

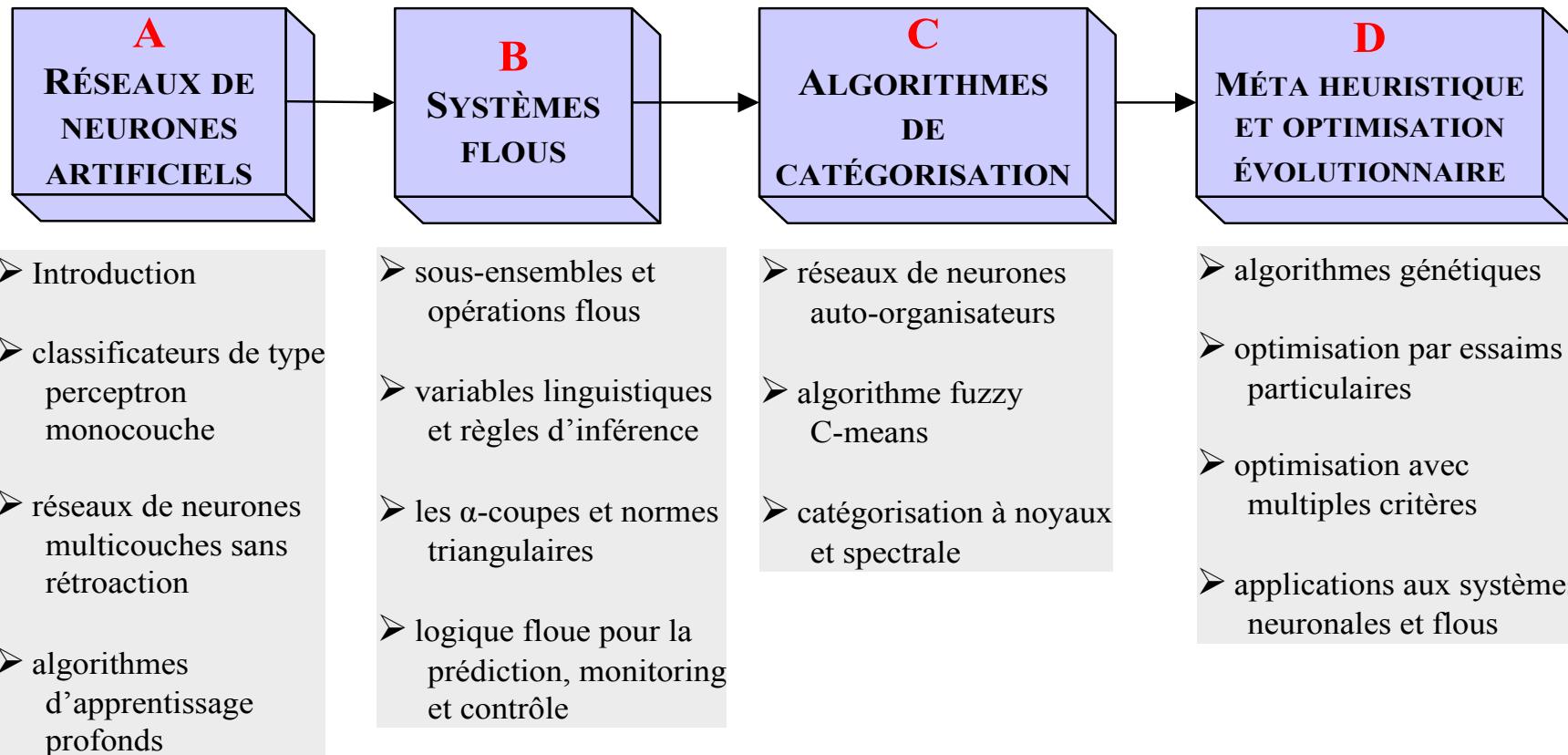
SYS843

A.2 Classificateurs de type perceptron monocouche

**Eric Granger
Ismail Ben Ayed**

Hiver 2016

Contenu du cours



Contenu du cours

A.2 Classificateur du type perceptron monocouche

- 1) Modèles de classification, régions de décision
- 2) Fonctions discriminantes
- 3) Machine linéaire et classification à distance minimum
- 4) Le concept de l'apprentissage non paramétrique
- 5) Perceptron (discret, continue)
- 6) Réseau perceptron monocouche (discret, continue)

1) Modèles de classification

Définitions

- ▶ **Forme (Pattern):** Description quantitative d'un objet, évènement ou phénomène:
 - Spatiales (ordonnées dans l'espace): images, vidéos
 - Temporelles (ordonnées dans le temps): signaux de paroles, électrocardiogrammes
- ▶ **Classification:** Assignation d'une forme à une classe définie *a priori* (par apprentissage). La **reconnaissance** est une classification de formes pas utilisées pour l'apprentissage.

[Zeruda 82]

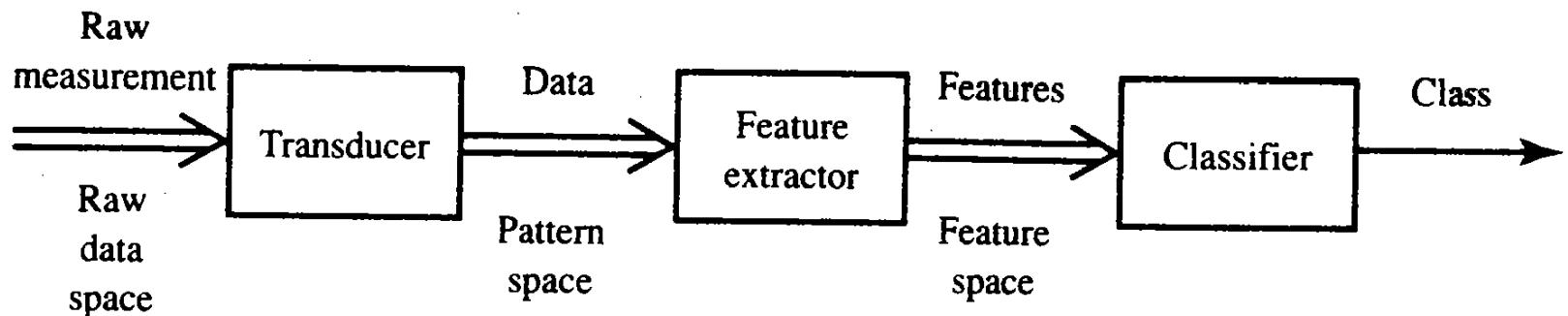
1) Modèles de classification

Définitions

► **Caractéristiques (*features*):** Mesures effectuées sur la forme

- Réduire la dimension sans affecter la classification
- Augmenter la dimension pour améliorer la classification
(on va voir des exemples en fin de cours)

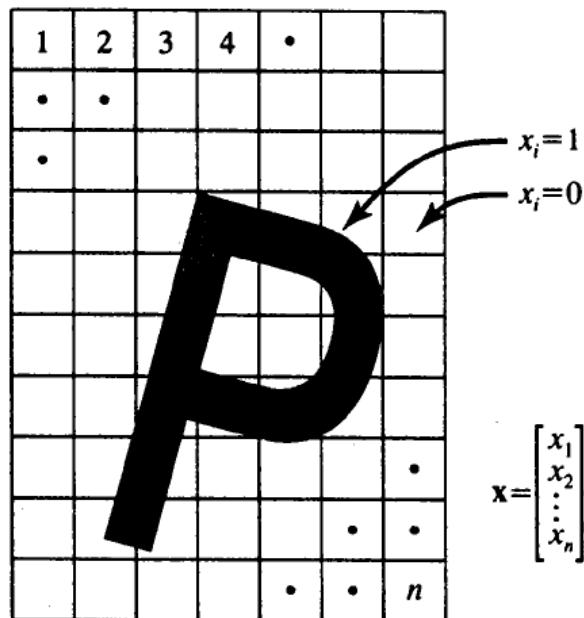
Diagramme général des systèmes



1) Modèles de classification

► Exemples de caractéristiques (Vecteur x)

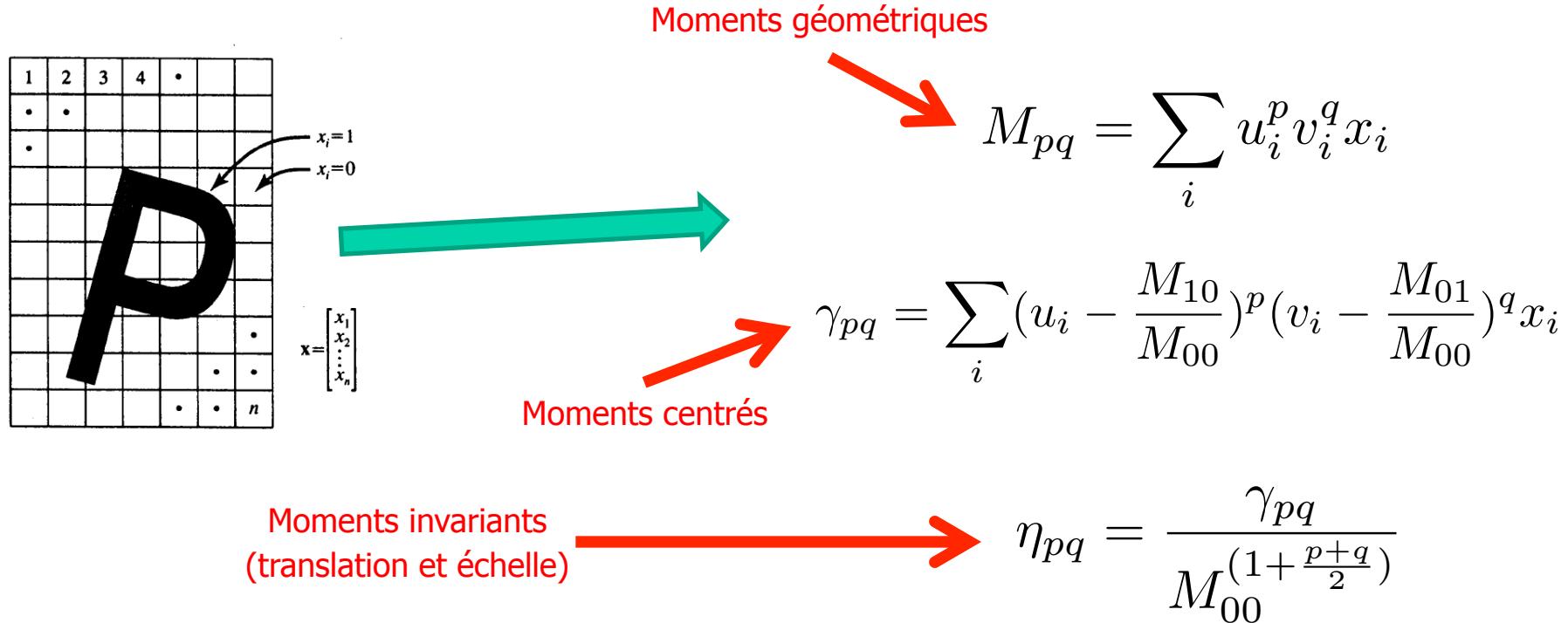
- Forme numérique des données brutes (ex., image binaire)



1) Modèles de classification

► Exemples de caractéristiques (Vecteur x)

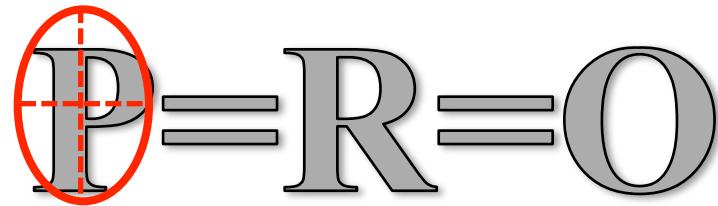
- Transformation des données brutes (ex., moments invariants)



1) Modèles de classification

► Exemples de caractéristiques (Vecteur x)

- Transformation des données brutes (ex., moments invariants)
- Importance de la dimension:
 - Réduire trop la dimension rend la séparation des données difficile, ex., moments d'ordre 2 (information d'élongation)

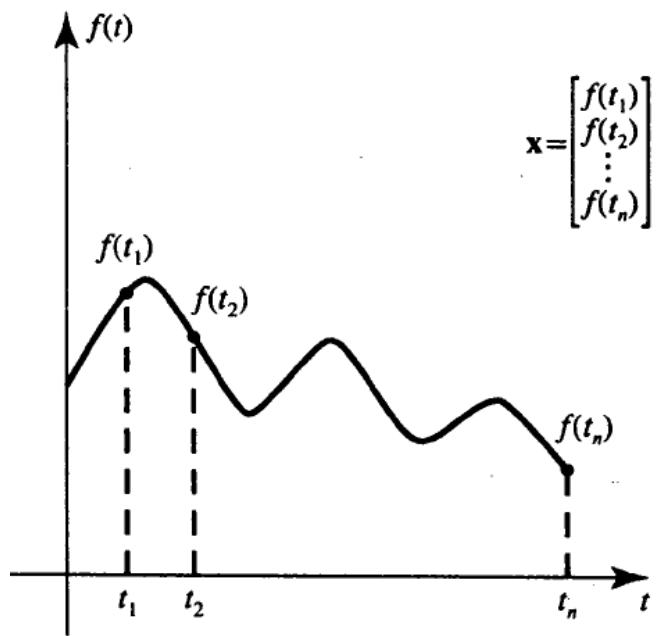


- Larges dimensions peuvent rendre le problème difficile de point de vue computationnel

1) Modèles de classification

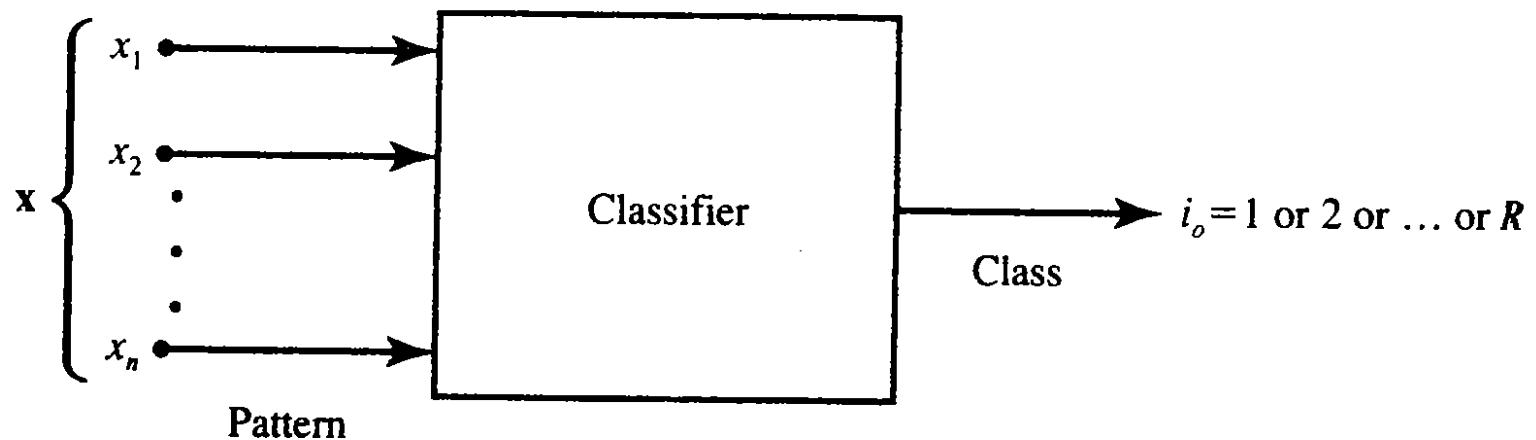
► Exemples de caractéristiques (Vecteur x)

- Transformation des données brutes (ex., échantillonnage d'un signal temporel)



1) Modèles de classification

- ▶ Notations dans ce qui suit:



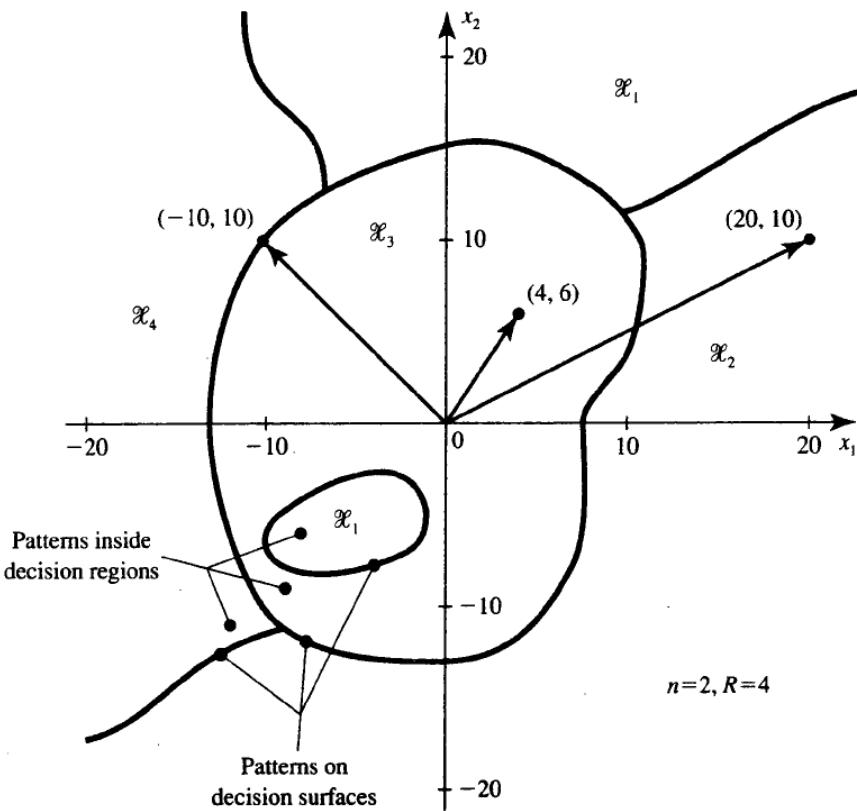
1) Modèles de classification

► Notations dans ce qui suit:

- L'entrée du classificateur sera représentée par le vecteur $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$
- Classification de \mathbf{x} : Fonction de décision $i_o(\mathbf{x})$
- $i_o = 1, 2, \dots, R$: Classes définies *a priori*
- La fonction de classification (décision) $i_o = i_o(\mathbf{x})$: Transformation ('mapping') du vecteur \mathbf{x} dans l'une des classes

1) Modèles de classification

► Interprétation géométrique



2) Fonctions discriminantes

► Dans ce qui suit, on suppose:

- Observations x_1, x_2, \dots, x_p et leur classe sont connues.
- P (nombre de formes) est nettement plus grand que R (nombre de classes)
- Le classificateur a déjà été entraîné.
- **Fonctions discriminantes:** Permettra de comprendre comment les réseaux de neurones doivent être entraînés.

