Université de Rennes

istic Informatique Électronique

ENSAI École nationale de la statistique et de l'analyse de l'information

Chapitre 4. Méthodes liées à la convexification

Claude Petit, Insee et université de Rennes - claude.petit@univ-rennes.fr

2025-2026

Overview

1. Convexification de l'ensemble des classifieurs et de la perte
2. Espaces de Hilbert à noyau auto-reproduisant
3. Machines à vecteurs de support
4. Boosting

1. Convexification de l'ensemble des classifieurs et de la perte

Fonction et ensemble convexe

Ensemble convexe

Soi E un espace vectoriel et $C \subset E$. C est convexe si pour tout $x, y \in C$ et $\lambda \in [0,1]$, $\lambda x + (1-\lambda)y \in C$.

Fonction convexe

Soit $C \subset E$ un ensemble convexe. Une fonction $f : C \longrightarrow \mathbb{R}$ est convexe si $\forall x, y \in C,\ \forall \lambda \in [0,1]$

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$$

Une fonction convexe définie sur un ensemble convexe fermé atteint son minimum.

Problématique : pourquoi convexifier ?

- La minimisation du risque empirique est difficile à mettre en œuvre à cause de la non-convexité de $\mathcal{G}$ et de la non-convexité de $R_n(g)$.
- Minimiser une fonction convexe dans un ensemble convexe est un problème « facile » à résoudre.
- Il faut changer l'espace des classifieurs $\mathcal{G}$ pour qu'il devienne convexe.
- Il faut changer la forme du risque $R(g)$ pour en faire une fonction convexe.

On suppose dans ce chapitre, pour simplifier les formules, que $\mathbb{Y} = \{-1, 1\}$ au lieu de $\mathbb{Y} = \{0, 1\}$.

Convexification de l'espace de recherche $\mathcal{G}$

- Avant : $g : \mathbb{X} = \mathbb{R}^d \longrightarrow \{-1, 1\}$.
- Maintenant : $h : \mathbb{X} \longrightarrow \mathbb{R}$ car $\mathcal{F}(\mathbb{X}, \mathbb{R})$ est convexe.

On affecte à x l'étiquette -1 si $h(x) \leq 0$ et 1 si $h(x) > 0$:

$g(x) = \text{sgn}(h(x))$ où sgn est la fonction signe :

$$\text{sgn}(u) = \begin{cases} 1 & \text{si } u > 0 \\ -1 & \text{si } u \leq 0 \end{cases} \tag{1}$$

Toute fonction réelle h peut alors être considérée comme prédicteur.

Convexification de $\mathcal{G}$ en $\mathcal{H}$

On remplace $\mathcal{F}(\mathbb{X}, \{-1, 1\})$ par $\mathcal{F}(\mathbb{X}, \mathbb{R})$
On remplace $g \in \mathcal{G}$ par $h \in \mathcal{H} \subset \mathcal{F}(\mathbb{X}, \mathbb{R})$

Forme de la perte de prédiction et du risque

La perte de prédiction est donnée par

$$\begin{aligned} l(y, h(x)) &= \mathbb{1}_{[y \neq \text{sgn}(h(x))]} = \mathbb{1}_{[y=1]}\mathbb{1}_{[1 \neq \text{sgn}(h(x))]} + \mathbb{1}_{[y=-1]}\mathbb{1}_{[-1 \neq \text{sgn}(h(x))]} \\ &= \mathbb{1}_{[y=1]}\mathbb{1}_{[h(x) \leq 0]} + \mathbb{1}_{[y=-1]}\mathbb{1}_{[h(x) > 0]} \\ &= \mathbb{1}_{[y=1]}\mathbb{1}_{[yh(x) \leq 0]} + \mathbb{1}_{[y=-1]}\mathbb{1}_{[yh(x) < 0]} \end{aligned}$$

$[yh(x) < 0] \subset [yh(x) \leq 0] \Rightarrow$
$l(y, h(x)) \leq \mathbb{1}_{[y=1]}\mathbb{1}_{[yh(x) \leq 0]} + \mathbb{1}_{[y=-1]}\mathbb{1}_{[yh(x) \leq 0]} \leq \mathbb{1}_{[yh(x) \leq 0]}$

$l(y, h(x)) \geq \mathbb{1}_{[yh(x) < 0]}$ et $\mathcal{H} \subset \mathcal{F}(\mathbb{X}, \mathbb{R}) / \ \mathbb{P}[h(X) = 0] = 0 \ \forall h \in \mathcal{H} \Rightarrow$

Forme de la fonction de perte et du risque

$$l(y, h(x)) = \mathbb{1}_{[yh(x) \leq 0]} = \mathbb{1}_{[0, +\infty[}(-yh(x))$$
$$R(g) = R(\text{sgn}(h)) = \mathbb{E}\left[\mathbb{1}_{[0, +\infty[}(-Yh(X))\right]$$

Convexification de la fonction de perte

On définit le minimiseur du risque empirique convexifié par

$$\widehat{h}_n \in \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{[0,+\infty[}(-Y_i h(X_i)) \tag{2}$$

$\widehat{h}_n$ appartient à $\mathcal{H}$. Mais $\mathbb{1}_{[0,+\infty[}$ n'est pas convexe. On la remplace par ϕ convexe, et on appelle ϕ-risque du classifieur h la quantité :

$$A(h) = R_n(\phi, h) = \mathbb{E}[\phi(-Yh(x))] \tag{3}$$

Le ϕ-classifieur de Bayes est le prédicteur minimisant le ϕ-risque :

$$h^\star \in \arg\min_{h \in \mathcal{F}} A(h) \tag{4}$$

Le minimiseur sur $\mathcal{H} \subset \mathcal{F}$ du ϕ-risque empirique est le prédicteur

$$\widehat{h}_n \in \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \phi(-Y_i h(X_i)) \tag{5}$$

Fonctions convexes usuelles pour le ϕ-risque

Objectif : choisir pour ϕ un majorant convexe de $\mathbb{1}_{[0,+\infty[}$ pour qu'un classifieur de ϕ-risque faible ait un faible risque de classification.

Les fonctions traditionnellement utilisées sont les suivantes :

Perte charnière	$\phi(x) = (1+x)_+$
Perte de Boosting	$\phi(x) = e^x$
Perte logistique	$\phi(x) = \log_2(1+e^x)$
Perte quadratique	$\phi(x) = (1+x)^2$

Fonctions de perte convexes

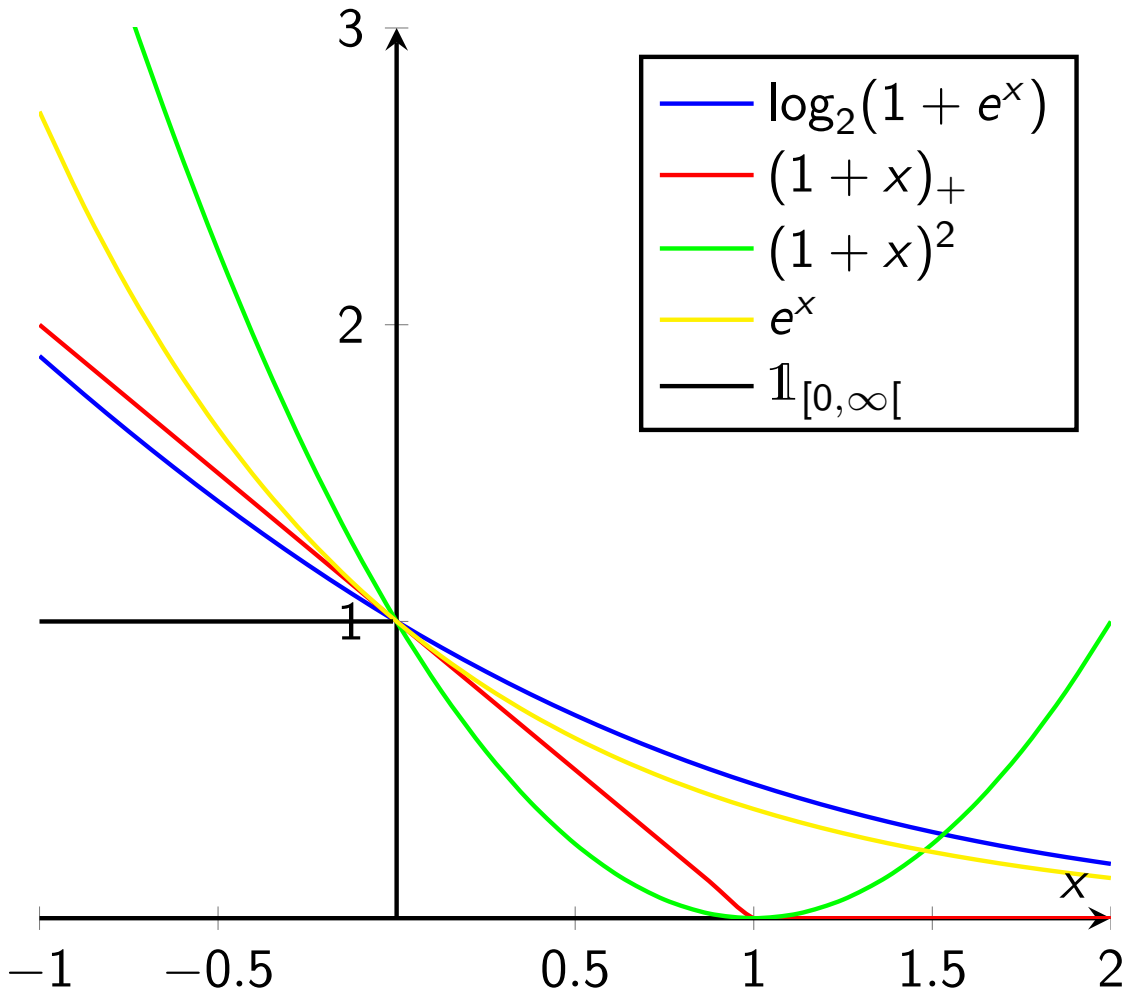


Figure 1 – Fonctions de perte convexe ϕ : courbes de $\phi(yh(x))$.

Consistance du minimiseur du ϕ-risque

Théorème de consistance

Soit $\phi : \mathbb{R} \longrightarrow \mathbb{R}$ une fonction convexe telle que $\forall x \in \mathbb{R}$,

$$x(\phi(x) - \phi(-x)) \geq 0. \tag{6}$$

Soit $\psi : [0,1] \longrightarrow \mathbb{R}$ la fonction définie par

$$\psi(p) = \inf_{u \in \mathbb{R}} (p\phi(-u) + (1-p)\phi(u)) \tag{7}$$

S'il existe $\gamma \in [0,1]$ et $c > 0$ tels que $\forall p \in [0,1]$,

$$|1 - 2p| \leq c \left(\phi(0) - \psi(p)\right)^{\gamma}, \tag{8}$$

alors pour toute fonction de prédiction h,

$$R[\mathrm{sgn}(h)] - R[g^{\star}] \leq c \left(A(h) - A(h^{\star})\right)^{\gamma} \tag{9}$$

Démonstration en exercice !

2. Espaces de Hilbert à noyau auto-reproduisant

Espace de Hilbert : définition

On suppose maintenant que $\mathcal{H}$ a une structure d'espace de Hilbert : espace vectoriel normé muni d'un produit scalaire qui en fait un espace métrique complet :

- l'espace euclidien $\mathbb{R}^n$ muni de $\langle x, y \rangle = x^T y$.
- l'ensemble des suites de carré sommable $l^2(\mathbb{N})$ muni de $\langle x, y \rangle = \sum_{k=1}^{+\infty} x_k y_k$.
- l'ensemble des fonctions de carré intégrable $L^2(\mathbb{R})$. muni de $\langle f, g \rangle = \int_{-\infty}^{+\infty} f(x) g(x) dx$.
- idem sur $\mathbb{C}$ en ajoutant un conjugué sur le second terme.

Les espaces de Hilbert généralisent en dimension infinie la notion d'espace euclidien : on peut travailler de façon "géométrique" dans l'espace : avec des normes, angles, projections orthogonales.

Espace de représentation des données

Données de $\mathbb{X}$ pas facilement manipulables $\Rightarrow$

- on les transforme en les envoyant...
- ... dans un espace (vectoriel) de représentation (de redescription, ou features space),
- via une fonction de représentation ϕ (features map).
- ici, l'espace de représentation sera un espace de Hilbert $\mathcal{H}$.

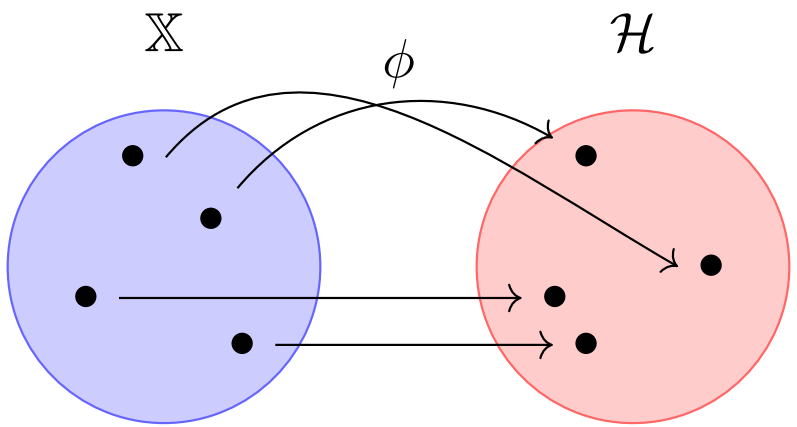


Figure 2 – $\mathcal{H}$ représente les données (features) issues de $\mathbb{X}$

Noyau de similarité

Plutôt qu'une représentation individuelle, il est souhaitable de représenter les données par couples en respectant les similarités entre objets de $\mathbb{X}$:

$$\begin{aligned} K :& \mathbb{X} \times \mathbb{X} \longrightarrow \mathbb{R} \\ & (x, y) \longrightarrow K(x, y) \end{aligned}$$

- noyau symétrique : $K(x, y) = K(y, x)$.
- semi-défini positif : $\forall x_i, x_j \in \mathbb{X}, \forall a_i, a_j \in \mathbb{R}, \sum_i \sum_j a_i a_j K(x_i, x_j) \geq 0$.
- qui définit une matrice carrée réelle : $(K(x_i, x_j))_{i,j}$ pour $i, j = 1, ..., n$.

On peut mélanger les deux points de vue en utilisant $\phi : \mathbb{X} \to \mathcal{H}$ et le noyau $K : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$:

$$K(x, y) = \langle \phi(x), \phi(y) \rangle$$

Pourquoi augmenter la dimension ? 1

• Si $\mathbb{X}$ possède beaucoup de paramètres (de features), $\mathcal{H}$ peut être de grande dimension.

• C'est même l'objectif de ϕ : travailler dans un espace où il y a beaucoup de « place », de degrés de liberté.

• Si ϕ choisie de façon adéquate, possible de ne pas avoir à effectuer explicitement les produits scalaires en grande dimension grâce à l'astuce du noyau (« kernel trick »).

• K mesure la similarité entre les données x et $y \sim$ généralisation d'une fonction de covariance.

Pourquoi augmenter la dimension ? 2

• Travailler dans un espace où il y a beaucoup de « place », de degrés de liberté.

