

Data Science - Machine Learning

Msc Supply Chain

Guillaume Metzler

Université Lumière Lyon 2
Laboratoire ERIC, UR 3083, Lyon

guillaume.metzler@univ-lyon2.fr

Spring

What is Data Science ?

What is Machine Learning ?

Introduction

What is Machine Learning ? I

Machine Learning is a sub-field of *artificial intelligence*. It belongs in the frontier of Computer and Applied Mathematics (like Statistics and Optimization or Linear Algebra). This topic is also part of the *Data Science* since it requires to collect/clean et analyze a collection of data in order to extract some important information for the task at end.

What is Machine Learning? II

This definition shows that several ingredients are important in Machine Learning in order to solve a task

- **The task** : what is the problem that I want to solve using the computer science tools? It is very important to clearly identify the task you are dealing with.
- **The data** : which data are available and will be used to develop an artificial intelligence that will help us to solve the task
- **The algorithm** : According to the task I want to solve, we have to find the correct formulation of it using the mathematical tools, develop an algorithm to find the solution.

What is Machine Learning? III

**What are for you
the two main phases
in Machine Learning?**

What is Machine Learning? IV

Learning/Training

This first part consists in learning the parameters of the model previously defined and which aims to solve our problem.

This phase requires a collection of data and, more precisely, a subset of the available data.

Tasks can be very different, such as classifying documents, distinguish between cats and dogs in a set of images or predict the value of stock markets.

This can be seen as a preparation phase where the model is learned and tested before being delivered to the customer.

What is Machine Learning? V

Test

We can also see it as the phase of putting our model **in production** but ... virtually! Because we need to be sure of its performance.

The system has been previously developed, validated and tested and it can then be deployed for practical usages.

During this phase, the model is constantly monitored to ensure that its effectiveness remains in line with the observations made when it was first trained. Sometimes, updates are necessary (re-training the model on more recent data) in order to maintain its effectiveness (this is known as *learning by doing*).

Data : cats and Dogs I

Let us have a look at a set of data



Data : cats and Dogs II

Images : these are matrices (one matrix with each entry corresponding to a pixel value) of numbers where each matrix represents a color intensity for a given pixel (superposition of three matrices for three color channels).

$$R = \begin{bmatrix} 1 \\ 1 \\ 0.9 \\ 0 \\ \dots \end{bmatrix}, \quad G = \begin{bmatrix} 1 \\ 1 \\ 0.1 \\ 0.3 \\ \dots \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0.7 \\ \dots \end{bmatrix}.$$

Data : cats and Dogs III

They are at the heart of learning algorithms and can therefore be of different kinds :

- gross (non transformed)
- transformed or created
- learned

We will, however, see a major problem linked to these data and which is specific to learning : **these data come from an unknown distribution, which opens up new avenues of research in Machine Learning!**

Let us now have a look how they can be used in practice !

Data : cats and Dogs IV

Our image descriptors are not invariant and may depend on a number of parameters or data collection conditions.

Exemples : shapes or patterns - size - colors

- the size varies with the notion of the distance to the image
- the shapes are complex to be represented
- the color can vary according to the light.

This means that the same image, taken under different conditions, can have different characteristics. (There are, however, algorithms for identifying identical elements in "different" images, such as *SIFT*).

Data : cats and Dogs V

It's rarely enough, and this is even truer in an industrial context, to work with raw data. We often seek to transform them in order to create new variables or to transform basic variables :

- With the help of business knowledge and experts who know the important information or discriminating factors.
- Using mathematical transformations or learning models : neural networks, auto-encoders, metric learning, kernels,

Now let's look at this data in practical terms, *i.e.* how to use it. If we adopt an algorithmic presentation, we have :

- **Input** : our images (represented as a vector)
- **Output** : the class of the image (dog or cat)
- **Training set** : the set of images + label

Data : cats and Dogs VI



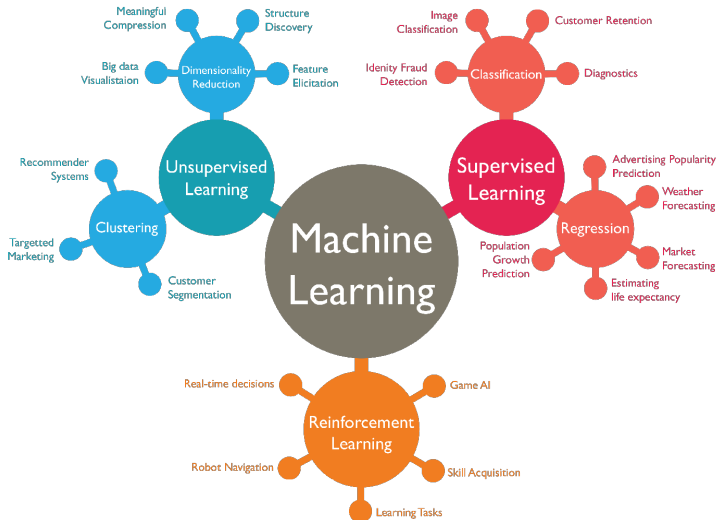
Data : cats and Dogs VII

Once training is complete, it's a good idea to find out what happens if our model can be put into production. So we're going to reproduce production conditions.

