Support Vector Machines

Machine Learning

Cédric RICHARD
Université Nice Sophia Antipolis

Principes MRE et MRS

Principe MRE. Il s'agit de minimiser la fonctionnelle de risque

$$P_e(d) = \int \frac{1}{2}|y - d(\boldsymbol{x}; \boldsymbol{w}, b)|\, p(\boldsymbol{x}, y)\, d\boldsymbol{x}\, dy.$$

La densité $p(\boldsymbol{x}, y)$ étant inconnue, la minimisation de $P_e(d)$ se traduit par celle du risque empirique

$$P_{emp}(d) = \frac{1}{2n} \sum_{i=1}^{n} |y_i - d(\boldsymbol{x}_i; \boldsymbol{w}, b)|$$

calculable sur les données constituant l'ensemble d'apprentissage $\mathcal{A}_n$.

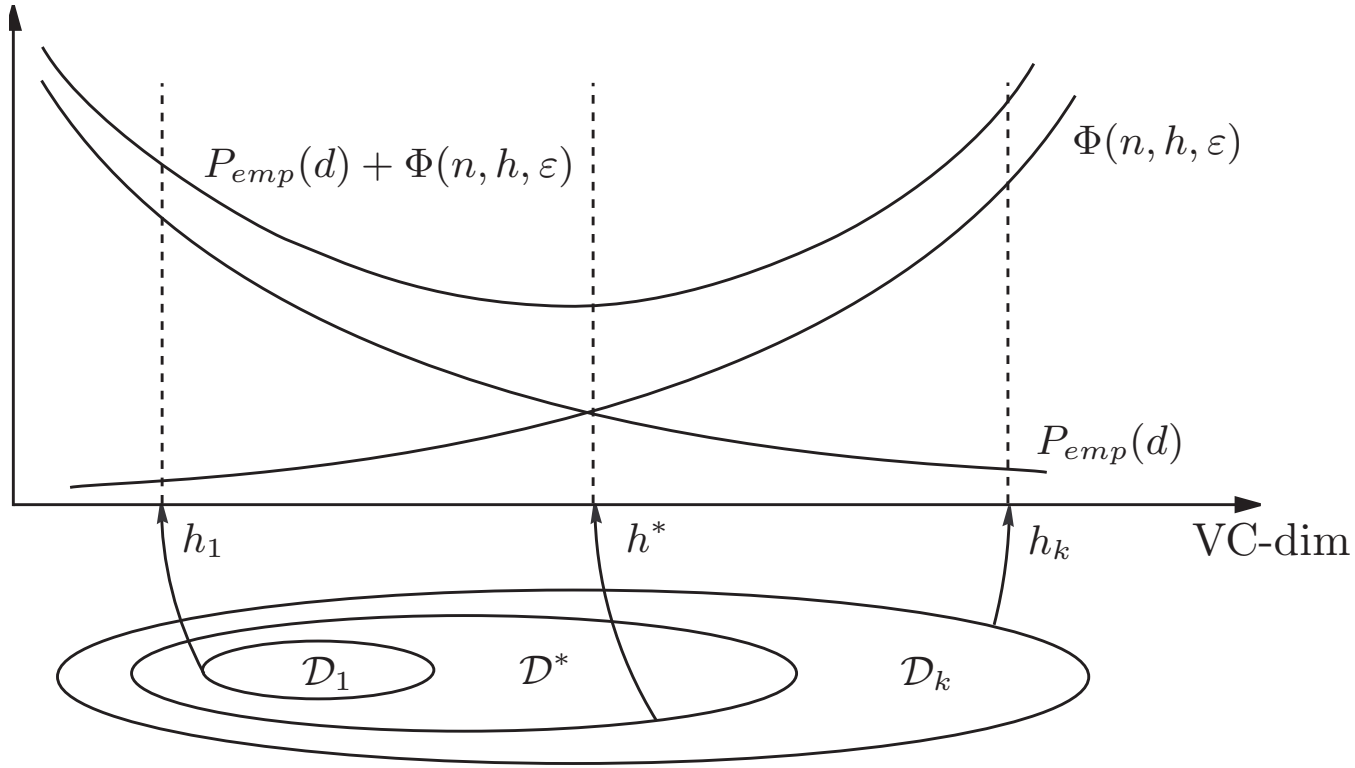
Principe MRS. Avec une probabilité $1 - \varepsilon$, on a :

$$P_e(d) \leq P_{emp}(d) + \sqrt{\frac{h\, \ln\left(\frac{2n}{h} + 1\right) - \ln\frac{\varepsilon}{4}}{n}}.$$

Le principe MRS suggère de minimiser par rapport à h le risque garanti défini par la borne supérieure $P_{emp}(d) + \Phi(n, h, \varepsilon)$, plutôt que le risque empirique.

Le principe MRS

Illustration

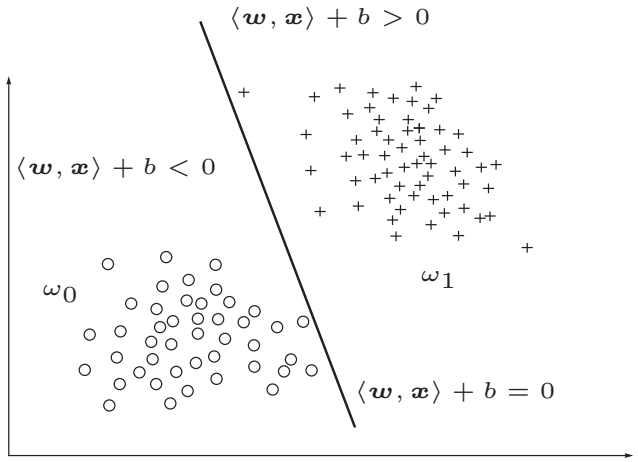


L'algorithme du Perceptron

L'algorithme du Perceptron vise à produire une solution d'erreur d'apprentissage minimum en minimisant le risque empirique suivant :

$$(\boldsymbol{w}^*, b^*) = \arg\min_{(\boldsymbol{w},b)} \sum_{i=1}^{n} |y_i - d(\boldsymbol{x}_i; \boldsymbol{w}, b)|.$$

- ▷ Pourquoi la solution obtenue aurait-elle les meilleures performances ?
- ▷ Est-ce que la minimisation de l'erreur empirique est une bonne idée ?
- ▷ Existe-t-il une alternative ?



Classifieurs linéaires

Définition

On considère un problème de classification à 2 classes de n données dans $\mathbb{R}^l$, étant donné un ensemble d'apprentissage

$$\mathcal{A}_n = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}.$$

On pose $y_i = (-1)$ si $\boldsymbol{x}_i \in \omega_0$, et $y_i = (+1)$ si $\boldsymbol{x}_i \in \omega_1$.

Un **classifieur linéaire** est défini par

$$d(\boldsymbol{x}; \boldsymbol{w}, b) = \text{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b).$$

Séparabilité linéaire

Définition

On considère un problème de classification à 2 classes de n données dans $\mathbb{R}^l$, étant donné un ensemble d'apprentissage

$$\mathcal{A}_n = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}.$$

On pose $y_i = (-1)$ si $\boldsymbol{x}_i \in \omega_0$, et $y_i = (+1)$ si $\boldsymbol{x}_i \in \omega_1$.

