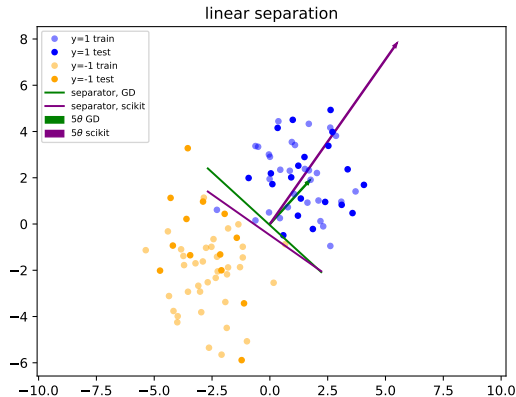


# Fondamentaux théoriques du machine learning



## Overview of lecture 4

Scoring, multiclass problems, cross validation

- Regression

- Binary classification

- Multi-class classification

- Cross-validation

Classification

- Problem statement

- Convexification of the risk and calibration

- Logistic regression

Gradient algorithms

Dimensionality reduction

- Principal component analysis

Clustering

## Scoring, multiclass problems, cross validation

Regression

Binary classification

Multi-class classification

Cross-validation

## Classification

Problem statement

Convexification of the risk and calibration

Logistic regression

## Gradient algorithms

## Dimensionality reduction

Principal component analysis

## Clustering

# Scoring

Many possibilities are available to evaluate the quality of an estimator.

# Regression

- ▶  $\mathcal{X} = \mathbb{R}^d$
- ▶  $\mathcal{Y} = \mathbb{R}$ .

Until now we used the squared error or the mean squared error (MSE) :

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_{pred,i} - y_{label,i})^2 \quad (1)$$

Without normalisation, it is also called **residual sum of squares** (RSS)

$$RSS = \sum_{i=1}^n (y_{pred,i} - y_{label,i})^2 \quad (2)$$

## Coefficient of determination

Also called  $R^2$ .  $R^2 \leq 1$ . We introduce the Total sum of squares (TSS)

$$TSS = \sum_{i=1}^n (y_{label,i} - \bar{y})^2 \quad (3)$$

where  $\bar{y}$  is the mean of the labels :

$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_{label,i} \quad (4)$$

Then

$$R^2 = 1 - \frac{RSS}{TSS} \quad (5)$$

## Coefficient of determination

$$TSS = \sum_{i=1}^n (y_{label,i} - \bar{y})^2 \quad (6)$$

$$RSS = \sum_{i=1}^n (y_{pred,i} - y_{label,i})^2 \quad (7)$$

Finally we define the Explained sum of squares ESS :

$$ESS = \sum_{i=1}^n (y_{pred,i} - \bar{y})^2 \quad (8)$$

Then if the predictions are linear, then

$$TSS = ESS + RSS \quad (9)$$

## Scikit metrics

`https://scikit-learn.org/stable/modules/model\_evaluation.html`



## Binary classification

We now review some metrics for binary classification problems.

# Accuracy

Most simple scoring : accuracy.

$$\frac{\text{number of correct predictions}}{\text{number of samples}} \quad (10)$$

## Precision and recall

Precision : "Quand tu dis que c'est positif, c'est positif".

$$\frac{TP}{TP + FP} \quad (11)$$

Recall / sensitivity (rappel) : "Quand c'est positif, tu dis que c'est positif".

$$\frac{TP}{TP + FN} \quad (12)$$

See also : specificity and 1-specificity.

## F score

Also called  $F1$ . It quantifies the tradeoff between precision and recall.

$$F1 = 2 \times \frac{\text{sensitivity} \times \text{precision}}{\text{sensitivity} + \text{precision}} \quad (13)$$

**Exercise 1:** What are the extremal values of  $F$ ?

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

- └ Scoring, multiclass problems, cross validation
- └ Multi-class classification

## Binary classification

Standard classifiers such as logistic regression and support vector machines are **binary**.

However, sometimes  $|\mathcal{Y}| = p > 2$ . In these situations, several binary classifiers are aggregated in order to perform the classification.

- ▶ one-vs-rest scheme : learns a binary classifier per class. At prediction time, select the class with highest score.
- ▶ one-vs-one scheme : learns as many binary classifiers as there are pairs of classes.

- └ Scoring, multiclass problems, cross validation
- └ Multi-class classification

## Number of classifiers

We assume  $|\mathcal{Y}| = p$ .

**Exercise 2:** How many classifiers need to be built :

- ▶ with the one-vs-rest scheme?
- ▶ with the one-vs-one scheme?

## Number of classifiers

- ▶ one-vs-rest is the standard approach.
- ▶ one-vs-one need to build a number of classifiers that is quadratic in  $p$ , ( $\mathcal{O}(p^2)$ ).
- ▶ However one-vs-one might still be useful since less samples are used by each binary classifiers, since we only need the samples from the two selected classes. If the complexity of the classifier scales badly with the number of samples  $n$  (like for kernel methods), one-vs-one might be faster.

- └ Scoring, multiclass problems, cross validation
- └ Multi-class classification

# Scikit

Scikit has a builtin implementation of both schemes.

<https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html>



- └ Scoring, multiclass problems, cross validation
- └ Multi-class classification

# Softmax

It is also possible to directly learn  $p$  real outputs and convert the vector of outputs to a probability by normalizing (it is what is done with softmax regression in neural networks).

[https://fr.wikipedia.org/wiki/Fonction\\_softmax](https://fr.wikipedia.org/wiki/Fonction_softmax)

- └ Scoring, multiclass problems, cross validation
- └ Multi-class classification

## Confusion matrix

```
https://en.wikipedia.org/wiki/Confusion_matrix  
https://scikit-learn.org/stable/modules/generated/  
sklearn.metrics.confusion_matrix.html
```

## Classification report

It is also possible to define precision, recall (and thus F1) for each class. In scikit, classification report prints these quantities for each class

`https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification\_report.html`

## Multi-class vs multi-label

Don't mix multi-class problems and multi-label problems.

- ▶ multi-class : several output classes are possible
- ▶ multi-label : we have to make several predictions for each input

A problem can be both multi-class and multi-label.

## Hyperparameters (summary of TP3)

All learning algorithms have hyperparameters. Examples :

- ▶ regularization parameter
- ▶ learning rate schedule
- ▶ kernel widths
- ▶ tree depth for cart
- ▶ number of trees for random forest

## Hyperparameter tuning

Sometimes, we have theoretical results that guarantee that a hyperparameter value is a good choice (e.g. the learning rates for GD, SGD, SAG).