We now want to know whether our model performs well on new data? Is it able to identify the kind of images not encountered during its training phase?



Different context in Machine Learning I



Different context in Machine Learning II

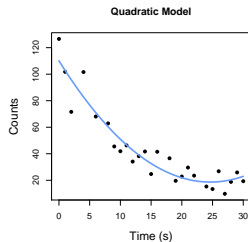
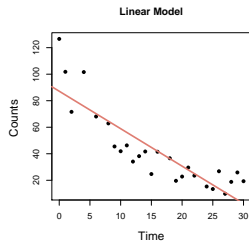
Machine Learning can be separated into three main categories

- **Supervised Learning** : includes a large number of classification algorithms (SVM - k -NN, ...), as well as similarity learning (Metric Learning, Transfer Learning) and regression tasks
- **Unsupervised Learning** : These include clustering algorithms (K-means or HCA) as well as unsupervised transfer learning.
- **reinforcement Learning** : Learning by doing experiments.

Different context in Machine Learning III

Supervised Learning : Regression

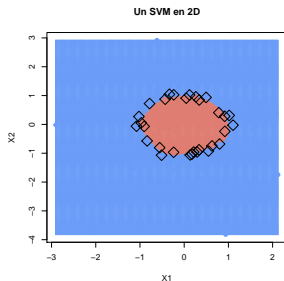
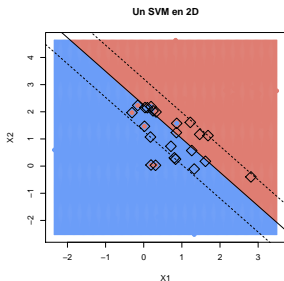
The system has access to a set of data described both by descriptors and by a label provided by an *oracle*. The aim will be to learn a "rule" from the descriptors, enabling the prediction of the value of the data and also that of new data.



Different context in Machine Learning IV

Supervised Learning : Classification

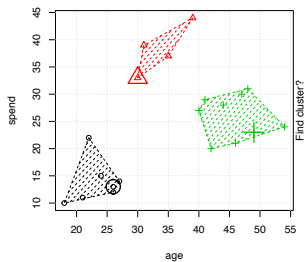
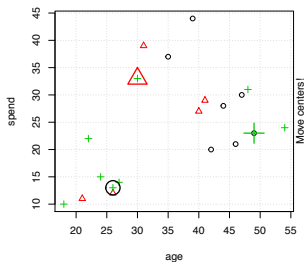
The system has access to a set of data described both by descriptors and by a label provided by an "oracle". The objective will be to learn a "rule" from the descriptors, allowing the prediction of the label of the data, but also that of new data.



Different context in Machine Learning V

Unsupervised Learning : Clustering

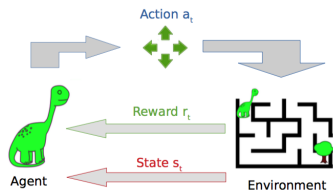
No labels are available during the learning phase. The main idea is to group together examples showing **similarities** or **resemblances** to create groups (**clustering**). This can also be used for **density estimation** or to learn a **new data representation**.



Different context in Machine Learning VI

Reinforcement Learning

The model must learn to perform the right **actions** according to the **context**, *i.e.* it must make the right **decisions** according to the **observations made**. This is done using a system of **rewards**. In this type of learning, there's no data to tell it which actions are optimal in any given situation. It is left to discover for itself what it should do, using a trial-and-error method.



Data and Representation I

The previous examples show that data is at the heart of machine learning algorithms. Data can come in a variety of forms and formats, be structured or unstructured, and sometimes contain anomalies or missing values...

In *Machine Learning*, it is customary to view this data as a set of m examples of the same nature. The most natural representation chosen is in the form of a vector \mathbf{x} in dimension d : $\mathbf{x} = (x_1, x_2, \dots, x_d)$ where $\mathbf{x} \in \mathbb{R}^d$

Data and Representation II

the chosen representation, *i.e.* the meaning of the vector, will depend of the nature of the data.

Images can be represented by vectors (after transformation, CNN and *vectorization*).

genetic data for a gene sequence (dimension = length of the gene, vector = encoding)

sounds as sequences of signals (dimension = frequency, vector of amplitudes)

textual documents as a set of words (dimension = number of words in the corpus, vector of occurrences)

metadata : authors, dates, ...

Real data can be very complex, with a high degree of redundancy and variability.

Normalize the data I

In addition to the representation aspect, we also need to take an interest in the values taken by each descriptor in our data, so as not to give casual importance to a subset. This could prove particularly harmful for certain algorithms based on the notion of distance!