L'équation d'un hyperplan est définie à une constante multiplicative près :

$$\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0 \quad \Longleftrightarrow \quad \langle \gamma\, \boldsymbol{w}, \boldsymbol{x} \rangle + \gamma\, b = 0, \quad \gamma \in \mathbb{R}^*$$

Les classes ω_0 et ω_1 sont dites **linéairement séparables** s'il existe $\boldsymbol{w}$ et b tels que

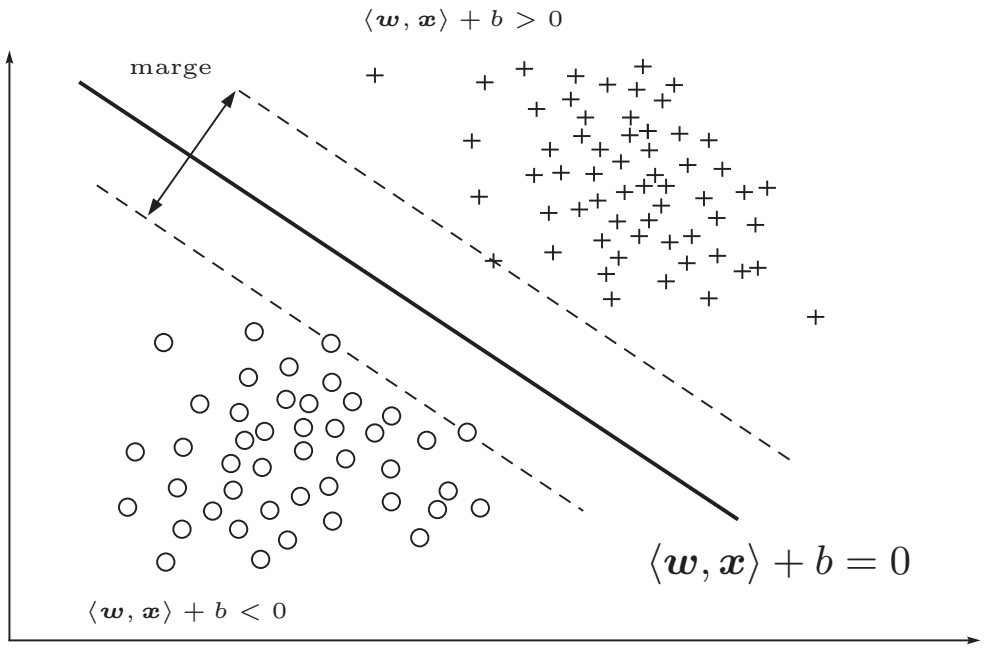
$$\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b \geq +1 \qquad \forall \boldsymbol{x}_i \in \omega_1$$
$$\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b \leq -1 \qquad \forall \boldsymbol{x}_i \in \omega_0$$

Dans la suite, on résume ce critère de séparabilité ainsi

$$y_i \left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b \right) - 1 \geq 0 \qquad \forall (\boldsymbol{x}_i, y_i) \in \mathcal{A}_n$$

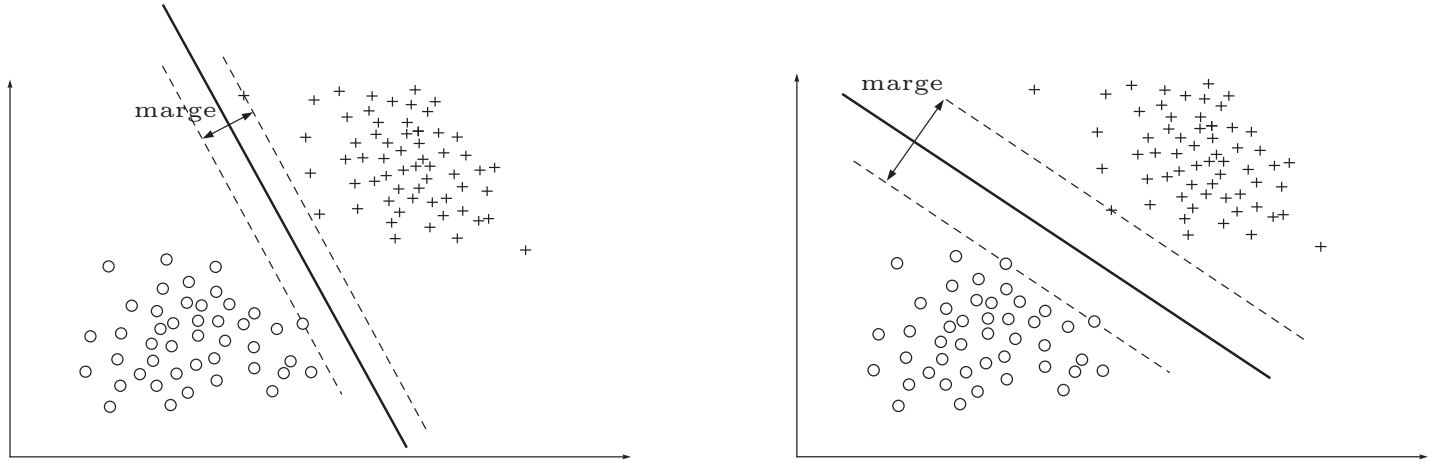
Un nouveau principe d'induction

Parmi les séparateurs conduisant à une erreur empirique minimum, il convient de **choisir celui de marge maximum** (Vapnik 1965, 1992).



Un nouveau principe d'induction

Illustration



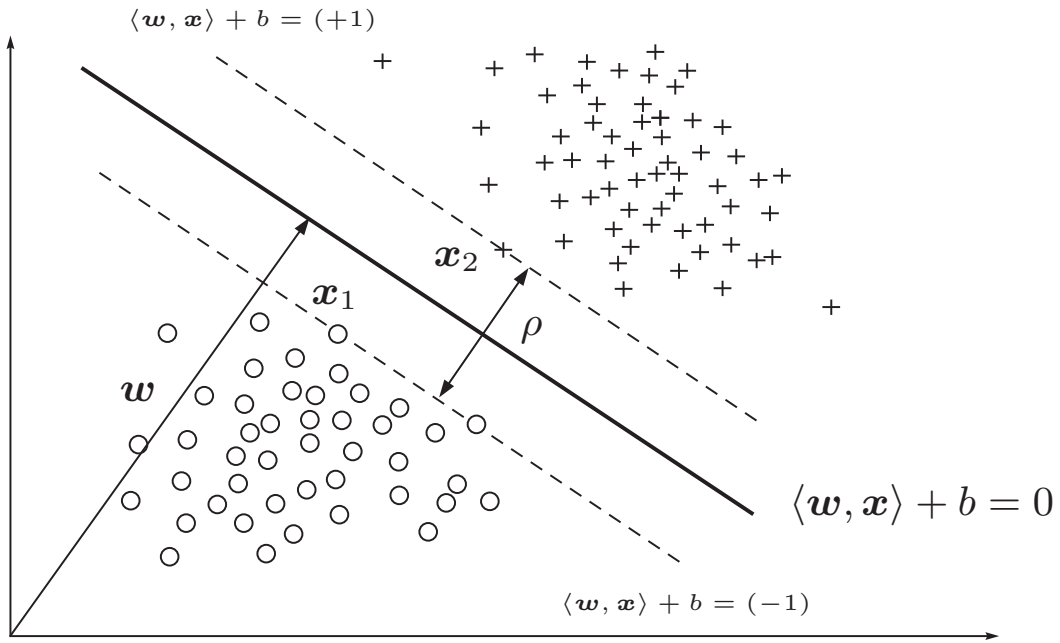
Faible marge : probablement de faibles performances en généralisation

Large marge : probablement de bonnes performances en généralisation

Ceci va être justifié de manière plus rigoureuse à présent.

Un nouveau principe d'induction

Calcul de la marge



On a $\langle \boldsymbol{w}, \boldsymbol{x}_2 \rangle + b = (+1)$ et $\langle \boldsymbol{w}, \boldsymbol{x}_1 \rangle + b = (-1)$. Il en résulte directement que

$$\rho = \left\langle \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}, \boldsymbol{x}_2 - \boldsymbol{x}_1 \right\rangle = \frac{2}{\|\boldsymbol{w}\|}$$

Un nouveau principe d'induction

Maximisation de la marge

On justifie la maximisation de la marge ρ, principe de base des SVM, par le résultat suivant issu de la théorie statistique de l'apprentissage.

Théorème 1. *Considérons les hyperplans de la forme* $\langle \boldsymbol{w}, \boldsymbol{x} \rangle = 0$, *où* $\boldsymbol{w}$ *est normalisé de sorte qu'ils soient sous forme canonique par rapport à* $\mathcal{A}_n$, *soit :*

$$\min_{\boldsymbol{x} \in \mathcal{A}_n} |\langle \boldsymbol{w}, \boldsymbol{x} \rangle| = 1.$$

L'ensemble des fonctions de décision $d(\boldsymbol{x}; \boldsymbol{w}) = sgn\langle \boldsymbol{w}, \boldsymbol{x} \rangle$ *définies à partir de* $\mathcal{A}_n$ *et vérifiant la contrainte* $\|\boldsymbol{w}\| \leq \Lambda$ *à une VC-dimension* h *vérifiant :*

$$h \leq R^2 \Lambda^2,$$

où R *est le rayon de la plus petite sphère centrée sur l'origine contenant* $\mathcal{A}_n$.

