

IFT3395/6390

Fondements de l'apprentissage machine

**Classificateurs linéaires  
(et fonctions discriminantes linéaires)**

Professeur: Pascal Vincent

# Qu'est-ce qu'un classifieur linéaire?

- Un algorithme d'apprentissage pour la classification qui permet d'apprendre une fonction de décision  $f(x)$  pour un problème de classification
- Si sa fonction de décision **peut s'exprimer** sous une des formes simples suivantes:

## Classification binaire (2 classes):

$f(\mathbf{x}) = \text{sign}(g(\mathbf{x}))$  où  $g(\mathbf{x})$  est une **fonction discriminante linéaire**:  
réponse: classe -1 ou +1

**Paramètres:**

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \mathbf{w}^t \mathbf{x} + b$$

$$\theta = \{\mathbf{w}, b\}, \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$$

vecteur de poids      bias

## Classification multiclasse ( $m$ classes):

$f(\mathbf{x}) = \text{argmax}(g(\mathbf{x}))$  réponse: classe entre 1 et  $m$

**Paramètres:**

$$g(\mathbf{x}) = \mathbf{W}^T \mathbf{x} + \mathbf{b}$$
$$g(\mathbf{x}) = (g_1(\mathbf{x}), g_2(\mathbf{x}), \dots, g_m(\mathbf{x}))$$

$$\theta = \{\mathbf{W}, \mathbf{b}\}, \mathbf{W} \in \mathbb{R}^{d \times m}, \mathbf{b} \in \mathbb{R}^m$$

matrice de poids      vecteur de biais

# Relation entre cas binaire et multiclass avec $m=2$

$$\begin{aligned}f(\mathbf{x}) &= \arg \max [\mathbf{W}^T \mathbf{x} + \mathbf{b}] \\&= \arg \max [(\mathbf{W}_{:1}^T \mathbf{x} + b_1, \quad \mathbf{W}_{:2}^T \mathbf{x} + b_2)] \\&= 2 \quad \text{si} \quad \mathbf{W}_{:2}^T \mathbf{x} + b_2 > \mathbf{W}_{:1}^T \mathbf{x} + b_1 \quad (\text{et } 1 \text{ sinon}) \\&= 2 \quad \text{si} \quad (\mathbf{W}_{:2} - \mathbf{W}_{:1})^T \mathbf{x} + (b_2 - b_1) > 0 \quad (\text{et } 1 \text{ sinon}) \\&= 2 \quad \text{si} \quad \underbrace{\text{sign}((\mathbf{W}_{:2} - \mathbf{W}_{:1})^T \mathbf{x} + (b_2 - b_1))}_{\mathbf{w}} = +1 \quad (\text{et } 1 \text{ sinon}) \\&= 1.5 + \frac{1}{2} \text{sign}(\mathbf{w}^T \mathbf{x} + b)\end{aligned}$$

# Géométrie d'une fonction discriminante *linéaire*

- Régions de décision:

$$R_+ = \{x \mid g(x) > 0\} \quad \text{et} \quad R_- = \{x \mid g(x) < 0\}$$

- Frontière de décision  $H = \{x \mid g(x) = 0\}$

- Discriminante linéaire:  $g(x) = \mathbf{w}^T \mathbf{x} + b$

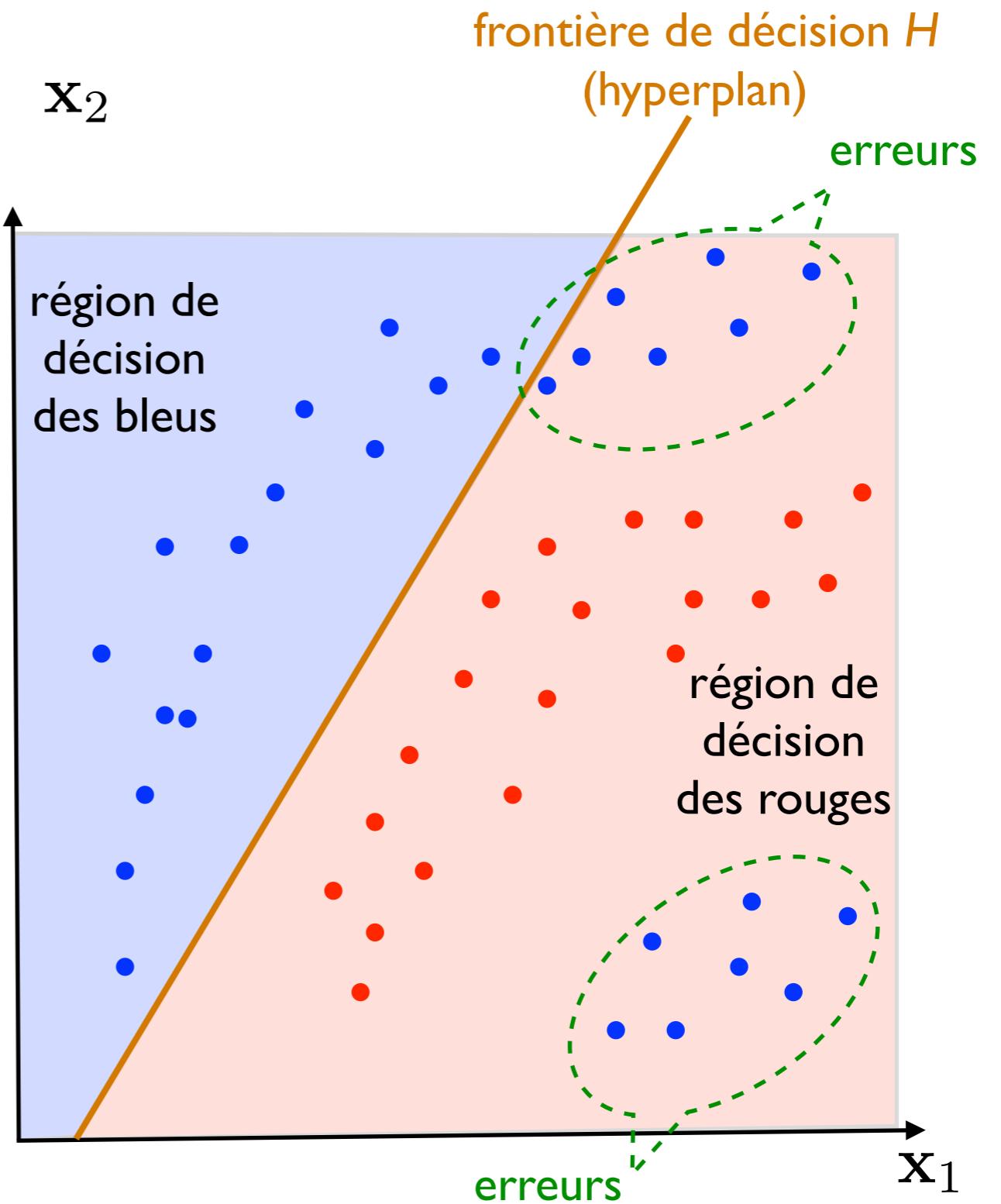
➡ Frontière de décision:  $H = \{x \mid \mathbf{w}^T \mathbf{x} + b = 0\}$

C'est un **hyper-plan**.

➡ Régions de décision: les deux **demi-espace** que sépare cet hyper-plan.

# Géométrie d'une fonction discriminante linéaire

La frontière de décision est un hyperplan.