Exemple : Let $\mathbf{x} = (x_1, x_2)$, $\mathbf{x}' = (x'_1, x'_2)$ where $x_1, x'_1 \in [0, 1]$ and $x_2, x'_2 \in [500, 1000]$, the distance

$$d(\mathbf{x}, \mathbf{x}') = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2}$$

is close to the distance between x_2 and x'_2 .

One solution is to normalize the data, which in most cases increases algorithm performance by giving the same weight to each variable.

Normalize the data II

There are several data normalization processes, the best known/used of which are as follows :

- **scaling** or **min-max normalization** so that values lie within $[0, 1]$ or $[-1, 1]$

$$x = \frac{x - \min(x)}{\max(x) - \min(x)} \quad \text{where} \quad x = 1 - 2 \times \frac{x - \min(x)}{\max(x) - \min(x)},$$

- **standardization** : we transform each variable such they follow a normal distribution

$$x = \frac{x - \mu(x)}{\sigma(x)},$$

Normalize the data III

- **normalization** : divide each vector by its norm so that $\|\mathbf{x}\| = 1$

$$\mathbf{x} = \frac{\mathbf{x}}{\|\mathbf{x}\|}.$$

An important assumption

As mentioned earlier, we never have access to the whole distribution when learning a model, but only to a **small sample**. To ensure that the model learned on these data is capable of generalizing to new data, we need to assume that the new data are *i.i.d.*

Définition 1.1: Distribution of the data

A training set S is said to be *i.i.d.* if all the examples come from the same probability distribution **unknown** \mathcal{D} and are mutually independent.

→ A fundamental assumption in Machine Learning

Data and Dimension

If we have enough examples in a space of "limited" dimension, we show that our models are capable of generalization, since the data allow us to correctly approximate the distribution.

Quid in the other case? Curse of dimensionality!

This results in the appearance of phenomena that do not appear in reasonable dimensions : such as volumes that become abnormally small, which can have consequences for certain algorithms.

→ Reduce the space of representation of the data !

Learning

Learning I

Let $\mathcal{D} = \mathcal{X} \times \mathcal{Y}$ the unknown distribution of the data. The space \mathcal{X} is called the **input space** or **feature space**, in general $\mathcal{X} \subset \mathbb{R}^d$. The space \mathcal{Y} is **the label space** or even **the output space** :

- $\mathcal{Y} = \emptyset$: unsupervised learning
- $\mathcal{Y} = \mathbb{R}$: regression
- $\mathcal{Y} = \{0, 1\}$: binary classification
- $\mathcal{Y} = \{1, \dots, C\}$: multi-class classification
- $\mathcal{Y} = \{0, 1\}^C$: multi-label classification

Objective : find an algorithm \mathcal{A} , which generate an hypothesis h able to solve the task.

Problem : in practise \mathcal{D} is unknown.

Learning II

We only have access to a finite sample $S = \{\mathbf{x}_i, y_i\}_{i=1}^m \underset{i.i.d.}{\sim} \mathcal{D} = \mathcal{X} \times \mathcal{Y}$

where $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} \subset \mathbb{Z}$ (classification) or $\subset \mathbb{R}$ (regression).

In the following we suppose $\mathbf{x}_i \in \mathbb{R}^d$ and we denote

$$X = (\mathbf{x}_1, \dots, \mathbf{x}_m) = \begin{pmatrix} \mathbf{x}_{11} & \cdots & \mathbf{x}_{1d} \\ \vdots & \ddots & \vdots \\ \mathbf{x}_{m1} & \cdots & \mathbf{x}_{md} \end{pmatrix}.$$

We aim to find an **hypothesis** h , such that $h(\mathbf{x}) = y$, we want it to be powerfull on our sample S but also on new instances. (\mathbf{x}, y) .

How to learn such an hypothesis?

Learning III

Let us come back to our classification task using images of cats and dogs.



The aim is to minimize the algorithm's errors, or more generally, to minimize what is known as a **risk**.

Learning IV

In order to learn and find the best h^* hypothesis, we need to define a criterion that quantifies the quality of the learned hypothesis. This criterion will take the form of **Performance measure** (which we will then seek to maximize) or, more traditionally, **Risk** (which we will seek to minimize). Ideally, the aim is to minimize the risk over the entire data distribution, known as the **Real Risk**.

The real risk $\mathcal{R}(h)$, also called **risk in generalization** of a hypothesis h , corresponds to the average error (hence the expectation) of the hypothesis h over the entire distribution.

$$\mathcal{R}(h) = \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} [\mathbb{1}_{\{h(\mathbf{x}) \neq y\}}].$$

Learning V

In supervised learning, the aim is to learn a hypothesis h that will minimize the real risk, *i.e.* the risk on our entire distribution. Unfortunately, this real risk $\mathcal{R}(h)$ cannot be calculated, given the unknown nature of \mathcal{D} .

