

# Support Vector Machines and kernel methods

MDI343 - Machine Learning

---

Florence d'Alché-Buc

Contact: [florence.dalche@telecom-paristech.fr](mailto:florence.dalche@telecom-paristech.fr),  
MS Big Data, Télécom ParisTech, France

Rappels

SVM linéaires

Passage au cas non linéaire et noyaux

Pour aller plus loin: Support Vector Regression

Conclusion et References

## Cadre probabiliste et statistique 1/2

- Soit  $X$  un vecteur aléatoire de  $\mathcal{X} = \mathbb{R}^p$
- $Y$  une variable aléatoire discrète  $\mathcal{Y} = \{-1, 1\}$
- Soit  $\mathbb{P}$  la loi de probabilité jointe de  $(X, Y)$
- Soit  $\mathcal{S}_n = \{(x_1, y_1), \dots, (x_n, y_n)\}$ , i.i.d. sample from  $\mathbb{P}$ .

## Cadre probabiliste et statistique 2/2

- Soit  $f : \mathbb{R}^p \rightarrow \{-1, +1\}$  une fonction de classification binaire:  
 $f(x) = \text{sign}(h(x))$  avec  $h : \mathbb{R}^p \rightarrow \mathbb{R} \in \mathcal{H}$
- Soit  $\ell : \{-1, +1\} \times \mathbb{R} \rightarrow \mathbb{R}$  une fonction de perte
- Risque empirique  $R_n(h) = \frac{1}{n} \sum_i \ell(y_i, h(x_i))$  et un terme régularisateur  $\Omega(h)$  qui mesure la *complexité* de  $h$ .
- On cherche :  $\hat{h} = \arg \min_{h \in \mathcal{H}} R_n(h) + \lambda \Omega(h)$

# Et en pratique, comment fait-on ?

## Méthodologie pour développer une approche discriminante

- Définir
  - **l'espace de représentation** des entrées

# Et en pratique, comment fait-on ?

## Méthodologie pour développer une approche discriminante

- Définir
  - l'**espace de représentation** des entrées
  - la **classe des fonctions** de classification binaire considérées

# Et en pratique, comment fait-on ?

## Méthodologie pour développer une approche discriminante

- Définir
  - l'**espace de représentation** des entrées
  - la **classe des fonctions** de classification binaire considérées
  - la **fonction de coût** à minimiser pour obtenir le meilleur classifieur dans cette classe

# Et en pratique, comment fait-on ?

## Méthodologie pour développer une approche discriminante

- Définir
  - l'**espace de représentation** des entrées
  - la **classe des fonctions** de classification binaire considérées
  - la **fonction de coût** à minimiser pour obtenir le meilleur classifieur dans cette classe
  - l'**algorithme de minimisation** de cette fonction de coût



# Et en pratique, comment fait-on ?

## Méthodologie pour développer une approche discriminante

- Définir
  - l'**espace de représentation** des entrées
  - la **classe des fonctions** de classification binaire considérées
  - la **fonction de coût** à minimiser pour obtenir le meilleur classifieur dans cette classe
  - l'**algorithme de minimisation** de cette fonction de coût
  - une **méthode de sélection de modèle** pour définir les hyperparamètres

Rappels

SVM linéaires

Passage au cas non linéaire et noyaux

Pour aller plus loin: Support Vector Regression

Conclusion et References

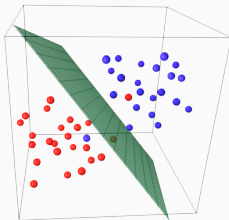
# Séparateur linéaire

## Définition

Soit  $\mathbf{x} \in \mathbb{R}^p$

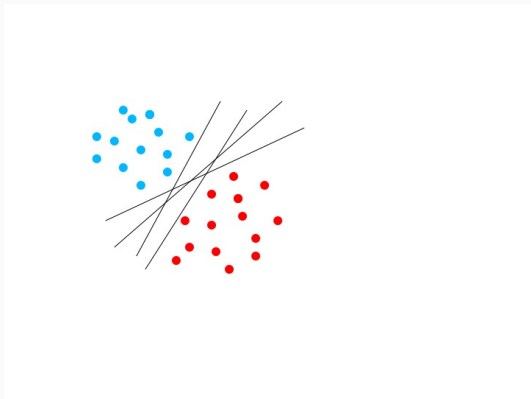
$$h(\mathbf{x}) = \text{signe}(\mathbf{w}^T \mathbf{x} + b)$$

