Introduction to Machine Learning

César Hernández,
Ekhiñe Irurozki,
Aritz Pérez

Data pre-processing and more...

César Hernández

Outline

- Introduction
- Technologies, software and “to know how the companies do it”
- Important information links.
- Data pre-processing
  1) Data checking.
     - Outliers detection.
     - Missing values.
  2) Numeric data standardization & normalization.
  3) Numeric data discretization.
  4) Feature selection.
  5) Feature extraction.
  6) Handling unbalanced data.

Introduction

Introduction

- Which is my **purpose** when using ML with my data?

- Did I collect my data?
  - If so, did I do it right?
  - If not, can I be (almost) sure that my data was rightly collected?
  - What does “right” mean?
  - Right in terms of data representation and structure, volume, … (problem dependent)

- Is my data useful? (Reliable, representative, noisy, …)

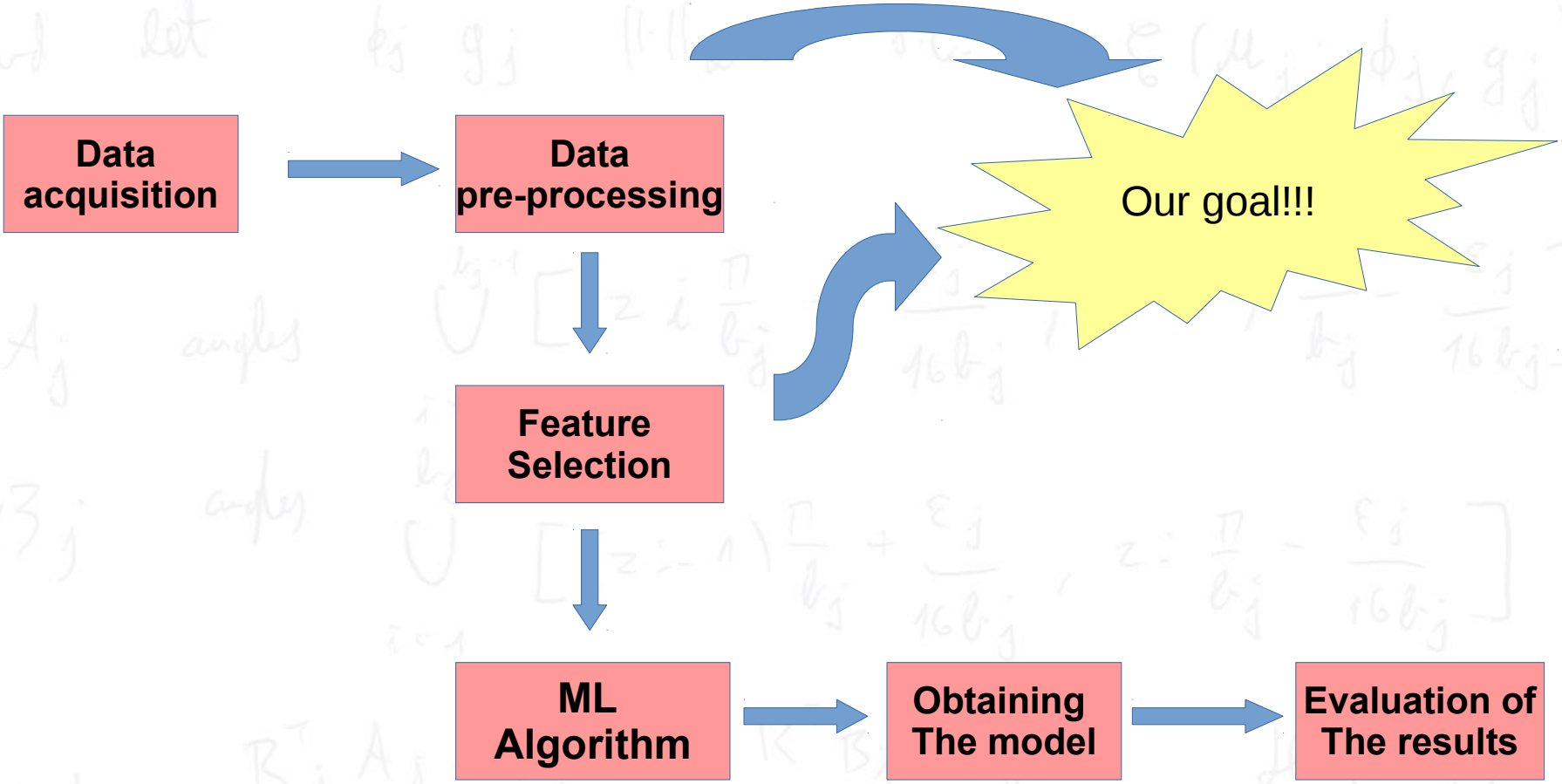
- Once my goal and my options in term of ML algorithms (computation or memory limitations, etc.) are clear, how will I evaluate the goodness of my results and infer valid conclusions?