En conséquence, plus $\rho = 2/\|\boldsymbol{w}\|$ est grand, plus h est petit.

Support vector machines (hard margin)

Formulation du problème d'optimisation

Maximiser la marge, définie par $\rho = \frac{2}{\|\boldsymbol{w}\|}$, est équivalent à minimiser $\|\boldsymbol{w}\|^2$. On implémente le principe MRS en résolvant le problème d'optimisation suivant :

Minimiser $\frac{1}{2}\|\boldsymbol{w}\|^2$

sous les contraintes $y_i\left(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b\right) \geq 1, \qquad 1 \leq i \leq n.$

Remarque. Cette formulation est valide pour des classes linéairement séparables.

Support vector machines (hard margin)

Rappels sur les multiplicateurs de Lagrange

La minimisation d'une fonction convexe $f(\boldsymbol{x})$ sous les contraintes $g_i(\boldsymbol{x}) \leq 0$, $i = 1, \ldots, n$, est équivalente à la recherche du point selle du Lagrangien

$$L(\boldsymbol{x}; \boldsymbol{\alpha}) = f(\boldsymbol{x}) + \sum_{i=1}^{n} \alpha_i \, g_i(\boldsymbol{x}).$$

Le minimum est recherché par rapport à $\boldsymbol{x}$. Le maximum l'est par rapport aux n facteurs de Lagrange α_i, qui doivent être positifs ou nuls.

Les conditions, dites de Karush-Kuhn-Tucker, sont satisfaites à l'optimum :

$$\alpha_i^* \, g_i(\boldsymbol{x}^*) = 0, \qquad i = 1, \ldots, n.$$

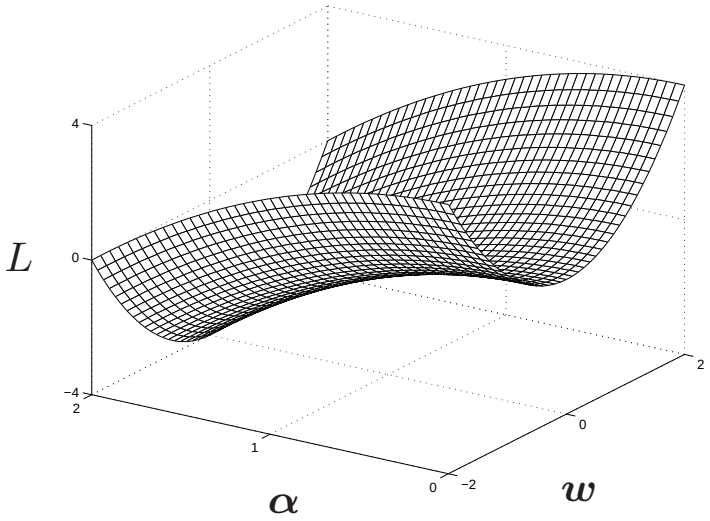
Support vector machines (hard margin)

Résolution par la méthode de Lagrange

On résout le problème précédent à l'aide de la méthode du Lagrangien

$$L(\boldsymbol{w}, b; \boldsymbol{\alpha}) = \frac{1}{2} \|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \alpha_i \{y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) - 1\}, \quad \alpha_i \geq 0.$$

La fonction L doit être minimisée par rapport aux variables primales $\boldsymbol{w}$ et b et maximisée par rapport aux variables duales α_i.



Support vector machines (hard margin)

Formulation du problème dual

Les conditions d'optimalité formulées à l'égard du Lagrangien

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

se traduisent par des dérivées nulles par rapport aux variables primales et duales :

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

Un rapide calcul mène aux relations suivantes qui, injectées dans l'expression du Lagrangien, fourniront le problème dual à résoudre.

$$\sum_{i=1}^{n} \alpha_i^* \, y_i = 0 \qquad \boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i^* \, y_i \, \boldsymbol{x}_i.$$

Support vector machines (hard margin)

Formulation du problème dual

Le problème d'optimisation dual s'exprime finalement ainsi :

$$\text{Maximiser } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \, \alpha_j \, y_i \, y_j \, \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$$

$$\text{sous les contraintes } \sum_{i=1}^{n} \alpha_i \, y_i = 0, \quad \alpha_i \geq 0, \qquad \forall i = 1, \ldots, n.$$

Support vector machines (hard margin)

Formulation de la solution et support vectors

Le vecteur normal au plan séparateur optimum s'exprime sous la forme :

$$\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i^* \, y_i \, \boldsymbol{x}_i$$

D'après les conditions de Karush-Kuhn-Tucker, à l'optimum on a :

$$\alpha_i^* \{ y_i (\langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle + b^*) - 1 \} = 0.$$

Cas 1 : $y_i(\langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle + b^*) > 1$
On a $\alpha_i^* = 0$, signifiant que $\boldsymbol{x}_i$ n'apparaît pas dans l'expression de $\boldsymbol{w}^*$.

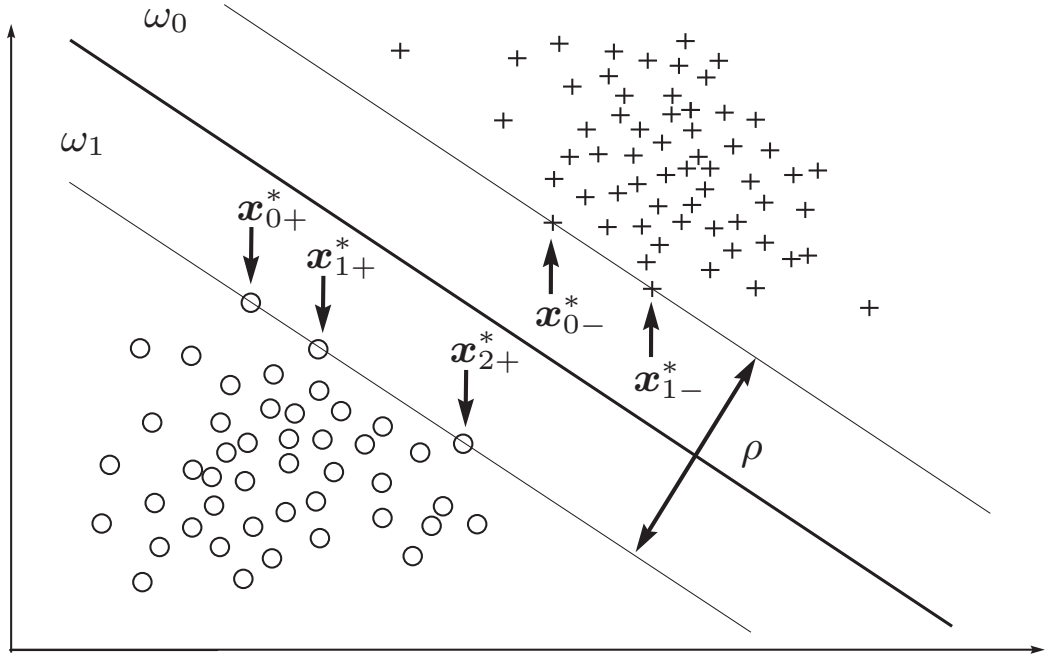
Cas 2 : $y_i(\langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle + b^*) = 1$
On a $\alpha_i^* \neq 0$ et $\boldsymbol{x}_i$ se trouve sur la marge. On déduit b^* de tels points.

Le vecteur $\boldsymbol{w}^*$ n'est défini qu'à partir des $\boldsymbol{x}_i$ situé sur la marge, les *Support Vectors*.

Support vector machines (hard margin)

Support vectors

Les support vectors sont indiqués ci-dessous par les flèches.



Performances en généralisation des SVM

Le fait que l'hyperplan optimum s'exprime uniquement à partir des support vectors est remarquable car, en général, leur nombre est faible.