We can only estimate it or measure it on our training set. This estimate is called **Empirical Risk**, denoted $\mathcal{R}_S(h)$.

From a practical point of view, it's this quantity that we're going to try to minimize ... well ... not quite, as we'll see later! Let's take a look at its definition.

Learning VI

In unsupervised learning : This notion of risk is also present, but no longer refers to an error term.

It is simply used to refer to the quantity you wish to minimize in a minimization problem, e.g. a sum of variances in the case of clustering, or a distance between distributions in the case of unsupervised domain adaptation.

Let us come back to the definition of risk

Learning VII

Définition 2.1: Empirical risk

Let $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ be a training set. The empirical risk $\mathcal{R}_S(h)$, also known as the risk of error, of a hypothesis h corresponds to the average error on our training set, or the expectation of this error on S .

$$\mathcal{R}_S(h) = \mathbb{E}_{(\mathbf{x}, y) \sim S} [\mathbb{1}_{\{h(\mathbf{x}) \neq y\}}] = \frac{1}{m} \sum_{i=1}^m \mathbb{1}_{\{h(\mathbf{x}_i) \neq y_i\}} = \mathbb{P}[h(\mathbf{x}) \neq y].$$

Learning VIII

In this case, this empirical risk measures the probability that the h hypothesis is wrong in the prediction made for the \mathbf{x} example.

But minimizing errors (in a binary way) is a difficult problem for (NP-hard) algorithms. So we rarely try to minimize the error rate as is. In practice, we're more likely to seek to minimize a risk based on a surrogate for the error rate, through the use of **loss functions**.

Loss functions I

This notion of risk of error is specific to the so-called *0-1 loss* (correct prediction 0, prediction error 1). However, these are not the only losses used, and the latter is only **very rarely used in practice**.

Définition 2.2: Loss function

A loss function ℓ is a function $\mathcal{H} \times \mathcal{Z} \rightarrow \mathbb{R}^+$ which is used to measure the disagreement between the prediction made by the hypothesis h , $h(\mathbf{x})$ and the value to be predicted y . \mathcal{H} is called hypothesis space.

Loss functions II

Classification error or 0-1 loss

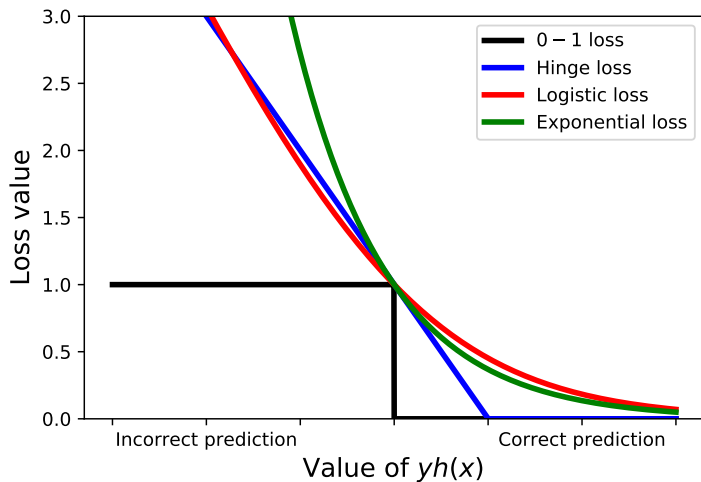
$$\ell(h(\mathbf{x}), y) = \begin{cases} 1 & \text{if } h(x) \times y < 0, \\ 0 & \text{else.} \end{cases}$$

This loss has a number of drawbacks : it is not convex, it is not derivable, and is therefore of little interest for *optimization algorithms*.

→ use of **surrogate**, so-called substitution functions, from 0-1 loss which have the advantage of being **convex** and sometimes even **differentiable**..

Different surrogates will be used depending on the algorithm you wish to use. These overrides are presented as upper bounds of the 0-1 loss.

Loss functions III



Loss functions IV

- **the hinge loss** : it is particularly useful for learning models such as *SVM* (classification or regression)

$$\ell(h(\mathbf{x}), y) = \max(0, 1 - yh(\mathbf{x})),$$

- **the logistic loss** : encountered when using a textitlogistic regression model (classification)

$$\ell(h(\mathbf{x}), y) = \frac{1}{\ln(2)} \ln(1 + \exp(-yh(\mathbf{x}))),$$

- **the exponential loss** : met in the context of boosting *boosting*

$$\ell(h(\mathbf{x}), y) = \exp(-yh(\mathbf{x})).$$

A new version of the risk

Définition 2.3: Empirical and real risk

The real risk of a hypothesis h according to a loss ℓ , noted $\mathcal{R}_S^\ell(h)$ is defined as the average value of the loss on the distribution. $\mathcal{R}_S(h)$,

$$\mathcal{R}_S^\ell(h) = \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} [\ell(h(\mathbf{x}) \neq y)].$$

Let $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$. The empirical risk $\mathcal{R}_S(h)$ of a hypothesis h corresponds to the average error of ℓ on S .

$$\mathcal{R}_S^\ell(h) = \mathbb{E}_{(\mathbf{x}, y) \sim S} [\ell(h(\mathbf{x}) \neq y)] = \frac{1}{m} \sum_{i=1}^m \ell(h(\mathbf{x}_i) \neq y_i).$$

What is a good model ?

