UNSUPERVISED LEARNING

María Hernández Rubio - @maria_hr

About me...

María Hernández Rubio

@maria_hr

Mathematics & Computer Science, UAM
MsC Computational Intelligence, UAM

Senior Data Scientist at BBVA Data & Analytics

Joined BBVA in 2011, Smart Cities, external consultant
Joined Beeva (now BBVA Next Technologies) in 2013
Joined BBVA D&A in 2014

- Urban Analysis, C360, RecSys
- (Non-) Customer Intelligence
- Smart Replies (NLP)

Unsupervised learning

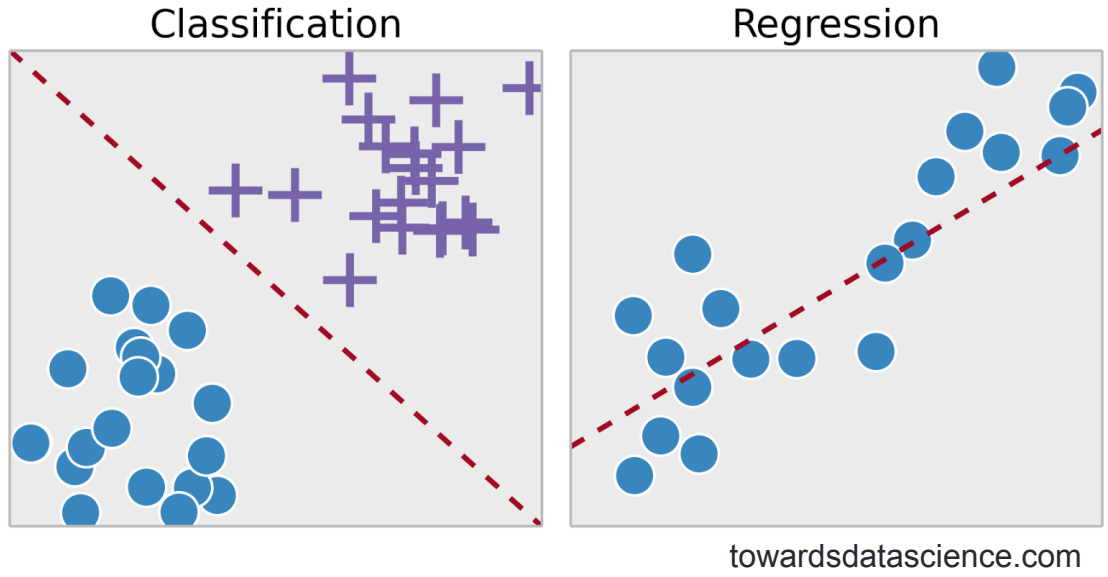
- **Unsupervised Learning vs Supervised Learning**
- **Clustering**
  - K-means
  - Hierarchical clustering
  - DBScan & OPTICS
- **Dimmensionality Reduction**
  - PCA
  - Matrix Factorization
    - SVD
    - NMF
  - Autoencoders
- **Additional Material**

UNSUPERVISED LEARNING

VS Supervised Learning

Supervised learning

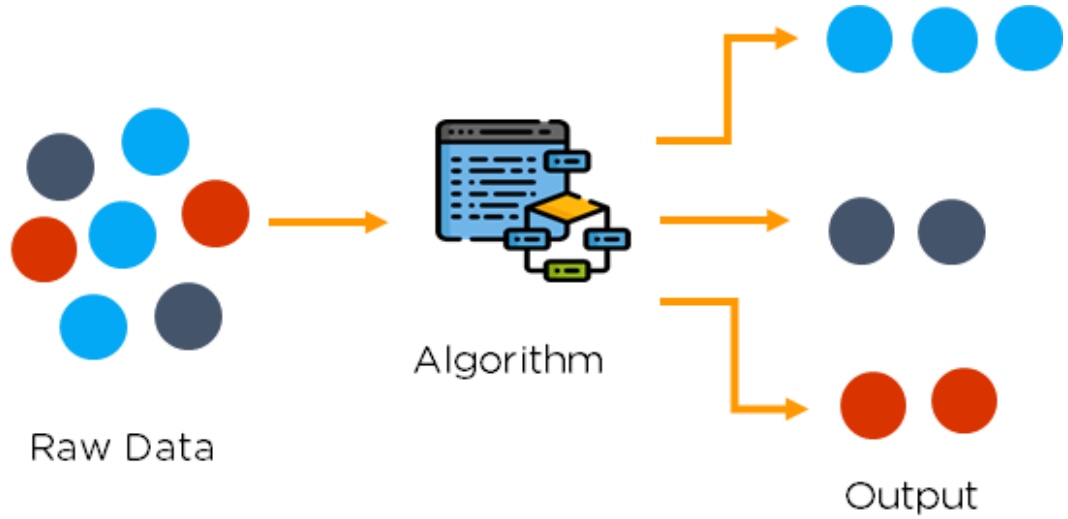
- In supervised learning, the objective is to **estimate the true label** (class/ number) from the available data **features**.
- We *do* have a ground truth
- Training consists of maximizing/ minimizing a function of real and estimated values.
  - Regression: minimizing *difference* between *y* and *y'* (*MAE, MSE, RMSE*,…)
  - Classification: maximizing *precission*, *accuracy*, *F1*,…



Training set: $\{(\mathbf{x}^1, y^1), (\mathbf{x}^2, y^2), \ldots, (\mathbf{x}^m, y^m)\}$

UNsupervised learning

- In **un**supervised learning, the objective is to **learn the inherent structure** in the data.
- Main example: clustering
- No labels are provided, so no objective way to compare model perfomance. Algorithms are heuristic, not model-based.

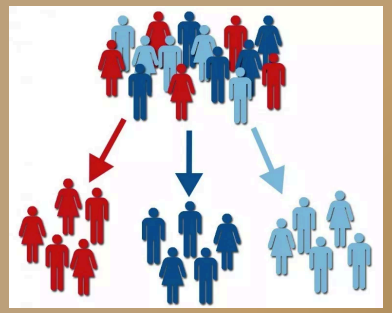


Source: Quora

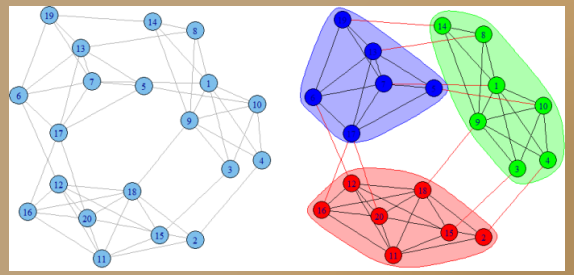
Training set: $\{x^1, x^2, \ldots, x^m\}$

Applications of unsupervised learning

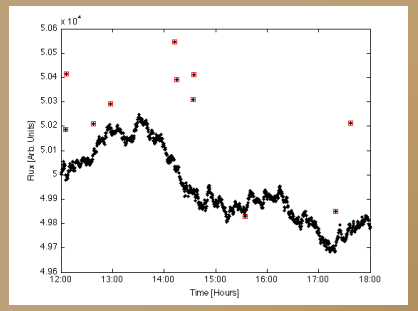
Segmentation



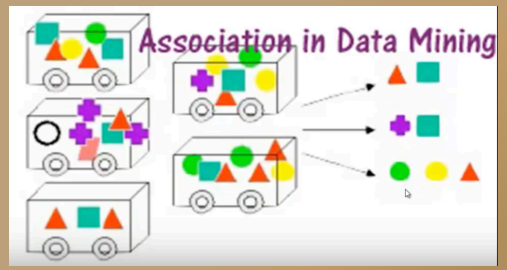
Social Network Analysis



Outlier detection



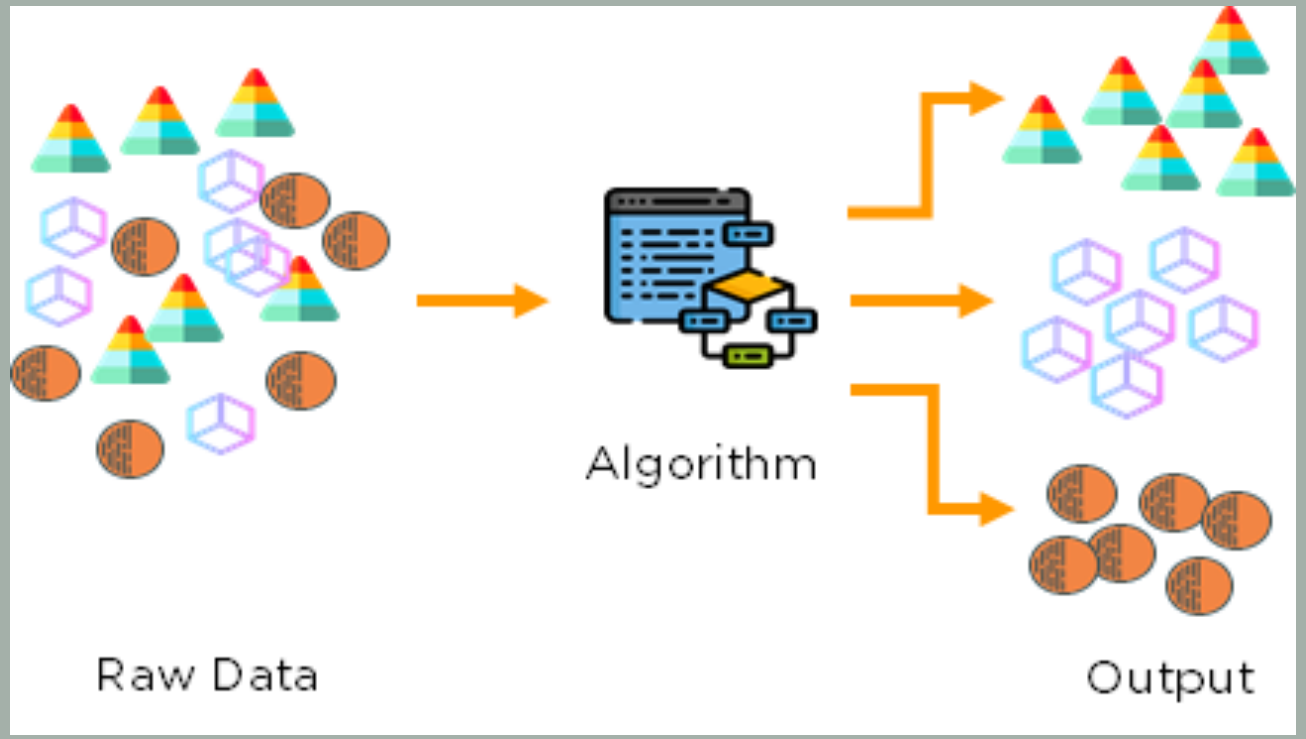
Association Rules



CLUSTERING

Grouping data

Clustering



Source: Quora

Unsupervised learning

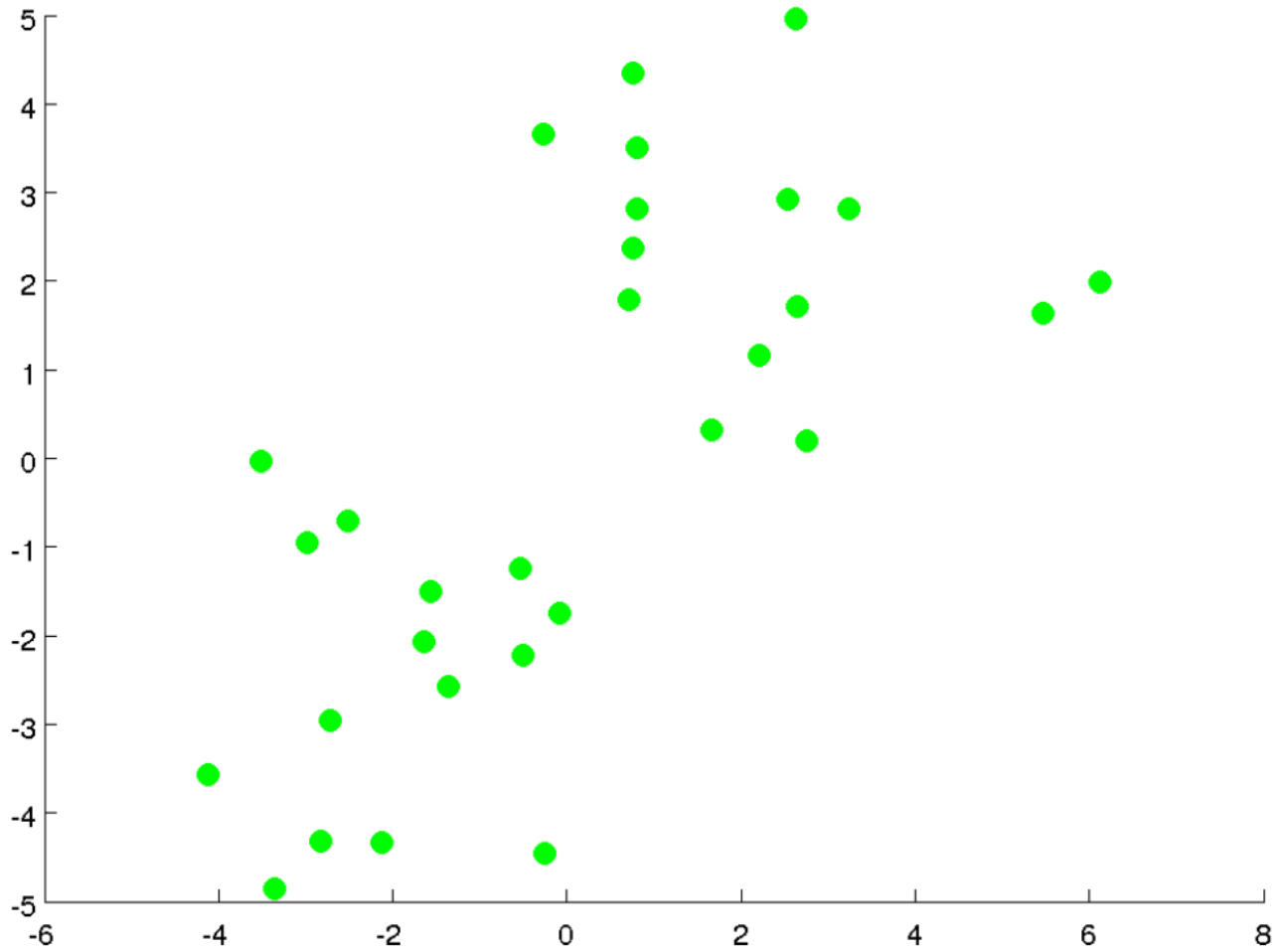
- Unsupervised Learning vs Supervised Learning
- **Clustering**
  - **K-means**
  - Hierarchical clustering
  - DBScan
  - OPTICS
- Dimmensionality Reduction
  - PCA
  - Matrix Factorization
    - SVD
    - NMF
  - Autoencoders
- Additional Material

K-means

Input data:

- K (number of clusters)
- Training set $\{x^1, x^2, \ldots, x^n\}$

1. Assign the points to closest centroid
2. Recompute centroids



Images from Machine Learning course @Coursera

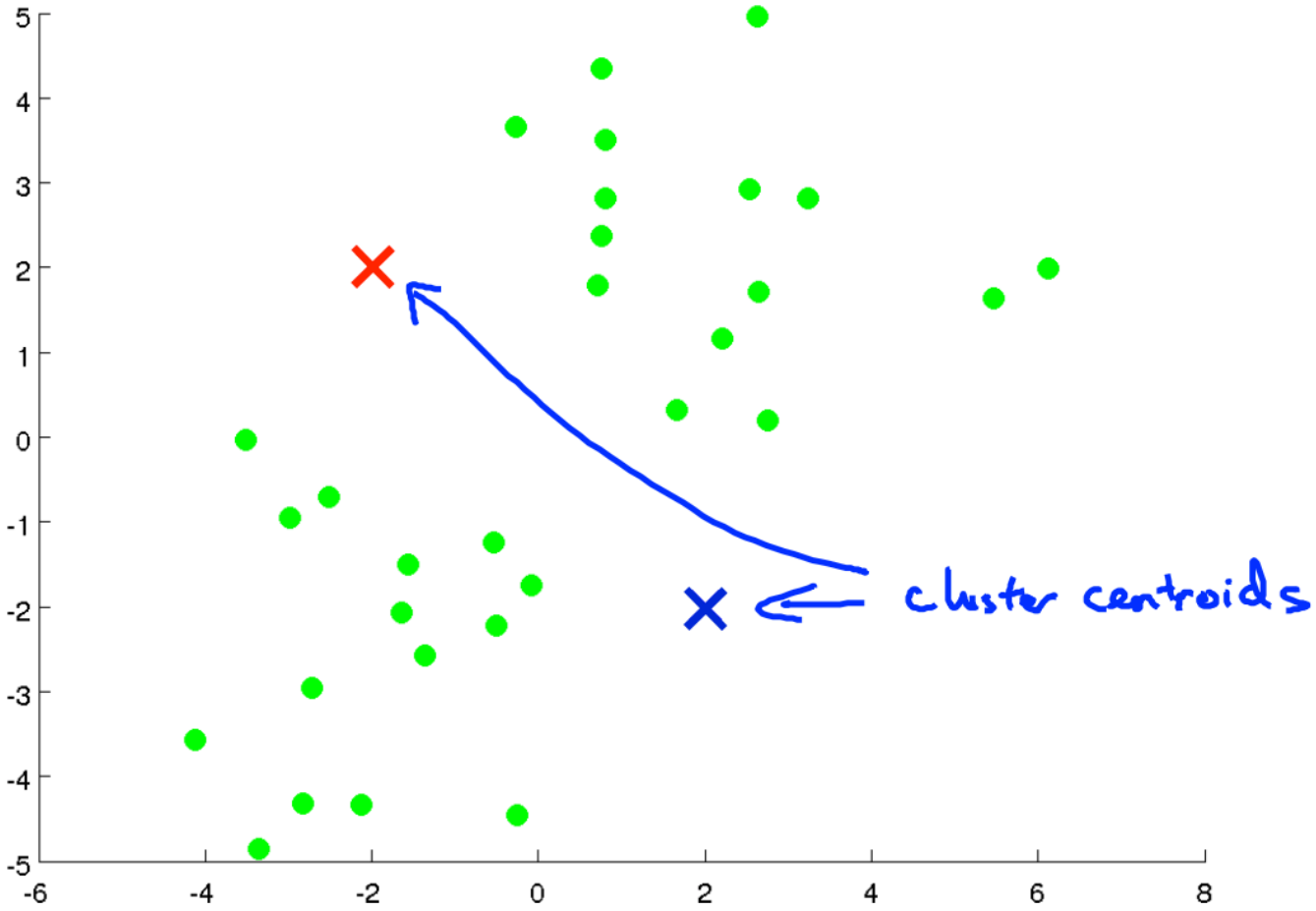
K-means

Input data:

- K (number of clusters)
- Training set $\{x^1, x^2, \ldots, x^n\}$

1. Assign the points to closest centroid
2. **Recompute centroids**

(first iteration initialize random)



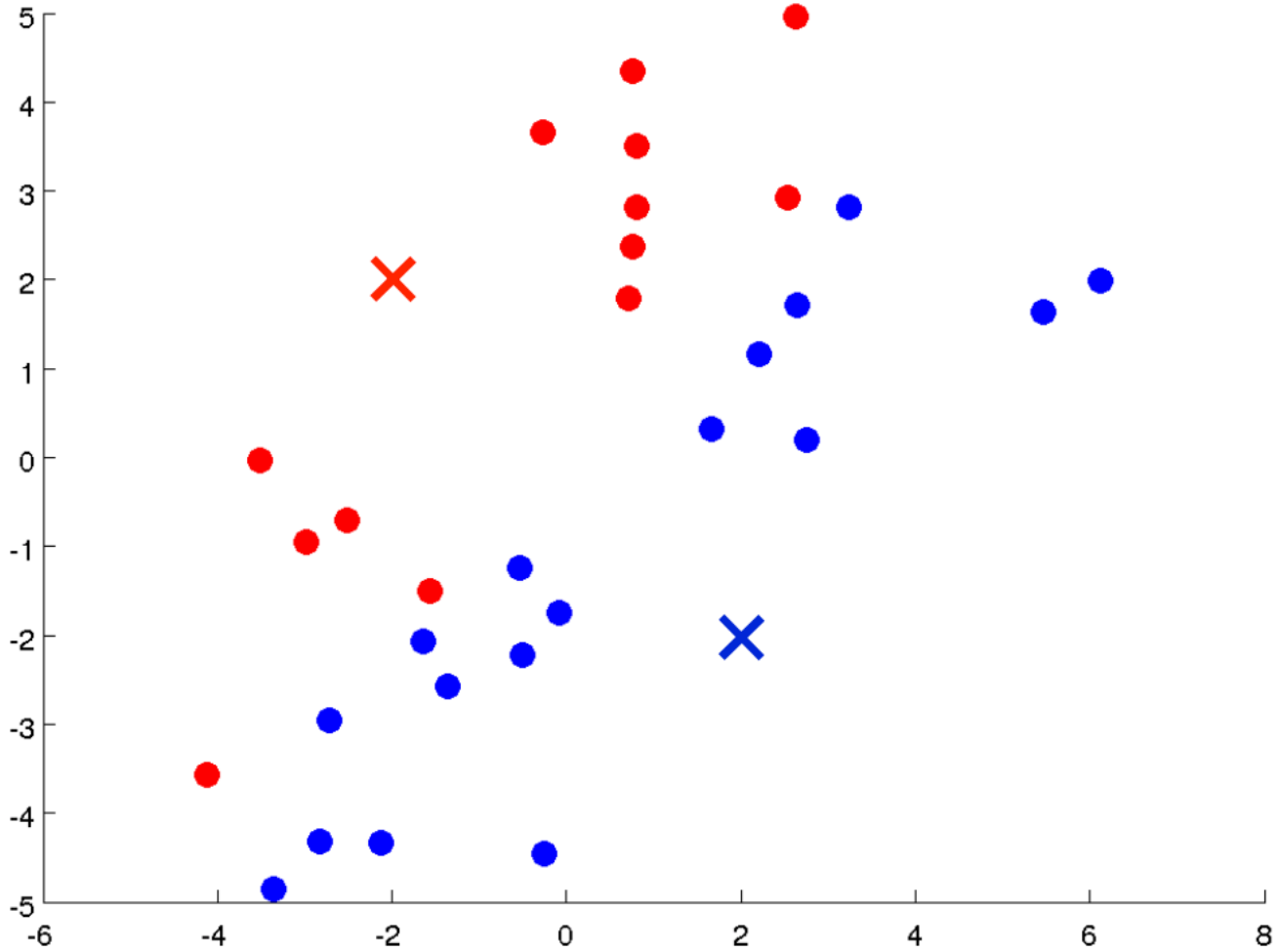
Images from Machine Learning course @Coursera

K-means

Input data:

- K (number of clusters)
- Training set $\{x^1, x^2, \ldots, x^n\}$

1. **Assign the points to closest centroid**
2. Recompute centroids



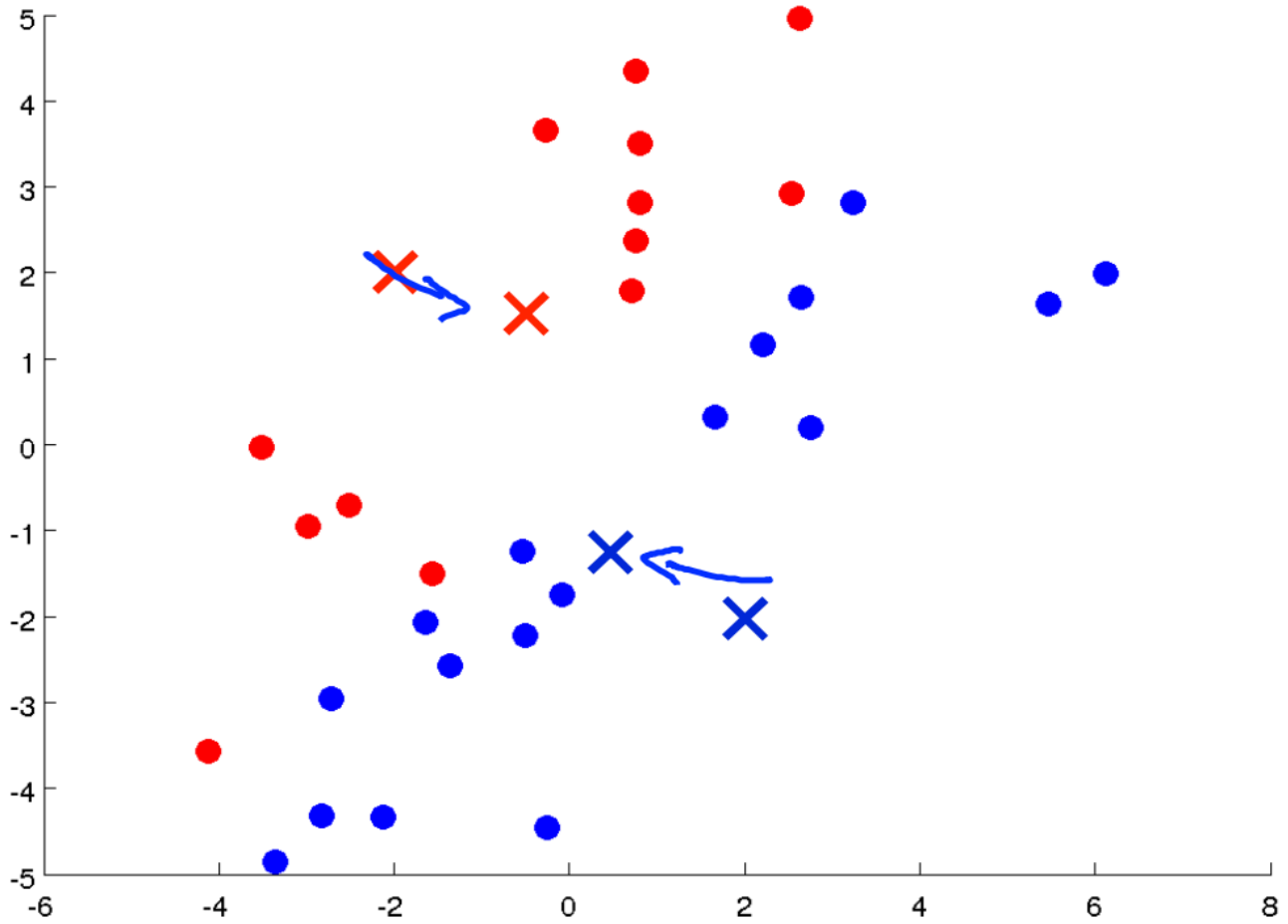
Images from Machine Learning course @Coursera

K-means

Input data:

- K (number of clusters)
- Training set $\{x^1, x^2, \ldots, x^n\}$

1. Assign the points to closest centroid
2. **Recompute centroids**



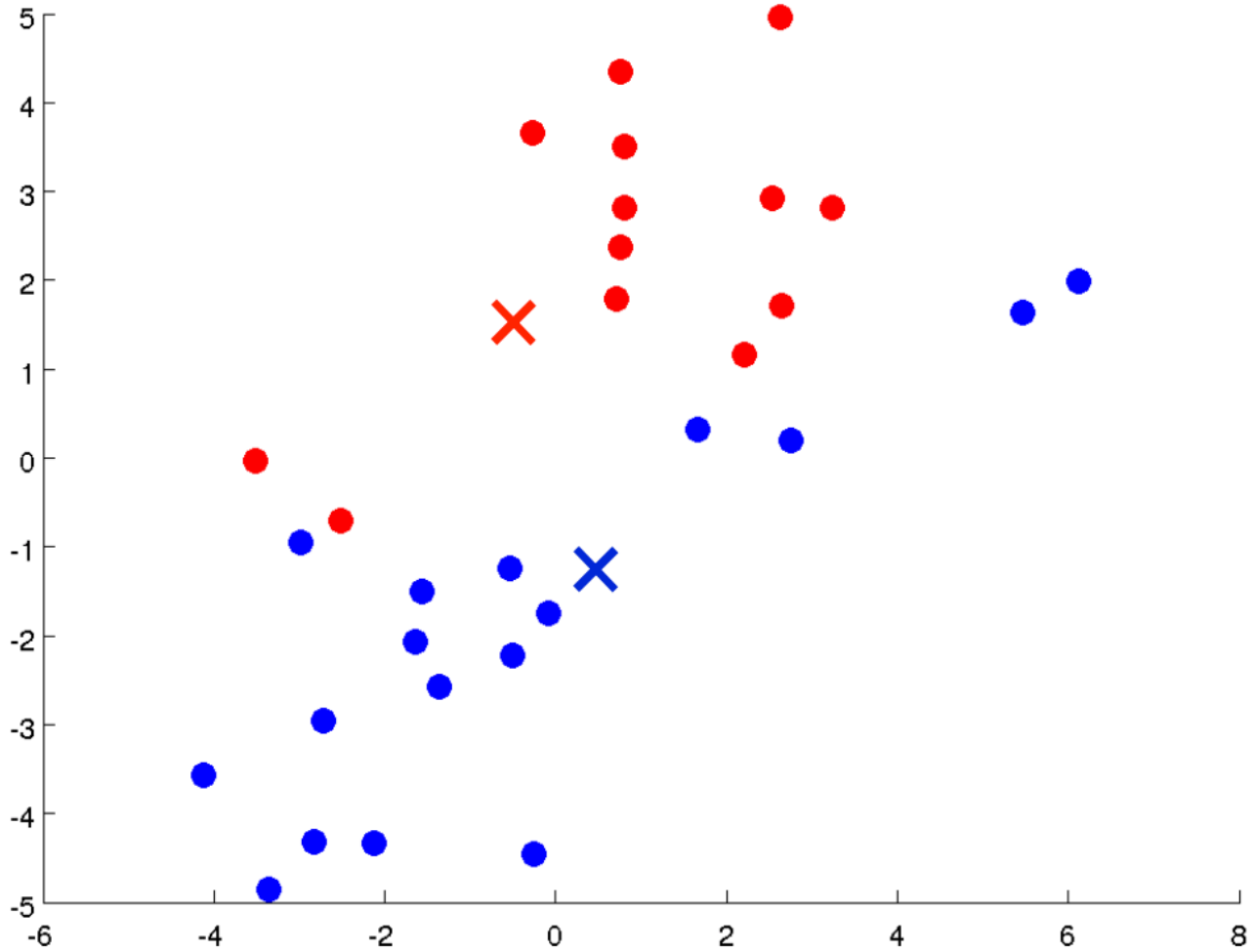
Images from Machine Learning course @Coursera

K-means

Input data:

- K (number of clusters)
- Training set $\{x^1, x^2, \ldots, x^n\}$

1. **Assign the points to closest centroid**
2. Recompute centroids



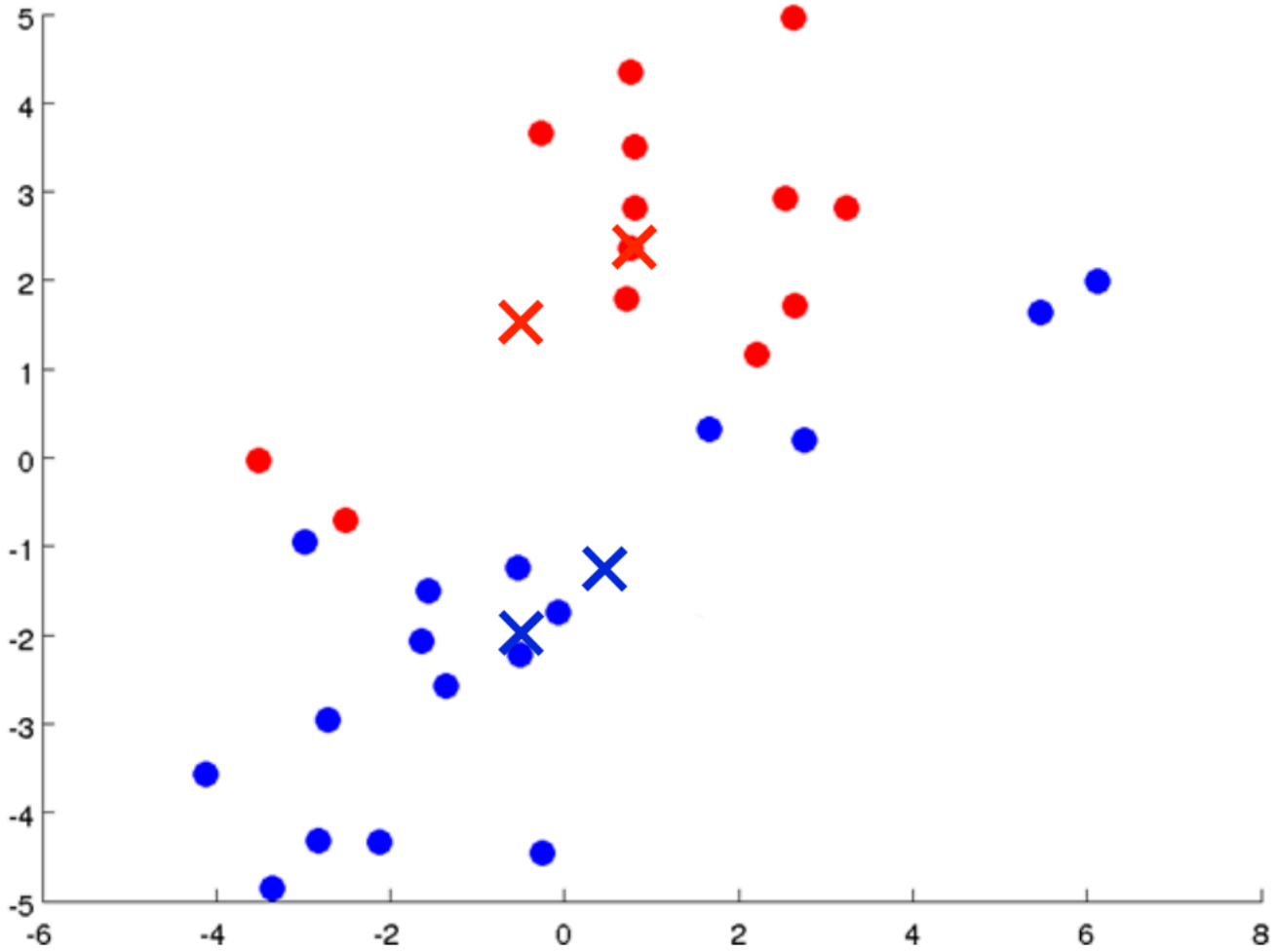
Images from Machine Learning course @Coursera

K-means

Input data:

- K (number of clusters)
- Training set $\{x^1, x^2, \ldots, x^n\}$

1. Assign the points to closest centroid
2. **Recompute centroids**



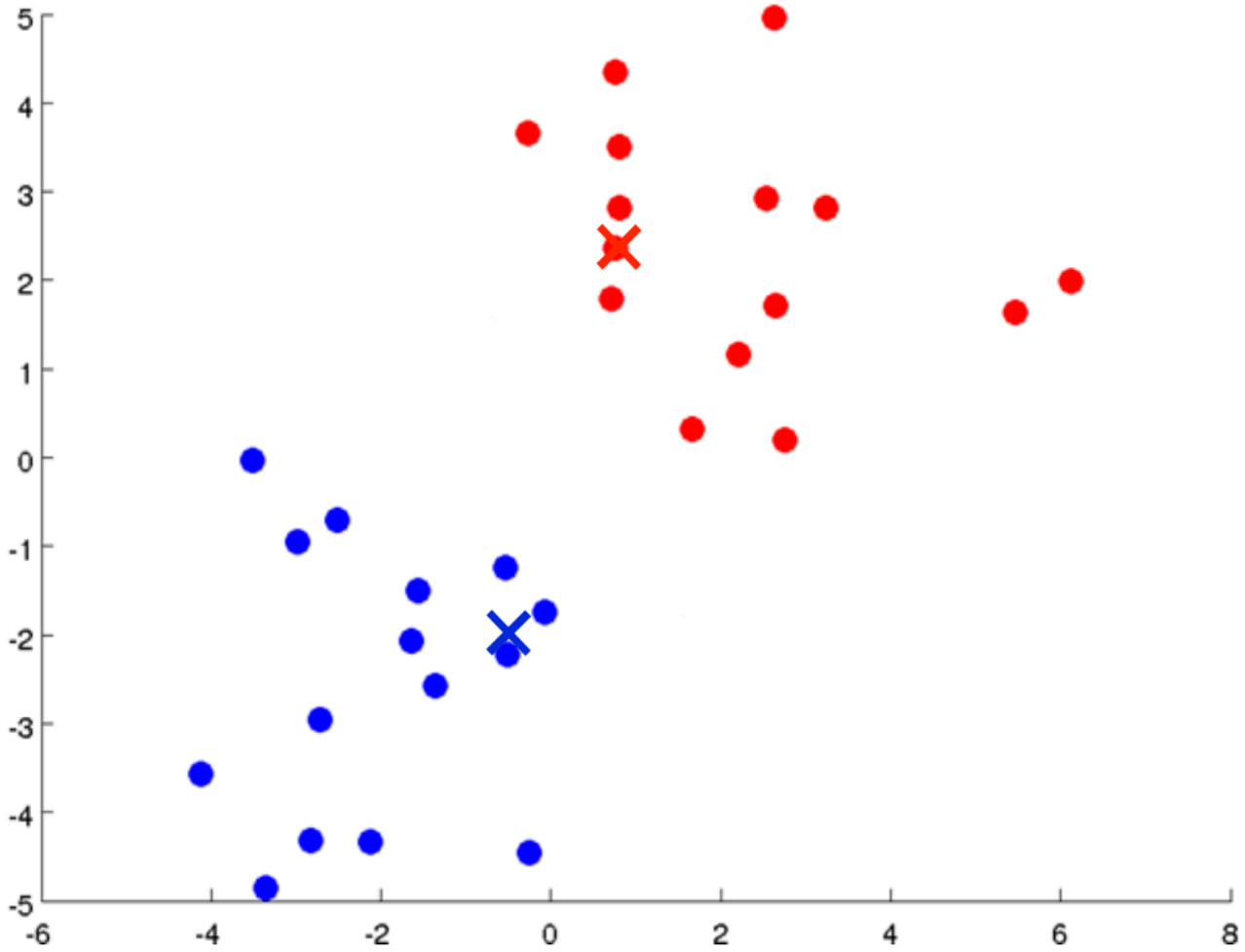
Images from Machine Learning course @Coursera

K-means

Input data:

- K (number of clusters)
- Training set $\{x^1, x^2, \ldots, x^n\}$

1. **Assign the points to closest centroid**
2. Recompute centroids



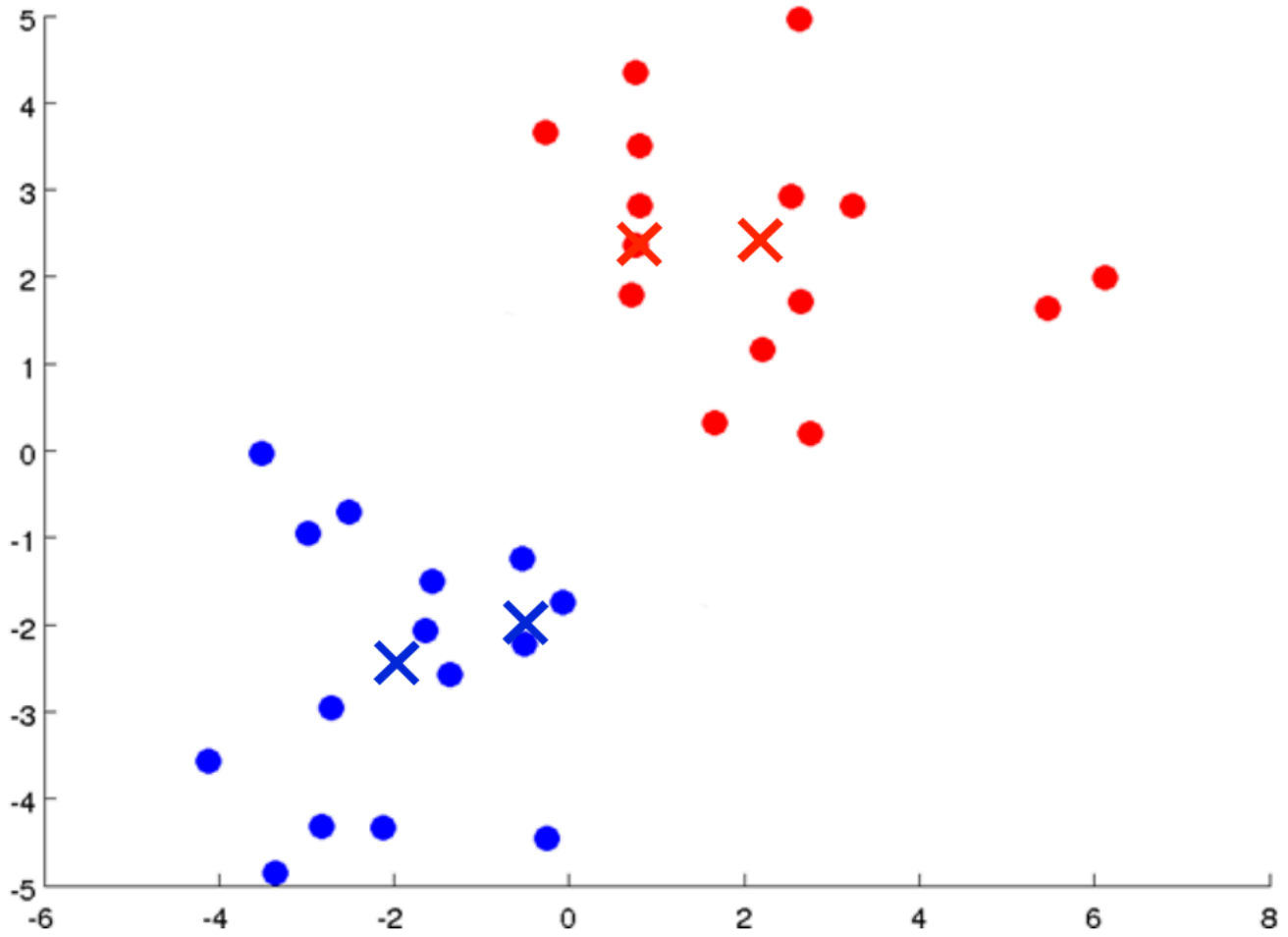
Images from Machine Learning course @Coursera

K-means

Input data:

- K (number of clusters)
- Training set $\{x^1, x^2, \ldots, x^n\}$

1. Assign the points to closest centroid
2. **Recompute centroids**



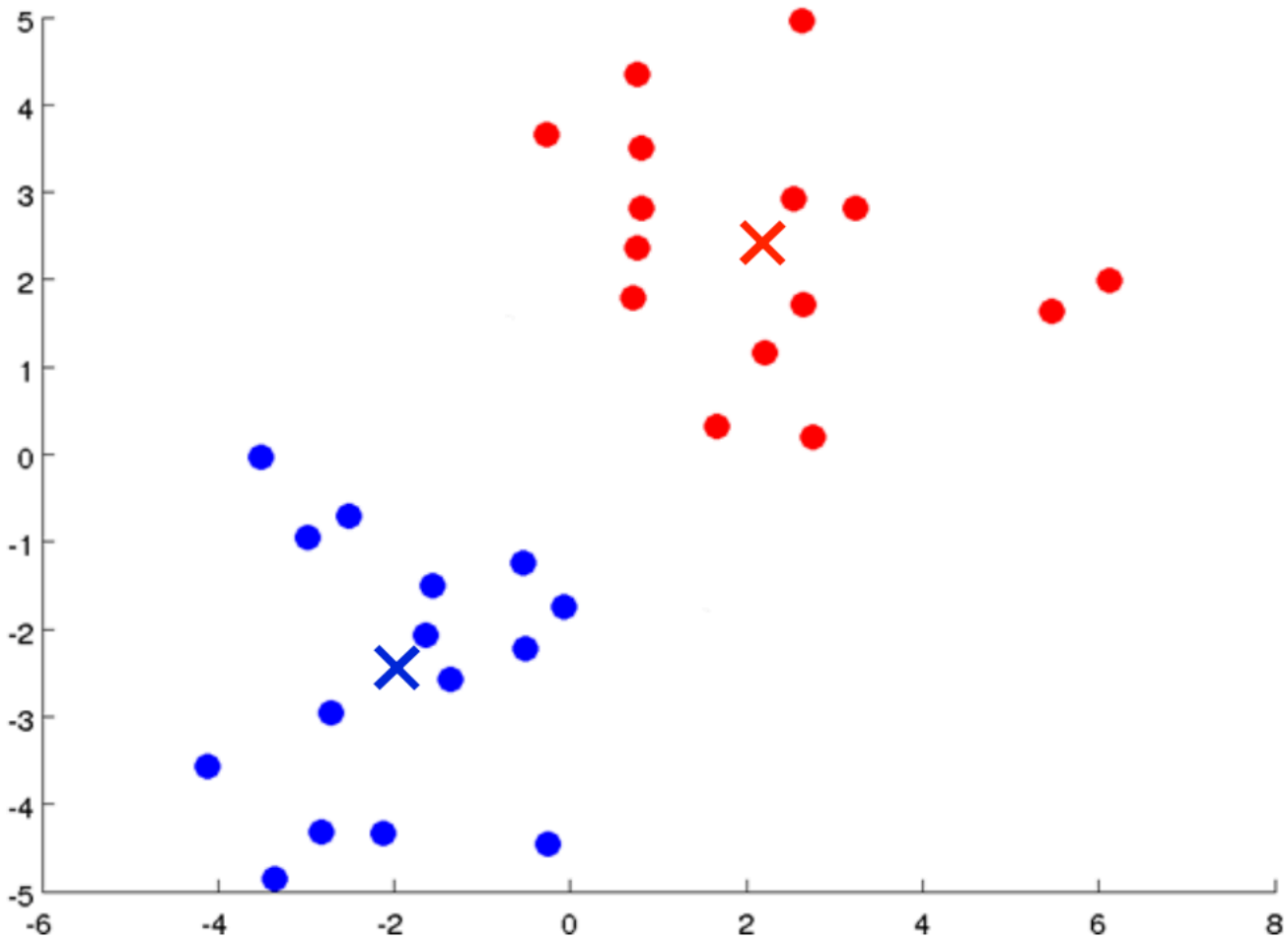
Images from Machine Learning course @Coursera

