Support Vector Machines

Master parcours SSD - UE Apprentissage Statistique II

Pierre Mahé - bioMérieux & Université de Grenoble-Alpes

Ce cours

Support Vector Machines - Machines à Vecteur de Support [1] :

- un algorithme
  - de classification binaire
  - basé sur des séparateurs linéaires
- qui se généralise :
  - à la régression
  - à la classification multi-classe
  - au cadre non-linéaire grâce aux fonctions noyaux
- très performant sur de nombreux problèmes réels
  - haute dimension
  - données structurées
- relativement simple à paramétrer

1. ou Séparateurs à Vaste Marge.

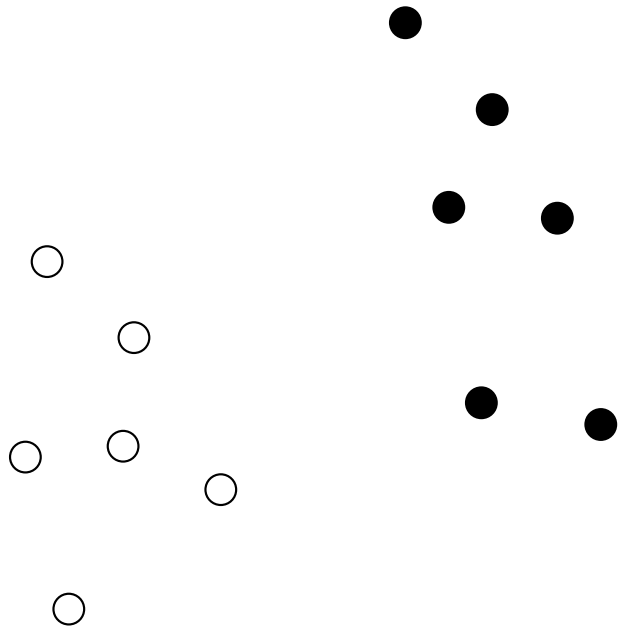
Plan

1. Séparateurs linéaires
2. SVMs linéaires & données séparables
3. SVMs linéaires & données non-séparables
4. SVMs non-linéaires & fonctions noyaux
5. SVMs en pratique
6. Mise en oeuvre avec `Scikit-Learn`

Séparateurs linéaires

Cadre général

Cadre général : **classification binaire**

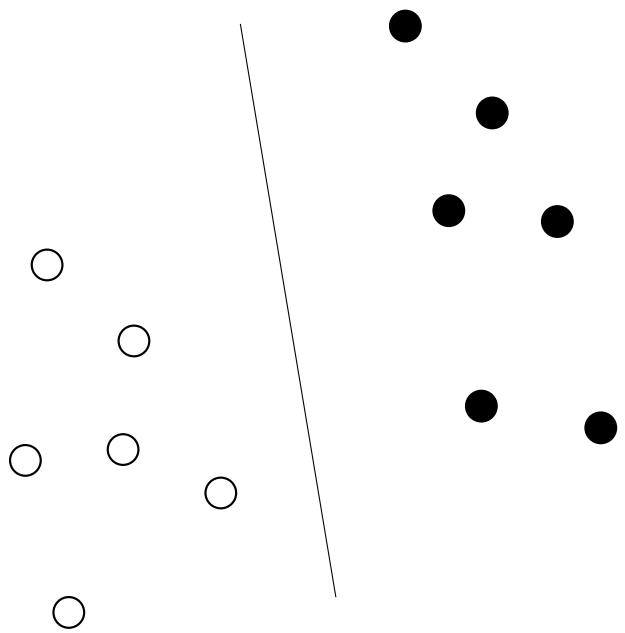


$\Rightarrow$ couples $(x_i, y_i) \in \mathbb{R}^p \times \{-1, 1\}$, pour $i = 1, ..., n$.

- on prendra $p = 2$ pour les illustrations

Séparateur linéaire

Si $p = 2$: séparateur linéaire = droite



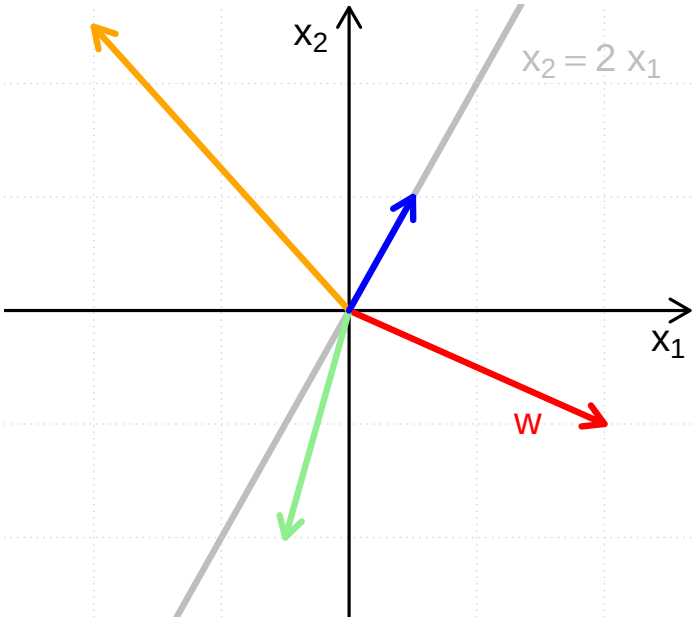
Une droite D est définie par un couple $(w, b) \in \mathbb{R}^2 \times \mathbb{R}$:

$$\boxed{D_{w,b} = \{x \in \mathbb{R}^2 : \langle w, x \rangle + b = 0\}}$$

$\Rightarrow$ w est un vecteur normal de $D_{w,b}$.

Séparateur linéaire - illustration vecteur normal

Illustration : $x = [x_1 \;\; x_2] \in \mathbb{R}^2$ et la droite $x_2 = 2x_1$.



On a donc $2x_1 - x_2 = 0$,
soit $\langle w, x \rangle + b = 0$ avec :

- $w = [2 \;\; -1]$
- $b = 0$

$\Rightarrow$ w est $\perp$ à la droite.

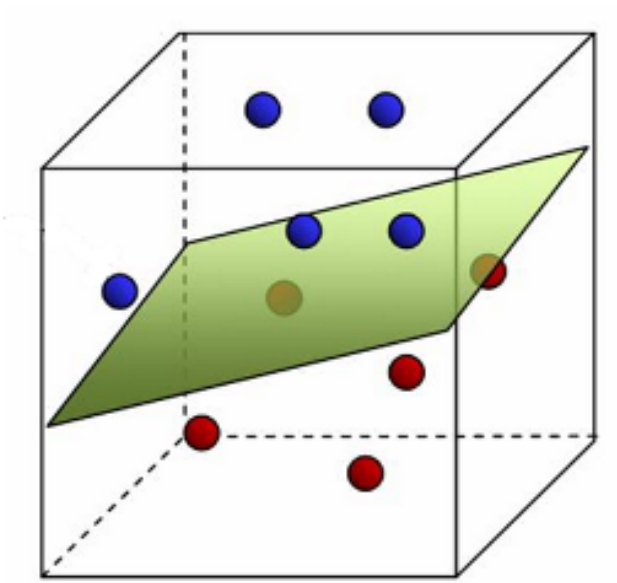
- les vecteurs $\perp$ à w définissent les points de la droite
- les vecteurs "pointant" dans la direction de w ont un score positif : $\langle w, . \rangle + b > 0$ (angle $\in [-\pi/2, \pi/2]$)
- les vecteurs "pointant" dans la direction inverse ont un score négatif : $\langle w, . \rangle + b < 0$ (angle $\in [\pi/2, 3\pi/2]$)

Séparateur linéaire

Plus généralement, on parle d'hyperplan :

$$\boxed{H_{w,b} = \{x \in \mathbb{R}^p : \langle w, x \rangle + b = 0\}}$$

avec $(w, b) \in \mathbb{R}^p \times \mathbb{R}$ et $p \geq 2$.



$\Rightarrow$ une surface en 3D [2].

2. image tirée de `http://fouryears.eu/2009/02/26/`

Séparateur linéaire

Soit le jeu d'apprentissage $\mathcal{T} = \{(x_1, y_1), ..., (x_n, y_n)\}$.

- $(x_i, y_i) \in \mathbb{R}^p \times \{-1, 1\}$.

$\mathcal{T}$ est linéairement séparable si il existe un hyperplan $H_{w,b}$ permettant de discriminer correctement les données :

$$\begin{cases} \langle w, x_i \rangle + b > 0 & \text{si } y_i = 1, \\ \langle w, x_i \rangle + b < 0 & \text{si } y_i = -1. \end{cases}$$

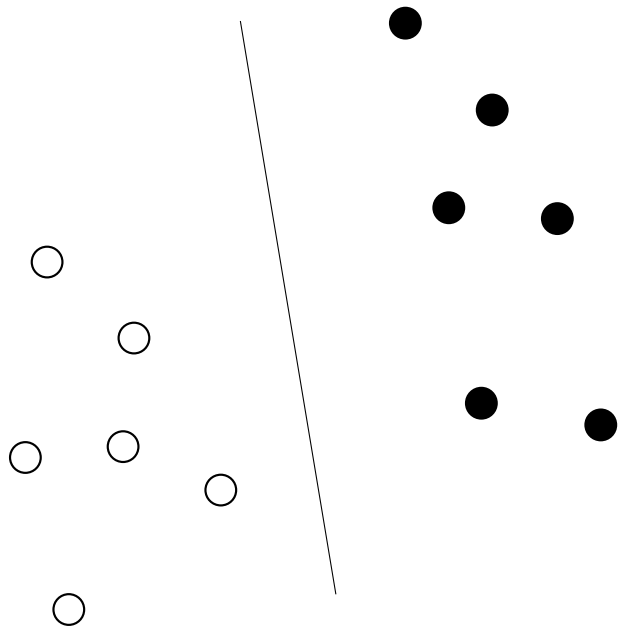
Soit de manière équivalente :

$$\boxed{y_i\big(\langle w, x_i \rangle + b\big) > 0, \quad \forall i = 1, ..., n.}$$

$\Rightarrow$ le signe de $f(x) = \langle w, x \rangle + b$ donne la classe +1/-1.