L'équation :  $\mathbf{w}^T \mathbf{x} + b = 0$  définit un hyperplan dans l'espace euclidien  $\mathbb{R}^p$

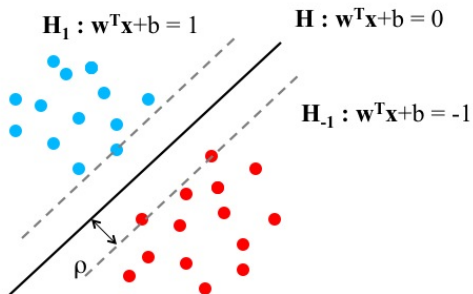


Exemple: données d'apprentissage en 3D et séparateur linéaire

## Cas de données linéairement séparables



Exemple en 2D: quelle droite choisir ?



## Notion de marge géométrique

- Pour séparer les données, on considère un triplet d'hyperplans:
  - $H: \mathbf{w}^T \mathbf{x} + b = 0$ ,  $H_1: \mathbf{w}^T \mathbf{x} + b = 1$ ,  $H_{-1}: \mathbf{w}^T \mathbf{x} + b = -1$
- On appelle *marge géométrique*,  $\rho(\mathbf{w})$  la plus petite distance entre les données et l'hyperplan  $H$ , ici donc la moitié de la distance entre  $H_1$  et  $H_{-1}$
- Un calcul simple donne :  $\rho(\mathbf{w}) = \frac{1}{\|\mathbf{w}\|}$ .

## Comment déterminer $w$ et $b$ ?

- Maximiser la marge  $\rho(\mathbf{w})$  tout en séparant les données de part et d'autre de  $H_1$  et  $H_{-1}$
- Séparer les données bleues ( $y_i = 1$ ) :  $\mathbf{w}^T \mathbf{x}_i + b \geq 1$
- Séparer les données rouges ( $y_i = -1$ ) :  $\mathbf{w}^T \mathbf{x}_i + b \leq -1$

## Optimisation dans l'espace primal

$$\underset{\mathbf{w}, b}{\text{minimiser}} \quad \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{sous la contrainte} \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \quad i = 1, \dots, n.$$

## Référence

Boser, B. E.; Guyon, I. M.; Vapnik, V. N. (1992). "A training algorithm for optimal margin classifiers". Proceedings of the fifth annual workshop on Computational learning theory - COLT '92. p. 144.



# Programmation quadratique sous contraintes d'inégalités affines

Problème primal:

$$\begin{array}{ll} \underset{\mathbf{w}, b}{\text{minimiser}} & \frac{1}{2} \|\mathbf{w}\|^2 \\ \text{sous la contrainte} & 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b) \leq 0, \quad i = 1, \dots, n. \end{array}$$

# Programmation quadratique sous contraintes d'inégalités affines

On peut définir une nouvelle fonction objectif: la somme de la fonction à minimiser + la somme des contraintes multipliées par des coefficients positifs , dits de Lagrange

## Lagrangien

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 + \sum_i \alpha_i (1 - y_i(\mathbf{w}^T \mathbf{x}_i + b))$$
$$\forall i, \alpha_i \geq 0$$

La solution de ce problème convexe sous contraintes d'inégalités affines peut être obtenue en résolvant un problème de point de selle :  $\min_{\mathbf{w}} \max_{\alpha} \mathcal{L}(\mathbf{w}, \alpha)$ . Lorsque la fonction est convexe en  $\mathbf{w}$  (b) et concave en  $\alpha$ , nous pouvons intervertir le min et le max.

