

# Apprentissage automatique

## - Algorithme du perceptron et machines à vecteurs de support -

Hachem Kadri

Université Aix-Marseille – LIS (Equipe [Qarma](#))

<http://qarma.lis-lab.fr/>



# Plan

Perceptron

SVM

## Classification linéaire binaire

$$X \subset \mathbb{R}^n, Y = \{-1, 1\}$$

**Définition.** Un *classifieur linéaire* est une fonction de la forme

$$f(x) = \begin{cases} 1 & \text{si } \langle w, x \rangle + b \geq 0 \\ -1 & \text{sinon.} \end{cases}$$

où  $w \in \mathbb{R}^n$ ,  $b \in \mathbb{R}$ ,  $\langle w, x \rangle$  désigne le produit scalaire entre  $w$  et  $x$  : si  $w = (w_1, \dots, w_n)$  et  $x = (x_1, \dots, x_n)$ ,  $\langle w, x \rangle = \sum_{i=1}^n w_i x_i$ .

**Interprétation géométrique :**  $\langle w, x \rangle + b = 0$  est l'équation d'un hyperplan qui sépare  $X$  en deux demi-espaces correspondant aux deux classes.

### Exemple

$$X = \mathbb{R}^2$$

Classifieur linéaire défini par  $w = (1, 2)$  et  $b = -1$  :

$$f(x_1, x_2) = \begin{cases} 1 & \text{si } x_1 + 2x_2 - 1 \geq 0 \\ -1 & \text{sinon.} \end{cases}$$

Par exemple,  $f(0, 0) = -1$  et  $f(1, 1) = 1$ .

Hyperplan d'équation  $x_1 + 2x_2 - 1 = 0$

## Expressivité des perceptrons

- ▶ Les classifieurs linéaires peuvent sembler *a priori* très peu expressifs : pourquoi des données naturelles se répartiraient-elles de part et d'autres d'un hyperplan ?
- ▶ Cette intuition n'est pas forcément vérifiée en très grande dimension (cas de classification de textes, par exemple).
- ▶ Cela suggère de plonger les données initiales dans un espace de grande dimension (voir chapitre 5).

*A complex pattern-classification problem, cast in a high-dimensional space nonlinearly, is more likely to be linearly separable than in a low-dimensional space, provided that the space is not densely populated. (T.M. Cover, 1965)*

## Données linéairement séparables

Un échantillon  $S = \{(x_1, y_1), \dots, (x_l, y_l)\} \subset (X \times Y)^l$  est *linéairement séparable* s'il existe un classifieur linéaire qui classe correctement tous les exemples de  $S$ .

### Exemples :

$S = \{((0, 0), -1), ((1, 0), 1), ((0, 1), -1)\}$  est linéairement séparable.

$S = \{((0, 0), -1), ((1, 0), 1), ((0, 1), 1), ((1, 1), -1)\}$  n'est pas linéairement séparable (XOR).

# Perceptrons

- ▶ Perceptrons (Rosenblatt 1958, Minsky/Papert 1969) : généralisation d'un modèle plus simple proposé par (McCulloch/Pitts neurons, 1942)
- ▶ un perceptron à  $n$  entrées est décrit par un vecteur de pondération  $\vec{w} = (w_1, \dots, w_n)^T \in \mathbb{R}^n$ , un seuil  $\theta \in \mathbb{R}$  et permet de calculer la fonction suivante :

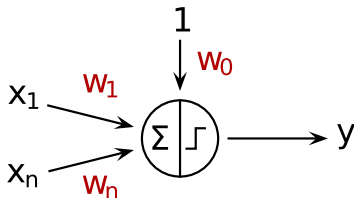
$$(x_1, \dots, x_n) \mapsto y = \begin{cases} 1 & \text{if } x_1 w_1 + x_2 w_2 + \dots + x_n w_n \geq \theta \\ 0 & \text{if } x_1 w_1 + x_2 w_2 + \dots + x_n w_n < \theta \end{cases}$$

## Perceptrons

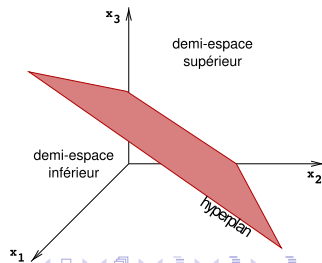
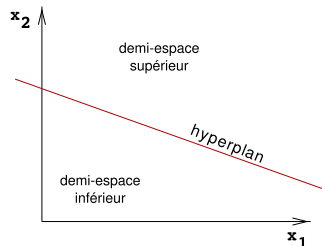
- pour plus de commodité : remplacer le seuil par un poids supplémentaire (**biais**)  $w_0 = -\theta$
- un perceptron avec un vecteur de pondération  $\vec{w}$  et un biais  $w_0$  effectue le calcul suivant :

$$(x_1, \dots, x_n) \mapsto y = f(w_0 + \sum_{i=1}^n (w_i x_i)) = f(w_0 + \langle \vec{w}, \vec{x} \rangle)$$

avec  $f(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$



- ▶ données  $(x_1, \dots, x_n)$   
→  $\in$  un espace de dimension  $n$
- ▶ points vérifiants  $w_0 + \langle \vec{w}, \vec{x} \rangle = 0$   
→ hyperplan défini par  $w_0$  et  $\vec{w}$
- ▶ points vérifiants  $w_0 + \langle \vec{w}, \vec{x} \rangle > 0$   
→ points d'un coté de l'hyperplan
- ▶ points vérifiants  $w_0 + \langle \vec{w}, \vec{x} \rangle < 0$   
→ points de l'autre coté de l'hyperplan
- ▶ un perceptron divise l'espace des données en deux demi-espaces  
→ situés de part et d'autre de l'hyperplan





## algorithme d'apprentissage du perceptron

- ▶ les perceptrons peuvent être automatiquement adaptés à des tâches d'apprentissage  $\Rightarrow$  Apprentissage supervisé : Classification
- ▶ algorithme d'apprentissage du perceptron :  
données :
  - un ensemble de données  $\mathcal{P} \subseteq \mathbb{R}^n$  : exemples positifs
  - un autre ensemble  $\mathcal{N} \subseteq \mathbb{R}^n$  : exemples négatifstache :
  - générer un perceptron qui retourne 1 pour tous les exemples de  $\mathcal{P}$  et 0 pour les exemples de  $\mathcal{N}$
- ▶ évidemment, il y a des cas dans lesquels l'algo d'apprentissage du perceptron n'est pas capable de résoudre le problème de classification
  - $\rightarrow$  exemple :  $\mathcal{P} \cap \mathcal{N} \neq \emptyset$
  - $\rightarrow$  données non linéairement séparables
  - $\rightarrow$  solution potentielle : transférer les données dans un autre espace dans lequel les données sont linéairement séparables (plus de détail, Chap. 5 : Méthodes à noyau)

## algorithme d'apprentissage du perceptron

► **Lemme (séparabilité stricte) :**

Si  $\exists$  un perceptron qui classe parfaitement les données d'apprentissage, alors  $\exists$  un perceptron qui classe ces données sans qu'aucune ne soit sur la frontière de décision,  $w_0 + \langle \vec{w}, \vec{x} \rangle \neq 0$

Preuve :

Soit  $(w_0, \vec{w})$  un perceptron qui classe parfaitement toutes les données d'apprentissage. D'où

$$w_0 + \langle \vec{w}, \vec{x} \rangle \begin{cases} \geq 0 & \forall \vec{x} \in \mathcal{P} \\ < 0 & \forall \vec{x} \in \mathcal{N} \end{cases}$$

