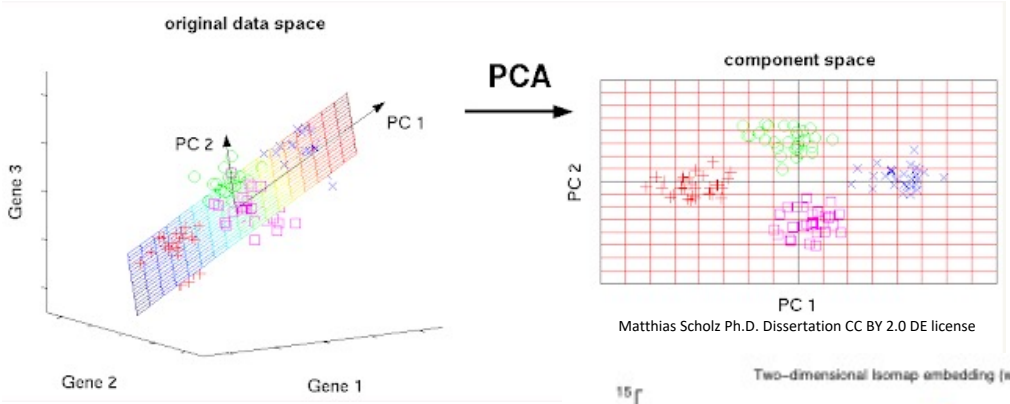
Credit: **David Sheehan** – Clustering with Scikit with GIFs
https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/

Clustering in scikit-learn

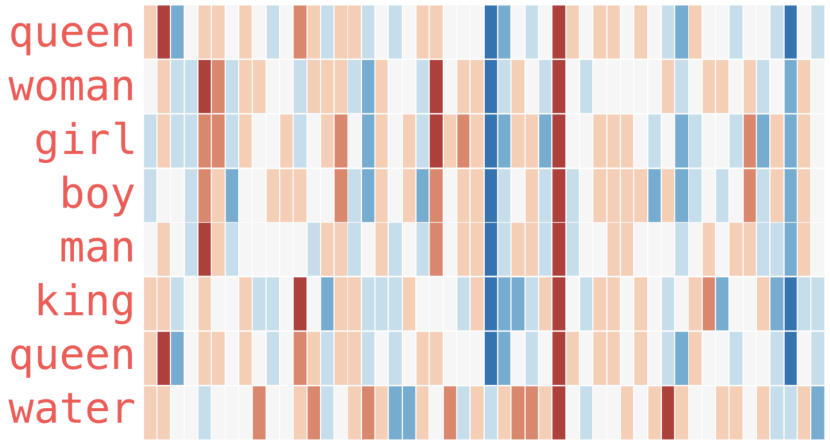


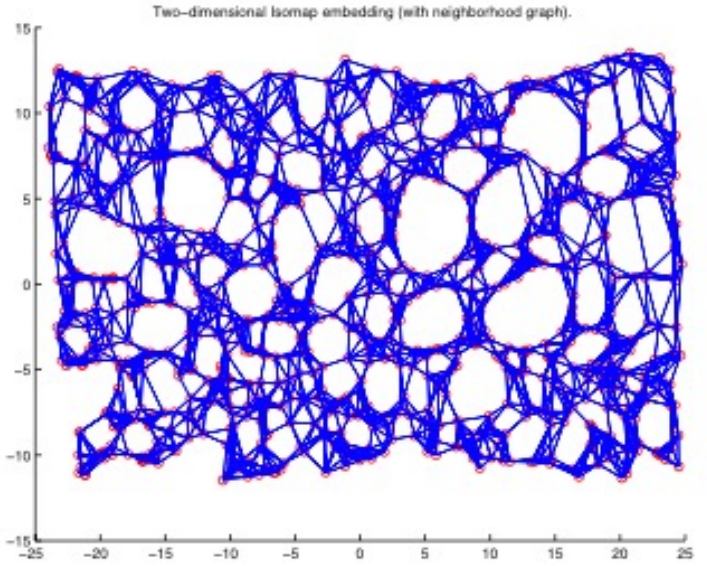
https://scikit-learn.org/stable/modules/clustering.html

Dimensionality reduction vs. Representation learning vs. Manifold Learning

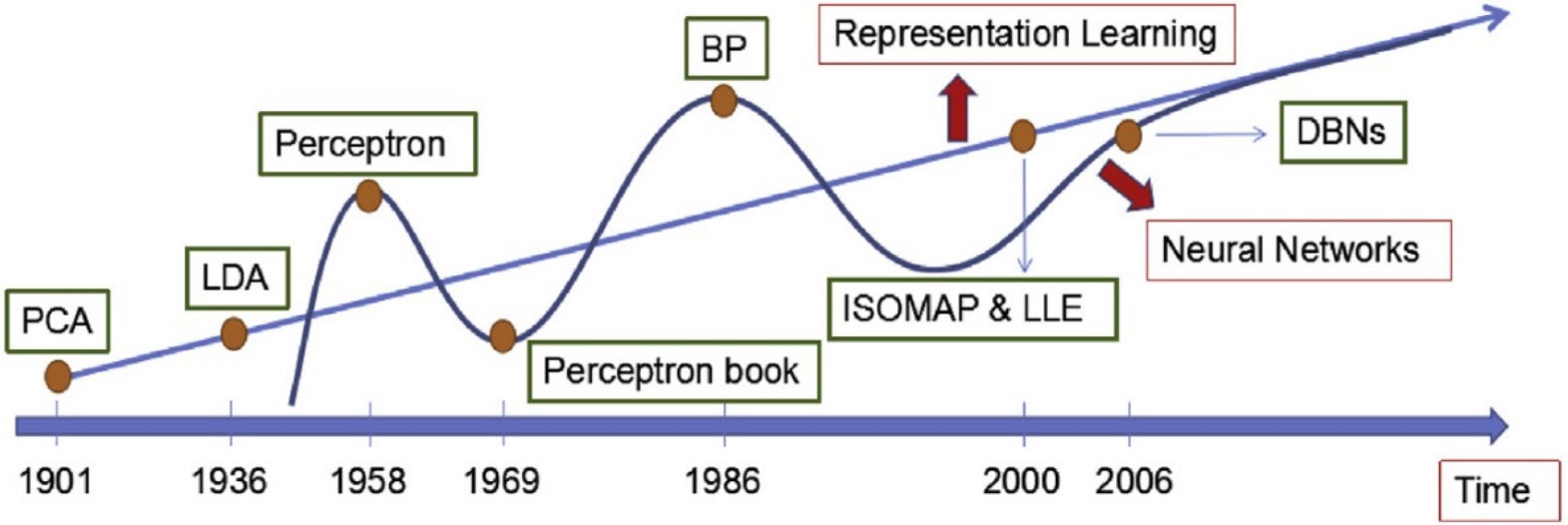


Matthias Scholz Ph.D. Dissertation CC BY 2.0 DE license





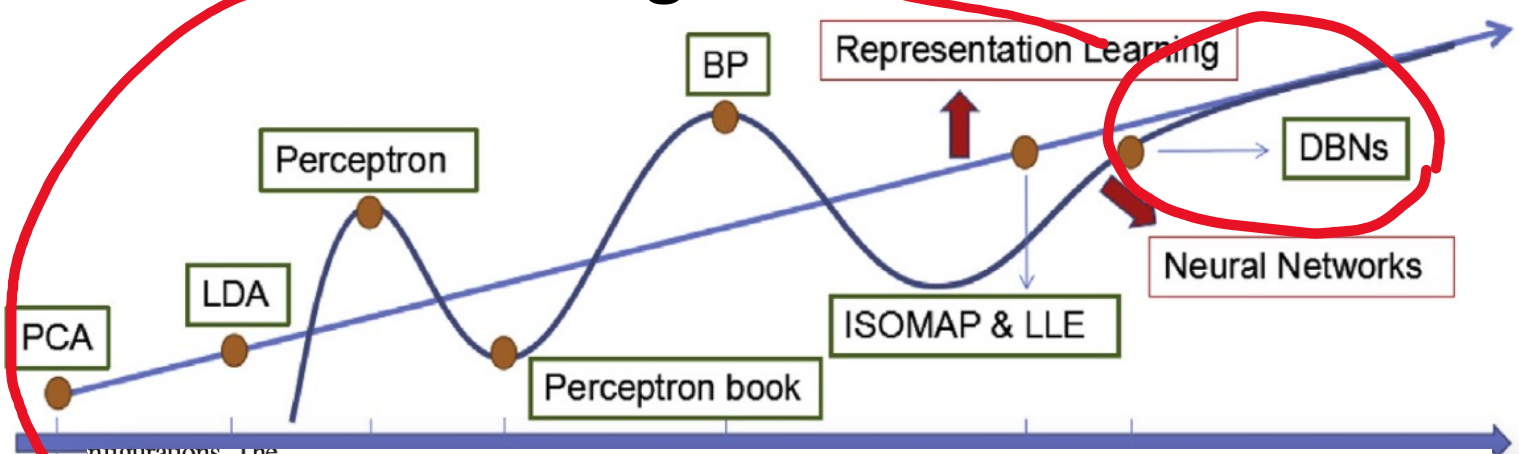
Representation learning timeline



2016 - Zhong - An overview on data representation learning: From traditional feature learning to recent deep learning

Luis G. Moyano - Fundamentos de ML Instituto Balseiro

Representation learning timeline



Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton* and R. R. Salakhutdinov

High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors. Gradient descent can be used for fine-tuning the weights in such "autoencoder" networks, but this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.