En l'extremum, on a

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \mathbf{w} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i = 0$$

$$\nabla_b \mathcal{L}(b) = - \sum_{i=1}^n \alpha_i y_i = 0$$

$$\forall i, \alpha_i \geq 0$$

$$\forall i, \alpha_i [1 - y_i(\mathbf{w}^T \mathbf{x}_i + b)] = 0$$

## Obtention des $\alpha_i$ : résolution dans l'espace dual

$$\mathcal{L}(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

- Maximiser  $\mathcal{L}$  sous les contraintes  $\alpha_i \geq 0$  et  $\sum_i \alpha_i y_i = 0, \forall i = 1, \dots, n$
- Faire appel à un solveur quadratique

Aller plus loin: slides sur le site pédagogique

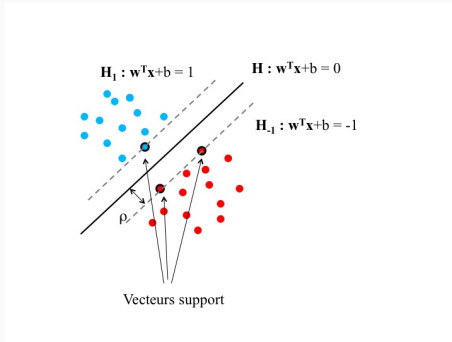
Supposons que les multiplicateurs de Lagrange  $\alpha_i$  soient déterminés :

## Equation d'un SVM linéaire

$$f(\mathbf{x}) = \text{signe}\left(\sum_{i=1}^n \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b\right)$$

Pour classer une donnée  $\mathbf{x}$ , ce classifieur combine linéairement les valeurs de classe  $y_i$  des données support avec des poids du type  $\alpha_i \mathbf{x}_i^T \mathbf{x}$  dépendant de la ressemblance entre  $\mathbf{x}$  et les données support au sens du produit scalaire.

# Vecteurs "supports"

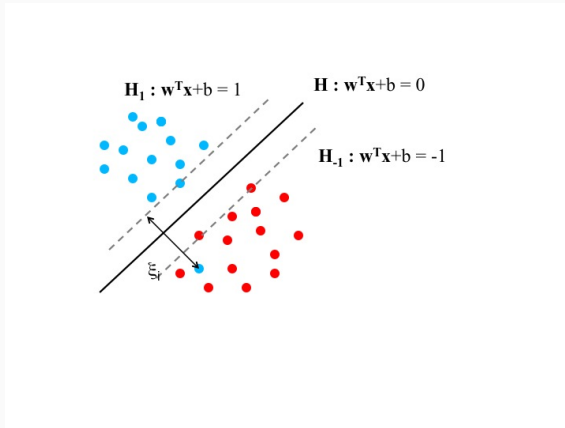


Les données d'apprentissage  $\mathbf{x}_i$  telles que  $\alpha_i \neq 0$  sont sur l'un ou l'autre des hyperplans  $H_1$  ou  $H_{-1}$ . Seules ces données dites *vecteur de support* comptent dans la définition de  $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$

NB :  $b$  est obtenu en choisissant une donnée support ( $\alpha_i \neq 0$ )

## Problème: dans le cas données non séparables

Si on applique un solveur quadratique à des données du type:



alors, l'algorithme ne peut pas réussir à satisfaire les contraintes...

# Cas réaliste: SVM linéaire dans le cas données non séparables

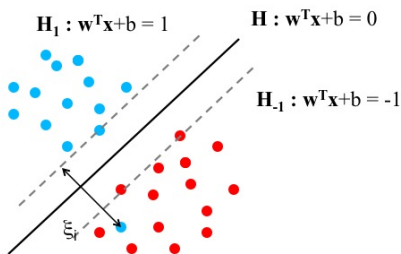
Introduire une variable d'écart  $\xi_i$  pour chaque donnée:

## Problème dans le primal

$$\begin{aligned} \min_{\mathbf{w}, b, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \\ \text{sous les contraintes} \quad & y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \quad i = 1, \dots, n. \\ & \xi_i \geq 0 \quad i = 1, \dots, n. \end{aligned}$$



