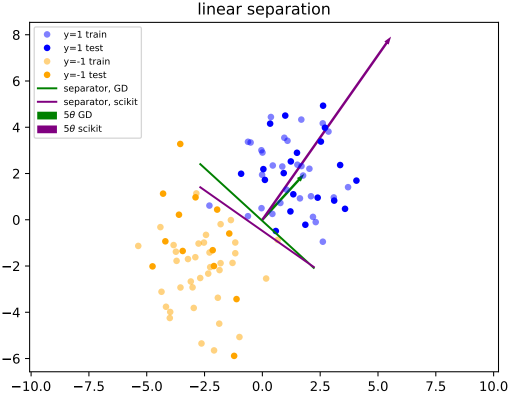
Fondamentaux théoriques du machine learning



Overview of lecture 4

Ridge regression

Gradient algorithms

Classification

- Problem statement
- Convexification of the risk and calibration
- Logistic regression

Dimensionality reduction

- Principal component analysis

Regularization of the empirical risk

In the previous practical session, we studied, Ridge regression, a form **regularization** of the objective function for the linear regression problem.
This enforces the unicity of the solution, even when $X^T X$ is not invertible (X is the design matrix like before) at the cost of introducing a **bias** in the estimator. The unicity is garanteed by the **strong convexity** of the new loss function (studies in the next exercises).

Ridge regression estimator

We use the same notations as in the previous lectures.

- $X \in \mathbb{R}^{n,d}$
- $y \in \mathbb{R}^n$
- $\theta \in \mathbb{R}^d$

$$\hat{\theta}_\lambda = \underset{\theta \in \mathbb{R}^d}{\arg\min} \left(\frac{1}{n} ||y - X\theta||_2^2 + \lambda ||\theta||_2^2 \right) \tag{1}$$

with $\lambda > 0$.

Ridge regression estimator

Proposition

The Ridge regression estimator is unique even if X^TX is not inversible and is given by

$$\hat{\theta}_\lambda = \frac{1}{n}(\hat{\Sigma} + \lambda I_d)^{-1}X^Ty$$

with

$$\hat{\Sigma} = \frac{1}{n}X^TX \in \mathbb{R}^{d,d} \tag{2}$$

Statistical analysis of ridge regression

We can compare the excess risk (difference between risk and Bayes risk) to that of OLS.

Proposition

Under the linear model assumption, with fixed design setting, the ridge regression estimator has the following excess risk

$$E[R(\hat{\theta}_\lambda)] - R^* = \lambda^2 \theta^{*T} (\hat{\Sigma} + \lambda I_d)^{-2} \hat{\Sigma} \theta^* + \frac{\sigma^2}{n} tr[\hat{\Sigma}^2 (\hat{\Sigma} + \lambda I_d)^{-2}] \quad (3)$$

Choice of λ

Is it possible that the excess risk is smaller with ridge regression than OLS ?

Proposition

With the choice

$$\lambda^* = \frac{\sigma\sqrt{tr(\hat{\Sigma})}}{||\theta^*||_2\sqrt{n}} \tag{4}$$

then

$$E[R(\hat{\theta}_\lambda] - R^* \leq \frac{\sigma\sqrt{tr(\hat{\Sigma})}||\theta^*||_2}{\sqrt{n}} \tag{5}$$

Choice of λ

Ridge

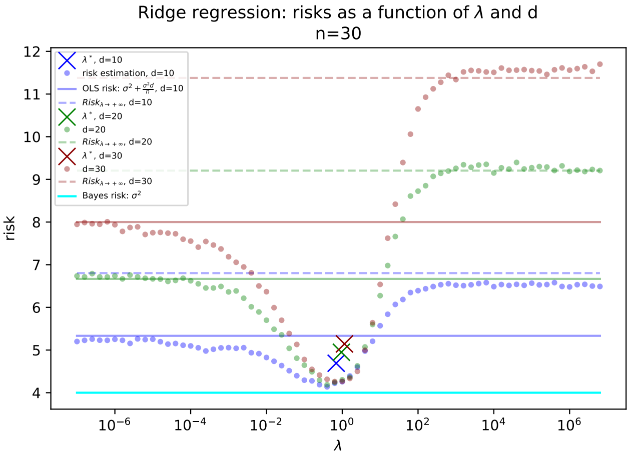
$$E[R(\hat{\theta}_\lambda] - R^* \leq \frac{\sigma\sqrt{tr(\hat{\Sigma})}||\theta^*||_2}{\sqrt{n}} \tag{6}$$

OLS

$$E[R(\hat{\theta}] - R^* = \sigma^2\frac{d}{n} \tag{7}$$

- $\frac{1}{n}$ (OLS) vs $\frac{1}{\sqrt{n}}$ (ridge), with different constants
- dimension-free bound for Ridge (maybe in the project)

Optimal λ



Hyperparameter search

- In practical situations, the quantities involved in the computation of λ^* in 4 are typically unknown. However this equation shows that there may **exist** a λ with a better prediction performance than OLS, which can be found by cross validation in practice (previous and next TP).
- λ is an example of **hyperparameter**.