Soit  $\epsilon = \min\{-(w_0 + \langle \vec{w}, \vec{x} \rangle) \mid \vec{x} \in \mathcal{N}\}$ . Alors :

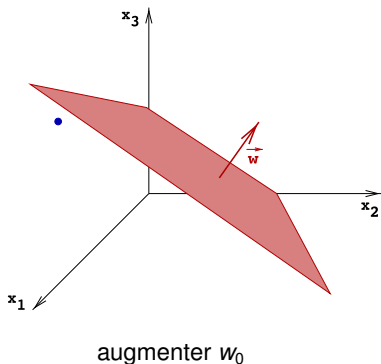
$$w_0 + \frac{\epsilon}{2} + \langle \vec{w}, \vec{x} \rangle \begin{cases} \geq \frac{\epsilon}{2} > 0 & \forall \vec{x} \in \mathcal{P} \\ \leq -\frac{\epsilon}{2} < 0 & \forall \vec{x} \in \mathcal{N} \end{cases}$$

Ainsi le perceptron  $(w_0 + \frac{\epsilon}{2}, \vec{w})$  prouve le lemme.

## Algorithme d'apprentissage du perceptron

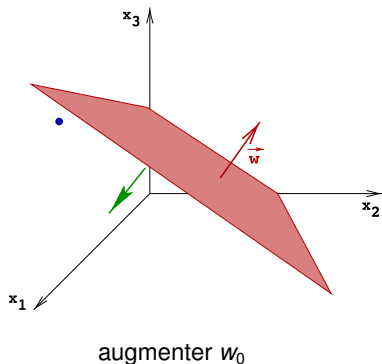
- ▶ Erreur de classification de  $\vec{x} \in \mathcal{P}$   
→  $w_0 + \langle \vec{w}, \vec{x} \rangle < 0$
- ▶ Comment peut-on modifier  $w_0$  et  $\vec{w}$  pour remédier à cette erreur  
→ augmenter  $w_0 + \langle \vec{w}, \vec{x} \rangle$ 
  - augmenter  $w_0$
  - Si  $x_i > 0$  augmenter  $w_i$
  - Si  $x_i < 0$  diminuer  $w_i$
- ▶ Algorithme : ajouter  $\vec{x}$  à  $\vec{w}$  et 1 à  $w_0$   
→ Procéder par analogie pour les exemples négatifs  $\vec{x} \in \mathcal{N}$

- ▶ Erreur de classification de  $\vec{x} \in \mathcal{P}$ 
  - $w_0 + \langle \vec{w}, \vec{x} \rangle < 0$
- ▶ Comment peut-on modifier  $w_0$  et  $\vec{w}$  pour remédier à cette erreur
  - augmenter  $w_0 + \langle \vec{w}, \vec{x} \rangle$ 
    - augmenter  $w_0$
    - Si  $x_i > 0$  augmenter  $w_i$
    - Si  $x_i < 0$  diminuer  $w_i$
- ▶ Algorithme : ajouter  $\vec{x}$  à  $\vec{w}$  et 1 à  $w_0$ 
  - Procéder par analogie pour les exemples négatifs  $\vec{x} \in \mathcal{N}$



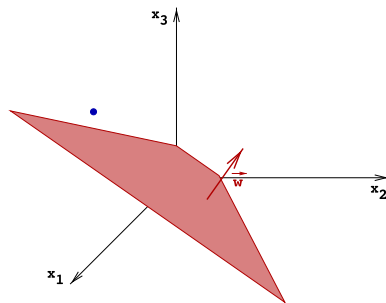
## Algorithme d'apprentissage du perceptron

- ▶ Erreur de classification de  $\vec{x} \in \mathcal{P}$   
 →  $w_0 + \langle \vec{w}, \vec{x} \rangle < 0$
- ▶ Comment peut-on modifier  $w_0$  et  $\vec{w}$  pour remédier à cette erreur  
 → augmenter  $w_0 + \langle \vec{w}, \vec{x} \rangle$ 
  - augmenter  $w_0$
  - Si  $x_i > 0$  augmenter  $w_i$
  - Si  $x_i < 0$  diminuer  $w_i$
- ▶ Algorithme : ajouter  $\vec{x}$  à  $\vec{w}$  et 1 à  $w_0$   
 → Procéder par analogie pour les exemples négatifs  $\vec{x} \in \mathcal{N}$



## Algorithme d'apprentissage du perceptron

- ▶ Erreur de classification de  $\vec{x} \in \mathcal{P}$   
 →  $w_0 + \langle \vec{w}, \vec{x} \rangle < 0$
- ▶ Comment peut-on modifier  $w_0$  et  $\vec{w}$  pour remédier à cette erreur  
 → augmenter  $w_0 + \langle \vec{w}, \vec{x} \rangle$ 
  - augmenter  $w_0$
  - Si  $x_i > 0$  augmenter  $w_i$
  - Si  $x_i < 0$  diminuer  $w_i$
- ▶ Algorithme : ajouter  $\vec{x}$  à  $\vec{w}$  et 1 à  $w_0$   
 → Procéder par analogie pour les exemples négatifs  $\vec{x} \in \mathcal{N}$



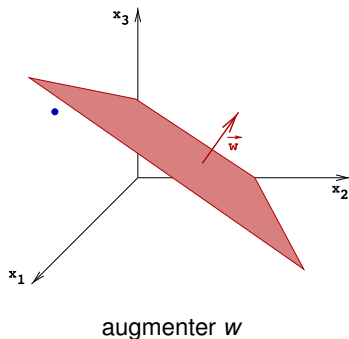
augmenter  $w_0$

## Algorithme d'apprentissage du perceptron

- ▶ Erreur de classification de  $\vec{x} \in \mathcal{P}$   
→  $w_0 + \langle \vec{w}, \vec{x} \rangle < 0$
- ▶ Comment peut-on modifier  $w_0$  et  $\vec{w}$  pour remédier à cette erreur  
→ augmenter  $w_0 + \langle \vec{w}, \vec{x} \rangle$ 
  - augmenter  $w_0$
  - Si  $x_i > 0$  augmenter  $w_i$
  - Si  $x_i < 0$  diminuer  $w_i$
- ▶ Algorithme : ajouter  $\vec{x}$  à  $\vec{w}$  et 1 à  $w_0$   
→ Procéder par analogie pour les exemples négatifs  $\vec{x} \in \mathcal{N}$

## Algorithme d'apprentissage du perceptron

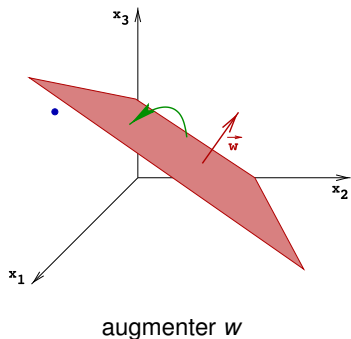
- ▶ Erreur de classification de  $\vec{x} \in \mathcal{P}$   
 →  $w_0 + \langle \vec{w}, \vec{x} \rangle < 0$
- ▶ Comment peut-on modifier  $w_0$  et  $\vec{w}$  pour remédier à cette erreur  
 → augmenter  $w_0 + \langle \vec{w}, \vec{x} \rangle$ 
  - augmenter  $w_0$
  - Si  $x_i > 0$  augmenter  $w_i$
  - Si  $x_i < 0$  diminuer  $w_i$
- ▶ Algorithme : ajouter  $\vec{x}$  à  $\vec{w}$  et 1 à  $w_0$   
 → Procéder par analogie pour les exemples négatifs  $\vec{x} \in \mathcal{N}$





