Formation Machine Learning - Support Vector Machine

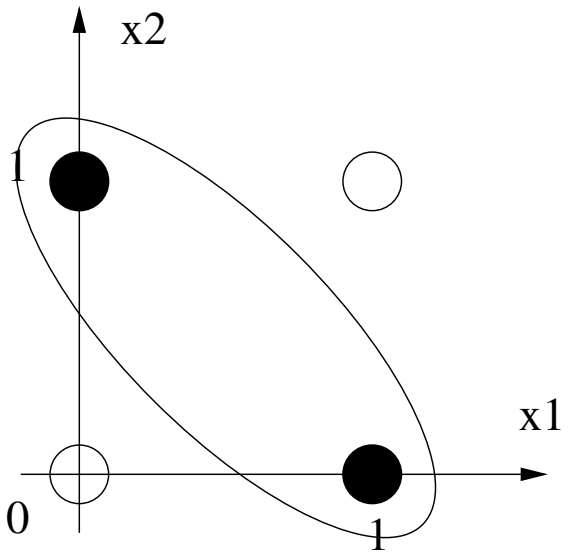
Jean-Marc Martinez

Université Paris-Saclay
CEA, Département de Modélisation des Systèmes et Structures
91191 Gif-sur-Yvette, France

CEA Maison de la simulation 24-26 mai 2023

Rappel de l'introduction au SVM

- La fonction logique **XOR** non linéaire séparable.
- Le **XOR** devient linéairement séparable dans $\{x_1, x_2, x_1x_2\}$



$$y = \begin{cases} 1 & \texttt{si} \quad x_1 + x_2 - 2x_1x_2 \geq 0.5 \\ 0 & \texttt{sinon} \end{cases}$$

- Solution par les SVM : transformation des données $\rightarrow$ **espace de re-description** dans lequel le problème devient linéairement séparable

Classification supervisée

• Nous allons *reconsidérer* le problème de la classification supervisée sans faire appel à une modélisation probabiliste.

$$\boxed{\textit{loss function} \neq \text{likelihood}}$$

• Nous nous limiterons aux problèmes à 2 classes.

• Notation : Base d'exemples $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ où $x_i \in \mathbb{R}^p$ représente les caractéristiques de l'exemple x_i associé à une des 2 classes codées par $y_i \in \{-1, +1\}$.

• Définition : un problème de classification est **linéairement séparable** lorsque les 2 classes peuvent être séparées par une forme linéaire (affine) des entrées x

• Construction d'un SVM fait appel aux méthodes d'optimisation sous contraintes que nous allons introduire.

Optimisation sous contraintes égalités

- Soit une fonction $f : x \in \mathbb{R}^p \rightarrow \mathbb{R}$.
- Recherche du minimum de f sous les contraintes $g_i(x) = 0, i = 1, \ldots, m$.
- On suppose f et g différentiables.

Construction du lagrangien

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$

où λ_i sont les multiplicateurs de Lagrange.

Conditions nécessaires

Si le problème a une solution x^* sous les contraintes, on a les conditions nécessaires :

$$\begin{array}{lcll} \nabla_x L_x(x^*, \lambda^*) & = & \nabla_x f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla_x g_i(x^*) & = & 0 \\ \nabla_\lambda L(x^*, \lambda^*) & = & g_i(x^*) & = & 0, \quad i = 1, \ldots, m \end{array}$$

Equivalence

Optimisation dans $\mathbb{R}^p$ sous m contraintes = Optimisation sans contraintes dans $\mathbb{R}^{p+m}$

Exemple : distance minimale d'un point à un plan

Lagragien

La distance minimale entre le point u et le plan défini par $w^t x + b = 0$ est obtenue en minimisant $||x - u||^2$ sous la contrainte $w^t x + b = 0$:

$$L(x, \lambda) = ||x - u||^2 + \lambda(w^t x + b)$$

Conditions nécessaires

$$\begin{aligned} (a) \quad \nabla_x L(x^*, \lambda^*) &= 2(x^* - u) + \lambda^* w = 0 \\ (b) \quad \nabla_\lambda L(x^*, \lambda^*) &= w^t x^* + b = 0 \end{aligned}$$

Résolution

$$\begin{aligned} (a) \Rightarrow x^* &= u - \frac{\lambda^*}{2} w \\ (b) \Rightarrow w^t(u - \frac{\lambda^*}{2} w) + b &= 0 \Rightarrow \lambda^* = 2\frac{w^t u + b}{||w||^2} \\ ||x^* - u|| &= \frac{||\lambda^*}{2} ||w|| \\ \Rightarrow ||x^* - u|| &= \frac{|w^t u + b|}{||w||} \end{aligned}$$

Optimisation sous contraintes inégalités

Soit une fonction $f : x \in \mathbb{R}^p$. Recherche du minimum de f sous les contraintes $g_i(x) \geq 0$ pour $i = 1, \ldots, m$. On suppose f et g différentiables.

Lagrangien

$$L(x, \mu) = f(x) - \sum_{i=1}^{m} \mu_i g_i(x)$$

où μ_i sont les multiplicateurs de Lagrange.

Conditions nécessaires de Karusch, Kuhn et Tucker

Si le problème a une solution x^* sous les m contraintes inégalités, on a les conditions :

$$\begin{aligned} \nabla_x f(x^*) - \sum_{i=1}^{m} \mu_i^* \nabla_x g_i(x^*) &= 0 \\ g_i(x^*) \leq 0, \quad \mu_i^* \geq 0, \quad \mu_i^* g_i(x^*) &= 0, i = 1, \ldots, m \end{aligned}$$

Définition d'une contrainte active

De $\mu_i^* g_i(x^*) = 0$, on en déduit soit $\mu_i^* = 0$ soit $g_i(x^*) = 0$.

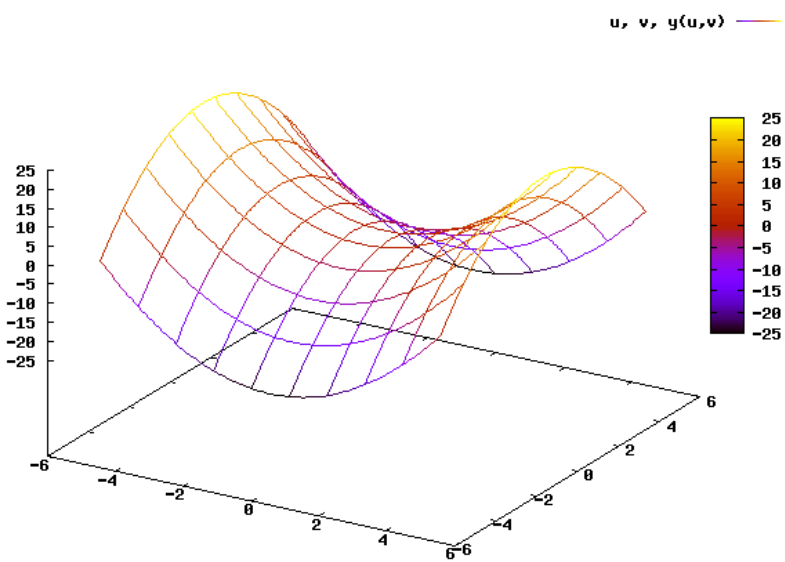
$$\mu_i^* > 0 \Rightarrow g_i(x^*) = 0 \quad \texttt{contrainte active}$$

