

# Noyaux, SVM

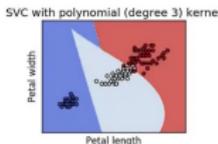
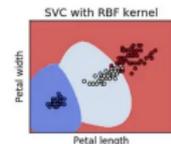
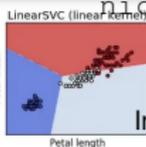
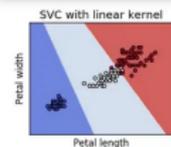
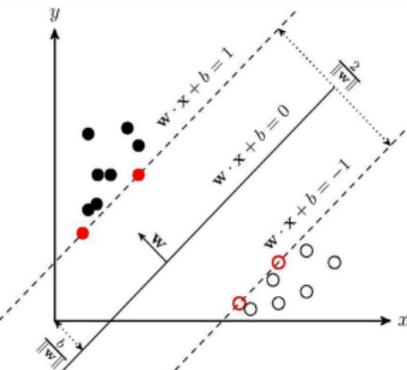
Cours 5  
ML Master DAC

Nicolas Baskiotis

[nicolas.baskiotis@sorbonne-universite.fr](mailto:nicolas.baskiotis@sorbonne-universite.fr)

équipe MLIA,  
Institut des Systèmes Intelligents et de Robotique (ISIR)  
Sorbonne Université

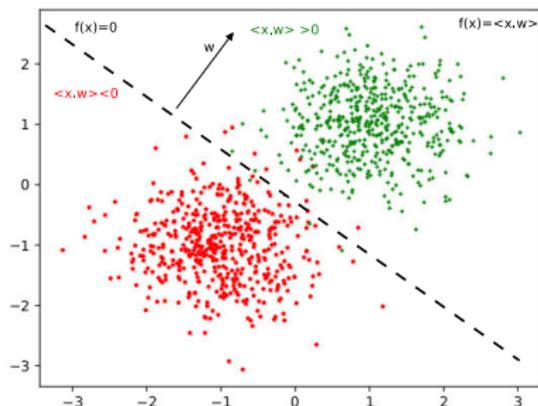
S2 (2024-2025)



# Plan

- 1 Retour sur le perceptron
- 2 Support Vector Machine : principe
- 3 Intro à l'optimisation sous contraintes
- 4 SVM : l'optimisation
- 5 The Kernel Trick - le tour de passe-passe non linéaire

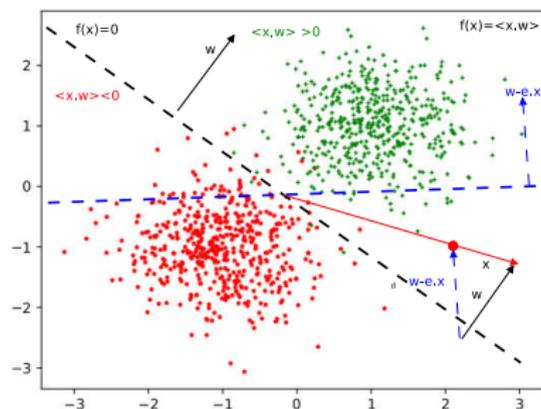
# Rappel de l'algorithme



## Principe :

- Séparateur linéaire :  $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle + b$
- Algorithme d'apprentissage :  
Répéter :
  - ▶ Tirer  $(\mathbf{x}, y)$  au hasard
  - ▶ Si  $yf(\mathbf{x}) > 0$  ne rien faire (point bien classé)
  - ▶ Sinon  $\mathbf{w} = \mathbf{w} + y\mathbf{x}$

# Rappel de l'algorithme

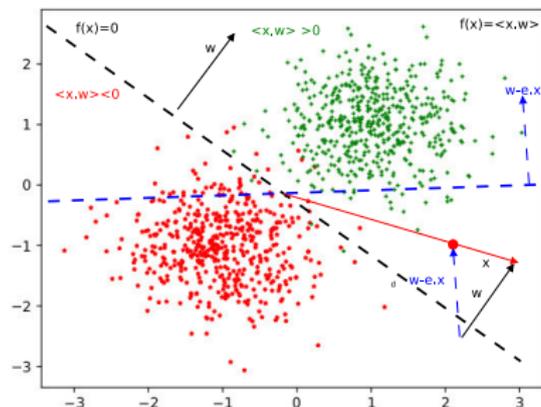


## Considérations géométriques

Que représente :

- $w$  par rapport à la séparatrice ?
- $\langle w, x \rangle$  ?
- la règle de mise à jour : si  $(y\langle w, x \rangle) < 0$  corriger  $w = w + yx$  ?

# Rappel de l'algorithme



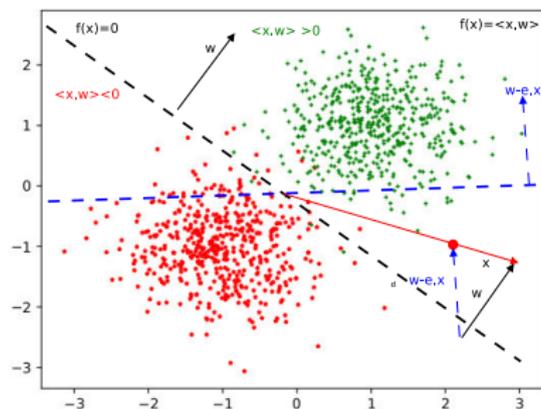
## Considérations géométriques

Que représente :

- $w$  par rapport à la séparatrice ?  $\Rightarrow$  la normale à l'hyper plan
- $\langle w, x \rangle = \cos(\theta) \|x\| \|w\|$ , angle entre les deux vecteurs.
- la règle de mise à jour : si  $(\langle w, x \rangle) < 0$  corriger  $w = w + yx$  ?

