UE: Machine Learning Fundamentals

Part I : Supervised Learning

Massih-Reza Amini

http://ama.liglab.fr/~amini/Cours/ML/ML.html

Université Grenoble Alpes
Laboratoire d'Informatique de Grenoble
Massih-Reza.Amini@imag.fr

Organization

- Formation
  - Theoretical courses - 12 weeks (3 ECTS), MoSIG and MSIAM
  - Practical courses - 9 hours in December MSIAM students `Marianne.Clausel@imag.fr`
- Practical information (important dates, timetables, defence schedule, etc.) are available at `http://mosig.imag.fr`
- Research projects `http://projets-mastermi.imag.fr/pcarre`

Program[1]

1. Supervised Learning:
   - The Empirical Risk Minimization principle
   - Binary models and their link with the ERM principle
   - Unconstrained Convex Optimization
   - Consistency of the ERM principle

   ——————— Midterm Exam ———————

   - Multi-class classification

2. Unsupervised Learning:
   - Generative models and the EM algorithm
   - CEM algorithm

3. Semi-supervised Learning:
   - Graphical and Generative models
   - Discriminant models

[1] Based on Chapters 1, 2, 3 & 5 of [Amini 15]

Learning and Inference

The process of inference is done in three steps:

1. Observe a phenomenon,
2. Construct a model of the phenomenon,
3. Do predictions.

Learning and Inference

The process of inference is done in three steps:

1. Observe a phenomenon,
2. Construct a model of the phenomenon,
3. Do predictions.

- These steps are involved in more or less all natural sciences! *All that is necessary to reduce the whole nature of laws similar to those which Newton discovered with the aid of calculus, is to have a sufficient number of observations and a mathematics that is complex enough* (Marquis de Condorcet, 1785)

Learning and Inference

The process of inference is done in three steps:

1. Observe a phenomenon,
2. Construct a model of the phenomenon,
3. Do predictions.

- These steps are involved in more or less all natural sciences!
- The aim of learning is to automate this process,

Learning and Inference

The process of inference is done in three steps:

1. Observe a phenomenon,
2. Construct a model of the phenomenon,
3. Do predictions.

- These steps are involved in more or less all natural sciences!
- The aim of learning is to automate this process,
- The aim of the learning theory is to formalize the process.

Induction vs. deduction

- Induction is the process of deriving general principles from particular facts or instances.

- Deduction is, in the other hand, the process of reasoning in which a conclusion follows necessarily from the stated premises; it is an inference by reasoning from the general to the specific.

  This is how mathematicians prove theorems from axioms.

Pattern recognition

If we consider the context of supervised learning for pattern recognition:

- The data consist of pairs of examples (vector representation of an observation, class label),
- Class labels are often $\mathcal{Y} = \{1, \ldots, K\}$ with K large (but in the theory of ML we consider the binary classification case $\mathcal{Y} = \{-1, +1\}$),
- The learning algorithm constructs an association between the vector representation of an observation $\rightarrow$ class label,
- Aim: Make few errors on unseen examples.

Pattern recognition (Exemple)

IRIS classification, Ronald Fisher (1936)

Iris Setosa

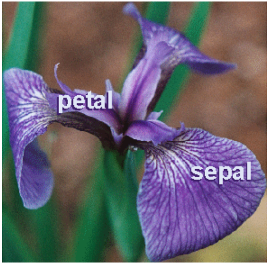
Iris Versicolor

Iris Virginica

Pattern recognition (Exemple)

- First step is to formalize the perception of the flowers with relevant common characteristics, that constitute the features of their vector representations.

- This usually requires expert knowledge.



Pattern recognition (Exemple)

- If observations are from a Field of Irises

Pattern recognition (Exemple)

- If observations are from a Field of Irises then they become

Fisher's *Iris* Data

longueur des sépales (en cm) *(Sepal length)*	largeur des sépales (en cm) *(Sepal width)*	longueur des pétales (en cm) *(Petal length)*	largeur des pétales (en cm) *(Petal width)*	Espèce *(Species)*
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
4.6	3.1	1.5	0.2	*I. setosa*
5.0	3.6	1.4	0.2	*I. setosa*

...

7.0	3.2	4.7	1.4	*I. versicolor*
6.4	3.2	4.5	1.5	*I. versicolor*
6.9	3.1	4.9	1.5	*I. versicolor*
5.5	2.3	4.0	1.3	*I. versicolor*
6.5	2.8	4.6	1.5	*I. versicolor*

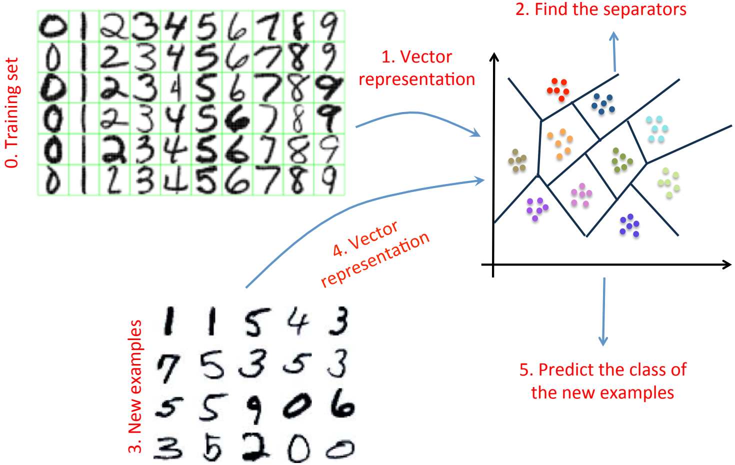
...

6.3	3.3	6.0	2.5	*I. virginica*
5.8	2.7	5.1	1.9	*I. virginica*
7.1	3.0	5.9	2.1	*I. virginica*
6.3	2.9	5.6	1.8	*I. virginica*
6.5	3.0	5.8	2.2	*I. virginica*

Pattern recognition (Exemple)

- The constitution of vectorised observations and their associated labels is generally time consuming.
- Many studies are now focused on representation learning using deep neural networks
- Second step: Learning translates then in the search of a function that maps vectorised observations (inputs) to their associated outputs

Pattern recognition

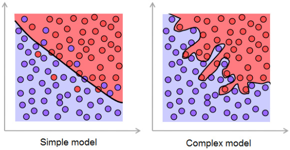


Approximation - Interpolation

It is always possible to construct a function that exactly fits the data.

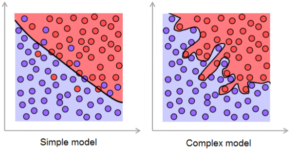
Approximation - Interpolation

It is always possible to construct a function that exactly fits the data.



Approximation - Interpolation

It is always possible to construct a function that exactly fits the data.



Is it reasonable?

Occam razor

Idea: Search for regularities (or repetitions) in the observed phenomenon, generalization is done from the passed observations to the new futur ones $\Rightarrow$ Take the most simple model ...

But how to measure the simplicity ?

1. Number of constantes,
2. Number de parameters,
3. ...

Basic Hypotheses

Two types of hypotheses:

- Past observations are related to the future ones
  → The phenomenon is stationary

- Observations are independently generated from a source
  → Notion of independence

Aims

→ How can one do predictions with past data? What are the hypotheses?

- Give a formel definition of learning, generalization, overfitting,
- Characterize the performance of learning algorithms,
- Construct better algorithms.

Probabilistic model

Relations between the past and future observations.

- Independence: Each new observation provides a maximum individual information,
- identically Distributed : Observations provide information on the phenomenon which generates the observations.

Formally

We consider an input space $\mathcal{X} \subseteq \mathbb{R}^d$ and an output space $\mathcal{Y}$.

Assumption: Example pairs $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ are *identically* and *independently* distributed (`i.i.d`) with respect to an unknown but fixed probability distribution $\mathcal{D}$.

Samples: We observe a sequence of m pairs of examples $(\mathbf{x}_i, y_i)$ generated `i.i.d` from $\mathcal{D}$.

Aim: Construct a prediction function $f : \mathcal{X} \to \mathcal{Y}$ which predicts an output y for a given new $\mathbf{x}$ with a minimum probability of error.

Supervised Learning

- Discriminant models directly find a classification function $f : \mathcal{X} \to \mathcal{Y}$ from a given class of functions $\mathcal{F}$;
- The function found should be the one having the lowest probability of error

$$R(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}} L(x, y) = \int_{\mathcal{X}\times\mathcal{Y}} L(f(x), y)\, d\mathcal{D}(x, y)$$

Where L is a risk function defined as

$$L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$$

The risk function considered in classification is usually the misclassification error:

$$\forall (x, y); L(f(x), y) = [\![f(x) \neq y]\!]$$

Where $[\![\pi]\!]$ is equal to 1 if the predicate π is true and 0 otherwise.

Empirical risk minimization (ERM) principle

- As the probability distribution $\mathcal{D}$ is unknown, the analytic form of the true risk cannot be driven, so the prediction function cannot be found directly on $R(f)$.

- Empirical risk minimization (ERM) principle: Find f by minimizing the unbiased estimator of R on a given training set $S=(x_i,y_i)_{i=1}^m$:

$$\hat{R}_m(f,S)=\frac{1}{m}\sum_{i=1}^{m}L(f(x_i),y_i)$$

- However, without restricting the class of functions this is not the right way of proceeding (occam razor) ...

ERM principle, problem

Suppose that the input dimension is $d = 1$, let the input space $\mathcal{X}$ be the interval $[a, b] \subset \mathbb{R}$ where a and b are real values such that $a < b$, and suppose that the output space is $\{-1, +1\}$. Moreover, suppose that the distribution $\mathcal{D}$ generating the examples $(\mathbf{x}, y)$ is an uniform distribution over $[a, b] \times \{-1\}$. Consider now, a learning algorithm which minimizes the empirical risk by choosing a function in the function class $\mathcal{F} = \{f : [a, b] \to \{-1, +1\}\}$ (also denoted as $\mathcal{F} = \{-1, +1\}^{[a,b]}$) in the following way ; after reviewing a training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ the algorithm outputs the prediction function f_S such that

$$f_S(\mathbf{x}) = \begin{cases} -1, & \text{if } \mathbf{x} \in \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \\ +1, & \text{otherwise} \end{cases}$$

Consistency of the ERM principle

- For the above problem, the found classifier has an empirical risk equal to 0, and that for any given training set. However, as the classifier makes an error over the entire infinite set $[a, b]$ except on a finite training set (of measure zero), its generalization error is always equal to 1.

- So the question is : *in which case the ERM principle is likely to generate a general learning rule?*
  $\Rightarrow$ The answer of this question lies in a statistical notion called consistency.