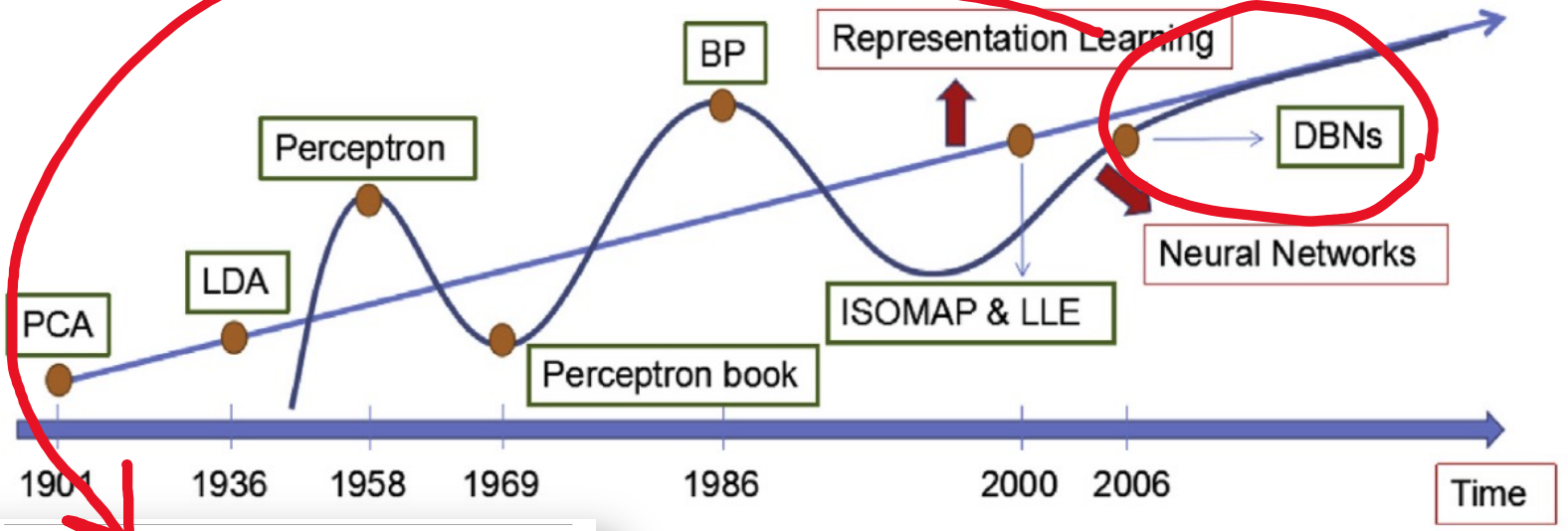
Dimensionality reduction facilitates the classification, visualization, communication, and storage of high-dimensional data. A simple and widely used method is principal components analysis (PCA), which finds the directions of greatest variance in the data set and represents each data point by its coordinates along each of these directions. We describe a nonlinear generalization of PCA that uses an adaptive, multilayer "encoder" network

28 JULY 2006 VOL 313 SCIENCE www.sciencemag.org

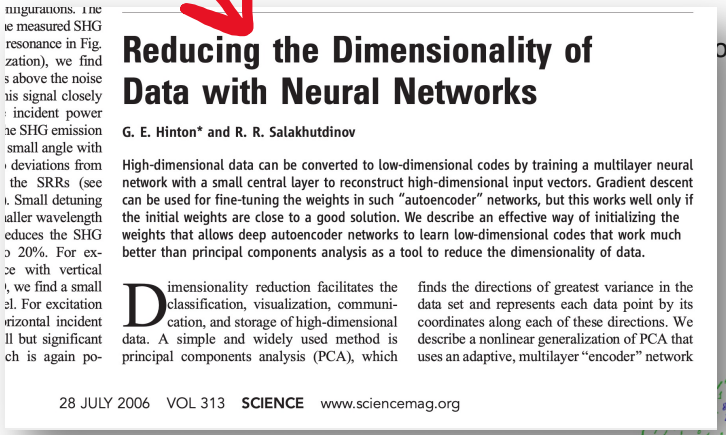
Hinton et al., Reducing the Dimensionality of Data with Neural Networks, Science 2006

Luis G. Moyano - Fundamentos de ML Instituto Balseiro

Representation learning timeline



on learning: From traditional feature learning to recent deep learning



Reducing the Dimensionality of Data with Neural Networks

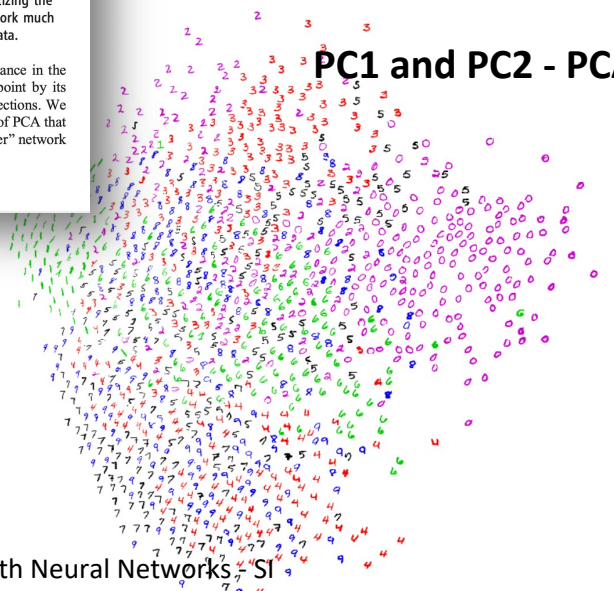
G. E. Hinton* and R. R. Salakhutdinov

High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors. Gradient descent can be used for fine-tuning the weights in such "autoencoder" networks, but this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.

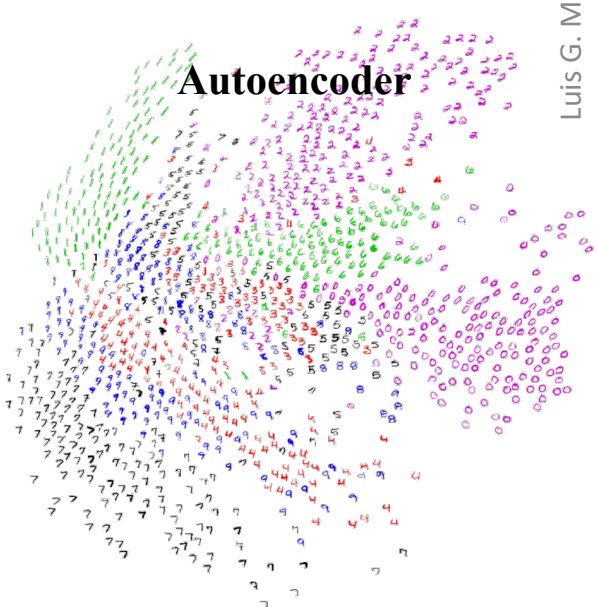
Dimensionality reduction facilitates the classification, visualization, communication, and storage of high-dimensional data. A simple and widely used method is principal components analysis (PCA), which finds the directions of greatest variance in the data set and represents each data point by its coordinates along each of these directions. We describe a nonlinear generalization of PCA that uses an adaptive, multilayer "encoder" network

