Université des Antilles et de la Guyane **avril 2003**
Pointe à Pitre

RECONNAISSANCE DES FORMES
Méthodes stochastiques

(notes de cours 2HI - version 1.3)

TABLE DES MATIERES

FIGURES

1 GENERALITES

La reconnaissance des formes consiste en une automatisation de tâches de perception artificielle réalisées usuellement par le cerveau et le système sensoriel humain

exemple:

- reconnaître
  - un caractère manuscrit
  - un son, un signal
  - un objet dans une image numérique.

Une forme est une représentation simplifiée du monde extérieur définie sous une forme acceptable par l'ordinateur (par exemple un vecteur de réels, ou bien un mot d'un langage donné, ...).

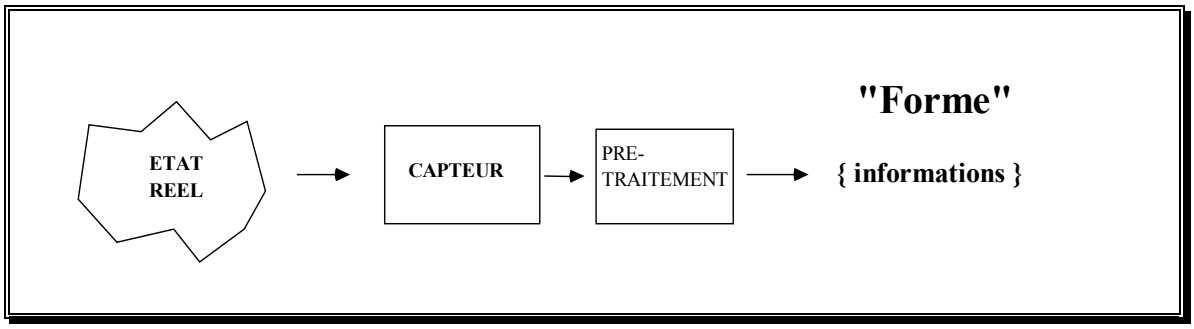


Figure 1 - De la réalité à la « forme »

Un système de reconnaissance des formes peut comporter une phase d'apprentissage qui va consister à « apprendre » à reconnaître des formes sur la base d'échantillons. Lorsque cette phase d'apprentissage sera achevée le système sera alors prêt à fonctionner pour reconnaître des formes inconnues qui lui seront soumises.

Mais un système de reconnaissance des formes peut être aussi un système qui trie (fait des « paquets » homogènes suivant certains critères) un ensemble de formes inconnues. Il n y a alors pas d'apprentissage à proprement parler.

On voit donc qu'il y a de nombreuses approches possibles pour ce problème.

Le schéma général d'un système de reconnaissance des formes avec un phase d'apprentissage est donné par la figure suivante:

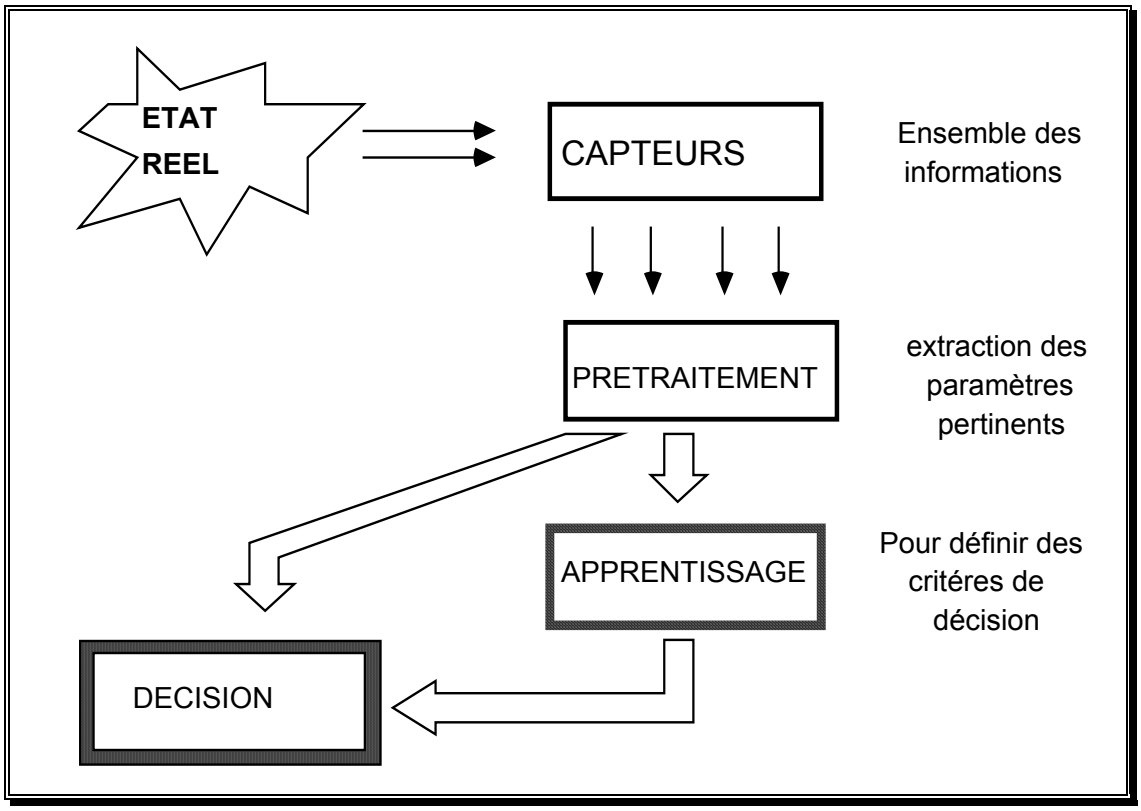


Figure 2 - schéma général d'un système de reconnaissance des formes

1.1 APPRENTISSAGE

Dans le cas d'apprentissage il s'agit en fait de fournir au système un ensemble de formes qui sont déjà connues (on connaît la classe de chacune d'elles). C'est cet ensemble d'apprentissage qui va permettre de « régler » le système de reconnaissance de façon à ce qu'il soit capable de reconnaître ultérieurement des formes de classe inconnue.

Il y a plusieurs cas possibles:

1) - Nombre de classes connu
   - On connaît la classe de chaque forme de l'ensemble d'échantillons

   On parle alors d'apprentissage supervisé (ou avec « professeur »)

2) - Nombre de classes connu ou non
   - On ne connaît pas la classe des échantillons (ce qu'on sait c'est seulement que l'ensemble des échantillons représente ce qu'on doit connaître)

   On parle alors d'apprentissage non supervisé (ou sans « professeur »)

En fait le rôle du module d'apprentissage consiste à caractériser chaque classe par exemple par des relations entre les paramètres définissant la forme

Exemple:

Supposons que des formes soient définies par des vecteurs de R^2 (donc deux paramètres réels !) et supposons disposer d'échantillons de deux classes d'objets .
On peut alors représenter ces échantillons dans l'espace R^2 des paramètres.

L'apprentissage peut alors par exemple consister à trouver automatiquement un courbe séparant les échantillons de chacune des deux classes. C'est la recherche de cette courbe qui va être l'apprentissage des deux classes:

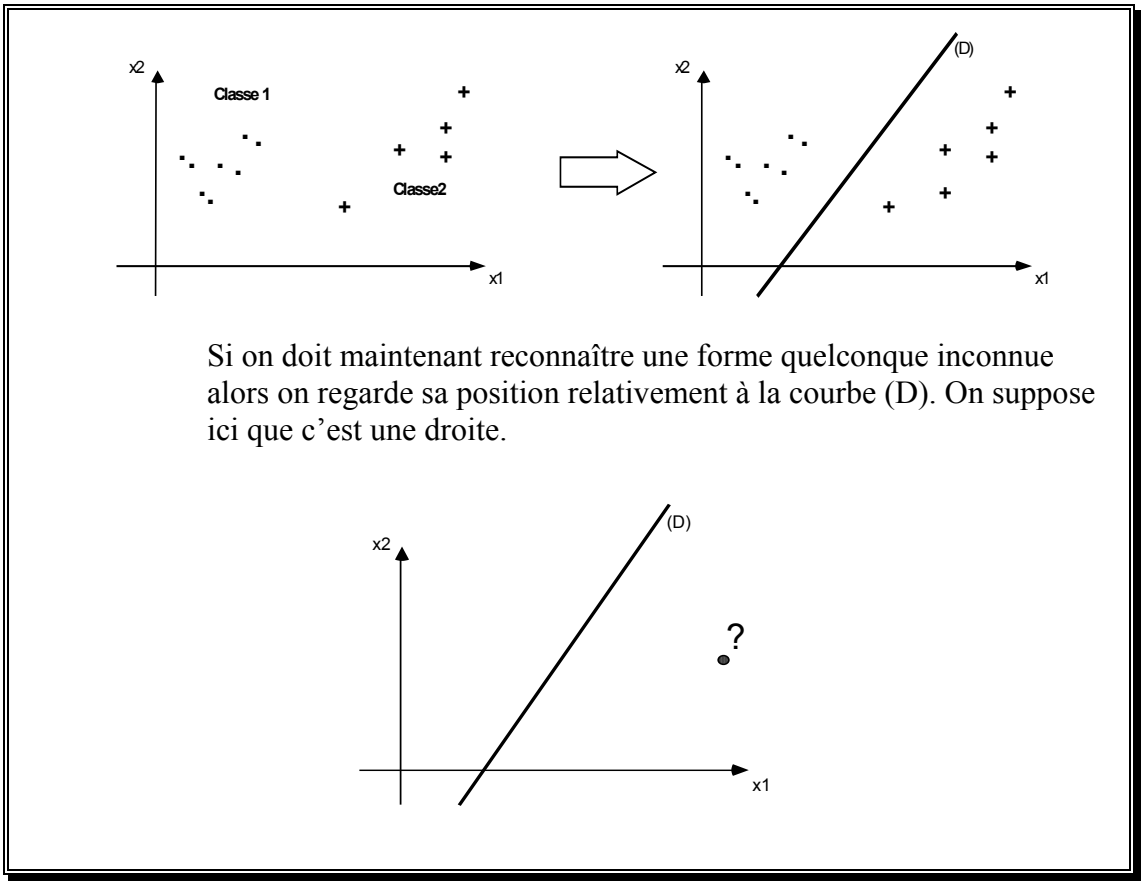


Figure 3 - exemple d'apprentissage et de règle de décision induite

1.2 REMARQUES

L'espace des paramètres définissant une forme n'est pas toujours R^d. Ainsi en reconnaissance syntaxique, une forme peut être définie comme un mot appartenant au langage généré par une grammaire donnée.

L'apprentissage est un processus complexe et des boucles de contrôle peuvent éventuellement être prévues de manière à valider un ensemble d'échantillons d'apprentissage.

En effet les formes échantillons fournies par un utilisateur peuvent être biaisées et/ou non exhaustives (représentation partielle et tronquée de la classe de formes correspondante)

Si on a une idée a priori sur les classes à reconnaître (par exemple une classe sera dite « gaussienne » si toutes les formes de cette classe sont voisines d'une forme moyenne prototype et si la probabilité d'avoir des formes éloignées de la moyenne sont de plus en plus faible avec l'éloignement).

Dans ces cas de figure, on a des processus d'apprentissage simplifiés et quelques formes échantillons permettront de fixer les paramètres d'un modèle de reconnaissance.

1.3 APPROCHE « SYSTEME »

Il existe une approche qualifiée d'approche « système »:

Dans la théorie des systèmes , un système peut être considéré comme une boite noire avec une entrée (la forme) et une sortie (la décision de classification de la forme).

Le système d'apprentissage peut alors être représenté par le schéma suivant:

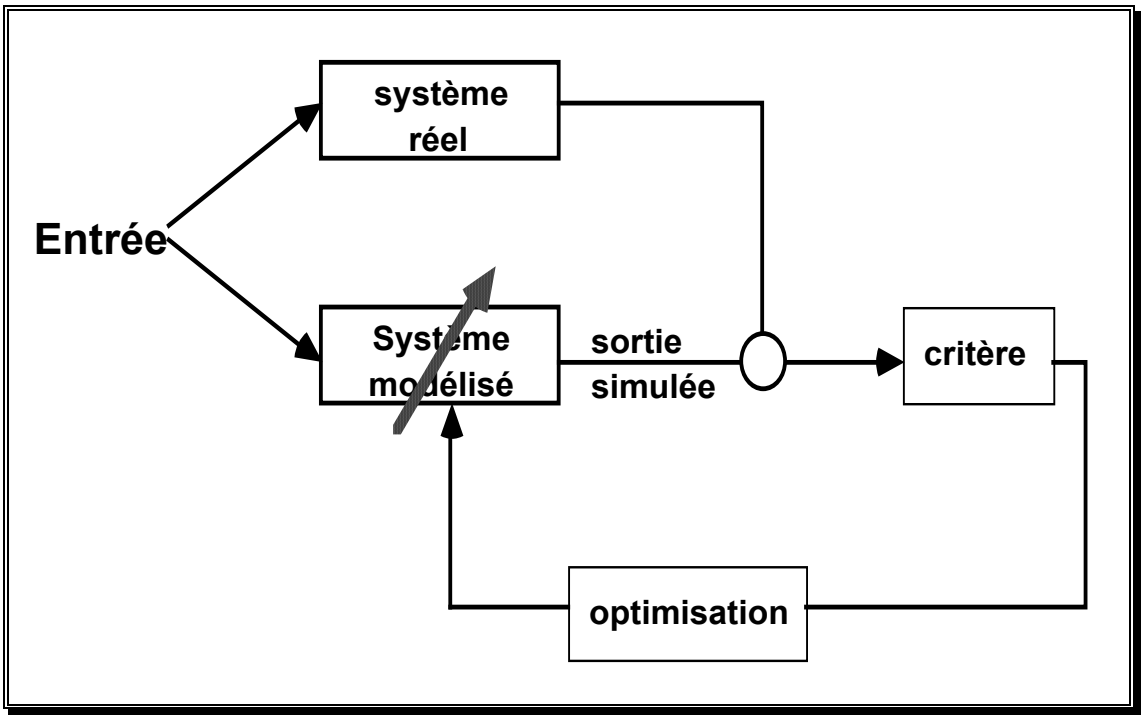


Figure 4 - Approche "système" de la reconnaissance des formes

1.4 DOMAINES D'APPLICATION

1.4.1 Reconnaissance des formes sur signaux temporels

- Signal de parole

  reconnaissance de la parole (qu'a t on dit ?)

  reconnaissance du locuteur (qui a parlé ?)

- Signaux biomédicaux

  ex : électrocardiogramme...

- surveillance d'instruments , diagnostics de panne

1.4.2 Reconnaissance des formes dans les images numériques

- Lecture automatique de caractères

- reconnaissance d'empreintes digitales

- radiographies

- analyse de scènes , interprétation d'images, « computer vision »

Dans ces dernières approches on voit rapidement apparaître la nécessité de coupler les techniques « pures et dures « de la reconnaissance des formes classiques avec les techniques de l'intelligence artificielle...

2 METHODES STOCHASTIQUES

2.1 INTRODUCTION

Dans ces méthodes de reconnaissance des formes ou il va s'agir de regrouper les formes inconnues dans des classes, on va considérer que chacune des classes est régie par un phénomène stochastique (par exemple on dira qu'on s'intéresse à une classe de comportement gaussien : chaque forme de la classe a une certaine probabilité de se produire et la loi de probabilité correspondante est gaussienne)

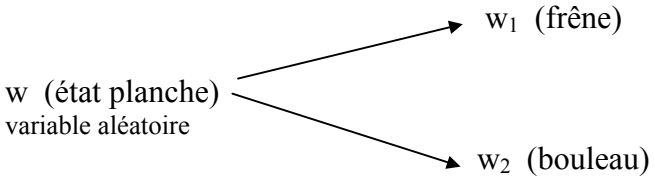
2.2 DECISION BAYESIENNE

La théorie de la décision Bayésienne est la théorie centrale des méthodes stochastiques où les problèmes de décision sont traités en termes de probabilités. Le point central de cette théorie est la règle de Bayes qui permet en fait de choisir l'hypothèse ayant la probabilité la plus élevée.

2.2.1 présentation

Nous allons prendre un exemple tout à fait simple permettant de comprendre le rôle des différents éléments entrant dans la méthode de décision Bayésienne.

Nous allons supposer que l'on s'intéresse à une entreprise de sciage de troncs d'arbres ayant la particularité de ne traiter que des frênes et des bouleaux. Cette entreprise ne sort donc que des planches de frêne et de bouleau. On va alors considérer l'état d'une planche (état = "frêne" ou "bouleau") comme une variable aléatoire w qui peut prendre deux valeurs w_1 et w_2



2.2.1.1 probabilité à priori d'une classe

Supposons maintenant que l'on connaît les quantités respectives de frêne et de bouleau qui rentrent dans la scierie, alors on connaît en fait les proportions de planches de frêne et de bouleau en sortie !

Supposons pour fixer les idées qu'on reçoit 2/3 de frêne et 1/3 de bouleau, alors on peut dire qu'en sortie deux planches sur trois en moyenne en sortie sont du frêne et une planche sur trois est du bouleau.

Si maintenant on doit maintenant décider la classe d'une planche quelconque qui sort et sur laquelle on n'a aucune information (on ne la voit pas , on ne la touche pas , on ne la sent pas ! ...) , on pourra valablement dire que c'est du frêne) et ainsi on minimise la probabilité de se tromper (puisqu'on a deux chances sur trois d'être dans le vrai).

En fait on peut dire qu'on dispose d'une information capitale qui est

- la probabilité à priori de w_1 $p(w_1) = 2/3$
- la probabilité à priori de w_2 $p(w_2) = 1/3$

On voit immédiatement le rôle capital que joue la probabilité à priori de chaque classe.

Si on ne dispose même pas de la probabilité à priori de chacune des classes de formes recherchées alors en général on partagera la probabilité de façon égalitaire sur chacune des classes en concurrence.

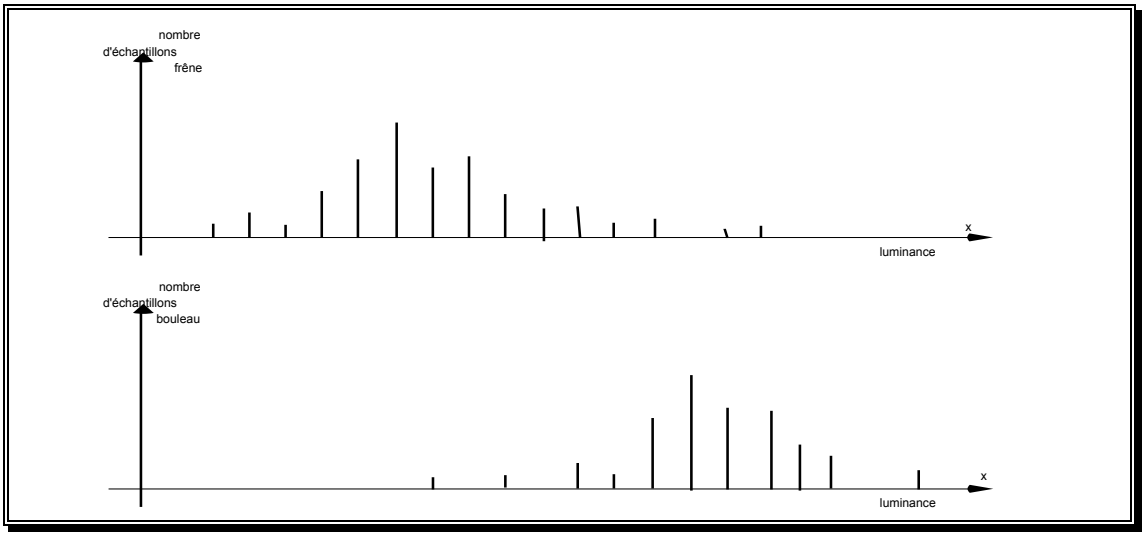
Si on peut réaliser un apprentissage , c'est à dire qu'on vous fournit des échantillons représentatifs de chacune des classes, on peut supposer que ces échantillons sont non seulement représentatifs de chacune des classes mais aussi de l'ensemble des classes , dans ce cas la proportion de chacune des classes dans l'ensemble des échantillons permettra d'obtenir les probabilités à priori des classes.