Consistency of the ERM principle (2)

This concept indicates two conditions that a learning algorithm has to fulfil, namely

(a) the algorithm must return a prediction function whose empirical error reflects its generalization error when the size of the training set tends to infinity :
$\forall \epsilon > 0, \lim_{m\to\infty} \mathbb{P}(|\hat{\mathfrak{L}}(f_S, S) - \mathfrak{L}(f_S)| > \epsilon) = 0$, denoted as,

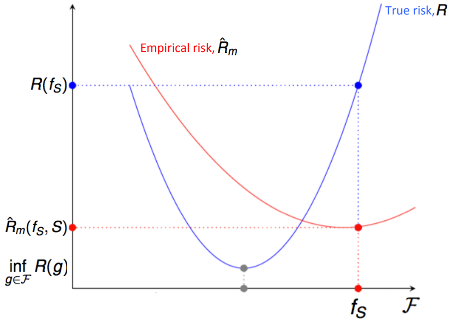
$$\hat{\mathfrak{L}}(f_S, S) \xrightarrow{\mathbb{P}} \mathfrak{L}(f_S)$$

(b) in the asymptotic case, the algorithm must allow to find the function which minimises the generalization error in the considered function class :

$$\hat{\mathfrak{L}}(f_S, S) \xrightarrow{\mathbb{P}} \inf_{g\in\mathcal{F}} \mathfrak{L}(g)$$

Consistency of the ERM principle (3)

These two conditions imply that the empirical error $\hat{\mathfrak{L}}(f_S, S)$ of the prediction function found by the learning algorithm over a training S, f_S, converges in probability to its generalization error $\mathfrak{L}(f_S)$ and $\inf_{g\in\mathcal{F}} \mathfrak{L}(g)$:



Study the consistency of the ERM principle

The fundamental result of the learning theory [Vapnik 88, theorem 2.1, p.38] concerning the consistency of the ERM principle, exhibits another relation involving the supremum over the function class in the form of an unilateral uniform convergence and which stipulates that :

The ERM principle is consistent if and only if :

$$\forall \epsilon > 0, \lim_{m \to \infty} \mathbb{P}\left(\sup_{f \in \mathcal{F}} \left[\mathfrak{L}(f) - \hat{\mathfrak{L}}(f, S) \right] > \epsilon \right) = 0$$

Study the consistency of the ERM principle

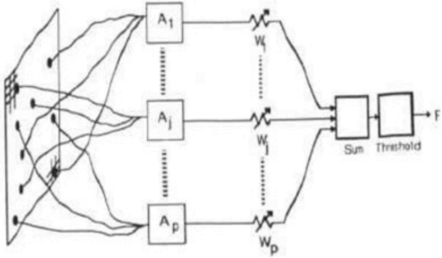
- A direct implication of this result is an uniform bound over the generalization error of all prediction functions $f \in \mathcal{F}$ learned on a training set S of size m and which writes :

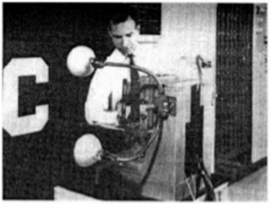
$$\forall \delta \in]0,1], \mathbb{P}\left(\forall f \in \mathcal{F}, (\mathfrak{L}(f) - \hat{\mathfrak{L}}(f,S)) \leq \mathfrak{C}(\mathcal{F}, m, \delta)\right) \geq 1 - \delta$$

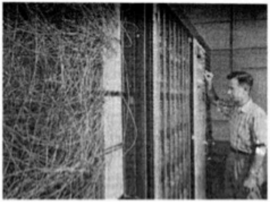
Where $\mathfrak{C}$ depends on the size of the function class, the size of the training set, and the desired precision $\delta \in]0,1]$. There are different ways to measure the size of a function class and the measure commonly used is called complexity or the capacity of the function class.

Usual binary classification models

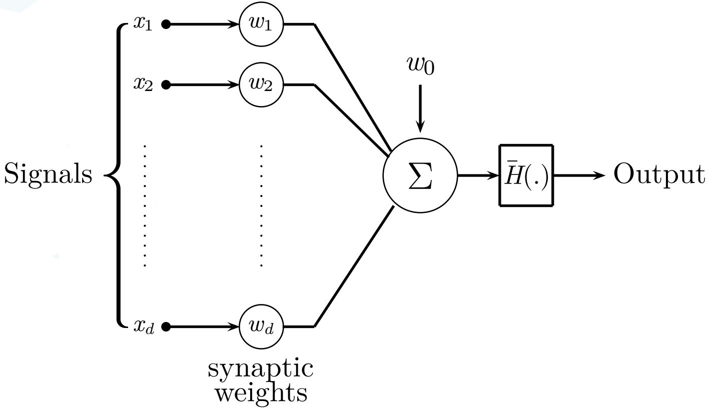
Perceptron [Rosenblatt, 1958]







Perceptron [Rosenblatt, 1958]



- Linear prediction function

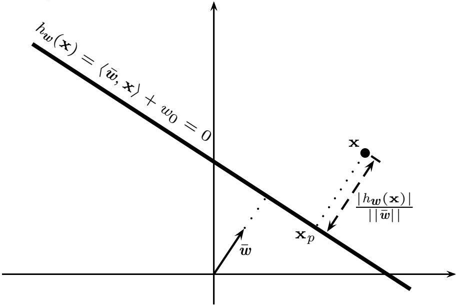
$$
\begin{array}{rcl}
h_{\boldsymbol{w}} : \mathbb{R}^d & \rightarrow & \mathbb{R} \\
\mathbf{x} & \mapsto & \langle \bar{\boldsymbol{w}}, \mathbf{x} \rangle + w_0
\end{array}
$$

Perceptron [Rosenblatt, 1958]

- Linear prediction function

$$
\begin{aligned}
h_{\boldsymbol{w}} : \mathbb{R}^d &\rightarrow \mathbb{R} \\
\mathbf{x} &\mapsto \langle \bar{\boldsymbol{w}}, \mathbf{x} \rangle + w_0
\end{aligned}
$$

- Find the parameters $\boldsymbol{w} = (\bar{\boldsymbol{w}}, w_0)$ by minimising the distance between the misclassified examples to the decision boundary.



Learning Perceptron parameters

- Objective function

$$\hat{\mathcal{L}}(\boldsymbol{w}) = -\sum_{i' \in \mathcal{I}} y_{i'} \left(\langle \bar{\boldsymbol{w}}, \mathbf{x}_{i'} \rangle + w_0 \right)$$

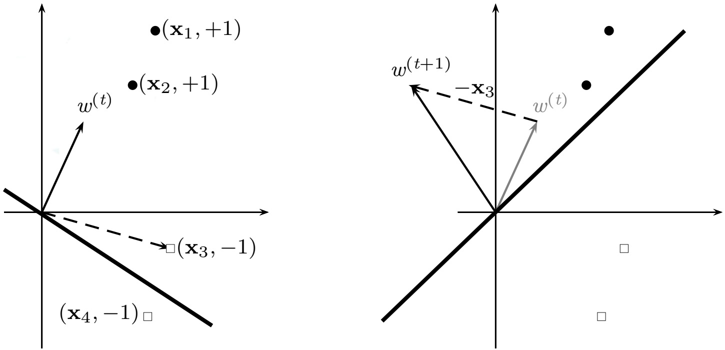
- Derivatives of with respect to the parameters

$$\begin{aligned} \frac{\partial \hat{\mathcal{L}}(\boldsymbol{w})}{\partial w_0} &= -\sum_{i' \in \mathcal{I}} y_{i'}, \\ \nabla \hat{\mathcal{L}}(\bar{\boldsymbol{w}}) &= -\sum_{i' \in \mathcal{I}} y_{i'} \mathbf{x}_{i'} \end{aligned}$$

- Perceptron: on-line parameter updates

$$\forall (\mathbf{x}, y), \text{ if } y(\langle \bar{\boldsymbol{w}}, \mathbf{x} \rangle + w_0) \leq 0 \text{ then } \begin{pmatrix} w_0 \\ \bar{\boldsymbol{w}} \end{pmatrix} \leftarrow \begin{pmatrix} w_0 \\ \bar{\boldsymbol{w}} \end{pmatrix} + \eta \begin{pmatrix} y \\ y\mathbf{x} \end{pmatrix}$$

Graphical depiction of the online update rule



Perceptron (algorithm)

Algorithm 1 The algorithm of perceptron

Training set $S = \{(x_i, y_i) \mid i \in \{1, \dots, m\}\}$
Initialize the weights $w^{(0)} \leftarrow 0$
$t \leftarrow 0$
Learning rate $\eta > 0$
repeat
    Choose randomly an example $(x^{(t)}, y^{(t)}) \in S$
    if $y \left\langle w^{(t)}, x^{(t)} \right\rangle < 0$ **then**
        $w_0^{(t+1)} \leftarrow w_0^{(t)} + \eta \times y^{(t)}$
        $w^{(t+1)} \leftarrow w^{(t)} + \eta \times y^{(t)} \times x^{(t)}$
    end if
    $t \leftarrow t + 1$
until $t > T$

☞ But does this updates converge?

Perceptron (convergence)

[Novikoff, 1962] showed that

- if there exists a weight $\bar{\boldsymbol{w}}^*$, such that $\forall i \in \{1,\ldots,m\}, y_i \times \langle \bar{\boldsymbol{w}}^*, x_i\rangle > 0$,
- then, by denoting $\rho = \min_{i\in\{1,\ldots,m\}} \left(y_i \left\langle \frac{\bar{\boldsymbol{w}}^*}{||\bar{\boldsymbol{w}}^*||}, x_i \right\rangle\right)$,
- and, $R = \max_{i\in\{1,\ldots,m\}} ||x_i||$,
- and, $\bar{\boldsymbol{w}}^{(0)} = 0$, $\eta = 1$,
- we have a bound over the maximum number of updates ℓ :

$$\ell \leq \left\lfloor \left(\frac{R}{\rho}\right)^2 \right\rfloor$$

Homework

1. We suppose that all the examples in the training set are within a hypersphere of radius R (i.e. $\forall \mathbf{x}_i \in S, ||\mathbf{x}_i|| \leq R$). Further, we initialise the weight vector to be the null vector (i.e. $w^{(0)} = 0$) as well as the learning rate $\epsilon = 1$. Show that after t updates, the norme of the current weight vector satisfies :
$$||w^{(t)}||^2 \leq t \times R^2 \tag{1}$$
hint : You can consider $||w^{(t)}||^2$ as $||w^{(t)} - w^{(0)}||^2$
2. Using the the same condition than in the previous question, show that after t updates of the weight vector we have
$$\left\langle \frac{w^*}{||w^*||}, w^{(t)} \right\rangle \geq t \times \rho \tag{2}$$
3. Deduce from equations (1) and (2) that the number of iterations t is bounded by
$$t \leq \left\lfloor \left(\frac{R}{\rho}\right)^2 \right\rfloor$$
where $\lfloor x \rfloor$ represents the floor function (This result is due to Novikoff, 1966).

