Une introduction aux machines à vecteurs supports (SVM)

Plan

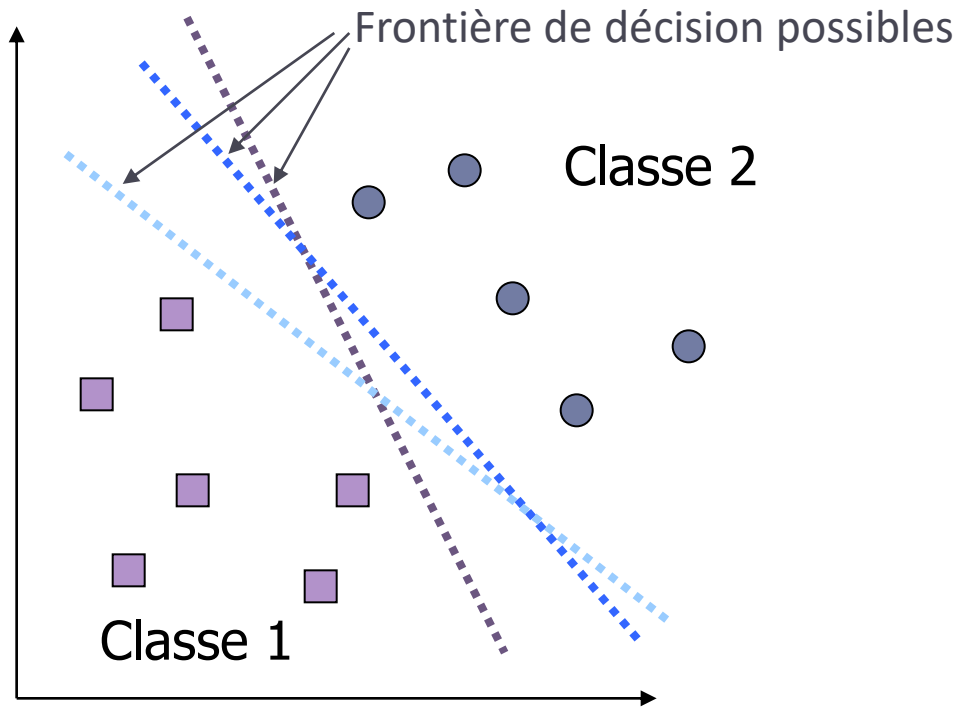
- Historique
- Quelle est la bonne frontière de séparation pour deux classes linéairement séparables ?
  - La solution SVM
- Adaptation aux cas non linéairement séparables: l'astuce des fonctions noyau
- Exemples d'application
- Conclusion

Historique du SVM

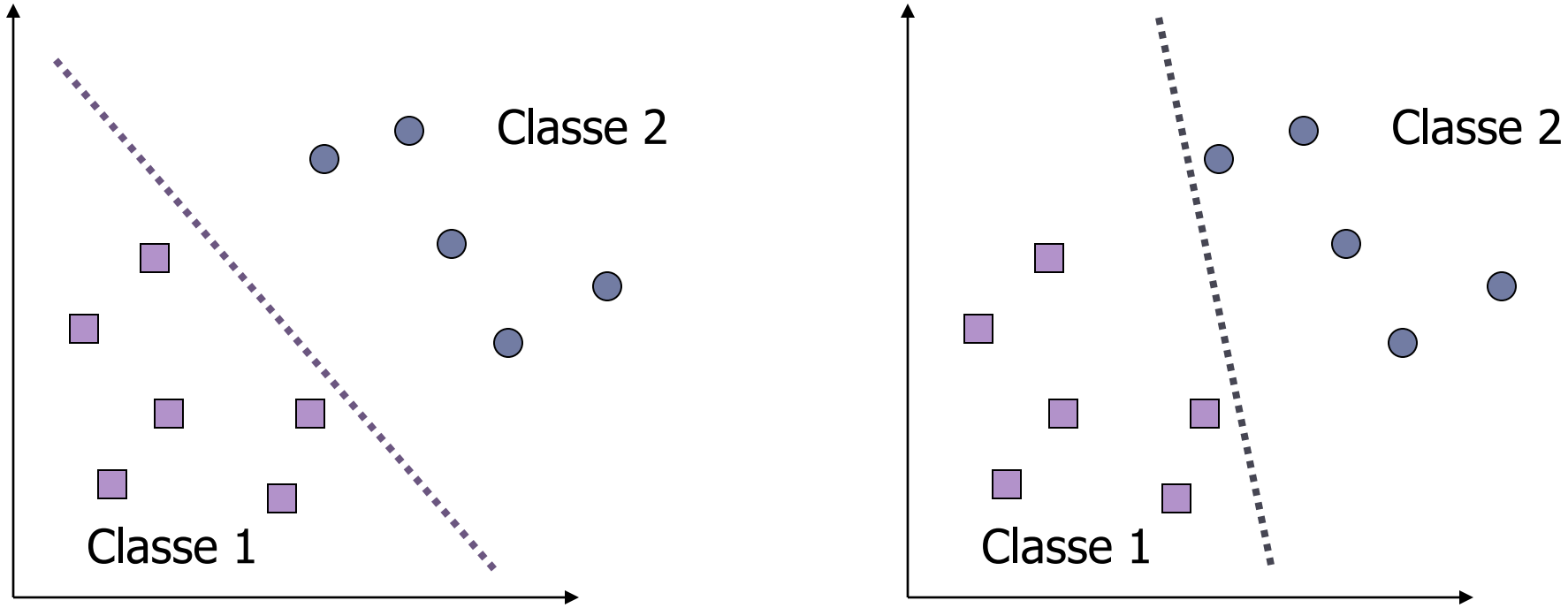
- Classifieur dérivé de la théorie de l'apprentissage statistique par Vapnik et Chervonenkis (~1994)
- Devenu populaire depuis qu'il a permis des performances égales ou meilleures aux RNA pour reconnaitre l'écriture manuscrite en partant d'images formées de pixels.
- Proche de :
  - Séparateurs à vastes marges
  - Méthodes à fonctions noyau
  - Réseaux de neurones à bases radiales

Problème à deux classes linéairement séparables

- Plusieurs surfaces de décision existent pour séparer les classes ; laquelle choisir ?

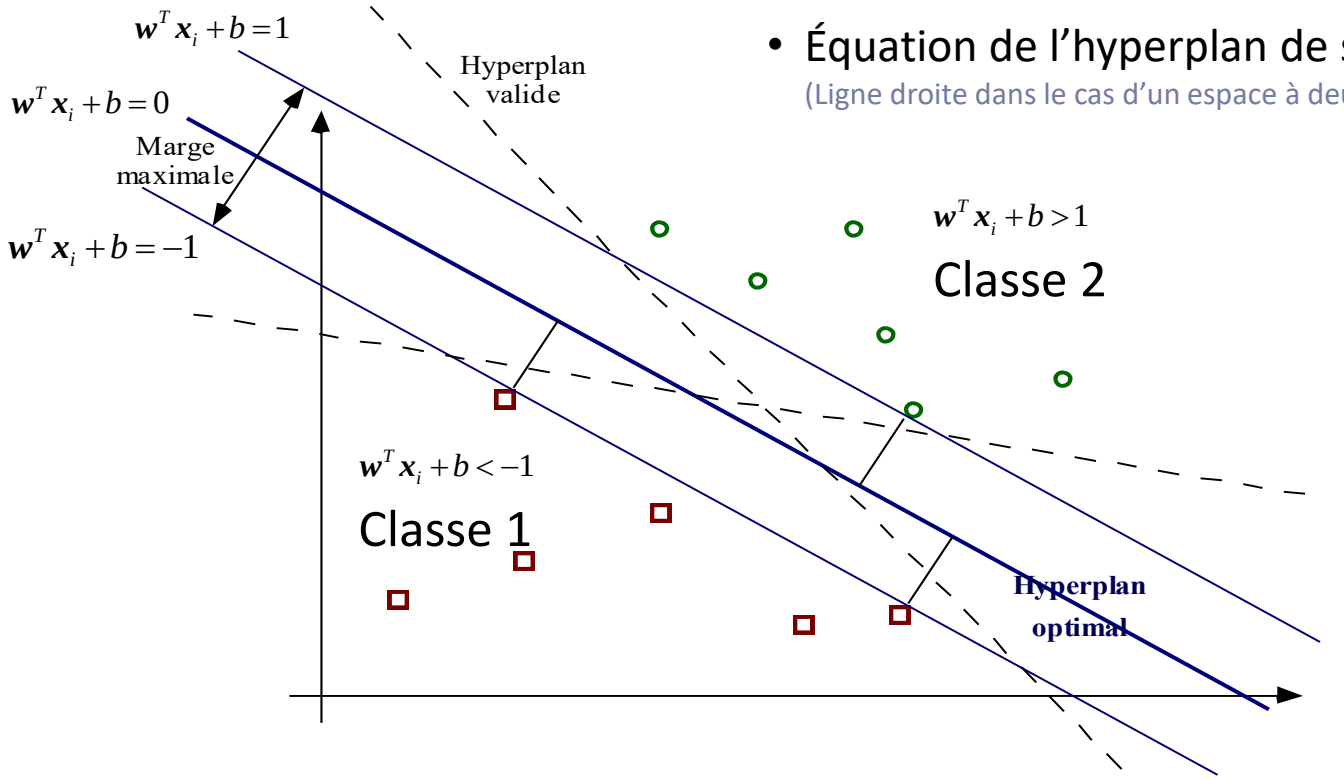


Exemples de mauvais choix



- Pour minimiser la sensibilité au bruit, la droite de décision doit être aussi éloignée que possible des données de chaque classe

Hyperplan de plus vaste marge



- Équation de l'hyperplan de séparation : $y = w^T x + b$
  (Ligne droite dans le cas d'un espace à deux dimensions)

- Si $\{x_1, ..., x_n\}$ est l'ensemble des données et $y_i \in \{1,-1\}$ est la classe de chacune, on devrait avoir :

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \qquad \forall i$$

  tout en ayant une distance optimale entre chaque x_i et le plan de séparation

Optimisation de la marge

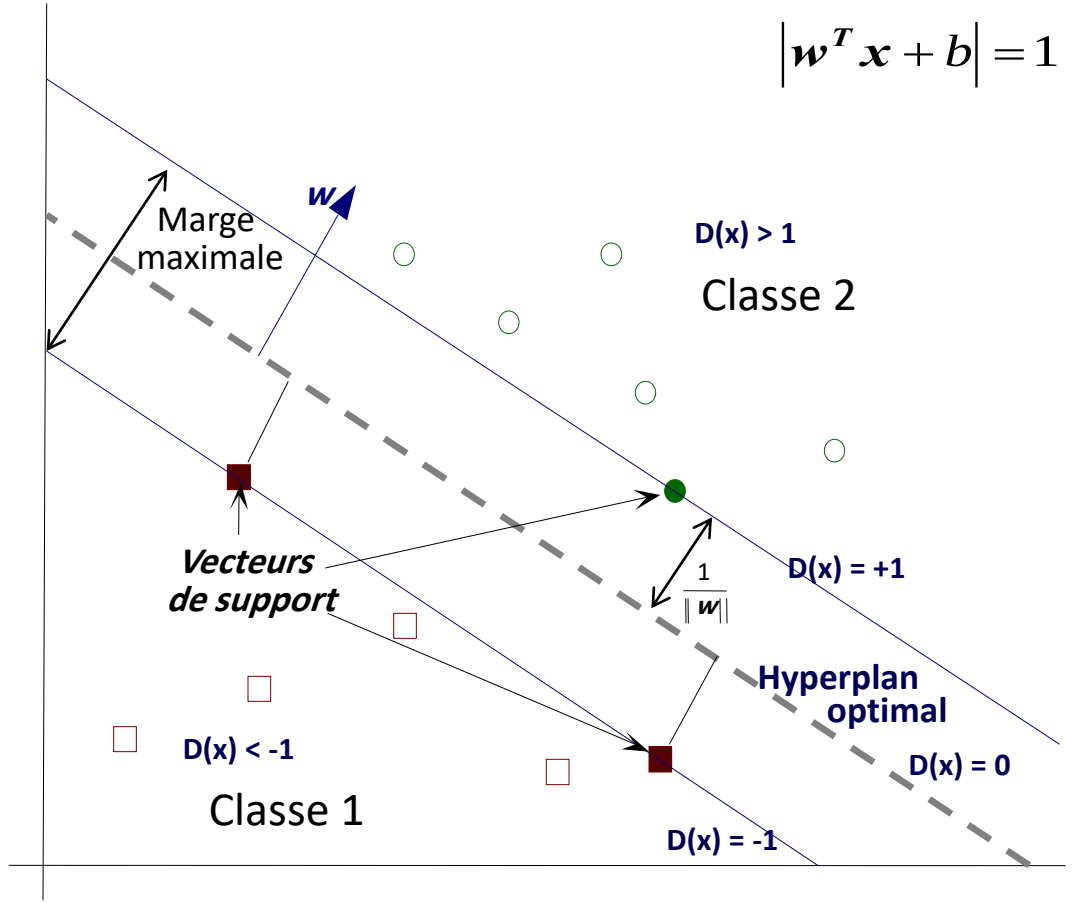
- Distance normale d'un point à l'hyperplan :

$$D(\boldsymbol{x}) = \frac{\left|\boldsymbol{w}^T \boldsymbol{x} + b\right|}{\|\boldsymbol{w}\|}$$

