

# VIETNAMESE SPRING SCHOOL in STATISTICS AND MACHINE LEARNING

## CHAPTER II. Supervised classification

Agnès LAGNOUX

[lagnox@univ-tlse2.fr](mailto:lagnox@univ-tlse2.fr)

# LECTURE SUPPORTS

All the files of this lecture are available on my webpage:

`https://perso.math.univ-toulouse.fr/lagnoux/enseignements/`

You will find them at the **bottom** of the page.

# LECTURE OUTLINE

- ① Introduction
- ② Supervised classification
  - Linear regression
  - $k$ -nearest neighbors
  - Discriminant factor analysis
  - Naive Bayesian
  - Logistic regression
- ③ Unsupervised classification
  - Hierarchical clustering analysis
  - $k$ -means

## Introduction to supervised learning

- Predictive performance evaluation

- Database slicing: learning, validation, testing

## Linear regression

## The $k$ -nearest neighbors

## Model-based approaches

- Discriminant analysis

- Naive Bayesian

- Logistic regression

## Other supervised classification algorithms

# Plan

## Introduction to supervised learning

- Predictive performance evaluation

- Database slicing: learning, validation, testing

## Linear regression

## The $k$ -nearest neighbors

## Model-based approaches

- Discriminant analysis

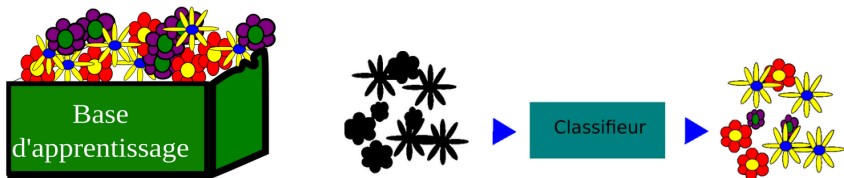
- Naive Bayesian

- Logistic regression

## Other supervised classification algorithms

## Introduction to supervised learning

*The objective of supervised classification is the algorithmic categorization of objects. It consists in assigning a class or category to each object (or individual) to be classified, based on statistical data.*



## Introduction to supervised learning

In this context, we are interested in a vector input:

$X = (X_1, \dots, X_p) \in \mathbb{R}^p$  and an output  $Y$  (usually real).

We observe  $n$  individuals described by their values of  $X$  and  $Y$ .

We therefore have the following data.

$$X_n = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} \quad \text{et} \quad Y_n = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}.$$

*We want to predict the class  $y_0$  of a new input*

$x_0 = (x_{01}, x_{02}, \dots, x_{0p})$ .

## Predictive performance - Confusion matrix

The confusion matrix counts the occurrences of predictions according to the true values.

		Predicted class			
		$\ell = 1$	$\ell = 2$	$\dots$	$\ell = K$
True class	$k = 1$	$n_{11}$	$n_{12}$	$\dots$	$n_{1K}$
	$k = 2$	$n_{21}$	$n_{22}$	$\dots$	$n_{2K}$
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	$k = K$	$n_{K1}$	$n_{K2}$	$\dots$	$n_{KK}$

where  $n_{k\ell}$  is the number of observations of class  $k$  predicted in the class  $\ell$  :

$$n_{k\ell} = \sum_{i=1}^n \mathbb{1}_{\{Y_i=k, \hat{Y}_i=\ell\}}.$$



## Predictive performance - Empirical risk

The **empirical risk** (average cost of misclassification) of the  $g$  classification rule is

$$R(g) = \frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K \sum_{\ell=1}^K C_{k\ell} \mathbb{1}_{\{Y_i=k, \hat{Y}_i=\ell\}} = \frac{1}{n} \sum_{k=1}^K \sum_{\ell=1}^K C_{k\ell} n_{k\ell}.$$

In the case of a 0-1 cost, we find the **empirical error rate**

$$R(g) = \frac{1}{n} \sum_{k=1}^K \sum_{\ell=1, \ell \neq k}^K n_{k\ell}.$$

## Predictive performances in binary classification

The variable to be explained has two modalities, for example {1,2}.

**Vocabulary:** “predict 1”=positive and “predict 2”=negative (e.g.).

The **confusion matrix** is then :

		Prediction	
		Positive	Negative
Truth	Positive P	True positive=TP Correct detection OK	False negative=FN Non detection Underestimation
	Negative N	False positive=FP False alarm Overestimation	True negative=TN Correct rejection OK

## Predictive performances in binary classification

**True positives** are the class 1 individuals that are correctly classified as 1.

**False negatives** are the class 1 individuals that are incorrectly classified as 2.

**False positives** are class 2 individuals that are incorrectly classified as 1.

**True negatives** are the class 2 individuals that are correctly classified as 2.

One can then calculate the percentages rows and columns.

## Predictive performances in binary classification

		Prediction			
		Positive	Negative		
True P	True	True positive Correct detection TP	False neg=FN Non detection Underestimation	True positive rate TPR $\frac{TP}{P}$	False negative rate FNR $\frac{FN}{P}$
	False	False pos=FP False alarm Overestimation	True negative Correct rejection TN	False positive rate FPR $\frac{FP}{N}$	True negative rate TNR $\frac{TN}{N}$
		Positive predictive value $PPV = \frac{TP}{TP+FP}$	False omission rate $FOR = \frac{FN}{TN+FN}$	Positive Likelihood R $LR+ = \frac{TPR}{FPR}$	Negative Likelihood R $LR- = \frac{FNR}{TNR}$
		False discovery rate $FDR = \frac{FP}{TP+FP}$	Negative predictive value $NPV = \frac{TN}{TN+FN}$	Markedness $PPV + NPV - 1$	Diagnostic odds ratio $DOR = \frac{LR+}{LR-}$

## Predictive performances - True positive rate

The **true positive rate** (TPR) is calculated using the following equation:

$$TPR = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} = \frac{TP}{TP + FN} = \frac{TP}{P}.$$

The true positive rate is equal to the number of true positives divided by the number of true positives plus the number of false negatives.

This is a percentage line.

True positive rate (TPR) is also called **sensitivity**, **recall**.

## Predictive performances - False positive rate

The **false positive rate** (FPR) is obtained using the following equation:

$$FPR = \frac{\text{false positives}}{\text{false positives} + \text{true negatives}} = \frac{FP}{FP + TN} = \frac{FP}{N}.$$

The false positive rate is equal to the number of false positives divided by the number of false positives plus the number of true negatives.

This is a percentage line.

The false positive rate (FPR) corresponds to **1 - specificity**.