Now that we have data S , a loss ℓ and an algorithm, how do we know if our model is performing well ?

To do this, we'll try to minimize the empirical risk $\mathcal{R}_S(h)$

$$\mathcal{R}_S(h) = \frac{1}{m} \sum_{i=1}^m \ell(h(\mathbf{x}_i), y).$$

and try to estimate the real risk $\mathcal{R}(h)$ or risk in generalization

$$\mathcal{R}(h) = \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} [\ell(h(\mathbf{x}), y)].$$

But we don't know the distribution... so we'll use a validation process !

Cross validation I

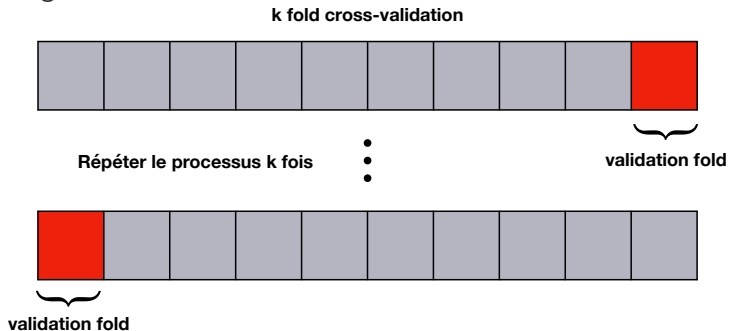
It's a question of conserving a part of the training set data that won't be used to learn the parameters of our model ; we'll use them to make an initial evaluation (training - validation) but we can do even better.

Définition 2.4: k -fold Cross-Validation

This is an algorithmic means of estimating the generalization performance of a given algorithm on unknown data. The principle is simple : we divide our data into k groups and use $k - 1$ groups to learn our model, with the last group used to evaluate it.

Cross validation II

We run a series of k experiments using classical validation, but we change the training and validation for each run.



In the end, our algorithm will therefore be evaluated on all available data, enabling a more reliable estimate.

Cross validation III

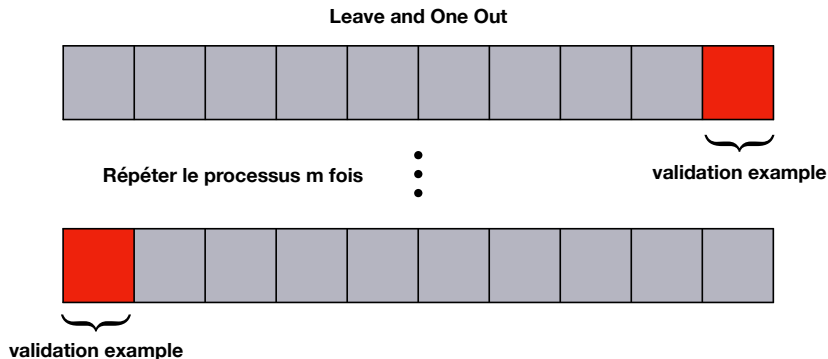
Mean error in CV.

The evaluation of $\frac{1}{k} \sum_{j=1}^k \mathcal{R}_{S_j}(h)$, called the mean error in cross-validation, quantifies how well our algorithm is able to generalize on new data from the same distribution.

It's therefore a good indicator of whether or not we can potentially put our algorithm into production. We'll see that this isn't its only use... It would also be good to be able to test our procedure on data not yet encountered by the model.

Leave and One Out

The principle remains the same, but this time we use a single example to validate each run, so we have to perform a larger number of training runs (m instead of k).



We retain the model with the best results on average over the m -folds used for validation.

And then... ?

We have S data, a ℓ loss and an algorithm and process for selecting the best model... so, a priori, we have everything we need to get started (modulo the presentation of the models).

Actually, no... but first there's one last thing to look at/study! What happens if the error observed on the training set is significantly different from the error observed in validation ?

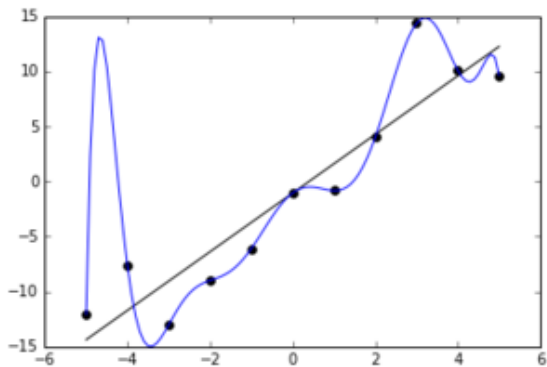
- if the validation error is smaller than the training set error,
- or conversely, if the validation error is greater than the training set error.