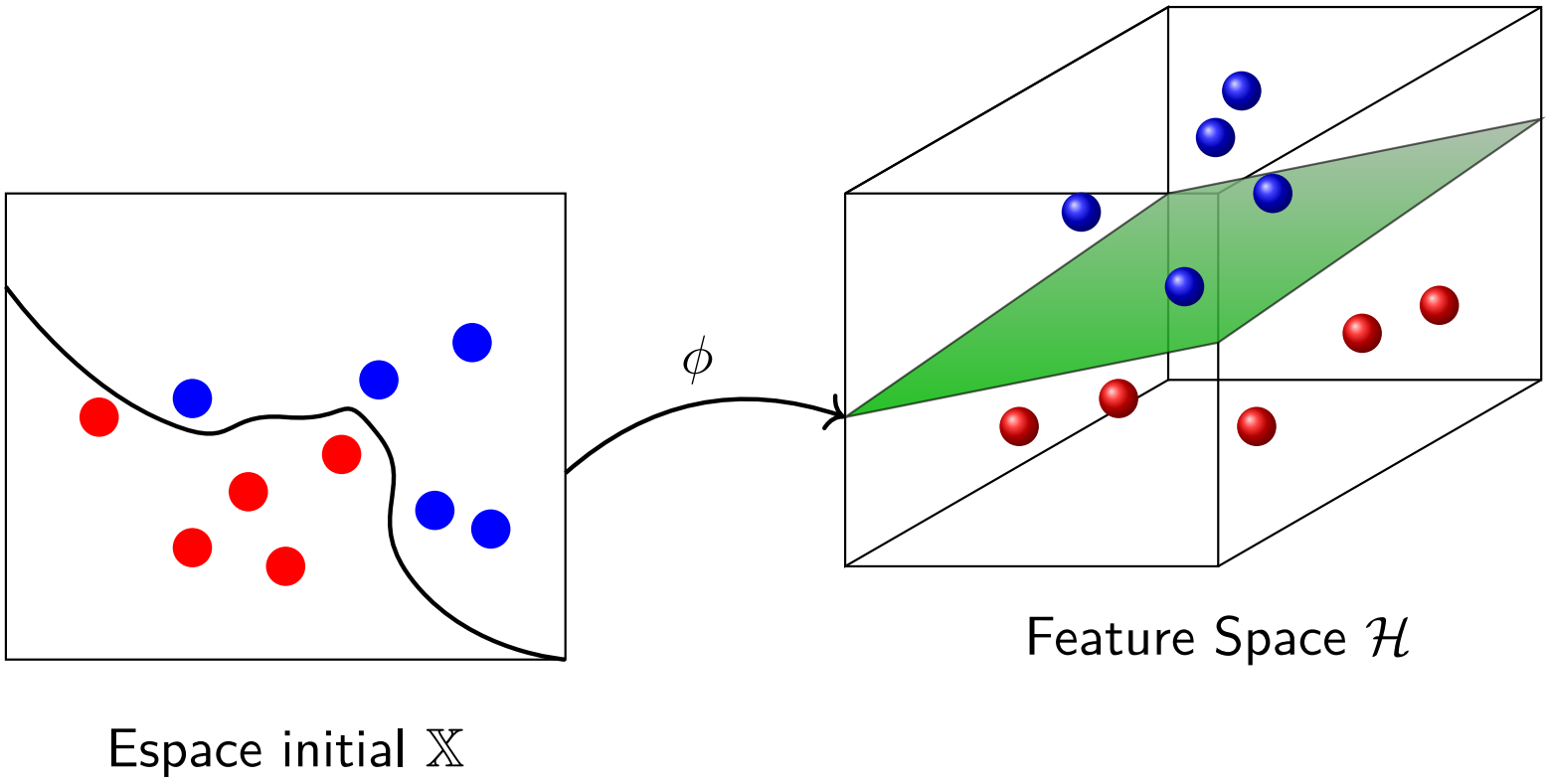


Figure 3 – On envoie les données initiales dans un espace (abstrait) de dimension plus grande, afin de pouvoir séparer plus facilement les classes.

Pourquoi augmenter la dimension ? 3

• K mesure la similarité entre les données x et $y \sim$ généralisation d'une fonction de covariance.

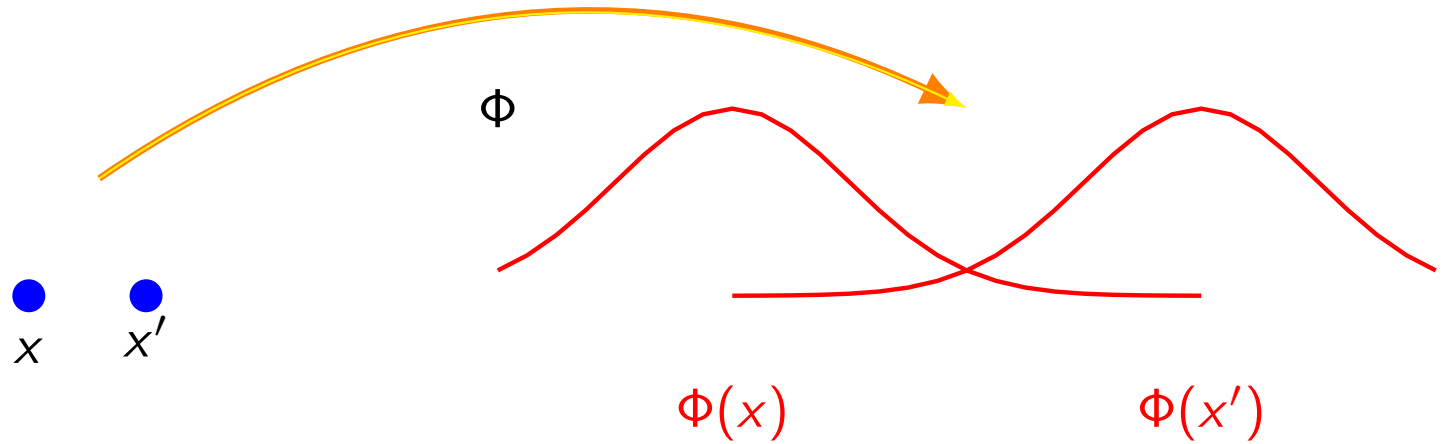


Figure 4 – La fonction de représentation ϕ transforme en fait un point de $\mathbb{X}$ en une fonction $K_x = \phi(x)$ de $\mathbb{X}$ dans $\mathbb{R}$ (considérée comme un élément de l'espace de Hilbert $\mathcal{H}$).

Les noyaux usuels

- Vanille : $K(x, y) = \langle x, y \rangle$.
- Sigmoïde : $K(x, y) = \tanh(\langle x, y \rangle)$.
- Gaussien (RBF) : $K(x, y) = \exp\left(-\frac{1}{2\sigma^2}||x - y||^2\right)$.
- ReLU : $K(x, y) = \min(x, y) = x - \text{ReLU}(x - y) = y - \text{ReLU}(y - x)$.
- Polynomial : $K(x, y) = (c + \langle x, y \rangle)^p$.
- Exponentiel : $K(x, y) = \exp\left(-\gamma||x - y||\right)$.

L'espace de représentation associé au noyau gaussien est un espace de Hilbert de dimension infinie.

On remarque que ϕ n'apparaît pas dans les formules.

Un théorème abstrait mais très important

Théorème de Mercer-Kolmogorov-Aronszajn (MKA)

Si K est un noyau continu, défini positif sur un ensemble $\mathbb{X}$, alors il existe un unique espace de Hilbert $\mathcal{H}$ et une application de représentation $\phi : \mathbb{X} \to \mathcal{H}$ tel que tels que $\forall x, y \in \mathbb{X}$,

$$K(x, y) = \langle \phi(x), \phi(y) \rangle$$

Mercer (1909) pour $\mathbb{X}$ compact de $\mathbb{R}^d$, Kolmogorov (1941) pour $\mathbb{X}$ dénombrable et Aronszajn (1944) pour le cas général.

- $\mathcal{H}$ est unique, mais pas ϕ.
- On espère que si $\mathcal{H}$ est de grande dimension (éventuellement infinie) les données vont devenir plus facile à classifier (séparer).
- L'astuce du noyau (kernel trick) : on n'a pas besoin de spécifier ϕ et d'évaluer $\phi(x)$ et $\phi(y)$, seul compte $K(x, y)$.

Dual d'un espace de Hilbert et théorème de Riesz

- On souhaite choisir pour $\mathcal{H}$ un espace de fonctions $h : \mathbb{X} \longrightarrow \mathbb{R}$.
- Une forme linéaire continue (FLC) sur un espace de Hilbert est une fonction linéaire continue de $\mathcal{H}$ dans $\mathbb{R}$.
- Leur ensemble est un e.v. $\mathcal{H}'$ appelé dual de $\mathcal{H}$.
- Le théorème de représentation de Riesz dit que $\mathcal{H} \simeq \mathcal{H}'$.

Théorème de représentation de Riesz

$\forall L \in \mathcal{H}'$, $\exists! f \in \mathcal{H}$ tel que $L(h) = \langle f, h \rangle$, $\forall h \in \mathcal{H}$.
Réciproquement, $\forall f \in \mathcal{H}$, $L(h) = \langle f, h \rangle$ définit un élément de $\mathcal{H}'$.

L application linéaire continue $\iff$ $|L(h)| \leq \delta ||h||$, $\forall h \in \mathcal{H}$
$\iff$ $||L|| < \delta$ $\iff$ L bornée.

RKHS

$\forall x \in \mathbb{X}$, soit $L_x : \mathcal{H} \longrightarrow \mathbb{R}$ telle que $L_x(h) = h(x)$.
L_x reproduit l'action de h sur x, c'est la fonction d'évaluation en x.

- $\forall x \in \mathbb{X}$, $L_x \in \mathcal{H}'$.
- $\mathcal{H}$ est un RKHS si, et seulement si $\forall x \in \mathbb{X}$, L_x est continue.
- Le théorème de Riesz $\Rightarrow$

$$\forall x \in \mathbb{X}, \exists! K_x \in \mathcal{H} \text{ tel que } L_x(h) = \langle K_x, h \rangle = h(x), \ \forall h \in \mathcal{H}$$

On peut poser $K(x, y) = \langle K_x, K_y \rangle = \langle \phi(x), \phi(y) \rangle$, c.-à-d. $K_x = \phi(x)$.
K est un noyau reproduisant, unique, symétrique, semi-défini positif.

Correspondance entre noyau et RKHS (réciproque de Aronszajn)

$\mathcal{H}$ est un RKHS si, et seulement s'il existe un noyau reproduisant (unique) K vérifiant $K(x, y) = \langle K_x, K_y \rangle$, $\forall x, y \in \mathbb{X}$.

Tout élément de $\mathcal{H}$ s'écrit de façon unique $h = \sum_{i=1}^{n} \alpha_i K_{x_i}$:
$h(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i)$.

Le théorème du représentant

Théorème du représentant

- Soit $K : \mathbb{X} \times \mathbb{X} \longrightarrow \mathbb{R}$ un noyau symétrique, défini positif.
- Soit $\mathcal{H}$ son RKHS.
- Soit $\mathcal{D}_n = \{(x_i, y_i)\} \in (\mathbb{X} \times \mathbb{R})^n$ un échantillon.
- Soit l une fonction de perte.
- Soient $\lambda > 0$ et J fonction réelle strictement croissante.

Alors toute solution $\widehat{h}_n$ du problème de minimisation

$$\arg\min_{h \in \mathcal{H}} \left(\frac{1}{n} \sum_{i=1}^{n} l(y_i, h(x_i)) + \lambda J\left(||h||\right) \right)$$

s'écrit de façon unique sous la forme

$$\widehat{h}_n(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i)$$

Beaucoup d'espaces abstraits...

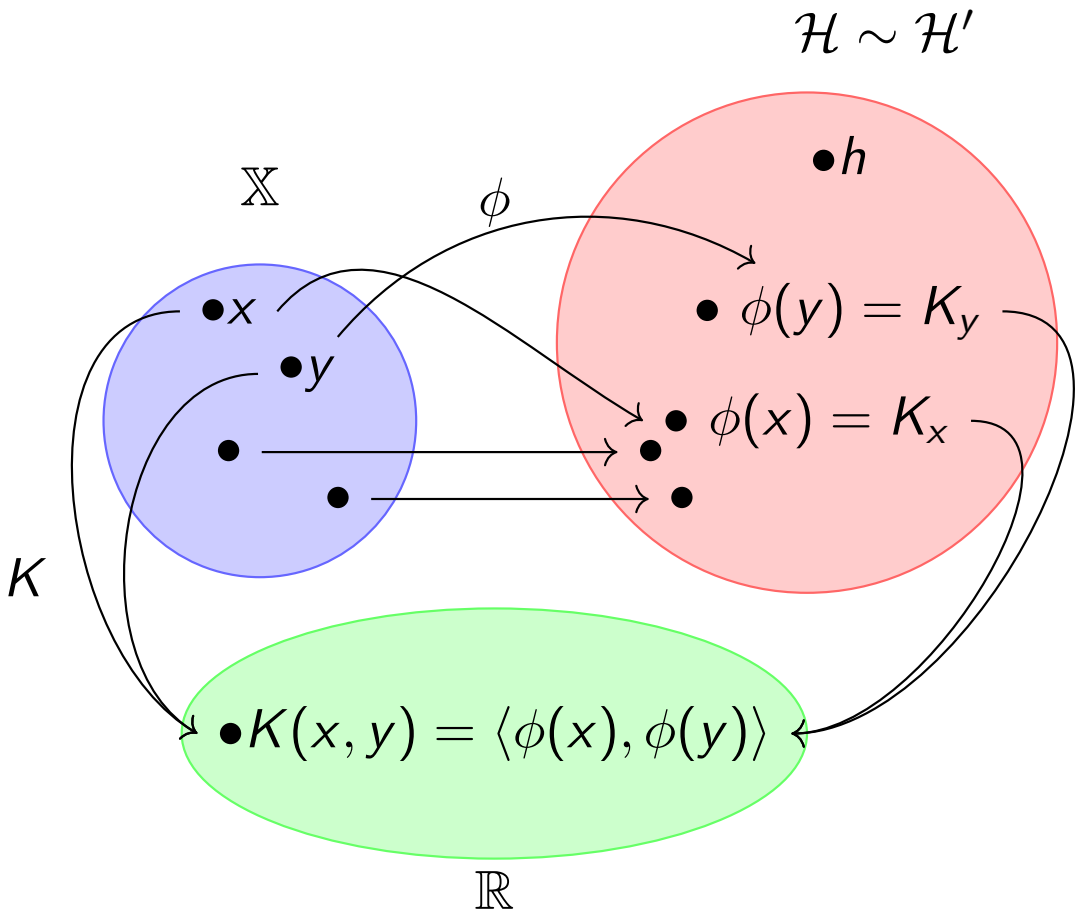


Figure 5 – Les espaces $\mathbb{X}$, $\mathcal{H}$, $\mathcal{H}'$ et $\mathbb{R}$, ainsi que les fonctions ϕ et K.

3. Machines à vecteurs de support

Machine à vecteurs de support : premier point de vue

Pour un $t > 0$, on pose

$$\widehat{h}_{\text{SVM}}(t) \in \underset{h \in \mathcal{H} : ||h|| \leq t}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \phi(-Y_i h(X_i)) \tag{10}$$

$$\iff \widehat{h}_{\text{SVM}}(\lambda) \in \underset{h \in \mathcal{H}}{\arg\min} \left(\frac{1}{n} \sum_{i=1}^{n} \phi(-Y_i h(X_i)) + \lambda ||h||^2 \right). \tag{11}$$

- $\widehat{h}$ est une fonction et $\mathcal{H}$ est un espace de fonctions de $\mathbb{X}$ dans $\mathbb{R}$.
- En classification, on choisit $\phi(x) = (1+x)_+$. En régression, $\phi(x) = x^2$; l'expression $\phi(-Yh(X))$ est remplacée par $\phi(Y - h(X))$.
- Si $\mathcal{H}$ espace de Hilbert à noyau auto-reproduisant (RKHS) alors on dit que $\widehat{h}_n = \widehat{h}_{\text{SVM}}$ est un prédicteur à base de noyau, ou machine à vecteurs de support (SVM).
- L'hyperparamètre t (ou λ) est calculé par validation croisée.