**However :**

- ▶ often, these parameters values depend on constants that are problem-dependent and sometimes not available :
  - ▶ variance  $\sigma^2$
  - ▶ smoothness constant  $L$
- ▶ we may not have a theoretical result at all (true for some aspects of deep learning)
- ▶ some values of the hyperparameter might work **better** than the theoretical value.

- └ Scoring, multiclass problems, cross validation
- └ Cross-validation

## Hyperparameter tuning

**Conclusion** : it is most often necessary to experiment and test in order to find relevant hyperparameters.

## Train / validation / test sets

The dataset can be split into 3 parts :

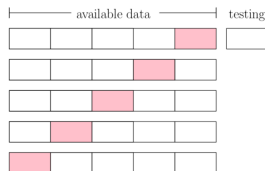
- ▶ train set : used to optimize each model
- ▶ validation set : used to compute a validation error of each model (error on this dataset). The model with lowest validation error can be chosen.
- ▶ test set : used to test the final model. We can not use it for validation (choice of the hyperparameters) : otherwise the estimation of the test error becomes biased.

**Problem** : there might be a high variability in the validation procedure. The found hyperparameters might depend a lot on the initial choice of the validation set.



## Cross-validation

Cross-validation validation is another method that allows the use of more training data. The train set is split in  $k$  folds (often 5 or 10), and  $k$  validation errors are computed (one for each fold). The model with the lowest average validation error is chosen, **and then** trained on the whole train set.



Due to the higher number of computations, cross-validation might be slower than standard train/validation/test split.

## Choice of the set of possible HPs

Grid search is a method for testing hyper parameters (exhaustive search among a given list of values).

- ▶ grid search : exhaustive
- ▶ random search, successive halving.
- ▶ bayesian optimization (optuna, skopt)

## Learning curves

[https://scikit-learn.org/stable/auto\\_examples/model\\_selection/plot\\_learning\\_curve.html](https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html)

Many model selection methods exist :

[https://scikit-learn.org/stable/model\\_selection.html](https://scikit-learn.org/stable/model_selection.html)

Scoring, multiclass problems, cross validation

Regression

Binary classification

Multi-class classification

Cross-validation

## Classification

Problem statement

Convexification of the risk and calibration

Logistic regression

Gradient algorithms

Dimensionality reduction

Principal component analysis

Clustering

## General classification problem

- ▶  $\mathcal{X} = \mathbb{R}^d$
- ▶  $\mathcal{Y} = \{-1, 1\}$  or  $\mathcal{Y} = \{0, 1\}$ .
- ▶  $l(y, z) = 1_{y \neq z}$  ("0-1" loss)
- ▶  $F = \mathcal{Y}^{\mathcal{X}}$

# Problem

Optimizing on  $F = \mathcal{Y}^{\mathcal{X}}$  is equivalent to optimizing in the set of subsets of  $\mathcal{X}$ .

We cannot differentiate on this hypothesis space and it is not clear how to regularize.

# Subsets

## Exercise 3 : Combinatorial problem

If we wanted to try all applications in  $\mathcal{Y}^{\mathcal{X}}$ , if  $|\mathcal{X}| = n$ , how many applications would there be ?

## Real-valued function

Instead of an application in  $\mathcal{Y}^{\mathcal{X}}$ , we will learn  $g : \mathcal{X} \rightarrow \mathbb{R}$  and define  $f(x) = \text{sign}(g(x))$  with

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$



# Risk

The risk (generalization error) of  $f = \text{sign} \circ g$  is.

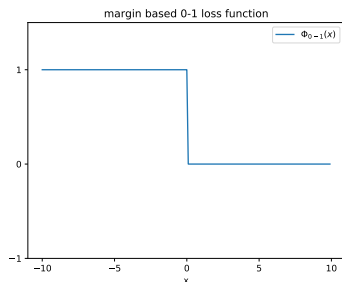
$$\begin{aligned} R(g) &= P(\text{sign}(g(x)) \neq y) \\ &= E \left[ 1_{\text{sign}(g(x)) \neq y} \right] \\ &= E \left[ 1_{yg(x) < 0} \right] \end{aligned} \tag{14}$$

## Remark : several solutions

If  $f^*$  is the Bayes predictor, there might be many optimal functions  $g$ , i.e : such that  $\text{sign}(g(x)) = f^*(x)$ .

Margin based 0-1 loss function  $\Phi_{0-1}$ 

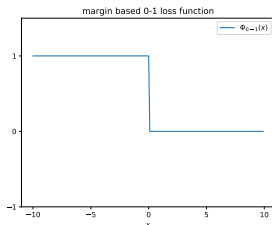
$$\begin{aligned} R(g) &= E \left[ 1_{\text{sign}(g(x)) \neq y} \right] \\ &= E \left[ 1_{yg(x) < 0} \right] \\ &= E \left[ \Phi_{0-1}(yg(x)) \right] \end{aligned} \tag{15}$$



## Empirical risk minimization

The corresponding empirical risk writes :

$$\frac{1}{n} \sum_{i=1}^n \Phi_{0-1}(y_i g(x_i)) \quad (16)$$

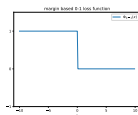


What is the issue with this objective function ?

## Empirical risk minimization

The corresponding empirical risk writes :

$$\frac{1}{n} \sum_{i=1}^n \Phi_{0-1}(y_i g(x_i)) \quad (17)$$



What is the issue with this objective function ?

- ▶ non-convex
- ▶ the differential is not defined or non informative.

## Convex surrogate

Key idea : replace  $\Phi_{0-1}$  by another function  $\Phi$  that is easier to optimize (convexity) but still represents the correctness of the classification.

### Definition

The  $\Phi$ -risk is defined as

$$R_{\Phi}(g) = E \left[ \Phi(yg(x)) \right] \quad (18)$$

The empirical  $\Phi$ -risk is defined as

$$R_{\Phi,n}(g) = \frac{1}{n} \sum_{i=1}^n \Phi(y_i g(x_i)) \quad (19)$$

# Most common convex surrogates

## Definition

Logistic loss

$$\Phi(u) = \log(1 + e^{-u}) \quad (20)$$

With linear predictors, this loss will lead to **logistic regression** (which is classification despite its name).