Overfitting I

Définition 2.5: Overfitting

In statistics, overfitting occurs when the model focuses so much on the data that it also tries to learn the noise present. This phenomenon occurs when the learned model is too complex, or when the training set is too small in relation to the number of model parameters (degrees of freedom).

Overfitting II

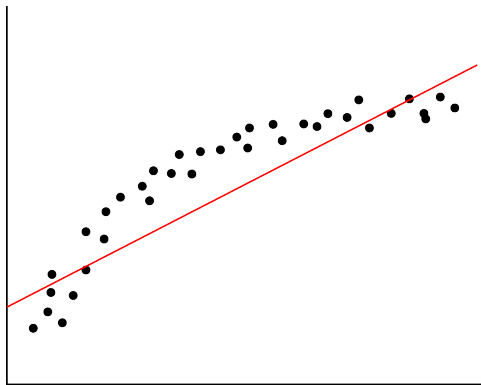


Underfitting I

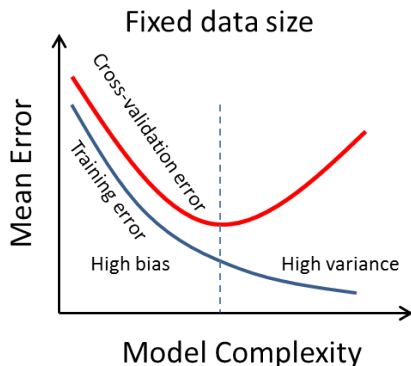
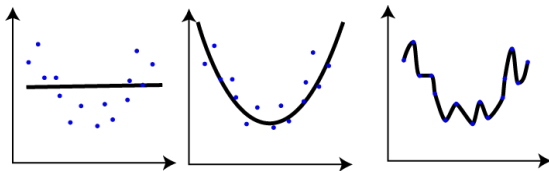
Définition 2.6: Underfitting

Underfitting occurs when the model used is unable to capture the trend present in the data. This occurs especially when the model used is far too simple, such as using a linear model to approximate a quadratic curve.

Underfitting II



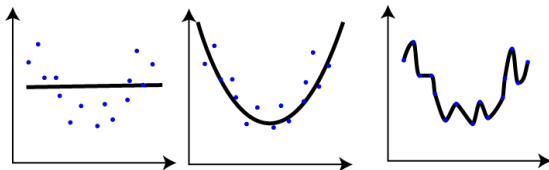
Underfitting-Overfitting I



How to know when there is overfitting or underfitting?

- if the training error is \ll then the error in CV, **overfitting**
- if the training error \gg the error in CV or is simply high, **underfitting**

Underfitting-Overfitting II



Let's consider that our hypotheses come from a set \mathcal{H} , called hypothesis space

Overfitting : it comes into play when the hypothesis space is **too rich**, *i.e.* if our hypothesis space is much **too large** and contains **potentially complex models**.

Underfitting : it comes into play when the hypothesis space is **poor**, *i.e.* if our hypothesis space is much **too small** and contains models that are **simple**.

What to do ?

To deal with underfitting, you can simply change the model class you're learning. For overfitting, on the other hand, we need to find a way to *simplify the learned model*.

Occam Razor Also known as the principle of simplicity or parsimony, it involves finding the simplest explanation (model) to explain the data :
no sunt multiplicanda entia praeter necessitatem" (William of Ockham)

It consists in adding a **penalty** term in the optimization problem.

Penalty

Introducing a penalty term into our optimization problem means finding a solution that strikes a compromise between *performance* and *simplicity* :

$$h^* = \arg \min_{h \in \mathcal{H}} \mathcal{R}_S(h) + \text{pen}(n, d)$$

This penalty term depends on both the dimension d of the data and the size of the dataset n . It is generally increasing with respect to n .

$$\text{pen}(n, d) \Leftrightarrow \mathcal{H}_n$$

$$\mathcal{H}_1 \subset \mathcal{H}_2 \subset \dots \subset \mathcal{H}_n$$

This notion is generally complex to determine, especially when it evolves as a function of sample size. For this reason, this approach is rarely used to find a model that is both simple and efficient.

Instead, we'll try to consider a very large set of assumptions and restrict

Regularization

Introducing a regularization term into our optimization problem

$$h^* = \arg \min_{h \in \mathcal{H}} \mathcal{R}_S(h).$$

Définition 2.7: Regularization

A regularization is a term used in statistics, and more particularly in Machine Learning, which refers to the **addition of an additional term to an optimization problem** to avoid overlearning. It takes the form of a function that **penalizes models that are too complex** : it will therefore seek to smooth models to make them smoother, or restrict the values that a model's parameters can take.

