Classication, Apprentissage, Décision

Chapitre cinquième : Perceptron

Stéphane Ayache, Cécile Capponi, François Denis, **Rémi Eyraud**,
Hachem Kadri, Liva Ralaivola

Master 2 IS-IN

Plan du cours

Plan

Classification linéaire binaire

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

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

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.

Un exemple

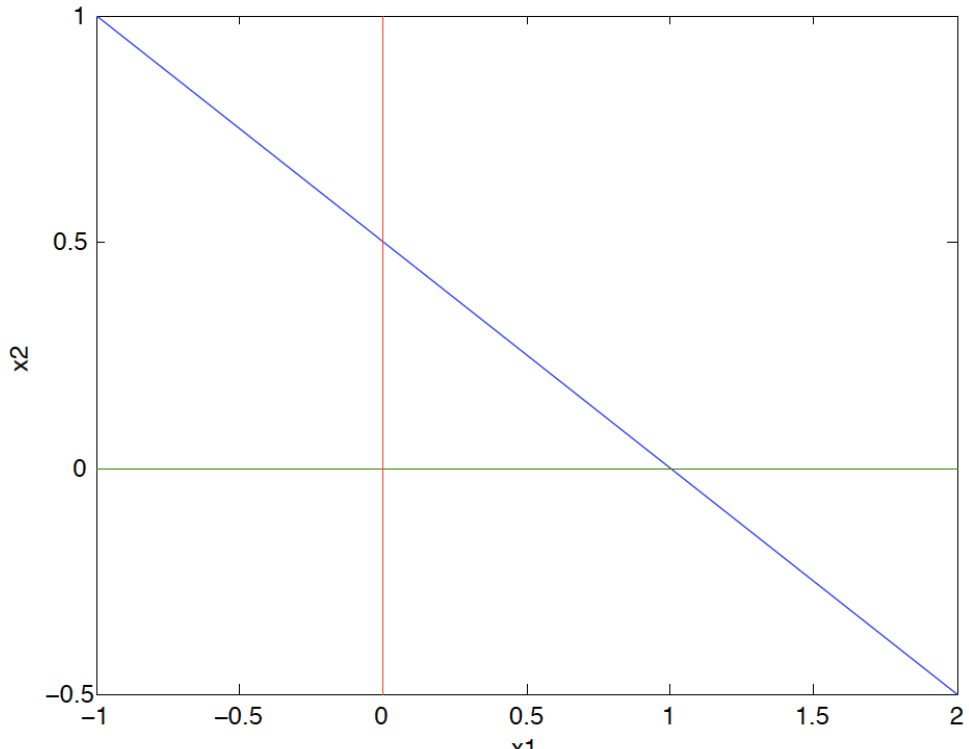
$X = \mathbb{R}^2$

Classifieur linéaire f 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 $x_1 + 2x_2 - 1 = 0$ (c-à-d $x_2 = -\frac{1}{2}x_1 + \frac{1}{2}$)



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 vidéos, par exemple).
- Cela suggère de plonger les données initiales dans un espace de grande dimension (voir chapitre suivant).

  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 $S = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subset (X \times Y)^n$ 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 d entrées est décrit par un vecteur de pondération $\overrightarrow{w} = (w_1, \ldots, w_d)^\top \in \mathbb{R}^d$, un seuil $\theta \in \mathbb{R}$ et permet de calculer la fonction suivante :

$$(x_1, \ldots, x_d) \longmapsto y = \begin{cases} +1 & \text{if } x_1 w_1 + x_2 w_2 + \ldots + x_d w_d \geq \theta \\ -1 & \text{if } x_1 w_1 + x_2 w_2 + \ldots + x_d w_d < \theta \end{cases}$$

Perceptrons

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

$$(x_1, \ldots, x_d) \longmapsto y = f(b + \sum_{i=1}^{d}(w_i x_i)) = f(b + \langle \overrightarrow{w}, \overrightarrow{x} \rangle)$$

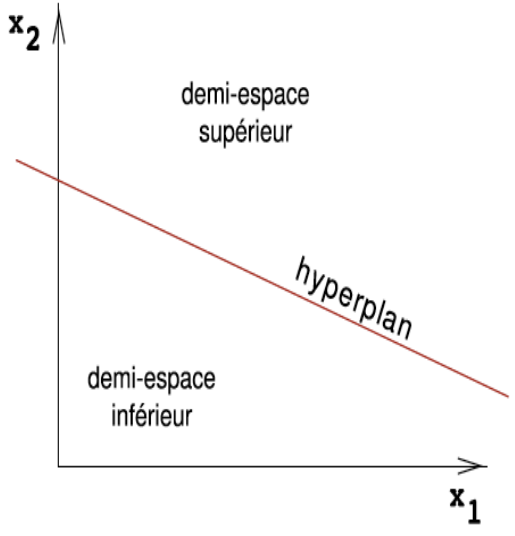
avec $f(z) = \begin{cases} +1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$

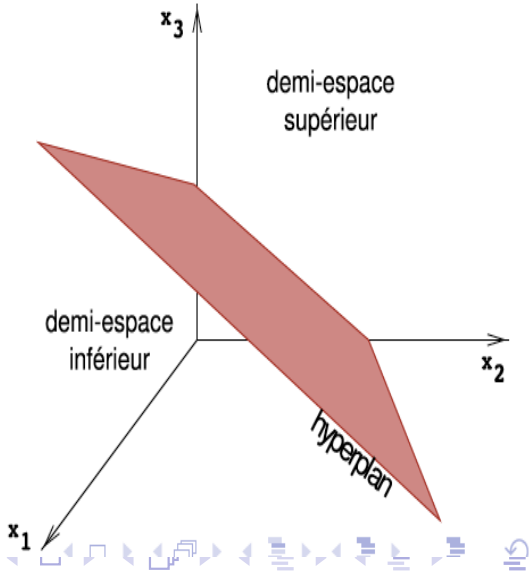
Perceptrons : Remarques

- Les isométries et les homothéties preservent la séparabilité
- En rajoutant une dimension, on peut ne plus avoir besoin du biais : ajouter une coordonnée x_{d+1} égale à 1 pour toutes les données et poser $b = w_{d+1}$
- il existe une infinité d'hyperplans séparant des données séparables...

Perceptrons : interprétation géométrique

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





algorithme d'apprentissage du perceptron

- les perceptrons peuvent être automatiquement adaptés à des taches d'apprentissage $\Rightarrow$ Apprentissage supervisé : Classification
- algorithme d'apprentissage du perceptron :
  données :
  • un ensemble de données $\mathcal{P} \subseteq \mathbb{R}^d$: exemples positifs
  • un autre ensemble $\mathcal{N} \subseteq \mathbb{R}^d$: exemples négatifs
  tache :
  • générer un perceptron qui retourne 1 pour tous les exemples de $\mathcal{P}$ et -1 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
  $\longrightarrow$ exemple : $\mathcal{P} \bigcap \mathcal{N} \neq \emptyset$
  $\longrightarrow$ données non linéairement séparables
  $\longrightarrow$ solution potentielle : transférer les données dans un autre espace dans lequel les données sont linéairement séparables (plus de détail dans le chapitre suivant)

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, $b + \langle \vec{w}, \vec{x} \rangle \neq 0$
  Preuve :
  Soit $(b, \vec{w})$ un perceptron qui classe parfaitement toutes les données d'apprentissage. D'où

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

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

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

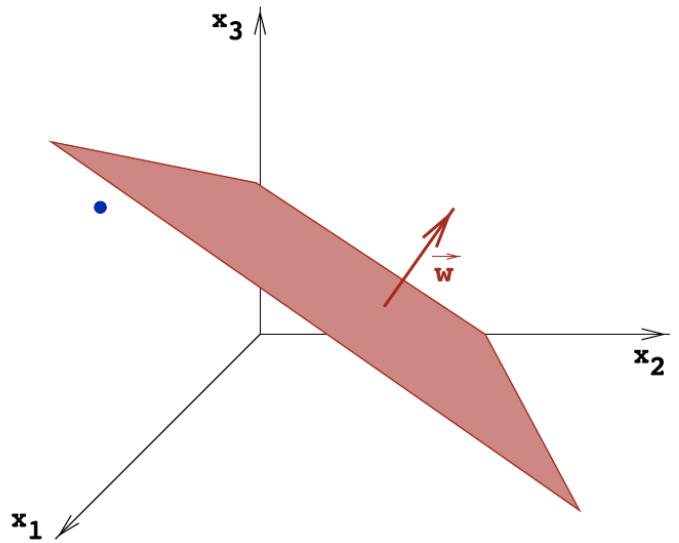
  Ainsi le perceptron $(b + \frac{\epsilon}{2}, \vec{w})$ prouve le lemme.

