TS349: Pattern Recognition

2/2 - Classification Methods

Rémi Giraud

remi.giraud@enseirb-matmeca.fr

2024-2025

3A Électronique / Option TSI

Introduction

Objectives

- **Vocabulary**
  (shape, pattern, descriptor, feature extraction, local, dense, keypoints, invariance, supervised, unsupervised, etc...)
- **Basic principles of pattern recognition methods** for image analysis.
- **Basic principles of data classification** using unsupervised and supervised methods.
- **Implementation and evaluation** of some approaches for pattern recognition.

Organization

Pattern Descriptors

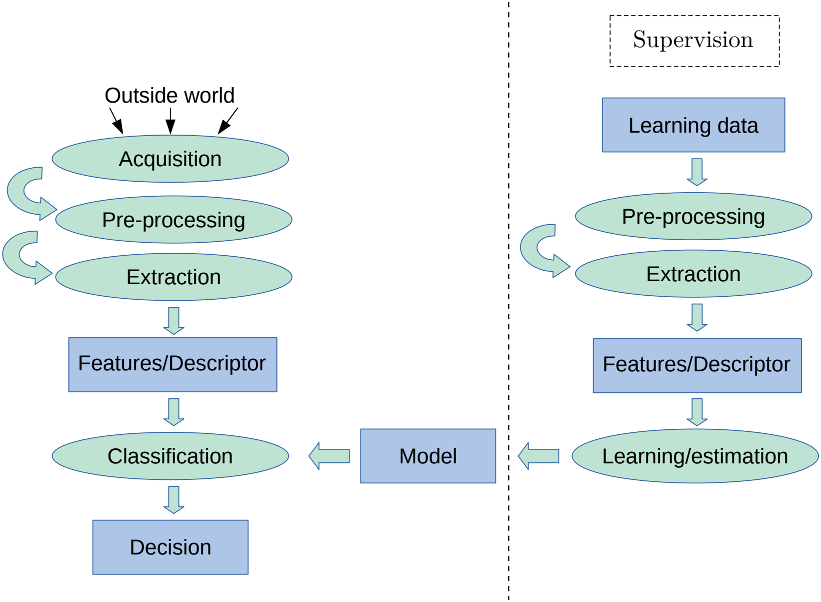
Course	Shape descriptors & extraction	1h20
Practical n°1	Shape recognition (Hough Transform)	4h
Course	Pattern descriptors & Dimension reduction	2h
Practical n°2	Texture classification (LBP, HOG)	3h20

Classification Methods

Course	Unsupervised classification	1h20
Practical n°1	Point cloud clustering	4h
Course	Supervised classification	1h20
Practical n°2	Digits classification	4h

Evaluation Practicals (x0.25) + Final exam (x0.75) 1h20

PR conception cycle



Classifier Properties

What would be the desired (or not) classifier properties?

Objective: accurately predict the class corresponding to an input descriptor

Properties:

- Accuracy (on what evaluation metric?)
- Allowing errors
- Use/need of learning data
- Robustness to outliers (very different features compared to the dataset)
- Binary decision/class probabilities
- Fast to train/apply
- Need for parameter tuning

What do we give to the classifier?

In our context, n image data described by p features (descriptor).

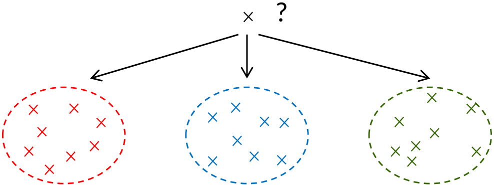
Can be seen as n statistical samples (or individuals), described by p variables.

A sample of n statistical samples (or individuals), described by p variables.

	Variable 1	Variable 2	...	Variable p
$\mathbf{X}_1$	x_{11}	x_{12}		
$\mathbf{X}_2$	x_{21}	...		
...	...	...		
$\mathbf{X}_n$	x_{n1}	...		

Objective

From this description, we want to classify each statistical sample into a given category.

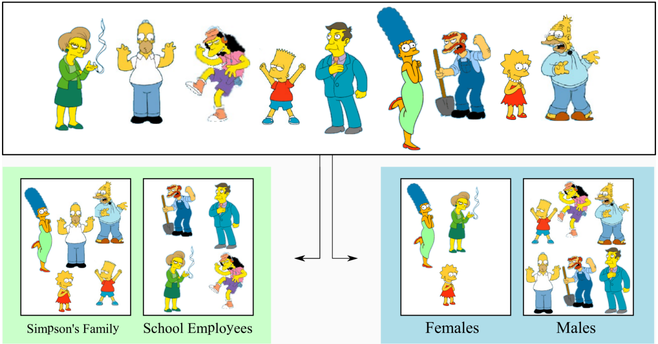


Unsupervised / Clustering

Without example data → no learning, the classes are blind.

Unsupervised / Clustering

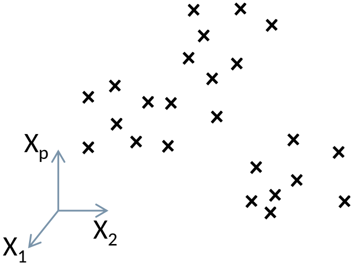
Without example data → no learning, the classes are blind.



(Source: Kasun Ranga Wijeweera)

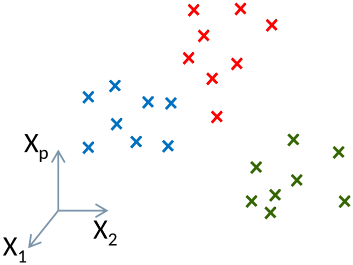
Unsupervised / Clustering

Without example data → no learning, the classes are blind.



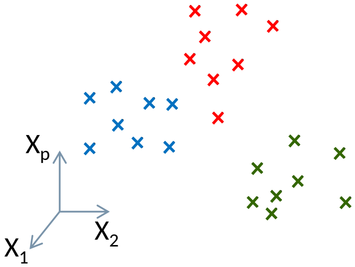
Unsupervised / Clustering

Without example data → no learning, the classes are blind.



Unsupervised / Clustering

Without example data → no learning, the classes are blind.



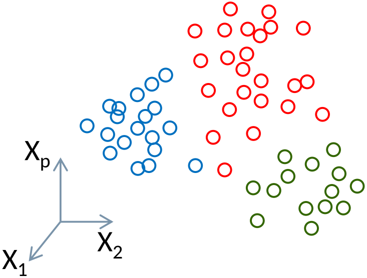
→ Allow to find structures in the data.

→ Guide the statistical studies, visualization, pre-processing, etc.

Approaches: Hierarchical grouping, K-means, etc.

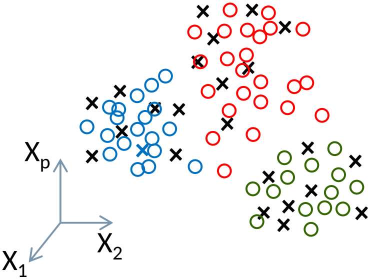
Supervised

With training data (o) available → used to classify the test data (x).



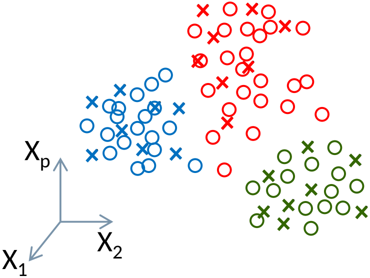
Supervised

With training data (o) available → used to classify the test data (x).



Supervised

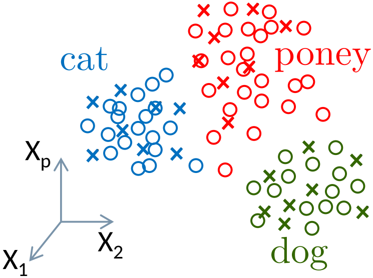
With training data (o) available → used to classify the test data (x).



Supervised

With training data (o) available → used to classify the test data (x).

Class information called semantic.



→ Automatic analysis/detection/recognition of data.

Approaches: Parametrics, nearest neighbors, ... deep learning.

Full PR system: Salmon vs Sea Bass

Context

Conveyor belt equipped with camera sensors, and we want to sort automatically split fish categories: Salmon and Sea Bass

Problem

Describe the main recommendations/instructions to consider to set up this system of recognition of Pisces

Full PR system: Salmon vs Sea Bass

What are the main steps of the system?

- Capture the image
- Isolate the fish
- Take measures
- Issue a decision

What are the problems in collecting the data?

- Lighting conditions
- Position of the fish on the treadmill (direction, rotation, concealment)
- White noise (camera)

What information allows recognition?

- Length
- Thickness
- Weight
- Number and shape of fins
- Shape of the tail, the head...

Full PR system: Salmon vs Sea Bass

Which characteristics to select?

- An expert (fishmonger) provides the following information:

  “a bass is usually bigger than a salmon”

  → use of length as a feature

  → decision following a threshold (boundary)

- How to choose such a threshold?

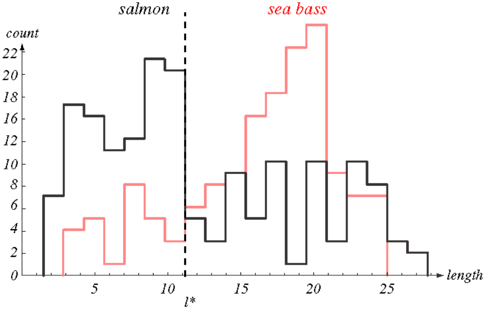
  - Calculation of a length histogram for both classes from a training set
  - Search for the threshold (partitioning into two classes):

    Manually (expertise)

    Automatically (for instance by maximizing information, entropy, ...)

Full PR system: Salmon vs Sea Bass

Histogram example with a selected threshold $l*$



What can we deduce from this?

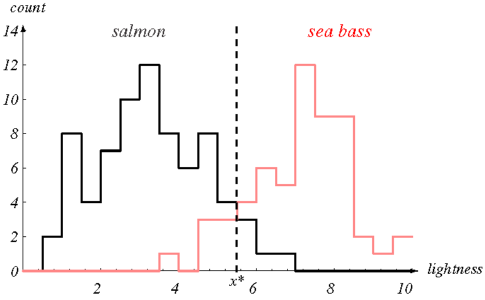
Although bass is larger than salmon on average, there are many samples that are not correctly classified based on a threshold

→ Test another characteristic to better separate the classes

(*e.g.*, the luminous intensity if generally the salmon are darker than bass)

Full PR system: Salmon vs Sea Bass

New histogram with a selected threshold $x*$



What can we deduce from this?

We see that the threshold selected for the light intensity allows to better differentiate the two classes of fish, but that the decision is not perfect

Full PR system: Salmon vs Sea Bass

Is it enough?

- Consider the cost of decision errors
- For example, if the objective is to fill cans, of customers may not appreciate having a different product...

How to improve the recognition?

- Consideration of multiple characteristics (vector)

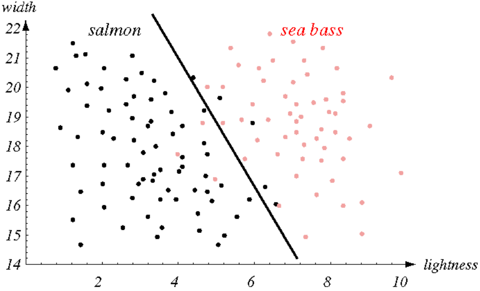
  “Bars are often darker and thicker than salmon”
- Two characteristics can be used to decide:

  - Lightness : x_1

  - Thickness : x_2

Full PR system: Salmon vs Sea Bass

Representation of samples according to thickness and lightness



A decision boundary can be obtained by drawing a straight line separating at better the classes

What can we deduce from this?

In this example, the result is close to that obtained for only the brightness

Use a curve instead of a line

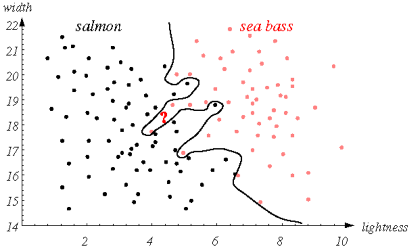
Full PR system: Salmon vs Sea Bass

First idea

- Learning model (curve) making it possible to obtain a zero error considering the learning set

  - Identical reality for the tests?

  - Behavior of learning if open world (new set of fish)?

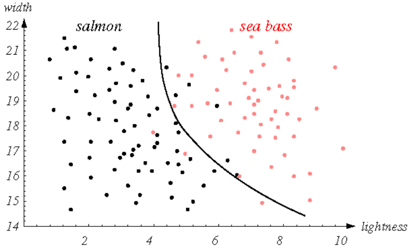


Generally

- “Simple” curve or set of curves separating at best the classes

  → Sufficiently large set of samples representative of reality

- Other possibility: introducing a reject class



Full PR system: Salmon vs Sea Bass

If we add other characteristics, can we still improve the recognition rate?