- Marge max. avant d'atteindre les frontières des deux classes ($\left|\boldsymbol{w}^T \boldsymbol{x} + b\right| = 1$) :

$$m = \frac{2}{||\mathbf{w}||}$$

- Maximiser m revient à minimiser $\|\boldsymbol{w}\|$ tout en préservant le pouvoir de classification :



$$min \frac{1}{2} \|\boldsymbol{w}\|^2 \; sous \; la \; contrainte : y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \geq 1, \forall i$$

Problème d'optimisation quadratique

- Maximiser le pouvoir de généralisation du classeur revient à trouver $\boldsymbol{w}$ et b tels que :

$$\frac{1}{2}\|\boldsymbol{w}\|^2 \quad \text{est minimum}$$

et

$$y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i+b\right)\geq 1, \qquad i=1,\ldots,n$$

- Si d est la dimension des $\boldsymbol{x}_i$ (nombre d'entrées), cela revient à régler d+1 paramètres (les éléments de $\boldsymbol{w}$, plus b)
  - Possible par les méthodes d'optimisation classiques (e.g., optimisation quadratique) seulement si d pas trop grand (< qqs 10^3)
  - L'approche SVM utilise les multiplicateurs de Lagrange pour une solution plus simple

Les multiplicateurs de Lagrange en 30 s

- On veut maximiser ou minimiser $f(x)$ sous la contrainte $g(x)=0$
- Solutions possibles :
  - Résoudre $g(x)=0$ et substituer la/les racines trouvées dans $f(x)$; résoudre alors $f'(x) = 0$: pas toujours facile !
  - Considérer que $f(x)$ et $g(x)=0$ évoluent pareillement au voisinage de l'extrémum recherché. Leurs tangentes sont alors colinéaires et :

    $f'(x) = \alpha g'(x)$ (ou $f'(x) - \alpha g'(x) = 0$), α étant à déterminer

    - La méthode des multiplicateurs de Lagrange regroupe la fonction a optimiser et la contrainte en une seule fonction $\Lambda(x,\alpha)= f(x)-\alpha g(x)$
      - La solution de $d\Lambda(x,\alpha)/dx = 0$ donne le point où les tangentes sont colinéaires; en même temps, $d\Lambda(x,\alpha)/d\alpha = 0$ répond à la contrainte.

Cas à plusieurs dimensions

- On veut trouver le minimum (ou maximum) d'une fonction $f(\boldsymbol{x})$ en respectant un nombre de contraintes $g_i(\boldsymbol{x})=0,\ i=1,\ldots,n$
- On peut monter qu'à l'extrémum recherché : $\nabla f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \nabla g_i(\boldsymbol{x})$, (colinéarité des tangentes)

  ou encore $\nabla f(\boldsymbol{x}) - \sum_{i=1}^{n} \alpha_i \nabla g_i(\boldsymbol{x}) = 0$, pour des coefficients α_i à déterminer

  - Si on forme la fonction (lagrangien) : $\Lambda(\boldsymbol{x},\alpha) = f(\boldsymbol{x}) - \sum_{i=1}^{l} \alpha_i g_i(\boldsymbol{x})$

    Alors : $\nabla_x \Lambda(\boldsymbol{x},\alpha) = \nabla f(\boldsymbol{x}) - \sum_{i=1}^{l} \lambda_i \nabla g_i(\boldsymbol{x})$

    et $\nabla_{\alpha_i} \Lambda(\boldsymbol{x},\alpha) = g_i(x)$

    => la solution de $\nabla_x \Lambda(\boldsymbol{x},\alpha) = 0$ mène à un extrémum qui respecte les contraintes.

Forme duale du problème d'optimisation

- Ici, la fonction à optimiser sous contraintes est celle de l'hyperplan de séparation à marge maximale, qui est défini par les paramètres ($\boldsymbol{w}$, b) correspondants.
- Donc, partant d'un ensemble de donnes données $\{(\boldsymbol{x}_i, y_i)\}$, de l'ensemble de contraintes $\{(\boldsymbol{x}_i^T \boldsymbol{w}+b)y_i - 1 = 0\}$ et des paramètres à optimiser ($\boldsymbol{w}$, b), on a:

$$\begin{cases} \Lambda(\boldsymbol{w}, b, \alpha) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \alpha_i \{(\boldsymbol{x}_i^T \boldsymbol{w}+b)y_i - 1\} \\ \forall i \quad \alpha_i \geq 0 \end{cases}$$

Le minimum recherché est donné par la solution de $\nabla_{w,b}\Lambda(\boldsymbol{w}, b, \boldsymbol{\alpha}) = 0$

- Il existe une formulation duale du problème plus facile à résoudre :
  - Théorème de Kuhn-Tucker : $\min_{\boldsymbol{\alpha}} x\left(\max_{\boldsymbol{x}} \Lambda(\boldsymbol{x}, \boldsymbol{\alpha})\right) = \max_{\boldsymbol{x}}\left(\min_{\boldsymbol{\alpha}} \Lambda(\boldsymbol{x}, \boldsymbol{\alpha})\right)$

  ⇒ Solution alternative pour trouver $\boldsymbol{w}$ et b :

  résoudre $\nabla_{\boldsymbol{\alpha}}\Lambda(\boldsymbol{w}, b, \boldsymbol{\alpha}) = 0$ sous la contrainte $\nabla_{w,b}\Lambda(\boldsymbol{w}, b, \boldsymbol{\alpha}) = 0$

Forme duale du problème d'optimisation

- On résout l'équation du lagrangien par rapport à $\boldsymbol{\alpha}$ au lieu de $\boldsymbol{w}, b$
- Avantage de résoudre $\nabla_{\alpha}\Lambda(\boldsymbol{w}, b, \boldsymbol{\alpha}) = 0$ au lieu de $\nabla_{w,b}\Lambda(\boldsymbol{w}, b, \boldsymbol{\alpha}) = 0$:
  - La complexité du problème d'optimisation devient proportionnelle à n (nombre de paires d'apprentissage ($\boldsymbol{x}_i$, y_i)) et non d (la dimension de $\boldsymbol{w}$ qui est donnée par celle des $\boldsymbol{x}_i$)
  - Possible d'obtenir des solutions pour des problèmes impliquant ≈ 10^5 exemples
  - Problème pour lequel le maximum global des α_i peut toujours être trouvé

Formulation du problème dual

- Partant de $\begin{cases} \Lambda(\boldsymbol{w},b,\boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \alpha_i\{(\boldsymbol{x}_i^T \boldsymbol{w}+b)y_i - 1\} \\ \forall i \quad \alpha_i \geq 0 \end{cases}$

$\nabla_{w,b}\Lambda(\boldsymbol{w},b,\boldsymbol{\alpha}) = 0$ donne $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i$

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

- On a par substitution dans $\Lambda(\boldsymbol{w},b,\boldsymbol{\alpha})$:

$$\begin{cases} \Lambda(\boldsymbol{w},b,\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \left(\boldsymbol{x}_i^T \boldsymbol{x}_j\right) \\ \forall i \quad \alpha_i \geq 0 \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases}$$

- Il faut trouver $\boldsymbol{\alpha}$ qui maximise $\Lambda(\boldsymbol{w},b,\boldsymbol{\alpha})$: résoudre $\nabla_{\alpha}\Lambda(\boldsymbol{w},b,\boldsymbol{\alpha}) = 0$
  - n équations linéaires homogènes à n inconnues (les composants de $\boldsymbol{\alpha}$)

Solution du problème d'optimisation

$$\begin{cases} \hat{\boldsymbol{w}} = \sum_{i=1}^{n_s} \hat{\alpha}_i y_i \boldsymbol{x}_i \\ \hat{b} = y_s - \sum_{i=1}^{n_s} \hat{\alpha}_i y_i (\boldsymbol{x}_i^T \boldsymbol{x}_s) \\ D(\boldsymbol{x}) = (\hat{\boldsymbol{w}}^T \boldsymbol{x} + \hat{b}) \end{cases}$$

- ^ : estimé
- n_S : nombre de vecteurs de support ($\boldsymbol{x}_i$ avec $\alpha \neq 0$)
- $(\boldsymbol{x}_S, y_S)$: vecteur de support arbitraire (pour trouver $\hat{b}$)

- Les données $\boldsymbol{x}_i$ avec $\alpha \neq 0$ sont appelées vecteurs de support. Ils correspondent aux points les plus proches de la surface de séparation
- Dans l'expression du Lagrangien pour déterminer $\hat{\boldsymbol{w}}$ et $\hat{b}$, *les donnés $\boldsymbol{x}$ apparaissent uniquement pour former des produits scalaires*

$$\Lambda(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \left(\boldsymbol{x}_i^T \boldsymbol{x}_j \right)$$

Caractéristiques de la solution

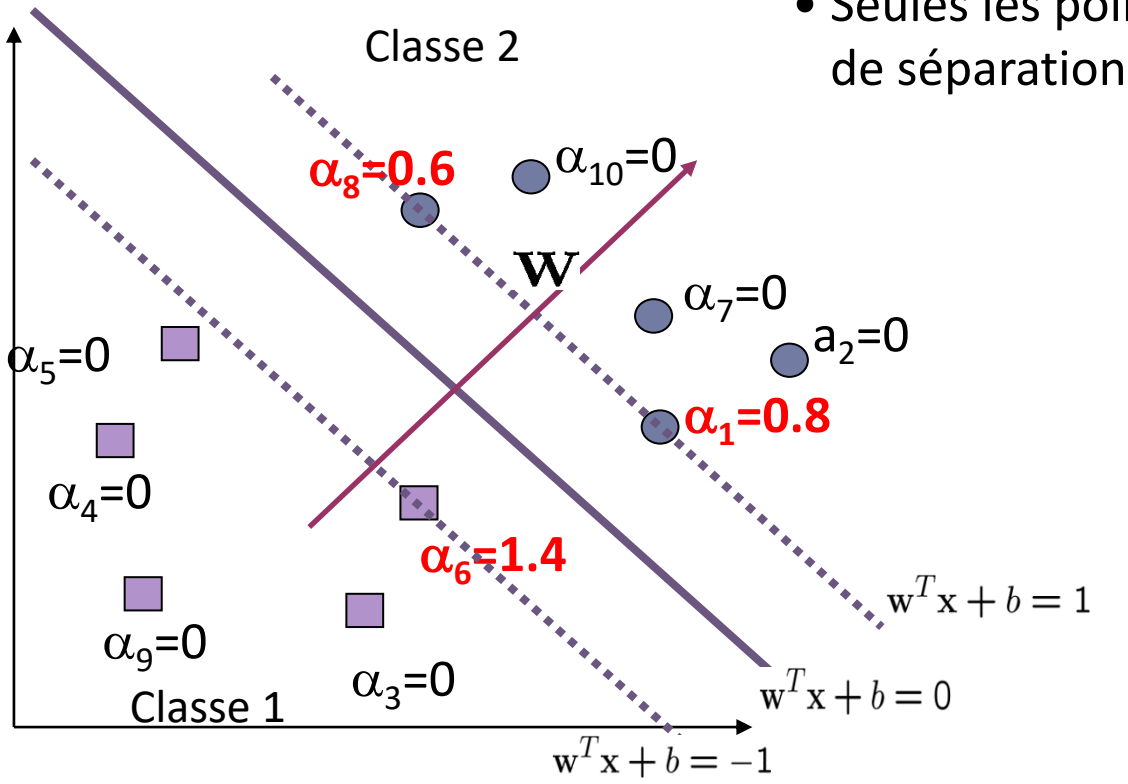
- Puisque plusieurs α_i sont nuls, $\boldsymbol{w}$ est une combinaison linéaire d'un petit nombre de données
- La surface de décision est uniquement déterminée par les n_s vecteurs de support trouvés:

$$\hat{\boldsymbol{w}} = \sum_{i=1}^{n_s} \hat{\alpha}_i y_i \boldsymbol{x}_i$$

$$\hat{b} = y_s - \sum_{i=1}^{n_s} \hat{\alpha}_i y_i (\boldsymbol{x}_i^T \boldsymbol{x}_s)$$