Perceptron Program

```
#include "defs.h"
void perceptron(X, Y, w, m, d, eta, T)
double **X;
double *Y;
double *w;
long int m;
long int d;
double  eta;
long int T;
{
  long int i, j, t=0;
    double ProdScal;
  // Initialisation of the weight vector
  for(j=0; j<=d; j++)
    w[j]=0.0;

  while(t<T)
  {
    i=(rand()%m) + 1;
    for(ProdScal=w[0], j=1; j<=d; j++)
       ProdScal+=w[j]*X[i][j];
    if(Y[i]*ProdScal<= 0.0){
      w[0]+=eta*Y[i];
      for(j=1; j<=d; j++)
         w[j]+=eta*Y[i]*X[i][j];
    }
    t++;
  }
}
```

source: http://ama.liglab.fr/~amini/Perceptron/

ADAptive LInear NEuron [Widrow & Hoff, 1960]

- ADAptive LInear NEuron
- Linear prediction function :

$$\begin{aligned} h_{\boldsymbol{w}} : \mathcal{X} &\rightarrow \mathbb{R} \\ x &\mapsto \langle \bar{w}, x \rangle + w_0 \end{aligned}$$

- Find parameters that minimise the convex upper-bound of the empirical 0/1 loss

$$\hat{\mathcal{L}}(\boldsymbol{w}) = \frac{1}{m} \sum_{i=1}^{m} (y_i - h_{\boldsymbol{w}}(\mathbf{x}_i))^2$$

- Update rule : stochastic gradient descent algorithm with a learning rate $\eta > 0$

$$\forall (\mathbf{x}, y), \begin{pmatrix} w_0 \\ \bar{\boldsymbol{w}} \end{pmatrix} \leftarrow \begin{pmatrix} w_0 \\ \bar{\boldsymbol{w}} \end{pmatrix} + \eta (y - h_{\boldsymbol{w}}(\mathbf{x})) \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \quad (3)$$

Adaline

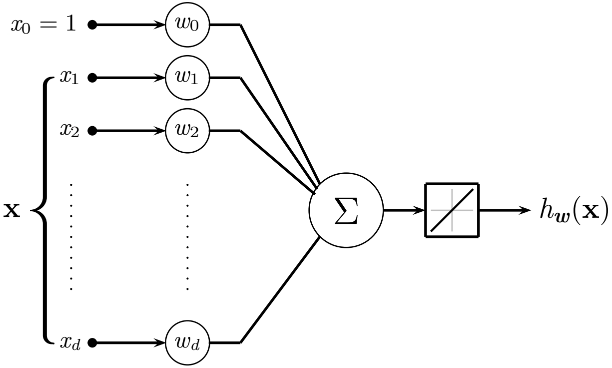
- ADAptive LInear NEuron
- Linear prediction function :

$$\begin{aligned} h_{\boldsymbol{w}} : \mathcal{X} &\rightarrow \mathbb{R} \\ x &\mapsto \langle \bar{w}, x \rangle + w_0 \end{aligned}$$

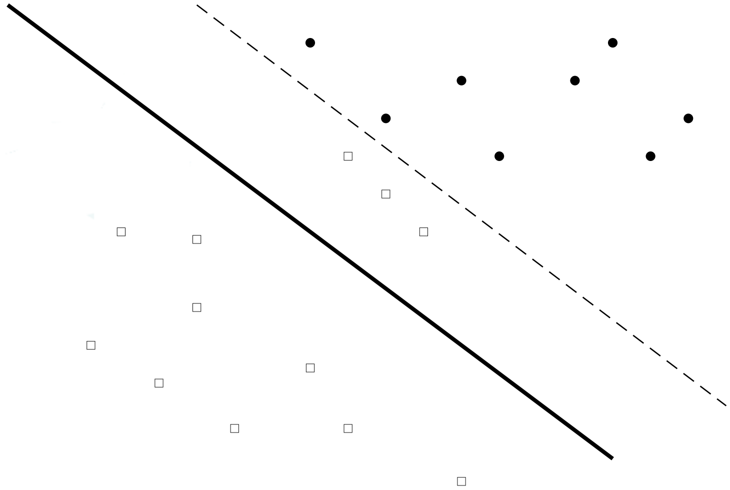
Algorithm 2 The algorithm of Adaline

```
Training set $S = \{(x_i, y_i) \mid i \in \{1, \ldots, m\}\}$
Initialize the weights $w^{(0)} \leftarrow 0$
$t \leftarrow 0$
Learning rate $\eta > 0$
repeat
    Choose randomly an example $(x^{(t)}, y^{(t)}) \in S$
    $w_0^{(t+1)} \leftarrow w_0^{(t)} + \eta \times (y^{(t)} - h_w(x^{(t)}))$
    $\bar{w}^{(t+1)} \leftarrow \bar{w}^{(t)} + \eta \times (y^{(t)} - h_w(x^{(t)})) \times x^{(t)}$
    $t \leftarrow t + 1$
until $t > T$
```

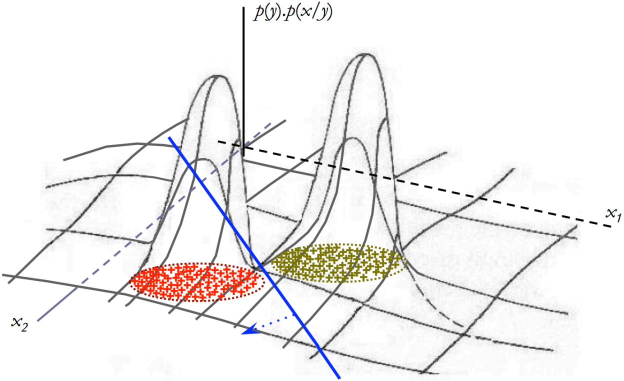
Formal models



Perceptron vs Adaline



Logistic regression: generative models



Each example $\mathbf{x}$ is supposed to be generated by a mixture model of parameters Θ:

$$P(\mathbf{x} \mid \Theta) = \sum_{k=1}^{K} P(y=k) P(\mathbf{x} \mid y=k, \Theta)$$

Logistic regression: generative models

- The aim is then to find the parameters Θ for which the model explains the best the observations,
- That is done by maximizing the log-likelihood of data $S = \{(\mathbf{x}_i, y_i); i \in \{1, \dots, m\}\}$

$$\mathcal{L}(\Theta) = \ln \prod_{i=1}^{m} P(\mathbf{x}_i \mid \Theta)$$

- Classical density functions are Gaussian density functions

$$P(\mathbf{x} \mid y = k, \Theta) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma_k|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu_k)^\top \Sigma_k^{-1}(\mathbf{x}-\mu_k)}$$

Logistic regression: generative models

- Once the parameters Θ are estimated; the generative model can be used for classification by applying the Bayes rule:

$$\begin{aligned}\forall \mathbf{x}; y^* &= \underset{k}{\operatorname{argmax}}\, P(y=k\mid \mathbf{x})\\ &\propto \underset{k}{\operatorname{argmax}}\, P(y=k)\times P(\mathbf{x}\mid y=k,\Theta)\end{aligned}$$

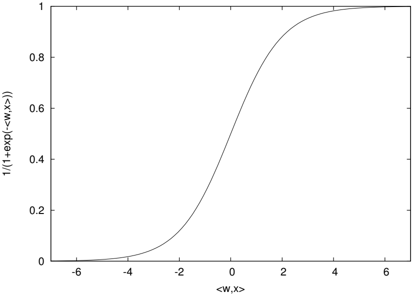
- Problem: in most real life applications the distributional assumption over data does not hold,
- The Logistic Regression model does not make any assumption except that

$$\ln\frac{P(y=1\mid \mathbf{x})}{P(y=0\mid \mathbf{x})} = \langle \bar{w}, \mathbf{x}\rangle + w_0$$

Logistic regression

- The logistic regression has been proposed to model the posterior probability of classes via logistic functions.

$$
\begin{aligned}
P(y=1 \mid x) &= \frac{1}{1+e^{-\langle \bar{w},x\rangle - w_0}} = g_w(x) \\
P(y=0 \mid x) &= 1 - P(y=1 \mid x) = \frac{1}{1+e^{\langle \bar{w},x\rangle + w_0}} = 1 - g_w(x) \\
P(y \mid x) &= (g_w(x))^y (1-g_w(x))^{1-y}
\end{aligned}
$$



Logistic regression

- For

$$\begin{aligned} g : \mathbb{R} &\rightarrow \;]0,1[\\ x &\mapsto \frac{1}{1+e^{-x}} \end{aligned}$$

we have

$$g'(x) = \frac{\partial g}{\partial x} = g(x)(1-g(x))$$

- Model parameters w are found by maximizing the complete log-liklihood, which by assuming that m training examples are generated independently, writes

$$\begin{aligned} \mathcal{L} = \ln \prod_{i=1}^{m} P(x_i, y_i) &= \ln \prod_{i=1}^{m} P(y_i \mid x_i) + \ln \prod_{i=1}^{m} P(x_i) \\ &\approx \sum_{i=1}^{m} \ln \left[(g_w(x_i))^{y_i} (1-g_w(x_i))^{1-y_i} \right] \end{aligned}$$

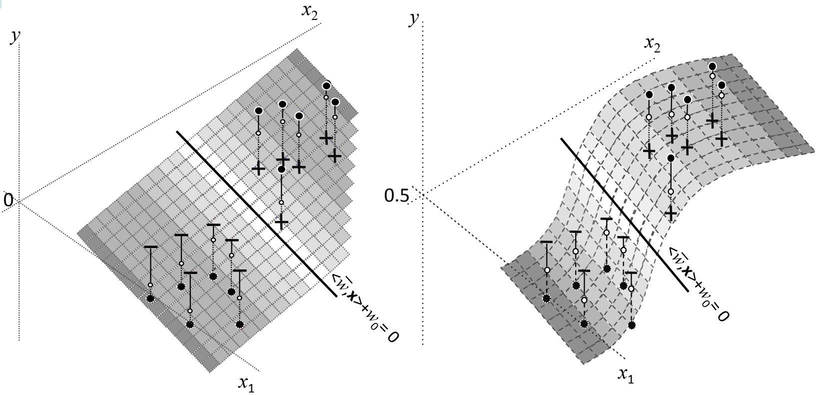
Logisitic Regression : link with the ERM principle

- If we consider the function $h_w : x \mapsto \langle \bar{w}, x \rangle + w_0$, the maximization of the log-liklihood $\mathcal{L}$ is equivalent to the minimization of the empirical logistic loss in the case where $\forall i, y_i \in \{-1, +1\}$.

$$\hat{\mathcal{L}}(\boldsymbol{w}) = \frac{1}{m} \sum_{i=1}^{m} \ln(1 + e^{-y_i h_w(x_i)})$$

- Minimization can be carried out with usual convex optimization techniques (i.e. conjugate gradient or the quasi-newton method)

Adaline vs Logistic regression



ADAptive BOOSTing [Schapire, 1999]

- The Adaboost algorithm generates a set of weak learners and combines them with a majority vote in order to produce an efficient final classifier.

- Each weak classifier is trained sequentially in the way to take into account the classification errors of the previous classifier
  - ☞ This is done by assigning weights to training examples and at each iteration to increase the weights of those on which the current classifier makes misclassification.
  - ☞ In this way the new classifier is focalized on *hard* examples that have been misclassified by the previous classifier.