Théorème : optimisation convexe (utile pour les SVM)

Si x^* est une solution du problème $\min_x f(x)$ sous $g_i(x) \geq 0$ (m contraintes) avec $f()$ et les $g_i()$ convexes, alors il existe un multiplicateur $\mu^* \in \mathbb{R}_+^m$ tel que le couple (x^*, μ^*) soit solution du problème du point selle :

$$\max_{\mu \geq 0} \min_x \{L(x, \mu) = f(x) - \mu^T g(x)\} = \max_{\mu \geq 0} f(x(\mu)) - \mu^T g(x(\mu))\}$$

• Variable *primale* x en fonction de la variable *duale* $x = x(\mu)$ puis recherche de la valeur μ^* qui maximise le lagragien ne dépendant plus que de μ. Puis calcul de la solution $x^*(\mu^*)$.



Exemple

Minimiser $x^T A_{n,n} x$ (A symétrique) sous m contraintes $C_{m,n} x \leq d$

$$L(x, \mu) = \frac{1}{2} x^T A x + \mu^T (Cx - d)$$

La fonction coût et la contrainte sont convexes.

$$\nabla_x L = Ax + C^T \mu$$

Le minimum par rapport à x :

$$\nabla_x L(x^*) = 0 \Rightarrow x^* = -A^{-1} C^T \mu$$

On obtient la fonctionnelle duale à maximiser

$$H(\mu) = -\frac{1}{2} \mu^T C A^{-1} C^T \mu - \mu^T d, \quad \mu_i \geq 0$$

Hyperplan séparant les 2 classes

• Pour un problème **linéairement séparable**, l'apprentissage consiste donc à déterminer un **hyperplan séparateur**.

• Le problème consiste à déterminer les coefficients (w, b) tels que la fonction de prédiction se code par :

$$f(x|w, b) = \texttt{signe}(w^T x + b)$$

• Problème linéairement séparable $\rightarrow$ il existe un hyperplan (w, b) dans $\mathbb{R}^p$ pour lequel :

$$y_i(w^T x_i + b) > 0, \quad i = 1, 2, \ldots, n$$

• Il suffit donc de trouver une solution (w, b) vérifiant les n contraintes.

• Remarque : il existe plusieurs hyperplans ! ! !

Hyperplan à marge optimale

- Afin d'avoir un classifieur **robuste**, retenons celui défini par l'hyperplan le plus éloigné des points x_i.
- Cela revient à maximiser la distance $d(x_i)$ des points x_i à l'hyperplan.

$$d(x_i) = \frac{|w^T x_i + b|}{||w||_2}$$

- Paramètres (w, b) vérifiant les n contraintes, définis à une constante près
- Forme de normalisation en retenant la solution (w, b) telle que :

$$\min_i |w^T x_i + b| = 1$$

- Hyperplan à marge optimale est donc obtenu en minimisant $||w||_2$

Optimisation quadratique sous contraintes linéaires

• La recherche des paramètres de l'hyperplan (w, b) se met sous la forme d'un lagrangien : minimiser $||w||^2$ sous les n contraintes $y_i(w^T x_i + b) >= 1$.

$$L(w, b, \mu_{[1:n]}) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \mu_i [y_i(w^T x_i + b) - 1]$$

avec les 2 conditions (a) et (b) de Karush-Kuhn-Tucker :

$$(a) : \quad \mu_i [y_i(w^T x_i + b) - 1] = 0, \quad (b) : \quad \mu_i \geq 0$$

• La solution s'obtient en recherchant le minimun du lagrangien par rapport à (w, b) (variables primales) et son maximum par rapport aux μ_i (variables duales)

• Critère, contraintes convexes $\rightarrow$ optimisation convexe. Méthode de résolution

1. exprimer la solution optimales (w^*, b^*) en fonction des variables $\boldsymbol{\mu}$
2. reformulation du lagragien en fonction des μ_i
3. calculer les valeurs optimalee μ_i^* puis en déduire (w^*, b^*)

Optimisation convexe

- Notation $\mu = (\mu_1, \mu_2, \ldots, \mu_n) \in R^{n+}$
- Problème primal, la solution doit annuler le gradient du lagragien par rapport aux variables primales

$$\begin{array}{rcrcrcl} \nabla_w L(w,b,\mu) & = & w - \sum_{i=1}^n \mu_i y_i x_i & \Rightarrow & w^* & = & \sum_{i=1}^n \mu_i y_i x_i \\ \nabla_b L(w,b,\mu) & = & -\sum_{i=1}^n \mu_i y_i & \Rightarrow & \sum_{i=1}^n \mu_i y_i & = & 0 \end{array}$$

- Problème dual obtenu en exprimant (w, b) en fonction μ dans le lagragien :

$$\begin{aligned} L(w^*, b^*, \mu) &= \frac{1}{2}||w^*||^2 - \sum_{i=1}^n \mu_i [y_i({w^*}^T x_i + b^*) - 1] \\ &= \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \mu_i \mu_j y_i y_j {x_i}^T x_j - \sum_{i=1}^n \mu_i [y_i(\sum_{j=1}^n \mu_j y_j {x_j}^T x_i + b) - 1] \end{aligned}$$

$$\Rightarrow \boxed{L(\mu) = \sum_{i=1}^n \mu_i - \frac{1}{2}\sum_{i,j=1}^n \mu_i \mu_j y_i y_j {x_i}^T x_j}$$

- Maximiser la **forme quadratique** $L(\mu)$ sous les contraintes $\mu_i \geq 0$ et $\sum_{i=1}^n \mu_i y_i = 0$

Minimisation quadratique sous contraintes linéaires

• Le problème se met donc sous forme d'une recherche *classique* d'un minimun d'une fonction quadratique sous contraintes linéaires.

• En notant K la matrice $n \times n$ d'éléments $y_i y_j x_i^T x_j$, le problème devient

$$\mu^* = \arg\min_{\mu}[\frac{1}{2}\mu^T K\mu - 1^T\mu]$$

sous les contraintes $\mu_i \geq 0$ et $\sum_i \mu_i y_i = 0$.

