Textual Data Analysis

J.-C. Chappelier

Laboratoire d'Intelligence Artificielle
Faculté I&C

Objectives of this lecture

Basics of textual data analysis:

- ➥ classification
- ➥ visualization: dimensionality reduction / projection

  (useful for a good understanding/presentation of classification/clustering results)

Is this course a Machine Learning Course?

CAVEAT/REMINDER

- NLP makes use of Machine Learning (as would Image Processing for instance)
- but good results require:
  - good preprocessing
  - good data (to learn from), relevant annotations
  - good understanding of the pros/cons, features, outputs, results, ...

☞ The goal of this course is to provide you with specific knowledge about NLP.

New:

☞ The goal of this lecture is to make some link between general ML and NLP.
This lecture is worth deepening with some real ML course.

Introduction: Data Analysis

WHAT does Data Analysis consist in?

"to represent in a live and intelligible manner the (statistical) informations, simplifying and summarizing them in diagrams"

[L. Lebart]

- ☞ classification (regrouping in the original space)
- ☞ "visualization": projection in a low-dimension space

$\updownarrow$ complementary

Classification/clustering consists in **regrouping** several objects in categories/clusters (i.e. subsets of objects)

Vizualisation: display in a intelligible way the internal structures of data

Contents

Supervized/unsupervized

The classification can be

- *supervized* (strict meaning of classification) :
  Classes are known *a priori*
  They are usually *meaningful* for the user

- *unsupervized* (called: *clustering*) :
  Clusters are based on the inner structures of the data (e.g. neighborhoods)
  Their meaning is really more dubious

Textual Data Analysis: relate documents(or words) so as to...
structure (supervized) / discover structure (unsupervized)

Classify what?

WHAT is to be classified?

Stating point: a **chart** (numbers) representing, *in a way or another*, a set of N objects (or "observations") $x^{(i)}$ characterized by m "features" $x_j^{(i)}$:

- continuous values (“importance” of a given feature for a given oject)
- contingency tables: co-occurrence counts (feature–feature)
- presence/absence of feature
- distance/(dis)similarity (symmetric square chart: object–object or feature–feature)

Two complementary points of view:

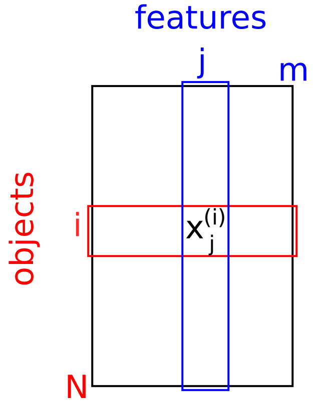
① N points in $\mathbb{R}^m$

② m points in $\mathbb{R}^N$

Not necessarily the same metrics:

objects similarities vs. features similarities

Classify what?



$x_j^{(i)}$= "importance" of feature j for object i

Textual Data Classification

- What is classified? (what objects?)
  - authors (1 object = several documents)
  - documents
  - paragraphs
  - "words"(/tokens) (vocabulary study, lexicometry)

- How to represent the objects? (what features?)
  - document indexing
  - choose the textual units that are meanigfull
  - choice of the metric/similarity

☞ preprocessing: "unsequentialize" text, suppress (meaningless) lexical variability

Frequently: lines = documents, columns = "words" (tokens, words, *n*-grams)
☞ the former two "visions" are complementary

Textual Data Classification: examples of applications

- Information Retrieval
- Open-Questions Survey (polls)
- emails classification/routing
- client survey (complaints analysis)
- Automated processing of ads
- ...