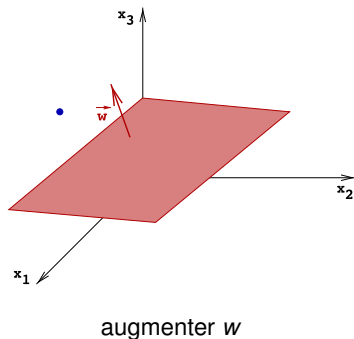
## Algorithme d'apprentissage du perceptron

- ▶ Erreur de classification de  $\vec{x} \in \mathcal{P}$   
 →  $w_0 + \langle \vec{w}, \vec{x} \rangle < 0$
- ▶ Comment peut-on modifier  $w_0$  et  $\vec{w}$  pour remédier à cette erreur  
 → augmenter  $w_0 + \langle \vec{w}, \vec{x} \rangle$ 
  - augmenter  $w_0$
  - Si  $x_i > 0$  augmenter  $w_i$
  - Si  $x_i < 0$  diminuer  $w_i$
- ▶ Algorithme : ajouter  $\vec{x}$  à  $\vec{w}$  et 1 à  $w_0$   
 → Procéder par analogie pour les exemples négatifs  $\vec{x} \in \mathcal{N}$



## Algorithme d'apprentissage du perceptron

- ▶ Erreur de classification de  $\vec{x} \in \mathcal{P}$   
 →  $w_0 + \langle \vec{w}, \vec{x} \rangle < 0$
- ▶ Comment peut-on modifier  $w_0$  et  $\vec{w}$  pour remédier à cette erreur  
 → augmenter  $w_0 + \langle \vec{w}, \vec{x} \rangle$ 
  - augmenter  $w_0$
  - Si  $x_i > 0$  augmenter  $w_i$
  - Si  $x_i < 0$  diminuer  $w_i$
- ▶ Algorithme : ajouter  $\vec{x}$  à  $\vec{w}$  et 1 à  $w_0$   
 → Procéder par analogie pour les exemples négatifs  $\vec{x} \in \mathcal{N}$



## Algorithme d'apprentissage du perceptron

**Entrées :** données d'apprentissage positives  $\mathcal{P}$  et négatives  $\mathcal{N}$

**Retourne :** un perceptron, si existe, qui classe parfaitement toutes les données d'apprentissage

1. initialiser arbitrairement le vecteur de pondération  $\vec{w}$  et le biais  $w_0$
2. **Tant que** il existe une donnée mal-classée  $\vec{x} \in \mathcal{P} \cup \mathcal{N}$  **faire**
3. **Si**  $\vec{x} \in \mathcal{P}$  **alors**
4.    $\vec{w} \leftarrow \vec{w} + \vec{x}$
5.    $w_0 = w_0 + 1$
6. **Sinon**
7.    $\vec{w} \leftarrow \vec{w} - \vec{x}$
8.    $w_0 = w_0 - 1$
9. **Fin si**
10. **Fin tant que**
11. **Retourner**  $w_0$  et  $\vec{w}$

## Algorithme d'apprentissage du perceptron : Exemple

$$\mathcal{N} = \{(1, 0)^\top, (1, 1)^\top\}, \mathcal{P} = \{(0, 1)^\top\}$$

→ **exercice**

## Algorithme d'apprentissage du perceptron : Convergence

► **Lemme :**

Si l'algorithme du perceptron converge, alors le perceptron  $(w_0, \vec{w})$  obtenu classe parfaitement tous les exemples d'apprentissage.

► **Théorème (Convergence) :**

Si  $\exists$  un perceptron qui classe correctement toutes les données d'apprentissage, alors l'algorithme du perceptron converge.

► **Lemme :**

Si au cours de l'exécution de l'algorithme du perceptron on rencontre deux fois le même vecteur de pondération, alors les données d'apprentissage utilisées ne sont pas linéairement séparables.

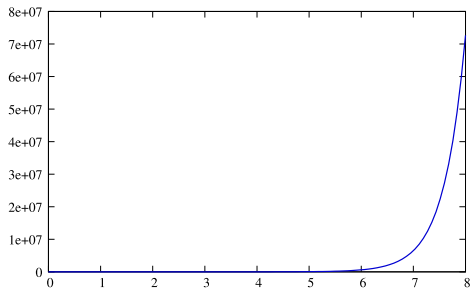
► **Lemme :**

Si l'algorithme du perceptron avec un biais  $w_0 = 0$  est exécuté sur un jeu de données non linéairement séparable, alors le vecteur de pondération se produira au moins deux fois .

## Algorithme d'apprentissage du perceptron : Convergence

### ► Lemme (Complexité) :

Si les données d'apprentissage sont linéairement séparables, alors le nombre d'itérations maximal de l'algorithme du perceptron est égale à  $(n + 1)^2 2^{(n+1) \log(n+1)}$

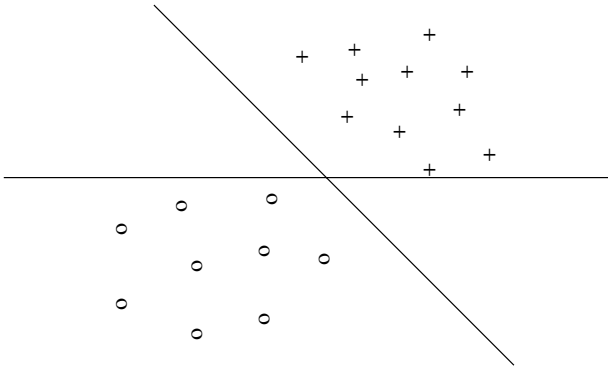


- temps d'exécution : complexité exponentielle  
→ autres implémentations - complexité  $O(n^{\frac{7}{2}})$  -

## Algorithme d'apprentissage du perceptron

- ▶ comment peut-on déterminer un "bon" perceptron si la tâche d'apprentissage ne peut pas être résolu parfaitement
- ▶ "bon" dans le sens d'un perceptron avec un nombre d'erreurs minimal
- ▶ l'algorithme du perceptron : le nombre d'erreurs ne décroît pas de façon monotone durant la phase d'apprentissage
- ▶ idée : mémoriser le meilleur vecteur de pondération rencontré au cours de l'exécution
- ▶ les perceptrons peuvent apprendre que des problèmes linéairement séparables
- ▶ contre-exemple :  $XOR(x_1, x_2)$   
 $\mathcal{P} = \{(0, 1)^T, (1, 0)^T\}, \mathcal{N} = \{(0, 0)^T, (1, 1)^T\}$
- ▶ un réseau de neurone associant plusieurs perceptrons est plus puissant

Entre ces deux solutions, laquelle est la meilleure ?





## Algorithme d'apprentissage du Perceptron (Rosenblatt, 1958)

Soit  $S = S_P \cup S_N \subset \mathbb{R}^{n+1} \times \{-1, 1\}$  un échantillon complété linéairement séparable.

Soit  $w$  le classifieur linéaire courant.

