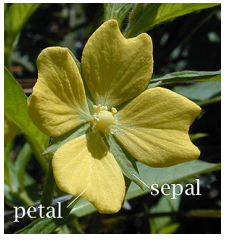
Apprentissage

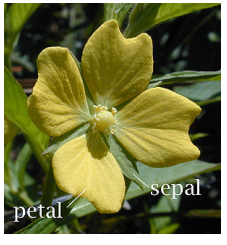
Apprentissage par optimisation numérique

Introduction



sépal		pétal		
long.	larg.	long.	larg.	type
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
.	.	.	.	.
7.0	3.2	4.7	1.4	versic.
6.4	3.2	4.5	1.5	versic.
.	.	.	.	.
6.3	3.3	6.0	2.5	virgin.
5.8	2.7	5.1	1.9	virgin.
.	.	.	.	.

Introduction : Un exemple classique



sépal		pétal		
long.	larg.	long.	larg.	type
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
.	.	.	.	.
7.0	3.2	4.7	1.4	versic.
6.4	3.2	4.5	1.5	versic.
.	.	.	.	.
6.3	3.3	6.0	2.5	virgin.
5.8	2.7	5.1	1.9	virgin.
.	.	.	.	.

$$f(x) = 0.202 \times \text{long.sep} + 0.264 \times \text{larg.sep} + 1.015 \times \text{long.pet} + 1.38 \times \text{larg.pet}$$

si $f(x) < 5.1$ alors setosa ;
si $5.1 < f(x) < 9.5$ alors versicolor ;
si $9.5 < f(x)$ alors virginica.

Introduction : Un autre exemple

Classification d'images, un exemple $=$ N pixels / niveaux de gris :

40 39 39 39 38 37 37 36 36 35 34 34 34 33 33 33
32 31 31 30 30 29 28 28 27 26 27 26 26 25 25 25
24 23 23 22 21 19 18 16 15 17 20 21 22 23 18 14
25 28 25 23 23 22 21 19 18 16 13 13 13 12 17 19
18 19 14 18 21 22 23 24 25 25 25 25 26 26 26 27
28 28 28 28 29 30 30 30 32 32 32 32 32 31 33 33
34 35 36 36 36 37 37 37 38 39 39 39 39 39 41 40
40 41 41 41 40 42 41 41 41 41 42 41 42 42 41 41
40 40 39 38 38 37 37 36 36 35 35 35 33 34 32 32
31 31 30 29 29 29 29 28 28 27 27 27 26 25 25 25
24 24 23 22 22 20 19 17 16 18 22 22 24 24 19 13
25 28 26 24 24 23 22 21 20 18 16 15 14 13 18 19

$\Rightarrow$ N variables numériques
On cherche une fonction $f(x_1, \ldots, x_N) \rightarrow \{-1, +1\}$ qui distingue par exemple les images d'un « a » des images d'une autre lettre.

Introduction : Le cadre d'apprentissage

On a des *exemples* décrits par n attributs **numériques**, et par leur classe, qui peut être -1 ou $+1$:

- l'espace des entrées est $\mathcal{X} = \pm R^n$
- f est la fonction *cible* à apprendre, $f : \mathbf{R}^n \to \{-1, +1\}$
- un exemple a la forme (x, u), où :
  - $x = (x_1, \ldots, x_n) \in \mathbf{R}^n$, et
  - $u \in \{-1, +1\}$: s'il n'y a pas de *bruit*, $u = f(x)$
- $\mathcal{E} \subseteq \mathbf{R}^n \times \{-1, +1\}$

Introduction : Le cadre d'apprentissage

On a des *exemples* décrits par n attributs **numériques**, et par leur classe, qui peut être -1 ou $+1$:

- l'espace des entrées est $\mathcal{X} = \pm R^n$
- f est la fonction *cible* à apprendre, $f : \mathbf{R}^n \to \{-1, +1\}$
- un exemple a la forme (x, u), où :
  - $x = (x_1, \ldots, x_n) \in \mathbf{R}^n$, et
  - $u \in \{-1, +1\}$: s'il n'y a pas de *bruit*, $u = f(x)$
- $\mathcal{E} \subseteq \mathbf{R}^n \times \{-1, +1\}$

On cherche à classer, donc on cherche

$$h : \mathbf{R}^n \to \mathbf{R} \text{ telle que } h(x) \times u > 0 \text{ pour tout } (x, u) \in \mathcal{E}.$$

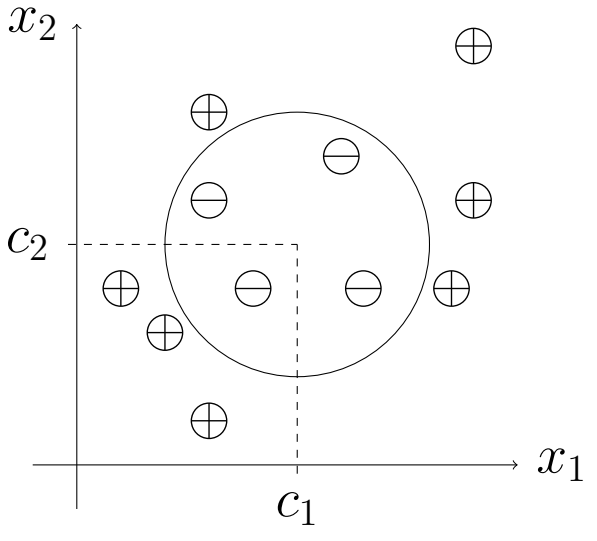
On est dans le cadre de la classification supervisée.

Remarque: on ne cherche pas exactement à avoir $h(x) = \pm 1$, on veut juste que $h(x)$ et y aient le même signe pour les exemples. On pourrait définir $h^s(x) = \mathsf{signe}(h(x)) = \pm 1$.

Introduction : Le cadre d'apprentissage

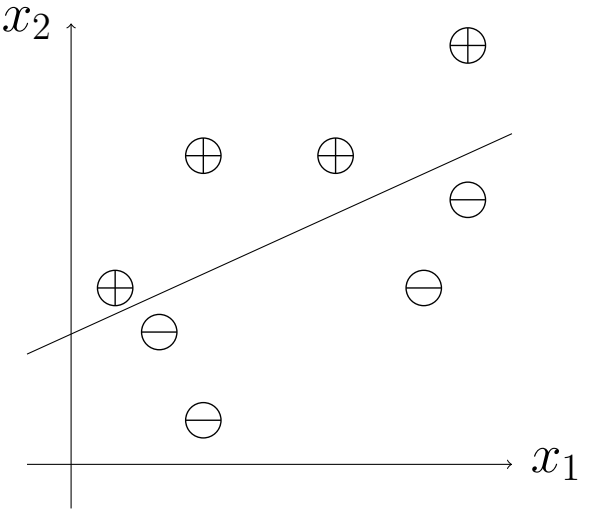
En dimension 2, on peut représenter sur un plan:

- les exemples par un $\oplus$ ou un $\ominus$ suivant le signe de $f(x)$;
- la courbe d'équation $h(x) = 0$.



Séparation non linéaire

$h(x_1, x_2) = (x_1 - c_1)^2 + (x_2 - c_2)^2 - R$



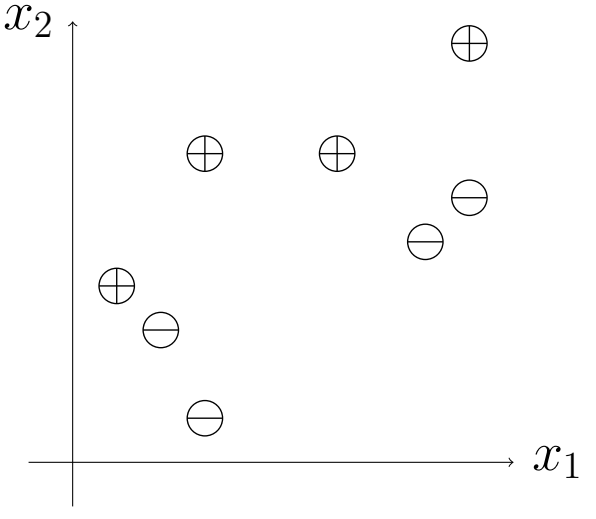
Séparation linéaire

$h(x_1, x_2) = a_1 x_1 + a_2 x_2 + b$

$\mathcal{E}$ est *linéairement séparable*

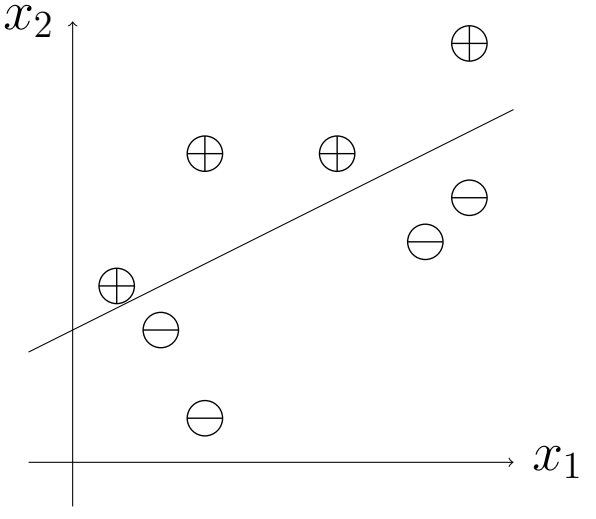
Apprentissage de séparation linéaire

- on cherche un hyperplan séparateur, d'équation
$$a_1x_1 + a_2x_2 + \ldots + a_nx_n + b = a \cdot x + b = 0$$
- on cherche $a = (a_1, \ldots, a_n) \in \mathbf{R}^n$, $b \in \mathbf{R}$
tels que $(a \cdot x + b) \times u > 0$ pour tout $(x, u) \in \mathcal{E}$
- on note $h_{a,b}$ l'hypothèse définie par
$$h_{a,b}(x) = a \cdot x + b = a_1x_1 + a_2x_2 + \ldots + a_nx_n + b.$$