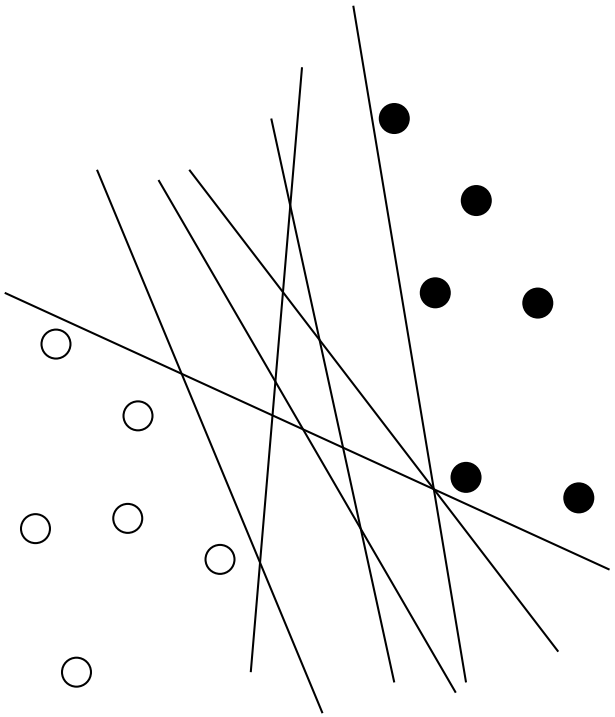
Séparateur linéaire

Illustration : un jeu de données linéairement séparable



Séparateur linéaire

Illustration : un jeu de données linéairement séparable

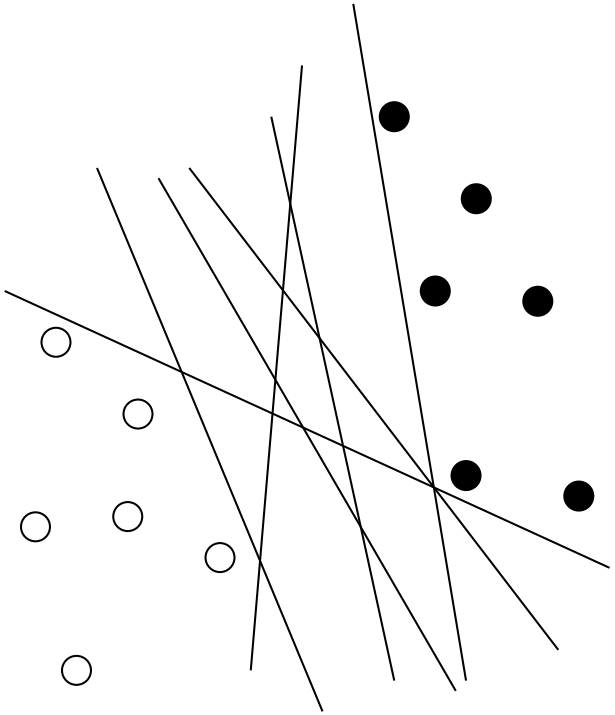


$\Rightarrow$ problème : une infinité d'hyperplans séparant les données !
$\Rightarrow$ lequel choisir ?

SVMs linéaires - cadre séparable

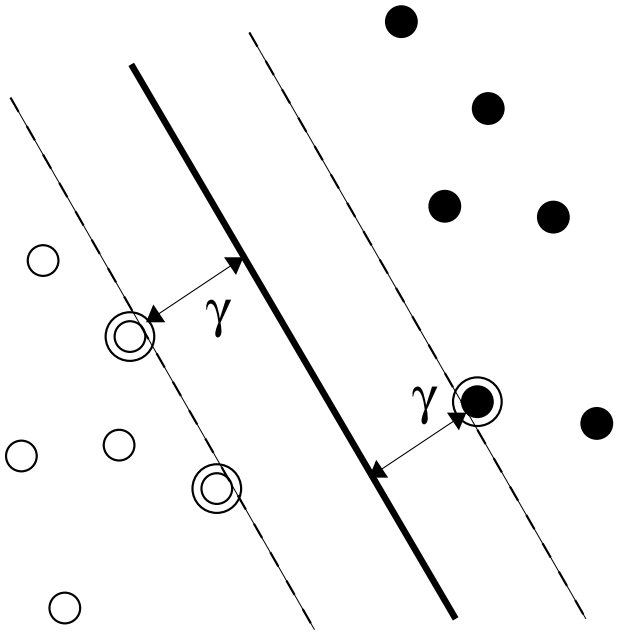
Séparateurs linéaires et marges

Hyperplans & données séparables : une infinité de solutions



Séparateurs linéaires et marges

Hyperplans & données séparables : une infinité de solutions



⇒ solution SVM : prendre celui avec la plus grande marge γ

Distances et marge

La distance du point x_i à $H_{w,b}$ vaut :

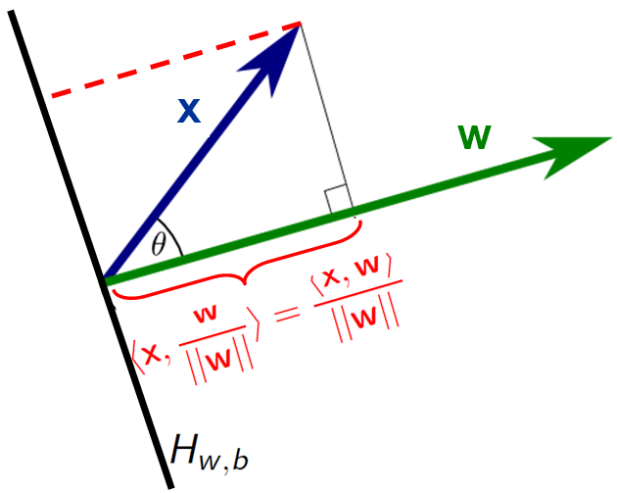
$$\frac{|(\langle w, x_i \rangle + b)|}{||w||}$$

soit :

$$\frac{y_i(\langle w, x_i \rangle + b)}{||w||}$$

pour un hyperplan séparateur.

- $\langle w, x_i \rangle + b > 0$ si $y_i = 1$
- $\langle w, x_i \rangle + b < 0$ si $y_i = -1$



Distances et marge

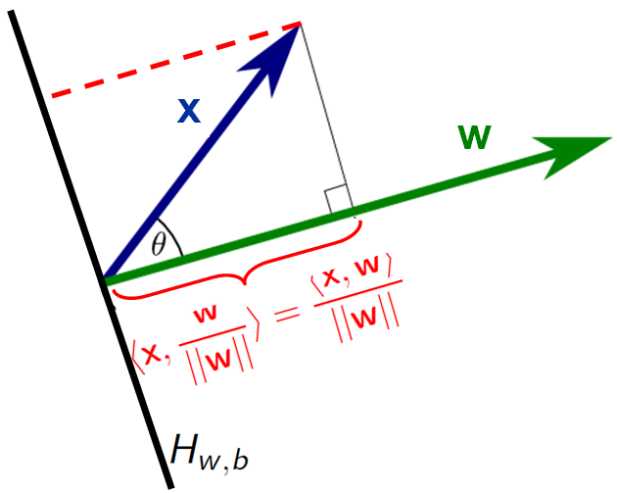
La distance du point x_i à $H_{w,b}$ vaut :

$$\frac{|(\langle w, x_i \rangle + b)|}{||w||}$$

soit :

$$\frac{y_i(\langle w, x_i \rangle + b)}{||w||}$$

pour un hyperplan séparateur.



- $\langle w, x_i \rangle + b > 0$ si $y_i = 1$
- $\langle w, x_i \rangle + b < 0$ si $y_i = -1$

$\Rightarrow$ la marge γ vaut donc : $\boxed{\gamma = \min_{i=1,\ldots,n} \frac{y_i(\langle w, x_i \rangle + b)}{||w||}}$

Hyperplan canonique

Un hyperplan est défini à un facteur multiplicatif près :

$$\boxed{\langle w, x\rangle + b = 0 \Leftrightarrow \langle \alpha w, x\rangle + \alpha b = 0 \quad \forall \alpha \in \mathbb{R}}$$

$\Rightarrow$ $H_{w,b}$ et $H_{\alpha w,\alpha b}$ définissent le même hyperplan.

Hyperplan canonique

Un hyperplan est défini à un facteur multiplicatif près :

$$\boxed{\langle w, x\rangle + b = 0 \Leftrightarrow \langle \alpha w, x\rangle + \alpha b = 0 \quad \forall \alpha \in \mathbb{R}}$$

$\Rightarrow$ $H_{w,b}$ et $H_{\alpha w, \alpha b}$ définissent le même hyperplan.

On définit un hyperplan (séparateur) canonique avec la contrainte :

$$\min_{i=1,\dots,n} y_i(\langle w, x_i\rangle + b) = 1$$

$\Rightarrow$ sa marge vaut donc :

$$\gamma = \min_{i=1,\dots,n} \frac{y_i(\langle w, x_i\rangle + b)}{||w||} \quad \Rightarrow \quad \boxed{\gamma = \frac{1}{||w||}}$$

SVM linéaire & séparable - formalisation

Le problème est donc de trouver $(w, b) \in \mathbb{R}^p \times \mathbb{R}$ tels que :

1. la marge $\frac{1}{||w||}$ soit maximale
2. l'hyperplan soit canonique : $\min_{i=1,\dots,n} y_i(\langle w, x_i \rangle + b) = 1$

SVM linéaire & séparable - formalisation

Le problème est donc de trouver $(w, b) \in \mathbb{R}^p \times \mathbb{R}$ tels que :

1. la marge $\frac{1}{||w||}$ soit maximale
2. l'hyperplan soit canonique : $\min_{i=1,\dots,n} y_i(\langle w, x_i \rangle + b) = 1$

$\Rightarrow$ se formalise comme un problème d'optimisation :

$$(w^*, b^*) = \underset{w \in \mathbb{R}^p, b \in \mathbb{R}}{\operatorname{argmin}} \quad \frac{1}{2}||w||^2$$
$$\text{s.t.} \quad y_i(\langle w, x_i \rangle + b) \geq 1, \ i = 1, \dots, n.$$

- un problème convexe avec une solution unique
- ($||w|| \to 1/2||w||^2$ pour simplifier sa résolution)

SVM linéaire & séparable - résolution

SVM : problème primal

$$(w^*, b^*) = \underset{w \in \mathbb{R}^p, b \in \mathbb{R}}{\operatorname{argmin}} \quad \frac{1}{2}||w||^2$$
$$\text{s.t.} \quad y_i(\langle w, x_i \rangle + b) \geq 1, \ i = 1, \ldots, n.$$

On peut le résoudre avec une formulation Lagrangienne :

1. introduire un paramètre $\alpha_i \geq 0$ pour chaque contrainte
   - i.e., pour chaque instance (x_i, y_i)
   - α_i = **multiplicateur de Lagrange**
2. ajouter les contraintes à la fonction objective :

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i \big(y_i(\langle w, x_i \rangle + b) - 1\big).$$

SVM linéaire & séparable - résolution

Formulation Lagrangienne :

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i \big(y_i(\langle w, x_i \rangle + b) - 1\big)$$

1. à minimiser selon (w, b)
   - minimisation de la fonction objective
2. à maximiser selon α
   - respect des contraintes :

$$\alpha_i \big(y_i(\langle w, x_i \rangle + b) - 1\big) \geq 0$$

si la contrainte est satisfaite.