Machine à vecteurs de support : classifieur SVM

- Comme $\mathcal{H}$ est un RKHS, la solution de

$$\underset{h\in\mathcal{H}:||h||\leq t}{\arg\min}\ \frac{1}{n}\sum_{i=1}^{n}\phi(-Y_i h(X_i)). \tag{12}$$

est de la forme (théorème du représentant)

$$\widehat{h}_{\mathrm{SVM}}(.) = \widehat{h}_n(.) = \sum_{i=1}^{n}\alpha_i K(.,x_i) \tag{13}$$

avec K noyau RKHS de l'espace $\mathcal{H}$, $\alpha_1, ...\alpha_n \in \mathbb{R}$ et $x_1, .., x_n$ données observées de l'échantillon.

- Les α_i sont facilement et rapidement déterminés par une méthode d'optimisation convexe.

- Les SVM ont été inventées par Vapnik dans les années 1990.

Machine à vecteurs de support : point de vue traditionnel

On dispose d'un échantillon $\mathcal{D}_n$ de n observations (x_k, y_k), avec $x_k \in \mathbb{R}^d$ et $y = \pm 1$, que l'on peut séparer linéairement par un hyperplan :

$$\begin{cases} w^T x_k + b > 0 & \text{si } y_k = 1 \\ w^T x_k + b < 0 & \text{si } y_k = -1 \end{cases}$$

L'hyperplan séparateur est caractérisé par son vecteur normal $w \in \mathbb{R}^d$ et un seuil b. Son équation est

$$w^T x + b = 0$$

La distance euclidienne d'un point $x \in \mathbb{R}^d$ à l'hyperplan H est donnée par :

$$d(x, H) = \frac{\left| w^T x + b \right|}{||w||}$$

Séparateur de plus grande marge

La plus petite distance entre les observations et un hyperplan séparateur (w, b) est

$$\frac{1}{||w||} \min_{i=1}^{n} y_i(w^T x_i + b)$$

Le double de ce minimum s'appelle la marge. L'hyperplan de plus grande marge a pour paramètres (w, b) les solutions de

$$\arg\max_{w,b} \left(\frac{1}{||w||} \min_{i=1}^{n} y_i(w^T x_i + b) \right)$$

Les observations qui réalisent le minimum sont appelées les vecteurs de support. Les séparateurs à vaste marge (SVM) sont des machines à vecteurs de support au sens défini précédemment.

Illustrations des SVM - 1 -

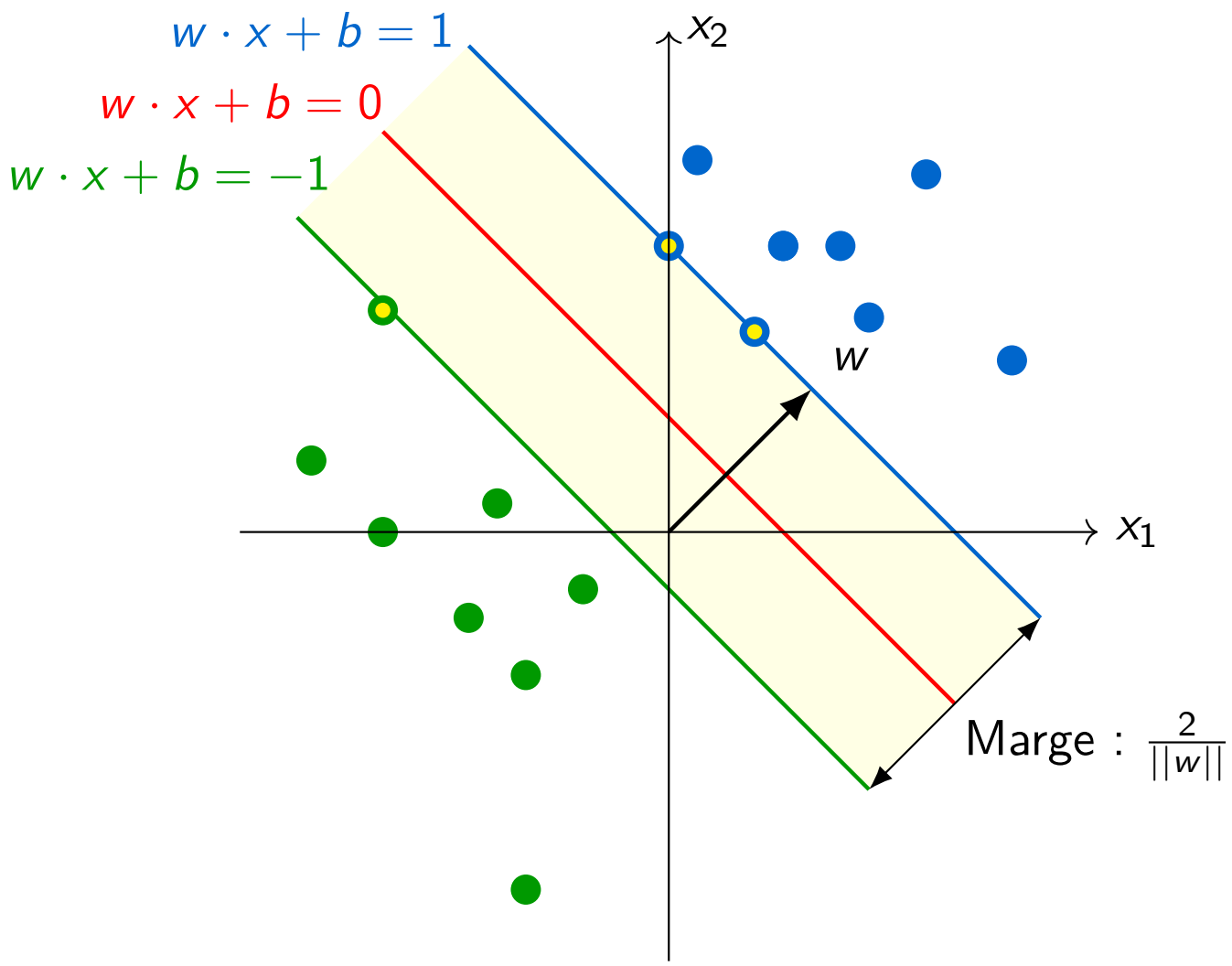


Figure 6 – 2 classes d'observations (bleu et vert) linéairement séparables. En rouge, l'hyperplan séparateur de plus grande marge, en vert et bleu les hyperplans frontières de chaque classe et vecteurs supports (centre jaune).

Illustrations des SVM - 2 -

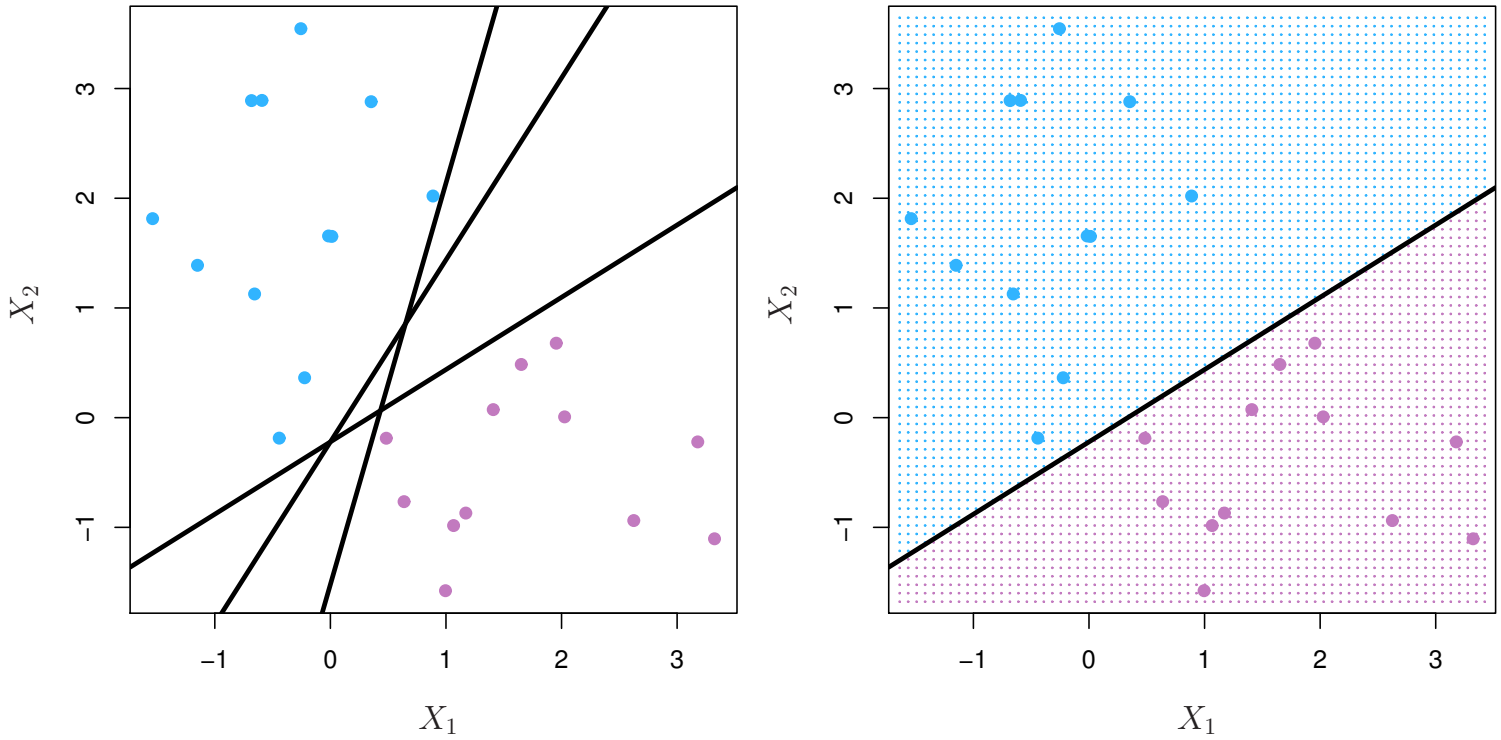


Figure 8 – Deux classes d'observations (bleu et rouge) mesurées par l'intermédiaire de deux variables x_1 et x_2. À gauche, trois exemples de plans séparateurs. À droite, le plan séparateur optimal, le plus éloigné des deux nuages. Crédit : Intro. to ML with Python. James, Witten, Hastie, Tibshirani.

Illustrations des SVM - 3 -

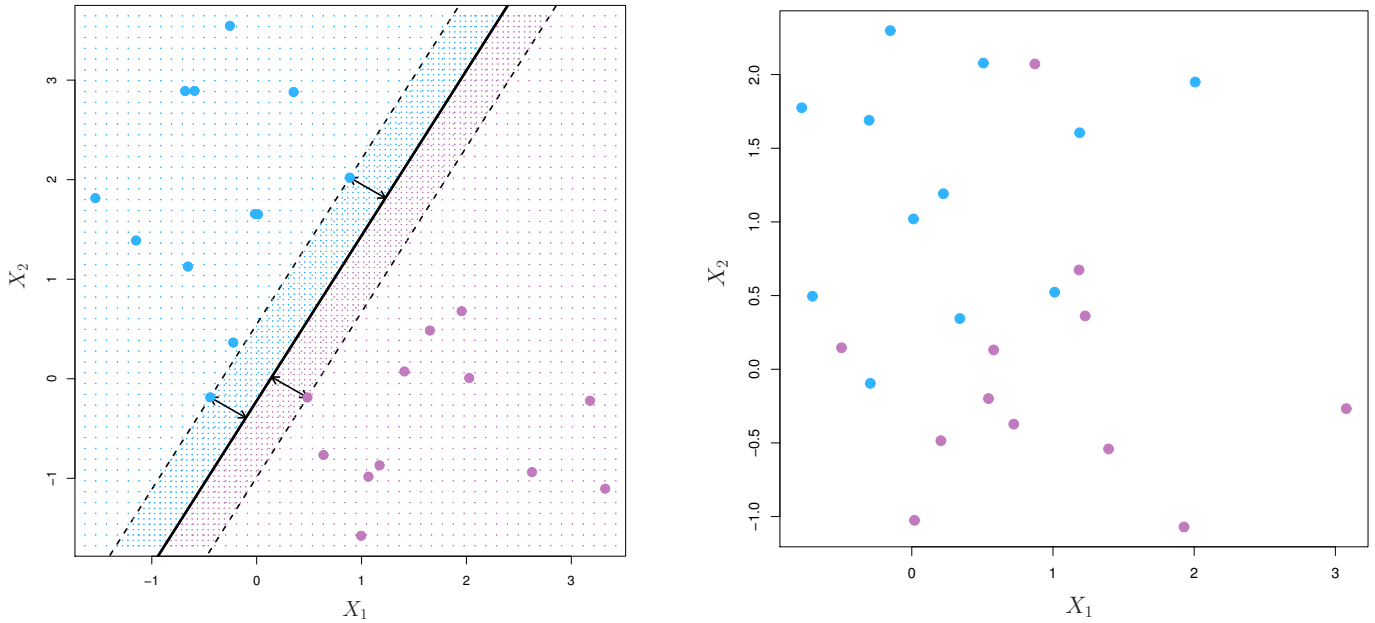


Figure 9 – Deux classes d'observations (bleu et rouge) mesurées par l'intermédiaire de deux variables x_1 et x_2. À gauche, le plan séparateur de plus grande marge avec les points les vecteurs de support à la frontière. À droite, un exemple de nuages non linéairement séparable à cause d'un point rouge ajouté par rapport au nuage précédent (trouver où...). Crédit : Introduction to Machine Learning with Python. James, Witten, Hastie, Tibshirani. Figures du chapitre 9.

Résolution du problème d'optimisation -1-

La marge vaut $\gamma = 1/||w||$. Maximiser $\gamma \iff$ minimiser $||w||$.

Condition de séparation des classes : $\forall i = 1, ..., n$:

$$\text{sgn}(h(x_i)) = y_i \iff y_i h(x_i) \geq 1 \iff y_i(w^T x_i + b) \geq 1$$

Problème d'optimisation (primal)

$$\begin{cases} \min_{w,b} ||w||^2/2 \\ \text{s.c. } y_i(w^T x_i + b) \geq 1 \ \forall i = 1, ..., n. \end{cases} \quad (14)$$

Problème d'optimisation convexe $\Rightarrow$ lagrangien + conditions de Karush-Kuhn-Tucker (KKT). $\Rightarrow$ problème dual.

Démo et résolution en exercice.

Résolution du problème d'optimisation -2-

Problème d'optimisation (dual)

$$\begin{cases} \underset{\alpha \in \mathbb{R}_+^n}{\arg\max} \left(\sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle / 2 \right) \\ \text{s.c. } \alpha_i \geq 0 \ \forall i = 1, ..., n, \ ; \ \sum_{i=1}^{n} \alpha_i y_i = 0. \end{cases} \tag{15}$$

On obtient d'abord les $\alpha_i^\star$, on en déduit $w^\star$ et $b^\star$ qui donnent l'hyperplan optimal :

$$w^\star = \sum_{i=1}^{n} \alpha_i^\star y_i x_i$$

$$\widehat{h}_n(x) = h^\star(x) = \sum_{i=1}^{n} \alpha_i^\star y_i \langle x_i, x \rangle + b^\star$$

Les données de l'échantillon correspondant à $\alpha_i^\star > 0$ sont les vecteurs support (seuls à contribuer à la solution).