(Dis)Symilarity Matrix

Most of classification techniques use distance measures or (dis)similarities: matrix of the distances between each data points: $\frac{N(N-1)}{2}$ values (symmetric with null diagonal)

distance (“metric”):

① $d(x,y) \geq 0$ and $d(x,y) = 0 \Longleftrightarrow x = y$

② $d(x,y) = d(y,x)$

③ $d(x,y) \leq d(x,z) + d(z,y)$

dissimilarity: ① and ② only

Some of the usual metrics/similarities

- Euclidian:

$$d(x,y) = \sqrt{\sum_{j=1}^{m} (x_j - y_j)^2}$$

- generalized ($p \in [1...\infty[$):

$$d_p(x,y) = \left(\sum_{j=1}^{m} (x_j - y_j)^p \right)^{1/p}$$

- χ^2:

$$d(x,y) = \sum_{j=1}^{m} \lambda_j \left(\frac{x_j}{\sum x_{j'}} - \frac{y_j}{\sum y_{j'}} \right)^2$$

where $\lambda_j = \frac{\sum_i \sum_j u_{ij}}{\sum_i u_{ij}}$ depends on some reference data (u_i, $i = 1...N$)

Some of the usual metrics/similarities

- cosine (similarity) :

$$\mathscr{S}(x,y) = \frac{\sum_{j=1}^{m} x_j y_j}{\sqrt{\sum_j {x_j}^2}\sqrt{\sum_j {y_j}^2}} = \frac{x}{||x||} \cdot \frac{y}{||y||}$$

- for probability distributions :
  - KL-divergence:

$$D_{KL}(x,y) = \sum_{j=1}^{m} x_j \log\left(\frac{x_j}{y_j}\right)$$

  - Jensen-Shannon divergence:

$$JS(x,y) = \frac{1}{2}\left(D_{KL}(x, \frac{x+y}{2}) + D_{KL}(y, \frac{x+y}{2})\right)$$

  - Hellinger distance:

$$d(x,y) = d_{\text{Euclid}}(\sqrt{x}, \sqrt{y}) = \sqrt{\sum_{j=1}^{m} (\sqrt{x_j} - \sqrt{y_j})^2}$$

Computational Complexity

Various complexities (depends on the method), but typically:
$\frac{N(N-1)}{2}$ distances

m computations for one single distance
☞ complexity in $m \cdot N^2$

Costly: $m \simeq 10^3$, $N \simeq 10^4$ ☞ $\rightarrow 10^{11}$!!

Classification as a mathematical problem

- supervized:
  - function approximation $f(x_1, ..., x_m) = C_k$
  - distribution estimation: $P(C_k | x_1, ..., x_m)$ or $P(x_1, ..., x_m | C_k)$
    - parametric: multi-gaussian, maximum likelihood, Bayesian inference, discriminative analysis
    - non-parametric: kernels, K nearest neighbors, LVQ, neural nets (Deep Learning, SVM)
  - inference:
    if $x_i = ...$ and $x_j = ...$ (etc.) then $C = C_k$
    ☞ decision trees
- unsupervized (clustering):
  - (local) minimization of a global criterion over the data set

Many different classification methods

How to choose? ☞ Several criteria

Task specification:

- supervized
- unsupervized

- hierarchical
- non hierarchical

- overlapping
- non overlapping (partition)

Model choices:

- generative models ($P(X, Y)$)
- discriminative models ($P(Y|X)$)

- parametric
- non parametric (= *many* parameters)

- linear methods (Statistics)
- trees (GOFAI)
- neural networks

Classification methods: examples

- supervized
  - **Naive Bayes**
  - **K-nearest neighbors**
  - ID3 – C4.5 (decision tree)
  - Kernels, Support Vector Machines (SVM)
  - Gaussian Mixtures
  - Neural nets: Deep Learning, SVM, MLP, Learning Vector Quantization
  - ...
- unsupervized
  - **K-means**
  - **dendrograms**
  - minimum spanning tree
  - Neural nets: Kohonen’s Self Organizing Maps (SOM)
  - ...

☞ The question you should ask yourself:
What is the optimized **criterion**?