SVM linéaire & séparable - résolution

Formulation Lagrangienne :

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i \big(y_i(\langle w, x_i \rangle + b) - 1\big)$$

$\Rightarrow$ minimiser selon (w, b) :

$$\frac{\partial L(w, b, \alpha)}{\partial w} = 0 \qquad \frac{\partial L(w, b, \alpha)}{\partial b} = 0$$

soit :

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \qquad \sum_{i=1}^{n} \alpha_i y_i = 0$$

SVM linéaire & séparable - résolution

On remplace w par $\sum_{i=1}^{n} \alpha_i y_i x_i$ dans $L(w, b, \alpha)$:

$$L(w, b, \alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

$\Rightarrow$ à maximiser selon $\alpha \in \mathbb{R}^n$.

SVM linéaire & séparable - résolution

On remplace w par $\sum_{i=1}^{n} \alpha_i y_i x_i$ dans $L(w, b, \alpha)$:

$$L(w, b, \alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

$\Rightarrow$ à maximiser selon $\alpha \in \mathbb{R}^n$.

SVM : problème dual

$$\alpha^* = \underset{\alpha \in \mathbb{R}^n}{\operatorname{argmax}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

$$\text{s.t.} \quad \alpha_i \geq 0, \quad i = 1, \ldots, n$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$

$\Rightarrow$ un problème quadratique (QP)

SVM linéaire & séparable - résolution

SVM : 2 problèmes d'optimisation équivalents [3] :

- problème primal : défini selon $(w, b) \in \mathbb{R}^p \times \mathbb{R}$
- problème dual : défini selon $\alpha \in \mathbb{R}^n$

liés par la relation $w = \sum_{i=1}^{n} \alpha_i y_i x_i$.

3. équivalence car dualité forte du problème convexe

SVM linéaire & séparable - résolution

SVM : 2 problèmes d'optimisation équivalents [3] :

- problème primal : défini selon $(w, b) \in \mathbb{R}^p \times \mathbb{R}$
- problème dual : défini selon $\alpha \in \mathbb{R}^n$

liés par la relation $w = \sum_{i=1}^{n} \alpha_i y_i x_i$.

A l'optimum on a :

$$\boxed{\alpha_i^* \left(y_i (\langle w^*, x_i \rangle + b^*) - 1 \right) = 0}, \quad \forall i = 1, \ldots, n.$$

$\Rightarrow$ deux cas de figure pour le point (x_i, y_i) :

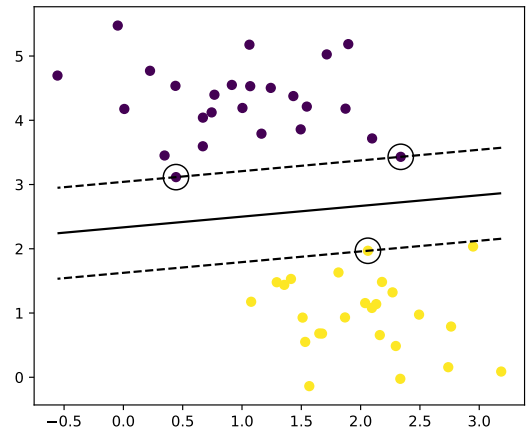
- $y_i(\langle w^*, x_i \rangle + b^*) > 1 \Rightarrow \alpha_i^* = 0$
- $y_i(\langle w^*, x_i \rangle + b^*) = 1 \Rightarrow \alpha_i^* > 0$

3. équivalence car dualité forte du problème convexe

Vecteurs support

Deux cas de figure pour (x_i, y_i) :

- $y_i(\langle w^*, x_i \rangle + b^*) > 1 \Rightarrow \alpha_i^* = 0$
  $\Rightarrow$ en dehors de la marge
- $y_i(\langle w^*, x_i \rangle + b^*) = 1 \Rightarrow \alpha_i^* > 0$
  $\Rightarrow$ sur la frontière de la marge



Vecteurs support

$\Rightarrow$ les points (x_i, y_i) tels que $\alpha_i^* > 0$

$\Rightarrow$ les seuls qui définissent l'hyperplan : $w = \sum_{i=1}^{n} \alpha_i y_i x_i$

En pratique

1. Calcul de w :
   - résoudre le problème dual à partir de $\mathcal{T} = \{(x_i, y_i)\}$
   - on obtient $\{\alpha_i^*\}_{i=1,\dots,n}$ et $w^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i$
2. Calcul de b :
   - pour les vecteurs supports on a $y_i\left(\langle w^*, x_i\rangle + b^*\right) = 1$
   - on en déduit $b^* = y_i - \langle w^*, x_i\rangle$ pour un VS (x_i, y_i).

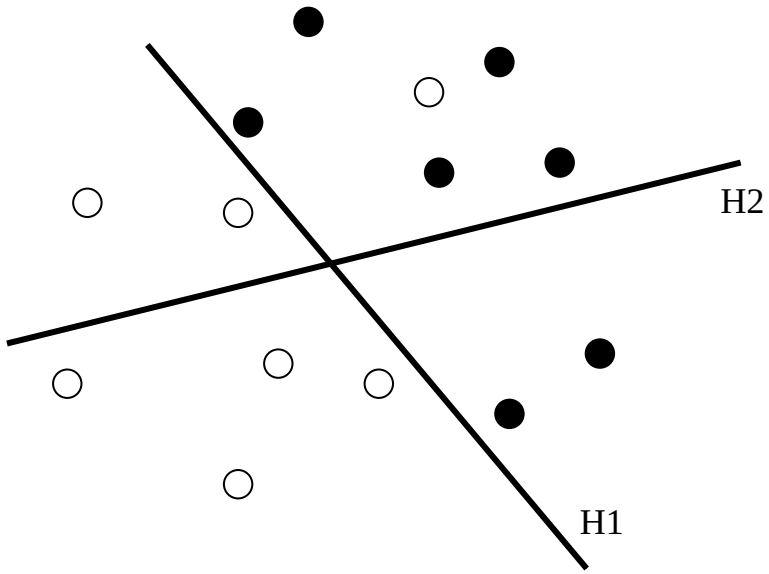
On obtient la fonction de décision :

$$f(x) = \langle w^*, x\rangle + b^* = \sum_{i=1}^{n} \alpha_i^* y_i \langle x_i, x\rangle + b^*$$

$\Rightarrow$ prédiction +1/-1 à partir du signe de $f(x)$.

SVMs linéaires - cadre non-séparable

Cadre non séparable



$\Rightarrow$ impossible de satisfaire toutes les contraintes :

$$\begin{cases} \langle w, x_i \rangle + b > 0 & \text{si } \ y_i = 1, \\ \langle w, x_i \rangle + b < 0 & \text{si } \ y_i = -1. \end{cases}$$

$\Rightarrow$ pas de solution avec la procédure précédente

SVM linéaire & non-séparable - formalisation

SVM linéaire pour données non séparables :

SVM linéaire & non-séparable - formalisation

SVM linéaire pour données non séparables :

1. relâcher les contraintes :

$$\boxed{y_i\big(\langle w, x_i\rangle + b\big) \geq 1} \quad \Rightarrow \quad \boxed{y_i\big(\langle w, x_i\rangle + b\big) \geq 1 - \xi_i}$$

$\Rightarrow$ la variable $\xi_i \geq 0$ (pour $i = 1, ..., n$) :

- quantifie l'erreur faite au point (x_i, y_i)
- est une variable "ressort" ("slack" variable)

SVM linéaire & non-séparable - formalisation

SVM linéaire pour données non séparables :

1. relâcher les contraintes :

$$\boxed{y_i\big(\langle w, x_i\rangle + b\big) \geq 1} \quad \Rightarrow \quad \boxed{y_i\big(\langle w, x_i\rangle + b\big) \geq 1 - \xi_i}$$

$\Rightarrow$ la variable $\xi_i \geq 0$ (pour $i = 1, ..., n$) :
   - quantifie l'erreur faite au point (x_i, y_i)
   - est une variable "ressort" ("slack" variable)

2. inclure l'erreur globale $\sum_{i=1}^{n} \xi_i$ dans la fonction objective

SVM linéaire & non-séparable - formalisation

SVM linéaire pour données non séparables :

1. relâcher les contraintes :

$$\boxed{y_i\big(\langle w, x_i\rangle + b\big) \geq 1} \quad \Rightarrow \quad \boxed{y_i\big(\langle w, x_i\rangle + b\big) \geq 1 - \xi_i}$$

   $\Rightarrow$ la variable $\xi_i \geq 0$ (pour $i = 1, ..., n$) :
   - quantifie l'erreur faite au point (x_i, y_i)
   - est une variable "ressort" ("slack" variable)

2. inclure l'erreur globale $\sum_{i=1}^{n} \xi_i$ dans la fonction objective

$\Rightarrow$ **"soft-margin"** SVM

- formulation précédente = **"hard-margin"** SVM

SVM linéaire & non-séparable - formalisation

Soft-margin SVM : problème primal

$$(w^*, b^*) = \underset{\substack{w\in\mathbb{R}^p, b\in\mathbb{R}, \\ \xi\in\mathbb{R}^n}}{\operatorname{argmin}} \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad y_i(\langle w, x_i\rangle + b) \geq 1-\xi_i, \quad i = 1,\dots,n,$$