Cas non linéairement séparables, lien entre les 2 points de vue

Si données non linéairement séparables (grande majorité des cas), on généralise par une perte convexe ϕ et on reécrit le pb. d'optimisation :

$$\min_{w,b} \left(||w||^2 + \frac{c}{n} \sum_{i=1}^{n} \phi(-y_i h(x_i)) \right) \iff \min_{w,b} \left(\frac{1}{n} \sum_{i=1}^{n} \phi(-y_i h(x_i)) + \lambda ||w||^2 \right)$$

avec $\lambda = 1/2c$ paramètre de régularisation.

Dans le pb. dual, $\langle x, y \rangle$ se transforme en $K(x, y) = \langle \phi(x), \phi(y) \rangle$ et... on retrouve le premier point de vue.

Illustration des SVM avec noyaux non linéaires -1-

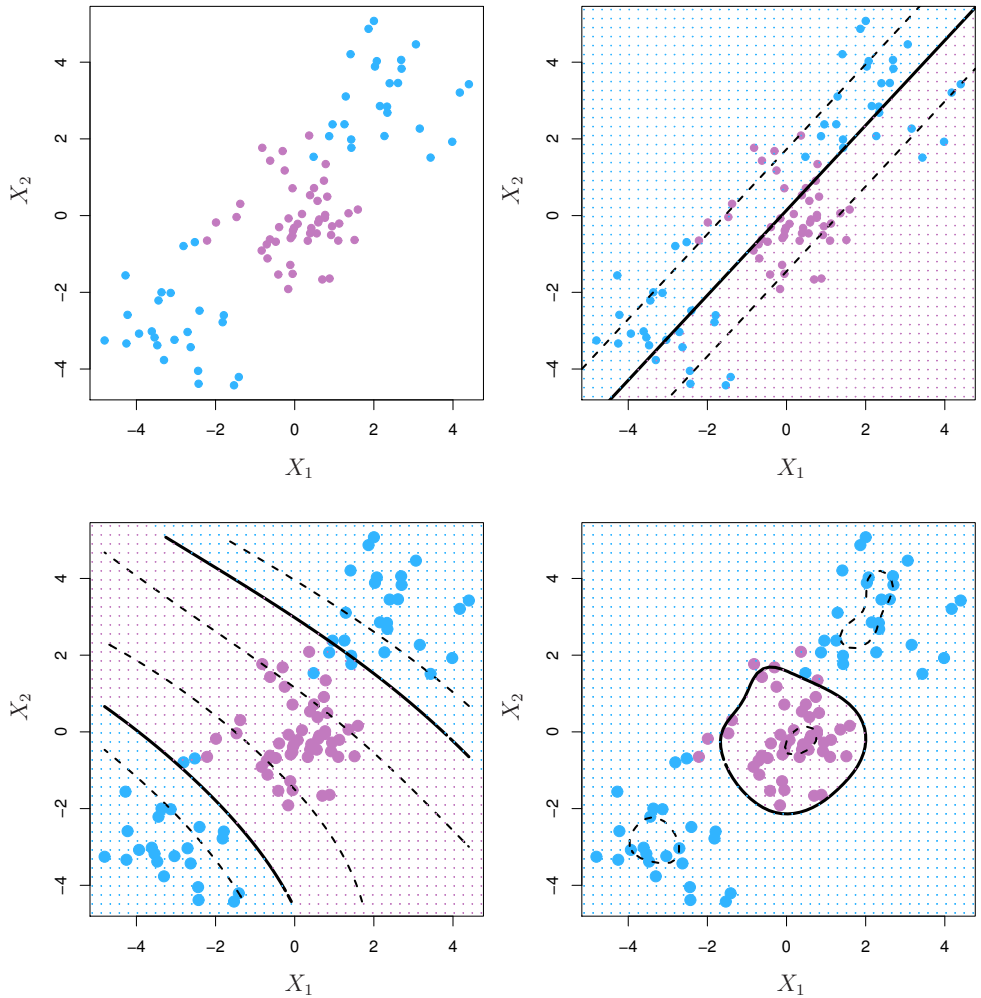
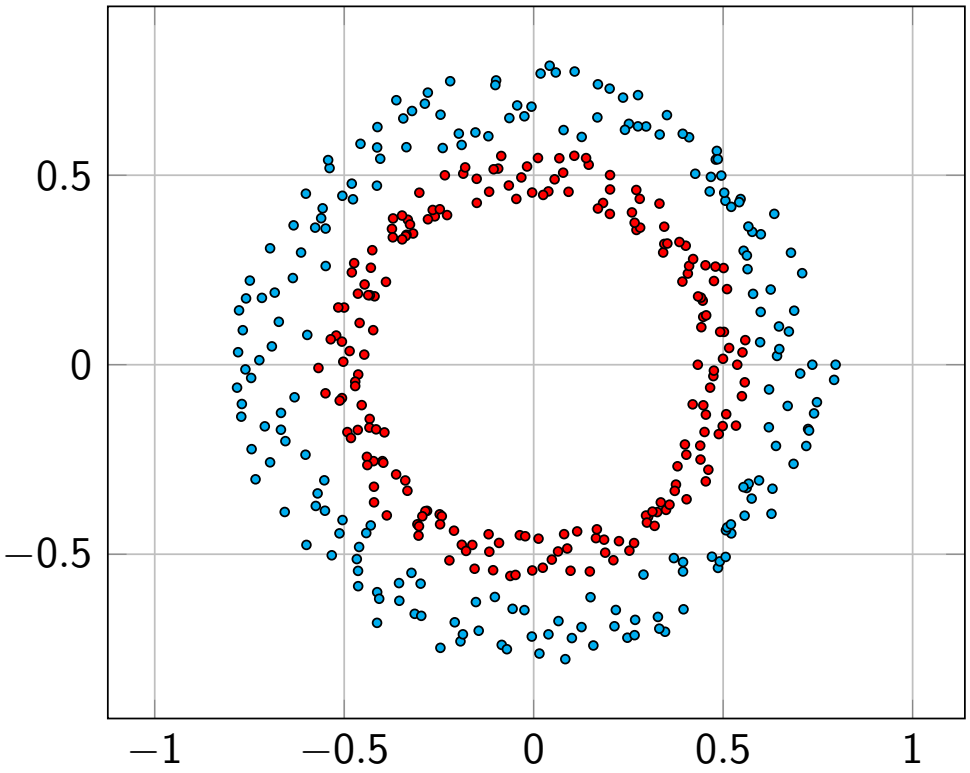
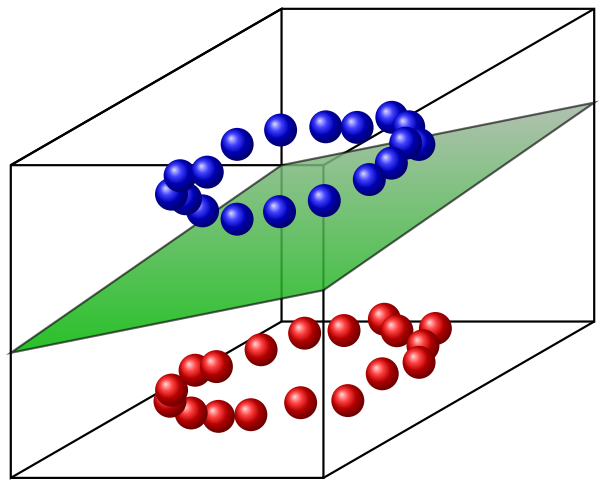


Figure 10 – 2 classes (bleu et rouge) mesurées par x_1 et x_2. En haut, données plus linéairement séparables. En bas, après changement de x en $\phi(x)$ (fonction non linéaire). À gauche, ϕ polynome degré 3, à droite, ϕ gaussienne radiale. Crédit : Intro. to ML with Pyton. James, Witten, Hastie.

Illustration des SVM avec noyaux non linéaires -2-





Feature Space $\mathcal{H}$

Figure 11 – Données non linéairement séparables en dimension 2, mais dont on se doute qu'elles sont à symétrie circulaire. En dimension 3, elles correspondent à deux nuages de hauteurs différentes qui sont linéairement séparables.
$\phi(x, y) = (x, y, x^2 + y^2)$ et $K(x, y) = \langle \phi(x), \phi(y) \rangle = x^T y + ||x||^2 ||y||^2$.

Marge souple : principe

Un cas particulier « presque linéaire » est donné par le pb. à marge souple : on autorise quelques écarts à l'hyperplan séparateur en introduisant une variable d'ajustement (« slack » variable pour varaible d'écart, souple, molle, etc.) :

$$\xi_i = (1 - y_i(wx_i + b))_+, \ i = 1, ..., n.$$

$$\begin{cases} \arg\min\limits_{w,b,\xi} \left(\frac{1}{2}||w||^2 + c\sum\limits_{i=1}^{n} \xi_i \right) \\ \\ \text{s.c. } \xi_i \geq (1 - y_i(wx_i + b))_+, \ i = 1, ..., n. \end{cases}$$

c grand : pénalise fortement les ξ_i (peu de souplesse).
c petit : on peut s'écarter sensiblement de l'hyperplan.

Dans le pb. dual, les ξ_i disparaissent. Seule $\neq$: condition $\alpha_i \leq c$.

Marge souple : illustration

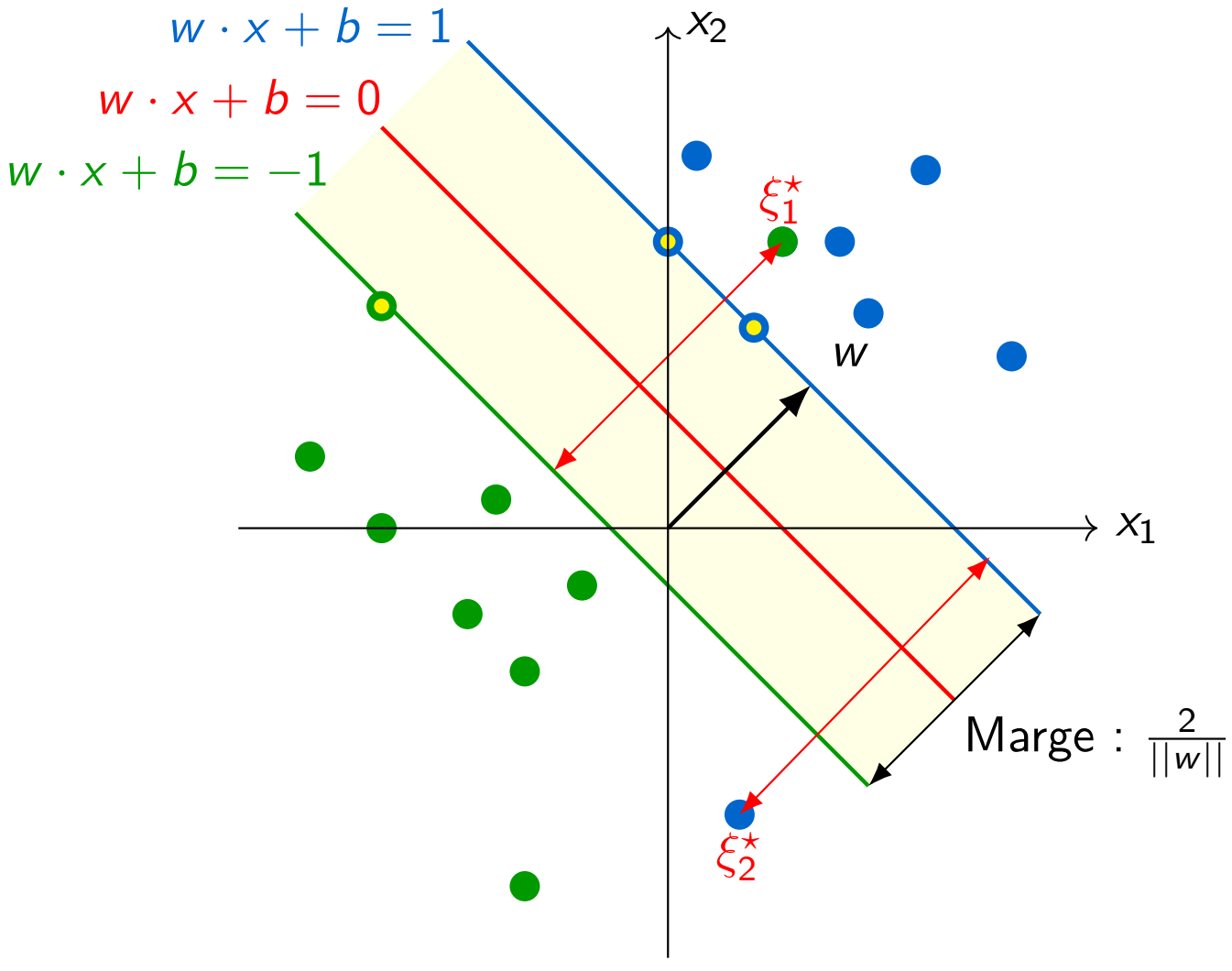


Figure 12 – Marge souple avec 2 classes (en bleu et vert) non linéairement séparables. 2 points (l'un vert, l'autre bleu) de part et d'autre de la marge. La distance $\xi_1^\star$ et $\xi_2^\star$ à la frontière les caractérisent.

Marge souple : conclusion

• Dans le probème dual, les ξ_i disparaissent et la seule $\neq$ avec le problème précédent est la condition $\alpha_i \leq c$. w ne dépend toujours que des vecteurs supports, pour lesquels la contrainte est saturée :

$$y_i(w^T x_i + b) = 1 - \xi_i.$$

• Les vecteurs supports sont les x_i tels que $\xi_i = 0$ ou $\xi_i > 0$, mais dans la marge (bien classé si $0 < \xi_i < 1$ et mal classé si $\xi_i > 1$). $\Longleftrightarrow$ On tolère quelques données mal classées mais permet d'être appliqué à des données non linéairement séparables.

SVR : régression à vecteurs de support -1-

• En régression, on cherche h approchant au mieux le nuage de points $(x_i, y_i)_i$. $h(x, w)$ dépend d'un paramètre à optimiser w.

• $\forall i = 1, ..., n$, $|h(x_i) - y_i|$ le $+$ petit possible et h la plus régulière possible (surapprentissage).

• $\Longleftrightarrow$ Minimiser la perte insensible (« insensitive ») :

$$|x|_\epsilon = \begin{cases} 0 & \text{si } |x| < \epsilon, \\ |x| - \epsilon & \text{sinon.} \end{cases} \tag{16}$$

$$|y - h(x)|_\epsilon = \max\left(0, |y - h(x)| - \epsilon\right). \tag{17}$$

• Pour la perte ϵ-insensible, écart $\leq \epsilon$ entre $h(x_i)$ et y_i non en compte.

• Définit un tube de largeur ϵ de part et d'autre de h à l'intérieur duquel doivent se trouver tous les points de l'échantillon.