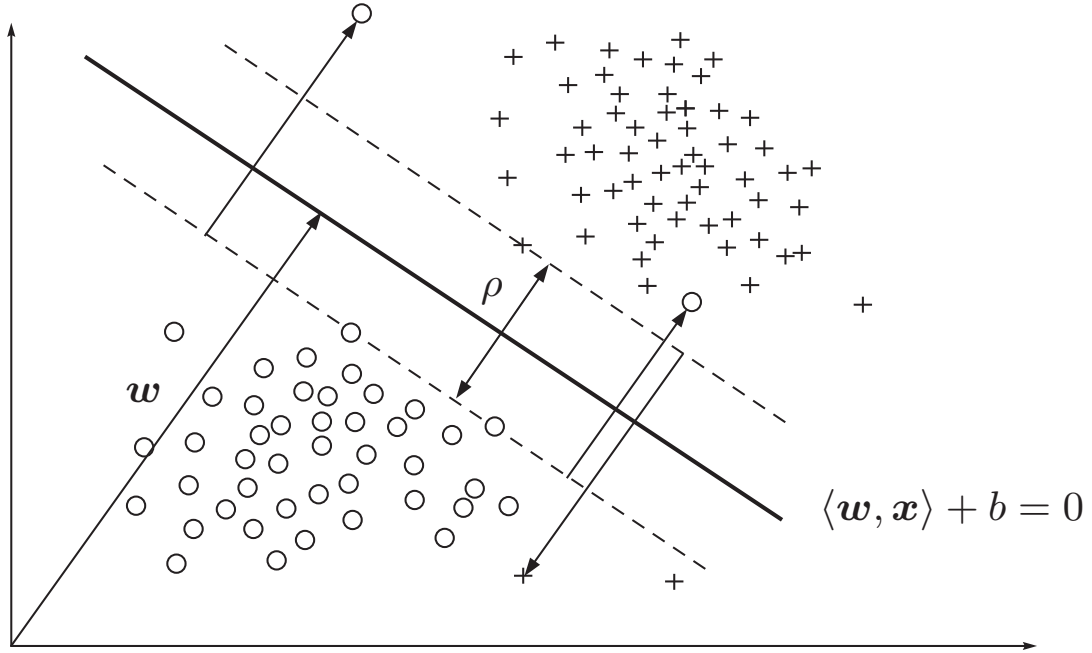
Le nombre n_{sv} de support vectors permet d'estimer les performances en généralisation du classifieur :

$$\mathrm{E}\{P_e\} \leq \frac{\mathrm{E}\{n_{sv}\}}{n}$$

Support vector machines (soft margin)

Mode de pénalisation

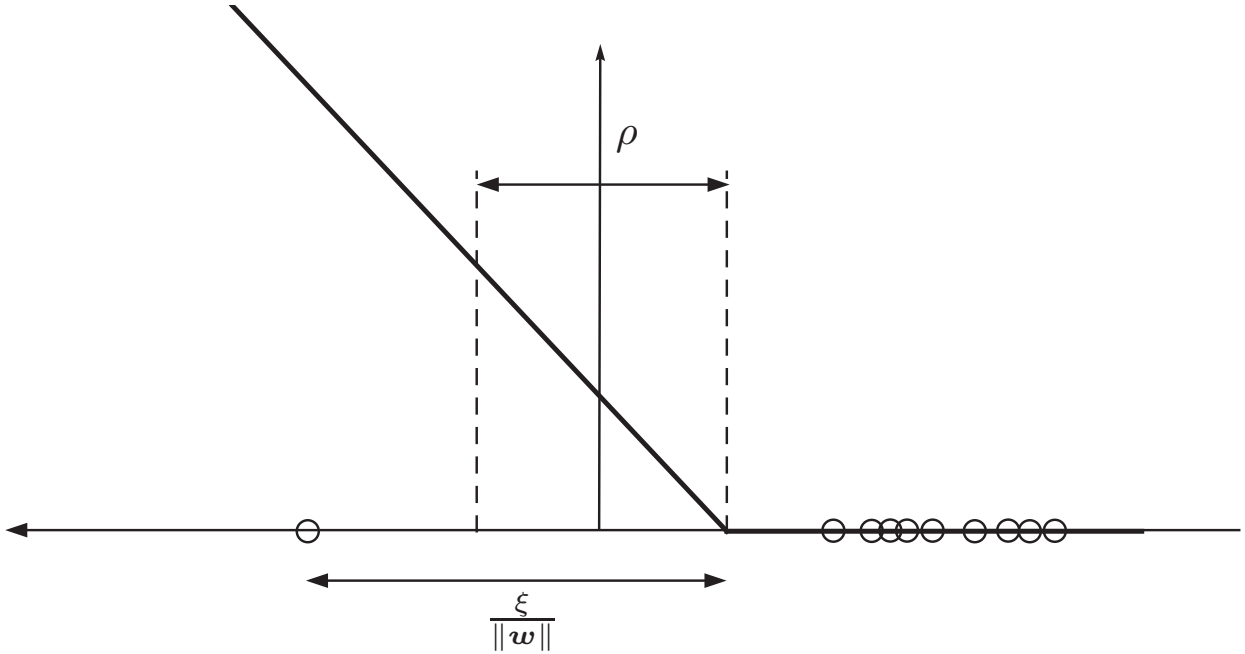
Lorsque les classes en compétition ne sont pas séparables linéairement, il convient de modifier la formulation du problème afin de pénaliser les données mal classées.



Support vector machines (soft margin)

Fonctions de pénalisation

Le mode de pénalisation le plus courant est relatif à la distance de l'échantillon mal-classé à la marge. On considère parfois le carré de cette dernière.



Support vector machines (soft margin)

Formulation du problème d'optimisation

Le schéma précédent incite à formuler le problème d'optimisation ainsi.

Minimiser $\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i, \quad C \geq 0$

sous les contraintes $y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \qquad 1 \leq i \leq n.$

Le terme $C\sum_{i=1}^{n}\xi_i$ a pour effet de pénaliser les échantillons mal classés. D'autres fonctions de pénalisation que celle-ci existent.

Support vector machines (soft margin)

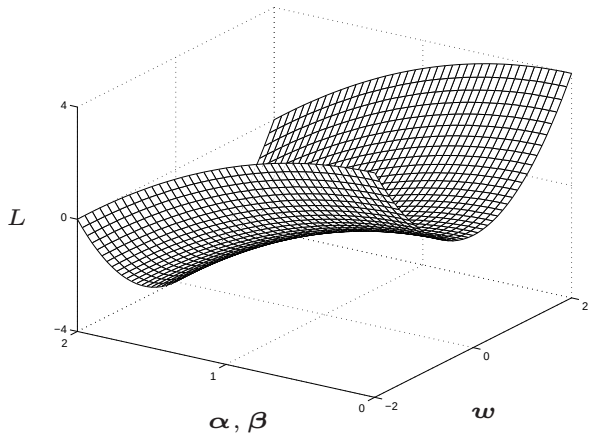
Résolution par la méthode de Lagrange

On résout le problème précédent à l'aide de la méthode du Lagrangien

$$L(\boldsymbol{w}, b, \boldsymbol{\xi}; \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\{y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) - 1 + \xi_i\} - \sum_{i=1}^{n}\beta_i\xi_i,$$

où les α_i et β_i sont des facteurs de Lagrange, positifs ou nuls.

La fonction L doit être minimisée par rapport aux variables primales $\boldsymbol{w}$ et b et maximisée par rapport aux variables duales α_i et β_i.



Support vector machines (soft margin)

Formulation du problème dual

Les conditions d'optimalité formulées à l'égard du Lagrangien

$$L(\boldsymbol{w}, b, \boldsymbol{\xi}; \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2} \|\boldsymbol{w}\|^2 + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \{y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) - 1 + \xi_i\} - \sum_{i=1}^{n} \beta_i \xi_i,$$

se traduisent par des dérivées nulles par rapport aux variables primales et duales :

$$\frac{\partial}{\partial \boldsymbol{w}} L(\boldsymbol{w}, b, \boldsymbol{\xi}; \boldsymbol{\alpha}, \boldsymbol{\beta}) = 0 \quad \Longrightarrow \quad \boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i^* \, y_i \, \boldsymbol{x}_i$$

$$\frac{\partial}{\partial b} L(\boldsymbol{w}, b, \boldsymbol{\xi}; \boldsymbol{\alpha}, \boldsymbol{\beta}) = 0 \quad \Longrightarrow \quad \sum_{i=1}^{n} \alpha_i^* \, y_i = 0$$

$$\frac{\partial}{\partial \boldsymbol{\xi}} L(\boldsymbol{w}, b, \boldsymbol{\xi}; \boldsymbol{\alpha}, \boldsymbol{\beta}) = 0 \quad \Longrightarrow \quad \beta_i^* = C - \alpha_i^*$$

Injectées dans l'expression du Lagrangien, ces relations fournissent le problème dual à résoudre.