2.2.1.2 loi de densité de probabilité d'une classe

Nous allons maintenant supposer que la scierie a installé une caméra N/B capable de saisir une image numérique de la surface de chaque planche qui sort et que cette image est traitée par un ordinateur qui peut fournir pour chaque image sa luminance moyenne. Ce paramètre étant considéré a priori comme relativement caractéristique de chaque essence de bois.

A partir d'échantillons (c'est à dire des planches qui sortent pour lesquelles on connaît la classe (w_1 ou w_2) et en même temps la luminance moyenne) on va chercher à estimer comment se distribue la luminance pour chacune des classes.

supposons avoir obtenu les histogrammes suivants:



A partir de là on peut obtenir les fonctions de densité de probabilité de chacune des classes $P(x/w_1)$ et $p(x/w_2)$ représentées ci dessous (suivant une méthode vue plus tard):

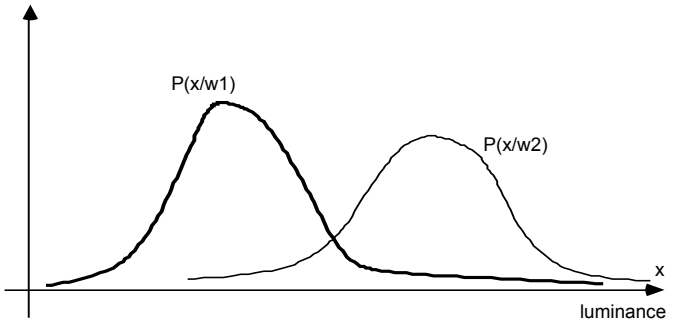


Figure 5 - Histogrammes des échantillons et densités de probabilité des classes

Notons que

$$\int_0^{+\infty} P(x/w_1) \times dx = 1 \quad et \quad \int_0^{+\infty} P(x/w_2) \times dx = 1$$

2.2.1.3 règle de Bayes et décision Bayésienne

La règle de Bayes permet de calculer ce qu'on appelle les probabilités à posteriori des classes , c'est à dire $P(w_1/x)$ et $P(w_2/x)$ à partir des probabilités à priori des classes $P(w_1)$ et $P(w_2)$ et des fonctions de densité des probabilités $p(x/w_1)$ et $p(x/w_2)$.

$$P(w_i/x) = \frac{P(x/w_i) \times P(w_i)}{P(x)} \quad \forall i = 1,2 \ \ avec \ \ P(x) = \sum_{j=1}^{2} P(x/w_j) \times P(w_j)$$

L'interprétation de $P(w_1/x)$ est la suivante: ayant obtenu une forme (décrite par une valeur x de luminance) $P(w_1/x)$ donne la probabilité d'avoir alors la classe w_1.

Dans notre exemple , avec les lois de densité de probabilité obtenues et les probabilités à priori données précédemment on obtient approximativement les densités de probabilité à posteriori suivantes

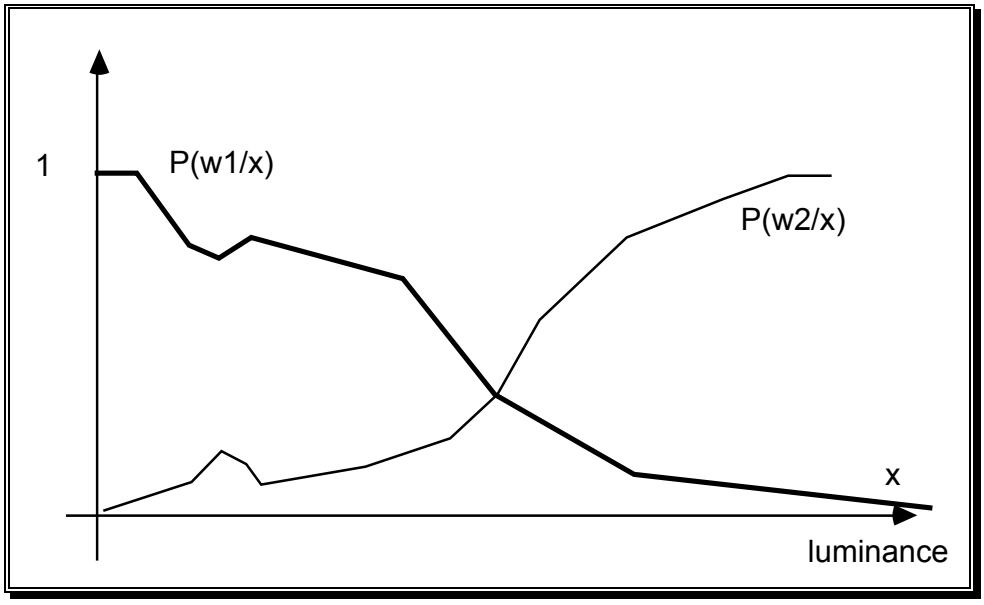


Figure 6 - densités de probabilité à posteriori des classes

La règle de décision Bayésienne est alors pratiquement immédiate:

Ayant remarqué une forme inconnue x (la forme est assimilée au paramètre qui la caractérise, ici un réel qui est la luminance moyenne de la planche) on dira

que la forme inconnue x est de la classe w_1 si $P(w_1/x) > P(w_2/x)$
sinon elle est de la classe w_2 Soit sous une forme simplifiée:

REGLE DE DECISION BAYESIENNE

Ayant observé x on décide :

w_1 si $P(w_1/x) > P(w_2/x)$
w_2 sinon

ou encore

w_1 si $P(x/w_1)xP(w_1) > P(x/w_2)xP(w_2)$

Dans cette règle on voit de façon particulièrement claire qu'il est nécessaire et suffisant de connaître pour toutes les classes cherchées la probabilité à priori et la loi de densité de probabilité de cette classe.

Justification de cette règle:

En fonction de la décision prise pour classer une forme x on augmentera ou diminuera la probabilité d'erreur (probabilité de se tromper dans le choix de la classe d'affectation pour x)

Ainsi ayant une forme x on peut dire P(erreur/x) = $P(w_1/x)$ si on décide w_2
$P(w_2/X)$ si on décide w_1

D'emblée on note que le bon choix est celui qui minimise la probabilité d'erreur.
L'erreur a en effet une probabilité minimale si on décide w_1 et que $P(w_1/x) > P(w_2/x)$
ou w_2 et que $P(w_2/x) > P(w_1/x)$

La probabilité moyenne d'erreur est donnée par la formule suivante

$$P(erreur) = \int_0^{+\infty} P(erreur / x) \times P(x) \times dx$$

Si pour toutes les valeurs possibles de x (luminance) on a P(erreur/x) minimum alors on est sûr que P(erreur) est minimal.

La règle de décision Bayésienne minimise donc à coup sûr la probabilité de se tromper.

2.2.1.4 Le choix des paramètres pour caractériser une forme

Nous donnons ici quelques éléments sur le problème du choix des paramètres pour caractériser une forme.

Dans l'exemple que nous avons pris nous avons décidé chaque forme (une planche qui sort de la scierie) était définie par sa luminance moyenne. Mais si nous regardons les lois de distribution des deux classes nous voyons que les deux classes ne sont pas bien séparées par ce seul paramètre puisque des échantillons eux mêmes seront mal classés par la règle de décision Bayésienne (même si on minimise la probabilité d'erreur , les erreurs seront dans ce cas de figure relativement nombreuses.

Supposons que l'on soit capable à partir de l'image fournie par la caméra de calculer le grain du bois (paramètre de texture), alors la forme sera maintenant définie par un vecteur de R^2

(x_1,x_2) avec x_1 :luminance moyenne
x_2 : grain

Si on a la répartition suivante pour les échantillons des classes on voit que le classifieur ne pourra être que plus performant

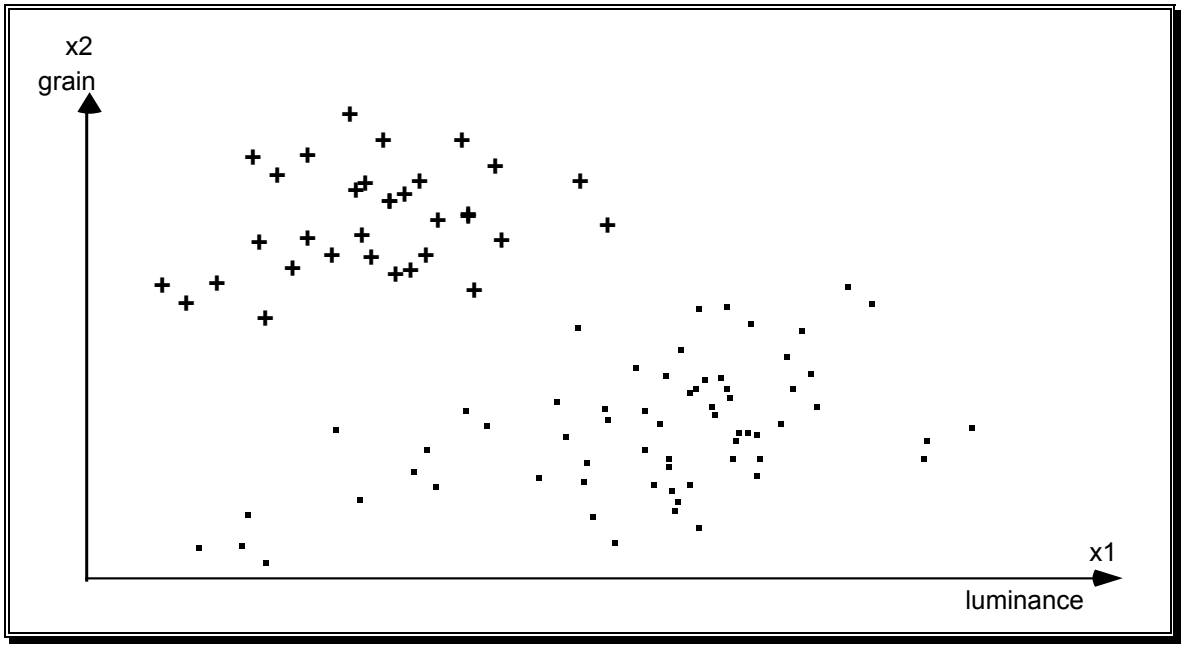


Figure 7 - choix des paramètres pour caractériser une forme

On voit qu'en projection sur l'axe x_1 (luminance) on retrouve les histogrammes et les lois de densité de probabilité initiales, mais que l'ajout d'un paramètre améliorera ici à coup sur le résultat de la classification par la règle de Bayes.

Mais les lois de densité de probabilités $P(x/w_1)$ et $P(w_1/x)$ sont maintenant des lois multidimensionnelles.

2.2.2 théorie de la decision bayesienne

Dans ce paragraphe nous allons aborder la décision Bayésienne dans le cas le plus général, puis nous définirons dans ce cadre la notion de fonctions discriminantes qui seront utilisées largement par la suite.

2.2.2.1 Formalisation de la décision Bayésienne dans le cas général

Nous allons en fait généraliser l'exemple élémentaire de la "scierie".

- plus de 1 paramètre de mesure
  (la forme est définie par un vecteur de paramètres (élément de R^d)

- Plus de deux classes (mais on suppose que les classes définies regroupent tous les états possibles des formes, c'est à dire que toute forme appartient obligatoirement à l'une des classes définies. Si ce n'est pas le cas on crée une classe supplémentaire dite de « rejet » pour tomber dans cette supposition)

- Les différents types d'erreurs lorsqu'on classe suivent des lois dépendantes des décisions

Soient les définitions suivantes:

$W=\{w_1,w_2,.....,w_s\}$ l'ensemble des s classes d'appartenance possibles

$A=\{a_1,a_2, ...,a_k\}$ l'ensemble des k actions (décisions) possibles.
En général a_i est la décision de classer dans la classe w_i

$\lambda(a_i,w_j)$ coût de la décision a_i quand la forme appartient à la classe w_j

$x= \{x_1,x_2, ... , x_d\}$ vecteur de paramètres décrivant une forme

$P(x/w_j)$ fonction de densité de probabilité de la classe w_j

$P(w_j)$ probabilité à priori de la classe w_j

alors,

$$P(w_j / x) = \frac{P(x / w_j) \times P(w\ )}{P(x)} \qquad avec \qquad P(x) = \sum_{k=1}^{s} P(x / w_k) \times P(w_k)$$

Le coût d'une décision a_i est égal à:

$$R(a_i / x) = \sum_{j=1}^{s} \lambda(a_i / w_J) \times P(w_j / x)$$

Cette valeur est appelée le risque conditionnel.

La règle de décision qui à une forme x va faire correspondre une décision a(x) ($a(x)= a_1$ ou a_2 .....ou a_s) devra minimiser le risque global R avec

$$R = \int R(a(x) / x) \times p(x) \times dx$$

ce qui est réalisé notamment si R(a(x)/x) est minimum pour tout vecteur x.

D'où la règle de décision Bayésienne:

Etant donné un vecteur x caractérisant une forme

Pour i= 1,2, ... k
calculer le risque conditionnel

$$R(a_i / x) = \sum_{j=1}^{s} \lambda(a_i / w_J) \times P(w_j / x)$$

et choisir la décision a_i telle que $R(\alpha_i/x)$ soit minimum

2.2.2.1.1 Cas particulier de deux classes

Nous considérons ici l'approche générale dans le cas où le nombre total de classes est de deux (et on suppose qu'il n'y a pas de classe de rejet.

action a_1 --> décision w_1 , action a_2 --> décision w_2

Soit $\lambda_{ij}=\lambda(a_i/w_j)$ coût de la décision w_i quand la vérité est w

Alors

$$R(a_1/x)=\lambda_{11} P(w_1/x) + \lambda_{12} P(w_2/x)$$
$$R(a_2/x)=\lambda_{21} P(w_1/x) + \lambda_{22} P(w_2/x)$$

avec $\lambda_{11} < \lambda_{12}$ car la décision qui correspond
$\lambda_{22} < \lambda_{21}$ à la vérité doit coûter moins cher !

Par exemple on décide w_1 si $R(a_1/x) < R(a_2/x)$

soit si

$$(\lambda_{21}-\lambda_{11}) P(w_1/x) > (\lambda_{12}-\lambda_{22}) P(w_2/x)$$

soit

$$(\lambda_{21}-\lambda_{11}) P(x/w_1) P(w_1) > (\lambda_{12}-\lambda_{22}) P(x/w_2) P(w_2)$$

Soit

on décide w_1 si $$\boxed{\frac{P(x/w_1)}{P(x/w_2)} > \frac{(\lambda_{12}-\lambda_{22})}{(\lambda_{21}-\lambda_{11})} \times \frac{P(w_2)}{P(w_1)}}$$

Ce rapport s'appelle le rapport de vraisemblance et en fait on choisit w1 si le rapport de vraisemblance est supérieur à un seuil (donné par le deuxième terme de l'inéquation.

Dans le cas où on prend $\lambda_{12} = \lambda_{21} = 1$ et $\lambda_{11} = \lambda_{22} = 0$ alors on retombe sur ce qu'on avait vu dans la scierie !

2.2.2.1.2 classification par taux d'erreur minimum

Où ... , sous certaines hypothèses on retrouve la règle de Bayes simple!....

On suppose que pour tout $i = 1.....s$ on a action a_i --> décision w_i

et qu'on applique une fonction de coût symétrique comme dans le paragraphe précédent.

$$\lambda(a_i/w_j) = \begin{cases} 0 & \text{si } i=j \\ 1 & \text{si } i \neq j \end{cases} \quad \text{pour } i,j = 1.....s$$

alors

$$R(a_i / x) = \sum_{j=1}^{s} \lambda(a_i / w_j) \times P(w_j / x)$$

$$= \sum_{j \neq i} P(w_j / x) = 1 - P(w_i / x)$$

Soit minimiser $R(a_i/x)$ équivaut à maximiser $P(w_i/x)$

c'est à dire

..... décider w_i ssi $P(w_i/x) > P(w_j/x)$ pour tout $j \neq i$

2.2.2.2 Fonctions discriminantes et classifieurs bayésiens

Pour disposer d'une règle de décision on va supposer que chaque classe recherchée est caractérisée par une fonction discriminante:

$g_i(x)$ pour $i=1, \ldots, s$

et les s fonctions discriminantes sont telles que

une forme x (vecteur de R^d) est classée dans la classe w_i si $g_i(x) > g_j(x)$ pour tout $j \neq i$

Le classifieur obtenu calcule donc pour un vecteur x donné les fonctions discriminantes puis cherche le maximum qui permet de choisir la classe. Ce classifieur peut être schématisé de la façon suivante:

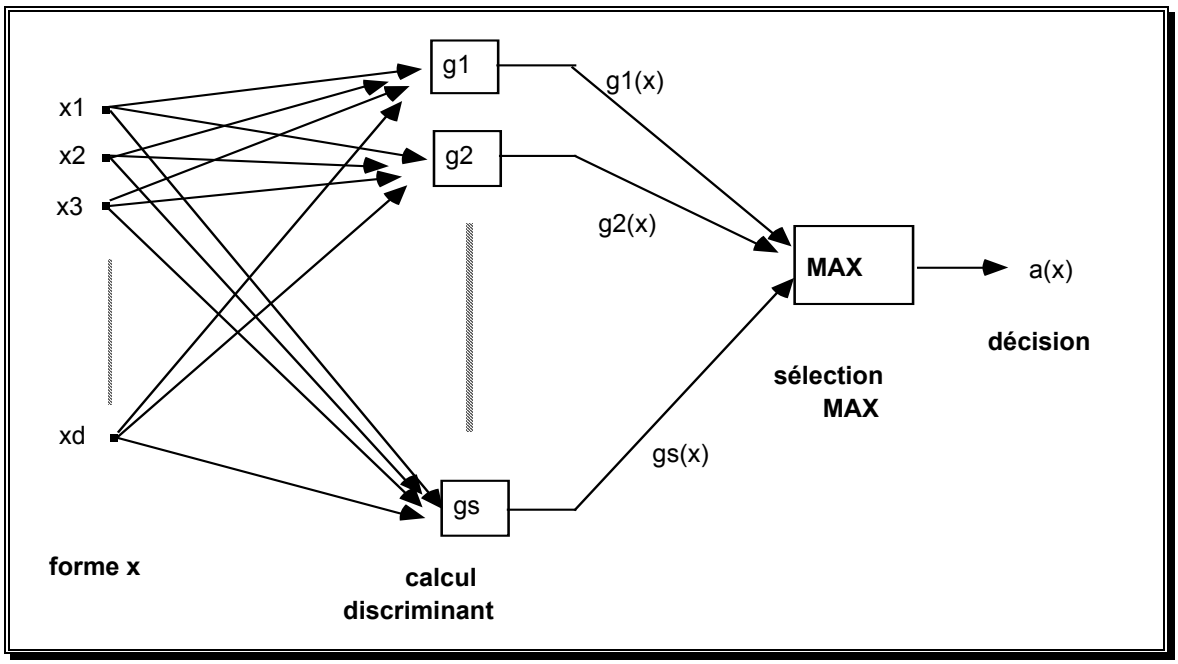


Figure 8 - fonctions discriminantes et décision

2.2.2.2.1 Fonctions discriminantes et décision Bayésienne

En partant des formulations vues en décision Bayésienne, on peut définir facilement une panoplie de fonctions discriminantes.