## Cas réaliste: SVM linéaire dans le cas données non séparables



Notion de marge douce

## Problème dans le dual

$$\begin{aligned} \max_{\alpha} \quad & \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{sous les contraintes} \quad & 0 \leq \alpha_i \leq C \quad i = 1, \dots, n. \\ & \sum_i \alpha_i y_i = 0 \quad i = 1, \dots, n. \end{aligned}$$

# Conditions de Karush-Kuhn-Tucker (KKT)

Soit  $\alpha^*$  la solution du problème dual:

$$\forall i, [y_i f_{w^*, b^*}(x_i) - 1 + \xi_i^*] \leq 0 \quad (1)$$

$$\forall i, \alpha_i^* \geq 0 \quad (2)$$

$$\forall i, \alpha_i^* [y_i f_{w^*, b^*}(x_i) - 1 + \xi_i^*] = 0 \quad (3)$$

$$\forall i, \mu_i^* \geq 0 \quad (4)$$

$$\forall i, \mu_i^* \xi_i^* = 0 \quad (5)$$

$$\forall i, \alpha_i^* + \mu_i^* = C \quad (6)$$

$$\forall i, \xi_i^* \geq 0 \quad (7)$$

$$\mathbf{w}^* = \sum_i \alpha_i^* y_i \mathbf{x}_i \quad (8)$$

$$\sum_i \alpha_i^* y_i = 0 \quad (9)$$

$$(10)$$

## Différents cas de figure

Soit  $\alpha^*$  la solution du problème dual:

- si  $\alpha_i^* = 0$ , alors  $\mu_i^* = C > 0$  et donc,  $\xi_i^* = 0$ :  $x_i^*$  est bien classé mais n'intervient pas dans le paramétrage de la fonction de décision
- si  $0 < \alpha_i^* < C$ , alors  $\mu_i^* > 0$  et donc  $\xi_i^* = 0$ :  $x_i^*$  est sur un des hyperplans  $H_1$  ou  $H_{-1}$ , est support
- si  $\alpha_i^* = C$ , alors  $\mu_i^* = 0$ ,  $\xi_i^* = 1 - y_i f_{w^*, b^*}(x_i)$ ,  $x_i^*$  est support

NB : on calcule  $b^*$  en utilisant un  $i$  tel que  $0 < \alpha_i^* < C$

## Quelques remarques

- certaines données support peuvent donc être de l'autre côté des hyperplans  $H_1$  ou  $H_{-1}$
- $C$  est un hyperparamètre qui contrôle le compromis entre la complexité du modèle et le nombre d'erreurs de classification du modèle.

## Optimisation dans l'espace primal

$$\min_{\mathbf{w}, b} \sum_{i=1}^n (1 - y_i(\mathbf{w}^T \mathbf{x}_i + b))_+ + \lambda \frac{1}{2} \|\mathbf{w}\|^2$$

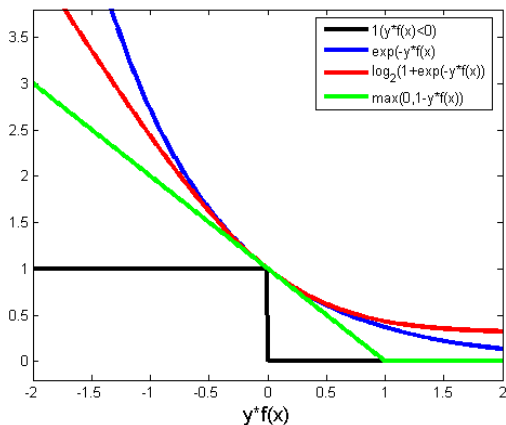
Avec:  $(z)_+ = \max(0, z)$

$f(\mathbf{x}) = \text{signe}(h(\mathbf{x}))$

Fonction de coût: Hinge loss  $L(\mathbf{x}, y, h(\mathbf{x})) = (1 - yh(\mathbf{x}))_+$