Support vector machines (soft margin)

Formulation du problème dual et solution

Le problème d'optimisation dual s'exprime finalement ainsi :

$$\text{Minimiser } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \, \alpha_j \, y_i \, y_j \, \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$$

$$\text{sous les contraintes } \sum_{i=1}^{n} \alpha_i \, y_i = 0, \quad 0 \leq \alpha_i \leq C, \qquad \forall i = 1, \ldots, n.$$

La solution du problème s'écrit finalement

$$d(\boldsymbol{x}; \boldsymbol{\alpha}^*, b^*) = \text{sign} \left(\sum_{sv} \alpha_i^* \, y_i \, \langle \boldsymbol{x}, \boldsymbol{x}_i \rangle + b^* \right)$$

Afin de déterminer b^*, on utilise les conditions de Karush-Kuhn-Tucker :

$$\alpha_i^* \{ y_i (\langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle + b^*) - 1 + \xi_i^* \} = 0, \qquad \beta_i^* \, \xi_i^* = 0.$$

Pour tout support vector $\boldsymbol{x}_i$ tel que $\alpha_i < C$, on a $\xi_i = 0$ et $b^* = y_i - \langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle$.

Support vector machines (soft margin)

Choix du paramètre C

Minimiser $\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i, \quad C \geq 0$

sous les contraintes $y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \qquad 1 \leq i \leq n.$

Le paramètre C établit un compromis entre la largeur de la marge, qui a un rôle régularisant, et le nombre d'échantillons mal classés.

C grand : petite marge, moins d'erreurs de classification

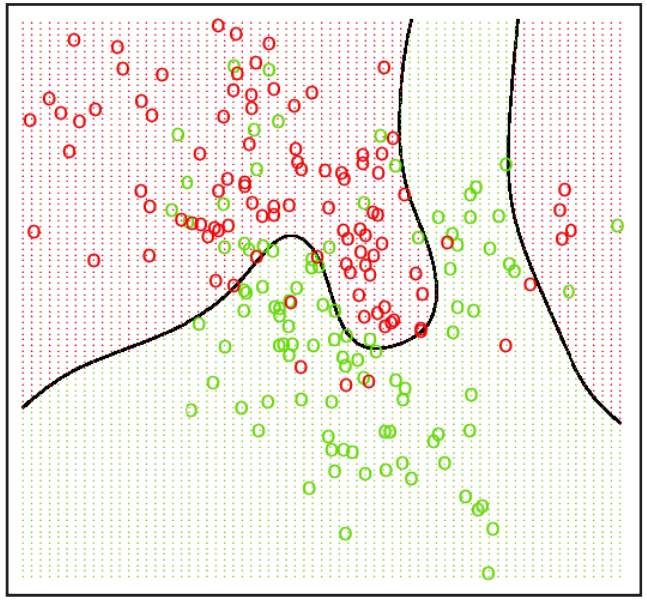
C petit : grande marge, plus d'erreurs de classification

Le choix du paramètre C peut être optimisé par validation croisée.