Neural networks

With neural networks, it seems that it is possible to have $d >> n$ but no overfitting (simplicity bias).

Context

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

Context

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

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

$$\hat{\theta} = (X^T X)^{-1} X^T y \tag{8}$$

High d might be prohibitive for a numerical resolution.

Context

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

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

Context

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

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

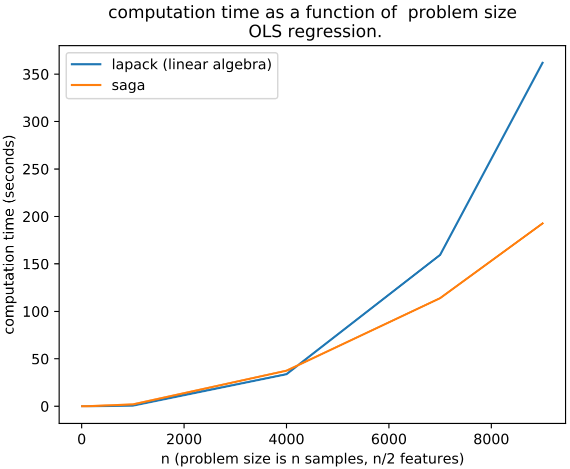
Convergence speed

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

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

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

SGD vs Lapack



Gradient descent

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

$$\theta \leftarrow \theta - \gamma \nabla_f(\theta) \tag{9}$$

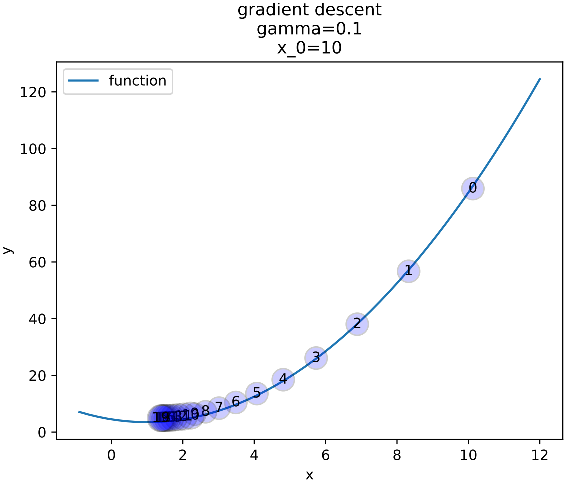
γ is called the **learning rate**, and is a very important hyperparameter. We will study some methods to tune it in practical sessions, like line seach.

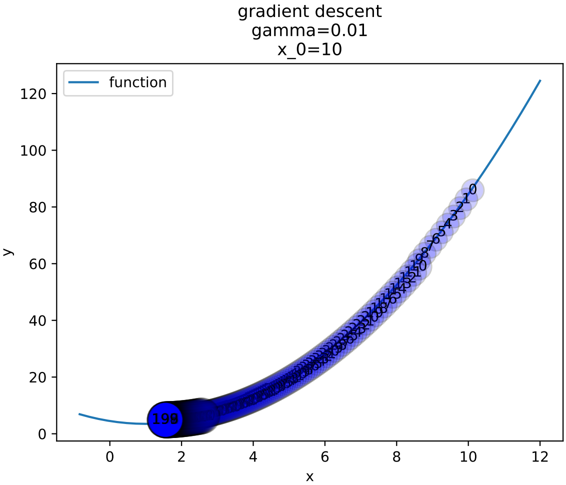
First-order method

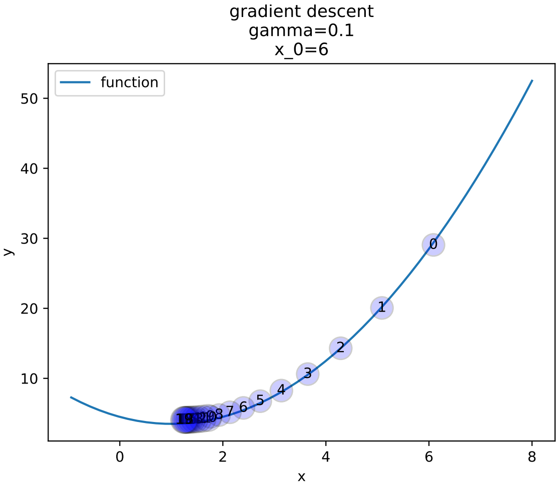
GD is called a first-order method because it makes use of the first order derivative.
The direction $-\nabla_\theta f(\theta)$ is the direction that minimizes the first order Taylor expansion of f in θ.