28 JULY 2006 VOL 313 SCIENCE www.sciencemag.org

PC1 and PC2 - PCA



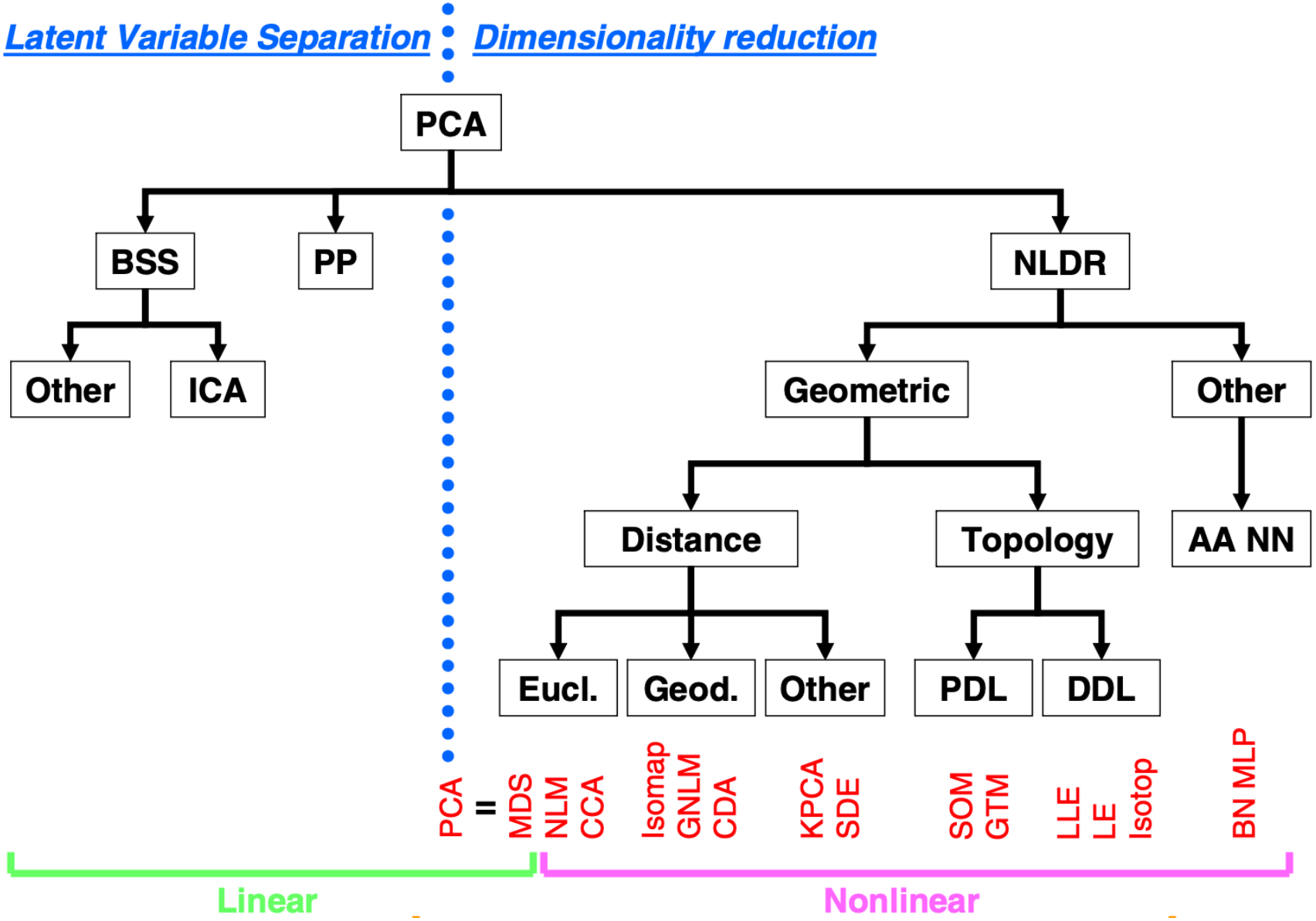
Autoencoder



2006 - Hinton - Reducing the Dimensionality of Data with Neural Networks - SI

Luis G. Moyano - Fundamentos de ML Instituto Balseiro

DR Taxonomy

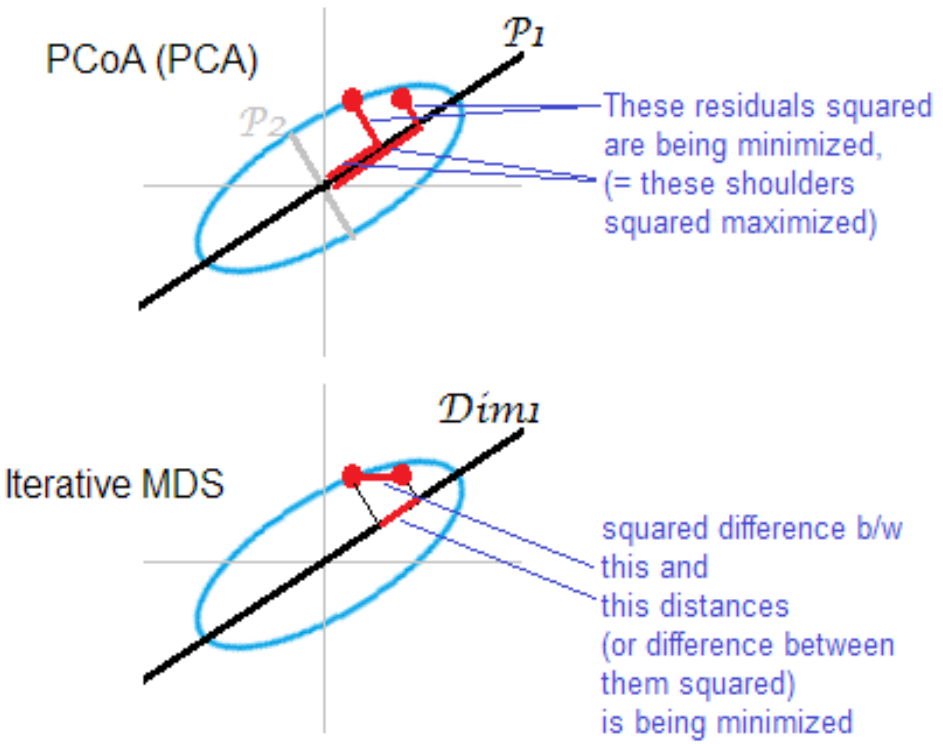


2007 - Lee - Nonlinear Dimensionality Reduction

Projective & manifold methods

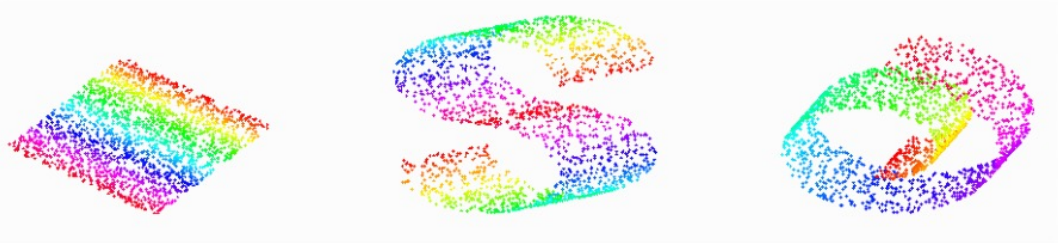
Projective

- PCA
- kernel PCA
- MDS
- others!



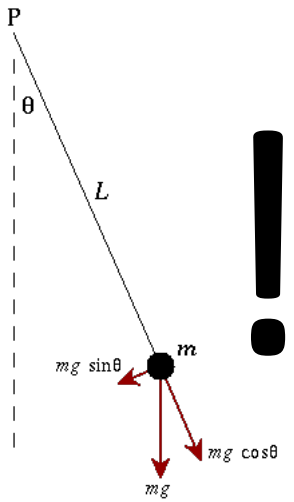
Manifold

- Isomap
- LLE
- LE
- others!



Principal Components Analysis - PCA

AKA Karhunen-Loève transform



- Earliest linear model
  - along with LDA (Fisher 1936)
- Transformation of variables to find new axes under a given optimization
- *Feature extraction*
  - Not *feature selection*
- Dimensionality reduction
  - Visualization
  - Denoising
  - Compression

LIII. *On Lines and Planes of Closest Fit to Systems of Points in Space.* By KARL PEARSON, *F.R.S., University College, London* *.