SVR : régression à vecteurs de support -2-

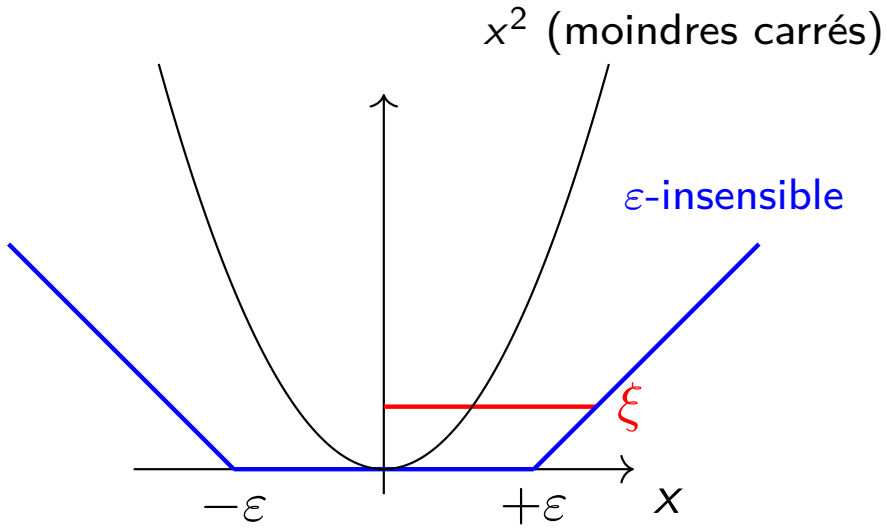


Figure 13 – La fonction de perte ϵ-insensible comparée à la perte au sens des moindres carrées.

SVR : régression à vecteurs de support -3-

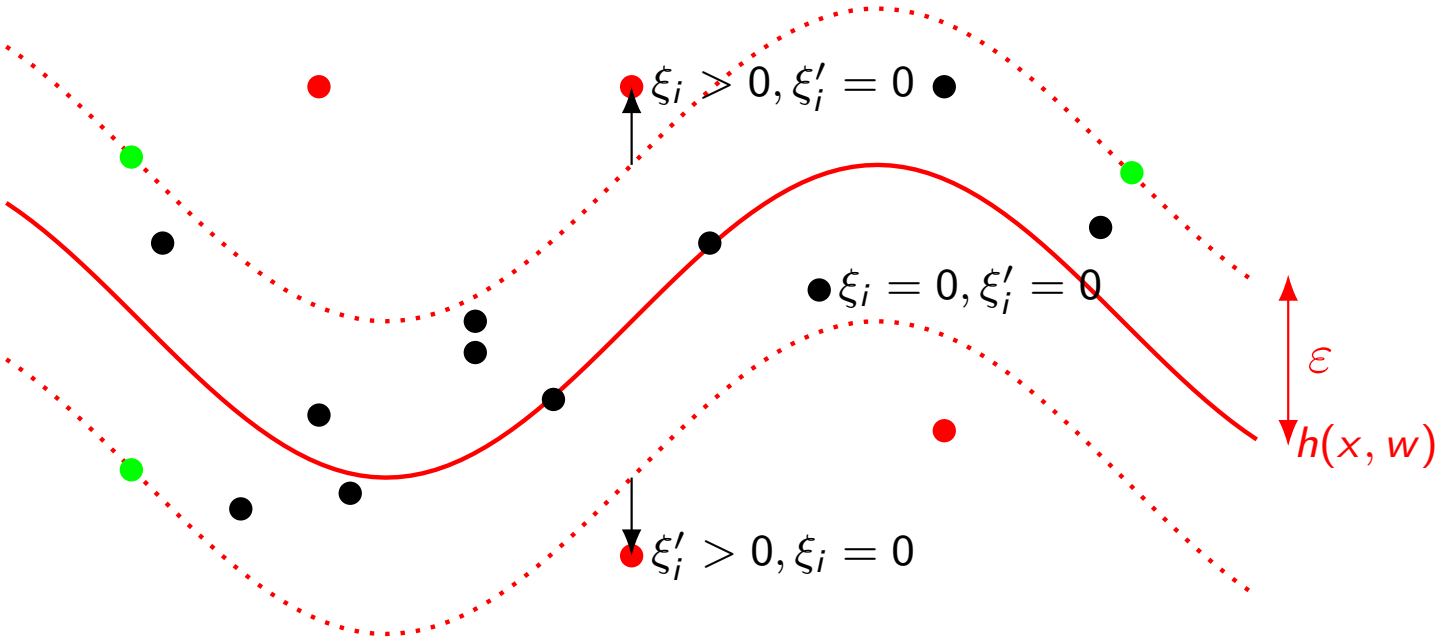


Figure 14 – Dans la SVR, la fonction perte définie un ϵ-tube de part et d'autre de la fonction de régression h dans laquelle doivent se trouver tous les points de l'échantillon. Il est possible de définir des variables d'écarts ξ comme pour une marge souple. En noir, les observations se trouvant dans le ϵ-tube, en rouge, les données à l'extérieur (pour lesquelles l'une des variables d'écart est non nulle) et en vert les données qui correspondent aux vecteurs de support.

SVR : régression à vecteurs de support -4-

$y_i \in \mathbb{R} \Rightarrow$ on doit résoudre :

$$\begin{cases} \arg\min\limits_{w,b} \left(||w||^2/2 \right) \\ \text{s.c. } |y_i - (w^t x_i + b)| \leq \epsilon, \ \forall i = 1, ..., n. \end{cases}$$

Les (x_i, yi) doivent être dans un tube de rayon ϵ autour de $(x, f(x))$ (appelé ϵ-tube).

Pas de solution si ϵ trop petit. On introduit des écarts possibles :

$$\xi_i = \begin{cases} |y_i - (w^t x_i + b)| - \epsilon & \text{si } |y_i - (w^t x_i + b)| > \epsilon, \\ 0 & \text{sinon.} \end{cases}$$

Problème d'optimisation primal SVR

$$\begin{cases} \arg\min\limits_{w,b,\xi,\xi'} \left(||w||^2/2 + c \sum_{i=1}^{n} (\xi_i + \xi'_i) \right) \\ \text{s.c. } y_i - w^t x_i + b \leq \epsilon + \xi_i \ , \ w^t x_i + b - y_i \leq \epsilon + \xi'_i \ , \ \forall i = 1, .., n. \end{cases}$$

SVR : régression à vecteurs de support -5-

Problème d'optimisation dual SVR

$$\begin{cases} \arg\min\limits_{\alpha,\alpha'} -\frac{1}{2}\sum\limits_{i,j}(\alpha_i - \alpha'_i)(\alpha_j - \alpha'_j)x_i x_j - \epsilon\sum\limits_{i}(\alpha_i + \alpha'_i) + \sum\limits_{i} y_i(\alpha_i - \alpha'_i) \\ \text{s.c. } 0 \leq \alpha_i, \alpha'_i \leq C,\ \forall i = 1, .., n\ ; \sum_i(\alpha_i - \alpha'_i) = 0. \end{cases}$$

Après résolution, le régresseur s'écrit sous la forme :

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha'_i)\langle x_i, x\rangle + b$$

et si des noyaux non linéaires sont utilisés, il s'exprime sous la forme :

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha'_i)K(x_i, x) + b.$$

SVR : exemple 1 -6-

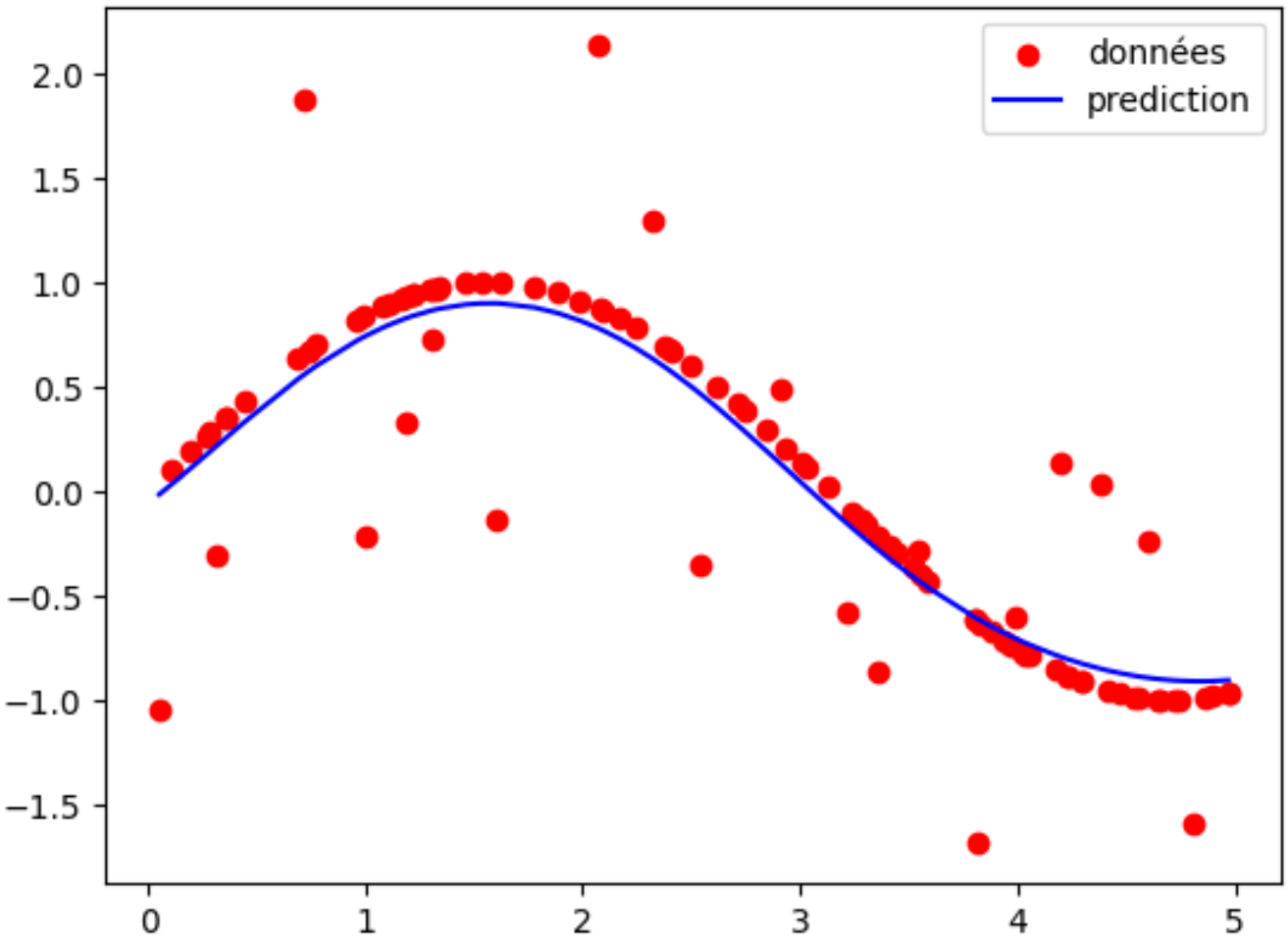


Figure 15 – Un nuage de points (en rouge) générés autour d'une sinusoïde perturbée par un bruit aléatoire et la courbe de régression SVR (en bleu).

SVR : exemple 2 -7-

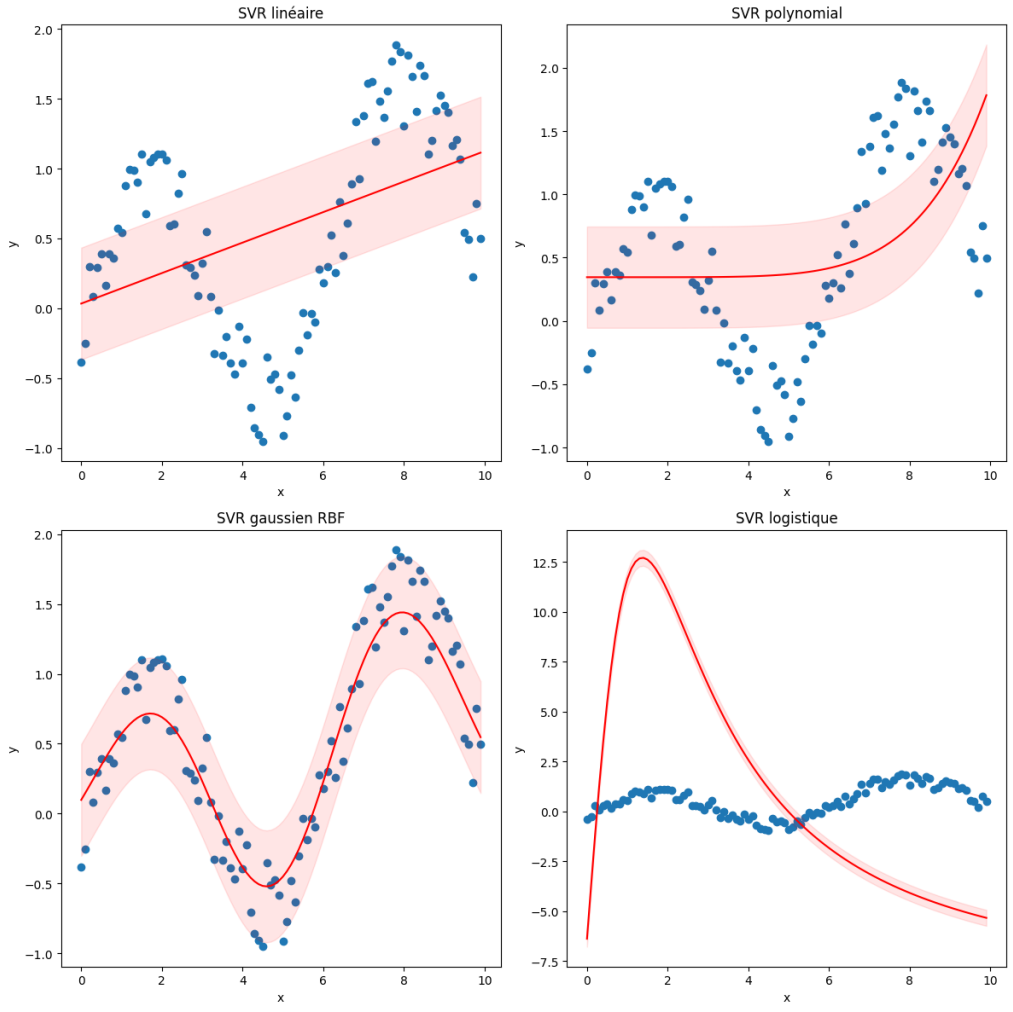


Figure 16 – Nuage de points (en bleu) autour d'une sinusoïde et courbes de régressions SVR (en rouge, avec les ϵ-tubes en rose, pour $\epsilon = 0.5$) pour 4 types de noyaux.