Regularized Empirical Risk I

the new problem can be formulated as :

$$h^* = \arg \min_{h \in \mathcal{H}} \mathcal{R}_S(h) + \lambda \|h\|,$$

where

- λ is a regularization parameter (also called a hyper-parameter to distinguish it from the h model parameters),
- $\|\cdot\|$ is a norm function.

So we're going to return the hypothesis that is capable of achieving the best compromise between performance and complexity.

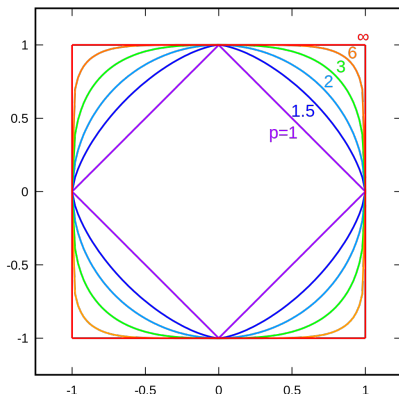
Regularized Empirical Risk II

This type of problem has already been encountered in the context of linear models, more precisely when we tried to learn a linear model which is **parcimonious**, *i.e.* whose solutions depend on only a small number of variables, *i.e.*

$$\min_{\beta \in \mathbb{R}^{d+1}} \|\mathbf{y} - \mathbf{X}\beta\| + \lambda \|\beta\|_1$$

Usual norms

Some examples of norms : $\|h\|_p = \left(\sum_{i=1}^d |h_i|^p \right)^{\frac{1}{p}}$.

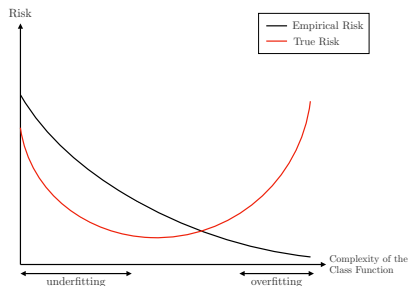


Choice of the parameter λ

Let us go back to the optimization problem :

$$h^* = \arg \min_h \sum_{i=1}^m \ell(h(\mathbf{x}), y) + \lambda \|h\|,$$

where λ : regularization parameter, controls the complexity of our hypothesis, indirectly, it is also said to control the richness of our hypothesis space \mathcal{H} .



Choose λ where the two curves intersect! But how?

Choice of λ and... let us come back to the cross validation

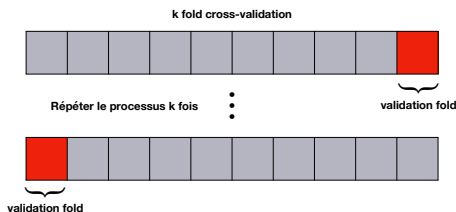
Our training set can also be used to verify our model's ability to generalize for the choice of a particular hyper-parameter.

- Standard validation : we separate our learning sample into two sets *train/valid* (2/3 - 1/3)
- k-fold cross validation : partition of the sample into k sets. $k - 1$ are used for learning and 1 for validation.
- Leave and One Out : we learn from all examples except 1, which is used to validate the model

k-fold cross validation as a way to tune our model

It works as follows :

- we define a grid of values for the hyper-parameter to be optimized : $\lambda \in (\lambda_1, \lambda_2, \dots, \lambda_p)$
- for each value of λ_i we calculate our average cross-validation error using a k -CV : $\frac{1}{k} \sum_{j=1}^k \mathcal{R}_{S_j}(h, \lambda_i)$



- we retain the value of λ_i for which the average CV error is lowest.

Few remarks

- To avoid the problem of overlearning, we use one (or more) regularization terms in our optimization problem.
- This is combined with k-fold cross validation (we commonly take $k = 10$) to ensure that the model generalizes well.
- Attention : these approaches are empirical ! To do the job properly, you'd have to study generalization guarantees

$$\left| \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \ell(h(\mathbf{x}), y) - \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{S}} \ell(h(\mathbf{x}), y) \right| \leq \mathcal{O}(\sqrt{1/m}, \lambda^{-1})$$

- Sometimes you need to distinguish between the optimization problem and the performance criterion you wish to optimize.

Generalization Bounds I

As we have just seen, the aim of these limits is to measure the probability of measuring a certain deviation between the empirical risk and the generalized risk..

These studies are relatively old and stem from learning *PAC* (Probably Approximately Correct) and in particular from the bounds *PAC* [Valiant, 1984].

$$\mathbb{P}(|\mathcal{R}(\cdot) - \mathcal{R}_S(\cdot)| \geq \varepsilon) \leq \delta,$$

where $\varepsilon > 0$ and $\delta \in [0, 1]$.

Generalization Bounds II

Terminals are generally presented in the following format

$$|\mathcal{R}(\cdot) - \mathcal{R}_S(\cdot)| \leq \varepsilon(\delta, m),$$

where $1 - \delta$ is interpreted as a level of confidence in your terminal. The function ε is then decreasing in the number of examples m but increasing in δ . You can show that the rate of convergence is generally in $\mathcal{O}(\ln(m)\sqrt{m})$ or sometimes even in $\mathcal{O}(1\sqrt{m})$.