AdaBoost, algorithm

Algorithm 3 The algorithm of Boosting

Training set $S = \{(x_i, y_i) \mid i \in \{1, \dots, m\}\}$
Initialize the initial distribution over examples $\forall i \in \{1, \dots, m\}, D_1(i) = \frac{1}{m}$
T, the maximum number of iterations (or classifiers to be combined)
for t=1,...,T **do**
  Train a weak classifier $f_t : \mathcal{X} \rightarrow \{-1, +1\}$ by using the distribution D_t
  Set $\epsilon_t = \sum_{i: f_t(x_i) \neq y_i} D_t(i)$
  Choose $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
  Update the distribution of weights

$$\forall i \in \{1, \dots, m\}, D_{t+1}(i) = \frac{D_t(i)\, e^{-\alpha_t y_i f_t(x_i)}}{Z_t}$$

  Where,

$$Z_t = \sum_{i=1}^{m} D_t(i)\, e^{-\alpha_t y_i f_t(x_i)}$$

end for
The final classifier: $\forall x, F(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t f_t(x)\right)$

source: `http://ama.liglab.fr/~amini/RankBoost/`

AdaBoost, algorithm

Algorithm 4 The algorithm of Boosting

Training set $S = \{(x_i, y_i) \mid i \in \{1, \dots, m\}\}$

Initialize the initial distribution over examples $\forall i \in \{1, \dots, m\}, D_1(i) = \frac{1}{m}$

T, the maximum number of iterations (or classifiers to be combined)

for t=1,...,T **do**

  Train a weak classifier $f_t : \mathcal{X} \rightarrow \{-1, +1\}$ by using the distribution D_t

  Set $\epsilon_t = \sum_{i:f_t(x_i) \neq y_i} D_t(i)$

  Choose $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$

  Update the distribution of weights

$$\forall i \in \{1, \dots, m\}, D_{t+1}(i) = \frac{D_t(i)\, e^{-\alpha_t y_i f_t(x_i)}}{Z_t}$$

  Where,

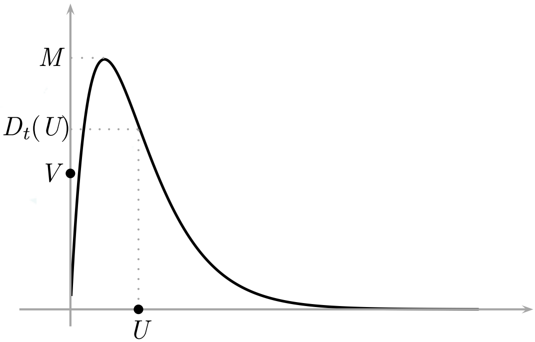
$$Z_t = \sum_{i=1}^{m} D_t(i)\, e^{-\alpha_t y_i f_t(x_i)}$$

end for

The final classifier: $\forall x, F(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t f_t(x)\right)$

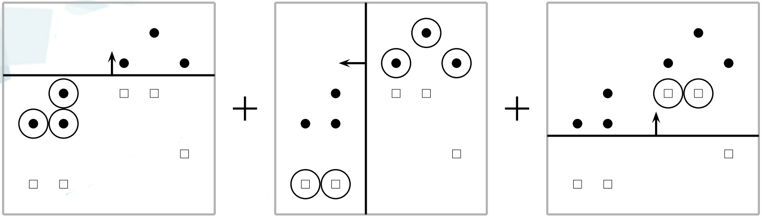
source: `http://ama.liglab.fr/~amini/RankBoost/`

How to sample using a distribution D_t

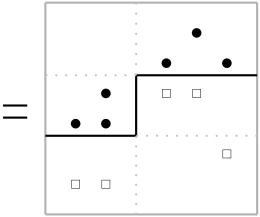


- Choose randomly an index $U \in \{1, \ldots, m\}$ and a real-value $V \in [0, \max_{i \in \{1,\ldots,m\}} D_t(i)]$, if $D_t(U) > V$ then accept the example $(\mathbf{x}_U, y_U)$.

AdaBoost, geometry interpretation



$\alpha_1 = 0.5$ $\alpha_2 = 0.1$ $\alpha_3 = 0.75$



Homework

1. If we denote by $\forall x, H(x)=\sum_{t=1}^{T}\alpha_t f_t(x)$ and $F(x)=\text{sign}(H(x))$ show that

$$\frac{1}{m}\sum_{i=1}^{m}[\![y_i\neq F(x_i)]\!]\leq\frac{1}{m}\sum_{i=1}^{m}e^{-y_iH(x_i)}$$

2. Deduce that

$$\frac{1}{m}\sum_{i=1}^{m}e^{-y_iH(x_i)}=\sum_{i=1}^{m}Z_1D_2(i)\prod_{t>1}e^{-y_i\alpha_tf_t(x_i)}$$

And,

$$\frac{1}{m}\sum_{i=1}^{m}e^{-y_iH(x_i)}=\prod_{t=1}^{T}Z_t \tag{4}$$

Homework

3. The minimization of (4) is carried out by minimizing each of its terms. Using the definition of ϵ_t show that:

$$\forall t, Z_t = \epsilon_t e^{\alpha_t} + (1-\epsilon_t)e^{-\alpha_t}$$

4. Further show that the minimum of the normalisation term, with respect to the combination weights, α_t is reached for $\alpha_t = \frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}$
5. By posing $\gamma_t = \frac{1}{2} - \epsilon_t$, and when $\epsilon_t < \frac{1}{2}$ show that

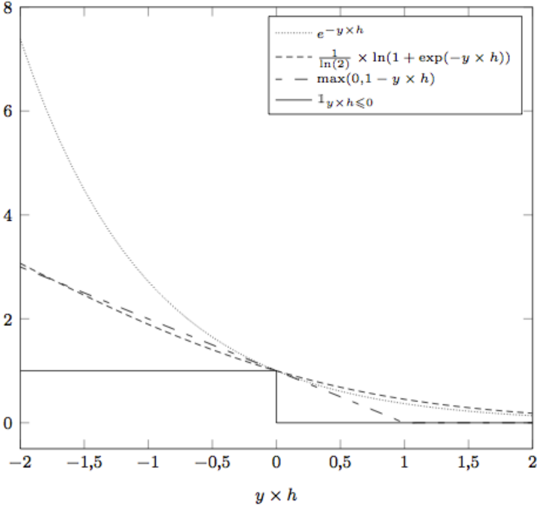
$$\forall t, Z_t = \sqrt{1-4\gamma_t^2} \leq e^{-2\gamma_t^2}$$

6. Finally show that the empirical misclassification error decreases exponentially to 0

$$\frac{1}{m}\sum_{i=1}^{m}[\![y_i \neq F(x_i)]\!] \leq \prod_{t=1}^{T} Z_t \leq e^{-2\sum_{t=1}^{T}\gamma_t^2}$$

Unconstrained convex optimization

Common convex upper bounds for the misclassification error



Property

- The learning problem casts into a easier unconstrained convex optimization problem.
- Consider the Taylor formula of the objective function around its minimiser

$$\hat{\mathcal{L}}(\boldsymbol{w}) = \hat{\mathcal{L}}(\boldsymbol{w}^*) + (w - w^*)^\top \underbrace{\nabla\hat{\mathcal{L}}(\boldsymbol{w}^*)}_{=0} + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^\top \mathbf{H}(\boldsymbol{w} - \boldsymbol{w}^*) + o(\| w - w^* \|^2)$$

- The Hessian matrix is symmetric and from Schwarz theorem its eigenvectors $(\boldsymbol{v}_i)_{i=1}^d$ form an orthonormal basis.

$$\forall (i,j) \in \{1,\dots,d\}^2, \mathbf{H}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i, \text{ et } \boldsymbol{v}_i^\top \boldsymbol{v}_j = \begin{cases} +1 : \text{si } i = j, \\ 0 : \text{otherwise.} \end{cases}$$

Property (2)

- Every weight vector $\boldsymbol{w} - \boldsymbol{w}^*$ can be uniquely decomposed in this basis

$$\boldsymbol{w} - \boldsymbol{w}^* = \sum_{i=1}^{d} q_i v_i$$

- That to say

$$\hat{\mathcal{L}}(\boldsymbol{w}) = \hat{\mathcal{L}}(\boldsymbol{w}^*) + \frac{1}{2}\sum_{i=1}^{d} \lambda_i q_i^2$$

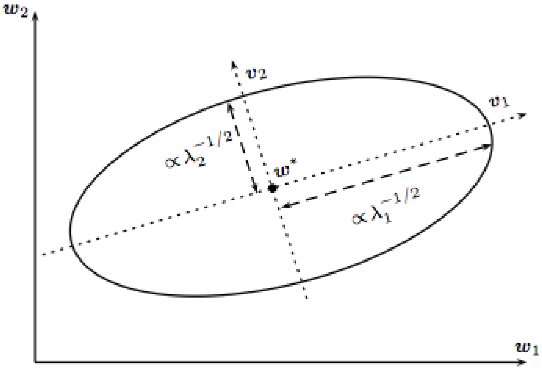
- Furthermore the Hessian matrix is definite positive, because of the definition of the global minimum

$$(\boldsymbol{w} - \boldsymbol{w}^*)^\top \mathbf{H}(\boldsymbol{w} - \boldsymbol{w}^*) = \sum_{i=1}^{d} \lambda_i q_i^2 = 2(\hat{\mathcal{L}}(\boldsymbol{w}) - \hat{\mathcal{L}}(\boldsymbol{w}^*)) \geq 0$$

All the eigenvalues of $\mathbf{H}$ are then positive.

Property (3)

- This implies that the level lines of $\hat{\mathcal{L}}$, defined by weight points for which $\hat{\mathcal{L}}$ is constant, are ellipses

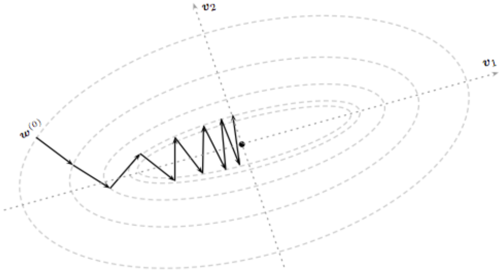


Gradient descent algorithm [Rumelhart et al., 1986]

- The gradient descent algorithm is an iterative algorithm that updates the weight vectors at each step :

$$\forall t \in \mathbb{N}, \boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)})$$

Where $\eta > 0$ is the learning rate



Convergence of the gradient descent algorithm

- Take the decomposition of any vector $\boldsymbol{w}-\boldsymbol{w}^*$ in the orthonormal basis $(\boldsymbol{v}_i)_{i=1}^d$ formed by the eigenvectors of the Hessian matrix

$$\nabla\hat{\mathcal{L}}(\boldsymbol{w})=\sum_{i=1}^d q_i\lambda_i\boldsymbol{v}_i$$

- Let $\boldsymbol{w}^{(t)}$ be the weight vector obtained from $\boldsymbol{w}^{(t-1)}$ after applying the gradient descent rule

$$\boldsymbol{w}^{(t)}-\boldsymbol{w}^{(t-1)}=\sum_{i=1}^d\left(q_i^{(t)}-q_i^{(t-1)}\right)\boldsymbol{v}_i=-\eta\nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t-1)})=-\eta\sum_{i=1}^d q_i^{(t-1)}\lambda_i\boldsymbol{v}_i$$

- So

$$\forall i\in\{1,\dots,d\}, q_i^{(t)}=(1-\eta\lambda_i)^t q_i^{(0)}$$

and the algorithm convergence if

$$\eta<\frac{1}{2\lambda_{max}}$$

OK but how to find the good learning rate? Line search

At each iteration t, on $\boldsymbol{w}^{(t)}$

- Estimate the descent direction $\mathbf{p}_t$ (i.e. $\mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)}) < 0$)
- Update

$$\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} + \eta_t \mathbf{p}_t$$

// Where η_t is a positive learning rate making $\boldsymbol{w}^{(t+1)}$ be acceptable for the next iteration.