- Pour classer une nouvelle donnée $\boldsymbol{z}$
  - Calculer $\hat{\boldsymbol{w}}^T \boldsymbol{z} + \hat{b} = \sum_{i=1}^{n_s} \hat{\alpha}_i y_i \left(\boldsymbol{x}_i^T \boldsymbol{z}\right) + \hat{b}$ et classer $\boldsymbol{z}$ dans la classe 1 si le résultat est positif et dans la classe 2 s'il est négatif

Interprétation géométrique

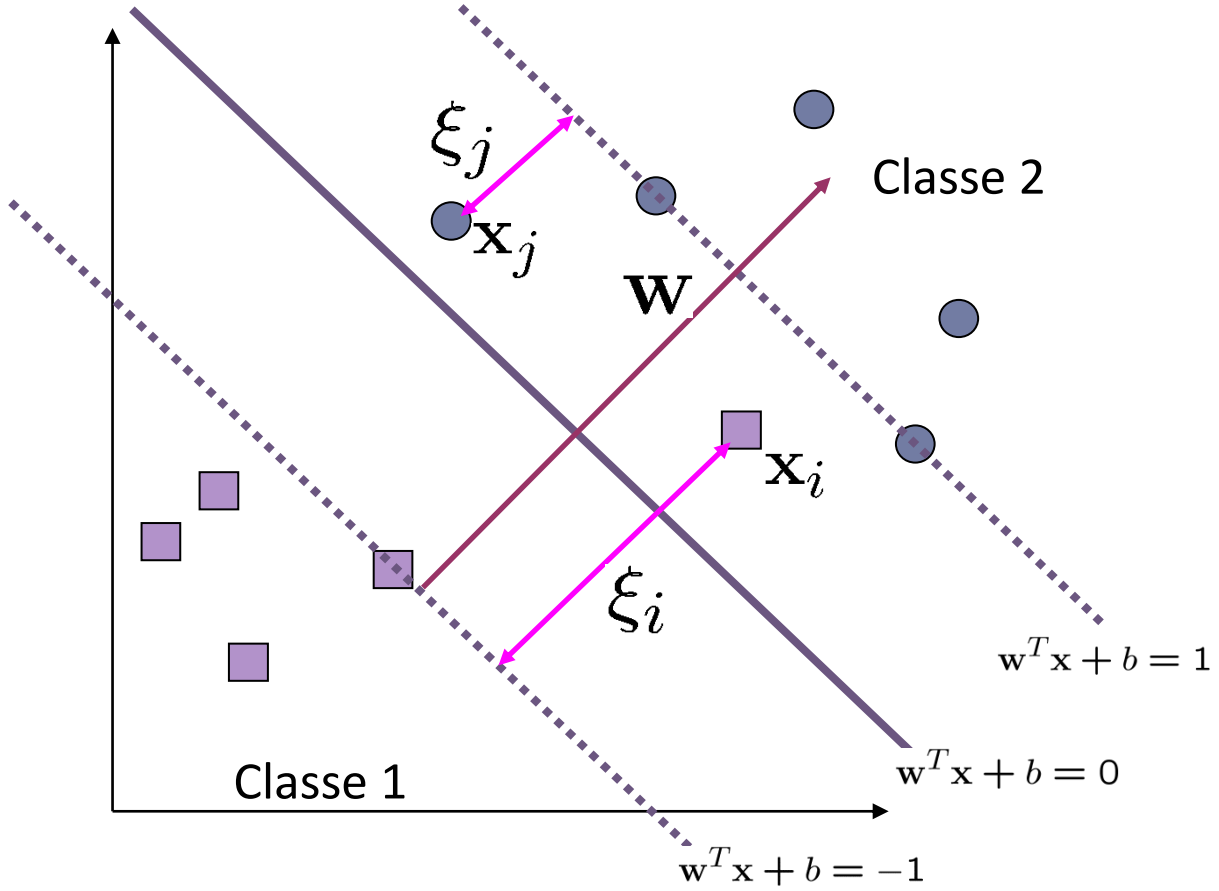


- Seules les points les plus proches de la surface de séparation influent sur sa définition

- Il existe des limites théorique pour l'erreur de classification de données nouvelles
  - Plus grande la marge, plus petite la limite
  - Plus petit le nombre de SV, plus petite la limite

Et pour un cas non linéairement séparable ?

- On peut introduire une marge d'erreur ξ_i pour la classification



Hyperplan à marges douces

- ξ_i= 0 s'il n'existe pas d'erreur pour x_i
  - ξ_i sont des variables qui donnent du "mou" aux marges optimales

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

- Nous voulons minimiser

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

  - C : paramètre de compromis entre l'erreur et la marge
- Le problème d'optimisation devient

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

Détermination de l'hyperplan de séparation

- La forme duale du problème est

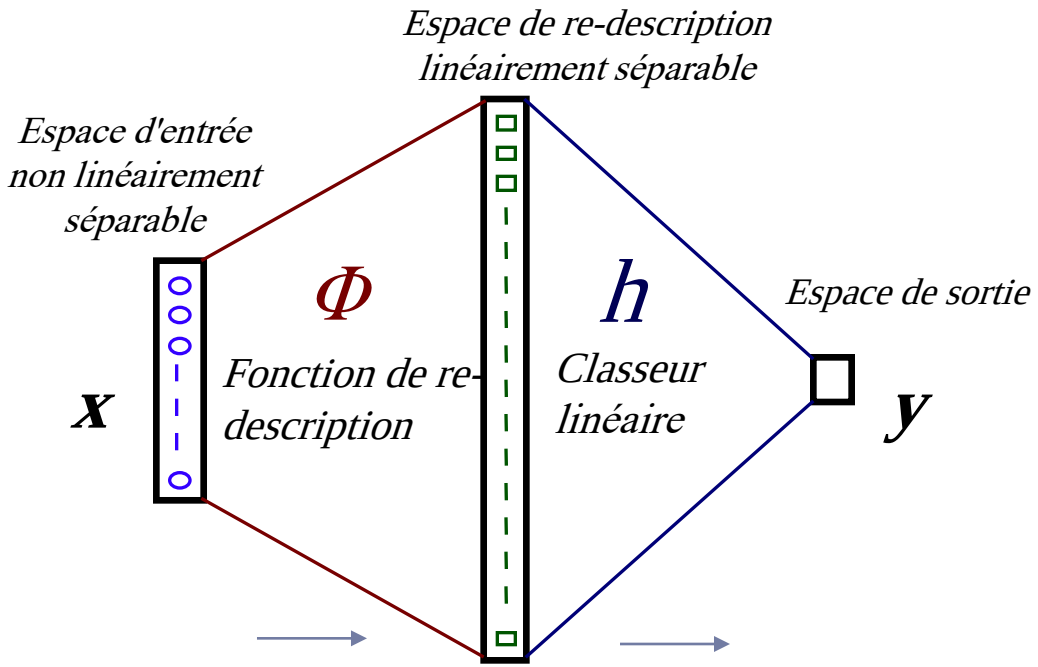
$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

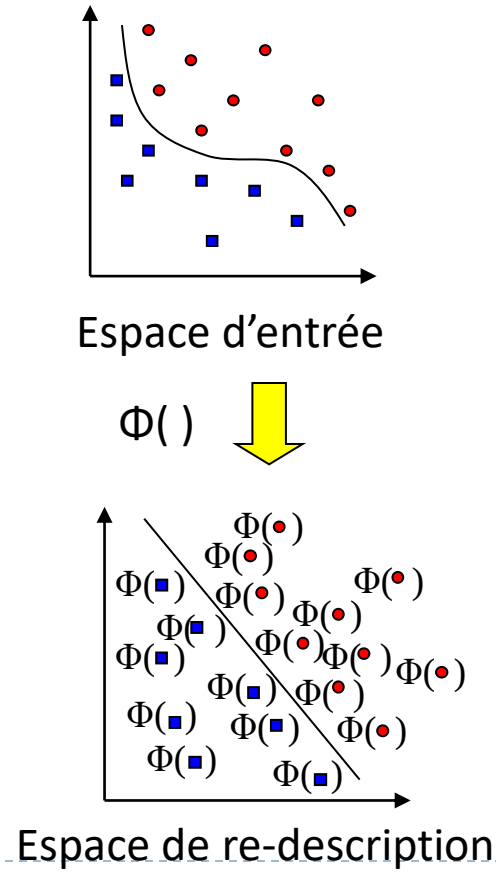
$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

- $\boldsymbol{w}$ est aussi donné par $\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$
- La seule différence avec le cas linéairement séparable est qu'il existe une limite supérieure C aux α_i

Extension à une surface de séparation non-linéaire

- « Simplifier les choses » en projetant les x_i dans un espace qui les rend linéairement séparables





Modification due à la transformation

- Substituer les arguments transformés dans les produits scalaires lors de la phase d'apprentissage,

Problème original :

$$\max. \; W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Après transformation :

$$\max. \; W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{\Phi}(\mathbf{x}_i)^T \boldsymbol{\Phi}(\mathbf{x}_i)$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

- Cependant, trouver $\Phi()$ n'est pas évident !

Modification due à la transformation

- Les nouvelles données $\boldsymbol{z}$ sont toujours classées dans la classe 1 si $f \geq 0$, la classe 2 sinon :

Original :

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

$$f = \mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^T \mathbf{z} + b$$

Après xformation :

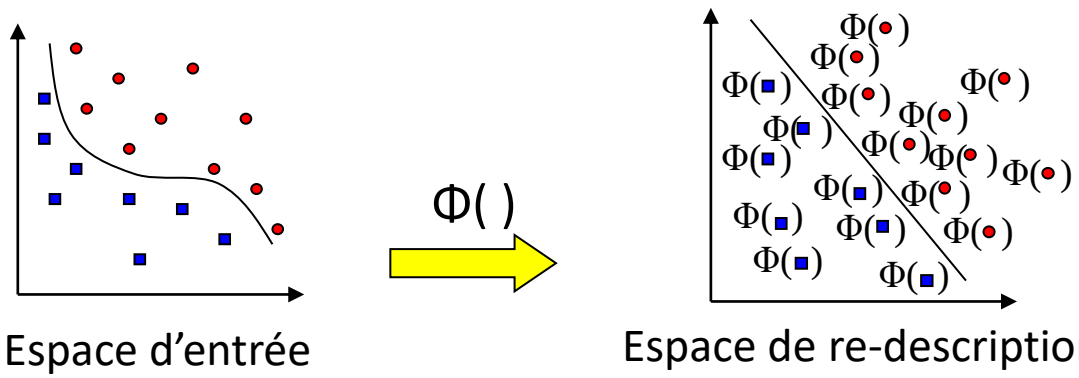
$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \phi(\mathbf{x}_{t_j})$$

$$f = \langle \mathbf{w}, \phi(\mathbf{z}) \rangle + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \boldsymbol{\Phi}(x_{t_j})^T \boldsymbol{\Phi}(z) + b$$

et la surface de séparation dans le nouvel espace est : $D(\boldsymbol{x}) = \sum_{j=1}^{s} \hat{\alpha}_j y_j \Phi(\boldsymbol{x}_j)^T \Phi(\boldsymbol{x}_j) + \hat{b}$

Extension à une surface de séparation non-linéaire

- Problèmes cependant :
  - Φ = ?
  - Grand effort de calcul potentiel (*d* explose !)
- SVM à fonctions noyaux résout les deux problèmes
  - Efficacité computationnelle
  - La transformation désirée des données est faite implicitement !