# Coût d'erreur de classification

- Fonction de décision associée à une fonction discriminante linéaire:

$$y = f_{\theta}(\mathbf{x}) = \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$$

- Coût (perte) d'erreur de classification:

cibles

$$t^{(i)} \in \{-1, +1\}$$

$$L(y, t) = I_{\{y \neq t\}} = I_{\{(\langle \mathbf{w}, \mathbf{x} \rangle + b)t < 0\}}$$

- Taux d'erreur de classification d'une telle fonction sur un ensemble  $D_n = \{(x^{(1)}, t^{(1)}), \dots, (x^{(n)}, t^{(n)})\}$

$$\hat{R}_{\text{classif}}(f_{\theta}, D_n) = \frac{1}{n} \sum_{i=1}^n L(f_{\theta}(x^{(i)}), t^{(i)})$$

# Notion de séparabilité linéaire

- On dit d'un ensemble  $D_n = \{(x^{(1)}, t^{(1)}), \dots, (x^{(n)}, t^{(n)})\}$  qu'il est **linéairement séparable** si (et seulement si) il existe une fonction discriminante linéaire qui fait 0 erreurs de classification sur cet ensemble.
- Géométriquement: si il existe un hyper-plan qui sépare parfaitement les exemples de  $D_n$  (de sorte que tous ceux d'une classe se trouvent dans un demi-espace et ceux de l'autre classe dans l'autre demi-espace)
- Formellement. Si  $t^{(i)} \in \{-1, +1\}$   $D_n$  est linéairement séparable si et seulement si
$$\exists \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R} \quad | \quad \forall i \in \{1, \dots, n\}, \quad \underbrace{\text{sign}(\mathbf{w}^T x^{(i)} + b)}_{f(x^{(i)})} = t^{(i)}$$

# De nombreux algos permettant d'apprendre un «classifieur linéaire»

- Régression linéaire (régularisée ou non) avec des cibles  $+1, -1$ . (Mais pas très approprié pour la classif).
- Classifieur de plus proche moyenne (ou centroid, barycentre)
- Classifieur de Bayes avec densités Gaussiennes (si contraintes à avoir la même covariance).
- Régression logistique
- Algorithme du Perceptron
- Analyse Discriminante Linéaire de Fisher (LDA)
- Machines à vecteurs de support (linéaire)
- ...

# De nombreux algos permettant d'apprendre un «classifieur linéaire»

## Qu'ont ils de semblable?

- Ils engendrent tous une fonction de décision *exprimable* sous la forme paramétrée précédemment définie.
- Pour la classification binaires (2 classes), leur frontière de décision est toujours un *hyperplan*

## En quoi différent-ils?

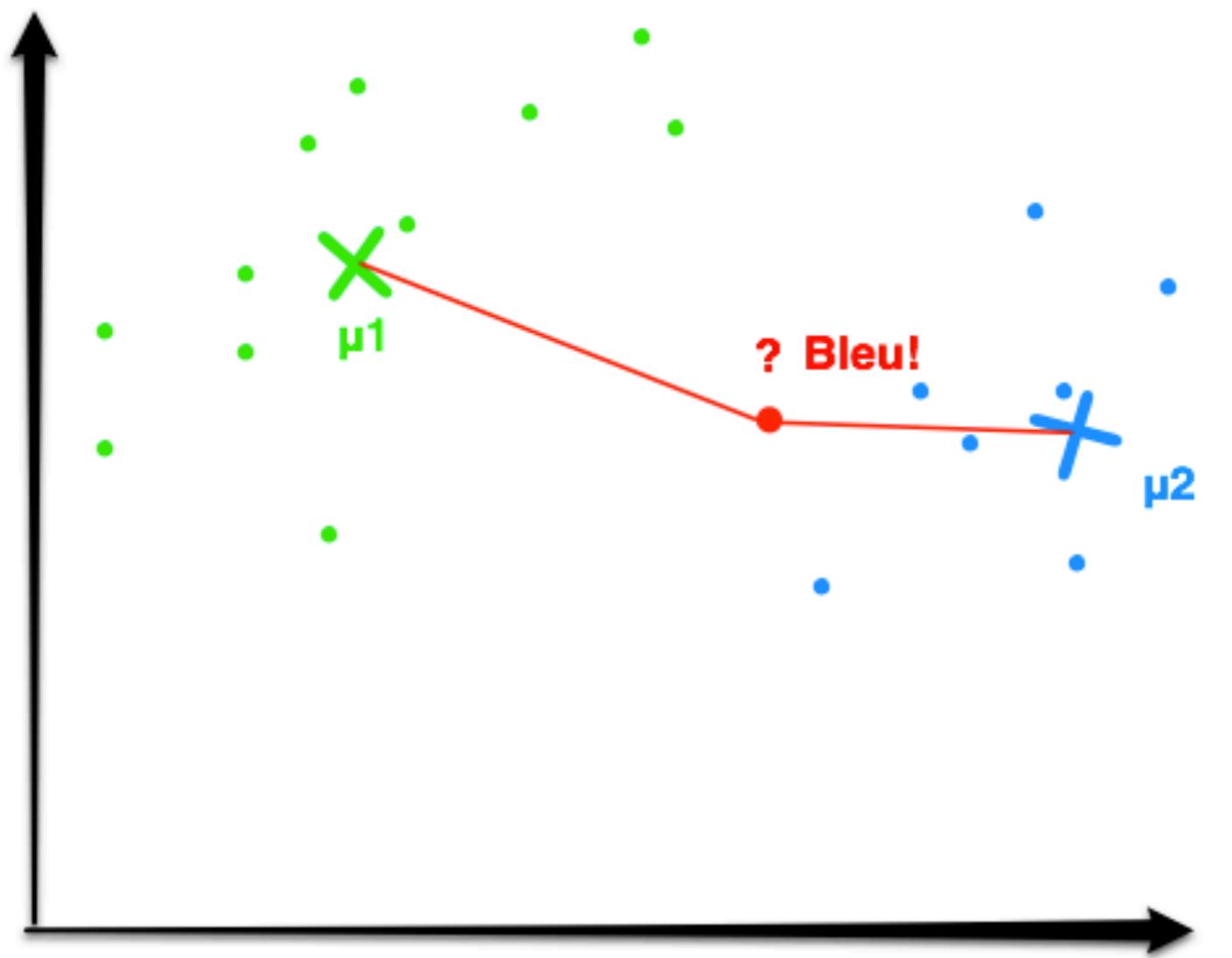
- Pour un même ensemble d'entraînement  $D_n$  ils apprendront possiblement des valeurs de paramètres *differentes*.  
=> donnent lieu à un hyperplan positionné différemment.  
=> Certains généraliseront mieux à de nouveaux exemples

# Ex I: classifieur de plus proche moyenne (centroid/barycentre)

- Paramètres:  $\theta = \{\mu_1, \dots, \mu_m\}$  tous vecteurs de dimension  $d$ .
- Entraînement:  
 **$\mu_m$  = moyenne des exemples de la classe  $m$**
- Décision:  
$$f(x) = \arg \min[(d_{L_2}(\mathbf{x}, \mu_1), \dots, d_{L_2}(\mathbf{x}, \mu_m))]$$

où  $d_{L_2}$  est la distance Euclidienne

# Ex I : classifieur de plus proche moyenne (centroid/barycentre)



# Le classifieur de plus proche moyenne est un classifieur linéaire

Multiclasse:

$$\begin{aligned}f(x) &= \arg \min [(d_{L_2}(\mathbf{x}, \mu_1), \dots, d_{L_2}(\mathbf{x}, \mu_m))] \\&= \arg \min [(\|\mathbf{x} - \mu_1\|^2, \dots, \|\mathbf{x} - \mu_m\|^2)] \\&= \arg \min [(\|\mathbf{x}\|^2 - 2\mu_1^T \mathbf{x} + \|\mu_1\|^2, \dots, \|\mathbf{x}\|^2 - 2\mu_m^T \mathbf{x} + \|\mu_m\|^2)] \\&= \arg \min [(-2\mu_1^T \mathbf{x} + \|\mu_1\|^2, \dots, -2\mu_m^T \mathbf{x} + \|\mu_m\|^2)] \\&= \arg \max [(2\mu_1^T \mathbf{x} - \|\mu_1\|^2, \dots, 2\mu_m^T \mathbf{x} - \|\mu_m\|^2)] \\&= \arg \max [\mathbf{W}^T \mathbf{x} + \mathbf{b}]\end{aligned}$$

$$\text{avec } \mathbf{W}_{:k} = 2v\mu_k \quad \text{et} \quad b_k = -\|\mu_k\|^2$$

Binaire:

$$\begin{aligned}f(\mathbf{x}) &= \text{sign}(\|\mathbf{x} - \mu_-\| - \|\mathbf{x} - \mu_+\|) \\&= \text{sign}(\underbrace{2(\mu_+ - \mu_-)^T}_{\mathbf{w}} \mathbf{x} + \underbrace{(\|\mu_-\|^2 - \|\mu_+\|^2)}_{b})\end{aligned}$$

# Ex 2: Classifieur de Bayes avec densités Gaussiennes

Voir document du même nom pour une dérivation détaillée de ces résultats

- Si les matrices de covariance sont contraintes à être identiques  $\Sigma_i = \Sigma$  cela donne lieu à un classifieur linéaire.
- Si en plus les classes sont à priori équiprobables  $P(Y = C_i) = P(Y = C_j)$  cela équivaut à un classifieur de plus proche moyenne (basé sur la distance Euclidienne si les Gaussiennes sont sphériques/isotropiques, et sur la distance de Mahalanobis dans le cas plus général)
- Si les matrices de covariance sont différentes, on obtient des fonctions discriminantes quadratiques  
=> ça ne donne plus un classifieur linéaire.