Classification multiclasses : exemple des iris de Fisher -1-

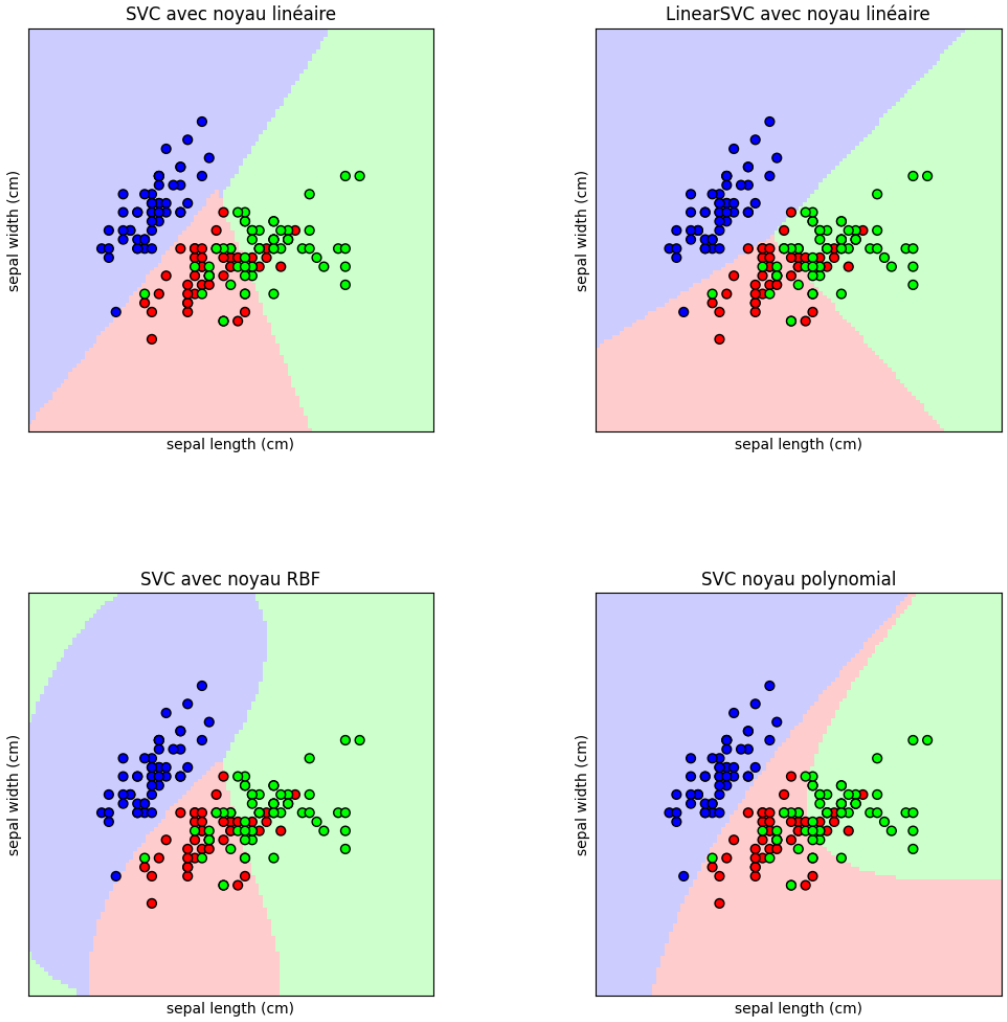


Figure 17 – SVM : classification des 3 espèces d'iris de Fisher. 4 noyaux différents proposés. Régions dédiées à chaque espèce : dans le plan caractérisé par la longueur et la largeur du sépale.

Classification multiclasses : exemple des iris de Fisher -2-

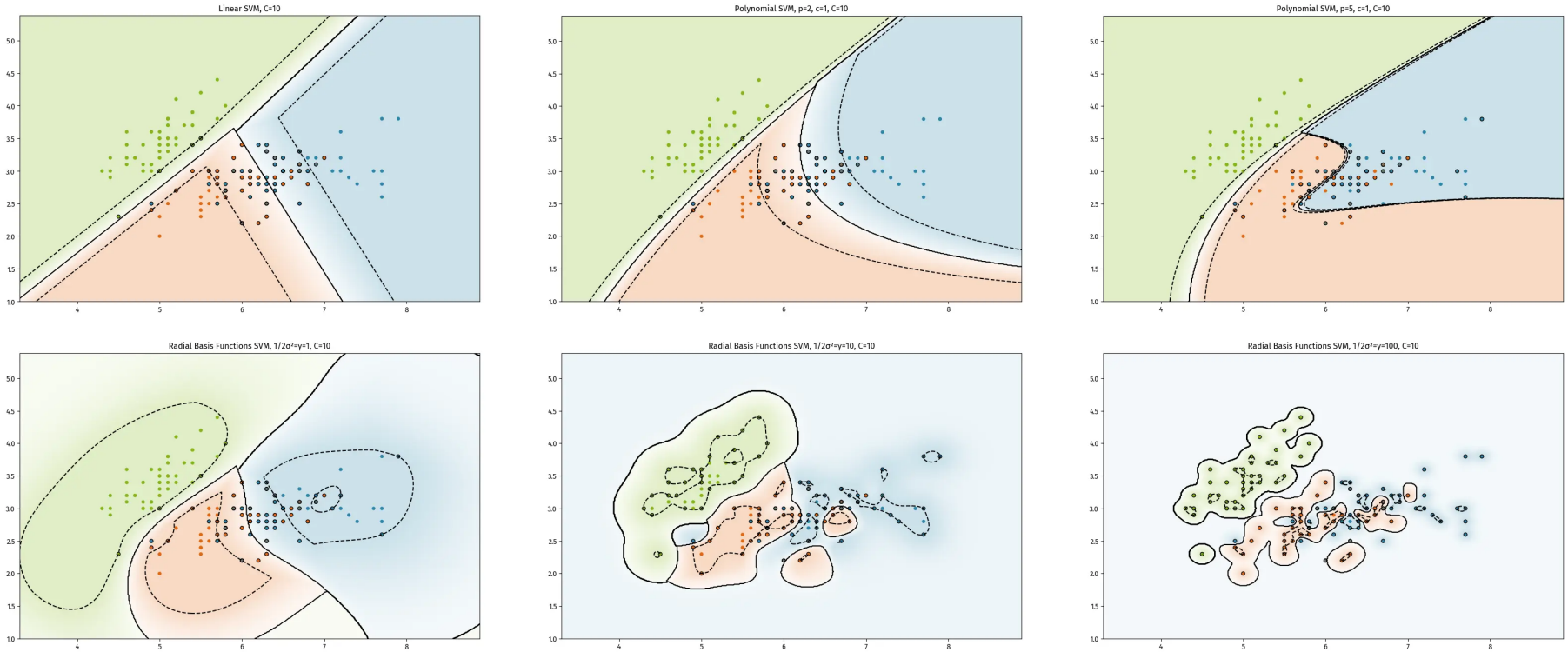


Figure 18 – SVM : classification des 3 espèces d'iris de Fisher. 4 noyaux différents proposés, dont 3 sont des noyaux gaussiens RBF (ligne du bas).

Remarques finales

- SVM permettent également de faire de la régression.
- SVM considérées comme 1 des meilleures méthodes de classification.
- En grande dimension ($d >> n$) les SVM sont efficaces.
- Mais n'estiment pas de probabilité ($\neq$ reg. logistique).
- Bonne alternative aux réseaux de neurones : plus faciles à entraîner.
- Mais pas toujours interprétable, pas toujours très rapides.
- Souvent moins performant que les forêts aléatoires.

Bibliographie

- Learning With Kernels : Support Vector Machines, Regularization, Optimization and Beyond, 2002.
- Hofmann, Schölkopf, Smola, Kernel methods in machine learning, Annals of Statistics, 2008.
- Burges, A tutorial on support vector machines for pattern recognition, Data Mining, 1998.
- Smola, Schölkopf, A tutorial on support vector regression, Statistics and Computing, 2004.
- + Chapitre SVM (p.367) Intro to Statistical Learning with Python.
- + Chapitre 8 du polycopié de Frédéric Sur.

4. Boosting

Encore une histoire de convexification

Autre façon de convexifier l'ensemble des prédicteurs $\mathcal{H} \subset \mathcal{F}(\mathbb{X}, \{-1, 1\})$:

$$\mathcal{H}_\lambda = \{\sum_{m=1}^{M} \lambda_m h_m;\ \lambda_m \geq 0,\ h_m \in \mathcal{H},\ \sum_m \lambda_m \leq \lambda\}. \tag{18}$$

$\mathcal{H}_\lambda$ est convexe. On pose

$$\widehat{h}_{n,\lambda} \in \arg\min_{h \in \mathcal{H}_\lambda} A_n(h) \tag{19}$$

- Si $\lambda = \lambda_n \to +\infty$, $\lambda_n \phi'(\lambda_n) \sqrt{\ln n / n} \to +\infty$ et $\mathcal{H}$ a une dimension de Vapnik finie, alors $\widehat{h}_{n,\lambda_n}$ est universellement consistent.
- Problème difficile à résoudre car de dimension infinie.
- L'algorithme Adaboost permet de le résoudre.

Algorithme AdaBoost -1-

Adaptive Boosting (Freund et Shapire 1996). Principe : la sagesse des foules (wisdom of the many).

- Classifieur performant à partir de classifieurs faibles (réduit le biais).
- Donne plus de poids aux observations difficiles à prédire.
- Simple, rapide et facile à implémenter : très peu de paramètres.
- Flexible (s'adapte à tout type de classifieur faible sans it a priori).
- Permet également la régression.
- Versatile : beaucoup d'applications (reconnaissance d'images, de textes, moteurs de recherche).

Mais....

- Comportement vis à vis du sur-apprentissage ambigu.
- Entrainement séquentiel coûteux en temps de calcul.
- Sensible aux valeurs aberrantes et au bruit.

Algorithme AdaBoost -2-

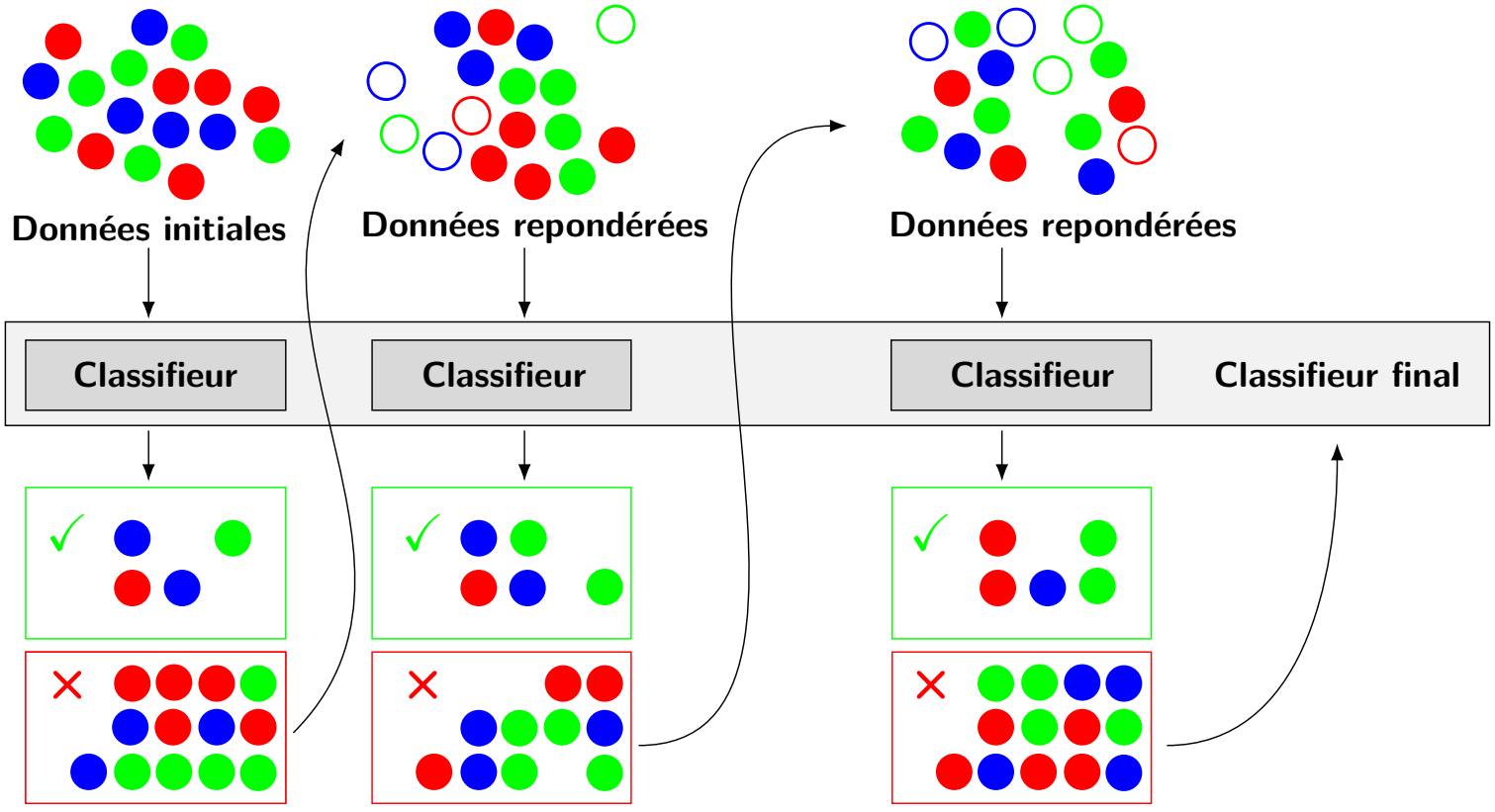
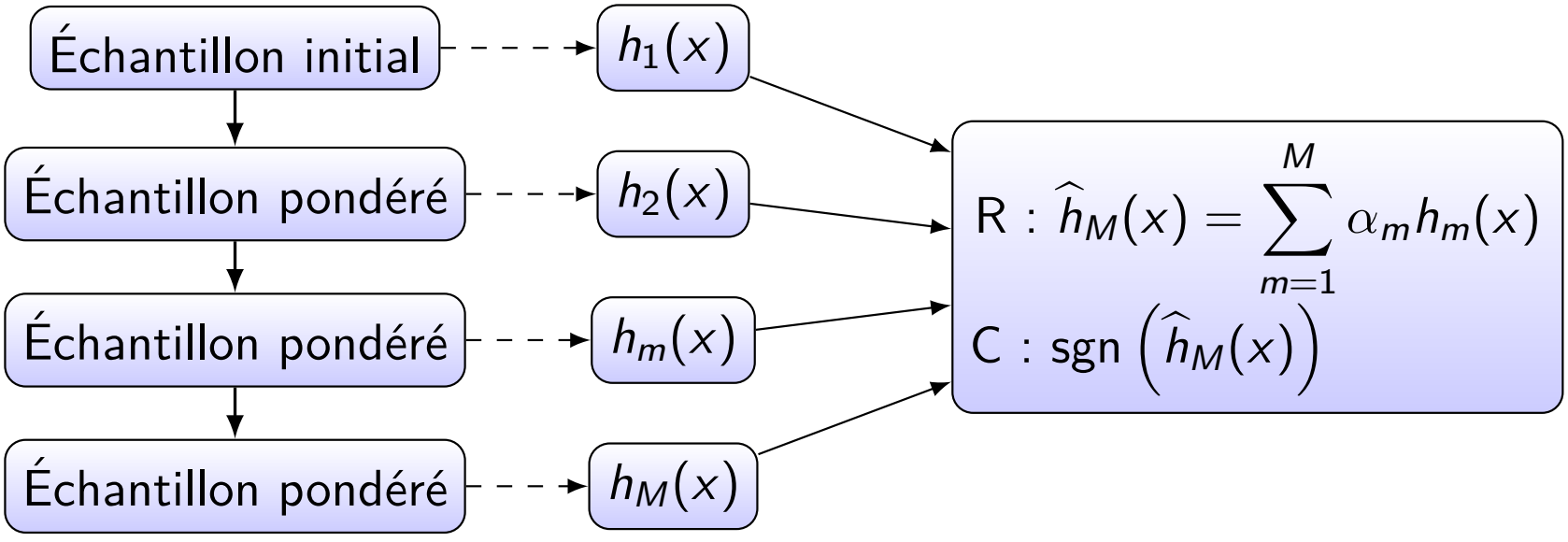


Figure 19 – Boosting : un classifieur faible est entraîné sur l'ensemble des données initiales. Pondération $\neq$ des données bien et mal classées favorisant ces dernières. Second classifieur entraîné sur données pondérées et remet à jour les pondérations, etc. Classifieur final = moyenne pondérée des classifieurs faibles.

Algorithme AdaBoost -3-

Diagramme plus explicite pour illustrer le Boosting :



Forme du prédicteur Boosting (C : classifieur, R : régresseur).