## Most common convex surrogates

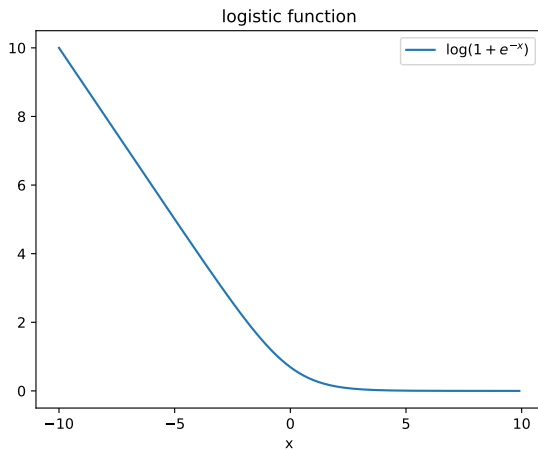
If  $\mathcal{Y} = \{0, 1\}$ ,  $\hat{y}$  is the prediction and  $y$  is the correct label, then we sometimes write :

$$l(\hat{y}, y) = y \log(1 + e^{-\hat{y}}) + (1 - y) \log(1 + e^{\hat{y}}) \quad (21)$$

(cross entropy loss)



# Logistic function



## Most common convex surrogates

### Definition

Hinge loss

$$\Phi(u) = \max(1 - u, 0) \quad (22)$$

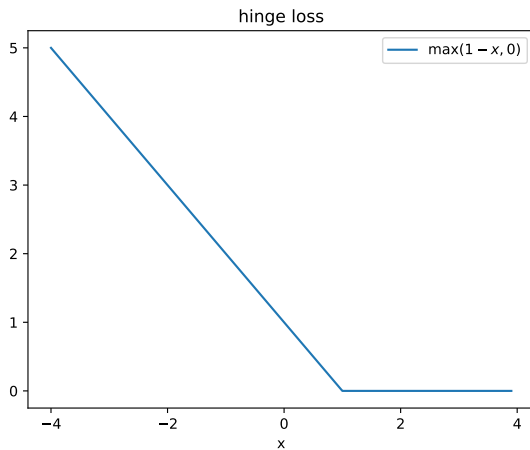
With linear predictors, this loss will lead to **Support vector machines**.

### Definition

Squared hinge loss

$$\Phi(u) = (\max(1 - u, 0))^2 \quad (23)$$

# Hinge loss

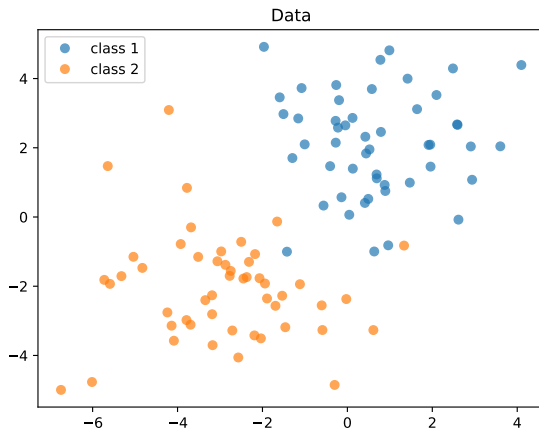


Under some technical hypotheses, minimizing the empirical  $\Phi$  risk leads to a good generalization error for the "0-1" loss (notion of calibration function).

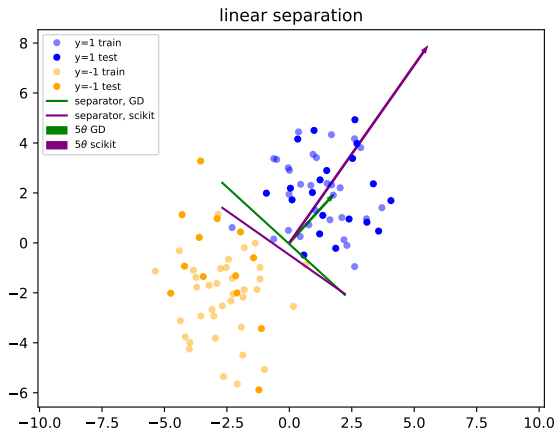
# Logistic regression

- ▶  $g(x) = \langle x, \theta \rangle = x^T \theta$ .
- ▶  $f(x) = \text{sign}(\langle x^T \theta \rangle)$
- ▶ It can be seen as "linear regression applied to classification".

## Data to separate



# Data to separate



# Logistic regression

In this section we use the setting  $\mathcal{Y} = \{0, 1\}$ .

- ▶ prediction :  $\hat{y} = x^T \theta$
- ▶ surrogate loss : cross-entropy loss.

$$l(\hat{y}, y) = y \log(1 + e^{-\hat{y}}) + (1 - y) \log(1 + e^{\hat{y}}) \quad (24)$$



## Logistic regression estimator

If  $l$  is the logistic loss, it is defined as

$$\hat{\theta}_{logit} = \arg \min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n l(x_i^T \theta, y_i) \quad (25)$$

We will show that this empirical risk is convex as a function of  $\theta$  (exercises for next week).

## No closed-form solution

Since the loss is convex, to minimize it is sufficient to look for the cancellation of the gradient. However, the corresponding equation has no closed-form solution.

We thus need to use iterative algorithms in order to find a minimizer (e.g. : gradient descent, Newton's method, etc)

## Practical usage of logistic regression

In practice, it is common practice to :

- ▶ regularize the logistic loss to avoid overfitting, for instance with a  $L2$  penalty (as in ridge regression)
- ▶ use feature maps and classify with  $\phi(x)$  instead of  $x$ .

# Probabilistic interpretation of logistic regression

Later in the course, we will study a probabilistic interpretation of logistic regression.

## Scoring, multiclass problems, cross validation

- Regression

- Binary classification

- Multi-class classification

- Cross-validation

## Classification

- Problem statement

- Convexification of the risk and calibration

- Logistic regression

## Gradient algorithms

## Dimensionality reduction

- Principal component analysis

## Clustering

# Context

In machine learning, we often encounter problems in high dimension, where closed-form solutions are not available, or where even if they are available, the necessary computation time is too large.

## Context

In machine learning, we often encounter problems in high dimension, where closed-form solutions are not available, or where even if they are available, the necessary computation time is too large.

**Example 1** : Computing the OLS estimator requires a matrix inversion, which is  $\mathcal{O}(d^3)$ .

$$\hat{\theta} = (X^T X)^{-1} X^T y \quad (26)$$

## Context

In machine learning, we often encounter problems in high dimension, where closed-form solutions are not available, or where even if they are available, the necessary computation time is too large.