$\Rightarrow \langle w + yx, x \rangle = \langle w, x \rangle + y\|x\|^2$ , permet donc d'augmenter ou de diminuer l'angle

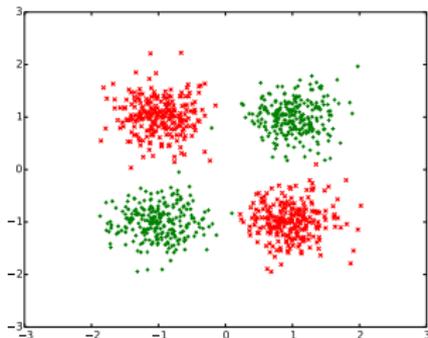
# Rappel de l'algorithme



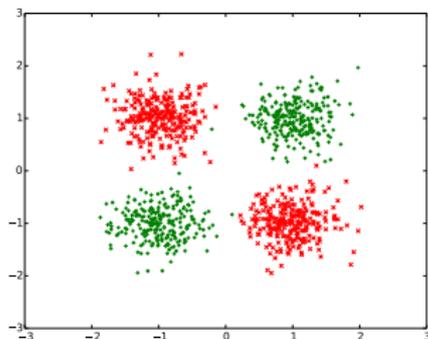
## Questions

- Solution unique ?
- Certaines solutions meilleures que d'autres ?

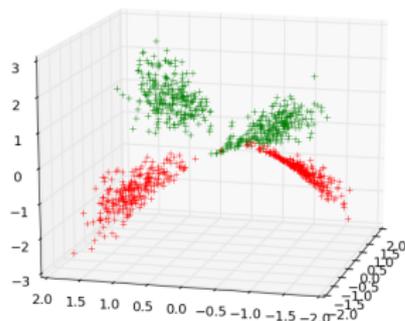
# Données non séparables linéairement



# Données non séparables linéairement



$x_1, x_2$



$x_1, x_2$  et  $x_3 = x_1 * x_2$

## Solutions

- Utiliser des fonctions non-linéaires (réseau de neurones)
- Augmenter les dimensions : projection des données dans un espace de dimension supérieure

# Projection des données

## Exprimer les données dans un espace de plus grande dimension

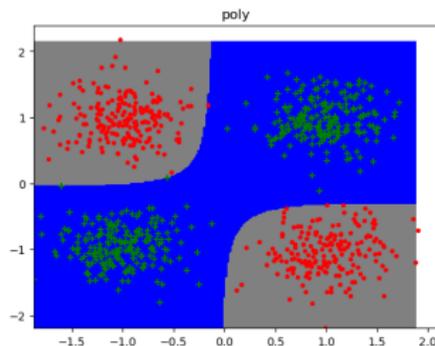
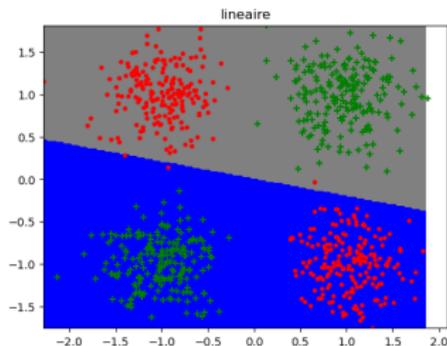
- On considère une fonction  $\phi(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$ , généralement  $d' \gg d$ .
- Nouvel ensemble d'apprentissage :  $\{(\mathbf{z}^k, y^k)\}_{k=1}^N$  avec  $\mathbf{z}^i = \phi(\mathbf{x}^i)$
- Modèle linéaire :  $\sum_{i=1}^{d'} w_i z_i = \sum_{i=1}^{d'} w_i \phi(x_i)$
- Même algo d'optimisation : descente de gradient avec le coût perceptron

## Exemple : Projection polynomiale

On considère  $\mathbf{x} \in \mathbb{R}^3$ , pour une projection de degré 2 :

- $\phi(\mathbf{x}) = (x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3, x_1^2, x_2^2, x_3^2) \in \mathbb{R}^9$
- Nouvelle forme de la frontière de décision :  
 $w_1x_1 + w_2x_2 + w_3x_3 + w_4x_1x_2 + w_5x_1x_3 + w_6x_2x_3 + w_7x_1^2 + w_8x_2^2 + w_9x_3^2 = 0$

# Projection des données



## Exemple : Projection polynomiale

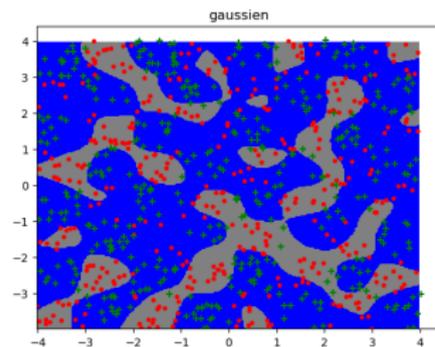
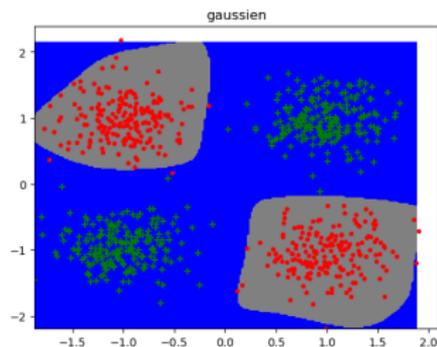
On considère  $\mathbf{x} \in \mathbb{R}^3$ , pour une projection de degré 2 :

- $\phi(\mathbf{x}) = (x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3, x_1^2, x_2^2, x_3^2) \in \mathbb{R}^9$

- Nouvelle forme de la frontière de décision :

$$w_1x_1 + w_2x_2 + w_3x_3 + w_4x_1x_2 + w_5x_1x_3 + w_6x_2x_3 + w_7x_1^2 + w_8x_2^2 + w_9x_3^2 = 0$$