2) Fonctions discriminantes

- Considérons R fonctions discriminantes:

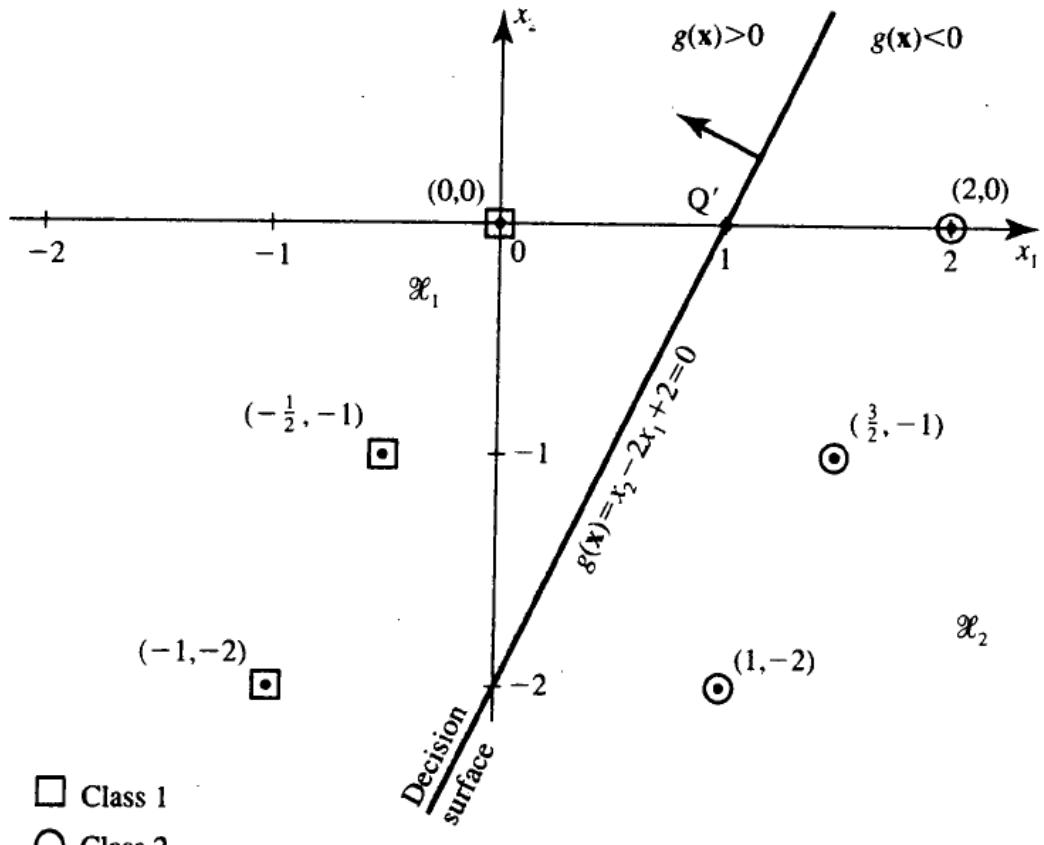
$$g_1(\mathbf{x}), g_2(\mathbf{x}), \dots, g_R(\mathbf{x})$$

- L'observation inconnue \mathbf{x} appartient à la classe i ssi:
$$g_i(\mathbf{x}) > g_j(\mathbf{x}), \text{ pour } i, j = 1, 2, \dots, R, \text{ et } i \neq j$$
- Équation de la surface de décision entre deux régions contiguës:

$$g_i(\mathbf{x}) - g_j(\mathbf{x}) = 0$$

2) Fonctions discriminantes

- Un exemple simple ($n=2, R=2$):

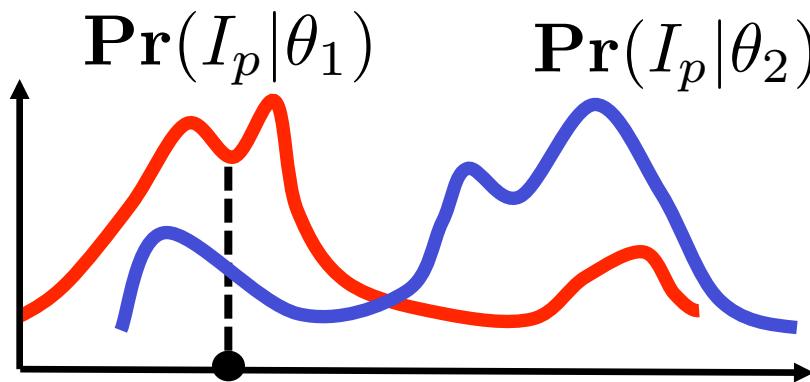


$$g_1(\mathbf{x}) = \begin{bmatrix} -1 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 2$$

$$g_2(\mathbf{x}) = \begin{bmatrix} 1 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

2) Fonctions discriminantes

- Exemple réel (segmentation d'images)



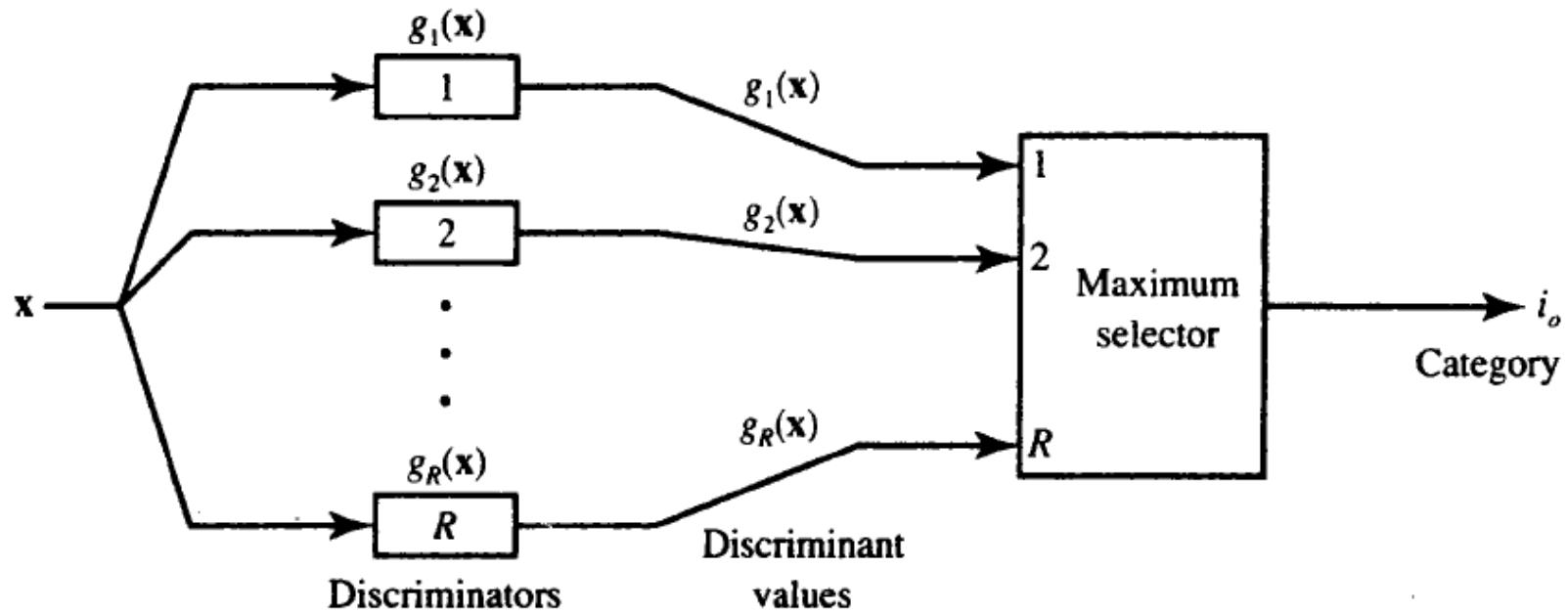
$$\log(\Pr(I_p | \theta_1)) > \log(\Pr(I_p | \theta_2))$$

$\downarrow g_1$ $\downarrow g_2$

Estimation Bayésienne
(Maximum de vraisemblance)

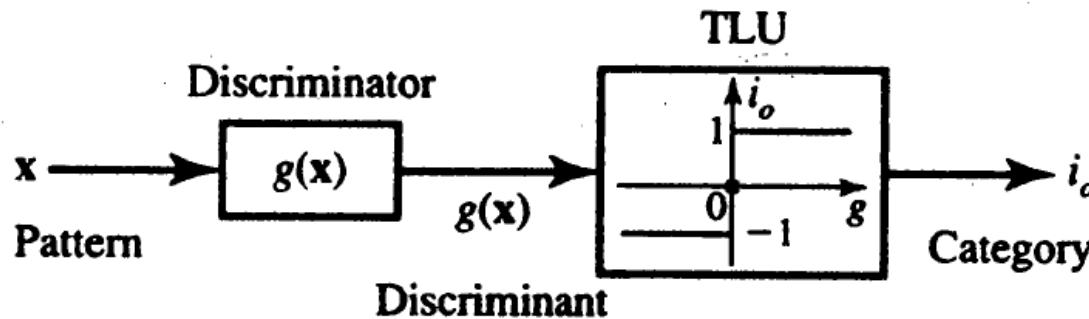
2) Fonctions discriminantes

- Architecture: Cas du problème de classification à R classes:



2) Fonctions discriminantes

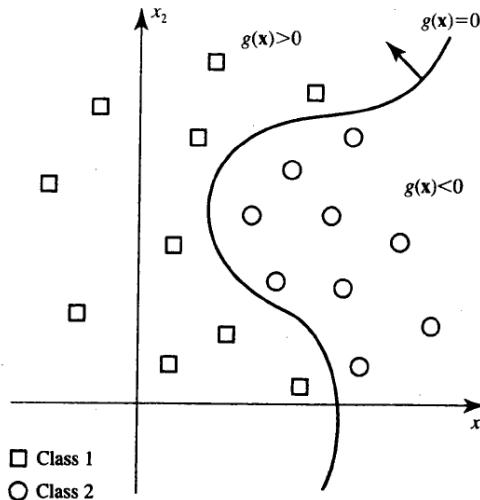
- Architecture simplifiée (Cas $R=2$)



$$i_o = \operatorname{sgn} [g(\mathbf{x})] = \begin{cases} -1 & \text{pour } g(\mathbf{x}) < 0 \text{ (classe2)} \\ \text{indéfinie} & \text{pour } g(\mathbf{x}) = 0 \\ 1 & \text{pour } g(\mathbf{x}) > 0 \text{ (classe1)} \end{cases}$$

2) Fonctions discriminantes

- Nous verrons que les réseaux de neurones peuvent implanter des fonctions de décision **non-linéaires**.



- Fonctions discriminantes (linéaire ou non-linéaire) étant choisies:
Conception d'un classificateur: évaluer par **apprentissage** les coefficients des fonctions discriminantes à partir des formes (exemples) ayant des classes connues.

3) Machine linéaire

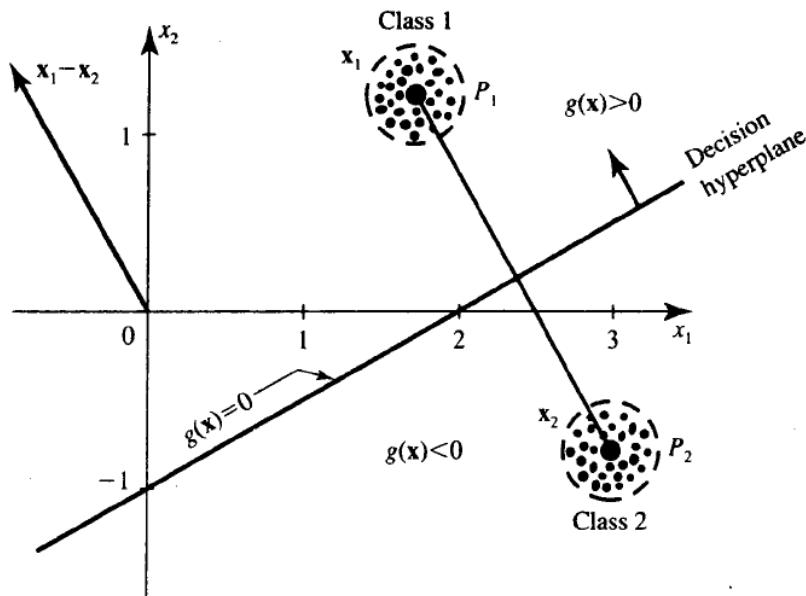
► Classification à distance minimum:

■ Pour le moment, supposons:

- L'ensemble d'entraînement et la classe d'appartenance de tous ses éléments sont connus: **l'apprentissage est supervisé**
- **Les fonctions discriminantes sont linéaires** et seulement leurs coefficients sont ajustés par la procédure d'entraînement.

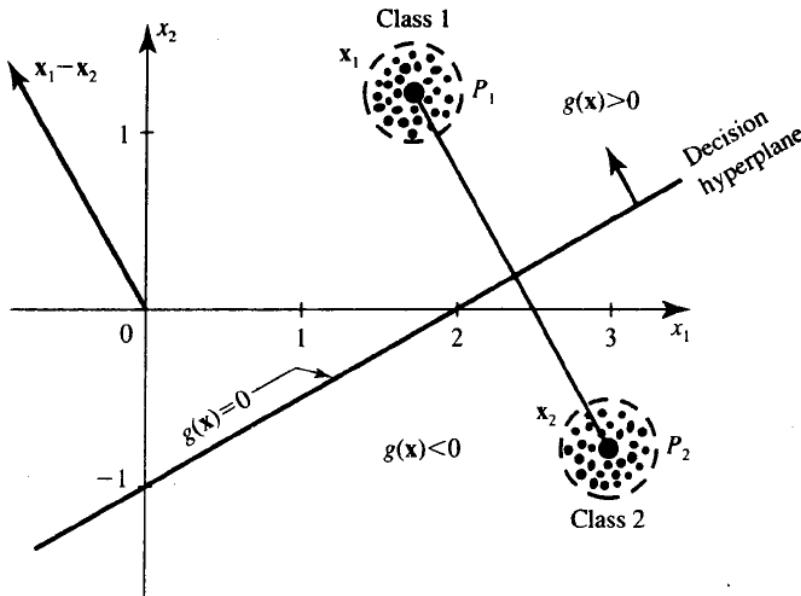
3) Machine linéaire

- Posons $R = 2$ classes, et les vecteurs x définis dans un espace Euclidien. Pour ce qui est de la classification linéaire, la surface de décision est un hyperplan.