Algorithme d'apprentissage du perceptron

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

Algorithme d'apprentissage du perceptron

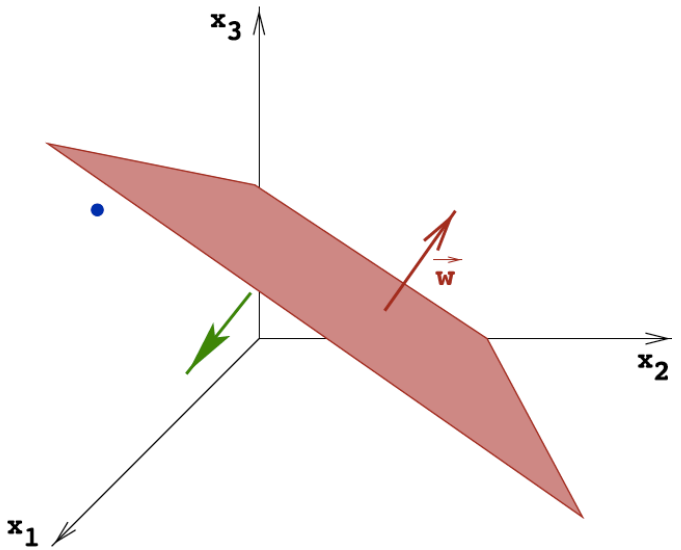
- Erreur de classification de $\overrightarrow{x} \in \mathcal{P}$
  $\longrightarrow b + \langle \overrightarrow{w}, \overrightarrow{x} \rangle < 0$
- Comment peut-on modifier b et $\overrightarrow{w}$ pour remédier à cette erreur
  $\longrightarrow$ augmenter $b + \langle \overrightarrow{w}, \overrightarrow{x}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



augmenter w_0

Algorithme d'apprentissage du perceptron

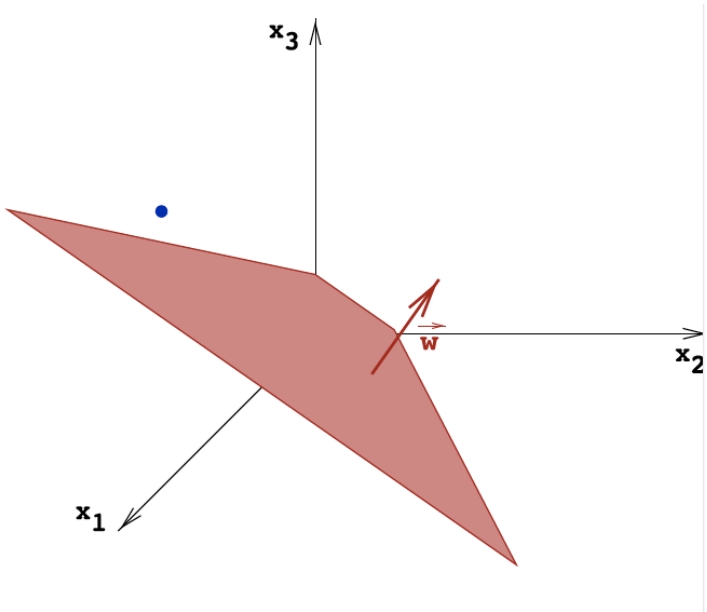
- Erreur de classification de $\overrightarrow{x} \in \mathcal{P}$
  $\longrightarrow b + \langle \overrightarrow{w}, \overrightarrow{x} \rangle < 0$
- Comment peut-on modifier b et $\overrightarrow{w}$ pour remédier à cette erreur
  $\longrightarrow$ augmenter $b + \langle \overrightarrow{w}, \overrightarrow{x}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



augmenter w_0

Algorithme d'apprentissage du perceptron

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



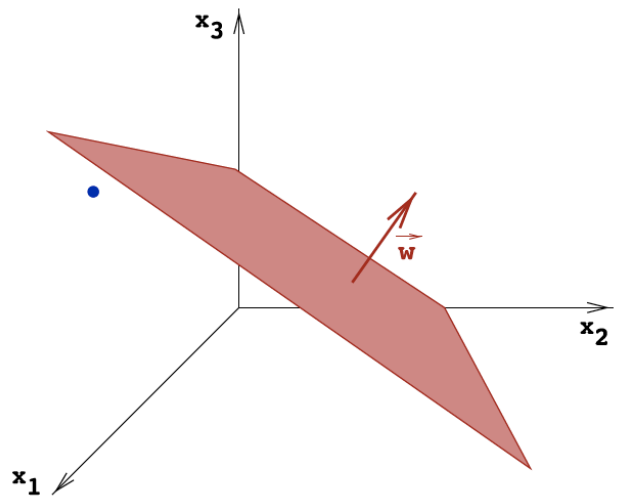
augmenter w_0

Algorithme d'apprentissage du perceptron

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

Algorithme d'apprentissage du perceptron

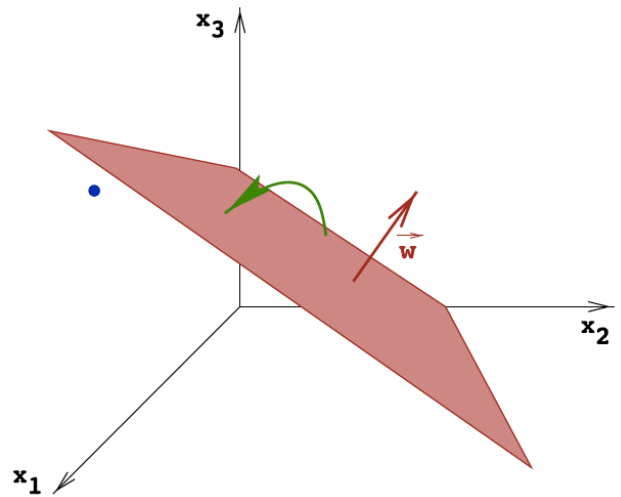
- Erreur de classification de $\overrightarrow{x} \in \mathcal{P}$
  $\longrightarrow b + \langle \overrightarrow{w}, \overrightarrow{x} \rangle < 0$
- Comment peut-on modifier b et $\overrightarrow{w}$ pour remédier à cette erreur
  $\longrightarrow$ augmenter $b + \langle \overrightarrow{w}, \overrightarrow{x}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



augmenter w

Algorithme d'apprentissage du perceptron

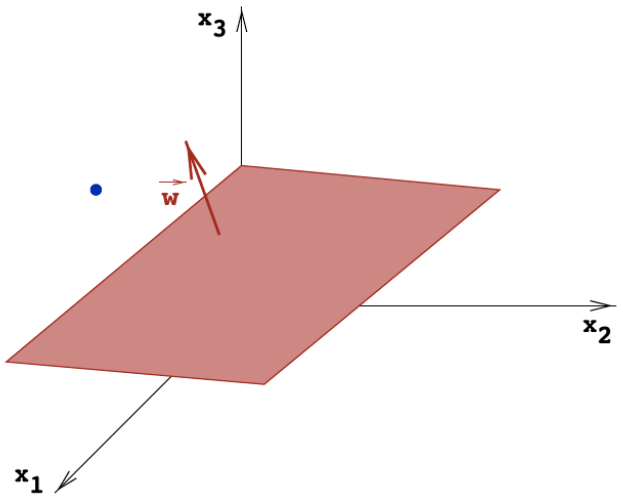
- Erreur de classification de $\overrightarrow{x} \in \mathcal{P}$
  $\longrightarrow b + \langle \overrightarrow{w}, \overrightarrow{x} \rangle < 0$
- Comment peut-on modifier b et $\overrightarrow{w}$ pour remédier à cette erreur
  $\longrightarrow$ augmenter $b + \langle \overrightarrow{w}, \overrightarrow{x}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



augmenter w

Algorithme d'apprentissage du perceptron

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



augmenter w

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 b
2. **Tant que** il existe une donnée mal-classée $\vec{x} \in \mathcal{P} \bigcup \mathcal{N}$ **faire**
3. **Si** $\vec{x} \in \mathcal{P}$ **alors**
4. $\quad \vec{w} \leftarrow \vec{w} + \vec{x}$
5. $\quad b = b + 1$
6. **Sinon**
7. $\quad \vec{w} \leftarrow \vec{w} - \vec{x}$
8. $\quad b = b - 1$
9. **Fin si**
10. **Fin tant que**
11. **Retourner** b et $\vec{w}$

Algorithme d'apprentissage du perceptron : Exemple

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

$\longrightarrow$ exercice

Algorithme d'apprentissage du perceptron : Convergence

- Lemme :

  Si l'algorithme du perceptron converge, alors le perceptron $(b, \overrightarrow{w})$ 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 $b = 0$ est exécuté sur un jeu de données non linéairement séparable, alors le même 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
  $\rightarrow$ 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 décroit 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 : $XOR(x_1, x_2)$
  $\mathcal{P} = \{(0,1)^\top, (1,0)^\top\}, \mathcal{N} = \{(0,0)^\top, (1,1)^\top\}$
- un réseau de neurone associant plusieurs perceptrons est plus puissant