K-means

Input data:

- K (number of clusters)
- Training set $\{x^1, x^2, \ldots, x^n\}$

1. **Assign the points to closest centroid**
2. Recompute centroids



Images from Machine Learning course @Coursera

K-means

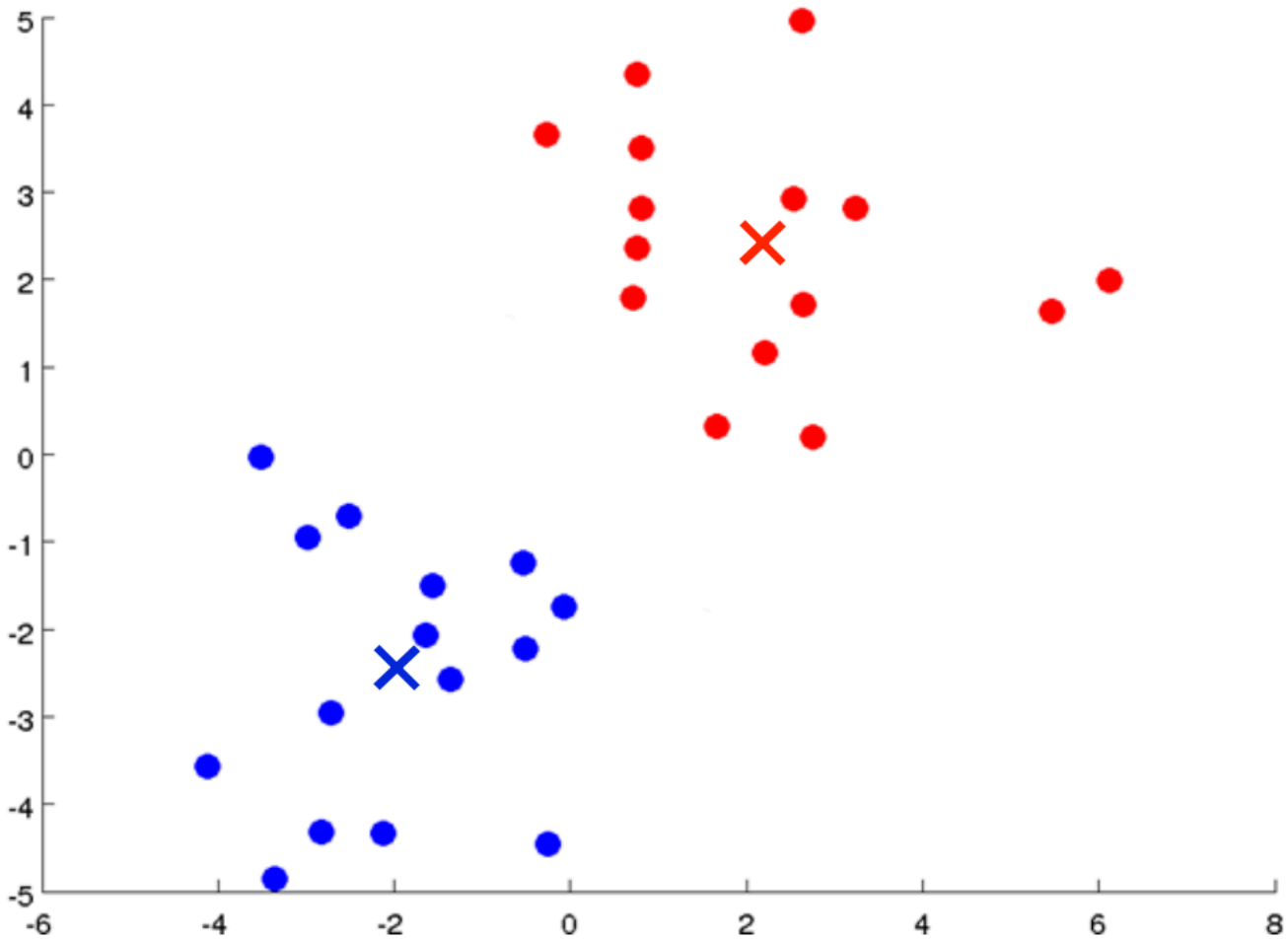
Input data:

- K (number of clusters)
- Training set $\{x^1, x^2, \ldots, x^n\}$

1. Assign the points to closest centroid
2. **Recompute centroids**

(no change)

END.



Images from Machine Learning course @Coursera

K-means algorithm

```
Kmeans(K, data):
      Randomly initialize K cluster centroids (mu(1),..., mu(k))
      Repeat until convergence:
            # assign cluster
            for d in data:
                  assign d to closest cluster centroid

            # recompute clusters' centroid
            for k = 1 to K:
                  mu(k) = mean of data points assigned to cluster k
```

You will code it in notebook!

Quality of K-means clustering

How to choose K (number of clusters)?

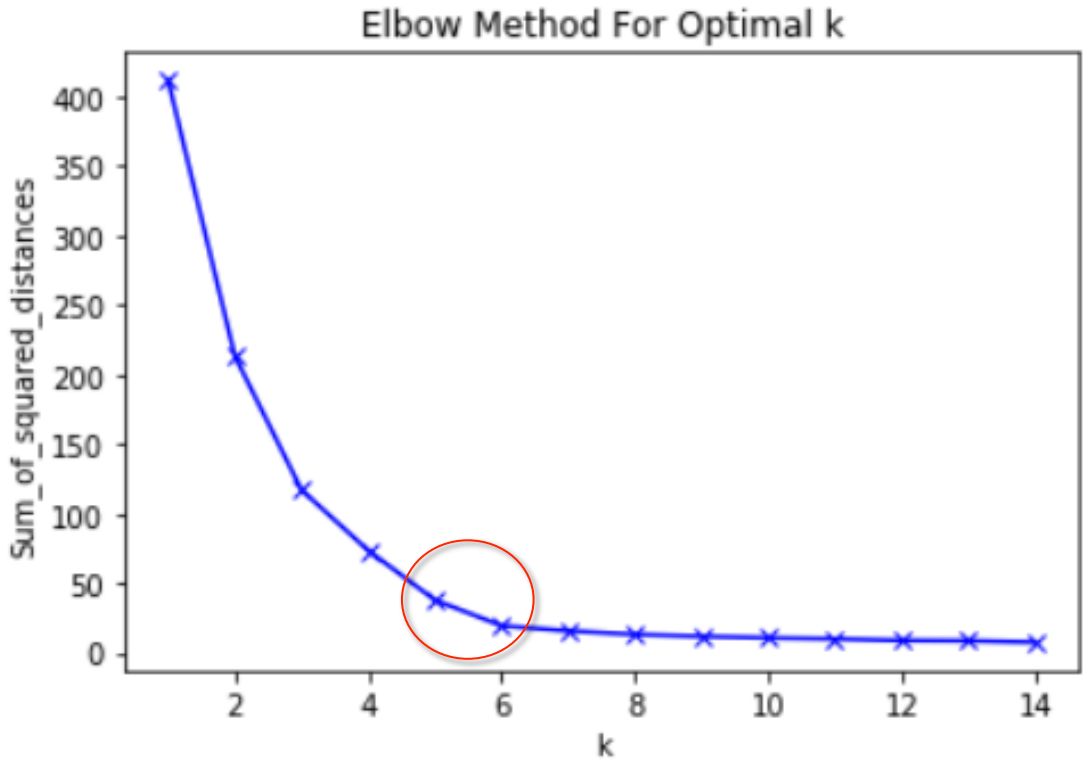
Elbow method

Optimization function

$$\min J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K) = \min \sum_{i=1}^{m} ||x^{(i)} - \mu_{c(i)}||^2$$

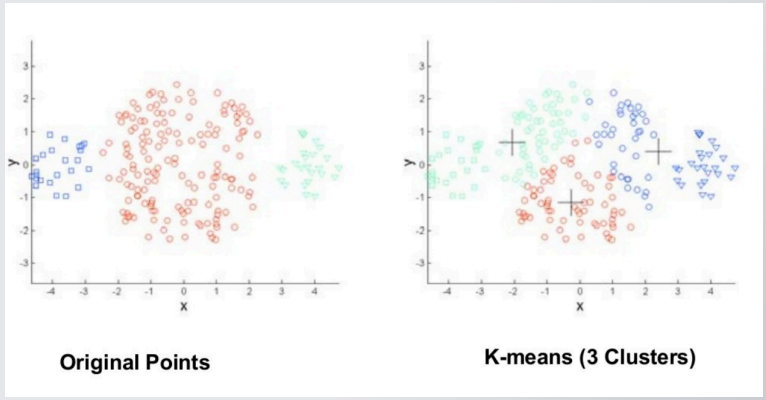
Compute the cost metric for different *partitions* (different number of K or different initialization)

Plot cost function versus number of clusters and choose K such that the cost decreases the velocity of improvement

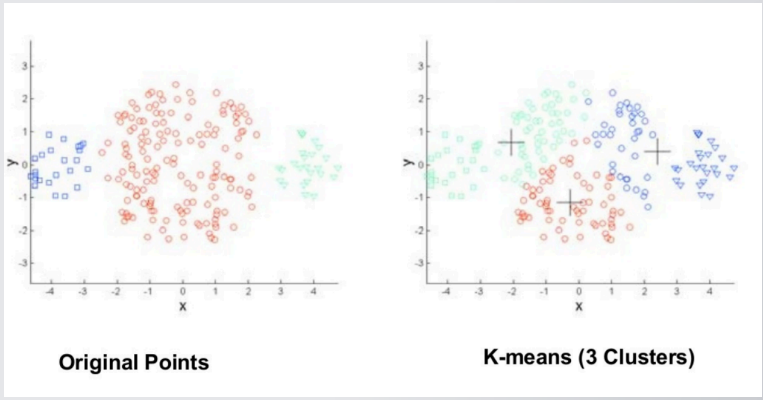


K-means limitations

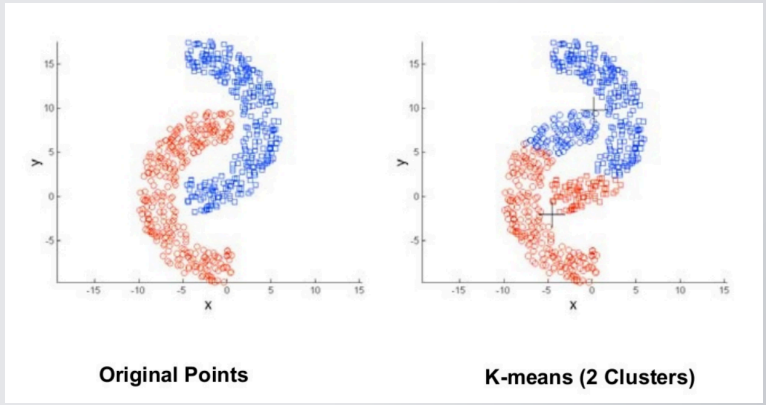
Clusters of different sizes



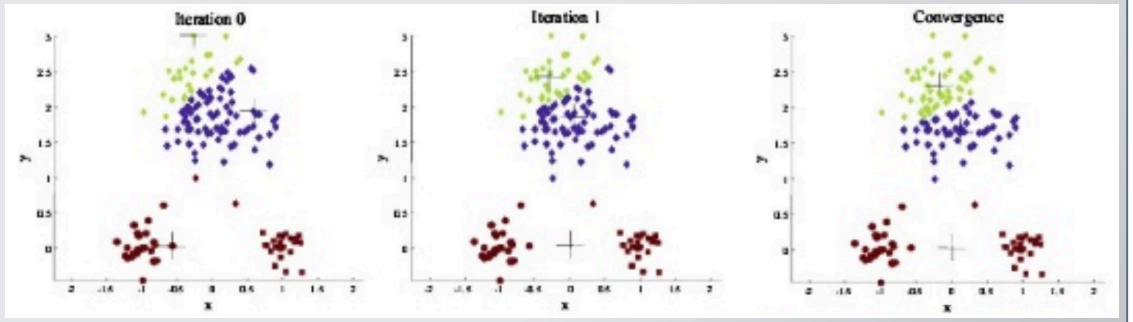
Clusters of different density



Assumes spherical distribution



Very sensitive to initialization



Images from “Algorithmic methods of Data Science”, Sapienza University

K-means

Some observations:

- It is common to choose initial centroids as K random points from training data.
- What is **convergence**?
  - #points that switch cluster, distance of new centroids to old ones, #iterations,…
- Limitations:
  - Most important: assumes spherical data.
  - Sensitive to initialization. Solution: repeat many times and choose best solution.
  - Outliers can cause problems
  - Different densities and sizes can cause problems

K-means - Code

R

```
cl <- kmeans(iris[,1:4], 3) #stats package

help(kmeans)
cl$cluster      # A vector of cluster id of each point
cl$centers      # A matrix of cluster centres
cl$totss        # Total sum of squares
cl$withinss     # Vector of within-cluster sum of squares
cl$tot.withinss # Total within-cluster sum of squares (=sum(withinss))
cl$betweenss    # between-cluster ss (=totss - tot.withinss)
cl$size         # Number of points in each cluster
```

Python

```
from sklearn.cluster import KMeans
model = KMeans(n_clusters = 3)
model = model.fit(data)

model.cluster_centers_
model.labels_
model.inertia_            # sse
```

Unsupervised learning

- Unsupervised Learning vs Supervised Learning
- **Clustering**
  - K-means
  - **Hierarchical clustering**
  - DBScan
  - OPTICS
- Dimmensionality Reduction
  - PCA
  - Matrix Factorization
    - SVD
    - NMF
  - Autoencoders
- Additional Material

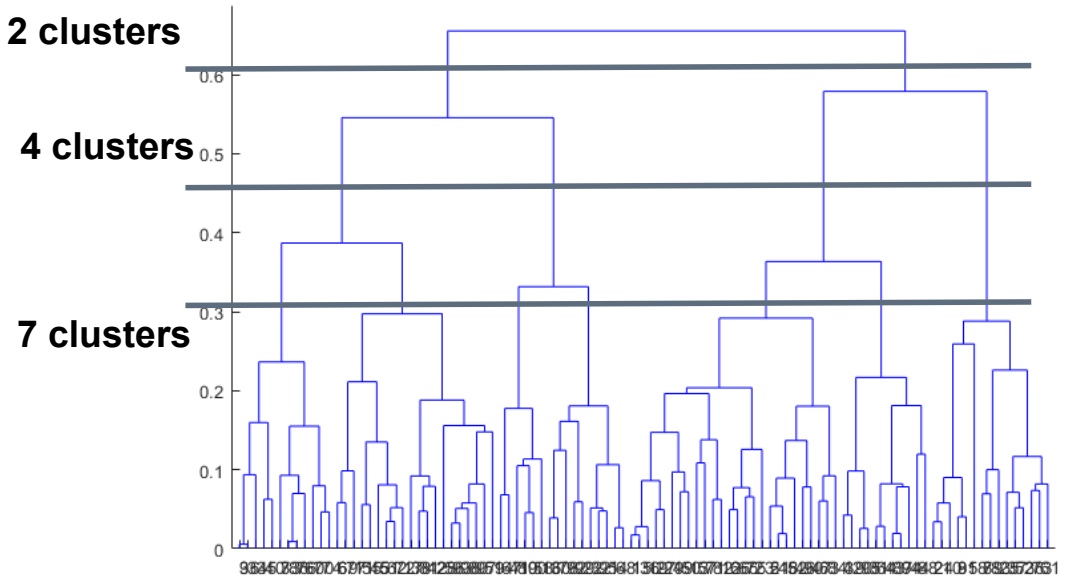
Hierarchical clustering

Assign close points to the same cluster.