# Ex 3: régression linéaire

$$D_n = \{(x^{(1)}, t^{(1)}), \dots, (x^{(n)}, t^{(n)})\} \quad t^{(i)} \in \{-1, +1\}$$

- Rien ne nous empêche d'appliquer la régression linéaire (ou de ridge) avec des cibles +1 et -1
- Mais celle-ci minimise une erreur quadratique  
 $L(y, t) = (y - t)^2$   
plutôt que l'erreur de classification...
- => très sensible aux «outliers» (valeurs extrêmes)  
La frontière de décision va être grandement influencée «tirée» par les points les plus loin d'elle.
- => ne généralisera pas bien pour un pb de classification.

# Et si on minimisait directement l'erreur de classification?

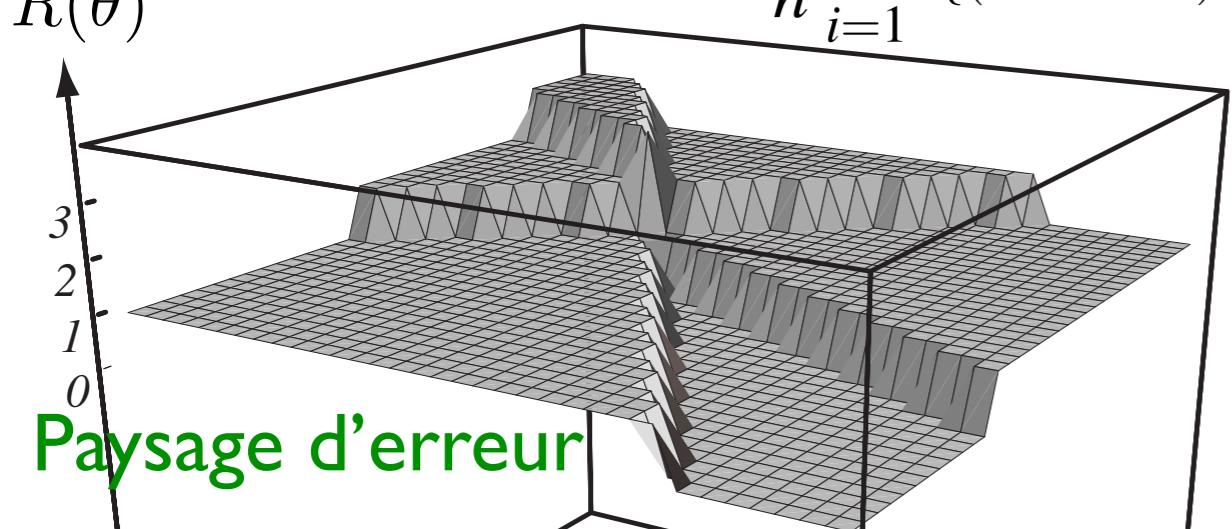
$$D_n = \{(x^{(1)}, t^{(1)}), \dots, (x^{(n)}, t^{(n)})\} \quad t^{(i)} \in \{-1, +1\}$$

$$y = f_\theta(\mathbf{x}) = \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$$

$$L(y, t) = I_{\{y \neq t\}} = I_{\{(\langle \mathbf{w}, \mathbf{x} \rangle + b)t < 0\}}$$

$$J(\theta) = \hat{R}(f_\theta, D_n) = \frac{1}{n} \sum_{i=1}^n L(\mathbf{w}^T \mathbf{x}^{(i)} + b, t^{(i)})$$

$$= \frac{1}{n} \sum_{i=1}^n I_{\{(\mathbf{w}^T \mathbf{x}^{(i)} + b)t^{(i)} < 0\}}$$



Par descente de gradient?

**Problème:** gradient nul partout!  
(car la dérivée de la fonction indicatrice par rapport aux paramètres est nulle)

=> On ne peut pas utiliser la descente de gradient.

# Ex4: algorithme du Perceptron

## (description simple, version en-ligne / stochastique)

$$D_n = \{(x^{(1)}, t^{(1)}), \dots, (x^{(n)}, t^{(n)})\} \quad t^{(i)} \in \{-1, +1\}$$

- Initialiser les paramètres  $w$  et  $b$  à 0
- Répéter tant qu'il demeure des points mal classifiés:
  - ▶ Choisir un prochain exemple  $(x, t)$  au hazard dans l'ensemble  $D_n$ .
  - ▶ Si (avec la valeur actuelle des paramètres)  $x$  est correctement classifié c.a.d. si  $(w^T x + b)t > 0$  alors on ne touche à rien.
  - ▶ Sinon (il est mal classifié) modifier:
$$w \leftarrow w + t x$$
$$b \leftarrow b + t$$

# Faits importants sur le perceptron

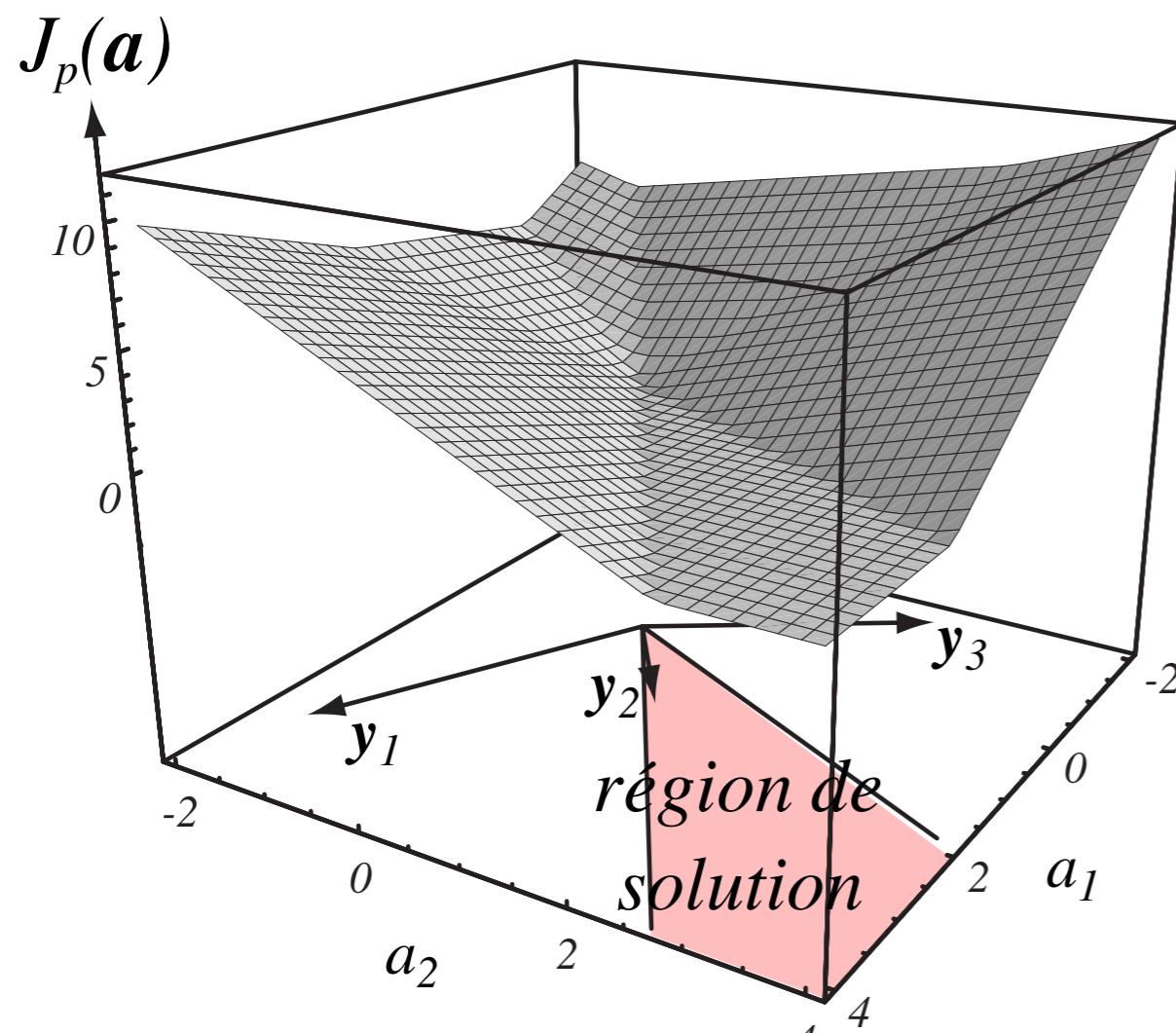
- Inventé en 1957 par Frank Rosenblatt.
- Inspiré par les théories cognitives de Friedrich Hayek et de Donald Hebb.
- Première tentative de «neurone artificiel»
- Si les données sont linéairement séparables, la version stochastique (avec sélection aléatoire de l'exemple) est garantie de trouver un hyper-plan séparateur (donnant 0 erreurs d'entraînement) en un nombre fini d'itérations.
- Si non linéairement séparable: boucle à l'infini.  
=> il faut un autre critère d'arrêt.