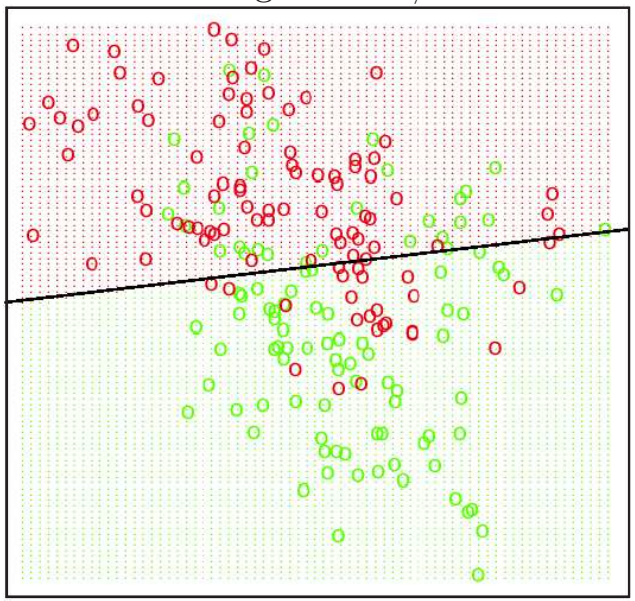
Exemples de mises en œuvre

Détecteur de Bayes et discriminant de Fisher

Classifieur de Bayes



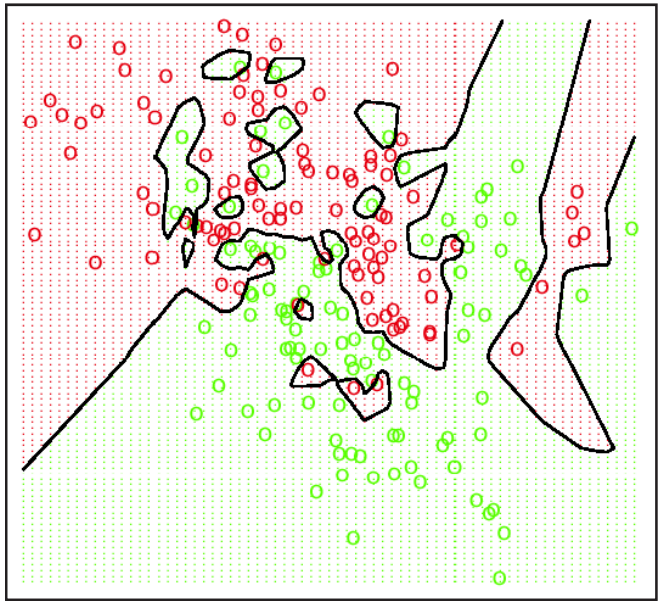
Régression 0/1



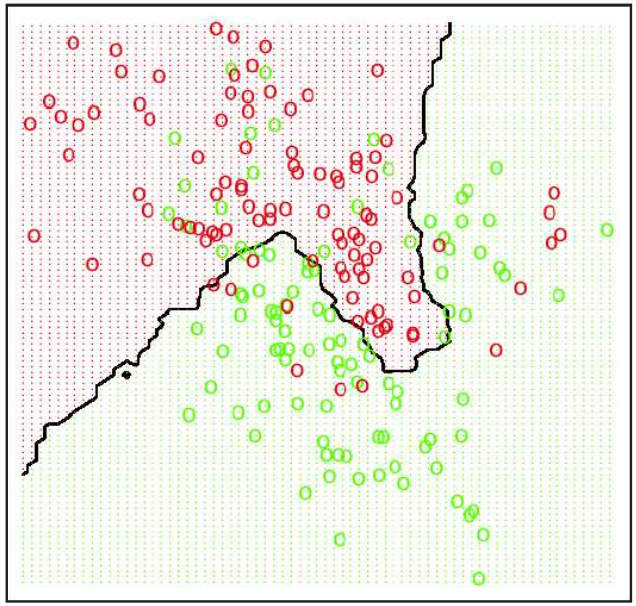
Exemples de mises en œuvre

K plus-proches-voisins

1-ppv



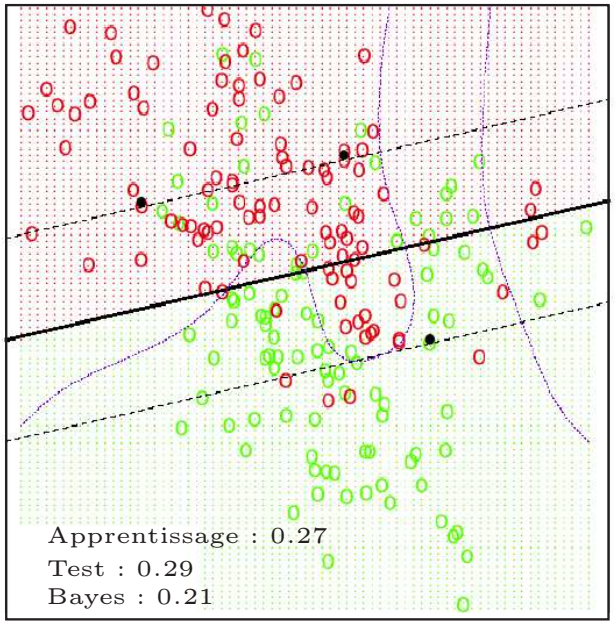
15-ppv



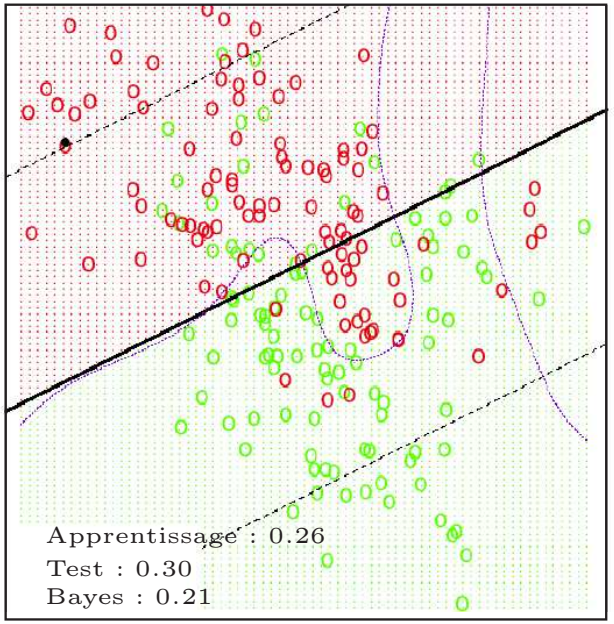
Exemples de mises en œuvre

Support vector machines

$C = 10^4$



$C = 10^{-2}$



Support vector machines

Du linéaire au non-linéaire

Les classifieurs linéaires ont des capacités de classification limitées. Pour y remédier, on peut les mettre en œuvre après transformation non-linéaire des données :

$$\boldsymbol{x} \longrightarrow \boldsymbol{\phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \ldots]^t$$

où les $\varphi_i(\boldsymbol{x})$ sont des fonctions non-linéaires préalablement choisies.

Un classifieur linéaire en $\boldsymbol{\phi}(\boldsymbol{x})$ est non-linéaire par rapport à $\boldsymbol{x}$

Support vector machines

Exemple : classifieur polynomial

On considère $\boldsymbol{x} = [x(1)\ x(2)\ x(3)]^t$. Considérons la transformation suivante :

$$\begin{array}{lll} \phi_1(\boldsymbol{x}) = x(1) & \phi_4(\boldsymbol{x}) = x(1)^2 & \phi_7(\boldsymbol{x}) = x(1)\,x(2) \\ \phi_2(\boldsymbol{x}) = x(2) & \phi_5(\boldsymbol{x}) = x(2)^2 & \phi_8(\boldsymbol{x}) = x(1)\,x(3) \\ \phi_3(\boldsymbol{x}) = x(3) & \phi_6(\boldsymbol{x}) = x(3)^2 & \phi_9(\boldsymbol{x}) = x(2)\,x(3) \end{array}$$

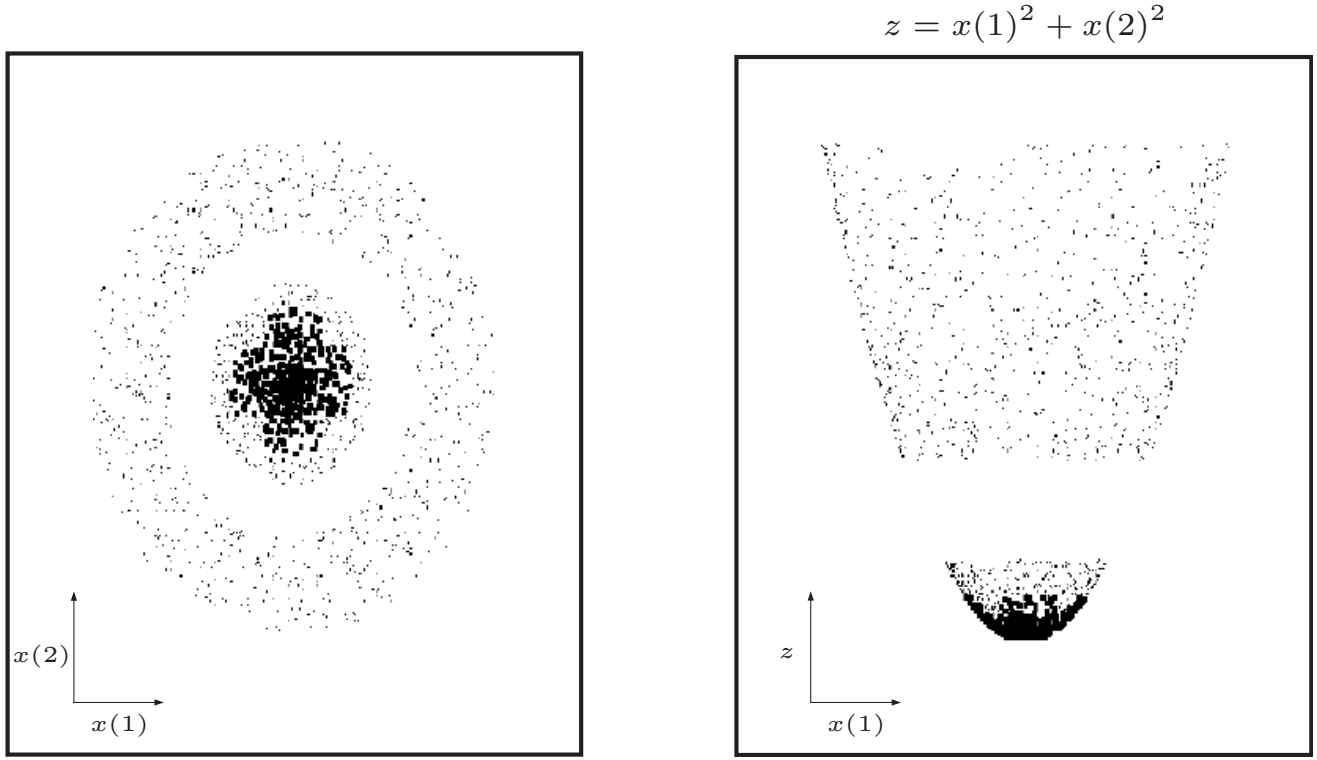
Un classifieur linéaire dans l'espace transformé $\{\boldsymbol{\phi}(\boldsymbol{x})\}_{\boldsymbol{x}\in\mathbb{R}^3}$, c'est-à-dire

$$d(\boldsymbol{x}; \boldsymbol{w}, b) = \text{sign}(\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x})\rangle + b),$$

est un classifieur polynomial de degré 2 par rapport à $\boldsymbol{x}$.

Support vector machines

Exemple : classifieur polynomial



La transformation polynomiale rend les données linéairement séparables.

Support vector machines

Optimisation dans un espace transformé

Le problème d'optimisation dual s'exprime ainsi

$$\text{Minimiser } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \, \alpha_j \, y_i \, y_j \, \langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}_j) \rangle$$

$$\text{sous les contraintes } \sum_{i=1}^{n} \alpha_i \, y_i = 0, \quad 0 \leq \alpha_i \leq C, \qquad \forall i = 1, \ldots, n.$$

La solution du problème s'écrit

$$d(\boldsymbol{x}; \boldsymbol{\alpha}^*, b^*) = \text{sign} \left(\sum_{sv} \alpha_i^* \, y_i \, \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}_i) \rangle + b^* \right).$$

On remarque que

- on n'a jamais besoin de calculer explicitement $\boldsymbol{\phi}(\boldsymbol{x})$;
- si $\boldsymbol{x}$ est de grande dimension, celle de $\boldsymbol{\phi}(\boldsymbol{x})$ l'est encore d'avantage, parfois infinie.

Support vector machines

Le coup du noyau

Si l'on parvient à définir un noyau $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}_j) \rangle$ tel que :

- la surface de décision associée soit performante

$$d(\boldsymbol{x}; \boldsymbol{\alpha}^*, b^*) = \text{sign}\left(\sum_{sv} \alpha_i^* \, y_i \, \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) + b^* \right)$$

- il soit aisé de calculer $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$, même pour des données de grande dimension...