- ▶ Si  $(x, y) \in S_P$  est mal classé,  $\langle w, x \rangle < 0$  et il faudrait augmenter  $\langle w, x \rangle$ ,
- ▶ si  $(x, y) \in S_N$  est mal classé,  $\langle w, x \rangle \geq 0$  et il faudrait diminuer  $\langle w, x \rangle$ ,

Idée : prendre  $w_{new} = w + xy$ .

Dans le premier cas, on a  $\langle w_{new}, x \rangle = \langle w, x \rangle + \|x\|^2$ ; dans le second cas, on a  $\langle w_{new}, x \rangle = \langle w, x \rangle - \|x\|^2$ .

## Algorithme d'apprentissage du Perceptron

### Algorithme d'apprentissage du Perceptron

**Entrée :**  $S = \{(x_1, y_1), \dots, (x_l, y_l)\}$ , un échantillon complété linéairement séparable de  $\mathbb{R}^{n+1} \times \{-1, 1\}$

$w_0 = 0 \in \mathbb{R}^{n+1}$ ,  $k = 0$

**Répéter**

**Pour**  $i = 1$  à  $l$

**Si**  $y_i \langle w_k, x_i \rangle \leq 0$  **alors**

$w_{k+1} = w_k + y_i x_i$

$k = k + 1$

**FinPour**

**Jusqu'à ce qu'il n'y ait plus d'erreurs**

**Sortie :**  $w_k$

## Exercice

Utilisez l'algorithme du perceptron pour séparer l'échantillon  $\{((0, 0), -1), ((0, 1), 1), ((1, 0), 1), ((1, 1), 1)\}$ . Dessinez l'hyperplan obtenu.

$k$	$w_k$	$x_k$ mal classé	$y_k$
0	0 0 0	0 0 1	-1
1	0 0 -1	...	...
...	...	...	...

## Propriétés

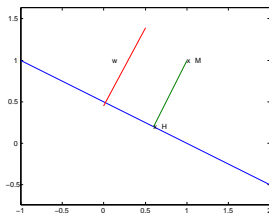
- ▶ L'algorithme du Perceptron est une procédure *on-line*, par *correction d'erreurs* (*error-driven*).
- ▶ L'algorithme est *correct* : lorsqu'il converge, l'hyperplan retourné sépare les données fournies en entrée
- ▶ L'algorithme est *complet* : si  $S$  est linéairement séparable, l'algorithme converge.
- ▶ Dans le pire des cas, le nombre d'itérations est égal à  $(n+1)^2 2^{(n+1) \log(n+1)}$ . Complexité exponentielle !
- ▶ Très mauvaise tolérance au bruit.

## La notion de marge

On peut multiplier l'équation  $\langle w, x \rangle + b = 0$  d'un hyperplan par un réel non nul sans modifier l'hyperplan qu'elle définit.

On peut donc supposer que  $w$  vérifie  $\|w\| = 1$ .

Dans ce cas, la distance d'un exemple  $(x, y)$  à un hyperplan séparateur est égal à  $y(\langle w, x \rangle + b)$ .



## Complexité de l'algorithme et marge (suite)

**Théorème (Novikoff) :** Soit  $S = \{(x_1, y_1), \dots, (x_l, y_l)\}$  un échantillon d'apprentissage non trivial (i.e.  $\exists i, j \ y_i y_j = -1$ ). Supposons que

- ▶  $\forall i, \|x_i\| \leq 1$  et
- ▶  $\exists w, b, \gamma > 0$  tels que  $\forall i, y_i(\langle w, x_i \rangle + b) \geq \gamma$ .

Alors, le nombre d'erreurs  $(y_i(\langle w_k, x_i \rangle + b_k) \leq 0)$  commises pendant l'exécution de l'algorithme est au plus égal à  $(2/\gamma)^2$ .

### Remarques :

- ▶  $\gamma$  est une borne inférieure de la *marge* du problème
- ▶ Quelles que soient les données, on peut toujours se ramener au moyen d'une homothétie-translation au cas où  $\text{Max}\|x_i\| = 1$ .

## Complexité de l'algorithme et marge

$$S = \{((0, 0), -1), ((0, 1), 1), ((1, 0), 1), ((1, 1), 1)\}.$$

Au moyen d'une translation de vecteur  $(-1/2, -1/2)$  suivie d'une homothétie de rapport  $\sqrt{2}$ , on obtient l'échantillon équivalent

$$S = \{((- \frac{\sqrt{2}}{2}, - \frac{\sqrt{2}}{2}), -1), ((- \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}), 1), ((\frac{\sqrt{2}}{2}, - \frac{\sqrt{2}}{2}), 1), ((\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}), 1)\}.$$

On a bien  $\text{Max} ||x_i|| = 1$ . On vérifie que la marge du problème est égale à  $1/2$ .

Le théorème prédit que le nombre de corrections de l'algorithme est inférieur ou égal à 8.

## Forme duale de l'algorithme du perceptron

**Remarque :** l'hypothèse finale est une combinaison linéaire des exemples d'apprentissage.

$$w = \sum_{i=1}^I \alpha_i y_i x_i.$$

Les nombres  $\alpha_i$  sont positifs et égaux au nombre de fois où une mauvaise classification de  $x_i$  a entraîné une mise à jour du perceptron. Ils peuvent être vus comme une représentation duale de la solution :

$$f(x) = \text{sgn}(\langle w, x \rangle + b) = \text{sgn} \left( \sum_{i=1}^I \alpha_i y_i \langle x_i, x \rangle + b \right).$$



## Forme duale de l'algorithme du perceptron

**entrée :**  $S = \{(x_1, y_1), \dots, (x_l, y_l)\}$ , un échantillon complété linéairement séparable

$\alpha = 0 \in \mathbb{R}^l$

**répéter**

**Pour**  $i = 1$  à  $l$

**Si**  $y_i(\sum_{j=1}^l \alpha_j \langle x_j, x_i \rangle) \leq 0$  **alors**

$\alpha_i = \alpha_i + 1$

$k = k + 1$

**FinSi**

**FinPour**

**Jusqu'à ce qu'il n'y ait plus d'erreurs**

**Sortie :**  $\alpha$

## Exercice

Utilisez l'algorithme du perceptron pour séparer l'échantillon  $\{((0, 0), -1), ((0, 1), 1), ((1, 0), 1), ((1, 1), 1)\}$ . Dessinez l'hyperplan obtenu.

$k$	$\alpha_k$	$x_k$ mal classé	$y_k$
0	0 0 0 0	0 0 1	-1
1	1 0 0 0	...	...
...	...	...	...

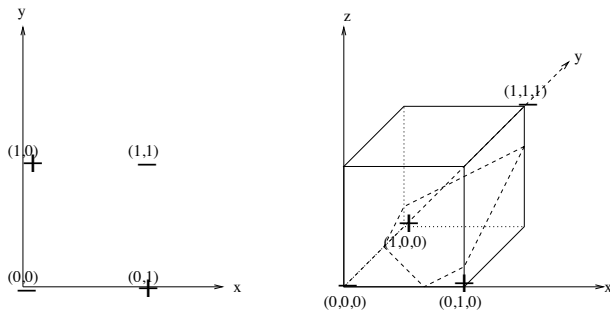
## Propriétés de l'algorithme dual

- ▶ dans la représentation duale, le nombre de paramètres de la solution ne dépend pas de la dimension de l'espace dans lequel les  $x_i$  sont plongés,
- ▶ les exemples d'apprentissage ne sont pris en compte par l'algorithme que par l'intermédiaire de leurs produits scalaires.
- ▶ On appelle **Matrice de Gram** la matrice  $G = (\langle x_i, x_j \rangle)_{1 \leq i, j \leq l}$  : elle suffit à trouver une solution.