- **Agglomerative**: each observation starts in its own cluster and clusters with the smallest intergroup dissimilarity are combined.
- **Divisive**: all points start in one cluster and clusters are split into two groups with the largest between-group dissimilarity.

It requires to define a **metric** as the distance to use between two points and a **linkage criterion** as the measure of dissimilarity between two sets of points.

The output is not a clustering partition, but a dendrogram. We need to manually choose either the number of clusters or the maximum distance.

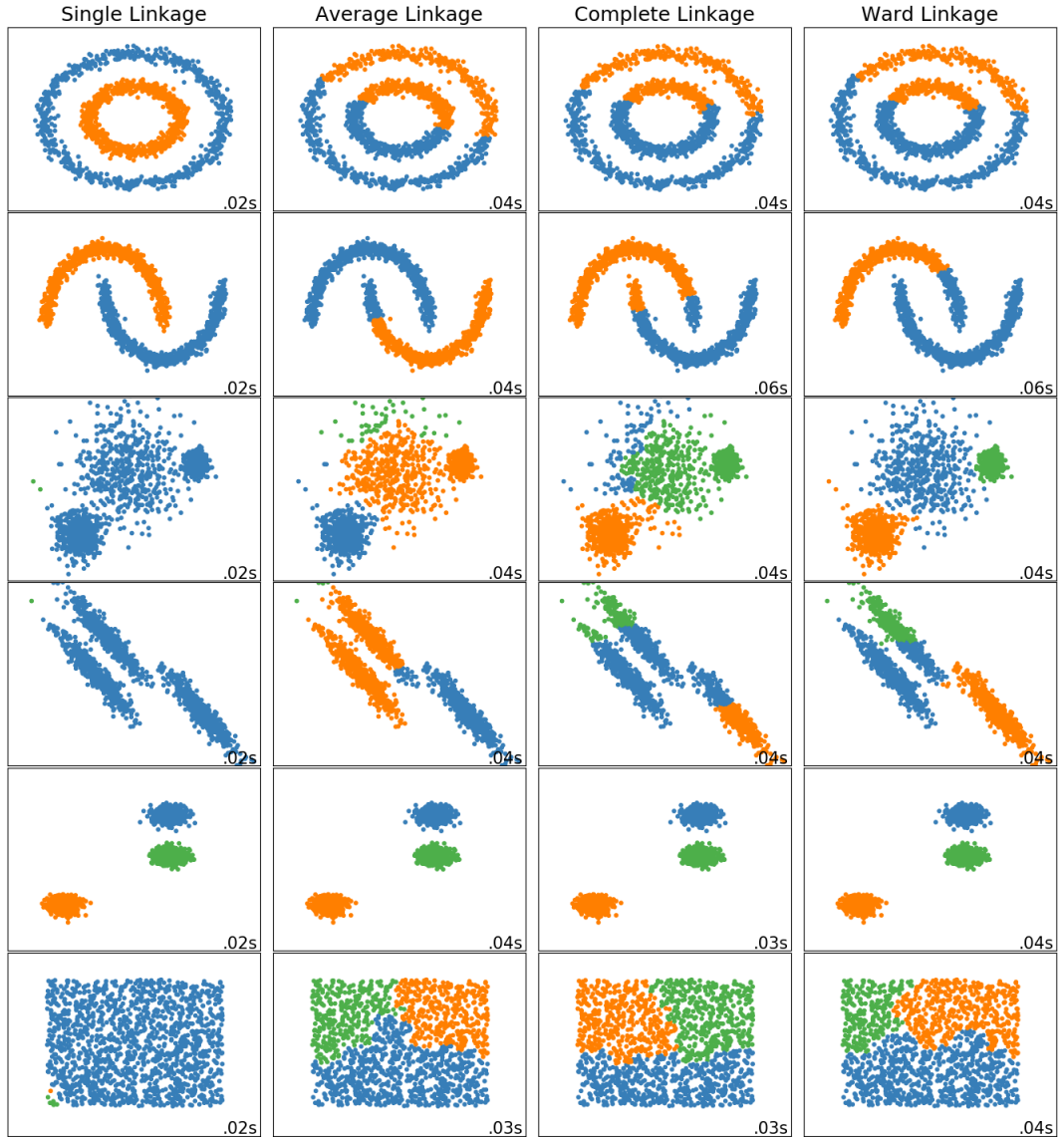


linkage : *{"ward", "complete", "average", "single"}, optional (default="ward")*

Which linkage criterion to use. The linkage criterion determines which distance to use between sets of observation. The algorithm will merge the pairs of cluster that minimize this criterion.

- ward minimizes the variance of the clusters being merged.
- average uses the average of the distances of each observation of the two sets.
- complete or maximum linkage uses the maximum distances between all observations of the two sets.
- single uses the minimum of the distances between all observations of the two sets.

Hierarchical clustering



effect of Linkage criterion

More information about linkage and distance
https://scikit-learn.org/stable/auto_examples/cluster/plot_agglomerative_clustering.html

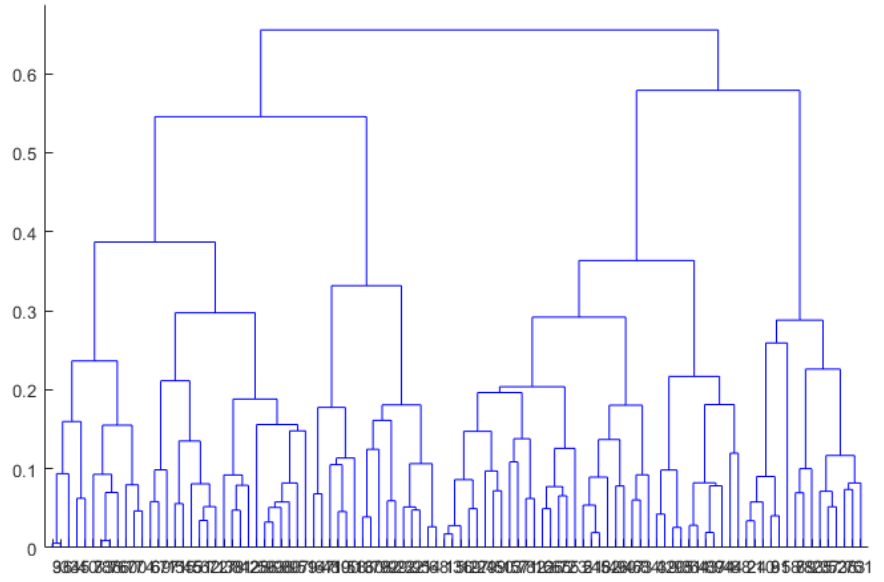
Hierarchical clustering

Pros:

- ✓We don’t need to specify number of clusters.
- ✓It does not make any assumption in the data shape so it’s suitable for any shape.

Cons:

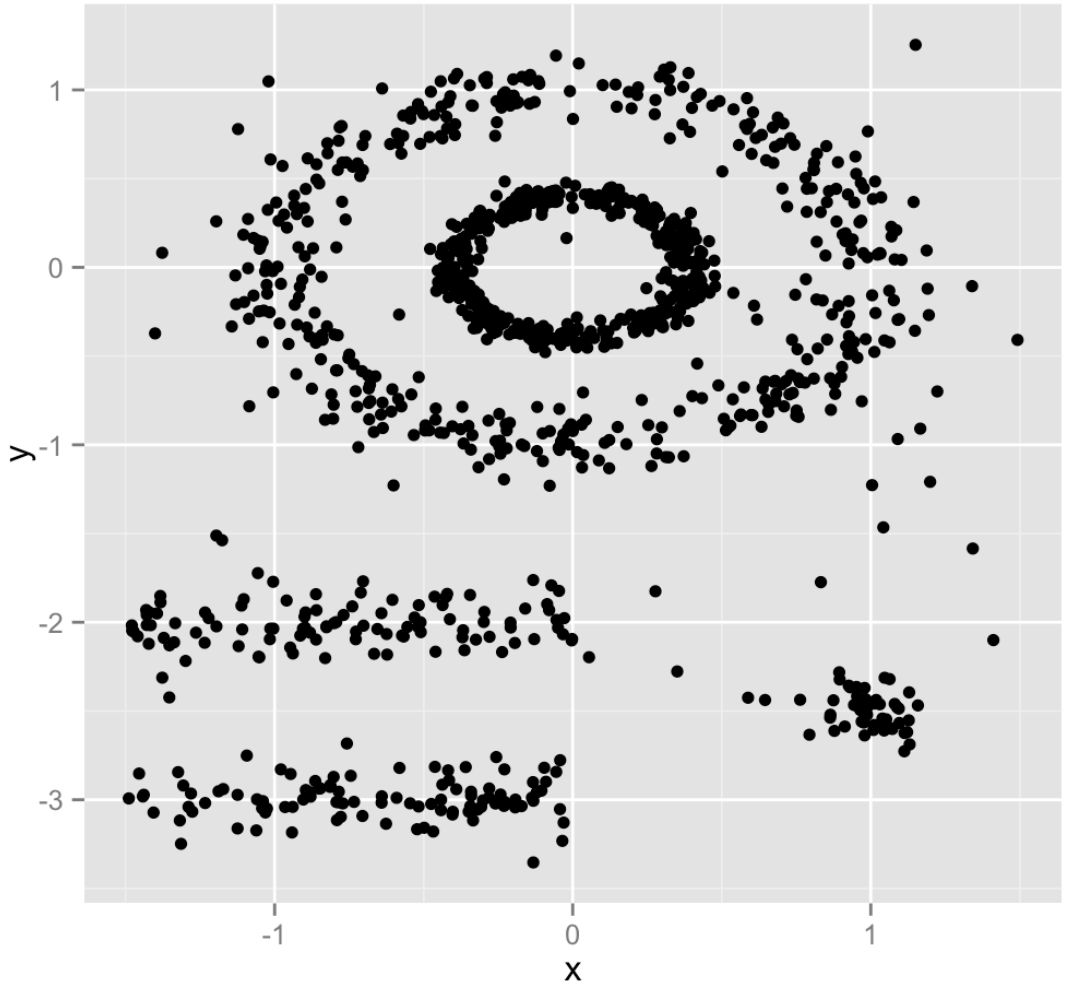
- x We need to specify a threshold distance
- x Very sensitive to distance and linkage criterion



Unsupervised learning

- Unsupervised Learning vs Supervised Learning
- **Clustering**
  - K-means
  - Hierarchical clustering
  - **Density-based clustering: DBScan and OPTICS.**
- Dimmensionality Reduction
  - PCA
  - Matrix Factorization
    - SVD
    - NMF
  - Autoencoders
- Additional Material

DBScan



- Non-convex clusters
- Outliers/noise
- Different density
- Different size

DBScan

Density-**B**ased **S**patial **C**lustering and **A**pplication with **N**oise (Ester et al. 1996)

Key idea: ***density***. Points that are density-reachable from another point belong to the same cluster.

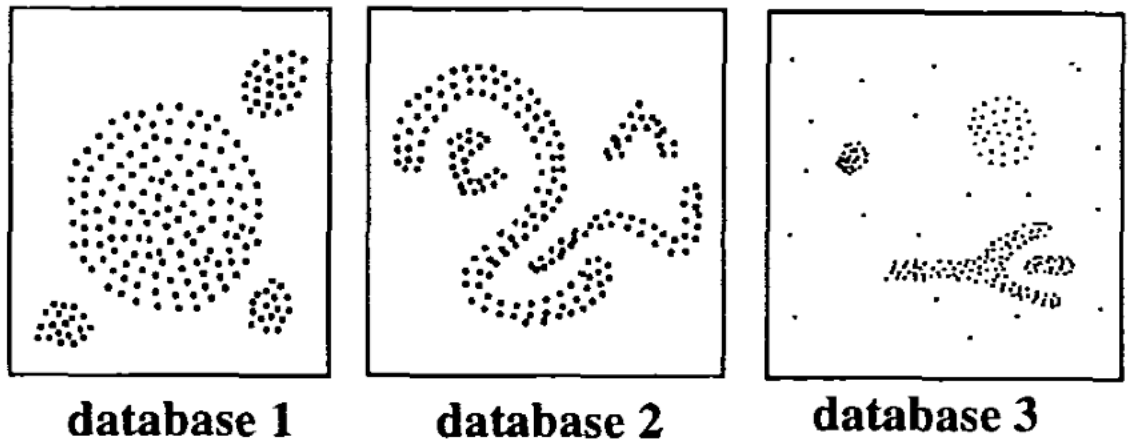
- `eps`: distance between points
- `MinPts`: minimum number of neighbors at distance eps for a point to be considered *core point*

Overview of the algorithm:

- Assign every core point to a new cluster
- Merge two points if you can reach one from another only through core points.

Pros: #clusters (k) is automatically found.

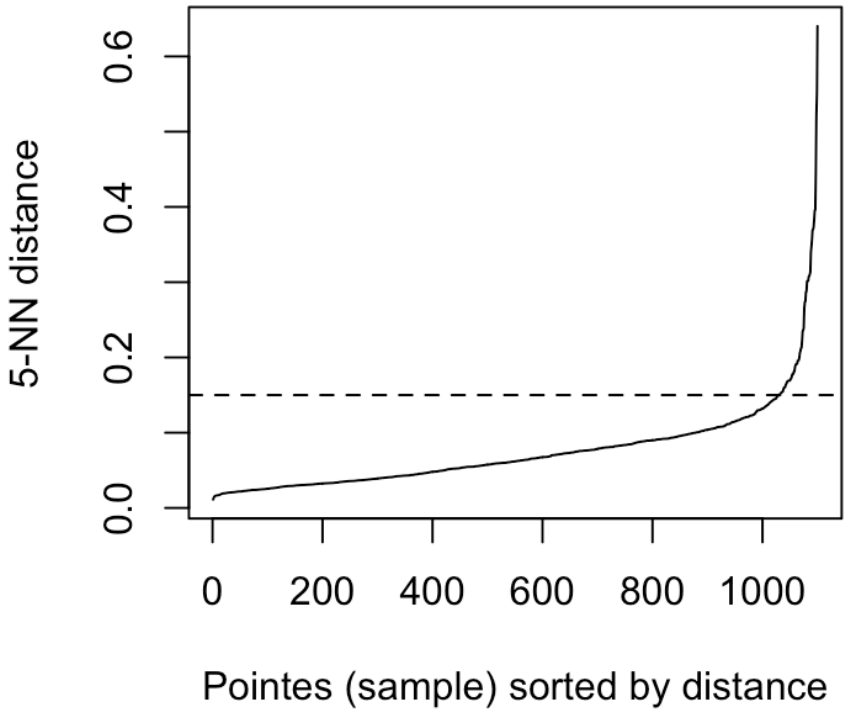
Cons: we need to specify `eps` and `MinPts`.



DBScan

How to choose `eps` and `MinPts`.