• Existence d'algorithmes d'optimisation convexe efficace pour déterminer μ^*. On déduit la solution w^* :

$$w^* = \sum_{i=1}^{n} \mu_i^* y_i x_i$$

• Seules les composantes $\mu_i > 0$ contribuent à la définition de l'hyperplan. Notons Sv l'ensemble des indices de ces composantes :

$$w^* = \sum_{s \in Sv} \mu_s^* y_s x_s$$

• La définition du séparateur ne dépend que des points x_s.

Modèle linéaire du SVM

- La condition de Karush-Kuhn-Tucker (a) implique que tous les points supports vérifient $y_s(w^T x_s + b) = 1$.
- Le seuil b peut donc être calculé par un des points supports ou par l'ensemble en prenant la moyenne (plus précis numériquement).
- En notant $|Sv|$ le nombre de points supports, la valeur b^* est calculée par :

$$b^* = \frac{1}{|Sv|} \sum_{s \in Sv} (y_s - w^{*T} x_s)$$

- Le classifieur à marge optimale est donc défini par :

$$f(x|w^*, b^*) = \texttt{signe}(w^{*T} x + b^*)$$

- Cette méthode se ramène à la sélection de quelques points notés x_s qui contribuent à la définition de l'hyperplan. En général $|Sv| << n$.
- Ces points sont les **supports** à la construction de l'hyperplan.
- D'où le nom donné à cette méthode, **SVM** pour Support Vector Machine.

Calcul de la marge égale à $1/||\mathbf{w}^*||$

- Le vecteur w^* s'exprime à partir des points supports :

$$w^* = \sum_{j \in Sv} \mu_j^* y_j x_j$$

- Pour chaque point support $i \in Sv$, on a $y_i(w^{*T} x_i + b^*) = 1$. D'où :

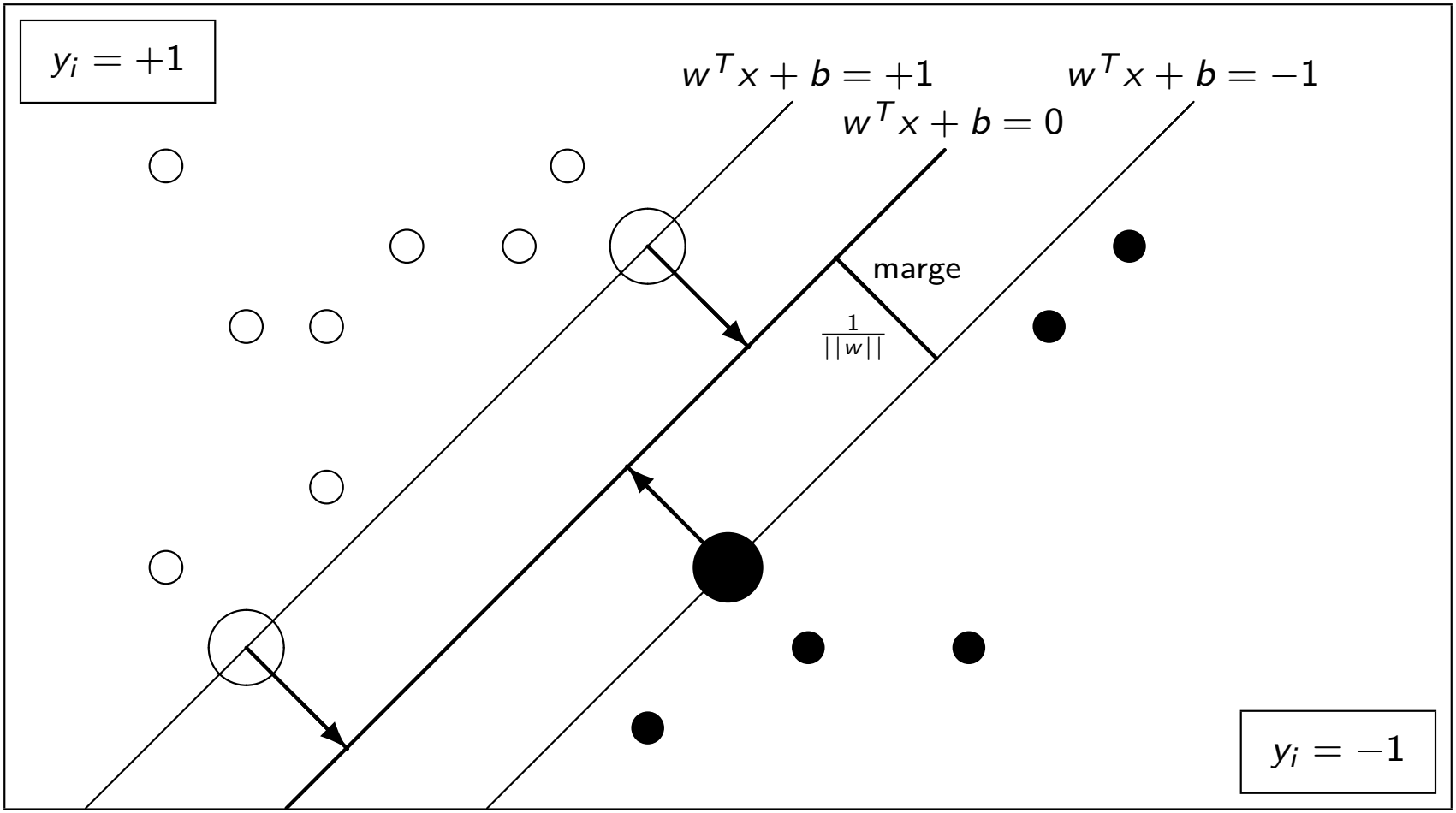
$$y_i[\underbrace{(\sum_{j \in Sv} \mu_j^* y_j x_j^T)}_{w^{*T}} x_i + b^*] = 1 \Rightarrow y_i \sum_{j \in Sv} \mu_j^* y_j x_j^T x_i = 1 - y_i b^*$$

- D'où la valeur de la marge donnée à partir de :

$$\begin{aligned} \Rightarrow ||w^*||^2 &= \sum_{i \in Sv} \mu_i^* y_i \sum_{j \in Sv} \mu_j^* y_j x_i^T x_j = \sum_{i \in Sv} \mu_i^* (1 - y_i b^*) \\ &= \sum_{i \in Sv} \mu_i^* - b \sum_{i \in Sv} \mu_i^* y_i = \sum_{i \in Sv} \mu_i^* \end{aligned}$$

$$\Rightarrow \boxed{\text{Marge} = \frac{1}{||w^*||} = \frac{1}{\sqrt{\sum_{i \in Sv} \mu_i^*}}}$$