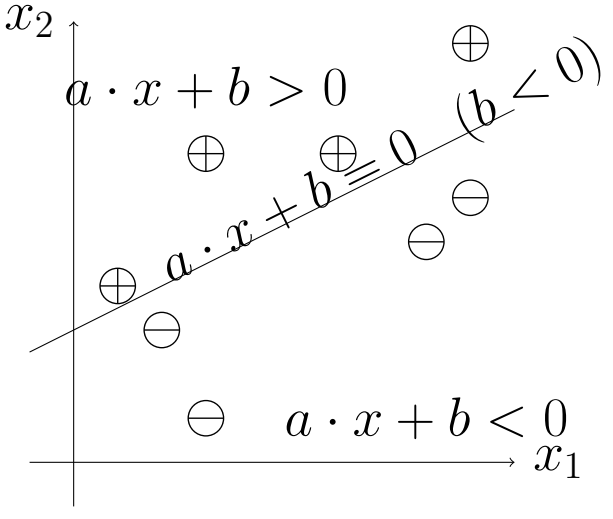
Apprentissage de séparation linéaire

- on cherche un hyperplan séparateur, d'équation
$$a_1x_1 + a_2x_2 + \ldots + a_nx_n + b = a \cdot x + b = 0$$
- on cherche $a = (a_1, \ldots, a_n) \in \mathbf{R}^n$, $b \in \mathbf{R}$
tels que $(a \cdot x + b) \times u > 0$ pour tout $(x, u) \in \mathcal{E}$
- on note $h_{a,b}$ l'hypothèse définie par
$$h_{a,b}(x) = a \cdot x + b = a_1x_1 + a_2x_2 + \ldots + a_nx_n + b.$$



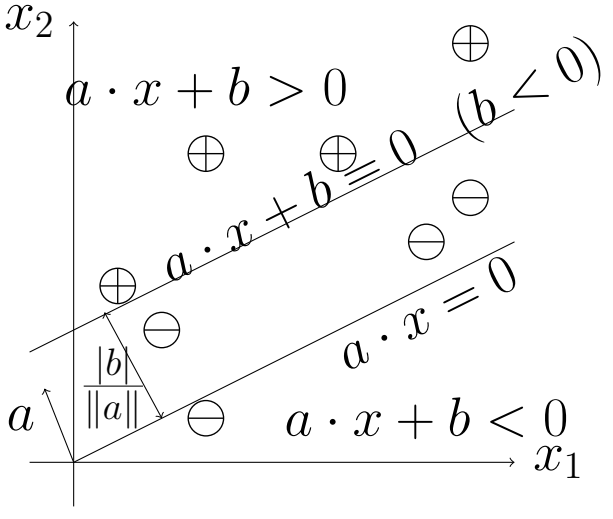
Apprentissage de séparation linéaire

- on cherche un hyperplan séparateur, d'équation
$$a_1x_1 + a_2x_2 + \ldots + a_nx_n + b = a \cdot x + b = 0$$
- on cherche $a = (a_1, \ldots, a_n) \in \mathbf{R}^n$, $b \in \mathbf{R}$
tels que $(a \cdot x + b) \times u > 0$ pour tout $(x, u) \in \mathcal{E}$
- on note $h_{a,b}$ l'hypothèse définie par
$$h_{a,b}(x) = a \cdot x + b = a_1x_1 + a_2x_2 + \ldots + a_nx_n + b.$$



Apprentissage de séparation linéaire

- on cherche un hyperplan séparateur, d'équation
$$a_1x_1 + a_2x_2 + \ldots + a_nx_n + b = a \cdot x + b = 0$$
- on cherche $a = (a_1, \ldots, a_n) \in \mathbf{R}^n$, $b \in \mathbf{R}$
tels que $(a \cdot x + b) \times u > 0$ pour tout $(x, u) \in \mathcal{E}$
- on note $h_{a,b}$ l'hypothèse définie par
$$h_{a,b}(x) = a \cdot x + b = a_1x_1 + a_2x_2 + \ldots + a_nx_n + b.$$



Apprentissage de séparation linéaire : Le perceptron

Trouver a et b tels que $a \cdot x + b \times u > 0$ pour tout $(x, u) \in \mathcal{E}$?
Si on part de a, b quelconques : pour tout $(x, u) \in \mathcal{E}$,

Apprentissage de séparation linéaire : Le perceptron

Trouver a et b tels que $a \cdot x + b \times u > 0$ pour tout $(x, u) \in \mathcal{E}$?
Si on part de a, b quelconques : pour tout $(x, u) \in \mathcal{E}$,

- si $u = +1$ et $a \cdot x + b \leq 0$:
  $a' = a + \tau x$, $b' = b + \tau \ \Rightarrow a' \cdot x + b' = a \cdot x + b + \tau(1 + x^2) > a \cdot x + b$

Apprentissage de séparation linéaire : Le perceptron

Trouver a et b tels que $a \cdot x + b \times u > 0$ pour tout $(x, u) \in \mathcal{E}$?
Si on part de a, b quelconques : pour tout $(x, u) \in \mathcal{E}$,

- si $u = +1$ et $a \cdot x + b \leq 0$:
  $a' = a + \tau x$, $b' = b + \tau \Rightarrow a' \cdot x + b' = a \cdot x + b + \tau(1 + x^2) > a \cdot x + b$
- si $u = -1$ et $a \cdot x + b \geq 0$:
  $a' = a - \tau x$, $b' = b - \tau \Rightarrow a' \cdot x + b' = a \cdot x + b - \tau(1 + x^2) < a \cdot x + b$

Apprentissage de séparation linéaire : Le perceptron

Trouver a et b tels que $a \cdot x + b \times u > 0$ pour tout $(x, u) \in \mathcal{E}$?
Si on part de a, b quelconques : pour tout $(x, u) \in \mathcal{E}$,

- si $u = +1$ et $a \cdot x + b \leq 0$:
  $a' = a + \tau x$, $b' = b + \tau \Rightarrow a' \cdot x + b' = a \cdot x + b + \tau(1 + x^2) > a \cdot x + b$
- si $u = -1$ et $a \cdot x + b \geq 0$:
  $a' = a - \tau x$, $b' = b - \tau \Rightarrow a' \cdot x + b' = a \cdot x + b - \tau(1 + x^2) < a \cdot x + b$

L'algorithme

1. $a \leftarrow (0, \ldots, 0)$; $b \leftarrow 0$; `pas_fini` $\leftarrow$ `vrai`;
2. Tant que `pas_fini` faire
   (a) `pas_fini` $\leftarrow$ `faux`;
   (b) pour chaque $(x, u) \in \mathcal{E}$, si $(a \cdot x + b) \times u \leq 0$ alors
   $a \leftarrow a + \tau \times u \times x$;
   $b \leftarrow b + \tau \times u$;
   `pas_fini` $\leftarrow$ `vrai`;
3. Retourner (a, b).

($\tau > 0$ permet de régler la vitesse de convergence)

Apprentissage de séparation linéaire : Le perceptron

Propriétés

- algorithme *incrémental* : il suffit de le relancer si de nouveaux exemples sont ajoutés.
- convergence garantie si $\mathcal{E}$ est *linéairement séparable*, c'est-à-dire s'il existe une hypothèse linéaire qui permet de classer $\mathcal{E}$;
- ne converge pas dans le cas contraire.

Remarque : si on part de coefficients nuls, on peut exprimer le résultat de l'algorithme sous la forme

$$a^{(N)} = \sum_{(x,u)\in\mathcal{E}} \alpha_x x$$

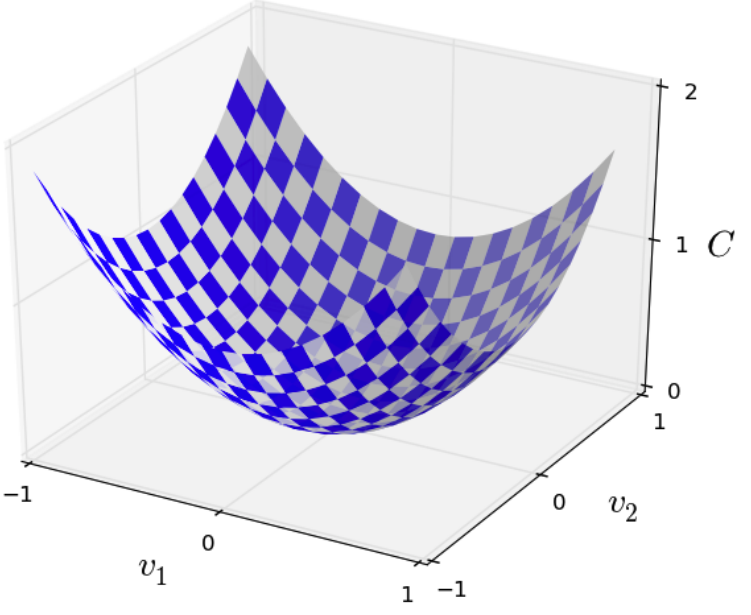
où $\alpha_x = \tau N(x) u$

Apprentissage de séparation linéaire : Descente en gradient

Dans la pratique, exemples pas toujours exacts
(exemple : reconnaissance de caractères)
$\Rightarrow$ on n'a pas toujours des exemples séparables
$\Rightarrow$ chercher la *meilleure* séparation possible

Erreur empirique pour a, b et $\mathcal{E}$ donnés :

$$E(a, b, \mathcal{E}) = \frac{1}{2} \sum_{(x,u)\in\mathcal{E}} (a \cdot x + b - u)^2$$



Apprentissage de séparation linéaire : Descente en gradient

On cherche a, b donnant une erreur minimale
(dans le cas ou $\mathcal{E}$ n'est pas linéairement séparable, il n'est pas possible d'avoir une erreur nulle).