**Example 2 :** The cancellation of the gradient of the objective function with logistic loss has no closed-form solution.



## Context

Instead, we often use **iterative** algorithm such as Gradient descent (GD) or Stochastic gradient descent (SGD). SGD is the standard optimization algorithm for large-scale machine learning because (brutal summary) :

- ▶ SGD is roughly  $n$  times faster than GD
- ▶ SGD's convergence is often satisfactory

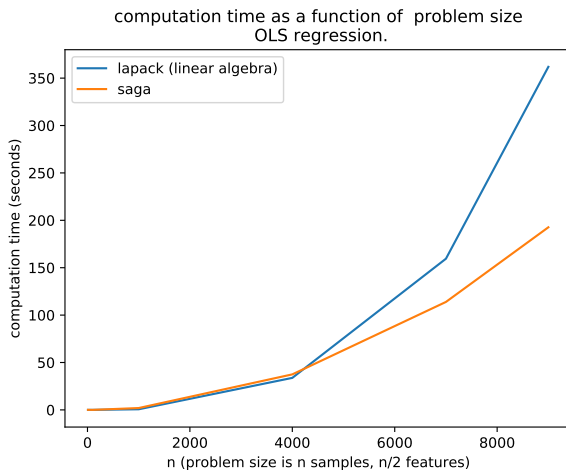
## Convergence speed

The properties and speed of convergence of GD and SGD are important properties that are extensively studied. They typically depend on :

- ▶ the convexity or strong convexity of the objective function  $f$  (often the empirical risk).
- ▶ the regularity (smoothness constant) of the objective function  $f$ .
- ▶ the statistical properties of the dataset

We will study some of these aspects during the practical sessions.

# SGD vs Lapack



## Gradient descent

We want to minimize a function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$ . **Gradient descent** iteratively performs the following update, until some criterion is met.

$$\theta \leftarrow \theta - \gamma \nabla f(\theta) \quad (27)$$

$\gamma$  is called the **learning rate**, and is a very important hyperparameter. We will study some methods to tune it in practical sessions, like line search.

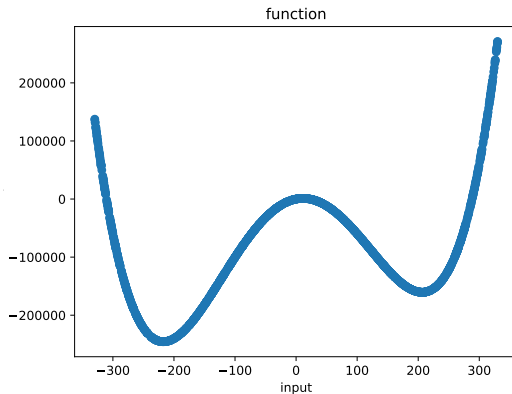
## First-order method

GD is called a first-order method because it make use of the first order derivative.

The direction  $-\nabla_{\theta} f(\theta)$  is the direction that minimizes the first order Taylor expansion of  $f$  in  $\theta$ .

## Local minima

GD is a greedy algorithm, and if  $f$  is non-convex, it might fall in local minima.



# Stochastic gradient descent

In machine learning, we often consider an objective function of the form

$$f(\theta) = \frac{1}{n} \sum_{i=1}^n l(y_i, g_{\theta}(x_i)) + \Omega(\theta) \quad (28)$$

## Batch gradient

In machine learning, we often consider an objective function of the form

$$f(\theta) = \frac{1}{n} \sum_{i=1}^n l(y_i, g_{\theta}(x_i)) + \Omega(\theta) \quad (29)$$

Gradient descent computes the **batch gradient** (also called **full gradient**) of  $f$ , which requires at least  $n$  calculations, and each calculation also has a complexity that depends on the dimension  $d$ . When  $n$  and  $d$  are large, this can be problematic.



# SGD

In machine learning, we often consider an objective function of the form

$$f(\theta) = \frac{1}{n} \sum_{i=1}^n l(y_i, g_{\theta}(x_i)) + \Omega(\theta) \quad (30)$$

SGD replaces the full gradient  $\nabla_{\theta} f(\theta)$  by :

$$\nabla_{\theta} \left[ l(y_i, g_{\theta}(x_i)) + \Omega(\theta) \right] \quad (31)$$

with  $i$  being **randomly sampled** in  $[1, n]$  (this is why it is called **stochastic**.)