# Projection des données



## Exemple : Projection Gaussienne sur une base

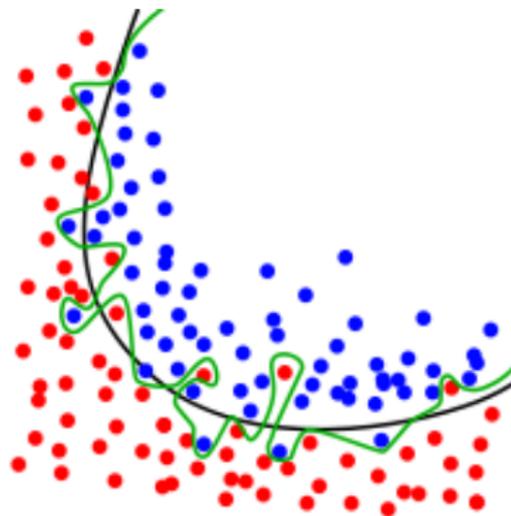
On considère une base  $\{\mathbf{x}^{'j}\}_{j=1}^M$  de point dans  $\mathbb{R}^d$ , la projection d'un exemple  $\mathbf{x}$  est constitué des  $M$  distances gaussiennes aux points  $\mathbf{x}^{'j}$  de la base :

- $\phi(\mathbf{x}) = (e^{-\|\mathbf{x}-\mathbf{x}'^1\|^2/2\sigma}, e^{-\|\mathbf{x}-\mathbf{x}'^2\|^2/2\sigma}, \dots, e^{-\|\mathbf{x}-\mathbf{x}'^M\|^2/2\sigma})$
- Nouvelle forme de la frontière de décision :  
 $w_1 e^{-\|\mathbf{x}-\mathbf{x}'^1\|^2/2\sigma} + w_2 e^{-\|\mathbf{x}-\mathbf{x}'^2\|^2/2\sigma} + \dots + w_M e^{-\|\mathbf{x}-\mathbf{x}'^M\|^2/2\sigma} = 0$
- Seuls les points  $\mathbf{x}'$  proche de  $\mathbf{x}$  jouent un rôle (et donc seuls les poids associés).
- Très proche d'un  $k$ -NN avec prise en compte de la distance, **mais** les poids sont appris automatiquement !

# Projection des données

## Oui mais ...

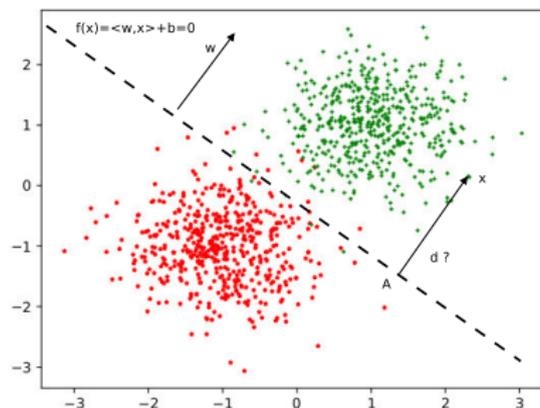
- Quelle projection ?
- Et le sur-apprentissage ?
- Et les “mauvaises” données (le bruit) ?



# Plan

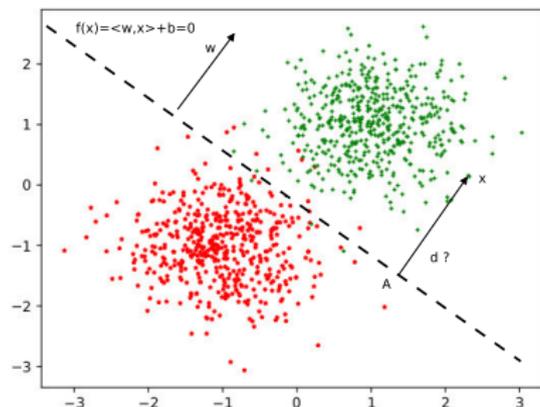
- 1 Retour sur le perceptron
- 2 Support Vector Machine : principe**
- 3 Intro à l'optimisation sous contraintes
- 4 SVM : l'optimisation
- 5 The Kernel Trick - le tour de passe-passe non linéaire

# Considérations géométriques



## Distance géométrique à la séparatrice

# Considérations géométriques

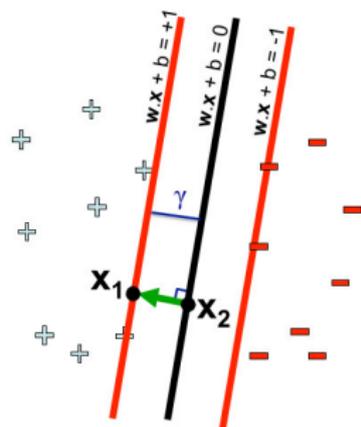


## Distance géométrique à la séparatrice

- $d = |AX|$ , or  $X = A + d \frac{\mathbf{w}}{\|\mathbf{w}\|}$  (pour un exemple au dessus de la séparatrice)
- De plus,  $f(A) = \langle \mathbf{w}, A \rangle + b = 0$ , donc  $\langle \mathbf{w}, X - d \frac{\mathbf{w}}{\|\mathbf{w}\|} \rangle + b = 0$
- Soit  $d = \frac{\langle \mathbf{w}, X \rangle + b}{\|\mathbf{w}\|} = \frac{f(\mathbf{x})}{\|\mathbf{w}\|}$ .
- $f(\mathbf{x})$  est la distance *fonctionnelle* à la séparatrice

Remarque :  $\mathbf{w}$  est défini à une constante multiplicative près...