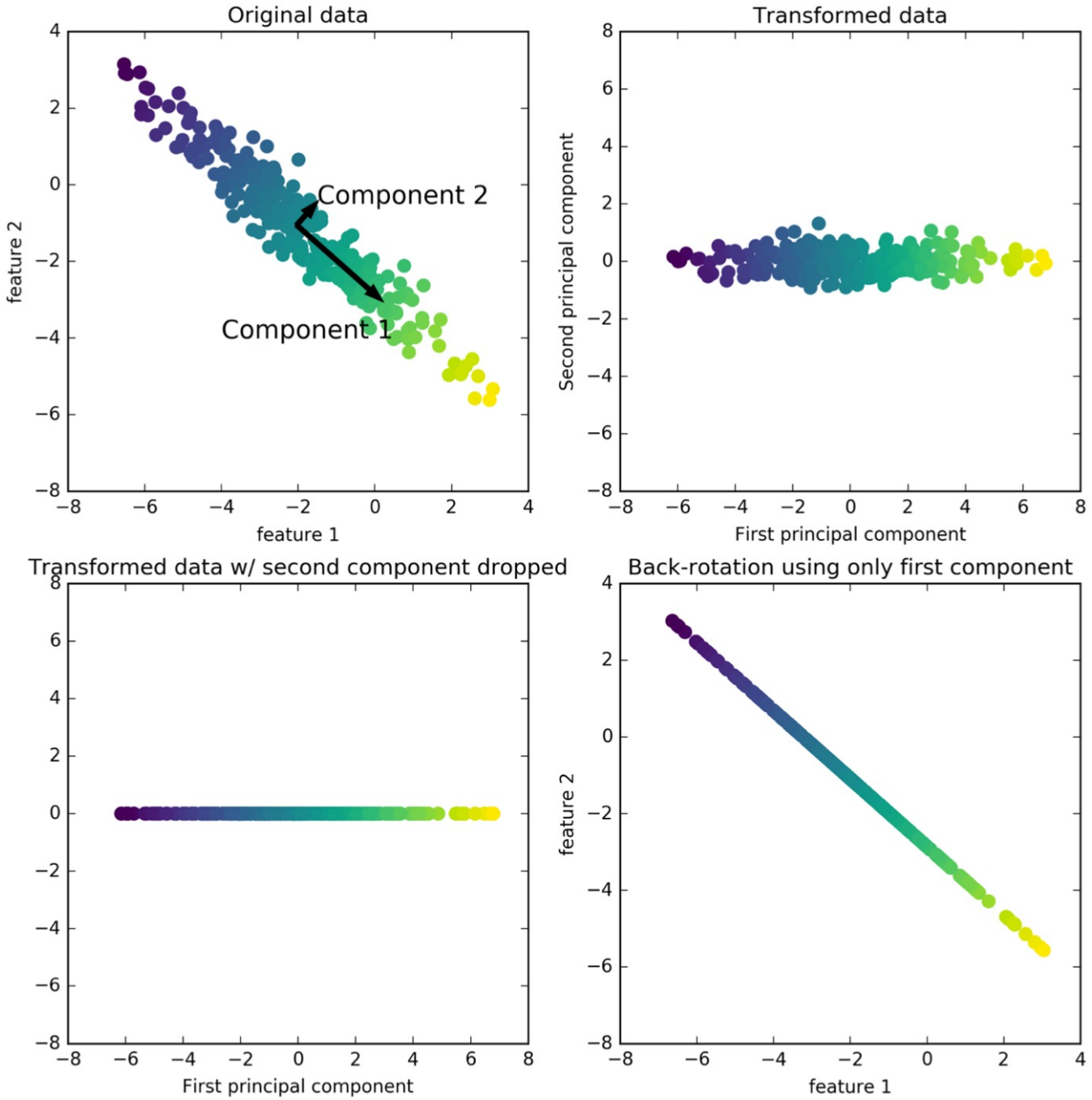
1901

(1) IN many physical, statistical, and biological investigations it is desirable to represent a system of points in plane, three, or higher dimensioned space by the "best-fitting" straight line or plane. Analytically this consists in taking

$$y = a_0 + a_1 x, \quad \text{or} \quad z = a_0 + a_1 x + b_1 y,$$
$$\text{or} \quad z = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3 + \ldots + a_n x_n,$$

wh … e
" … n
in relation to the observed corresponding values of the variables. In nearly all the cases dealt with in the text-books

K. Pearson. *On lines and planes of closest fit to systems of points in space*. Philosophical Magazine, 2:559–572, 1901.

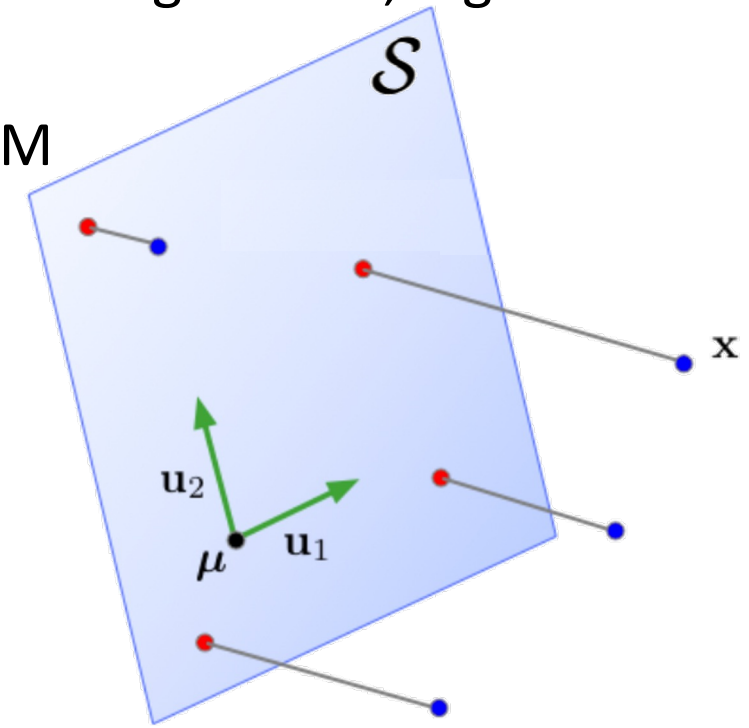

Original data
Component 2
Component 1
feature 2
feature 1
Transformed data
Second principal component
First principal component
Transformed data w/ second component dropped
First principal component
Back-rotation using only first component
feature 2
feature 1

PCA

- PCA is a linear model with a closed-form solution.
- It's useful for understanding lots of other algorithms, e.g. autoencoders, etc.
- N points in D dimension, subspace in M

$$z_{nj} = \mathbf{x}_n^{\mathrm{T}} \mathbf{u}_j$$

$$\hat{\mathbf{x}}_i = \mathbf{W}\mathbf{z}_i$$



- The DxM **W** matrix has unit norm and othogonal columns

Goal: find an M-dimensional subspace $\mathcal{S} \subset \mathbb{R}^D$ such that point is "well represented" by its projection $\mathbf{x}_n^{\mathrm{T}} \mathbf{u}_j$ (i.e., closest point in $\mathcal{S}$).

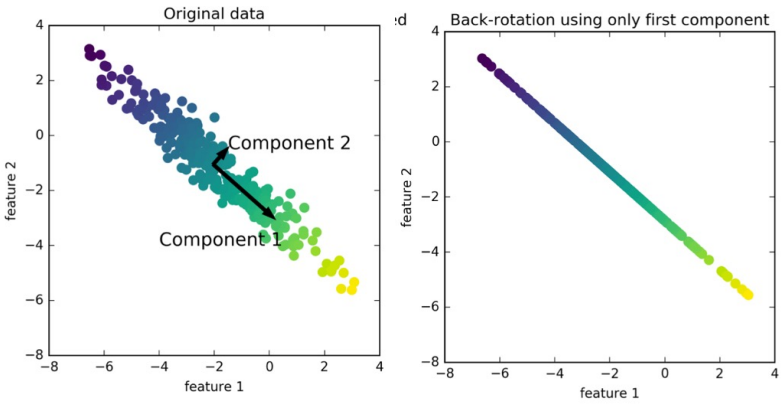
PCA rationale

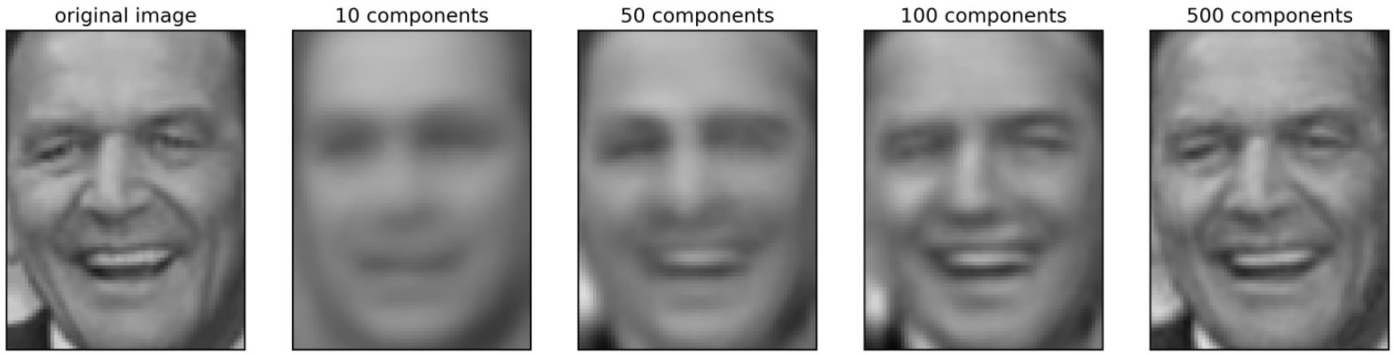
- Preprocess (i.e. standardize) variables, *usually*
- Find reconstruction/representation **x** by some criteria:
  - Minimal reconstruction error
  - Maximization variance of the unit vector in subspace $\mathcal{S}$

  ] Equivalent
- If reconstruction data lie near actual data, they can approximate distances
  - And if M << D, then it is much cheaper to work with **x** than with **z**
- The subspace mapping, easier to manipulate and visualize, is a representation of the data.