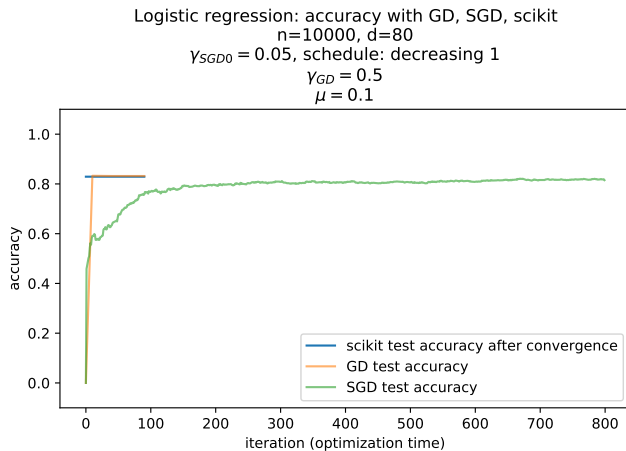


Figure – accuracy

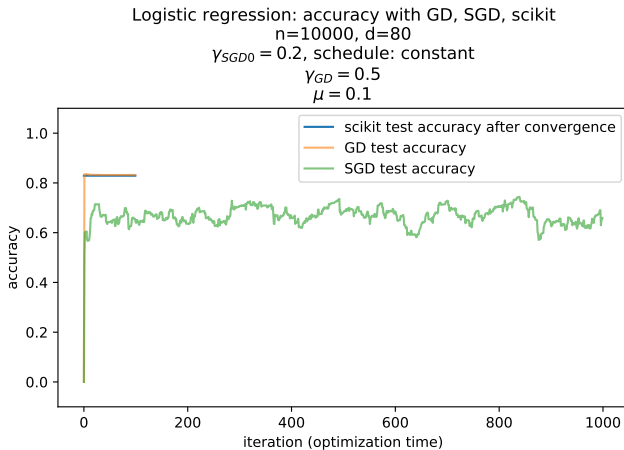


Figure – accuracy

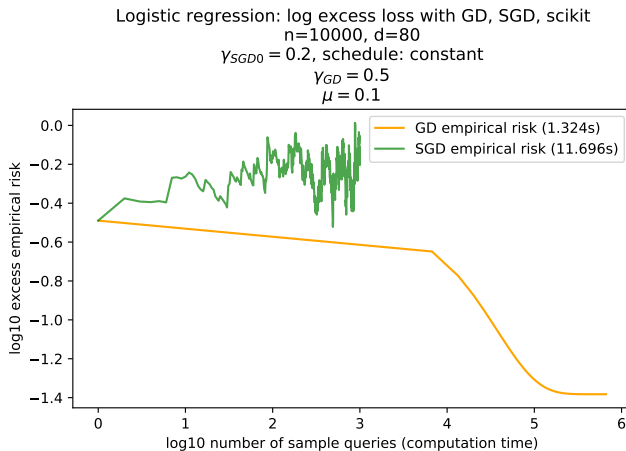


Figure – accuracy

## Scoring, multiclass problems, cross validation

- Regression

- Binary classification

- Multi-class classification

- Cross-validation

## Classification

- Problem statement

- Convexification of the risk and calibration

- Logistic regression

## Gradient algorithms

## Dimensionality reduction

- Principal component analysis

## Clustering

# Dimensionality reduction

We consider the space  $\mathcal{X}$  that contains the data, for either supervised or unsupervised learning. In machine learning, we often have  $\mathcal{X} \in \mathbb{R}^d$ .

- ▶ If  $d$  is large (e.g.  $\geq 10^4$ ), the algorithms that run on the data might become too slow to be used, as their algorithmic complexity depends on  $d$  (potentially in a quadratic or exponential way, curse of dimensionality)

# Dimensionality reduction

We consider the space  $\mathcal{X}$  that contains the data, for either supervised or unsupervised learning. In machine learning, we often have  $\mathcal{X} \in \mathbb{R}^d$ .

- ▶ If  $d$  is large (e.g.  $\geq 10^4$ ), the algorithms that run on the data might become too slow to be used, as their algorithmic complexity depends on  $d$  (potentially in a quadratic or exponential way, curse of dimensionality)
- ▶ **However**, often the data might actually occupy a **subspace** of lower dimension  $q$ , or it may be possible to project the data on such a subspace without losing too much information.
  - ▶ Working in a subspace of lower dimension might speed up the algorithms.
  - ▶ It may also allow visualization of the data.

# Main methods of dimensionality reduction

- ▶ **feature selection** : selecting a subset of the original dimensions.
- ▶ **feature extraction** : computing new features from the original features.



# Principal component analysis (PCA)

- ▶ PCA is a **linear feature extraction** technique.
- ▶ Points in  $\mathbb{R}^d$  are linearly projected on a well chosen affine subspace of  $\mathbb{R}^q$ , with  $q \leq d$ .

## Formalization as a (empirical) variance maximimisation problem

Without loss of generality, we assume the data are **centered**, which means that

$$\bar{x} = \sum_{i=1}^n x_n = 0 \in \mathbb{R}^d \quad (32)$$

We note  $X$  is the design matrix as in OLS. The **first principal component** is a vector  $w \in \mathbb{R}^d$ , with  $\|w\| = 1$ , such that  $\hat{Var}(w^T x)$  is maximal, where  $\hat{Var}$  denotes the empirical variance.

## Variance

$$\overline{w^T x} = w^T \bar{x} = 0 \quad (33)$$

Hence,

$$\begin{aligned} \hat{Var}(w^T x) &= \frac{1}{n-1} \sum_{i=1}^n ((w^T x)_i - \overline{w^T x})^2 \\ &= \frac{1}{n-1} \sum_{i=1}^n (w^T x_i)^2 \end{aligned} \quad (34)$$

## Variance maximisation problem

We can then formulate the problem as finding  $w$ ,  $\|w\| = 1$  such that

$$\sum_{i=1}^n (w^T x_i)^2 \quad (35)$$

is maximal.

## First principal component

We look for  $w$ ,  $\|w\| = 1$  such that

$$\sum_{i=1}^n (w^T x_i)^2 \tag{36}$$

is maximal.

### Proposition

$w$  is the eigenvector of  $X^T X$  with largest eigenvalue  $\lambda_{\max}$ .

## First principal component

We look for  $w$ ,  $\|w\| = 1$  such that

$$\sum_{i=1}^n (w^T x_i)^2 \quad (37)$$

is maximal.

### Proposition

$w$  is the eigenvector of  $X^T X$  with largest eigenvalue  $\lambda_{\max}$ .

**Exercise 4:** Show the proposition.

## Reconstruction error

Alternately, we can formulate the problem as a **reconstruction error minimization**.

$$\mathbb{R}^d = \text{Vect}(w) \oplus \text{Vect}(w)^\perp \quad (38)$$

and if  $\|w\| = 1$ ,

$$\begin{aligned} \forall x \in \mathbb{R}^d, \|x\|^2 &= \|(x^T w)w\|^2 + \|x - (x^T w)w\|^2 \\ &= (x^T w)^2 + \|x - (x^T w)w\|^2 \end{aligned} \quad (39)$$

## Reconstruction error

We can formulate the problem as a **reconstruction error minimization**. If  $\|w\| = 1$ ,

$$\begin{aligned}\forall x \in \mathbb{R}^d, \|x\|^2 &= \|(x^T w)w\|^2 + \|x - (x^T w)w\|^2 \\ &= (x^T w)^2 + \|x - (x^T w)w\|^2\end{aligned}\tag{40}$$

Hence,

$$\begin{aligned}\sum_{i=1}^n \|x_i\|^2 &= \sum_{i=1}^n (x_i^T w)^2 + \sum_{i=1}^n \|x_i - (x_i^T w)w\|^2 \\ &= \hat{Var}(w^T x) + \sum_{i=1}^n \|x_i - (x_i^T w)w\|^2\end{aligned}\tag{41}$$



## Reconstruction error

$$\sum_{i=1}^n \|x_i\|^2 = \hat{Var}(w^T x) + \sum_{i=1}^n \|x_i - (x_i^T w)w\|^2 \quad (42)$$

We can see  $\sum_{i=1}^n \|x_i - (x_i^T w)w\|^2$  as a **reconstruction error**, when the data are projected on  $\text{Vect}(w)$ .