SVM linéaire - Illustration

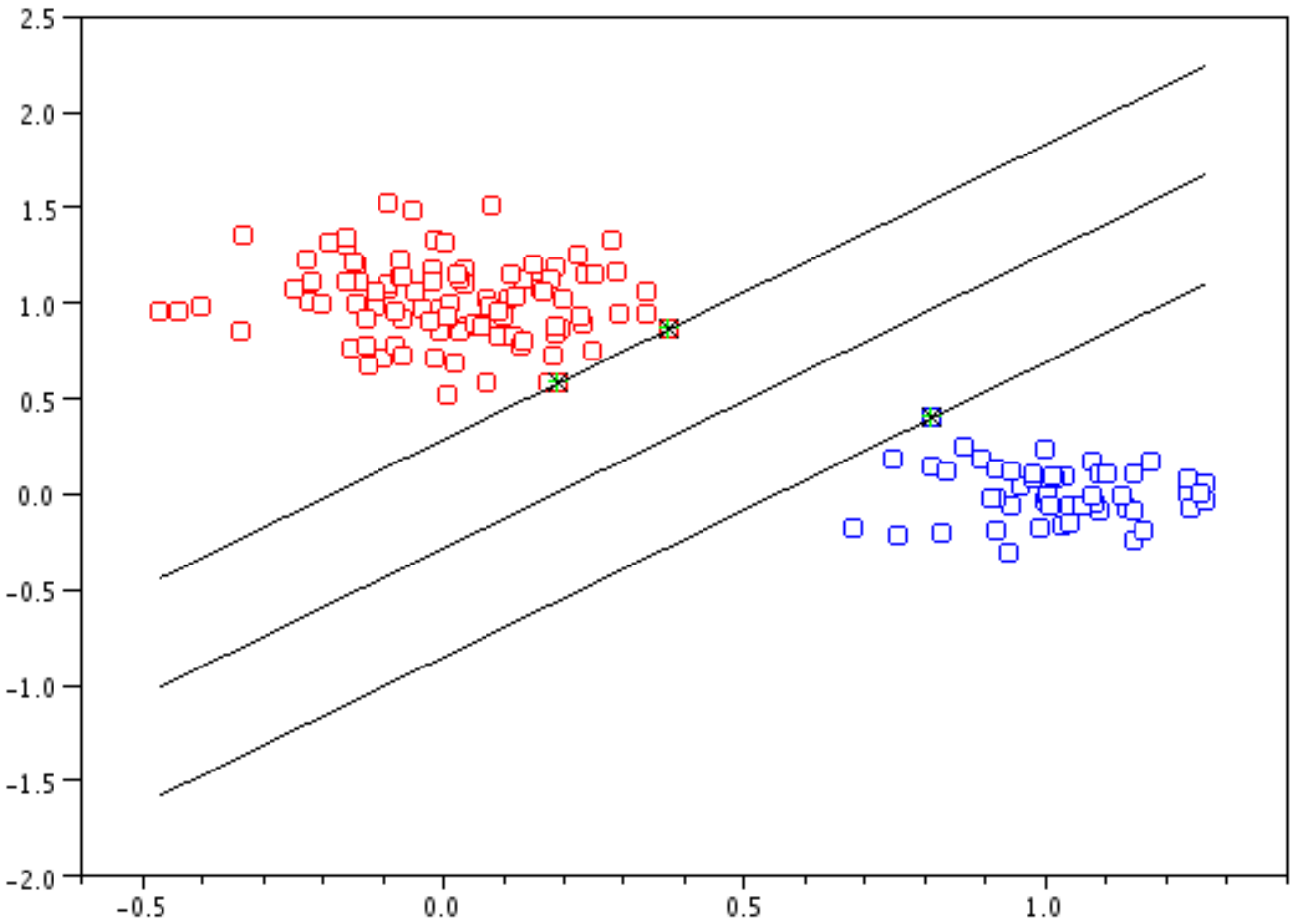


SVM à 3 vecteurs support, ceux pour lesquels la contrainte est active $y_s(w_s^T + b) = 1$

Echantillon linéairement séparable

Classe 1 : 100 points $\sim \mathcal{N}([1;0], 0.02)$ Classe 2 : 50 points $\sim \mathcal{N}([0;1], 0.04)$

Points supports pour le calcul des paramètres du séparateur (w^*, b^*)



Cas non linéairement séparable

• Lorsque les 2 classes ne sont pas linéairement séparables, certaines contraintes ne seront pas satisfaites.

• Pour cela on introduit des variables auxiliaires $\xi_i \geq 0, i = 1, 2, \ldots, n$ (*slack variables*) permettant de *relâcher* les contraintes :

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, 2, \ldots, n$$

où ξ_i est l'écart (éventuel) à la contrainte $y_i(w^T x_i + b) \geq 1$.

• On cherchera donc à minimiser également la somme des écarts ξ_i :

$$\frac{1}{2}||w||^2 + C \sum_{i=1}^{n} \xi_i$$

où C est un hyper paramètre caractérisant le compromis entre une marge optimale et le nombre d'erreurs de classement.

• Les petites valeurs de C permettent d'obtenir un modèle de prédiction plus *lisse* en négligeant les *outliers* (données bruitées). Choix de sa valeur par évaluation d'un critère (*grid search*).

Optimisation convexe

- En notant μ, ν les $2 * n$ multiplicateurs de Lagrange, le lagrangien est :

$$L(w, b, \xi, \mu, \nu) = \underbrace{\frac{1}{2}||w||^2 + C\sum_i \xi_i}_{\texttt{critère}} - \underbrace{\sum_i \mu_i(\xi_i + y_i(w^T x_i + b) - 1) - \sum_i \nu_i \xi_i}_{\texttt{contraintes}}$$

- Même méthode de résolution que le cas linéairement séparable

- Bonne nouvelle ! ! ! le cas non linéairement séparable s'obtient en imposant de plus la borne supérieure C sur les μ_i : $0 \leq \mu_i \leq C$

$$\begin{array}{lcl@{\qquad}lcl} \mu^* & = & \arg\min_\mu[\frac{1}{2}\mu^T K \mu - 1^T \mu] & K_{ij} & = & y_i y_j x_i^T x_j \\ w^* & = & \sum_{s \in Sv} \mu_s^* y_s x_s & 0 < & \mu^*_{s \in Sv} & \leq C \\ b^* & = & \frac{1}{|Sb|}\sum_{s \in Sb}(y_s - w^{*T} x_s) & 0 < & \mu^*_{s \in Sb} & < C \end{array}$$

- Modèle du classifieur à marge optimale identique à celui du cas linéairement séparable :

$$f(x|w^*, b^*) = \texttt{signe}(w^{*T} x + b^*)$$

- Marge s'obtient en prenant en compte les *slack variables* ξ_i^* et l'hyperparamètre C :