Algorithme d'apprentissage du Perceptron (Rosenblatt, 1958)

Soit $S = S_P \cup S_N \subset \mathbb{R}^{d+1} \times \{-1, 1\}$ un échantillon 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 : $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, un échantillon complété linéairement séparable de $\mathbb{R}^{d+1} \times \{-1, 1\}$
$w_0 = 0 \in \mathbb{R}^{d+1}, k = 0$
Répéter
    Pour $i = 1$ **à** n
        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

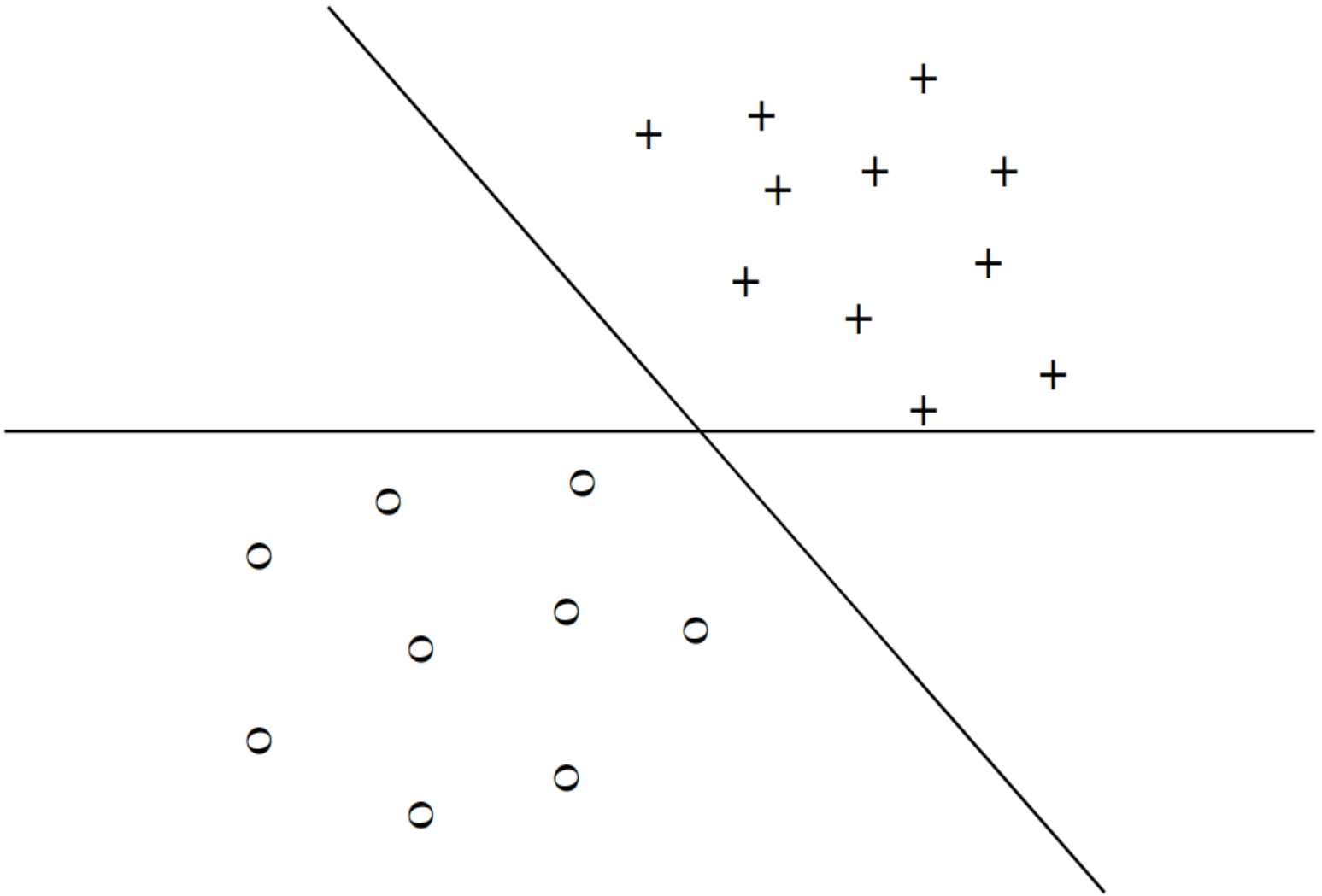
Utilisez l'algorithme du perceptron pour séparer l'échantillon $\{((0,0),-1),((0,1),1),((1,0),1),((1,1),1)\}$. Dessinez l'hyperplan obtenu.

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.

Entre ces deux solutions, laquelle est la meilleure ?

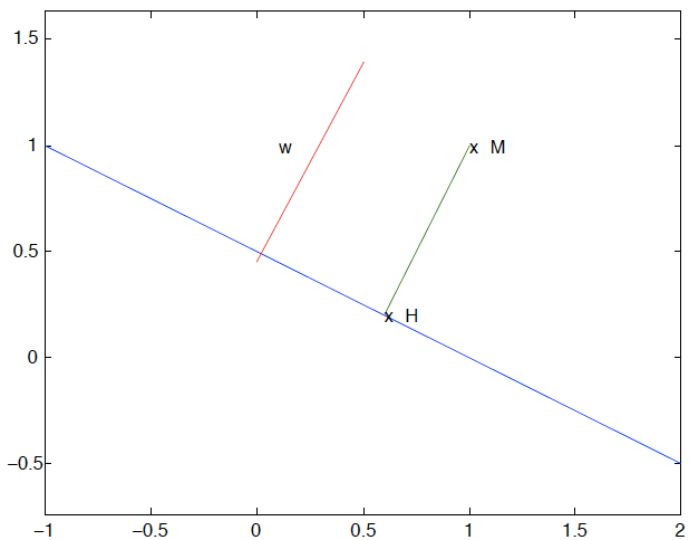


La notion de marge

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 $y(\langle w, x \rangle + b)$.



$$\begin{aligned} d(M, H) &= y_M \langle w, x_M - x_H \rangle \\ &= y_M(\langle w, x_M \rangle + b) \end{aligned}$$

puisque $\langle w, x_H \rangle + b = 0$.

Complexité de l'algorithme et marge

Théorème (Novikoff) : Soit $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ 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 :

- γ 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 $Max||x_i|| = 1$.

Complexité de l'algorithme et marge (suite)

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

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 $Max||x_i|| = 1$. 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.

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

Les nombres α_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 :

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

Forme duale de l'algorithme du perceptron

entrée : $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, un échantillon complété linéairement séparable
$\alpha = 0 \in \mathbb{R}^n$
répéter
    Pour $i = 1$ **à** n
        Si $y_i(\sum_{j=1}^{n} \alpha_j \langle x_j, x_i \rangle) \leq 0$ **alors**
            $\alpha_i = \alpha_i + 1$
            $k = k + 1$
        FinSi
    FinPour
Jusqu'à ce qu'il n'y ait plus d'erreurs
Sortie : α

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	α_k	x_k mal classé	y_k
0	0 0 0 0	0 0 1	-1
1	1 0 0 0	...	...
...	...	...	...

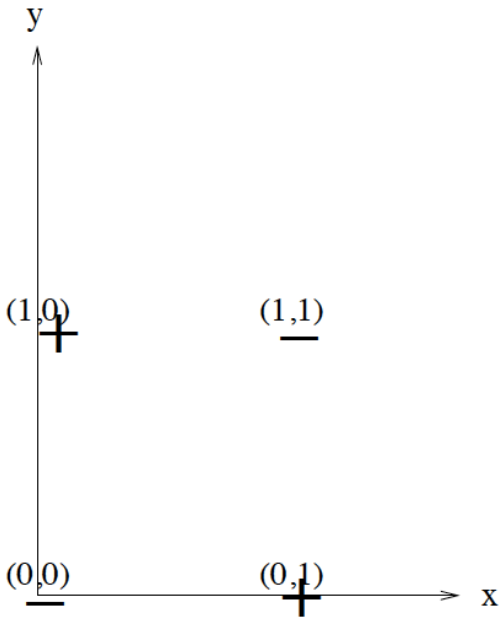
Propriétés de l'algorithme dual

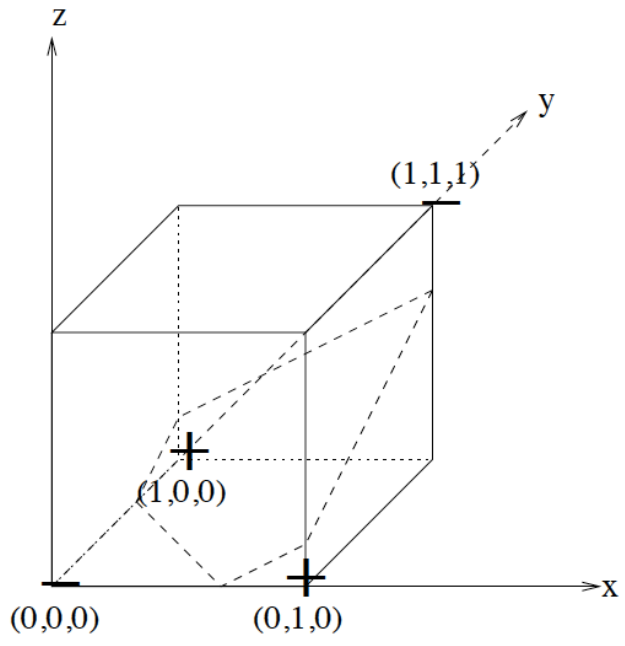
- dans la représentation duale, le nombre de paramètres de la solution ne dépend pas de la dimension de l'espace dans lequel les x_i sont plongés,
- les exemples d'apprentissage ne sont pris en compte par l'algorithme que par l'intermédiaire de leurs produits scalaires.
- On appelle **Matrice de Gram** 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