Maximizing the variance of the projections is equivalent to minimizing the reconstruction errors obtained by projection.

## Several principal components

Most of the time, we project the data on several principal components.

- ▶ 1] compute the first principal component  $w_1$
- ▶ 2] project the data on  $\text{Vect}(w_1)^\perp$
- ▶ 3] start again on the projected data

## Reconstruction error

The interpretation stays the same. If  $p_F(x)$  is the projection of  $x$  on the subspace spanned by the principal components :

$$||x||^2 = ||p_F(x)||^2 + ||x - p_F(x)||^2 \quad (43)$$

The principal components are the largest eigenvectors of  $X^T X \in \mathbb{R}^d$ ,  $d$  with norm 1. They are **orthogonal** to each other.

## Inertia

We can again define an inertia :

$$I_F = \sum_{i=1}^n \|x - p_F(x)\|^2 \quad (44)$$

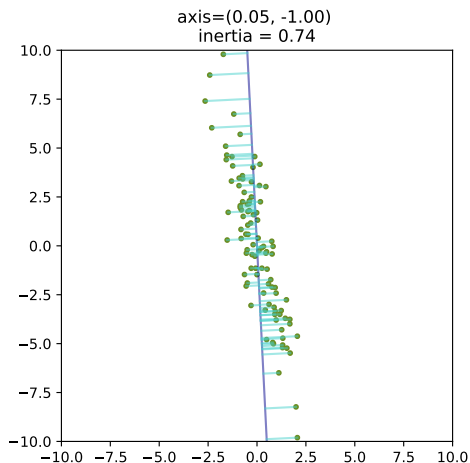
We look for the subspace that minimizes the inertia  $I_F$ .

# Inertia

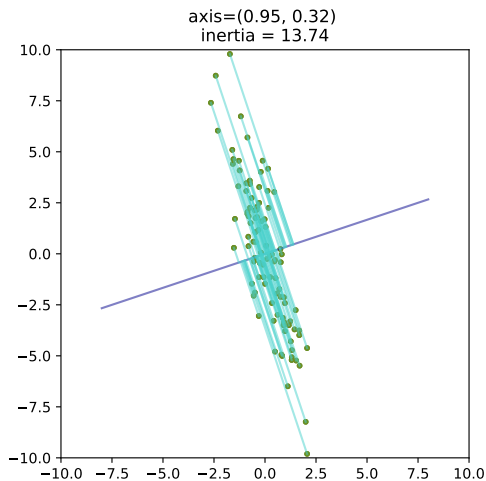
## Exercise 5 : No inertia

In what situations could we have  $I_F = 0$ ?

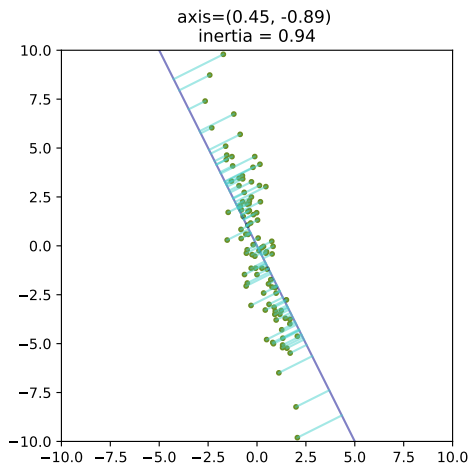
# Inertia



# Inertia



# Inertia





## Iris dataset

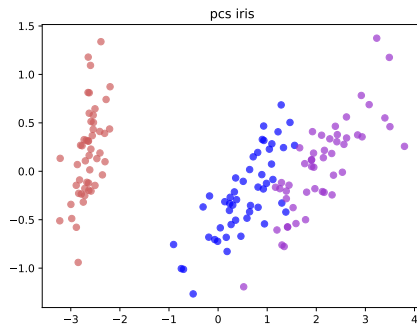


Figure – PCA performed on the iris dataset. We see that the principal components are able to separate the data.

In this paper, astrophysicists use PCA in order to test a new star temperature prediction method [Bermejo et al., 2013]

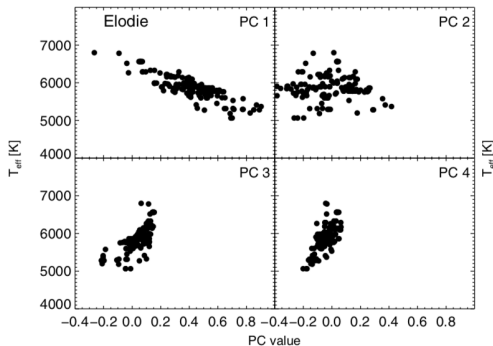
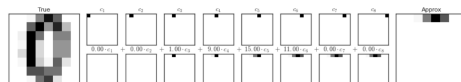


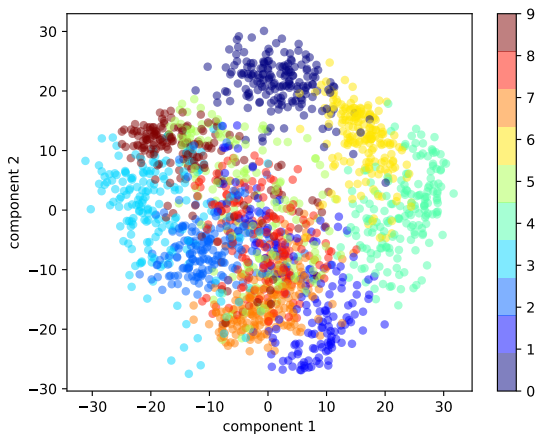
Figure – PCA used in order to predict temperature.

## PCA on digits

- ▶ We can perform the PCA on a dataset consisting in  $8 \times 8$  pixels images of digits, in order to see if the PCA allows a visualization of some structure in the data.



## PCA on digits



## PCA on digits : reconstruction

With 8 principal components, we can monitor the reconstruction of the images (originally in 64 dimensions)

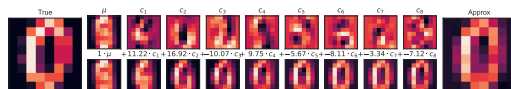


Figure – Reconstruction of 0

<https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html>

## PCA on digits : reconstruction

With 8 principal components, we can monitor the reconstruction of the images (originally in 64 dimensions)

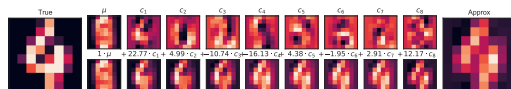


Figure – Reconstruction of 4

<https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html>

## Explained variance

As always, a natural question is : what is a relevant number of principal components ?