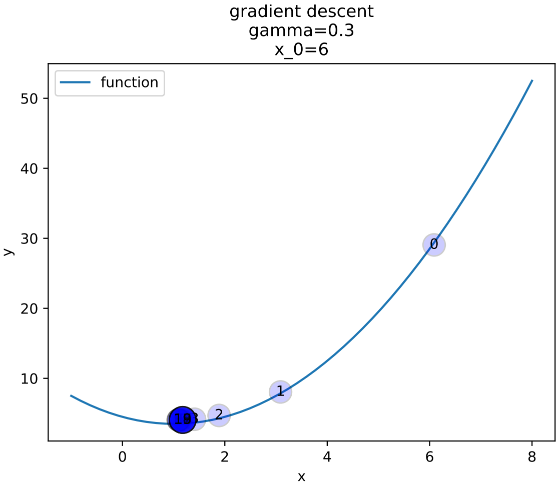
$$f(\theta + h) = f(\theta) + \langle \nabla_\theta f(\theta) | h \rangle + o(h) \tag{10}$$

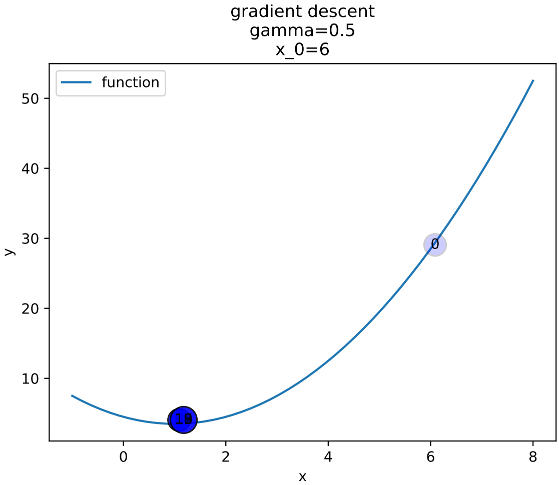
In the following examples, we use the function $f : x \rightarrow (x-1)^2 + 3$

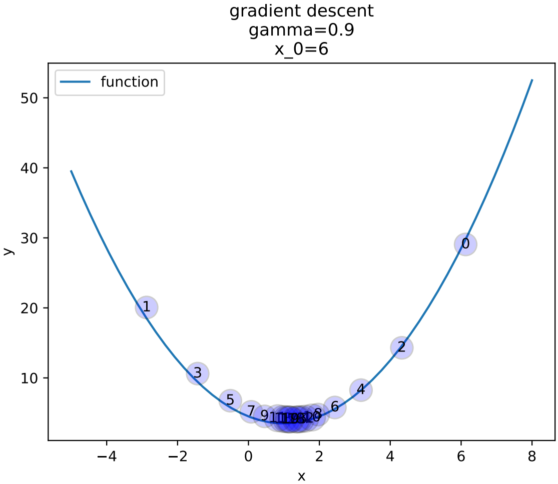


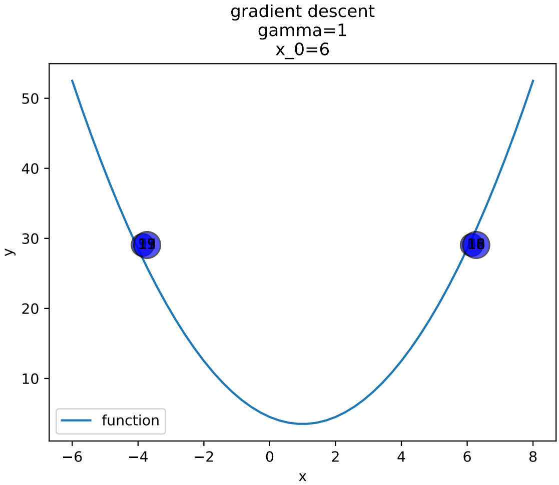

gradient descent
gamma=0.01
x_0=10
function
y
x
0
20
40
60
80
100
120
0
2
4
6
8
10
12
0
1
2
3
4
5
6
7
8
9
10
199

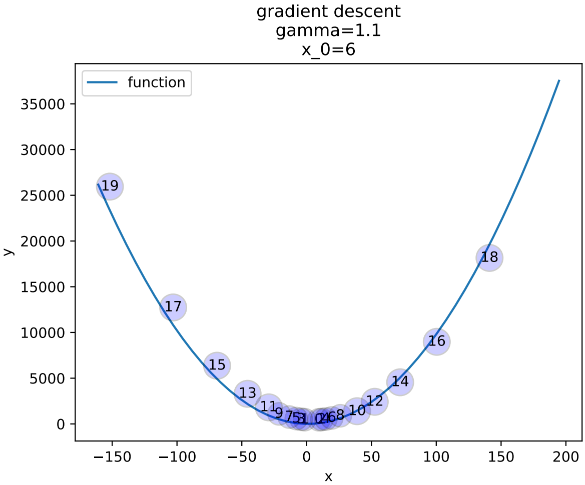

gradient descent
gamma=0.1
x_0=6
function
y
x
50
40
30
20
10
0
2
4
6
8
0
1
2
3
4
5
6
7
8