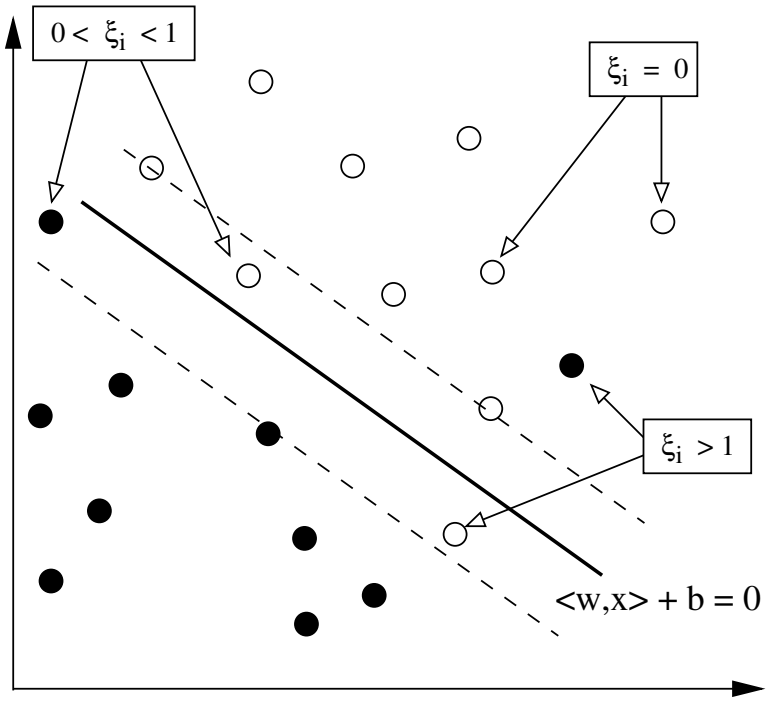
$$\xi_i \geq 0, \quad i = 1,\dots,n.$$

(hyper) Paramètre C

- compromis marge / erreur d'apprentissage
- à fixer / optimiser par l'utilisateur
- importance critique

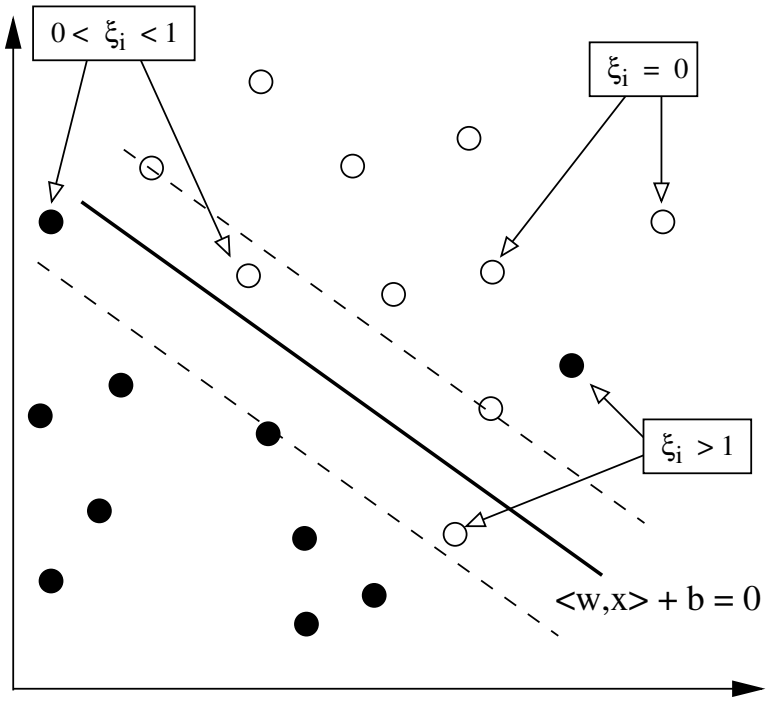
$\Rightarrow$ hyperparamètre de régularisation

Illustration - slack-variables



- $\xi_i = 0$: point (x_i, y_i) bien classé en dehors de la marge
- $0 < \xi_i < 1$: point (x_i, y_i) bien classé mais dans la marge
- $\xi_i > 1$: point (x_i, y_i) mal classé

Illustration - slack-variables



- $\xi_i = 0$: point (x_i, y_i) bien classé en dehors de la marge
- $0 < \xi_i < 1$: point (x_i, y_i) bien classé mais dans la marge
- $\xi_i > 1$: point (x_i, y_i) mal classé

$$\Rightarrow \sum_{i=1}^{n} \xi_i > \#\text{ d'erreurs de classification}$$

SVM linéaire & non-séparable - résolution

Soft-margin SVM : problème primal

$$
\begin{aligned}
(w^*, b^*) = \underset{\substack{w\in\mathbb{R}^p, b\in\mathbb{R}, \\ \xi\in\mathbb{R}^n}}{\operatorname{argmin}} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i \\
\text{s.t.} \quad & y_i(\langle w, x_i\rangle + b) \geq 1-\xi_i, \quad i = 1,\dots,n, \\
& \xi_i \geq 0, \qquad\qquad\qquad\quad i = 1,\dots,n.
\end{aligned}
$$

SVM linéaire & non-séparable - résolution

Soft-margin SVM : problème primal

$$
\begin{aligned}
(w^*, b^*) = \underset{\substack{w\in\mathbb{R}^p, b\in\mathbb{R},\\ \xi\in\mathbb{R}^n}}{\operatorname{argmin}} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i \\
\text{s.t.} \quad & y_i(\langle w, x_i\rangle + b) \geq 1-\xi_i, \quad i = 1,\dots,n, \\
& \xi_i \geq 0, \qquad\qquad\qquad\quad i = 1,\dots,n.
\end{aligned}
$$

$\Rightarrow$ Formulation Lagrangienne :

$$
\begin{aligned}
L(w, b, \xi, \alpha, \nu) = & \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i \\
& - \sum_{i=1}^{n}\alpha_i\big(y_i(\langle w, x_i\rangle + b) - 1 + \xi_i\big) - \sum_{i=1}^{n}\nu_i\xi_i,
\end{aligned}
$$

avec $\alpha_i \geq 0$ et $\nu_i \geq 0$ pour $i = 1, ..., n$.

SVM linéaire & non-séparable - résolution

Même procédure pour résoudre le problème :

- minimiser $L(w, b, \xi, \alpha, \nu)$ par rapport à w, b et ξ.
- maximiser $L(w, b, \xi, \alpha, \nu)$ par rapport à α, ν.

SVM linéaire & non-séparable - résolution

Même procédure pour résoudre le problème :

- minimiser $L(w, b, \xi, \alpha, \nu)$ par rapport à w, b et ξ.
- maximiser $L(w, b, \xi, \alpha, \nu)$ par rapport à α, ν.

Soft-margin SVM : problème dual

$$\alpha^* = \underset{\alpha \in \mathbb{R}^n}{\operatorname{argmax}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n$$

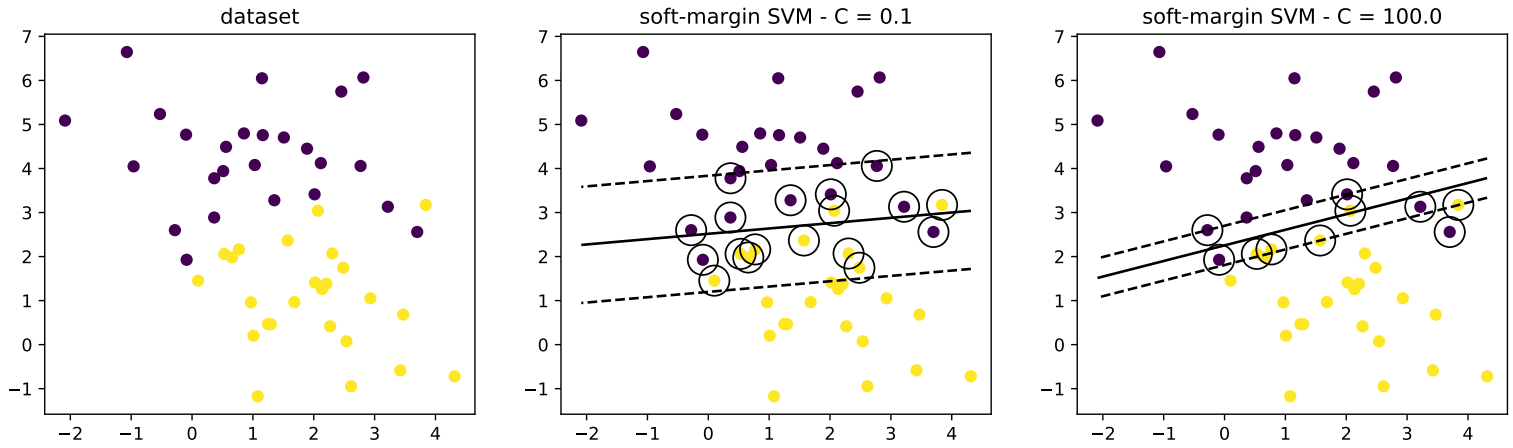
$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$

$\Rightarrow$ soft-margin = re-définition des contraintes sur les α_i.

$\Rightarrow$ pas d'impact sur l'algorithme d'optimisation.