$\Rightarrow$ on « descend » le long de la surface représentant $E(a, b, \mathcal{E})$ en fonction de a et b : $\frac{\delta E}{\delta a}$ et $\frac{\delta E}{\delta b}$ donnent la direction de la « montée ».

Algorithme :

1. $a \leftarrow (0, \ldots, 0)$; $b \leftarrow 0$;
2. Tant que `pas_fini` faire
   (a) pour chaque $(x, u) \in \mathcal{E}$,
   $$a \leftarrow a + \tau \times (u - (a \cdot x + b)) \times x \ ;$$
   $$b \leftarrow b + \tau \times (u - (a \cdot x + b)) \ ;$$
3. Retourner (a, b).

Apprentissage de séparation linéaire : Descente en gradient

Propriétés

- convergence peut être très lente ;
- convergence même si les exemples ne sont pas linéairement séparables.

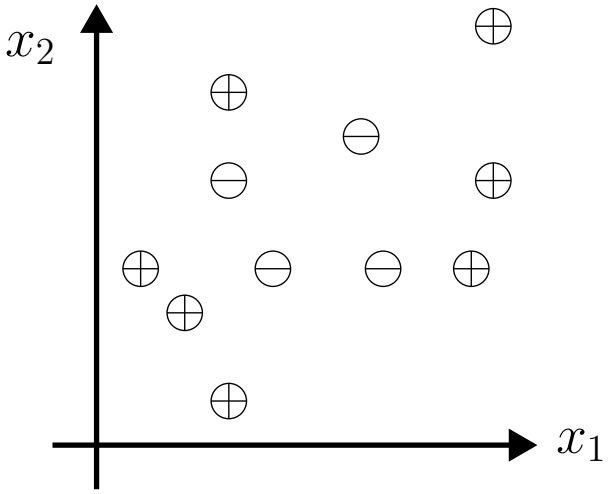
Apprentissage de séparation linéaire : C'est un problème d'optimis

Trouver a, b qui minimisent $\frac{1}{2}\Sigma_{(x,u)\in\mathcal{E}}(a \cdot x + b - u)^2$:

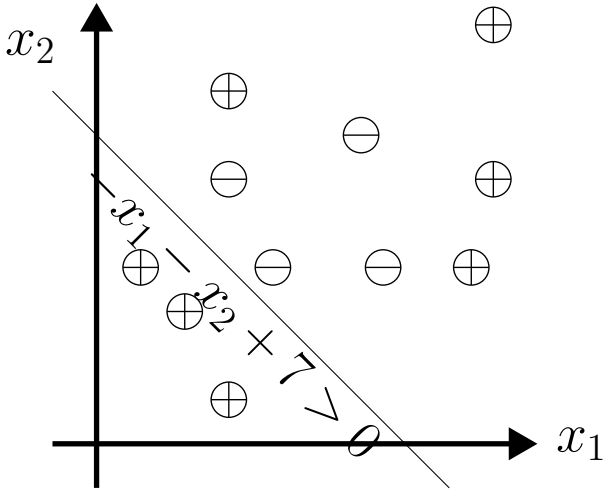
- on peut utiliser d'autres types d'erreurs : $\log(1 + e^{-u(a\cdot x+b)})$ (perte logistique)
  mène à d'autres algorithmes
- régularisation = terme dans l'expression à minimiser
  impose des contraintes supplémentaires sur a et b, surtout pour éviter le surapprentissage (overfitting).
  par exemple : minimiser $\frac{1}{2}\Sigma_{(x,u)\in\mathcal{E}}(a \cdot x + b - u)^2 + C\|a, b\|$
  C = constante à régler
- nombreux algorithmes existent pour résoudre ces pb. d'optimisation spécifiques

Voir par exemple [YHL12].

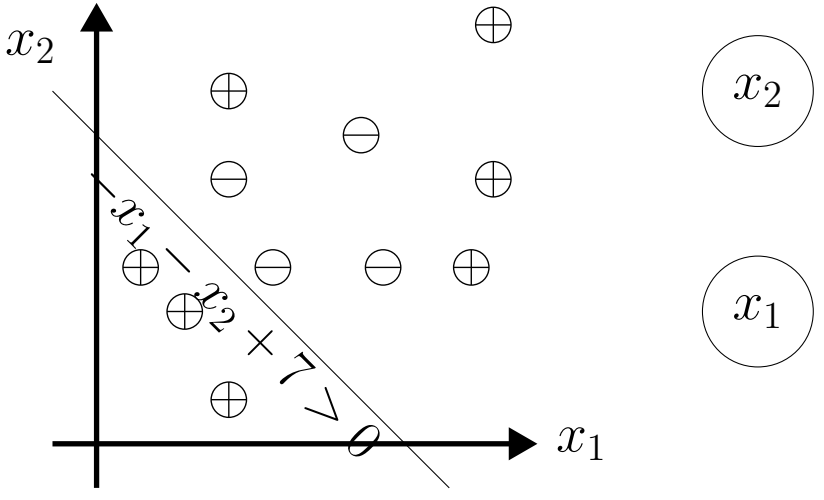
Réseaux de classifieurs linéaires



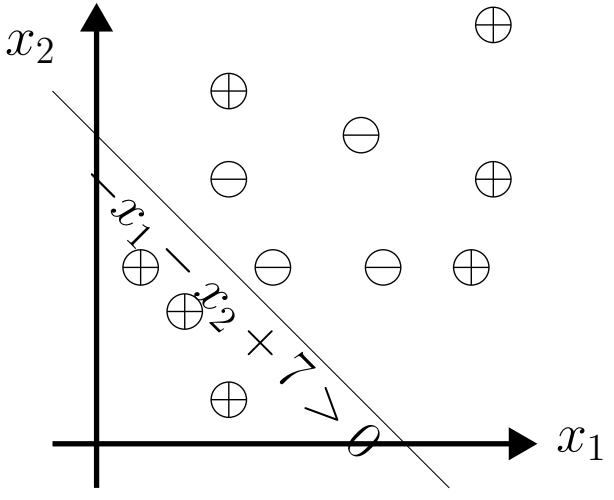
Réseaux de classifieurs linéaires

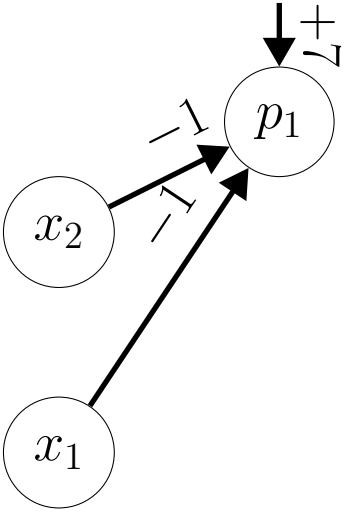


Réseaux de classifieurs linéaires



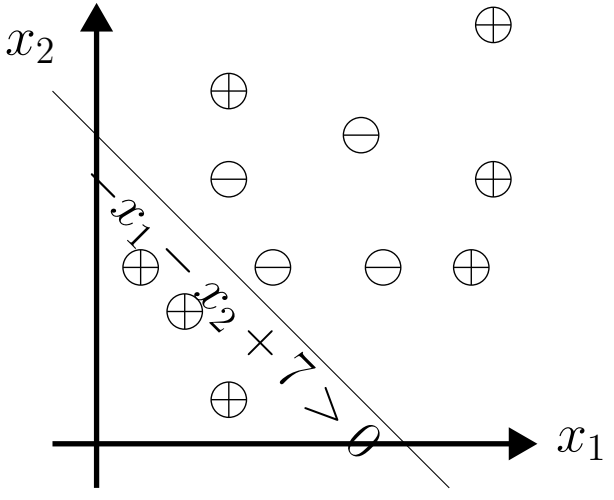
Réseaux de classifieurs linéaires

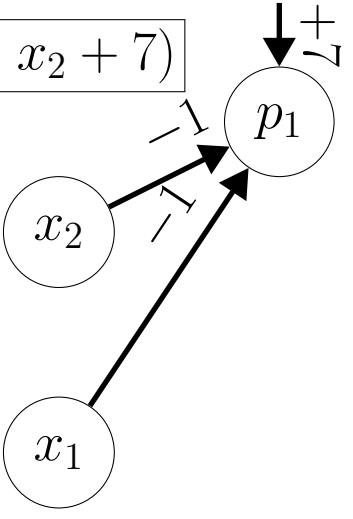




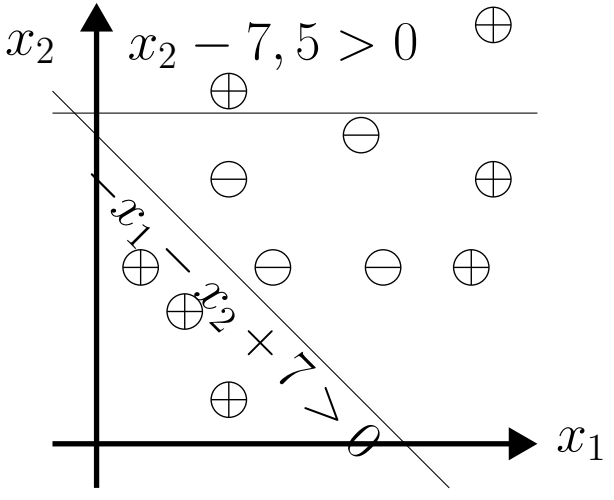
Réseaux de classifieurs linéaires

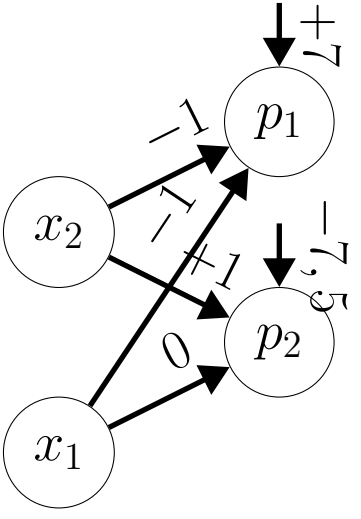
$$p_1(x_1, x_2) = \texttt{signe}(-x_1 - x_2 + 7)$$