## Extensions

1. Plongements non linéaires
2. Perceptrons linéaires avec couches cachées
3. Perceptrons non linéaires (avec couches cachées)
4. Séparateurs linéaires optimaux - Machines à Vecteurs Supports ou Séparateurs à vaste Marge (SVM)
5. Méthodes à noyaux

# Plongements non linéaires



$$(x,y) \longrightarrow (x,y,xy)$$

$$C : -x - y + 2xy + 1/3 = 0 \longleftarrow P : -x - y + 2z + 1/3 = 0$$

## Séparation linéaire après plongement non linéaire

Soit  $S = \{(0, 0), -1; (0, 1), 1; (1, 0), 1; (1, 1), -1\}$ .

On considère le plongement  $\phi$  de  $\mathbb{R}^2$  dans  $\mathbb{R}^3$  défini par

$$\phi(x^{(1)}, x^{(2)}) = (x^{(1)}, x^{(2)}, x^{(1)}x^{(2)}).$$

$$S_\phi = \{(\phi(x), y) | (x, y) \in S\} =$$

$$\{(0, 0, 0), -1; (1, 0, 0), 1; (0, 1, 0), 1; (1, 1, 1), -1\}.$$

$$\bar{S}_\phi = \{(0, 0, 0, 1), -1; (1, 0, 0, 1), 1; (0, 1, 0, 1), 1; (1, 1, 1, 1), -1\}.$$

### Algorithme du perceptron

$k$	$\mathbf{w}_k$	$\mathbf{x}_k$ mal classé	$y_k$
0	0 0 0 0	0 0 0 1	-1
1	0 0 0 -1	1 0 0 1	1
2	1 0 0 0	1 1 1 1	-1
3	0 -1 -1 -1	1 0 0 1	1
...	...	...	...
17	1 1 -3 -1		

ce qui conduit

- ▶ au plan séparateur d'équation  $x^{(1)} + x^{(2)} - 3x^{(3)} - 1 = 0$  dans  $\mathbb{R}^3$  et
- ▶ à la courbe séparatrice d'équation  $x^{(1)} + x^{(2)} - 3x^{(1)}x^{(2)} - 1 = 0$  dans  $\mathbb{R}^2$ .

## Séparation linéaire après plongement non linéaire (suite)

On considère un plongement  $\phi$  de  $\mathbb{R}^2$  dans  $\mathcal{H}$ , espace muni d'un produit scalaire, vérifiant

$$\langle \phi(x), \phi(y) \rangle_{\mathcal{H}} = 1 + x^{(1)}y^{(1)} + x^{(2)}y^{(2)} + x^{(1)}y^{(1)}x^{(2)}y^{(2)}.$$

Matrice de Gram de  $S_{\phi}$  :

$$(\langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}})_{1 \leq i, j \leq |S|} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 2 \\ 1 & 1 & 2 & 2 \\ 1 & 2 & 2 & 4 \end{pmatrix}$$

### Algorithme dual du perceptron

$k$	$\mathbf{n}_k$	$\phi(\mathbf{x}_k)$ mal classé	$y_k$
0	0 0 0 0	$\mathbf{x}_1$	-1
1	1 0 0 0	$\mathbf{x}_2$	1
2	1 1 0 0	$\mathbf{x}_4$	-1
3	1 1 0 1	$\mathbf{x}_2$	1
...	...	...	...
17	6 4 4 3		

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^4 n_i y_i \langle \phi(x_i), \phi(x) \rangle_{\mathcal{H}}\right)$$

$$= \operatorname{sgn}(-6 + 4(1 + x^{(1)}) + 4(1 + x^{(2)}) - 3(1 + x^{(1)} + x^{(2)} + x^{(1)}x^{(2)}))$$

## Kernel trick

On appelle **noyau** toute fonction  $k : X \times X \rightarrow \mathbb{R}$  qui peut être interprétée comme un produit scalaire dans un plongement  $\Phi$  :

$$k(x, y) = \langle \Phi(x), \Phi(y) \rangle$$

Tout algorithme d'apprentissage qui n'utilise que les produits scalaires des données (matrice de Gram) peut être *kernelisé*.

Le perceptron à noyau est un classifieur

$$f : x \mapsto \text{signe}\left(\sum_{i=1}^l \alpha_i y_i k(x, x_i)\right)$$

qui est

- ▶ linéaire dans l'espace de plongement (avec toutes les garanties associées) et
- ▶ non linéaire dans l'espace initial.



## Perceptron à noyau

**entrée :**  $S = \{(x_1, y_1), \dots, (x_l, y_l)\}$ , un échantillon complété

$\alpha = 0 \in \mathbb{R}^l$

**répéter**

**Pour**  $i = 1$  à  $l$

**Si**  $y_i(\sum_{j=1}^l \alpha_j y_j k(x_j, x_i)) \leq 0$  **alors**

$\alpha_i = \alpha_i + 1$

**FinSi**

**FinPour**

**Jusqu'à ce qu'il n'y ait plus d'erreurs**

**Sortie :**  $x \mapsto \text{signe}(\sum_i \alpha_i y_i k(x, x_i))$

## Exemples de noyaux :

### Noyau polynomial homogène

$$X = \mathbb{R}^n, k(x, y) = \left( \sum_{i=1}^n x_i y_i \right)^d$$

### Noyau polynomial

$$X = \mathbb{R}^n, k(x, y) = \left( 1 + \sum_{i=1}^n x_i y_i \right)^d.$$

### Noyau gaussien :

$$k(x, y) = \exp \left( -\frac{\|x - y\|^2}{2\sigma^2} \right)$$

La dimension de l'espace de plongement est finie pour les noyaux polynomiaux et infini (espace de Hilbert) pour le noyau gaussien ... mais le plongement est virtuel.

## Caractérisation des noyaux

**Théorème** : une fonction  $k : X \times X \rightarrow \mathbb{R}$  est un noyau ssi pour tout  $m$ -uplet  $x_1, \dots, x_m$  d'éléments de  $X$ , la matrice de Gram  $k(x_i, x_j)_{1 \leq i, j \leq m}$  est *définie positive*, c'est-à-dire que pour tous réels  $c_1, \dots, c_m$ ,

$$\sum_{i,j} c_i c_j k(x_i, x_j) \geq 0.$$

**A retenir** : on sait (théoriquement) caractériser les fonctions noyaux et déterminer un plongement correspondant.

## Séparation linéaire après plongement non linéaire (suite)

Sur l'exemple précédent, on considère le noyau Gaussien

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right).$$

Matrice de Gram de  $S_\phi$  :

$$(\langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}})_{1 \leq i, j \leq |S|} = \begin{pmatrix} 1.00 & 0.61 & 0.61 & 0.37 \\ 0.61 & 1.00 & 0.37 & 0.61 \\ 0.61 & 0.37 & 1.00 & 0.61 \\ 0.37 & 0.61 & 0.61 & 1.00 \end{pmatrix}$$

**Algorithme du perceptron à noyau** :  $n = [1, 1, 1, 1]$

- ▶ convergence en 4 étapes
- ▶ chaque exemple a été mal classé une et une seule fois

ce qui conduit au classifieur

$$f(x) = \text{sg}(-e^{-\|x\|^2/2} + e^{-((x^{(1)}-1)^2 + (x^{(2)})^2)/2} + e^{-((x^{(2)}-1)^2 + (x^{(1)})^2)/2} - e^{-((x^{(1)}-1)^2 + (x^{(2)}-1)^2)/2})$$

## Exercice

Soit

$$S = \{(-1, 0), 1; (0, 1), 1; (1, 0), 1; (0, 0), -1\}$$

et

$$k(x, y) = 1 + x^{(1)}y^{(1)} + x^{(2)}y^{(2)} + (x^{(1)}y^{(1)})^2.$$

1. Dessinez  $S$ ,
2. Appliquez le perceptron à noyau à ce jeu de données,
3. Dessinez la courbe séparatrice dans l'espace initial.