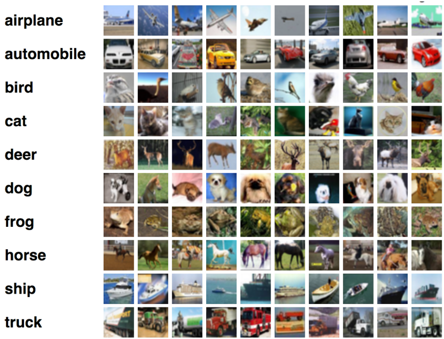
Potential issues?

- Correlation between characteristics
- Reliability of characteristics/measurements
- Noise (and corrections on measurements)
- Simplified space compared to reality
- Curse of Dimensionality

Image classification

Different types of classification

Global classification: a single label per image



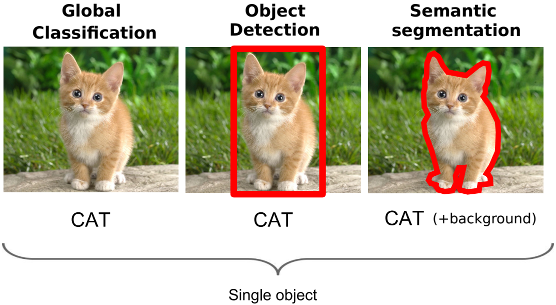
- Data: All image pixels
- Variables: Intensity or RGB colors?
- Class number: $K = 10$ (on this dataset CIFAR-10)

Image classification

Different types of classification

Object detection: Classification + Localization of object (bounding box)

Semantic segmentation: Classification + Segmentation (class for each pixel)

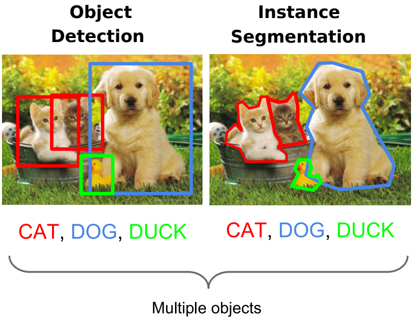


- Data: All image pixels
- Variables: Intensity or RGB colors?
- Class number: $K = \mathrm{N}$ (cat, dog, duck, ...)

Image classification

Different types of classification

Instance: Differentiation of objects of the same class

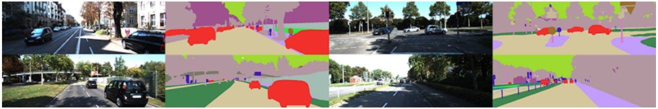


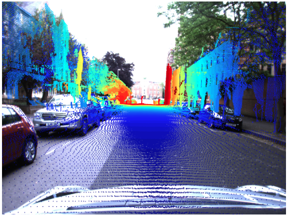
- Data: All image pixels
- Variables: Intensity or RGB colors?
- Class number: $K =$ N (cat, dog, duck, ...)

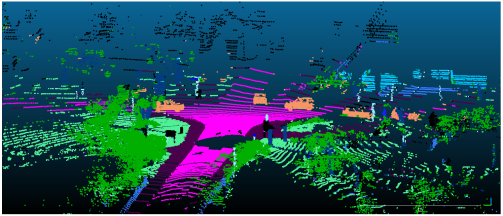
Image classification

Different types of classification

Other modalities, for example LiDAR point clouds:







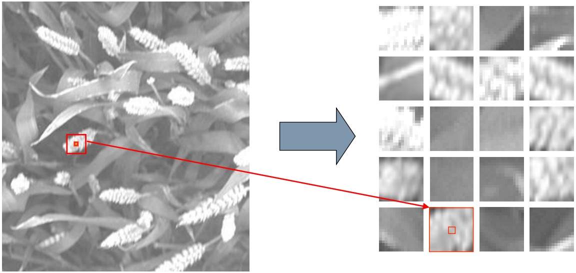
- Data: All points
- Variables: 3D position (X, Y, Z) $+$ reflectance?
- Class number: $K = n$ (building, road, car, pedestrian, etc...)

Image Classification

Some applications - Field data analysis

Wheat ear recognition for automatic counting:

Pixel segmentation into sub-windows (patches):

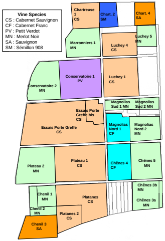


- Data: The n extracted patches
- Variables: texture descriptors computed on the patches? Measures?
- Class number: $K = 2$ (wheat ear, leaf or background)

Some applications - Field data analysis

Recognition of plant species through hyperspectral image analysis:

Objective: To automatically classify vine pixels into grape varieties.



Ground truth:
cépages on the domain

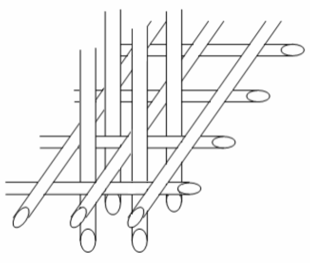
Hyperspectral image in fake colors
(144 spectral bands)

- Data: All image pixels
- Variables: reflectance ont the 144 channels ([400,950] nm) so $p = 144$
- Class number: $K = n$ (the different cépages)

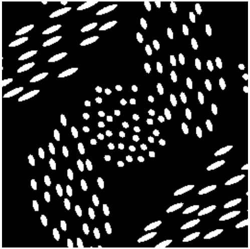
Image Classification

Some applications - Material analysis

3D fiber orientations in a composite material from a 2D section:



3D structure of the material



Appearance of the fibers on a 2D section (after segmentation)

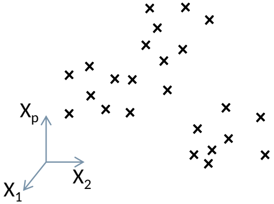
- Data: All image pixels
- Variables: geometrical shape descriptors (perimeter, surface, etc.)
- Class number: $K = 3$ (fibers in X, Y, or Z)

Unsupervised Classification

Unsupervised classification

Sample of n statistical data, described by p variables.

	Variable 1	Variable 2	...	Variable p
$\mathbf{X}_1$	x_{11}	x_{12}		
$\mathbf{X}_2$	x_{21}	...		
...	...	...		
$\mathbf{X}_n$	x_{n1}	...		



Search for the **best partition** of this sample:

- blindly (without example information),
- based on a certain criterion (a distance d).

Which criteria to use?

How to evaluate the quality of the classification according to this criteria?

Problem of computational complexity

- **High computational complexity**

  Total number of partitions of a set of n individuals:

n	1	2	3	4	5	...	n	...	11	12
P_n	1	2	5	15	52	...	$\sum_{k=0}^{n-1} \binom{n}{k} P_k$	...	678970	4213597

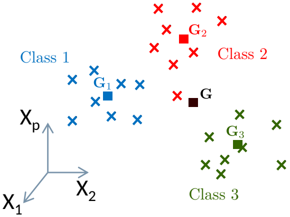
  $\rightarrow$ Impossible to consider all possible partitions to choose the best one.

- **We choose potentially sub-optimal methods:**
  - Hierarchical ascending classification
  - K-means method

Example of classification criteria

The relation intra / inter inertias

For K classes C_i, of barycenters $\mathbf{G}_i$, containing n_i elements



- Total inertia:

$$I_{tot} = \frac{1}{n}\sum_{i=1}^{n} d(\mathbf{x}_i, \mathbf{G})^2$$

- Inter classes inertia:

$$I_{inter} = \frac{1}{n}\sum_{i=1}^{K} n_i d(\mathbf{G}_i, \mathbf{G})^2$$

- Barycenters:

$$\mathbf{G}_i = \frac{1}{n_i}\sum_{\mathbf{x}\in C_i} \mathbf{x} \qquad \mathbf{G} = \frac{1}{K}\sum_{i=1}^{K} \mathbf{G}_i$$

- Intra classes inertia:

$$I_{intra} = \frac{1}{n}\sum_{i=1}^{K} I_i \quad \text{with} \quad I_i = \sum_{\mathbf{x}\in C_i} d(\mathbf{x}, \mathbf{G}_i)^2$$

→ An “appropriate” partition: **low intra inertia** and **high inter inertia**.

Hierarchical Ascending Classification (HAC)

Objective: Build a smaller set of classes through successive groupings

- **Algorithm:**

```
Start
   Create a class by sample ($n$ classes).
   Repeat
      Compute the distances between classes
      Select the couple of classes with the minimal distance
      Aggregate the two classes in one
   Until there is only one class remaining.
End
```

- **Need for:**
  - **Distance** between sample / classes
  - **Aggregation strategy**

Hierarchical Ascending Classification (HAC)

Objective: Build a smaller set of classes through successive groupings

- **Algorithm:**

```
Start
  Create a class by sample ($n$ classes).
  Repeat
    Compute the distances between classes
    Select the couple of classes with the minimal distance
    Aggregate the two classes in one
  Until there is only one class remaining.
End
```

- **Need for:**
  - **Distance** between sample / classes
  - **Aggregation strategy**

- **Complexity:** $\mathcal{O}(n^3) \rightarrow$ quite important

Distances

For two samples $\mathbf{x}$ and $\mathbf{y}$ (vectors of size p)

- **Minkowski** distance (L_n norm, general case):

$$d_n(\mathbf{x}, \mathbf{y}) = \left(\sum_{j=1}^{p} |x_j - y_j|^n \right)^{1/n}$$

- **Hamming** distance (L_1 norm):

$$d_1(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{p} |x_j - y_j|$$

- **Euclidean** distance (L_2 norm):

$$d_2(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{j=1}^{p} |x_j - y_j|^2}$$

- **Maximum** distance (∞ norm):

$$d_\infty(\mathbf{x}, \mathbf{y}) = \max_{j=1 \ldots p} |x_j - y_j|$$

Distances

For two samples $\mathbf{x}$ and $\mathbf{y}$ (vectors of size p)

- **Quadratic** distance: $d(\mathbf{x},\mathbf{y})^2 = (\mathbf{x}-\mathbf{y})^t\mathbf{M}(\mathbf{x}-\mathbf{y})$

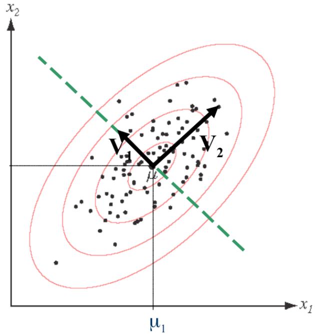
  where $\mathbf{M}$ is a matrix of size $p\times p$.

  Particular cases:

  - Identity matrix: $\mathbf{M} = \mathbf{I}$
  - **Mahalanobis** distance: $\mathbf{M} = \mathbf{C}^{-1}$ where $\mathbf{C}$ is the covariance matrix.

  → Projection following the eigen vectors of $\mathbf{C}$

  → Normalization over each axis



Aggregation strategies

- **Simple link** or minimal jump:

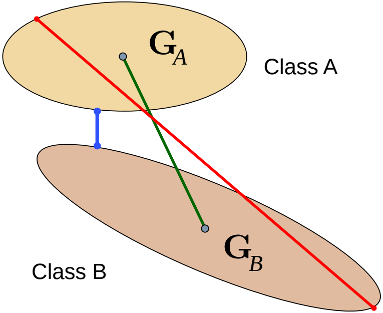
$$D(\mathbf{A},\mathbf{B}) = \min_{\mathbf{i}\in\mathbf{A},\mathbf{j}\in\mathbf{B}} d(\mathbf{i},\mathbf{j})$$

- **Complete link**:

$$D(\mathbf{A},\mathbf{B}) = \max_{\mathbf{i}\in\mathbf{A},\mathbf{j}\in\mathbf{B}} d(\mathbf{i},\mathbf{j})$$

- **Average link**:

$$D(\mathbf{A},\mathbf{B}) = d(\mathbf{G}_A,\mathbf{G}_B)$$



- **Ward's method**:

$$D(\mathbf{A},\mathbf{B}) = \frac{n_A n_B}{n_A + n_B} d(\mathbf{G}_A,\mathbf{G}_B)$$

(ensuring at each step that the within-class inertia is as low as possible)

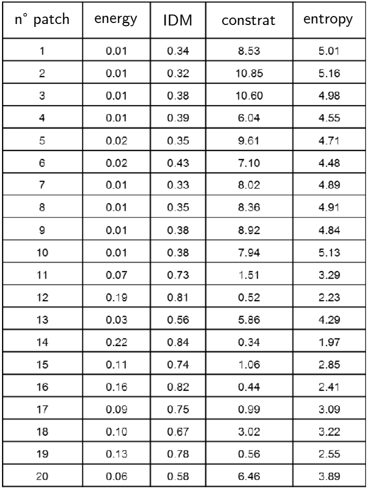
Example: wheat ear recognition by image analysis

Texture descriptors

On each image patch, we calculate 4 statistical attributes derived from co-occurrence matrices. [Har79].