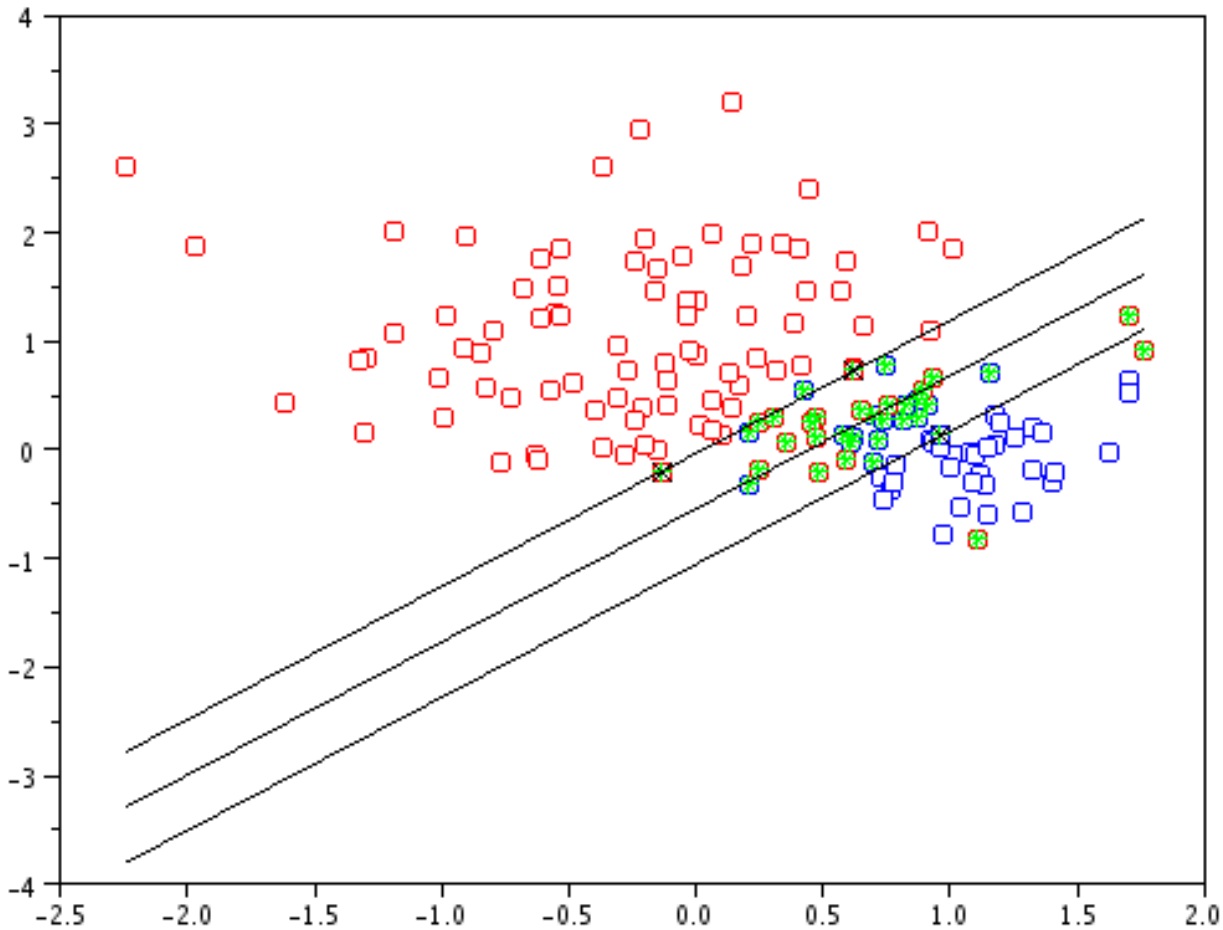
$$\boxed{\frac{1}{||w^*||} = \frac{1}{\sqrt{\sum_{i \in Sv}(\mu_i^* - C\xi_i^*)}} \text{ où } \xi_i^* = \max(0, 1 - y_i(w^{*T} x_i + b^*))}$$

- Remarque : si le problème est linéairement séparable $\xi_i = 0$, on retrouve le SVM du cas linéairement séparable avec $0 \leq \mu^* < C = \infty$.

Echantillon non linéairement séparable

Classe 1 : 100 points $\sim \mathcal{N}([0,1], 0.1)$ Classe 2 : 50 points $\sim \mathcal{N}([1,0], 0.05)$

Points supports pour le calcul de w^* dont 3 notés $\otimes$ pour le calcul de b^*



Introduction aux SVM non linéaires

• Le classifieur LINEAIRE a été défini à partir de produits scalaires défini dans l'espace des entrées entre le point x et les points supports x_s.

$$f(x|w^*, b^*) = \texttt{signe}({w^*}^T x + b^*) = \texttt{signe}(\sum_s \mu_s^* y_s x_s^T x + b^*)$$