# Perceptron et descente de gradient

$$D_n = \{(x^{(1)}, t^{(1)}), \dots, (x^{(n)}, t^{(n)})\} \quad t^{(i)} \in \{-1, +1\}$$

- Le perceptron correspond à une descente de gradient sur l'objectif suivant:

$$J_{\text{perceptron}}(\theta) = \frac{1}{n} \sum_{i=1}^n I_{\{(\mathbf{w}^T \mathbf{x}^{(i)} + b)t^{(i)} < 0\}} (-(\mathbf{w}^T \mathbf{x}^{(i)} + b)t^{(i)})$$



- On peut en dériver des optimisations par descente de gradient stochastique (donne l'algo détaillé précédemment), batch, ou mini-batch.

# Ex 5: La Régression Logistique (cas de deux classes)

**Attention: il s'agit d'un algo de classification!** (en dépit du nom)

Pour une tâche de **classification binaire**:

$$t \in \{0, 1\}$$

On veut estimer la **probabilité conditionnelle**:

$$y \simeq P(t = 1 | \mathbf{x})$$

**On choisit**

$$y \in [0, 1]$$

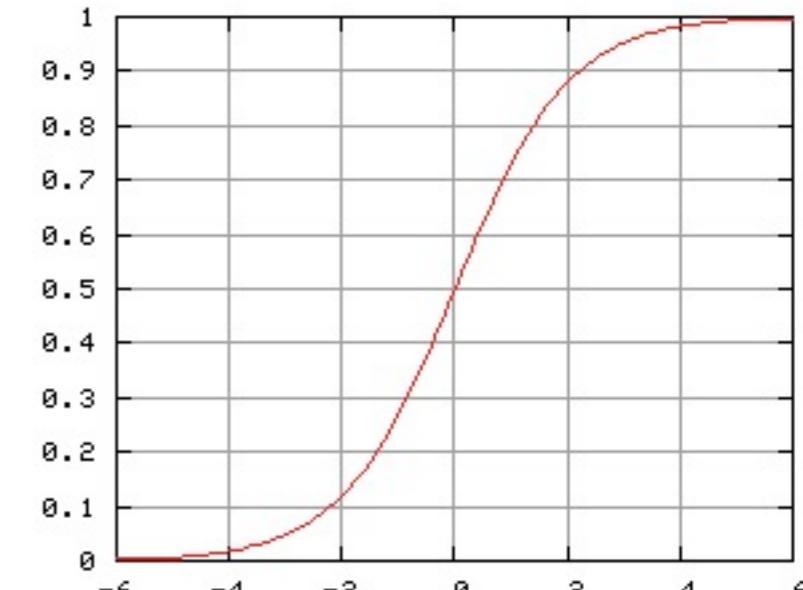
Une fonction de prédiction **non-linéaire** donnant une **probabilité**

$$f_{\theta}(\mathbf{x}) = f_{\mathbf{w}, b}(\mathbf{x}) = \underbrace{\text{sigmoid}(\langle \mathbf{w}, \mathbf{x} \rangle + b)}$$

non-linéarité, fonction de transfert ou d'activation

$$\text{logistic sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

La sigmoïde logistique est l'inverse de la fonction de lien *logit* dans la terminologie des Modèles Linéaires Généralisés (GLMs).



**Perte d'entropie croisée («cross-entropy»)**

$$L(y, t) = -(t \ln(y) + (1 - t) \ln(1 - y))$$

Ici, pas de solution analytique mais l'optimisation est **convexe**

# optimisation par descente de gradient

$$D_n = \{(x^{(1)}, t^{(1)}), \dots, (x^{(n)}, t^{(n)})\} \quad t^{(i)} \in \{0, 1\}$$

$$\hat{R}_\lambda(f_\theta, D_n) = \left( \sum_{i=1}^n L(f_\theta(\mathbf{x}^{(i)}), t^{(i)}) \right) + \underbrace{\lambda \Omega(\theta)}_{\text{terme de régularisation}}$$

- On initialise les paramètres aléatoirement
- On les mets à jour itérativement en suivant le gradient

Soit **descente de gradient batch (par lot)**:

BOUCLE:  $\theta \leftarrow \theta - \eta \frac{\partial \hat{R}_\lambda}{\partial \theta}$

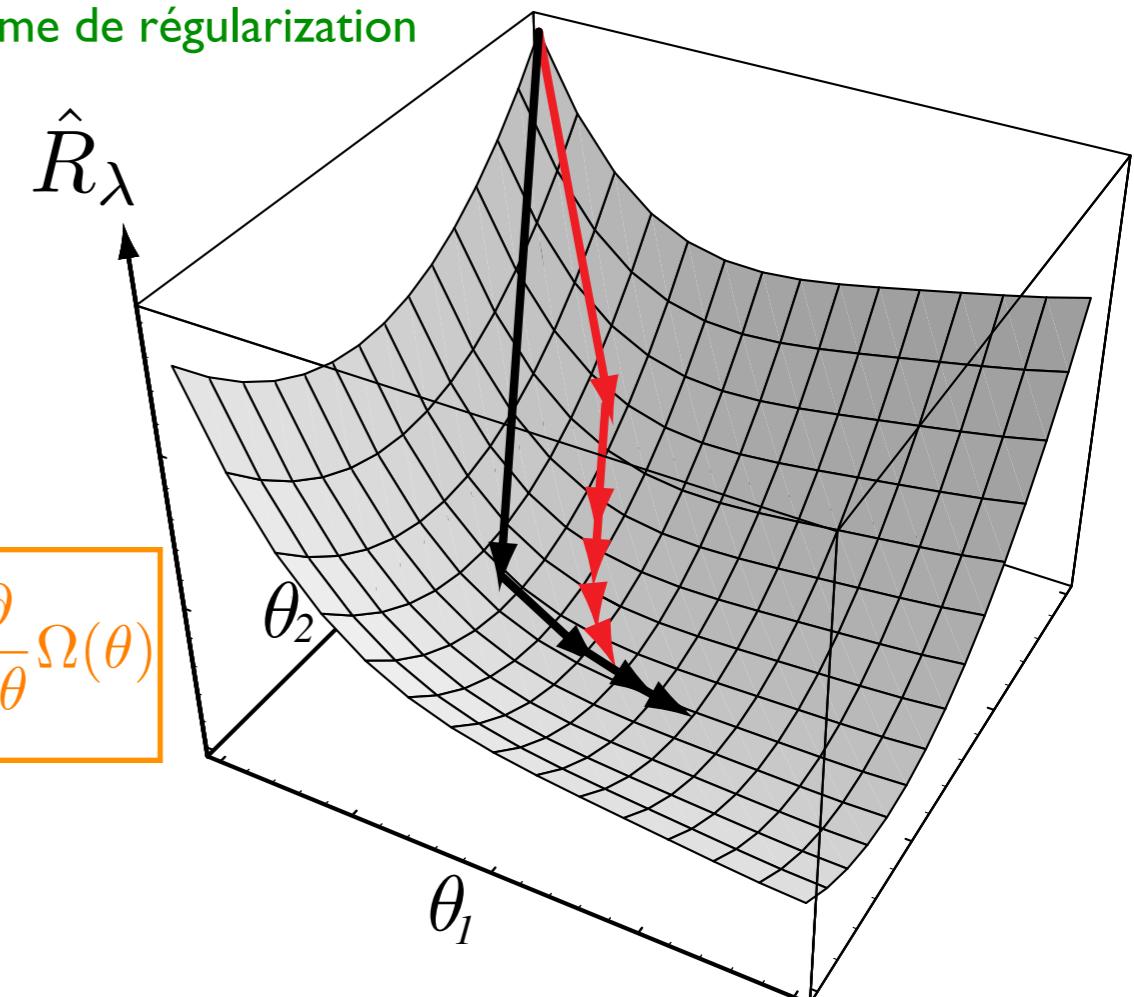
$$= \boxed{\left( \sum_{i=1}^n \frac{\partial}{\partial \theta} L(f_\theta(\mathbf{x}^{(i)}), t^{(i)}) \right) + \lambda \frac{\partial}{\partial \theta} \Omega(\theta)}$$