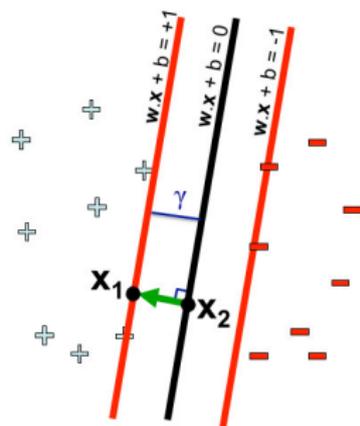
# Unicité de la solution



## Si séparable

- Introduction d'une contrainte : la distance fonctionnelle aux points les plus proches est fixée à 1
- $\Rightarrow f(\mathbf{x}) = \mathbf{w}\mathbf{x}_1 + b = \pm 1$
- Tous les points sont tq :  $yf(\mathbf{x}) \geq 1$ .
  - On note  $\gamma$  la distance euclidienne :  $\mathbf{x}_1 - \mathbf{x}_2 = \gamma \frac{\mathbf{w}}{\|\mathbf{w}\|}$
  - $\gamma$  : distance entre l'hyperplan (frontière) et le point  $\mathbf{x}_1$  le plus proche
  - $\gamma$  est appelé la marge (symétrique ? pourquoi ?)

# Se donner de la marge



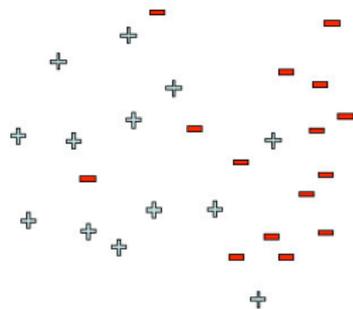
## Si séparable

- $f(\mathbf{x}) = \mathbf{w}\mathbf{x}_1 + b = \pm 1$  et  $\mathbf{x}_1 - \mathbf{x}_2 = \gamma \frac{\mathbf{w}}{\|\mathbf{w}\|}$
  - On a  
 $f(\mathbf{x}_1) - f(\mathbf{x}_2) = 1 = (\mathbf{w}\mathbf{x}_1 + b) - (\mathbf{w}\mathbf{x}_2 + b) = \mathbf{w}(\mathbf{x}_1 - \mathbf{x}_2)$
  - Soit  $\gamma \frac{\mathbf{w}\mathbf{w}}{\|\mathbf{w}\|} = 1$ , donc  $\gamma \|\mathbf{w}\| = 1$ , soit  $\gamma = \frac{1}{\|\mathbf{w}\|}$
- ⇒ Maximiser la marge  $\Leftrightarrow$  minimiser  $\|\mathbf{w}\|$  !
- Nouvelle formulation :  
minimiser  $\|\mathbf{w}\|^2$  tel que

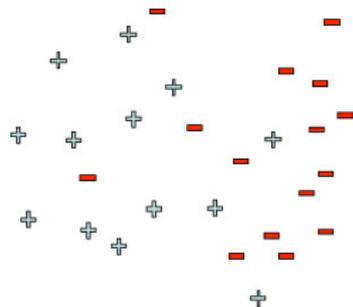
$$\forall i, (\mathbf{w}\mathbf{x}^i + b)y^i \geq 1$$

Problème d'optimisation quadratique convexe

# Et si les données sont bruitées ?



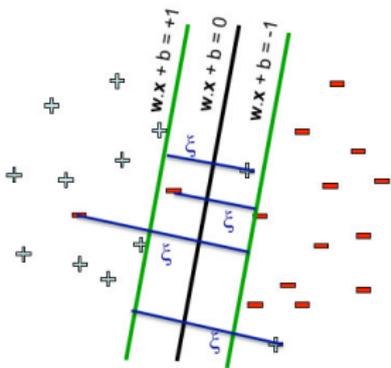
# Et si les données sont bruitées ?



## Prendre en compte les erreurs

- Minimiser  $\|\mathbf{w}\| + K\#\text{Erreurs}$   
tel que  $(\mathbf{w}\mathbf{x}^i + b)y^i \geq 1$
- Problème NP difficile (et les problèmes inhérents au coût 0-1).

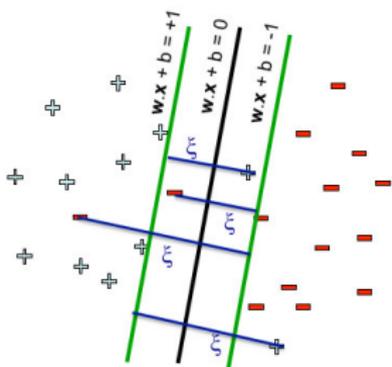
# Et si les données sont bruitées ?



## Approche Support Vector Machine

- Introduire des variables “ressorts” (slack)  $\xi^i$  : on tolère une “débordement”  $(\mathbf{w}\mathbf{x}^i + b)y^i \geq 1 - \xi_i$ , avec  $\xi_i \geq 0$ .
  - Ce “débordement” doit être le plus petit possible
- ⇒ Minimiser  $\|\mathbf{w}\|^2 + K \sum \xi_i$   
tq  $(\mathbf{w}\mathbf{x}^i + b)y^i \geq 1 - \xi_i$  et  $\xi_i \geq 0$
- Si la marge est plus grande que 1 → pas de coût de débordement, sinon coût linéaire :  
$$\begin{cases} \xi_i = 0 & \text{si } (\mathbf{w}\mathbf{x}^i + b)y^i \geq 1 \\ \xi_i = 1 - (\mathbf{w}\mathbf{x}^i + b)y^i & \text{si } (\mathbf{w}\mathbf{x}^i + b)y^i < 1 \end{cases}$$
- ⇒  $\xi_i = \max(0, 1 - (\mathbf{w}\mathbf{x}^i + b)y^i)$  (hinge-loss !)
- Pourquoi la constante  $K$  ? Comment la choisir ?