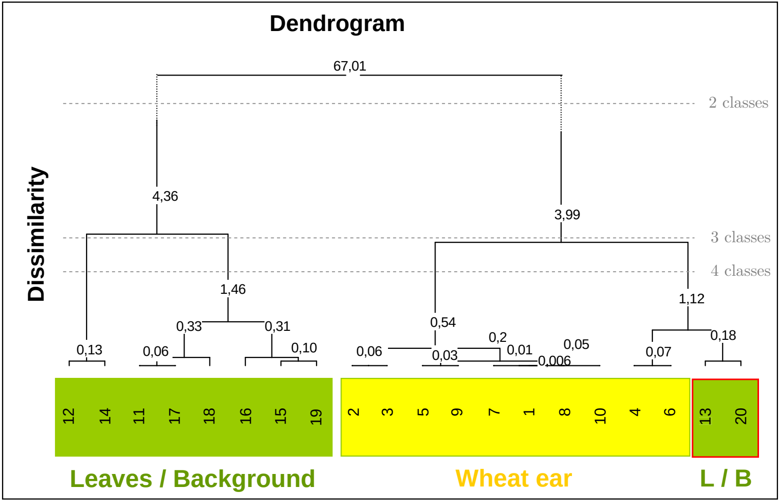
Question

Do these attributes allow for an effective differentiation of the two classes?

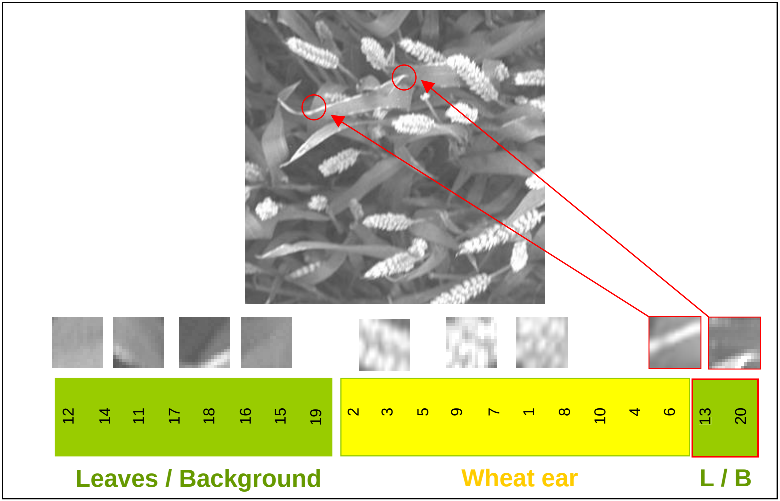


n° patch	energy	IDM	constrat	entropy
1	0.01	0.34	8.53	5.01
2	0.01	0.32	10.85	5.16
3	0.01	0.38	10.60	4.98
4	0.01	0.39	6.04	4.55
5	0.02	0.35	9.61	4.71
6	0.02	0.43	7.10	4.48
7	0.01	0.33	8.02	4.89
8	0.01	0.35	8.36	4.91
9	0.01	0.38	8.92	4.84
10	0.01	0.38	7.94	5.13
11	0.07	0.73	1.51	3.29
12	0.19	0.81	0.52	2.23
13	0.03	0.56	5.86	4.29
14	0.22	0.84	0.34	1.97
15	0.11	0.74	1.06	2.85
16	0.16	0.82	0.44	2.41
17	0.09	0.75	0.99	3.09
18	0.10	0.67	3.02	3.22
19	0.13	0.78	0.56	2.55
20	0.06	0.58	6.46	3.89

Example: wheat ear recognition by image analysis



Example: wheat ear recognition by image analysis

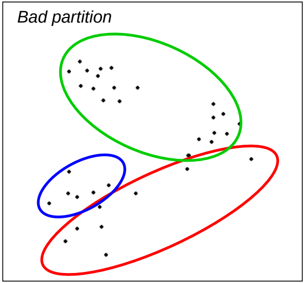


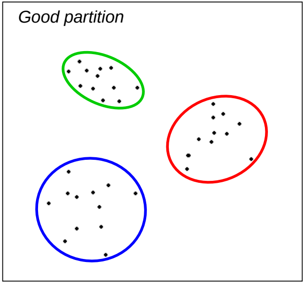
The method of the moving centers, or K-means

[MacQueen1967]

- **Hypothesis:** the number K of classes is known.
- **Principle:** Find the best partition of the set of individuals into K groups: providing the most compact and farthest groups possible from each other.

  $\rightarrow$ Minimizing intra class inertia and maximizing inter class inertia.





K-means algorithm

Algorithm:

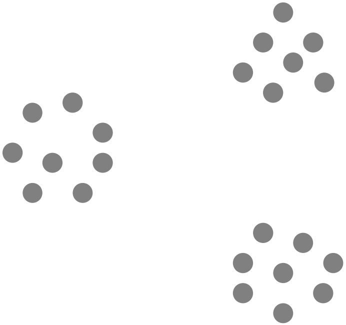
```
Start
  Choose the centers ($K$ points $\mathbf{z}_1$,...,$\mathbf{z}_K$ in the data space).
  Repeat
    Segment the space into $K$ classes $C_1$,...,$C_K$
    ($C_i$ is composed of the points closest from $\mathbf{z}_i$ than the other centers $\mathbf{z}_j$)
    Replace the $\mathbf{z}_i$ by the barycenters $\mathbf{G}_i$ of classes $C_i$
  Until minimization of the intra class inertia.
Fin
```

Remarks:

- The algorithm converges towards a **local minimum** of intra class inertia.
- If a class gets empty, we can draw a new random seed.
- Computational complexity of the algorithm...

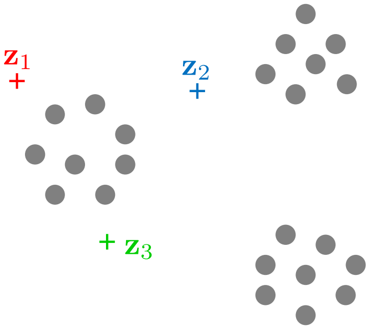
Lloyd's algorithm: Example

- Choice of the class number: 3.



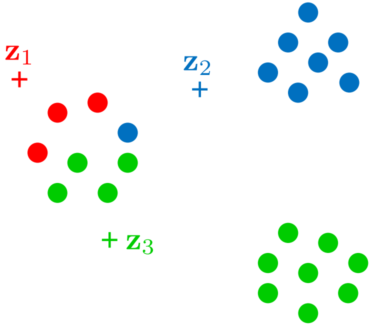
Lloyd's algorithm: Example

- Initialization of the class centers: $\mathbf{z}_1$, $\mathbf{z}_2$, and $\mathbf{z}_3$ (by random sampling).



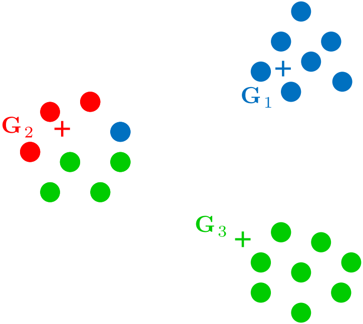
Lloyd's algorithm: Example

- Segmentation of the space into 3 classes.



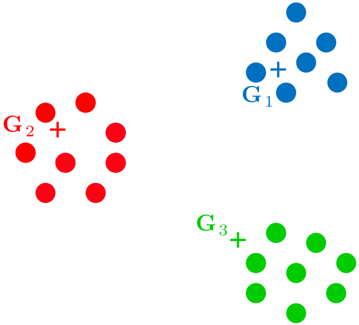
Lloyd's algorithm: Example

- Update of the class centers: barycenters $\mathbf{G}_1$, $\mathbf{G}_2$, and $\mathbf{G}_3$.



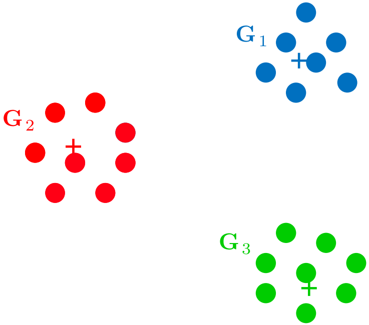
Lloyd's algorithm: Example

- Segmentation of the space into 3 classes using $\mathbf{G}_1$, $\mathbf{G}_2$, and $\mathbf{G}_3$.



Lloyd's algorithm: Example

- Update of the class centers: barycenters $\mathbf{G}_1$, $\mathbf{G}_2$, and $\mathbf{G}_3$.



Remarks on the K-means method

Advantages:

- Simplicity of implementation
- Unique parameter: K
- Computational complexity according to the number if iterations: $\mathcal{O}(N_{\text{iter}}\ K\ n\ (p))$

Remarks on the K-means method

Limitations and solutions:

- Computational complexity:
  $\rightarrow$ A pre-processing limiting the number of variables ($< p$) can facilitate the aggregation (ex. PCA)
- The number K of classes results from:
  - A knowledge a priori,
  - Successive tries,
  - An automatic method (hierarchical aggregation for instance).
- Impact of the initialization (convergence towards a local minimum):
  $\rightarrow$ improved initialization (e.g. K-means$++$).
- Sensitivity to outlier data:
  $\rightarrow$ Robust estimation of the centers (e.g. K-medoids).
  Barycenters $=$ "central" points, *i.e.*, points having the lowest average distance to the other points in the same class.
  Complexity $\mathcal{O}(n) \rightarrow \mathcal{O}(n^2)$

K-means++ [David2007]

Optimization of the initialization:

We choose as centers spaced points among the samples X

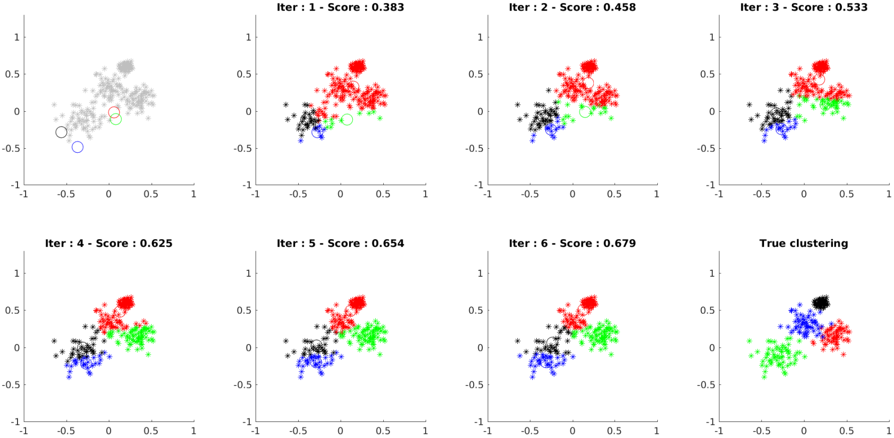
Let $D(x)$, the smallest distance of a point x to the already existing G_i.

Algorithm:

- 1 - Choice of the first center G_1, randomly taken among the data X
- 2.1 - Computation of the $D(x)$
- 2.2 - Choice of a new center $G_i = x \in X$
  randomly sampled following the probability law $\mathcal{L}(D) = \frac{D(x)^2}{\sum_{x \in X} D(x)^2}$
- 3 - Repeat 2.1 and 2.2 until K centers are selected

Example of the K-means on a 2D point cloud ($K = 4$)

- Impact of the initialization (convergence towards a local minimum)



Iter : 1 - Score : 0.383

Iter : 2 - Score : 0.458

Iter : 3 - Score : 0.533

Iter : 4 - Score : 0.625

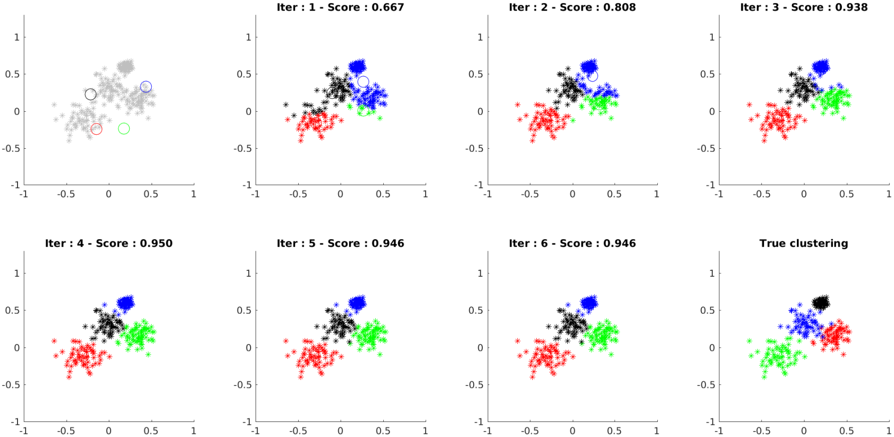
Iter : 5 - Score : 0.654

Iter : 6 - Score : 0.679

True clustering

Example of the K-means on a 2D point cloud ($K = 4$)

- Impact of the initialization (convergence towards a local minimum)



Iter : 1 - Score : 0.667

Iter : 2 - Score : 0.808

Iter : 3 - Score : 0.938

Iter : 4 - Score : 0.950

Iter : 5 - Score : 0.946

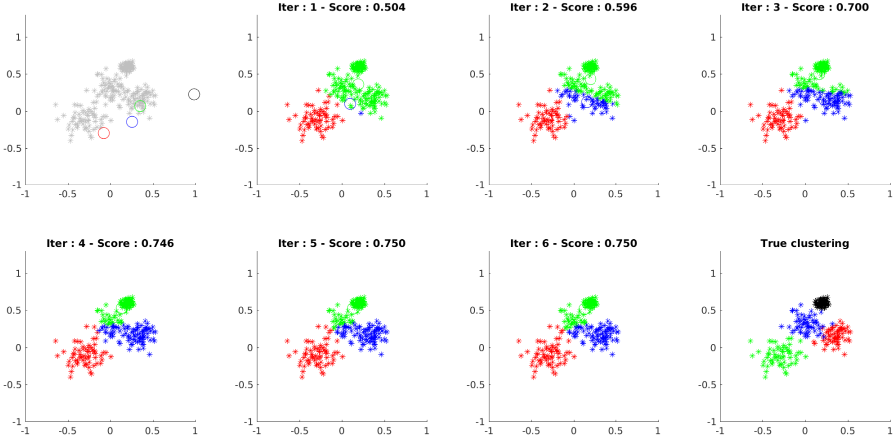
Iter : 6 - Score : 0.946

True clustering

Example of the K-means on a 2D point cloud ($K = 4$)

- Impact of the initialization (convergence towards a local minimum)

Risk of loosing some clusters!