Introduction

Summarizing:

- Know where you want to go.
- Know how to go there.
- Know (if you can) that you will eventually reach there.
- The first step in your journey is **preprocessing** your data.
- It has been usually **neglected**.
- In real-world problems, data preprocessing takes around **80%** of your **working time**.
- It is crucial for the subsequent steps. The **bigger** your **data**, the **more important** their preprocessing process.

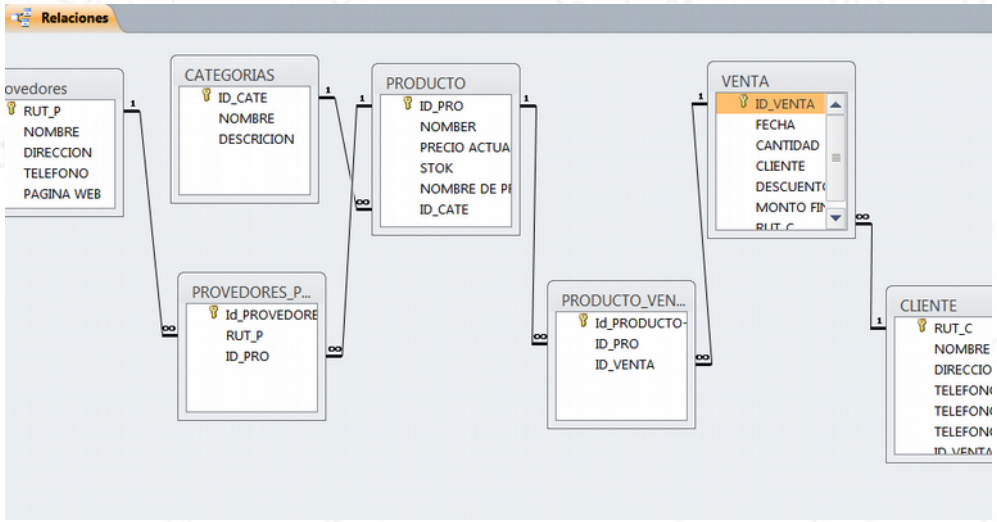
Introduction

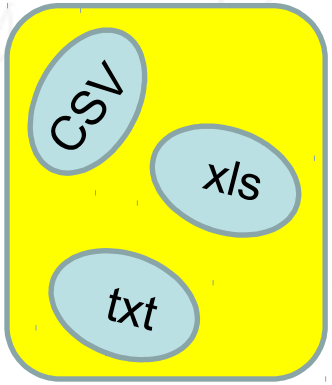


Technologies, software and "to know how the companies do it"