L'astuce des fonctions noyau

- Définition d'une fonction noyau :

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

  $\Rightarrow$ La connaissance de $K(\ )$ permet de calculer un produit scalaire où intervient $\Phi(\ )$ sans connaitre l'expression de $\Phi(\ )$

- Or, seuls des produits scalaires interviennent dans la solution du problème d'optimisation
  - Un autre avantage d'utiliser $K()$ est qu'il représente intuitivement la similarité entre les $\boldsymbol{x}$ et $\boldsymbol{y}$, obtenue de nos connaissances a priori
  - Cependant, $K(\boldsymbol{x},\boldsymbol{y})$ doit satisfaire certaines conditions (conditions de Mercer) pour que le $\Phi\ (\ )$ correspondant existe

Les conditions de Mercer

- Pour une fonction *K* symétrique, il existe une fonction Φ telle que :

$$K(\boldsymbol{x},\boldsymbol{x}') = \Phi(\boldsymbol{x}).\Phi(\boldsymbol{x}') = \sum_{i=1}^{m} g_i(\boldsymbol{x}).g_i(\boldsymbol{x}')$$

ssi, pour toute fonction *f* telle que : $\int f(\boldsymbol{x})^2 d\boldsymbol{x}$ *est fini*

l'on a : $\int K(\boldsymbol{x},\boldsymbol{x}')\ f(\boldsymbol{x}) f(\boldsymbol{x}')\ d\boldsymbol{x}\, d\boldsymbol{x}' \geq 0$

- Si cette condition est vérifiée, on peut appliquer la fonction noyaux dans le SVM
- MAIS cela ne dit pas comment construire Φ

Exemple illustratif

- Définissons la fonction noyau $K(\boldsymbol{x},\boldsymbol{y})$ telle que, pour toute paire de vecteurs $\boldsymbol{x}=(x_1, x_2)$ et $\boldsymbol{y}=(y_1, y_2)$:

$$K(\boldsymbol{x},\boldsymbol{y}) = (1 + x_1 y_1 + x_2 y_2)^2$$

- Considérons maintenant une transformation Φ qui prend un vecteur de dimension 2 et le projette dans un espace de dimension 6 :

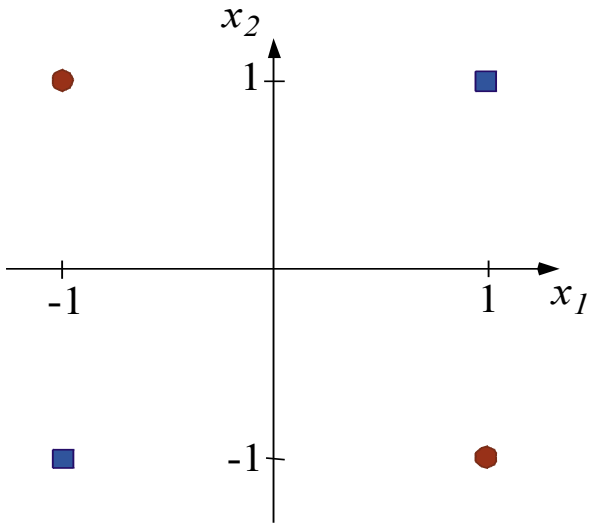
$$\Phi(\boldsymbol{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1 x_2)$$

On peut voir en effectuant le calcul que

$$\langle \Phi(\boldsymbol{x}), \Phi(y) \rangle = (1 + x_1 y_1 + x_2 y_2)^2$$
$$= K(\boldsymbol{x},\boldsymbol{y})$$

On peut donc obtenir le résultat sans avoir à passer par l'espace transformé!

Illustration : le cas du XOR



Index i	x_i	y
1	(1,1)	1
2	(1,-1)	-1
3	(-1,-1)	1
4	(-1,1)	-1

- Il faut résoudre :

$$\begin{cases} \max\limits_{\alpha}\left(\sum\limits_{i=1}^{4}\alpha_i - \frac{1}{2}\sum\limits_{i=1}^{4}\sum\limits_{j=1}^{4}\alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i,\boldsymbol{x}_j)\right) \\ \forall i \;\; 0 \le \alpha_i \le C \\ \sum\limits_{i=1}^{4}\alpha_i y_i = 0 \end{cases}$$

- Si on reprend la fonction noyau $K(\boldsymbol{x},\boldsymbol{y}) = (1 + x_1y_1 + x_2y_2)^2$, on obtient les équations suivantes pour le Lagrangien :

$$Q(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}(9\alpha_1^2 - 2\alpha_1\alpha_2 + 2\alpha_1\alpha_3 - 2\alpha_1\alpha_4 + 9\alpha_2^2 - 2\alpha_2\alpha_3 + 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2)$$

$$\alpha_1 - \alpha_2 + \alpha_3 - \alpha_4 = 0$$

Illustration : le cas du XOR

- Le maximum de *Q(a)* est obtenu en prenant ses dérivées par rapport aux α_i et en trouvant les valeurs de α_i qui les annulent :

$$\begin{cases} 1-9\alpha_1+\alpha_2-\alpha_3+\alpha_4=0 \\ 1+\alpha_1-9\alpha_2+\alpha_3-\alpha_4=0 \\ 1-\alpha_1+\alpha_2-9\alpha_3-\alpha_4=0 \\ 1+\alpha_1-\alpha_2+\alpha_3-9\alpha_4=0 \end{cases}$$

  - La valeur optimale des multiplicateurs de Lagrange est :

$$\hat{\alpha}_1=\hat{\alpha}_2=\hat{\alpha}_3=\hat{\alpha}_4=\frac{1}{8}$$

  - Les 4 données du où exclusif sont donc des vecteurs de support, puisque aucune valeur trouvée de α n'est nulle

Illustration : le cas du XOR

- Dans l'espace de re-description :

$$\begin{cases} \hat{\boldsymbol{w}} = \sum_{i=1}^{n_s} \hat{\alpha}_i y_i \Phi(\boldsymbol{x}_i) \\ \hat{b} = y_s - \sum_{i=1}^{n_s} \hat{\alpha}_i y_i K(\boldsymbol{x}_i^T \boldsymbol{x}_s) \\ D(\boldsymbol{x}) = \sum_{j=1}^{s} \hat{\alpha}_j y_j K(\boldsymbol{x}_j, \boldsymbol{x}) + \hat{b} \end{cases}$$

- Donc : $\hat{\boldsymbol{w}} = \frac{1}{8}\left[-\Phi(\boldsymbol{x}_1) + \Phi(\boldsymbol{x}_2) + \Phi(\boldsymbol{x}_3) - \Phi(\boldsymbol{x}_4)\right]$

$$= \frac{1}{8}\left\{ -\begin{pmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ -\sqrt{2} \\ -\sqrt{2} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ -\sqrt{2} \\ \sqrt{2} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ \sqrt{2} \\ -\sqrt{2} \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ \sqrt{2} \\ \sqrt{2} \end{pmatrix} \right\} = \begin{pmatrix} 0 \\ 0 \\ -1/\sqrt{2} \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

(on connait Φ() dans cet exemple, mais il n'est pas requis en général, car l'équation de la marge dépend seulement de K())

$$\hat{b} = 1 - \frac{1}{8}\sum_{j=1}^{4} y_j K(\boldsymbol{x}_j, \boldsymbol{x}_1) = 1 + \frac{1}{8}\sum_{j=1}^{4} (-1)^j K(\boldsymbol{x}_j, \boldsymbol{x}_1) = 0$$

et :

$$D(\boldsymbol{x}) = \frac{1}{8}\sum_{j=1}^{4} y_j K(\boldsymbol{x}_j, \boldsymbol{x}) = -\frac{1}{8}\sum_{j=1}^{4} (-1)^j K(\boldsymbol{x}_j, \boldsymbol{x}) = -x_1 x_2$$

(on aurait obtenu le même résultat en utilisant Φ() :

$$\hat{w}^{\mathrm{T}}\Phi(\boldsymbol{x}) = \left(0, 0, \frac{-1}{\sqrt{2}}, 0, 0, 0\right) \begin{pmatrix} 1 \\ x_1^2 \\ \sqrt{2}x_1 x_2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \end{pmatrix} = -x_1 x_2$$

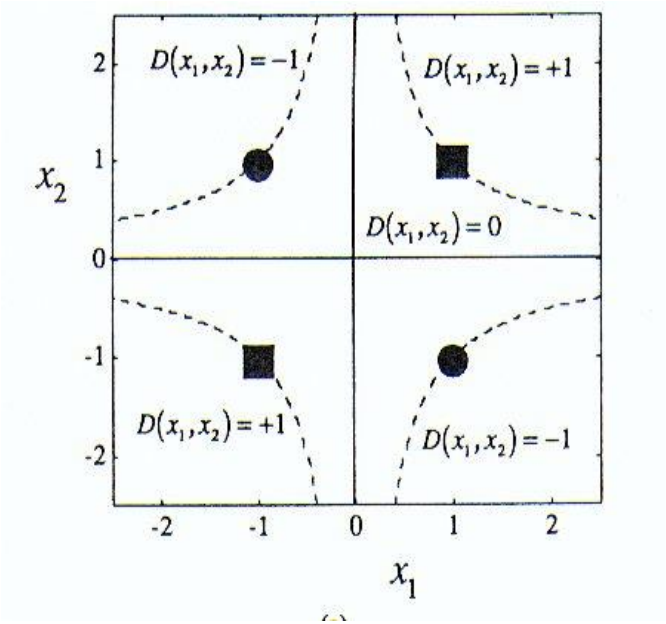
)

- La marge optimale est :

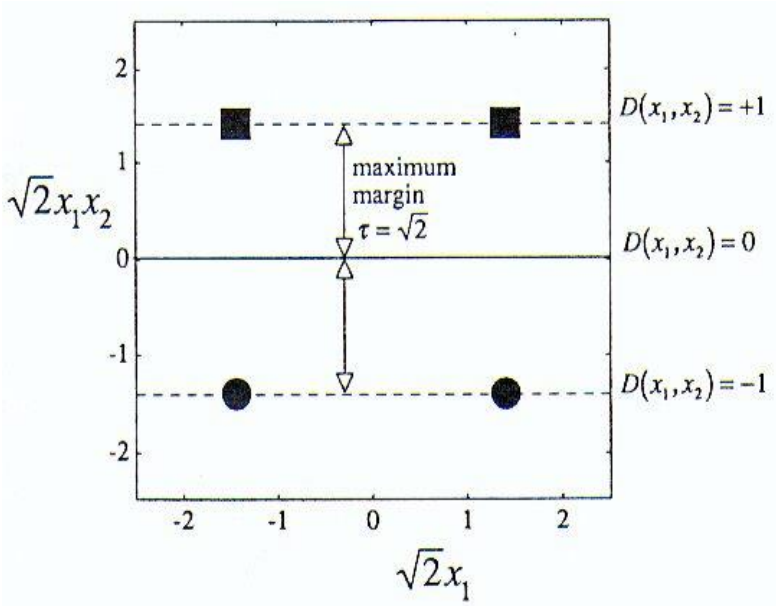
$$\frac{1}{2}\|\hat{\boldsymbol{w}}\|^2 = \frac{1}{2}\hat{\boldsymbol{w}}^T\hat{\boldsymbol{w}} = \frac{1}{2}\left(\sum_{i=1}^{4}\hat{\alpha}_i y_i \Phi(\boldsymbol{x}_i)\right)^T\left(\sum_{j=1}^{4}\hat{\alpha}_j y_j \Phi(\boldsymbol{x}_i)\right)$$

$$= \frac{1}{2}\cdot\frac{1}{8^2}\sum_{i=1}^{4}\sum_{j=1}^{4} y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{4} \Rightarrow \|\hat{\boldsymbol{w}}\| = \frac{1}{\sqrt{2}}$$