Iter : 1 - Score : 0.504

Iter : 2 - Score : 0.596

Iter : 3 - Score : 0.700

Iter : 4 - Score : 0.746

Iter : 5 - Score : 0.750

Iter : 6 - Score : 0.750

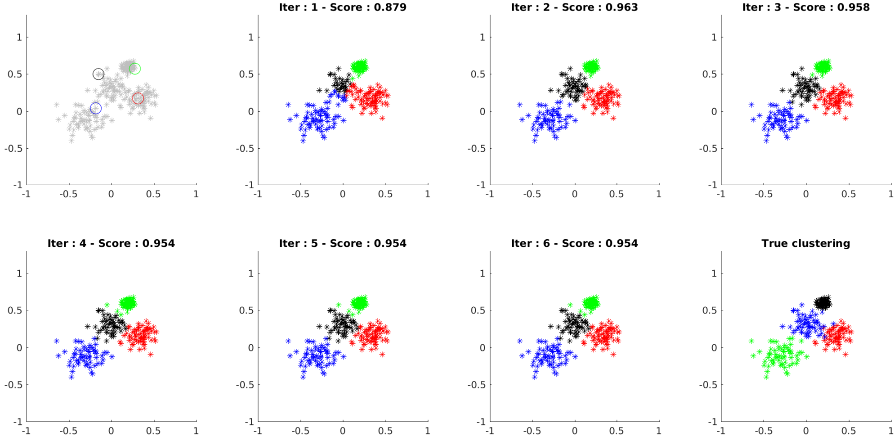
True clustering

Example of the K-means on a 2D point cloud ($K = 4$)

- Impact of the initialization (convergence towards a local minimum)

  Risk of loosing some clusters!

  $\rightarrow$ K-means++: relevant initialization



Iter : 1 - Score : 0.879

Iter : 2 - Score : 0.963

Iter : 3 - Score : 0.958

Iter : 4 - Score : 0.954

Iter : 5 - Score : 0.954

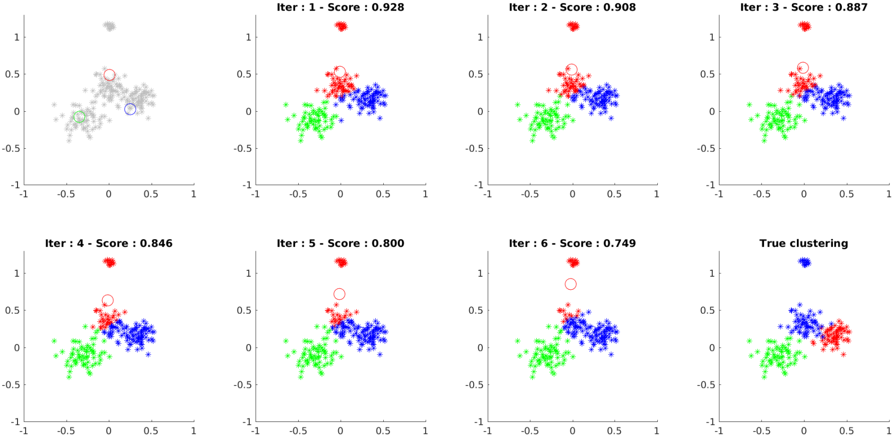
Iter : 6 - Score : 0.954

True clustering

Example of K-means of a 2D point cloud ($K = 3$)

- Sensitivity to outlier data

Averaging the data can lead to irrelevant class centers!

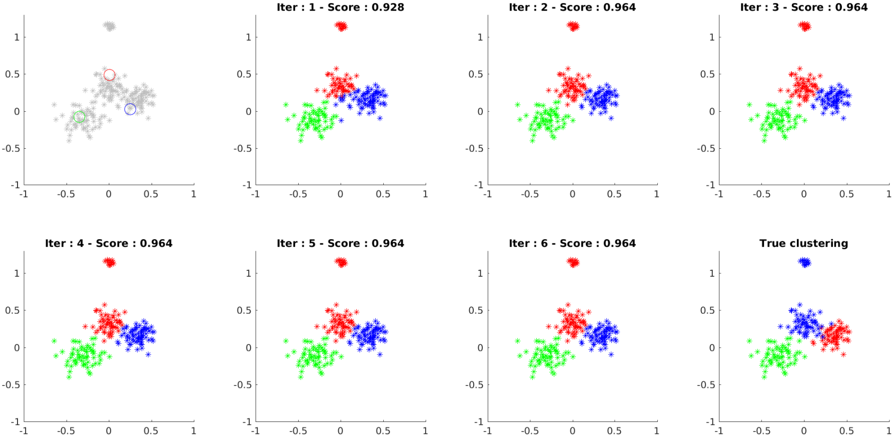


Example of K-means of a 2D point cloud ($K = 3$)

- Sensitivity to outlier data

  Averaging the data can lead to irrelevant class centers!

  $\rightarrow$ K-medoids: less sensitive to outlier data

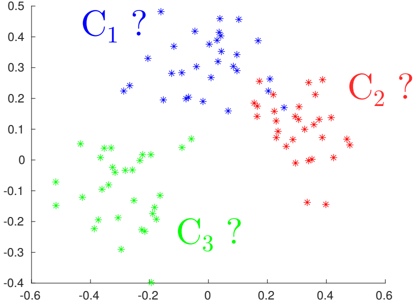


Evaluation of performance

Accurate rate (Overall Accuracy)

$$OA = \frac{\sum_{i=1}^{n} \delta_{C_{[n_i]},\hat{C}_{[n_i]}}}{n} = \frac{\text{number of accurate classification}}{\text{total number}}$$

Abstract classes → need for testing all combinations



Application example: Over-segmentation of images

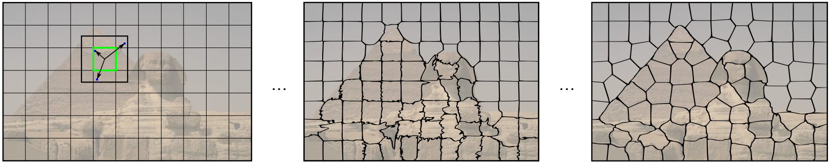
Decomposition of the image into *superpixels*: homogeneous connected regions

→ Reduction of the number of considered elements

→ Respect of the object boundaries

Algorithm [Achanta2012]:

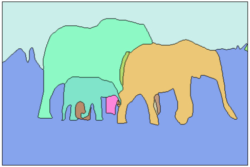
- Initialization of the centers as a regular grid
- Locally constrained K-means
  Color and spatial distance on all pixels
- Refinement step to ensure connectivity for all clusters



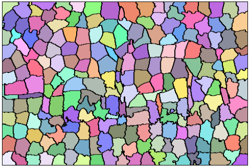
Evaluation of performance

Comparison between a segmentation containing one or several objects and an over-segmentation of an image containing numerous regions

image



ground truth $\mathcal{G}$



segmentation into superpixels $\mathcal{S}$

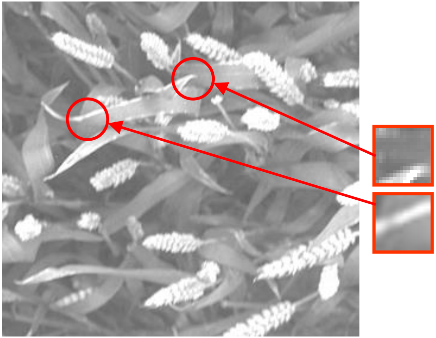
Achievable Segmentation Accuracy (ASA) metric:

$$ASA(\mathcal{S},\mathcal{G}) = \frac{1}{\sum_{S_k\in\mathcal{S}} |S_k|} \sum_{S_k\in\mathcal{S}} \max_{G_i\in\mathcal{G}} |S_k \cap G_i|$$

Example: wheat ear recognition by image analysis

K-means result with 2 classes:

Same result as with the HAC: most of patches are well classified (except leaves 3 and 10)



Class	Class 1	Class 2
Number	8	12
	leaf 1	ear 1
	leaf 2	ear 2
	leaf 4	ear 3
	leaf 5	ear 4
	leaf 6	ear 5
	leaf 7	ear 6
	leaf 8	ear 7
	leaf 9	ear 8
		ear 9
		ear 10
		leaf 3
		leaf 10

K-means vs Hierarchical Ascending Classification

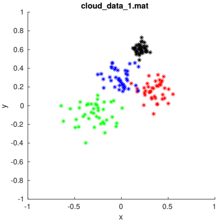
With K-means:

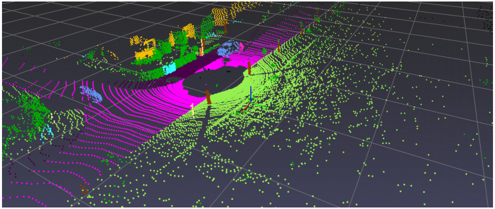
- A data point can change its cluster between two iterations.
  With hierarchical clustering, an assignment is final.
- Different initializations can lead to different solutions
  One can study a set of solutions by modifying the starting centroids.
- Not easy to estimate a relevant number of clusters, nor to visualize the proximity between clusters or objects.

→ Complementarity of the methods

Practical nº1: Unsupervised classification

- **Data:** 2D ans 3D point clouds to classifiy





- **Hierarchical ascending classification**
  - Implementation, comparison of 4 aggregation strategies
- K**-means algorithm,** K**-means++,** K**-medoids**
  - Implementation, computation of the intra and inter classes inertias
  - Over-segmentation, computation of the ASA

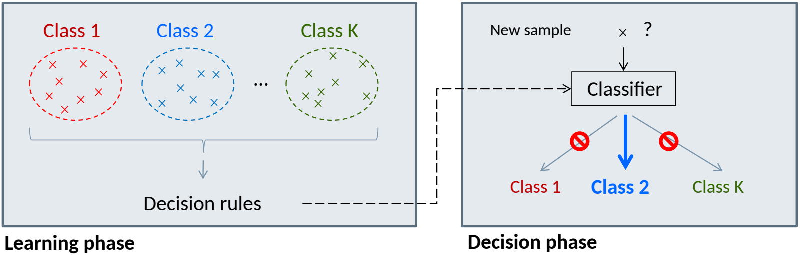
Supervised Classification

The process of supervised classification

The classification of individuals into K categories is not done "blindly".

It requires a **learning phase** where we learn how to recognize individuals, i.e. to associate them with one or the other of the categories.

A new individual is then "classified" into the most similar category: this is the **decision phase**.