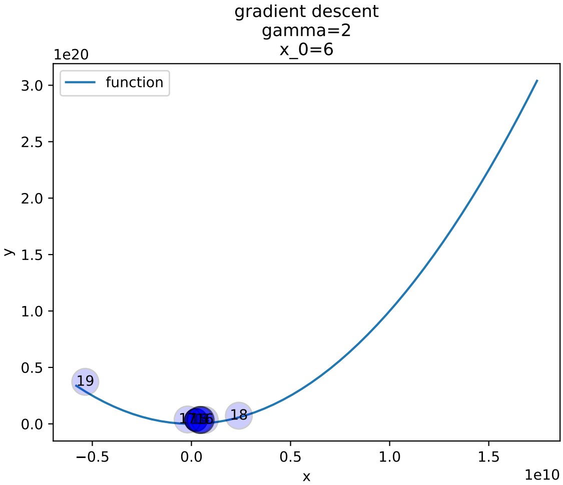

gradient descent
gamma=0.3
x_0=6
function
y
x
50
40
30
20
10
0
2
4
6
8
0
1
2


gradient descent
gamma=0.5
x_0=6
function
y
50
40
30
20
10
0
2
4
6
8
x
0
19


gradient descent
gamma=0.9
x_0=6
function
y
x
50
40
30
20
10
−4
−2
0
2
4
6
8
0
1
2
3
4
5
6
7
8
9

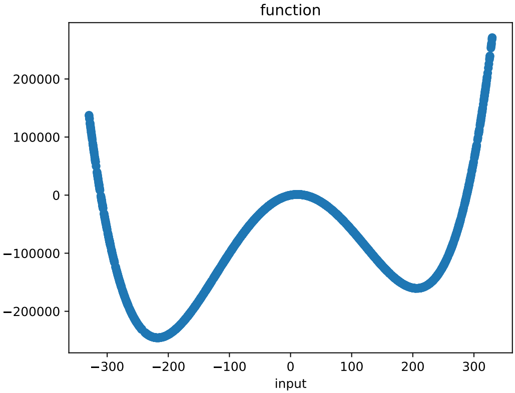

gradient descent
gamma=1
x_0=6
50
40
30
20
10
y
−6
−4
−2
0
2
4
6
8
x
19
18
function


gradient descent
gamma=1.1
x_0=6
function
y
x
35000
30000
25000
20000
15000
10000
5000
0
−150
−100
−50
0
50
100
150
200
19
18
17
16
15
14
13
12
11
10
9
7
5
3
1
0
2
4
6
8


gradient descent
gamma=2
x_0=6
1e20
function
3.0
2.5
2.0
1.5
1.0
0.5
0.0
y
19
18
−0.5
0.0
0.5
1.0
1.5
x
1e10

Local minima

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



Stochastic gradient descent

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

$$f(\theta) = \frac{1}{n} \sum_{i=1}^{n} l(y_i, g_\theta(x_i)) + \Omega(\theta) \tag{11}$$

Example :

- $\Omega(\theta) = \lambda ||\theta||^2$.
- $l(y_i, g_\theta(x_i)) = ||y_i - \langle \theta | x_i \rangle||^2$

Batch gradient

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

$$f(\theta) = \frac{1}{n}\sum_{i=1}^{n} l(y_i, g_\theta(x_i)) + \Omega(\theta) \tag{12}$$

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

SGD

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

$$f(\theta) = \frac{1}{n}\sum_{i=1}^{n} l(y_i, g_\theta(x_i)) + \Omega(\theta) \tag{13}$$

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

$$\nabla_\theta \Big[l(y_i, g_\theta(x_i)) + \Omega(\theta) \Big](\theta) \tag{14}$$