Le tour est joué !

Le coup du noyau

Noyaux polynomiaux

Dans le cas de la transformation polynomiale de degré 2, on montre aisément que :

$$\langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}') \rangle = (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^2 \triangleq \kappa(\boldsymbol{x}, \boldsymbol{x}')$$

▷ Le calcul de produit scalaire peut s'effectuer dans $\mathbb{R}^2$!

Plus généralement, on s'intéresse à $\kappa(\boldsymbol{x}, \boldsymbol{x'}) = (1 + \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}') \rangle)^q$, avec $\boldsymbol{x} \in \mathbb{R}^l$.

$$\kappa(\boldsymbol{x}, \boldsymbol{x}') = (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^q = \sum_{j=0}^{q} \binom{q}{j} \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^j .$$

Chaque composante $\langle \boldsymbol{x}, \boldsymbol{x}' \rangle^j = [x(1)\, x'(1) + \ldots + x(l)\, x'(l)]^j$ de cette expression peut être développée en une somme pondérée de monômes de degré j de la forme

$$[x(1)\, x'(1)]^{j_1} [x(2)\, x'(2)]^{j_2} \ \ldots \ [x(l)\, x'(l)]^{j_l}$$

avec $\sum_{i=1}^{l} j_i = j$. Ceci mène directement à l'expression de $\boldsymbol{\phi}(\boldsymbol{x})$...

Le coup du noyau

Théorème de Mercer

On s'intéresse aux fonctions $\kappa(\boldsymbol{x}, \boldsymbol{x}')$ pouvant faire fonction de produit scalaire dans un espace $\mathcal{H}$. On appelle *noyau* une fonction symétrique κ de $\mathcal{X} \times \mathcal{X}$ dans $\mathbb{R}$.

Théorème 2. *Si κ est un noyau continu d'un opérateur intégral défini positif, ce qui signifie que*

$$\iint \varphi(\boldsymbol{x})\, \kappa(\boldsymbol{x}, \boldsymbol{x}')\, \varphi^*(\boldsymbol{x}')\, d\boldsymbol{x}\, d\boldsymbol{x}' \geq 0$$

pour tout $\varphi \in \mathcal{L}^2(\mathcal{X})$, il peut être décomposé sous la forme

$$\kappa(\boldsymbol{x}, \boldsymbol{x}') = \sum_{i=1}^{\infty} \lambda_i\, \psi_i(\boldsymbol{x})\, \psi_i(\boldsymbol{x}'),$$

où ψ_i et λ_i sont les fonctions propres (orthogonales) et valeurs propres (positives) du noyau κ, respectivement, telles que

$$\int \kappa(\boldsymbol{x}, \boldsymbol{x}')\, \psi_i(\boldsymbol{x})\, d\boldsymbol{x} = \lambda_i\, \psi_i(\boldsymbol{x}').$$

Le coup du noyau

Théorème de Mercer

Il est aisé de voir qu'un noyau κ satisfaisant au théorème de Mercer peut faire fonction de produit scalaire dans un espace transformé $\mathcal{H}$. Il suffit d'écrire :

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{pmatrix} \sqrt{\lambda_1}\,\psi_1(\boldsymbol{x}) \\ \sqrt{\lambda_2}\,\psi_2(\boldsymbol{x}) \\ \cdots \end{pmatrix}$$

Dans ces conditions, on vérifie bien que l'on retrouve : $\langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x'}) \rangle = \kappa(\boldsymbol{x}, \boldsymbol{x'})$.

On définit l'espace $\mathcal{H}$ comme étant celui engendré par les fonctions propres ψ_i du noyau κ, c'est-à-dire

$$\mathcal{H} \triangleq \{f(\cdot) \mid f(x) = \sum_{i=1}^{\infty} \alpha_i\, \psi_i(x),\ \alpha_i \in \mathbb{R}\}.$$

Le coup du noyau

Exemples de noyaux de Mercer

On peut montrer que les noyaux suivants vérifie la condition de Mercer, et correspondent donc à un produit scalaire dans un espace $\mathcal{H}$.

Noyaux projectifs	
sigmoidal	$\frac{1}{\eta_0}\tanh(\beta_0\langle \boldsymbol{x}, \boldsymbol{x}'\rangle - \alpha_0)$
monomial de degré q	$\langle \boldsymbol{x}, \boldsymbol{x}'\rangle^q$
polynomial de degré q	$(1 + \langle \boldsymbol{x}, \boldsymbol{x}'\rangle)^q$

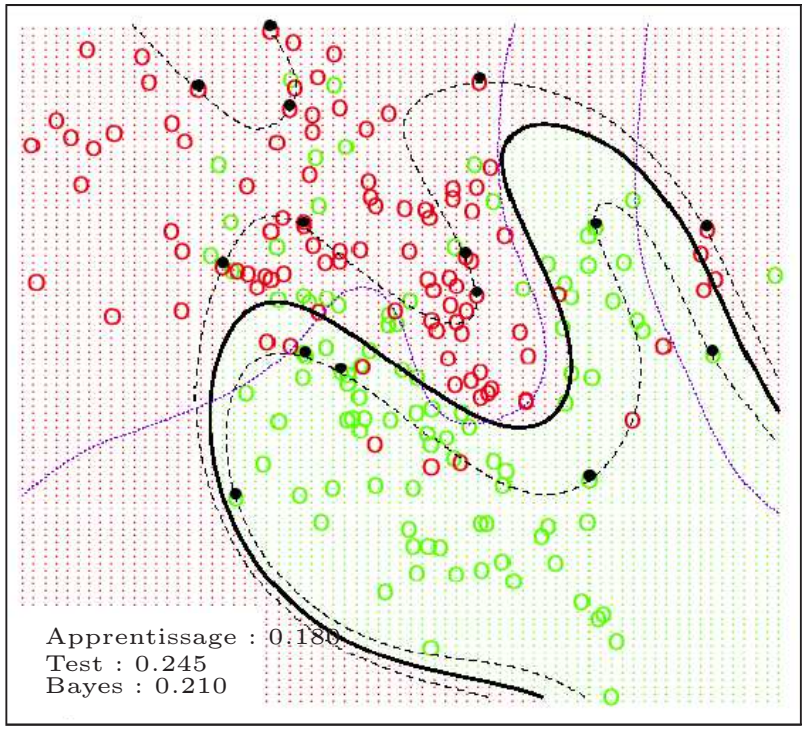
Noyaux radiaux	
uniforme	$\frac{1}{\eta_0}\,\mathbb{1}_{\|\boldsymbol{x}-\boldsymbol{x}'\|\leq\beta_0}$
Epanechnikov	$\frac{1}{\eta_0}\,(\beta_0^2 - \|\boldsymbol{x}-\boldsymbol{x}'\|^2)\,\mathbb{1}_{\|\boldsymbol{x}-\boldsymbol{x}'\|\leq\beta_0}$
Cauchy	$\frac{1}{\eta_0}\,\frac{1}{1+\|\boldsymbol{x}-\boldsymbol{x}'\|^2/\beta_0^2}$

... et encore $\kappa_1(\boldsymbol{x}, \boldsymbol{x}') + \kappa_2(\boldsymbol{x}, \boldsymbol{x}')$, $\kappa_1(\boldsymbol{x}, \boldsymbol{x}') \cdot \kappa_2(\boldsymbol{x}, \boldsymbol{x}')$, ...