## Predictive performances - Predictive Value

The **predictive value** (PPV) is obtained by using the following equation:

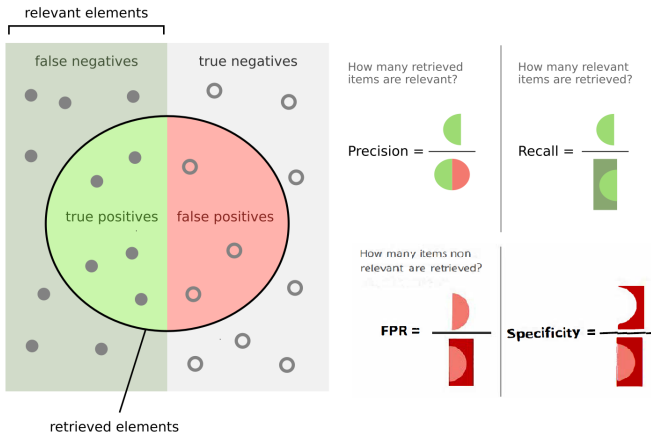
$$PPV = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} = \frac{TP}{TP + FP}.$$

The predictive value is equal to the number of true positives divided by the number of true positives plus false positives. This is the proportion of predictions 1 that are actually class 1 entries. This proportion compares to the **prevalence** which is the proportion of class 1 in the data.

This is a percentage column.

The predictive value (PPV) is also called **precision**.

# Predictive performances: precision/recall/specificity



Source : <https://en.wikipedia.org/wiki/F-score>



## Predictive performances - $F_1$ -score

In binary classification, the  $F_1$ -score depends on

- the **positive predictive value** (PPV), also called **precision**,
- the **true positive rate** (TPR), also called **recall** or **sensitivity**

as follows:

$$F_1 = \frac{2PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}.$$

In other words, it is the **harmonic mean** of precision and recall. To obtain  $F_1$ , we must therefore calculate the true positive (TP), false positive (FP), and false negative (FN) rates of the model on a dataset.

It measures the classification rule's ability to correctly predict class 1 entries and not predict 1 of the class 2 entries.

## Predictive performances - $F_1$ -score

In the case where the predictions are no longer binary (multi-class), the F-measure is calculated by making the **average of  $F_1$  scores for each class**. This average can be done in different ways:

- the “micro” approach where the TP, FP and FN rates of each class are simply added to calculate the F-measure;
- the “macro” approach where the F-measure is the arithmetic mean of the  $F_1$ -score for each class.

## Predictive performances - Kappa de Cohen

In statistics, the *kappa method* (kappa) measures agreement between observers during qualitative coding into categories. The calculation of *kappa* is done as follows:

$$\kappa = \frac{P_{\text{agreement}} - P_{\text{random}}}{1 - P_{\text{random}}},$$

where  $P_{\text{agreement}}$  is the proportion of agreement between coders and  $P_{\text{random}}$  is the probability of a random agreement.

- If the coders are in total agreement,  $\kappa = 1$ .
- If they are totally in disagreement (or agree due to random),  $\kappa \leq 0$ .

## Predictive performances - Kappa de Cohen

Marc and Simon are responsible for defining who will be accepted or not at the final exam in a group of 50 students. Each of them checks the copy of each student and notes received or not (YES or NO):

Simon \ Marc	YES	NO
	YES	NO
YES	$a=20$	$b=5$
NO	$c=10$	$d=15$

The agreement between evaluators is:

$$P_{\text{agreement}} = \frac{a + d}{a + b + c + d} = \frac{20 + 15}{50} = 0.7.$$

To calculate the probability of agreement “at random”, we note:

- Simon scored YES to 25 students, or 50% of the cases.
- Marc scored YES in 60%, 30 out of 50 students.

## Predictive performances - Kappa de Cohen

The probability that both teachers evaluate YES is

$$P_{YES} = \frac{a+b}{a+b+c+d} \times \frac{a+c}{a+b+c+d} = 0.5 \times 0.6 = 0.3$$

Analogously, the probability that both teachers evaluate NO is:

$$P_{NO} = \frac{c+d}{a+b+c+d} \times \frac{b+d}{a+b+c+d} = 0.5 \times 0.4 = 0.2$$

The global probability that the teachers agree is:

$$P_{random} = P_{YES} + P_{NO} = 0.3 + 0.2 = 0.5$$

Kappa's formula then gives:

$$\kappa = \frac{P_{agreement} - P_{random}}{1 - P_{random}} = \frac{0.7 - 0.5}{1 - 0.5} = 0.4$$

## Predictive performances

Other criteria in binary classification:

- Prevalence =  $\frac{P}{P+N}$ ,
- Accuracy =  $\frac{TP+TN}{P+N}$ ,
- Balanced-accuracy =  $\frac{TPR+TNR}{2}$ ,
- Fowlkes-Mallows index (FM) =  $\sqrt{PPV \times TPR}$ ,
- Matthews correlation coefficient (MCC)  
=  $\sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$ ,
- Threat score (TS), critical success index (CSI), Jaccard index  
=  $\frac{TP}{TP+FN+FP}$ ,
- Informedness (BM) =  $TPR + TNR - 1$ ,
- Prevalence threshold (PT) =  $\frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$ .

## Predictive performances - ROC curve

### What is the ROC curve?

The ROC curve, or receiver operating characteristic curve, is a graphical representation used to evaluate the performance of a binary classification model.

It illustrates the trade-off between **sensitivity** (TPR) and **specificity** (1 - FPR) on different threshold parameters.

The ROC curve is particularly useful for determining the extent to which a model distinguishes between two classes, making it an essential tool in statistics, data analysis and data science.

## Predictive performances - ROC curve

Understanding the rate of true positives and the rate of false positives

The ROC curve plots TPR versus FPR at different threshold levels, providing a comprehensive view of model performance.

Interpreting the ROC curve

The shape of the ROC curve gives an overview of the efficiency of a classification model.

A curve that slopes towards the upper left corner indicates a model with high sensitivity and specificity, while a curve closer to the diagonal line suggests a model with low discriminative power.



## Predictive performances - AUC

The **area under the ROC curve** (AUC) is an essential measure of model performance. It is calculated by integrating the area under the ROC curve, providing a single scalar value that summarizes the model's ability to distinguish between classes. An AUC

- of 0.5 suggests **absence of discriminating ability**,
- from 0.7 to 0.8 is considered as **acceptable**,
- greater than 0.8 indicates **good performance**,
- greater than 0.9 suggests **excellent performance**,
- of 1 indicates **perfect classification**.

## Predictive performances - Take home message

A good model is both:

- sensitive and specific. This is measured with the ROC curve and the AUC.
- sensitive and accurate. This is measured with the  $F1$ -measure.

# Database slicing: learning, validation, testing

## Why slicing data?

- Prevent overfitting.
- Test the generalizing capacity of a model on new data.

## A classic scheme

- Training set (70%): to adjust the model.
- Validation set (20%): to adjust the hyperparameters.
- Test set (10%): to evaluate final performance.

Possible problems: small dataset  $\Rightarrow$  statistical bias.

## Cross-validation: principle


**Cross-validation** is a method of estimating model reliability based on a sampling technique.

In  **$K$ -fold cross-validation**, the original sample is divided into  $K$  samples (or blocks), then one of the  $K$  samples is selected as the validation set, while the other  $K - 1$  samples constitute the training set.