Ainsi les fonctions suivantes répondent aux critères d'une fonction discriminante

> $\mathbf{g_i(x) = -R(\alpha_i/x)}$ car un maximum pour g_i correspond à un minimum du coût

> $\mathbf{g_i(x) = P(w_i/x)}$

> $\mathbf{g_i(x) = P(x/w_i) \times P(w_i)}$

Ensuite

si $g_i(x)$ est une fonction discriminante alors

> f ($g_i(x)$) si f est une fonction croissante et monotone

Voici donc quelques fonctions discriminantes classiques extraites de la théorie de la décision Bayésienne:

$$g_i(x) = P(w_i / x)$$

$$g_i(x) = \frac{P(x / w_i) \times P(w_i)}{\sum_{j=1}^{s} P(x / w_j) \times P(w_j)}$$

$$g_i(x) = P(x / w_i) \times P(w_i)$$

$$g_i(x) = \mathbf{log}\left[P(x / w_i)\right] + \mathbf{log}\left[P(w_i)\right]$$

2.2.2.2.2 Séparation de l'espace des paramètres en régions

L'utilisation des s fonctions discriminantes pour réaliser la classification de formes amène à découper l'espace des observations (espace des paramètres x) en s régions R_1 R_2 ...R_s.

Chaque région (qui n'est pas forcément connexe) est définie de la façon suivante:

Si $g_i(x) > g_j(x)$ pour tout $j \neq i$ alors x appartient à R_i
et on assigne x à w_i

On peut alors définir des frontières de décision :

Si deux régions Ri et Rj sont contiguës alors l'équation de la frontière est définie par

$g(x) = (g_i(x)-g_j(x)) = 0$

Exemples

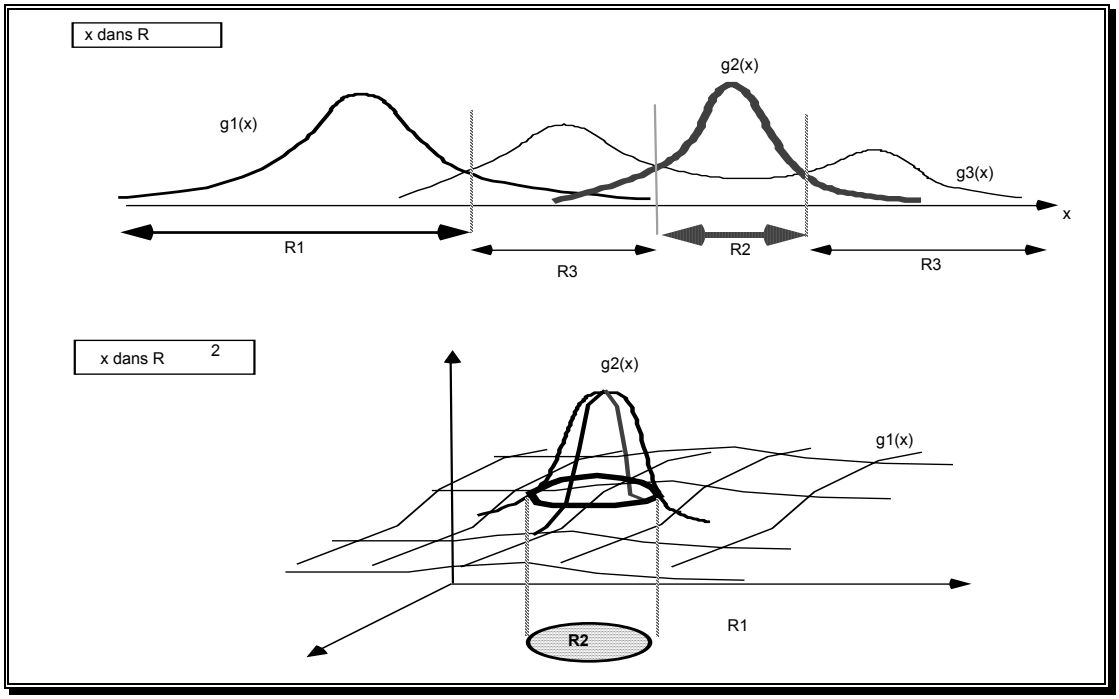


Figure 9 - Frontières de décision dans l'espace des observations

2.2.2.2.3 Fonctions discriminante : cas de deux classes

Dans ce cas on peut calculer $g(x)=g_1(x)-g_2(x)$

La décision est alors définie par x est de w1 si $g(x) > 0$

exemples:

$$g(x)=P(w_1/x) - P(w_2/x)$$

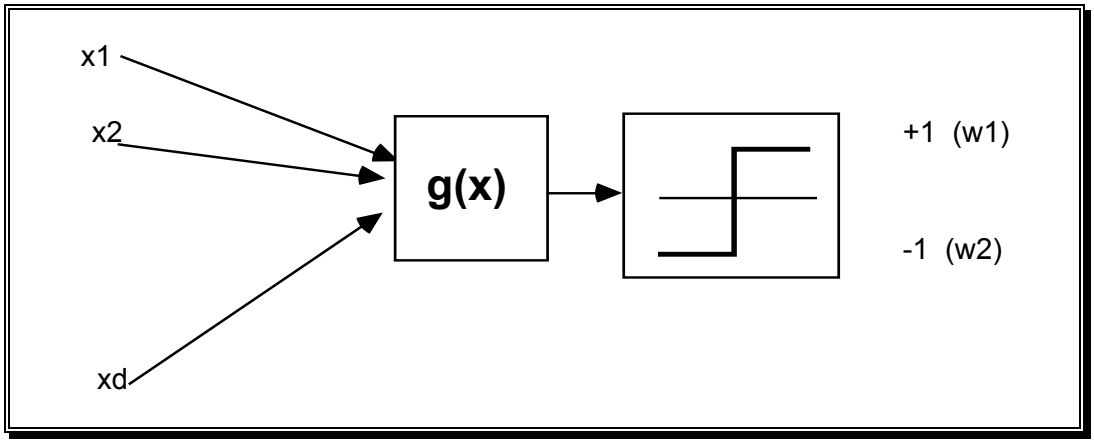


Figure 10 - fonctions discriminantes dans le cas de deux classes

2.2.2.2.4 probabilité d'erreur et fonctions discriminantes bayésiennes

Voyons d'abord le cas de deux classes et du classifieur bayésien:

On a donc deux régions R1 et R2

Le classifieur se trompe dans les cas suivant:

- x tombe dans R1 mais en fait x appartient à w2
- x tombe dans R2 mais en fait x appartient à w1

On a donc

$$p(erreur/x) = P(x \in R_2, w_1) + P(x \in R_1, w_2)$$
$$= P(x \in R_2 / w_1) \times P(w_1) + P(x \in R_1 / w_2) \times P(w_2)$$

Soit en calculant la probabilité d'erreur sur toutes les valeurs de x

$$P(erreur) = \int_{R_1} p(x/w_1) \times P(w_1) \times dx + \int_{R_2} p(x/w_2) \times P(w_2) \times dx$$

Exemple sur l'espace d'observation R avec deux classes « gaussiennes »:

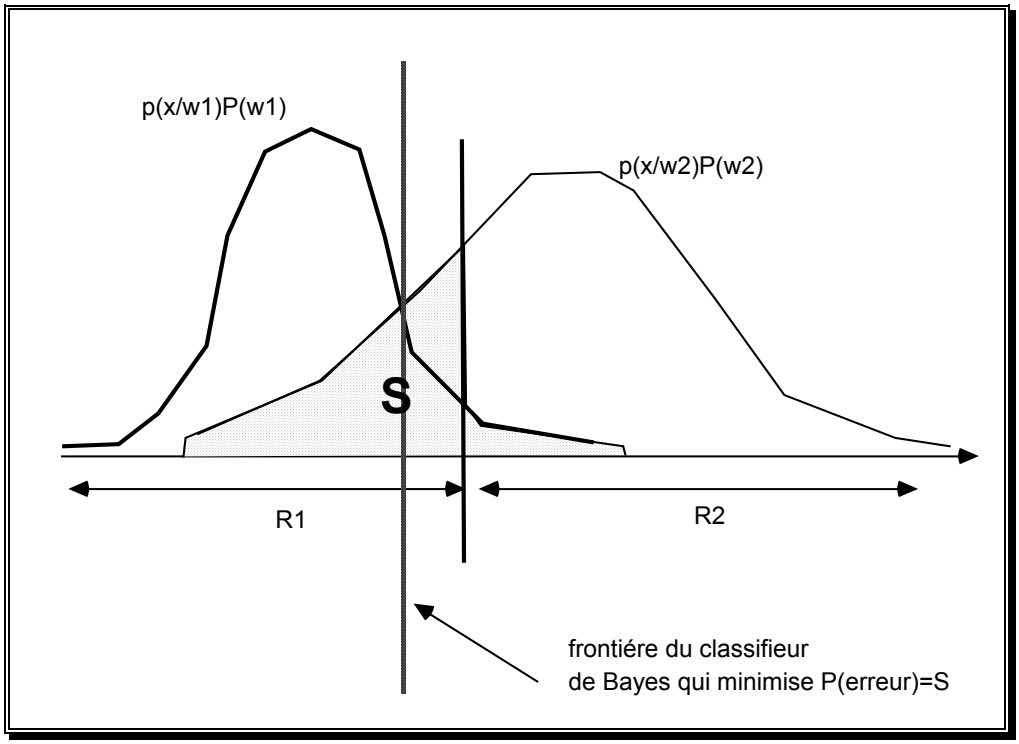


Figure 11 - Minimisation de la probabilité d'erreur

On peut aussi voir le problème par une maximisation de probabilité de classification correcte.

Ainsi si on a s classes possibles on a

$$p(correct / x) = \sum_{i=1}^{s} p(x \in R_i, w_i) = \sum_{i=1}^{s} p(x \in R_i / w_i) \times P(w_i)$$

ce qui donne

$$\boxed{P(correct) = \sum_{i=1}^{s} \int_{R_i} p(x / w_i) \times P(w_i) \times dx}$$

Le classifieur de Bayes maximise cette probabilité

2.3 LOI NORMALE ET DECISION BAYESIENNE

2.3.1 Introduction

Nous avons vu qu'un classifieur bayésien est parfaitement défini quand on connaît pour chaque classe

$P(x/w_i)$ et $P(w_i)$

La loi de distribution $p(x/w_i)$ peut être aussi complexe qu'on le veut , mais bien sûr les temps de calcul risquent d'être sérieusement élevés.

C'est pourquoi il est souvent possible de faire des hypothèses sur cette loi de densité de probabilité qui décrit en fait la distribution des formes d'une classe particulière sur l'espace des observations (vecteur x).

Une hypothèse remarquable et souvent prise comme une vérité fondamentale est de supposer que les classes sont gaussiennes.

La loi de densité de probabilité de la classe w_i , $p(x/w_i)$ est alors une loi normale définie de façon analytique par quelques paramètres.

Cette hypothèse apparaît fondée quand pour une classe donnée les vecteurs x définissant chaque forme sont répartis de façon continue et « harmonieuse » autour d'un vecteur prototype moyen μ.

2.3.1.1 Définition d'une loi normale

Voyons d'abord le cas où le vecteur x est en fait un nombre réel. La loi normale est dite « à une dimension »

Pour simplifier les écritures nous noterons la loi de densité de probabilité de la classe w_i $p(x/w_i)$ sous la forme $p(x)$ quand il n y aura pas de confusions possibles.

La loi normale notée $N(\mu,\sigma)$ s'écrit

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \times e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

$$avec \quad \mu = E(x) = \int_{-\infty}^{+\infty} x \times p(x) \times dx$$

$$et \quad \sigma^2 = E\left((x-\mu)^2\right) = \int_{-\infty}^{+\infty} (x-\mu)^2 \times p(x) \times x$$

Figure 12 - Loi normale à une dimension

Voyons maintenant le cas où le vecteur x est dans R^d

alors on a la loi normale multivariée notée $N(\mu, \sum)$

$$p(x) = \frac{1}{(2\pi)^{d/2} \times |\Sigma|^{1/2}} \times e^{\left[-\frac{1}{2}(x-\mu)^t \times \Sigma^{-1} \times (x-\mu)\right]}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_d \end{bmatrix} \qquad \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \cdot \\ \cdot \\ \mu_d \end{bmatrix}$$

Σ : matrice de covariance

$|\Sigma|$: déterminant

Note: Pour bien comprendre la notion de matrice de covariance lire l'annexe de ce polycopié sur l'analyse en composantes principales

Méthode d'estimation de μ et $\sum$:

A partir d'échantillons d'une classe, il est possible d'estimer les paramètres μ et $\sum$ de la loi normale suivie par cette classe.

si on a donc n observations de formes de la classe, vecteurs $x=(x_1\ x_2\ ...\ x_d)^t$

alors on a

$$\mu_i = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\sum = \left[\sigma_{kl}\right]_{\substack{k=1...d \\ l=1....d}}$$

$$\sigma_{kl} = \frac{1}{n}\sum_{x} (x_k - \mu_k) \times (x - \mu_l)$$

La matrice $\sum$ est symétrique et définie positive
(sauf cas ou deux composantes du vecteur x sont systématiquement identiques ou bien si une composante est à variance nulle (donc composante constante)).

La connaissance de la matrice de covariance d'une classe de formes emmagasine toute l'information sur la façon dont se répartissent les formes de la classe dans l'espace des observations (ici R^d).

Ainsi soit a un vecteur unité (||a||=1) définissant une direction dans R^d

alors $\mathbf{y = a^t x}$ est la projection de x suivant a

et $\mathbf{a^t \sum a}$ représente la variance de la projection de x suivant a

Si on considère les n échantillons définissant la classe alors ces échantillons forment un « nuage » de points dans R^d de centre μ et de « forme » « $\sum$ ».

Si on veut représenter dans l'espace R^d tous les x qui ont une probabilité identique fixée de se produire cela correspond à un p(x) constante donné et donc à

$$\mathbf{(x-\mu)^t \sum^{-1} (x-\mu)=Cte}$$

Ce qui est l'équation d'un ellipsoïde dans R^d

Chacun des ellipsoïdes répondant à une équation de ce type représente un ensemble de points équiprobables pour la classe.

Les vecteurs propres de la matrice $\sum$ représentent les axes principaux du nuage de points échantillons.

Toutes ces notions sont abordées dans l'annexe sur l'analyse en composantes principales.

On peut alors définir une distance entre une forme x et une classe représentée par sa moyenne μ, c'est la distance de Mahanalobis:

Exemple dans le cas de R^2:

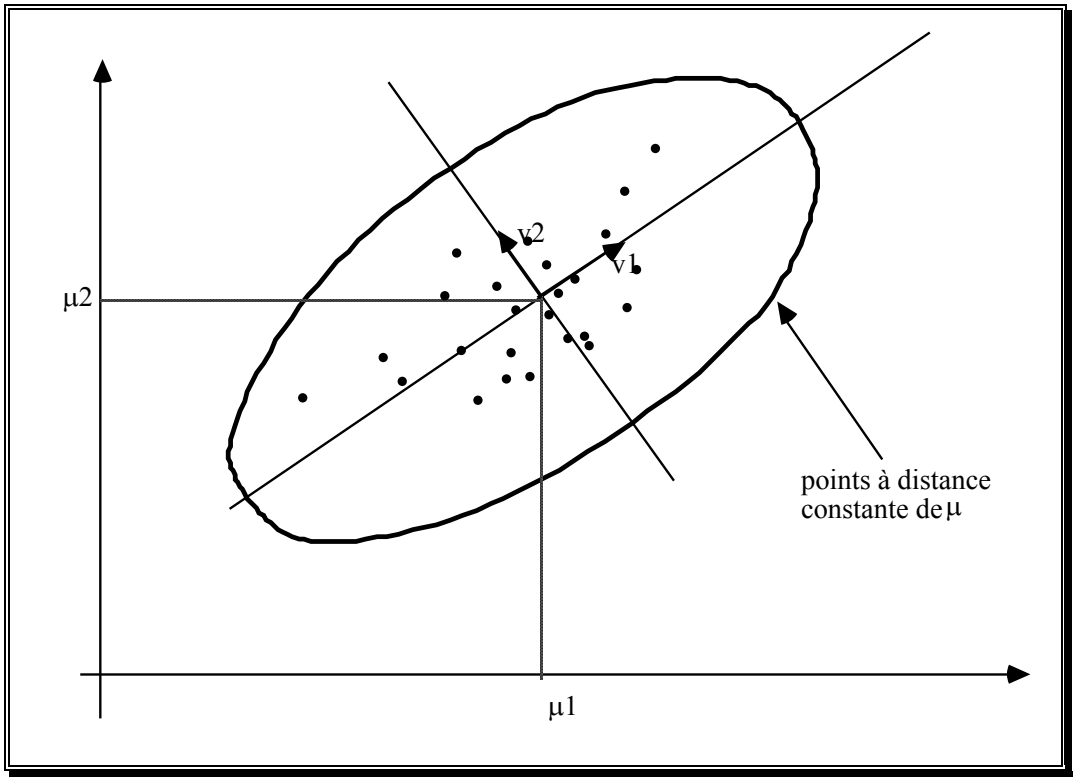


Figure 13 - Distance de Mahanalobis dans R^2

Eparpillement des échantillons autour de la moyenne:

-> volume de l'hyper-ellipsoide défini par une distance de Mahanalobis de r

$$V = V_d \times |\Sigma|^{-1/2} \times r^d$$

avec V_d volume de la sphère unité dont la valeur est définie comme suit:

d pair --> $V_d = \dfrac{\pi^{d/2}}{(d/2)!}$, **d impair** --> $V_d = \dfrac{2^d \times \pi^{(d-1)/2} \times \left(\dfrac{d-1}{2}\right)!}{d!}$

On voit que l'éparpillement des échantillons de la classe autour de la moyenne est proportionnel à $|\sum|^{1/2}$

2.3.1.2 Fonctions discriminantes pour la loi normale

Possibilité de fonction discriminante

$g_{wi}(x)=\log(p(x/w_i))+\log(P(w_i))$

Si on fait hypothèse que la classe w_i suit une loi normale $N(\mu_{wi},\sum_{wi})$

alors:

$$g_{w_i}(x) = -\frac{1}{2}\left(x-\mu_{w_i}\right)^t \times \sum_{w_i}^{-1} \times \left(x-\mu_{w_i}\right) - \frac{d}{2}\log(2\pi) - \frac{1}{2}\log\left|\Sigma_{w_i}\right| + \log(p(w_i))$$

Cette fonction discriminante relativement complexe peut éventuellement se simplifier suivant les hypothèses restrictives faites sur les classes recherchées.

En effet chaque classe w_i dispose de sa fonction discriminante $g_{wi}(x)$, mais si dans chacune de ces fonctions des termes sont indépendants de w_i alors on pourra les éliminer de l'expression de toutes les fonctions $g_{wi}(x)$, ce qui ne changera rien à la méthode de décision induite.

Voici différents cas possibles:

nous appellerons chaque composante du vecteur x un paramètre.

- $\sum_{wi}= \sigma^2$ I pour toutes les classes,

  I est la matrice unité de taille dxd $\begin{vmatrix} 1 & 0 & . & 0 \\ 0 & 1 & . & . \\ . & . & . & 0 \\ 0 & . & 0 & 1 \end{vmatrix}$

chaque paramètre a même variance et tous les paramètres sont totalement indépendants et ceci pour toutes les classes.

>> $\sum_{wi}=\sum$ pour tous les w_i

Pour chaque classe les matrices de covariance sont identiques.

>>> $\sum_{wi}$ est quelconque et dépend de chaque classe

la seule hypothèse restrictive sur les classes est ici le fait que chaque classe suit une loi gaussienne (mais chaque classe suit une loi spécifique).