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

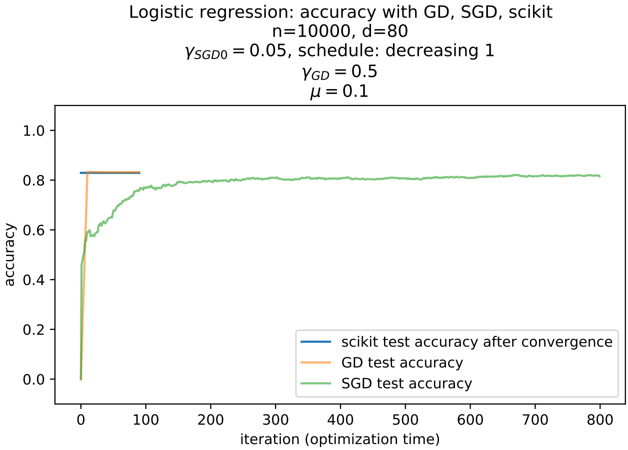


Figure – accuracy

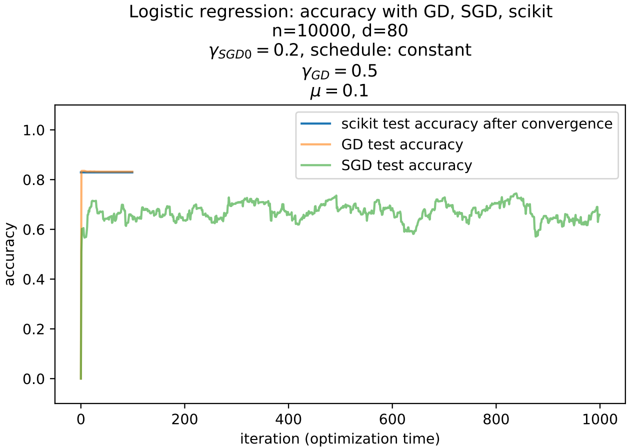


Figure – accuracy

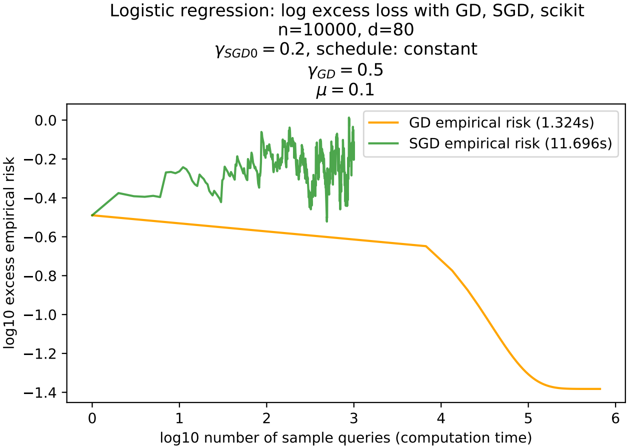


Figure – accuracy

Ridge regression

Gradient algorithms

Classification

- Problem statement
- Convexification of the risk and calibration
- Logistic regression

Dimensionality reduction

- Principal component analysis

General classification problem

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

Problem

Optimizing on $F = \mathcal{Y}^{\mathcal{X}}$ is equivalent to optimizing in the set of subsets of $\mathcal{X}$.
We cannot differentiate on this hypothesis space and it is not clear how to regularize.

Subsets

Exercice 1 : **Combinatorial problem**

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

Real-valued function

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

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

Risk

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

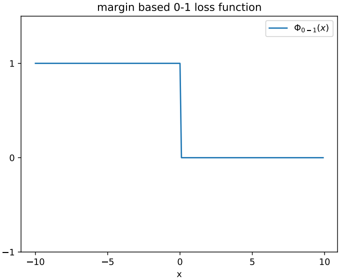
$$\begin{aligned} R(g) &= P(\text{sign}(g(x)) \neq y) \\ &= E\left[1_{\text{sign}(g(x)) \neq y}\right] \\ &= E\left[1_{yg(x)<0}\right] \end{aligned} \tag{15}$$

Remark : several solutions

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

Margin based 0-1 loss function Φ_{0-1}

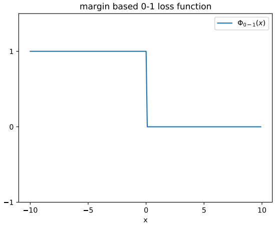
$$\begin{aligned} R(g) &= E\left[1_{\text{sign}(g(x))\neq y}\right] \\ &= E\left[1_{yg(x)<0}\right] \\ &= E\left[\Phi_{0-1}(yg(x))\right] \end{aligned} \tag{16}$$



Empirical risk minimization

The corresponding empirical risk writes :

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

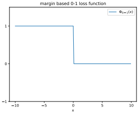


What is the issue with this objective function ?

Empirical risk minimization

The corresponding empirical risk writes :

$$\frac{1}{n}\sum_{i=1}^{n}\Phi_{0-1}(y_i g(x_i)) \tag{18}$$



What is the issue with this objective function ?

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

Convex surrogate

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

Definition

The Φ-risk is defined as

$$R_{\Phi}(g) = E\Big[\Phi(yg(x))\Big] \tag{19}$$

The empirical Φ-risk is defined as

$$R_{\Phi,n}(g) = \frac{1}{n}\sum_{i=1}^{n}\Phi(y_i g(x_i)) \tag{20}$$

Most common convex surrogates

Definition

Logistic loss

$$\Phi(u) = \log(1 + e^{-u}) \tag{21}$$

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

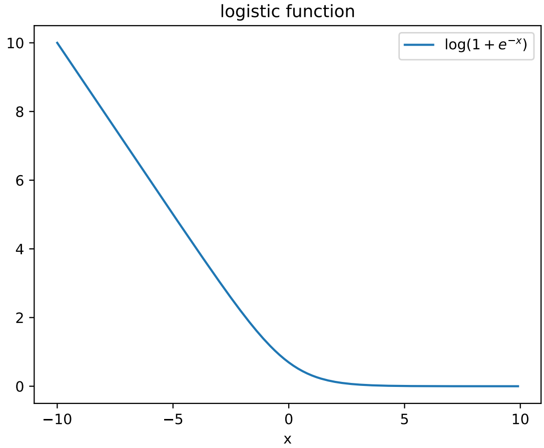
Most common convex surrogates

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

$$l(\hat{y}, y) = y \log(1 + e^{-\hat{y}}) + (1 - y) \log(1 + e^{\hat{y}}) \tag{22}$$

(cross entropy loss)