# Et si les données sont bruitées ?



## Approche Support Vector Machine

- Introduire des variables “ressorts” (slack)  $\xi^i$  : on tolère une “débordement”  $(\mathbf{w}\mathbf{x}^i + b)y^i \geq 1 - \xi_i$ , avec  $\xi_i \geq 0$ .
  - Ce “débordement” doit être le plus petit possible
- ⇒ Minimiser  $\|\mathbf{w}\|^2 + K \sum \xi_j$   
tq  $(\mathbf{w}\mathbf{x}^i + b)y^i \geq 1 - \xi_i$  et  $\xi_i \geq 0$
- Si la marge est plus grande que 1 → pas de coût de débordement, sinon coût linéaire :  
$$\begin{cases} \xi_i = 0 & \text{si } (\mathbf{w}\mathbf{x}^i + b)y^i \geq 1 \\ \xi_i = 1 - (\mathbf{w}\mathbf{x}^i + b)y^i & \text{si } (\mathbf{w}\mathbf{x}^i + b)y^i < 1 \end{cases}$$
- ⇒  $\xi_i = \max(0, 1 - (\mathbf{w}\mathbf{x}^i + b)y^i)$  (hinge-loss !)
- Pourquoi la constante  $K$  ? Comment la choisir ?

## Formulation

- Minimiser :  $\|\mathbf{w}\|^2 + K \sum \ell(y^i, \mathbf{w}\mathbf{x}^i + b)$ , avec  $\ell(y, \hat{y}) = \max(0, 1 - \hat{y})$
- $\|\mathbf{w}\|^2$  → terme de régularisation pour contrôler le sur-apprentissage.

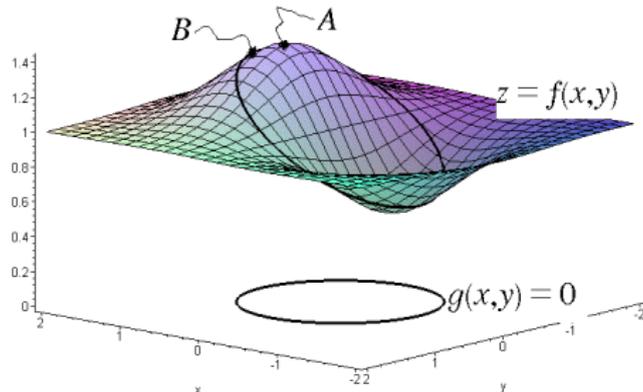
# Plan

- 1 Retour sur le perceptron
- 2 Support Vector Machine : principe
- 3 Intro à l'optimisation sous contraintes**
- 4 SVM : l'optimisation
- 5 The Kernel Trick - le tour de passe-passe non linéaire

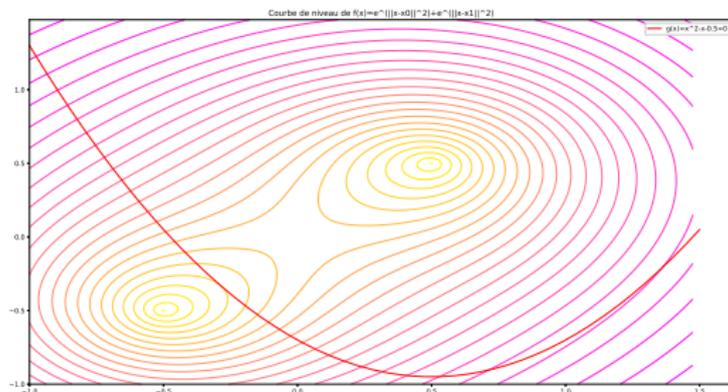
# Optimisation avec contraintes

Permet de résoudre les problèmes de type :

minimiser  $f(\mathbf{x})$  avec un ensemble de contraintes  $c_i(\mathbf{x}) \leq 0$



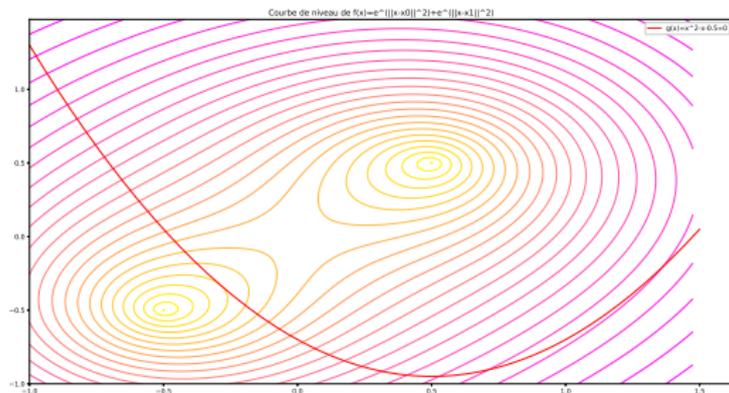
# Optimisation avec contraintes d'égalité



## Formulation et intuition

- Problème du type :  $\min_{\mathbf{x}} f(\mathbf{x})$  tq  $g(\mathbf{x}) = 0$

# Optimisation avec contraintes d'égalité



## Formulation et intuition

- Problème du type :  $\min_{\mathbf{x}} f(\mathbf{x})$  tq  $g(\mathbf{x}) = 0$
- Au point optimal  $\mathbf{x}_0$ ,  $\nabla f(\mathbf{x}_0) = \lambda \nabla g(\mathbf{x}_0)$ , les gradients sont alignés
  - ▶ soit  $\mathbf{x}_0$  est un minimum de  $f \rightarrow \lambda = 0$
  - ▶ soit en suivant  $g$ , la valeur de  $f$  ne change pas  $\rightarrow g$  tangente à l'isocourbe de  $f$