**Leave-one-out cross-validation** (LOOCV) is the special case of  $K$  block cross-validation with  $K = n$ . That is, at each learning-validation iteration, learning is performed on  $n - 1$  observations and validation on the single remaining observation.

## Cross-validation

More precisely, for leave-one-out cross-validation:

- For  $i = 1, \dots, n$ ,
  - estimate the rule on the data without the  $i$ -th data,
  - predict this data  $i$  with this rule.
- compute the performance criterion on these  $n$  predictions.  
 a single vector of possible predictions.

for cross-validation  $K$ -folds:

- Split the data into  $K$  sub-samples of the same size (respecting, if possible the proportions of the classes).
- For  $k = 1, \dots, K$ ,
  - estimate the rule on the private data of sample  $k$ ,
  - predict the data of sample  $k$  with this rule.
- Compute the performance criterion on these  $K$  predictions.  
 several vectors of possible predictions.

# Cross-validation

## Evaluate generalized model performance

- Cross-validation is used to assess whether the model trained on a subset of data is capable of predicting correctly on other data (generalization).

## Identify biases linked to overlearning

- Cross-validation helps detect whether performance on training data is significantly better than that on validation data.

## Compare with other models

- When comparing supervised models ( $k$ -NN, LDA/QDA, SVM, decision trees), it is essential to do so with a fair procedure.
- Using cross-validation provides a robust and comparable evaluation.

# Cross-validation

## Manage class imbalances

- If classes are not balanced, sub-samples in a dataset may not reflect the full distribution.
- Cross-validation ensures rotation of observations and alleviates this problem.

## Optimize variable selection (feature selection)

- If you use a prior variable selection step before building your model (e.g., choosing discriminant variables based on their relevance), cross-validation ensures that this selection does not bias the evaluation. Variables should only be selected from the training data at each iteration of cross-validation.

## Importance of data preparation

- Data cleaning: management of missing values, standardization / normalization of variables.
- Exploratory visualization to understand relationships in the data.



# Plan

## Introduction to supervised learning

- Predictive performance evaluation

- Database slicing: learning, validation, testing

## Linear regression

## The $k$ -nearest neighbors

## Model-based approaches

- Discriminant analysis

- Naive Bayesian

- Logistic regression

## Other supervised classification algorithms

## Linear regression: how it works?

The aim of **linear regression** is to express an output variable  $Y$  as a function of an input variable  $X$  in a linear fashion, i.e.  $Y = aX + b$ .

This model therefore has two parameters  $a$  and  $b$ , whose **optimal values** must be found during the **learning phase**. Several techniques exist for estimating these parameters, the most widely used being

- the **least square method**,
- the **deviation method**,
- the **maximum likelihood method**.

Linear regression is a **supervised learning algorithm**, with  $N$  input-output pairs making up the data set  $(x_i, y_i)_{i=1, \dots, n}$ . These known data will be used to **estimate the model parameters**.

## Linear regression: how it works?

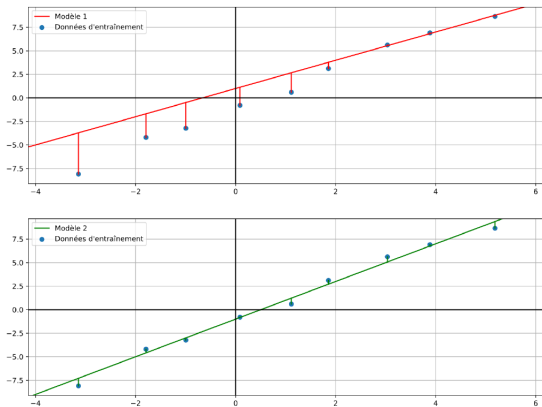
Here, we'll take the example of the least-squares method, which is more widespread than the others. The principle of linear regression algorithms remains similar for the other methods.

In the **case of the least-squares method**, we look for the parameters  $a$  and  $b$  that **minimize a cost function**

$$C = \sum_{i=1}^n (y_i - ax_i - b)^2.$$

This function corresponds to the sum of squared deviations between predictions and expected values. These deviations to be minimized are called **residuals**.

## Linear regression: an example



Two linear regression models. The first model shows large deviations between predicted and expected values, while the second minimizes the squares of these deviations.

## Linear regression: the $r$ coefficient

The **linear regression coefficient** can be calculated to estimate whether two variables can be linearly related. This coefficient is calculated from the standard deviations  $\sigma_X$  and  $\sigma_Y$  of the variables and from the covariance between the input and output variables:

$$r = \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y}.$$

If the coefficient is close in absolute value to 1, this means that the variables can be linearly linked. It is therefore a way of checking that linear regression is a coherent choice of model.

## Linear regression: applications

Linear regression has many applications. For example, you could measure the current flowing through a resistor for different voltage values applied across its terminals. The current measurement may be noisy, making it impossible to directly determine the value of the resistance passed through. Linear regression can be used to estimate the value of this resistance by minimizing the difference between the measured current and the value estimated by the  $U = RI$  model. This same method can be used to estimate parameters in many fields.

In practice, there are many software packages available for performing linear regression. Most spreadsheets have this functionality, and some programming languages offer libraries for performing regression (scikit learn for Python, operator 'for Matlab...).

## Linear regression: limitations

- (1) This method is limited to the estimation of linear models, which severely restricts its possible applications. The principle of linear regression can, however, be extended to polynomial regressions. The latter are more complex to implement, as they require more parameters (and more choice in the number of parameters) and face the problems of over- and under-learning.
- (2) The estimate obtained is sensitive to the measurement noise introduced in the training data. Thus, outlier data (far removed from the model to be estimated) will influence parameter estimates. This influence depends on the linear regression method used: the least squares method considers the square of the residuals, and an outlier will therefore shift the model more than in the case of the least deviations method (which uses the absolute values of the residuals).

## Linear regression: limitations

Some methods allow you to incorporate a priori knowledge, further limiting the effect of outliers.

To reduce the effect of noise on estimation, it is important to have a sufficiently large number of data, which also constrains the experimental protocol.



# Plan

## Introduction to supervised learning

- Predictive performance evaluation

- Database slicing: learning, validation, testing

## Linear regression

## The $k$ -nearest neighbors

## Model-based approaches

- Discriminant analysis

- Naive Bayesian

- Logistic regression

## Other supervised classification algorithms

## The $k$ -nearest neighbors: principle

The  $k$ -nearest neighbors algorithm or  $k$ -NN, like linear regression, is a supervised learning algorithm, so we have the set of  $(x_i, y_i)_{i=1, \dots, n}$  labeled data.

- This is a **non-parametric model** (unlike regression), i.e. the model has no parameters whose value must be optimized.  
⚠  $k$  is not a parameter but a **hyperparameter**.
- In the case of a **classification algorithm**, the outputs  $y_i$  correspond to the possible classes (and not continuous values, as in regression). The input  $x_i$  is a vector of dimension  $p$ , containing the different variables associated with an input, an individual.