Classification: an estimation problem

The output random variable, called G

- It is the variable we seek to predict.
- It is a categorical variable.
- It takes its values into a finite set:

$$G \in \mathcal{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_k, \ldots, \mathcal{G}_K\}$$

Classification examples:

- $G \in \mathcal{G} = \{$Ear, Leaves/Background$\}$
- $G \in \mathcal{G} = \{$X Fibers, Y Fibers, Z Fibers$\}$
- $G \in \mathcal{G} = \{$Ground, Merlot noir, Sauvignon, ... $\}$
- $G \in \mathcal{G} = \{$Dog, Cat, Boat, Plane, ... $\}$

Classification: an estimation problem

The input random variable, called X

- In the general case, it is a vector of random variables

$$\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_j, \ldots, \mathbf{X}_p)^t \in \mathbb{R}^p$$

- Realization of X on the sample i:

$$\mathbf{x}_i = (x_{i1}, \ldots, x_{ij}, \ldots, x_{ip})^t \in \mathbb{R}^p, \qquad i = 1, \ldots n$$

Reformulation of the classification problem

- Making the best possible prediction $\hat{G}$ of the output variable G from an input sample $\mathbf{x}$.
  Like G, $\hat{G}$ takes its values into $\mathcal{G}$:

$$\hat{G} \in \mathcal{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_k, \ldots, \mathcal{G}_K\}$$

The learning: a two steps phase

The learning:

- Analyze the characteristics of each class and determine the rules for classification of new individuals
- Performed on the learning set (i.e. a sample of individuals whose class membership is known a priori)

The validation:

- Applying decision rules to a new sample
- Done on the test set, i.e. individuals whose class membership is known a priori but not use in the classification process (blind classification)
- Comparison of the classification results to prior knowledge

Learning/Training and testing sets

The learning/training set:

- To develop the classifier (build decision rules)

$$E_A = \{(\mathbf{x}_1, g_1), \ldots (\mathbf{x}_{n_a}, g_{n_a})\}$$

with x_i a realization of X (individual to be classified) and $g_i \in \mathcal{G}$ its class

The test/testing set (or validation):

- To test the classifier on known data that has not been used for training:

$$E_V = \{(\mathbf{x}'_1, g'_1), \ldots (\mathbf{x}'_{n_v}, g'_{n_v})\}$$

with x'_i a realization of X (individual to be classified) and $g'_i \in \mathcal{G}$ its class

To measure the performance of a classifier (on the validation set)

Confusion matrix $\mathbf{C} = [c_{kl}]_{k,l \in \{1,\dots,K\}}$

predicted class

real class

$$\begin{array}{c|ccccc}
 & \hat{G}_1 & \cdots & \hat{G}_l & \cdots & \hat{G}_K \\
\hline
\mathcal{G}_1 & c_{11} & \cdots & c_{1l} & \cdots & c_{1K} \\
\vdots & & \ddots & & & \\
\mathcal{G}_k & c_{k1} & & c_{kl} & & c_{kK} \\
\vdots & & & & \ddots & \\
\mathcal{G}_K & c_{K1} & & c_{Kl} & & c_{KK}
\end{array}$$

c_{kl}: number of elements of $\mathcal{G}_k$ in $\hat{G}_l$

Accuracy rate

$$OA = \frac{\text{Trace}(\mathbf{C})}{n_v} = \frac{\sum_k c_{kk}}{\sum_{k,l} c_{kl}} = \frac{\text{number of accurate classification}}{\text{total number}}$$

To measure the performance of a classifier (on the validation set)

Kappa Coefficient

Measures the efficiency of the classifier following randomness:

$$kappa = \frac{p_c - p_h}{1 - p_h}$$

where p_c is the accuracy rate ($p_c = OA$),
and p_h is the accuracy rate due to randomness:

$$p_h = \frac{1}{n_v^2} \sum_k \mathbf{c_{k.}} \mathbf{c_{.k}} \quad \text{where} \quad \mathbf{c_{k.}} = \sum_l c_{kl} \quad \text{and} \quad \mathbf{c_{.k}} = \sum_l c_{lk}$$

(Landis & Koch)

Quality of the classifier	**Kappa**
Excellent	1,00 - 0,81
Good	0,80 - 0,61
Average	0,60 - 0,41
Low	0,40 - 0,21
Negligible	0,20 - 0
Bad	< 0

To measure the performance of a classifier (on the validation set)

Kappa Coefficient

Measures the efficiency of the classifier following randomness:

$$kappa = \frac{p_c - p_h}{1 - p_h}$$

where p_c is the accuracy rate ($p_c = OA$),
and p_h is the accuracy rate due to randomness:

$$p_h = \frac{1}{n_v^2} \sum_k \mathbf{c_{k.}} \mathbf{c_{.k}} \quad \text{where} \quad \mathbf{c_{k.}} = \sum_l c_{kl} \quad \text{and} \quad \mathbf{c_{.k}} = \sum_l c_{lk}$$

Examples:

- Perfect classifier, balanced classes

$$c_{ij} = 0 \quad \forall i \neq j \qquad p_h = \frac{1}{n_v^2} K \frac{n_v^2}{K^2} = \frac{1}{K}$$

$$c_{ii} = \frac{n_v}{K}$$

To measure the performance of a classifier (on the validation set)

Kappa Coefficient

Measures the efficiency of the classifier following randomness:

$$kappa = \frac{p_c - p_h}{1 - p_h}$$

where p_c is the accuracy rate ($p_c = OA$),
and p_h is the accuracy rate due to randomness:

$$p_h = \frac{1}{n_v^2} \sum_k \mathbf{c_{k.}} \mathbf{c_{.k}} \quad \text{where} \quad \mathbf{c_{k.}} = \sum_l c_{kl} \quad \text{and} \quad \mathbf{c_{.k}} = \sum_l c_{lk}$$

Examples:

- Unbalanced classes

$$\begin{bmatrix} 85 & 4 \\ 6 & 5 \end{bmatrix} \qquad p_h = \frac{1}{100^2}\left((85+4)*(85+6)+(5+6)*(5+4)\right) = \frac{8198}{10000} = 0.8198$$

$$kappa = \frac{p_c - p_h}{1 - p_h} = \frac{0.9 - 0.8198}{1 - 0.8198} = 0.4451$$

Bayesian Classification: The estimation problem

Cost criteria

To estimate the realization $\hat{G}$ of G, a cost criteria or function $L(k,l)$ is necessary:

$L(k,l)$ is the price to pay if an observation of $\mathcal{G}_k$ is classified in $\mathcal{G}_l$

Examples:

- Binary cost: $L(k,l) = \begin{cases} 0 & \text{if } k=l \\ 1 & \text{otherwise} \end{cases}$

- With different penalizations according to the risk level:

$$\begin{array}{c c} & \begin{array}{cc} \text{healthy} & \text{sick} \end{array} \\ \begin{array}{c} \text{healthy} \\ \text{sick} \end{array} & \begin{bmatrix} 0 & \text{q} \\ \text{p} & 0 \end{bmatrix} \end{array} \quad \text{with} \;\; \text{p>q}$$

The estimation problem

Expected value of the prediction error EPE

The principle of the classifier is to minimize the EPE, defined by:

$$EPE = E_{\mathbf{G},\mathbf{X}}[L(\mathbf{G}, \hat{G}(\mathbf{X}))]$$

By conditioning with respect to $\mathbf{X}$, we get:

$$EPE = E_{\mathbf{X}} E_{\mathbf{G}|\mathbf{X}}[L(\mathbf{G}, \hat{G}(\mathbf{X}))] = E_{\mathbf{X}} \sum_{k=1}^{K} p(\mathbf{G} = \mathcal{G}_k | \mathbf{X} = \mathbf{x}) L(\mathcal{G}_k, \hat{G}(\mathbf{X}))$$

In practice, we do not work on the set of possible $\mathbf{X}$ but on a particular value $\mathbf{x}$. We search to solve:

$$\hat{G}(\mathbf{x}) = \underset{g \in \mathcal{G}}{\text{argmin}} \sum_{k=1}^{K} p(\mathcal{G}_k | \mathbf{x}) L(\mathcal{G}_k, g)$$

The estimation problem

Case of the binary cost (0/1)

- $L(k,l) = \begin{cases} 0 & \text{if } k = l \\ 1 & \text{otherwise} \end{cases}$

The principle of the classifier is to minimize the EPE, defined by:

$$\hat{G}(\mathbf{x}) = \underset{g \in \mathcal{G}}{\text{argmin}} \sum_{\mathcal{G}_k \neq g} p(\mathcal{G}_k|\mathbf{x}) = \underset{g \in \mathcal{G}}{\text{argmin}} \left[1 - p(g|\mathbf{x})\right]$$

This corresponds to:

$$\hat{G}(\mathbf{x}) = \underset{g \in \mathcal{G}}{\text{argmax}}\ p(g|\mathbf{x})$$

It is called a maximum a posteriori probability (MAP) estimate: we choose the class g that maximizes the probability a posteriori $p(g|\mathbf{x})$.

The estimation problem

Bayes theorem

$$p(g|\mathbf{x}) = \frac{f(\mathbf{x}|g)p(g)}{f(\mathbf{x})}$$

with:

$p(g) = p(G = g)$	the a priori probability of the class g
$f(\mathbf{x}) = f_{\mathbf{X}}(\mathbf{x})$	the probability density of the input variable $\mathbf{X}$
$f(\mathbf{x}|g) = f_{\mathbf{X}}(\mathbf{x}|G = g)$	the probability density of $\mathbf{X}$ in class g

The Bayesian classifier:

$$\hat{G}(\mathbf{x}) = \underset{g \in \mathcal{G}}{\text{argmax}}\ f(\mathbf{x}|g)p(g)$$

In practice:

- Probabilities $p(g)$ and laws $f(\mathbf{x}|g)$ are known (a priori): not realistic!
- Otherwise, they have to be estimated...

Bayesian learning

To perform a Bayesian learning, we have to know:

- The law (type and parameters) of each class
- The a priori probabilities of each class

If they are not know, we have to estimate them.

A priori probabilities $p(g)$

Estimated from the learning set:

$$\hat{p}(g) = \frac{\#\{\mathbf{x} \in E_A | \mathbf{x} \in g\}}{\#\{\mathbf{x} \in E_A\}} = \frac{n_g}{n_A}$$

Bayesian learning

Conditional law $f(x|g)$

Let $E_g \subset E_A$ be the learning individuals in the class g:

$$E_g = \{\mathbf{x}_1, \ldots, \mathbf{x}_{n_g}\}$$

We suppose that we have a law model for $f(\mathbf{x}|g)$.

Let $\boldsymbol{\theta_g} = \{\theta_{g,1}, \ldots \theta_{g,m}\}$ be the parameters of this law.

Finding $f(\mathbf{x}|g)$, is finding $\boldsymbol{\theta_g}$.

Maximum Likelyhood Estimator: $\hat{\boldsymbol{\theta}}_{\boldsymbol{g}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ f(E_g|\boldsymbol{\theta})$

If we consider the individuals in E_g as independent:

$$\hat{\boldsymbol{\theta}}(g) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{i=1}^{n_g} f(\mathbf{x}_i|\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{i=1}^{n_g} \log f(\mathbf{x}_i|\boldsymbol{\theta})$$

Resolution: Solve the partial derivative system:

$$\forall l, \qquad \frac{\partial}{\partial \theta_l} \sum_{i=1}^{n_g} \log f(\mathbf{x}_i|\boldsymbol{\theta}) = 0$$

Linear Discriminant Analysis

Principle

The one of the Bayesian classification, under certain hypotheses...

Hypotheses

- Conditional laws: Gaussian laws of average $\boldsymbol{\mu}_k$...
- ... and same covariance matrix:

$$\mathbf{V}_k = \mathbf{V}, \qquad \forall k = \{1, \dots, K\}$$

The probability densities become:

$$f(\mathbf{x}|g_k) = \frac{1}{2\pi|\mathbf{V}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^t \mathbf{V}^{-1} (\mathbf{x}-\boldsymbol{\mu}_k)}$$

Linear Discriminant Analysis

Frontier between two classes g_k and g_l

Defined by:

$$\begin{aligned} p(g_k|\mathbf{x}) = p(g_l|\mathbf{x}) &\Leftrightarrow \frac{p(g_k|\mathbf{x})}{p(g_l|\mathbf{x})} = 1 \\ &\Leftrightarrow \log \frac{p(g_k|\mathbf{x})}{p(g_l|\mathbf{x})} = 0 \\ &\Leftrightarrow \log \frac{p(\mathbf{x}|g_k)p(g_k)}{p(\mathbf{x}|g_l)p(g_l)} = 0 \\ &\Leftrightarrow \log \frac{p(\mathbf{x}|g_k)}{p(\mathbf{x}|g_l)} + \log \frac{p(g_k)}{p(g_l)} = 0 \\ &\Leftrightarrow \log \frac{p(g_k)}{p(g_l)} - \frac{1}{2}\mu_k^t \mathbf{V}^{-1}\mu_k + \frac{1}{2}\mu_l^t \mathbf{V}^{-1}\mu_l + x^t \mathbf{V}^{-1}(\mu_k - \mu_l) = 0 \end{aligned}$$

→ Linear frontier in $\mathbf{x}$

In dimension p, it is a hyperplan (dimension $p-1$)

Linear Discriminant Analysis

Discriminative functions (linear)

Defined by:

$$\begin{aligned}\delta_k(\mathbf{x}) &= \log\left(f(\mathbf{x}|g_k)p(g_k)\right)\\ &= \mathbf{x}^t\mathbf{V}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^t\mathbf{V}^{-1}\boldsymbol{\mu}_k + \log p(g_k), \qquad \forall k \in \{1,\dots,K\}\end{aligned}$$

They define the decision rules:

$$\hat{G}(\mathbf{x}) = \underset{k}{\operatorname{argmax}}\ \delta_k(\mathbf{x})$$

Remarks

The Gaussian hypothesis is, in practice, not very restrictive.