À chaque itération, $h_m(x) = h_{m-1}(x) + \alpha h$
où α et h doivent rendre h_m minimal pour R_n.

Algorithme AdaBoost -4- : l'algorithme

Initialisation : $w_i(0) = 1/n$. Puis à chaque itération $m = 1, ..., M$,

1. Entraîner $h_m(x)$ sur l'échantillon pondéré par $w = (w_i(m))_i$:

$$h_m \in \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} w_i(m) \mathbb{1}_{[y_i \neq h(x_i)]}. \tag{20}$$

2. Calculer l'erreur normalisée :

$$\epsilon_m = \sum_{i=1}^{n} w_i(m) \mathbb{1}_{[y_i \neq h_m(x_i)]} / ||w||_1. \tag{21}$$

3. Calculer le poids de l'itération m : $\alpha_m = \ln \sqrt{(1 - \epsilon_m)/\epsilon_m}$.

4. Mettre à jour les poids des observations :

$$w_i(m+1) = w_i(m) \exp\left(\alpha_m \mathbb{1}_{[y_i \neq h_m(x_i)]}\right), \ \forall i = 1, ..., n. \tag{22}$$

$$\Rightarrow \widehat{h}_M(x) = \sum_{m=1}^{M} \alpha_m h_m(x) \tag{23}$$

Algorithme AdaBoost -5- : pourquoi ça marche

$\forall m$, $h_m = h_{m-1} + \widehat{\alpha}\widehat{h}$, avec h_m min. ϕ-risque empirique / perte Boosting :

$$(\widehat{h}, \widehat{\alpha}) \in \underset{h \in \mathcal{H}, \alpha \in \mathbb{R}_+}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \phi\left(-y_i [h_{m-1}(x_i) + \alpha h(x_i)]\right) \tag{24}$$

$$= \underset{h,\alpha}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} w_i(m) \exp\left(-y_i \alpha h(x_i)\right) \tag{25}$$

$$= \underset{h,\alpha}{\arg\min} \left(\frac{e^{-\alpha}}{n} \sum_{y_i = h(x_i)} w_i(m) + \frac{e^{\alpha}}{n} \sum_{y_i \neq h(x_i)} w_i(m) \right) \tag{26}$$

$\phi(x) = e^x$, $w_i(m) = \exp(-y_i h_{m-1}(x_i))$.

On trouve $\widehat{h}$, on déduit $\widehat{\alpha} = \alpha_m$: démo en exercice et dans le polycopié.

Algorithme AdaBoost -6- : propriétés

Les classifieurs faibles utilisés ne doivent pas être trop faibles : $\epsilon_m = 1/2 - \gamma_n$ avec $\gamma_n > 0$.

Majoration de l'erreur empirique :

$$R_n(\widehat{h}_M) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{[y_i \neq \widehat{h}_M(x_i)]} \leq \exp\left(-2\sum_{m=1}^{M}\gamma_m^2\right) \tag{27}$$

Majoration de l'erreur de généralisation (Freund et Shapire) :

$$R(\widehat{h}_M) = \mathbb{P}[\widehat{h}_M(X) \neq Y] \leq R_n(\widehat{h}_M) + \mathcal{O}\left(\sqrt{\frac{MV}{n}}\right) \tag{28}$$

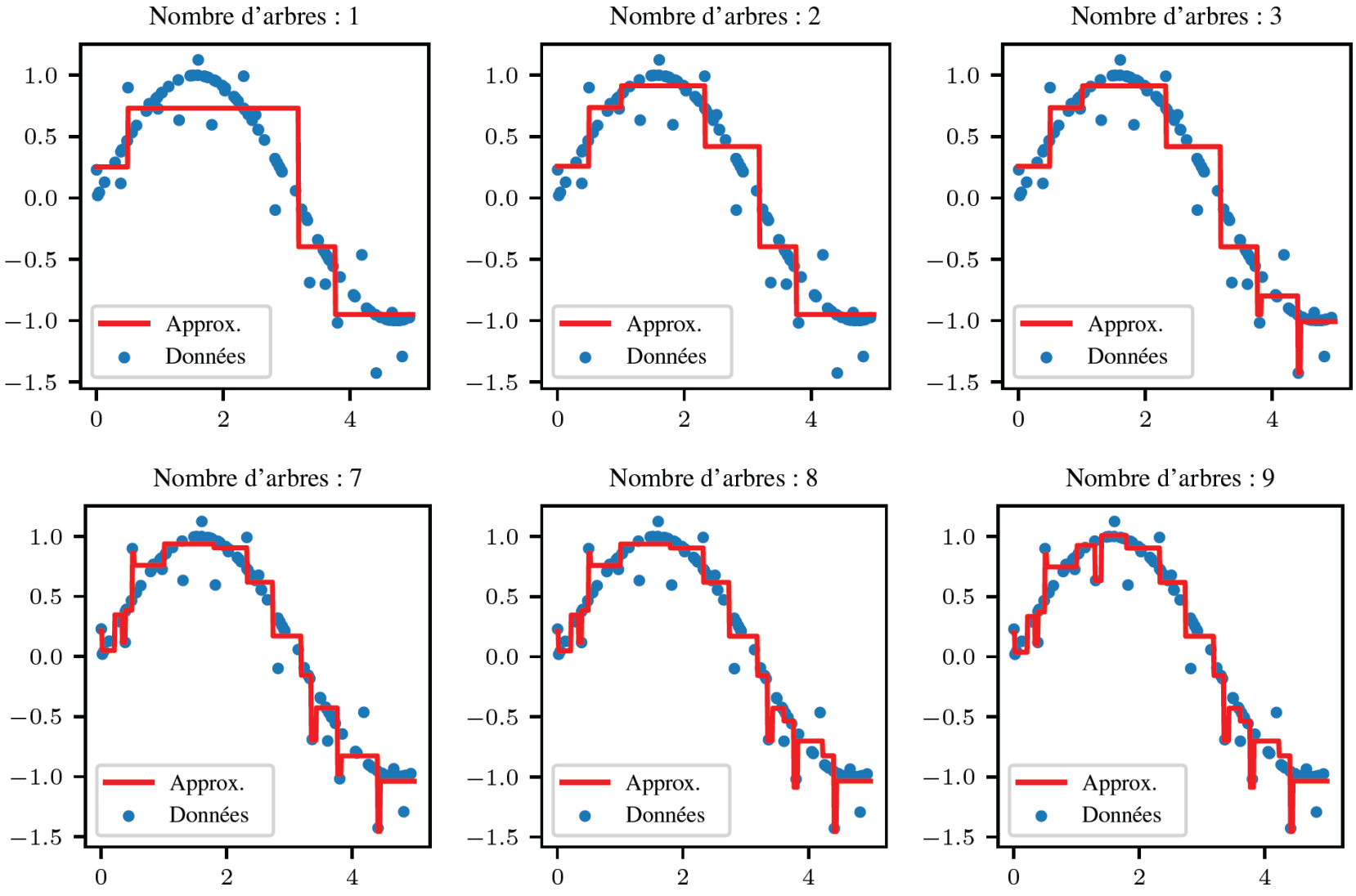
V dimension de Vapnik de l'espace des classifieurs faibles.

Bartlett et Traskin : si $M = M_n = n^{1-\epsilon}$, $\epsilon \in]0,1[$, alors Adaboost est fortement et universellement consistant.

Généralisations

- À des étiquettes non binaires : SAMME Stagewise Additive Modeling Multi-class Exponentiel (c.f. polycopié).
- À de la régression : L^2 Boosting (c.f. polycopié).
- À d'autres fonctions de perte : logistique, etc.
- À d'autres méthodes de minimisation du risque empirique : $\Rightarrow$ Gradient Boosting et XGBoost (eXtreme Gradient Boosting).

Algorithme Adaboost -7- : performances



Régression Boosting sur des arbres de profondeur 2.

Gradient Boosting -0- : Descente de gradient -1-

- Gradient Boosting = Boosting par descente de gradient.

- La descente de gradient s'inspire de la méthode de Newton-Raphson. Si u strictement convexe de $\mathbb{R}$ dans $\mathbb{R}$,

$$x_{n+1} = x_n - \frac{u(x_n)}{u'(x_n)}.$$

En x_n l'équation de la tangente à la courbe de u est

$$y = u(x) = u(x_n) + u'(x_n)(x - x_n)$$

- $y = 0 \Rightarrow$ la tangente coupe $(0x)$ en x_{n+1}.
- De proche en proche, on se rapproche d'un zéro de u.

- Descente de gradient : remplacer u par u', fixer η /

$$x_{n+1} = x_n - \eta u'(x_n)$$

Gradient Boosting -0- : Descente de gradient -2-

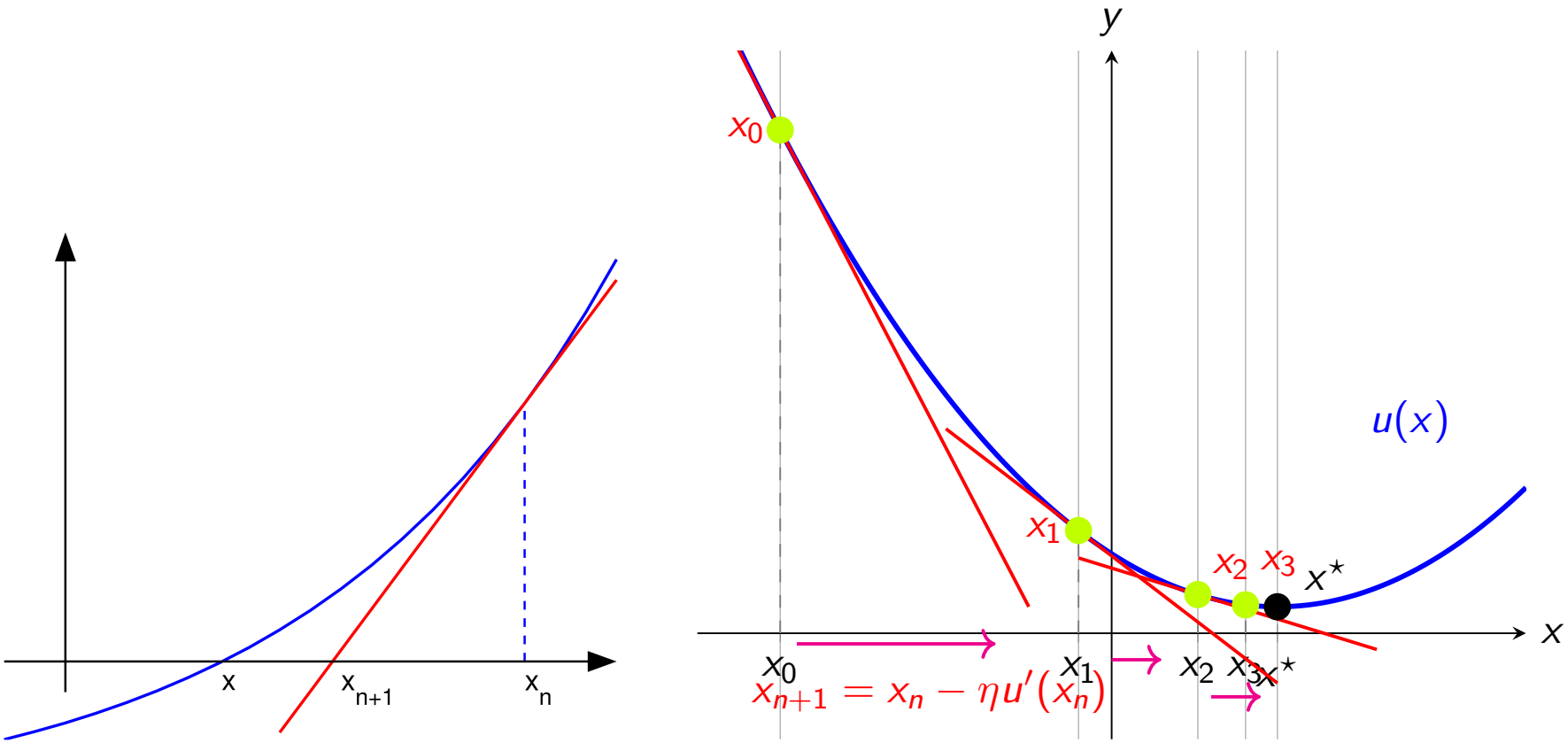


Figure 20 – Illustration de l'algorithme de Newton-Raphson (à gauche) et de celui de la descente de gradient (à droite).

Gradient Boosting -0- : Descente de gradient -3-

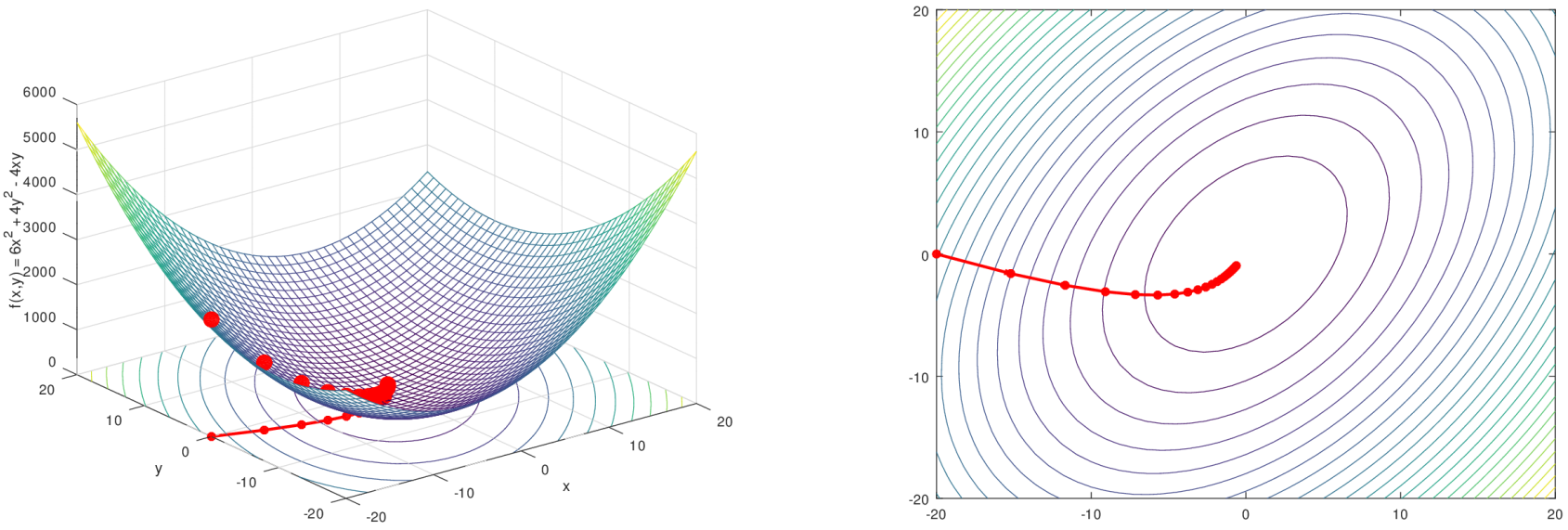


Figure 21 – Illustration de l'algorithme de descente de gradient en dimension 2.