PCA as minimal reconstruction error

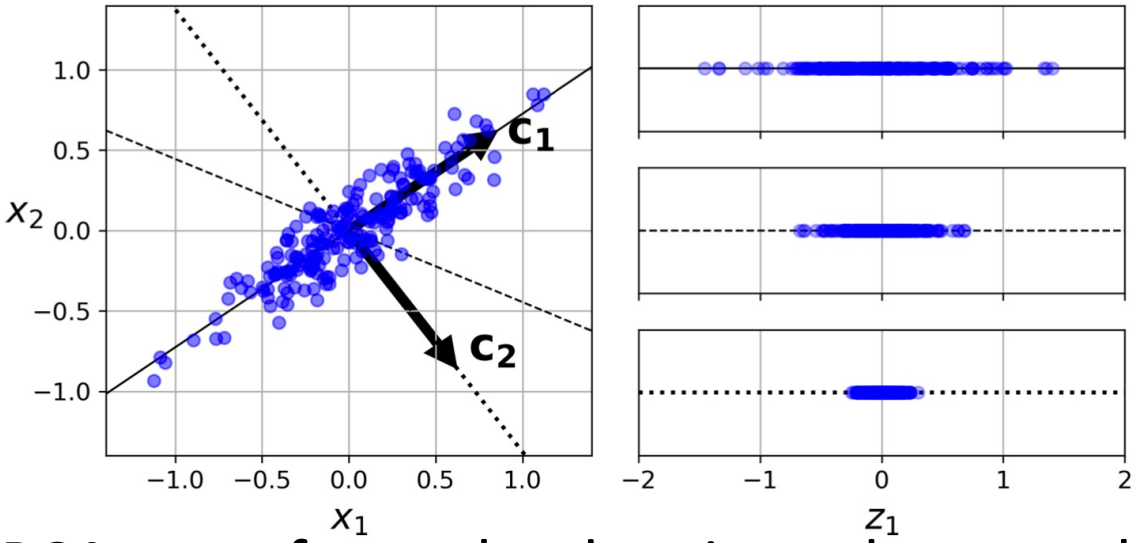
$$J(\mathbf{W}, \mathbf{Z}) = \frac{1}{N} \sum_{i=1}^{N} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2$$

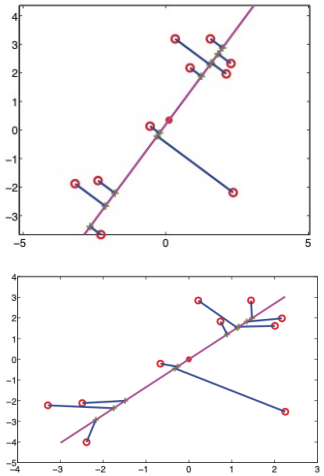




PCA as maximal variance

- PCA achieves maximal (explained) variance (loses the least amount of information).
- The *explained variance ratio is* used to select the best number of dimensions.



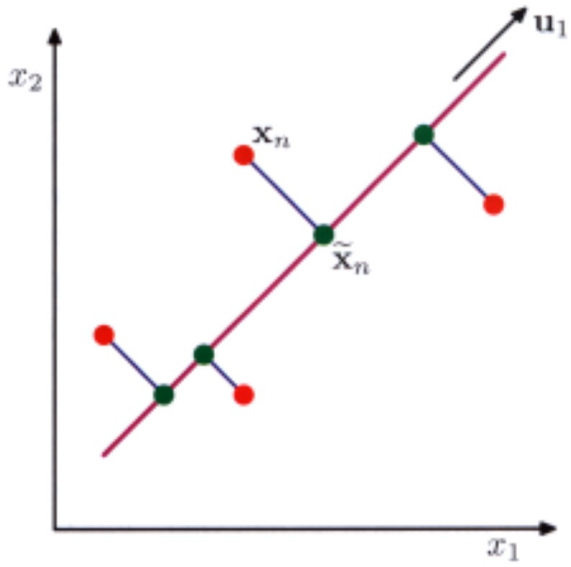


- PCA transform the data in such a way that the components are statistically uncorrelated with each other.

PCA - Maximum Variance Formulation

Correlation matrix eigenvectors and eigenvalues

- Example for a single PC (i.e. M=1)
- Project data points into $\mathbf{u}_1$ via $\mathbf{u}_1^{\mathrm{T}}\mathbf{x}_n$ subject to $\mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1 = 1$



- Subtract $\overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$
- Variance equal to

$$\frac{1}{N}\sum_{n=1}^{N}\left\{\mathbf{u}_1^{\mathrm{T}}\mathbf{x}_n - \mathbf{u}_1^{\mathrm{T}}\overline{\mathbf{x}}\right\}^2 = \mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1 \quad \text{where}$$

$$\mathbf{S} = \mathbf{X}\mathbf{X}^T$$

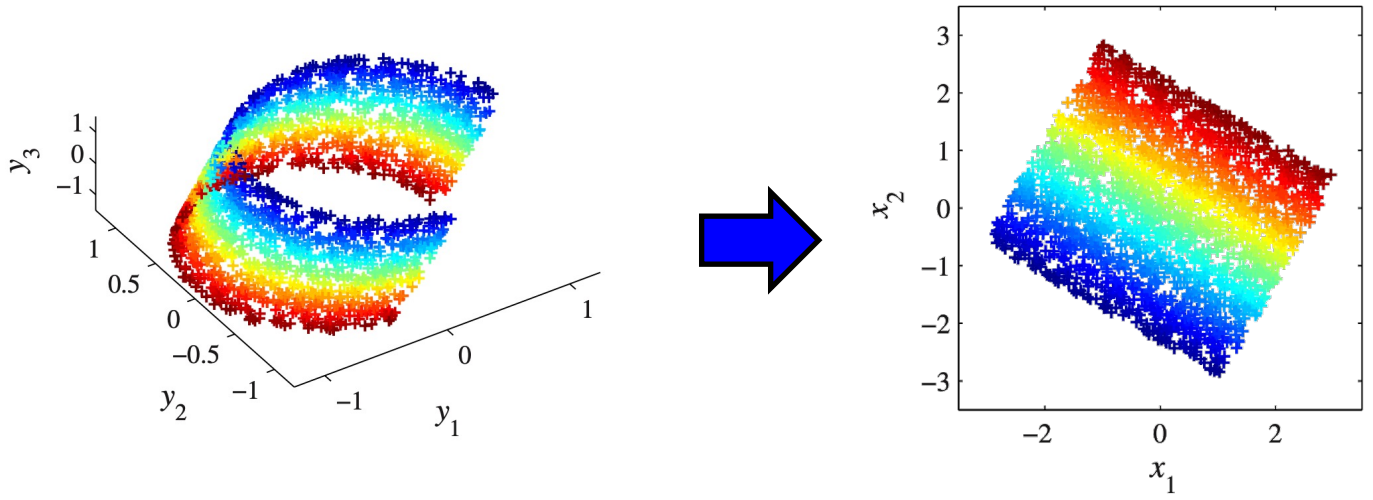
$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^{\mathrm{T}}.$$

- Find minimum via Lagrange multiplier:

$$\mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1 + \lambda_1\left(1 - \mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1\right). \qquad \mathbf{S}\mathbf{u}_1 = \lambda_1\mathbf{u}_1$$ 👍