# Un outil magique : le lagrangien

## Multiplicateurs de Lagrange

- Fonction auxiliaire :  $\mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda g(\mathbf{x})$  dont on cherche l'optimum ( $\lambda$  : multiplicateur de Lagrange)
- On cherche  $\nabla \mathcal{L}_{\mathbf{x}, \lambda}(\mathbf{x}, \lambda) = 0$ , soit
  - ▶  $\frac{\partial \mathcal{L}}{\partial \lambda} = 0 = g(\mathbf{x})$  (contrainte d'égalité)
  - ▶  $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda) = 0, \quad \nabla_{\mathbf{x}} f(\mathbf{x}) = \lambda \nabla_{\mathbf{x}} g(\mathbf{x})$

## Remarques

- Condition nécessaire mais pas suffisante ! (le signe du déterminant du Hessien donne la condition suffisante)
- Généralisable à un nombre quelconques de contraintes : introduire autant de multiplicateurs que de contraintes
- Avantage : problème avec contraintes  $\Rightarrow$  problème sans contraintes ! (mais au prix de nouvelles variables)

# Optimisation avec contraintes d'inégalité

## Formulation

$\min_{\mathbf{x}} f(\mathbf{x})$  tel que  $c_1(\mathbf{x}) \leq 0, \dots, c_n(\mathbf{x}) \leq 0$  et  $g_1(\mathbf{x}) = 0, \dots, g_m(\mathbf{x}) = 0$

## Multiplicateurs de Lagrange - formulation duale

- Pour chaque contrainte d'inégalité  $c_i$ , on introduit une variable  $\lambda_i \geq 0$
- Pour chaque contrainte d'égalité  $g_j$ , on introduit une variable  $\mu_j \in \mathbb{R}$
- Formulation duale :  $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) + \sum_i \lambda_i c_i(\mathbf{x}) + \sum_j \mu_j g_j(\mathbf{x})$
- $\min_{\mathbf{x}} f(\mathbf{x}) \text{ tq } c_i(\mathbf{x}) \leq 0, g_j(\mathbf{x}) = 0 \iff \max_{\boldsymbol{\lambda}, \boldsymbol{\mu}} \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$

## Conditions nécessaires d'optimalité de Karush Kuhn Tucker (KKT)

Si  $(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$  est optimal, alors

- $\nabla \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = 0$  (stationarité)
- $\forall i c_i(\mathbf{x}^*) \leq 0, \forall j g_j(\mathbf{x}^*) = 0$  (admissibilité primale)
- $\forall i \lambda_i^* \geq 0$  (admissibilité duale)
- $\lambda_i^* c_i(\mathbf{x}^*) = 0$  (complémentarité) : si  $c_i(\mathbf{x}^*) < 0$ , alors  $\lambda_i = 0$  (contrainte inactive), sinon  $\lambda_i > 0$  et  $c_i(\mathbf{x}^*) = 0$ .

# Plan

- 1 Retour sur le perceptron
- 2 Support Vector Machine : principe
- 3 Intro à l'optimisation sous contraintes
- 4 SVM : l'optimisation**
- 5 The Kernel Trick - le tour de passe-passe non linéaire

# La recette magique

Dans le cas simple (sans variables slack)

## Le problème primal

minimiser  $\mathbf{w}, b$   $\frac{1}{2} \|\mathbf{w}\|^2$  tel que  $y^i(\mathbf{w}\mathbf{x}^i + b) \geq 1$

## Fonction de Lagrange

- $L(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_i \alpha_i (y^i(\mathbf{w}\mathbf{x}^i + b) - 1)$
- Si  $\mathbf{w}, b$  sont des minimums, alors il existe  $\alpha_i \geq 0$  tel que le gradient du lagrangien soit nul.
- Conditions d'optimalité (Karush Kuhn Tucker) :

$$\alpha_i (y^i(\mathbf{w}\mathbf{x}^i + b) - 1) = 0 \rightarrow \begin{cases} \alpha_i = 0 \\ \alpha_i > 0 \Rightarrow (y^i(\mathbf{w}\mathbf{x}^i + b) - 1) = 0 \end{cases}$$

# Résolution

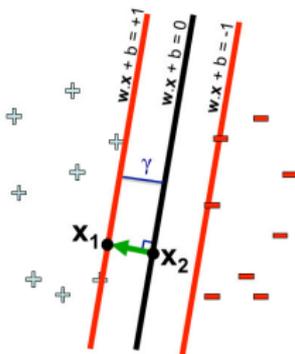
## Lagrangien

- $L(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_i \alpha_i (y^i (\mathbf{w}\mathbf{x}^i + b) - 1)$

## Dérivées

- $\nabla_{\mathbf{w}} L(\mathbf{w}, b, \alpha) = \mathbf{w} - \sum_i \alpha_i y^i \mathbf{x}^i = 0$ , donc  $\mathbf{w} = \sum_i \alpha_i y^i \mathbf{x}^i$
  - $\nabla_b L(\mathbf{w}, b, \alpha) = \sum_i \alpha_i y^i = 0$
- ⇒ maximiser <sub>$\alpha$</sub>   $-\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^i y^j \langle \mathbf{x}^i, \mathbf{x}^j \rangle + \sum_i \alpha_i$   
tel que  $\sum \alpha_i y^i = 0$  et  $\alpha_i \geq 0$ .

# Remarques importantes



- $\mathbf{w} = \sum_i \alpha_i y^i \mathbf{x}^i$  : le vecteur de poids est une combinaison linéaire des exemples d'apprentissage !
- Il y a (même beaucoup) de  $\alpha_i$  qui sont nuls  $\Rightarrow$  exemples non pris en compte (normal ?)
- La fonction de décision résultante :

$$f(\mathbf{x}) = \langle \mathbf{w}\mathbf{x} \rangle + b = \sum_i \alpha_i y^i \langle \mathbf{x}^i \mathbf{x} \rangle + b$$

ne fait intervenir que des produits scalaires entre les exemples d'apprentissage et l'exemple à classifier.