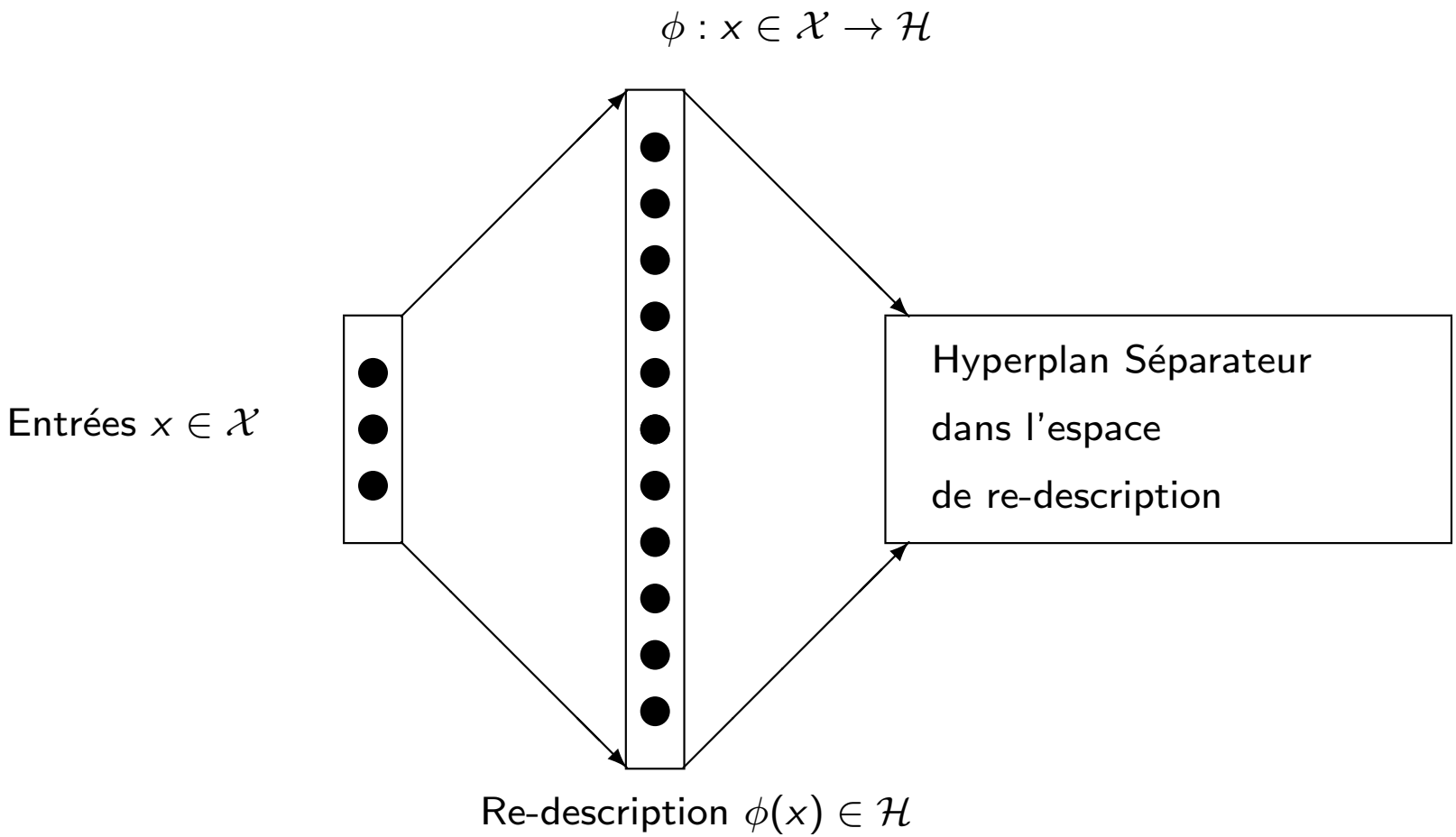
• Nous avons vu pour l'exemple du XOR (problème non linéairement séparable dans $\mathbb{R}^2$) que des transformations polynomiales des entrées

$$x = (x(1), x(2)) \rightarrow \phi(x) = (x(1), x(2), x(1)^2, x(2)^2, x(1)x(2))$$

permettent la construction d'un séparateur linéaire dans l'espace transformé de dimension 5.

Espace de re-description

Choix d'une application $\phi()$ de l'espace des entrées vers un espace de re-description (généralement de plus grande dimension) de façon à *linéariser* le classifieur



Introduction aux SVM non linéaires (suite)

• Pour l'exemple du XOR, considérons la transformation polynomiale suivante de degré 2 :

$$x = \underbrace{(x(1), x(2))}_{\mathcal{X}\text{: espace des entrées}} \rightarrow \phi(x) = \underbrace{(1, \sqrt{2}x(1), \sqrt{2}x(2), x(1)^2, x(2)^2, \sqrt{2}x(1)x(2))}_{\text{un espace de redescription}}$$

• En retenant le noyau $k(x_i, x_j) = (1 + x_i^T x_j)^2$ pour obtenir des fonctions polynomiales de degré 2 dans $\mathbb{R}^2$ (5 coefficients), on a la propriété suivante :

$$\phi(x_i)^T \phi(x_j) = (1 + x_i^T x_j)^2$$

• L'intérêt de cette approche est de remplacer le calcul des produits scalaires dans l'espace de re-description (grande dimension) par une fonction $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$: de plus faible dimension et sans expliciter la transformation $\phi(x)$:

$$k(x_i, x_j) = (1 + x_i^T x_j)^2$$

• Par exemple si $x \in \mathbb{R}^p$, le polynôme multidimensionnel de degré d comporte $(p + d)!/p!d!$ monômes (nombre de composantes de ϕ). L'évaluation du produit scalaire $\phi(x_i)^T \phi(x_j)$ est beaucoup plus couteuse que celle de la fonction $k(x_i, x_j) = (1 + x_i^T x_j)^d$.

• Dans le cas de MNIST, les images des chiffres sont codées sur des matrices 28×28, c'est à dire par $p = 784$ composantes. Si on retient un polynôme de dégré $d = 3$, les polynômes $\phi(x)$ ont 80931145 composantes à comparer aux produits scalaires entre vecteurs de 784 composantes ! ! !

Introduction aux méthodes à noyaux

• On se restreint à une classe particulière pour les fonctions k, les noyaux symétriques définis positifs (s.d.p.)

• Un noyau défini positif sur un ensemble $\mathcal{X}$ est une fonction $k()$ qui associe à 2 objets de $\mathcal{X}$ un nombre réel représentant leur *similarité* :

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

• On exige la symétrie et la positivité. Pour toute séquence finie $x_1, x_2, \ldots, x_n$, la matrice K d'éléments $k(x_i, x_j)$, matrice de Gram est définie positive, c'est à dire $\forall \beta \in \mathbb{R}^n : \beta^T K \beta \geq 0$.

• Quelques exemples de noyaux définis positifs pour $\mathcal{X} = \mathbb{R}^p$

- linéaire $k(x, y) = x^T y$,
- polynomial $k(x, y) = (1 + x^T y)^d$
- gaussien $k(x, y) = \exp[-\alpha ||x - y||_2^2]$ appelé également RBF (Radial Basis Functions)

RKHS : Espaces de Hilbert à noyaux reproduisants

- Un noyau défini positif k permet de construire un espace de fonctions.
- Muni d'un produit scalaire, c'est un espace de Hilbert $\mathcal{H}_k$ à noyau reproduisant appelé RKHS, Reproducing Kernel Hilbert Space
- Le RKHS se définit très simplement par ses éléments (des fonctions) et par un produit scalaire (entre fonctions).
  1. $\forall x \in \mathcal{X},\ k_x = k(x, .) \in \mathcal{H}_k$
  2. $< k_x, k_y >= k(x, y)$
- Exemples pour $x = a \in \mathbb{R}^p$
  1. $k(x, y) = x^T y \rightarrow$ fonctions affines $k_a(y) = \sum_{i=1}^{p} y_i a_i$
  2. $k(x, y) = (1 + x^T y)^d \rightarrow$ polynômes multi-dimensionnels $k_a(y) = (1 + \sum_{i=1}^{p} y_i a_i)^d$
  3. $k(x, y) = \exp[-\alpha ||x - y||_2^2] \rightarrow$ noyaux gaussiens $k_a(y) = \exp[-\alpha \sum_{i=1}^{p} (y_i - a_i)^2]$
- Intérêt est de pouvoir travailler sur des espaces de très grande dimension (polynômes), voire de dimension infinie (noyaux gaussiens, de Laplace, ...).