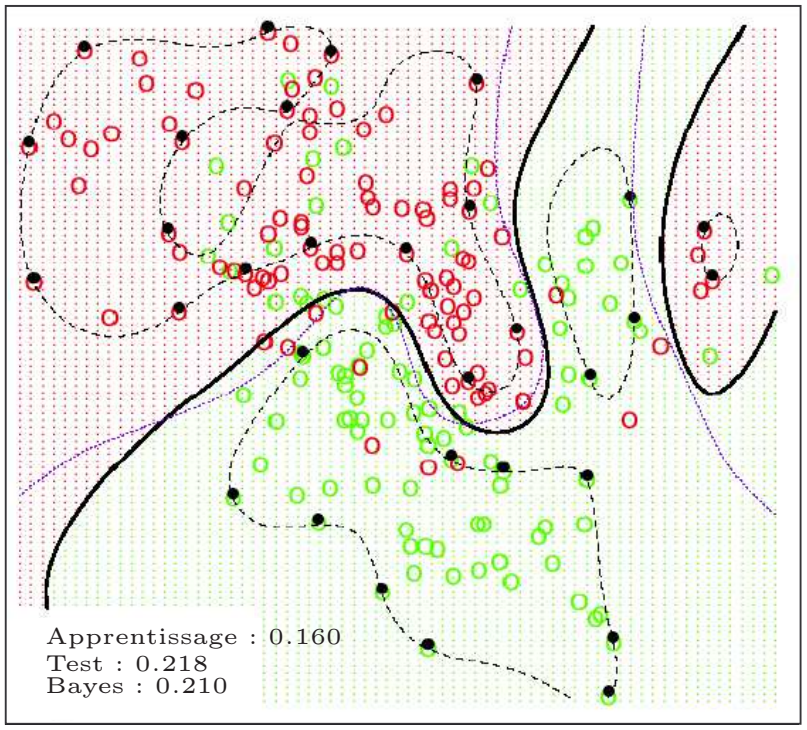
Exemples de mises en œuvre

Support vector machines

noyau polynomial de degré 4



noyau gaussien



Le coup du noyau

En résumé

Parmi les possibilités offertes par le coup du noyau, on note en particulier :

— Tout algorithme pouvant s'exprimer à partir de produits scalaires peut bénéficier du coup du noyau en vue d'une extension au cas non-linéaire.

— Grâce au coup du noyau, tout algorithme de reconnaissance des formes est en mesure de traiter des données autres que numériques, par exemple alphabétiques.

Premier bilan sur les SVM

Qualités

Par rapport aux techniques concurrentes telles que les réseaux de neurones artificiels, les SVM possèdent d'immenses qualités.

1. Solution unique
   $\longrightarrow$ Problème quadratique

2. Processus de régularisation intégré, solution sparse
   $\longrightarrow$ Fonction coût et contraintes d'inégalité en découlant

3. Extension aisée au cas non-linéaire, solution non boite noire
   $\longrightarrow$ Kernel trick

Premier bilan sur les SVM

Une difficulté majeure

La mise en œuvre de l'algorithme d'apprentissage par une approche directe s'avère difficile car la taille du problème est celle de la base d'apprentissage $\mathcal{A}_n$.

$$\begin{aligned} &\text{Minimiser} \quad W(\boldsymbol{\alpha}) = \tfrac{1}{2}\,\boldsymbol{\alpha}^t \boldsymbol{H} \boldsymbol{\alpha} + \mathbb{1}_n^t \boldsymbol{\alpha} \\ &\text{sous les contraintes} \quad \boldsymbol{y}^t \boldsymbol{\alpha} = 0, \quad 0 \cdot \mathbb{1}_n \leq \boldsymbol{\alpha} \leq C \cdot \mathbb{1}_n. \end{aligned}$$

Des solutions algorithmiques

Méthodes par décomposition

Des algorithmes performants existent, basés sur la décomposition du problème d'optimisation en sous-problèmes.

Minimiser

$$W(\boldsymbol{\alpha}_B) = \tfrac{1}{2}\,(\boldsymbol{\alpha}_B \;\; \boldsymbol{\alpha}_N)^t \begin{pmatrix} \boldsymbol{H}_{BB} & \boldsymbol{H}_{BN} \\ \boldsymbol{H}_{NB} & \boldsymbol{H}_{NN} \end{pmatrix} (\boldsymbol{\alpha}_B \;\; \boldsymbol{\alpha}_N) - (\mathbb{1}_B \;\; \mathbb{1}_N) \begin{pmatrix} \boldsymbol{\alpha}_B \\ \boldsymbol{\alpha}_N \end{pmatrix}$$

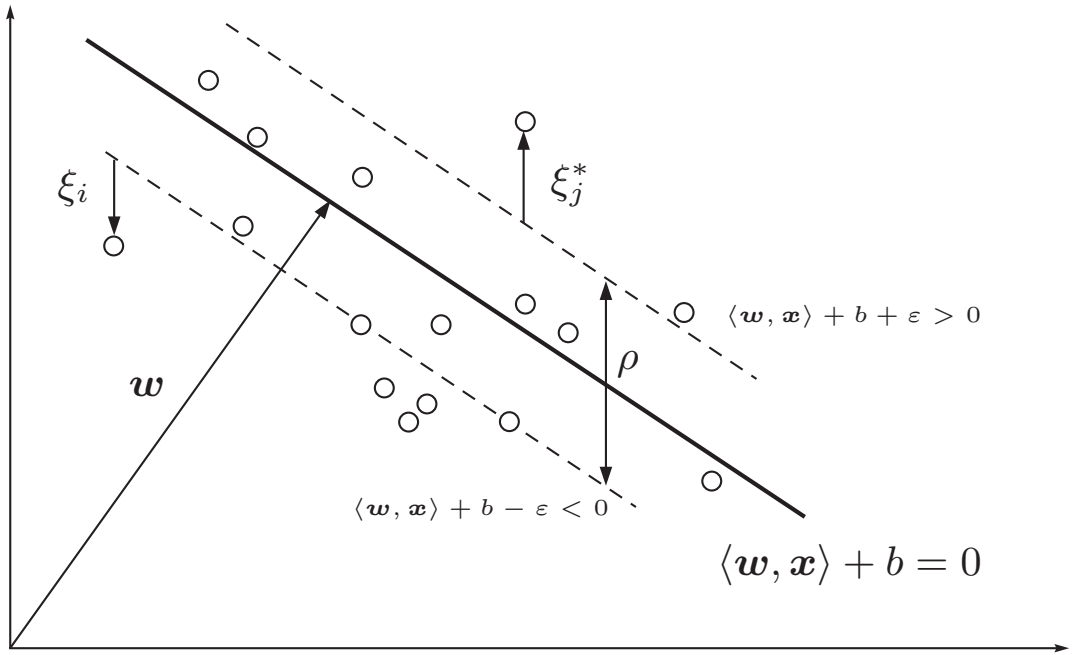
sous les contraintes $\quad \boldsymbol{\alpha}_B^t\, \boldsymbol{y}_B + \boldsymbol{\alpha}_N^t\, \boldsymbol{y}_N = 0 \quad 0 \cdot \mathbb{1}_B \leq \boldsymbol{\alpha}_B \leq C \cdot \mathbb{1}_B.$

Plusieurs stratégies ont été proposées pour décomposer le problème. La plus extrême consiste à limiter B à deux éléments. La résolution est alors analytique.

Application à la régression

Support vector regression

L'idée de marge, sur laquelle repose la presque totalité des qualités des SVM, peut être appliquée à la résolution de problèmes de régression.



Application à la régression

Support vector regression

Le problème d'optimisation correspondant s'exprime de la façon suivante.

$$\begin{aligned} &\text{Minimiser } \tfrac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \\ &\text{sous les contraintes} \quad y_i - (\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq \varepsilon + \xi_i, \quad \xi_i \geq 0 \\ &\qquad\qquad\qquad\quad (\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) - y_i \geq \varepsilon + \xi_i^*, \quad \xi_i^* \geq 0 \end{aligned}$$

Le kernel trick s'applique de la même façon que pour les SVM, tout comme les algorithmes de résolution.

Application à la régression

Support vector regression

On résout le problème précédent à l'aide de la méthode du Lagrangien. On aboutit au problème dual suivant :

$$\text{Minimiser } W(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = \tfrac{1}{2} \sum_{i,j=1}^{n} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) + \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$$
$$+ \varepsilon \sum_{i=1}^{n} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i \, (\alpha_i - \alpha_i^*)$$

sous les contraintes

$$\sum_{i=1}^{n} \alpha_i = \sum_{i=1}^{n} \alpha_i^*, \quad 0 \leq \alpha_i, \alpha_i^* \leq C, \quad \forall i = 1, \ldots, n.$$

La solution s'écrit finalement sous la forme suivante, permettant ainsi la mise en œuvre du coup du noyau :

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} (\alpha_i^* - \alpha_i) \langle \boldsymbol{x}, \boldsymbol{x}_i \rangle.$$

Application à la régression

Support vector regression