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

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

Perceptron linéaire à seuil avec une couche cachée

Entrée : $(x_1, \ldots, x_d, x_{d+1} = 1)$

Couche cachée : h perceptrons $y_1, \ldots, y_h$ (+ une entrée constante $y_{h+1} = 1$) où pour $1 \leq i \leq h$,

$$y_i = \begin{cases} 1 \text{ si } \sum_{j=1}^{d+1} \alpha_i^j x_j \geq 0 \\ 0 \text{ sinon.} \end{cases}$$

Sortie : un perceptron s défini par

$$s = f(x_1, \ldots, x_{d+1}) = \begin{cases} 1 \text{ si } \sum_{i=1}^{h+1} \beta_i y_i \geq 0 \\ 0 \text{ sinon.} \end{cases}$$

Perceptron linéaire à seuil avec une couche cachée

Théorème : toute fonction booléenne est calculable par un tel perceptron

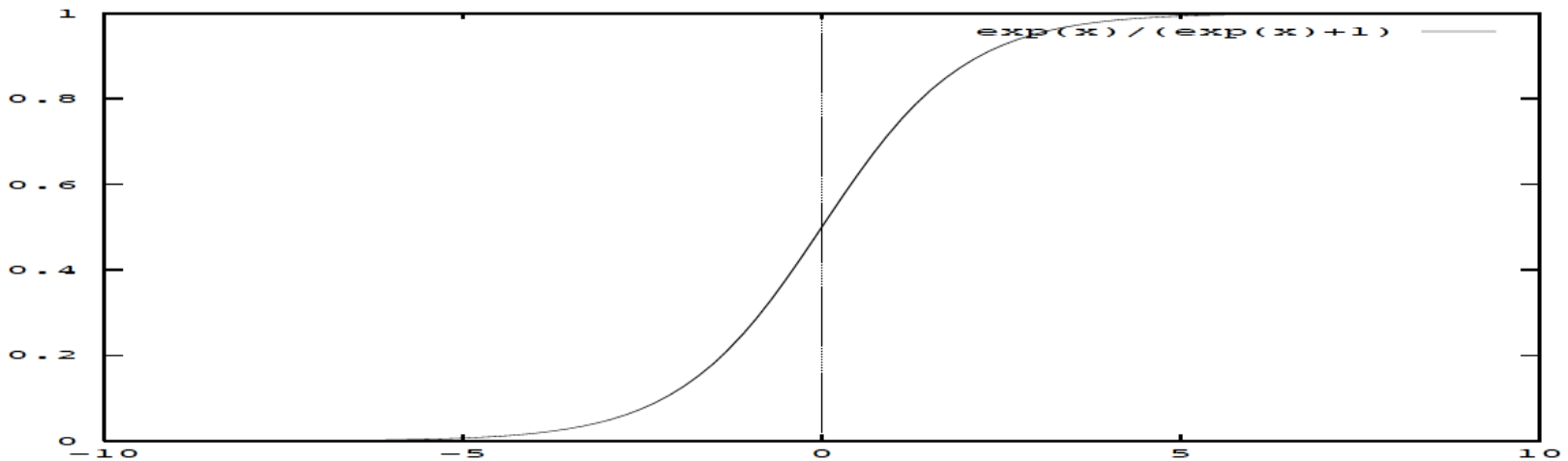
- bonne expressivité ;
- pas d'algorithme d'apprentissage !

La fonction sigmoïde

$$\sigma_k(x) = 1/(1 + e^{-kx})$$

On peut voir les fonctions sigmoïdes comme des fonctions indéfiniment dérivables qui lissent la fonction linéaire à seuil et dont les dérivées sont très simples à calculer :

$$\sigma' = \sigma(1 - \sigma)$$



Perceptron avec fonction d'activation sigmoïde et une couche cachée

Entrée : $x_1, \ldots, x_d, x_{d+1} = 1$

Couche cachée : $y_1, \ldots, y_h, y_{h+1} = 1$ où pour $1 \leq i \leq h$,

$$y_i = \sigma\left(\sum_{j=1}^{d+1} \alpha_i^j x_j\right)$$

Sortie :

$$s = f(x_1, \ldots, x_{d+1}) = \sigma\left(\sum_{i=1}^{h+1} \beta_i y_i\right).$$

Classification binaire : on attribue la classe 1 si $s \geq 1/2$ et 0 sinon ; s peut s'interpréter comme un indice de confiance.

Classification n-aire : n neurones normalisés en sortie.

Perceptron avec fonction d'activation sigmoïde et une couche cachée

Théorème : Ces perceptrons permettent d'approcher d'aussi près qu'on le souhaite une classe très riche de fonctions.

- très bonne expressivité ;
- il existe un algorithme d'apprentissage : l'algorithme de rétropropagation du gradient.

Séparateurs linéaires optimaux

Soit $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, $x_i \in R^d$ et $y_i \in \{-1, 1\}$ un échantillon linéairement séparable.
Il existe un unique hyperplan de marge maximale qui est solution du problème d'optimisation quadratique convexe suivant :

$$\text{Minimiser } ||w||^2$$

sous les contraintes

$$y_i f(x_i) \geq 1 \text{ pour tout } i = 1 \ldots l$$

où $f(x) = \langle w, x \rangle + b$.

Séparateurs linéaires optimaux

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

$$\text{Minimiser } w_1^2 + w_2^2$$

sous les contraintes

$$-b \geq 1, w_2 + b \geq 1, w_1 + b \geq 1, w_1 + w_2 + b \geq 1$$

La seule solution est $w_1 = w_2 = 2$ et $b = -1$.

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 modifier linairement 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 $\frac{1}{2\sqrt{2}} = \frac{\sqrt{2}}{4}$.

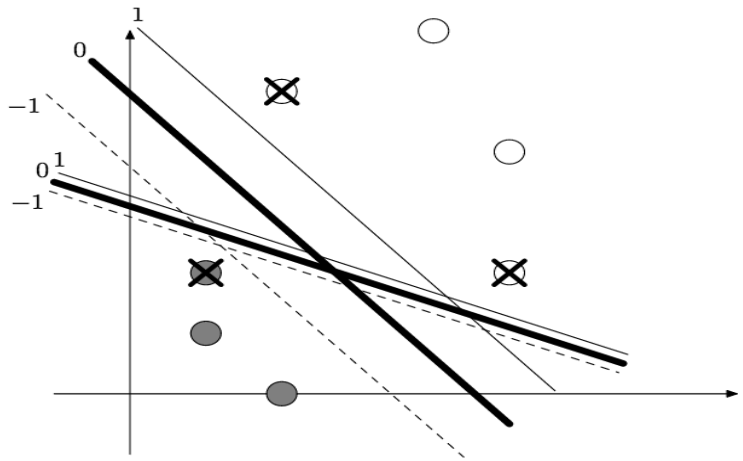
Hyperplan optimal

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

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

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

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

$$||w||^2$$

sous les contraintes

$$y_i(\langle w, x_i \rangle + b) \geq 1 \text{ pour tout } (x_i, y_i) \in S$$

Nouvelles variables (multiplicateurs de Lagrange) : $\alpha_i \geq 0$
Nouvelle fonction : le Lagrangien

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

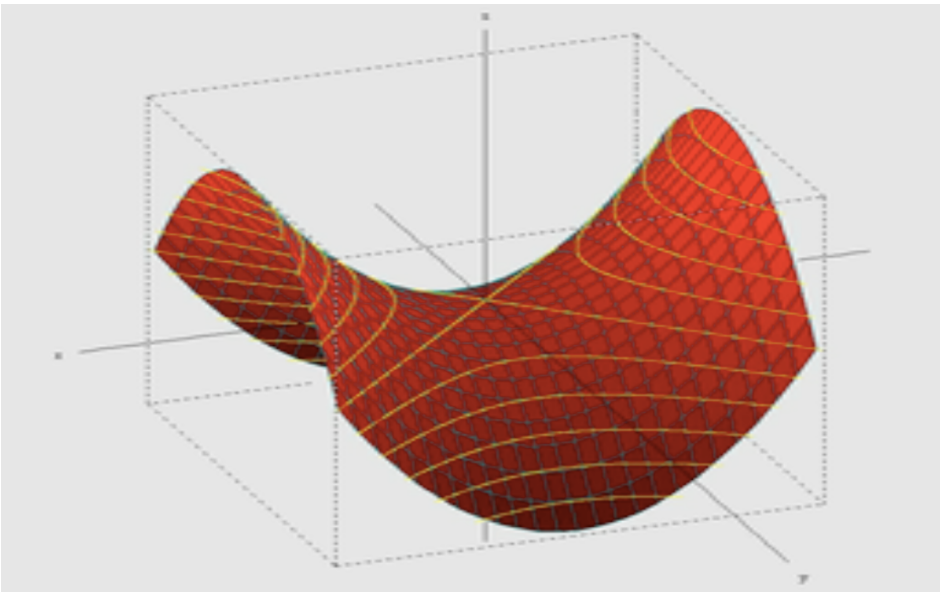
La solution (*unique* grâce à la convexité) (w^*, b^*, α^*) est un *point selle* du Lagrangien : $L(w, b, \alpha^*)$ est minimal en (w^*, b^*) et $L(w^*, b^*, \alpha)$ est maximal en α^*.