Illustration - paramètre C

Illustration : impact du paramètre C



$\Rightarrow$ C faible :

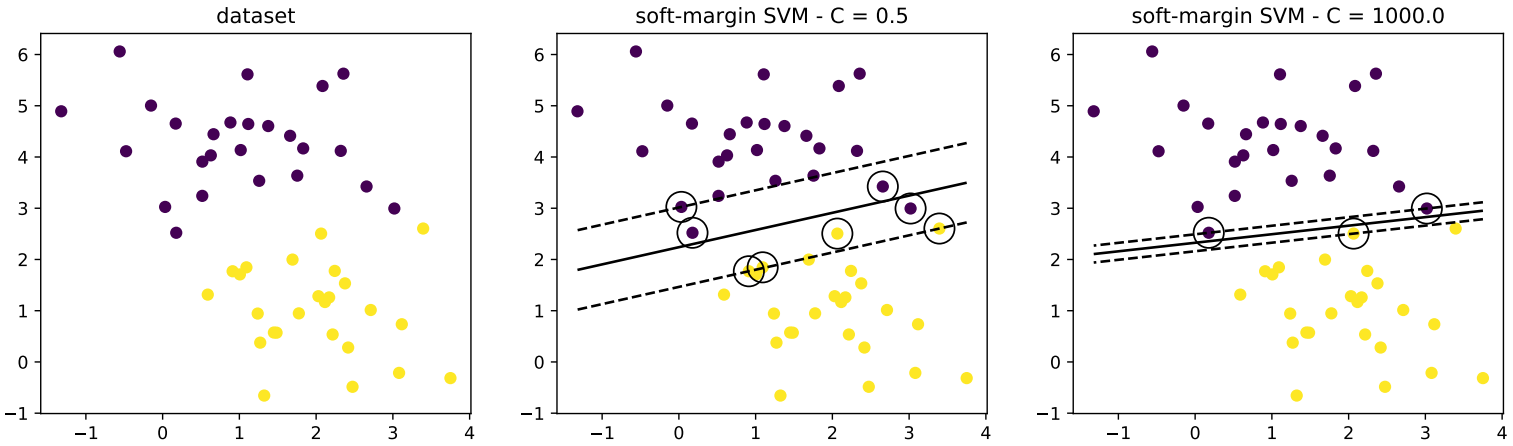
- marge importante
- beaucoup de points dans la marge / mal classés

$\Rightarrow$ C élevé :

- peu de points dans la marge / mal classés
- marge faible

Illustration - paramètre C

Illustration : SVM soft-margin & données séparables



$\Rightarrow$ même si séparable, SVM "hard-margin" pas forcément la meilleure solution !

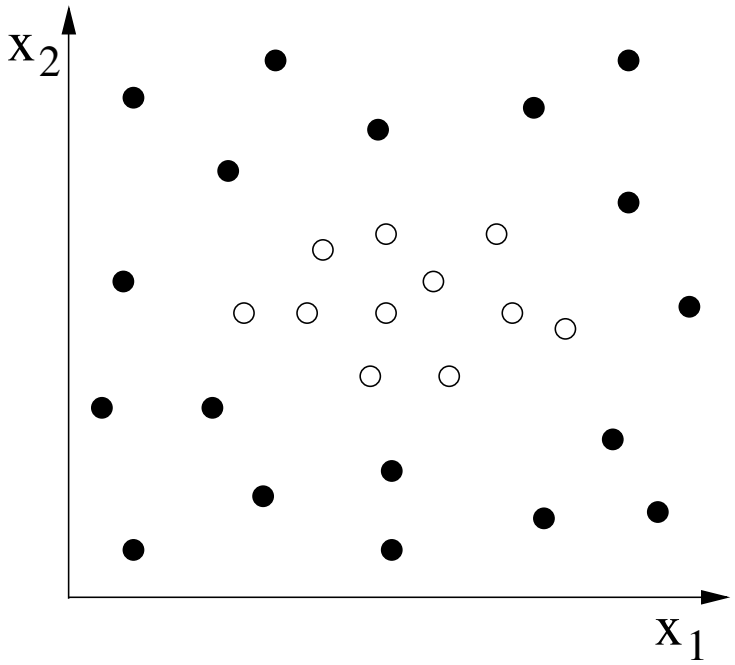
$\Rightarrow$ soft-margin = robustesse aux bruit / points aberrants

$\Rightarrow$ toujours optimiser le paramètre C

SVMs non-linéaires et noyaux

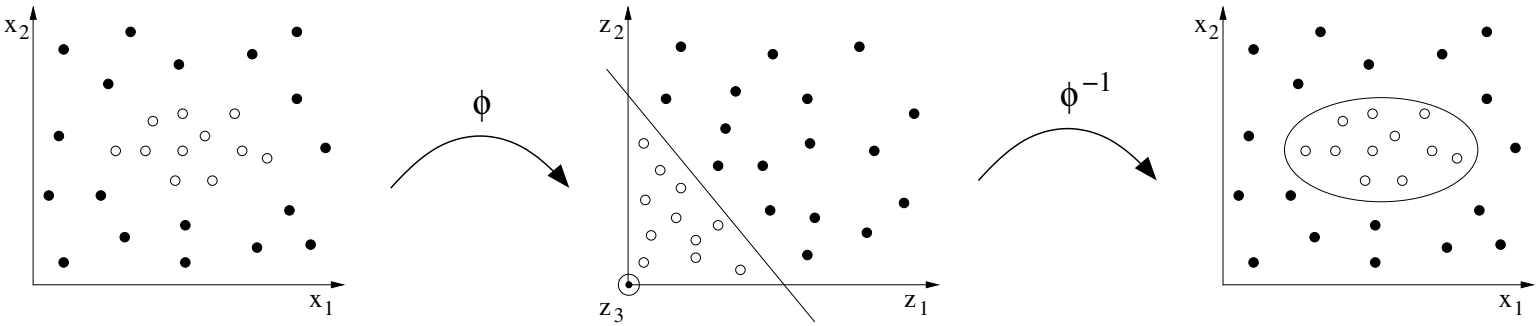
Motivation...

Modèle linéaire : parfois fondamentalement inadapté



SVM non-linéaire

Comment faire ?

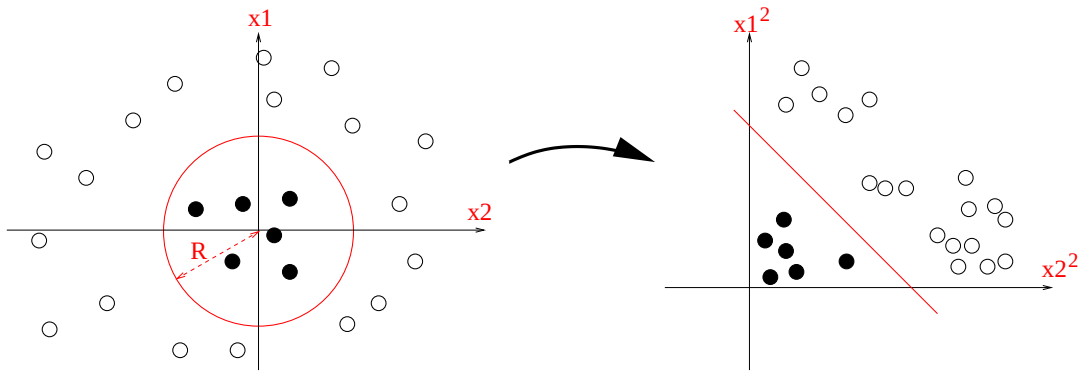


1. considérer une transformation non linéaire $\phi : \mathcal{X} \to \mathcal{F}$
   - $\mathcal{X}$ = input space (e.g., $\mathbb{R}^p$)
   - $\mathcal{F}$ = feature space (e.g., $\mathbb{R}^d$, $d >> p$)
2. apprendre une SVM linéaire dans $\mathcal{F}$

$\Rightarrow$ le modèle obtenu par ϕ^{-1} est non-linéaire dans $\mathcal{X}$.

Exemple

Exemple : projection polynomiale



Pour $x = (x_1, x_2) \in \mathbb{R}^2 \Rightarrow \Phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \in \mathbb{R}^3$

- expansion polynomiale d'ordre 2

$\Rightarrow$ un cercle / une ellipse dans $\mathcal{X}$ = un hyperplan dans $\mathcal{F}$

- e.g., equation du cercle : $x_1^2 + x_2^2 = R^2$

SVM non-linéaire : implémentation

1. Résoudre le problème dual dans $\mathcal{F}$:

$$\alpha^* = \underset{\alpha \in \mathbb{R}^n}{\operatorname{argmax}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \Phi(x_i), \Phi(x_j) \rangle$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$

SVM non-linéaire : implémentation

1. Résoudre le problème dual dans $\mathcal{F}$:

$$\alpha^* = \underset{\alpha \in \mathbb{R}^n}{\operatorname{argmax}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \Phi(x_i), \Phi(x_j) \rangle$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$

2. La fonction de décision est alors donnée par (le signe de) :

$$f(x) = \sum_{i=1}^{n} \alpha_i^* y_i \langle \Phi(x_i), \Phi(x) \rangle + b^*$$

L'astuce noyau - "the kernel trick"

Remarque clé :

les données n'apparaissent que via des opérations de produit-scalaire $\langle \Phi(.), \Phi(.) \rangle$

L'astuce noyau - "the kernel trick"

Remarque clé :

> les données n'apparaissent que via des opérations de produit-scalaire $\langle \Phi(.), \Phi(.) \rangle$

$\Rightarrow$ le kernel-trick :

- ne pas calculer Φ explicitement
- calculer directement le noyau $K(.,.) = \langle \Phi(.), \Phi(.) \rangle$

L'astuce noyau - "the kernel trick"

Remarque clé :

> les données n'apparaissent que via des opérations de produit-scalaire $\langle \Phi(.), \Phi(.) \rangle$

$\Rightarrow$ le kernel-trick :

- ne pas calculer Φ explicitement
- calculer directement le noyau $K(.,.) = \langle \Phi(.), \Phi(.) \rangle$

Conséquences :

- parfois plus simple / efficace de calculer le noyau
- permet de considérer des projections non-explicites
- extension aux données structurées

SVM "kernelisée" : implémentation

1. Résoudre le problème dual dans $\mathcal{F}$:

$$\alpha^* = \underset{\alpha \in \mathbb{R}^n}{\operatorname{argmax}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$

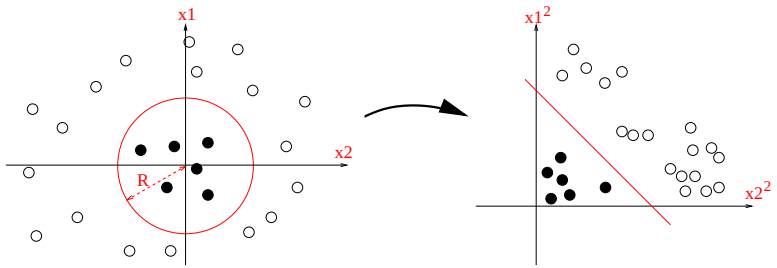
2. La fonction de décision est alors donnée par (le signe de) :

$$f(x) = \sum_{i=1}^{n} \alpha_i^* y_i K(x_i, x) + b^*$$

$\Rightarrow$ le produit scalaire original $\langle x_i, x_j \rangle$ est le noyau linéaire

Noyau polynomial

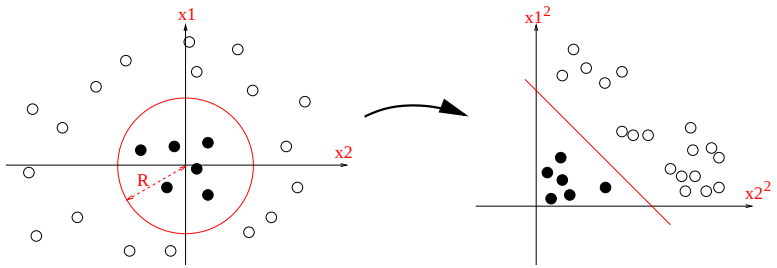
Revenons à la projection polynomiale d'ordre 2 :



pour $x = (x_1, x_2) \in \mathbb{R}^2 \Rightarrow \Phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \in \mathbb{R}^3$.