## The $k$ -nearest neighbors: principle

For each new input  $x_0$ , we measure the distance between  $x_0$  and the inputs  $x_i$  for all  $i$  in  $\{1, \dots, n\}$ .

We then select the  $k$  elements closest to the input.

The class predicted by the algorithm then corresponds to the majority class, i.e. the most frequent class in the  $k$ -NNs selected.

Different distance measures (Euclidean, Manhattan...) can be used for this algorithm, depending on the problem studied.

## The $k$ -nearest neighbors: an example

A famous and commonly used example of  $k$ -NNs concerns Fisher's iris database. In 1936, botanist Edgar Anderson measured 150 iris specimens of three different varieties: Setosa iris, Versicolor iris and Virginica iris. For each specimen, he measured the length and width of the petals and sepals.



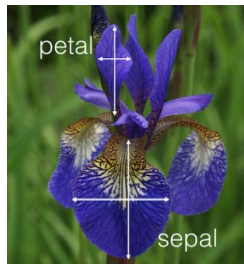
setosa



versicolor



virginica



## The $k$ -nearest neighbors: an example

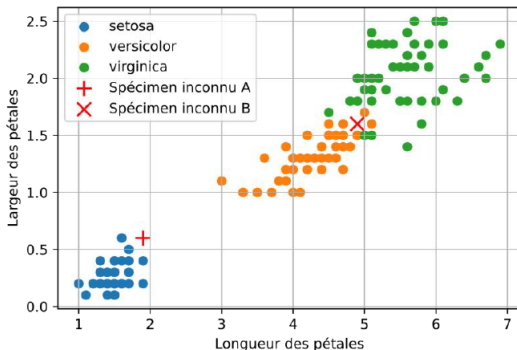
For the purposes of this example, we're only interested in the length and width of the petals.

The input  $x_i$  is therefore a vector of dimension 2 containing the length and width of the petal of individual number  $i$ .

The output  $y_i$  is an integer: 0, 1 or 2, corresponding to the specimen class (setosa, versicolor or virginica) of the individual number  $i$ .

The 150 specimens are divided into 3 varieties and positioned according to petal length and width.

## The $k$ -nearest neighbors: an example



Specimens are grouped according to variety. Indeed, specimens of the same variety have similar characteristics. It therefore seems coherent to predict whether a specimen belongs to a variety based on other specimens in the vicinity (in the sense of Euclidean distance).

## The $k$ -nearest neighbors: an example

Let's now consider two unknown specimens A and B and try to predict the variety to which they belong.

- The first specimen, called A, is represented by a '+' on the graph. It is an iris with petals 1.9 cm long and 0.6 cm wide. This iris has characteristics closer to those of setosa iris specimens than to other specimens. Specimen A can therefore be classified in the setosa variety.
- The second specimen, called B, has petals 4.9 cm long and 1.6 cm wide and is represented by an 'x' on the graph. We can then look at the 5 known iris specimens closest to specimen B. These include 1 iris virginica and 4 iris versicolor. Specimen B is therefore classified in the versicolor variety.

## The $k$ -nearest neighbors: an example

We were therefore able to classify two unknown specimens using the  $k$ -NN method.

Note that we have used the algorithm here for classification purposes, but it can be **extended to regression**. In this case, the elements are not associated with a class but with a value. The output of the algorithm then corresponds to the average value of the  $k$ -NN.



## The $k$ -nearest neighbors: applications

The  $k$ -NN algorithm is frequently used in both classification and regression. It can also be used in shape recognition, with inputs containing the circumference, area or contour signature of the shape, and outputs corresponding to the various possible shapes.

This algorithm has the advantage of being relatively robust if sufficient training examples are provided.

It can also be easily implemented in various programming languages.

## The $k$ -nearest neighbors: limitations

⚠ The value of  $k$  has a strong influence on prediction!

**$k$  too small:** elements out of the ordinary will more easily influence the prediction. The generalization of the model for new elements will therefore be less good. This is the **overfitting** problem.

For example, if  $k = 1$ , a versicolor iris with abnormally small petals will be taken into account by the model, which runs the risk of misclassifying a setosa iris with dimensions close to this outlier.

**$k$  too large:** the model will take into account data that are too far apart, and the majority class will be predicted too often. This is called **underfitting**: the model uses not enough the training data.

For example, if  $k = N$ , all irises will be taken into account and the predicted class will be the same regardless of petal size.

## The $k$ -nearest neighbors: applications and limitations

Choosing the value of  $k$  isn't easy: it's not enough to take it large or small. Different methods exist to get an idea of the possible value of  $k$ .

It's also important to **test the algorithm** on a set of known data to check the quality of its predictions. We can then separate the known data set into two sets, the **training set** and the **test set**. The former will be used to make the prediction, while the latter will be used to check the algorithm's performance and avoid over- and under-learning.

## The $k$ -nearest neighbors: applications and limitations

One of the main problems encountered when implementing the  $k$ -NN algorithm is the **dimension curse**. Indeed, for the algorithm to work optimally, a sufficient number of training data is required so that the points studied are always close to known examples.

When studying higher-dimensional problems, it is therefore essential to have a **large amount of training data**. For this reason, the  $k$ -NN algorithm quickly becomes unusable: beyond 4 or 5 dimensions, the number of data required becomes too large.

# Plan

Introduction to supervised learning

Predictive performance evaluation

Database slicing: learning, validation, testing

Linear regression

The  $k$ -nearest neighbors

Model-based approaches

Discriminant analysis

Naive Bayesian

Logistic regression

Other supervised classification algorithms

# Introduction to discriminant analysis

**Discriminant factor analysis** or simply **discriminant analysis** (DA) is a statistical technique that aims to

- describe,
- explain,
- predict

the membership to predefined groups (classes, modalities of the variable to be predicted...) of a **set of observations** (individuals, examples...) from a series of **predictive variables** (descriptors, exogenous variables...).

## Introduction to discriminant analysis - Applications

- In **medecine**, to detect high-risk cardiac groups based on characteristics such as diet, smoking or not, family history, ...
- In **banking**, to assess the reliability of credit applicants based on their income, the number of people they support, the outstanding credits they holds, ...
- In **biology**, to assign an object to its family based on its physical characteristics. See Fisher's irises.
- In **computer science**, for optical character recognition based on simple information, such as the presence or absence of symmetry, the number of extremities, ...

## Introduction to descriptive discriminant analysis

**Discriminant analysis** can be a **descriptive** technique. The objective is to propose a new representation system, i.e. latent variables formed from linear combinations of predictive variables, which make it possible to discern groups of individuals as much as possible.

In this sense, it is similar to factor analysis because it makes it possible to propose a graphical representation in a reduced space and more particularly to **principal component analysis** (PCA) calculated on the conditional gravity centers of point clouds with a particular metric. We also speak of **canonical discriminant analysis**, particularly in Anglo-Saxon software.