Bayesian approach

Probabilitic modeling: the classification is made according to $P(C_k|x)$:
an object $x^{(i)}$ is classified in category

$$\underset{C}{\operatorname{argmax}} P(C|x = x^{(i)})$$

Discriminative: model $P(C_k|x)$ directly;

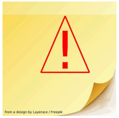
Generative: assume we know $P(C_k)$ and $P(x|C_k)$,
then using Bayes formula:

$$P(C|x = x^{(i)}) = \frac{P(x = x^{(i)}|C) \cdot P(C)}{P(x = x^{(i)})} = \frac{P(x^{(i)}|C) \cdot P(C)}{\sum_C \left[P(C) \cdot P(x^{(i)}|C)\right]}$$

$P(C)$: "*prior*" $P(C|x)$: "*posterior*" $P(x|C)$: "*likelihood*"

In practice, those distributions are hardly known.
All the difficulty consists in "learning" (estimating) them from samples making several hypotheses.

Naive Bayes



Supervised generative probabilistic (non overlaping) model:

Classification is made using the Bayes formula

$P(C)$ is estimated directly on a typical example

What is "naive" in this approach is the computation of $P(x|C)$

Hypothesis: feature independance:

$$P(x|C) = \prod_{j=1}^{m} p(x_j|C)$$

The $p(x_j|C)$ (*a priori* much fewer than the $P(x|C)$) are estimated on typical examples (learning corpus).

In the case of Textual Data: features = indexing terms (e.g. lemmas)

☞ This hypothesis is most certainly **wrong**
but **good enough** in practice

(multinomial) Logistic regression

Supervised *discriminative* probabilistic (non overlaping) model:
Directly model $P(C|x)$ as:

$$P(C|x) = \prod_{j=1}^{m} f(x_j, C) = \frac{\exp(\sum_{j=1}^{m} w_{C,j}\, x_j)}{\sum_{C'} \exp(\sum_{j'=1}^{m} w_{C',j'}\, x_{j'})}$$

where $w_{C,j}$ is a parameter, the “weight” of x_j for class C
(x_j being here some numerical representation of j-th indexing term: 0–1, frequency, log-normalized, ...).

The parameters $w_{C,j}$ can be learned using various approximation algorithms (e.g. iterative or batch; IGS, IRLS, L-BGFS, ...), for instance:

$$w_{C,j}{}^{(t+1)} = w_{C,j}{}^{(t)} + \alpha \left(\delta_{C,\widehat{C}_n} - P(C|x_n) \right) x_{nj}$$

with α a learning parameter (step strength/speed) and $\delta_{C,\widehat{C}_n}$ the Kronecker delta function between class C and expected class $\widehat{C}_n$ for sample input x_n.

K nearest neighbors – Parzen window

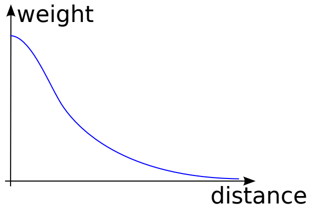
non hierachical non overlapping classification

K nearest neighbors:

very simple:
classify a new object according to the majority class in its K nearest neighbors (vote).
(no learning phase)

Parzen window:

same idea, but the votes are weighted according to the distance to the new object



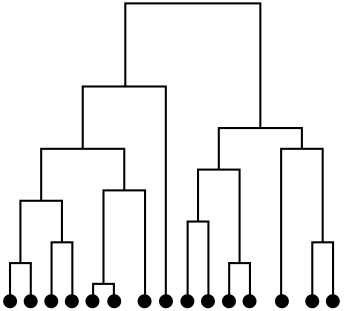
Dendrograms

It's a bottom-up hierachical clustering (= unsupervised)
Starts form a distance chart between the N objects

① Regroup in one cluster the two closest "elements" and consider the new cluster as a new element

② compute the distances between this new element and the others

③ loop in ① while there are more than one element



☞ representation in the form of a binary tree

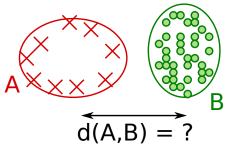
Complexity: $\mathcal{O}(N^2 \log N)$

Dendrograms: "linkage" scheme (1/2)

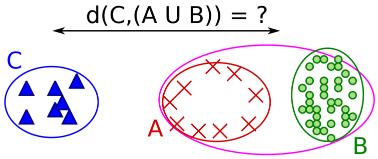
"*regroup the two closest elements*" ☞ closest?