The ADL is simple in principle, is relatively effective for many problems, as long as the classes remain (roughly) separable linearly.

Linear Discriminant Analysis

Learning

The parameters of the distributions are estimated on the training set:

Probabilities a priori: $\hat{p}(g_k) = n_k/n_A$

Conditional Probabilities (estimations according to the ML):

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k} \sum_{i=1}^{n_k} \mathbf{x}_i = \begin{bmatrix} \mu_{k,1} \\ \vdots \\ \mu_{k,p} \end{bmatrix}$$

$$\hat{\mathbf{V}} = \frac{1}{n_A} \sum_{k=1}^{K} \sum_{i=1}^{n_k} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^t = \frac{1}{n_A} \sum_{k=1}^{K} \begin{bmatrix} \sigma^2_{k,11} & \sigma^2_{k,12} & \cdots & \sigma^2_{k,1p} \\ \sigma^2_{k,21} & \sigma^2_{k,22} & \cdots & \sigma^2_{k,2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^2_{k,p1} & \sigma^2_{k,p2} & \vdots & \sigma^2_{k,pp} \end{bmatrix}$$

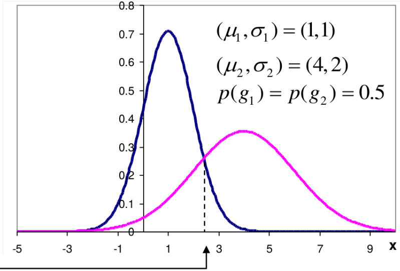
→ Equal covariance matrix assumption.

Example 1 (with known probability laws)

Two normal laws in dimension 1

- Input variable: $x \in \mathbb{R}$
- Two classes g_1, g_2 equally probable: $p(g_1) = p(g_2) = 0.5$
- Normal distributions:

$$f(x|g_k) = \frac{1}{\sigma_k\sqrt{2\pi}} e^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}}$$



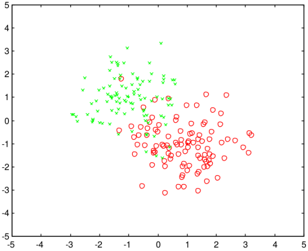
Bayesian decision frontier

Example 2 (with known probability laws)

Two normal laws in dimension 2

- Input variable: $\mathbf{x} \in \mathbb{R}^2$
- Two classes g_1, g_2 equally probable: $p(g_1) = p(g_2) = 0.5$
- Normal distributions:

$$f(\mathbf{x}|g_k) = \frac{1}{2\pi|\mathbf{V}_k|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^t \mathbf{V}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)}$$

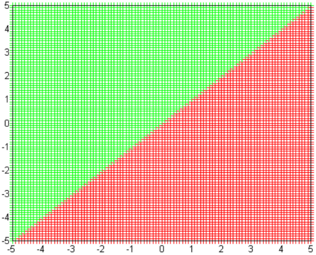


Two samples in $\mathbb{R}^2$

$$\mu_1 = (-1,1)^{-t}$$

$$\mu_2 = (1,-1)^{-t}$$

$$\mathbf{V}_1 = \mathbf{V}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



Bayesian decision frontier

Quadratic Discriminant Analysis

Principle

Same as LDA, but with slighly different hypotheses...

Hypotheses

- Conditional laws: Normal (or Gaussian) distribution with mean $\boldsymbol{\mu}_k$...
- ... and different covariance matrices:

$$\mathbf{V}_k \neq \mathbf{V}_l, \qquad \forall k, l = \{1, \ldots, K\}, k \neq l$$

The density probability functions are:

$$f(\mathbf{x}|g_k) = \frac{1}{2\pi|\mathbf{V}_k|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^t \mathbf{V}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)}$$

Decision frontiers

They derived from quadratic equations...

Quadratic Discriminant Analysis

Discriminative functions (quadratic too)

Defined by:

$$\delta_k(\mathbf{x}) = -\frac{1}{2}\log|\mathbf{V}_k| - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^t\mathbf{V}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) + \log p(g_k), \qquad \forall k \in \{1,\ldots,K\}$$

They define the decision rules:

$$\hat{G}(\mathbf{x}) = \underset{k}{\text{argmax}}\ \delta_k(\mathbf{x})$$

Remarks

The QDA is initially more attractive than the LDA because it can adapt to the case of different covariance distributions.

However, it may pose estimation difficulties in the case of a small training set...

Quadratic Discriminant Analysis

Learning

The parameters of the distributions are estimated on the training set:

Probabilities a priori: $\hat{p}(g_k) = n_k / n_A$

Conditional Probabilities (estimations according to the ML):

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k} \sum_{i=1}^{n_k} \mathbf{x}_i = \begin{bmatrix} \mu_{k,1} \\ \vdots \\ \mu_{k,p} \end{bmatrix}$$

$$\hat{\mathbf{V}}_k = \frac{1}{n_k} \sum_{i=1}^{n_k} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^t = \begin{bmatrix} \sigma^2_{k,11} & \dots & \sigma^2_{k,1p} \\ \vdots & \ddots & \vdots \\ \sigma^2_{k,p1} & \vdots & \sigma^2_{k,pp} \end{bmatrix}$$

Quadratic Discriminant Analysis

Learning

The parameters of the distributions are estimated on the training set:

Probabilities a priori: $\hat{p}(g_k) = n_k/n_A$

Conditional Probabilities (estimations according to the ML):

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k}\sum_{i=1}^{n_k}\mathbf{x}_i = \begin{bmatrix}\mu_{k,1}\\ \vdots\\ \mu_{k,p}\end{bmatrix}$$

$$\hat{\mathbf{V}}_k = \frac{1}{n_k}\sum_{i=1}^{n_k}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^t = \begin{bmatrix}\sigma^2_{k,11} & \dots & \sigma^2_{k,1p}\\ \vdots & \ddots & \vdots\\ \sigma^2_{k,p1} & \vdots & \sigma^2_{k,pp}\end{bmatrix}$$

Remarks

The number of parameters to estimate is larger than for the LDA:

$$(K-1) + Kp + K\left(\frac{p(p+1)}{2}\right) \quad \text{instead of} \quad (K-1) + Kp + \frac{p(p+1)}{2}$$

Linear Discriminant Analysis on “augmented” data

Principle

Sometimes quadratic discriminant analysis is replaced by a linear analysis on *augmented data*.

For example, in the case of two input variables: $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2)^t$

We will apply linear discriminant analysis on:

$$\mathbf{X}' = (\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_1\mathbf{X}_2, \mathbf{X}_1^2, \mathbf{X}_2^2)^t$$

Remarks

The results are generally quite similar to those of quadratic analysis.

“Naïve” Discriminant Analysis

Variables are considered to be **independent** from each other.

The covariance matrices become diagonal:

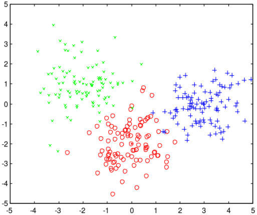
$$\hat{\mathbf{V}}_k = \frac{1}{n_k} \sum_{i=1}^{n_k} \begin{bmatrix} \sigma^2_{k,11} & 0 & \dots & 0 \\ 0 & \sigma^2_{k,22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \vdots & \sigma^2_{k,pp} \end{bmatrix} \text{ et } \hat{\mathbf{V}}_k^{-1} = \begin{bmatrix} \frac{1}{\sigma^2_{k,11}} & 0 & \dots & 0 \\ 0 & \frac{1}{\sigma^2_{k,22}} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \vdots & \frac{1}{\sigma^2_{k,pp}} \end{bmatrix}$$

→ The calculations are simpler, and this can potentially reduce the risk of overfitting (since the dependence between features is completely ignored).

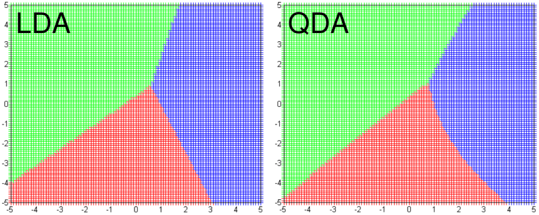
Examples

3 2D normal laws with the same covariance matrix

$(\mathbf{V} = \sigma^2 \mathbf{I})$



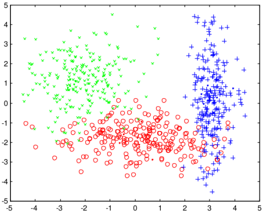
Three classes in R^2



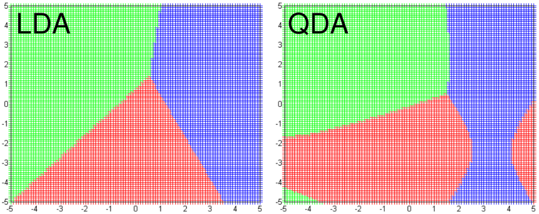
Decision frontiers (computed on learning data)

Examples

3 2D normal laws with different covariance matrices



Three classes in R^2

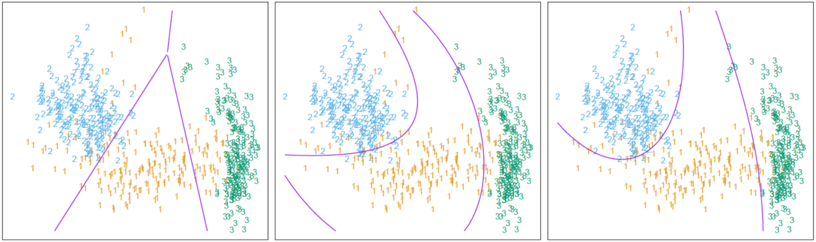


Decision frontiers (computed on learning data)

Examples

[Hasti2001], The Elements of Statistical Learning, Springer.

$K = 3$ classes, with a Gaussian Mixture Model distribution, $p = 2$.



LDA on

$$X = \begin{pmatrix} X_1 & X_2 \end{pmatrix}^t$$

LDA on

$$X' = \begin{pmatrix} X_1 & X_2 & X_1X_2 & X_1^2 & X_2^2 \end{pmatrix}^t$$

QDA on

$$X = \begin{pmatrix} X_1 & X_2 \end{pmatrix}^t$$

Conclusion on the Bayesian Classification

Generative model

A prototype is created for each class

A boundary between classes can then be calculated

Different possible modeling

Using different distributions, depending on our knowledge of the problem

- Normal distribution, multinomial distribution, etc.
- Or more complex models (such as mixture of Gaussians)

Reference classifier

Often used as a reference classifier for its simplicity

No hyperparameters to adjust, linear time learning

The k nearest neighbor method (k-NN

Principle

For $\mathbf{x}$ a data to classify, the idea is to examine the k individuals closest to x in the training set and to choose, for the decision, the most represented class.

$$\hat{G}(\mathbf{x}) = \underset{g \in G}{\text{argmax}}\; Card\{\mathbf{y} \in N_k(\mathbf{x}) | G(\mathbf{y}) = g\}$$

$N_k(\mathbf{x})$ is the "neighborhood" of $\mathbf{x}$ consisting of its k closest neighbors.

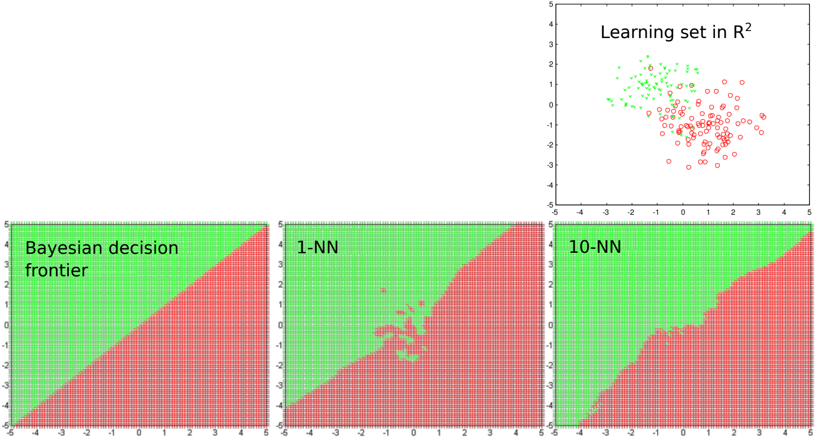
Remarks

- If $k = 1$, the point $\mathbf{x}$ is simply assigned the class of its closest neighbor.
- No assumption is made about the nature of the classes or the type of separators: it is a non-parametric method.
- There is no proper learning involved.
- The k-NN method uses the notion of neighborhood which itself implies a notion of proximity and therefore the use of a metric.