Espaces de Hilbert à noyaux reproduisants (suite)

- A tout $x \in \mathcal{X}$ est associé un élément $\phi_x \in \mathcal{H}_k$:

$$\begin{array}{ccc} \mathcal{X} & \rightarrow & \mathcal{H}_k \\ \phi_x & = & k(x, .) \end{array}$$

- Propriété du kernel trick.

$$< \phi_{x_i}, \phi_{x_j} >_{\mathcal{H}_k} = k(x_i, x_j)$$

  - Le choix d'un noyau défini positif permet donc de travailler dans des espaces de grande dimension (même infinie) de façon à *linéariser* la séparatrice
  - Pas nécessaire d'expliciter les fonctions de l'espace de re-description grâce au kernel trick
  - On reprend donc les formules établies pour le SVM linéaire et il suffit de remplacer les produits scalaires $x_i^T x_j$ par $< \phi_{x_i}, \phi_{x_j} >_{H_k}$ c'est à dire par $k(x_i, x_j)$

SVM à noyaux (1)

- La solution s'exprime comme dans le cas linéaire à partir de la résolution d'une minimisation quadratique sous contraintes linéaires.

$$\begin{aligned} \mu^* &= \arg\min_{\mu} \frac{1}{2} \sum_{i,j} \mu_i \mu_j y_i y_j < \phi_{x_i}, \phi_{x_j} >_{\mathcal{H}_k} - \sum_i \mu_i \\ 0 &\leq \mu_i \leq C \quad , \quad \sum_i \mu_i y_i = 0 \end{aligned}$$

- Et via le kernel trick les produits scalaires sont remplacés par les éléments de la matrice de similarité

$$\begin{aligned} \mu^* &= \arg\min_{\mu} \frac{1}{2} \sum_{i,j} \mu_i \mu_j y_i y_j k(x_i, x_j) - \sum_i \mu_i \\ &= \arg\min_{\mu} [\frac{1}{2} \mu^T K \mu - 1^T \mu] \end{aligned}$$

- On retrouve la formulation du SVM linéaire ! ! ! via une *simple* optimisation convexe

SVM à noyaux (2)

- Formalisation du modèle de prédiction via le kernel :

$$\begin{aligned} f(x|w^*, b^*) &= \texttt{signe}(< w^*, \phi_x >_{\mathcal{H}_k} + b^*) \\ &= \texttt{signe}(< \sum_{s \in Sv} \mu_s^* y_s \phi_{x_s}, \phi_x >_{\mathcal{H}_k} + b^*) \\ &= \texttt{signe}(\sum_{s \in Sv} \mu_s^* y_s < \phi_{x_s}, \phi_x >_{\mathcal{H}_k} + b^*) \\ &= \texttt{signe}(\sum_{s \in Sv} \mu_s^* y_s k(x_s, x) + b^*) \end{aligned}$$

- Le seuil b^* est défini par les points supports $Sb \subset Sv$ vérifiant les prédictions. Soit $x_s \in Sb$ un des ces points, on a

$$b^* = y_s - < w^*, \phi_{x_s} >_{\mathcal{H}_k} = y_s - \sum_{k \in Sv} \mu_k^* y_k k(x_k, x_s)$$

  ou bien par la moyenne de tous ces points (plus de précision numérique) :

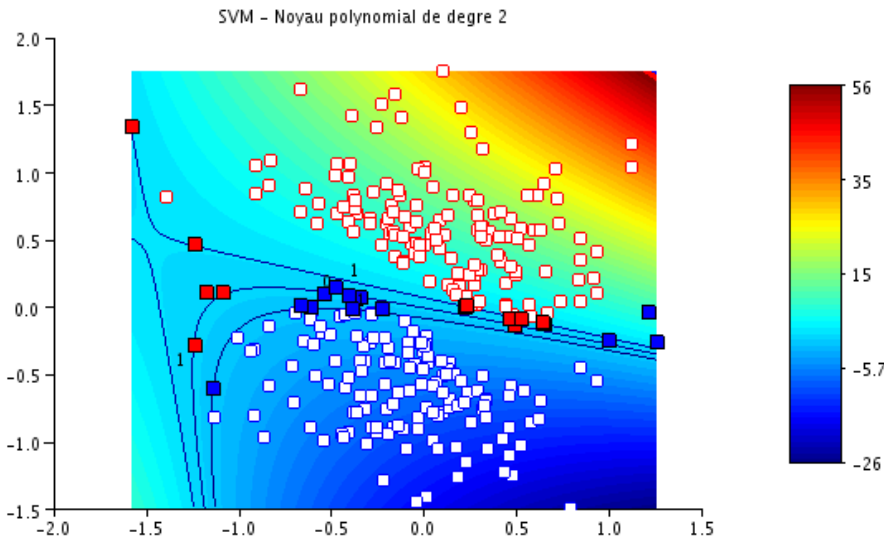
$$b^* = \frac{1}{|Sb|} \sum_{s \in Sb} (y_s - \sum_{k \in Sv} \mu_k^* y_k k(x_k, x_s))$$

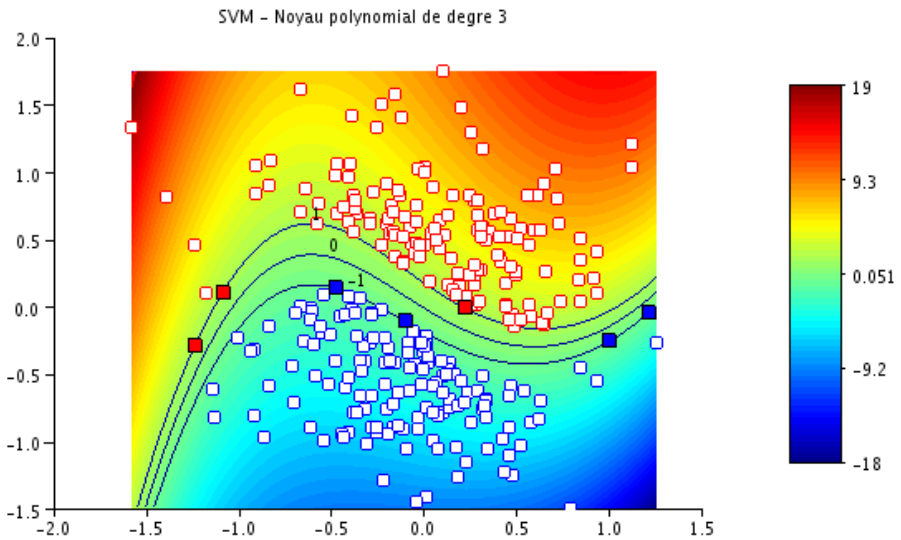
- Toute prédiction en x s'exprime à partir d'une combinaison linéaire des similarités entre x et les points supports x_s. En notant $\alpha_s^* = \mu_s^* y_s$:

$$f(x|\alpha^*, b^*) = \texttt{signe}(\sum_{s \in Sv} \alpha_s^* k(x_s, x) + b^*)$$