Two questions:

1. How to define the distance between two clusters (two sets of elements)? (based on the distances between the elements)



2. How to (efficiently) compute distance between a former cluster and a (new) merge of two clusters? (based on the former distances between clusters)



Dendrograms: "linkage" scheme (2/2)

"*regroup the two closest elements*" ☞ closest?

Let A and B be two subclusters: what is their distance? (Lance-Williams algorithm)

method	definition $D(A,B) =$	merging $D(A \cup B, C) =$
single linkage:	$\min\limits_{x \in A, y \in B} d(x,y)$	$\min\Big(D(A,C), D(B,C)\Big)$
complete linkage:	$\max\limits_{x \in A, y \in B} d(x,y)$	$\max\Big(D(A,C), D(B,C)\Big)$
average linkage:	$\frac{1}{\lvert A\rvert \cdot \lvert B\rvert} \sum\limits_{x \in A, y \in B} d(x,y)$	$\frac{\lvert A\rvert \cdot D(A,C) + \lvert B\rvert \cdot D(B,C)}{\lvert A\rvert + \lvert B\rvert}$

K-means

non hierachical non overlapping clustering

① choose *a priori* the number of clusters : K

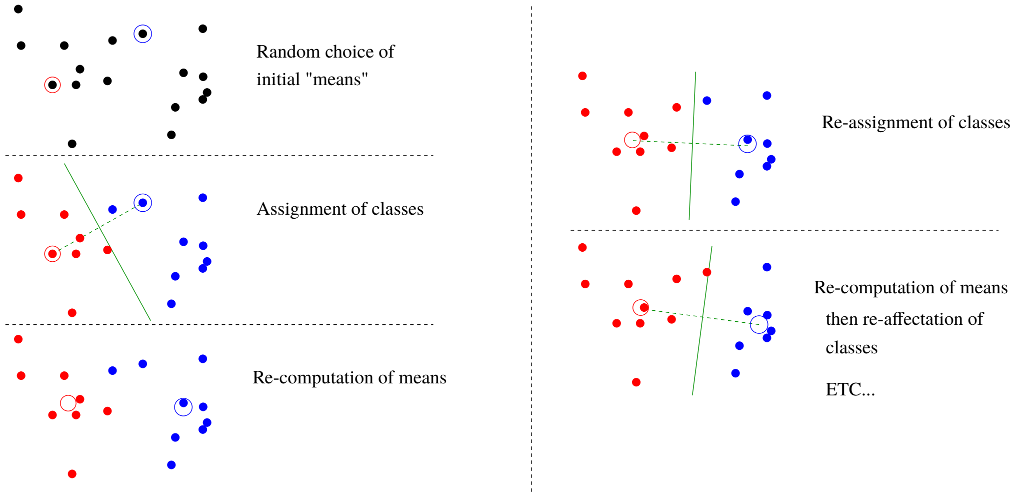
② randomly draw K objects as clusters' representatives ("clusters' centers")

③ partition the objects with respect to the K centers (closest)

④ recompute the K centers as the mean of each cluster

⑤ loop in ③ until convergence (or any other stoping criterion).

K-means (2) : example with $K = 2$


Random choice of initial "means"
Assignment of classes
Re-computation of means
Re-assignment of classes
Re-computation of means then re-affectation of classes
ETC...