2.3.1.2.1 paramètres indépendants et de variances identiques

Dans ce cas on a $\sum_{wi} = \sigma^2 I$ pour toutes les classes

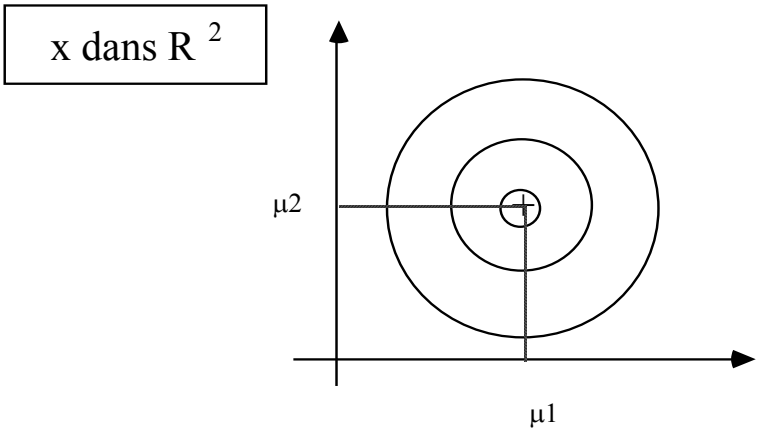
$$x = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_d \end{bmatrix} \quad \textbf{et} \quad \sum_{wi} = \begin{bmatrix} \sigma^2 & 0 & . & 0 \\ 0 & \sigma^2 & . & . \\ . & . & . & 0 \\ 0 & . & 0 & \sigma^2 \end{bmatrix}$$

Pour toutes les classes on a donc

- x_i et x_j statistiquement indépendants (en d'autres termes étant donné x_i on ne peut absolument pas prédire x_j) pour tout i et tout j

- chaque paramètre (pour chaque classe!) a une variance de σ^2

La forme de chaque classe est du type hyper-sphére



$$\left|\sum_{w_i}\right| = \sigma^{2d} \quad \textbf{et} \quad \sum_{w_i}^{-1} = \frac{1}{\sigma^2} \times I$$

et donc

$$\left|\sum_{w_i}\right| \quad et \quad \frac{d}{2} \times \textbf{log}(2\pi)$$ sont indépendants de w_i

On en déduit que l'on peut choisir comme fonction discriminante pour chaque classe

$$g_{w_i}(x) = -\frac{\left\|x - \mu_{w_i}\right\|^2}{2\sigma^2} + \log\left(P(_{w_i})\right)$$

avec la norme Euclidienne

$$\left\|x - \mu_{w_i}\right\|^2 = \left(x - \mu_{w_i}\right)^t \times \left(x - \mu_{w_i}\right)$$

--> Si on fait hypothèse supplémentaire que les classes sont équiprobables (tous les $P(w_i)$ sont égaux) alors les fonctions discriminantes se simplifient encore puisqu'on peut obtenir:

$$g_{w_i}(x) = -(x - \mu_{w_i})^t \times (x - \mu_{w_i})$$

La règle de décision devient alors,

Pour une observation x , calculer sa distance Euclidienne à chaque centre de classe μ_{w1} , μ_{w2}, ...μ_{ws} et l'assigner à la classe la plus proche.

Ce classifieur s'appelle classifieur de la distance minimum. On voit que les classes sont parfaitement représentées par une seule forme , une forme prototype. Ceci est très rarement satisfaisant !

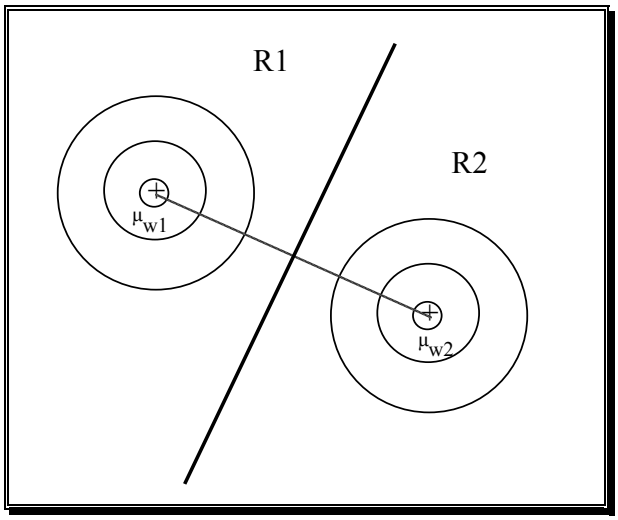


Figure 14 - classifieur de la distance minimum pour deux classes

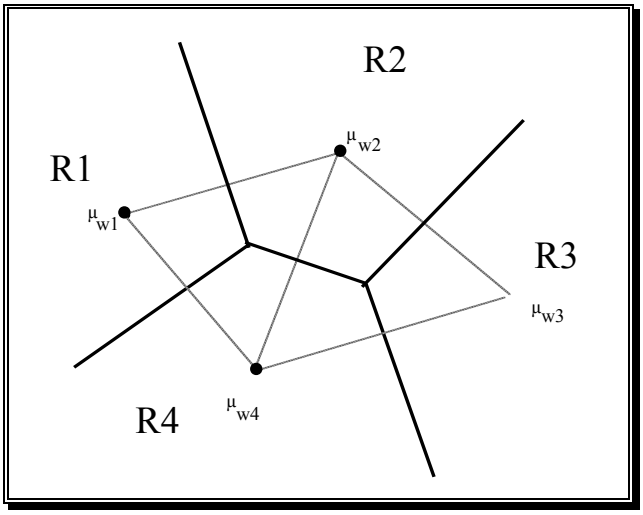


Figure 15 - classifieur de la distance minimum pour quatre classes

--> **Si les $P(w_i)$ sont distincts**

Après développement de la formule on obtient:

$$g_{w_i}(x) = -\frac{1}{2\sigma^2}\left[x^t x - 2\mu_i{}^t x + \mu_i{}^t \mu_i\right] + \mathbf{log}(P(w_i))$$

où $x^t x$ est bien sûr indépendant de w_i, on peut donc le supprimer dans toutes les fonctions discriminantes.

On obtient donc après un petit calcul sympa:

$$g_{w_i}(x) = K_i{}^t x + K_{i0}$$

$$avec \quad K_i = \frac{1}{\sigma^2} \times \mu_{w_i} \quad et \quad K_{i0} = -\frac{1}{2\sigma^2} \times \mu_{w_i}^t \times \mu_{w_i} + \mathbf{log}(P(w_i))$$

Comme ces fonctions discriminantes sont linéaires , les surfaces de décision sont des hyperplans. Ainsi l'hyperplan séparant les régions Ri et Rj (si ces régions sont adjacentes) a pour équation $g_{wi}(x)=g_{wj}(x)$.

L'équation de cet hyperplan est donc $g_{wi}(x)-g_{wj}(x)=0$ que l'on peut écrire sous la forme :

$$W^t \times (x - x_0) = 0$$

$$avec \quad W = \mu_{w_i} - \mu_{w_j}$$

$$et \quad x_0 = \frac{1}{2} \times (\mu_{w_i} + \mu_{w_j}) - \frac{\sigma^2}{\left\|\mu_{w_i} - \mu_{w_j}\right\|^2} \times \log\left(\frac{P(w_i)}{P(w_j)}\right) \times (\mu_{w_i} - \mu_{w_j})$$

C'est l'équation d'un hyperplan perpendiculaire au vecteur W et passant par x_0

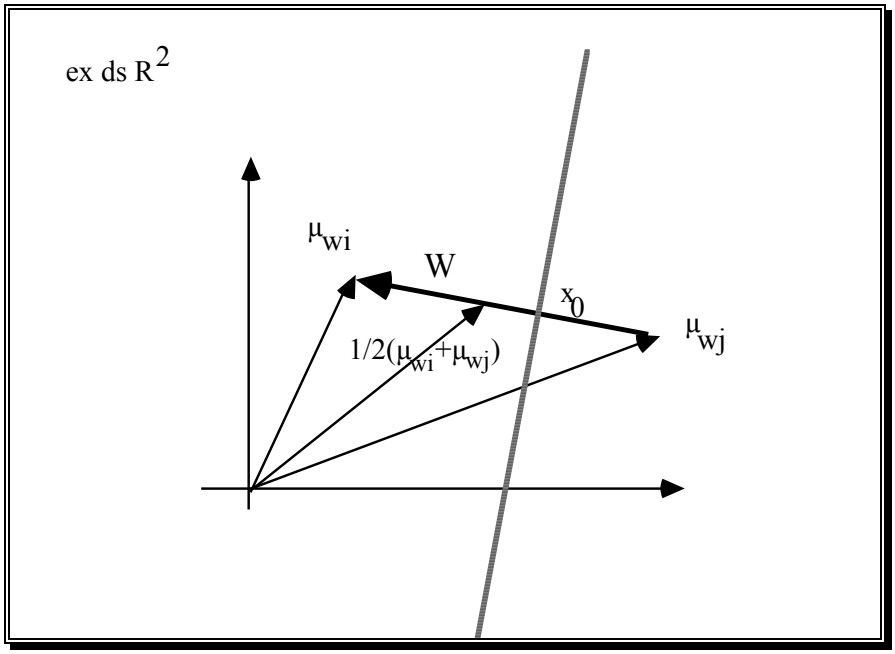


Figure 16 - surfaces de décision pour classes de matrice de covariance égale à IxCte

Si de plus on a P(wi)=P(wj) alors l'hyperplan séparateur passe par le milieu du segment joignant les extrémités des vecteurs μ_{wi} et μ_{wj}

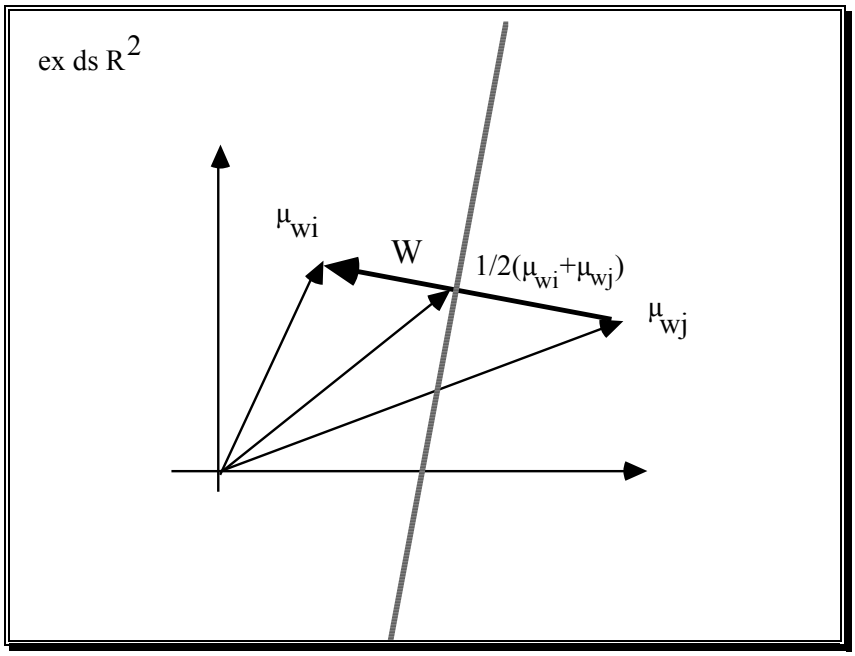


Figure 17 - surfaces de décision pour classes de matrice de covariance égale à IxCte et de probabilités à priori identiques

Si $\sigma^2 <<<<< \left\| \mu_{w_i} - \mu_{w_j} \right\|^2$

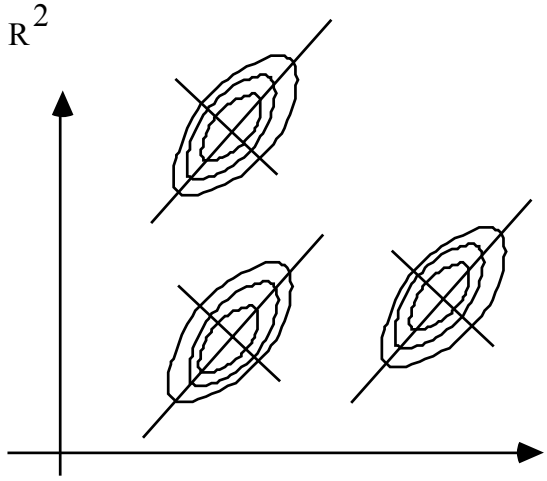
alors $P(w_i)$ et $P(w_j)$ influent peu et donc $x0 \approx \frac{1}{2}\left(\mu_{w_i} + \mu_{w_j}\right)$

Sinon si $P(w_i) > P(w_j)$, alors l'hyperplan séparateur est plus proche de μ_{wj} que de μ_{wi}

2.3.1.2.2 Cas où toutes les classes ont même matrice de covariance

Dans ce cas toutes les classes recherchées ont une matrice de covariance identique (ce qui se comprendre comme toutes les classes ont même « forme » et même « taille » , seuls les centres de classe sont différents.

ex ds R^2



Les fonctions discriminantes pour des classes suivant une loi normale peuvent être nous l'avons vu:

$$g_{w_i}(x) = -\frac{1}{2}\left(x - \mu_{w_i}\right)^t \times \sum_{w_i}^{-1} \times \left(x - \mu_{w_i}\right) - \frac{d}{2}\log(2\pi) - \frac{1}{2}\log\left|\Sigma_{w_i}\right| + \log(p(w_i))$$

Dans ces fonctions on peut supprimer les termes qui sont indépendants de i :

on obtient alors:

$$\boxed{g_{w_i}(x) = -\frac{1}{2}\left(x - \mu_{w_i}\right)^t \times \sum^{-1} \times \left(x - \mu_{w_i}\right) + \log(p(w_i))}$$

avec

$\left(x - \mu_{w_i}\right)^t \times \sum^{-1} \times \left(x - \mu_{w_i}\right)$ qui représente la distance de Mahanalobis de la forme x à la moyenne de la classe μ_{wi}

Si l'on considère maintenant deux classes w_i et w_j contiguës dans l'espace des paramètres(ou des observations , ou des formes...ici R^d) alors la frontière est définie par

$$\mathbf{g(x)=(g_{wi}(x)-g_{wj}(x))=0}$$

soit l'équation d'un hyperplan

$$\mathbf{W^t} \text{ x } \mathbf{(x-x_0) =0}$$

avec
(après quelques calculs sympas à faire au lieu d'aller au cinetoche!)

$$W = \sum^{-1} \times \left(\mu_{w_i} - \mu_{w_j} \right)$$

$$x_0 = \frac{1}{2}\left(\mu_{w_i} + \mu_{w_j} \right) - \frac{\mathbf{log}\left(\frac{P(w_i)}{P(w_j)} \right)}{\left(\mu_{w_i} - \mu_{w_j} \right) \times \sum^{-1} \times \left(\mu_{w_i} - \mu_{w_j} \right)} \times \left(\mu_{w_i} - \mu_{w_j} \right)$$

On voit que

- le vecteur W n'est pas forcément dans la direction $(\mu_{wi}-\mu_{wj})$

donc l'hyperplan séparateur des deux classes considérées n'est pas forcément perpendiculaire au vecteur $(\mu_{wi}-\mu_{wj})$

- Si $P(w_i)= P(w_j)$ alors cet hyperplan passe par le milieu du segment joignant les extrémités des vecteurs μ_{wi} et μ_{wj}

Dans $R^{2,}$

si $P(w_i)= P(w_j)$, on obtiendrait:

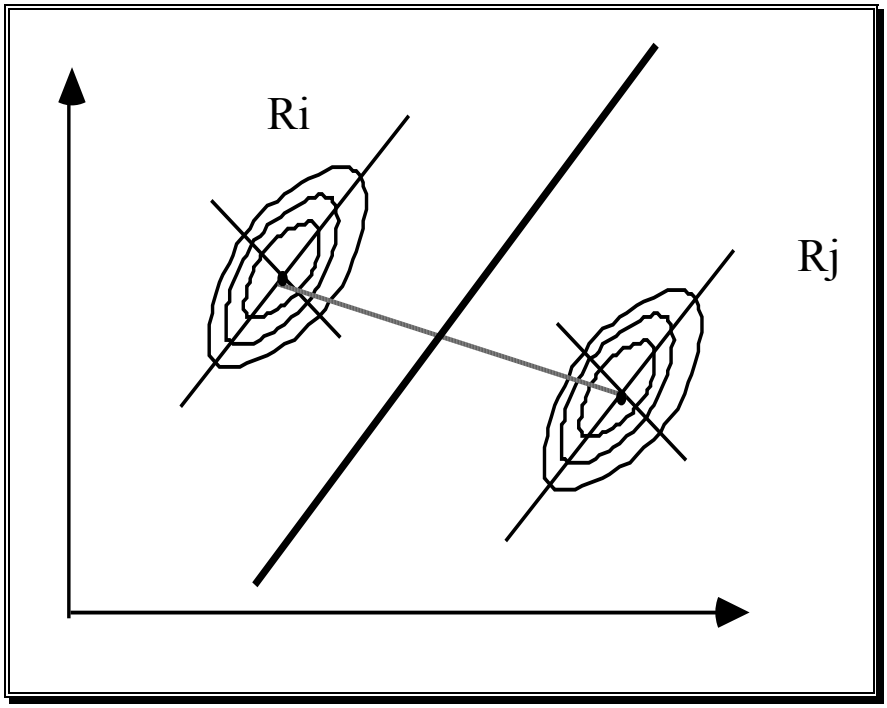


Figure 18 - surfaces de décision pour des classes de matrices de covariance identique

2.3.1.2.3 Cas où les classes ont des matrices de covariance distinctes

Reprenons la forme générale des fonctions discriminantes:

$$g_{w_i}(x) = -\frac{1}{2}\left(x-\mu_{w_i}\right)^t \times \sum_{w_i}^{-1} \times \left(x-\mu_{w_i}\right) - \frac{d}{2}\log(2\pi) - \frac{1}{2}\log\left|\sum_{w_i}\right| + \log(p(w_i))$$

On voit que le seul terme que l'on puisse éliminer (dans toutes les fonctions) est $\frac{d}{2}\log(2\pi)$. On obtient alors une fonction du second degré en x de la forme

$$g_{w_i}(x) = x^t \times W_1 \times x + W_2{}^t \times x + W_3$$

avec

$$W_1 = -\frac{1}{2}\sum_{w_i}^{-1}$$

$$W2 = \sum_{w_i}^{-1} \times \mu_{w_i}$$

$$W3 = -\frac{1}{2}\mu_{w_i}^t \times \sum_{w_i}^{-1} \times \mu_{w_i} - \frac{1}{2}\log\left(\left|\sum_{w_i}\right|\right) + \log(P(w_i))$$

Les surfaces de décision obtenues sont des hyperquadratiques

- paire d'hyperplans
- hypersphéres
- hyperellipsoides
- hyperparaboloides
- hyperhyperboloides

Exemples pour deux classes (les ellipses ou cercles représentent dans différentes configurations des probabilité d'appartenance toutes identiques)

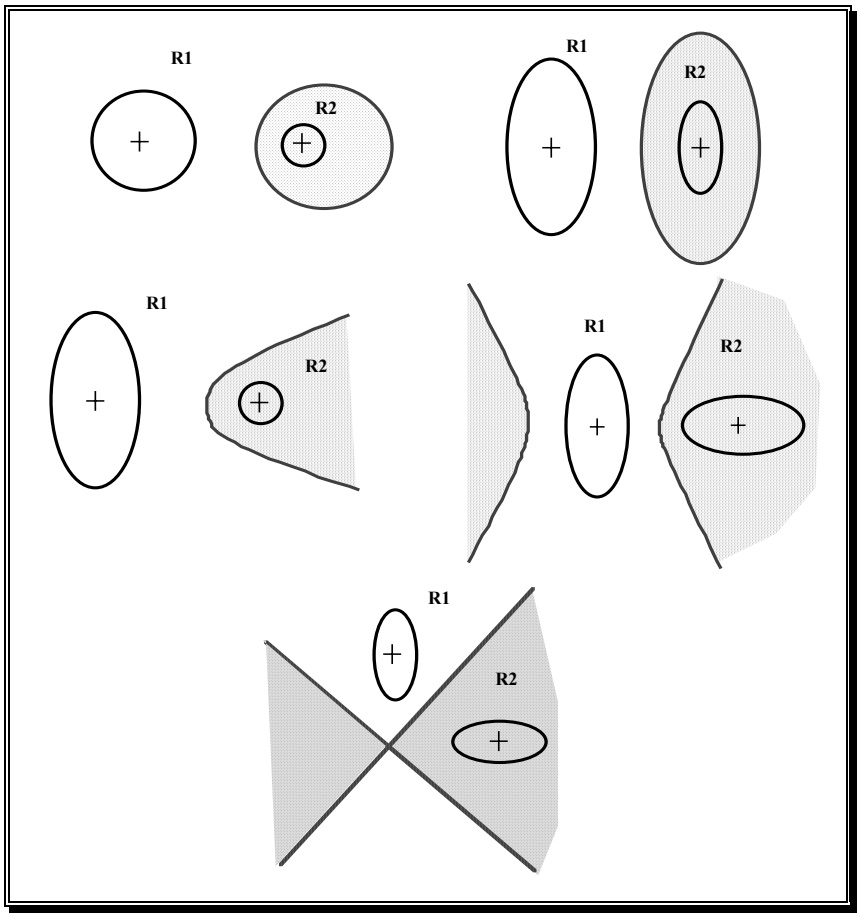


Figure 19 - Exemples de surfaces de décision dans le cas général (classes gaussiennes)

2.3.1.2.4 quelques remarques

Nous avons appliqué la méthode de décision Bayésienne en faisant l'hypothèse relativement restrictive que toutes les classes suivaient des lois gaussiennes (en gros: loi de densité de probabilité en « cloche »). Puis nous avons fait successivement des hypothèses de plus en plus restrictives sur les classes qui à chaque fois modifiaient la complexité des méthodes de décision induites. Il est bien évident qu'il faut vérifier que les méthodes sont acceptables.