Logistic function



Most common convex surrogates

Definition

Hinge loss

$$\Phi(u) = max(1 - u, 0) \tag{23}$$

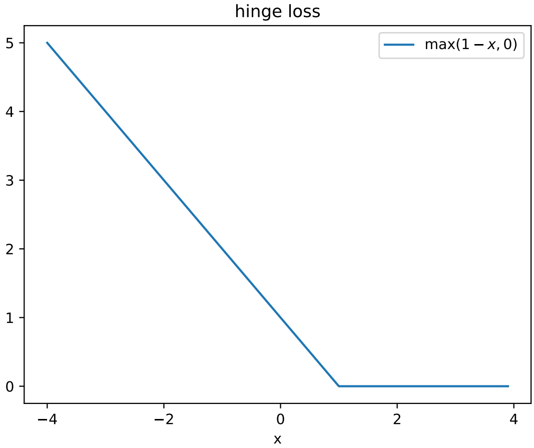
With linear predictors, this loss will lead to **Support vector machines**.

Definition

Squared hinge loss

$$\Phi(u) = (max(1 - u, 0))^2 \tag{24}$$

Hinge loss

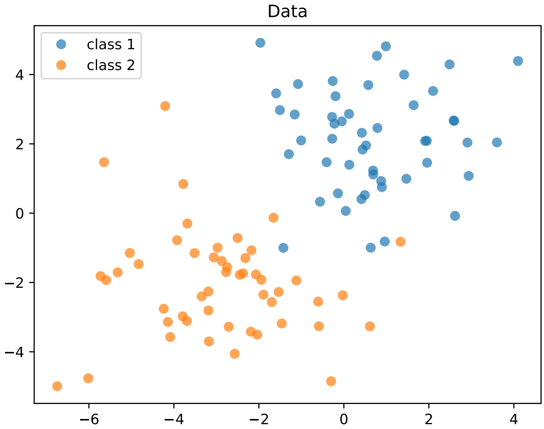


Under some technical hypotheses, minimizing the empirical Φ risk leads to a good generalization errror for the "0-1" loss (notion of calibration function).

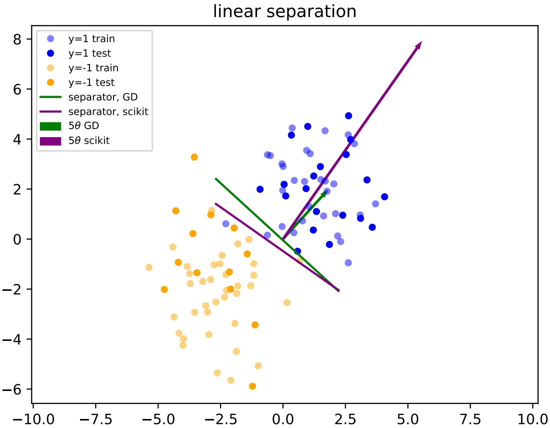
Logistic regression

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

Data to separate



Data to separate



Logistic regression

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

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

$$l(\hat{y}, y) = y \log(1 + e^{-\hat{y}}) + (1 - y) \log(1 + e^{\hat{y}}) \tag{25}$$

Logistic regression estimator

If l is the logistic loss, it is defined as

$$\hat{\theta}_{logit} = \arg\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} l(x_i^T \theta, y_i) \tag{26}$$

We will show that this empirical risk is convex as a function of θ (exercises).

No closed-form solution

Since the loss is convex, to minimize it is sufficient to look for the cancellation of the gradient. However, the corresponding equation has no closed-form solution.
We thus need to use iterative algorithms in order to find a minimizer (e.g. : gradient descent, Newton's method, etc)

Practical usage of logistic regression

In practice, it is common practice to :

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

Probabilistic interpretation of logistic regression

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

Ridge regression

Gradient algorithms

Classification

- Problem statement
- Convexification of the risk and calibration
- Logistic regression

Dimensionality reduction

- Principal component analysis

Dimensionality reduction

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

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

Dimensionality reduction

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

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

Main methods of dimensionality reduction

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

Principal component analysis (PCA)

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

Formalization as a (empirical) variance maximimisation problem

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

$$\bar{x} = \sum_{i=1}^{n} x_n = 0 \in \mathbb{R}^d \tag{27}$$

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

Variance

$$\overline{w^T x} = w^T \overline{x} = 0 \tag{28}$$

Hence,

$$\begin{aligned} \hat{Var}(w^T x) &= \frac{1}{n-1} \sum_{i=1}^{n} \left((w^T x)_i - \overline{w^T x}\right)^2 \\ &= \frac{1}{n-1} \sum_{i=1}^{n} \left(w^T x_i\right)^2 \end{aligned} \tag{29}$$

Variance maximisation problem

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

$$\sum_{i=1}^{n} \left(w^T x_i\right)^2 \tag{30}$$

is maximal.

First principal component

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

$$\sum_{i=1}^{n} \left(w^T x_i\right)^2 \tag{31}$$

is maximal.

Proposition

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

First principal component

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

$$\sum_{i=1}^{n} \left(w^T x_i\right)^2 \tag{32}$$

is maximal.

Proposition

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

Exercice 2 : Show the proposition.