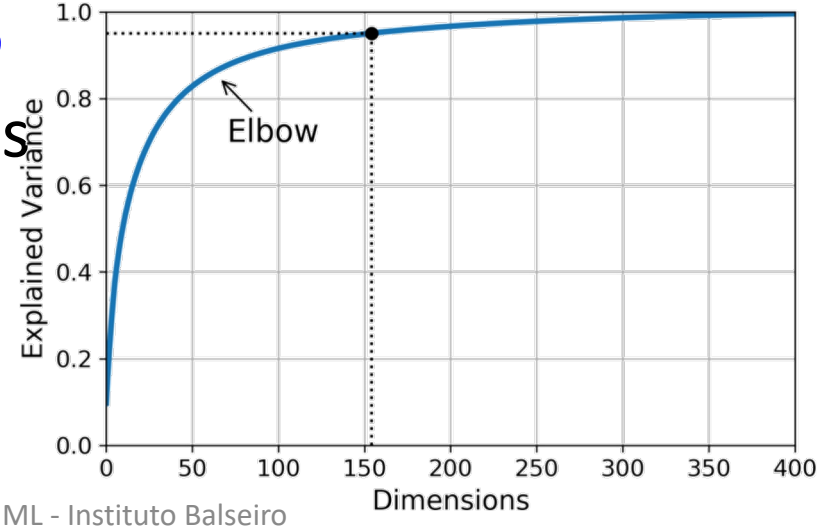
How many dimensions?

- Intrinsic dimensionality



- Use *explained variance ratio* as a function of dimension as an *ad-hoc* heuristic

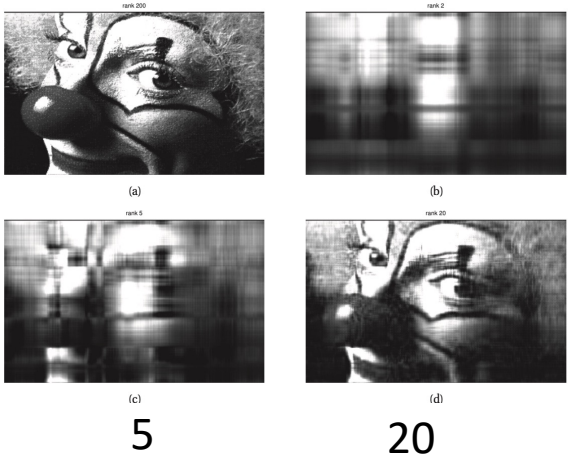
$$F(\mathcal{D}_{\text{train}}, L) = \frac{\sum_{j=1}^{L} \lambda_j}{\sum_{j'=1}^{L_{max}} \lambda_{j'}}$$

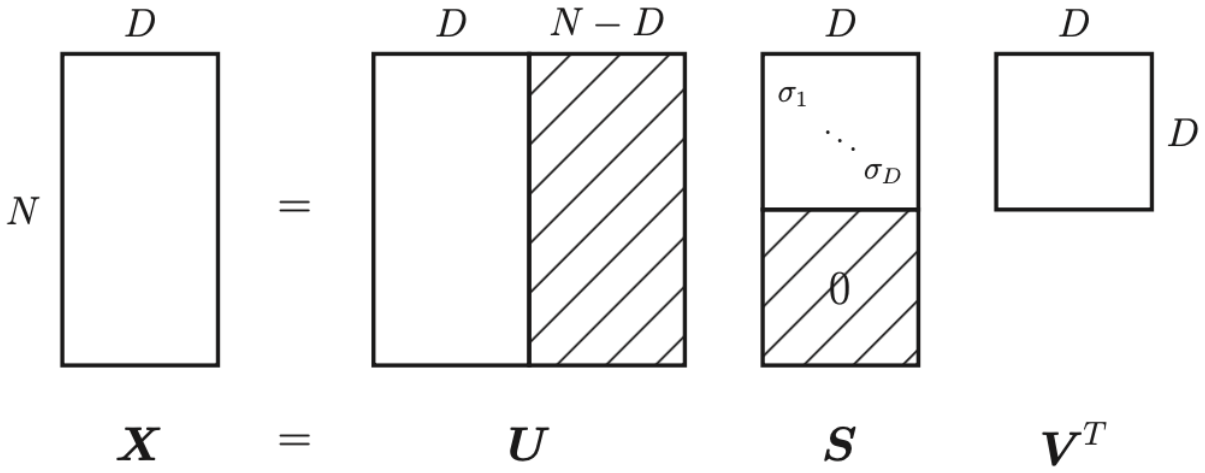


SVD

Economy-size:

$$\underbrace{\mathbf{X}}_{N\times D} = \underbrace{\mathbf{U}}_{N\times N}\ \underbrace{\mathbf{S}}_{N\times D}\ \underbrace{\mathbf{V}^T}_{D\times D}$$

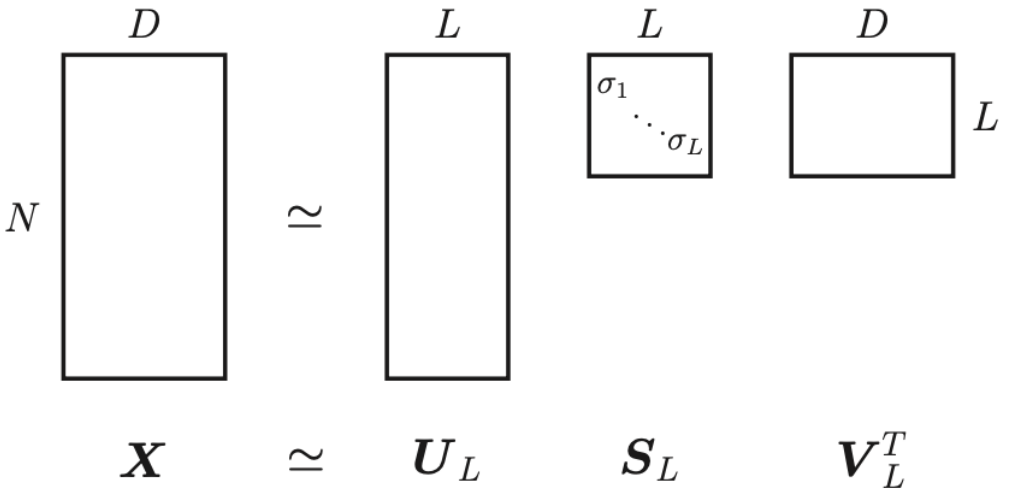




(a)

Truncated:

$$\underbrace{\mathbf{X}}_{N\times D} \simeq \underbrace{\hat{\mathbf{U}}_L}_{N\times L}\ \underbrace{\hat{\mathbf{S}}_L}_{L\times L}\ \underbrace{\hat{\mathbf{V}}_L^T}_{D\times L}$$