$yh(\mathbf{x})$  est appelée marge du classifieur

## Approche régularisée: comparaison des termes de pertes



Rappels

SVM linéaires

Passage au cas non linéaire et noyaux

Définition des noyaux

Retour aux SVM

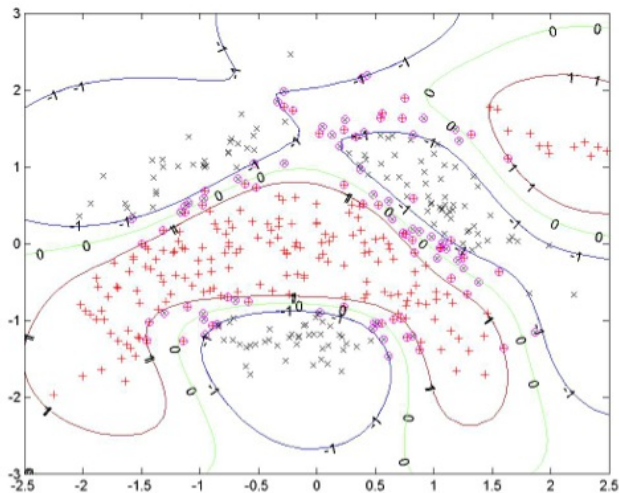
Pour aller plus loin: Support Vector Regression

Conclusion et References



# Support Vector Machine : le cas non linéaire

On voudrait pouvoir sparer non linéairement des données comme ici:



Le problème de l'hyperplan de marge optimale ne fait intervenir les données d'apprentissage qu'à travers de produits scalaires.

$$\max_{\alpha} \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

sous les contraintes  $0 \leq \alpha_i \leq C \quad i = 1, \dots, n.$

$$\sum_i \alpha_i y_i = 1, \dots, n.$$

Si je transforme les données à l'aide d'une fonction  $\phi$  (non linéaire), et que je pose le problème suivant dans l'espace primal:

### Problème dans le primal

$$\min_{\mathbf{w}, b, \xi} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i$$

$$\text{sous les contraintes} \quad y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \quad i = 1, \dots, n.$$
$$\xi_i \geq 0 \quad i = 1, \dots, n.$$

.. je peux apprendre une fonction de séparation non linéaire.

et si je sais calculer les produits scalaires  $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ , je peux résoudre le problème dual suivant:

$$\max_{\alpha} \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

sous les contraintes  $0 \leq \alpha_i \leq C, i = 1, \dots, n.$

$$\sum_i \alpha_i y_i, i = 1, \dots, n.$$

Pour classer une nouvelle donnée  $\mathbf{x}$ , je n'ai besoin que de savoir calculer  $\phi(\mathbf{x}_i)^T \phi(\mathbf{x})$ :

$$f(\mathbf{x}) = \text{signe}(\sum_i \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + b)$$

Au lieu de définir  $\phi$ , je remplace  $\mathbf{x}_i^T \mathbf{x}_j$  par l'image par une fonction  $k$  :  $k(\mathbf{x}_i, \mathbf{x}_j)$  telle qu'il existe un espace de re-description (*feature space*)  $\mathcal{F}$  et une fonction de re-description (*feature map*)  $\phi : \mathcal{X} \rightarrow \mathcal{F}$  et  $\forall(\mathbf{x}, \mathbf{x}') \in \mathcal{X}, k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$ , alors je peux appliquer le même algorithme d'optimisation (résolution dans le dual) et j'obtiens :

$$f(\mathbf{x}) = \text{signe}(\sum_{i=1}^n \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b)$$

Des telles fonctions  $k$  existent et sont appelées *noyaux*.