K-means (3)

cluster representatives:
mean (centre of gravity): $R_k = \frac{1}{N_k} \sum_{x \in C_k} x$

☞ The algorithm is convergent because the **intra-class variance** can only **decrease**

$$v = \sum_{i=1}^{K} \sum_{x \in C_i} p(x) d(x, R_i)^2$$

($p(x)$: probability of the objects)

BUT it converges to a **local** minimum;
improvements:

- stable clusters
- Deterministic Annealing

Other methods similar to K-means:

- having several representatives
- compute representatives at each binding of an individual
- choose representatives among the objects

about Word Embedings & Deep Learning

Of course nowadays (since the $\simeq$5 years “deep learning buzz”),
many NLP classification tasks are done using so-called “*Word embedding*” and “*Deep learning*”
(sometimes with a bit of confusion and surprisingly oblivious of the large body of past work)

“*Word embedding*” are indeed numerical representation of the “words” usefull as “features” to represent documents

“*Deep learning*” refers to Neural Networks-based approaches to classification (although

- there is **NO need** of deep learning for good word-embeddings
- <u>not</u> all Neural Network models (NN) are deep learners

)

☞ much more about all this in two weeks (dedicated lecture)

Classification: evaluation

- classification (supervised): evaluation is "easy" $\rightarrow$ test corpus (some known samples kept for testing only)
- clustering (unsupervised): objective evaluation is more difficult: what are the criteria?

(supervised) Classification: **REMINDER** (see “*Evaluation*” lecture)

- Check IAA (if possible)
- Measure the misclassification error on the test corpus
  ☞ **!!** really **separated** from the learning set (and also from the validation set, if any)
  ☞ criteria: confusion matrix, error rate, ..
- Is the difference in the results **statistically significant**?

Clustering (unsupervised learning) evaluation

There is no absolute scheme with which to evaluate clustering, but a variety of ad-hoc measures from diverse areas/point-of-view.

For K non overlapping clusters (with objects having a probability p), standard measures include:

Intra-cluster variance (to be minimized):
$$v = \sum_{k=1}^{K} \sum_{x \in C_k} p(x)\, d(x, \overline{x_k})^2$$

Inter-cluster variance (to be maximized):
$$V = \sum_{k=1}^{K} \underbrace{\left(\sum_{x \in C_k} p(x) \right)}_{=p(C_k)} d(\overline{x_k}, \overline{x})^2$$

The best way is to *think* to how you want to assess the quality of a clustering w.r.t. your needs:

usually: high intra-cluster similarity and low inter-cluster similarity

(but what does "*similar*" mean?...)

One way also is to have manual evaluation of the clustering.

Note: if you already have a gold-standard with *classes*: why not use (supervised) classification in the first place??
(rather than using a supervised corpus to assess unsupervised methods...)

“Visualization”

Visualize: project/map data in 2D or 3D

More generally: techniques presented in this section are to lower the dimension of data

☞ go form N-D to n-D with $n < N$ or even $n \ll N$

☞ usually means: go from *sparse* to *dense* representation

visualization: projection in a low-dimension space

classification: regrouping in the original space

$\updownarrow$ complementary

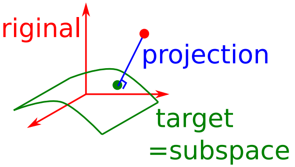
Which one to start with, depends on your data/application
(can even loop between the two)

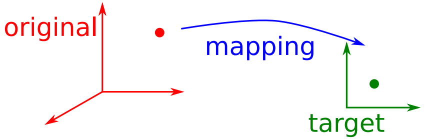
Several approaches

- Simple methods (but poorly informative): ordered list, "thermometer-like", histograms
- some of the classification methods can be used:
  - use/display the classes

  e.g. dendrograms with minimal spanning tree
- Linear and non-linear projections/mappings

  (projection: in the same space as original data
  mapping: in some other space)





Linear projections

Projections on selected sub-spaces of the original space

- Principal Components Analysis (PCA) [Pearson 1901]:
  object–feature chart (continuous values)
  feature similarity: correlations
  object similarity: distance on the feature space
- Correspondance Analysis:
  contingency tables
  row/column symmetry (features)
  χ^2 metric

☞ Singular value decomposition

Principal Components Analysis (PCA)