Hyperplan optimal (cas général)

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

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

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



Hyperplan optimal (cas général)

On a :

$$\begin{cases} \nabla_x L(w, b, \alpha) = w - \sum_{i=1}^{n} \alpha_i y_i x_i \\ \dfrac{\partial L(w, b, \alpha)}{\partial b} = \sum_{i=1}^{n} \alpha_i y_i \end{cases}$$

Ce qui donne les condition d'optimalité :

$$\begin{cases} \nabla_x L(w, b, \alpha) = 0 \Rightarrow w = \sum_{i=1}^{n} \alpha_i y_i x_i \\ \dfrac{\partial L(w, b, \alpha)}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases}$$

Hyperplans optimaux : problème dual

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

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

On a

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

Ce qui donne

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

qu'on doit maximiser sous les contraintes $\sum_{i=1}^{n} \alpha_i y_i = 0$ 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 α_i^* : $w^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i$
- Calcul de b^* à partir d'un vecteur support : $\langle w^*, x_i \rangle + b^* = y_i$.

Problème dual (exemple)

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

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

$$= \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2}(25\alpha_1^2 + 4\alpha_2^2 + 12\alpha_1\alpha_2)$$

qu'on doit maximiser sous les contraintes

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

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.
- $\sum_{i=1}^{n} \alpha_i^* y_i = 0$
- $w^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i$

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 $\sum_{i=1}^{n} \alpha_i^* y_i = 0$ et $w^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i$, 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.

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.

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

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 $G = (\langle x_i, x_j \rangle)_{1 \leq i,j \leq n}$ pour construire le classifieur linéaire correspondant :

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

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

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

Kernel trick

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

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

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

$$f : x \mapsto \mathit{signe}(\sum_{i=1}^{n} \alpha_i k(x, x_i))$$

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

Perceptron à noyau

entrée : $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, un échantillon complété linéairement séparable
$\alpha = 0 \in \mathbb{R}^n$
répéter
  Pour $i = 1$ **à** n
    Si $y_i(\sum_{j=1}^{n} \alpha_j y_j k(x_j, x_i)) \leq 0$ **alors**
      $\alpha_i = \alpha_i + 1$
      FinSi
    FinPour
  Jusqu'à ce qu'il n'y ait plus d'erreurs
  Sortie : $x \mapsto signe(\sum_i \alpha_i y_i k(x, x_i))$

Exemples de noyaux :

Noyau polynomial homogène

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

Noyau polynomial

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

Noyau gaussien :

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

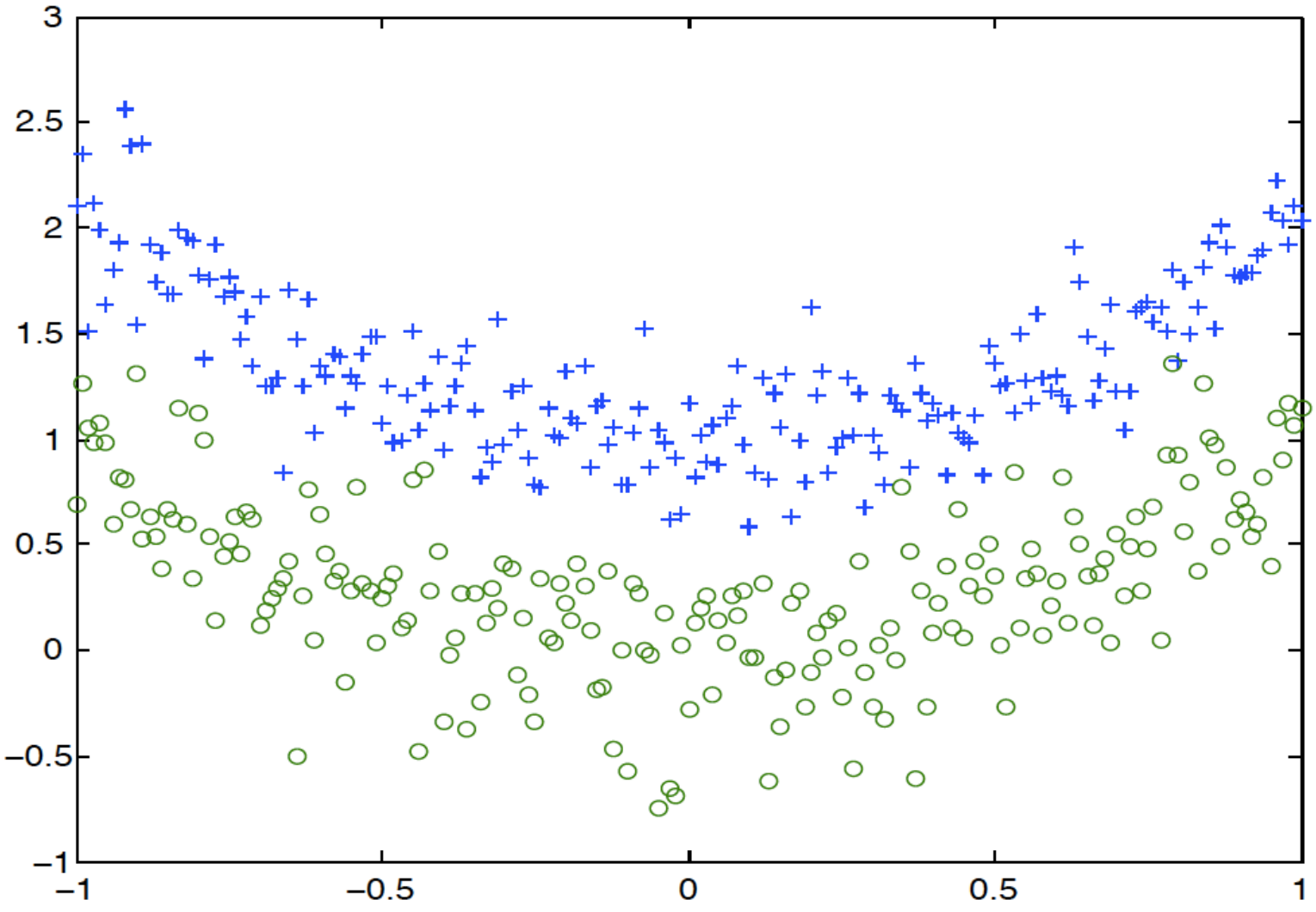
La dimension de l'espace de plongement est finie pour les noyaux polynomiaux et infini (espace de Hilbert) pour le noyau gaussien . . . mais le plongement est virtuel.