A common quantity that is used is **explained variance**. Each component  $w_k$  carries a percentage of the total variance of the data.

$$\frac{\hat{Var}(w_k^T x)}{\sum_{j=1}^d \hat{Var}(x^j)} \quad (45)$$

where  $\hat{Var}(x^j)$  is the variance of the component  $j$ .

$$\hat{Var}(x^j) = \sum_{i=1}^n (x_i^j)^2 \quad (46)$$

## Number of components

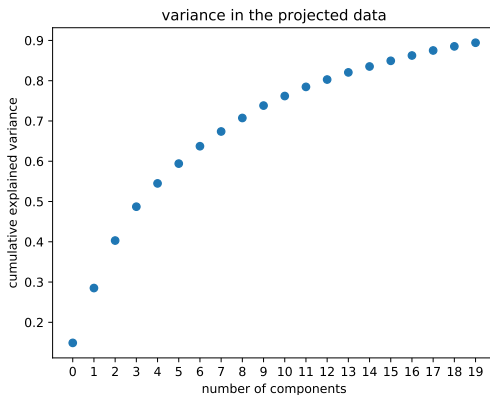
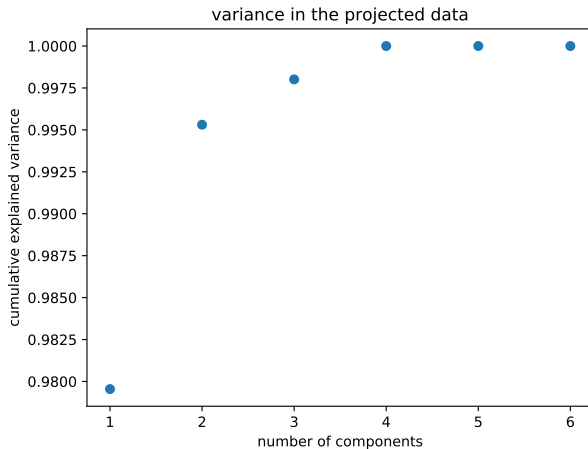


Figure – Variance of the projected data as a function of the number of components (digits dataset)



## Number of components

Exercise 6: What happens with this dataset?



## Number of components

**Conclusion :** PCA can help determine whether some components carry no information in the data.

# Shortcomings of PCA

PCA is sensitive to :

- ▶ outliers
- ▶ initial data scaling

# Unsupervised learning

From a number of samples  $x_i$ , you want to retrieve information on their structure : **modelisation**.

# Unsupervised learning

From a number of samples  $x_i$ , you want to retrieve information on their structure : **modelisation**. The three main unsupervised learning problems are :

- ▶ clustering
- ▶ density estimation
- ▶ dimensionality reduction

# Clustering

**Clustering** consists in partitioning the data.  $\forall i, x_i \in \mathcal{X}^n$ .

$$D_n = \{(x_i)_{i \in [1, \dots, n]}\} \quad (47)$$

# Clustering

**Clustering** consists in partitioning the data.  $\forall i, x_i \in \mathcal{X}^n$ .

$$D_n = \{(x_i)_{i \in [1, \dots, n]}\} \quad (48)$$

A **partition** is a set of  $K$  subsets  $A_k \subset D_n$ , such that



$$\cup_{k \in [1, \dots, K]} A_k = D_n \quad (49)$$



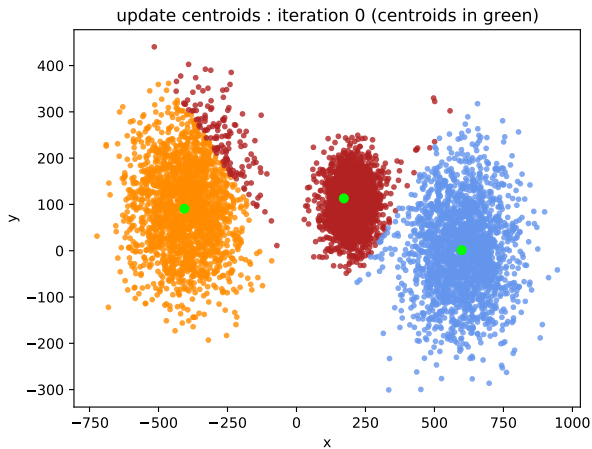
$$\forall k \neq k', A_k \cap A_{k'} = \emptyset \quad (50)$$

## Partitions

- ▶ **Example 1** :  $A$  is the set of even integers,  $B$  the set of odd integers. Is  $(A, B)$  a partition of  $\mathbb{N}$ ?
- ▶ **Example 2** :  $C$  is the set of multiples of 2,  $D$  the set of multiples of 3. Is  $(C, D)$  a partition of  $\mathbb{N}$ ?



## Example : partition of data



## Applications of clustering

Example applications :

- ▶ spam filtering [Sharma and Rastogi, 2014, ]
- ▶ fake news identification  
[Hosseinimotlagh and Papalexakis, 2018, ]
- ▶ marketing and sales
- ▶ document analysis [Zhao and Karypis, 2002, ]
- ▶ traffic classification [Woo et al., 2007, ]

Some of these applications can be considered to be semi-supervised learning.

## Vector quantization

**Vector quantization** consists in computing **prototypes**

$\Omega = (\omega_k)_{k \in [1, \dots, K]} \in \mathcal{X}^K$  that represent the data well.

This implies that a **metric** is defined on  $\mathcal{X}$ .

Most often, this is interesting if  $K \ll n$ .

## Voronoi subsets

We assume a loss  $L$  is defined on  $\mathcal{X}$ . The Voronoi subset of  $\omega$  is defined as

$$V(\omega) = \{x \in D_n, \arg \min_{\omega' \in \Omega} L(\omega', x) = \omega\} \quad (51)$$

- ▶ We assume that  $\arg \min$  returns one single element.
- ▶ The Voronoi subsets form a partition of  $D_n$ .

## Distortion

To measure the quality of a Voronoï partition, we introduce the **distortion**  $R(\Omega)$ .