Illustration : le cas du XOR



Séparatrice dans l'espace d'entrée



Séparatrice dans l'espace *Φ(x)*

$$D(x) = -x_1x_2$$

$$\sqrt{2}\,x_1x_2 = 0$$

Autre Exemple

- Supposons 5 nombres x_1=1, x_2=2, x_3=4, x_4=5, x_5=6, avec
  - 1, 2, 6 $\in$ classe 1 (y=1)
  - 4, 5 $\in$ classe 2 (y=-1)
  - Donc: $\{(x_i, y_i)\}_{i=1,\ldots,5}$ ={(1,1), (2,1), (4,-1), (5,-1), (5,1)}
- Utilisons à nouveau le noyau polynomial de degré 2
  - K($\boldsymbol{x},\boldsymbol{y}$) = $(1+\boldsymbol{x}^T\boldsymbol{y})^2$
  - C est choisi égal à 100
- Trouvons d'abord $\{\alpha_i\}_{i=1,\ldots,5}$ pour satisfaire :

$$\text{max.} \quad \sum_{i=1}^{5}\alpha_i - \frac{1}{2}\sum_{i=1}^{5}\sum_{j=1}^{5}\alpha_i\alpha_j y_i y_j (x_i x_j + 1)^2$$

$$100 \geq \alpha_i \geq 0, \sum_{i=1}^{5}\alpha_i y_i = 0$$

Exemple

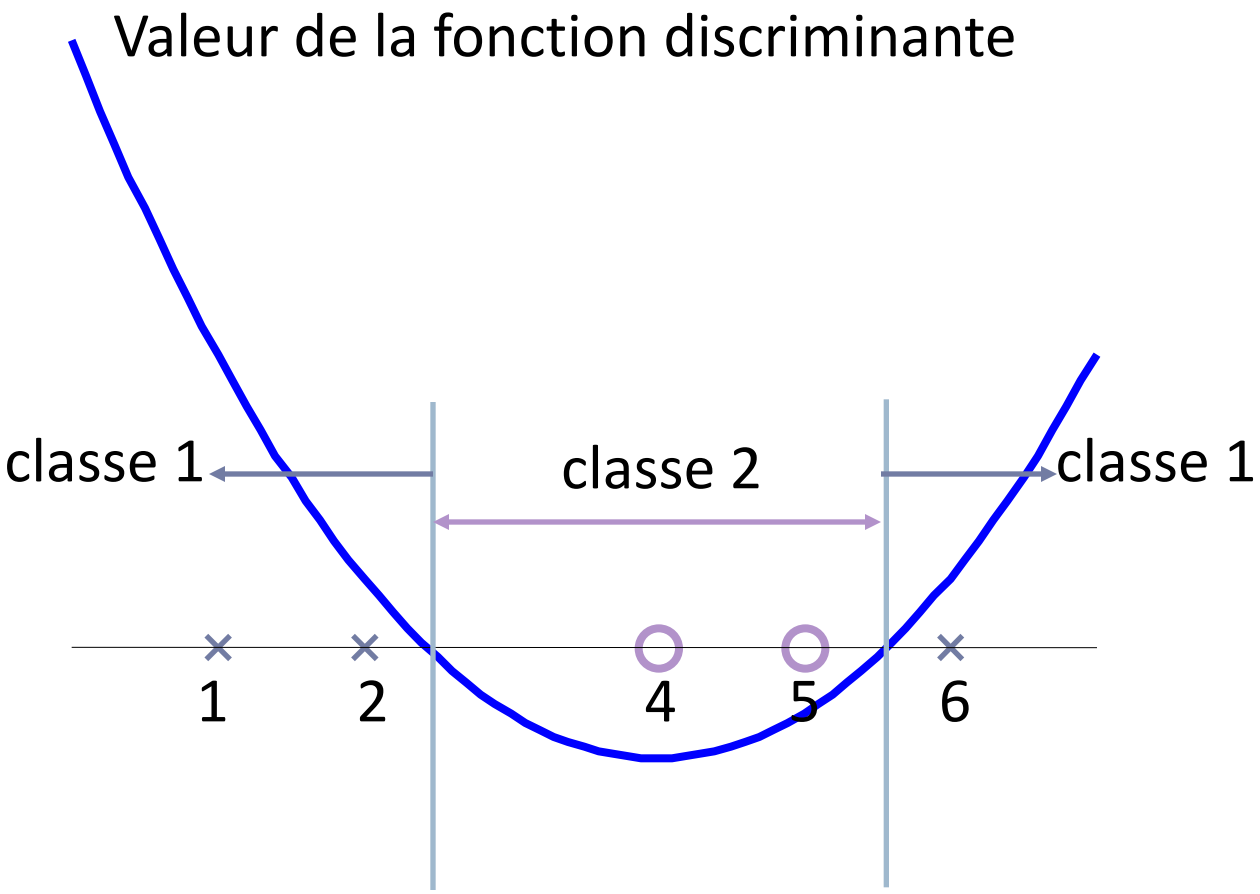
- La solution est :
  - α_1=0, α_2=2.5, α_3=0, α_4=7.333, α_5=4.833
    - Les vecteur supports sont donc $\{x_2=2, x_4=5, x_5=6\}$
- La fonction discriminante est

$$f(y) = 2.5(1)(2y+1)^2 + 7.333(-1)(5y+1)^2 + 4.833(1)(6y+1)^2 + b$$
$$= 0.6667x^2 - 5.333x + b$$

  - *b* trouvé en résolvant f(2)=1 ou f(5)=-1 ou f(6)=1, puisque x_2, x_4, x_5 sont dans $y_i(\mathbf{w}^T\phi(z) + b) = 1$ et tous donnent b=9

$$\rightarrow \quad f(y) = 0.6667x^2 - 5.333x + 9$$

Exemple


Valeur de la fonction discriminante
classe 1
classe 2
classe 1
1
2
4
5
6

Exemples de fonctions noyaux

- Noyau polynomial de degré *d*

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^d$$

- Noyau à fonction à base radiale de dispersion σ

$$K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/(2\sigma^2))$$

  - Très proche des RN avec fonctions à base radiale

- Sigmoïde avec paramètres κ et θ

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa\mathbf{x}^T\mathbf{y} + \theta)$$

  - Ne satisfait pas la condition de Mercer pour tous κ et θ

- La recherche d'autres fonctions noyaux pour diverses applications est très active !

Exemple d'application avec SciKit

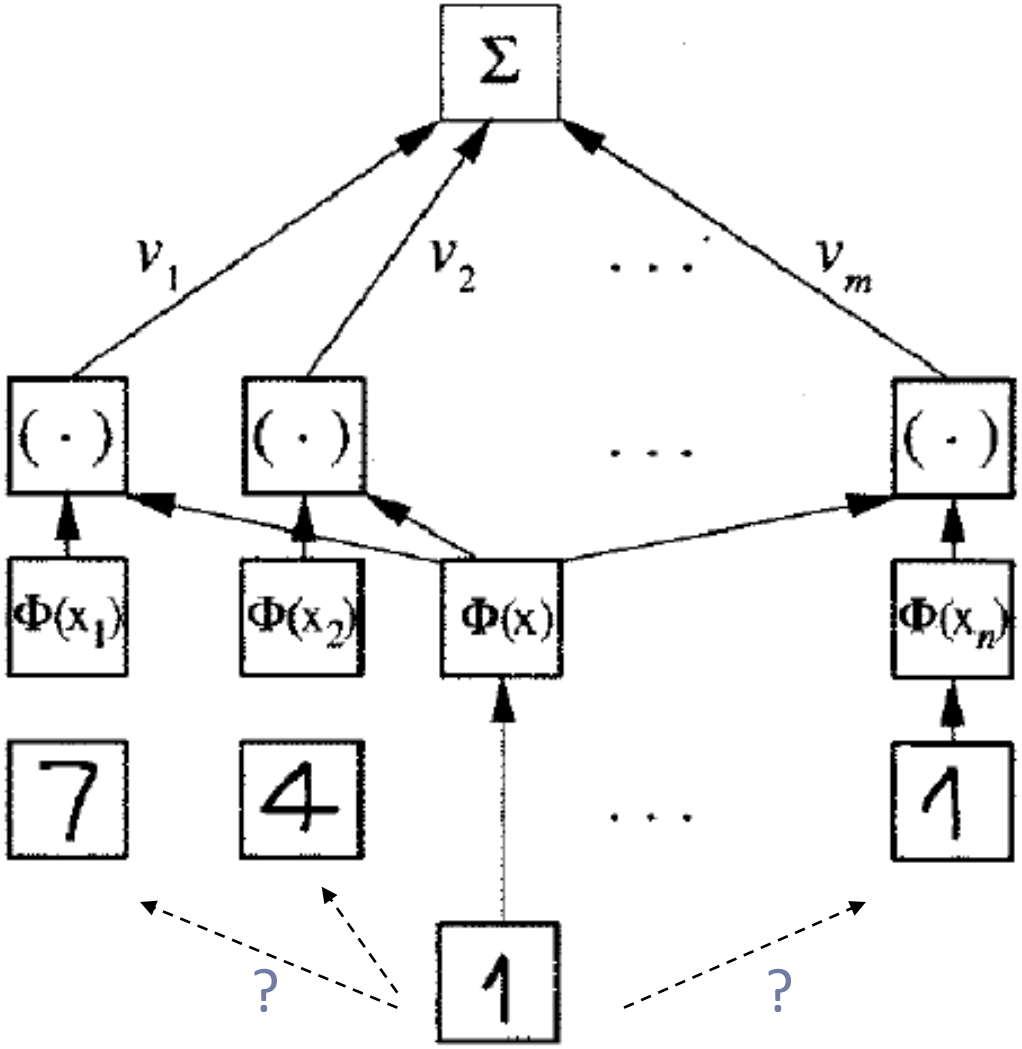
```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn import metrics

#Load dataset
cancer = datasets.load_breast_cancer()
# print the names of the 30 features
print("Features: ", cancer.feature_names)
# print the label type of cancer('malignant' 'benign')
print("Labels: ", cancer.target_names)
# print data rows and features
print("rows, features per row: ", cancer.data.shape)

# Split dataset into 70% training set and 30% test set
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target,
test_size=0.3,random_state=109)
```

```
#Create a svm Classifier
clf = svm.SVC(kernel='linear') # default=rbf
#Train the model using the training sets
clf.fit(X_train, y_train)
#Predict the response for test dataset
y_pred = clf.predict(X_test)

# Model Accuracy
print("Accuracy: ", metrics.accuracy_score(y_test, y_pred)
```

Classification multi-classes

- SVM est à la base un classifieur binaire
- On peut changer la formulation pour permettre la classification multi-classe
  - L'ensemble des données est divisé en deux parts de multiples façons, et classé ensuite
    - Un contre tous ou un contre chaque alternative
    - Un SVM séparé est formé pour chaque division
  - La classification multi-classes est accomplie en combinant la sortie de tous les SVM
    - Prône à l'explosion combinatoire des possibilités!

Example d'application des SVM : Reconnaissance de l'écriture manuscrite



output $\Sigma\, v_i\, k\,(x,x_i) + b$

weights

dot product $(\Phi(x)\cdot\Phi(x_i)) = k(x,x_i)$

mapped vectors $\Phi(x_i)$, $\Phi(x)$

support vectors $x_1 \ldots x_n$

test vector x

Sommaire: étapes de la classification

- Préparer la matrice des patrons
- Choisir la fonction noyau à utiliser
- Choisir les paramètres de la fonction noyau et la valeur de *C* (valeurs suggérées par le logiciel SVM ou essai-erreur).
- Exécuter l'algorithme d'apprentissage pour trouver α_i

- Les données nouvelles peuvent être classées en fonctions des α_i et des vecteurs supports trouvés

Effet des paramètres de contrôle.

- Apprentissage de données en damier
  - Apprentissage de deux classes
  - SVM à fonction noyau gaussienne