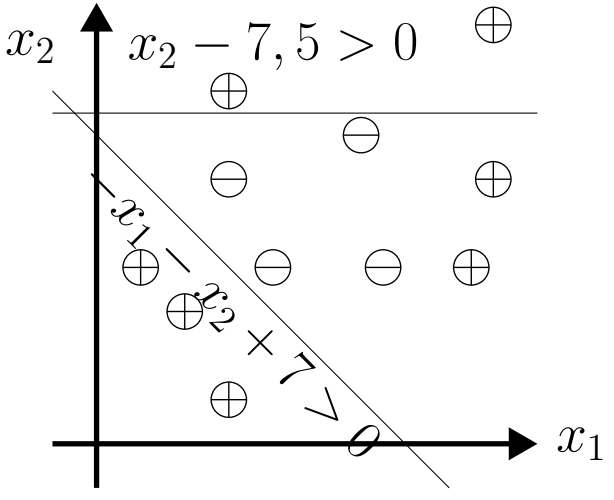
Réseaux de classifieurs linéaires

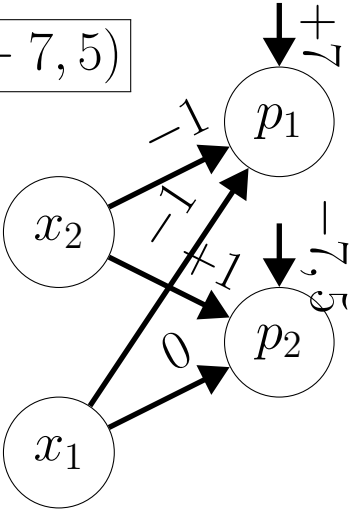




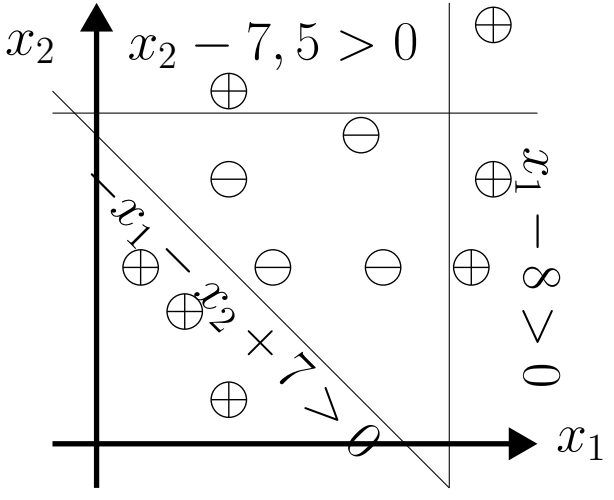
Réseaux de classifieurs linéaires

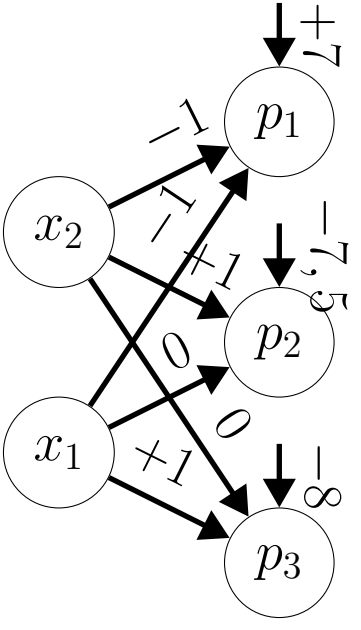
$$p_2(x_1, x_2) = \texttt{signe}(x_2 - 7,5)$$





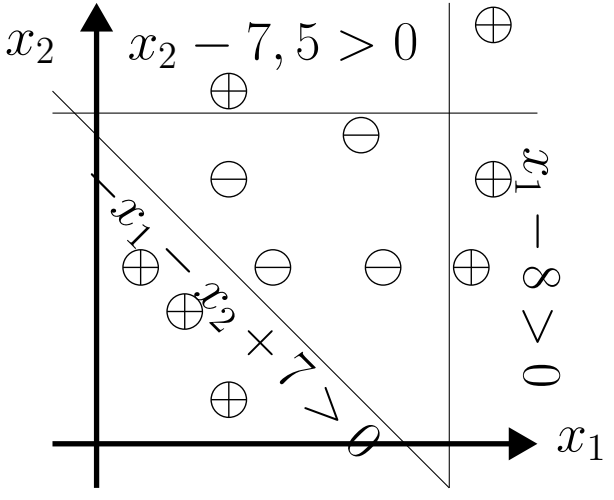
Réseaux de classifieurs linéaires

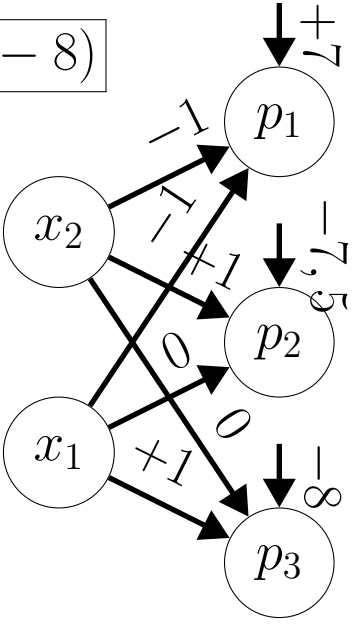




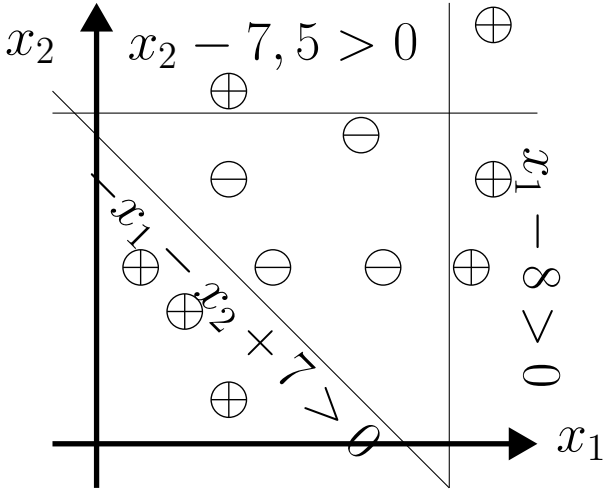
Réseaux de classifieurs linéaires

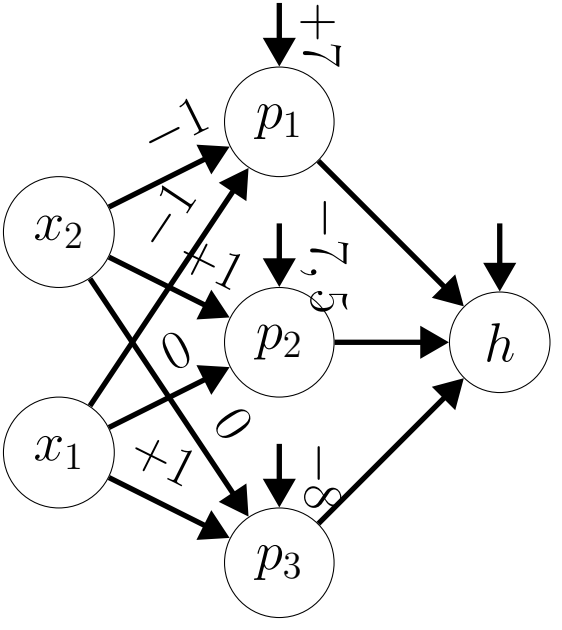
$$p_3(x_1, x_2) = \texttt{signe}(x_1 - 8)$$