For each  $x$ , we note  $h_{\Omega}(x) = \arg \min_{\omega' \in \Omega} L(\omega', x)$ .

$$R(\Omega) = \frac{1}{n} \sum_{i=1}^n L(x_i, h_{\Omega}(x_i)) \quad (52)$$

## Distortion

For each  $x$ , we note  $h_{\Omega}(x) = \arg \min_{\omega' \in \Omega} L(\omega', x)$ .

$$\begin{aligned} R(\Omega) &= \frac{1}{n} \sum_{i=1}^n L(x_i, h_{\Omega}(x_i)) \\ &= \frac{1}{n} \sum_{\omega \in \Omega} \sum_{x \in V(\omega)} L(x_i, h_{\Omega}(x_i)) \\ &= \frac{1}{n} \sum_{\omega \in \Omega} V_{\Omega}(\omega) \end{aligned} \tag{53}$$

with

$$V_{\Omega}(\omega) = \sum_{x \in V(\omega)} L(x_i, h_{\Omega}(x_i)) \tag{54}$$

## Minimum of distortion

We want to find the prototypes for which the distortion is **minimal**.

- ▶ The set of prototypes minimizing distortion might not be unique.
- ▶ We need to tune  $K$  (number of prototypes).

# Vector quantization techniques

- ▶ K-means
- ▶ Growing neural gas (GNG)
- ▶ Self-organizing maps



# K-means clustering

- ▶  $\mathcal{X} = \mathbb{R}^d$ .
- ▶  $L(x, y) = \|x - y\|^2$ .

## Objective function

With

- ▶  $\Omega = \{\omega_1, \dots, \omega_K\} \in \mathbb{R}^{K,d}$ .
- ▶  $z_i^k = 1$  if  $x_i$  is assigned to  $\omega_k$ ,  $z_i^k = 0$  otherwise.  
 $z = (z_i^k) \in \mathbb{R}^{n,K}$ .

we define the objective function

$$J(\Omega, z) = \sum_{i=1}^n \sum_{k=1}^K z_i^k \|x_i - \omega_k\|^2 \quad (55)$$

It is also called the **inertia**.

## K-means algorithm

**Result:**  $\Omega \in \mathbb{R}^{K,d}$

$\Omega \leftarrow$  Random initialization;

$z = M$  where  $M$  is the  $n \times K$  matrix with 0's;

**while** *Convergence criteria is not satisfied* **do**

- | a] Greedily minimize  $J$  with respect to  $z$ ;
- | b] Minimize  $J$  with respect to  $\Omega$ ;

**end**

return  $\Omega$

**Algorithm 1:** K-means (Lloyd algorithm)

## Stopping criterion

To stop the algorithm, the norm of the difference between  $\Omega_t$  and  $\Omega_{t+1}$  must be smaller than a given tolerance. (e.g.  $1e^{-4}$ ). Here it is a norm between matrix (Frobenius norm) :

$$\|A\|_F = \sqrt{\sum_{i=1}^n A_{ij}^2} \quad (56)$$

## Minimization

We focus on step b]. How can we minimize  $J$  with respect to  $\Omega$ ?

## Minimization

### Exercise 7 : Convexity :

Show that  $J(\Omega, z)$  is convex with respect to  $\Omega$ .

$$J(\Omega, z) = \sum_{i=1}^n \sum_{k=1}^K z_i^k \|x_i - \omega_k\|^2 \quad (57)$$

Hence, to minimize  $J$  with respect to  $\Omega$ , we just need to cancel the gradient.

## Minimization

### Exercise 8 : Gradient :

Compute the gradient of  $J(\Omega, z)$  with respect to  $\Omega$  and deduce the minimizer  $\Omega^*$ .

- ▶  $z$  is fixed
- ▶ we can see  $\Omega$  has a vector of  $\mathbb{R}^{Kd}$ .

# Convexity

Is  $J(\Omega, z)$  convex in  $z$ ?



# Convexity

Is  $J(\Omega, z)$  convex in  $z$ ?

**No**, as  $z$  is not even defined on a convex set.

Hence, the function might have **local minima**.

## Suboptimal clustering

**Exercise 8 : Local minima** : propose a setting (dataset, initialization) for which the algorithm outputs a bad set of centroids.

## Suboptimal clustering

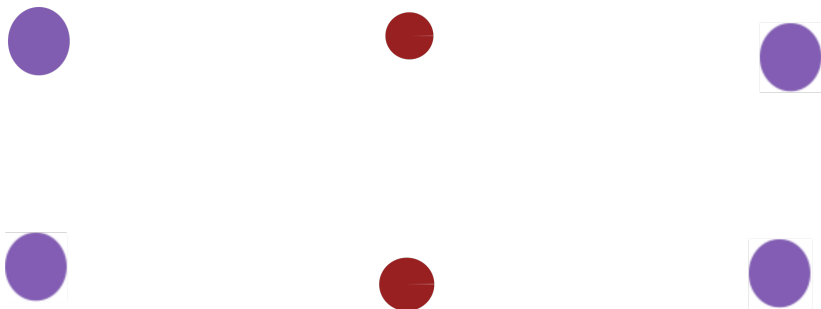


Figure – Initialization of centroids in red.

## Random initialization

Hence, the result of K-means strongly depends on the initial position of the centroids.

A common approach is to restart the algorithm several times and select the result with lowest inertia.




## Drawbacks of inertia minimization

K-means is based on the minimization of the inertia and hence on the euclidean distance. **However**, in some contexts, the euclidean distance is not the adapted metric.

https:

[//scikit-learn.org/stable/modules/clustering.html](https://scikit-learn.org/stable/modules/clustering.html)

## References I

-  Bermejo, J. M., Ramos, A. A., and Prieto, C. A. (2013). Astrophysics A PCA approach to stellar effective temperatures. 95 :1–9.
-  Hosseinimotlagh, S. and Papalexakis, E. E. (2018). Unsupervised content-based identification of fake news articles with tensor decomposition ensembles. *Proceedings of the WSDM MIS2 : Misinformation and Misbehavior Mining on the Web Workshop*, pages 1–8.
-  Sharma, A. and Rastogi, V. (2014). Spam Filtering using K mean Clustering with Local Feature Selection Classifier. *International Journal of Computer Applications*, 108(10) :35–39.

## References II



Woo, D. M., Park, D. C., Song, Y. S., Nguyen, Q. D., and Tran, Q. D. N. (2007).

Terrain classification using clustering algorithms.

*Proceedings - Third International Conference on Natural Computation, ICNC 2007*, 1 :315–319.



Zhao, Y. and Karypis, G. (2002).

Evaluation of hierarchical clustering algorithms for document datasets.

*International Conference on Information and Knowledge Management, Proceedings*, (August 2002) :515–524.