$$K(x, x') = e^{-\frac{\|x-x'\|^2}{2\sigma^2}}$$

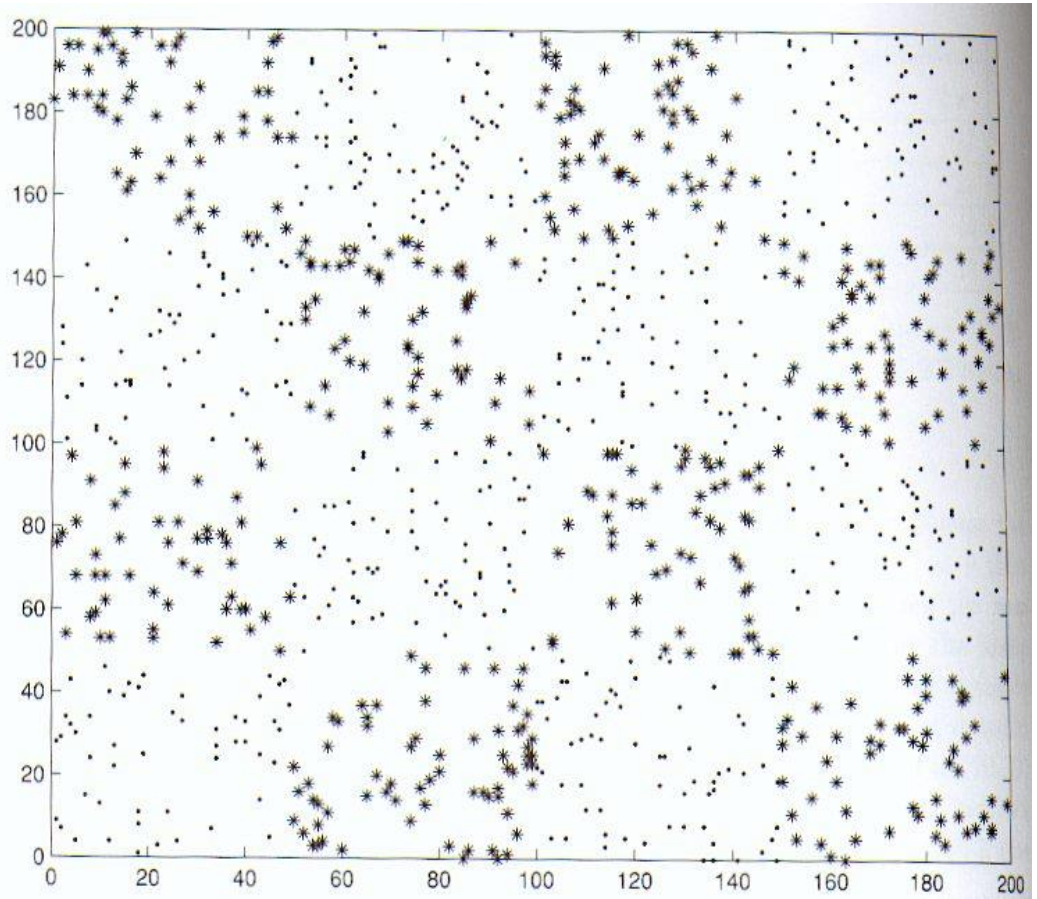


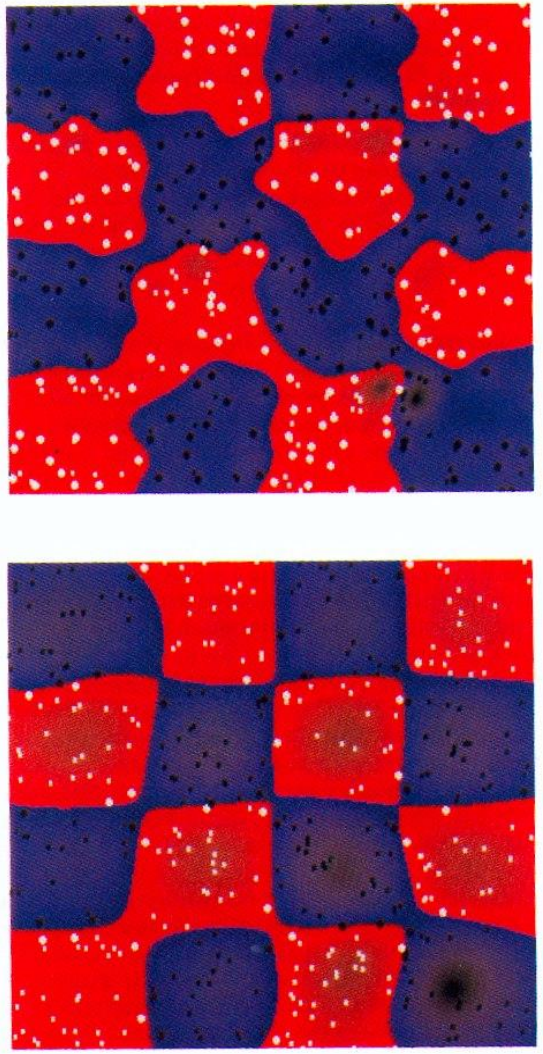
Figure 10.2 Training points for checkerboard pattern

Effet des paramètres de contrôle

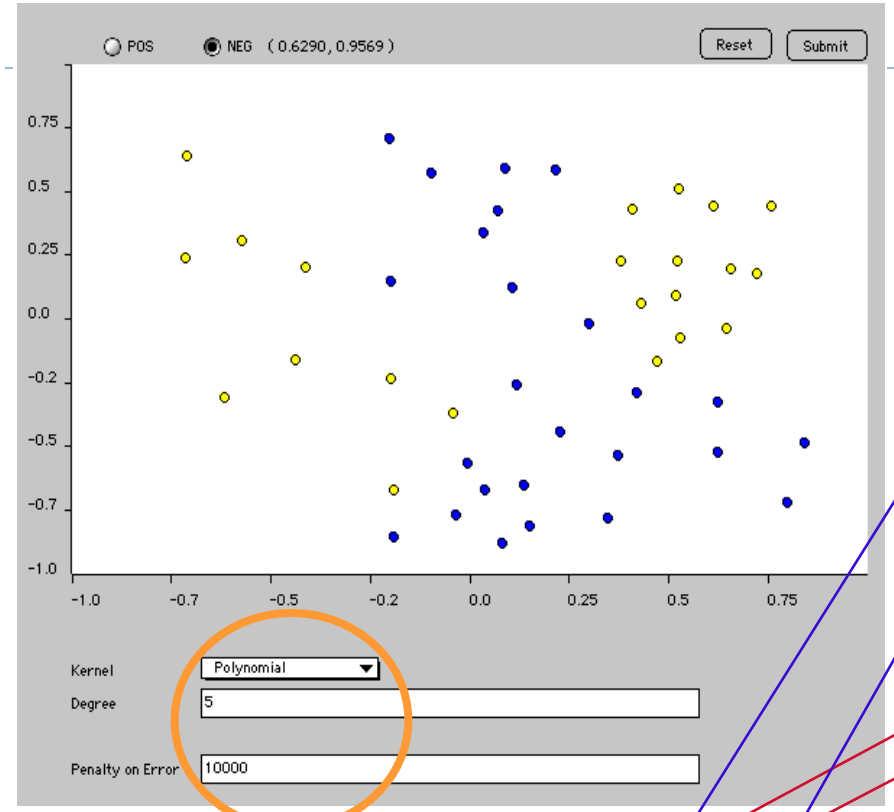
- Apprentissage de deux classes
  - exemples tirés uniformément sur l'échiquier
- SVM à fonctions noyau gaussienne

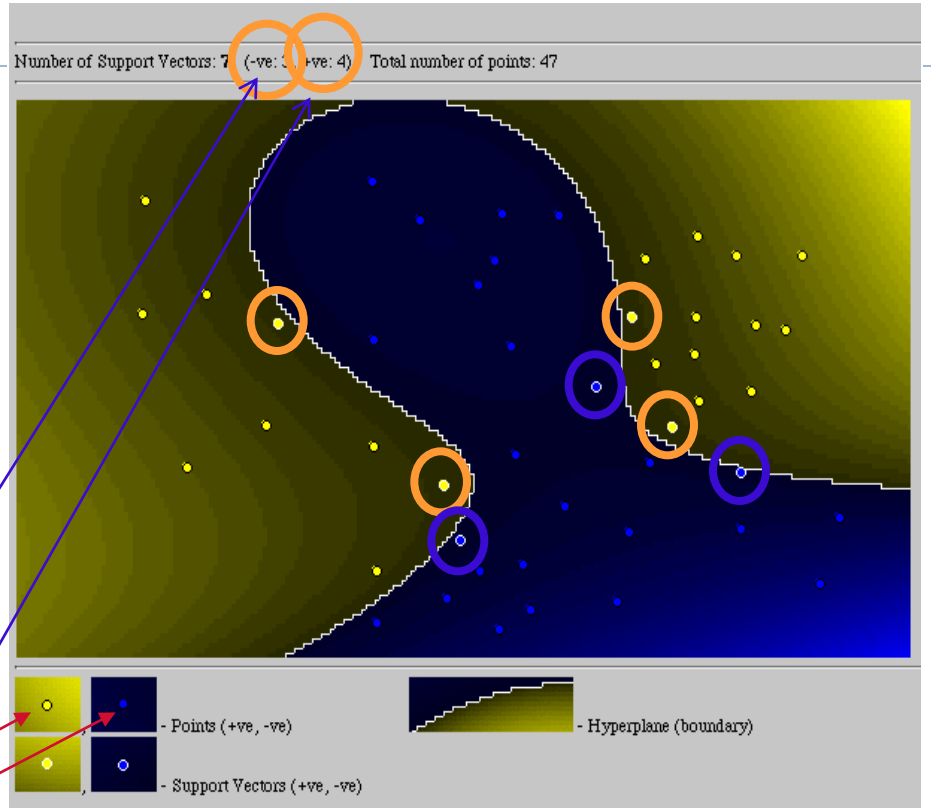
$$K(\boldsymbol{x},\boldsymbol{x}') = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2\sigma^2}}$$

- Ici deux valeurs de σ
  - En haut : petite valeur
  - En bas : grande valeur
- Les gros points sont des exemples critiques
  - Plus en haut qu'en bas



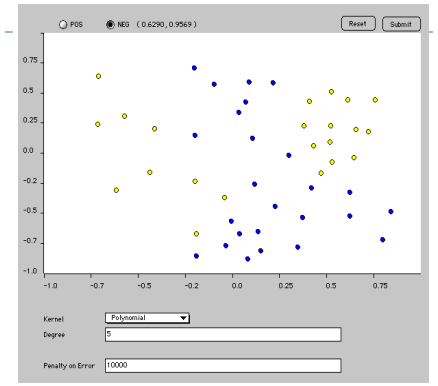
Une applette de démonstration

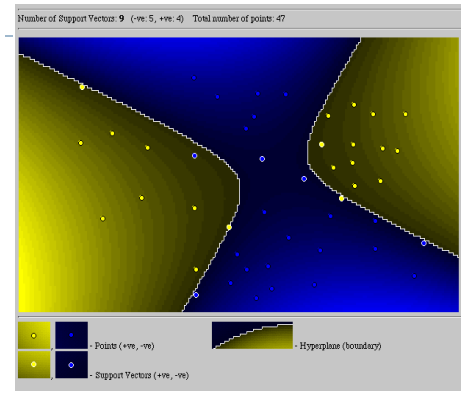


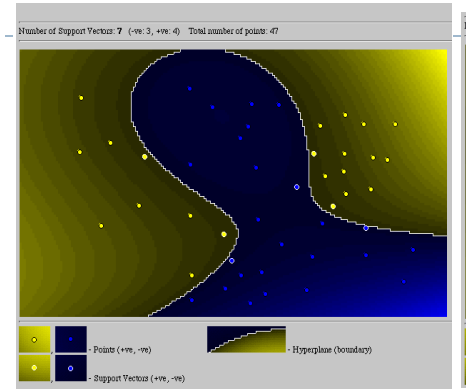


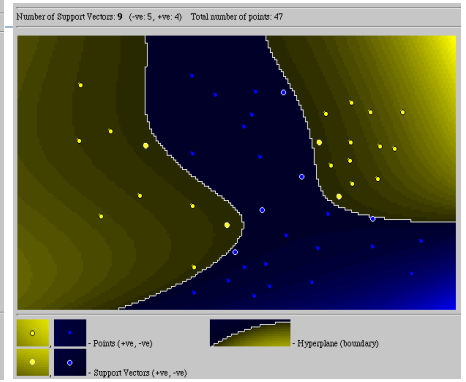
- http://svm.cs.rhul.ac.uk/pagesnew/GPat.shtml
- 47 exemples (22 +, 25 -)
- *Exemples critiques* : 4 + et 3 -
- **Ici** *fonction polynomiale* **de** *degré 5* **et** *C = 10000*

Paramètres de contrôle : les fonctions noyau





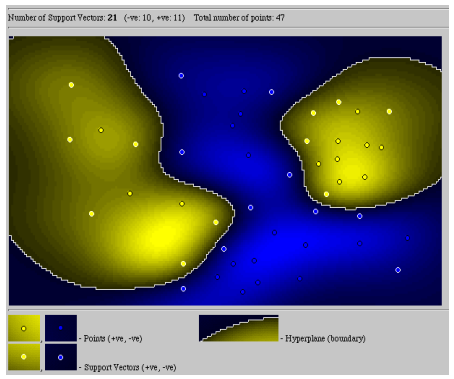


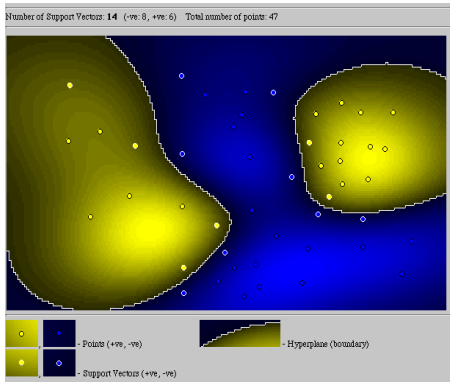


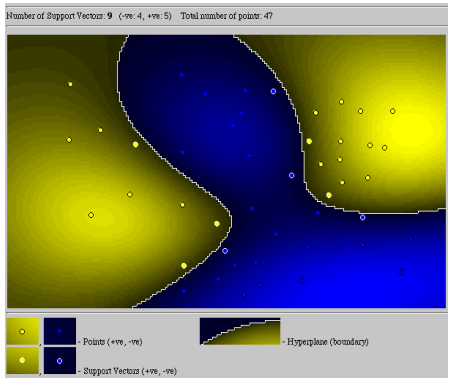
(5-, 4+) **(3-, 4+)** **(5-, 4+)**

- 47 exemples (22 +, 25 -)
- *Exemples critiques* : 4 + et 3 -

Ici *fonction polynomiale* **de** *degré 2, 5*, 8 **et** *C = 10000*







(10-, 11+) **(8-, 6+)** **(4-, 5+)**

Ici *fonction Gaussienne* **de** σ *= 2, 5*, 10, 20 **et** *C = 10000*

Domaines d'application des SVMs

- Traitement d'images
  - Reconnaissance de caractères manuscrits
  - Reconnaissance de scènes naturelles
  - Reconnaissance de visages
  - *Entrées* : image bidimensionnelle en couleur ou en tons de gris codée en vecteur de pixels ou traits
  - *Sortie* : classe (chiffre / personne)