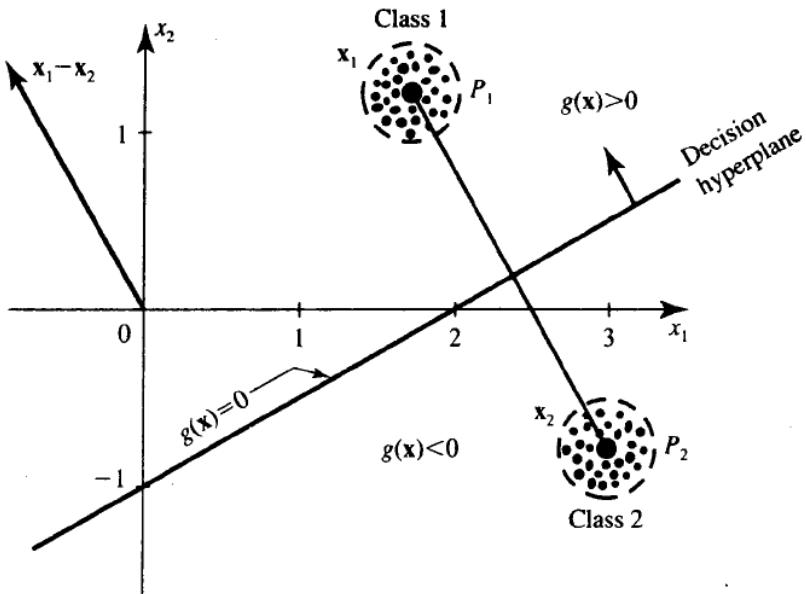
3) Machine linéaire

- 2 nuages de points appartenant aux classes 1 et 2: Considérons les centres des nuages aussi bien que l'hyperplan qui passe par le point milieu du segment entre ces deux points.



3) Machine linéaire

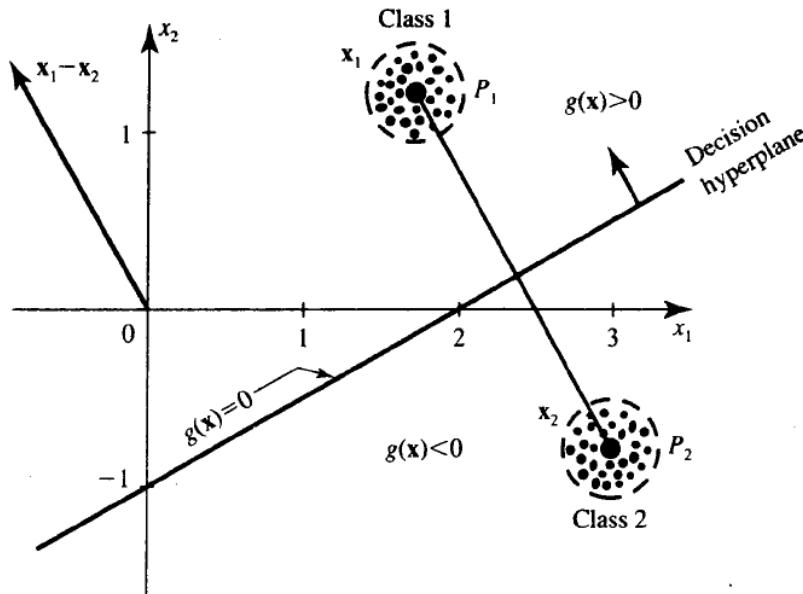
- Cet hyperplan est donc perpendiculaire au vecteur $x_1 - x_2$



3) Machine linéaire

- Connaissant un vecteur perpendiculaire à cet hyperplan et un point appartenant à ce plan (le point du milieu), on obtient l'équation de l'hyperplan:

$$g(\mathbf{x}) = (\mathbf{x}_1 - \mathbf{x}_2)^t \mathbf{x} + \frac{1}{2} (\|\mathbf{x}_2\|^2 - \|\mathbf{x}_1\|^2) = 0$$



3) Machine linéaire

- L'équation de l'hyperplan précédente peut être écrite sous la forme:

$$w_1x_1 + w_2x_2 + \dots + w_nx_n + w_{n+1} = 0$$



$$w^t x + w_{n+1} = 0$$



$$\begin{bmatrix} w \\ w_{n+1} \end{bmatrix}^t \begin{bmatrix} x \\ 1 \end{bmatrix} = 0$$

3) Machine linéaire

- Le vecteur de poids est défini par: $w = x_1 - x_2$

$$w_1x_1 + w_2x_2 + \dots + w_nx_n + w_{n+1} = 0$$

$w \doteq \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ \vdots \\ w_n \end{bmatrix}$

$w_{n+1} = \frac{1}{2} (\|x_2\|^2 - \|x_1\|^2)$

3) Machine linéaire

- Classificateur à distance minimum pour R classes:
 - Nous avons besoins de $R(R-1)/2$ fonctions discriminantes (une pour chaque paire de classes)
 - C'est possible que pour une paire de classes, les deux régions \mathfrak{R}_i et \mathfrak{R}_j ne sont pas contiguës
 - Dans ce cas, $g_i(\mathbf{x}) - g_j(\mathbf{x}) = 0$ (l'équation de la surface de décision) n'a pas de solution

3) Machine linéaire

- Classificateur à distance minimum pour R classes:
 - Chaque classe est représentée par un prototype. Donc, on a un R vecteurs prototypes: $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_R$
 - La distance Euclidienne entre un point à classer et un prototype i est donnée par:

$$\|\mathbf{x} - \mathbf{x}_i\| = \sqrt{(\mathbf{x} - \mathbf{x}_i)^t(\mathbf{x} - \mathbf{x}_i)}$$

- Pour chaque point à classifier, le classificateur évalue la distance entre le point et tous les prototypes. Le point est attribué à la classe i correspondant au prototype le plus proche.

3) Machine linéaire

- On peut réécrire les distances point-prototype de la façon suivante:

$$\|x - x_i\|^2 = \cancel{x^t x} - 2x_i^t x + x_i^t x_i, \text{ pour } i = 1, 2, \dots, R$$

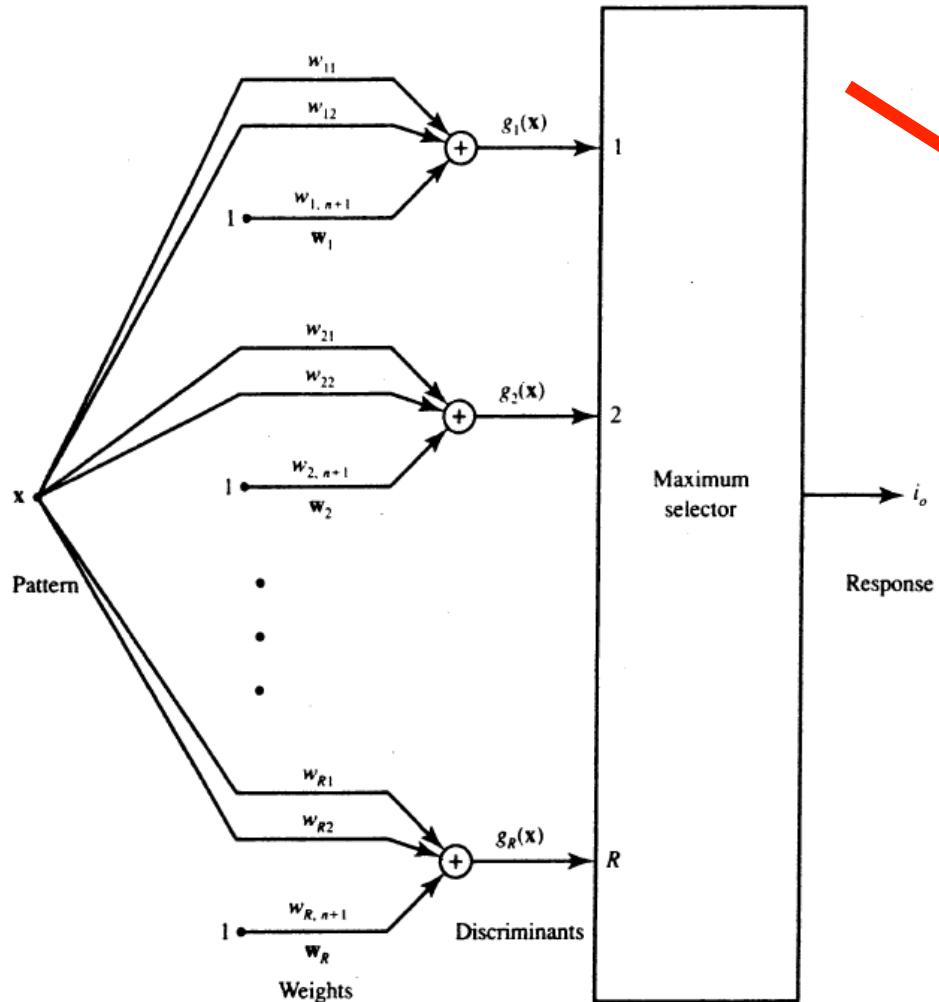
Indépendant de i

- Par conséquent:

$$g_i(x) = x_i^t x - \frac{1}{2}x_i^t x_i, \text{ pour } i = 1, 2, \dots, R$$

On attribue à l'observation la classe pour laquelle cette fonction est maximale

3) Machine linéaire



Aussi appelé
corrélateur

3) Machine linéaire

- Surface de décision entre deux régions contiguës:

$$\begin{aligned} g_i(\mathbf{x}) - g_j(\mathbf{x}) &= 0 \\ \downarrow & \\ w_i^t \mathbf{x} + w_{i,n+1} - w_j^t \mathbf{x} - w_{j,n+1} &= 0 \\ \text{---} & \\ g_i(\mathbf{y}) = w_i^t \mathbf{y} &\xrightarrow{\text{Vecteur augmenté}} \mathbf{y} \doteq \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \end{aligned}$$

3) Machine linéaire

- Un exemple simple dans le cas $R=3$ et $n=2$:

$$\mathbf{x}_1 = \begin{bmatrix} 10 \\ 2 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 2 \\ -5 \end{bmatrix}, \quad \mathbf{x}_3 = \begin{bmatrix} -5 \\ 5 \end{bmatrix}$$



$$\mathbf{w}_1 = \begin{bmatrix} 10 \\ 2 \\ -52 \end{bmatrix}, \quad \mathbf{w}_2 = \begin{bmatrix} 2 \\ -5 \\ -14.5 \end{bmatrix}, \quad \mathbf{w}_3 = \begin{bmatrix} -5 \\ 5 \\ -25 \end{bmatrix}$$



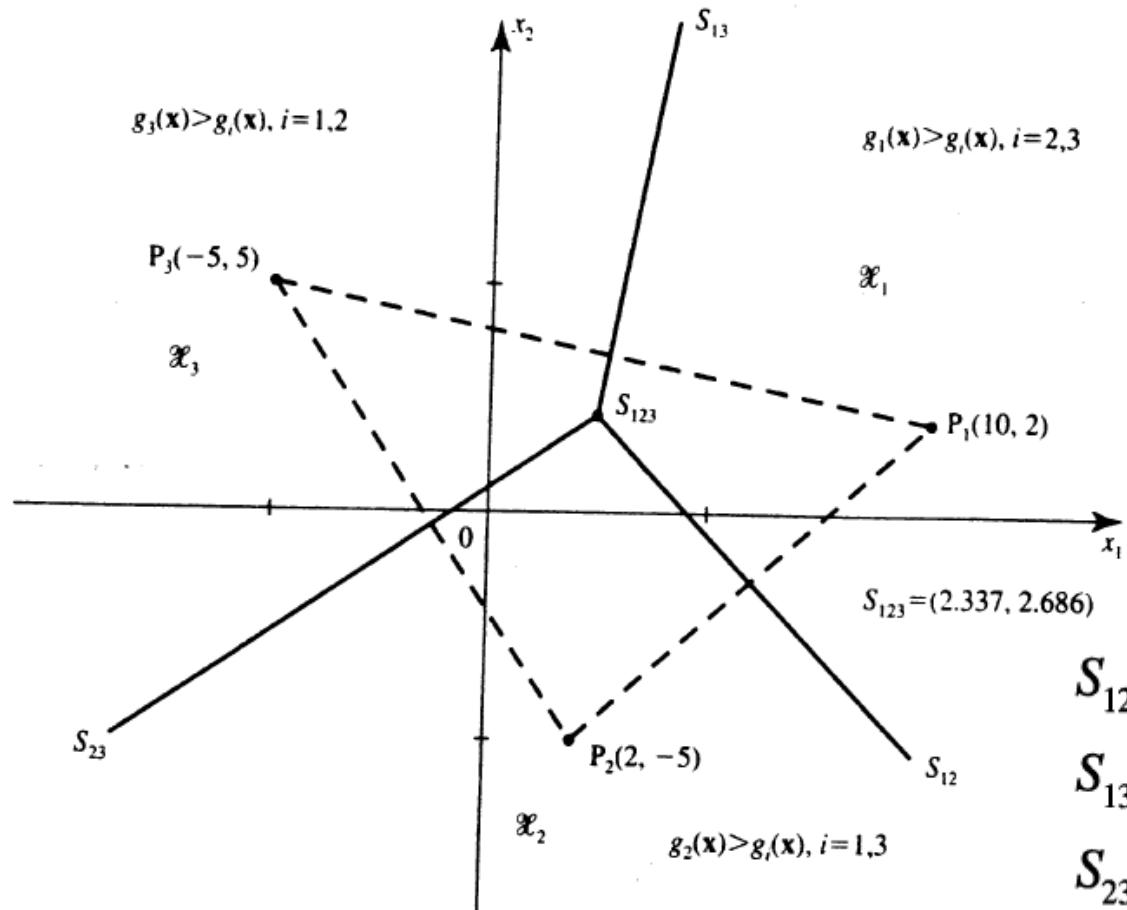
$$\mathbf{w}_i = \mathbf{x}_i$$

*En appliquant
les expressions:*

$$\mathbf{w}_{i,n+1} = -\frac{1}{2} \mathbf{x}_i^t \mathbf{x}_i, \quad \text{pour } i = 1, 2, \dots, R$$

3) Machine linéaire

- Interprétation géométrique:



$$g_1(x) = 10x_1 + 2x_2 - 52$$

$$g_2(x) = 2x_1 - 5x_2 - 14.5$$

$$g_3(x) = -5x_1 + 5x_2 - 25$$



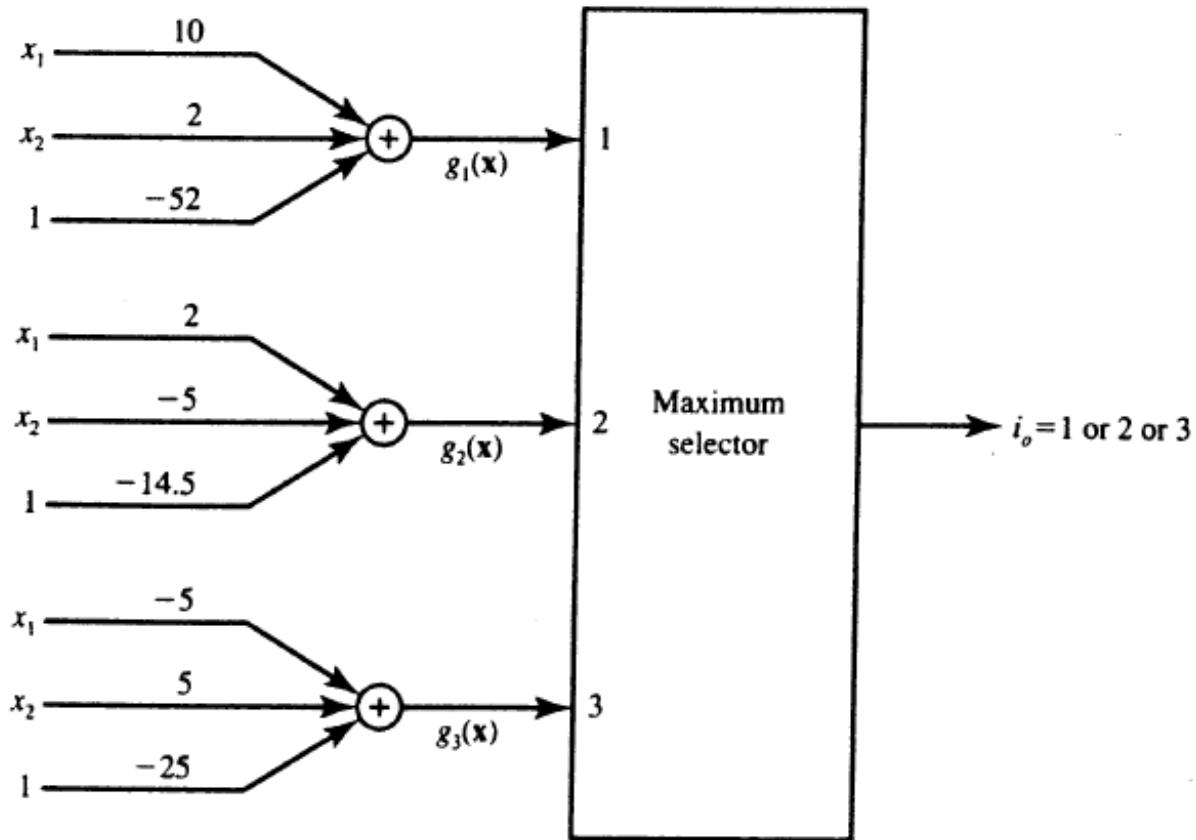
$$S_{12}: 8x_1 + 7x_2 - 37.5 = 0$$

$$S_{13}: -15x_1 + 3x_2 + 27 = 0$$

$$S_{23}: -7x_1 + 10x_2 - 10.5 = 0$$

3) Machine linéaire

- Architecture de l'exemple



3) Machine linéaire

■ En résumé:

- Si une machine linéaire peut classer correctement toutes les formes de toutes les régions, alors on dit que l'ensemble de formes est **linéairement séparable**
- Plus formellement: L'ensemble des formes est linéairement séparable si R fonctions linéaires existent et vérifient les conditions suivantes:

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \text{ pour tous les } \mathbf{x} \in \mathfrak{R}_i, \\ i = 1, 2, \dots, R; \quad \text{et} \quad j = 1, 2, \dots, R, i \neq j$$

3) Machine linéaire

Exemples de formes qui ne peuvent pas être séparées linéairement en 2D

- Soit rendre les fonctions discriminantes plus complexes
- Soit augmenter la dimension des données (on va voir cela en fin de cours)

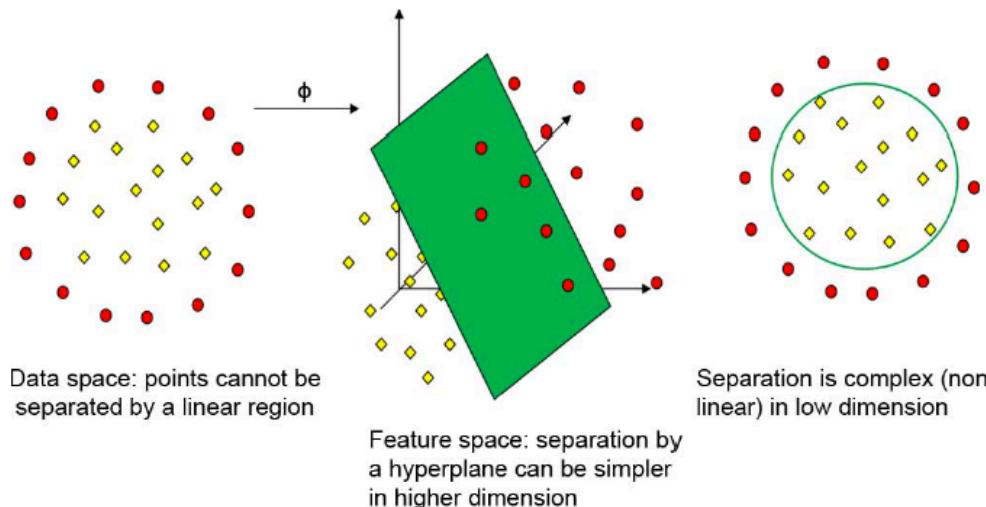


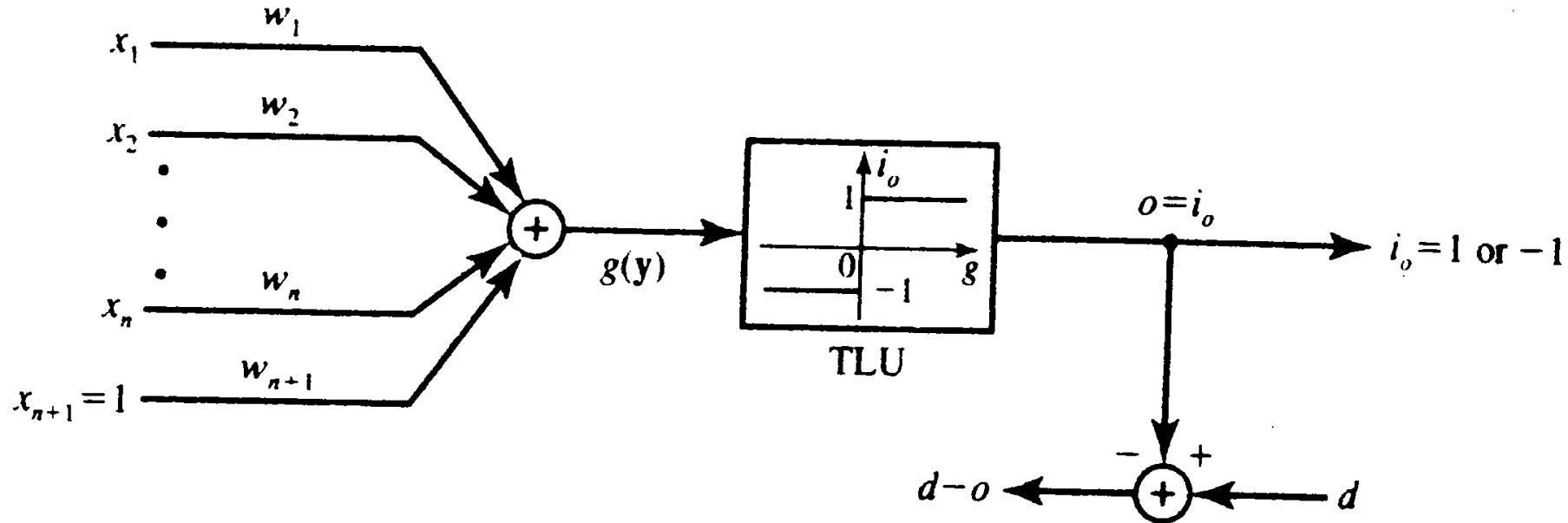
Fig. from [Ben Salah *et al.*, IEEE TIP 2011]

4) Apprentissage non paramétrique

- Nous avons vu que les coefficients des fonctions discriminantes linéaires peuvent être déterminés analytiquement en se basant sur la connaissance *a priori* d'un ensemble de formes et de leur classe d'appartenance.
- Dans ce qui suit, nous allons examiner un premier type de **réseaux de neurones** utilisé comme classificateur: **Le perceptron à fonction d'activation binaire**

4) Apprentissage non paramétrique

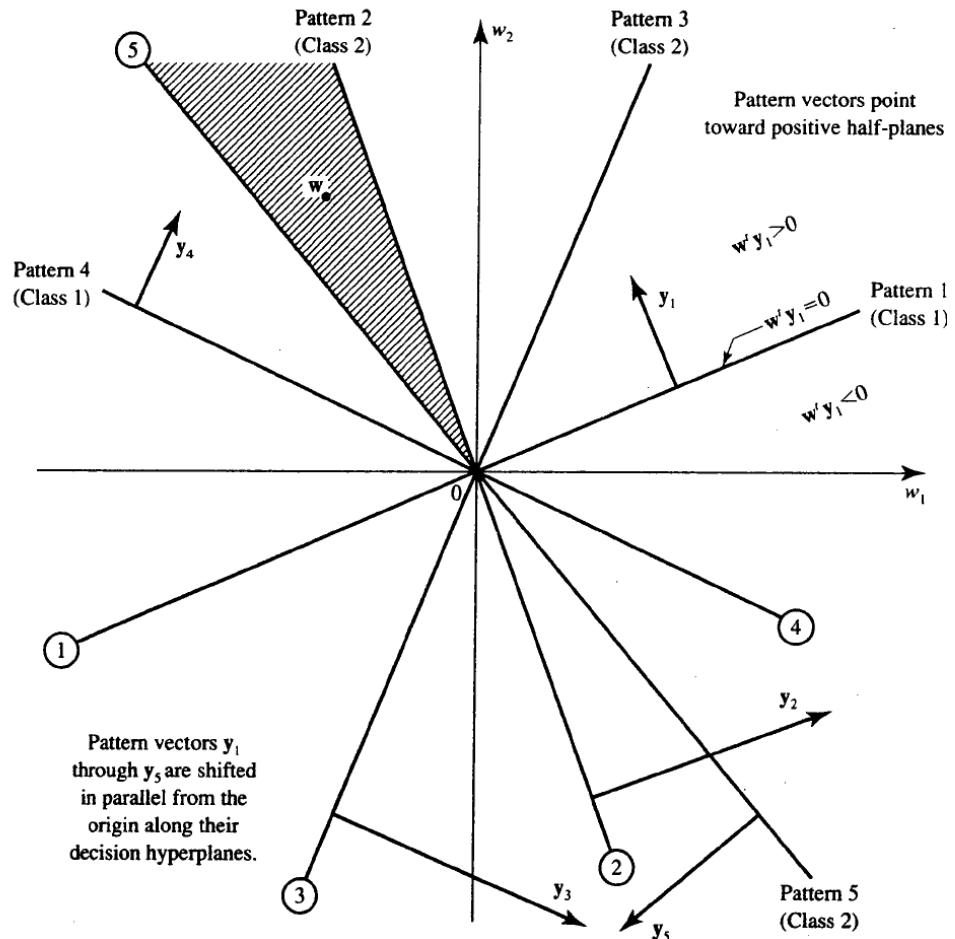
Architecture: Perceptron discret (2 classes)



4) Apprentissage non paramétrique

- Soit un ensemble d'entraînement x_1, x_2, \dots, x_p présenté au réseau
- Pour chaque forme d'entraînement, la réponse désirée est présentée au réseau par un professeur
- Le réseau apprend par expérience en comparant la réponse désirée à la réponse actuelle
- Les poids sont ajustés pour chaque **mauvaise** réponse fournie au réseau

4) Apprentissage non paramétrique



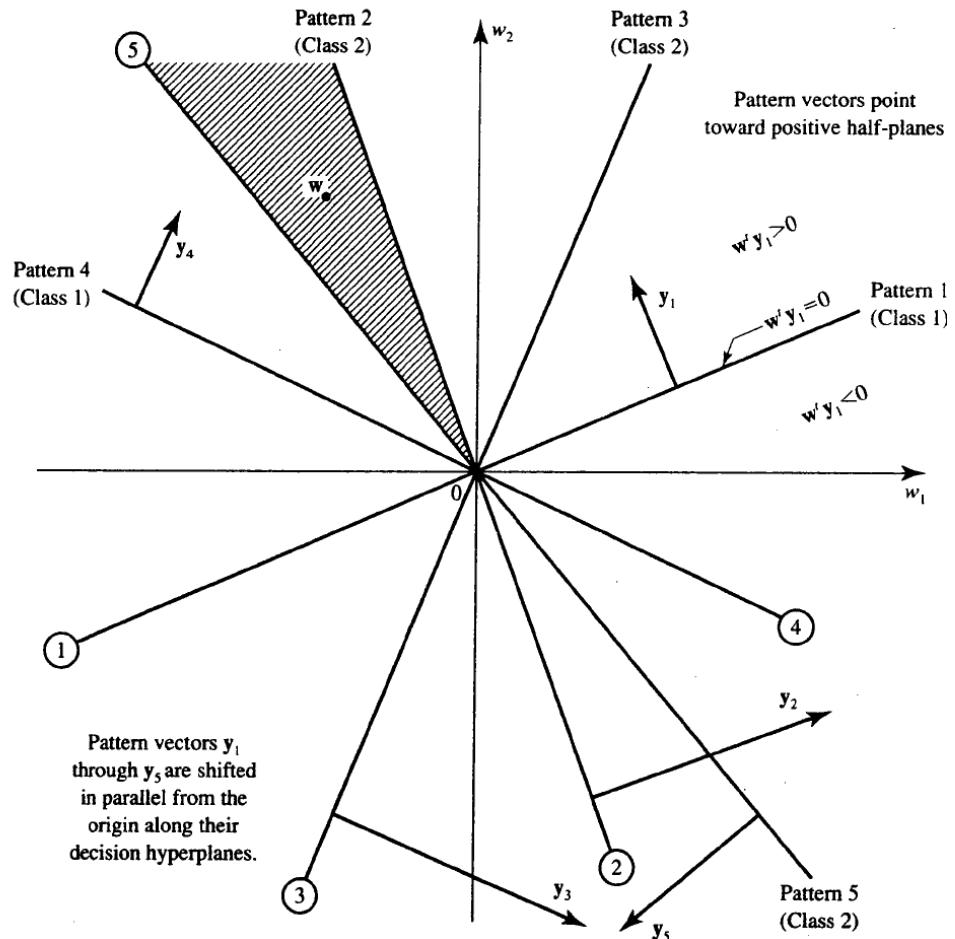
- Soit un le vecteur entrée augmentée \mathbf{y} . La surface de décision correspond à:

$$\mathbf{w}^T \mathbf{x} + w_{n+1} = 0$$



$$\mathbf{w}^T \mathbf{y} = 0$$

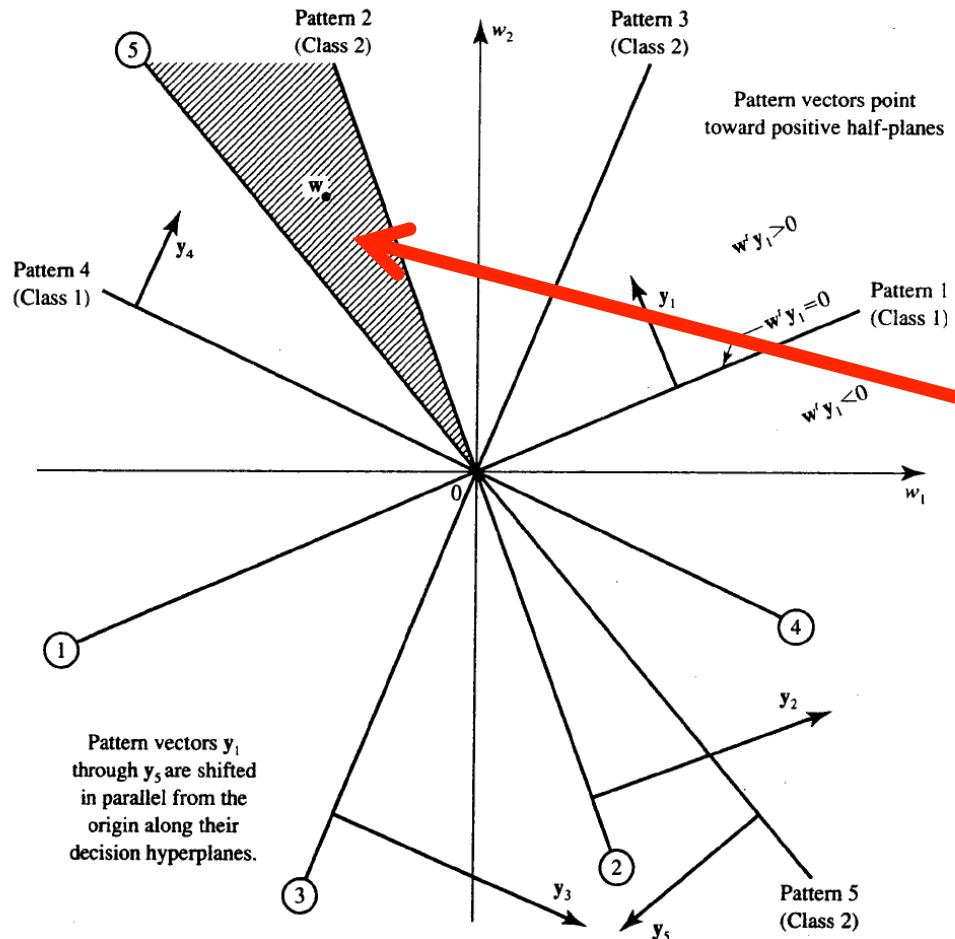
4) Apprentissage non paramétrique



- Tous les hyperplans coupent l'origine.
- Vecteur y est perpendiculaire au plan et pointe toujours vers la région positive:

$$w^T y > 0$$

4) Apprentissage non paramétrique



Correspond à l'intersection des régions de décisions positives pour les prototypes de la classe 1 et négatives pour les prototypes de la classe 2.