Noyau polynomial

Revenons à la projection polynomiale d'ordre 2 :



pour $x = (x_1, x_2) \in \mathbb{R}^2 \Rightarrow \Phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \in \mathbb{R}^3$.

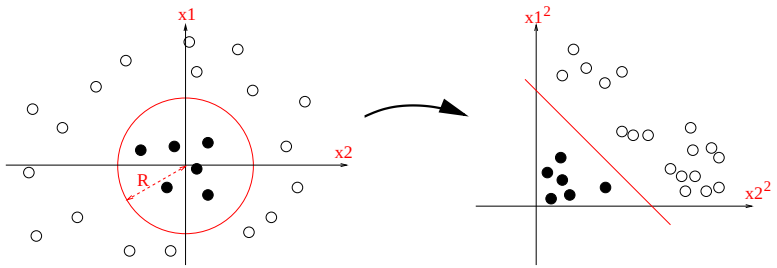
Produit scalaire : $\langle \Phi(x), \Phi(x') \rangle = x_1^2 x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2 x_2'^2$

$$= \left(x_1x_1' + x_2x_2'\right)^2$$

$$= \langle x, x' \rangle^2$$

Noyau polynomial

Revenons à la projection polynomiale d'ordre 2 :



pour $x = (x_1, x_2) \in \mathbb{R}^2 \Rightarrow \Phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \in \mathbb{R}^3$.

Produit scalaire : $\langle \Phi(x), \Phi(x') \rangle = x_1^2 x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2 x_2'^2$

$$= \left(x_1x_1' + x_2x_2'\right)^2$$

$$= \langle x, x' \rangle^2$$

$\Rightarrow$ Noyau polynomial : $\boxed{K(x, x') = \left(\gamma \langle x, x' \rangle + r\right)^d}$

- plus simple et efficace à calculer
- (gain exponentiel quand p et/ou d augmentent)

Fonctions noyaux

A chaque fonction de projection $\Phi : \mathcal{X} \to \mathcal{F}$ correspond une fonction noyau $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$:

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle$$

Fonctions noyaux

A chaque fonction de projection $\Phi : \mathcal{X} \to \mathcal{F}$ correspond une fonction noyau $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$:

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle$$

Inversement : à chaque fonction symmétrique et définie positive $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ correspond une projection $\phi : \mathcal{X} \to \mathcal{F}$:

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle$$

Fonctions noyaux

A chaque fonction de projection $\Phi : \mathcal{X} \to \mathcal{F}$ correspond une fonction noyau $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$:

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle$$

Inversement : à chaque fonction symmétrique et définie positive $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ correspond une projection $\phi : \mathcal{X} \to \mathcal{F}$:

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle$$

$\Rightarrow$ toute fonction définie positive peut être utilisée comme noyau pour les SVMs :

- sans forcément connaître explicitement Φ (ni $\mathcal{F}$)
- sans forcément savoir la calculer

Noyaux définis positifs

Définition :

- pour tout $n \in \mathbb{N}$, $x_1, ..., x_n \in \mathcal{X}$ and $\alpha_1, ..., \alpha_n \in \mathbb{R}$:

$$\sum_i \sum_j \alpha_i \alpha_j K(x_i, x_j) \geq 0$$

- pour tout $n \in \mathbb{N}$, and $x_1, ..., x_n \in \mathcal{X}$ la matrice de Gram :

$$\mathbf{G}[i, j] = K(x_i, x_j)$$

est définie positive (toutes ses valeurs propres sont ≥ 0)

Noyaux définis positifs

Définition :

- pour tout $n \in \mathbb{N}$, $x_1, ..., x_n \in \mathcal{X}$ and $\alpha_1, ..., \alpha_n \in \mathbb{R}$:

$$\sum_i \sum_j \alpha_i \alpha_j K(x_i, x_j) \geq 0$$

- pour tout $n \in \mathbb{N}$, and $x_1, ..., x_n \in \mathcal{X}$ la matrice de Gram :

$$\mathbf{G}[i, j] = K(x_i, x_j)$$

est définie positive (toutes ses valeurs propres sont ≥ 0)

Illustration : le produit scalaire est défini-positif :

$$\sum_i \sum_j \alpha_i \alpha_j \langle \phi(x_i), \phi(x_j) \rangle = \langle \sum_i \alpha_i \phi(x_i), \sum_j \alpha_j \phi(x_j) \rangle$$
$$= \left\| \sum_i \alpha_i \phi(x_i) \right\|^2 \geq 0.$$

Fonctions noyaux & mesures de similarité

Un noyau peut être vu comme une mesure de similarité

- analogie avec le produit-scalaire / la corrélation

$\Rightarrow$ il est parfois plus naturel de raisonner en terme de similarité qu'en terme de représentation en descripteurs.

Fonctions noyaux & mesures de similarité

Un noyau peut être vu comme une mesure de similarité

- analogie avec le produit-scalaire / la corrélation

$\Rightarrow$ il est parfois plus naturel de raisonner en terme de similarité qu'en terme de représentation en descripteurs.

Exemple : le noyau RBF (Radial Basis Function)

$$\boxed{K(x, x') = \exp\left(-\gamma \| x - x' \|^2\right)}$$

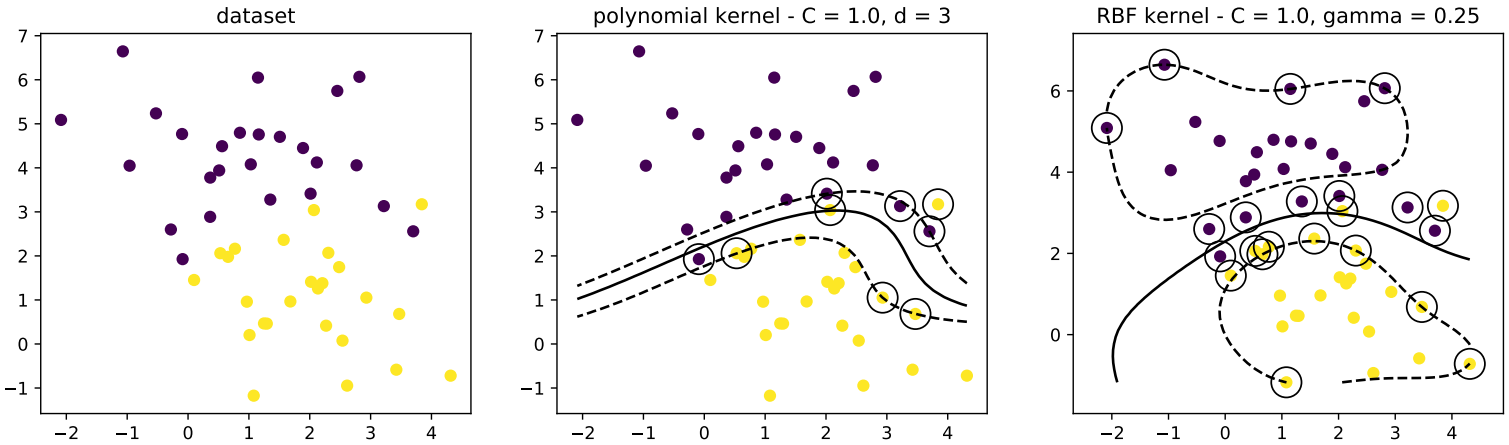
est un noyau valide entre vecteurs x et x'.

- i.e., il est symmétrique et défini-positif

$\Rightarrow$ la dimension du feature space correspondant est infinie
$\Rightarrow$ on sait expliciter Φ mais pas le calculer