Wolfe conditions

- To find the sequence $(\boldsymbol{w}^{(t)})_{t\in\mathbb{N}}$ following the line search rule, the following necessary condition

$$\forall t \in \mathbb{N}, \hat{\mathcal{L}}(\boldsymbol{w}^{(t+1)}) < \hat{\mathcal{L}}(\boldsymbol{w}^{(t)})$$

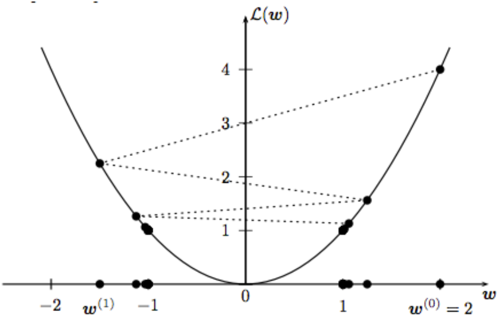
  is not sufficient to guarantee the convergence of the sequence to the minimiser of $\hat{\mathcal{L}}$.

- In two situations, the previous condition is satisfied but there is no convergence

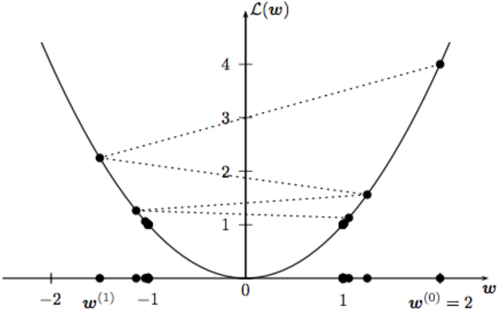
1. The decreasing of $\hat{\mathcal{L}}$ is too small with respect to the length of the jumps

Consider the following example $d = 1$; $\hat{\mathcal{L}}(\boldsymbol{w}) = \boldsymbol{w}^2$ with $\boldsymbol{w}^{(0)} = 2$, $(\mathbf{p}_t = (-1)^{t+1})_{t\in\mathbb{N}^*}$ and $(\eta_t = (2 + \frac{3}{2^{t+1}}))_{t\in\mathbb{N}^*}$. The sequence of updates would then be

$$\forall t \in \mathbb{N}^*, \boldsymbol{w}^{(t)} = (-1)^t(1 + 2^{-t})$$



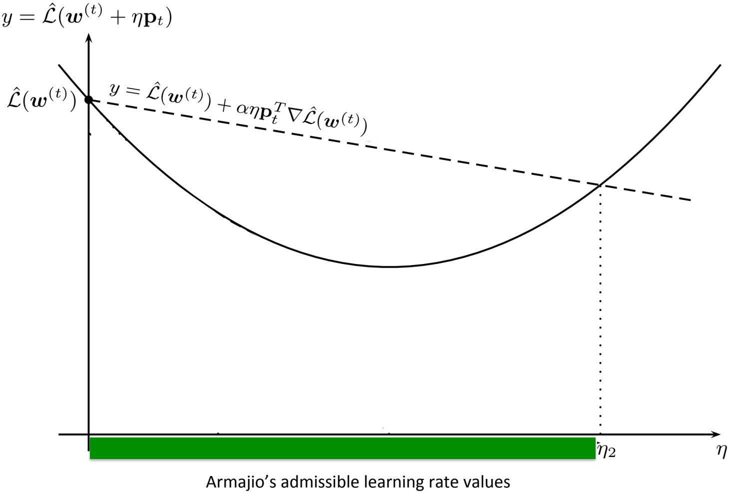
1. The decreasing of $\hat{\mathcal{L}}$ is too small with respect to the length of the jumps



$\Rightarrow$ Armijo condition : require that for a given $\alpha \in (0,1)$,

$$\forall t \in \mathbb{N}^*, \hat{\mathcal{L}}(\boldsymbol{w}^{(t)} + \eta_t \mathbf{p}_t) \leq \hat{\mathcal{L}}(\boldsymbol{w}^{(t)}) + \alpha \eta_t \mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)})$$

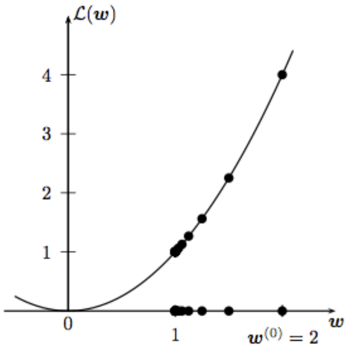
Armajio condition



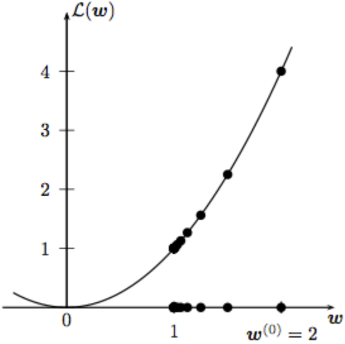
2. The jumps of the weight vectors are too small

Consider the following example $d = 1$; $\hat{\mathcal{L}}(\boldsymbol{w}) = \boldsymbol{w}^2$ with $\boldsymbol{w}^{(0)} = 2$, $(\mathbf{p}_t = -1)_{t\in\mathbb{N}^*}$ and $(\eta_t = (2^{-t+1}))_{t\in\mathbb{N}^*}$. The sequence of updates would then be

$$\forall t \in \mathbb{N}^*, \boldsymbol{w}^{(t)} = (1 + 2^{-t})$$



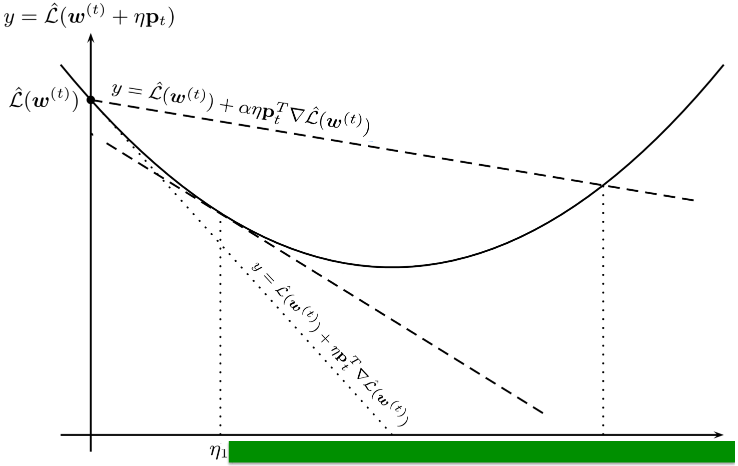
2. The jumps of the weight vectors are too small



$\Rightarrow \exists \beta \in (\alpha, 1)$ such that

$$\forall t \in \mathbb{N}^*, \mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)} + \eta_t \mathbf{p}_t) \geq \beta \mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)})$$

Armajio condition



Existence of learning rates verifying Wolfe conditions

- Let $\mathbf{p}_t$ be a descent direction of $\hat{\mathcal{L}}$ at $\boldsymbol{w}^{(t)}$. Suppose that the function $\psi_t : \eta \mapsto \hat{\mathcal{L}}(\boldsymbol{w}^{(t)} + \eta \mathbf{p}_t)$ is derivative and lower bounded, then there exists η_t verifying both Wolfe conditions.

proof:

1. consider

$$E = \{a \in \mathbb{R}_+ \mid \forall \eta \in]0, a], \hat{\mathcal{L}}(\boldsymbol{w}^{(t)} + \eta \mathbf{p}_t) \leq \hat{\mathcal{L}}(\boldsymbol{w}^{(t)}) + \alpha \eta \mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)})\}$$

As $\mathbf{p}_t$ is a descent direction of $\hat{\mathcal{L}}$ at $\boldsymbol{w}^{(t)}$ then for all $\alpha < 1$ there exists $\bar{a} > 0$ such that

$$\forall \eta \in]0, \bar{a}], \hat{\mathcal{L}}(\boldsymbol{w}^{(t)} + \eta \mathbf{p}_t) < \hat{\mathcal{L}}(\boldsymbol{w}^{(t)}) + \alpha \eta \mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)})$$

Existence of learning rates verifying Wolfe conditions

2. So $E \neq \emptyset$. Furthermore, as the function ψ_t is lower bounded, the largest rate in E, $\hat{\eta}_t = \sup E$, exists. By continuity of ψ_t we have

$$\hat{\mathcal{L}}(\boldsymbol{w}^{(t)} + \hat{\eta}_t \mathbf{p}_t) < \hat{\mathcal{L}}(\boldsymbol{w}^{(t)}) + \alpha \hat{\eta}_t \mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)})$$

3. Let $(\eta_n)_{n\in\mathbb{N}}$ be a convergence sequence to $\hat{\eta}_t$ by higher values, i.e. $\forall n \in \mathbb{N}, \eta_n > \hat{\eta}_t$ and $\lim_{n\to+\infty} \eta_n = \hat{\eta}_t$. As $(\eta_n)_{n\in\mathbb{N}} \notin E$ we get

$$\forall n \in \mathbb{N}, \hat{\mathcal{L}}(\boldsymbol{w}^{(t)} + \eta_n \mathbf{p}_t) > \hat{\mathcal{L}}(\boldsymbol{w}^{(t)}) + \alpha \eta_n \mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)})$$

So

$$\hat{\mathcal{L}}(\boldsymbol{w}^{(t)} + \hat{\eta}_t \mathbf{p}_t) = \hat{\mathcal{L}}(\boldsymbol{w}^{(t)}) + \alpha \hat{\eta}_t \mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)})$$

Existence of learning rates verifying Wolfe conditions

4. We finally get

$$\mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)} + \hat{\eta}_t \mathbf{p}_t) \geq \alpha \mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)}) \geq \beta \mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)})$$

Where $\beta \in (\alpha, 1)$ and $\mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)}) < 0$.

$\Rightarrow$ The learning rate $\hat{\eta}_t$ verifies both Wolfe conditions

Does it work?