Ou **descente de gradient stochastique**:

BOUCLE:  
    Choisir i dans 1...n

$$\theta \leftarrow \theta - \eta \frac{\partial}{\partial \theta} \left( L(f_\theta(\mathbf{x}^{(i)}), t^{(i)}) + \frac{\lambda}{n} \Omega(\theta) \right)$$

Ou **d'autres méthodes de descente de gradient**  
(gradient conjugué, Newton, gradient naturel, ...)



# Limitations de la Régression Logistique: ça demeure un classifieur linéaire!

On décide classe 1 si  $P(t = 1|x) > 0.5$  (classe 0 sinon).

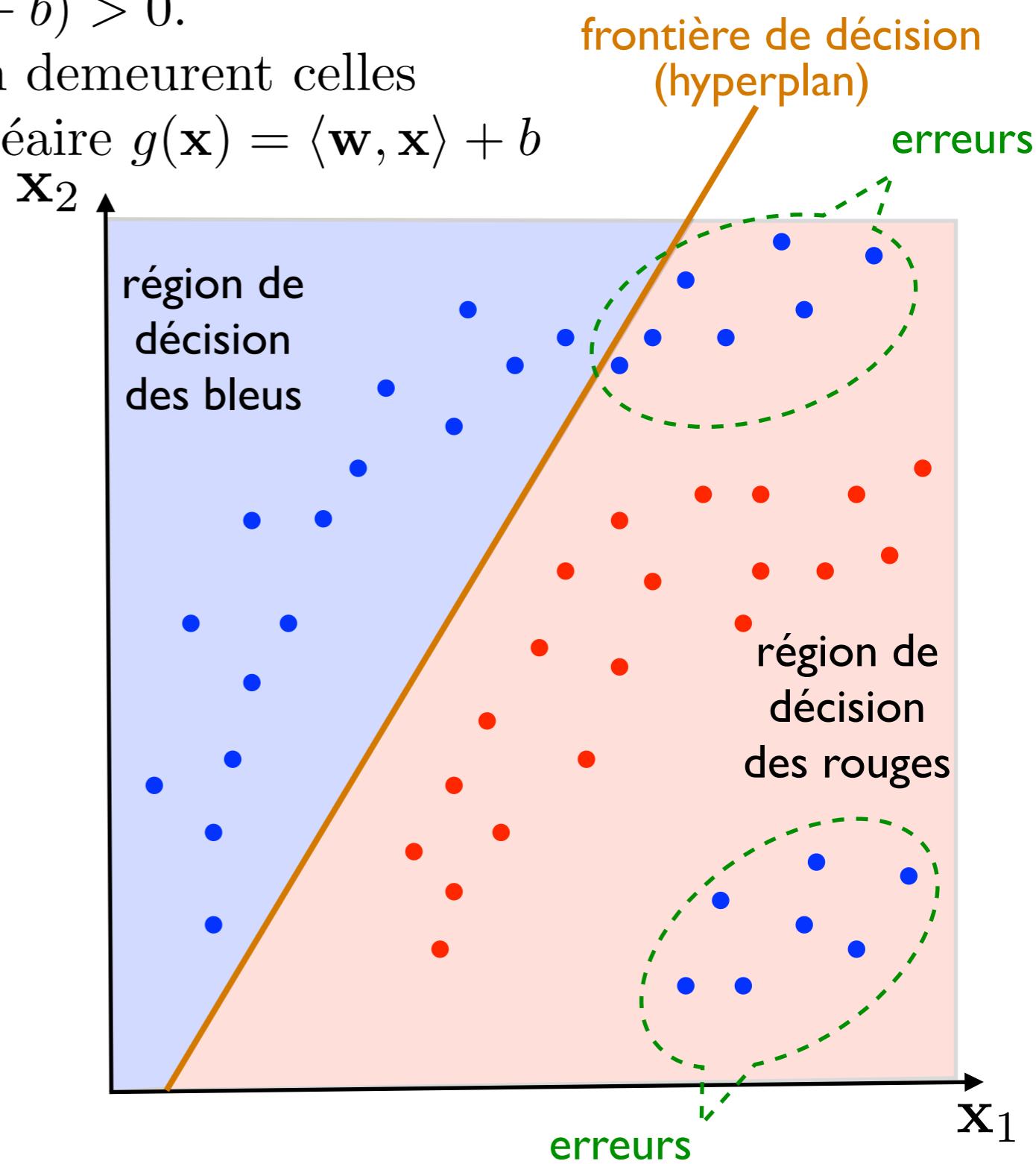
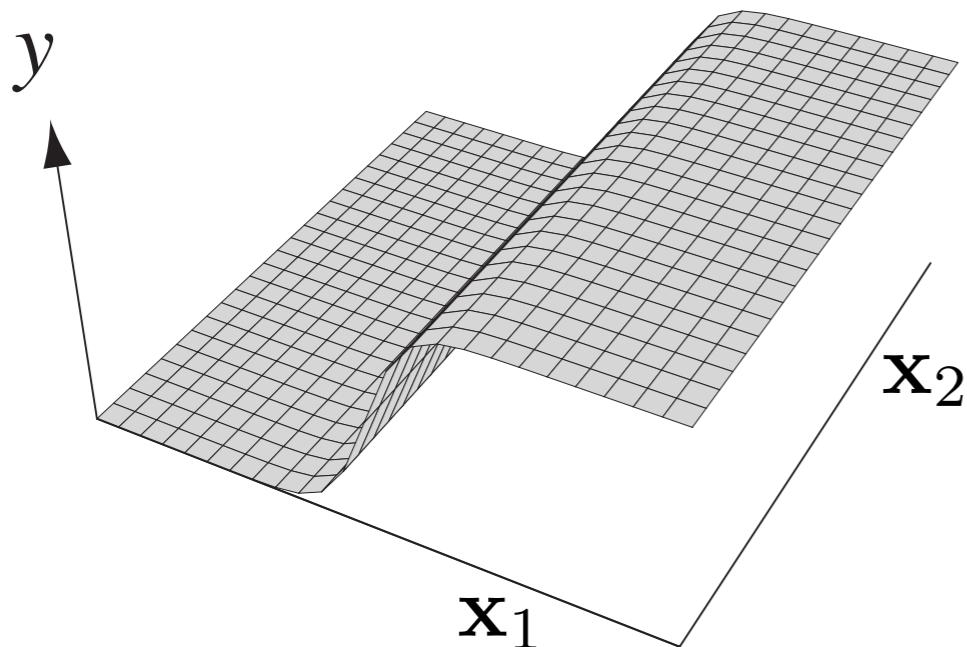
Or  $\text{sigmoid}(\langle \mathbf{w}, \mathbf{x} \rangle + b) > 0.5 \equiv \langle \mathbf{w}, \mathbf{x} \rangle + b > 0$ .

Donc les régions et frontière de décision demeurent celles associées à la fonction discriminante linéaire  $g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$

La frontière de décision est un hyperplan.