4) Apprentissage non paramétrique

- **Cas A:** Soit le vecteur d' entraînement \mathbf{y}_1 , et le vecteur de poids initial \mathbf{w}^I positionné dans la partie négative du plan:

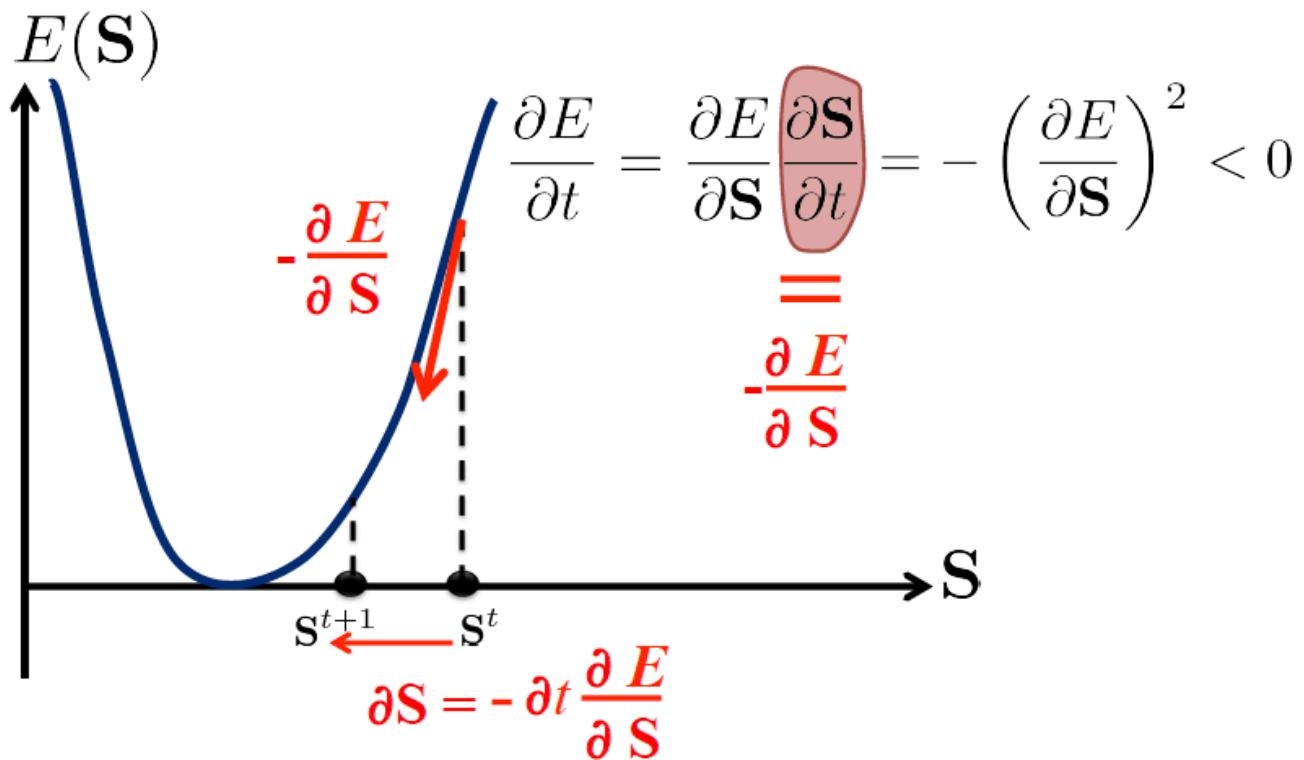
$$\mathbf{w}^I \mathbf{y}_1 < 0$$

- Une façon simple de corriger la fonction discriminante est d'ajuster le vecteur de poids en se basant sur la direction du gradient de la fonction discriminante:

$$\nabla_{\mathbf{w}}(\mathbf{w}^T \mathbf{y}_1) = \mathbf{y}_1$$

Descente de gradient (Rappel)

- Interprétation de la descente de gradient:



4) Apprentissage non paramétrique

- Cas A: Soit le vecteur d' entrainement y_1 , et le vecteur de poids initial w^1 positionné dans la partie négative du plan:

$$w^1 \cdot y_1 < 0$$

- Dans ce cas, une forme appartenant à la *classe 1* est mal classée. Donc, on ajuste le vecteur des poids de la façon suivante:

$$w' = w^1 + c y_1$$

Constante positive
(+: « montée du gradient »)

4) Apprentissage non paramétrique

- **Cas B:** La fonction discriminante est positive, mais le prototype appartient à la *classe 2 (pas la classe 1)*. Dans ce cas, on ajuste le vecteur des poids de la façon suivante:

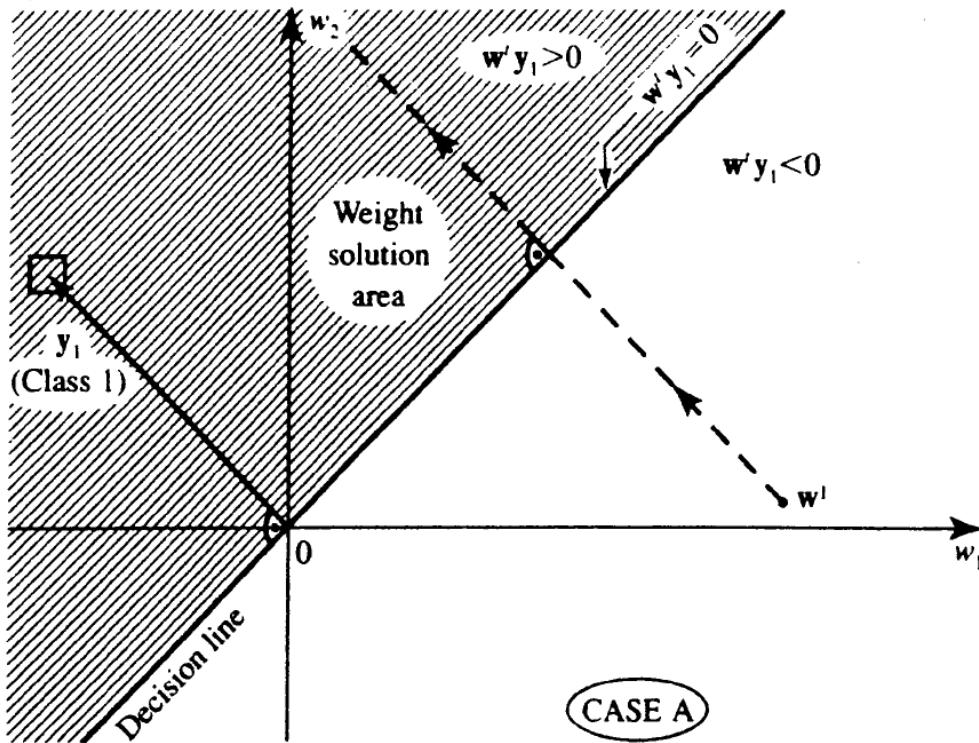
$$\mathbf{w}' = \mathbf{w}^1 - c\mathbf{y}_1$$



Constante positive
(-: « descente du gradient »)

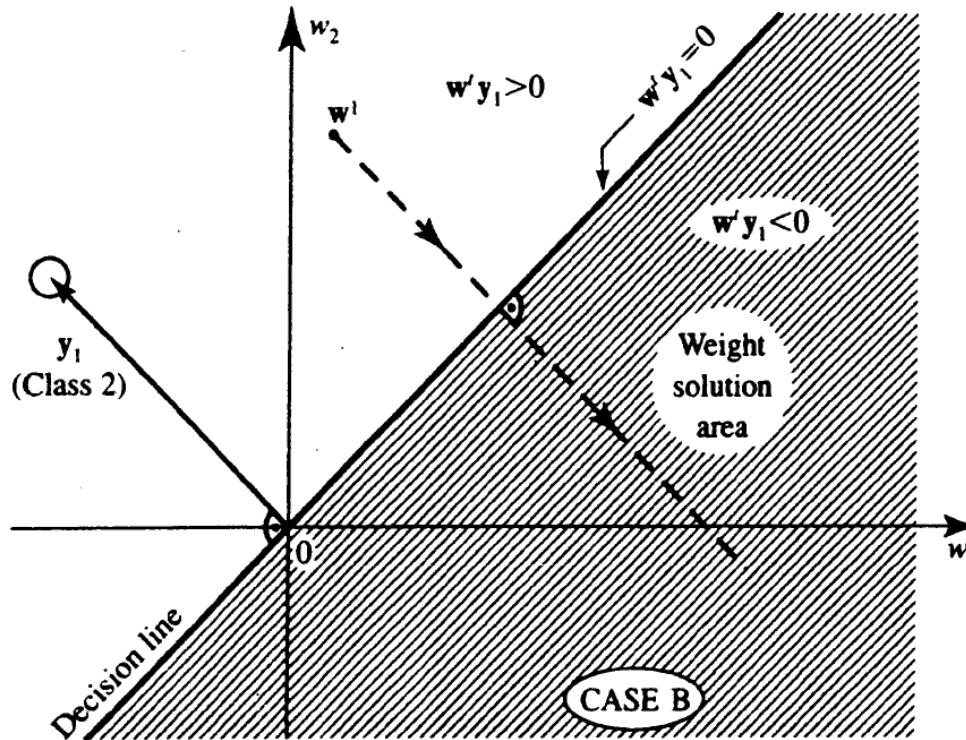
4) Apprentissage non paramétrique

- Illustration graphique de l'ajustement de poids (Cas A)

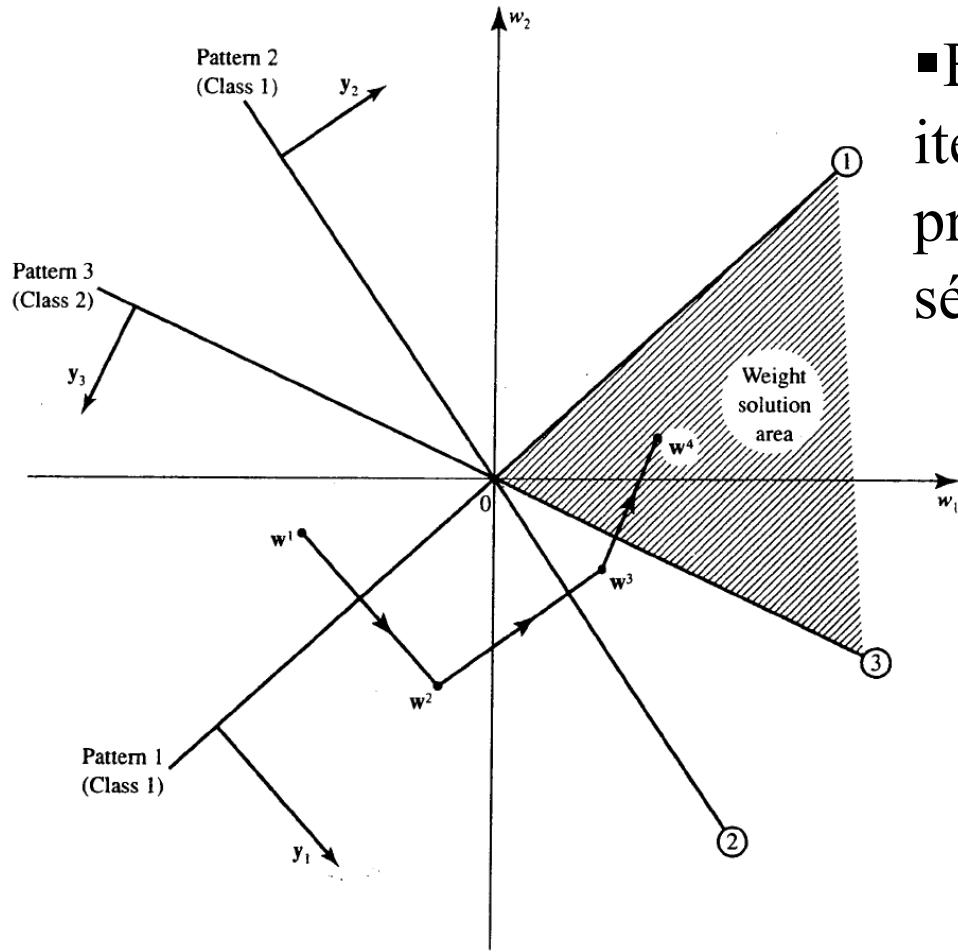


4) Apprentissage non paramétrique

- Illustration graphique de l'ajustement de poids (Cas B)



4) Apprentissage non paramétrique



- Exemple d'ajustement itératif de poids pour trois prototypes présentés séquentiellement au réseau:

$$w^2 = w^1 + cy_1$$

$$w^3 = w^2 + cy_2$$

$$w^4 = w^3 - cy_3$$



*Corrections apportées
aux poids*

5) Perceptron discret ($R = 2$ classes)

(Single Discrete Perceptron Training Algorithm: SDPTA)

► **On suppose:**

- L'ensemble d'entraînement constitué de P vecteurs d'apprentissage \mathbf{x}_i et de leur classe d'appartenance d_i , pour $i = 1, 2, \dots, P$:

$$\{(\mathbf{x}_1, d_1), (\mathbf{x}_2, d_2), \dots, (\mathbf{x}_P, d_P)\},$$

où \mathbf{x}_i est de dimension ($n \times 1$) et d_i de dimension (1×1)

- le vecteur augmenté: $y_i = \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix}$, pour $i = 1, 2, \dots, P$
- k représente le numéro de l'itération ou cycle d'entraînement
- p représente le numéro du vecteur considéré dans l'ensemble d'entraînement

5) Perceptron discret ($R = 2$ classes)

► Initialisation:

- **Étape 1:** Choisir le taux d'apprentissage $c > 0$
- **Étape 2:** Les poids du vecteur \mathbf{w} sont initialisés avec des petites valeurs choisies aléatoirement. De plus, $k \leftarrow 1$; $p \leftarrow 1$; $E \leftarrow 0$

► Le cycle d'apprentissage:

- **Étape 3:** Un vecteur \mathbf{y} est présenté au perceptron et la sortie o est évaluée:

$$\mathbf{y} \leftarrow \mathbf{y}_p ; d \leftarrow d_p ; o \leftarrow \text{sgn}(\mathbf{w}^t \mathbf{y})$$

- **Étape 4:** Mise à jour du vecteur de poids \mathbf{w} (*s'il y a erreur de classification*):

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta \mathbf{w} = \mathbf{w} + \frac{c}{2} [d - o] \mathbf{y}$$

5) Perceptron discret ($R = 2$ classes)

► Le cycle d'apprentissage: (suite)

- **Étape 5:** Mise à jour de l'erreur accumulé du cycle d'apprentissage (la fonction de coût a minimiser):

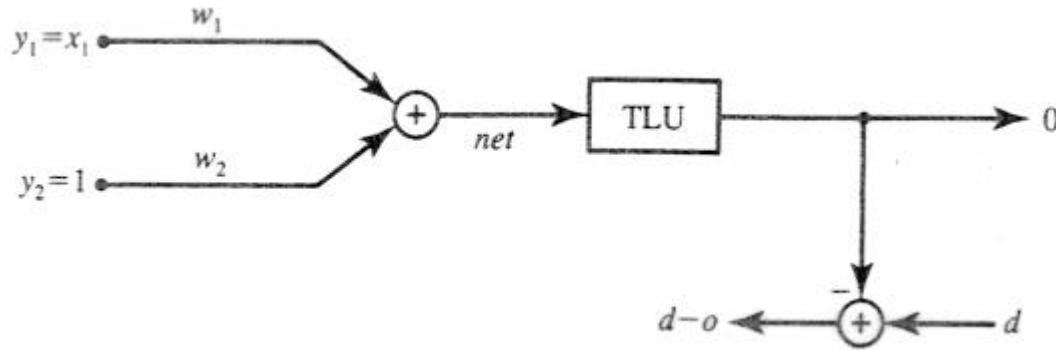
$$E \leftarrow E + \frac{1}{2} (d - o)^2$$

- **Étape 6:** Si $p < P$, alors $k \leftarrow k + 1$; $p \leftarrow p + 1$ et aller à l'étape 3. Sinon aller à l'étape 7.
- **Étape 7:** Le cycle d'apprentissage est complété ($p = P$).
Si $E = 0$, alors la phase d'apprentissage est terminée et le vecteur de poids w est conservé.
Si $E > 0$, alors $E \leftarrow 0$; $p \leftarrow 1$ et aller à l'étape 3. (et incrémenter k)

FIN (SDPTA)

5) Perceptron discret ($R = 2$ classes)

- Exemple: l'apprentissage non-paramétrique d'un perceptron discret (2 classes, 1 dimension)



- L'ensemble d'entraînement augmenté correspond à:
$$\mathbf{y}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{y}_2 = \begin{bmatrix} -0.5 \\ 1 \end{bmatrix}, \mathbf{y}_3 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \mathbf{y}_4 = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$$
- Posons arbitrairement $c = 1$ et $\mathbf{w}^1 = [-2.5 \ 1.75]^t$ et la règle d'apprentissage (fixe): $\mathbf{w}' = \mathbf{w} \pm \Delta\mathbf{w} = \mathbf{w} \pm c\mathbf{y}$

5) Perceptron discret ($R = 2$ classes)

► **Exemple:** l'apprentissage non-paramétrique d'un perceptron discret (2 classes, 1 dimension)

- La règle d'ajustement pour un perceptron discret peut être reformulée par:

$$\Delta \mathbf{w}^k = \frac{c}{2} [d_k - o_k] \mathbf{y}_k = \frac{c}{2} [d_k - sgn(\mathbf{w}^{kt} \mathbf{y}_k)] \mathbf{y}_k$$

et alors:

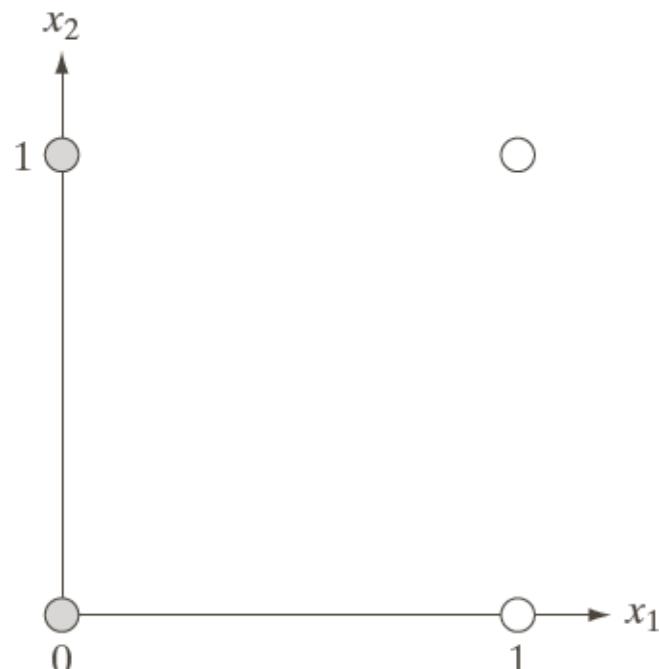
$$\Delta \mathbf{w}^k = +c\mathbf{y} \quad \text{pour une erreur sur la classe 1}$$

$$\Delta \mathbf{w}^k = -c\mathbf{y} \quad \text{pour une erreur sur la classe 2}$$

$$\Delta \mathbf{w}^k = 0 \quad \text{aucune erreur}$$

5) Perceptron discret ($R = 2$ classes)