Input: a matrix $\overline{M}$ objects (rows) – features (columns) (of size $N \times m$ with $N > m$)

centered: $\overline{M}_{i\bullet} = x^{(i)} - \overline{x}$

Singular value decomposition (SVD) of $\overline{M}$:

eigenvalue decomposition of $\overline{M}\,\overline{M}^t$ (i.e. the covariance matrix (multiplied by $(N-1)$))

☞ $$\overline{M} = U \Lambda V^t$$

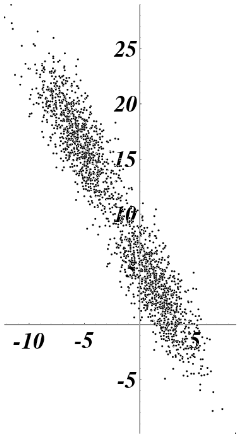
Λ diagonal, ordered: $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_m \geq 0$

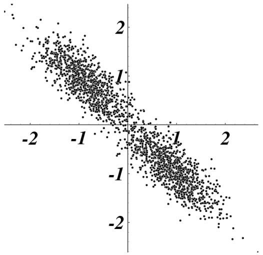
U of size $N \times m$ with orthogonal columns

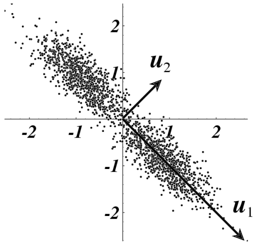
and V orthogonal, of size $m \times m$

PCA (2)

The "*principal components*" are the columns of $\overline{M}\,V$ (or of V)







PCA (3)

Projection in a low dimension space:

$$\widetilde{M} = U_q \Lambda_q V_q^t$$

with $q < m$ and X_q matrices reduced to only the q first singular values

$\widetilde{M}$ is the best approximation of rank q of $\overline{M}$.

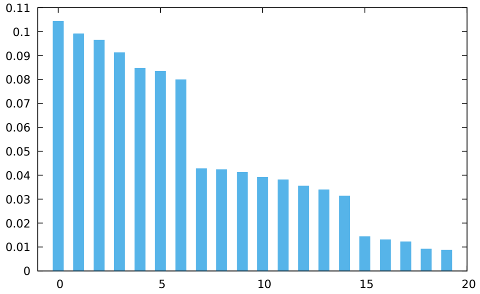
"*better approximation*" w.r.t several criteria:

L_2 norm, biggest variance (trace and determinant of the corvariance matrix), Frobenius norm, ...

This means that the subspace of the first q principal components is the best linear approximation of dimension q of the data, "best" in the sense of the distance between the original data points and their projection.

PCA (4): how to choose dimension q?

- sometimes imposed by the application (e.g. for visualization $q = 2$ or 3)
- otherwise: make use of the spectrum:
  - simple: choose q where there is a "big step" (a.k.a. "elbow") in $\lambda_i / \sum_j \lambda_j$ plot (a.k.a. "Cattell's scree plot" or "explained variance"):



  - advanced: see:
    Tom Minka, *Automatic choice of dimensionality for PCA*, NIPS, 2000.
    `https://tminka.github.io/papers/pca/`

PCA (4)

Simple and **efficient** approximation method using sub-spaces (i.e. **linear** manifolds)

Weaknesses:

① linear method (precisely what makes it easy to use!)

② since the methods maximizes the (co)variance, it is strongly dependant on the measure units used for the features

In practice, except when the variance is *really* what has to be maximized, the data are renormalized before: it is then the correlation matrix which is decomposed rather than the (co)variance.

"Projection Pursuit"

Linear projection methods on a low dimension space (1, 2 ou 3) but maximizing another criterion than (co)variance.