Caractérisation des noyaux

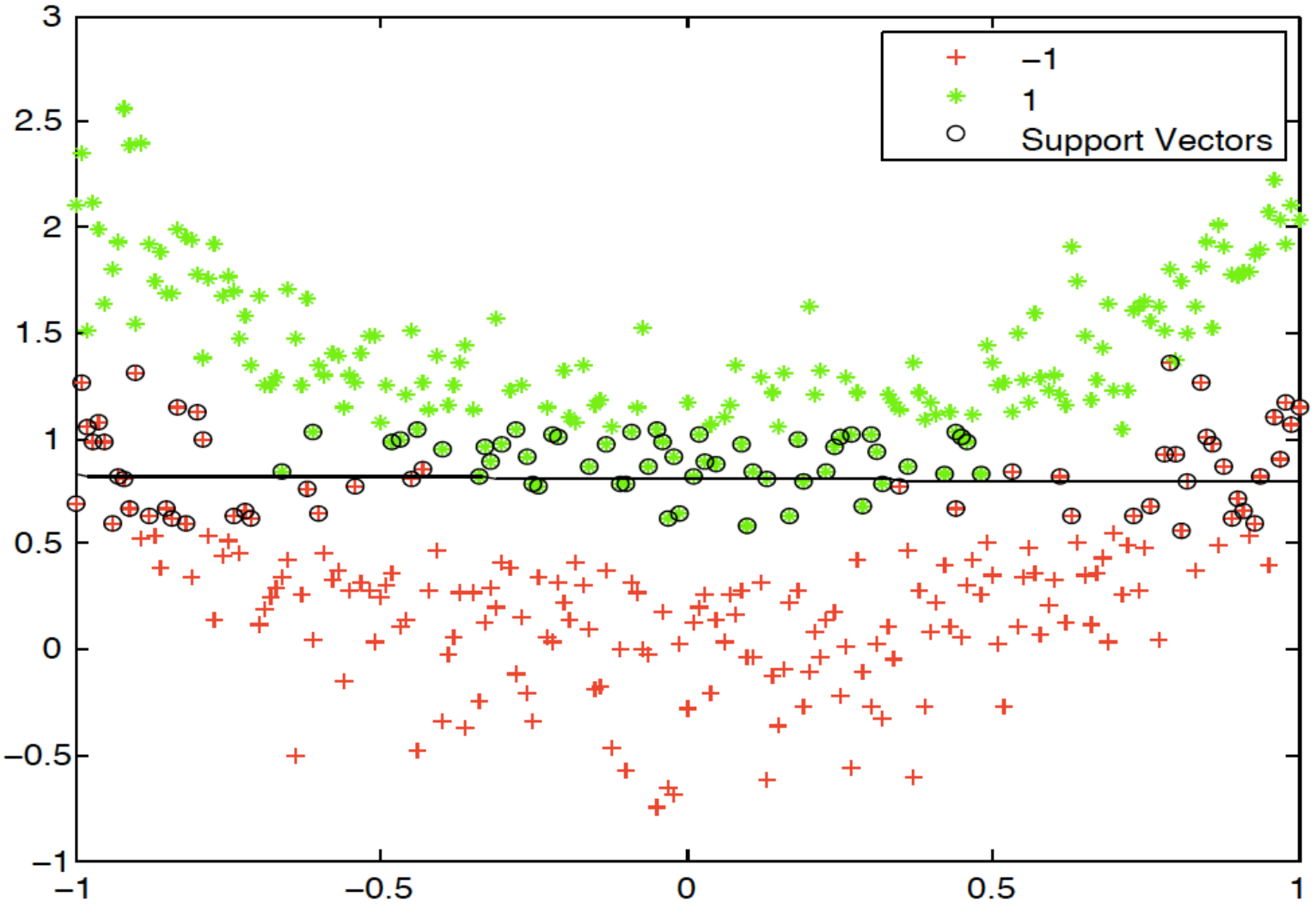
Théorème : une fonction $k : X \times X \to \mathbb{R}$ est un noyau ssi pour tout m-uplet $x_1, \ldots, x_m$ d'éléments de X, la matrice de Gram $k(x_i, x_j)_{1 \leq i,j \leq m}$ est *définie positive*, c'est-à-dire que pour tous réels $c_1, \ldots, c_m$,

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

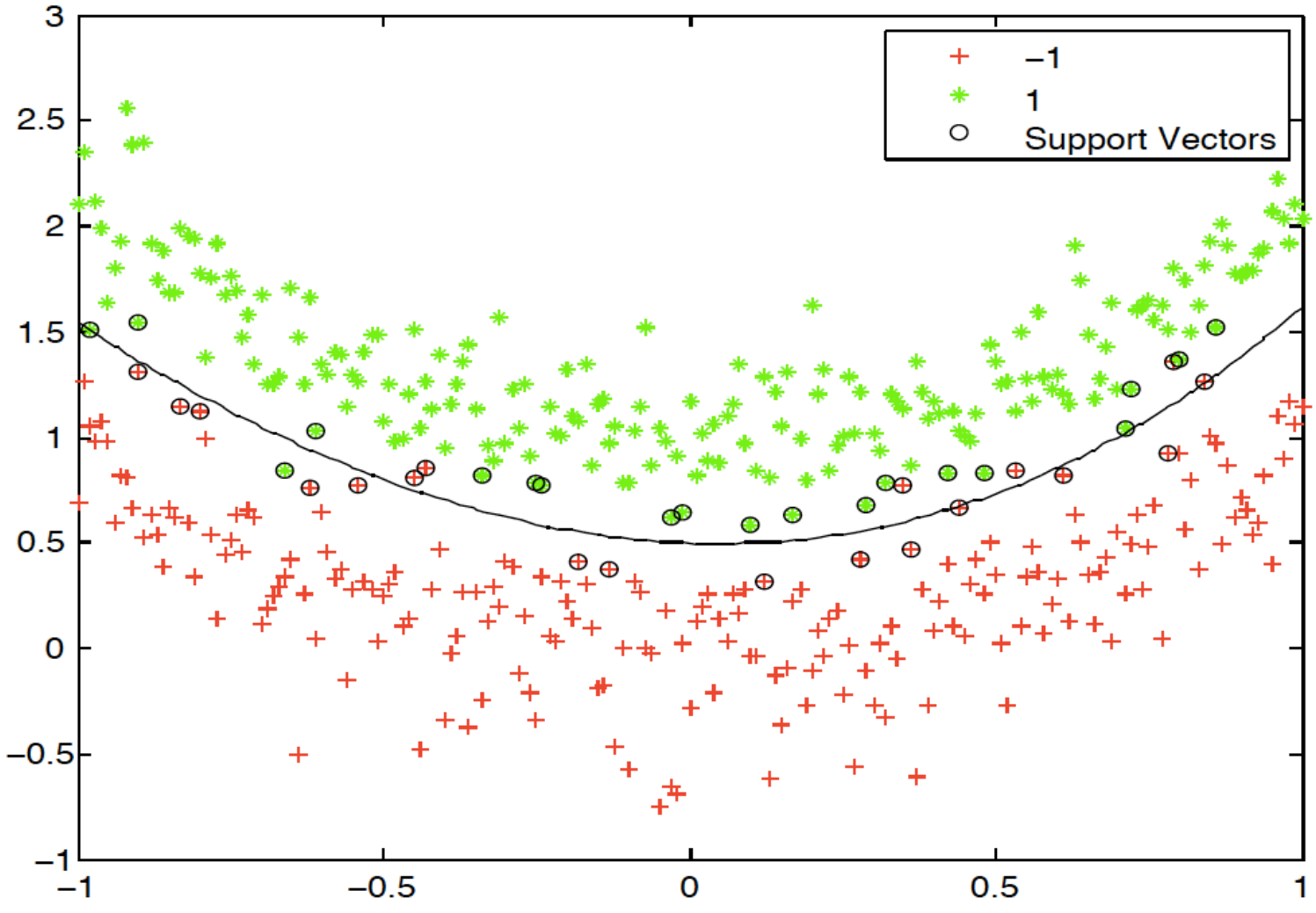
A retenir : on sait (théoriquement) caractériser les fonctions noyaux et déterminer un plongement correspondant.