Exemple simple:



$$\begin{aligned} \bullet &\in C_1 \\ \circ &\in C_2 \end{aligned}$$

Vecteurs augmentés
(Classe 1)

$$\mathbf{y}_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{y}_2 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

Vecteurs augmentés
(Classe 2)

$$\mathbf{y}_3 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{y}_4 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

5) Perceptron discret ($R = 2$ classes)

Exemple simple:

- Supposons que: $c = 1$, $\mathbf{w}^1 = \mathbf{0}$
- Et qu'on traite les formes d'apprentissage dans l'ordre suivant (voire les pages qui suivent):

$$\mathbf{w}^{1t} \mathbf{y}_1 = (0 \quad 0 \quad 0) \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = 0$$

$$\mathbf{w}^2 = \mathbf{w}^1 + \mathbf{y}_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

5) Perceptron discret ($R = 2$ classes)

Exemple simple:

- Supposons que: $c = 1$, $\mathbf{w}^1 = \mathbf{0}$
- Et qu'on traite les formes d'apprentissage dans l'ordre suivant (voire les pages qui suivent):

$$\mathbf{w}^{2t} \mathbf{y}_2 = (0 \quad 0 \quad 1) \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = 1$$
$$\mathbf{w}^3 = \mathbf{w}^2 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

5) Perceptron discret ($R = 2$ classes)

Exemple simple:

- Supposons que: $c = 1$, $\mathbf{w}^1 = \mathbf{0}$
- Et qu'on traite les formes d'apprentissage dans l'ordre suivant (voire les pages qui suivent):

$$\mathbf{w}^{3t} \mathbf{y}_3 = (0 \quad 0 \quad 1) \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = 1$$

$$\mathbf{w}^4 = \mathbf{w}^3 - \mathbf{y}_3 = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix}$$

5) Perceptron discret ($R = 2$ classes)

Exemple simple:

- Supposons que: $c = 1$, $\mathbf{w}^1 = \mathbf{0}$
- Et qu'on traite les formes d'apprentissage dans l'ordre suivant (voire les pages qui suivent):

$$\mathbf{w}^{4t} \mathbf{y}_4 = \begin{pmatrix} -1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = -1$$
$$\mathbf{w}^5 = \mathbf{w}^4 = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix}$$

5) Perceptron discret ($R = 2$ classes)

- On continue avec le prochain cycle
- On s'arrête quand un cycle entier est passé sans aucune erreur
- Finalement la convergence est obtenue à $k=14$:

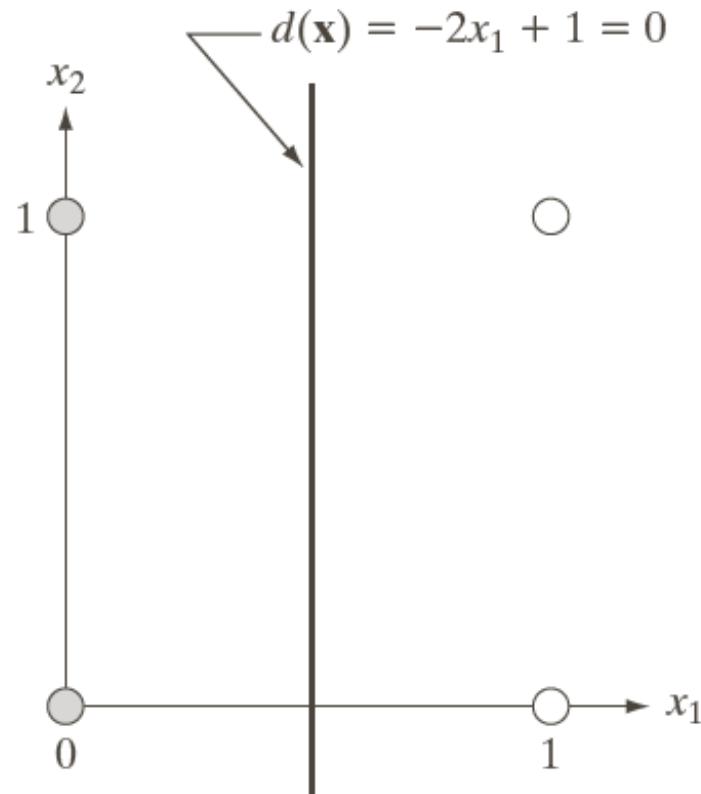
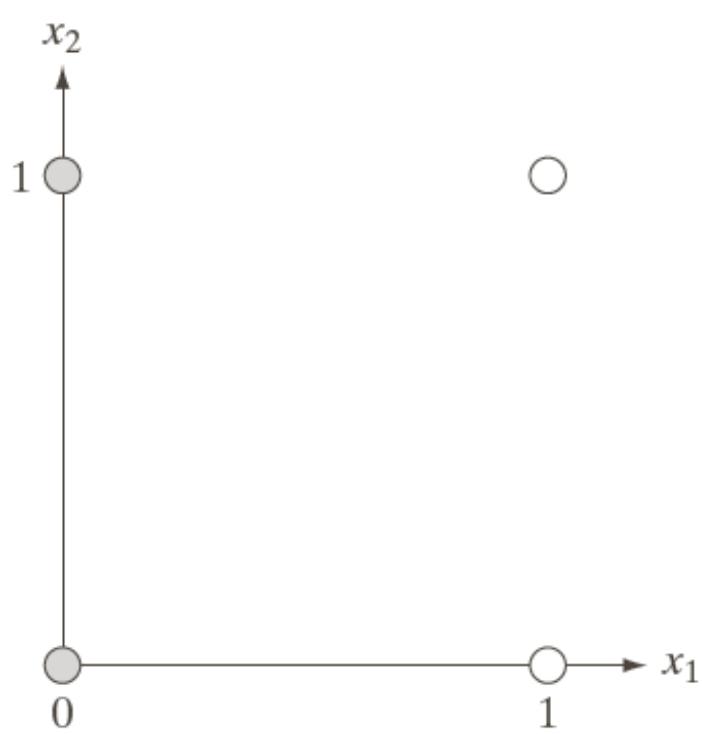
$$\mathbf{w}^{14} = (-2, 0, 1)^t$$



Fonction de décision

$$d(\mathbf{x}) = -2x_1 + 1$$

5) Perceptron discret ($R = 2$ classes)



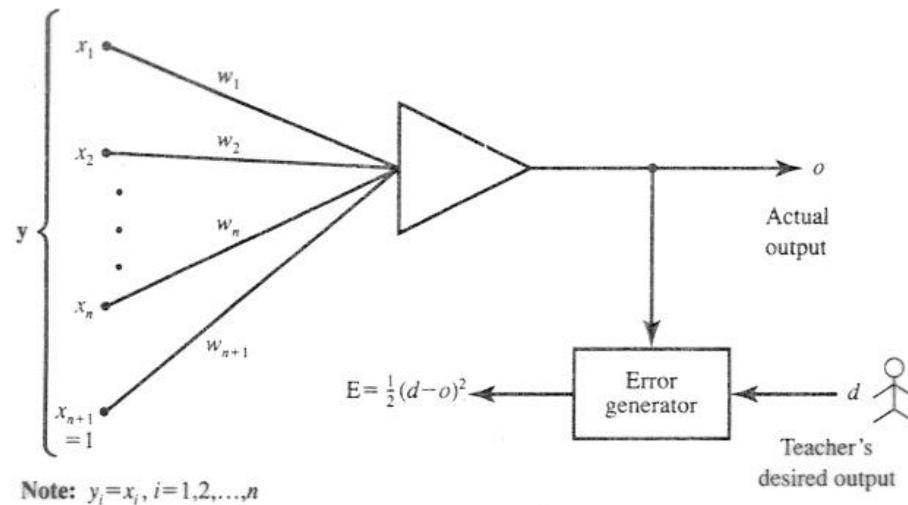
5) Perceptron discret ($R = 2$ classes)

Classification – données lin. séparables ($R = 2$ classes)

- ▶ Selon le **théorème de convergence du perceptron** (Zurada, 1982), l'apprentissage termine dans un nombre fini d'itérations si les formes des deux classes sont linéairement séparables.
 - k_0 : nombre d'étapes d'apprentissages (itérations) après lequel aucun erreur de classification a lieu avec la base d'entraînement.
 - Le nombre d'étapes k_0 requis pour compléter l'apprentissage dépend de l'ordre de présentation des vecteurs de l'ensemble d'entraînement et de la valeur de c .

6) Perceptron continu ($R = 2$ classes)

Architecture du perceptron continu



- **La fonction d'activation continue a l'avantage :**
 - de procurer un meilleur contrôle lors de l'entraînement pour l'ajustement des poids (car o est continu);
 - d'être dérivable, ce qui permet l'évaluation du gradient de l'erreur évaluée à la sortie du neurone.

6) Perceptron continu ($R = 2$ classes)

- ▶ **Principe de l'entraînement**
- ▶ **On tente de résoudre le problème d'ajustement des poids selon la descente de gradient des erreurs:**
 - Soit un vecteur de poids \mathbf{w} choisi arbitrairement, alors le gradient de la fonction d'erreur $\nabla E(\mathbf{w})$ peut être évalué
 - La prochaine valeur de \mathbf{w} est obtenue en ajustant les poids dans la direction négative du gradient, soit la direction de la plus grande pente:

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \eta \nabla E(\mathbf{w})$$

6) Perceptron continu ($R = 2$ classes)

Principe de l'entraînement

- **Notre objectif est de minimiser la fonction d'erreur $E(w)$ dans l'espace des poids de dimension ($n+1$):**
 - Définissons E_k , la fonction d'erreur à minimiser à l'étape k de la procédure d'entraînement :

$$E_k = \frac{1}{2}(d^k - o^k)^2, \text{ ou}$$

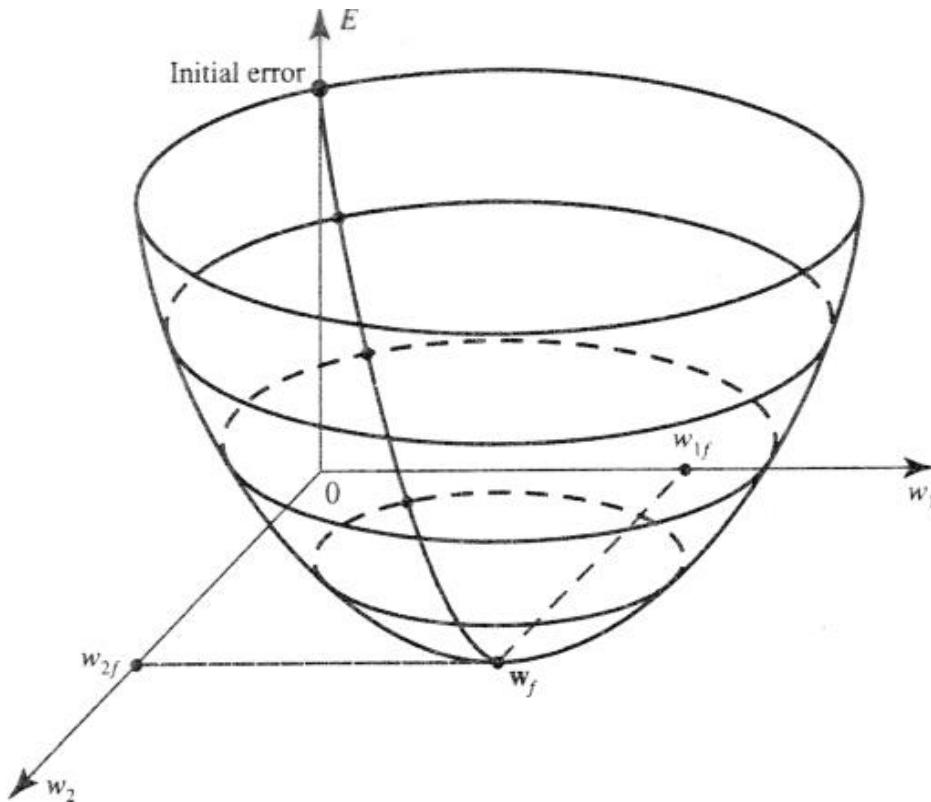
$$E_k = \frac{1}{2}[d^k - f(w^k y^k)]^2$$

(la différence au carré entre la réponse désirée d^k et la sortie du neurone o^k)

6) Perceptron continu ($R = 2$ classes)

Principe de l'entraînement

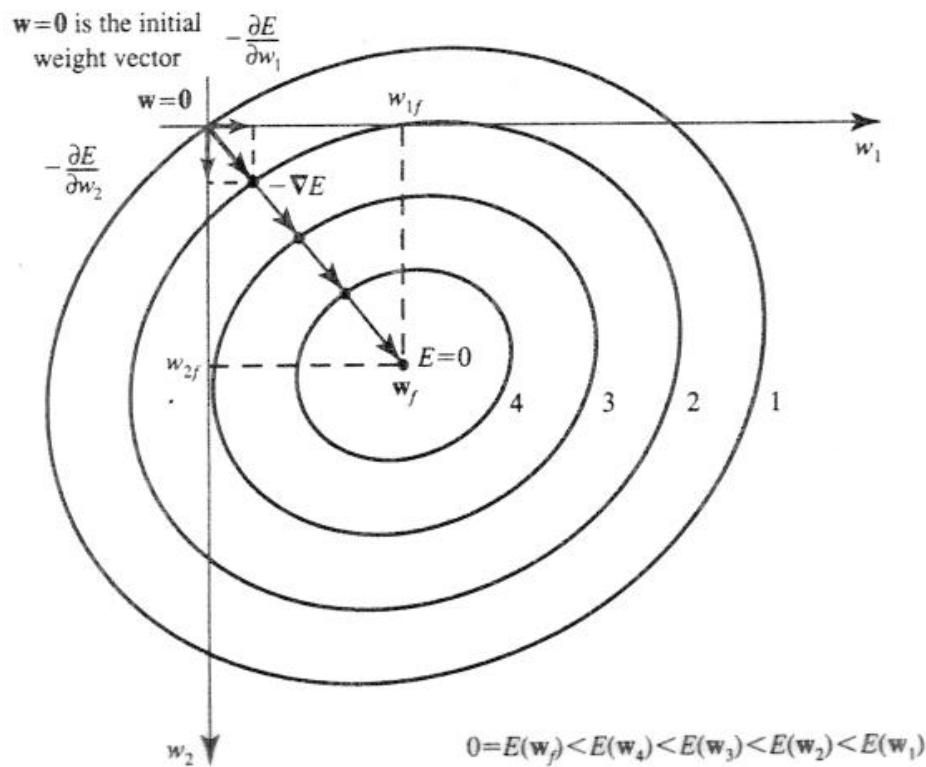
- ▶ Illustration de la fonction d'erreur dans l'espace des poids avec $n = 1$:



6) Perceptron continu ($R = 2$ classes)

Principe de l'entraînement

- ▶ Illustration de la fonction d'erreur dans l'espace des poids avec $n = 1$:



6) Perceptron continu ($R = 2$ classes)

Principe de l'entraînement

► La minimisation de la fonction d'erreur nécessite l'évaluation du gradient de l'erreur:

$$\nabla E(w) = \frac{1}{2} \nabla [d - f(\text{net})]^2$$

— pour un espace de dimension $(n+1)$, l'équation précédente devient (après remplacement):

$$\nabla E(w) \doteq \begin{bmatrix} \frac{\partial E}{\partial w_1} \\ \frac{\partial E}{\partial w_2} \\ \vdots \\ \vdots \\ \frac{\partial E}{\partial w_{n+1}} \end{bmatrix} = -(d - o) f'(net) \begin{bmatrix} \frac{\partial (\text{net})}{\partial w_1} \\ \frac{\partial (\text{net})}{\partial w_2} \\ \vdots \\ \vdots \\ \frac{\partial (\text{net})}{\partial w_{n+1}} \end{bmatrix}$$

6) Perceptron continu ($R = 2$ classes)

Principe de l'entraînement

- Étant donnée que $net = \mathbf{w}^t \mathbf{y}$, alors nous avons:

$$\frac{\partial(\text{net})}{\partial w_i} = y_i, \quad \text{pour } i = 1, 2, \dots, n+1$$

et l'équation du gradient de la fonction d'erreur se résume à:

$$\nabla E(\mathbf{w}) = -(d-o) f'(net) \mathbf{y}$$

où

$$\frac{\partial E}{\partial w_i} = -(d-o) f'(net) y_i$$

Remarque: Cette règle d'entraînement est équivalente à la règle d'apprentissage delta présentée dans partie A.1.