# Dans le cas compliqué (avec slack)

## Le problème primal

minimiser<sub>w,b</sub>  $\frac{1}{2}\|\mathbf{w}\|^2 + K \sum_i \xi_i$  tel que  $y^i(\mathbf{w}\mathbf{x}^i + b) \geq 1 - \xi_i$  et  $\xi_i \geq 0$

## Lagrangien

$$\bullet L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 + K \sum_i \xi_i - \sum_i \alpha_i (y^i(\mathbf{w}\mathbf{x}^i + b) + \xi_i - 1) - \sum_i \eta_i \xi_i$$

## Dérivées

- $\bullet \nabla_{\mathbf{w}} L(\mathbf{w}, b, \alpha, \xi, \eta) = \mathbf{w} - \sum_i \alpha_i y^i \mathbf{x}^i = 0$
  - $\bullet \nabla_b L(\mathbf{w}, b, \alpha, \xi, \eta) = \sum_i \alpha_i y^i = 0$
  - $\bullet \nabla_{\xi} L(\mathbf{w}, b, \alpha, \xi, \eta) = K - \alpha_i - \eta_i = 0$
- $\Rightarrow$  maximiser <sub>$\alpha$</sub>   $-\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^i y^j \langle \mathbf{x}^i, \mathbf{x}^j \rangle + \sum_i \alpha_i$   
tel que  $\sum \alpha_i y^i = 0$  et  $\alpha_i \in [0, K]$ .

- $\bullet$  Conditions d'optimalité (KKT) :

$$\begin{cases} \alpha_i (y^i(\mathbf{w}\mathbf{x}^i + b) + \xi_i - 1) = 0 \\ \eta_i \xi_i = 0 \end{cases} \rightarrow \begin{cases} \alpha_i = 0 \Rightarrow y^i(\mathbf{w}\mathbf{x}^i + b) \geq 1 \\ 0 < \alpha_i < K \Rightarrow (y^i(\mathbf{w}\mathbf{x}^i + b) - 1) = 1 \\ \alpha_i = K \Rightarrow (y^i(\mathbf{w}\mathbf{x}^i + b) - 1) \leq 1 \end{cases}$$

# Plan

- 1 Retour sur le perceptron
- 2 Support Vector Machine : principe
- 3 Intro à l'optimisation sous contraintes
- 4 SVM : l'optimisation
- 5 The Kernel Trick - le tour de passe-passe non linéaire**

# LE détail important

Dans toutes les formulations, ce qui importe c'est le produit scalaire !

- minimiser  $\mathbf{w}, b$   $\frac{1}{2} \|\mathbf{w}\|^2 + K \sum_i \xi_i$  tel que  $y^i(\mathbf{w}\mathbf{x}^i + b) \geq 1 - \xi_i$  et  $\xi_i \geq 0$
- ⇔ maximiser  $\alpha$   $-\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^i y^j \langle \mathbf{x}^i, \mathbf{x}^j \rangle + \sum_i \alpha_i$   
tel que  $\sum \alpha_i y^i = 0$  et  $\alpha_i \in [0, K]$ .
- Sauf dans  $\mathbf{w}^* = \sum_i \alpha_i y^i \mathbf{x}^i$
- mais :  $f(\mathbf{x}) = \sum_i \alpha_i y_i \langle \mathbf{x}^i, \mathbf{x} \rangle + b$

# Pourquoi donc ?

## Non linéarité → projection

- On veut considérer une projection  $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^{n'}$ ,  $n \ll n'$
- Par exemple,  $\phi(\mathbf{x}) = (x_1, x_2, \dots, x_d, x_1x_1, x_1x_2, \dots, x_dx_d)$
- On peut noter :  $K(\mathbf{x}^i, \mathbf{x}^j) = \langle \phi(\mathbf{x}^i), \phi(\mathbf{x}^j) \rangle$
- $\mathbf{w}^* = \sum_i \alpha_i y^i \phi(\mathbf{x}^i)$

## Le SVM fait intervenir les quantités :

Espace de départ

$$\langle \mathbf{x}^1, \mathbf{x}^2 \rangle$$

$$\langle \mathbf{w}, \mathbf{x} \rangle$$

$$f(\mathbf{x}) = \sum_i \alpha_i y^i \langle \mathbf{x}^i, \mathbf{x} \rangle + b$$

Espace projeté

$$\langle \phi(\mathbf{x}^1), \phi(\mathbf{x}^2) \rangle = K(\mathbf{x}^1, \mathbf{x}^2)$$

$$\langle \mathbf{w}, \phi(\mathbf{x}) \rangle = \sum_i \alpha_i y^i \langle \phi(\mathbf{x}^i), \phi(\mathbf{x}) \rangle = \sum_i \alpha_i y^i K(\mathbf{x}^i, \mathbf{x})$$

$$f(\mathbf{x}) = \sum_i \alpha_i y^i \langle \phi(\mathbf{x}^i), \phi(\mathbf{x}) \rangle + b = \sum_i \alpha_i y^i K(\mathbf{x}^i, \mathbf{x}) + b$$

# Projection et produit scalaire

## Exemple : projection polynômiale

Soit  $\phi_2(\mathbf{x}) = (x_0x_0, x_0x_1, \dots, x_0x_d, x_1x_0, x_1x_1, \dots, x_dx_d)$  avec  $x_0 = 1$  :