On peut concevoir de mesurer cela à partir des échantillons représentatifs des classes (par exemple à partir des histogrammes)

Notons que si l'on trouve des classes qui visiblement ne sont pas gaussiennes (en regardant l'histogramme des paramètres) , on peut peut-être se ramener à ce cas là en découpant artificiellement la classe concernée en sous classes.

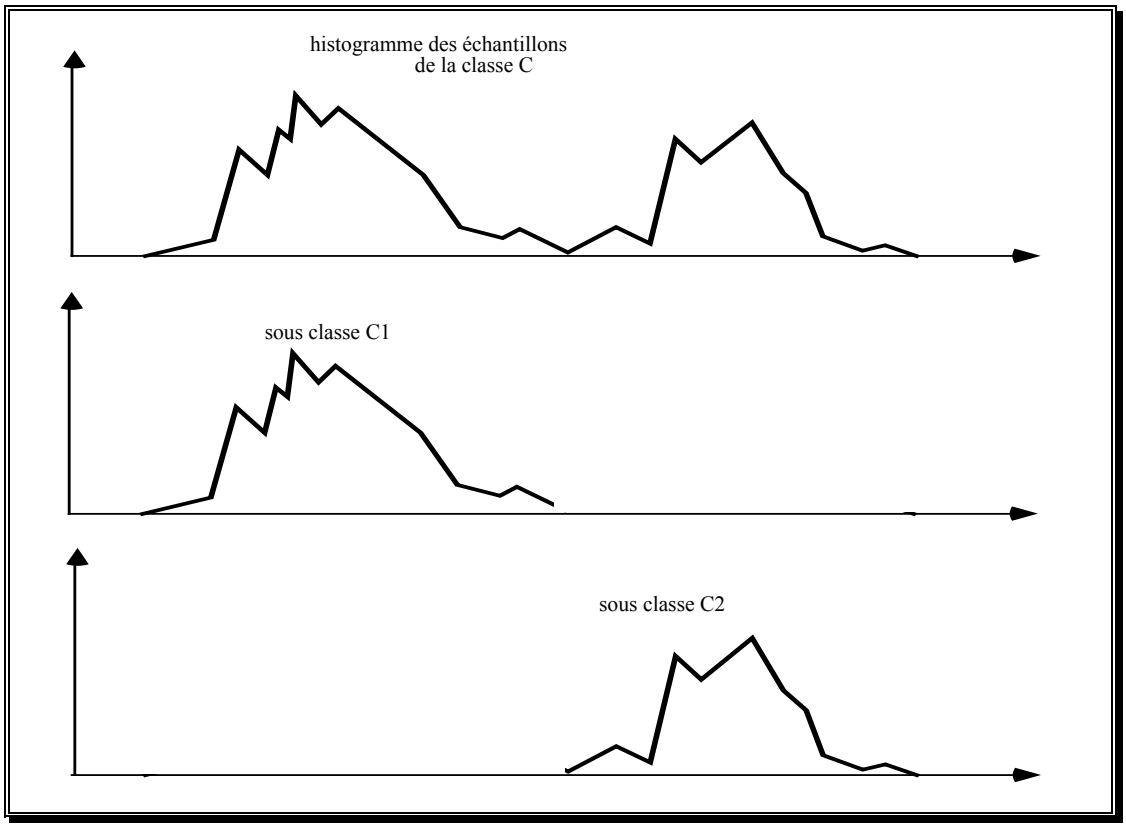


Figure 20 - classe non gaussienne et hypothèse gaussienne

2.4 APPRENTISSAGE

2.4.1 généralités

Comme nous l'avons vu dans le cadre de la décision Bayésienne:

Si l'on connaît pour chacune des classes w_i recherchées la probabilité à priori de la classe $P(w_i)$ et d'autre part la loi de densité de probabilité de la classe $p(x/w_i)$ alors on peut définir grâce à la décision Bayésienne un classifieur optimal.

Si l'on se place dans ce cadre apprendre à reconnaître les classes consiste alors à découvrir $P(w_i)$ et $p(x/w_i)$ à l'aide des données fournies qui sont les exemples de formes de la classe : les échantillons.

On supposera que ces échantillons sont bien représentatifs de la classe et en donnent une vision globale. Malheureusement ces hypothèses ne sont pas toujours satisfaites.

Le principe de l'apprentissage consiste donc à obtenir $P(w_i)$ et $p(x/w_i)$ à partir d'un ensemble d'échantillons mais il arrive que l'on puisse disposer de données à priori par exemple la probabilité à priori d'une classe...

Il existe maintenant plusieurs approches plus ou moins complexes:

2.4.1.1 Méthodes paramétriques

Si l'on fait une hypothèse sur la forme de la loi de densité de probabilité $p(x/w_i)$, par exemple par ce qu'on a des informations initiales sur la « forme » de la classe alors la loi générale est connue et sa forme particulière est modulée par un nombre fini de paramètres.

L'apprentissage à l'aide des échantillons va simplement consister à trouver ces paramètres.

- exemple 1 :

si les formes sont définies sur R et que l'on sait que la loi $p(x/w_i)$ est une loi normale $N(\mu,\sigma)$ nous allons utiliser les échantillons de la classe pour estimer les vraies valeurs des deux paramètres réels μ et σ. On disposera alors de l'expression analytique de $p(x/w_i)$

- exemple 2:

Si les formes sont définies sur R^d et que l'on sait que la classe suit une loi normale $N(\mu, \sum)$ alors on utilise les échantillons pour estimer μ et $\sum$ c'est à dire en fait un nombre de paramètres défini par la formule suivante:

$$\frac{1}{2}(d^2 + 3 \times d)$$

paramètres: $\mu_1,\mu_2.....\mu_d$, σ_{11}, $\sigma_{12}.....$ σ_{1d}, σ_{22}, $\sigma_{23}.....$ σ_{2d}, σ_{33}, $\sigma_{34}.....$ σ_{dd}

- exemple 3 :

$p(x/w_i)$ peut suivre toute autre loi que normale mais définie par des paramètres.

2.4.1.2 Méthodes non paramétriques

Ici on ne fait aucune hypothèse sur la « forme » de la classe ce qui signifie que la loi de densité de probabilité $p(x/w_i)$ peut avoir une forme totalement imprévisible.

Il faut donc trouver une méthode permettant d'obtenir $p(x/w_i)$ pour toute forme x.

$$p(x/w_i)=f(x,\text{échantillons})$$

Les méthodes que nous verrons dans ce cadre considèrent un voisinage (au sens large) de la forme x et prennent en compte la densité des échantillons dans ce voisinage et éventuellement la distance des échantillons à x.

Plus les échantillons sont proches de x, plus ils sont nombreux et plus $p(x/w_i)$ sera élevé.

Si les échantillons sont représentatifs de la classe et s'ils sont suffisamment nombreux alors on peut espérer obtenir une loi $p(x/w_i)$ proche de la vraie loi suivie par la classe.

2.4.2 methodes paramétriques

2.4.2.1 Introduction

La loi de densité de probabilité de la classe w_i est supposée être une loi connue (par exemple une loi normale, un loi de Poisson, une loi gamma,....), cette loi étant totalement définie par la connaissance des paramètres caractéristiques de la loi.

par exemple μ et σ pour une loi monodimensionnelle $N(\mu,\sigma)$

Plusieurs approches sont possibles et entre autres:

- méthode du maximum de vraisemblance:

Les paramètres de la loi recherchés sont considérés comme « fixes » et le choix optimal est celui qui maximise la probabilité d'obtenir les échantillons fournis pour définir la classe.

c'est cette méthode que nous décrirons par la suite.

- méthode d'estimation Bayésienne

Les paramètres cherchés sont considérés comme des variables aléatoires ayant une distribution a priori connue.

Notons au passage que nous n'étudierons que l'apprentissage supervisé, c'est à dire que pour chaque échantillon fourni on connaît la classe correspondante.

En apprentissage non supervisé on peut avoir un ensemble d'échantillons représentatifs des classes cherchées mais l'étiquette de chaque échantillon est inconnue. Nous traiterons ce cas dans le cadre de la classification automatique.

2.4.2.2 Estimation par le maximum de vraisemblance

Soient E_1 , E_2 , ..., E_c c ensembles d'échantillons représentant les c classes de formes recherchées. On supposera que ces échantillons ont été « tirés » indépendamment (mais en accord avec la loi de densité de probabilité $p(x/w_i)$ de la classe à laquelle ils appartiennent respectivement)

On fait de plus hypothèse que la loi $p(x/w_i)$ a une forme paramétrique connue déterminée par la valeur d'un vecteur θ_{wi} de paramètres.

Nous prendrons pour fixer les idées: $p(x/w_i)$ est une loi normale $N(\mu_{wi},\sum_{wi})$

alors θ_{wi} donne en fait μ_{wi} et $\sum_{wi}$

Et si les formes x sont définies dans R^d alors,

$$\theta_{wi}=\{ \mu_1,\mu_2.....\mu_d, \sigma_{11}, \sigma_{12}..... \sigma_{1d}, \sigma_{22}, \sigma_{23}..... \sigma_{2d}, \sigma_{33}, \sigma_{34}..... \sigma_{dd}\}$$

Pour indiquer que $p(x/w_i)$ dépend de θ_{wi} on le notera $p(x/w_i,\theta_{wi})$

2.4.2.2.1 Méthode d'estimation des paramètres de la loi de densité de probabilité

Dans les calculs qui suivent on fait hypothèse (raisonnable au demeurant !) que les échantillons de E_i ne donnent pas d'informations sur θ_{wj} si $j \neq i$

On peut alors raisonner indépendamment classe par classe.

On considère donc maintenant que pour la classe considérée on dispose d'un ensemble E d'échantillons tirés indépendamment suivant la loi de densité de probabilité p(x,θ)

Nous noterons E={X1,X2, ... , Xn}

Alors la probabilité d'obtenir le « tirage » E si la classe suit une loi définie par le paramètre θ est donnée par:

$$\boxed{p(E,\theta) = \prod_{k=1}^{n} p(X_k,\theta)}$$

On cherche donc la valeur du vecteur θ , $\hat{\theta}$ maximise p(E, θ)

Notons $\theta = (\theta_1\ \theta_2 \ldots\ldots \theta_p)$ et $\nabla_\theta = \begin{bmatrix} \frac{\partial}{\partial\theta_1} \\ \cdot \\ \cdot \\ \frac{\partial}{\partial\theta_n} \end{bmatrix}$

Soit l'expression

$$l(\theta) = \mathbf{log}\left[p(E,\theta\right]$$

$$= \sum_{k=1}^{n} \mathbf{log}\left[p(X_k,\theta\right]$$

Alors

$$\nabla_\theta l = \sum_{k=1}^{n} \nabla_\theta \mathbf{log}\left[p(X_k,\theta\right]$$

Alors une condition nécessaire pour que p(E,θ) soit maximum est que

$$\nabla_\theta l = 0$$

2.4.2.2.2 Estimation des paramètres pour une classe gaussienne (une dimension)

Alors $\theta=\{\theta_1\ \theta_2\}$ avec $\theta_1=\mu$ et $\theta_2=\sigma^2$

la loi normale dans R s'écrit $$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \times e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

donc

$$\log\left[p(X_k, \theta)\right] = \frac{1}{2}\log\left(2\pi \times \theta_2\right) - \frac{1}{2\theta_2} \times \left(X_k - \theta_1\right)^2$$

$$\nabla_\theta \log\left[p(Xk, \theta)\right] = \begin{bmatrix} \frac{1}{\theta_2} \times \left(X_k - \theta_1\right) \\ -\frac{1}{2 \times \theta_2} + \frac{\left(X_k - \theta_1\right)^2}{2 \times {\theta_2}^2} \end{bmatrix}$$

alors

$$\nabla_\theta l = 0 \Rightarrow \sum_{k=1}^{n} \frac{1}{\hat{\theta}_2} \times \left(X_k - \hat{\theta}_1\right) = 0 \quad et \quad -\sum_{k=1}^{n} \frac{1}{\hat{\theta}_2} + \sum_{k=1}^{n} \frac{\left(X_k - \hat{\theta}_1\right)^2}{\hat{\theta}_2^2} = 0$$

d'où le résultat bien connu pour estimer moyenne et variance d'une loi normale à partir d'échantillons:

$$\hat{\theta}1 = \hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} X_k \qquad \textbf{moyenne arithmétique}$$

$$\hat{\theta}_2 = \hat{\sigma}^2 = \frac{1}{\mathbf{n}}\sum_{k=1}^{n}\left(\mathbf{X_k} - \hat{\mu}\right)^2 \qquad \textbf{écart type}$$

2.4.2.2.3 Estimation des paramètres pour une classe gaussienne multidimensionnelle

Nous allons d'abord étudier le problème de l'estimation de la moyenne μ dans le cas d'une loi normale multidimensionnelle. On a donc:

$$p(Xk,\mu) = \frac{1}{(2\pi)^{d/2} \times \left|\sum\right|^{1/2}} \times e^{\left[-\frac{1}{2}(X_k-\mu)^t \times \sum^{-1} \times (X_k-\mu) \right]}$$

Σ : matrice de covariance

d'où

$$\log\left[p(X_k,\mu)\right] = -\frac{1}{2}\log\left[(2\pi)^d \times \left|\sum\right| \right] - \frac{1}{2}(X_k-\mu)^t \times \sum^{-1} \times (X_k-\mu)$$

et $$\nabla_\mu \log\left(p(X_k,\mu)\right) = \sum^{-1} \times (Xk-\mu)$$

L'estimation de la moyenne, $\hat{\mu}$ est définie par:

$$\nabla_\mu l = \sum_{k=1}^{n}\left(\sum^{-1} \times (X_k - \hat{\mu})\right) = 0 \quad soit\left(\sum_{k=1}^{n} X_k\right) - n \times \hat{\mu} = 0$$

$$soit \quad \hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} X_k \quad \textbf{moyenne arithmétique}$$

On retrouve le résultat bien classique :

la moyenne arithmétique des échantillons (moyenne arithmétique de chacune des composantes car on est dans R^d) fournit une estimation optimale de la moyenne μ de la loi normale.

Finalement on trouve (relativement longs calcul pour l'estimation de la matrice de covariance):

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} X_k \quad et \quad \hat{\sum} = \frac{1}{n}\sum_{k=1}^{n}(Xk-\mu)^t \times (X_k-\mu)$$

2.4.3 methodes non parametriques

2.4.3.1 Introduction

Dans les méthodes d'apprentissage paramétriques , les échantillons permettent d'estimer les paramètres de la loi de densité de probabilité de chaque classe, mais toutes les lois de distribution classiques sont unimodales, elles ne peuvent donc prétendre représenter toutes les posssibilités pour les classes.

D'autre part faire des hypothèses erronées sur une classe peut amener à des résultats catastrophiques , comme dans l'exemple suivant, où on dispose d'échantillons répartis suivant l'exemple ci-dessous (on suppose que l'espace des observations est R) et où on fait l'hypothèse que la loi suivie par la classe est normale.

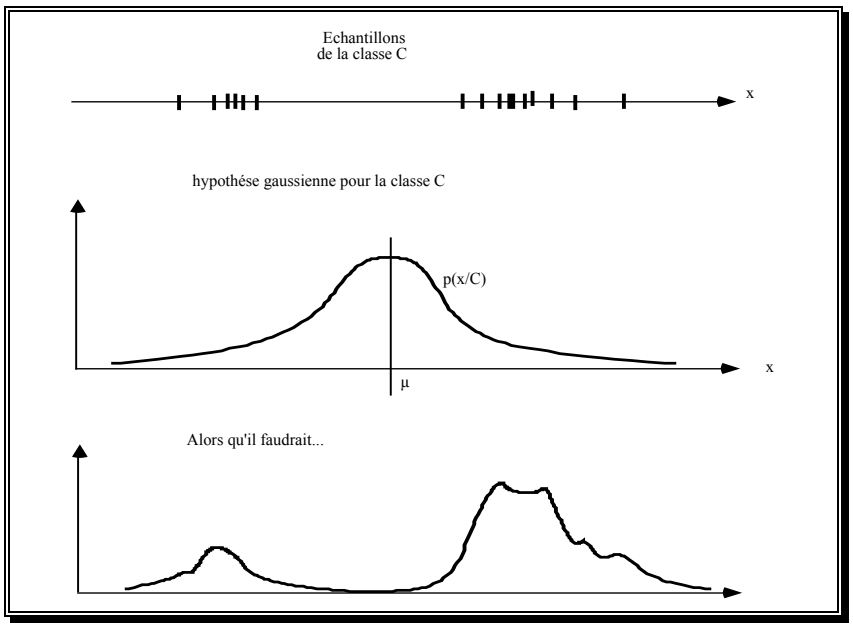


Figure 21 - De la nécessité de l'apprentissage non paramétrique..

Les méthodes non paramétriques vont prendre en compte les échantillons et surtout leur répartition spatiale dans l'espace des paramètres pour obtenir des loi de densité de probabilités plus proche de la réalité de la classe.

Les méthodes que nous verrons auront pour objectif d'estimer les lois $p(x/w_i)$ ou bien $p(w_i/x)$ directement à partir des échantillons.

2.4.3.2 principes de base

Nous allons décrire ici les principes de base sur lesquels repose l'apprentissage non paramétrique, et notamment la notion de voisinage et de prise en compte des échantillons dans ce voisinage.

Nous nous intéresserons ici à l'apprentissage des lois de densité de probabilité $p(x/w_i)$.
Notons que nous noterons $p(x/w_i)$ $p(x)$ pour simplifier dans la mesure où la méthode décrite s'appliquera pour chacune des classes indépendamment des autres.