- ✓Set `MinPts` manually (`MinPts` ~ 5)
- ✓Select `eps` based on the *k-distances plot*:
  - Compute the average distance of each point to its k-nearest neighbors (k = MinPts)
  - Sort the distances ascending and plot them
  - Select `eps` that corresponds to the knee (~elbow method)



DBScan - Code

R

```
myiris <- as.matrix(iris[,1:4])
dbscan::kNNdistplot(myiris, k=5)

cl <- dbscan::dbscan(myiris, eps=0.5, minPts = 5)
cl$cluster
```

Python

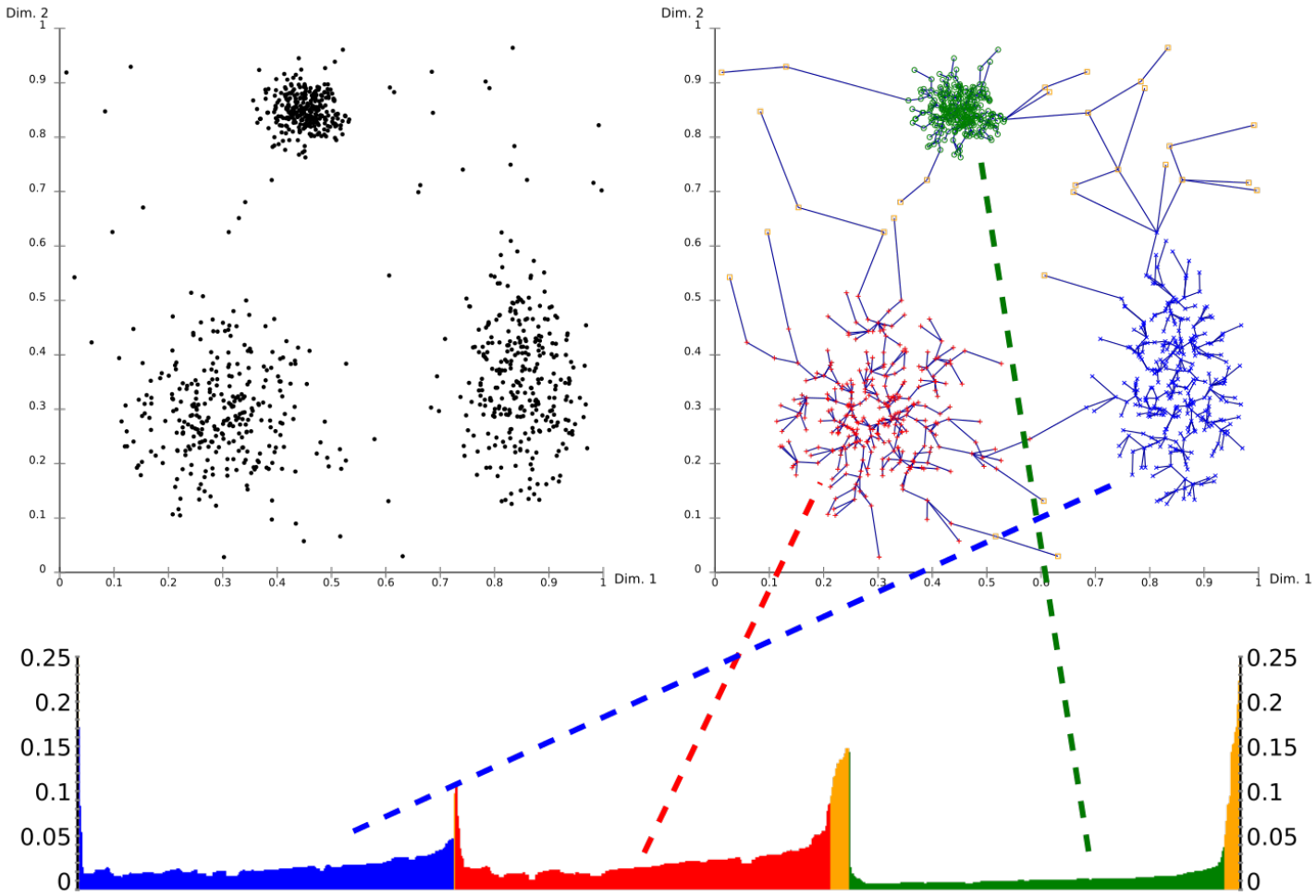
```
from sklearn.cluster import DBSCAN
dbs = DBSCAN(eps=0.4, min_samples=5)
dbs = dbs.fit(data)

dbs.labels_
```

OPTICS

Ordering **p**oints **t**o **i**dentify **c**lustering **s**tructure

- Density-based clustering
- Based on DBSCAN
- Solves dbscan problem with diverse density clusters:
  - by considering a eps by each core-point.
  - Assigning points to the same cluster if they are under a set density.



Let’s code!

01_Clustering_Before_Class.ipynb

DIMMENSIONALITY REDUCTION

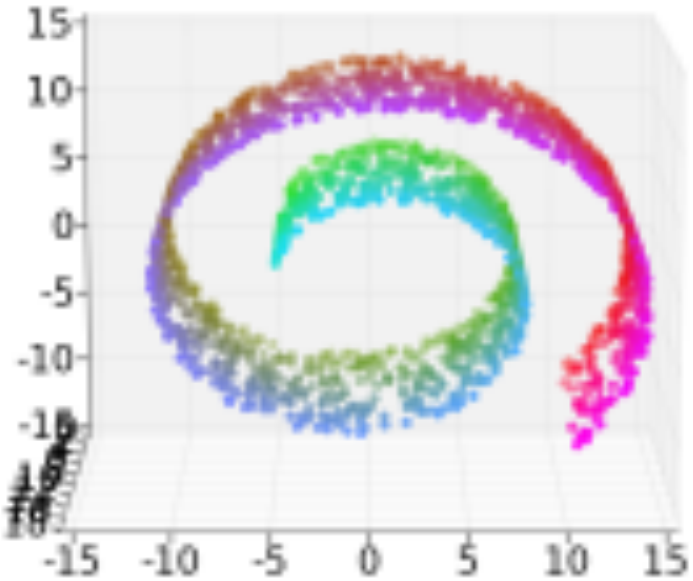
Synthesizing data

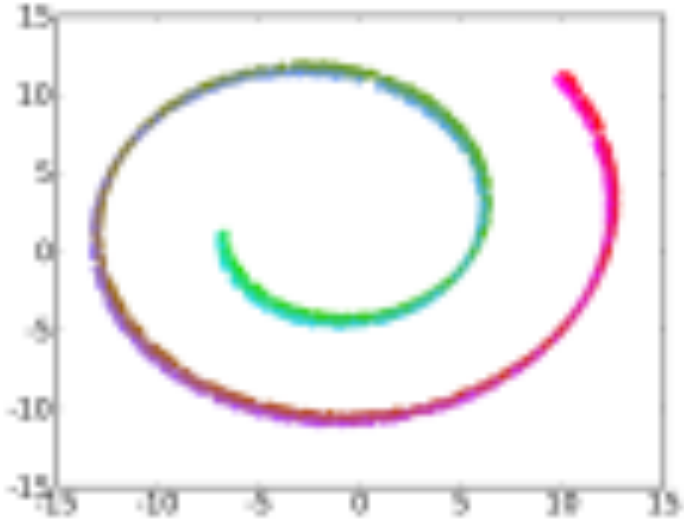
Dimensionality reduction

Generate a *projected* dataset where a few components (less than the original dataset) capture as much information as possible.

Why? Curse of dimensionality

- **PCA**: linear transformation based on eigenvalues
- **Matrix Factorization:**
  - **SVD**: linear transformation based on singular values
  - **Low-rank matrix factorization / NMF**: matrix multiplication
- **Autoencoder**: reconstruct original data after a dimensionality reduction





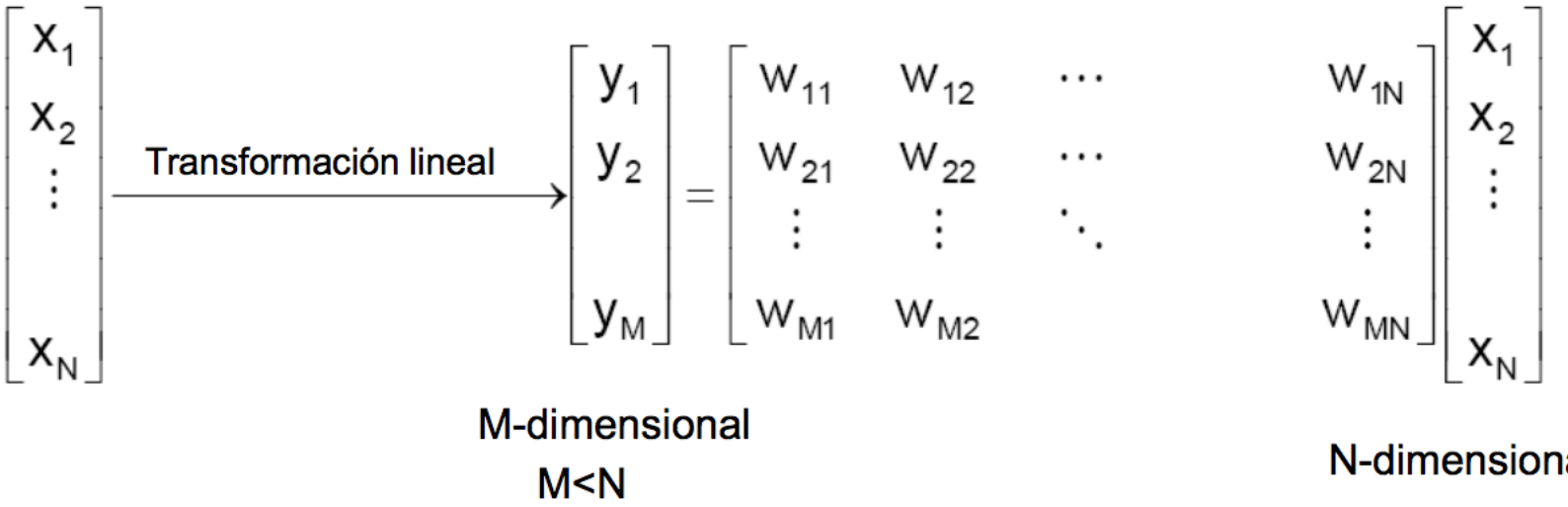
Unsupervised learning

- Unsupervised Learning vs Supervised Learning
- Clustering
  - K-means
  - Hierarchical clustering
  - Density-based clustering: DBScan and OPTICS.
- **Dimmensionality Reduction**
  - **PCA**
  - Matrix Factorization
    - SVD
    - NMF
  - Autoencoders
- Additional Material

PCA

- Linear transformation of the original data

$$y = W^T X, \qquad X_{N\times 1} \quad W_{N\times M} \quad y_{M\times 1}$$



How do we choose W?
How do we want y to be?

PCA – How do we choose **W**?

- Linear transformation of the original data $y = W^T X,$

$X_{N\times 1}$

$W_{N\times M}$

$y_{M\times 1}$

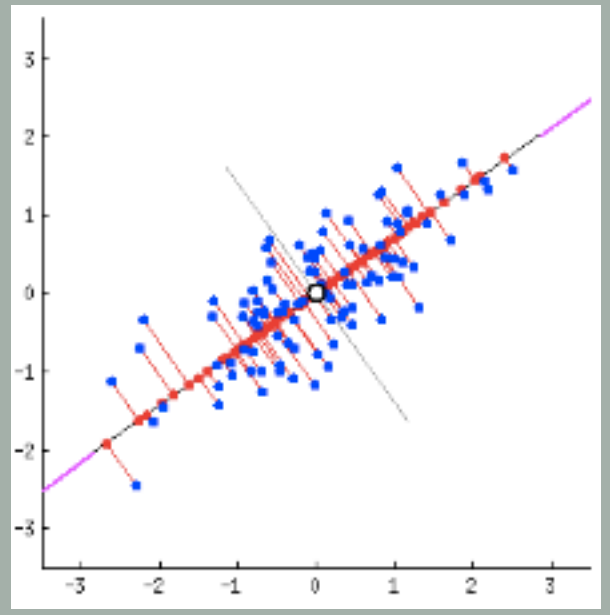
$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ w_{M1} & w_{12} & \cdots & w_{MN} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{12} & \cdots & w_{NN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

PCA

Goal:

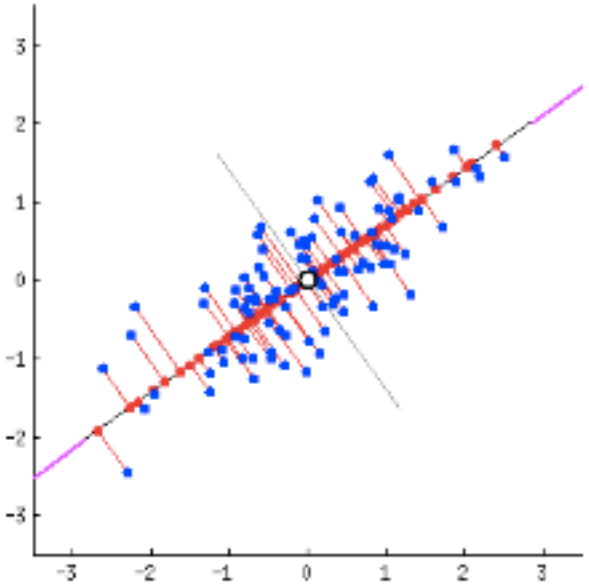
Find a collection of unit vectors $\{w_i\}$, such that:

1. The variance of the projected dataset onto w_i is maximized and
2. w_i is orthogonal w_j $\forall i \neq j$



How do we choose W?
How do we want y to be?

PCA – Procedure

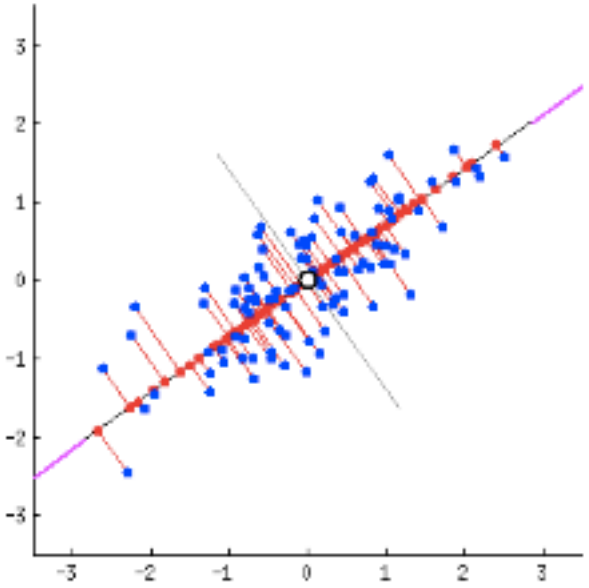


Data = X [dxN] d samples of dimension N

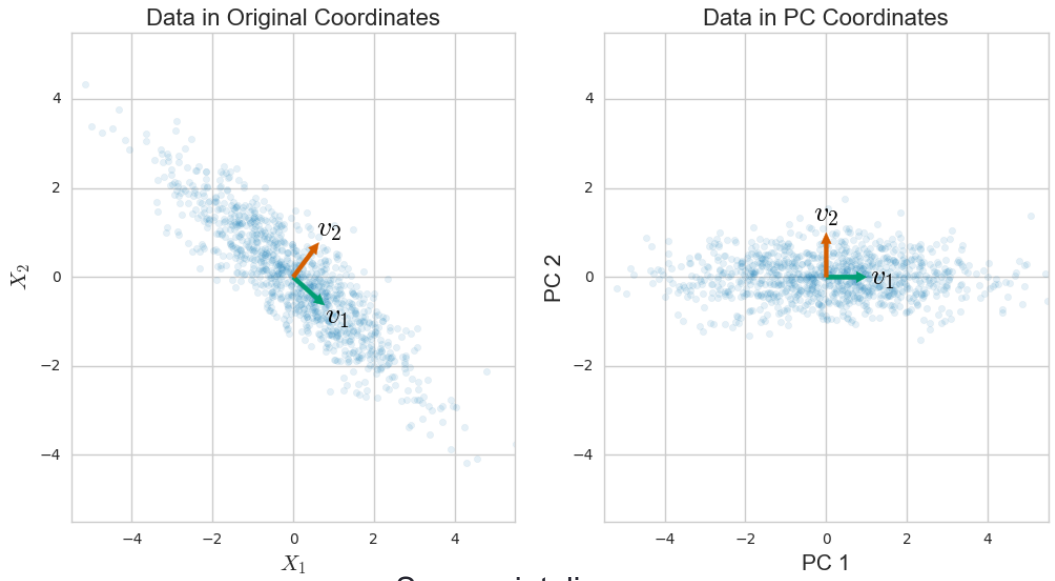
1. Compute **Covariance matrix** of the data → NxN matrix
2. Compute **eigenvectors and eigenvalues** of the covariance matrix
    1. Make eigenvectors size 1 (by dividing by the norm)
    2. Sort them in decreasing order of the eigenvalues
3. Select top M eigenvalues such that more than x% of total variance is explained (total variance = sum of eigenvalues)
4. Select corresponding M eigenvectors and construct matrix W whose columns are the eigenvectors
5. *Rotate* data to the new space with XxW [(dxN) x (NxM) = (dxM)]

Note: here we are using programming languages dataframes notation, where rows of data represent samples and columns of represent features. Usually in mathematics the notation is the opposite and each example is a column vector. That's why you will see W^tX.