## Introduction to predictive discriminant analysis

**Discriminant analysis** can be a **predictive** technique. In this case, it involves building a **classification function** (**assignment rule**) that predicts the group to which an individual belongs based on the values taken by the predictive variables.

In this sense, this technique is similar to supervised techniques in **machine learning** such as **k-nn**, **decision trees**, **neural networks**, ...

It is based on a **probabilistic framework**. The best known is certainly the multinormal distribution hypothesis (normal law). With the homoscedasticity hypothesis (the point clouds have the same shape), we obtain **linear discriminant analysis** (LDA): very attractive in practice because the classification function is expressed as a linear combination of the predictive variables, easy to analyze and interpret.

# Introduction to predictive discriminant analysis

This technique is, with **logistic regression**, widely used in "scoring", when we want for example to characterize the appetite - the propensity to buy - of a customer facing a new product.

⚠ In this course, we are interested in **predictive discriminant analysis**.

## Principle of discriminant analysis

We have a sample of  $n$  observations distributed in  $K$  groups of sizes  $n_k$ . The number of classes  $K$  is fixed in advance.

The inputs are always vectorial:  $X \in \mathbb{R}^p$  and the output defining the groups is discrete:  $Y \in \{1, \dots, K\}$ .

Let  $\mu_k$  denote the centers of gravity of the conditional point clouds and  $\Sigma_k$  their variance-covariance matrix. Let us assume that the conditional distribution of  $X$  to class  $Y$  is parametric and Gaussian:  $(X|Y = k) \sim \mathcal{N}(\mu_k, \Sigma_k)$ . We therefore have

$$f_{[Y=k]}(x) = \frac{1}{(2\pi)^{p/2} |\Sigma_k|^{1/2}} \exp\left\{-\frac{1}{2}(x - \mu_k)^\top \Sigma_k^{-1} (x - \mu_k)\right\}.$$

Unknown parameters:  $\pi_k = \mathbb{P}(Y = k)$ ,  $\mu_k$  and  $\Sigma_k$  for  $k = 1, \dots, K$ .

## Principle of discriminant analysis

Unknown parameters to estimate

$$\theta = (\pi_1, \dots, \pi_K, \mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_K).$$

Log-likelihood of sample  $(x_1, y_1), \dots, (x_n, y_n)$ :

$$\begin{aligned}\ell(\theta) &= \log \prod_{i=1}^n f_{X,Y}(x_i, y_i) = \sum_{i=1}^n \log(\pi_{y_i} f_{[Y=y_i]}(x_i)) \\ &= \sum_{k=1}^K n_k \log(\pi_k) + \sum_{k=1}^K \sum_{i; y_i=k} \log f_{[Y=k]}(x_i).\end{aligned}$$

Maximum likelihood estimators

$$\begin{aligned}\hat{\pi}_k &= \frac{n_k}{n}, \quad \hat{\mu}_k = \frac{1}{n_k} \sum_{i; y_i=k} x_i, \\ \hat{\Sigma}_k &= \frac{1}{n_k} \sum_{i; y_i=k} (x_i - \hat{\mu}_k)^\top (x_i - \hat{\mu}_k).\end{aligned}$$

## Principle of discriminant analysis

The **Bayes classification rule** writes as:

$$g(x) = \arg \max_{k \in \{1, \dots, K\}} \mathbb{P}(Y = k | X = x) \quad (\text{direct approach})$$

$$= \arg \max_{k \in \{1, \dots, K\}} f_{[Y=k]}(x) \mathbb{P}(Y = k) \quad (\text{indirect approach})$$

$$= \arg \max_{k \in \{1, \dots, K\}} (\log f_{[Y=k]}(x) + \log \pi_k).$$

👉 Approaches based on a model then consists in **learning the law of  $Y$  given  $X$**  to then deduce the **classification rule  $g$** .

## Principle of discriminant analysis - remarks

The **direct approach** consists in directly learning the law of  $Y$  given  $X$ . For example, in logistic regression:

$$\mathbb{P}(Y = 1|X = x) = \frac{\exp(x^\top \beta)}{1 + \exp(x^\top \beta)}$$

where  $\beta$  is estimated from the training data.

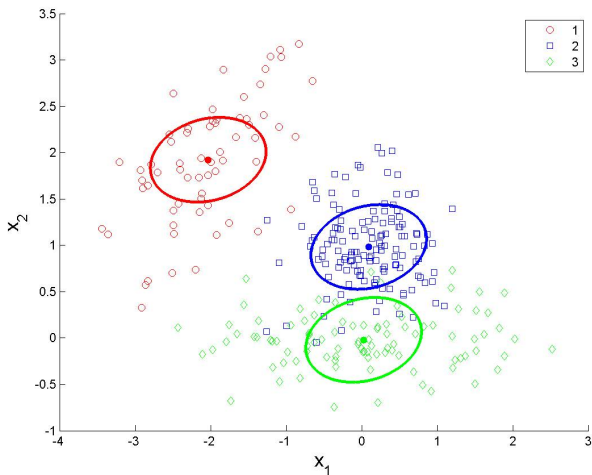
The **indirect approach** uses Bayes' formula

$$\mathbb{P}(Y = k|X = x) = \frac{f_{[Y=k]}(x)\mathbb{P}(Y = k)}{\sum_{k=1}^K f_{[Y=k]}(x)\mathbb{P}(Y = k)} = \frac{f_{[Y=k]}(x)\pi_k}{\sum_{k=1}^K f_{[Y=k]}(x)\pi_k}.$$

It is then sufficient to learn the distribution of  $X$  given  $Y$  and the distribution of  $Y$ . For example, in DA, the distribution of  $X$  given  $Y$  is Gaussian and the parameters are estimated from the training data as we have just seen.

## Linear discriminant analysis - an example

First, we assume that  $\Sigma_k = \Sigma$  for  $k = 1, \dots, K$ .



## Linear discriminant analysis

Under the assumption of equality of covariance matrices, we obtain:

$$\log \mathbb{P}(Y = k | X = x) = \log(f_{[Y=k]}(x) \mathbb{P}(Y = k)) = C(x) + L_k(x)$$

where  $C$  is a function that does not depend on the class  $k$  and

$$L_k(x) = x^\top \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^\top \Sigma^{-1} \mu_k + \log \pi_k.$$

So the Bayes classification rule becomes

$$g(x) = \arg \max_{k \in \{1, \dots, K\}} L_k(x).$$

$L_k$  is then called **linear discriminant function** and measures a class membership score.



## Linear Discriminant Analysis

To conclude, LDA assigns the input  $x$  to the class that maximizes  $\hat{L}_k(x)$ , where we have replaced in the expression of  $L_k(x)$  the unknown quantities  $\mu_k$  and  $\pi_k$  by their estimators.