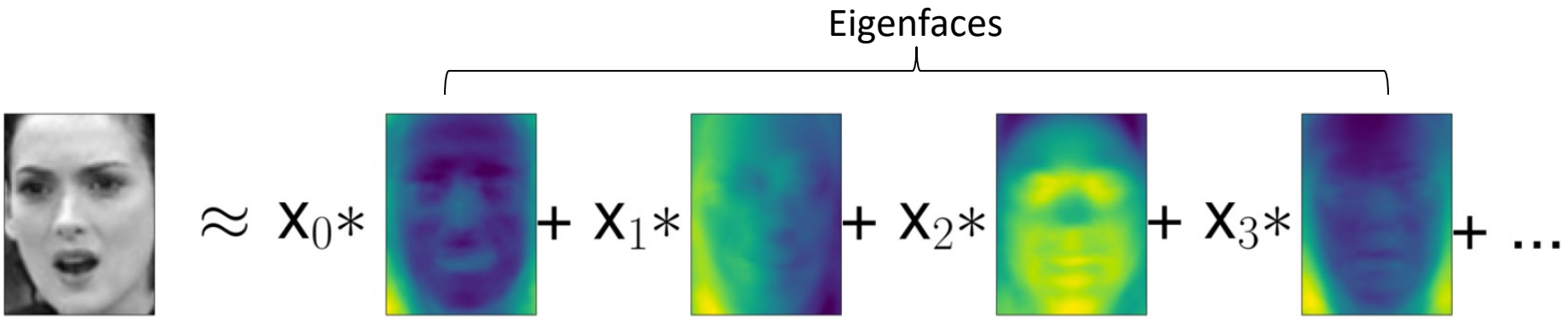


PCs are the eigenvectors of covariance matrix

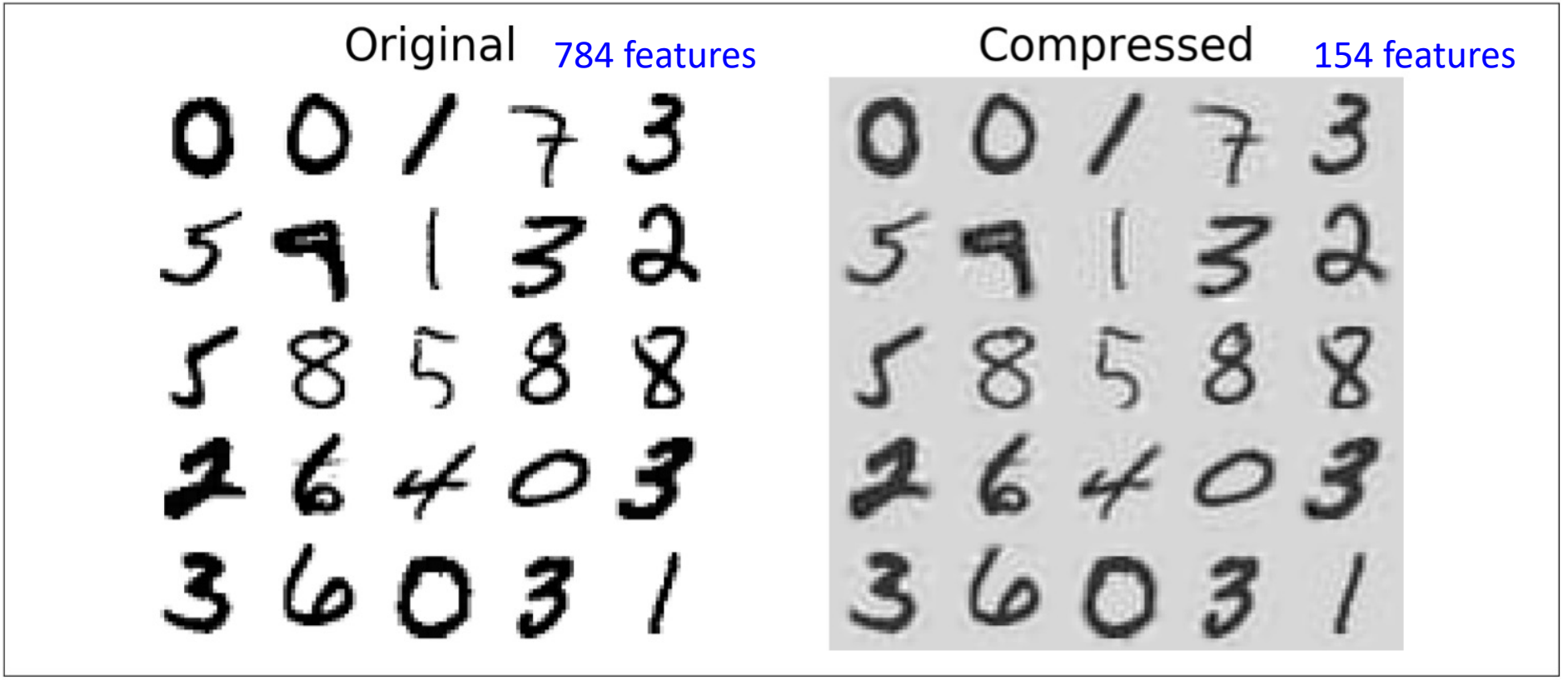
- PCs are the eigenvectors of the covariance matrix.
- A frequently used method derives the eigenvectors by means of Singular Value Decomposition (SVD).

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T \qquad \begin{aligned} \mathbf{V} &= \text{evec}(\mathbf{X}^T\mathbf{X}) \\ \mathbf{S}^2 &= \text{eval}(\mathbf{X}^T\mathbf{X}) \end{aligned}$$

- Typically, some trailing components are discarded

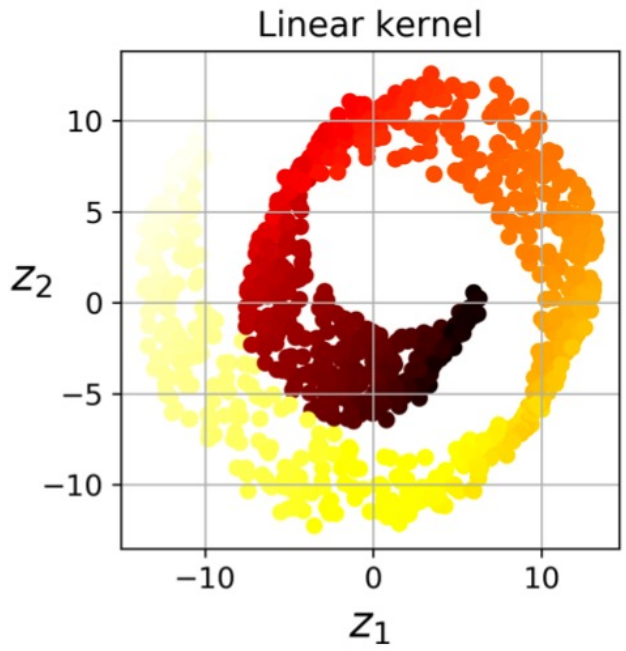


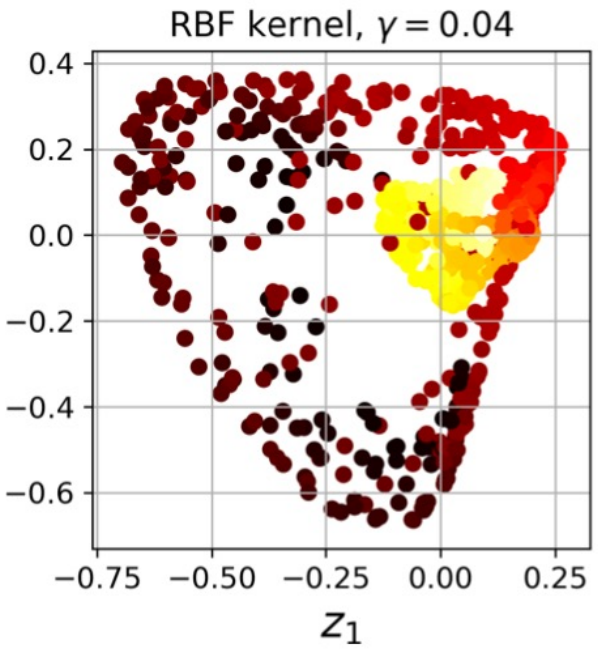
PCA for compression

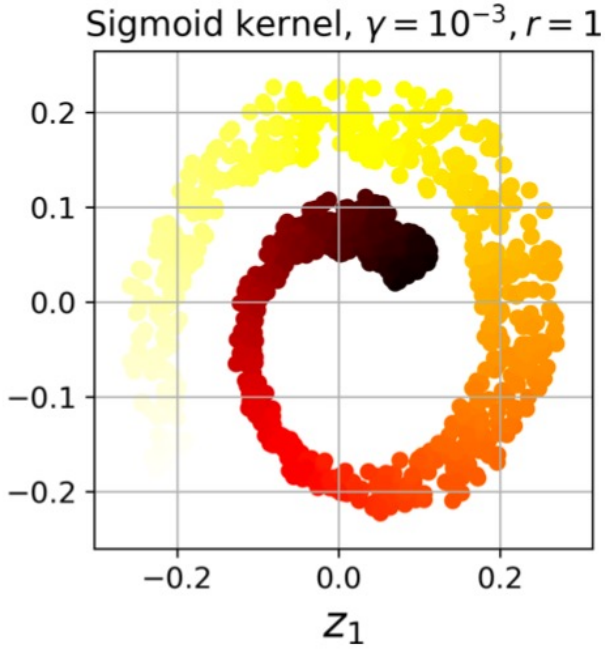


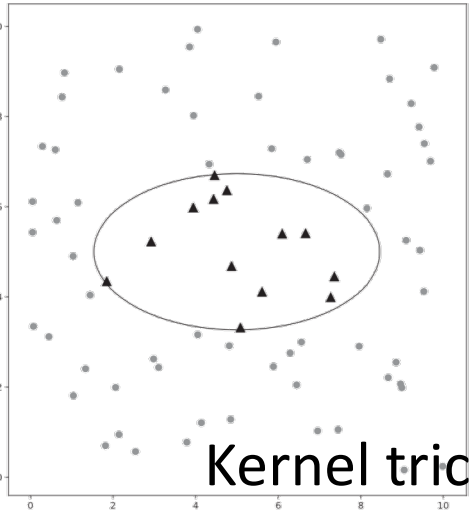
MNIST compression preserving 95% of the variance