PCA – Intuition



- We want data to be projected such as we keep the directions of maximum variance.
- Also, we want to remove correlated dimensions (i.e. covariance among the dimensions should be zero) → linearly independent
- Think of an ellypse. PCA ~ fitting a n-dimensional ellipsoid to the data.
- It means that the covariance matrix shoud have:
  - Large numbers as the main diagonal elements (variance)
  - Zero values outside the diagonal
  - i.e. a *Diagonal matrix*
- So we want to transform the original data points into points such that their covariance matrix is a diagonal matrix. **Diagonalization**.
  - In the ellypse simil, get rid of small axis direction.



Source: intoli.com

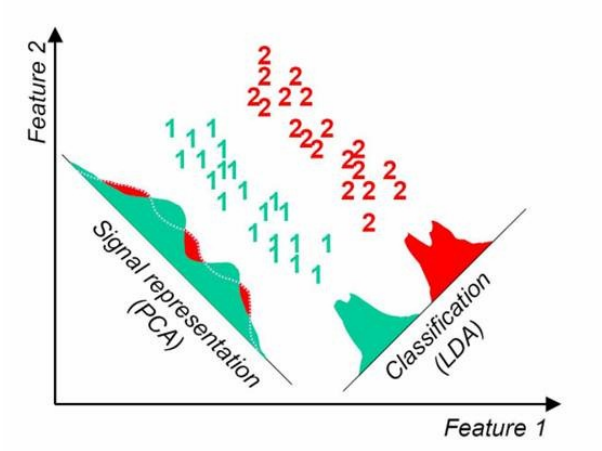
PCA – Mathematical derivation

A bit of theory…

- $C_x = X^T X$ is symmetric by construction
- → C_x is diagonalizable with orthogonal matrix P[1]
- → exists C_y diagonal and P such that $C_y = P^{-1} C_x P$
  - Where columns of P are the eigenvectors of C_x
  - As P is orthogonal $P^{-1} = P^T$
- → $C_y = P^T X^T X P$
- We call Y=XP → $C_y = Y^T Y$
- → C_y is the covariance of a matrix Y, the projection of X transformed by the eigenvector matrix.
- → It happens to be diagonal → Y has independent component and if P is sorted in descending order of eigenvalues then first components are those of maximum variation.

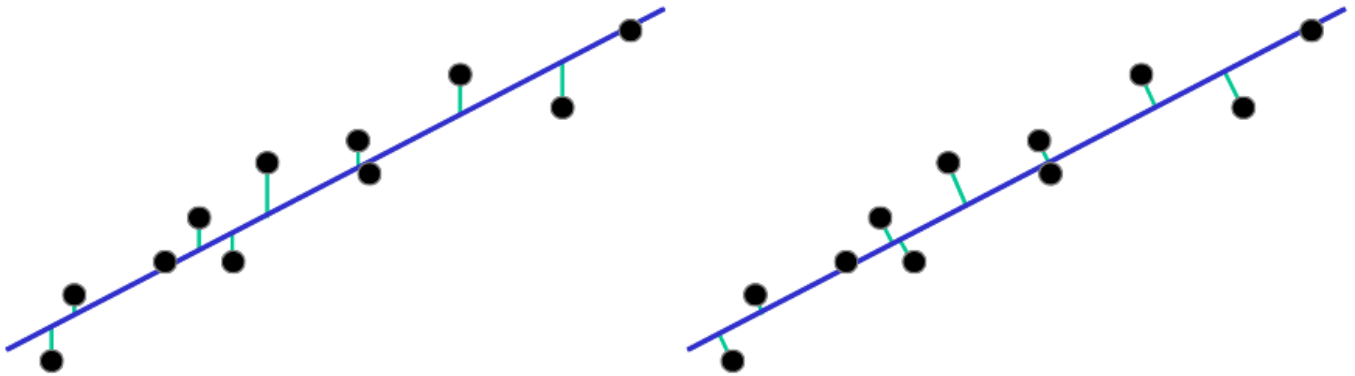
[1]proof: http://control.ucsd.edu/mauricio/courses/mae280a/lecture11.pdf

PCA considerations



- PCA is very sensitive to data scale. It is very important to put original data in the same scale (mean zero and standard deviation = 1).
- PCA only rotates the coordinates such that new directions are those of maximum variance → no guarantee that this information is good for classification.
- New variables (or *components*) does not have any business meaning.
- We can use correlation matrix (instead of covariance) and it does not require normalized data.

PCA and Linear Regression



PCA - Code

R

```
# stats
data.pca <- prcomp(cov(data), center = TRUE, scale. = TRUE, retx=TRUE)
data.pca$rotation             #eigenvectors
data.pca$x[, 1:NCOMPONENTS]   #rotated data
summary(data.pca)

# caret
library(caret)
model.pca.caret <- preProcess(data, method = c("center", "scale", "pca"))
data.pca.caret <- predict(model.pca.caret, data)
data.pca.caret[, 1:NCOMPONENTS]
```

Python

```
from sklearn.decomposition import PCA
pca_model = PCA(n_components = 5)
data_pca_sklearn = pca_model.fit_transform(data_norm)

pca_model.explained_variance_
pca_model.explained_variance_ratio_
np.cumsum(pca_model.explained_variance_ratio_)
```

Let’s code!

02_Dimmensionality_Reduction_Before_Class.ipynb

Unsupervised learning

- Unsupervised Learning vs Supervised Learning
- Clustering
  - K-means
  - Hierarchical clustering
  - Density-based clustering: DBScan and OPTICS.
- **Dimmensionality Reduction**
  - PCA
  - **Matrix Factorization**
    - **SVD**
    - NMF
  - Autoencoders
- Additional Material

Matrix Factorization - SVD

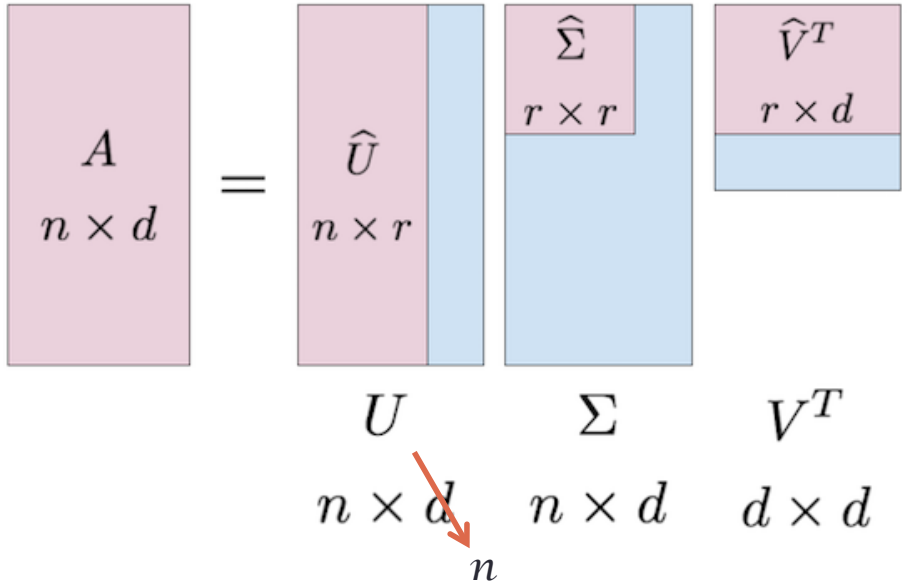
- Let A be a nxd matrix, then exists a factorization $A = UDV^T$ with:
  - D diagonal matrix with non-negative entries, of same dimension as A (nxd)
  - U, V orthogonal matrices of dimension nxn and dxd respectively
- If we truncate U and D to take the first r components (r<d)
  - D' diagonal matrix of dimension rxr
  - U' dimension nxr

we get an approximation of A' ~ U'D'V'.

- We take U'D' (dimension nxd) such that

||A-A'|| is small.

- Elements of D are called **singular values**.

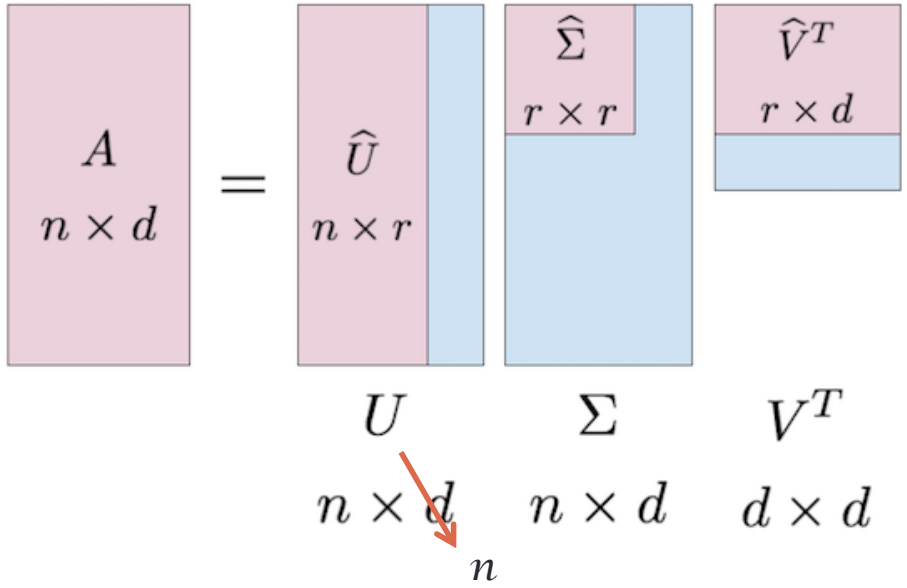


PCA and SVD

- PCA is a particular case of SVD where

A = correlation matrix ($A=X^TX$)

  - There always exists SVD factorization
  - We need square and semi-positive defined matrix for the PCA decomposition to exist.



- Columns of U are the eigenvectors of XX^T,
- Columns of V are the eigenvectors of X^TX
- Square root of singular values of Sigma are the eigenvalues of XX^T
- **Home exercise:** prove the previous properties

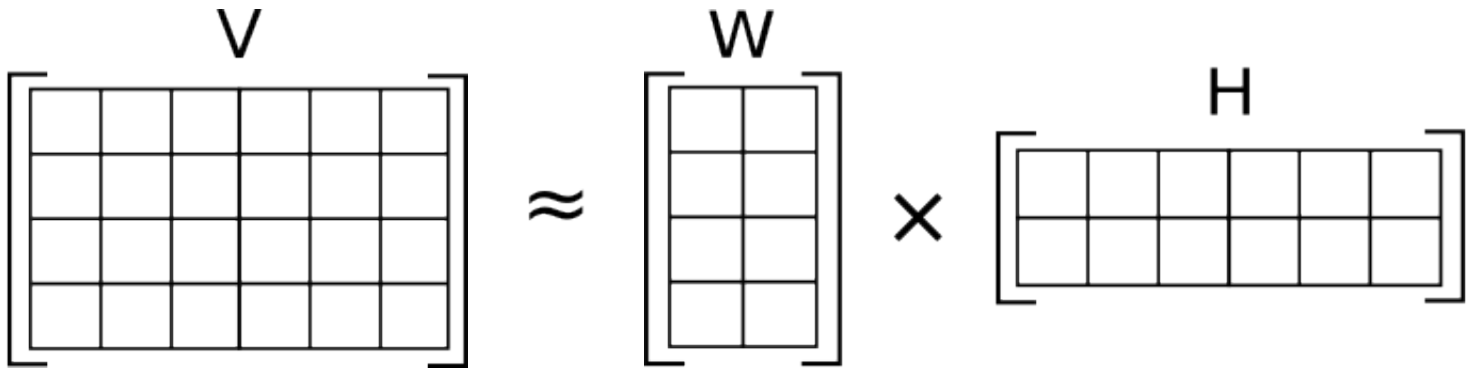
Unsupervised learning

- Unsupervised Learning vs Supervised Learning
- Clustering
  - K-means
  - Hierarchical clustering
  - Density-based clustering: DBScan and OPTICS.
- **Dimmensionality Reduction**
  - PCA
  - **Matrix Factorization**
    - SVD
    - **NMF**
  - Autoencoders
- Additional Material

Matrix Factorization - NMF

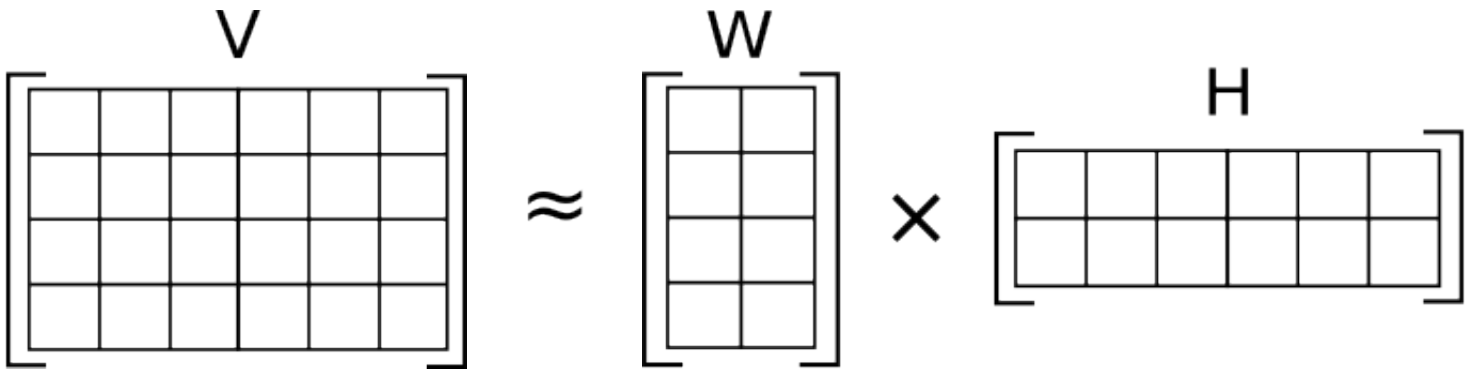
- Let V be a non-negative matrix of dimension n x m,
- NMF algorithm decomposes the matrix into a low rank, sparse and non-negative factors such that the original data can be approximated as

  V ~ WH, with
  - W and H have non-negative elements
  - W has dimension nxk, H has dimension kxm, k<<min(n,m)



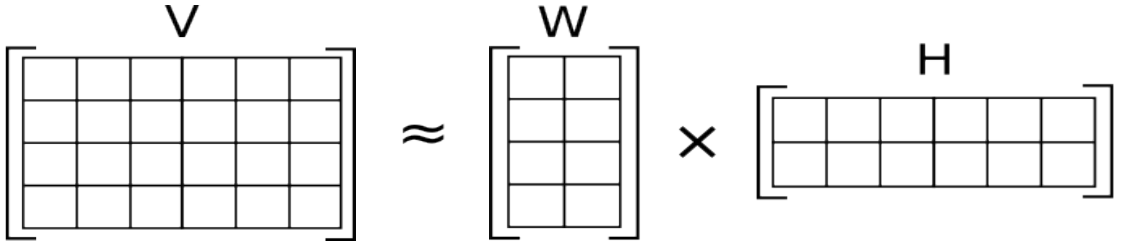
Matrix Factorization - NMF

- V [n x m], W [n x k], H [k x m]



- Think of k as “features” that forms each element in V
- Examples:
  - V = user-item movie matrix (recommender systems)
  - V = document-word frequency matrix (text classification)
- There are algorithms that compute this factorization.