Generalization Bounds III

Uniform Deviation

The construction of such bounds is based on the convergence of empirical risk towards generalized risk (see the strong law of large numbers).

$$\lim_{m \rightarrow \infty} \frac{1}{m} \sum_{i=1}^m \ell(y_i, h(\mathbf{x}_i)) = \mathbb{E}_{(y, \mathbf{x}) \sim \mathcal{D}} [\ell(y, h(\mathbf{x}))].$$

We'll refer to them as *uniform* bounds, since they are valid for all hypotheses $h \in \mathcal{H}$.

$$\left\{ \sup_{h \in \mathcal{H}} \left| \mathcal{R}^\ell(h) - \mathcal{R}_S^\ell(h) \right| \geq \varepsilon \right\}.$$

Generalization Bounds IV

These bounds are generally established with probabilistic results such as concentration inequalities. In addition, some of these bounds may depend on the complexity of the hypothesis space [Bousquet et al., 2004].

Théorème 2.1: Uniform Generalization Bound

Let \mathcal{H} be a hypothesis space of finite size, S a learning set of size m obtained *i.i.d.* from \mathcal{D} , ℓ a loss taking its values in $[0, 1]$ and $\delta > 0$. Then, for any $h \in \mathcal{H}$, with probability at least $1 - \delta$, we have :

$$\mathcal{R}^\ell(h) \leq \mathcal{R}_S^\ell(h) + \sqrt{\frac{\ln |\mathcal{H}| + \ln(2/\delta)}{2m}}.$$

Generalization Bounds V

In general, \mathcal{H} is rarely of finite size. Just imagine the set of regression lines in a plane, or the set of lines that could separate a collection of points with two different labels.

There are measures for evaluating the complexity of hypothesis spaces, such as the Vapnik-Chervonenkis dimension (VC-dim) [Vapnik and Chervonenkis, 1971] or even the *Rademacher Complexity* [Koltchinskii and Panchenko, 2000].

These notions are rather difficult to handle, so we'll concentrate on another approach in this course.

Generalization Bounds VI

Uniform Stability

This is a more recent approach to establishing these probabilistic bounds, and does not involve assessing the complexity of the hypothesis space.

Instead, we focus solely on the value of the hyper-parameter that controls the complexity of the hypothesis space. The disadvantage of this method is that it only applies to convex optimization problems.

In a few words, we're going to look at the impact of a change in the learning sample on loss. More formally

Generalization Bounds VII

Définition 2.8: Uniform Stability

A learning algorithm \mathcal{A} has a uniform stability constant in $\frac{\beta}{m}$ with respect to the cost function ℓ and the set of parameters θ , where $\beta > 0$ if :

$$\forall S, \forall i, 1 \leq i \leq m, \sup_{\mathbf{x}} |\ell(\theta_S, \mathbf{x}) - \ell(\theta_{S^i}, \mathbf{x})| \leq \frac{\beta}{m},$$

where S is our training set of size m , θ_S the model parameters learned with S , θ_{S^i} the same parameters learned with S^i where S^i is obtained by replacing the i -th example \mathbf{x}_i of S by another example \mathbf{x}'_i drawn according to \mathcal{D} . $\ell(\theta_S, \mathbf{x})$ is the loss value evaluated in \mathbf{x} .

Generalization Bounds VIII




Théorème 2.2: Bound via Uniform Stability

Let $\delta > 0$ and $m > 1$. Let an algorithm have a uniform stability constant in β/m with respect to a cost function ℓ bounded by K . Then, with probability at least $1 - \delta$ depending on the draw of S , we have :

$$\mathcal{R}^\ell(\theta_S) \leq \mathcal{R}_S^\ell(\theta_S) + \frac{2\beta}{m} + (4\beta + K) \sqrt{\frac{\ln 1/\delta}{2m}},$$

where $\mathcal{R}^\ell(\cdot)$ is the real risk $\mathcal{R}_S^\ell(\cdot)$ is the empirical risk on S .

Références I

-  Bousquet, O., Boucheron, S., and Lugosi, G. (2004).
Introduction to Statistical Learning Theory, pages 169–207.
Springer Berlin Heidelberg.
-  Koltchinskii, V. and Panchenko, D. (2000).
Rademacher processes and bounding the risk of function learning.
In Giné, E., Mason, D. M., and Wellner, J. A., editors, *High Dimensional Probability II*, pages 443–457. Birkhäuser Boston.
-  Valiant, L. G. (1984).
A theory of the learnable.
Communication of the ACM, 27(11) :1134–1142.

Références II



Vapnik, V. and Chervonenkis, A. (1971).

On the uniform convergence of relative frequencies of events to their probabilities.

Theory of Probability & Its Applications, 16(2) :264–280.