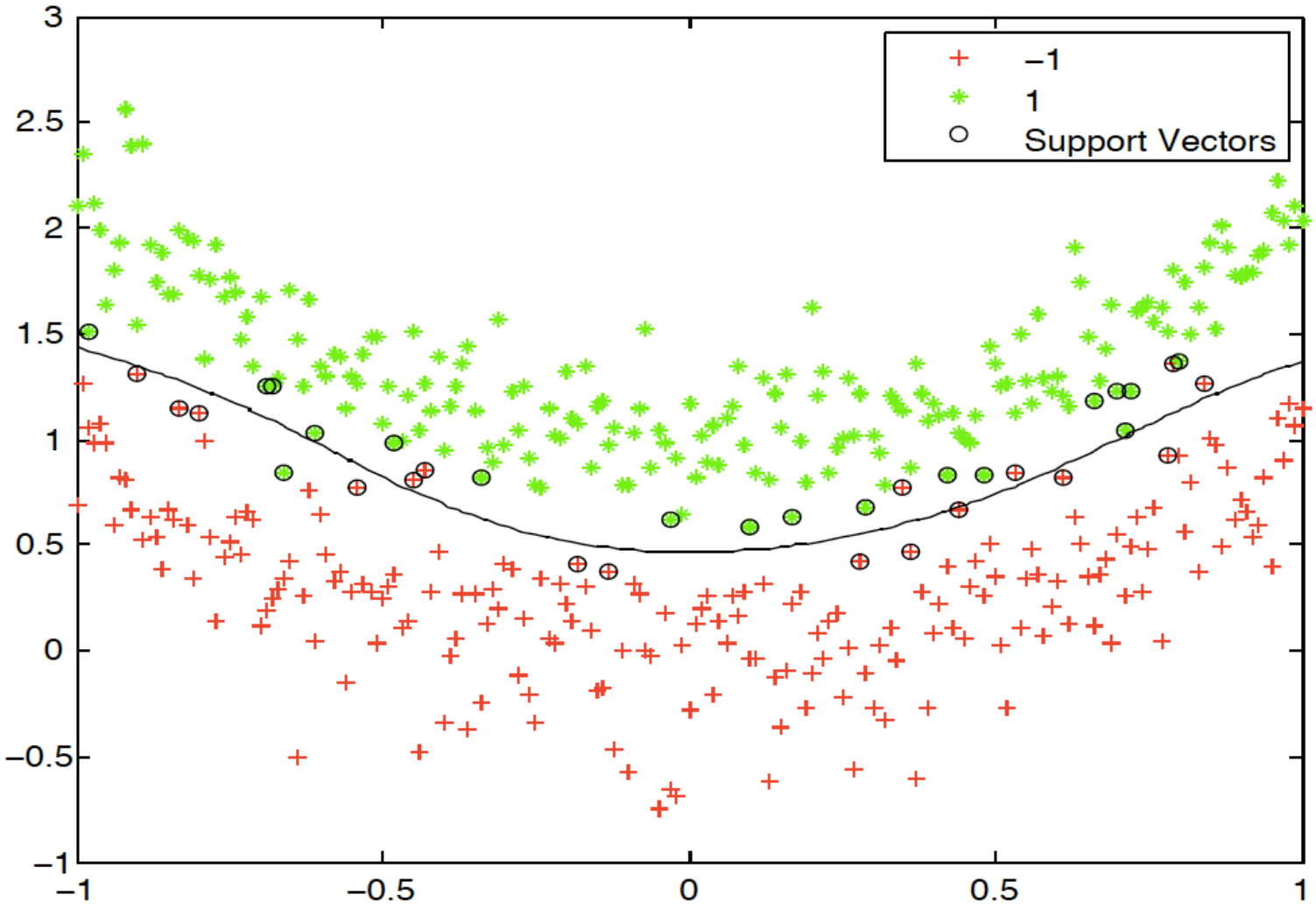
402 points générés à partir de 2 paraboles parallèles avec bruit gaussien.



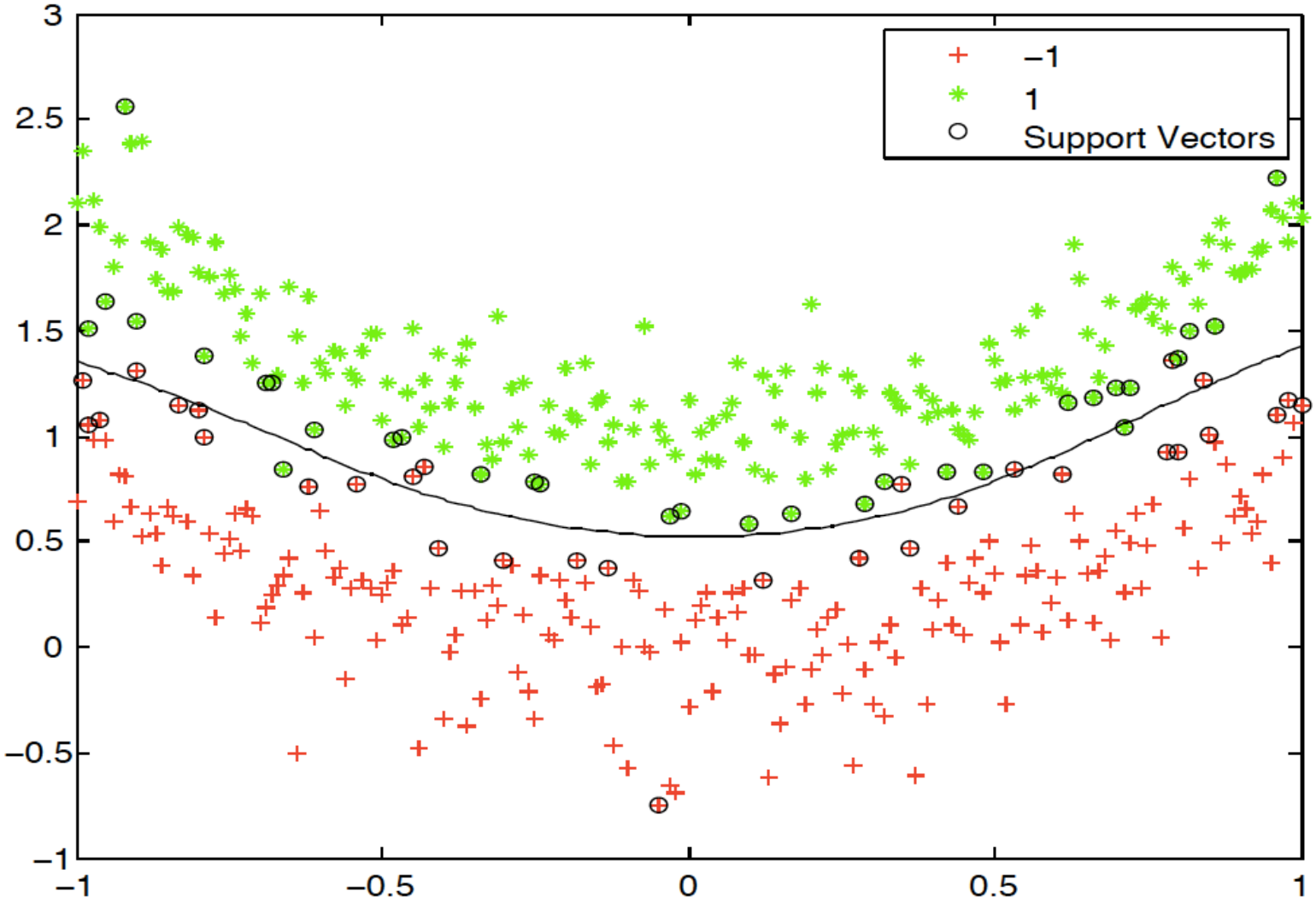
Séparateur linéaire optimal avec marges souples : 104 vecteurs supports.



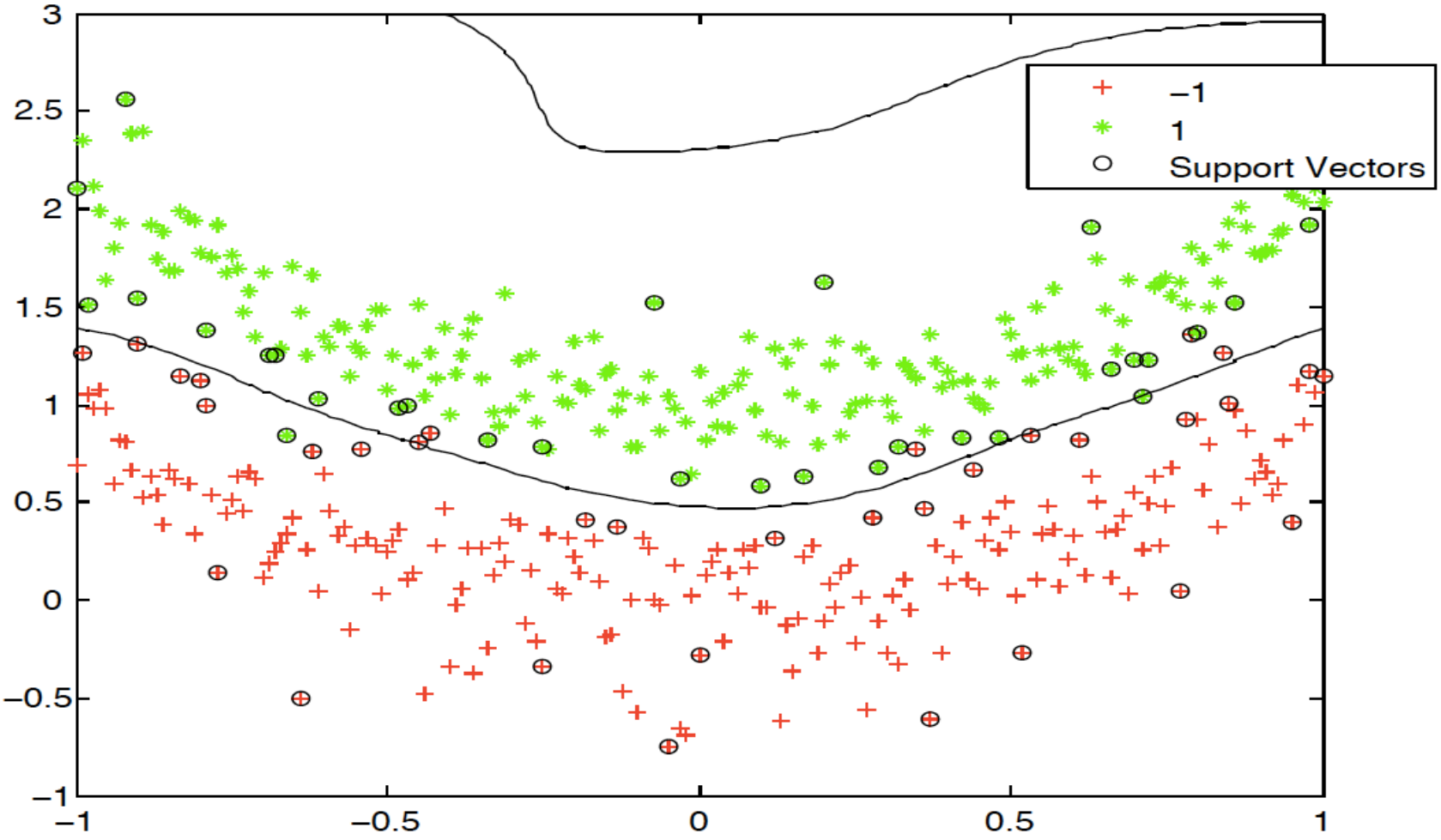
Noyau quadratique : 38 vecteurs supports.