The k nearest neighbor method

Example

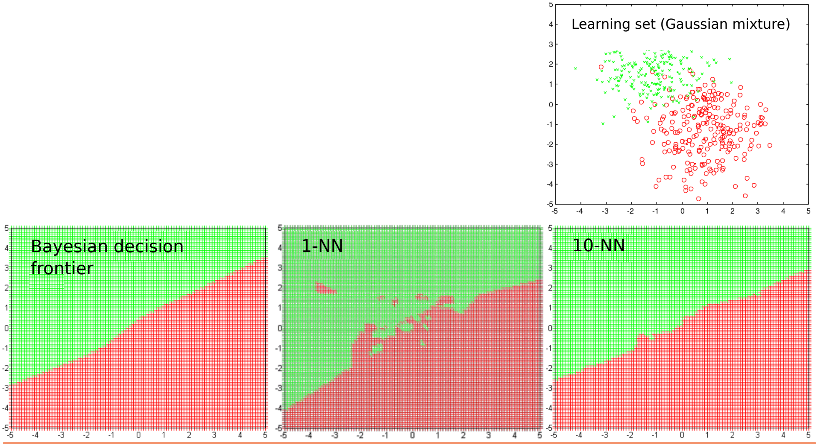
2 normal laws in 2 dimension



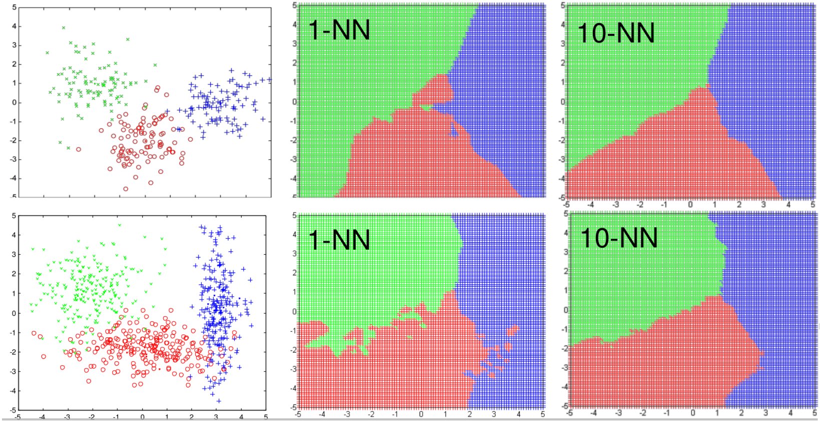
The k nearest neighbor method

Example

2 normal mixture laws in 2 dimension



The k nearest neighbor method: other examples



Support vector machine *(SVM, Vapnik et al., 1995)*

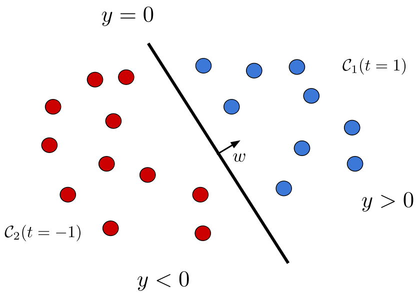
Context

Binary classifier: for each point x_i, we have $t_i = 1$ or $t_i = -1$

Principle

We search for an optimal hyperplane of the form: $y(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0 = 0$

Maximizing the size of the margin between two classes, that is, the region where the boundary can be orthogonally moved without causing misclassification.



Support vector machine *(SVM, Vapnik et al., 1995)*

Context

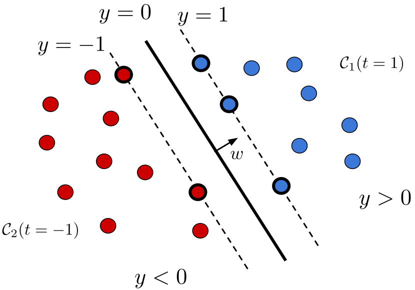
Binary classifier: for each point x_i, we have $t_i = 1$ or $t_i = -1$

Principle

We search for an optimal hyperplane of the form: $y(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0 = 0$

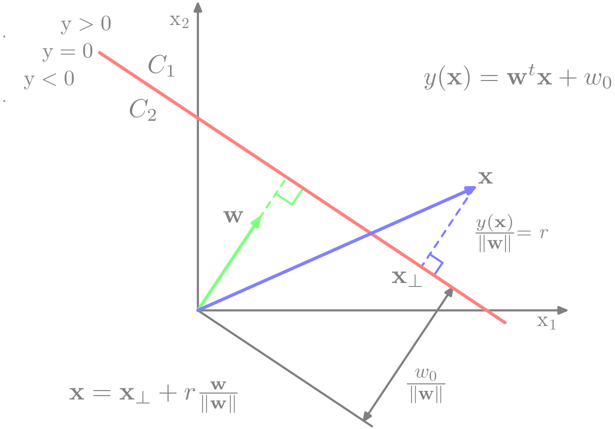
Maximizing the size of the margin between two classes, that is, the region where the boundary can be orthogonally moved without causing misclassification.

The support vectors are the points on which the margins are based.



Support vector machine

Projection system

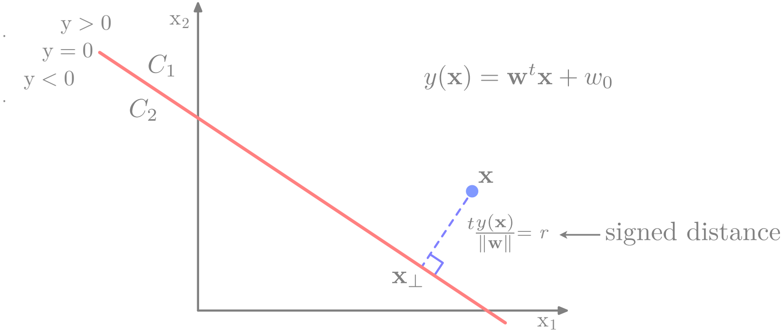


$$y(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0$$

$$\mathbf{x} = \mathbf{x}_\perp + r\,\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

Support vector machine

Projection system



$$y(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0$$

$\frac{t\,y(\mathbf{x})}{\|\mathbf{w}\|} = r$ ⟵ signed distance

$$\mathbf{x} = \mathbf{x}_\perp + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

Support vector machine

The margin

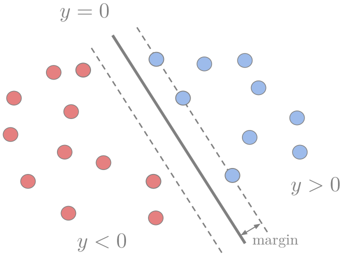
The shortest signed distance between the decision surface and the learning data.

The signed distance for a set $(\mathbf{x}_i, t_i)$ is:

$$\frac{t_i y(\mathbf{x}_i)}{\|\mathbf{w}\|} = \frac{t_i(\mathbf{w}^t\mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

Maximizing the margin implies to solve:

$$\operatorname*{argmax}_{\mathbf{w},b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_i [t_i(\mathbf{w}^t\mathbf{x}_i + b)] \right\}$$



Support vector machine

The margin

The shortest signed distance between the decision surface and the learning data.

The signed distance for a set $(\mathbf{x}_i, t_i)$ is:

$$\frac{t_i y(\mathbf{x}_i)}{\|\mathbf{w}\|} = \frac{t_i(\mathbf{w}^t\mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

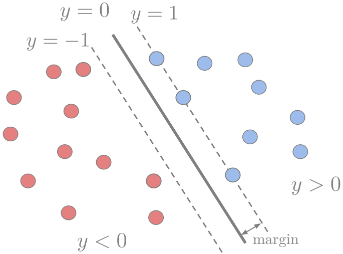
Maximizing the margin implies to solve:

$$\underset{\mathbf{w},b}{\text{argmax}} \left\{ \frac{1}{\|\mathbf{w}\|} \min_i [t_i(\mathbf{w}^t\mathbf{x}_i + b)] \right\}$$

The margin is the same if we multiply $\mathbf{w}$ and b by a constant, so we can set as constraint:

$$t_i(\mathbf{w}^t\mathbf{x}_i + b) = 1$$

for the closest point $(\mathbf{x}_i, t_i)$ of the decision surface.



Support vector machine

The margin

The shortest signed distance between the decision surface and the learning data.

The signed distance for a set $(\mathbf{x}_i, t_i)$ is:

$$\frac{t_i y(\mathbf{x}_i)}{\|\mathbf{w}\|} = \frac{t_i(\mathbf{w}^t \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

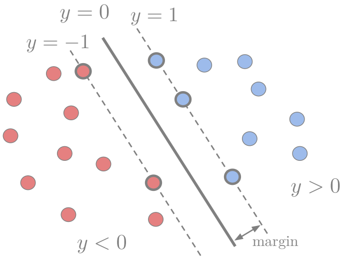
Maximizing the margin implies to solve:

$$\underset{\mathbf{w},b}{\text{argmax}} \left\{ \frac{1}{\|\mathbf{w}\|} \min_i [t_i(\mathbf{w}^t \mathbf{x}_i + b)] \right\}$$

The margin is the same if we multiply $\mathbf{w}$ and b by a constant, so we can set as constraint:

$$t_i(\mathbf{w}^t \mathbf{x}_i + b) = 1$$

for the closest point $(\mathbf{x}_i, t_i)$ of the decision surface. → Optimal margin



Support vector machine

Resolution

if the learning set is linearly separable, with the previous constraint:

$$\underset{\mathbf{w},b}{\text{argmax}}\left\{\frac{1}{\|\mathbf{w}\|}\min_i[t_i(\mathbf{w}^t\mathbf{x}_i+b)]\right\} \quad\rightarrow\quad \underset{\mathbf{w},b}{\text{argmin}}\left\{\frac{1}{2}\|\mathbf{w}\|^2\right\}$$

with $\quad t_i(\mathbf{w}^t\mathbf{x}_i+b)\geqslant 1 \quad \forall i=1,...,n$

Quadratic optimization problem. Solved using Lagrange multipliers:

$$L(\mathbf{w},b,\mathbf{a})=\frac{1}{2}\|\mathbf{w}\|^2-\sum_{i=1}^{n}a_i\left(t_i(\mathbf{w}^t\mathbf{x_i}+b)-1\right)$$

By cancelling the derivatives, we get:

$$\mathbf{w}=\sum_{i=1}^{n}a_it_i\mathbf{x_i} \qquad \sum_{i=1}^{n}a_it_i=0$$

With more work and using the Karush-Kuhn-Tucker conditions, we finally get:

$$a_i(t_iy(\mathbf{x}_i)-1)=0 \quad\Rightarrow\quad \forall i=1,...,n \quad a_i=0 \quad \text{or} \quad t_iy(\mathbf{x}_i)=1$$

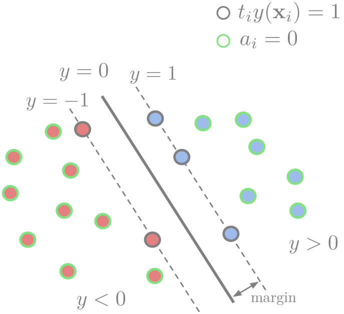
→ The $\mathbf{x}_i$ such as $a_i>0$ are called support vectors.

Support vector machine

Prediction using the support vectors

Only the support vectors are considered:

$$
\begin{aligned}
y(\mathbf{x}) &= \mathbf{w}^t\mathbf{x} + b \\
&= \left(\sum_{i=1}^{n} a_i t_i \mathbf{x}_i\right)^t \mathbf{x} + b \\
&= \sum_{i=1}^{n} a_i t_i \mathbf{x}_i^t \mathbf{x} + b
\end{aligned}
$$



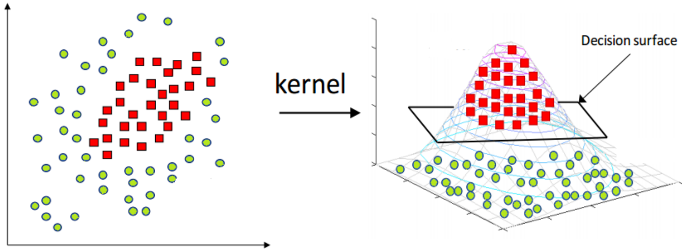
Support vector machine

Kernel trick

Implicit projection of the data into a higher dimensional space.

Addition of a kernel to compute distances: polynomial kernel, Gaussian kernel.

$$y(\mathbf{x}) = \mathbf{w}^t \phi(\mathbf{x}) + b$$



→ Allows to learn non-linear boundaries between classes

Support vector machine

The non separable case

Definition of a "soft" margin that allows misclassified samples.

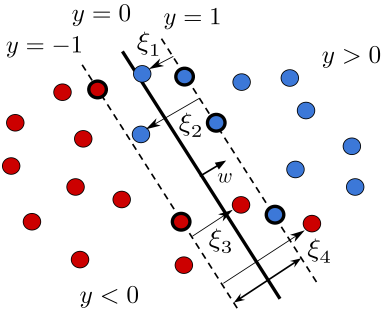
The classification errors $\xi_i > 0$ are such that $\sum_{i=1}^{n} \xi_i \leqslant cste$.