# Plan

Perceptron

SVM

## Retour sur la notion de marge

Soit  $S$  un échantillon linéairement séparable et soit  $H$  un hyperplan séparateur, d'équation  $\langle w, x \rangle + b = 0$ .

On peut choisir  $w$  et  $b$  tel que le point  $M$  le plus proche de  $H$  satisfasse :

$$f(x_M) = \langle w, x_M \rangle + b = \begin{cases} 1 & \text{si } M \text{ est positif} \\ -1 & \text{sinon.} \end{cases}.$$

Dans ce cas,

- ▶ la marge de  $H$  est égale à  $1/\|w\|$  et
- ▶ tous les points de  $S$  vérifient  $yf(x) \geq 1$ .

## Calcul de la marge (exemple)

Soit  $S = \{((0, 1), +), ((2, 0), -)\}$ .

La droite d'équation  $f(x, y) = -x + y - 1/2 = 0$  sépare  $S$ .

On a  $f(0, 1) = 1/2$  et  $f(2, 0) = -5/2$ .

On normalise l'équation en la multipliant par 2 :  $-2x + 2y - 1 = 0$ .

$$w = (-2, 2)^T, \|w\| = \sqrt{8} = 2\sqrt{2}$$

et la marge est égale à  $\frac{1}{2\sqrt{2}} = \frac{\sqrt{2}}{4}$ .



## SVM : Hyperplan optimal

Soit  $S = \{(x_1, y_1), \dots, (x_l, y_l)\} \subset \mathbb{R}^n \times \{-1, 1\}$  un échantillon linéairement séparable.

Il existe un **unique hyperplan de marge maximale** qui est solution du problème d'optimisation suivant :

$$\begin{cases} f(x) = \langle w, x \rangle + b \\ y_i f(x_i) \geq 1 \text{ pour tout } i = 1 \dots l \\ \text{Minimiser } \|w\|^2. \end{cases}$$

Optimisation quadratique sous contraintes linéaires (convexes)

## Hyperplan optimal (exemple)

Soit  $S = \{((4, 3), 1), ((0, 2), 1), ((0, 0), -1)\}$ .

Le problème d'optimisation à résoudre est :

Minimiser  $w_1^2 + w_2^2$  sous les contraintes

$$4w_1 + 3w_2 + b \geq 1, 2w_2 + b \geq 1, -b \geq 1.$$

Les deux dernières équations impliquent  $w_2 \geq 1$  et donc  $w_1^2 + w_2^2 \geq 1$ .

On en déduit la solution optimale :  $w_1 = 0, w_2 = 1, b = -1$ .

Équation de l'hyperplan optimal :  $y = 1$

## Hyperplan optimal (cas général)

Minimiser

$$||\mathbf{w}||^2$$

sous les contraintes

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \text{ pour tout } i = 1 \dots l$$

Nouvelles variables (multiplicateurs de Lagrange) :  $\alpha_i \geq 0$

Nouvelle fonction : le Lagrangien

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \mathbf{w}^2 - \sum_{i=1}^l \alpha_i [y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1]$$

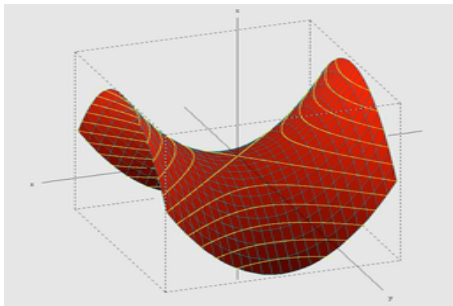
La solution (*unique* grâce à la convexité)  $(\mathbf{w}^*, b^*, \alpha^*)$  est un *point selle* du Lagrangien :  $L(\mathbf{w}, b, \alpha^*)$  est minimal en  $(\mathbf{w}^*, b^*)$  et  $L(\mathbf{w}^*, b^*, \alpha)$  est maximal en  $\alpha^*$ .

## Hyperplan optimal (cas général)

$$L(w, b, \alpha) = \frac{1}{2} w^2 - \sum_{i=1}^l \alpha_i [y_i (\langle w, x_i \rangle + b) - 1]$$

La solution (*unique* grâce à la convexité)  $(w^*, b^*, \alpha^*)$  est un *point selle* du Lagrangien :

$L(w, b, \alpha^*)$  minimal en  $(w^*, b^*)$ ;  $L(w^*, b^*, \alpha)$  maximal en  $\alpha^*$ .



## Hyperplan optimal : propriétés

On appelle **vecteur support** toute donnée  $x_i$  telle que  $\alpha_i^* \neq 0$ .

Soit  $SV$  l'ensemble des vecteurs supports de  $S$ .

### Propriétés

- ▶ Si  $x_i \in SV$ ,  $\langle w^*, x_i \rangle + b^* = y_i$
- ▶ Si  $x_i \notin SV$ , la contrainte correspondante n'est pas active,
- ▶ Les problèmes d'optimisation associés à  $S$  et à  $SV$  ont la même solution.
- ▶ Le rapport  $\#SV/\#S$  est une borne supérieure de l'estimateur *leave-one-out* de l'erreur en généralisation de la solution.
- ▶  $\sum_{i=1}^l \alpha_i^* y_i = 0$
- ▶  $w^* = \sum_{i=1}^l \alpha_i^* y_i x_i$

La solution s'exprime en fonction des vecteurs supports

## Exemple

Soit  $S = \{((4, 3), 1), ((0, 2), 1), ((0, 0), -1)\}$ . On a trouvé

$$w^* = (0, 1) \text{ et } b^* = -1.$$

Si l'on pose  $\sum_{i=1}^l \alpha_i^* y_i = 0$  et  $w^* = \sum_{i=1}^l \alpha_i^* y_i x_i$ , on trouve

$$\begin{cases} \alpha_1 + \alpha_2 - \alpha_3 &= 0 \\ 4\alpha_1 &= 0 \\ 3\alpha_1 + 2\alpha_2 &= 1 \end{cases}$$

On trouve

$$\alpha_1 = 0, \alpha_2 = 1/2 \text{ et } \alpha_3 = 1/2$$

Les vecteurs supports sont  $x_2$  et  $x_3$ .

## Hyperplan optimal : problème dual

Si l'on remplace  $w$  par  $\sum_{i=1}^l \alpha_i y_i x_i$  dans le Lagrangien

$$L(w, b, \alpha) = \frac{1}{2} w^2 - \sum_{i=1}^l \alpha_i [y_i (\langle w, x_i \rangle + b) - 1]$$

on trouve

$$W(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle$$

qu'on doit maximiser sous les contraintes  $\sum_{i=1}^l \alpha_i y_i = 0$  et  $\alpha_i \geq 0$ .