Illustration

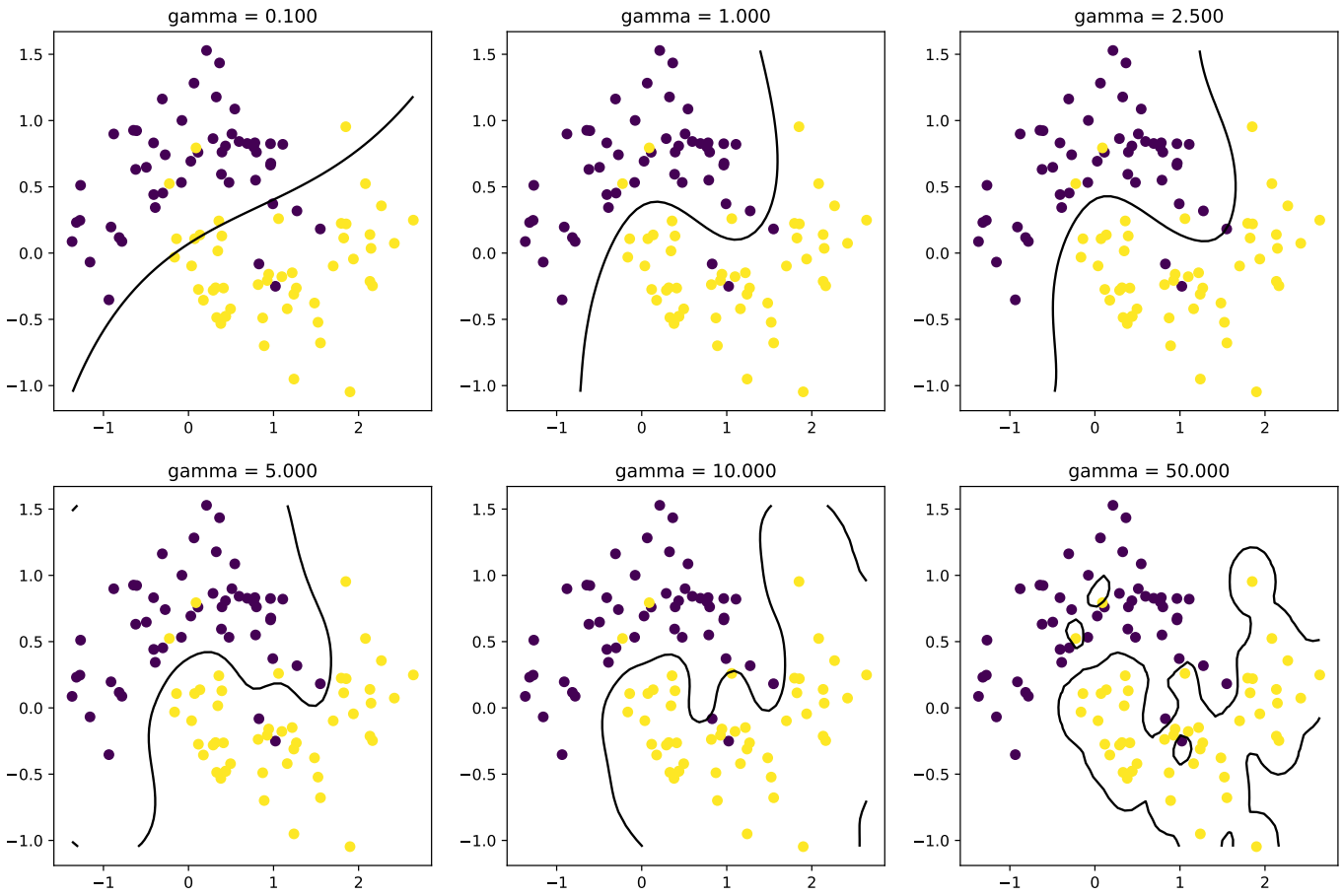
Illustration : noyau polynomial vs noyau RBF



- polynomial : linéaire après expansion polynomiale
- RBF : une Gaussienne sur chaque point d'apprentissage :

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b = \sum_{i=1}^{n} \alpha_i y_i \exp\left(-\gamma \| x_i - x \|^2\right) + b$$

Illustration - noyau RBF : influence de γ



Largeur de bande $\gamma \sim 1/\sigma^2$ pour une gaussienne

- γ élevé : modèle local / γ faible : modèle global

SVMs en pratique

SVM en pratique

SVM "stricto sensu" : algorithme de classification binaire

Quelques extensions / adaptations :

- régression
- classification multiclasse
- jeux de données déséquilibrés
- critère de confiance dans la prédiction
- mise en oeuvre sur de gands jeux de données

SVM pour la régression

Extension aux problèmes de régression :

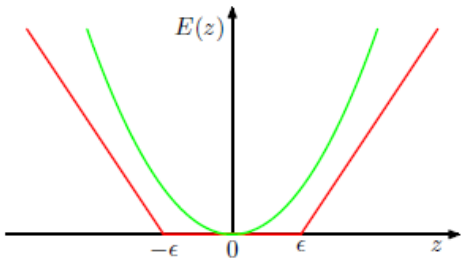
1. remplacer les contraintes :

$$y_i\big(\langle w, x_i\rangle + b\big) \geq 1 - \xi_i \quad \Rightarrow \quad \boxed{|y_i - \big(\langle w, x_i\rangle + b\big)| \leq \epsilon + \xi_i}$$

2. garder le même algorithme d'optimisation :

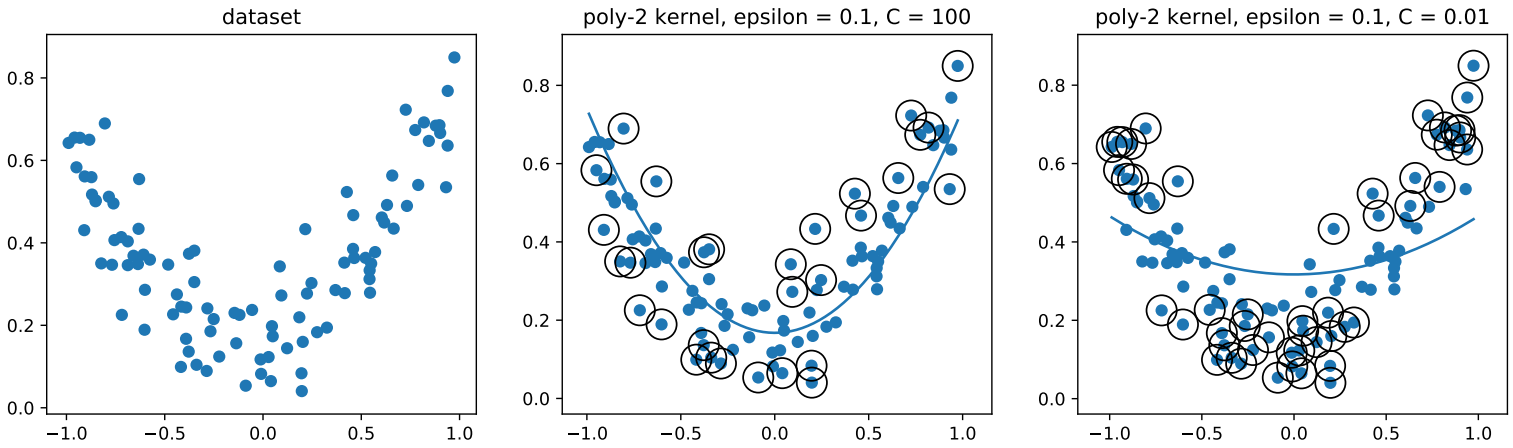
$$(w^*, b^*) = \underset{\substack{w\in\mathbb{R}^p, b\in\mathbb{R}, \\ \xi\in\mathbb{R}^n}}{\operatorname{argmin}} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i \text{ sous les contraintes}$$

$\Rightarrow$ ϵ-insensitive loss function :



SVM pour la régression

Illustration : exemple tiré de Géron (2017)



- avec noyaux : méthode de régression non-linéaire
  - ici, noyau polynomial
- deux paramètres à optimiser : ϵ et C
  - (plus ceux du noyau)

SVM pour la classification multiclasse

Classification multi-classe : $y_i \in \{1, ..., K\}$

En général : par combinaison de classifieurs binaires

- stratégie One-vs-All ou One-vs-Rest
  - construire K classifieurs
  - classifieur $\#k$: classe k contre tout le reste
  - prédiction : classe donnant le score $f_k(x)$ le plus élevé
- stratégie One-vs-One
  - construire $K(K-1)/2$ classifieurs
  - chaque classifieur compare deux classes
  - prédiction : classe obtenant le plus de votes
    - un nombre entre 0 et $K-1$

Remarque : autre statégies "nativement" multiclasses

- proche du formalisme des SVMs "structurées"

SVM & jeux de données déséquilibrés

Fonction objective des SVMs : $\boxed{\frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i}$

- "compte" globalement le nombre d'erreurs
- sensible au déséquilibre des classes

SVM & jeux de données déséquilibrés

Fonction objective des SVMs : $\boxed{\frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i}$

- "compte" globalement le nombre d'erreurs
- sensible au déséquilibre des classes

SVM à coûts asymétriques :

$$\boxed{\frac{1}{2}||w||^2 + C_+ \sum_{i:y_i=+1}\xi_i + C_- \sum_{j:y_j=-1}\xi_j}$$

- C_+ : coût des erreurs de la classe +1
- C_- : coût des erreurs de la classe -1

$\Rightarrow \nearrow$ le coût de la classe minoritaire compense le déséquilibre

SVM & critère de confiance dans la prédiction

La fonction $\boxed{f(x) = \langle w, x \rangle + b} \propto$ la distance à l'hyperplan

Elle permet notamment :

1. de définir une courbe ROC :
   - seuil de décision par défaut = 0 (le signe de $f(x)$)
   - ajuster le seuil $\leftrightarrow$ modifier b $\leftrightarrow$ **translater l'hyperplan**
2. de définir un critère de confiance :
   - valeur élevée : loin de la frontière $\Rightarrow$ bonne confiance
   - valeur faible : proche de la frontière $\Rightarrow$ confiance faible

$\Rightarrow$ le score $f(x)$ peut être transformé en critère probabiliste

- typiquement : par une **fonction sigmoïdale**
- paramètres à optimiser sur le jeu d'apprentissage

SVM pour les grands jeux de données

SVM : 2 problèmes d'optimisation équivalents

- problème primal : défini selon $(w, b) \in \mathbb{R}^p \times \mathbb{R}$
- problème dual : défini selon $\alpha \in \mathbb{R}^n$

liés par la relation $w = \sum_{i=1}^{n} \alpha_i y_i x_i$.

SVM pour les grands jeux de données

SVM : 2 problèmes d'optimisation équivalents

- problème primal : défini selon $(w, b) \in \mathbb{R}^p \times \mathbb{R}$
- problème dual : défini selon $\alpha \in \mathbb{R}^n$

liés par la relation $w = \sum_{i=1}^{n} \alpha_i y_i x_i$.

Pour utiliser des noyaux : travailler dans le dual

- librairies "dual" classiques : `LibSVM`, `SVMlight`

$\Rightarrow$ coût calculatoire : limitant quand beaucoup d'instances

SVM pour les grands jeux de données

SVM : 2 problèmes d'optimisation équivalents

- problème primal : défini selon $(w, b) \in \mathbb{R}^p \times \mathbb{R}$
- problème dual : défini selon $\alpha \in \mathbb{R}^n$

liés par la relation $w = \sum_{i=1}^{n} \alpha_i y_i x_i$.