Reconstruction error

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

$$\mathbb{R}^d = \text{Vect}(w) \bigoplus \text{Vect}(w)^{\perp} \tag{33}$$

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

$$\begin{aligned} \forall x \in \mathbb{R}^d, ||x||^2 &= ||(x^T w)w||^2 + ||x - (x^T w)w||^2 \\ &= (x^T w)^2 + ||x - (x^T w)w||^2 \end{aligned} \tag{34}$$

Reconstruction error

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

$$\begin{aligned} \forall x \in \mathbb{R}^d, ||x||^2 &= ||(x^T w)w||^2 + ||x - (x^T w)w||^2 \\ &= (x^T w)^2 + ||x - (x^T w)w||^2 \end{aligned} \tag{35}$$

Hence,

$$\begin{aligned} \sum_{i=1}^n ||x_i||^2 &= \sum_{i=1}^n (x_i^T w)^2 + \sum_{i=1}^n ||x_i - (x_i^T w)w||^2 \\ &= \hat{Var}(w^T x) + \sum_{i=1}^n ||x_i - (x_i^T w)w||^2 \end{aligned} \tag{36}$$

Reconstruction error

$$\sum_{i=1}^{n} ||x_i||^2 = \hat{Var}(w^T x) + \sum_{i=1}^{n} ||x_i - (x_i^T w)w||^2 \qquad (37)$$

We can see $\sum_{i=1}^{n} ||x_i - (x_i^T w)w||^2$ as a **reconstruction error**, when the data are projected on Vect(w).
Maximizing the variance of the projections is equivalent to minimizing the reconstruction errors obtained by projection.

Several principal components

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

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

Reconstruction error

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

$$||x||^2 = ||p_F(x)||^2 + ||x - p_F(x)||^2 \tag{38}$$

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

Inertia

We can define an inertia :

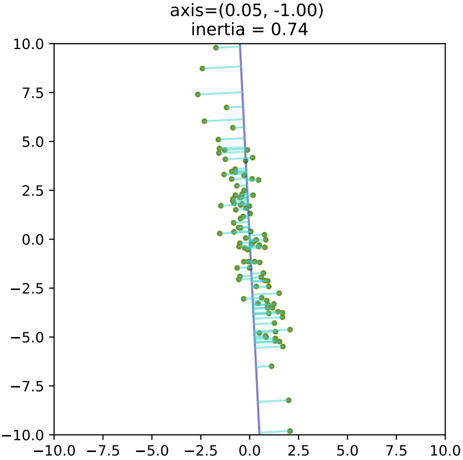
$$I_F = \sum_{i=1}^{n} ||x - p_F(x)||^2 \tag{39}$$

We lool for the subspace that minimizes the inertia I_F.

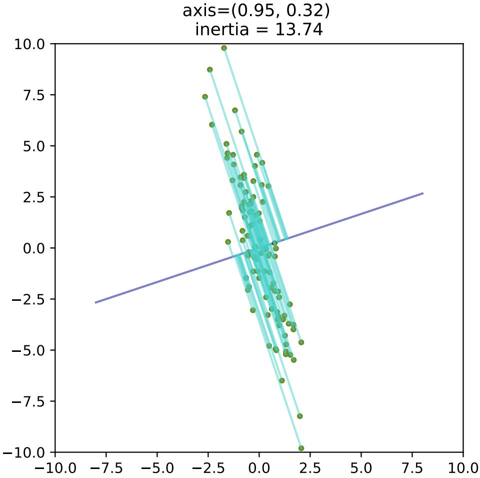
Inertia

Exercice 3 : **No inertia**
In what situations could we have $I_F = 0$?

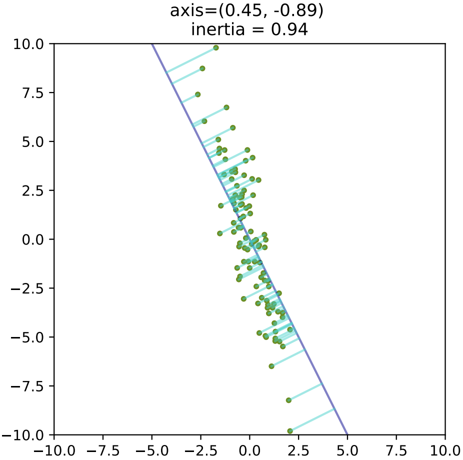
Inertia


axis=(0.05, -1.00)
inertia = 0.74
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

Inertia


axis=(0.95, 0.32)
inertia = 13.74
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

Inertia


axis=(0.45, -0.89)
inertia = 0.94
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

Iris dataset

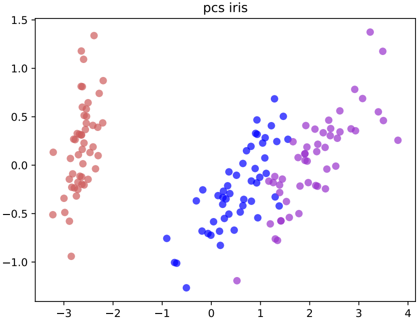


Figure – PCA performed on the iris dataset. We see that the principal components are able to separate the data.