The maximum likelihood estimator of  $\Sigma$  is the **intra-group variance-covariance matrix** defined by:

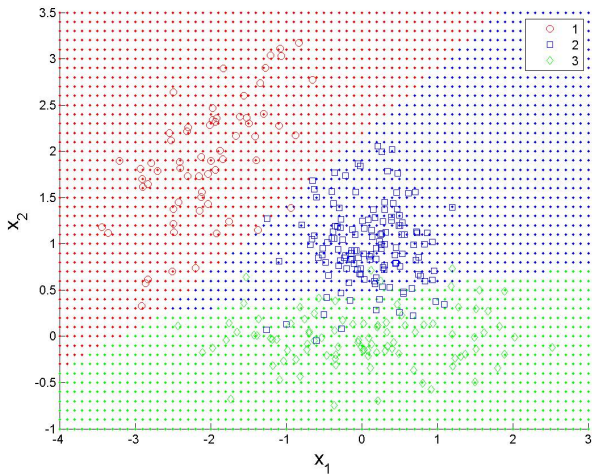
$$\hat{\Sigma} = \frac{1}{n} \sum_{k=1}^K n_k \hat{\Sigma}_k \quad \text{with} \quad \hat{\Sigma}_k = \frac{1}{n_k} \sum_{i; y_i=k} (x_i - \hat{\mu}_k)^\top (x_i - \hat{\mu}_k).$$

The posterior probabilities of the classes are calculated as follows:

$$\mathbb{P}(Y = k | X = x) = \frac{\exp(L_k(x))}{\sum_{\ell=1}^K \exp(L_\ell(x))}.$$

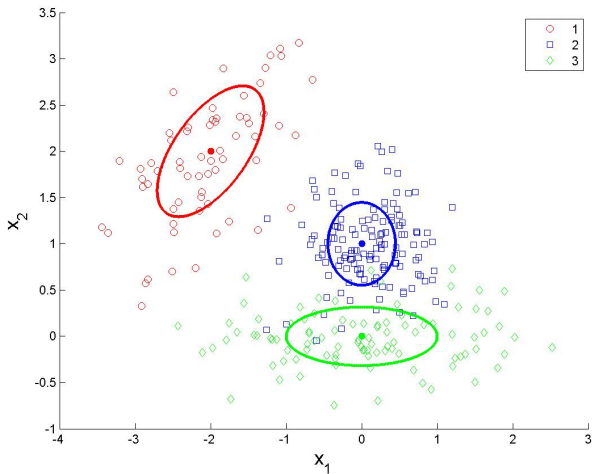
## Linear discriminant analysis - an example

The decision boundary between two classes  $k$  and  $\ell$  is described by a **linear** equation in  $x$ :  $\{x : L_k(x) = L_\ell(x)\}$ .



## Quadratic discriminant analysis - an example

We no longer assume that  $\Sigma_k = \Sigma$  for all  $k = 1, \dots, K$ .



## Quadratic discriminant analysis

Without the assumption of equality of covariances, we get

$$g(x) = \arg \max_{k \in \{1, \dots, K\}} Q_k(x) \quad \text{with}$$

$$Q_k(x) = -\frac{1}{2} \log |\Sigma_k|^{-1} - \frac{1}{2} (x - \mu_k)^\top \Sigma_k^{-1} (x - \mu_k) + \log \pi_k.$$

$Q_k$  is then called **quadratic discriminant function** and measures a membership score for the classes.

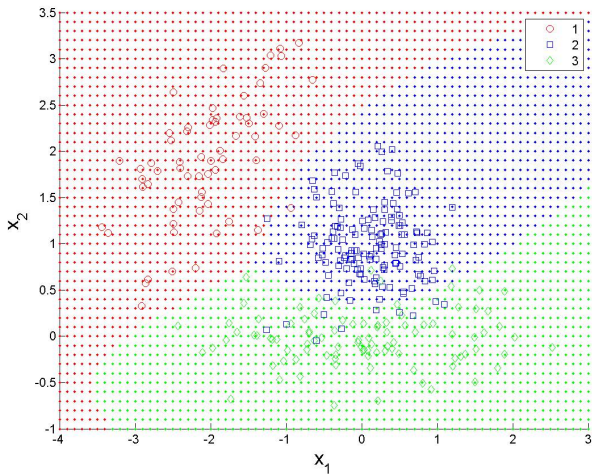
The posterior probabilities of the classes are calculated as follows:

$$\mathbb{P}(Y = k | X = x) = \frac{\exp(Q_k(x))}{\sum_{\ell=1}^K \exp(Q_\ell(x))}.$$

$-2Q_k$  is the **generalized Mahalanobis distance** between  $x$  and  $\mu_k$ .

## Quadratic Discriminant Analysis - An Example

The decision boundary between two classes  $k$  and  $\ell$  is described by a **quadratic** equation in  $x$ :  $\{x : Q_k(x) = Q_\ell(x)\}$ .



## Special case of binary classification where $K = 2$

We assume that  $\Sigma_k = \Sigma$  for  $k = 1, 2$ .

The **Fisher score** is defined by

$$\begin{aligned}\Delta(x) &= L_1(x) - L_2(x) \\ &= x^\top \Sigma^{-1}(\mu_1 - \mu_2) - \frac{1}{2}(\mu_1 + \mu_2)^\top \Sigma^{-1}(\mu_1 - \mu_2) + \log \frac{\pi_1}{\pi_2}.\end{aligned}$$

This score is a **linear function in  $x$** .

We assign  $x$  to class 1 if  $\Delta(x) \geq 0$  and to class 2 otherwise.

The a posteriori probability of belonging to class 1 is a logistic function of the Fisher score:

$$\mathbb{P}(Y = 1 | X = x) = \frac{\exp \Delta(x)}{1 + \exp \Delta(x)}.$$

## Special case of linear discriminant analysis with uniform probabilities

We assume that  $\Sigma_k = \Sigma$  and  $\pi_k = \mathbb{P}(Y = k) = 1/K$  for  $k = 1, \dots, K$ .

Under this additional assumption of equal prior probabilities, we obtain:

$$g(x) = \arg \min_{k \in \{1, \dots, n\}} D_k(x) \quad \text{with } D_k(x) = (x - \mu_k)^\top \Sigma^{-1} (x - \mu_k).$$

$D_k(x)$  is the **square of the Mahalanobis distance** (metric  $\Sigma^{-1}$ ) between  $x$  and the center  $\mu_k$  of class  $k$ .

We then assign  $x$  to the closest class.

This is called a **geometric classification rule**.

## Limitations of LDA and QDA

- LDA and QDA rely on statistical assumptions (normality of data and homogeneity of covariance matrices for LDA). These assumptions can be partially or totally violated in real datasets.
- Even if LDA or QDA are considered relatively simple models, they can overfit on complex or noisy datasets.



## Naive Bayesian

The input variables  $X = (X_1, \dots, X_p)$  are **quantitative** or **qualitative** and  $Y \in \{1, \dots, K\}$ .