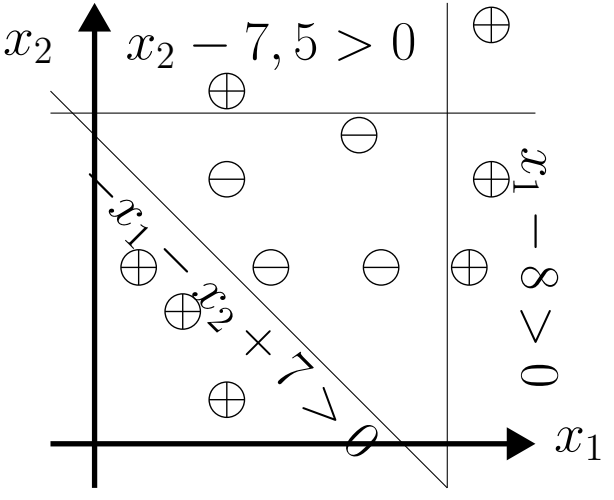
Réseaux de classifieurs linéaires

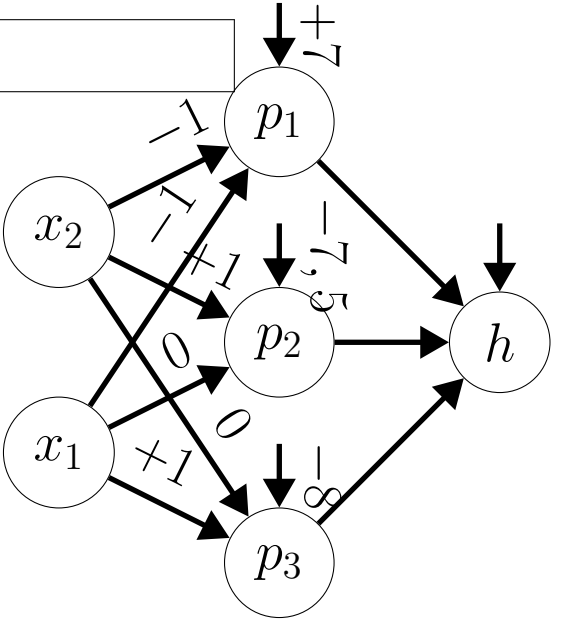




Réseaux de classifieurs linéaires

$h(x_1, x_2) =$





Réseaux de classifieurs linéaires : Définition

Réseau de classifieurs linéaires = graphe acyclique orienté :

- n nœuds d'entrées $X_1, \ldots, X_n$,
  correspondant aux n variables $x_1, \ldots x_n$
- 1 nœud de sortie H : l'hypothèse à apprendre
  (on peut aussi avoir plusieurs nœuds de sortie)
- 1 ou plusieurs « couches cachées »

$\Rightarrow$ $\mathcal{N}$ = l'ensemble des nœuds, $\mathcal{N}^*$ = nœuds des couches cachées

- $\mathtt{Pa}(N)$ = ensemble des parents de N, pour $N \in \mathcal{N}^* \cup \{H\}$;
- a_{MN} = poids de l'arc allant de M à N ;
  b_N = poids de N

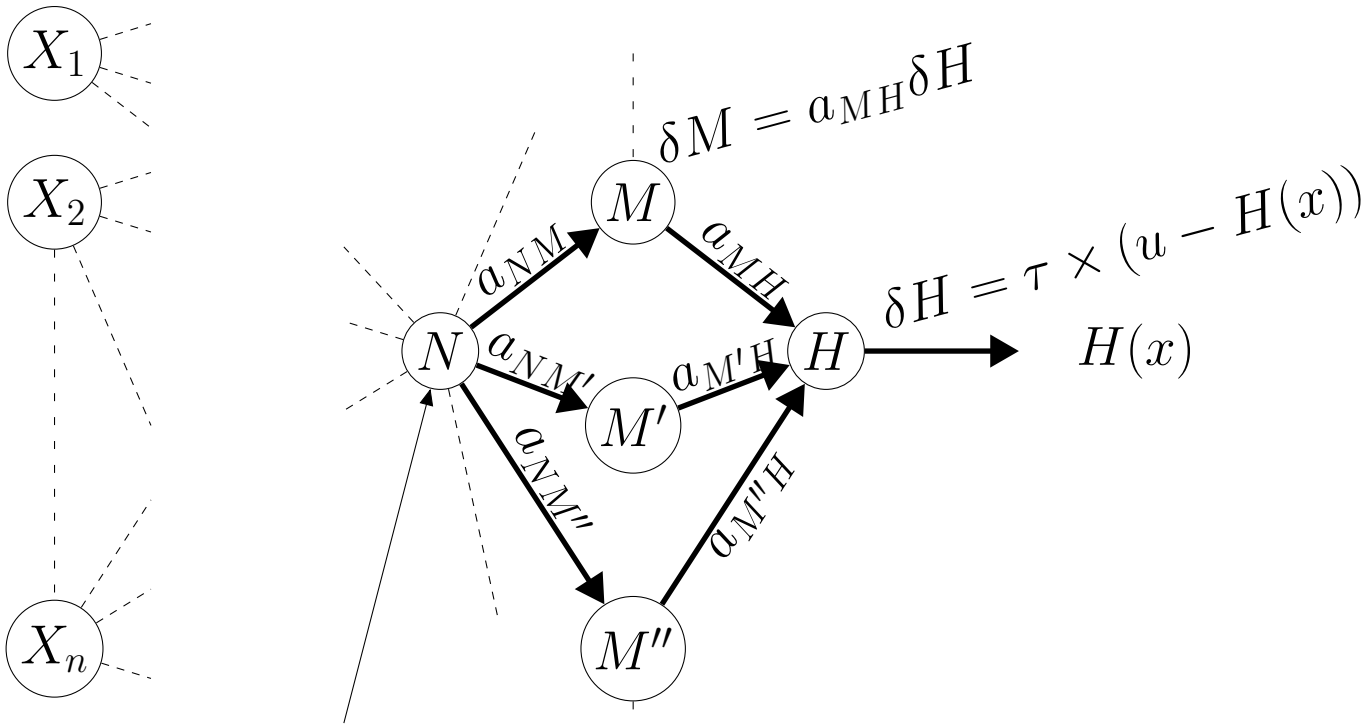
Calcul de $h(x)$ (propagation) :
$N(x) = \mathtt{signe}(\Sigma_{M \in \mathtt{Pa}(N)}\, a_{MN} M(x) + b_N) = \mathtt{signe}(a_N \cdot \mathtt{Pa}(N)(x) + b_N)$
où a_N est le vecteur des poids des arcs de $\mathtt{Pa}(N)$ à N

$\Rightarrow$ pour $x = (x_1, \ldots, x_n) \in R^n$ donné, on calcule $h(x)$ en propageant des variables à H

Réseaux de classifieurs linéaires : Définition

On montre que toute fonction continue de $\mathbf{R}^n$ dans $\mathbf{R}$ peut être approximée d'aussi près qu'on veut par un réseau de neurones à 1 couche cachée et une couche de sortie.

Réseaux de classifieurs linéaires : Apprentissage des coefficients



Réseaux de classifieurs linéaires : Apprentissage des coefficients

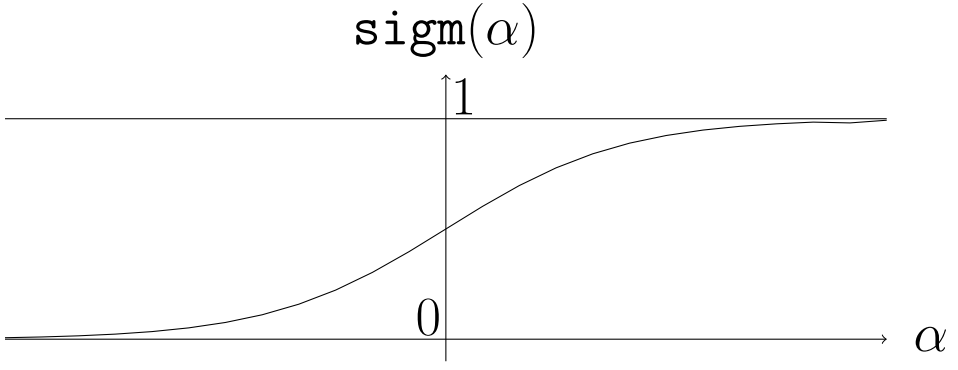
(Attention à l'initialisation : coefs $\neq 0$!)

Tant que « pas fini », faire :

1. Pour chaque $(x, u) \in \mathcal{E}$ faire
   (a) *// propagation*
   pour chaque $N \in \mathcal{N}^*$, en partant des successeurs des entrées et en allant vers la sortie, calculer $N(x)$;
   (b) *// calcul de la correction à apporter en H*
   $\delta H \leftarrow \tau \times (u - H(x))$;
   (c) *// rétropropagation de la correction*
   Pour chaque $N \in \mathcal{N}^*$, en partant des prédécesseurs de H et en allant vers les entrées, faire :
   $\delta N \leftarrow \Sigma_{N \in \mathtt{Pa}(M)}\, a_{NM} \delta M$;
   (d) *// mise à jour des coefficients de $\mathcal{N}$*
   Pour chaque $N \in \mathcal{N}^* \cup \{H\}$, faire :
   $a_N \leftarrow a_N + \delta N \times \mathtt{Pa}(N)(x)$; $b_N \leftarrow b_N + \delta N$;

Réseaux de neurones

$\texttt{sigm}(\alpha) = \frac{1}{1+e^{-\lambda\alpha}}$ où λ est une constante



Plus λ est grand, plus la courbe se rapproche d'une marche brusque en $\alpha = 0$.
$\Rightarrow N(x) = \texttt{sigm}(a_N \cdot \texttt{Pa}(N)(x) + b_N)$.
Remarque : on a maintenant $N(x) \in [0, 1]$.