Application : images couleurs

- Ex. : Base d'images Corel Stock Photo Collection
  - 200 catégories
  - 100 images / catégorie
- Codage
  - Pixel = vecteur dans espace à trois dimensions (RGB)
  - Image = histogramme (fraction des pixels d'une couleur donnée)

  Invariant / nombreuses opérations
- Noyau : 

$$K(x,z) = exp\left(-\frac{d(x,z)}{\sigma^2}\right)$$

$$d(x,z) = \sum_{i=1}^{n} \frac{(x_i - z_i)^2}{x_i + z_i}$$

(fonction c^2)

Domaines d'application des SVMs

- Catégorisation de textes
  - Classification de courriels
  - Classification de pages web
  - *Entrées* : document (texte ou html)
    - Approche « sac de mots »
    - Document = vecteur de mots (lemmatisés pondérés par tf-idf)
  - *Sortie* : catégorie (thème, spam/non-spam)
  - ***Noyau*** :
    - Produit scalaire des vecteurs
    - $C = \infty$ (marge dure)

Domaines d'application des SVMs

- Diagnostic médical
  - Évaluation du risque de cancer
  - Détection d'arythmie cardiaque
  - Évaluation du risque d'accidents cardio-vasculaires à moins de 6 ans
- *Entrées* : état du patient (sexe, age, bilan sanguin, …)
- *Sortie* :
  - Classe : à risque ou non
  - Probabilité d'accident à échéance donnée

Extensions

- Leçon à retenir des SVM:
  - *Un algorithme linéaire dans l'espace de re-description peut remplacer un algorithme non-linéaire dans l'espace d'entrée*
- Les algorithme linéaires classiques peuvent être généralisés en des versions non-linéaires en allant vers l'espace de re-description
  - ACP à noyaux, k-moyennes à noyaux, etc.
  - Régression
  - Détection de « nouveautés »

Régression vectorielle à support ε (ε-SVR)

- L'idée est de trouver $f(\vec{x}) = \vec{w} \cdot \vec{x} + b$ qui fait l'approximation d'un ensemble $\{\vec{x}_i, y_i\}_{i=1..N}$ avec une déviation maximale de $\pm\varepsilon$ des vraies valeurs de y
  - Les points doivent être compris dans la bande de largeur 2ε.

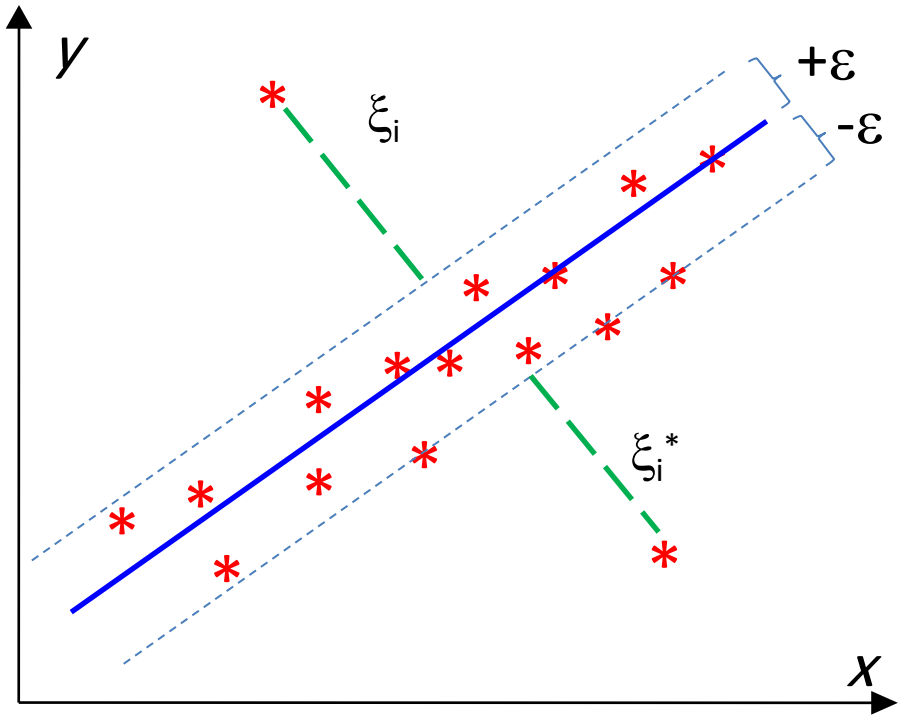
- Le problème d'optimisation est :

$$\text{Min } \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$

$$\text{subject to } \begin{cases} u_i - \mathbf{w}^T\mathbf{x}_i - b \leq \epsilon + \xi_i \\ \mathbf{w}^T\mathbf{x}_i + b - y_i \leq \epsilon + \xi_i^* \\ \xi_i \geq 0, \xi_i^* \geq 0 \end{cases}$$

ou ξ_i et ξ_i^* pénalisent les points hors cible

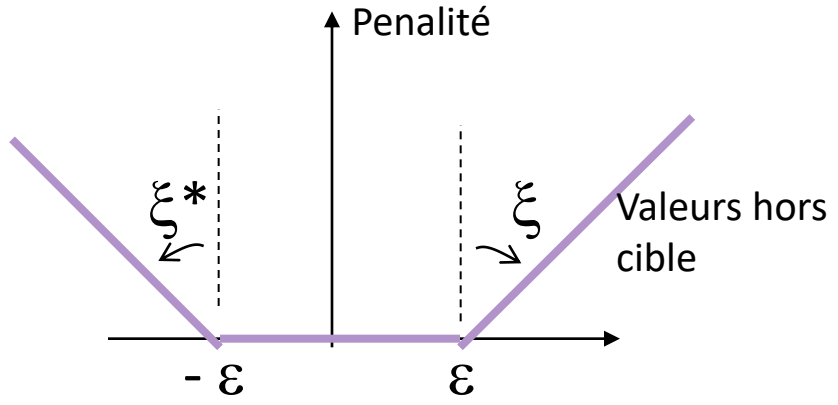
- Formulation similaire à SVM



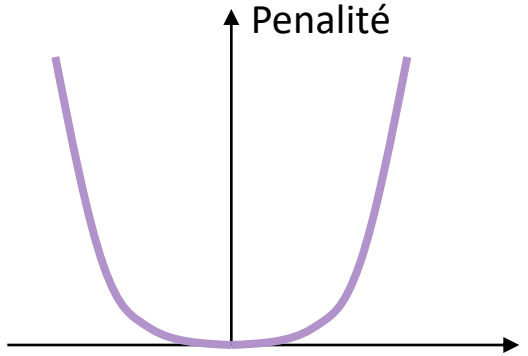
Régression vectorielle à support ε (ε-SVR)

- Régression linéaire dans l'espace de redescription
- À l'encontre de la régression par moindres carrés, la fonction d'erreur est une fonction de perte ε-insensible, et linéaire ensuite
  - Intuitivement, une erreur inférieure à ε est ignorée
  - Cela mène à des points de marge terses similaires à SVM

Fonction de perte ε -insensible

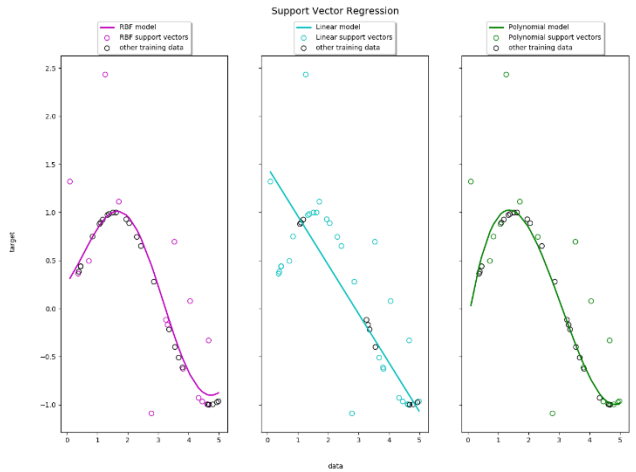


Fonction de perte quadratique



Exemple d'application pour la regression

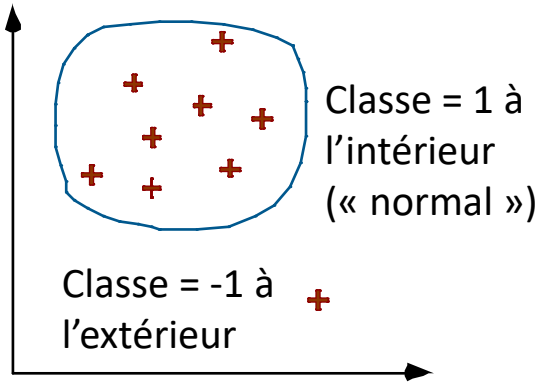
```
# Support Vector Regression (SVR) using linear and non-linear kernels
# 1D regression using linear, polynomial and RBF kernels.
import numpy as np
from sklearn.svm import SVR
import matplotlib.pyplot as plt
# Generate sample data
X = np.sort(5 * np.random.rand(40, 1), axis=0)
y = np.sin(X).ravel()
# Add noise to targets
y[::5] += 3 * (0.5 - np.random.rand(8))
# Fit regression model
svr_rbf = SVR(kernel='rbf', C=100, gamma=0.1, epsilon=.1)
svr_lin = SVR(kernel='linear', C=100, gamma='auto')
svr_poly = SVR(kernel='poly', C=100, gamma='auto', degree=3, epsilon=.1, coef0=1)
# Look at the results
lw = 2
svrs = [svr_rbf, svr_lin, svr_poly]
kernel_label = ['RBF', 'Linear', 'Polynomial']
model_color = ['m', 'c', 'g']

fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15, 10), sharey=True)

for ix, svr in enumerate(svrs):
                axes[ix].plot(X, svr.fit(X, y).predict(X), color=model_color[ix], lw=lw,
label='{}                    model'.format(kernel_label[ix]))
                axes[ix].scatter(X[svr.support_], y[svr.support_], facecolor="none",
                                                edgecolor=model_color[ix], s=50,
label='{} support
                vectors'.format(kernel_label[ix]))
                axes[ix].scatter(X[np.setdiff1d(np.arange(len(X)), svr.support_)],

                y[np.setdiff1d(np.arange(len(X)), svr.support_)], facecolor="none",
                                edgecolor="k", s=50, label='other training data')
                axes[ix].legend(loc='upper center', bbox_to_anchor=(0.5, 1.1),
ncol=1,                                         fancybox=True, shadow=True)
fig.text(0.5, 0.04, 'data', ha='center', va='center')
 fig.text(0.06, 0.5, 'targe
fig.suptitle("Support Vec

plt.show()
```

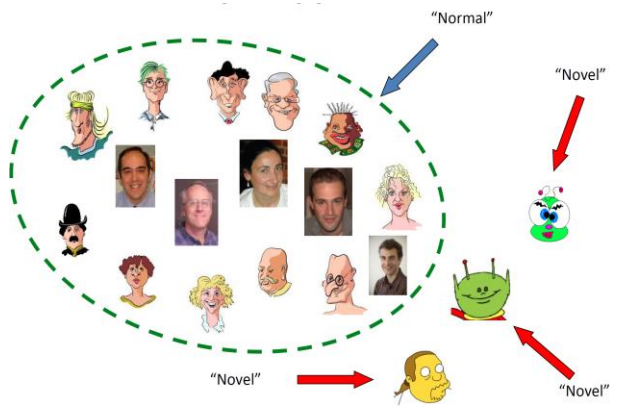


Source: scikit

SVM pour le groupement à une classe (One Class SVM)