- **Independence hypothesis** of the variables  $X_1, \dots, X_p$  conditionally on  $Y$ :

$$f(x|Y = k) = \prod_{j=1}^p f_j(x_j|Y = k),$$

where  $f_j(x_j|Y = k)$  is the notation used here to denote in a unified way:

- the conditional density of  $X_j$  given  $Y = k$  if  $X_j$  is continuous,
  - a conditional probability of  $X_j$  given  $Y = k$  if  $X_j$  is discrete.
- The indirect approach gives:

$$g(x) = \arg \max_{\ell \in \{1, \dots, K\}} \pi_{\ell} f(x|Y = \ell) = \arg \max_{\ell \in \{1, \dots, K\}} \pi_{\ell} \prod_{j=1}^p f_j(x_j|Y = \ell).$$

## Naive Bayesian

The  $K$  parameters  $p_{j\ell}$  and the  $p \times K$  conditional distributions  $f_j(x_j|Y = k)$  are to be **estimated on the training data** i.e. on a sample of pairs of i.i.d. random variables  $(X_1, Y_1), \dots, (X_n, Y_n)$  with the same distribution as  $(X, Y)$ .

If the variable  $X_j$  is **qualitative** with values in  $\mathcal{M}_j$ , we estimate the conditional probabilities  $f_j(x_j|Y = k)$  by the frequencies in the class  $k$  of the modalities  $x_j \in \mathcal{M}_j$  :

$$\hat{f}_j(x_j|Y = k) = \frac{\sum_{i: y_i = k}^n \mathbb{1}_{X_i^j = x_j}}{n_k}.$$

# Naive Bayesian

If the variable  $X_j$  is quantitative with values in  $\mathbb{R}$  :

- We can assume a **parametric form** for  $f_j(x_j|Y = k)$  and estimate the parameters by maximum likelihood. For example

$$\hat{f}_j(x_j|Y = k) = \frac{1}{\sqrt{2\pi\hat{\sigma}_{jk}^2}} \exp\left(-\frac{1}{2\hat{\sigma}_{jk}^2}(x_j - \hat{\mu}_{jk})^2\right)$$

where  $\hat{\mu}_{jk}$  is the empirical mean and  $\hat{\sigma}_{jk}^2$  is the corrected empirical variance of the variable  $X_j$  in class  $k$ .

- We can also estimate  $f_j(x_j|Y = k)$  nonparametrically using a **histogram** or a **kernel density estimator**.

## Naive Bayesian

⚠ The assumption of independence of the variables  $X_1, \dots, X_p$  conditionally to  $Y$  is generally false.

However, this approach is very common:

- it is simple, fast and works for a non-binary output variable, and input variables of any type.
- it allows to process high-dimensional data.

## Logistic regression

Here we seek to predict the behavior of an output variable  $Y$  based on the inputs  $X$ , as in linear or polynomial regression.

However, in the context of ordinary **logistic regression**,  $Y$  is not a real number but **takes the values 0 or 1** (it is a binary variable). It allows us to evaluate the effect of different explanatory variables on a variable of interest.

This machine learning method is widely used in the field of marketing to evaluate the sale or not of products following a decision or in the medical field to evaluate the cure or not of a patient.

## Logistic regression

The input variables are **quantitative** or **qualitative** and  $Y \in \{0, 1\}$ .

- The **qualitative** variables are recoded by the **category indicators** and  $X = (X_1, \dots, X_p) \in \mathbb{R}^p$  with  $X_j$  **quantitative** or **binary**.
- In logistic regression, we are interested in the **distribution of  $Y$  given  $X$**  which is a Bernoulli distribution of parameter  $p$  with:

$$\mathbb{P}(Y = 1 | X = x) = p$$

$$\mathbb{P}(Y = 0 | X = x) = 1 - p.$$

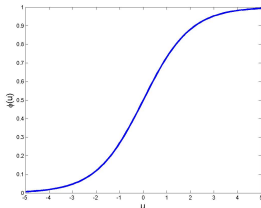
## Logistic regression

We assume that the probability  $p = \mathbb{P}(Y = 1|X = x)$  is a **logistic function** of a **linear score**

$$\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p \in \mathbb{R}$$

and the logistic function  $f : \mathbb{R} \rightarrow [0, 1]$  is defined by :

$$f(u) = \frac{\exp(u)}{1 + \exp(u)}.$$



## Logistic regression

We therefore model the **posteriori probability**  $p$  by:

$$p = \mathbb{P}(Y = 1 | X = x) = f\left(\beta_0 + \sum_{j=1}^p \beta_j x_j\right) = \frac{\exp(\beta_0 + \sum_{j=1}^p \beta_j x_j)}{1 + \exp(\beta_0 + \sum_{j=1}^p \beta_j x_j)}$$

The linear **score** is then:

$$\beta_0 + \sum_{j=1}^p \beta_j x_j = f^{-1}(p) = \log\left(\frac{p}{1-p}\right).$$

The function  $f^{-1}$  is called **logit function** with:

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right).$$



## Logistic regression

The **unknown parameters**  $(\beta_0, \dots, \beta_p)$  are estimated by maximum likelihood.

**Log-likelihood** (conditional) of the sample  $(X_1, Y_1), \dots, (X_n, Y_n)$  :

$$\begin{aligned}\ell(\beta_0, \dots, \beta_p) &= \log \prod_{i=1}^p \mathbb{P}(Y_i = y_i | X_i = x_i) \\ &= \log \prod_{i=1}^p p_i^{y_i} (1 - p_i)^{1-y_i}\end{aligned}$$

with

$$p_i = \mathbb{P}(Y_i = 1 | X_i = x_i) = \frac{\exp(\beta_0 + \sum_{j=1}^p \beta_j x_{ij})}{1 + \exp(\beta_0 + \sum_{j=1}^p \beta_j x_{ij})}.$$

## Logistic regression

⚠ The maximum likelihood estimator of  $\beta$  does not have an explicit form.

Software therefore uses optimization algorithms to estimate the parameters  $\beta_0, \dots, \beta_p$  on the training data.

The algorithm often used is that of **Newton-Raphson** which is an iterative method based on the following relation:

$$\beta^{(t)} = \beta^{(t-1)} - \left( \frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^\top} \Big|_{\beta^{(t-1)}} \right)^{-1} \frac{\partial \ell(\beta)}{\partial \beta} \Big|_{\beta^{(t-1)}}$$

where  $\beta = (\beta_0, \dots, \beta_p)$ .

## Logistic regression

The **classification rule**  $g$  then assigns a new observation  $x$  to class 1 if

$$\hat{p} = \frac{\exp(\hat{\beta}_0 + \sum_{j=1}^p \hat{\beta}_j x_j)}{1 + \exp(\hat{\beta}_0 + \sum_{j=1}^p \hat{\beta}_j x_j)} \geq 0.5$$

It is assigned to class 0 otherwise.