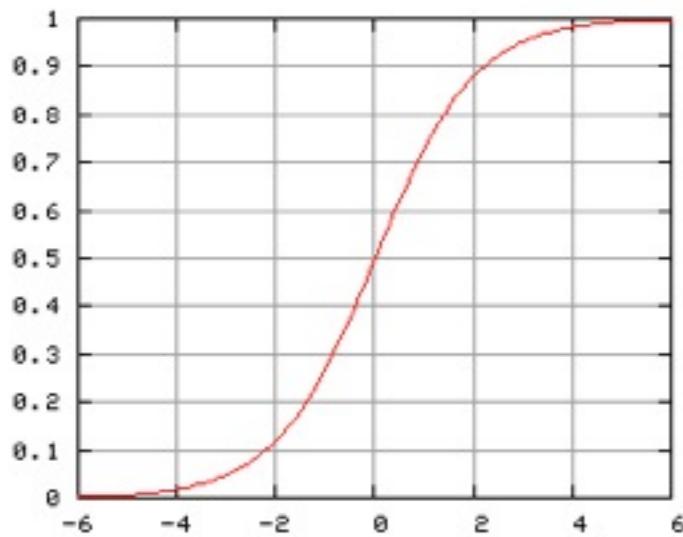
→ inappropriate si les classes ne sont pas linéairement séparables

(ex: voir figure)



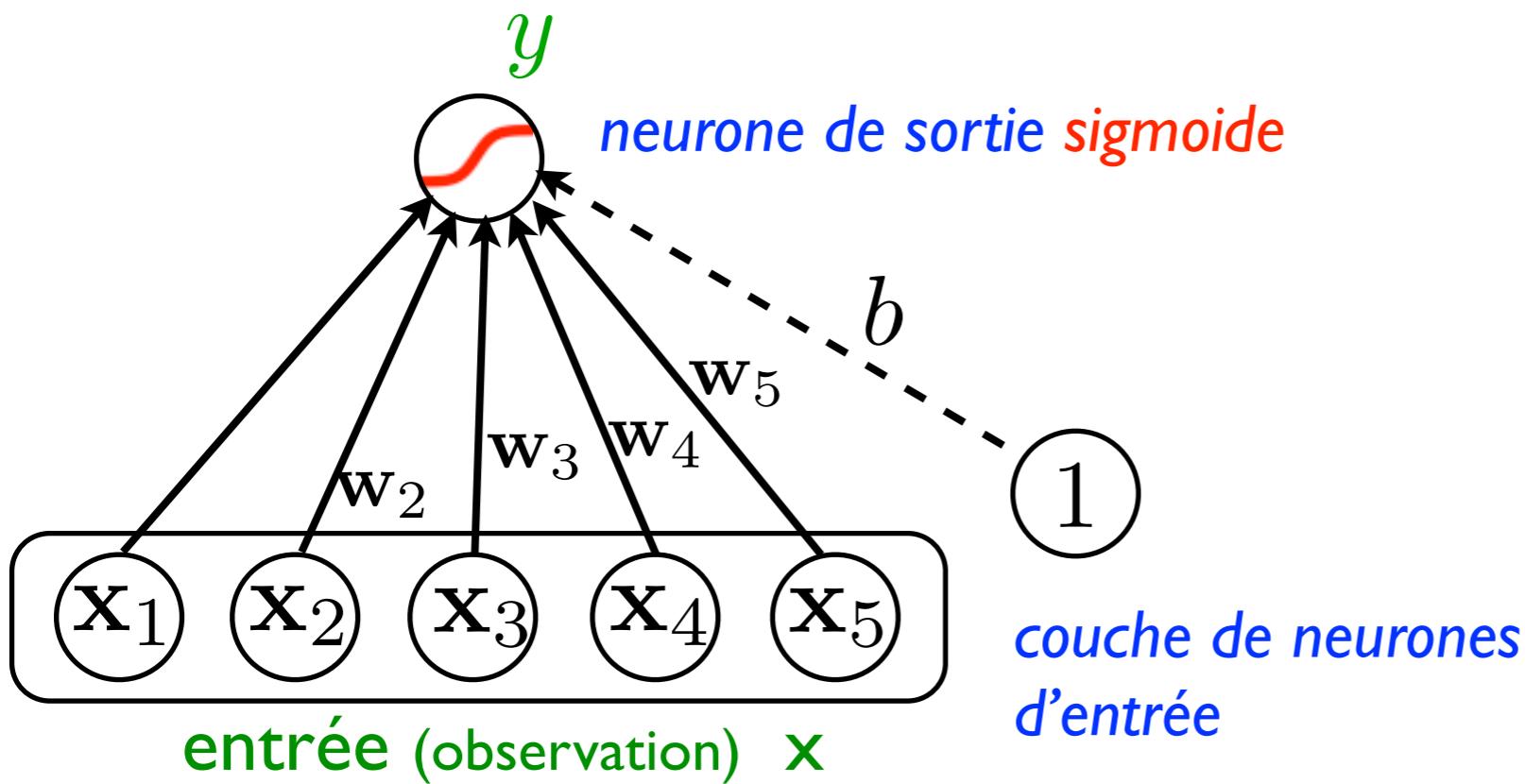
# Régression logistique

## Vision neuronale



La sigmoïde peut être vue comme

- Une alternative différentiable «douce» à la fonction escalier du Perceptron (Rosenblatt 1957). 
- Une approximation de la réponse en “taux d’impulsion” dans les neurones biologiques.



Ex 6:

# Les Machines à Vecteurs de Support

Support Vector Machine

(Vapnik & Lerner 1963, Cortes & Vapnik 1995)

- À la base: un autre algorithme qui apprend une simple fonction discriminante linéaire  $g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$
- Choisit un hyperplan séparateur particulier, motivation géométrique: **marge maximale**.
- Version pour construire des classifieurs non-linéaires utilise l'**astuce du noyau**.
- Algorithme très populaire (dans sa forme linéaire et dans sa forme non-linéaire à noyau).

# SVM: motivation géométrique

Données:

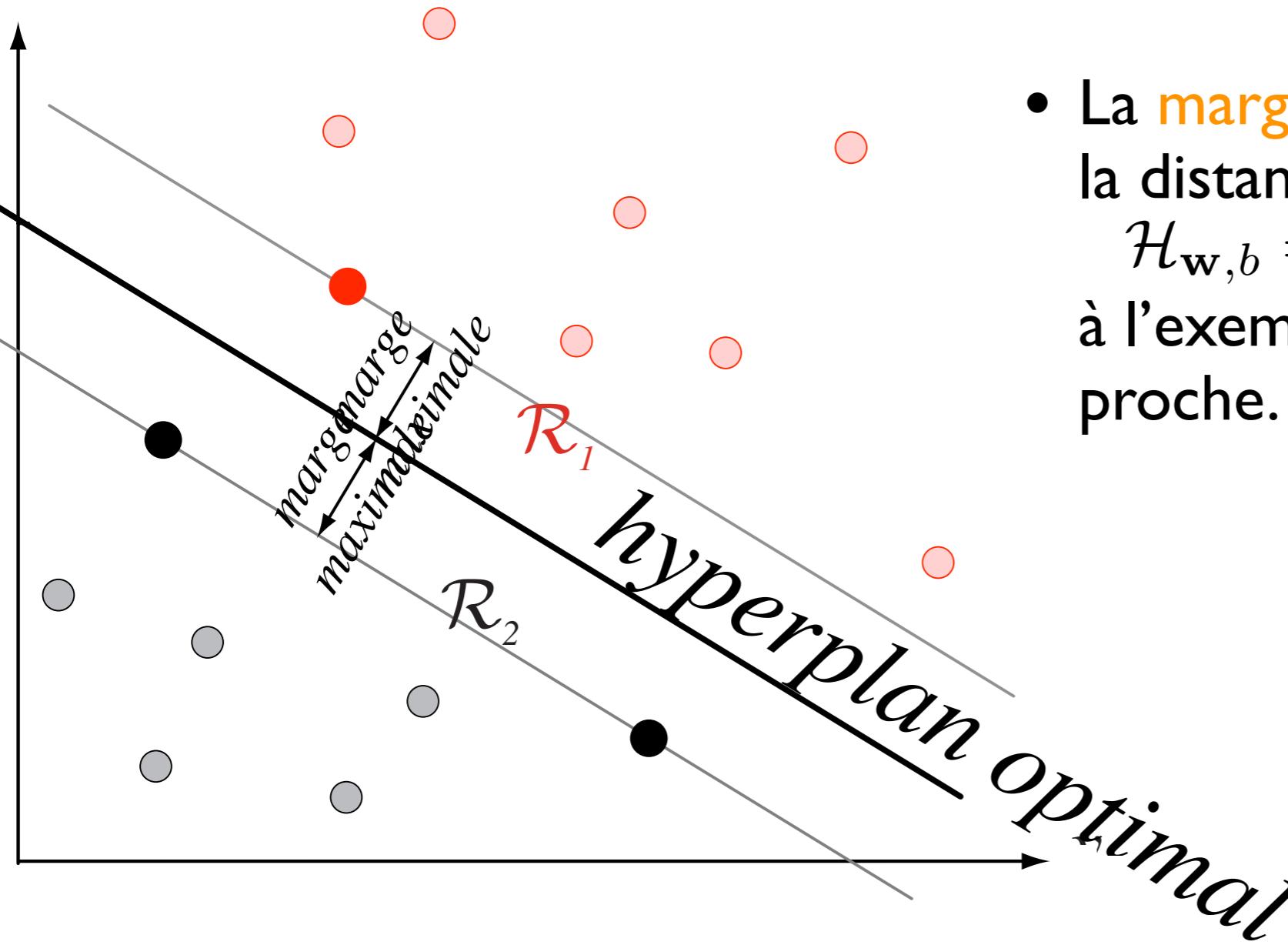
$$D_n = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$$