Rappels

SVM linéaires

Passage au cas non linéaire et noyaux

Définition des noyaux

Retour aux SVM

Pour aller plus loin: Support Vector Regression

Conclusion et References

## Définition

Soit  $\mathcal{X}$  un ensemble. Soit  $k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ , une fonction symétrique. La fonction  $k$  est appelée *noyau* positif défini si et seulement si quel que soit le sous-ensemble fini  $\{\mathbf{x}_1, \dots, \mathbf{x}_m\}$  de  $\mathcal{X}$  et le vecteur colonne  $\mathbf{c}$  de  $\mathbb{R}^m$ ,

$$\mathbf{c}^T K \mathbf{c} = \sum_{i,j=1}^m c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

N.B.: on impose donc que toute matrice construite à partir d'un nombre fini d'éléments de  $\mathcal{X}$  soit semi-définie positive.

## Théorème de Moore-Aronzajn (simplifié)

Soit  $K$  un noyau positif défini. Alors, il existe un espace de Hilbert  $\mathcal{F}$ , appelé *espace de redescription* et une fonction  $\phi : \mathcal{X} \rightarrow \mathcal{F}$ , appelée fonction de redescription (en anglais, feature map) telle que:

$$\langle \phi(x), \phi(x') \rangle_{\mathcal{F}} = k(x, x').$$

N.B.: pour un noyau  $k$ , il peut exister plusieurs couples  $(\mathcal{F}, \phi)$ .



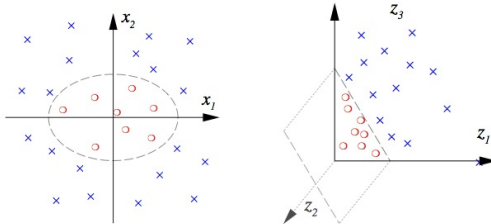
## Noyaux entre vecteurs

$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^p$

- Noyau linéaire :  $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$
- Noyau polynomial :  $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^d$
- Noyau gaussien :  $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$

## Exemple : noyau polynomial

$$\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2} x_1 x_2, x_2^2)$$



## Exemple : noyau polynomial

### Astuce du noyau

On remarque que  $\phi(\mathbf{x}_1)^T \phi(\mathbf{x}')$  peut se calculer sans travailler dans  $\mathbb{R}^3$

Je peux définir  $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}') = (\mathbf{x}^T \mathbf{x}')^2$

## Propriétés de fermeture

Soient  $k_1$  et  $k_2$  deux noyaux sur  $\mathcal{X} \times \mathcal{X}$ . Soit  $g : \mathcal{X} \rightarrow \mathbb{R}$ . Soit  $a \in \mathbb{R}$ .

$$k(x, x') = k_1(x, x') + k_2(x, x')$$

$$k(x, x') = ak_1(x, x')$$

$$k(x, x') = k_1(x, x')k_2(x, x')$$

$$k(x, x') = g(x)g(x')$$

$$k(x, x') = k_1(g(x), g(x'))$$

Les fonctions  $k$  ainsi définis sont des noyaux.

On peut construire des noyaux pour des données structurées : graphes, séquences, arbres et appliquer les SVM !

- Classifier des molécules
- Classifier des documents structurés
- Traiter des séquences biologiques
- ...

Rappels

SVM linéaires

Passage au cas non linéaire et noyaux

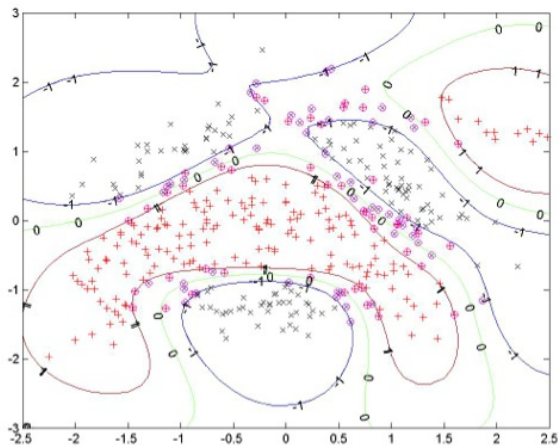
Définition des noyaux

Retour aux SVM

Pour aller plus loin: Support Vector Regression

Conclusion et References

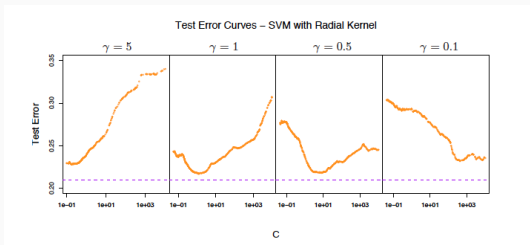
# Support Vector Machine : séparateur non linéaire par noyau gaussien



$$f(\mathbf{x}) = \text{sign}(\sum_{i=1}^n \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b)$$