## Logistic regression

Logistic regression can be extended to the case of **multiple classes**. We then speak of **multinomial logistic regression**.

We now have  $Y \in \{1, \dots, K\}$  and we note  $X = (1, X_1, \dots, X_p)$ .

The model takes the form

$$\begin{aligned}\log \frac{\mathbb{P}(Y = 1|X = x)}{\mathbb{P}(Y = K|X = x)} &= x^\top \beta_1 \\ \log \frac{\mathbb{P}(Y = 2|X = x)}{\mathbb{P}(Y = K|X = x)} &= x^\top \beta_2 \\ &\vdots \\ \log \frac{\mathbb{P}(Y = K-1|X = x)}{\mathbb{P}(Y = K|X = x)} &= x^\top \beta_{K-1}\end{aligned}$$

with  $\beta_1, \dots, \beta_{K-1}$  vectors of  $\mathbb{R}^{p+1}$ .

## Logistic regression

The  $K - 1$  vectors  $\beta_k$  are **estimated** by maximum likelihood on the **training data**.

The **posteriori probabilities** are then:

$$\mathbb{P}(Y = K | X = x) = \frac{1}{1 + \sum_{\ell=1}^{K-1} \exp(x^\top \beta_\ell)}$$

$$\mathbb{P}(Y = 1 | X = x) = \frac{\exp(x^\top \beta_1)}{1 + \sum_{\ell=1}^{K-1} \exp(x^\top \beta_\ell)}$$

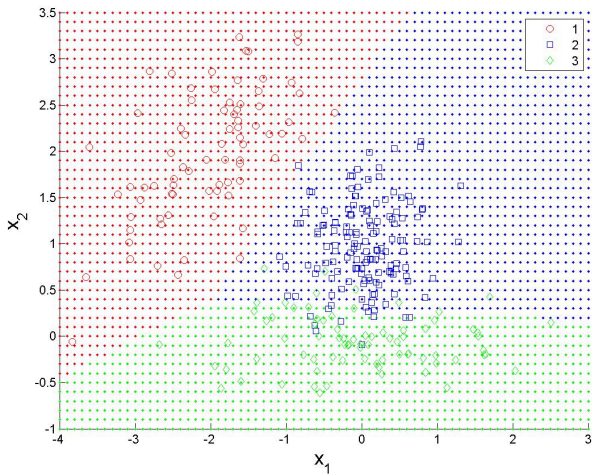
$$\vdots$$

$$\mathbb{P}(Y = K - 1 | X = x) = \frac{\exp(x^\top \beta_{K-1})}{1 + \sum_{\ell=1}^{K-1} \exp(x^\top \beta_\ell)}.$$

The **classification rule**  $g$  then assigns a new observation  $x$  to the most probable class a posteriori.

# Logistic Regression

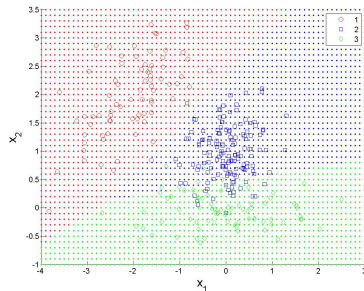
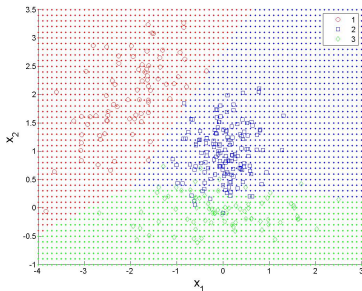
**Example:**  $p = 2$ ,  $K = 3$  classes



# Logistic Regression

## Comparison with Linear Discriminant Analysis.

Logistic regression (left) versus LDA (right).



# Plan

## Introduction to supervised learning

- Predictive performance evaluation

- Database slicing: learning, validation, testing

## Linear regression

## The $k$ -nearest neighbors

## Model-based approaches

- Discriminant analysis

- Naive Bayesian

- Logistic regression

## Other supervised classification algorithms



## Classification trees

Classification trees (or decision trees) and regression are methods for creating a decision model. This structure is made up of

- of root nodes, constituting the tree's inputs,
- of internal nodes, performing intermediate operations
- and of leaves (or terminal nodes), representing the value of the output variables.

This hierarchical structure is based on a chain of decisions creating the passage from one node to another, up to the leaves. The tree is generally built recursively by searching for each node the decision allowing the best sharing of the data set. After a decision, we then have several subsets of data on which we apply the same process.

## Classification trees

Decision trees have the advantage of having a clear decision process, guaranteeing good explainability of the solution. They have long been used before the development of artificial neural networks.

It is possible to break down more complex problems by using several classification trees, this is called **random forest**.

## Artificial neural networks

Neural networks are complex models that can be used in supervised learning for regression and classification.

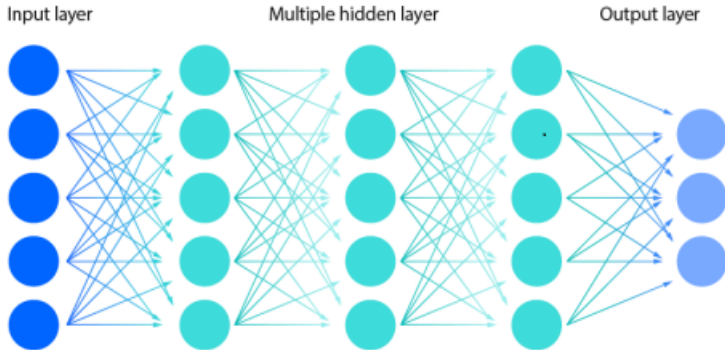
They are increasingly used in many fields because they allow complex phenomena to be modeled.

A neural network is generally composed of a succession of layers, each of which takes its inputs from the outputs of the previous one:

$$y = f_K \circ \dots \circ f_1(x) \quad \text{with} \quad f_j(z_j) = g_j(\alpha_j + \beta_j z_j).$$

They are found, for example, in image processing for pattern recognition, in speech processing and for the approximation of complex functions that take a long time to calculate.

# Artificial neural networks



Source: <https://www.ibm.com/fr-fr/topics/neural-networks>

## Support vector machines

Support vector machines are a generalization of linear classifiers and estimators (such as linear regression models) for higher-dimensional data.

The model then takes as input a parameter vector  $X$  and associates it with an output value  $Y$  using a function  $h(X)$ .

In the case of classification, separating hyperplanes are introduced as decision criteria. The space of the output value  $Y$  is separated into different zones, each corresponding to predictions.

For example, a class will be predicted if the value of  $Y$  is positive and another if it is negative.

# Cảm ơn sự chú ý của bạn !

## Câu hỏi ?

Major sources of the whole lecture

- Marie Chavent's lectures  
<https://marie-chavent.perso.math.cnrs.fr/>
- Juliette Chevallier's lectures  
<https://juliette-chevallier.pages.math.cnrs.fr/>