Modification de l'algorithme d'apprentissage :
Descente en gradient pour minimiser $E(h, \mathcal{E}) = \frac{1}{2}\Sigma_{(x,u)\in\mathcal{E}}(h(x) - u)^2$.
Avec $\lambda = 1$: $\texttt{sigm}'(\alpha) = \texttt{sigm}(\alpha) \times (1 - \texttt{sigm}(\alpha)) \Rightarrow$
$\delta H \leftarrow \tau \times H(x) \times (1 - H(x)) \times (u - H(x))$
$\delta N \leftarrow N(x) \times (1 - N(x)) \times \Sigma_{N\in\texttt{Pa}(M)}\, a_{NM}\delta M$

Réseaux de neurones

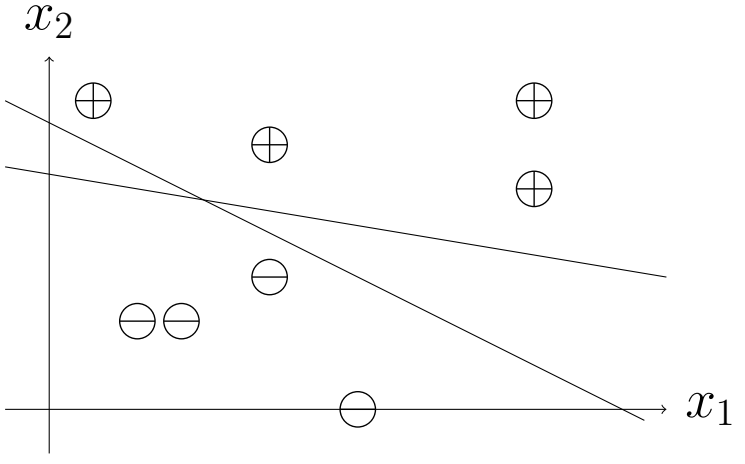
Minima locaux: appliquer plusieurs fois l'algorithme en prenant des poids initiaux différents ;
on garde alors le réseau qui donne la meilleure performance sur un ensemble d'exemples de validation

Surapprentissage: pour l'éviter, on peut surveiller la performance du réseau sur un ensemble de validation au cours de l'apprentissage, et arrêter d'apprendre lorsque la performance ne s'améliore pas.

Apprentissage de la structure: dans le cas où on n'a pas d'idée de la structure du réseau à apprendre, il faut apprendre aussi la structure. On peut pour cela partir d'un réseau avec peu de neurones en couche cachée, et ajouter un a un des neurones, en apprenant à chaque fois les poids du réseau: on arrête quand l'ajout de neurone n'améliore pas la performance.

Classes multiples: lorsqu'on doit apprendre un concept ayant un nombre fini de valeurs en sortie mais supérieur à 2, c'est-à-dire que $|\mathcal{Y}| > 2$, on peut apprendre un réseau de neurones ayant plusieurs neurones en sortie : un par classe.

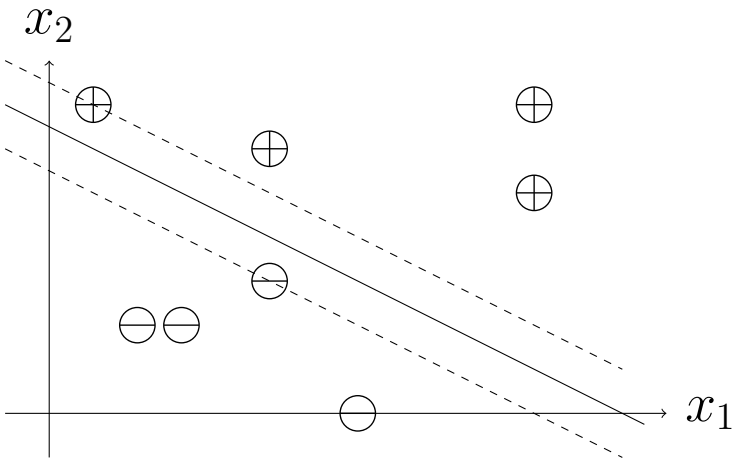
Séparateurs linéaires à vaste marge



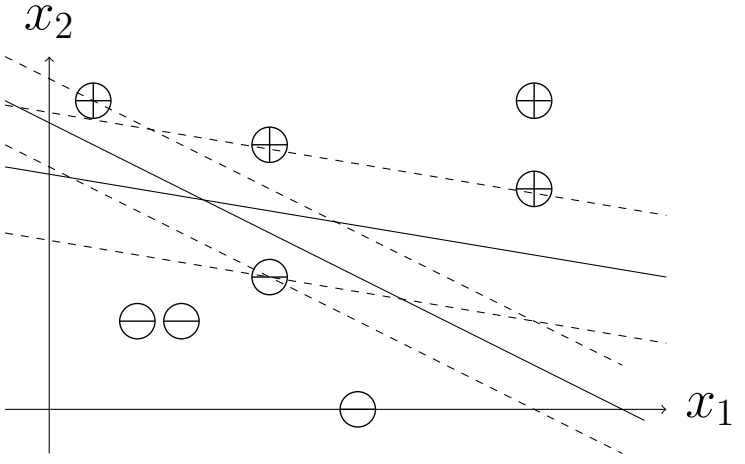
Si $\mathcal{E}$ est linéairement séparable, plusieurs droites sont possibles
$\Rightarrow$ laquelle choisir?

On veut la droite la plus prometteuse, c'est-à-dire celle qui classera au mieux les instances à venir (et non connues).

Séparateurs linéaires à vaste marge : Maximisation de la marge



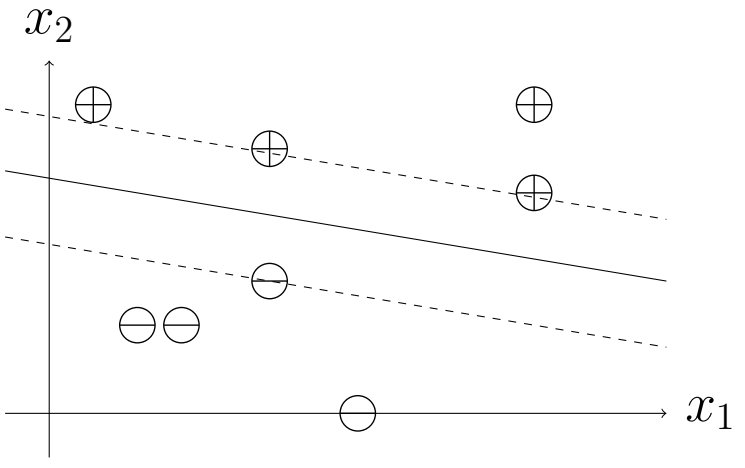
Séparateurs linéaires à vaste marge : Maximisation de la marge



Marge associée à une droite d :
c'est la largeur du « tube » le + large centré sur la droite et qui ne contient aucun exemple.

Plusieurs séparatrices possibles, toutes n'ont pas la même marge.

Séparateurs linéaires à vaste marge : Maximisation de la marge



On va chercher la droite associée à la plus grande marge
$\Rightarrow$ Séparateur à Vaste Marge (SVM).

Remarque: cette droite ne dépend en fait que de quelques exemples de $\mathcal{E}$, on les appelle les *exemples critiques*, ou *vecteurs supports*, *support vector* en Anglais $\Rightarrow$ Support Vector Machine.

Séparateurs linéaires à vaste marge : Maximisation de la marge

Pourquoi la marge la plus grande possible:

- graphiquement, semble satisfaisant
- semble la + sûre: si on fait une petite erreur sur la localisation de la droite, on obtiendra peu d'erreur.
- marche très bien en pratique

Séparateurs linéaires à vaste marge : Optimisation quadratique

On cherche donc

- $a \in \mathbf{R}^n$, $b \in \mathbf{R}$, tels que $u \times (a \cdot x + b) > 0$ pour tout $(x, u) \in \mathcal{E}$,
- et tels que l'hyperplan obtenu soit celui qui offre la meilleure marge.

Distance d'un x qq à l'hyperplan d'équations $= a \cdot x + b$:

$$\frac{|a \cdot x + b|}{\|a\|}$$

(Pourquoi ? . . .)

Séparateurs linéaires à vaste marge : Optimisation quadratique

On cherche donc

- $a \in \mathbf{R}^n$, $b \in \mathbf{R}$, tels que $u \times (a \cdot x + b) > 0$ pour tout $(x, u) \in \mathcal{E}$,
- et tels que l'hyperplan obtenu soit celui qui offre la meilleure marge.

Distance d'un x qq à l'hyperplan d'équations $= a \cdot x + b$:

$$\frac{|a \cdot x + b|}{\|a\|}$$

(Pourquoi ? ...)

Si $\pi(x)$ est la projection de x sur l'hyperplan, on a $a \cdot \pi(x) + b = 0 = a \cdot (\pi(x) - x) + a \cdot x + b = \|a\| \, \|\pi(x) - x)\| + a \cdot x + b$ donc $\|\pi(x) - x)\| = |a \cdot x + b| / \|a\|$

Séparateurs linéaires à vaste marge : Optimisation quadratique

Distance d'un x qq à l'hyperplan d'équations $= a \cdot x + b$:

$$\frac{|a \cdot x + b|}{\|a\|}$$

$\Rightarrow$ on cherche a et b

- qui maximisent $\frac{1}{\|a\|} \min_{(x,u)\in\mathcal{E}} u \times (a \cdot x + b)$
- sous les contraintes $u \times (a \cdot x + b) > 0$ pour tout $(x, u) \in \mathcal{E}$.

Pas très facile à résoudre...

Séparateurs linéaires à vaste marge : Optimisation quadratique

Distance d'un x qq à l'hyperplan d'équations $= a \cdot x + b$:

$$\frac{|a \cdot x + b|}{\|a\|}$$

$\Rightarrow$ on cherche a et b

- qui maximisent $\frac{1}{\|a\|} \min_{(x,u)\in\mathcal{E}} u \times (a \cdot x + b)$
- sous les contraintes $u \times (a \cdot x + b) > 0$ pour tout $(x, u) \in \mathcal{E}$.