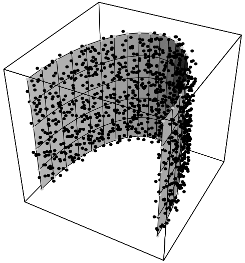
☞ No analytic solution: numerical optimization (iteration and local convergence)

$\Rightarrow$ The criterion has to be easily comptutable

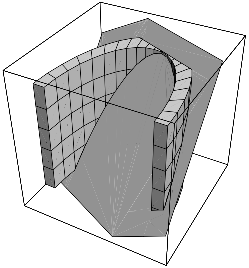
Several possible criteria:

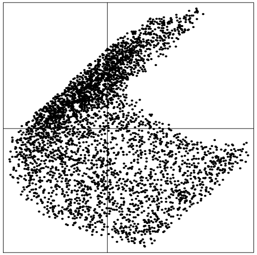
entropy, dispersion, higher momenta (> 2), divergence to normal distribution, ...

Linear vs. non-linear

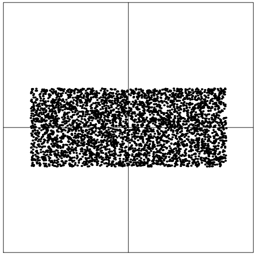


PCA:





non-linear method:

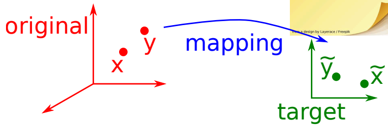


Non-linear Methods

- "principal curve" [Hastie & Stuetzle 89]
- ACC (neural net) [Demartines 94]
- Non-linear PCA (NLPCA) [Karhunen 94]
- Kernel PCA [Schölkopf, Smola, Müller 97]
- Gaussian process latent variable models (GPLVM) [Lawrence 03]

Multidimensional Scaling (MDS)

uses the chart of distances/dissimilarities between objects



Sammon Mapping: criterion:

$$C(d,\tilde{d}) = \sum_{x \neq y} \frac{\left(d(x,y) - \tilde{d}(\tilde{x},\tilde{y})\right)^2}{d(x,y)} = \sum_{x \neq y} \text{weight}(x,y) \cdot \text{error}(x,y)$$

where d is the dissimilarity in the original object space, and $\tilde{d}$ the dissimilarity in the projection space (e.g. Euclidian)

☞ more accurate representation of objects that are close

More recent alternatives:

- **t-SNE** (*t*-Distributed Stochastic Neighbor Embedding)
  [L.J.P. van der Maaten and G.E. Hinton; *Visualizing High-Dimensional Data Using t-SNE*; Journal of Machine Learning Research 9(Nov):2579-2605, 2008.]
- **UMAP** (Uniform Manifold Approximation and Projection for Dimension Reduction)
  [L. McInnes, Healy J., N. Saul and L. Grossberger; *UMAP: Uniform Manifold Approximation and Projection*; Journal of Open Source Software 3(29):861 (2018).]

Keypoints

- Many classification/clustering techniques (coming from different fields)
  ☞ Know their main characteristics, criteria
  ☞ Know at least two methods (e.g. Naive Bayes and K-means), that could be useful as baseline in any case.
- *A priori* choice of "the best method" is not easy:
  ☞ well define what you are looking for, means (time, samples, ...) you have access to
- It's even **more** difficult for Textual Data $\Rightarrow$ **preprocessing is really essential** (lemmatization, parsing, ...)
- Pay attention to use a proper methodolgy: good evaluation protocol, statistical tests, ...
- Classification/Clustering and Projection methods are complementary in (Textual) Data Analysis
- Use **several** representation/classification criteria
- Visualization: Focus on usefulness first:
  What does it bring/shows to the user? How is it useful?
  Pay attention not overwhelming the user...

References

- F. Sebastiani, *Machine learning in automated text categorization*, ACM Comput. Surv, 34(1): 1-47, 2002.
- C. Bishop, *Pattern Recognition and Machine Learning*, springer, 2006.
- B.D. Ripley, *Pattern recognition and Neural Networks*, Cambridge University Press, 1996.
- V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 2000.
- B. Schölkopf & A. Smola, *Learning with Kernels*, MIT Press, 2002.