$$\mathbf{x}_i \in \mathbb{R}^d, y_i \in \{-1, +1\}$$

- Algorithme qui apprend une **fonction discriminante linéaire**  $g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$  à **marge maximale**.
- La **marge** est la distance de l'hyperplan  $\mathcal{H}_{\mathbf{w}, b} = \{\mathbf{x} | \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}$  à l'exemple d'entraînement le plus proche.

La distance d'un point  $\mathbf{x}$  à l'hyperplan est la distance à sa projection orthogonale sur l'hyperplan:

$$\left| \frac{\langle \mathbf{w}, \mathbf{x} \rangle + b}{\|\mathbf{w}\|} \right|$$



# SVM à marge dure

La distance d'un point  $\mathbf{x}$  à l'hyperplan:

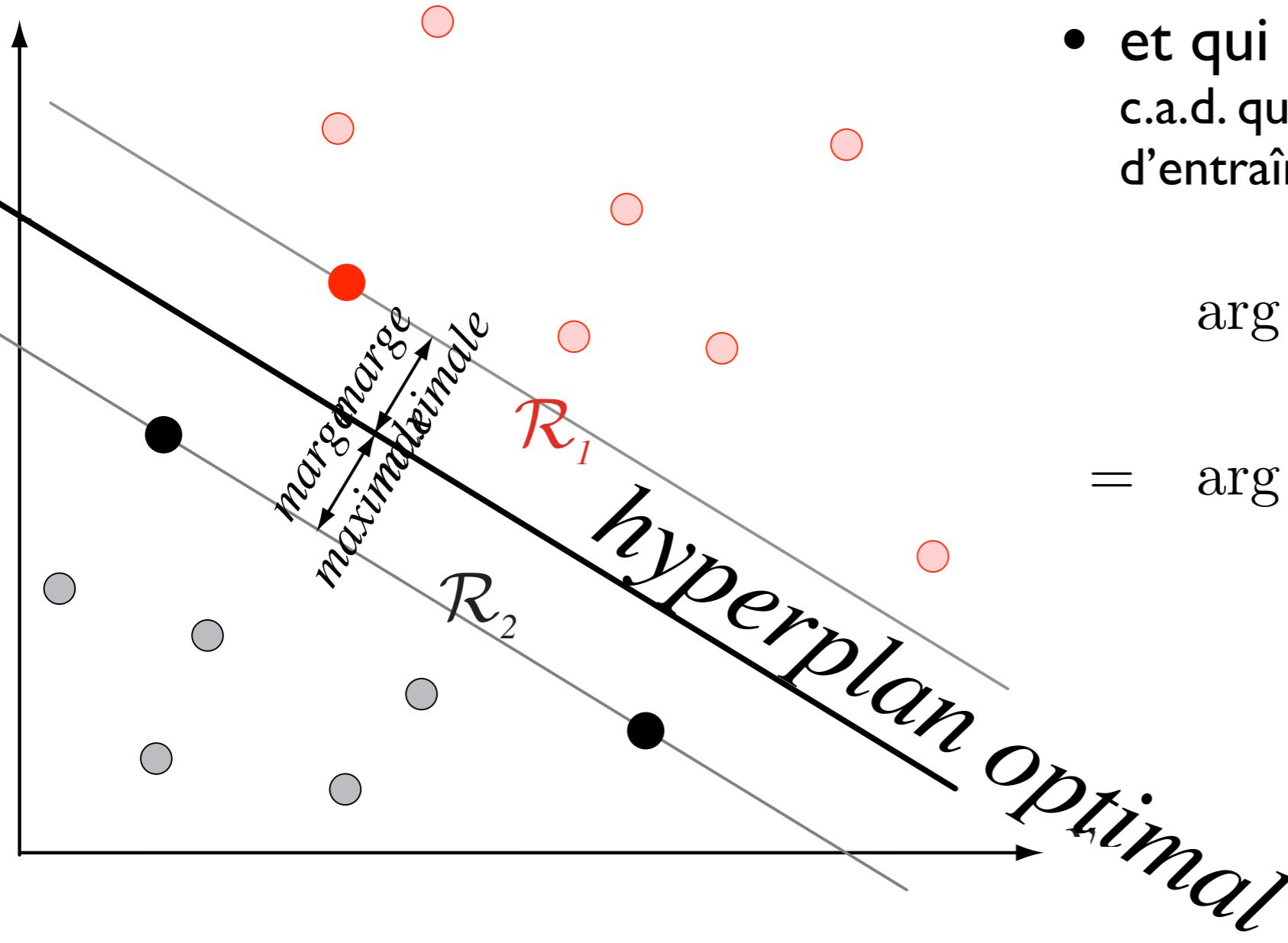
$$d(\mathbf{x}, \mathcal{H}_{\mathbf{w}, b}) = \left| \frac{\langle \mathbf{w}, \mathbf{x} \rangle + b}{\|\mathbf{w}\|} \right|$$

Trouver l'hyperplan «optimal» qui:

- sépare parfaitement les données  
 $\forall i \in \{1, \dots, n\}, (\langle \mathbf{w}, \mathbf{x}_i \rangle + b)y_i > 0$
- et qui a la **plus grande marge**  
c.a.d. qui est le plus loin du point d'entraînement le plus proche

$$\arg \max_{\mathbf{w}, b} \left\{ \min_i d(\mathbf{x}_i, \mathcal{H}_{\mathbf{w}, b}) \right\}$$

$$= \arg \max_{\mathbf{w}, b} \left\{ \min_i \frac{\langle \mathbf{w}, \mathbf{x}_i \rangle + b}{\|\mathbf{w}\|} y_i \right\}$$



# Marge = confiance

- Définition: marge géométrique d'un exemple  $(\mathbf{x}, y)$  est sa distance signée à l'hyperplan (signe indique si classification correcte):

$$\gamma^{(g)}(\mathbf{x}, y) = \frac{\langle \mathbf{w}, \mathbf{x} \rangle + b}{\|\mathbf{w}\|} y$$

- Remarque: la norme de  $\mathbf{w}$  ne change rien à l'hyperplan
- Définition: marge fonctionnelle d'un exemple  $(\mathbf{x}, y)$

$$\gamma(\mathbf{x}, y) = g(\mathbf{x})y = (\langle \mathbf{w}, \mathbf{x} \rangle + b)y$$

- Le signe de la marge indique si classification correcte ou erreur
- La magnitude (valeur absolue) de la marge indique la «confiance» du classifieur dans sa prédiction.
- Un exemple avec une large marge positive est classifié correctement avec une grande confiance.
- Un point avec une marge négative de forte magnitude est mal classifié avec une grande confiance.