Theorem (Zoutendijk)

Let $\hat{\mathcal{L}}:\mathbb{R}^d\rightarrow\mathbb{R}$ be a differentiable objective function with a lipschtizien gradient and lower bounded. Let $\mathfrak{A}$ be an algorithm generating $(\boldsymbol{w}^{(t)})_{t\in\mathbb{N}}$ defined by

$$\forall t\in\mathbb{N}, \boldsymbol{w}^{(t+1)}=\boldsymbol{w}^{(t)}+\eta_t\mathbf{p}_t$$

where $\mathbf{p}_t$ is a descent direction of $\hat{\mathcal{L}}$ and η_t a learning rate verifying both Wolfe conditions. By considering the angle θ_t between the descent direction $\mathbf{p}_t$ and the direction of the gradient :

$$\cos(\theta_t)=\frac{-\mathbf{p}_t^\top\nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t)})}{||\hat{\mathcal{L}}(\boldsymbol{w}^{(t)})||\times||\mathbf{p}_t||}$$

The following series is convergent

$$\sum_t \cos^2(\theta_t)||\nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t)})||^2$$

Proof of Zoutendijk's theorem

1. Using the second Wolfe's condition and by subtracting $\mathbf{p}_t^\top \nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t)})$ from both terms of the inequality, we get

$$\forall t, \mathbf{p}_t^\top (\nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t+1)}) - \nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t)})) \geq (\beta - 1) \left(\mathbf{p}_t^\top \nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t)})\right)$$

2. Using the lipschitzian property of the gradient of the objective function

$$\begin{aligned}
\mathbf{p}_t^\top (\nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t+1)}) - \nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t)})) &\leq ||\nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t+1)}) - \nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t)})|| \times ||\mathbf{p}_t|| \\
&\leq L||\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)}|| \times ||\mathbf{p}_t|| \\
&\leq L\eta_t||\mathbf{p}_t||^2
\end{aligned}$$

Proof of Zoutendijk's theorem

3. By combining both inequalities it comes

$$\forall t, 0 \leq (\beta - 1)(\mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)})) \leq L\eta_t ||\mathbf{p}_t||^2$$

4. For $\eta_t \geq \frac{\beta-1}{L} \frac{\mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)})}{||\mathbf{p}_t||^2} > 0$ we get from Armijo's condition

$$\begin{aligned}
\hat{\mathcal{L}}(\boldsymbol{w}^{(t)}) - \hat{\mathcal{L}}(\boldsymbol{w}^{(t+1)}) &\geq -\alpha\eta_t \mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)}) \\
&\geq \alpha \frac{1-\beta}{L} \frac{(\mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)}))^2}{||\mathbf{p}_t||^2} \\
&\geq \alpha \frac{1-\beta}{L} \cos^2(\theta_t) ||\nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)})||^2 \geq 0
\end{aligned}$$

Proof of Zoutendijk's theorem

5. The objective function is lower bounded, the sequence of general term $\hat{\mathcal{L}}(\boldsymbol{w}^{(t)}) - \hat{\mathcal{L}}(\boldsymbol{w}^{(t+1)}) > 0$ is convergent

6. Hence, the series

$$\sum_t \cos^2(\theta_t)||\nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t)})||^2$$

is convergent.

Corollary of Zoutendijk's theorem
Guarantee of convergence

- In the case where, the descente direction and the gradient are not orthogonal :

$$\exists\kappa > 0, \forall t \geq T, \cos^2(\theta_t) \geq \kappa$$

- Following Zoutendijk's theorem the series :

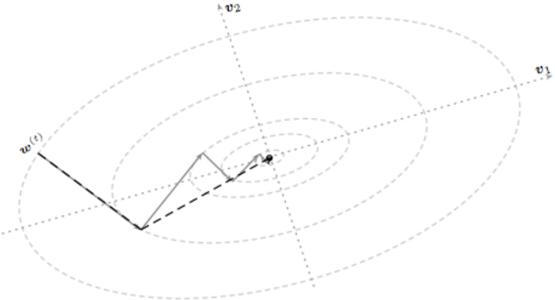
$$\sum_t ||\nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t)})||^2$$

is convergent.

- Hence, the sequence $(\nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t)}))_t$ tends to 0 when t tends to infinity.

Can we do better? Conjugate gradient method

- The adaptive search of the learning rate with the line search algorithm does not prevent the oscillations of the weight vector around the minimiser of the objective function



Conjugate gradient method

- One solution to this problem is to require that the gradient at point $\boldsymbol{w}^{(t+1)} + \eta \mathbf{p}_{t+1}$ be orthogonal to the previous direction $\mathbf{p}_t$
$$\mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t+1)} + \eta \mathbf{p}_{t+1}) = 0$$
- Consider the Taylor development of $\eta \mapsto \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t+1)} + \eta \mathbf{p}_{t+1})$

$$\nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t+1)} + \eta \mathbf{p}_{t+1}) = \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t+1)}) + \eta \mathbf{H}^{(t+1)} \mathbf{p}_{t+1}$$

It comes
$$\mathbf{p}_t^\top \mathbf{H}^{(t+1)} \mathbf{p}_{t+1} = 0$$

Directions $\mathbf{p}_t$ and $\mathbf{p}_{t+1}$ are said to be conjugate.

Conjugate gradient method

- At the neighbourhood of the minimiser of the objective function, where the quadratic approximation holds
- Suppose that we have d conjuguante directions $\{\mathbf{p}_t, t \in [\![0, d-1]\!]\}$

$$\forall (t, t') \in [\![0, d-1]\!]^2, t \neq t', \mathbf{p}_t^\top \mathbf{H} \mathbf{p}_{t'} = 0$$

- As the Hessian matrix is symmetric positive definite, we can show that the directions $\{\mathbf{p}_t\}$ are linearly independent and that they form a basis. We have

$$\boldsymbol{w}^* - \boldsymbol{w}^{(0)} = \sum_{t=0}^{d-1} \eta_t \mathbf{p}_t$$

Hence

$$\forall t, \eta_t = \frac{\mathbf{p}_t^\top \mathbf{H} (\boldsymbol{w}^* - \boldsymbol{w}^{(0)})}{\mathbf{p}_t^\top \mathbf{H} \mathbf{p}_t}$$

Conjugate gradient method

- Let

$$\boldsymbol{w}^{(t)} = \boldsymbol{w}^{(0)} + \sum_{i=0}^{t-1} \eta_i \mathbf{p}_i$$

We get the following update rule

$$\forall t \in [\![0, d-1]\!], \boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta_t \mathbf{p}_t$$

- From the mutual conjuguate property of $(\mathbf{p}_t)_{t=0}^{d-1}$:

$$\mathbf{p}_t^\top \mathbf{H} \boldsymbol{w}^{(t)} = \mathbf{p}_t^\top \mathbf{H} \boldsymbol{w}^{(0)}$$

That is

$$\forall t, \eta_t = -\frac{\mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)})}{\mathbf{p}_t^\top \mathbf{H} \mathbf{p}_t} \tag{5}$$

Conjugate gradient method

- With the previous definition of learning rates, it is simple to show that the current gradient is orthogonal to all the previous descent directions. In fact

$$\forall t, \nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t+1)}) - \nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t)}) = \mathbf{H}(\underbrace{\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)}}_{\eta_t\mathbf{p}_t})$$

- By multiplying $\mathbf{p}_t$ from the left and by the definition of η_t

$$\forall t, \mathbf{p}_t^\top(\nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t+1)}) - \nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t)})) = -\mathbf{p}_t^\top\nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t)})$$

Which gives

$$\forall t, \mathbf{p}_t^\top\nabla\hat{\mathcal{L}}(\boldsymbol{w}^{(t+1)}) = 0$$

Conjugate gradient method

- For a given index $t \in [\![0, d-1]\!]$, we finally get

$$\forall t', \forall t, t < t', \mathbf{p}_t^\top \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t')}) = 0 \tag{6}$$

- Hence, if the descent directions are conjugate after d updates

$$\forall t \in [\![0, d-1]\!], \boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta_t \mathbf{p}_t$$

  using the learning rate above, we arrive to a point where the gradient of the objective function at this point is orthogonal to all the descent direction, and which is the minimum.

Conjugate gradient algorithm

- The necessary condition to get the previous result is to have descent directions $(\mathbf{p}_t)_{t=0}^{d-1}$ that are mutually conjugated
- The following sequence

$$\begin{cases} \mathbf{p}_0 = -\nabla\hat{\mathcal{L}}(w^{(0)}) \\ \mathbf{p}_{t+1} = -\nabla\hat{\mathcal{L}}(w^{(t+1)}) + \beta_t\mathbf{p}_t \quad \text{si } t \geq 0 \end{cases}$$

- For

$$\forall t, \beta_t = \frac{\mathbf{p}_t^\top \mathbf{H} \nabla\hat{\mathcal{L}}(w^{(t+1)})}{\mathbf{p}_t^\top \mathbf{H} \mathbf{p}_t}$$

It is easy to show that the descent directions are mutually conjugated.

Conjugate gradient algorithm

- The coefficients (β_t) can be estimated without the use of the Hessian matrix (Hestenes and Stiefel, 52)

$$\forall t, \beta_t = \frac{\nabla^\top \hat{\mathcal{L}}(w^{(t+1)})\mathbf{H}\mathbf{p}_t}{\mathbf{p}_t^\top \mathbf{H}\mathbf{p}_t} = \frac{\nabla^\top \hat{\mathcal{L}}(w^{(t+1)})(\nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t+1)}) - \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)}))}{\mathbf{p}_t^\top (\nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t+1)}) - \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)}))}$$

Followed by others

$$\forall t, \beta_t = \frac{\nabla^\top \hat{\mathcal{L}}(w^{(t+1)})(\nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t+1)}) - \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)}))}{\nabla^\top \hat{\mathcal{L}}(w^{(t)})\nabla \hat{\mathcal{L}}(w^{(t)})} \text{ (Polak and Ribiere, 69)}$$

$$\forall t, \beta_t = \frac{\nabla^\top \hat{\mathcal{L}}(w^{(t+1)})\nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t+1)})}{\nabla^\top \hat{\mathcal{L}}(w^{(t)})\nabla \hat{\mathcal{L}}(w^{(t)})} \text{ (Fletcher and Reeves, 64)}$$

Conjugate gradient algorithm

```
void grdcnj(double **X, double *Y, long int m, long int d, double *w, double epsilon)
{
  long int    j, Epoque=0;
  double      *wold, OldLoss, NewLoss, *g, *p, *h, dgg, ngg, beta;
 //  wold,    p, g, h allocated
  for(j=0; j<=d; j++)
     wold[j]= 2.0*(rand() / (double) RAND_MAX)-1.0;
  NewLoss = FoncLoss(wold,  X, Y, m, d);
  OldLoss = NewLoss + 2*epsilon;
  g  = Gradient(wold, X, Y, m, d);
 for(j=0; j<=d; j++)
    p[j] = -g[j];  //  ▷ p_0 ← −∇L̂(w^(0))

  while(fabs(OldLoss-NewLoss) > (fabs(OldLoss)*epsilon))
  {
    OldLoss = NewLoss;
    rchln(wold, OldLoss, g, p, w, &NewLoss, X, Y, m, d);
    h  = Gradient(w, X, Y, m, d); // New gradient  ▷ ∇L̂(w^(t+1))
    for(dgg=0.0, ngg=0.0, j=0; j<=d; j++){
       dgg+=g[j]*g[j];
       ngg+=h[j]*h[j];
     }
    beta=ngg/dgg;
    for(j=0; j<=d; j++){
       wold[j]=w[j];
       g[j]=h[j];
       p[j]=-g[j]+beta*p[j]; // New descent direction
    }
  }
 }
```

Logisitc Regression Program

```
// Logistic function  x ↦ 1/(1+e^{-x})
double Logistic(double x)
{
   return (1.0/(1.0+exp(-x)));
}

// Estimation of the gradient vector
double *Gradient(double *w, double **X, double *y, long int m, long int d)
{
  double   ps, *g;
  long int i, j;

  g=(double *)malloc((d+1)*sizeof (double));
  for(j=0; j<=d; j++)
    g[j]=0.0;

  for(i=1; i<=m; i++){
     for(ps=w[0],j=1; j<=d; j++)
       ps+=w[j]*X[i][j];
     g[0]+=(Logistic(y[i]*ps)-1.0)*y[i];
     for(j=1; j<=d; j++)
       g[j]+=(Logistic(y[i]*ps)-1.0)*y[i]*X[i][j];
   }

  for(j=0; j<=d; j++)
        g[j]/=(double ) m;

  return(g);
}
```

Logisitc Regression Program

```
double FoncLoss(double *w, double **X, double *y, long int m, long int d)
{
  double   S=0.0, ps;
  long int i, j;

  for(i=1; i<=m; i++){
     for(ps=w[0],j=1; j<=d; j++)
       ps+=w[j]*X[i][j];
     S+= log(1.0+exp(-y[i]*ps));
  }
  S/=(double ) m;

  return (S);
}

void RegressionLogistique(double *w, DATA TrainingSet, LR_PARAM params)
{
   // Minimization of the logistic loss using the gradient conjuguate

    grdcnj(TrainingSet.X, TrainingSet.y, TrainingSet.m, TrainingSet.d, w, params.eps);

}
```

source: http://ama.liglab.fr/~amini/LR/

Consistency of the ERM principle

Estimation of the generalization error on a test set

- Remind that the examples of a test set are generated i.i.d. with respect to the same probability distribution $\mathcal{D}$ which has generated the training set,
- Consider f_S a learned function over the training set S, and let $T = \{(\mathbf{x}_i, y_i); i \in \{1, \dots, n\}\}$ be a test set of size n,
- $(f_S(\mathbf{x}_i), y_i) \mapsto L(f_S(\mathbf{x}_i), y_i)$ can be considered as the independent copies of the same random variable :

$$\begin{aligned}\mathbb{E}_{T\sim\mathcal{D}^n}\hat{\mathfrak{L}}(f_S, T) &= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{T\sim\mathcal{D}^n}L(f_S(\mathbf{x}_i), y_i)\\ &= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}L(f_S(\mathbf{x}), y) = \mathfrak{L}(f_S)\end{aligned}$$

$\Rightarrow$ The empirical error of f_S on the test set, $\hat{\mathfrak{L}}(f_S, T)$ is an unbiased estimator of its generalization error.