- Permet la détection de nouveautés ou d'anomalies
  - On cherche à séparer au maximum les points de la classe de l'origine





- On trouve $f(\vec{x}) = sign(\vec{w}\cdot\vec{x}+b)$ en minimisant :

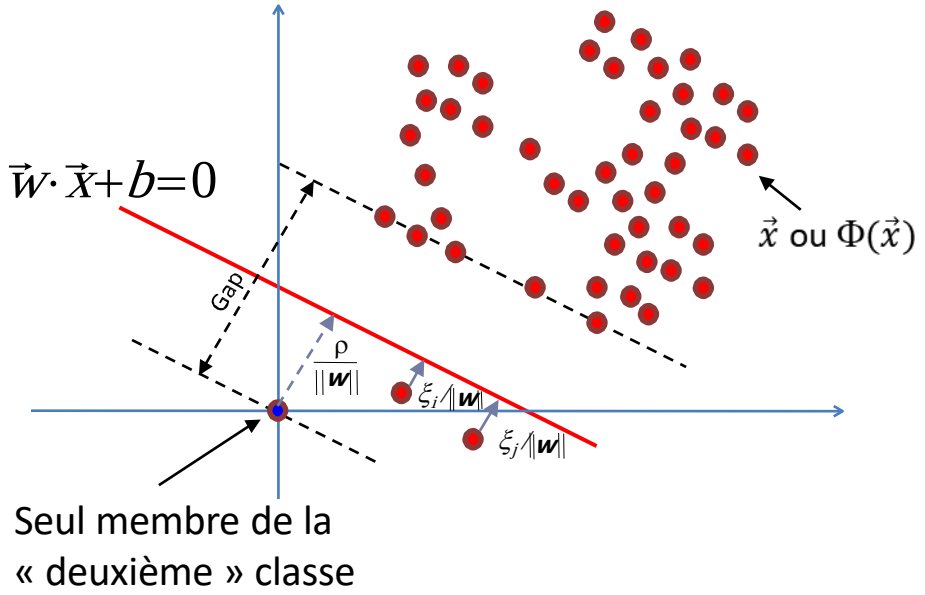
$$\tfrac{1}{2}\|\vec{w}\|^2 + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i + b$$

Sous les contraintes $\vec{w}\cdot\vec{x}+b \geq -\xi_i$

$$\xi_i \geq 0 \text{ pour } i = 1,\ldots,N$$

ν permet de régler la fraction de points singuliers

- Apprentissage non supervisé! (une seule classe)



Exemple d'utilisation pour la détection d'anomalies

```
print(__doc__)
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.font_manager
from sklearn import svm
xx, yy = np.meshgrid(np.linspace(-5, 5, 500), np.linspace(-5, 5, 500))
# Generate train data
X = 0.3 * np.random.randn(100, 2)
X_train = np.r_[X + 2, X - 2]
# Generate some regular novel observations
X = 0.3 * np.random.randn(20, 2)
X_test = np.r_[X + 2, X - 2]
# Generate some abnormal novel observations
X_outliers = np.random.uniform(low=-4, high=4, size=(20, 2))
# fit the model
clf = svm.OneClassSVM(nu=0.1, kernel="rbf", gamma=0.1)
clf.fit(X_train)
y_pred_train = clf.predict(X_train)
y_pred_test = clf.predict(X_test)
y_pred_outliers = clf.predict(X_outliers)
n_error_train = y_pred_train[y_pred_train == -1].size
n_error_test = y_pred_test[y_pred_test == -1].size
n_error_outliers = y_pred_outliers[y_pred_outliers == 1].size
# plot the line, the points, and the nearest vectors to the plane
Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.title("Novelty Detection")
plt.contourf(xx, yy, Z, levels=np.linspace(Z.min(), 0, 7), cmap=plt.cm.PuBu)
a = plt.contour(xx, yy, Z, levels=[0], linewidths=2, colors='darkred')
plt.contourf(xx, yy, Z, levels=[0, Z.max()], colors='palevioletred')
s = 40
b1 = plt.scatter(X_train[:, 0], X_train[:, 1], c='white', s=s, edgecolors='k')
b2 = plt.scatter(X_test[:, 0], X_test[:, 1], c='blueviolet', s=s,
                 edgecolors='k')
c = plt.scatter(X_outliers[:, 0], X_outliers[:, 1], c='gold', s=s,
                edgecolors='k')
plt.axis('tight')
plt.xlim((-5, 5))
plt.ylim((-5, 5))
plt.legend([a.collections[0], b1, b2, c],
           ["learned frontier", "training observations",
            "new regular observations", "new abnormal observations"],
           loc="upper left",
           prop=matplotlib.font_manager.FontProperties(size=11))
plt.xlabel(
    "error train: %d/200 ; errors novel regular: %d/40 ; "
    "errors novel abnormal: %d/40"
    % (n_error_train, n_error_test, n_error_outliers))
plt.show()
```

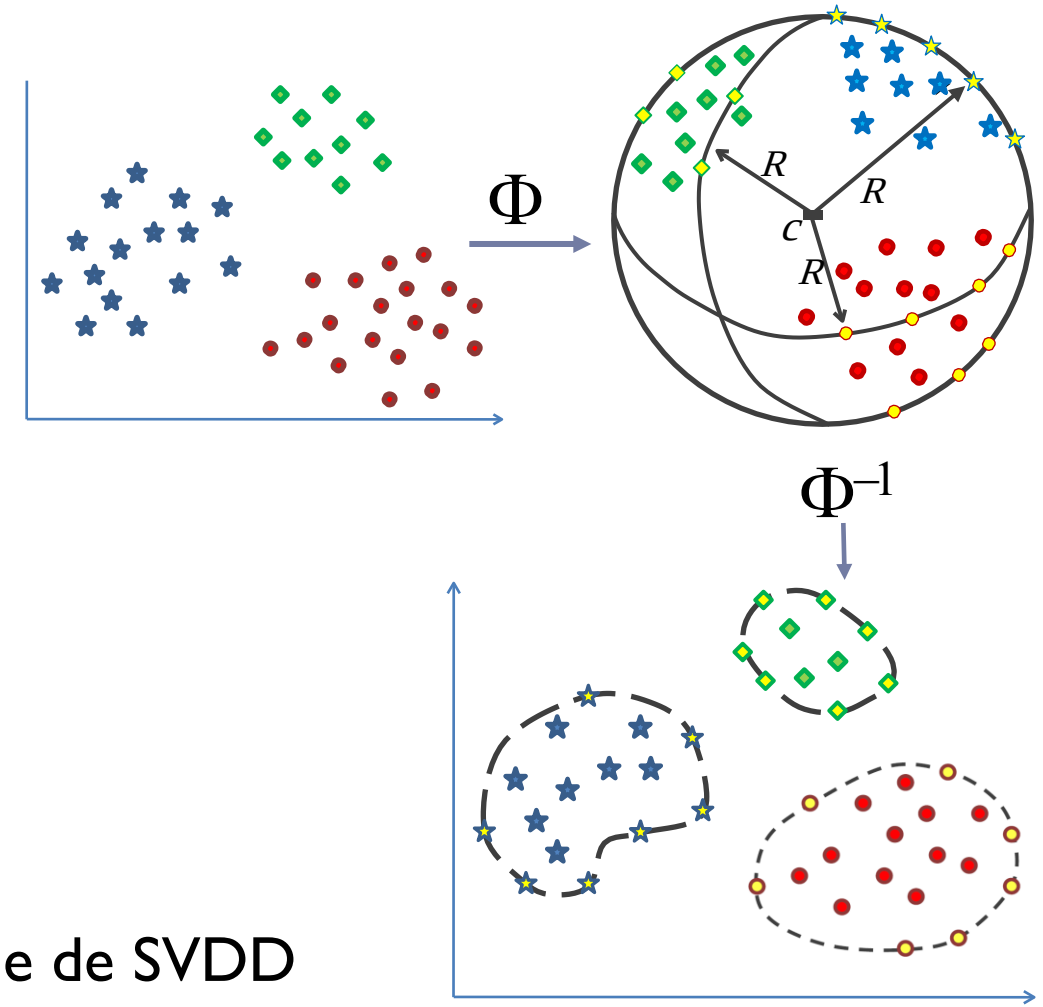

#https://docs.w3cub.com/scikit_learn/auto_examples/svm/plot_oneclass/#sphx-glr-download-auto-examples-svm-plot-oneclass-py

SVM pour le groupement de plusieurs classes

- SVDD (support vector domain description) trouve l'hypersphère de plus petit rayon à englober les données
  - Les vecteurs à la surface de l'hypesphère sont des vecteurs de support
- Pour $X = \{x_i\}$ et Φ une transformation de noyau radial (p. ex. gaussien), trouver la plus petite hypersphere de rayon R et centre c qui satisfait :

$$\|\Phi(x_j) - c\|^2 \leq R^2 + \xi_j$$

$$\xi_j \geq 0$$



- SVM à une classe avec noyau RBF est une forme de SVDD

Pourquoi ça marche ?

La marge est liée à la capacité en généralisation

- **Normalement, la classe des hyperplans de R^d est de $d_H = d + 1$**
- Mais la classe des hyperplans de marge $\frac{1}{\|w\|}$ tq. $\|w\|^2 \leq c$

  est bornée par : $d_H \leq \text{Min}(R^2 c, d) + 1$

  où R est le rayon de la plus petite sphère englobant l'échantillon d'apprentissage S

↙ **Peut être beaucoup plus petit que la dimension d de l'espace d'entrée X**

Forces et faiblesses des SVM

- Forces
  - L'apprentissage est relativement facile
    - Pas de minima locaux comme pour les RNA
  - L'algorithme est robuste face aux changements d'échelle
  - Le compromis entre la complexité du classeur et l'erreur de classification peut être gérée explicitement
  - Méthode générale
    - Des données non conventionnelles, telles des chaînes et des arbres peuvent servir d'entrées au SVM, à la place des vecteurs de traits
  - Résultats en général **équivalents et souvent meilleurs**
- Faiblesses
  - Il faut trouver la "bonne" fonction noyau
  - Problèmes i.i.d. (données indépendantes et identiquement distribuées)
  - Deux classes à la fois

Sources documentaires

Ouvrages / articles

- Cornuéjols & Miclet (02) : *Apprentisage artificiel. Concepts et algorithmes*. Eyrolles, 2002.
- Cristianini & Shawe-Taylor (00) : *Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.
- Herbrich (02) : *Learning kernel classifiers*. MIT Press, 2002.
- Schölkopf, Burges & Smola (eds) (98) : *Advances in Kernel Methods : Support Vector Learning*. MIT Press, 1998.
- Schölkopf & Smola (02) : *Learning with kernels*. MIT Press, 2002.
- Smola, Bartlett, Schölkopf & Schuurmans (00) : *Advances in large margin classifiers*. MIT Press, 2000.
- Vapnik (95) : *The nature of statistical learning.* Springer-Verlag, 1995.

Sites web

- http://www.kernel-machines.org/ **(point d'entrée)**
- http://www.support-vector.net **(point d'entrée)**

Implémentation des SVMs

- **Minimisation de fonctions différentiables convexes à plusieurs variables**
  - **Pas d'optima locaux**
  - **Mais :**
    - **Problèmes de stockage de la matrice noyau (si milliers d'exemples)**
    - **Long dans ce cas**
  - D'où mise au point de méthodes spécifiques
    - Gradient sophistiqué
    - Méthodes itératives, optimisation par morceaux
  - Plusieurs **packages publics** disponibles
    - SVMTorch
    - SVM^{Light}
    - SMO
    - …

Logiciels

- Une liste de réalisations de SVM se trouve à http://www.kernel-machines.org/software.html
- Certaines (tel LIBSVM) peuvent gérer la classification multi-classe
- SVMLight figure parmi les premières mises en oeuvres de SVM (écrit en c ; http://svmlight.joachims.org/)
- IL existe plusieurs boîtes à outils Matlab pour les SVM sur le web
- Scikit aussi inclut SVM

Autres ressources

- http://www.kernel-machines.org/
- http://www.support-vector.net/
- http://www.support-vector.net/icml-tutorial.pdf
- http://www.kernel-machines.org/papers/tutorial-nips.ps.gz
- http://www.clopinet.com/isabelle/Projects/SVM/applist.html

Conclusion

- SVM sont une alternative aux réseaux de neurones pour la classification binaire
- Concepts clés des SVM : maximiser la marge et exploiter l'astuce des noyaux
- Domaine de recherche encore d'intérêt
- Plusieurs mises en œuvre en logiciel libre existent et sont disponible sur le WEB !