6) Perceptron continu ($R = 2$ classes)

Principe de l'entraînement

- ▶ Pour ajuster les poids, il faut spécifier une fonction d'activation particulière:
 - Étant donné la fonction d'activation **continue bipolaire** ($\lambda=1$) :

$$f(\text{net}) = \frac{2}{1 + \exp(-\text{net})} - 1$$

on obtient la dérivée:

$$f'(\text{net}) = \frac{2 \exp(-\text{net})}{[1 + \exp(-\text{net})]^2}$$

Remarque: étant donné que les poids sont ajustables, le gain λ de la fonction d'activation sigmoïde peut être considéré fixe et non variable. Posons arbitrairement $\lambda=1$.

6) Perceptron continu ($R = 2$ classes)

Principe de l'entraînement

- ▶ Pour ajuster les poids, il faut spécifier une fonction d'activation particulière:
 - Enfin, on peut montrer que:

$$\frac{2 \exp(-net)}{[1 + \exp(-net)]^2} = \frac{1}{2}(1 - o^2)$$

- En effet, on peut dériver:

$$\frac{1}{2}(1 - o^2) = \frac{1}{2} \left[1 - \left(\frac{1 - \exp(-net)}{1 + \exp(-net)} \right)^2 \right] = \frac{2 \exp(-net)}{[1 + \exp(-net)]^2}$$

6) Perceptron continu ($R = 2$ classes)

- ▶ **Principe de l'entraînement**
- ▶ **Dans ce cas, l'équation du vecteur gradient devient:**

$$\nabla E(\mathbf{w}) = -\frac{1}{2} (d - o)(1 - o^2) \mathbf{y}$$

- ▶ Enfin, la règle d'apprentissage delta pour une fonction d'activation continue bipolaire (étape d'entraînement k):

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \eta \nabla E(\mathbf{w}) = \mathbf{w}^k + \frac{\eta}{2} (d^k - o^k)(1 - o^{k2}) \mathbf{y}^k$$

Note – problème de convergence des poids: La différence fondamentale entre le perceptron **discret** et le perceptron **continu** est que l'apprentissage du perceptron discret conduit toujours à une solution lorsque les vecteurs de l'ensemble d'entraînement sont linéairement séparables, ce qui n'est pas nécessairement le cas pour le perceptron continu (Wittner et Denker 1988).

6) Perceptron continu ($R = 2$ classes)

(Single Continuous Perceptron Training Algorithm: SCPTA)

► **On suppose:**

- l'ensemble d'entraînement constitué de P vecteurs d'apprentissage \mathbf{x}_i et de leur classe d'appartenance d_i , pour $i = 1, 2, \dots, P$:

$$\{(\mathbf{x}_1, d_1), (\mathbf{x}_2, d_2), \dots, (\mathbf{x}_P, d_P)\},$$

où \mathbf{x}_i est de dimension ($n \times 1$) et d_i de dimension (1×1)

- le vecteur augmenté: $\mathbf{y}_i = \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix}$, pour $i = 1, 2, \dots, P$
- k représente le numéro de l'itération ou cycle d'entraînement
- p représente le numéro du vecteur considéré dans l'ensemble d'entraînement

6) Perceptron continu ($R = 2$ classes)

► Algorithme d'apprentissage d'un perceptron continu

► Initialisation:

- **Étape 1:** Choisir les paramètres $\eta > 0$, $\lambda = 1$ et $E_{\max} > 0$
- **Étape 2:** Les poids du vecteur w sont initialisés avec des petites valeurs choisies aléatoirement. De plus, $k \leftarrow 1$; $p \leftarrow 1$; $E \leftarrow 0$

► Le cycle d'apprentissage:

- **Étape 3:** Un vecteur y est présenté au perceptron et la sortie o est évaluée:
$$y \leftarrow y_p ; d \leftarrow d_p ; o \leftarrow f(w^t y)$$

$$f(\text{net}) = \frac{2}{1 + \exp(-\text{net})} - 1$$

- **Étape 4:** Mise à jour du vecteur de poids w (*s'il y a erreur de classification*):
$$w \leftarrow w + \Delta w = w + \frac{1}{2} \eta (d - o)(1 - o^2) y$$

6) Perceptron continu ($R = 2$ classes)

► Algorithme d'apprentissage d'un perceptron continu

► Le cycle d'apprentissage: (suite)

- **Étape 5:** Mise à jour de l'erreur accumulé du cycle d'apprentissage (la fonction de coût a minimiser):

$$E \leftarrow E + \frac{1}{2} (d - o)^2$$

- **Étape 6:** Si $p < P$, alors $k \leftarrow k + 1$; $p \leftarrow p + 1$ et aller à l'étape 3. Sinon aller à l'étape 7.
- **Étape 7:** Le cycle d'apprentissage est complété ($p = P$).

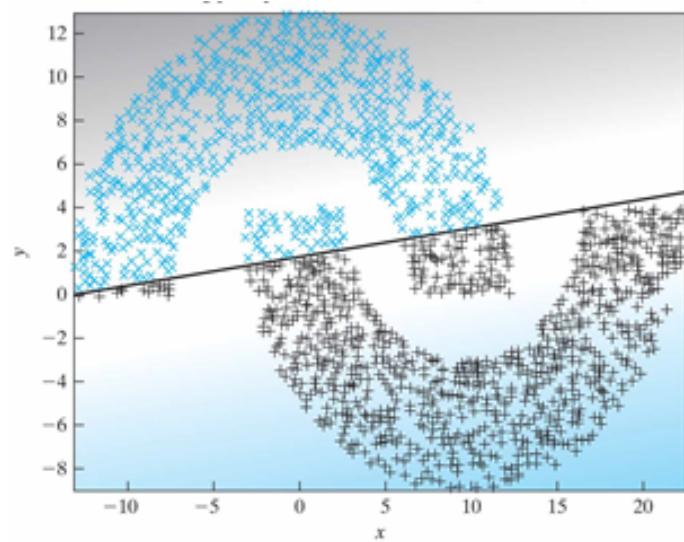
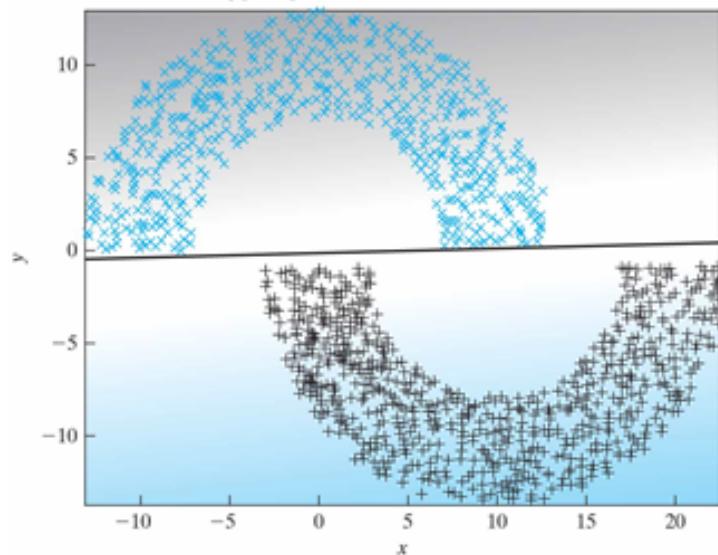
Si $E < E_{\max}$, alors la phase d'apprentissage est terminée et le vecteur de poids w , k et E sont conservés.

Si $E \geq E_{\max}$, alors $E \leftarrow 0$; $p \leftarrow 1$ et un nouveau cycle d'entraînement est requis. Aller à l'étape 3.

FIN (SCPTA)

6) Perceptron continu ($R = 2$ classes)

La frontière de décision change avec les paramètres



7) Réseau perceptron monocouche

- ▶ **Problèmes de classification à R classes**
 - ▶ **Supposons maintenant**
 - les R classes sont linéairement séparables deux à deux
 - chaque classe est linéairement séparable de l'ensemble des $(R-1)$ autres classes
 - Cette affirmation équivaut à dire qu'il existe R fonctions discriminantes linéaires telles que, pour $\forall \mathbf{x} \in \mathbb{R}^n$:
- $$g_i(\mathbf{x}) = \max \{ g_j(\mathbf{x}): j = 1, 2, \dots, R, i \neq j \}$$
- ou
$$g_i(\mathbf{x}) > g_j(\mathbf{x}), \text{ pour } i, j = 1, 2, \dots, R, i \neq j$$

7) Réseau perceptron monocouche

Classifieur à R classes – perceptrons discrets

► Pour résoudre ce problème, nous allons développer un procédure d'apprentissage pour un réseau monocouche composé de R perceptrons discrets

- Définissons le vecteur de poids augmenté:

$$\mathbf{w}_q = [w_{q,1} \ w_{q,2} \dots \ w_{q,n+1}]^t$$

- Lorsqu'un vecteur augmenté \mathbf{y} de la classe i est présenté à l'entrée du classificateur, les R fonctions de décision:

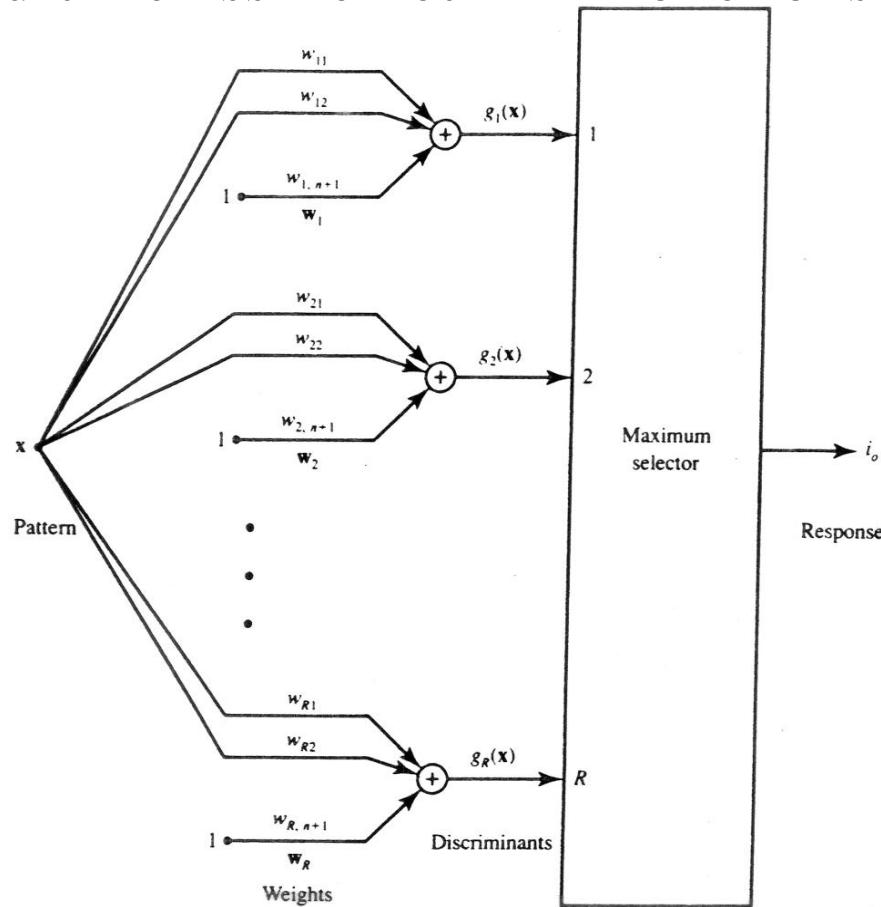
$$g_1(\mathbf{y}) = \mathbf{w}_1^t \mathbf{y}, \ g_2(\mathbf{y}) = \mathbf{w}_2^t \mathbf{y}, \dots, \ g_R(\mathbf{y}) = \mathbf{w}_R^t \mathbf{y}$$

sont évaluées.

7) Réseau perceptron monocouche

Classifieur à R classes – perceptrons discrets

- L'architecture d'un classificateur à R fonctions discriminantes



7) Réseau perceptron monocouche

Classifieur à R classes – perceptrons discrets

- **Apprentissage – le cas d'une bonne prédiction ($o = d$):**
 - Si $g_i(\mathbf{y}) = \mathbf{w}_i^t \mathbf{y}$ est plus grand que les $R-1$ autres fonctions discriminantes, alors les vecteurs de poids ne doivent pas être modifiés, étant donné que la classification est correcte
 - Les poids après corrections correspondent à :

$$\mathbf{w'}_1 = \mathbf{w}_1$$

$$\mathbf{w'}_2 = \mathbf{w}_2$$

....

$$\mathbf{w'}_R = \mathbf{w}_R \quad \dots \text{aucun changement}$$

7) Réseau perceptron monocouche

Classifieur à R classes – perceptrons discrets

- Apprentissage – le cas d'une mauvaise prédiction ($o \neq d$):
 - Si $g_i(\mathbf{y}) = \mathbf{w}_i^t \mathbf{y}$ n'est pas plus grand que les $R-1$ fonctions discriminantes, alors les vecteurs de poids sont modifiés
 - supposons que pour m fonctions discriminantes, nous avons $g_i(\mathbf{y}) = \mathbf{w}_i^t \mathbf{y} \leq g_m(\mathbf{y}) = \mathbf{w}_m^t \mathbf{y}$, alors les vecteurs de poids sont corrigés comme suit:

$$\mathbf{w'}_i = \mathbf{w}_i + c\mathbf{y}$$

$$\mathbf{w'}_m = \mathbf{w}_m - c\mathbf{y}$$

....

$$\mathbf{w'}_k = \mathbf{w}_k \quad \text{pour } k = 1, 2, \dots, R, k \neq i, m$$

7) Réseau perceptron monocouche

Classifieur à R classes – perceptrons discrets

► **Apprentissage – le cas d'une mauvaise prédiction ($o \neq d$):**

- Les équations précédentes peuvent également être présentées sous la forme de composants j de vecteurs:

$$w'_{ij} = w_{ij} + cy_j \quad \text{pour } j = 1, 2, \dots, n+1$$

$$w'_{mj} = w_{mj} - cy_j \quad \text{pour } j = 1, 2, \dots, n+1$$

$$w'_{kj} = w_{kj},$$

$$\text{pour } k = 1, 2, \dots, R, \quad k \neq i, m, \quad j = 1, 2, \dots, n+1$$

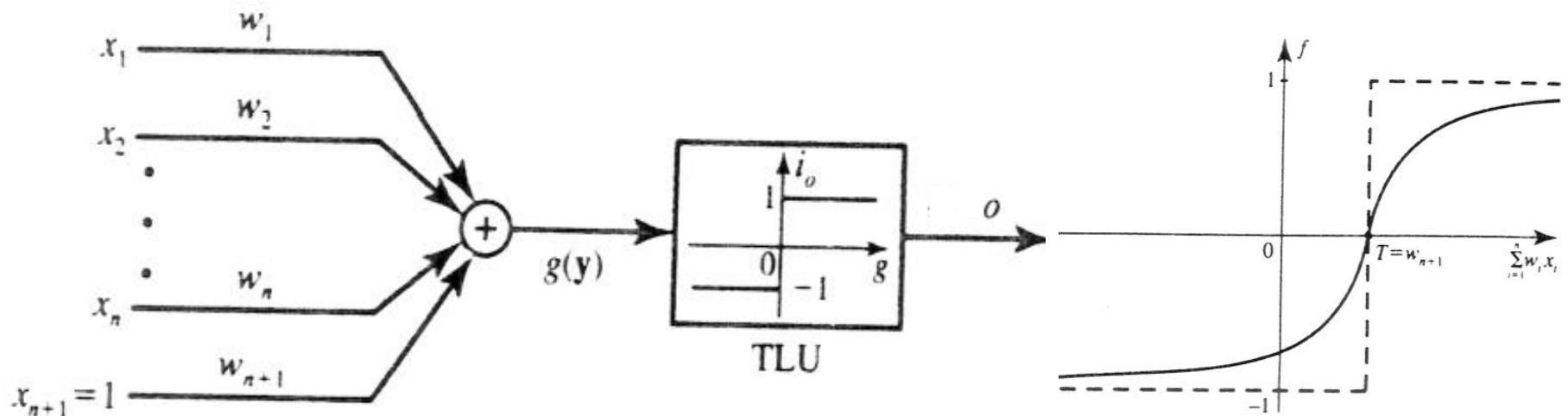
- **Stratégie:** les poids w_i augmentent selon cy si la sortie de o_i est trop faible; les poids w_m diminuent selon cy si la sortie de o_m est trop forte

7) Réseau perceptron monocouche

Classifieur à R classes – perceptrons discrets

► Ajustement du seuil:

- Rappelons que dans le cas du perceptron discret, le poids w_{n+1} permet l'ajustement du seuil T lors de l'apprentissage



7) Réseau perceptron monocouche

Classifieur à R classes – perceptrons discrets

► Ajustement du seuil:

- Si le neurone précédent est utilisé (avec $y_{n+1} = -1$), nous pouvons écrire:

$$\text{net} = \mathbf{w}^t \mathbf{x} + \text{biais} = \mathbf{w}^t \mathbf{x} - w_{n+1}$$

avec le seuil $T = w_{n+1}$.