[Hoeffding 63] Inequality

Let $X_1, \ldots, X_n$ be independent random variables and define the empirical mean of these variables : $S_n = X_1 + \cdots + X_n$. Assume that the X_i are almost surely bounded within the interval $[a_i, b_i]$. Then for any $\epsilon > 0$, the Theorem 2 of Hoeffding proves the inequalities

$$\mathbb{P}\left(S_n - \mathbb{E}[S_n] \geq \epsilon\right) \leq \exp\left(-\frac{2n^2\epsilon^2}{\sum_{i=1}^n (b_i - a_i)^2}\right)$$

$$\mathbb{P}\left(\mid S_n - \mathbb{E}[S_n] \mid \geq \epsilon\right) \leq 2\exp\left(-\frac{2n^2\epsilon^2}{\sum_{i=1}^n (b_i - a_i)^2}\right)$$

Estimation of the generalization error on a test set

- For each test example $(\mathbf{x}_i, y_i)$ let X_i be the random variable $\frac{1}{n}L(f_S(\mathbf{x}_i), y_i)$
- Further, all the random variables $X_i, i \in \{1, \dots, n\}$ are independent and that they take values in $\{0, 1\}$
- By noting that $\hat{\mathfrak{L}}(f_S, T) = \sum_{i=1}^n X_i$ and $\mathfrak{L}(f_S) = \mathbb{E}\left(\sum_{i=1}^n X_i\right)$, we have the following using [Hoeffding 63] inequality

$$\forall \epsilon > 0, \mathbb{P}\left(\mathfrak{L}(f_S) - \hat{\mathfrak{L}}(f_S, T) > \epsilon\right) \leq e^{-2n\epsilon^2}$$

- To better understand this result, let solve the equation $e^{-2n\epsilon^2} = \delta$ with respect to ϵ, hence we have $\epsilon = \sqrt{\frac{\ln 1/\delta}{2n}}$ and :

$$\forall \delta \in]0, 1], \mathbb{P}\left(\mathfrak{L}(f_S) \leq \hat{\mathfrak{L}}(f_S, T) + \sqrt{\frac{\ln 1/\delta}{2n}}\right) \geq 1 - \delta$$

Estimation of the generalization error on a test set

- For a small δ, according to the previous equation, we have the following inequality which stands with high probability and all test sets of size n :

$$\mathfrak{L}(f_S) \leq \hat{\mathfrak{L}}(f_S, T) + \sqrt{\frac{\ln 1/\delta}{2n}}$$

- From this result, we have a bound over the generalization error of a learned function which can be estimated using any test set, and in the case where n is sufficiently large, this bound gives a very accurate estimated of the latter.
- Example: suppose that the empirical error of a prediction function f_S over a test set T of size $n = 1000$ is $\hat{\mathfrak{L}}(f_S, T) = 0.23$. For $\delta = 0.01$, i.e. $\sqrt{\frac{\ln(1/\delta)}{2n}} \approx 0.047$, the generalization error of f_S is upperbounded by 0.277 with a probability at least 0.99.

A uniform generalization error bound

- As part of the study of the consistency of the ERM principle, we would now establish a uniform bound on the generalization error of a learned function depending on its empirical error over a training base.
- We cannot reach this result, by using the same development than previously.
- This is mainly due to the fact that when the learned function f_S has knowledge of the training data $S = \{(\mathbf{x}_i, y_i); i \in \{1, \ldots, m\}\}$, random variables $X_i = \frac{1}{m} L(f_S(\mathbf{x}_i), y_i); i \in \{1, \ldots, m\}$ involved in the estimation of the empirical error of f_S on S, are all dependent on each other.
  $\Rightarrow$ Indeed, if we change an example of the training set, the selected function f_S will also change, as well as the instantaneous errors of all the other examples.

Rademacher complexity [Koltchinskii 01]

- In the derivation of uniform generalization error bounds different capacity measures of the class of functions have been proposed. Among which the Rademacher complexity allows an accurate estimates of the capacity of a class of functions and it is dependent to the training sample

- The empirical Rademacher complexity estimates the richness of a function class $\mathcal{F}$ by measuring the degree to which the latter is able fit to random noise on a training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ of size m generated i.i.d. with respect to a probability distribution $\mathcal{D}$.

Rademacher complexity

- This complexity is estimated through Rademacher variables $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_m)^\top$ which are independent discrete random variables taking values in $\{-1, +1\}$ with the same probability 1/2, i.e. $\forall i \in \{1, \ldots, m\}; \mathbb{P}(\sigma_i = -1) = \mathbb{P}(\sigma_i = +1) = 1/2$, and is defined as :

$$\hat{\mathfrak{R}}_m(\mathcal{F}, S) = \frac{2}{m} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{f \in \mathcal{F}} \left| \sum_{i=1}^{m} \sigma_i f(\mathbf{x}_i) \right| \mid \mathbf{x}_1, \ldots, \mathbf{x}_m \right]$$

- Furthermore, we define the Rademacher complexity of the class of functions $\mathcal{F}$ independently to a given training set by

$$\mathfrak{R}_m(\mathcal{F}) = \mathbb{E}_{S \sim \mathcal{D}^m} \hat{\mathfrak{R}}_m(\mathcal{F}, S) = \frac{2}{m} \mathbb{E}_{S\boldsymbol{\sigma}} \left[\sup_{f \in \mathcal{F}} \left| \sum_{i=1}^{m} \sigma_i f(\mathbf{x}_i) \right| \right]$$

A uniform generalization error bound

Theorem (Generalization bound with the Rademacher complexity)

Let $\mathcal{X} \in \mathbb{R}^d$ *be a vectoriel space and* $\mathcal{Y} = \{-1, +1\}$ *an output space. Suppose that the pairs of examples* $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ *are generated i.i.d. with respect to the distribution probability* $\mathcal{D}$*. Let* $\mathcal{F}$ *be a class of functions having values in* $\mathcal{Y}$ *and* $L : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$ *a given instantaneous loss. Then for all* $\delta \in]0, 1]$*, we have with probability at least* $1 - \delta$ *the following inequality :*

$$\forall f \in \mathcal{F}, \mathfrak{L}(f) \leq \hat{\mathfrak{L}}(f, S) + \mathfrak{R}_m(L \circ \mathcal{F}) + \sqrt{\frac{\ln \frac{1}{\delta}}{2m}} \tag{7}$$

and also with probability at least $1 - \delta$

$$\mathfrak{L}(f) \leq \hat{\mathfrak{L}}(f, S) + \hat{\mathfrak{R}}_m(L \circ \mathcal{F}, S) + 3\sqrt{\frac{\ln \frac{2}{\delta}}{2m}} \tag{8}$$

Where $L \circ \mathcal{F} = \{(\mathbf{x}, y) \mapsto L(f(\mathbf{x}), y) \mid f \in \mathcal{F}\}$.

A uniform generalization error bound (1)

1. Link the supremum of $\mathfrak{L}(f) - \hat{\mathfrak{L}}(f, S)$ on $\mathcal{F}$ with its expectation

   The study of this bound is achieved by linking the supremum appearing, in the right hand side of the above inequality, with its expectation through a powerful tool developed for empirical processes by [McDiarmid 89], and known as the theorem of bounded differences

Let $I \subset \mathbb{R}$ be a real valued interval, and $(X_1, ..., X_m)$, m independent random variables taking values in I^m. Let $\Phi : I^m \rightarrow \mathbb{R}$ be defined such that : $\forall i \in \{1, ..., m\}, \exists c_i \in \mathbb{R}$ the following inequality holds for any $(x_1, ..., x_m) \in I^m$ and $\forall x' \in I$:

$$|\Phi(x_1, .., x_{i-1}, x_i, x_{i+1}, .., x_m) - \Phi(x_1, .., x_{i-1}, x', x_{i+1}, .., x_m)| \leq c_i$$

We have then

$$\forall \epsilon > 0, \mathbb{P}(\Phi(x_1, ..., x_m) - \mathbb{E}[\Phi] > \epsilon) \leq e^{\frac{-2\epsilon^2}{\sum_{i=1}^{m} c_i^2}}$$

A uniform generalization error bound (1)

1. Link the supremum of $\mathfrak{L}(f) - \hat{\mathfrak{L}}(f, S)$ on $\mathcal{F}$ with its expectation

consider the following function

$$\Phi : S \mapsto \sup_{f \in \mathcal{F}} [\mathfrak{L}(f) - \hat{\mathfrak{L}}(f, S)]$$

Mcdiarmid inequality can then be applied for the function Φ with $c_i = 1/m, \forall i$, thus :

$$\forall \epsilon > 0, \mathbb{P}\left(\sup_{f \in \mathcal{F}} [\mathfrak{L}(f) - \hat{\mathfrak{L}}(f, S)] - \mathbb{E}_S \sup_{f \in \mathcal{F}} [\mathfrak{L}(f) - \hat{\mathfrak{L}}(f, S)] > \epsilon\right) \leq e^{-2m\epsilon^2}$$

A uniform generalization error bound (2)

2. Bound $\mathbb{E}_S \sup_{f\in\mathcal{F}}[\mathfrak{L}(f) - \hat{\mathfrak{L}}(f,S)]$ with respect to $\mathfrak{R}_m(L\circ\mathcal{F})$

This step is a symmetrisation step and it consists in introducing a second virtual sample S' also generated i.i.d. with respect to $\mathcal{D}^m$ into $\mathbb{E}_S \sup_{f\in\mathcal{F}}[\mathfrak{L}(f) - \hat{\mathfrak{L}}(f,S)]$.