- ▶ Seuls les produits scalaires  $\langle x_i, x_j \rangle$  sont nécessaires pour trouver l'hyperplan optimal.
- ▶ Retrouver  $w^*$  à partir des  $\alpha_i^*$  :  $w^* = \sum_{i=1}^l \alpha_i^* y_i x_i$
- ▶ Calcul de  $b^*$  à partir d'un vecteur support :  $\langle w^*, x_i \rangle + b^* = y_i$ .

## Problème dual (exemple)

Soit  $S = \{((4, 3), 1), ((0, 2), 1), ((0, 0), -1)\}$ .

$$\begin{aligned} W(\alpha) &= \sum_{i=1}^I \alpha_i - \frac{1}{2} \sum_{i,j=1}^I y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle \\ &= \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2} (25\alpha_1^2 + 4\alpha_2^2 + 12\alpha_1\alpha_2) \end{aligned}$$

qu'on doit maximiser sous les contraintes

$$\alpha_1 + \alpha_2 - \alpha_3 = 0 \text{ et } \alpha_i \geq 0 \text{ pour } i = 1, 2, 3.$$



## Et lorsque les données ne sont pas séparables ?

- ▶ Trouver le classifieur linéaire qui minimise le nombre d'erreurs est un problème NP-dur.
- ▶ L'algorithme du Perceptron oscille, changeant d'hypothèses à chaque présentation d'un contre-exemple sans se stabiliser sur une solution intéressante.

**Idée :** se soucier davantage de la confiance avec laquelle la plupart des exemples sont correctement classés plutôt que du nombre d'exemples mal classés.

→ maximiser la marge d'un séparateur linéaire en tolérant un nombre limité d'exceptions : notion de **marge souple (soft margin)**.

## Soft margin optimisation

On introduit des **variables de relachement** (*slack variable*).

Minimiser

$$||\mathbf{w}'||^2 + C \sum_{i=1}^l \xi_i \text{ (ou } ||\mathbf{w}'||^2 + C \sum_{i=1}^l \xi_i^2)$$

sous les contraintes

$$\begin{cases} y_i(\langle \mathbf{w}', \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \text{ pour tout } i = 1, \dots, l \\ \xi_i \geq 0 \text{ pour tout } i = 1, \dots, l \end{cases}$$

**Idée** : on paie une *pénalité*  $C\xi_i$  pour tout exemple  $x_i$

- ▶ moins bien classé :  $0 < \xi_i < 1$  ou
- ▶ mal classé :  $\xi_i \geq 1$ .

## Comment réaliser un plongement des données ?

Pour tout algorithme d'apprentissage ne faisant intervenir que le produit scalaire des données

perceptron, séparateur optimal avec marges dures ou souples

il suffit de connaître la matrice de Gram  $G = (\langle x_i, x_j \rangle)_{1 \leq i, j \leq I}$  pour construire le classifieur linéaire correspondant :

$$f : x \mapsto \text{signe}\left(\sum_{i=1}^I \alpha_i \langle x, x_i \rangle\right).$$

Soit  $\Phi : X \rightarrow Y$  une fonction de plongement dans un espace  $Y$  avec produit scalaire. On obtiendra un classifieur linéaire (dans l'espace de plongement) défini par :

$$f : x \mapsto \text{signe}\left(\sum_{i=1}^I \alpha_i \langle \Phi(x), \Phi(x_i) \rangle\right).$$

## Kernel trick

On appelle **noyau** toute fonction  $k : X \times X \rightarrow \mathbb{R}$  qui peut être interprétée comme un produit scalaire dans un plongement  $\Phi$  :

$$k(x, y) = \langle \Phi(x), \Phi(y) \rangle$$

On peut appliquer les algorithmes du perceptron et de séparation optimale avec marges souples ou dures en remplaçant

$$\langle x_i, x_j \rangle \text{ par } k(x_i, x_j).$$

On obtient alors un classifieur

$$f : x \mapsto \text{signe}\left(\sum_{i=1}^l \alpha_i k(x, x_i)\right)$$

linéaire dans l'espace de plongement (avec toutes les garanties associées)  
et non linéaire dans l'espace initial.

## Exemples de noyaux :

### Noyau polynomial homogène

$$X = \mathbb{R}^n, k(x, y) = \left( \sum_{i=1}^n x_i y_i \right)^d$$

### Noyau polynomial

$$X = \mathbb{R}^n, k(x, y) = \left( 1 + \sum_{i=1}^n x_i y_i \right)^d.$$

### Noyau gaussien :

$$k(x, y) = \exp \left( -\frac{\|x - y\|^2}{2\sigma^2} \right)$$

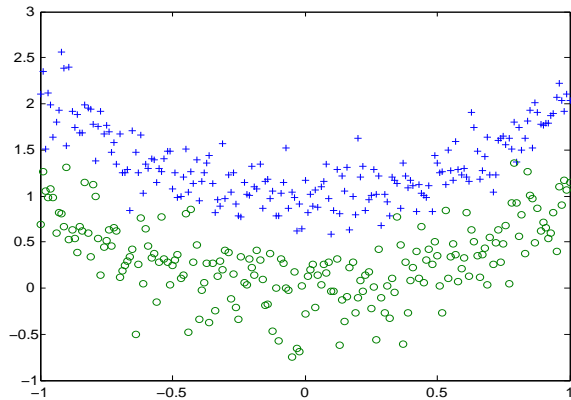
La dimension de l'espace de plongement est finie pour les noyaux polynomiaux et infini (espace de Hilbert) pour le noyau gaussien ... mais le plongement est virtuel.

## Caractérisation des noyaux

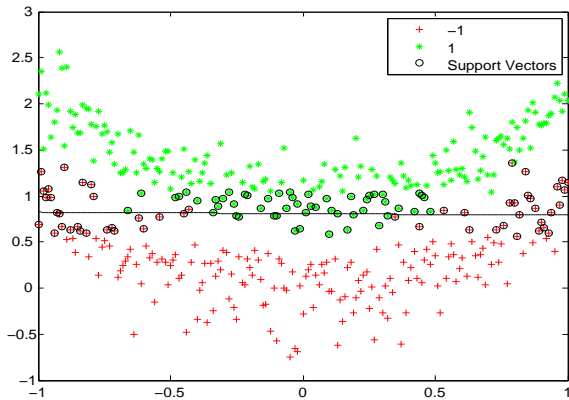
**Théorème** : une fonction  $k : X \times X \rightarrow \mathbb{R}$  est un noyau ssi pour tout  $m$ -uplet  $x_1, \dots, x_m$  d'éléments de  $X$ , la matrice de Gram  $k(x_i, x_j)_{1 \leq i, j \leq m}$  est *définie positive*, c'est-à-dire que pour tous réels  $c_1, \dots, c_m$ ,

$$\sum_{i,j} c_i c_j k(x_i, x_j) \geq 0.$$

**A retenir** : on sait (théoriquement) caractériser les fonctions noyaux et déterminer un plongement correspondant.

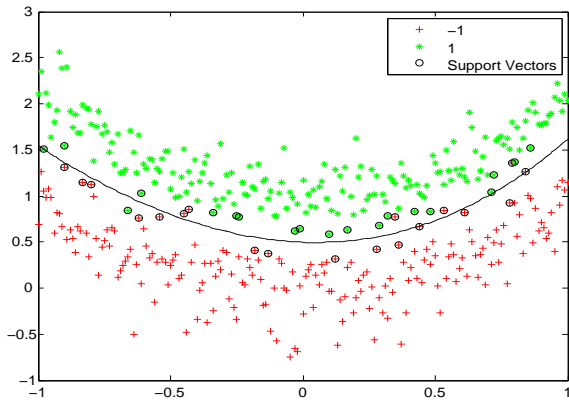


402 points générés à partir de 2 paraboles parallèles avec bruit Gaussien.