Pas très facile à résoudre...

...mais équivalent à chercher a et b

- qui maximisent $\frac{1}{\|a\|}$
- sous les contraintes $u \times (a \cdot x + b) \geq 1$ pour tout $(x, u) \in \mathcal{E}$.

(Pourquoi ? . . .)

Séparateurs linéaires à vaste marge : Optimisation quadratique

Distance d'un x qq à l'hyperplan d'équations $= a \cdot x + b$:

$$\frac{|a \cdot x + b|}{\|a\|}$$

$\Rightarrow$ on cherche a et b

- qui maximisent $\frac{1}{\|a\|} \min_{(x,u)\in\mathcal{E}} u \times (a \cdot x + b)$
- sous les contraintes $u \times (a \cdot x + b) > 0$ pour tout $(x, u) \in \mathcal{E}$.

Pas très facile à résoudre...

...mais équivalent à chercher a et b

- qui maximisent $\frac{1}{\|a\|}$
- sous les contraintes $u \times (a \cdot x + b) \geq 1$ pour tout $(x, u) \in \mathcal{E}$.

Toujours pas si facile à résoudre ?

Cela revient encore à chercher a, b

- qui *minimisent* $\|a\|^2$
- sous les contraintes $u \times (a \cdot x + b) \geq 1$ pour tout $(x, u) \in \mathcal{E}$.

Séparateurs linéaires à vaste marge : Optimisation quadratique

Chercher a, b

- qui *minimisent* $\|a\|^2$
- sous les contraintes $u \times (a \cdot x + b) \geq 1$ pour tout $(x, u) \in \mathcal{E}$.
- $n + 1$ inconnues : $a_1, \ldots, a_n, b$
- Objectif : minimiser $\|a\|^2$
- Contraintes : $u \times (a \cdot x + b) \geq 1$ pour tout $(x, u) \in \mathcal{E}$
- La fonction objectif est convexe

$\Rightarrow$ Une solution unique

$\Rightarrow$ Soluble en temps polynomial $O(n^3)$.
(Remarque : si $n = 256$ pixels...)

Exemple $\mathcal{E} = \{((2,2),+1), ((4,2),-1), ((3,3),+1), ((3,1),-1)\}$

Séparateurs linéaires à vaste marge : Problème dual

Rappel Problème dual d'1 problème d'optimisation

Exemple : trouver $x_1, x_2 \geq 0$ qui maximisent $x_1 + 6x_2$

sous contraintes (1) : $x_1 \leq 2$, (2) : $x_2 \leq 3$, (3) : $x_1 + x_2 \leq 4$?

On a :

$(1) + 6(2) \Rightarrow x_1 + 6x_2 \leq 20$

Mieux :

$5(2) + (3) \Rightarrow x_1 + 6x_2 \leq 19$

Plus généralement :

$\alpha_1(1) + \alpha_2(2) + \alpha_3(3) \Rightarrow (\alpha_1 + \alpha_3)x_1 + (\alpha_2 + \alpha_3)x_2 \leq 2\alpha_1 + 3\alpha_2 + 4\alpha_3$

$\Rightarrow$ **Problème dual :**

Trouver $\alpha_1, \alpha_2, \alpha_3$ qui minimisent $2\alpha_1 + 3\alpha_2 + 4\alpha_3$

sous contraintes $\alpha_1 + \alpha_3 \geq 1$, $\alpha_2 + \alpha_3 \geq 6$.

Séparateurs linéaires à vaste marge : Problème dual

Chercher a, b

- qui *minimisent* $\|a\|^2$
- sous les contraintes $u \times (a \cdot x + b) \geq 1$ pour tout $(x, u) \in \mathcal{E}$.

On montre que c'est équivalent au problème dual : chercher une famille de coefficients $(\alpha_x)_{(x,u)\in\mathcal{E}}$:

- qui maximise $\sum\limits_{(x,u)\in\mathcal{E}} \alpha_x - \frac{1}{2}\| \sum\limits_{(x,u)\in\mathcal{E}} \alpha_x u x\|^2$
- sous les contraintes $\begin{cases} 0 \leq \alpha_x \text{ pour tout } (x,u) \in \mathcal{E} \\ \sum\limits_{(x,u)\in\mathcal{E}\in\mathcal{E}} \alpha_x u = 0 \end{cases}$

$\Rightarrow$ Nouveau problème d'optimisation :

- $|\mathcal{E}|$ inconnues : un α_x pour chaque $(x, u) \in \mathcal{E}$
- Objectif : maximiser $\sum\limits_{(x,u)\in\mathcal{E}} \alpha_x - \frac{1}{2}\| \sum\limits_{(x,u)\in\mathcal{E}} \alpha_x u x\|^2$
- Contraintes : $0 \leq \alpha_x$, $\Sigma_{(x,u)\in\mathcal{E}}\, \alpha_x u = 0$
- **On n'a plus que $|\mathcal{E}|$ variables !**
  (Intéressant si $|\mathcal{E}| < n$)

Séparateurs linéaires à vaste marge : Problème dual

Chercher une famille de coefficients $(\alpha_x)_{(x,u)\in\mathcal{E}}$:

- qui maximize $\sum\limits_{(x,u)\in\mathcal{E}} \alpha_x - \frac{1}{2}\| \sum\limits_{(x,u)\in\mathcal{E}} \alpha_x u x\|^2$
- sous les contraintes $\begin{cases} 0 \leq \alpha_x \text{ pour tout } (x,u) \in \mathcal{E} \\ \sum\limits_{(x,u)\in\mathcal{E}} \alpha_x u = 0 \end{cases}$

Un hyperplan séparateur a alors pour équation:

$$h(x) = \sum_{(x',u')\in\mathcal{E}} \alpha_{x'} u' (x' \cdot (x - x^{(0)})) + u^{(0)} = 0$$

où $(x^{(0)}, u^{(0)})$ est un exemple critique, caractérisé par $\alpha_{x^{(0)}} \neq 0$. Cela revient à prendre $a = \Sigma_{(x,u)\in\mathcal{E}}\, \alpha_x u x$ et $b = u^{(0)} - x^{(0)} \cdot a$.

Dans les deux cas : fonction d'objectif convexe, contraintes linéaires

$\Rightarrow$ une solution unique, qu'on calcule en $O(\mathsf{nb\_variables}^3)$

Justification : ...

Séparateurs linéaires à vaste marge : Problème dual

Sous forme matricielle : problème d'*optimisation quadratique* $\Rightarrow$ algorithmes classiques (Matlab, R, ...)

trouver $\alpha = \begin{pmatrix} \alpha_{x_1} \\ . \\ . \\ \alpha_{x_R} \end{pmatrix}$ qui maximise $\alpha^t H \alpha + f^t \alpha$ sous les contraintes:

$$\begin{cases} -I\alpha \leq 0 \\ C\alpha = 0 \end{cases}$$

où $H_{ij} = -u_i u_j x_i x_j$, $f = \begin{pmatrix} 1 \\ . \\ . \\ 1 \end{pmatrix}$, $I = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ . & . & \dots & . \\ 0 & . & \dots & 1 \end{pmatrix}$, $C = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 0 & . & \dots & 0 \\ . & . & \dots & . \\ 0 & . & \dots & 0 \end{pmatrix}$

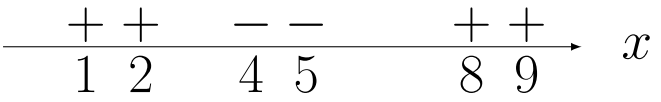
SVMs non linéaires

Dans la pratique, on peut rarement trouver une séparation linéaire. On peut alors :

- « projeter » les exemples dans un espace $\mathbf{R}^m$ avec $m > n$
- essayer d'apprendre une séparation linéaire dans $\mathbf{R}^m$.

SVMs non linéaires : Un exemple en dimension 1

$\mathcal{E} = \{(1,+1),(2,+1),(4,-1),(5,-1),(8,+1),(9,+1)\}$



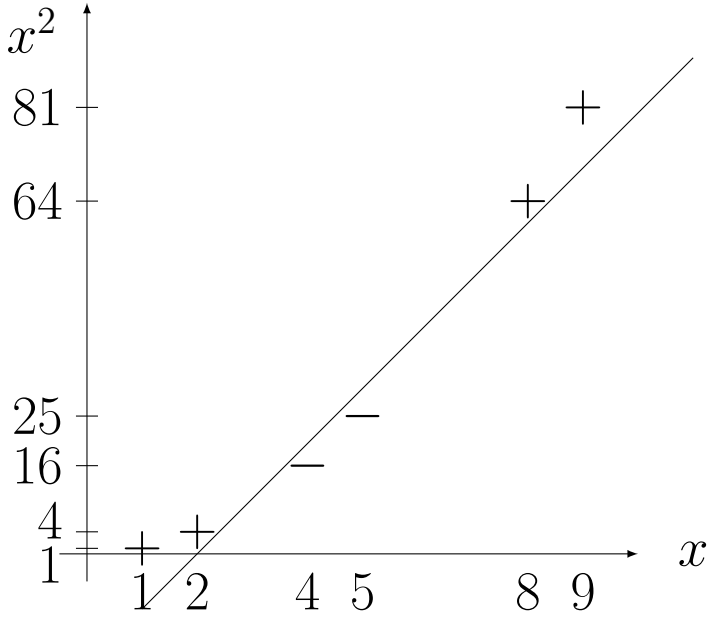
$\Rightarrow$ séparation linéaire = trouver a tel que

- $u = +1$ si $x > a$;
- $u = -1$ si $x < a$.