# Cas du noyau gaussien : rôle de $\gamma$

En plus de l'hyperparamètre  $C$ , il faudra aussi régler la valeur du paramètre  $\gamma$



Dans chacun des cas, on trouve un minimum différent pour  $C$ .



# Comment définir un noyau pour une application spécifique ?

- Utiliser les propriétés de fermeture pour construire de nouveaux noyaux
- Important: les noyaux peuvent être utilisés pour comparer de différents types de données
  - **Données structurées:** (sets), graphs, trees, sequences, ...
- **Apprentissage de noyaux:**
  - Apprentissage d'hyperparamètre: voir Chapelle et al. 2002
  - Apprentissage de noyau multiple: étant donnés  $k_1, \dots, k_m$ , apprendre une combinaison convexe  $\sum_i \beta_i k_i$  de noyaux (voir SimpleMKL Rakotomamonjy et al. 2008, ou Kloft et al. 2010)

Rappels

SVM linéaires

Passage au cas non linéaire et noyaux

Pour aller plus loin: Support Vector Regression

Conclusion et References

## Cadre probabiliste et statistique

Soit  $X$  un vecteur aléatoire de  $\mathcal{X} = \mathbb{R}^p$

$Y$  une variable aléatoire continue  $\mathcal{Y} = \mathbb{R}$

Soit  $P$  la loi de probabilité jointe de  $(X, Y)$ , loi fixée mais inconnue

Supposons que  $S_{app} = \{(x_i, y_i), i = 1, \dots, n\}$  soit un échantillon i.i.d. tiré de la loi  $P$

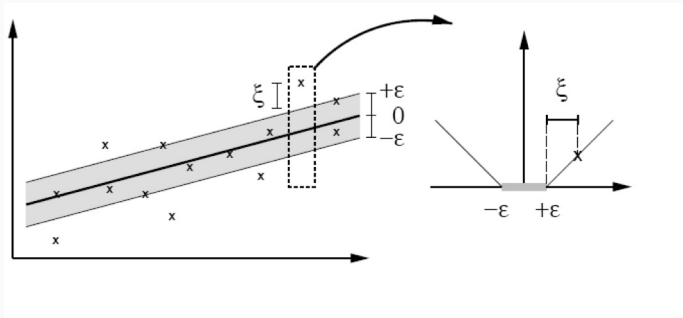
## Cadre probabiliste et statistique

- A partir de  $S_{app}$ , déterminer la fonction  $f \in \mathcal{F}$  qui minimise  $R(f) = \mathbb{E}_P[\ell(X, Y, f(X))]$
- $\ell$  étant une fonction de coût local qui mesure à quel point la vraie cible et la prédiction par le classifieur sont différentes

Pb: la loi jointe n'est pas connue : on ne peut pas calculer  $R(f)$

# Support Vector Regression

- Extend the idea of maximal soft margin to regression
- Impose an  $\epsilon$ -tube : perte  $\epsilon$ -insensible  $|y' - y|_\epsilon = \max(0, |y' - y| - \epsilon)$



# Support Vector Regression

## SVR in the primal space

Given  $C$  and  $\epsilon$

$$\min_{w,b,\xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i (\xi_i + \xi_i^*)$$

s.c.

$$\forall i = 1, \dots, n, y_i - f(x_i) \leq \epsilon + \xi_i$$

$$\forall i = 1, \dots, n, f(x_i) - y_i \leq \epsilon + \xi_i^*$$

$$\forall i = 1, \xi_i \geq 0, \xi_i^* \geq 0$$

$$\text{with } f(x) = w^T \phi(x) + b$$

General case :  $\phi$  is a feature map associated with a positive definite kernel  $k$ .