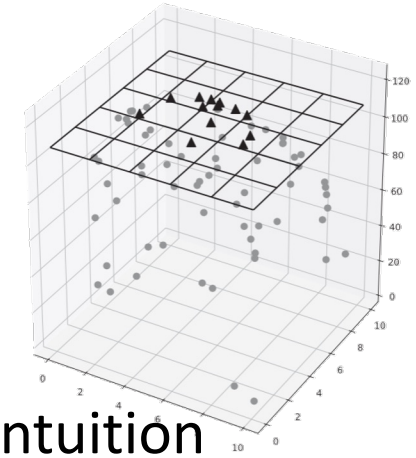
kernel PCA

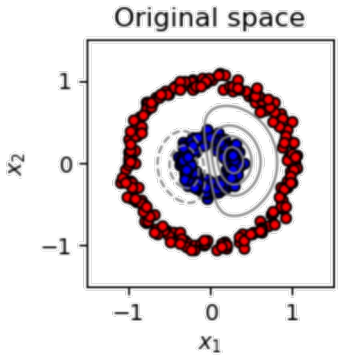


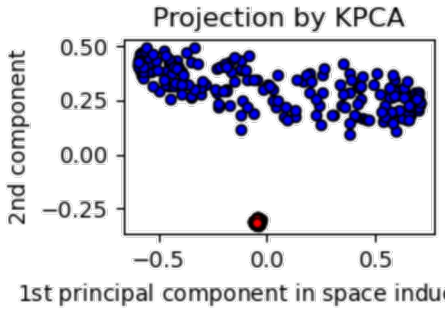






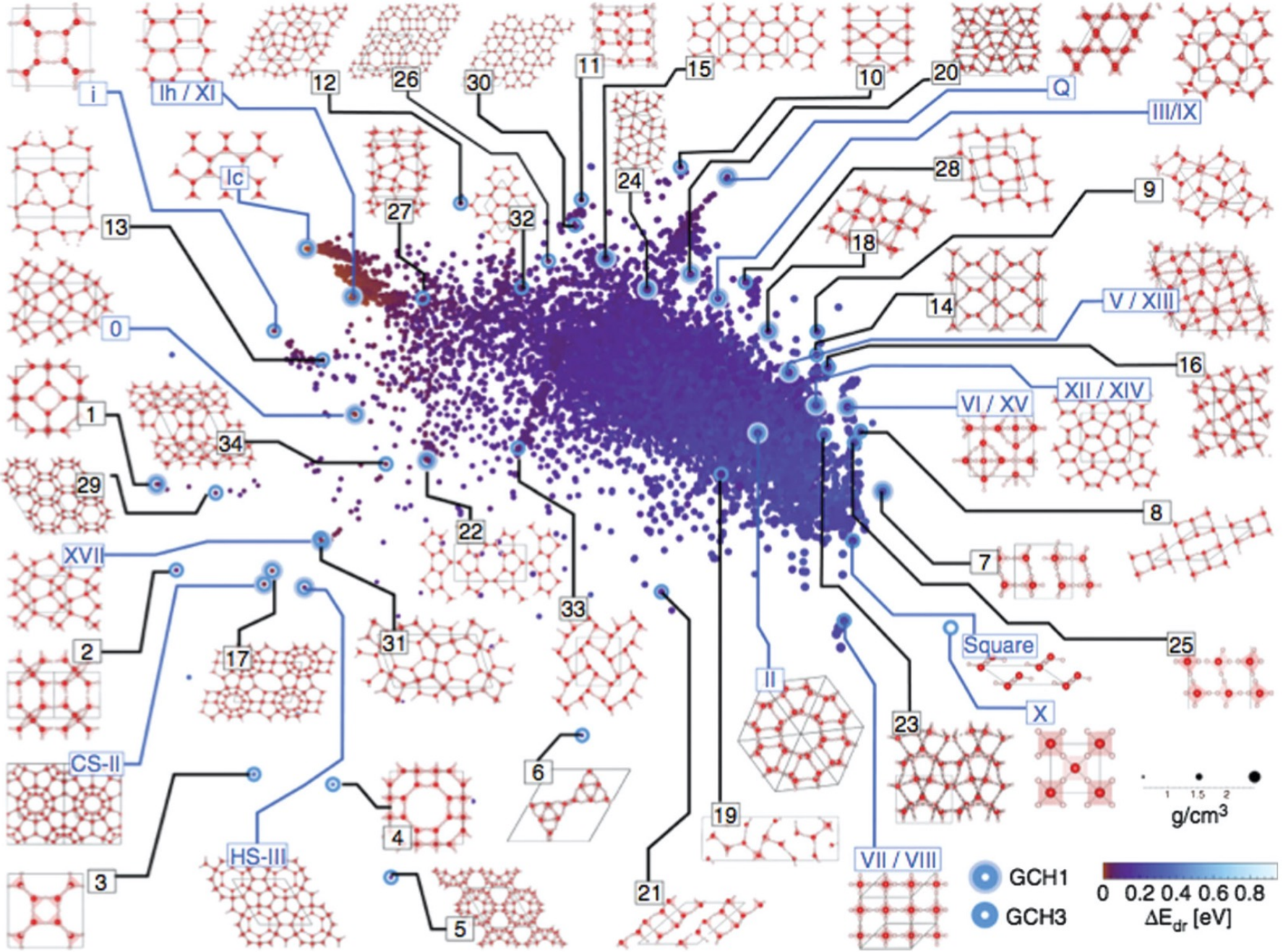






Kernel trick intuition

Kernel PCA for classification



kernel PCA in Engel, E. A., A. Anelli, M. Ceriotti, C. J. Pickard, and R. J. Needs, 2018, Nat. Commun. 9, 2173.

Multidimensional Scaling (MDS)

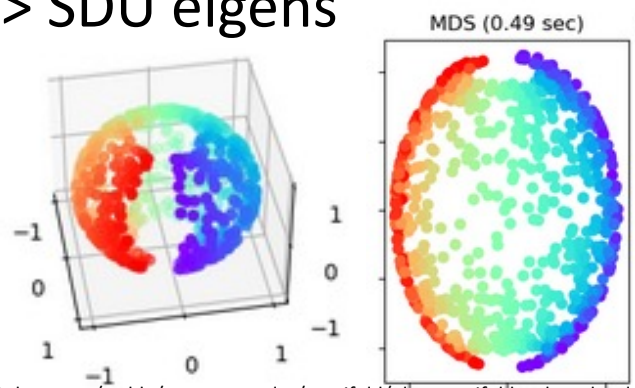
- Family of dimensionality reduction algorithms based on eigendecomposition
- Representation of data in low dimensionality aiming at preserving pairwise dissimilarities (distance):

$$d(\mathbf{y}(i), \mathbf{y}(j)) = \sqrt{\langle (\mathbf{y}(i) - \mathbf{y}(j)) \cdot (\mathbf{y}(i) - \mathbf{y}(j)) \rangle}$$

$$D := \begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,M} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,M} \\ \vdots & \vdots & & \vdots \\ d_{M,1} & d_{M,2} & \cdots & d_{M,M} \end{pmatrix}$$

Find vectors such that $\|x_i - x_j\| \approx d_{i,j}$

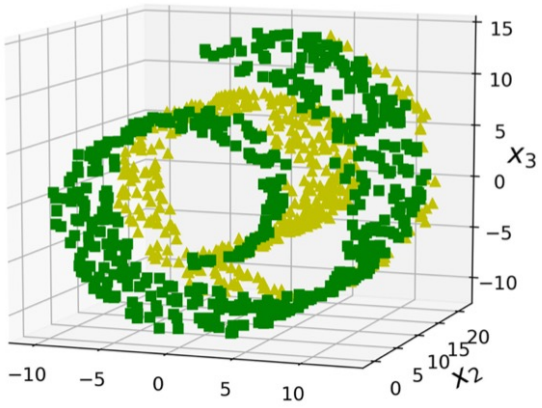
- Norm may be Euclidian (PCA) or other (metric/classic MDS)
- Distance -> dissimilarities (inner product) -> SDU eigens
- Target dimensions chosen beforehand
- **sklearn.manifold.MDS**



https://scikit-learn.org/stable/auto_examples/manifold/plot_manifold_sphere.html

ML Fundamentals – Lecture 7

- Unsupervised learning
- Clustering
  - k-means
  - Hierarchical methods
  - DBScan
- DR vs Representation vs Manifold learning
- Dimensionality reduction
  - PCA & kernel PCA
  - MDS



Next:
Manifold learning
& NLDR + Viz