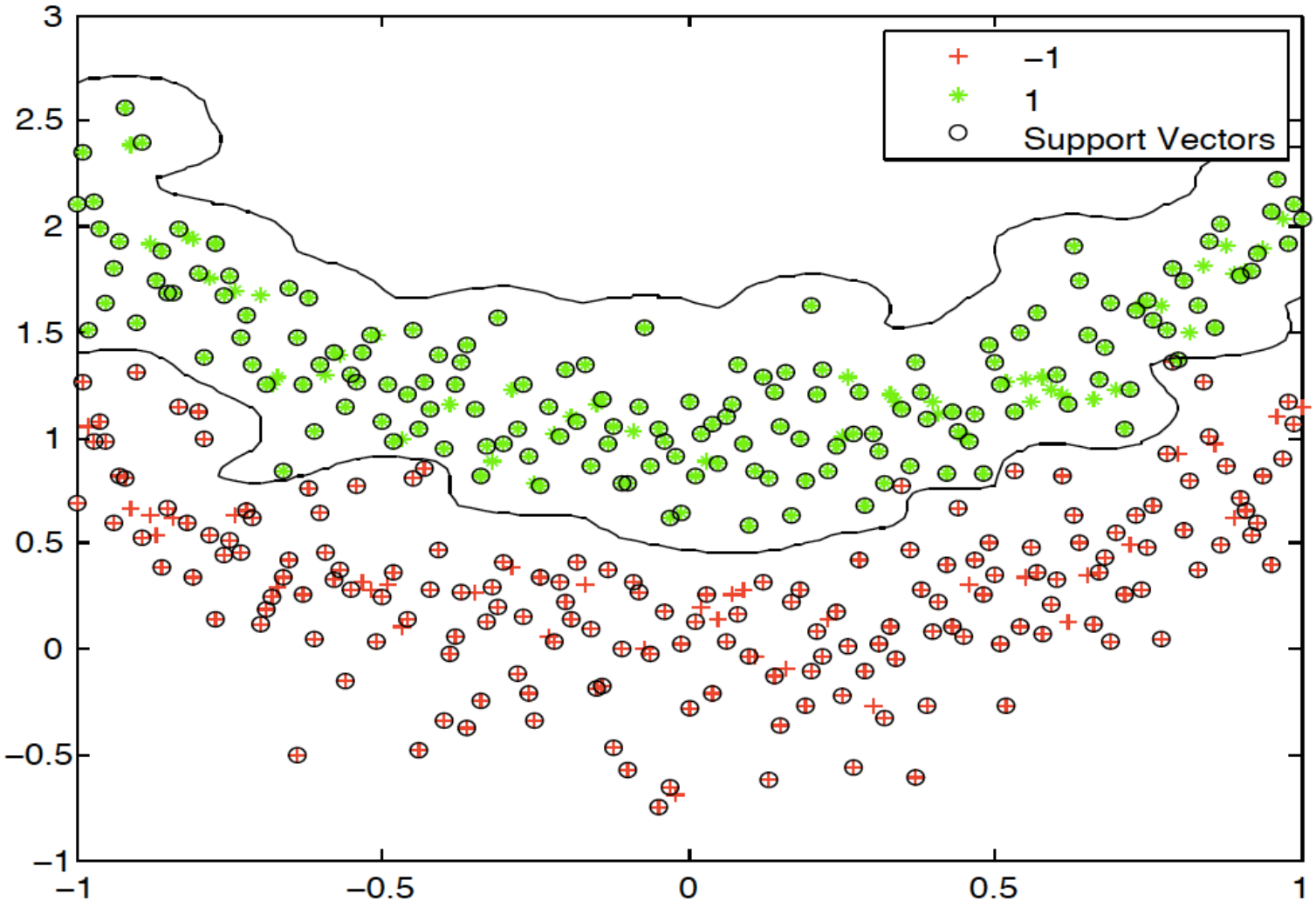
Noyau polynomial de degré 3 : 35 vecteurs supports.



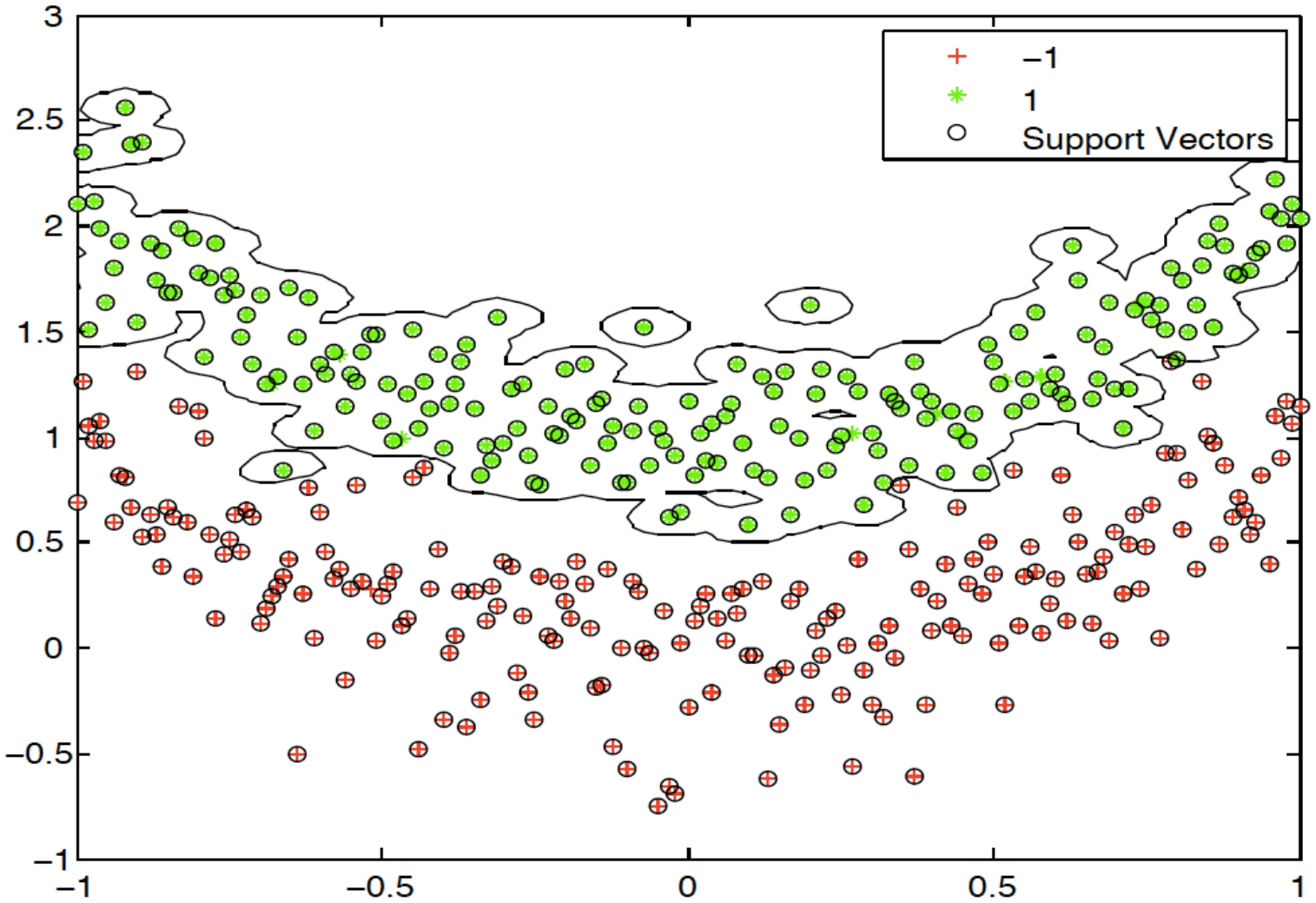
Noyau Gaussien, $\sigma = 1$: 62 vecteurs supports.



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 16×16 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.

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