Soit donc un domaine D inclus dans l'espace des attributs et pouvant être considéré comme un voisinage de x, forme pour laquelle on cherche $p(x/w_i)$

$$D \subset R^d$$

et supposons que

$$p(x/wi) \quad donc \quad p(x) \approx Cte \quad sur \quad D$$

alors

$$p(x \in D) = \int_D p(x') \times dx' \approx p(x) \times \int_D dx'$$

$$soit \quad p(x \in D) \approx p(x) \times V(D) \quad avec \quad V(D) \quad hypervolume \quad de \quad D$$

Soit

$$\boxed{\forall x \in D \qquad p(x) \approx \frac{p(x) \in D}{V(D)}}$$

Supposons maintenant disposer de **n échantillons au total** définissant la classe concernée et que parmi ces n échantillons **t échantillons se trouvent dans le domaine D**

Alors on peut dire que

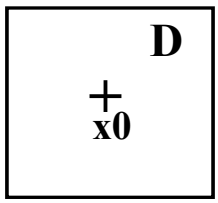
$$p(x \in D) \approx \frac{t}{n}$$

d'où

$$\boxed{p(x) \approx \frac{\left(t/n\right)}{V(D)}}$$

En pratique

Soit x_0 et $D(x_0)$ un domaine entourant x_0



Alors une estimation de $p(x_0)$ est:

$$\hat{p}(x_0) = \frac{(t/n)}{V(D(x_0))}$$

Cet estimateur de la probabilité d'obtenir la forme spécifique x_0 est la formule de base de l'apprentissage non paramétrique.

Les diverses variantes possibles joueront sur la définition du voisinage $D(x_0)$ qui sera liée à n et d'autre part sur la possibilité de faire intervenir dans la définition du voisinage la notion de proximité à x_0.

2.4.3.3 Choix d'un estimateur

Le problème posé est celui de la qualité de l'estimateur et de sa convergence.

Supposons dans un premier temps que $D(x_0)$ est fixé

Alors si m croît alors, $\frac{t}{n} \rightarrow vraie\ \ valeur\ \ de\ \ P(x \notin D(x_O))$

mais $\hat{p}(x0) = \frac{P(x \in D(x0))}{V(D(x0))}$ est en fait une valeur « moyennée » (ou lissée) sur le voisinage $D(x_0)$.

Pour gérer ce problème de lissage et s'approcher de la vraie valeur de p(x0), on a intérêt à diminuer $D(x_0)$ et donc $V(D(x_0))$

Mais si le nombre n d'échantillons qui est fini est faible alors on va trouver immanquablement de nombreux $D(x_0)$ sans échantillons!.. et donc des $p(x_0)$ nuls!

On en déduit qu'il est nécessaire de lier n et la taille de $D(x_0)$:

Nombre d 'échantillons	Domaine utilisé
1	D1(x0)
2	D2(x0)
.....	.....
n	Dn(x0)

Alors on a:

$$\hat{p}_n(x_0) = \frac{t_n / n}{V(D_n(x_0))}$$

.

avec t_n nombre d'échantillons dans $D(x_0)$ qui dépend de n bien sûr

On montre que

$\hat{p}_n(x_0) \to p(x_0)$ **vraie valeur si**

* $\lim_{n\to+\infty} V(D_n(x_0)) = 0$

** $\lim_{n\to+\infty} t_n = +\infty$

*** $\lim_{n\to+\infty} \frac{t_n}{n} = 0$

On trouve alors deux types de méthodes fondées sur les remarques précédentes:

1. <u>lier $V(D_n(x_0))$ à n</u>

   On a alors la **méthode du noyau** (fenêtres de Parzen)

2. <u>fixer t_n en fonction de n et faire croître $V(D_n(x_0))$ jusqu'à ce qu'il contienne t_n échantillons</u>

   On a alors la **méthode des t_n plus proches voisins**

2.4.3.4 Estimation par la méthode du noyau

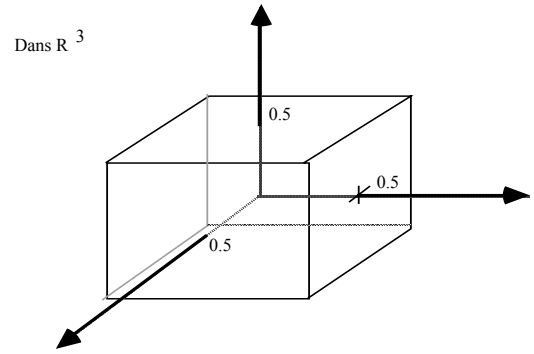
Dans cette méthode dite des fenêtres de Parzen , le point clé est le choix de $D_n(x_0)$

Par exemple on peut prendre un hypercube de coté h_n dans R^d (si on suppose que les formes x sont définies sur R^d)

Alors $V(D_n(x0))=(h_n)^d$

Définissons maintenant la fonction $\varphi(u)$ égale à 1 dans l'hypercube de coté 1 et centré à l'origine dans l'espace R^d

$$\begin{cases} \varphi(u) = 1 \quad si \quad |u_j| \le 0.5 \quad j = 1,....d \\ \varphi(u) = 0 \quad \textbf{sinon} \end{cases}$$



Alors si $D_n(x0)$ est un hypercube de coté h_n alors

$$x \in D_n(x_0) \Leftrightarrow \varphi\left(\frac{x_0 - x}{h_n}\right) = 1$$

et le nombre t_n d'échantillons qui se trouvent dans $D_n(x_o)$ est donné par la formule:

$$t_n = \sum_{i=1}^{n} \varphi\left(\frac{x_0 - x_i}{h_n}\right) \quad avec \quad x_i \quad échantillon \quad n° \quad i$$

d'où

$$\hat{p}_n(x_0) = \frac{t_n / n}{V(D_n(x_0))} = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{V(D_n(x_0))} \times \varphi\left(\frac{x_0 - x_i}{h_n}\right)$$

La fonction φ s'appelle le **noyau de l'estimateur**

Nous avons proposé une forme de ce noyau qui est un hypercube de volume unité centré à l'origine de l'espace des observations, mais il existe une infinité de choix pour la définition de ce noyau

2.4.3.4.1 Possibilités de choix pour le noyau

Il y a d'abord une condition de normalité de façon à ce que $\hat{p}_n(x_0)$ soit une loi de densité de probabilité et cette condition s'exprime par

$$\begin{cases} \varphi(x) \geq 0 \quad \forall x \in R^d \\ \int_{R^d} \varphi(x) \times dx = 1 \end{cases}$$

D'autre part la fonction φ doit permettre de définir une « fonction d'appartenance » à un voisinage de x_0

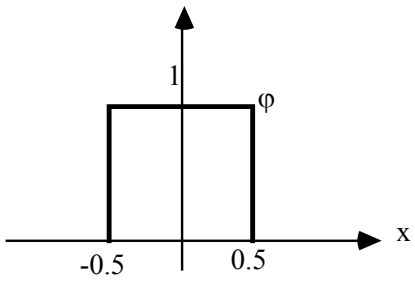
Si l'on représente les fonctions φ possibles (dans R pour simplifier !)

Noyau cubique

C'est le cas que nous avons envisagé précédemment où φ était définie dans un hypercube unité centré à l'origine.

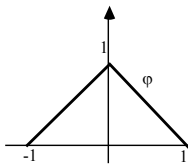
On a alors dans R:

$$\varphi(x) = 1 \quad si \quad |x| \leq 0.5$$
$$= 0 \quad si \quad |x| \succ 0.5$$



Noyau triangulaire

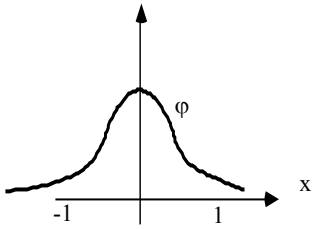
$$\varphi(x) = 1 - |x| \quad si \quad |x| \leq 1$$
$$= 0 \qquad si \quad |x| \succ 1$$



On voit bien ici que le domaine $D_n(x_0)$ ne sera plus un ensemble classique et φ pourra être interprétée comme le degré d'appartenance au domaine $D_n(x_0)$. L'appartenance totale (=1) ne pourra être attribuée qu'à $x=x_0$.

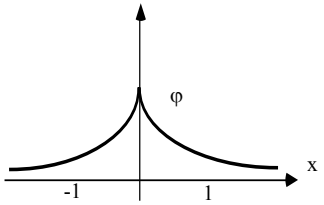
Noyau normal

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} \times e^{-\frac{x^2}{2}}$$

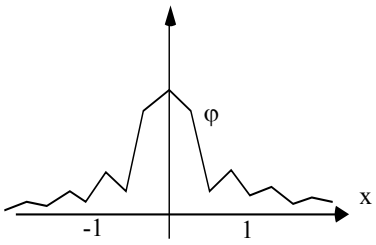


Noyau exponentiel

$$\varphi(x) = \frac{1}{2} \times e^{-|x|}$$



……..



d'où les noyaux possibles suivants $\varphi\left(\frac{x_0 - x_i}{h_m}\right)$ par exemple (formes définies sur R):

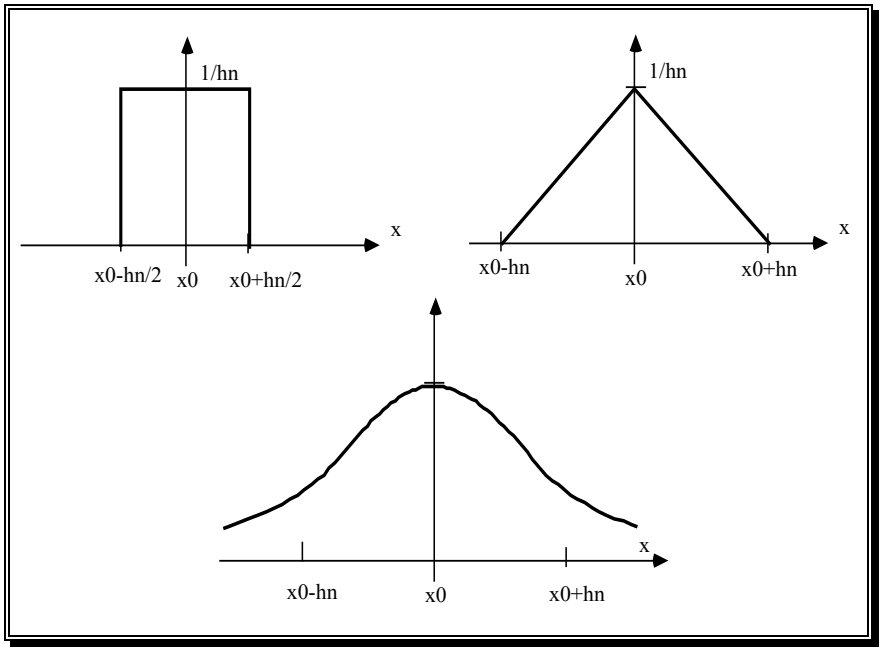


Figure 22 - exemples de noyaux de Parzen

l'estimateur de la probabilité au point x_0 est donnée par

$$\hat{p}_n(x_0) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{V(D_n(x_0))} \times \varphi\left(\frac{x_0 - x_i}{h_n}\right)$$

et sa convergence est assurée (en supposant que les formes sont définies dans R^d) si les conditions suivantes sont vérifiées:

$$\mapsto \lim_{|x|\to+\infty} |x|^d \times \varphi(x) = 0$$

$$\mapsto \lim_{n\to+\infty} h_n = 0$$

$$\mapsto \lim_{n\to+\infty} n \times (h_n)^d = +\infty$$

Compréhension de la méthode de l'estimateur:

L'estimation de p(x0) par $\hat{p}_n(x_0)$ peut se comprendre comme la « superposition » de n fonctions (les noyaux centrés sur les n échantillons définissant la classe)

On a vu différents types de noyaux et on comprend aisément que dans la définition de ces fonctions plus h_n est petit plus le nombre d'échantillons ayant une influence sur $\hat{p}_n(x_0)$
sera faible.

Dans la figure suivant nous présentons différentes valeurs de h_n et sa conséquence sur $\hat{p}_n(x_0)$

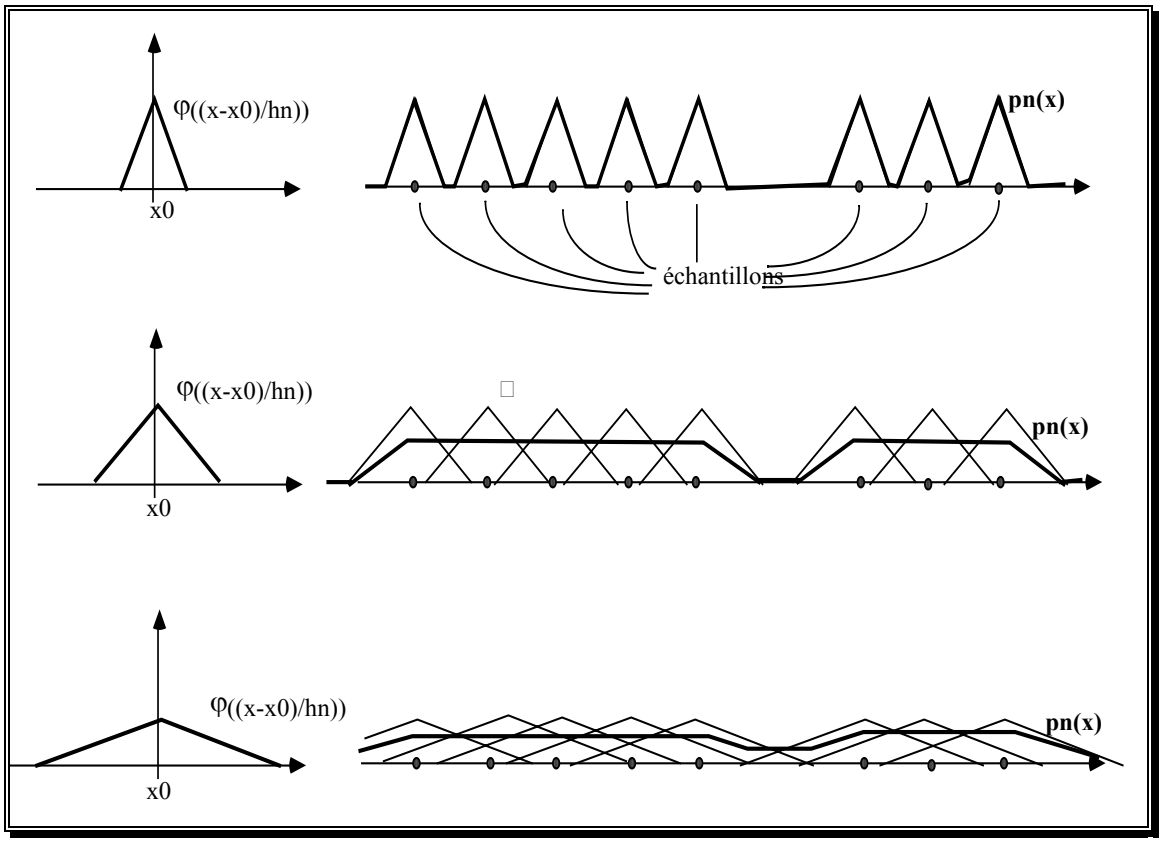


Figure 23 - Différents noyaux et estimation p(x0) correspondante

De façon à éviter d'avoir trop de points x_0 ou $\hat{p}_n(x_0)$ sera nulle du fait de l'absence d'échantillons proches on en conclut qu'il faut trouver une formule liant h_n et n où h_n sera relativement grand si le nombre n d'échantillons est petit.

Exemples:

$$h_n = \frac{h_0}{\sqrt{n}} \quad ou \quad bien \quad h_n = \frac{h_0}{\log n}$$

On voit qu'alors le problème important est le choix de h_0, notamment lorsque le nombre d'échantillons est faible.

2.4.3.5 Méthode des t_n plus proches voisins

Dans cette méthode l'idée est d'adapter la taille du « voisinage » de x_0 de façon à englober un nombre fixé t_n d'échantillons parmi les n échantillons définissant la classe.

Ainsi on est certain d'avoir suffisamment d'échantillons pour contribuer à la définition de $\hat{p}_n(x_0)$.

Soit $D_r(x_0)$ un domaine de volume unité centré en x_0

$$V[D_r(x_0)]=0$$

Soit $D(x_0,\alpha)$ le domaine homothétique de $D_r(x_0)$
de centre x_0 et de rapport d'homothétie α

Alors

$$V[D(x_0,\alpha)]=\alpha^d$$

La méthode choisie consiste donc à faire croître α jusqu'à ce que $D(x_0,\alpha)$ englobe t_n échantillons (on les appelle les t_n plus proches voisins)

L'estimateur de la densité de probabilité en x0 est donné par

$$\hat{p}_n(x0) = \frac{t_n / n}{V\left[D(x_0,\alpha)\right]}$$

On montre qu'il y a convergence de cet estimateur avec las choix suivants par exemple:

$$t_n = t_0 \times \sqrt{n} \quad ou \quad t_n = t_0 \times \mathbf{log}(n)$$

Le problème reste alors le choix de t_0

2.4.4 Estimation de probabilités a posteriori

Dans cette approche on va chercher à estimer la probabilité a posteriori pour chaque classe $p(w_i/x)$ à partir des échantillons. Cela revient en fait à considérer que la donnée des seuls échantillons définissant les c classes vont permettre d'estimer à la fois les probabilités à priori $P(w_i)$ et les lois de densités de probabilités $p(x/w_i)$.

On suppose donc disposer de n échantillons représentant les différentes classes possibles. On supposera de plus qu'il y en a K_i qui sont de la classe w_i .

$$n = \sum_{i=1}^{c} K_i$$

Soit un volume V autour de x contenant k échantillons dont k_i sont étiquetés w_i alors on peut estimer que:

$$P(w_i) = \frac{K_i}{n} \quad et \quad p(x/w_i) = \frac{k_i / K_i}{V}$$

$$donc \quad p(x/w_i) \times P(w_i) = \frac{k_i / n}{V}$$

d'où

$$p(w_i/x) = \frac{p(x/w_i) \times P(w_i)}{\sum_{j=1}^{c} p(x/w_j) \times P(w_j)} \approx \frac{ki/n / V}{\sum_{j=1}^{c} k_j / n / V} \approx \frac{k_i}{k}$$

Soit

$$\boxed{p(w_i/x) \approx \frac{k_i}{k}}$$

Pour le choix du volume V , on peut utiliser une technique voisine des fenêtres de Parzen ou bien des t_n plus proches voisins.

Dans le premier cas on peut ainsi choisir V égal à $\frac{h_0}{\sqrt{n}}$

Dans le deuxième cas on peut choisir V tel qu'il englobe un nombre fixé d'échantillons par exemple $t_0 \times \sqrt{n}$ échantillons

2.4.5 Méthodes de classification basée sur les échantillons

Les méthodes précédentes supposent calculer $p(x/w_i)$ ou plus directement $p(w_i/x)$, puis ensuite d'appliquer la règle de décision Bayésienne, mais certaines méthodes basées sur la fourniture d'échantillons vont avoir une approche plus globale. Elles vont utiliser directement les échantillons en tant que tels pour définir les classes et la méthode de décision associées.

2.4.5.1 Règle de décision du plus proche voisin

Cette méthode suppose disposer d'une distance dans l'espace des formes, et on est donc capable de calculer une distance entre une forme x quelconque à un échantillon.

Le principe de décision consiste tout simplement à calculer la distance de la forme inconnue x à tous les échantillons fournis. La forme est alors classée dans la classe de l'échantillon le plus proche (par exemple ci dessous x serait classé dans la classe w_2:

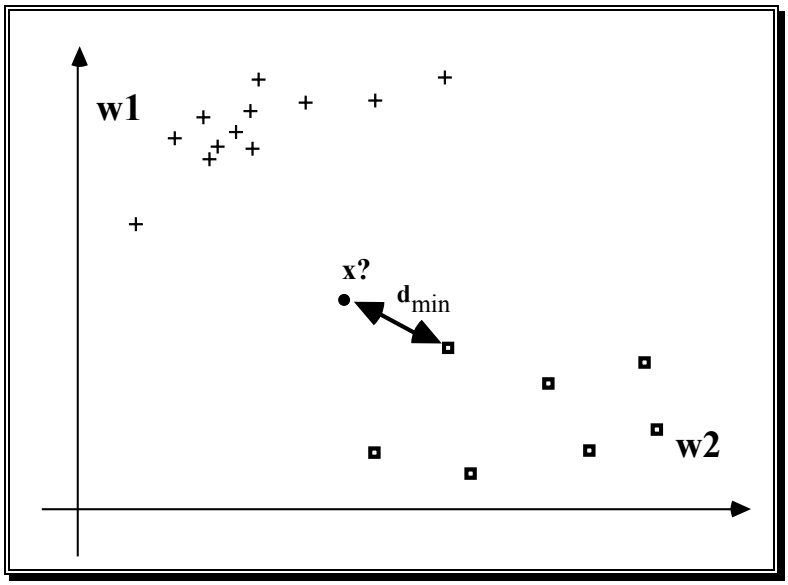


Figure 24 - Méthode de décision du plus proche voisin

2.4.5.2 Règle de décision des q plus proches voisins

Il s'agit d'une extension de la méthode précédente (plus proche voisin) : pour une forme inconnue x à classer, on va examiner la distance de x à tous les échantillons (qui définissent toutes les classes), puis on sélectionne les q plus proches échantillons et on affecte x à la classe majoritaire parmi ces q échantillons.