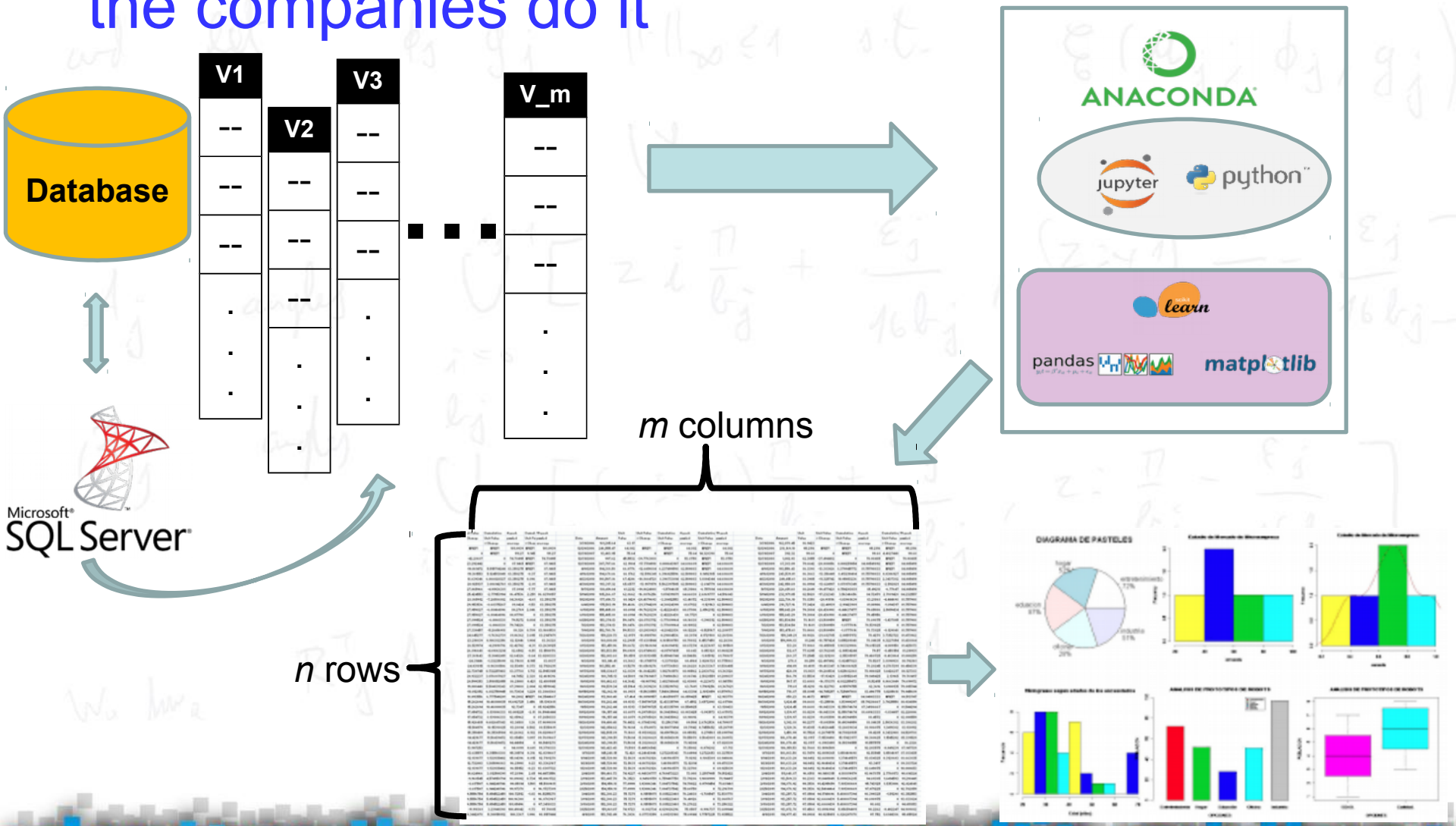
Technologies, software and “to know how the companies do it”

Queries using SQL (Structured Query Language)





Technologies, software and “to know how the companies do it”



Important information links

Important information links

- Github repository

  https://github.com/ceelch/Introduction_ML_2019_BCAM

- Anaconda, R, Rstudio, Weka, Spark

  https://www.anaconda.com/
  https://www.r-project.org/
  https://www.rstudio.com/products/rstudio/download/
  https://www.cs.waikato.ac.nz/ml/weka/downloading.html
  https://spark.apache.org/

- SQL practice
  https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_columns

1) Data checking

1) Data checking

- **Expert knowledge** is always relevant, not only in preprocessing.
- Due to expert knowledge you can:
  - Know which features are the most **relevant** (from experience).
  - Know the valid ranges of numerical variables (**domain knowledge**).
  - Identify **redundancies** in the features.
  - Set **bounds** to the modeling choices.