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

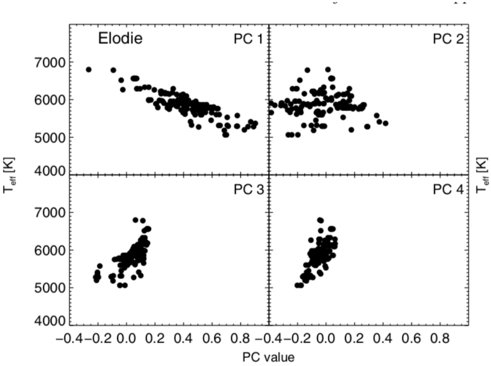
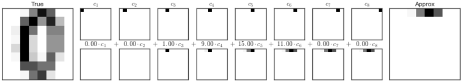


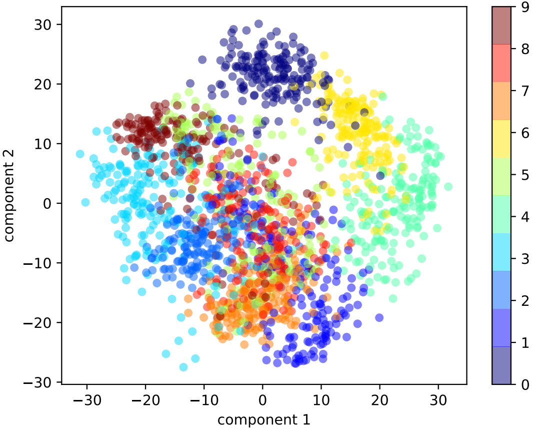
Figure – PCA used in order to predict temperature.

PCA on digits

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


True
c_1
c_2
c_3
c_4
c_5
c_6
c_7
c_8
Approx
$0.00 \cdot c_1 + 0.00 \cdot c_2 + 1.00 \cdot c_3 + 9.00 \cdot c_4 + 15.00 \cdot c_5 + 11.00 \cdot c_6 + 0.00 \cdot c_7 + 0.00 \cdot c_8$

PCA on digits



PCA on digits : reconstruction

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

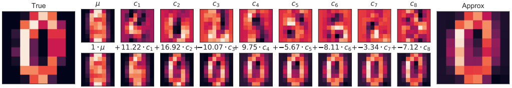


Figure – Reconstruction of 0

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

PCA on digits : reconstruction

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

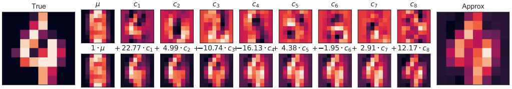


Figure – Reconstruction of 4

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

Explained variance

A natural question is : what is a relevant number of principal components ?
A common quantity that is used is **explained variance**. Each component w_k carries a percentage of the total variance of the data.

$$\frac{\hat{Var}(w_k^T x)}{\sum_{j=1}^{d} \hat{Var}(x^j)} \tag{40}$$

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

$$\hat{Var}(x^j) = \sum_{i=1}^{n} (x_i^j)^2 \tag{41}$$

Number of components

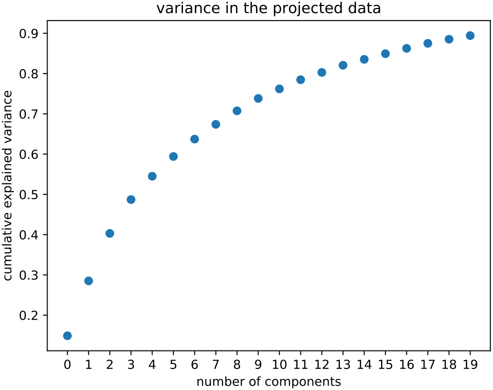
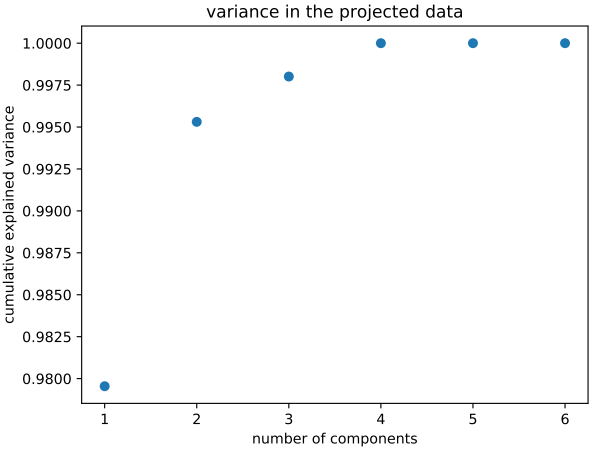


Figure – Variance of the projected data as a function of the number of components (digits dataset)

Number of components

Exercice 4 : What happens with this dataset ?



Number of components

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

Shortcomings of PCA

PCA is sensitive to :

- outliers
- initial data scaling

References I

Bermejo, J. M., Ramos, A. A., and Prieto, C. A. (2013).
Astrophysics A PCA approach to stellar effective temperatures.
95 :1–9.