Noyau polynomial

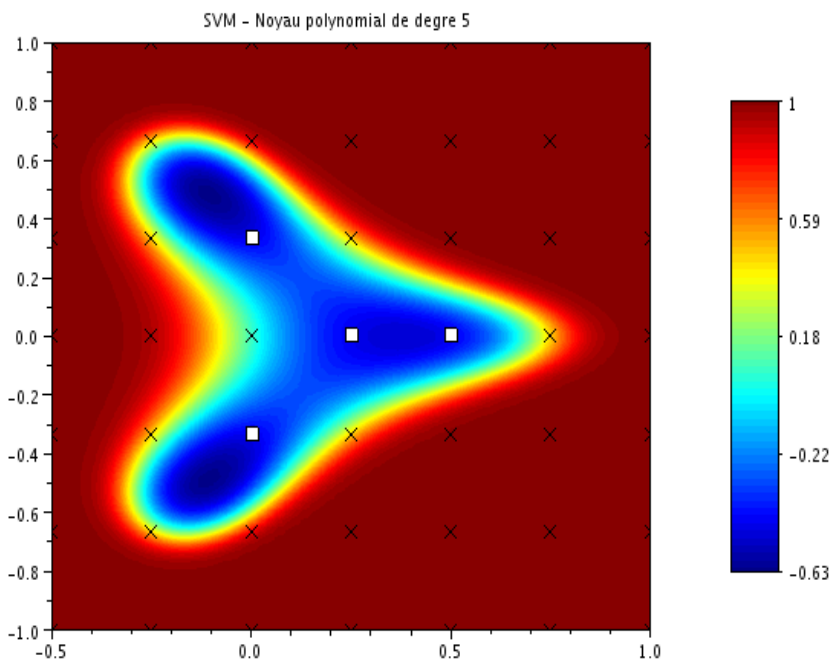
Points de la classe 1 : $y = +1$, points de la classe 2 : $y = -1$

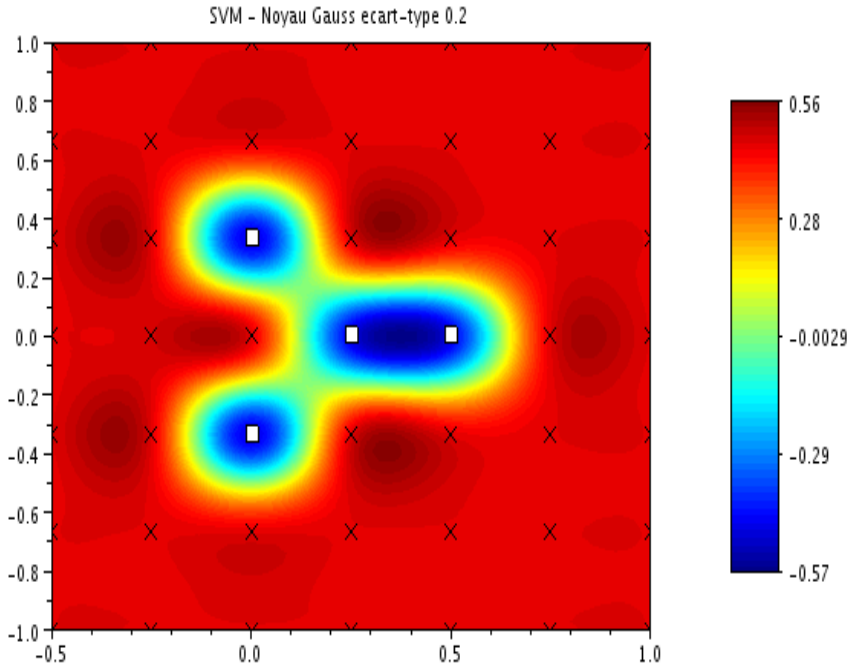




Noyau gaussien

- 2 classes imbriquées, une définie par 4 points (*square*, $y = -1$) contenue dans l'autre définie par les points (*cross*, $y = +1$)
- La visualisation porte sur les sorties transformées par la fonction tangente hyperbolique $y \rightarrow (1 - e^{-y})/(1 + e^{-y})$





Exemple de codage avec Scikit-Learn (S. Gazut CEA)

- Données d'apprentissage : Xtrain, Ytrain

```
classifier = SVC(kernel='poly', degree=3, gamma='auto', probability=True, C=100, coef0=1)
classifier = classifier.fit(Xtrain, Ytrain)
```

- Pour un choix des hyper-paramètres via *grid search*

```
parameters = {"kernel": ["poly", "poly", "poly"], 'degree': [3, 4, 5], "C": [0.1, 0.5, 1, 2, 10, 20, 50, 100]}
svmc = SVC(gamma='auto')
grille = GridSearchCV(estimator=svmc, param_grid=parameters, scoring="accuracy")
results = grille.fit(X, Y)

classifier = SVC(kernel=results.best_params_['kernel'],
            degree=results.best_params_['degree'], gamma='auto',
            probability=True, C=results.best_params_['C'], coef0=1)
```

- Prédictions pour de nouveaux points : Xnew

```
Ypred      = classifier.predict(Xnew)
```

- coef0 est un paramètre spécifique aux noyaux de type polynôme et sigmoïde. Par exemple, pour le noyau polynomial il correspond à la formulation suivante : k(x,y) = $(\text{coef0} + x^T y)^d$. Par défaut, il est fixé à 0 (noyau dit homogène).
- Probability = True, correspond au fait que la probabilité d'appartenance à la classe 1 sera estimée en plus de la valeur de prédiction du SVM. Il faut définir ce paramètre avant d'appeler la méthode fit. Globalement, cette probabilité d ?appartenance est estimée via la méthode de *platt scaling* qui correspond à un rescaling en appliquant une logistique en sortie.
- gamma = 'auto', le paramètre γ est utilisé pour pondérer l'écart entre les feature vectors notamment dans le noyau gaussien $k(u, v) = exp(-\gamma||u - v||^2)$. La valeur optimale de cet hyper-paramètre dépend du jeu de données (taille de l'espace, normalisation des données).

- Retenir
  - SVM outil efficace notamment en classification supervisée
  - Utilisation des *méthodes à noyaux* pour representer tout type de fonctions

- Références
  - Sites : http :/www.svms.org/tutorials et http :/www.support-vector-machines.org où on trouve de nombreux tutorials et références comme ceux de B. Scholkopf, des cours *matheux* sur les méthodes à noyaux comme ceux de J.Ph. Vert.
  - Librairie LIBSVM, Python Scikit-learn
  - A. Géron, Machine Learning avec Scikit-Learn, Dunod, 2019