1) Data checking

- **Data cleaning** can be done by applying expert knowledge or in a data-driven fashion.
- Get familiar with your data:
  - Look for **useless attributes** (noticeable even without expert knowledge).
  - Look for **corrupted data** (empty or constant variables).
- Just have a look and see what comes out, if the volume of your data allows it.

1) Data checking:

Outliers detection

Outliers detection

- Their detection can be by expert knowledge (e.g. known attribute's range) or data-driven. Let's focus on the latter.
- Outliers detection can be done in two ways:
  - Individual: Performed variable by variable. The simplest approaches consist on considering as outliers the values outside the intervals:

$$[\bar{X} - 3 * S_X, \bar{X} + 3 * S_X]$$

$$[Q_1 - 1.5 * IQR, Q_3 + 1.5 * IQR]$$

- Collective: Using all variables. The most representative one takes into account the distribution of the data, and is based on calculating an ellipsoid around the data points, using Mahalanobis distance. Te points outside it are considered as outliers.
- When detected the record is ignored or imputed (risky).

1) Data checking:

Missing values

Missing values

- **Missing values** are unexpected “holes” in the data.
- There are three strategies for dealing with missing values:
  - **Ignore records** (rows, samples, …) where missing values are detected.
  - Treat missing values in an attribute as **legitimate values** of the attribute.
  - **Impute** a value to “fill the holes”:
    - **Direct imputation**, in a supervised or unsupervised way, of a significant value, such as the mean or median (numerical attributes), the mode categorical attributes), or a suggestion by an expert.
    - More **advanced techniques** requiring some data analysis, such as k-nearest neighbours or expectation & maximization algorithm.

2) Numeric data standardization & normalization

2) Numeric data standardization & normalization

- When computing distances between pairs of samples, the scales of the different features is very relevant.

- Moreover, we cannot obviate the curse of dimensionality effect, i.e. in high-dimensional spaces all data is sparse. In simple words, all distances become huge.

- Therefore, if the algorithm we plan to apply after preprocessing implies distances and/or we are in a high-dimensional problem, we should transform all the features to a similar scale.

2) Numeric data standardization & normalization

- Some of the approaches are:
  - Mean centering: scaling to zero mean by subtracting the mean. Necessary, for instance, in approaches related to Mahalanobis distance, like PCA.
  - Standardization: scaling to zero mean and unit variance by subtracting the mean and dividing by the standard deviation.
  - Normalization: scaling to [0, 1] by subtracting the minimum and dividing by the range.
- In general, do **not** standardize or normalize your data unless you have a good reason for doing so.
- Other **transformations** from Statistics (logit, square root, power, Box-Cox, angular, etc.) are not considered because they are part of the posterior processing phase.

3) Numeric data discretization

3) Numeric data discretization

- The goal of discretization is reducing the number of values of a continuous attribute by grouping them into intervals (bins). The new values are the bins.
- Some methods require discrete attributes (some naïve Bayes and Bayesian networks methods, etc.).
- Sometimes, the results are better after discretization. Some methods do it implicitly (e.g. decision trees).
- In general, the computational cost of algorithms with discrete attributes is lower than their continuous versions.

3) Numeric data discretization

- Types of discretization: Unsupervised vs supervised.

- **Unsupervised**. Without using the attribute of interest.

- Some require to fix the number of bins, *k*, in advance:
  - If too large the bins could have not enough data to be significant.
  - If too little, we might loose the informative power of the attribute.
  - It is tricky to find out the right value.

- Examples:
  - **Equal-width**. Divide the range into *k* equal-width ranges. It could cause unbalance (e.g. the salary in a company).
  - **Equal-frequency**. Divide the range into *k* ranges with the same frequency. Better than equal-width in terms of clumping, but could produce odd artifacts with frequent or special values.