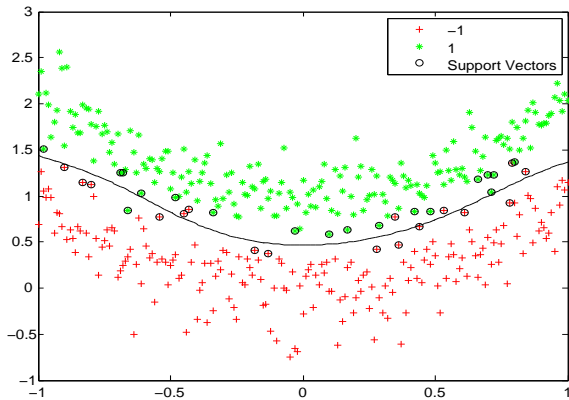


Séparateur linéaire optimal avec marges souples : 104 vecteurs supports.

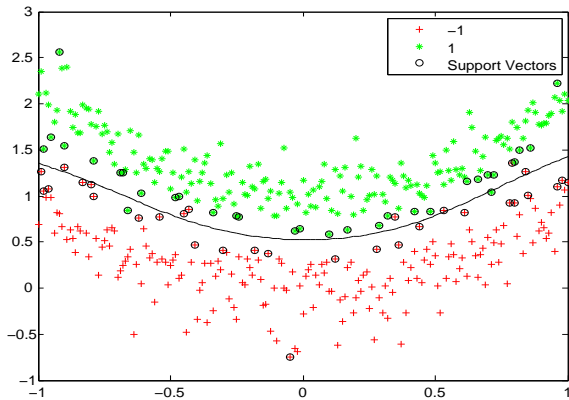




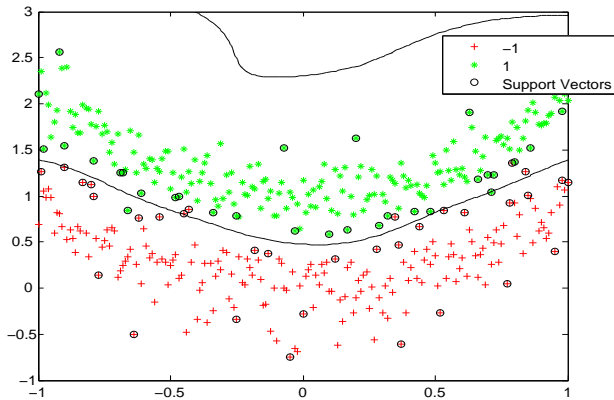
Noyau quadratique : 38 vecteurs supports.



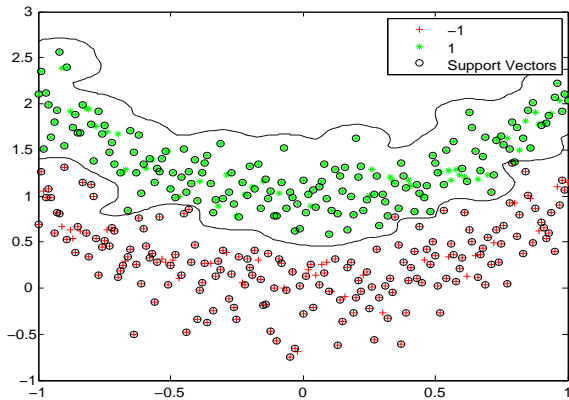
Noyau polynomial de degré 3 : 35 vecteurs supports.



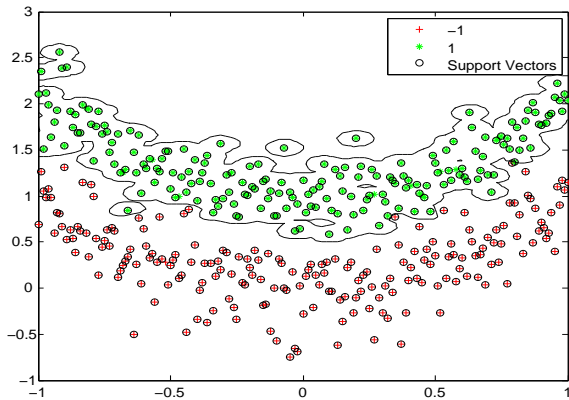
Noyau Gaussien,  $\sigma = 1$  : 62 vecteurs supports.



Noyau Gaussien,  $\sigma = 0.5$  : 64 vecteurs supports.



Noyau Gaussien,  $\sigma = 0.1$  : 321 vecteurs supports.



Noyau Gaussien,  $\sigma = 0.05$  : 390 vecteurs supports.

## The *US Postal Service (USPS)* database.

- ▶ 9298 chiffres manuscrits (7291 pour apprendre, 2007 pour tester) provenant d'enveloppes postées ou reçues à Buffalo ;
- ▶ Chaque chiffre est une image  $16 \times 16$  représenté par un vecteur de  $[-1, 1]^{256}$  ;
- ▶ Les jeu de tests est difficile : 2, 5% d'erreur en moyenne pour un humain ;
- ▶ Données disponibles à

*[http ://www.kernel-machines.org.data.html](http://www.kernel-machines.org.data.html).*

0540205091373973731137  
 004037138379622400007  
 7202722047014890025080  
 4597038973029411194107  
 606800968009681271130  
 0557102997095870173159  
 0296712967139673496816  
 4173576723087104821291  
 7334964814148080484334  
 848207470064824059542  
 4865514327442544720344  
 6419920874143221440774  
 4447114423044258455014  
 5440153538544095340203  
 1410304441967203306134  
 3333914130551825036900  
 0340641338411264141661



## Résultats (1) (Schölkopf, Smola)

Classification par SVMs sur les données USPS.

Noyau polynomial :  $k(x, y) = (\langle x, y \rangle / 256)^d$ .

Construction de 10 classifieurs (*méthode un contre tous*).

$d$	1	2	3	4	5	6	7
erreur %	8,9	4,7	4,0	4,2	4,5	4,5	4,7
Nb. moyen de VSs	282	237	274	321	374	422	491

## Résultats (2) (Schölkopf, Smola)

Classification par SVMs sur les données USPS.

Noyau gaussien :  $k(x, y) = \exp(-\|x - y\|^2 / (256c))$ .

Construction de 10 classifieurs (*méthode un contre tous*).

c	4,0	2,0	1,2	0,8	0,5	0,2	0,1
erreur %	5,3	5,0	4,9	4,3	4,4	4,4	4,5
Nb. moyen de VSs	266	240	233	235	251	366	722

## Comparaison des résultats (Schölkopf, Smola)

USPS+ : USPS + un ensemble de chiffres imprimés.

Classifieur	Ens. d'App.	Erreur sur ens. test
Decision tree, C4.5	USPS	16,2
Best two layer neural network	USPS	5,9
Five-layer network	USPS	5,1
Linear SVM	USPS	8,9
Hard margin SVM	USPS	4,6
SVM	USPS	4,0
Virtual SVM	USPS	3,2
Virtual SV, local kernel	USPS	3,0
Nearest neighbor	USPS+	5,9
Boosted neural net	USPS+	2,6
Human performance		2,5