$$
\begin{aligned}
\rightarrow \mathbb{E}_S \sup_{f\in\mathcal{F}}(\mathfrak{L}(f) - \hat{\mathfrak{L}}(f,S)) &= \mathbb{E}_S \sup_{f\in\mathcal{F}}[\mathbb{E}_{S'}(\hat{\mathfrak{L}}(f,S') - \hat{\mathfrak{L}}(f,S))]\\
&\leq \mathbb{E}_S\mathbb{E}_{S'} \sup_{f\in\mathcal{F}}[\mathfrak{L}(f,S') - \hat{\mathfrak{L}}(f,S)]
\end{aligned}
$$

$\rightarrow$ In the other hand,

$$
\begin{aligned}
&\mathbb{E}_S\mathbb{E}_{S'} \sup_{f\in\mathcal{F}}[\mathfrak{L}(f,S') - \hat{\mathfrak{L}}(f,S)]\\
&= \mathbb{E}_S\mathbb{E}_{S'}\mathbb{E}_{\boldsymbol{\sigma}} \sup_{f\in\mathcal{F}}\left[\frac{1}{m}\sum_{i=1}^{m}\sigma_i(L(f(\mathbf{x}'_i),y'_i) - L(f(\mathbf{x}_i),y_i))\right]
\end{aligned}
$$

A uniform generalization error bound (2)

2. Bound $\mathbb{E}_S \sup_{f\in\mathcal{F}}[\mathfrak{L}(f) - \hat{\mathfrak{L}}(f,S)]$ with respect to $\mathfrak{R}_m(L\circ\mathcal{F})$

By applying the triangular inequality $\sup = ||.||_\infty$ it comes

$$\mathbb{E}_S\mathbb{E}_{S'}\mathbb{E}_{\boldsymbol{\sigma}} \sup_{f\in\mathcal{F}}\left[\frac{1}{m}\sum_{i=1}^m \sigma_i(L(f(\mathbf{x}'_i),y'_i) - L(f(\mathbf{x}_i),y_i))\right] \leq$$

$$\mathbb{E}_S\mathbb{E}_{S'}\mathbb{E}_{\sigma} \sup_{f\in\mathcal{F}}\frac{1}{m}\sum_{i=1}^m \sigma_i L(f(\mathbf{x}'_i),y'_i) + \mathbb{E}_S\mathbb{E}_{S'}\mathbb{E}_{\boldsymbol{\sigma}} \sup_{f\in\mathcal{F}}\frac{1}{m}\sum_{i=1}^m (-\sigma_i) L(f(\mathbf{x}'_i),y'_i)$$

Finally as $\forall i, \sigma_i$ and $-\sigma_i$ have the same distribution we have

$$\mathbb{E}_S\mathbb{E}_{S'} \sup_{f\in\mathcal{F}}[\mathfrak{L}(f,S') - \hat{\mathfrak{L}}(f,S)] \leq \underbrace{2\mathbb{E}_S\mathbb{E}_{\boldsymbol{\sigma}} \sup_{f\in\mathcal{F}}\frac{1}{m}\sum_{i=1}^m \sigma_i L(f(\mathbf{x}_i),y_i)}_{\leq \mathfrak{R}_m(L\circ\mathcal{F})} \tag{9}$$

A uniform generalization error bound (2)

2. Bound $\mathbb{E}_S \sup_{f\in\mathcal{F}}[\mathfrak{L}(f) - \hat{\mathfrak{L}}(f,S)]$ with respect to $\mathfrak{R}_m(L\circ\mathcal{F})$

In summarizing the results obtained so far, we have:

1. $\forall f\in\mathcal{F}, \forall S, \mathfrak{L}(f) - \hat{\mathfrak{L}}(f,S) \leq \sup_{f\in\mathcal{F}}[\mathfrak{L}(f) - \hat{\mathfrak{L}}(f,S)]$
2. $\forall \epsilon > 0, \mathbb{P}\left(\sup_{f\in\mathcal{F}}[\mathfrak{L}(f) - \hat{\mathfrak{L}}(f,S)] - \mathbb{E}_S \sup_{f\in\mathcal{F}}[\mathfrak{L}(f) - \hat{\mathfrak{L}}(f,S)] > \epsilon\right) \leq e^{-2m\epsilon^2}$
3. $\mathbb{E}_S \sup_{f\in\mathcal{F}}(\mathfrak{L}(f) - \hat{\mathfrak{L}}(f,S)) \leq \mathfrak{R}_m(L\circ\mathcal{F})$

The first point of the theorem 2 is obtained by resolving the equation $e^{-2m\epsilon^2} = \delta$ with respect to ϵ.

A uniform generalization error bound (3)

3. Bound $\mathfrak{R}_m(L \circ \mathcal{F})$ with respect to $\hat{\mathfrak{R}}_m(L \circ \mathcal{F}, S)$

$\rightarrow$ Apply the McDiarmid inequality to the function $\Phi : S \mapsto \hat{\mathfrak{R}}_m(L \circ \mathcal{F}, S)$

$$\forall \epsilon > 0, \mathbb{P}(\mathfrak{R}_m(L \circ \mathcal{F}) > \hat{\mathfrak{R}}_m(L \circ \mathcal{F}, S) + \epsilon) \leq e^{-m\epsilon^2/2}$$

Thus for $\delta/2 = e^{-m\epsilon^2/2}$, we have with probability at least equal to $1 - \delta/2$:

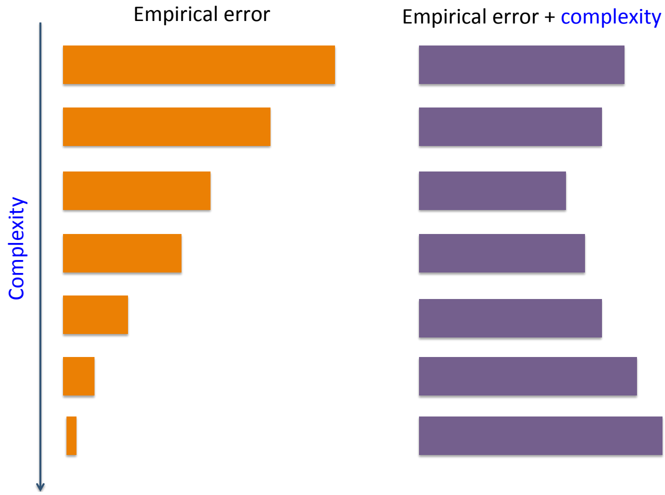
$$\mathfrak{R}_m(L \circ \mathcal{F}) \leq \hat{\mathfrak{R}}_m(L \circ \mathcal{F}, S) + 2\sqrt{\frac{\ln \frac{2}{\delta}}{2m}}$$

From the first point (Eq. 7) of the theorem 2, we have also with probability at least equal to $1 - \delta/2$:

$$\forall f \in \mathcal{F}, \forall S, \mathfrak{L}(f) \leq \hat{\mathfrak{L}}(f, S) + \mathfrak{R}_m(L \circ \mathcal{F}) + \sqrt{\frac{\ln \frac{2}{\delta}}{2m}}$$

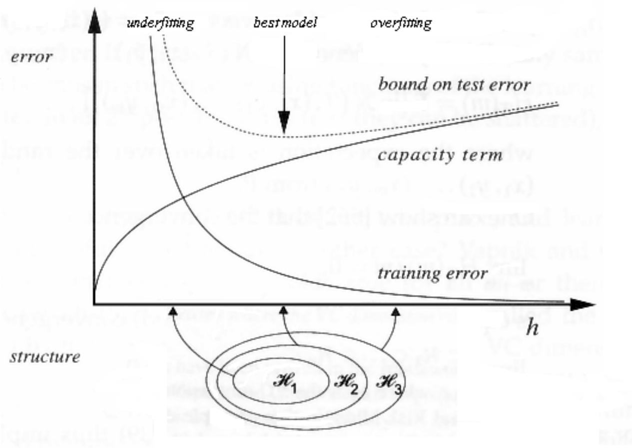
The second point (Eq. 8) of the theorem 2 is then obtained by combining the two previous results using the union bound.

Structural Risk Minimization



Structural Risk Minimization (2)

Image from : `http://www.svms.org/srm/`



Regularization

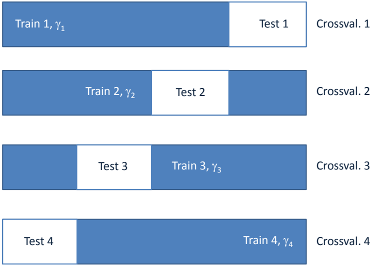
- Find a predictor by minimising the empirical risk with an added penalty for the size of the model,
- A simple approach consists in choosing a large class of functions $\mathcal{F}$ and to define on $\mathcal{F}$ a *regularizer*, typically a norm $|| \, g \, ||$, then to minimize the regularized empirical risk

$$\hat{f} = \underset{f\in\mathcal{F}}{\operatorname{argmin}} \hat{R}_m(f,S) + \underbrace{\gamma}_{hyperparameter} \times || \, f \, ||^2$$

- The hyper parameter, or the *regularisation parameter* allows to choose the right trade-off between fit and complexity.

K-fold cross validation

- Create a K-fold partition of the dataset
  - For each of K experiments, use $K-1$ folds for training and a different fold for testing, this procedure is illustrated in the following figure for $K=4$



- The value of the hyper parameter corresponds to the value of γ_k for which the testing performance is the highest on one of the folds.

In summary

- For induction, we should control the capacity of the class of functions.
- The study of the consistency of the ERM principle led to the second fundamental principle of machine learning called structural risk minimization (SRM).
- Learning is a compromise between a low empirical risk and a high capacity of the class of functions in use.

Homework

- Divide randomly each of the following datasets on 60% Training and 40% Test sets
  http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29
  http://archive.ics.uci.edu/ml/datasets/Ionosphere
  http://archive.ics.uci.edu/ml/datasets/Mushroom

- Learn each Perceptron, Adaline, Logistic Regression, Adaboost with perceptron on the training sets

- Compare their accuracy on the test sets

References

Massih-Reza Amini
Apprentissage Machine de la théorie à la pratique
éditions Eyrolles, 2015.

W. Hoeffding
Probability inequalities for sums of bounded random variables
Journal of the American Statistical Association, 58:13–30, 1963.

C. McDiarmid
On the method of bounded differences
Surveys in combinatorics, 141:148–188, 1989.

V. Koltchinskii
Rademacher penalties and structural risk minimization
IEEE Transactions on Information Theory, 47(5):1902–1914, 2001.

Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalker
Foundations of Machine Learning
2012.

References

A.B. Novikoff
On convergence proofs on perceptrons.
Symposium on the Mathematical Theory of Automata, 12: 615–622.
1962

F. Rosenblatt
The perceptron: A probabilistic model for information storage and organization in the brain.
Psychological Review, 65: 386–408.
1958

D. E. Rumelhart, G. E. Hinton and R. Williams
Learning internal representations by error propagation.
Parallel Distributed Processing: Explorations in the Microstructure of Cognition,
1986

R.E. Schapire
Theoretical views of boosting and applications.
In Proceedings of the 10th International Conference on Algorithmic Learning Theory, pages 13–25.
1999

References

G. Widrow and M. Hoff
Adaptive switching circuits.
Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record, 4: 96–104, 1960.

V. Vapnik.
The nature of statistical learning theory.
Springer, Verlag, 1998.