3) Numeric data discretization

- **Supervised**. Using the attribute of interest.
- In general, they have the advantage of **not having to fix *k* in advance**.

4) Feature selection

4) Feature selection

- With variables, there are **possible feature subsets**. Exploring all options is not frequently feasible.

- **Feature selection** consists on choosing a subset of the original features according to certain criteria to be optimized (**optimization problem**).

- Depending on the criteria and the way to optimize it we can get two types of outputs:

  - Just the subset of selected features.
  - A measurement of the **importance** of the features. This allows us to rank the features. Then we could get a subset by fixing a threshold for a cut-point of the importance.

4) Feature selection

- There are three types:
  - Filter methods: based on **intrinsic properties** of the data. The most widely used are **statistical** properties, such as correlation or mutual information. Not algorithm dependent (universal).
  - Wrapper methods: using some **learning task** in the selection. Unless the computational cost is too high, we would use the same learning task chosen for the subsequent processing. Usually much more **computationally expensive** than filters.
  - Embedded methods: included inside the algorithm (outside preprocessing).

- Examples of filters: **Correlation-based feature selection** (different definitions of correlation), **InfoGain** (mutual information), **ReliefF** (distance to the nearest sample from the same class and from a different class), **simmetrical uncertainty** (entropy of the sample and the class).

5) Feature extraction

5) Feature extraction

- **Feature extraction** is the action of defining new features from all or part of the original ones.

- Despite expert knowledge could be also used, we focus on data-driven approaches.

- When applying feature extraction, there is usually a price to pay in terms of **loss of interpretability** because of the lack of meaning of the artificial features.

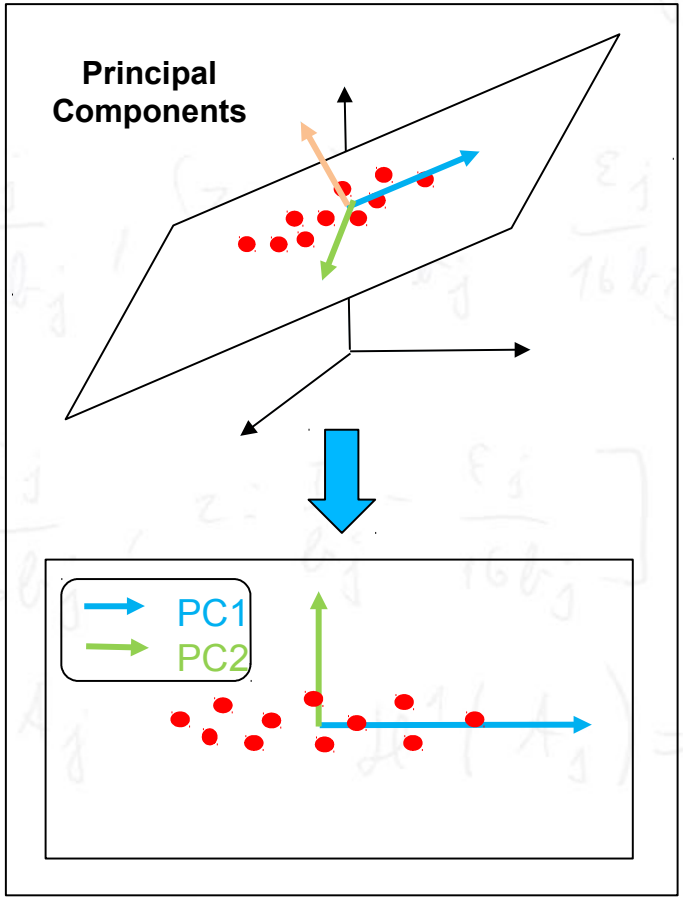
- There are both **supervised** and **unsupervised** methods.

- We will have a deeper look to the most famous unsupervised (and linear) method: **principal component analysis** (PCA).

5) Feature extraction

- **Principal Components Analysis**

PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.