Dans l'exemple suivant et pour 5 voisins on classerait x dans la classe w_1:

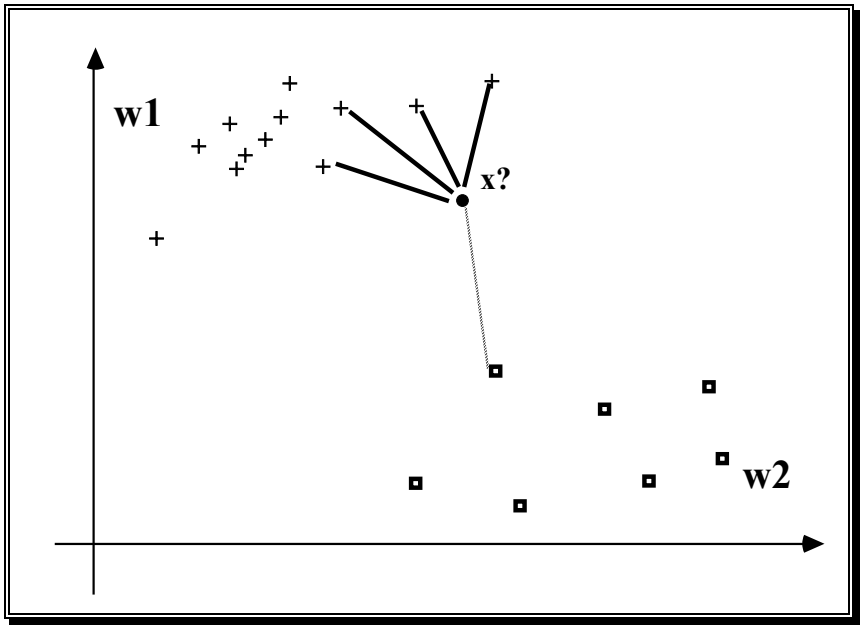


Figure 25 - Méthode des q plus proches voisins

Choix possible pour les distances:

- distance Euclidienne
- distance “city block“ (somme des valeurs absolues)
- distance de Tchebycheff (le sup)
- distance de Mahanalobis
- .....

2.5 CLASSIFICATION AUTOMATIQUE

2.5.1 Introduction

Dans cette approche il s’agit de faire une classification non supervisée (sans professeur), ce qui signifie qu’on ne fournit pas d’échantillons censés définir les classes . On peut comprendre le problème de la façon suivante: on dispose d’un ensemble de formes à répartir en paquets « homogènes »

On a donc un ensemble I de formes à classer . L’objectif à atteindre consiste à obtenir une partition P représentant au mieux les divers groupements pouvant exister au sein de cet ensemble.

Le nombre de classes peut être fixé ou non (à l’extrême si toutes les formes de l’ensemble I sont distinctes alors le nombre de classes peut être card(I)).

Un exemple de méthode de ce type est la méthode des nuées dynamiques (développée par Diday) où le nombre de classes à rechercher est fixé à priori à k.

2.6 EXERCICES

Donner l'expression générale de la distance de Mahalanobis d'une forme à une classe (définie par un ensemble de n échantillons dans R^d).

Expliquer les hypothèses faites permettant d'utiliser cette distance.
Indiquer son intérêt par rapport à la distance euclidienne pour la classification.
Donner l'expression de la matrice de covariance d'une classe et son interprétation.

Application pratique:

Soient deux classes suivant des lois normales et de probabilités a priori P(C1)=P(C2)=0.5.
On supposera que les formes sont des vecteurs de R^2 et que

$$\mu_{C_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mu_{C_2} = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \quad \sum_{C_1} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \quad \sum_{C_2} = \begin{bmatrix} 4/3 & -2/3 \\ -2/3 & 4/3 \end{bmatrix}$$

Donner l'équation de la frontière de décision bayésienne et la représenter graphiquement sommairement
note:

$$\sum_{C_1}^{-1} = \begin{bmatrix} 4/3 & -2/3 \\ -2/3 & 4/3 \end{bmatrix} \qquad \sum_{C_2}^{-1} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

Un eleveur de cabris dispose de deux races de cabris (race A et race B) et il a noté leurs poids respectifs dans le tableau suivant :

Poids en kg	Nombre race A	Nombre race B
3	1	
4	2	
5	2	
6	4	
7	5	1
8	9	3
9	10	4
10	8	7
11	5	10
12	4	7
13	1	5
14		3
15		1

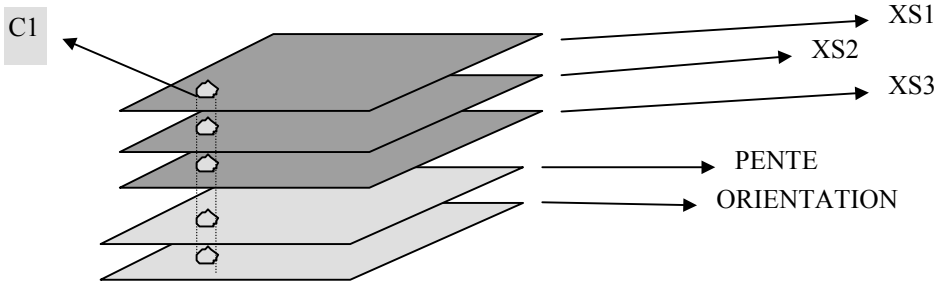
1. Si on choisit un cabri au hasard sur l'ensemble des cabris, quelle est la probabilité a priori qu'il soit de la race A (respectivement de la classe B)
2. calculer le poids moyen pour chaque race, de même que l 'écart-type
3. est il raisonnable de considérer les distributions de poids dans chaque race comme des distributions gaussiennes ?

4. si on choisit un cabri au hasard peut-on raisonnablement estimer sa race en utilisant la distance euclidienne de son poids à la moyenne de chaque race ?

Supposons disposer d'une image satellite SPOT XS (3 canaux) dont on veuille classer les points parmi 6 classes Ci (i=1,6) représentant les seules classes possibles dans cette image.

Ces six classes étant définies par des régions échantillon (on a représenté par exemple la région échantillon de la classe C1 sur le schéma). .

On suppose de plus disposer d'une image des pentes en chaque point et d'une image représentant les orientations en chaque point.



On suppose que les différentes classes ne se répartissent pas n'importe comment par rapport à la pente et à l'orientation. De plus on suppose que les classes C1 C2 C3 sont trois fois plus présentes que les autres classes dans la région étudiée.

1) Proposez le classifieur le plus performant possible (en appliquant la décision bayésienne et en faisant l'hypothèse de lois de distribution gaussiennes (sans plus de précisions) pour chacune des classes) et donner suffisamment de détails pour permettre une programmation aisée.

2) Quelle simplification peut on apporter si les classes sont en fait toutes équiprobables et de matrices de covariances identiques et diagonales

---

Supposons disposer d'une image satellite SPOT XS (3 canaux XS1,XS2,XS3) dont on veuille classer les points parmi 4 classes de végétation Ci (i=1,4) représentant les seules classes possibles dans cette image.

Ces quatre classes étant définies par des régions échantillon (on a représenté par exemple la région échantillon de la classe C1 sur le schéma).

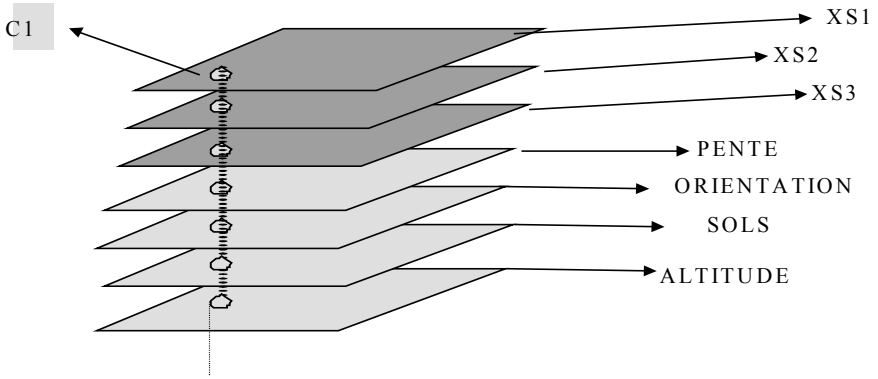
On suppose de plus disposer d'une image des pentes, d'une image représentant les orientations en chaque point, d'une image notant les types de sols par un numéro et du modèle numérique de terrain.

on suppose que les classes C1 C2 C3 sont trois fois plus présentes que la classe 4 dans la région étudiée.

1) Proposez le classifieur le plus performant possible (en appliquant la décision bayésienne et en faisant l'hypothèse de lois de distribution gaussiennes (sans plus de précisions) pour chacune des classes) et **en n'utilisant que les données SPOT**. Donner suffisamment de détails pour permettre un passage facile à la programmation.

2) Quelle simplification peut on apporter si les classes sont en fait toutes équiprobables et de matrices de covariance identiques et diagonales.

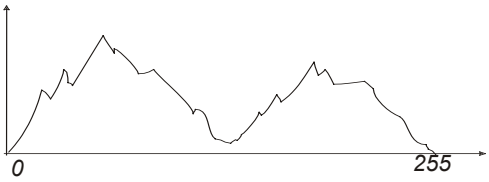
3) On suppose que les différentes classes ne se répartissent pas n'importe comment par rapport au contexte défini par « pente, orientation, type de sols et altitude ». Indiquer pourquoi on ne peut pas a priori faire l'hypothèse que les classes sont gaussiennes. Proposer de façon succincte un



classifieur prenant alors en compte le maximum de données.

4) En supposant maintenant traiter une autre image SPOT XS (avec n classes exhaustives) où toutes les classes ont des histogrammes (calculés sur les échantillons) dans XS1 quasiment « gaussiens » sauf la classe 2 qui a un histogramme de la forme suivante :

Indiquer comment procéder pour réaliser un classifieur bayésien utilisant les hypothèses de lois gaussiennes.



Module 18 - controle de TP (mai 97)

(traitement du signal et reconnaissance des formes)

On veut réaliser un programme de classification supervisée par la méthode dite des « bornes » (les classes sont supposées définies par les échantillons donnés ci-dessous).

Pour chaque classe et sur chaque canal on calcule les bornes minimales et maximales acceptables des niveaux de gris, de la façon suivante:

Pour chaque classe wi , on calcule pour chaque canal k le min_{wi} [k]et le max_{wi}[k] du niveau de gris des échantillons de la classe. Celà définit alors les bornes minimales et maximales.

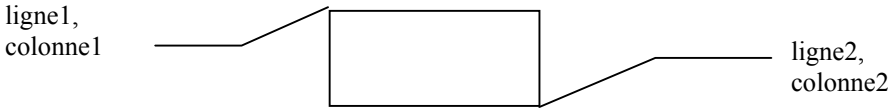
Un pixel quelconque est alors classé dans la classe wi si tous ses niveaux de gris sont compris dans les intervalles définis par les bornes minimales et maximales de la classe.

Un pixel peut donc être « non-classé », d'autre part il peut y avoir des pixels classés dans plusieurs classes alors on les considérera comme ambigus.

On affichera la classification en faisant apparaitre les points classés sans ambiguité dans l'une des classes.

Choix des échantillons :

On définit 3 classes sur l' image SPOT XS déja fournie de Nancy (images 512x512 notées etape01,etape02,etape03 sur le répertoire /home/wendling/Bandes_3) dont les échantillons sont définis par les rectangles suivants fournis:



classe	ligne1	colonne1	ligne2	colonne2
w1	390	83	405	94
w2	311	184	350	212
w3	437	462	451	476

Note:

vous pouvez utiliser Khoros en partie ou ....pas du tout et utiliser tous les programmes réalisés lors des Tps.

Supposons être dans l'espace R^2 muni de la distance euclidienne et disposer des échantillons suivants:
x_1 (2,2) x_2 (2,-2) x_3 (-2,-2) x_4 (-2,2) x_5 (0,0)

avec x_1 x_2 et x_3 appartenant à la classe w_1
et x_4 x_5 appartenant à la classe w_2

1) tracer la frontiére de décision suivant la règle du plus proche voisin
2) On note m1 et m2 les barycentres des échantillons des classes. Appliquer alors la règle du plus proche voisin aux barycentres et donner la nouvelle frontiére de décision obtenue. Commenter.

TP 1 - Reconnaissance des formes

Les entrées / sorties en C

getchar() permet de récupérer un caractère sur le "buffer d'entrée". Cette instruction renvoie le caractère le plus ancien se trouvant dans le buffer d'entrée. Lors de la lecture d'un texte caractère par caractère, ce buffer n'est rempli qu'au moment d'un retour-chariot.

putchar() permet d'envoyer un caractère sur le buffer de sortie.

printf("format", p1, p2, ...) permet d'envoyer à l'écran une chaîne de caractères. La commande est interprétée de la manière suivante :

- la chaîne de caractères qui compose le format est affichée jusqu'au premier caractère % qui peut y figurer ;

- Les caractères % et le (ou les) caractères qui suivent sont des spécificateurs de format. Ils sont remplacés par la valeur des variables situées après la chaîne de caractères.

scanf("format",&p1, &p2, ...) permet de lire au clavier différentes valeurs de variables.

Entrée / Sortie standard

stdin, stdout, stderr : entrée standard (clavier), sortie standard (écran) et sortie standard des erreurs (écran).

Pour forcer un affichage sur la sortie standard :

printf("affichage"); ou fprintf(stdout,"affichage");

Pour forcer une lecture sur l'entrée standard :

scanf("%d",&v); ou fscanf(stdin,"%d",&v);

Pour forcer les affichages, il peut être nécessaire de vider le buffer de sortie :

fflush(stdout);

Pour vider le buffer d'entrée et éviter la lecture d'une valeur obsolète, il peut être nécessaire de vider le buffer d'entrée :

fflush(stdin);

Spécificateurs de format

Les spécificateurs de formats pour les instructions printf et scanf sont :

% suivi d'un caractère qui indique le type de la conversion (pour l'affichage ou la lecture de la valeur d'une variable). Ce caractère peut être :

d -> décimaux
f -> réels signés
s -> chaînes de caractères
c -> caractères
u -> décimaux non signés
e -> flottant signés

\ suivi du caractère n pour un retour-chariot ou du caractère t pour une tabulation.

Les fichiers

La déclaration de l'identificateur d'un fichier est du type :

FILE *id_fichier;

L'ouverture d'un fichier s'effectue par l'appel à la fonction fopen :

id_fichier = fopen("nom_du_fichier_sur_le_support","mode");

Le mode est un ensemble de caractères indiquant le mode d'accès au fichier :

r -> lecture seule
w -> création en écriture
a -> concaténation
...

on peut préciser le mode en indiquant si on lit du binaire ou du texte (b ou t)

La fermeture d'un fichier se fait par l'appel à la fonction fclose :

fclose(id_fichier);

L'écriture formatée dans un fichier s'effectue à l'aide de la fonction fprintf :

fprintf(id_fichier, format, p1, p2, ...);

La lecture formatée s'effectue à l'aide de la fonction fscanf :

fscanf(id_fichier, format, &p1, &p2, ...);

La lecture par blocs dans un fichier s'effectue à l'aide de la fonction fread, d'en-tête :

size_t fread(void *tab, size_t t, size_t ninfo, FILE *fich);

tab est un pointeur sur la zone mémoire recevant les informations lues ;
t est la taille de chaque information lue (en octets) ;
ninfo est le nombre d'informations à lire ;
fich est l'identificateur du fichier, de type FILE *, sur lequel on opère la lecture. Cet identificateur ne doit pas être confondu avec le nom du fichier.

L'écriture dans un fichier s'effectue à l'aide de la fonction fwrite, d'en-tête :

size_t fwrite(void *tab, size_t t, size_t ninfo, FILE *fich);

La fonction fwrite renvoie le nombre d'informations effectivement écrites. Ce nombre d'informations peut être inférieur au nombre requis, en cas d'erreur.

Pour se positionner en un certain endroit d'un fichier, on utilise la fonction fseek, d'en-tête :

int fseek(FILE *fich, long decalage, int depart);

fich est l'identificateur du fichier dans lequel on cherche à se positionner ;
decalage est le nombre d'octets dont on doit se décaler à partir de la position donnée par depart ;
depart peut être égal à :
0 ou SEEK_SET (début) ;
SEEK_CUR (position courante dans le fichier) ;
SEEK_END (fin du fichier).

Attention : fseek renvoie une valeur non nulle en cas d'erreur (c'est-à-dire si on déborde du fichier).

La fonction feof, d'en-tête :

int feof(FILE *fichier);

renvoie une valeur non nulle si la fin du fichier est atteinte.

Exercice 1:

1.Créer un fichier de texte MNT contenant les 100 valeurs entières strictement positives suivantes à l'aide d'un éditeur de texte :

01 02 23 45 56 58 62 77 98 100

03 04 06 32 50 72 81 74 82 99

31 05 07 08 35 80 51 91 90 84

59 33 11 10 09 38 73 52 92 89

78 79 36 13 14 12 30 83 53 54

61 68 87 39 16 17 15 37 55 93

69 60 44 85 41 18 19 20 34 88

86 43 67 66 75 37 21 24 22 40

64 44 76 57 48 49 40 26 27 25

70 63 46 47 65 71 56 34 28 29

2.Ecrire un sous-programme C permettant de lire le fichier précédent et de stocker les données dans une matrice de taille 10*10

3.Tester le sous-programme précédent à l'aide d'un programme principal qui affiche les valeurs de la matrice.

Solution :
==========

```
#include "stdio.h"

#define TAILLE 10

typedef int mat[TAILLE][TAILLE];

void lire_fichier(mat m)
/* Lecture dans le fichier "mnt" */
{
      FILE *f;
      int i,j;

      if((f = fopen("mnt","r"))==NULL) printf("Lecture impossible !\n");
      else
      {
```

```
            fprintf(f,"Lecture possible :\n");
            for (i=0;i<TAILLE;i++)
                  for (j=0;j<TAILLE;j++)
                        fscanf(f,"%d",&m[i][j]);
      }
      fclose(f);
}

main()
{
      mat matrice;

/* Lecture du fichier */
      lire_fichier(matrice);
/* impression de mat[ ][ ] */
for (i=0;i<TAILLE;i++)
                  {
                     for (j=0;j<TAILLE;j++)
                        printf(f,"%d",m[i][j]);
                  printf("\n");
                   }

}
```

Exercice 2:

Créer un tableau de 20 entiers int (2lignes et 10 colonnes)
Et sauvegarder ce tableau dans un fichier puis relire les données à partir du fichier et les afficher sur écran pour vérification.