# SVM à marge dure

Problèmes d'optimisation équivalents  
(avec des données linéairement séparables)

- Maximiser la plus petite marge géométrique

$$\arg \max_{\mathbf{w}, b} \left\{ \min_i \frac{\langle \mathbf{w}, \mathbf{x}_i \rangle + b}{\|\mathbf{w}\|} y_i \right\}$$

- Maximiser la plus petite marge fonctionnelle, sous contrainte  $\|\mathbf{w}\|=1$

$$\arg \max_{\mathbf{w}, b} \left\{ \min_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) y_i \right\}, \text{ s.c. } \|\mathbf{w}\|^2 = 1$$

- Minimiser  $\|\mathbf{w}\|$  sous contrainte de marge fonctionnelle  $\geq 1$

$$\begin{aligned} & \arg \min_{\mathbf{w}} \|\mathbf{w}\|^2 \\ \text{s.c. } & \forall i \in \{1, \dots, n\} (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) y_i \geq 1 \end{aligned}$$

- Dans ces trois cas, on obtient la même solution, à la norme de  $\mathbf{w}$  près (qui ne change rien à l'hyperplan)
- La dernière formulation ci-dessus correspond à un problème de **Programmation Quadratique (QP) avec contraintes** (pour SVM à marge dure).

# SVM à marge souple

- Relaxation des contraintes pour tolérer des erreurs sur Dn

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

avec les contraintes  $\forall i \in \{1, \dots, n\}$

$$(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) y_i \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

- Les exemples en deça de la marge auront un  $\xi_i > 0$  ( $=0$  pour les autres)
- => Programmation Quadratique (QP) avec contraintes.
- **C est un hyper-paramètre** (scalaire positif) qui contrôle la tolérance aux erreurs.
- $C \rightarrow +\infty$  correspond à SVM à marge dure (aucune erreur tolérée, Dn doit être linéairement séparable)

# SVM à marge souple

## formulation sans contrainte

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \quad \text{s.c. } \xi_i \geq 0 \text{ et } (\langle \mathbf{w}, \mathbf{x}_i \rangle + b)y_i \geq 1 - \xi_i$$

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \quad \text{s.c. } \xi_i \geq 0 \text{ et } \xi_i \geq 1 - (\langle \mathbf{w}, \mathbf{x}_i \rangle + b)y_i$$

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \quad \text{s.c. } \xi_i \geq \max(0, 1 - (\langle \mathbf{w}, \mathbf{x}_i \rangle + b)y_i)$$

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \quad \text{s.c. } \xi_i = \max(0, 1 - (\langle \mathbf{w}, \mathbf{x}_i \rangle + b)y_i)$$

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \max(0, 1 - (\langle \mathbf{w}, \mathbf{x}_i \rangle + b)y_i)$$

**hinge loss:**  $\max(0, 1 - m)$  où  $m$  est la marge fonctionnelle

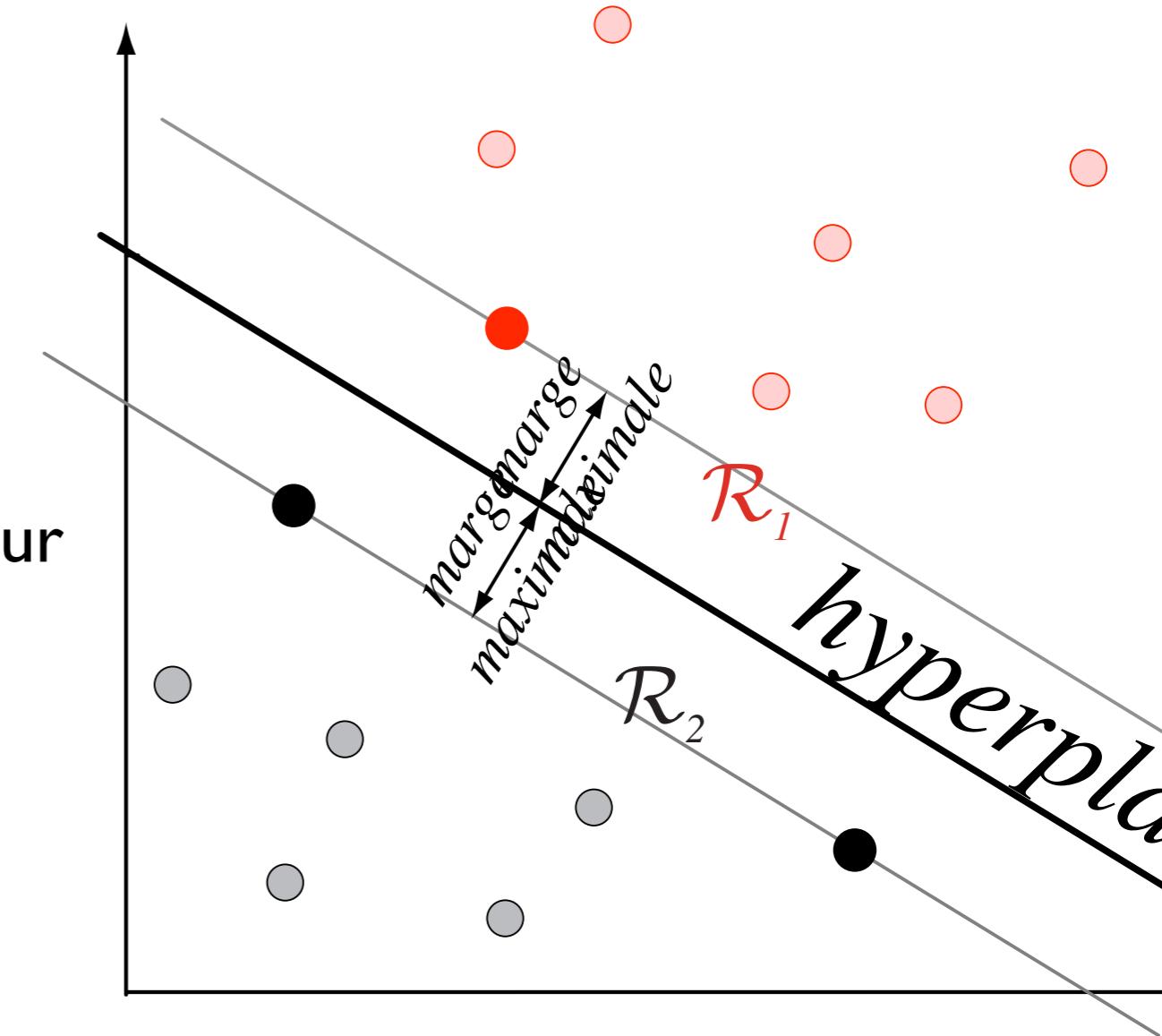
# SVM: forme de la solution

- La solution du problème de minimisation donne un  $w$  qui a la forme suivante:

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

avec  $\alpha_i \geq 0$

- Typiquement seule une petite fraction des exemples d'apprentissage auront leur  $\alpha_i > 0$ .
- La solution est donc exprimée en fonction de ces quelques exemples d'entraînement.
- On nomme ces points les **vecteurs de support**.
- Ils se situent sur la marge (ou en deçà si on tolère des erreurs, marge souple).



Cela permet une *formulation duale* de l'optimisation:  
où on apprend les  $\alpha_i$  plutôt que les  $w_j$

# SVM non linéaire et multiclassse

- Pour obtenir un SVM non-linéaire on applique l'astuce du noyau (voir deuxième partie)
- Pour obtenir un classifieur à m classes, on entraîne plusieurs SVM de classification binaire (typiquement m classifieurs «un contre tous»).
- Il y a de nombreuses autres extensions de SVM, notamment pour la régression.

# Machine à Vecteurs de Support (SVM) En résumé:

- Idée géométrique mène à un problème d'optimisation quadratique avec contraintes.
- Ceci donne l'algorithme “SVM à marge dure” et qui nécessite des données linéairement séparables.
- Une extension dite “à marge souple” permet de trouver une solution dans le cas non séparable.
- De plus l'astuce du noyau peut être appliquée pour donner un classifieur non-linéaire.
- Les SVMs à marge souple (à noyau ou non) sont un algorithme très très populaire (bonne performance, peu d'hyper-paramètres).

# Nombreuses techniques pour apprendre une fonction discriminante linéaire

- Algorithme du plus proche centroïde
- Classifieur de Bayes avec des densités Gaussiennes (avec contrainte de covariance identique pour les deux classes)
- Régression linéaire (ou ridge) avec des cibles  $-1 + 1$
- Perceptron
- Régression logistique (possiblement régularisée comme ridge)
- etc...
  - (Fisher) Linear Discriminant Analysis (LDA)
  - Machine à Vecteurs de Support (linéaire)

Toutes ces méthodes vont donner une frontière de décision différente mais il s'agira toujours d'un hyperplan! ( $\Rightarrow$  capacité limitée)