Gradient Boosting -0- : le problème de la notation -1-

• Si régression et $l(y, y') = (y - y')^2/2$. i fixé, $y'_i = h(x_i)$. Erreur de prédiction pour (x_i, y_i) : $y_i - y'_i = y_i - h(x_i)$.

$$\frac{\partial}{\partial y'} l(y, y') = \frac{1}{2} \frac{\partial l}{\partial y'} (y - y')^2 = y' - y. \tag{29}$$

• évaluée en $(y_i, h(x_i))$, $=$ à l'opposé de la perte.

$$R_n(h) = \frac{1}{n} \sum_{i=1}^{n} l(y_i, h(x_i)) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - h(x_i))^2. \tag{30}$$

• $R_n(h)$ fonction de h... et de $y_1, ..., y_n$ et $y'_1 = h(x_1), ..., y'_n = h(x_n)$. Le gradient de $R_n(h)$ (fonction de y'), évalué en $(y_i, h(x_i))_i$, est

$$\frac{\partial R_n(h)}{\partial y'_i}(y'_1, ..., y'_n)\Big|_{y'_i = h(x_i)} = \frac{1}{n} \frac{\partial l}{\partial y'_i}(y_i, y'_i)\Big|_{y'_i = h(x_i)} = \frac{1}{n} \left(h(x_i) - y_i\right). \tag{31}$$

Gradient Boosting -0- : le problème de la notation -2-

• On note ce vecteur, de façon peu rigoureuse et très ambiguë :

$$\nabla R_n(h) = \left(\frac{\partial l(y_i, h(x_i))}{\partial h(x_i)} \right)^T_{i=1,..,n} \tag{32}$$

• Vecteur, n coord ∂ R_n / 2^e coord. $(h(x_i))$, évalué en $(y_i, h(x_i))$.

• En fait, la minimisation de $R_n(h)$ est effectuée sur la fonction h et non sur les points de l'échantillon. On parle de gradient fonctionnel car minimisation sur h, à x_i et y_i fixés.

• À l'itération m, Boosting $\Rightarrow h_{m-1}(x)$, On veut $h_m(x)$

$$h_m(x) = h_{m-1}(x) + \alpha h(x) \tag{33}$$

où $\alpha h(x)$ minimise l'erreur. Si $\alpha h(x)$ égal à la perte quad., $\Rightarrow$ $h_m(x_i) = y_i \Rightarrow$ erreur prédiction nulle. On choisit comme expression de $h(x)$ le gradient du risque.

Gradient Boosting -1- : principe

- Adaboost : cas particulier d'algorithme de descente de gradient.
- $\widehat{h}_M = \sum_{m=1}^{M} \alpha_m h_m$ minimisant $R_n(h) = \sum_{i=1}^{n} l(h(x_i), y_i)/n$?
- Pas de solution explicite $\Rightarrow$ solution approchée.
- Idée : algorithme de descente de gradient de type Newton-Raphson.

- À une itération donnée, on a un classifieur h_{m-1} à améliorer.
- On lui ajoute g / $R_n(h_{m-1} + \alpha g)$ diminue au maximum.
- $\Rightarrow$ opposé du gradient : $g = -\nabla R_n(h_{m-1})$.

$$h_m(x) = h_{m-1}(x) - \alpha \nabla R_n(h_{m-1}) : \left(h_{m-1}(x_i) - \frac{\alpha}{n} \frac{\partial l}{\partial y_i'}(y_i, h_{m-1}(x_i)) \right)_i$$

Gradient Boosting -2- : principe - suite

Deux problèmes :

- La récurrence donne h_m uniquement aux points x_i.
- Le gradient g n'a aucune raison d'appartenir à $\mathcal{H}$, donc h_m non plus.

Pour obtenir $h_m(x)\ \forall x \in \mathbb{R}^d$ et assurer que $h_m \in \mathcal{H}$,
$\Rightarrow$ régression sur $\mathcal{D}_{n,m} = (x_i, u_i)_{i=1,..,n}$ avec

$$u_i = u_i(m) = -\frac{\partial l}{\partial y_i'}(y_i, h_{m-1}(x_i))$$

On cherche la fonction $h \in \mathcal{H}$ la plus proche de g :

La solution h_m s'obtient en ajustant le classifieur sur $(x_i, u_i)_i$.

On détermine ensuite la valeur optimale α_m de α.

Gradient Boosting -3- : l'algorithme de Friedman (1999)

c est une constante.

- Initialisation : $h_0(.) = \frac{1}{n} \arg\min_c \sum_{i=1}^{n} l(c, y_i)$
- Pour $m = 1, ..., M$, étant donné h_{m-1} calculé à l'itération précédente,

  $\forall i = 1, .., n$, calculer les pseudos-résidus u_i.

  ajuster un classifieur de $\mathcal{H}$ sur l'échantillon $(x_i, u_i)_i$:

$$\overline{h}_m = \arg\min_{h,\alpha} \sum_i [u_i - \alpha h(x_i)]^2$$

  déterminer α optimal :

$$\alpha_m = \arg\min_{\alpha} \sum_i l\left(h_{m-1}(x_i) + \alpha \overline{h}_m(x_i)\right)$$

  mise à jour : $h_m(x) = h_{m-1}(x) + \alpha_m \overline{h}_m(x)$.

En sortie, on obtient bien une combinaison linéaire d'estimateurs de $\mathcal{H}$:

$$\widehat{h}_M(x) = \sum_{m=1}^{M} \alpha_m h_m(x).$$

Gradient Boosting -4- : remarques sur l'algorithme

Plusieurs paramètres sont à calibrer :

- La fonction de perte.
- Le nombre M d'itérations.
- Le paramètre de régularisation α_m (learning rate : souvent ~ 0.1).
- Les paramètres de chaque classifieur constituant la combinaison linéaire.

La fonction de perte doit être dérivable et convexe. En classification binaire :

- Adaboost : $l(y, h(x)) = \exp(-yh(x))$.
- Logiboost : $l(y, h(x)) = \ln(1 + \exp(-2yh(x)))$.

En régression, L^2 boosting : $l(y, h(x)) = (y - h(x))^2/2$.

Gradient Boosting -5- : remarques finales

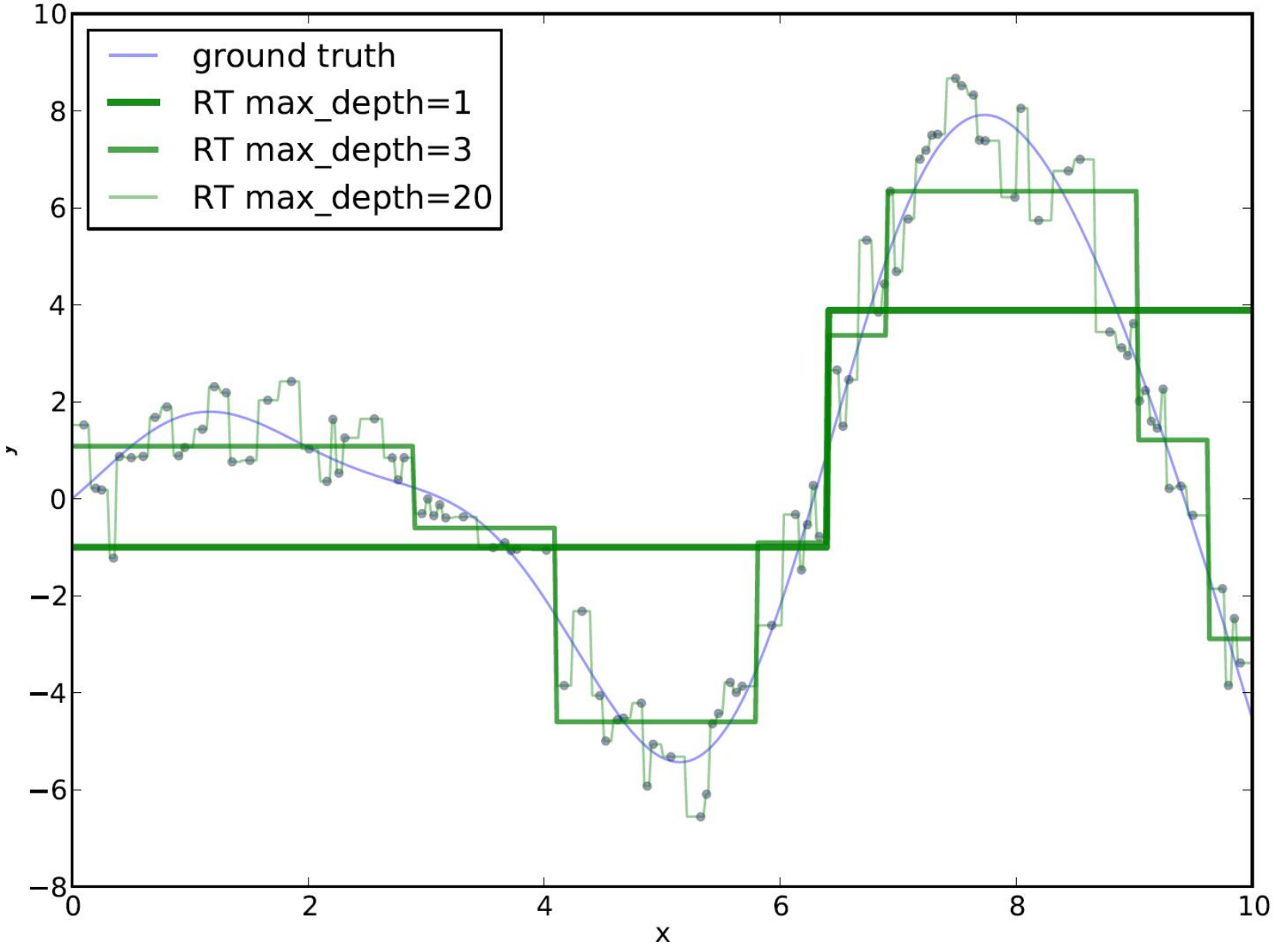
- On utilise souvent Adaboost et Gradient Boost avec des arbres CART peu profonds (classifieurs faibles) : $\mathcal{H}$ est un ensemble de combinaisons linéaires d'arbres de décision.
- Gradient Boosting stochastique : à chaque itération, on entraîne le classifieur sur un sous échantillon aléatoire de $\mathcal{D}_n$ (sans remise).
- Gradient Boosting avec $\alpha = 1$ et perte Boosting $\sim$ Adaboost.

La fonction de perte doit être dérivable et convexe. En classification binaire :

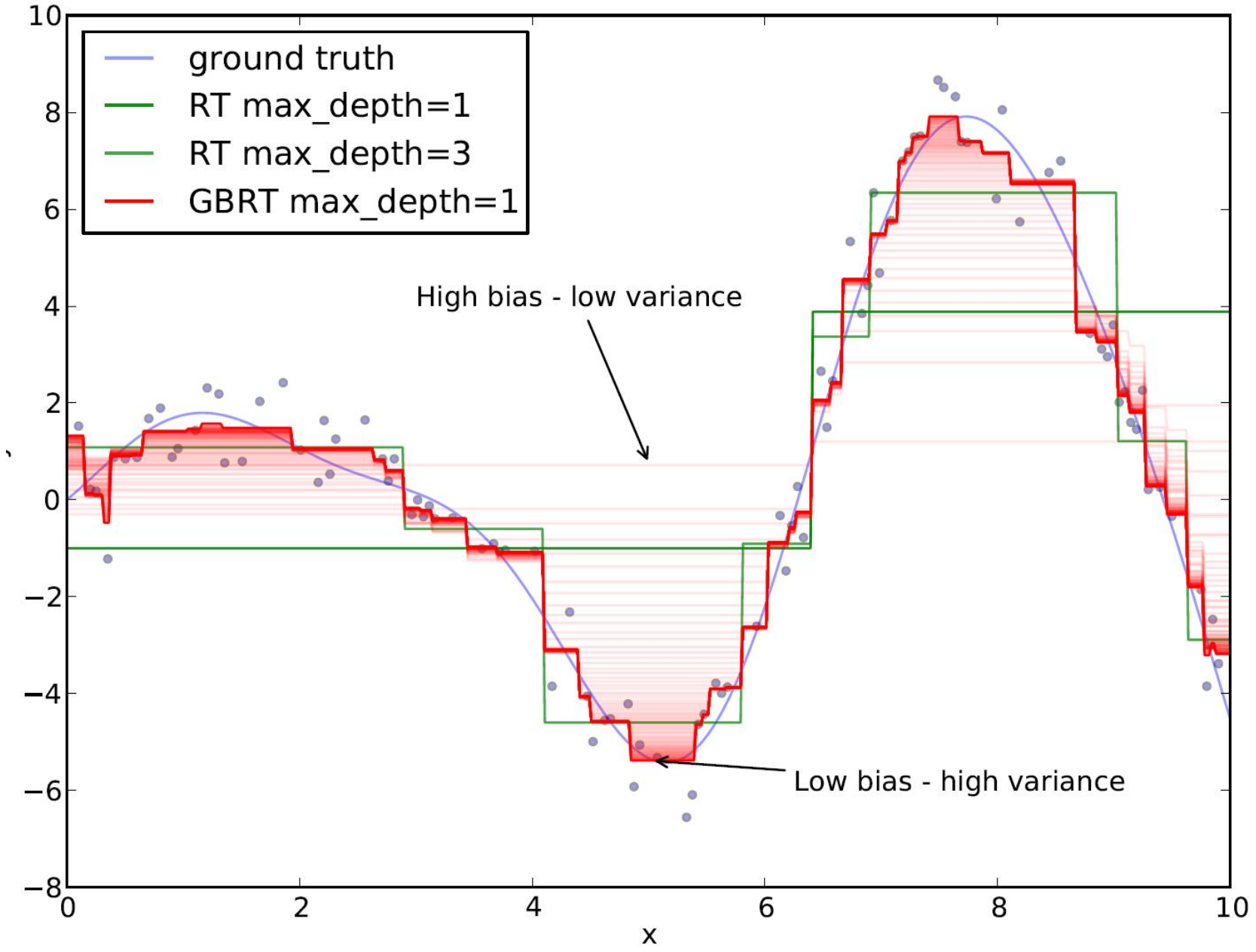
- Adaboost : $l(y, h(x)) = \exp(-yh(x))$.
- Logiboost : $l(y, h(x)) = \ln(1 + \exp(-2yh(x)))$.

En régression, L^2 boosting : $l(h(x), y) = (y - h(x))^2/2$.

Gradient Boosting -6- : performances



Gradient Boosting -7- : performances



Extreme Gradient Boosting : XGBoost

- Version sophistiquée de Gradient Boosting (Chen et Guestrin, 2016).
- Ajoute de la régularisation dans les entrainements des classifieurs faibles.
- Utilise Newton-Raphson à la place du gradient.
- développement limité au second ordre de la fonction de perte.
- Difficile à calibrer....
- ... mais très efficace si bien calibré.
- plus de détails dans le polycopié.

La fonction objectif possède un terme de régularisation :

$$\frac{1}{n}\sum_{i=1}^{n} l(h_m(x_i), y_i) + \sum_{j=1}^{s} \gamma(h_j)$$

$\gamma(h_j)$ pénalise h_j en fonction de sa complexité (nombre de feuilles si c'est un arbre).

γ élevé : complexité élevée (arbre profond).

Bibliographie

- Freund, Schapire. Experiments with a new boosting algorithm. roceedings of the 13th Inter. Conf. on Machine Learning. 1996.
- Freund, Shapire. Boosting. MIT Press. 2012.
- Friedman. Greedy function approximation : A gradient boosting machine. Annals of Statistics. 2001.
- Friedman, Hastie, Tibshirani. Additive logistic regression : A statistical view of boosting. Annals of statistics. 2000.
- Chen, Guestrin. XGBoost : A Scalable Tree Boosting System. 2016.
- + Chapitre 8 : Tree-Based Methods (p.331-363) Intro to Statistical Learning with Python.
- + Chapitre 7 du polycopié de Frédéric Sur.