- $\langle \phi_2(\mathbf{x}) \phi_2(\mathbf{x}') \rangle = \sum_{i=0}^d \sum_{j=0}^d (x_i x_j) (x'_i x'_j)$   
 $= \sum_{i=0}^d \sum_{j=0}^d (x_i x'_i) (x_j x'_j)$   
 $= (x_0 x'_0 + x_1 x'_1 + \dots + x_d x'_d)^2$   
 $= (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^2$   
 $K(\mathbf{x}, \mathbf{x}') = \langle \phi_2(\mathbf{x}), \phi_2(\mathbf{x}') \rangle = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^2$  beaucoup moins chère à calculer (linéaire en  $D$ ).
- Kernel Trick : remplacer le calcul coûteux (et pas toujours possible) de  $\langle \phi_2(\mathbf{x}), \phi_2(\mathbf{x}') \rangle$  par une fonction moins coûteuse  $K(\mathbf{x}, \mathbf{x}')$

# Une projection infinie : le noyau gaussien

Soit  $K(\mathbf{x}, \mathbf{x}') = e^{-\gamma\|\mathbf{x}-\mathbf{x}'\|^2}$  :

$$\begin{aligned}K(\mathbf{x}, \mathbf{x}') &= e^{-\gamma\|\mathbf{x}-\mathbf{x}'\|^2} \\&= e^{-\gamma(\|\mathbf{x}\|^2 + \|\mathbf{x}'\|^2 - 2\mathbf{x}\mathbf{x}')} \\&= e^{-\gamma\|\mathbf{x}\|^2} e^{-\gamma\|\mathbf{x}'\|^2} e^{2\gamma\mathbf{x}\mathbf{x}'} \\&= e^{-\gamma\|\mathbf{x}\|^2} e^{-\gamma\|\mathbf{x}'\|^2} \sum_{n=0}^{\infty} \frac{(2\gamma\mathbf{x}\mathbf{x}')^n}{n!} \\&= \sum_{n=0}^{\infty} \left( \sqrt{\frac{(2\gamma)^n}{n!}} e^{-\gamma\|\mathbf{x}\|^2} \phi_n(\mathbf{x}) \right) \left( \sqrt{\frac{(2\gamma)^n}{n!}} e^{-\gamma\|\mathbf{x}'\|^2} \phi_n(\mathbf{x}') \right)\end{aligned}$$

Avec  $\phi_n$  tel que  $\langle \phi_n(\mathbf{x}) \phi_n(\mathbf{x}') \rangle = (\langle \mathbf{x}\mathbf{x}' \rangle)^n$

$\Rightarrow K$  correspond au produit scalaire d'une concaténation infinie de projection

# Les noyaux

## Définition

- Forme généralisée de produit scalaire :  $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$
- Noyaux admissibles : tous ceux qui peuvent se mettre sous la forme d'un produit scalaire de deux projections (il existe  $\phi$  tel que ...).
- Mathématiquement : fonction semi-définie positive : pour toute fonction  $f$  carré intégrable,  $\int_{x,x'} f(x)k(x,x')f(x')dx dx' > 0$ .
- Ou, sur un échantillon  $\{x^1, \dots, x^n\}$ , si  $k$  est symétrique et pour tout  $c_i \in \mathbb{R}$ ,  $\sum_{i,j} c_i c_j k(x_i, x_j) \geq 0$ .

## Opération

Si  $k, k'$  sont des noyaux, alors sont aussi des noyaux:

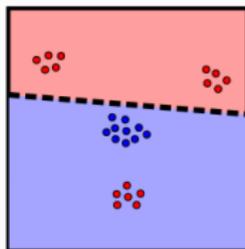
- $k(x, x') + k'(x, x')$
- $k(x, x') * k'(x, x')$
- $k(f(x), f'(x))$
- $f(k(x, x'))$  pour  $f$  polynôme
- $\exp(k(x, x'))$

# Quelques exemples

- Noyau gaussien :  $k(x, x') = \exp(-\|x - x'\|^2 / \sigma^2)$   
 $f(\mathbf{x}) = \sum_i \alpha_i y^i e^{-\frac{\|\mathbf{x}^i - \mathbf{x}\|^2}{\sigma^2}} \Rightarrow$  très proche du  $K$ -NN
- Kernel String pour le texte
- Noyau de convolution :  $k(x, x') = \sum_{w \in x} \sum_{w' \in x'} k'(w, w')$
- Noyau sur les arbres, les graphes ...
- Penser aux noyaux comme une mesure de similarité entre deux objets !
  - ▶ si éloignés  $\rightarrow 0$  (produit scalaire orthogonal)
  - ▶ si proches  $\rightarrow$  valeur maximale (vecteurs alignés)
  - ▶ si opposés  $\rightarrow$  valeur négative minimale (vecteurs opposés)

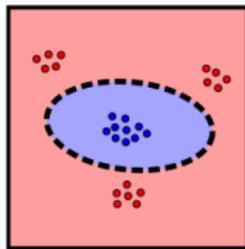
Noyau linéaire

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$$



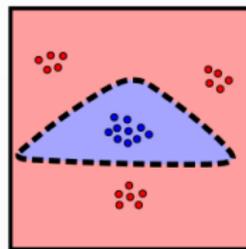
Noyau polynomial  
(degré 2)

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^2$$



Noyau gaussien

$$k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|}$$



# Conclusion

- Mythe : Les SVMs fonctionnent parce que l'on projette en très haute dimension
- ⇒ alors on aurait besoin de bien plus de données
- Combiné à la contrainte de marge.
  - On retrouve une forme générique des problèmes d'apprentissage :  
$$R(f) = \sum \ell(f(\mathbf{x}^i), y^i) + \Omega(f),$$
  
avec  $\ell$  une fonction de coût (risque empirique) et  $\Omega$  une régularisation sur la complexité de la fonction  $f$ .
  - Permet de régler le sur-apprentissage (ou de manière équivalent de contraindre la classe de fonction considérée).
  - Les noyaux s'adaptent à beaucoup d'autres méthodes (Régression ridge par exemple)