$\Rightarrow$ clairement impossible.

SVMs non linéaires : Un exemple en dimension 1

Par contre, si on « projette » chaque instance dans $\mathbf{R}^2$ à l'aide de la fonction $x \mapsto (x, x^2)$ alors on peut séparer les points obtenus par une droite:



SVMs non linéaires : Optim. dans un espace de dim. $> n$

De manière générale :

- Ensemble $\mathcal{E}$ d'exemples de $\mathbf{R}^n \times \{-1, +1\}$
- Séparation linéaire impossible

$\Rightarrow$ Chercher $\Phi : \mathbf{R}^n \to R^m$, $m > n$,
telle qu'il existe $(a, b) \in \mathbf{R}^m \times R$ avec :
$u(a \cdot \Phi(x) + b) \geq 1$ pour tout $(x, u) \in \mathcal{E}$

SVMs non linéaires : Optim. dans un espace de dim. $> n$

$\Rightarrow$ Nouveau problème d'optimisation :

$$\begin{cases} \text{Maximiser } \sum\limits_{(x,u)\in\mathcal{E}} \alpha_x - \frac{1}{2} \sum\limits_{(x,u),(x',u')\in\mathcal{E}} \alpha_x \alpha_{x'} u u' (\Phi(x)\cdot\Phi(x')) \\ \text{avec } 0 \leq \alpha_x \text{ pour tout } (x,u) \in \mathcal{E} \\ \text{et } \sum\limits_{x\in\mathcal{E}} \alpha_x u = 0 \end{cases}$$

La fonction séparatrice a alors pour équation :

$$\sum_{x'\in\mathcal{E}} \alpha_{x'} u' (\Phi(x')\cdot(\Phi(x) - \Phi(x^{(0)}))) + u^{(0)} = 0$$

ou encore $a \cdot \Phi(x) + b = 0$

avec $a = \sum\limits_{x'\in\mathcal{E}} \alpha_{x'} u' \Phi(x')$ et $b = u^{(0)} - \Phi(x^{(0)}) \cdot a$.

SVMs non linéaires : Optim. dans un espace de dim. $> n$

Problème : on arrive dans des espaces de très grande dimension par exemple, si $n = 256$ pixels, et si $\Phi(x) = (x_i x_j)_{1 \leq i,j \leq n}$, $m = \ldots \Rightarrow m^3 = \ldots$

$\Rightarrow$ le « kernel trick » !

SVMs non linéaires : Fonctions à noyau

Problème du changement d'espace : on risque de plonger le problème dans espace de très grande dimension, on a alors des problèmes pour calculer les produits scalaires

Heureusement, il existe des Φ pour lesquelles ce produit scalaire est facile à calculer à partir de l'espace d'origine.

Par exemple, si $\Phi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$, alors $\Phi(x_1, x_2) \cdot \Phi(x'_1, x'_2) = \ldots = ((x_1, x_2) \cdot (x'_1, x'_2))^2$.

Fonction noyau = fonction $k : \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}$ telle qu'il existe m et Φ :

- $\Phi : \mathbf{R}^n \to R^m$
- $k(x, x') = \Phi(x) \cdot \Phi(x')$

SVMs non linéaires : Fonctions à noyau

Problème d'optimisation associé :

- Trouver $(\alpha_x)_{(x,u)\in\mathcal{E}}$
- qui maximise $\sum_{x\in\mathcal{E}} \alpha_x - \frac{1}{2} \sum_{(x,u),(x',u')\in\mathcal{E}} \alpha_x \alpha_{x'} u u' k(x,x')$
- sous les contraintes: $\begin{cases} 0 \leq \alpha_x \text{ pour tout } x \in \mathcal{E} \\ \Sigma_{(x,u)\in\mathcal{E}}\, \alpha_x u = 0 \end{cases}$
- Solution :
  $h(x) = \texttt{signe}(\sum_{(x',u')\in\mathcal{E}} \alpha_{x'} y' k(x', x - x^{(0)}) + u^{(0)})$

SVMs non linéaires : Noyaux couramment utilisés

- Gaussien : $k(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$
- $k(x, y) = \tanh(\kappa x \cdot y - \delta)$ (cf réseaux de neurones)
- Polynômes contenant tous les termes $x_{i_1} \dots x_{i_k}$ jusqu'à un certain degré K.
- ...

SVMs non linéaires : Combinaisons de noyaux

si k et k' sont deux fonctions à noyaux, alors les fonctions suivantes sont à noyau (entre autres) :

- $ck(x, x')$
- $f(x)k(x, x')f(x')$
- $q(k(x, x'))$
- $\exp(k(x, x'))$
- $k + k'$
- $k \times k'$

...

où f est une fonction réelle quelconque, q est un polynome à coefficients ≥ 0.

SVMs non linéaires : Noyaux sur des espaces non numériques

Le noyau k doit représenter une notion de similarité entre exemples. Rien n'empêche de définir de telles fonctions sur des espaces autres que $\mathbf{R}^n$.

Une contrainte : la fonction k doit être

- symétrique :
- semi-définie positive : la matrice K des $k(x_i, x_j)$ doit vérifier

$$v^t \times K \times v \geq 0 \text{ pour tout } v \in \mathbf{R}^{|\mathcal{E}|}$$

Remarque : K est appelée la matrice de Gram

SVMs non linéaires : Noyaux sur des espaces non numériques

Par exemple, noyaux sur des ensembles :

- un ensemble de référence fini $\mathcal{A}$
  (Par exemple, pour la détection de spam, $\mathcal{A}$ est l'ensemble des mots du dictionnaire)
- un exemple est défini par une partie $A \subseteq \mathcal{A}$
- $k(A, A') = 2^{|A \cap A'|}$
- ou plus généralement $k(A, A') = \mu(A \cap A')$
  où μ est une probabilité sur $\mathcal{A}$.

On peut définir des noyaux sur des chaînes de caractères, des graphes, des arbres, ...

Bruit

En pratique : il y a des $(x, u) \in \mathcal{E}$ tels que $u \neq f(x)$
(où f est la fonction qu'on veut apprendre)

$\Rightarrow$ on risque de passer dans un espace de très grande dimension pour apprendre une fonction de risque empirique nul, et qui aura un risque réel élevé.

Il vaut mieux apprendre une fonction de risque réel non nul dans un espace de dimension plus faible.

Bruit

$\Rightarrow$ On va essayer de minimiser

$$a \cdot a + C \times (\text{distance entre mal classés et leur zone})$$

C est un paramètre à régler : plus il est grand, plus on pénalise les exemples mal classés.

$$\begin{cases} \text{Minimiser } \frac{1}{2}\|a\|^2 + C \sum\limits_{(x,u)\in\mathcal{E}} \epsilon_x \\ \text{avec } u(a \cdot x + b) \geq 1 - \epsilon_x \text{ pour tout } (x,u) \in \mathcal{E} \end{cases}$$

Formulation duale :

$$\begin{cases} \text{Maximiser } \sum\limits_{(x,u)\in\mathcal{E}} \alpha_x - \frac{1}{2}\| \sum\limits_{(x,u)\in\mathcal{E}} \alpha_x u x\|^2 \\ \text{avec } 0 \leq \alpha_x \leq C \text{ pour tout } (x,u) \in \mathcal{E} \\ \text{et } \sum\limits_{(x,u)\in\mathcal{E}} \alpha_x u = 0 \end{cases}$$

Exemple : *Ranking SVM* [Joa02]

Le problème : un ensemble de documents D, une requête $q \Rightarrow$ ordonner les documents de D en fonction de la requête.

Pour apprendre à ordonner, exemples de la forme (q, d_1, d_2) :
« pour la requête q, le document d_1 est plus pertinent que d_2 ».

$\Rightarrow$ Paires de vecteurs $(\Phi(q, d_1), \Phi(q, d_2))$, où $\Phi(q, d)$ indique l'adéquation entre la requête q et le document d, avec par exemple le nombre de mots communs, le *page-rank* de d, etc...
On cherche un vecteur a tel que $a \cdot \Phi(q, d_1) > a \cdot \Phi(q, d_2)$ pour tout exemple de cette forme.

$\Leftrightarrow$ trouver a tq $a \cdot (\Phi(q, d_1) - \Phi(q, d_2)) > 0$, pour tout exemple (q, d_1, d_2)

Maximisation de la marge :

$$\begin{cases} \text{Minimiser } \frac{1}{2}\|a\|^2 + C \sum\limits_{e \in \mathcal{E}} \epsilon_e \\ \text{avec } a \cdot (\Phi(q, d_1) - \Phi(q, d_2) \geq 1 - \epsilon_e \text{ pour tout } e = (q, d_1, d_2) \in \mathcal{E} \end{cases}$$

Conclusions

- Méthode appliquée dans de très nombreux domaines
- Plusieurs librairies existent, permettant d'utiliser la méthode en C++, Java, . . . , et dans d'autres logiciels : R, Matlab
- Les choix de la fonction à noyau et du paramètre C sont prépondéra
- Méthode « *boite noire* », c'est-à-dire qu'on ne peut pas « comprendre » le modèle appris
- Actuellement, difficile d'apprendre avec plus de 10^5 exemples
- Une fois le modèle appris, classification très facile à calculer

Bibliographie

[Joa02] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 133–142. ACM, 2002.

[YHL12] Guo-Xun Yuan, Chia-Hua Ho, and Chih-Jen Lin. Recent advances of large-scale linear classification. *Proceedings of the IEEE*, 100(9):2584 –2603, sept. 2012.

...