Matrix Factorization - NMF



V [n x m], W [n x k], H [k x m]

Example: user-item movie matrix

- V: rows = user, columns = items
- Each row of H can be seen as how the feature is described based on the movies that belong to them.
  - Ex: if row 1 represents feature ‘romantic’, it will have high values on movies such as Titanic, Notthing Hill or so on.
- Rows of W can be seen as *how much of each feature each user has.*

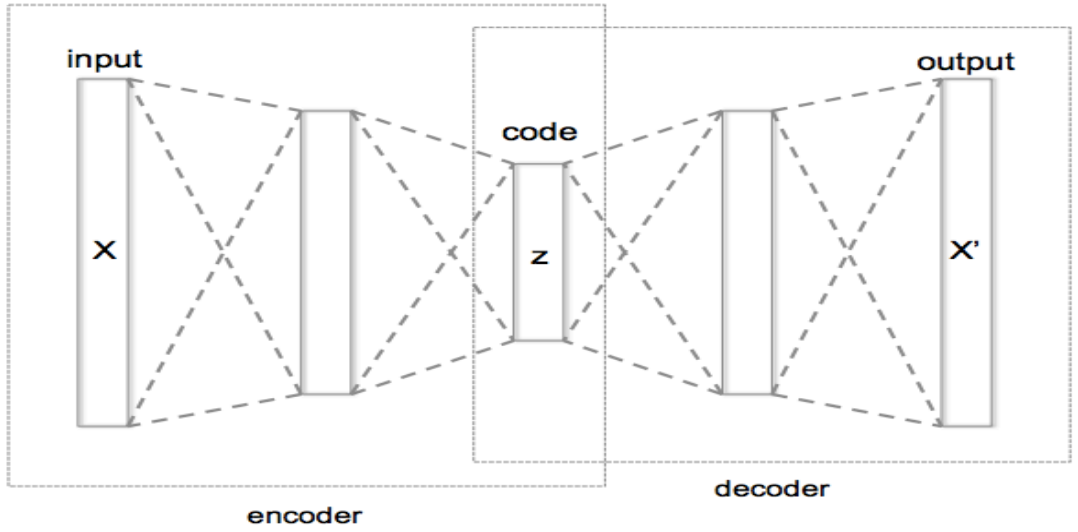
Example: document-word frequency matrix

- V: rows = documents, columns = words
- k can be seen as topics
- Rows of H: *how the topic is described through words*
- Rows of W: how much of topic k has document i

Unsupervised learning

- Unsupervised Learning vs Supervised Learning
- Clustering
  - K-means
  - Hierarchical clustering
  - Density-based clustering: DBScan and OPTICS.
- **Dimmensionality Reduction**
  - PCA
  - Matrix Factorization
    - SVD
    - NMF
  - **Autoencoders**
- Additional Material

Autoencoder



- Reconstruct original data after a compression.
- z is called *code*, *embedding* or *latent representation*.
- Used in many state-of-the-art results based on DL.

- Denoising Autoencoder [Pascal et al. 2010]: use as input a noised version of original data.
- Variational Autoencoder (VAE) [Kingma and Welling, 2014]: change in the objective function (minimizes the KL divergence of the encoded data from input and output). It learns a probability distribution to model input data, so we can sample from the distribution and generate new input data samples.

ADDITIONAL MATERIAL

Unsupervised learning

- Unsupervised Learning vs Supervised Learning
- Clustering
  - K-means
  - Hierarchical clustering
  - Density-based clustering: DBScan and OPTICS.
- Dimmensionality Reduction
  - PCA
  - Matrix Factorization
    - SVD
    - NMF
  - Autoencoders
- **Additional Material**

Association Rules (Market basket analysis)

Algorithm for finding *frequent* rules in data.

$$X \Rightarrow Y$$

- **Support:** ratio of appearance of X in a set of transactions.
- **Confidence:** ratio of X and Y happen wrt appearance of X (how often the rule is true) conf(X=>Y) = supp(X U Y) / supp(X)
- **Lift**: ratio of X and Y happen wrt appearance of X and appearance of Y=

  lift(X=>Y) = supp(X U Y)/[supp(X)supp(Y)]

Building a rule: “selecting frequent itemsets (above a support) with a minimum confidence and lift”

Transaction 1	🍎 🍺 ⚪ 🍗
Transaction 2	🍎 🍺 ⚪
Transaction 3	🍎 🍺
Transaction 4	🍎 🍐
Transaction 5	🍼 🍺 ⚪ 🍗
Transaction 6	🍼 🍺 ⚪
Transaction 7	🍼 🍺
Transaction 8	🍼 🍐

$$\text{Support}\{🍎\} = \frac{4}{8}$$

$$\text{Confidence}\{🍎 \rightarrow 🍺\} = \frac{\text{Support}\{🍎, 🍺\}}{\text{Support}\{🍎\}}$$

$$\text{Lift}\{🍎 \rightarrow 🍺\} = \frac{\text{Support}\{🍎, 🍺\}}{\text{Support}\{🍎\} \times \text{Support}\{🍺\}}$$

Images: KDnuggets

(not so) Advanced topics: **Expectation Maximization (EM)**

Algorithm for estimating latent (unobserved) parameters of a model based on unlabelled data.

EM_Algorithm[1]:

1. Start with a naïve or random initialization of parameters.
2. Expectation:
   - Probability of data, assuming current estimated parameters.
3. Maximization
   - Computes the parameters that maximize that probability (MLE).
4. Repeat 2-3 until convergence.

kMeans is a particular case of EM:

- Expectation: assign each point to its cluster centroid.
- Maximization: recompute centroids from the current node assignation.

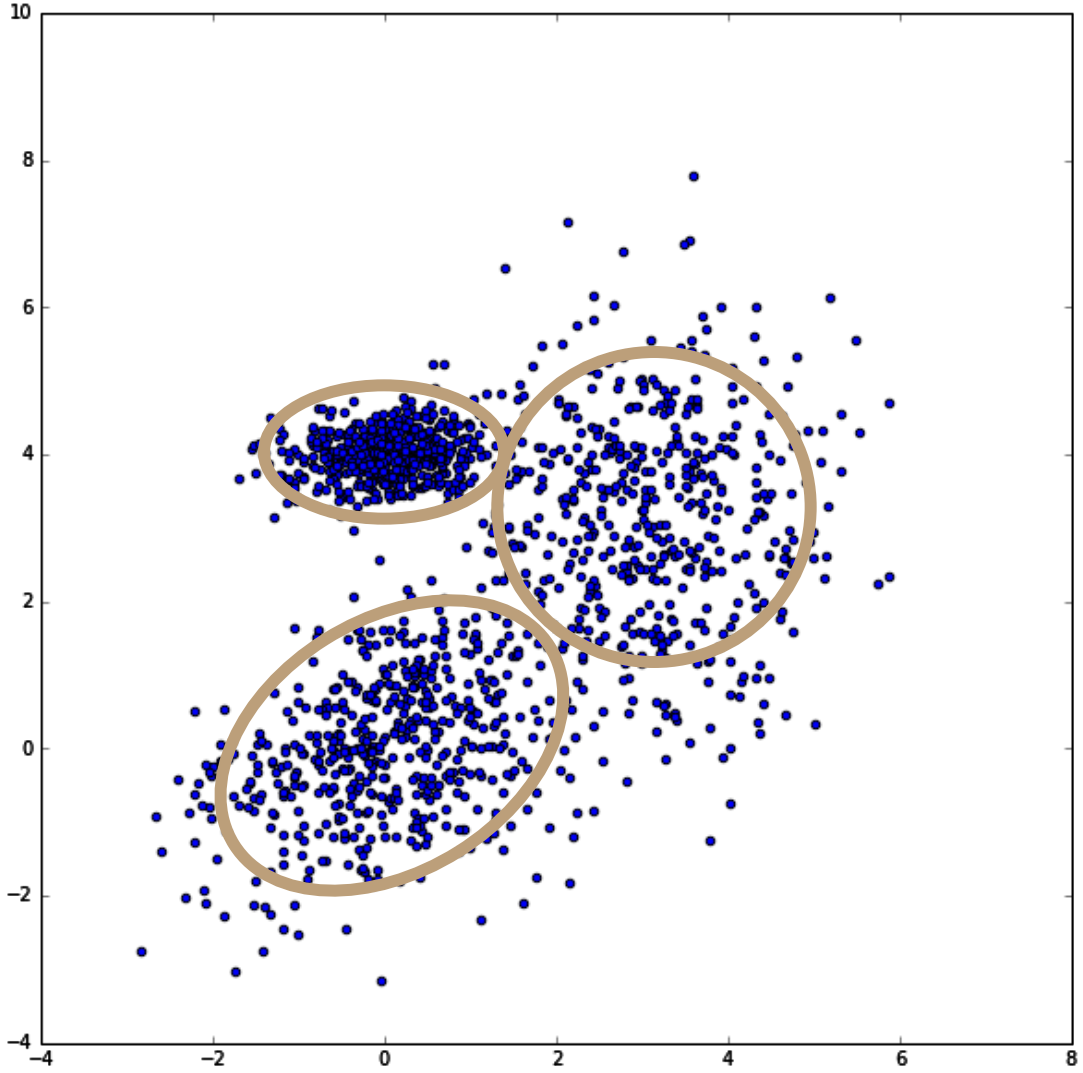
EM finds a local maximum; but may not find global optima.

[1]Dempster, A.P.; Laird, N.M.; Rubin, D.B. (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm". *Journal of the Royal Statistical Society, Series B*. **39** (1): 1–38. JSTOR 2984875. MR 0501537.

(not so) Advanced topics: **Gaussian Mixture Models (GMM)**

- Assume we have n data points $\{x_1, x_2, \ldots, x_n\}$ that comes from K multivariate Gaussian Distribution.
- We don't know which k each data point comes from → latent or unknown variables
- $z = \{z_1, x_2, \ldots, z_n\}$ where $z_i=k$, $k=1,\ldots,K$
- We also don't know the parameters of each gaussian distribution (μ_k, Σ_k)

- EM to find parameters
- BIC (or AIC) criteria to select number of components



WHAT DID YOU LEARN IN THIS LESSON?

Exercise

Final remarks

- Since it is unsupervised, there is no objective way of measuring performance.
  - How do we now if our clustering is ok?
- Pay attention to the scale of the data!!

Clustering:

- It is usually very useful in the first stages of a project, to explore and understand data.
- Kmeans is usually the most used algorithm because of its simplicity.
- It is usually not very useful since it does not let us make a decision, but as a supporting information.
  - Except for marketing, where a different marketing strategy can be made on each cluster → 'personna'
- I would recommend test dbscan when we actually want to perform a clustering as the solution.

Dimensionality reduction:

- It is usually very useful when we have high dimensionality or correlated data.
- Saves storage space and (more important!) computational time.
- It is the basis of most of state-of-the-art algorithms in Deep Learning (embeddings) and Recommender system
- Important: save PCA rotation computed on training set.

Bibliography and resources

Bibliography and resources:

- [Unsupervised Learning] Hastie, Trevor, Tibshirani, Robert and Friedman, Jerome. *The Elements of Statistical Learning*. New York, NY, USA: Springer New York Inc., 2001.
- [Wikipedia]: Clustering : https://en.wikipedia.org/wiki/Cluster_analysis
- K-means in R: https://stat.ethz.ch/R-manual/R-devel/library/stats/html/kmeans.html
- Dbscan in R: https://cran.r-project.org/web/packages/dbscan/dbscan.pdf
- K-means in scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
- Clustering in scikit-learn: https://scikit-learn.org/stable/modules/clustering.html
- Course “Algorithmic methods of Data Science”, Sapienza University
- Machine Learning Course [Coursera]: https://www.coursera.org/learn/machine-learning
- [DBSCAN] Ester, Martin, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.” In, 226–31. AAAI Press.
- [Autoencoders] Vincent, Pascal; Larochelle, Hugo; Lajoie, Isabelle; Bengio, Yoshua; Manzagol, Pierre-Antoine (2010). "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion". *The Journal of Machine Learning Research*. **11**: 3371–3408.
- [Autoencoders] Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
- [Expectation Maximization] Dempster, A.P.; Laird, N.M.; Rubin, D.B. (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm". *Journal of the Royal Statistical Society, Series B*. **39** (1): 1–38. JSTOR 2984875. MR 0501537.
- GMM in Scikit-learn:
  - https://scikit-learn.org/stable/modules/mixture.html
  - https://scikit-learn.org/stable/auto_examples/mixture/plot_gmm_covariances.html#sphx-glr-auto-examples-mixture-plot-gmm-covariances-py
- EM and GMM: http://www.blackarbs.com/blog/intro-to-expectation-maximization-k-means-gaussian-mixture-models-with-python-sklearn/3/20/2017

More information:

- Silhouette analysis for selecting the number of clusters: https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

More theory on PCA and SVD

- Understanding PCA
  - https://medium.com/@aptrishu/understanding-principle-component-analysis-e32be0253ef0
- SVD, PCA and LDA
  - https://www.lsv.uni-saarland.de/fileadmin/teaching/dsp/ss15/DSP_15_Lec6.pdf
- Explaining dimensionality reduction with SVD
  - https://stats.stackexchange.com/questions/342046/explaining-dimensionality-reduction-using-svd-without-reference-to-pca
- Relationship between SVD and PCA
  - https://stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-to-perform-pca?rq=1
- Dimensionality reduction with PCA and SVD
  - https://blog.paperspace.com/dimension-reduction-with-principal-component-analysis/
- PCA and SVD explanation in Jupyter
  - https://github.com/asdspal/dimRed/blob/master/PCA.ipynb
- Discussion on differences between PCA and OLS
  - https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues
- PCA: connection between maximinizing variance and minimizing error
  - https://stats.stackexchange.com/questions/32174/pca-objective-function-what-is-the-connection-between-maximizing-variance-and-m/136072#136072

UNSUPERVISED LEARNING

María Hernández Rubio - @maria_hr