- La sortie du neurone peut être exprimée par la relation:

$$f(\text{net}) = \begin{cases} >0 & \text{pour } \mathbf{w}^t \mathbf{x} > T \\ <0 & \text{pour } \mathbf{w}^t \mathbf{x} < T \end{cases}$$

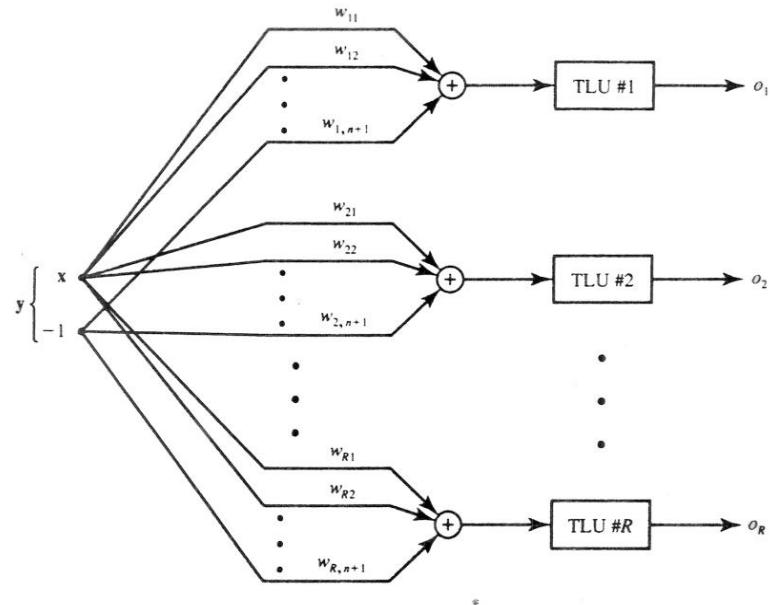
(le neurone est activé si $\text{net} > T$)

7) Réseau perceptron monocouche

Classifieur à R classes – perceptrons discrets

► Ajustement du seuil:

- L'architecture suivante basée sur R perceptrons avec un seuil de décision ajustable permet d'enlever l'opérateur MAX du classificateur à distance minimum précédent



7) Réseau perceptron monocouche

Classifieur à R classes – perceptrons discrets

► Ajustement du seuil:

- Pour l'**entraînement supervisé** de ce réseau, l'ajustement du vecteur de poids du $i^{\text{ième}}$ perceptron à l'étape k se résume à:

$$\mathbf{w}_i^{k+1} = \mathbf{w}_i^k + \frac{c^k}{2}(d_i^k - o_i^k)\mathbf{y}^k, \quad \text{pour } i = 1, 2, \dots, R$$

- **Représentation des classes:** Le nombre de neurones doit être plus grand ou égal au nombre de classes R .
 - cas local: un neurone par classe
 - cas distribué: un ou plusieurs neurones par classe

7) Réseau perceptron monocouche

Classifieur à R classes – perceptrons discrets

► Ajustement du seuil:

- Si l'implantation de notre classificateur a R classes est basée sur **une représentation locale des observations**, alors les valeurs désirées doivent respecter la formulation suivante (un neurone par classe):

$$d_i = 1, \quad d_j = -1, \quad \text{pour } j = 1, 2, \dots, R, \quad j \neq i$$

- Dans le cas où nous avons **une représentation distribuée des données** dans l'espace des formes, alors plusieurs neurones peuvent avoir une sortie désirée égale à +1.

7) Réseau perceptron monocouche

(R-Category Discrete Perceptron Training Algorithm: RDPTA)

► **On suppose:**

- une représentation locale (un neurone représente chaque classe)
- l'ensemble d'entraînement constitué de P vecteurs d'apprentissage \mathbf{x}_i et de leur classe d'appartenance \mathbf{d}_i , pour $i = 1, 2, \dots, P$:

$$\{(\mathbf{x}_1, \mathbf{d}_1), (\mathbf{x}_2, \mathbf{d}_2), \dots, (\mathbf{x}_P, \mathbf{d}_P)\},$$

où \mathbf{x}_i est de dimension ($n \times 1$) et \mathbf{d}_i de dimension ($R \times 1$)

- le vecteur augmenté: $\mathbf{y}_i = \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix}$, pour $i = 1, 2, \dots, P$
- k : le numéro de l'itération ou cycle d'entraînement
- p : numéro du vecteur considéré dans l'ensemble d'entraînement

7) Réseau perceptron monocouche

(R-Category Discrete Perceptron Training Algorithm: RDPTA)

► Initialisation:

- **Étape 1:** Choisir le taux d'apprentissage $\eta > 0$
- **Étape 2:** La matrice de poids $\mathbf{W} = [\mathbf{w}_{ij}]$ de dimension $R \times (n+1)$ est initialisée aléatoirement avec des petites valeurs, et $k \leftarrow 1$, $p \leftarrow 1$ et $E \leftarrow 0$

► Cycle d'apprentissage

- **Étape 3:** Un vecteur \mathbf{y} est présenté au réseau et les sorties sont calculées.

$$\mathbf{y} \leftarrow \mathbf{y}_p \text{ et } \mathbf{d} \leftarrow \mathbf{d}_p$$

$$o_i \leftarrow \text{sgn}(\mathbf{w}_i^T \mathbf{y}), \quad \text{pour } i = 1, 2, \dots, R$$

- **Étape 4:** Les poids sont ajustés

$$\mathbf{w}_i \leftarrow \mathbf{w}_i + \frac{1}{2} c (d_i - o_i) \mathbf{y}, \quad \text{pour } i = 1, 2, \dots, R$$

7) Réseau perceptron monocouche

(R-Category Discrete Perceptron Training Algorithm: RDPTA)

► Le cycle d'apprentissage: (suite)

- **Étape 5:** Mise à jour de l'erreur accumulé du cycle d'apprentissage (la fonction de coût a minimiser) pour $i = 1, 2, \dots, R$:

$$E \leftarrow E + \frac{1}{2} \sum_{i=1}^R (d_i - o_i)^2$$

- **Étape 6:** Si $p < P$, alors $k \leftarrow k + 1$; $p \leftarrow p + 1$ et aller à l'étape 3. Sinon aller à l'étape 7.
- **Étape 7:** Le cycle d'apprentissage est complété ($p = P$).

Si $E = 0$, alors la phase d'apprentissage est terminée et la matrice de poids \mathbf{W} est conservée.

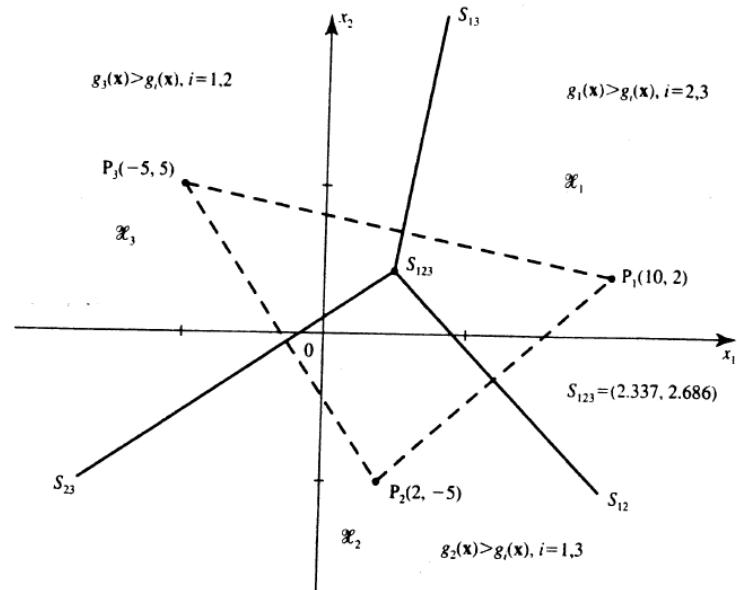
Si $E > 0$, alors $E \leftarrow 0$; $p \leftarrow 1$ et aller à l'étape 3. **FIN (RDPTA)**

7) Réseau perceptron monocouche

- **Exemple:** Reprenons l'exemple de conception d'un classifieur linéaire (a distance minimum) avec $R=3$ classes:
 - On avait une connaissance *a priori* des prototypes:

$$\mathbf{x}_1 = \begin{bmatrix} 10 \\ 2 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 2 \\ -5 \end{bmatrix}, \quad \mathbf{x}_3 = \begin{bmatrix} -5 \\ 5 \end{bmatrix}$$

- Interprétation géométrique:



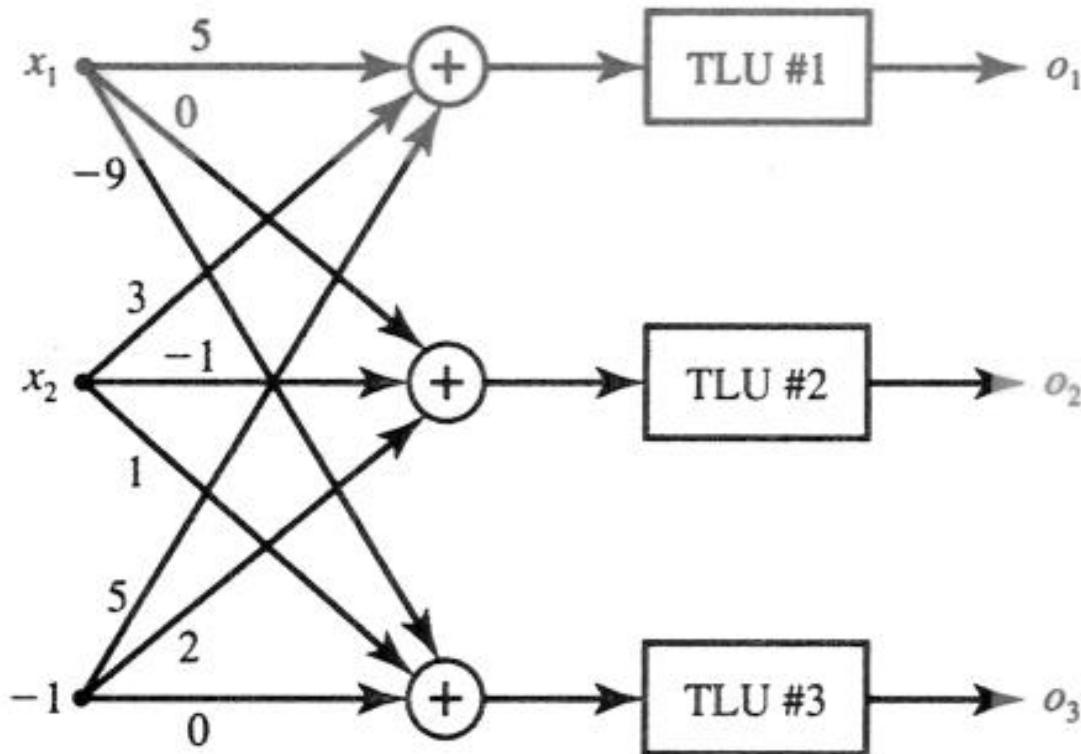
7) Réseau perceptron monocouche

► Exemple: classifieur linéaire avec $R=3$ classes

- On utilise un réseau monocouche avec 3 perceptrons discrets
- On veut évaluer les poids du classificateur en utilisant l'algorithme d'apprentissage précédent.
- Dans cet exemple, les poids sont initialisés à des valeurs arbitraires, et le réseau est entraîné avec les vecteurs étiquetés de l'ensemble d'entraînement.
- On présente les 3 vecteurs d'entraînement augmentées pendant plusieurs itérations
- On arrête quand $E_{tot} = 0$

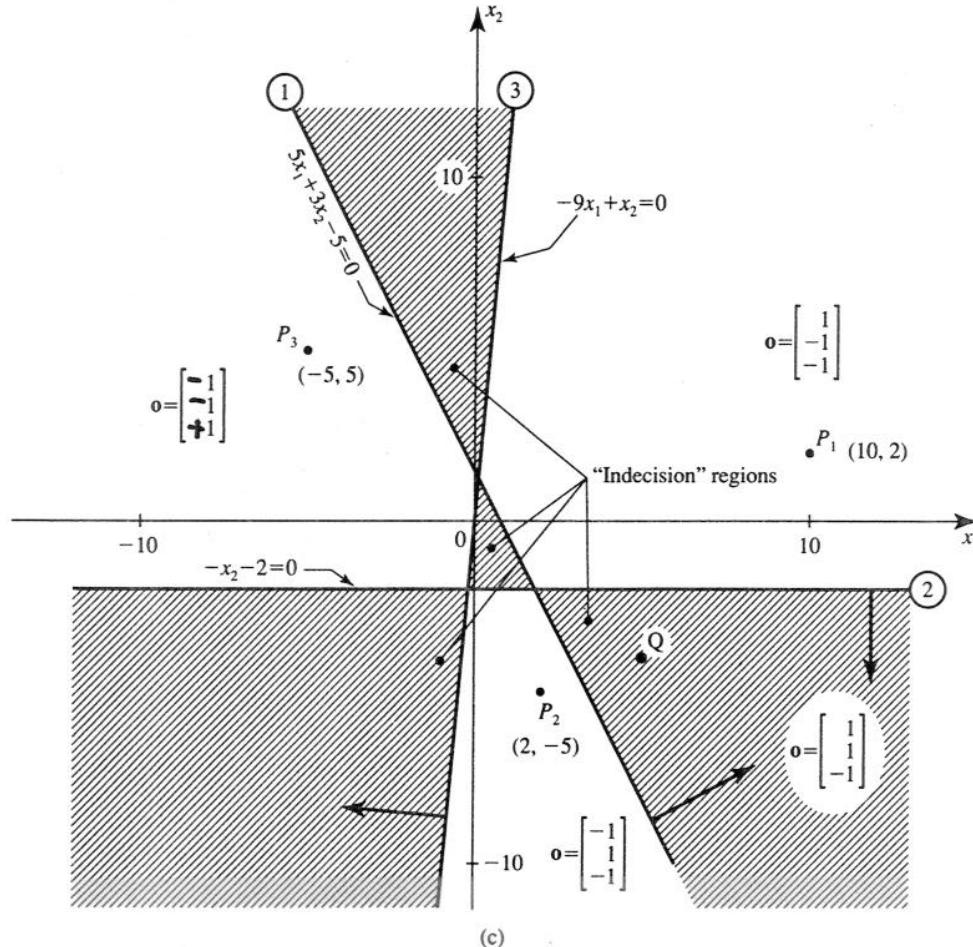
7) Réseau perceptron monocouche

- Exemple: le réseau de perceptrons et régions de décision



7) Réseau perceptron monocouche

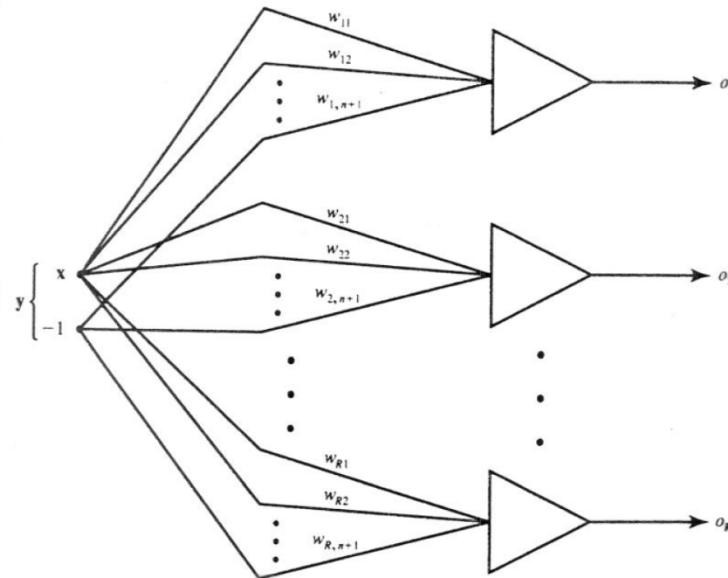
► Exemple: le réseau de perceptrons et régions de décision



7) Réseau perceptron monocouche

Classifieur à R classes – perceptrons continus

- ▶ On peut adapter l'algorithme d'apprentissage du perceptron continu pour un réseau monocouche de R neurones:
 - Architecture d'un classifieur linéaire avec perceptrons continus



7) Réseau perceptron monocouche

Classifieur à R classes – perceptrons continus

► **La règle d'apprentissage:** règle d'apprentissage delta généralisée (avec descente de gradient pour une couche de R neurones)

$$w_i^{k+1} = w_i^k + \frac{1}{2}\eta (d_i^k - o_i^k)(1 - o_i^{k2})y^k$$

pour $i = 1, 2, \dots, R$

- L'ajustement individuel des poids se résume à

$$w_{ij}^{k+1} = w_{ij}^k + \frac{1}{2}\eta (d_i^k - o_i^k)(1 - o_i^{k2})y_j^k$$

pour $j = 1, 2, \dots, n+1$, et $i = 1, 2, \dots, R$