Pour utiliser des noyaux : travailler dans le dual

- librairies "dual" classiques : `LibSVM`, `SVMlight`

$\Rightarrow$ coût calculatoire : limitant quand beaucoup d'instances

Pour un modèle linéaire : travailler dans le primal ou le dual

- choix dicté par $\min(n, p)$
- librairie "primal" classique : `LibLinear`

$\Rightarrow$ souvent la seule alternative pour les gros jeux de données
$\Rightarrow$ linéaire : souvent suffisant quand beaucoup de features

Conclusion

Conclusion

SVM : un algorithme incontournable

- solides bases théoriques
- très performant sur de nombreux problèmes

Conclusion

SVM : un algorithme incontournable

- solides bases théoriques
- très performant sur de nombreux problèmes

Différentes versions :

- hard-margin : données strictement séparables/ées
- soft-margin : tolère des erreurs de classification
- non-linéaire avec l'utilisation de noyaux

Conclusion

SVM : un algorithme incontournable

- solides bases théoriques
- très performant sur de nombreux problèmes

Différentes versions :

- hard-margin : données strictement séparables/ées
- soft-margin : tolère des erreurs de classification
- non-linéaire avec l'utilisation de noyaux

Soft-margin SVM :

- paramètre C : paramètre clé (régularisation)
- à optimiser par validation croisée
- considérer une grille logarithmique
  - e.g., $10^{-3}, 10^{-2}, ..., 10^{2}, 10^{3}$

Conclusion

Fonctions noyaux :

- extension non-linéaire des SVMs
- "kernel trick" : projection implicite
- noyau : fonction symmetrique et définie positive
- exemples incontournables : RBF et polynomial
- extensions aux données structurées

Conclusion

Fonctions noyaux :

- extension non-linéaire des SVMs
- "kernel trick" : projection implicite
- noyau : fonction symmetrique et définie positive
- exemples incontournables : RBF et polynomial
- extensions aux données structurées

En pratique :

- extension pour la régression
- adaptation au cadre multiclasse (OVR,OVO)
- implémentation primale pour gros volumes de données
  - restriction au cadre linéaire

SVMs & risque empirique pénalisé

SVMs & risque empirique pénalisé

SVMs : une formulation de risque empirique pénalisé :

$$(w^*, b^*) = \underset{w \in \mathbb{R}^p, b \in \mathbb{R}}{\operatorname{argmin}} \quad \sum_{i=1}^{n} h\big(y_i, f(x_i)\big) + \lambda ||w||^2$$

où $f(x) = \langle w, x \rangle + b$.

- risque empirique : erreur faite sur le jeu d'apprentissage
  - selon la fonction de coût $h\big(y, f(x)\big)$
- penalisation : pour contrôler le sur-apprentissage
  - risque empirique seul = risque de sur-apprentissage
    - e.g., haute dimension ou classes de fonctions complexes
  - pénalisation = régularisation
  - pour les SVMs : liée directement à la notion de marge

Fonctions de perte classiques

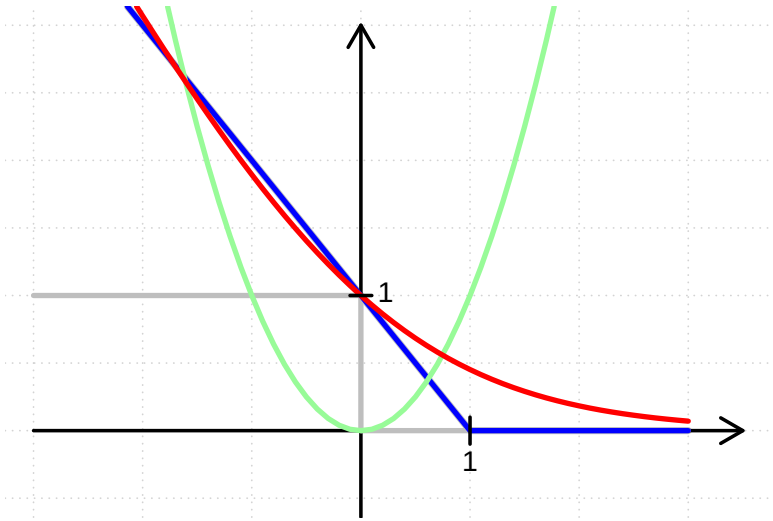
Régression :

- perte quadratique : $L\big(y, f(x)\big) = (y - f(x))^2$

Classification :

- perte hinge : $L\big(y, f(x)\big) = \big(1 - yf(x)\big)_+$
- perte logistique : $L\big(y, f(x)\big) = \log\left(1 + e^{-yf(x)}\right)$

(la quantité $yf(x)$ est parfois appelée la **marge** de (x, y))



Fonctions de pénalisation incontournables

Pénalité Ridge (ou L_2) : $\boxed{\Omega_{\text{Ridge}}(w) = ||w||_2^2 = \sum_{j=1}^{p} w_j^2}$

Pénalité Lasso (ou L_1) : $\boxed{\Omega_{\text{Lasso}}(w) = ||w||_1 = \sum_{j=1}^{p} |w_j|}$

Fonctions de pénalisation incontournables

Pénalité Ridge (ou L_2) : $\boxed{\Omega_{\text{Ridge}}(w) = ||w||_2^2 = \sum_{j=1}^{p} w_j^2}$

Pénalité Lasso (ou L_1) : $\boxed{\Omega_{\text{Lasso}}(w) = ||w||_1 = \sum_{j=1}^{p} |w_j|}$

Même effet : pénaliser les valeur élevées $\Rightarrow$ régularisation

- intuition : pente élevée $\rightarrow$ y varie + vite quand x varie

mais avec une géométrie différente :

- **lasso** : des coefficients w_j **exactement** $= 0$
- **ridge** : coefficients w_j petits mais **jamais nuls**

$\Rightarrow$ Lasso = méthode parcimonieuse, sélection de variables

Pertes & régularisation

Combinaisons pertes / pénalités :

	Quadratique	Hinge	Logistique
Ridge	ridge regression	SVM	ridge logistic-regression
Lasso	Lasso	L1-SVM	L1 logistic-regression

$\Rightarrow$ à suivre dans le cours statistique en haute dimension !

Mise en oeuvre Scikit-Learn

Classes Scikit-Learn

4 classes du module `svm` :

- `SVC` et `SVR` : classification et régression dans le dual
- `LinearSVC` et `LinearSVC` : idem dans le primal

Implémentation duale : basée sur `LibSVM`

- `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`

Implémentation primale : basée sur `LibLinear`

- `https://www.csie.ntu.edu.tw/~cjlin/liblinear/`

Autres classes / variantes non-couvertes dans ce cours :

- "nu"-SVC/SVR : autre manière de paramétrer les SVMs
- SVM "One Class"

Classe SVC (1/2)

Paramètres principaux :

- `C`
- `kernel` : 'linear', 'poly', 'rbf'
  - + paramètres associés : `gamma, degree, coef0`
  - également 'precomputed'
- `probability` (booléen) : transformation score $\rightarrow$ proba
- `class_weight` : coûts asymétriques
  - valeurs à utiliser ou 'balanced'
- `decision_function_shape` : 'ovo', 'ovr' (mutliclasse)

Classe SVC (1/2)

Paramètres principaux :

- `C`
- `kernel` : 'linear', 'poly', 'rbf'
  - + paramètres associés : `gamma, degree, coef0`
  - également 'precomputed'
- `probability` (booléen) : transformation score $\rightarrow$ proba
- `class_weight` : coûts asymétriques
  - valeurs à utiliser ou 'balanced'
- `decision_function_shape` : 'ovo', 'ovr' (mutliclasse)

Méthodes principales :

- `fit`
- `predict` : prédiction $= \text{sign}(\langle w, x \rangle + b)$
- `decision_function` : $\langle w, x \rangle + b$
- `predict_proba` : probabilité associée à la prédiction

Classe SVC (2/2)

Attributs principaux (calculés par l'appel à `fit`) :

- `support_` : indices des vecteurs support
- `support_vectors_` : vecteurs support
- `dual_coef_` : coefficients α_i (restreints aux VS)
- `coef_` : vecteur w (si `kernel = 'linear'`)
- `intercept_` : coefficient b.

Classe SVC (2/2)

Attributs principaux (calculés par l'appel à `fit`) :

- `support_` : indices des vecteurs support
- `support_vectors_` : vecteurs support
- `dual_coef_` : coefficients α_i (restreints aux VS)
- `coef_` : vecteur w (si `kernel = 'linear'`)
- `intercept_` : coefficient b.

Classe SVR : fonctionnement similaire

- Paramètre supplémentaire : `epsilon`
  - ϵ-insensitive loss-function
- Méthodes : essentiellement `fit` et `predict`.
- Attributs : les mêmes.

Classes `LinearSVC` et `LinearSVR`

Paramètres / méthode / attributs similaires à `SVC`.

Néanmoins :

- pas de paramètres liés aux noyaux (que linéaire)
- + de fléxibilité pour contrôler perte & régularisation
  - `penalty` : 'l1' ou 'l2'
  - `loss` : 'hinge' ou 'squared_hinge'
- paramètre `dual` (booléen) : implémentation duale ?
  - attention, `True` par défaut !
- option `decision_function_shape` → `multiclass`
  - 'ovr' ou 'crammer_singer'
- plus d'attributs liés au mode dual (coefficients α_i et VS)
- plus d'option `probability`

Classe `LinearSVR` : idem (avec option `epsilon`)

Définition / paramétrisation des noyaux

Noyaux disponibles pour `SVC/SVR` (option `kernel`) :

- `linear` : $K(x, x') = \langle x, x' \rangle$
- `poly` : $K(x, x') = \left(\gamma \langle x, x' \rangle + r\right)^d$
- `rbf` : $K(x, x') = \exp\left(-\gamma ||x - x'||^2\right)$
- `sigmoid` : $K(x, x') = \tanh\left(\gamma \langle x, x' \rangle + r\right)$

$\Rightarrow \gamma =$ argument `gamma` (par défaut $= 1/p$)
$\Rightarrow r =$ argument `coef0` (par défaut $= 0$)
$\Rightarrow d =$ argument `degree` (par défaut $= 3$)

Par défaut : `kernel` = 'rbf'

Références

A. Géron. *Hands-On Machine Learning with Scikit-Learn & TensorFlow*. O'Reilly, 2017.