Now we have to solve:

$$\min_{\mathbf{w},b} \|\mathbf{w}\| + C \sum_{i=1}^{n} \xi_i$$

under the constraints:

$$t_i(\mathbf{w}^t \mathbf{x}_i + b) \geqslant 1 - \xi_i \quad \xi_i \geqslant 0$$



Resolution with the same process.

The samples with $\xi_i > 0$ are also support vectors.

Hyperparameter $C > 0$

Set the trade-off between margin and errors:

- High C: high penalties, thin margin
- Low C: low penalties, large margin

Support vector machine

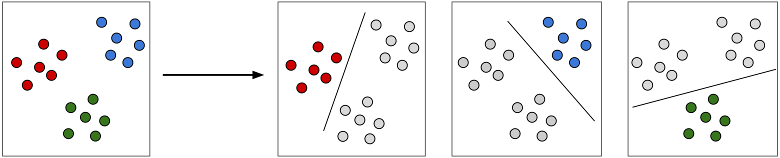
Multi-classes SVM

SVM is a binary classifier.

What if the dataset contains $K > 2$ different classes?

Different strategies are possible:

- **One-vs-all** (a SVM for each class)
- One-vs-one (a SVM for each pair of classes)



Conclusion

- The "**Ugly Duckling Theorem**" says there is no set of characteristics better than another for all the problems (or in the absence of a priori on the nature of the issue)

- The "**No Free Lunch Theorem**" says that in the absence of a priori information on the problem to be treated, there is no learning algorithm that is objectively superior to another.

  → There is simply no universally best algorithm.
  It is necessary to know the problem

- And what about **image** dedicated classification methods?

Bags of visual words

Classification by Bags of visual words (or bags of features)

- Approach inspired by bags of words for textual indexing
- Efficient for image-level classification

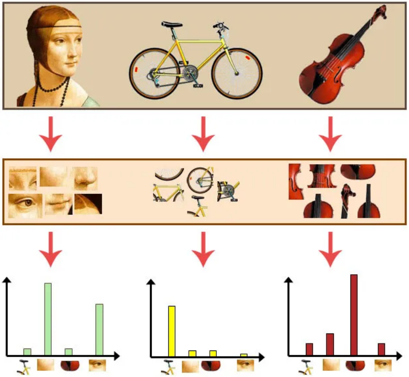
Method:

1)- Extract features on the learning set (keypoints, regular grid, ...)

2)- Learn a vocabulary of "visual words" on the learning set

3)- Describe each image by the histogram of its visual words

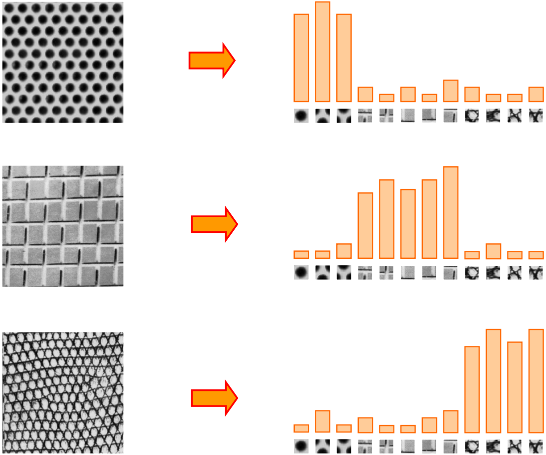
4)- Classify the image from this histogram, for example with SVM



State-of-the-art until 2012 (then Deep Learning)

Bags of visual words

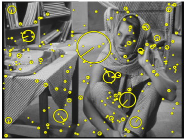
Example for texture clustering

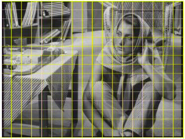


1) Extract features

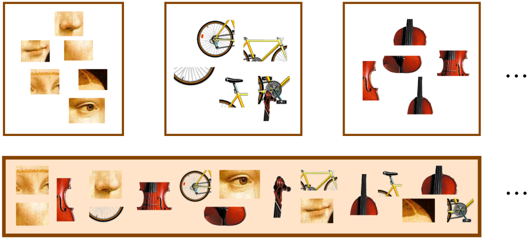
Extract features for each image of the learning set:

- At Keypoints (SIFT, SURF, ...)
- On a regular grid (block-wise) (intensity, LBP, HOG, ...)
- Random sampling ...





→ All the extracted features form the set of visual words

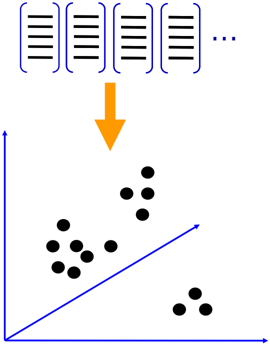


Regions of extracted features

Set of visual words

2) Learn the visual vocabulary

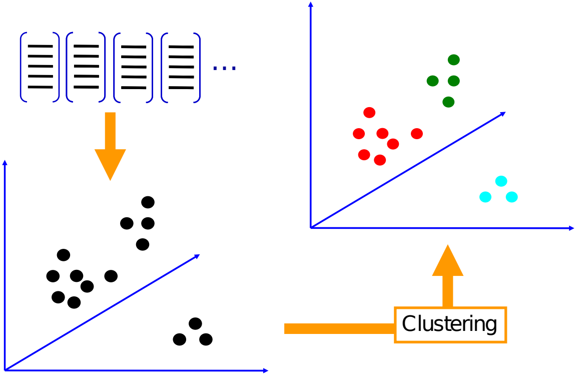
Use K-means to cluster the set of visual words (described by p variables)



→ Each cluster center corresponds to a visual word in the vocabulary

2) Learn the visual vocabulary

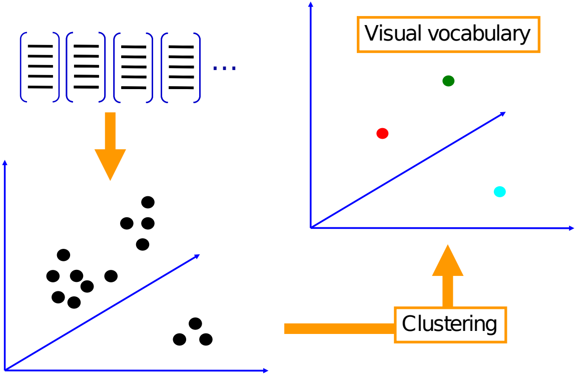
Use K-means to cluster the set of visual words (described by p variables)



→ Each cluster center corresponds to a visual word in the vocabulary

2) Learn the visual vocabulary

Use K-means to cluster the set of visual words (described by p variables)



→ Each cluster center corresponds to a visual word in the vocabulary

2) Learn the visual vocabulary

- Clustering is common for learning a visual vocabulary or codebook

  - Unsupervised learning process

  - Each cluster center produced by k-means becomes a codevector

  - "Universal" codebook if the training set is sufficiently representative

- The codebook is used for quantizing features

  - A codevector (visual word) quantizer takes a feature vector and maps it to the index of the nearest codevector in a codebook

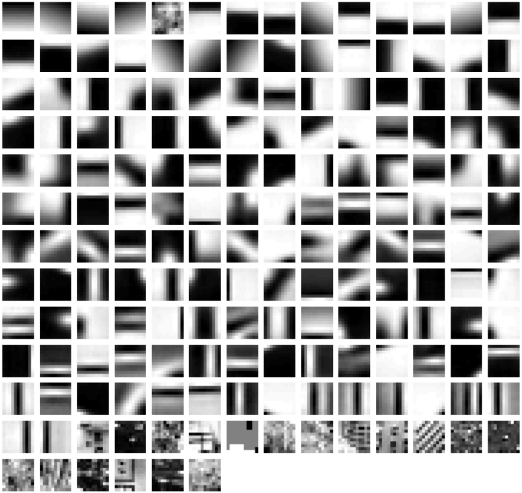
- How to choose vocabulary size?

  - Too small: visual words not representative of all patches

  - Too large: quantization artifacts, overfitting

2) Learn the visual vocabulary

Example of vocabulary

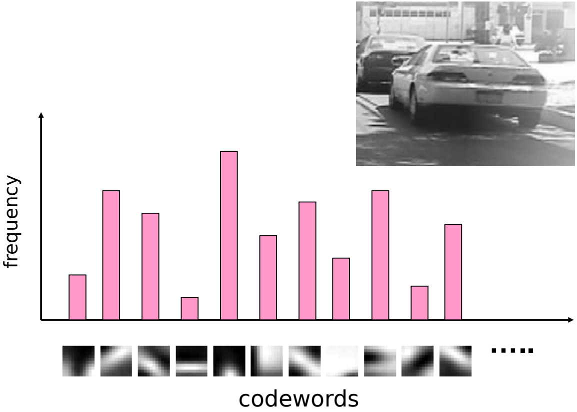


3) Image representation

For a new image:

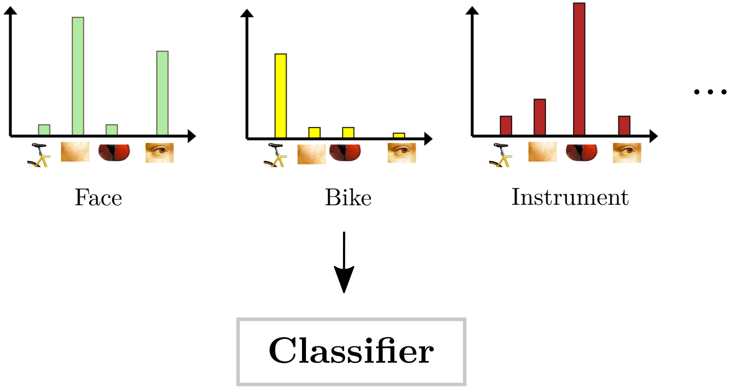
- Extract features
- Build a histogram of codeword frequencies:

  For each feature, increment the bin of the closest visual word



4) Classification

The histograms of the learning set can be fed to a classifier (e.g. SVM)



Face

Bike

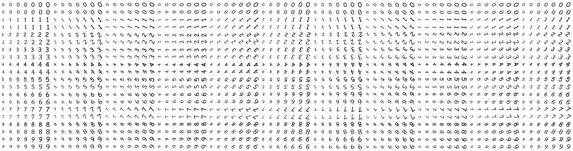
Instrument

. . .

Classifier

Practical nº2: Supervised Classification - Digit Recognition

- **Data:** Images of digits with different fonts, rotations, scales



- **Algorithms of supervised classification**
  - Linear discriminant and quadratic analysis
    Implementation, computation of performance with the confusion matrix, accuracy rate and Kappa coefficient
  - K nearest neighbor method
    Implementation, for K=1, then K=N
  - Support vector machine classifier
- **Choice of the parameters**
  - Measure of the influence of features and the size of the learning set

References

- Hastie, Tibshirani, Friedman (2001). The elements of Statistical Learning, Springer.
- Arthur, David et Vassilvitskii, Sergei (2007). K-moyennes$++$: the advantages of careful seeding, ACM-SIAM symposium on Discrete algorithms.
- Duda, Hart, Stork (2000). Pattern Classification, 2nd Edition, Wiley.
- Flach (2007). Machine Learning: The Art and Science of Algorithms that Make Sense of Data, Cambridge University Press.
- Theodoridis, Koutroumbas (2008). Pattern Recognition, Fourth Edition, Academic Press, Elsevier.
- Theodoridis, Pikrakis, Cavouras, Koutroumbas (2010). Introduction to Pattern Recognition: A Matlab Approach, Academic Press, Elsevier.
- Geoff Dougherty (2012). Pattern Recognition and Classification: An Introduction, Springer.

References

Slides inspired from:

- Marc Donias: https://donias.vvv.enseirb-matmeca.fr/ts326.html
- Michaël Clément: https://www.labri.fr/perso/mclement/
- Florent Grélard: https://fgrelard.github.io/#teaching
- Michel Crucianu: http://cedric.cnam.fr/~crucianm/rfmn.html
- Vincent Nozick: https://igm.univ-mlv.fr/~vnozick/teaching/slides/imac2_math/10_pca.pdf
- Serena Yeung: https://ai.stanford.edu/~syyeung/cvweb/tutorials.html
- Victor Powell: https://setosa.io/ev/principal-component-analysis/
- Mrinal Tyagi: https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f
- Shashmi Karanam: https://towardsdatascience.com/curse-of-dimensionality-a-curse-to-machine-learning-c122ee33bfeb
- Wikipédia: https://fr.wikipedia.org/wiki/Motif_binaire_local
- Datahacker: https://datahacker.rs/opencv-circle-detection-hough-transform/
- Niebles & Krishna: http://vision.stanford.edu/teaching/cs131_fall1718/files/14_BoW_bayes.pdf
- Hugo Larochelle: https://info.usherbrooke.ca/hlarochelle/cours/ift603_H2015/description.html