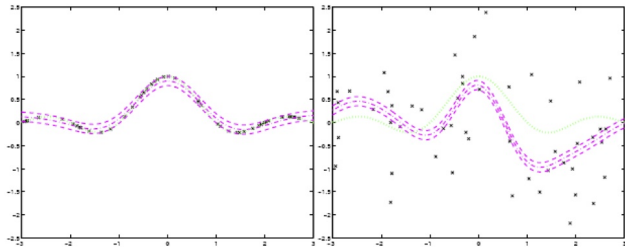
## Solution in the dual

$$\begin{aligned} \min_{\alpha, \alpha^*} \quad & \sum_{i,j} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(x_i, x_j) + \epsilon \sum_i (\alpha_i + \alpha_i^*) - \sum_i y_i (\alpha_i - \alpha_i^*) \\ \text{s.c.} \quad & \sum_i (\alpha_i - \alpha_i^*) = 0 \text{ and } 0 \leq \alpha_i \leq C \text{ and } 0 \leq \alpha_i^* \leq C \\ & w = \sum_{i=1}^n (\alpha_i - \alpha_i^*) \phi(x_i) \end{aligned}$$

**Solution**

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

# Support Vector Regression: example in 1D



*Identical* machine parameters ( $\epsilon = 0.2$ ), but different amounts of noise in the data.

B. Schölkopf, Canberra, February 2002



Rappels

SVM linéaires

Passage au cas non linéaire et noyaux

Pour aller plus loin: Support Vector Regression

Conclusion et References

# Conclusion 1

Classification supervisée par Support Vector Machine:

## Avantages

- 1 unique minimum, programmation quadratique sous contraintes linéaires
- Avec noyau universel comme le noyau Gaussien : c'est un approximateur universel
- Flexible : adapter le noyau au type de données, souvent l'unique solution dans le cas de données structurées
- Multi-classe: M classifieurs une classe contre toutes les autres, on prend le meilleur score (marge).
- Statistique : algorithme inspiré des résultats théoriques de Vapnik et Chervonenkis

Classification supervisée par Support Vector Machine:

### Désavantages

- Choix du noyau, pas d'apprentissage du noyau en tant que tel, seulement sélection de poids d'un noyau multiple
- Algorithme d'optimisation coûteux en temps et en mémoire
- Pour vraiment passer à l'échelle: passer par des approximations Nystrom (approximation de rang faible de la matrice de Gram) ou Random Fourier Features (approximation spectrale du noyau)

Astuce du noyau en général

- s'applique à d'autres algorithmes
- PCA  $\rightarrow$  Kernel PCA
- CCA  $\rightarrow$  Kernel CCA

S'applique en sortie !

- traiter des fonctions à sorties complexes et non à valeurs réelles
- requiert d'utiliser en entrée des noyaux à valeurs opérateurs (ex:matrices)

# References

## Historiques:

- BOSER, Bernhard E., Isabelle M. GUYON, and Vladimir N. VAPNIK, 1992. A training algorithm for optimal margin classifiers. In: COLT 92: Proceedings of the Fifth Annual Workshop on Computational Learning Theory. New York, NY, USA: ACM Press.
- CORTES, Corinna, and Vladimir VAPNIK, 1995. Support-vector networks. Machine Learning, 20(3).

## Revue:

- Article vraiment sympa, complet (un peu de maths) : [A tutorial review of RKHS methods in Machine Learning, Hofman , Schoelkopf, Smola, 2005](https://www.researchgate.net/publication/228827159_A_Tutorial_Review_of_RKHS_Methods_in_Machine_Learning)  
([https://www.researchgate.net/publication/228827159\\_A\\_Tutorial\\_Review\\_of\\_RKHS\\_Methods\\_in\\_Machine\\_Learning](https://www.researchgate.net/publication/228827159_A_Tutorial_Review_of_RKHS_Methods_in_Machine_Learning))