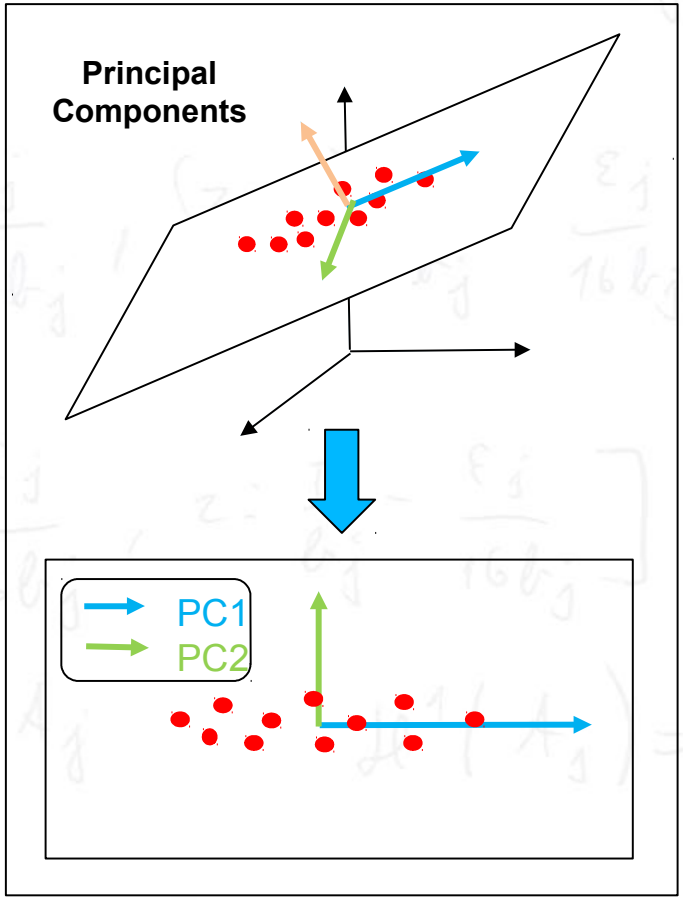


5) Feature extraction

- **Principal Components Analysis**

Algorithm.

1) Obtain the eigenvectors and eigenvalues of the covariance matrix.
2) Order the eigenvalues from highest to lowest and choose the *"k"* eigenvectors that correspond to the larger *"k"* eigenvectors (where *"k"* is the number of dimensions of the new characteristics subspace).
3) Build the projection matrix *W* with the *"k"* selected autovectors.
4) We transform the original "*X* standardized" dataset via *W* to obtain the new *k*-dimensional characteristics.



6) Handling unbalanced data

6) Handling unbalanced data

- **Unbalanced data** in classification happens when the class of interest has very low frequency.

- Example: Predicting if certain part of a machine is faulty or not, based on sensor data (fault detection problem).

- Cost-sensitive methods: penalize the errors in the minority class in order to guide the training towards a better representation of that class.

6) Handling unbalanced data

- Sampling-based methods:
  - Oversampling: add artificially samples of the minority class by copying (randomly or not) or creating instances.
  - Undersampling: Remove samples (randomly or not) of the majority class.
  - Combined: Both oversampling and undersampling.

- Pros & Cons:
  - Classes get equilibrated, promoting a better classification trade-off.
  - Undersampling removes samples that could contain relevant discriminant information.
  - Oversampling incorporates artificial/redundant information that could drive models (depending on their characteristics) towards non-realistic conclusions.

6) Handling unbalanced data

➔ Oversampling methods

- Selection methods:
  - *Random oversampling*. With or without replacement, pure or guided,…
  - Generation methods:
    - *Synthetic Minority Oversampling TEchnique* (SMOTE).
    - *ADAptive SYNthetic sampling method* (ADASYN).

➔ Undersampling methods

- Selection methods:
  - *Random undersampling*. With or without replacement, pure or guided,…

Introduction to Machine Learning

César Hernández,
Ekhiñe Irurozki,
Aritz Pérez

César Ernesto Hernández Hernández.

Postdoc Fellow.

Machine Learning.

chernandez@bcamath.org

http://www.bcamath.org/es/people/chernandez