Solution :
==========

```
#include "stdio.h"

#define TAILLE 10

typedef int mat[TAILLE][TAILLE];

void ecrire_fichier(int table[2][10])
/* ecriture dans le fichier "sauvegarde" */
{
      FILE *f;

      if((f = fopen("sauvegarde","wb"))==NULL) printf("Ecriture impossible !\n");
      else
          fwrite(void *table, 2, 20, f);

      fclose(f);
}

void lire_fichier(int table[2][10])
/* lecture dans le fichier "sauvegarde" */
{
      FILE *f;

      if((f = fopen("sauvegarde","rb"))==NULL) printf("lecture impossible !\n");
      else
          fread(void *sortie, 2, 20, f);
```

```
      fclose(f);
}

main()
{
     int tab [2][10],sortie[2][10];
  /* Création du fichier */
for (i=0;i<2;i++)
                  for (j=0;j<10;j++)
                     tab[i][j]=i*10+j;
/* Création du fichier */
      écrire_fichier(tab);
/* Lecture du fichier */
      lire_fichier(sortie);
/* impression de sortie[ ][ ] */
for (i=0;i<2;i++)
                   {for (j=0;j<10;j++)
                         printf(f,"%d",sortie[i][j]);
                   printf("\n");
                    }

}
```

Exercice 3:

Créer sur votre répertoire de travail le fichier image fruits.bmp.
Il s'agit d'un fichier .bmp d'une image couleur RVB 24 bits.

Cette image comporte en fait 61 lignes et 164 colonnes et chaque pixel comporte en fait 3 valeurs (unsigned char) représentant les trois couleurs rouge, vert, bleu.
Le fichier commence par un en-tête de 54 octets.
Ce fichier a donc pour taille: 54+61x164x3=30006 octets.

Si on veut lire l'image on saute donc les 54 premiers octets puis on lit successivement les pixels (3 octets représentant les couleurs R V B).

Donner le contenu de l'en tête et l'interpréter (voir annexe)

Visualiser cette image avec xv ou xview (?)

ANNEXE: contenu de l'entête

```
struct bmpHeaderType
                  {
                  char magic1, magic2; // identity 'BM'
                  long fileSize;       // size of file in bytes
                  long reserved;       // always 0
                  long pixelOffset;    // offset of data
                  long bmiSize;        // remaining header size
                  long cols, rows;     // width and height
                  int  nPlanes;        // number of color planes
                  int  bitsPerPixel;
                  long compression, cmpSize;   // compressed?
                  long xScale, yScale;
                  long colors, impColors;
                  };
```

TP 2 - Reconnaissance des formes

Le fichiers fruits2.bmp que vous avez sur votre répertoire de travail est en fait un fichier RVB 24 bits de type bitmap.

Il s'agit d'une image couleur où chaque pixel est codé sur 3 octets (un octet par couleur fondamentale R,V,B) sa taille est de 61 lignes et de 164 colonnes.

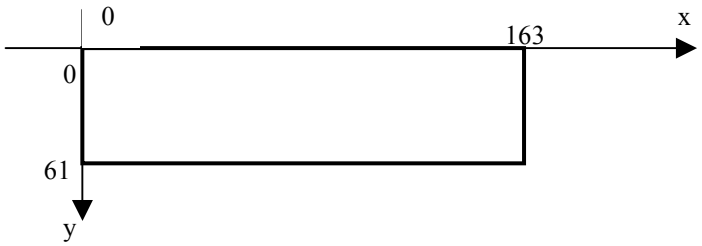
Le fichier comporte un entête de 54 octets dont le contenu est le suivant:

```
struct bmpHeaderType
        {
        char magic1, magic2; // identity 'BM'
        long fileSize;       // size of file in bytes
        long reserved;       // always 0
        long pixelOffset;    // offset of data
        long bmiSize;        // remaining header size
        long cols, rows;     // width and height
        int  nPlanes;        // number of color planes
        int  bitsPerPixel;
        long compression, cmpSize;   // compressed?
        long xScale, yScale;
        long colors, impColors;
        };
```

Cette image représente des fruits (raisins verts, raisins rouges et abricots). On suppose qu'il n'y a rien d'autre dans l'image.

Les formes à classer seront les pixels et chaque pixel appartient donc à l'une des trois classes possibles.

Les trois classes sont définies par des ensembles de pixels échantillons.
En supposant que l'image est représentée dans le repère suivant:



Echantillons "raisin vert" le rectangle défini par les "extrémités" (61,21) , (77,34)

Echantillons "raisin rouge" le rectangle défini par les "extrémités" (10,13) , (23,25)

Echantillons "abricots" le rectangle défini par les "extrémités" (133,2) , (147,11)

Exercice 1

En supposant que les classes sont représentées par le centre de gravité des échantillons qui les définissent .

Classer tous les points de l'image en supposant que la classe à choisir est celle pour laquelle la distance euclidienne entre le pixel à classer et le centre de gravité de la classe est minimum.

On créera le fichier image classée en affectant une couleur différente à chaque classe et on visualisera sur écran.

Exemple raisin rouge: R=255 V=0 B=0
Raisin vert : R=0 V=255 B=0
Abricot : R=255 V=128 B=128

3 METHODES DES « NUEES DYNAMIQUES »

Il s'agit de trouver sur l'ensemble I des formes à classer , une partition en k classes satisfaisant un critère global de qualité.

La méthode consiste à partir de « noyaux » initiaux autour desquels on agrège les formes de I pour obtenir une partition. Puis à partir de cette partition on cherche de nouveaux « noyaux » capables de générer une meilleure partition... et on itére le processus jusqu'à la stationnarité.

3.1.1.1 Formalisation de la méthode

Soient

- I l'ensemble des formes à classer
- P(I) l'ensemble des parties de I
- L un ensemble de k noyaux A_i

Nous appellerons « étalons » les éléments constitutifs de chaque noyau.

La méthode des nuées dynamiques consiste à trouver deux applications f et g telles que:

` L=g(P) g permet de passer d'une partition P à k noyaux

P=f(L) f permet de passer d'un ensemble de k noyaux à une partition P

Si nous notons P_i la partition obtenue à l'itération n° i alors:

$$P_{i+1} = [f \bullet g](P_i) \quad et \quad donc \quad P_{i+1} = [f \bullet g]^{i+1}(P_0)$$

Choix des fonctions f et g :

Le choix des fonctions f et g est réalisé en suivant les choix suivants permettant d'assurer une convergence.

P=f(L) avec $P_i = \left\{ x / x \in I \quad et \quad d(x, A_i) \leq d(x, A_j) \ \ \forall j \neq i \right\}$

soit chaque forme est associée au noyau le plus proche.

L=g(P) $L = \{A_1, A_2, ..., A_k\}$

avec A_i étant un ensemble de n_i éléments qui minimisent une fonction R(x,I,P)

(les éléments correspondants peuvent être ou non des éléments de l'ensemble I)

R(x,I,P) donne une notion de « distance » de la forme x à la classe i de la partition P

- exemple possible $R(x,I,P)= d(x,P_i)$ si $P=\{P_1,P_2,....,P_k\}$

On peut alors définir un critère de qualité de la partition P autour d'un ensemble de noyaux L par

$$\omega(V) = \sum_{i=1}^{k} \left(\sum_{x \in A_i} R(x, I, P) \right)$$

A la suite $V_0, V_1, ...V_n$

$$\begin{cases} V_i = (L_i, P_i) \\ V_{i+1} = (g(P_i), f(g(P_i))) \end{cases}$$

on associe la suite:

$$u_0 = \omega(V_0) \quad u_1 = \omega(V_1) \quad ..... \quad u_n = \omega(V_n)$$

En général u_n décroît et on arrive à un état stationnaire pour n>N

3.1.1.2 Exemples: ISODATA1, ISODATA2

ISODATA1:

On a donc un ensemble I de formes x à classer (on suppose que x appartient à R^d).

On suppose alors que chaque classe a une densité de probabilité définie parfaitement par la moyenne de la classe correspondante.

La méthode suit l'algorithme suivant

1 - Choisir le nombre de classes k
2 - choisir un ensemble aléatoire initial de moyennes (appartenant pas forcément à I)
$\mu_1,\mu_2, \ldots\ldots\ldots,\mu_k$
3 - Affecter chaque x de I à la classe w_i telle que
$d(x,\mu_i)<d(x,\mu_j)$ pour tout $j\neq i$ on obtient des partitions P_i
4 - Recalculer les nouvelles moyennes de chaque classe
5 - Si non stationnarité aller en 3
6 - Arrêt

Exemple dans R^2 avec deux classes:

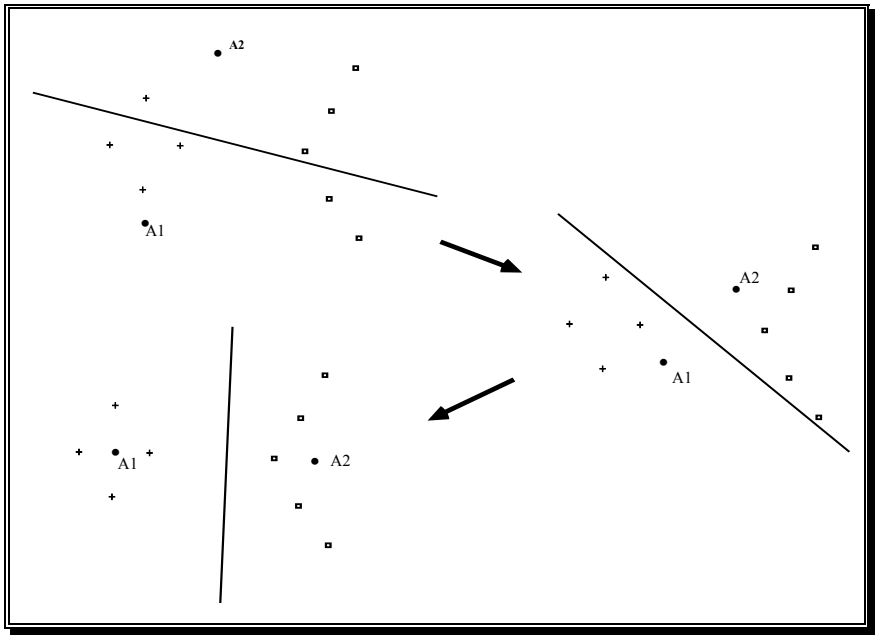


Figure 26 - ISODATA1 avec deux classes dans R^2

ISODATA2:

Dans ce cas on tient compte de la moyenne des classes (lors des itérations successives il s'agira des partitions Pi) mais aussi de la matrice de covariance.

D'où la méthode:

1 - Choisir le nombre de classes k
2 - Choisir pour chaque classe une moyenne aléatoire et une matrice de covariance aléatoire:

$$\left(\mu_1,\sum_1\right)\left(\mu_2,\sum_2\right)\ldots\ldots\ldots\ldots\left(\mu_k,\sum_k\right)$$

3 - Affecter chaque forme x à la partition Pi telle que la distance de Mahanalobis a cette classe soit minimale, soit

$$d_{mahanalobis}(x, A_i) \leq d_{mahanalobis}(x, A_j) \quad \forall j \neq i$$

$$avec \quad d_{mahanalobis}(x, A_i) = (x - \mu_i)^t \times \sum_i^{-1} \times (x - \mu_i)$$

4 - Recalculer les

$$(\mu_1, \sum_1)(\mu_2, \sum_2) \ldots\ldots\ldots\ldots\ldots (\mu_k, \sum_k)$$

5 - Si non stationnarité aller en 3

6 - Arrêt

3.2 EXERCICES

1- Décrire la méthode ISODATA1 en classification automatique

2- Appliquer graphiquement au cas où on recherche deux classes dans l'ensemble de points définis dans R^2 par (-1,1) , (-1,2), (1,1), (2,0), (2,1).
On considérera avoir les deux noyaux initiaux (1,1), (1,2)

3- Quelle différence y a t'il entre la méthode ISODATA1 et la méthode ISODATA2

En prenant des noyaux à un élément et en choisissant de rechercher deux classes de formes définies dans R^2 (par quelques points : (-1,1)(-1,2)(1,1)(2,0)(2,1)) , appliquer graphiquement l'algorithme des nuées dynamiques en prenant deux noyaux de départ ((1,1)(1,2)).

---

1) Décrire les méthodes particulières de classification non supervisées appelées ISODATA1 et ISODATA2.
2) Appliquer ISODATA1 aux formes suivantes de R^2 à classifier en deux classes :
(-2,1) (-1,2) (0,-2) (0,-1) (0,1) (0,2) (0,3) (1,-1),
avec les noyaux de départ (-2,-1) et (2,3).

---

1. Décrire les méthodes particulières de classification non supervisées appelées ISODATA1 et ISODATA2.

2. Appliquer ISODATA1 aux formes suivantes de R^2 à classifier en deux classes :
(-2,1) (-1,2) (0,-2) (0,-1) (0,1) (0,2) (0,3) (1,-1),
avec les noyaux de départ (-2,-1) et (2,3).

---

4 METHODES GEOMETRIQUES

4.1 INTRODUCTION

Dans ce type d'approches on va considérer l'espace des formes (R^d) comme un espace géométrique et la classification va être considérée comme faisant intervenir des métriques dans cet espace ou des hypersurfaces de séparation entre régions censées représenter les diverses classes cherchées.

Si les surfaces de séparation sont des hyperplans (équations linéaires) alors on parlera de fonctions discriminantes linéaires et le classifieur induit sera qualifié de machine linéaire. On parlera alors de séparation linéaire des classes.

4.2 SEPARATION LINEAIRE

On est alors dans le cas de fonctions discriminantes qui sont linéaires et les surfaces de séparation entre classes seront des hyperplans.

Ce type de classifieur est du type supervisé, c'est à dire que les classes sont définies par des échantillons. A partir de ces échantillons il faut arriver à trouver les surfaces de séparation entre classes qui définiront le classifieur. Bien entendu on fait là l'hypothèse que les classes sont linéairement séparables, c'est à dire que les paquets d'échantillons définissant les différentes classes recherchées peuvent être « séparés » par des hyperplans! Si ce n'est pas le cas on cherchera à trouver les hyperplans qui minimisent les erreurs entre les classes définies par leurs échantillons.

4.2.1 Surfaces de décision et fonctions discriminantes linéaires

4.2.1.1 Cas de deux classes

Ou bien chaque classe est définie par une fonction discriminante linéaire ($g_1(x)$ pour la classe w_1 et $g_2(x)$ pour la classe w_2) . Ces deux fonctions ayant été obtenues à partir des échantillons des deux classes suivant une méthode ad hoc.

Dans ce cas pour une forme inconnue x on décidera que x appartient à la classe w1 si g1(x) > g2(x) sinon x appartient à la classe w2.

Ou bien on considère qu'on dispose d'un fonction g(x) linéaire (par exemple g1(x)-g2(x) si on dispose de ces deux fonctions) qui pourra donc s'écrire:

$$g(x) = \omega^t \times x + \omega_0$$

$$avec \begin{cases} \omega & \textbf{appelé vecteur poids} \\ \omega_0 & \textbf{appelé poids du seuil} \end{cases}$$

La règle de décision associée étant la suivante:

$$\begin{cases} x \in w_1 \quad si \quad g(x) > 0 \\ x \in w_2 \quad si \quad g(x) < 0 \end{cases}$$

Ce qu'on peut comprendre comme :

x est assigné à la classe w1 si $\omega^t x$ dépasse le seuil $-\omega 0$

Dans le cas où g(x)=0 on peut considérer que x est non classé par exemple...

L'équation g(x)=0 définit en fait la surface de décision. Voici par exemple ce que cela donne dans R^2 :

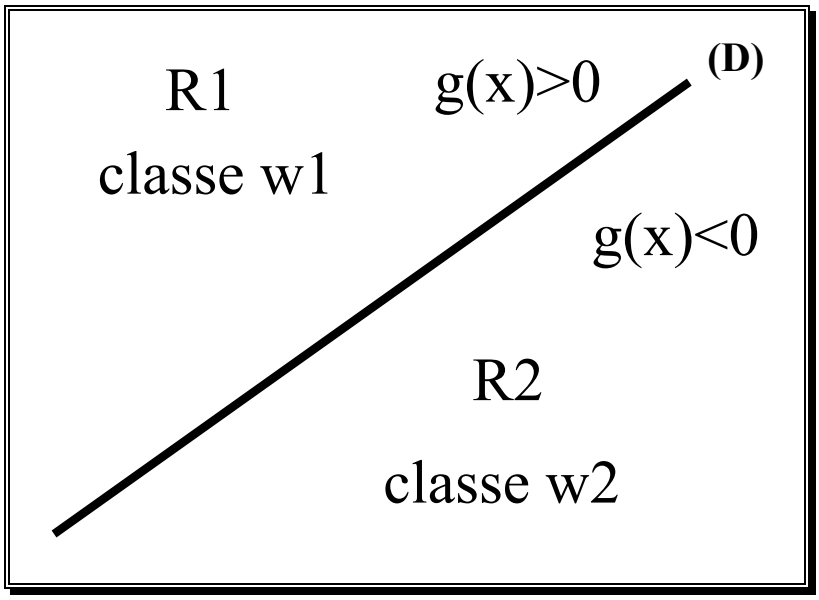


Figure 27 - Séparation linéaire dans R2 pour deux classes

4.2.1.1.1 Un peu de géométrie..

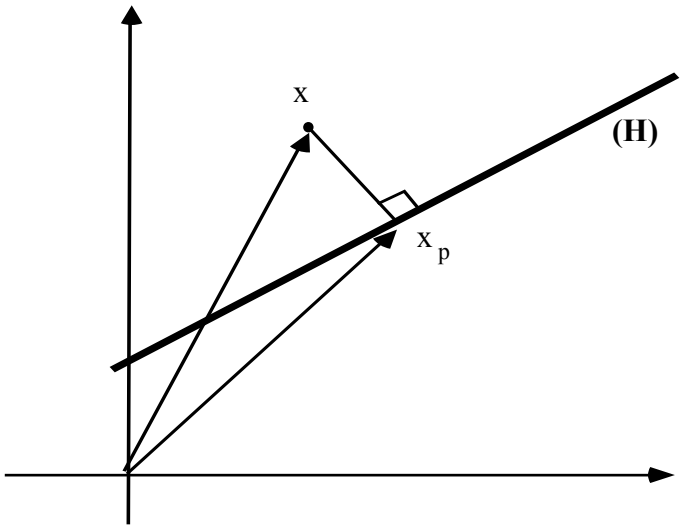
Supposons que les formes x1 et x2 distinctes soient sur la surface de décision entre les deux classes, alors on a:

$$\omega^t \times x_1 + \omega_0 = \omega^t \times x_1 + \omega_0 = 0$$

$$soit \quad \omega^t \times (x_1 - x_0) = 0$$

ce qui signifie que ω est orthogonal à l'hyperplan séparateur d'équation g(x)=0 puisqu'il est perpendiculaire à tout vecteur de cet hyperplan.

Soit maintenant une forme x quelconque et calculons sa distance euclidienne à l'hyperplan séparateur.



ω est un vecteur perpendiculaire à l'hyperplan. de norme

$$\|\omega\| = \sqrt{\omega^t \times \omega}$$

Alors $$x = x_p + r \times \frac{\omega}{\|\omega\|}$$

si x_p est la projection de x sur l'hyperplan H.

$$or \quad \omega^t \times x_p + \omega_0 = 0$$

$$donc \quad g(x) = \omega^t \times (x) + \omega_0 = \omega^t \times \left(x_p + r \times \frac{\omega}{\|\omega\|} \right) + \omega_0$$

$$soit \quad g(x) = r \times \|\omega\|$$

d'où

$$\boxed{r = \frac{g(x)}{\|\omega\|} \quad soit \quad d(x,H) = \frac{|g(x)|}{\|\omega\|}}$$

4.2.1.2 Cas de c classes

On peut alors considérer diverses approches pour construire le classifieur , notamment suivant que l'on considère que l'on a c problèmes du type « séparer $w_i \quad de \quad \overline{w}_j$ » ou bien $C_n^2 = \frac{c \times (c-1)}{2}$ problèmes du type « séparer $w_i \quad de \quad w_j$ ».

Voir ci dessous les deux cas indiqués pour 3 classes dans R^2:

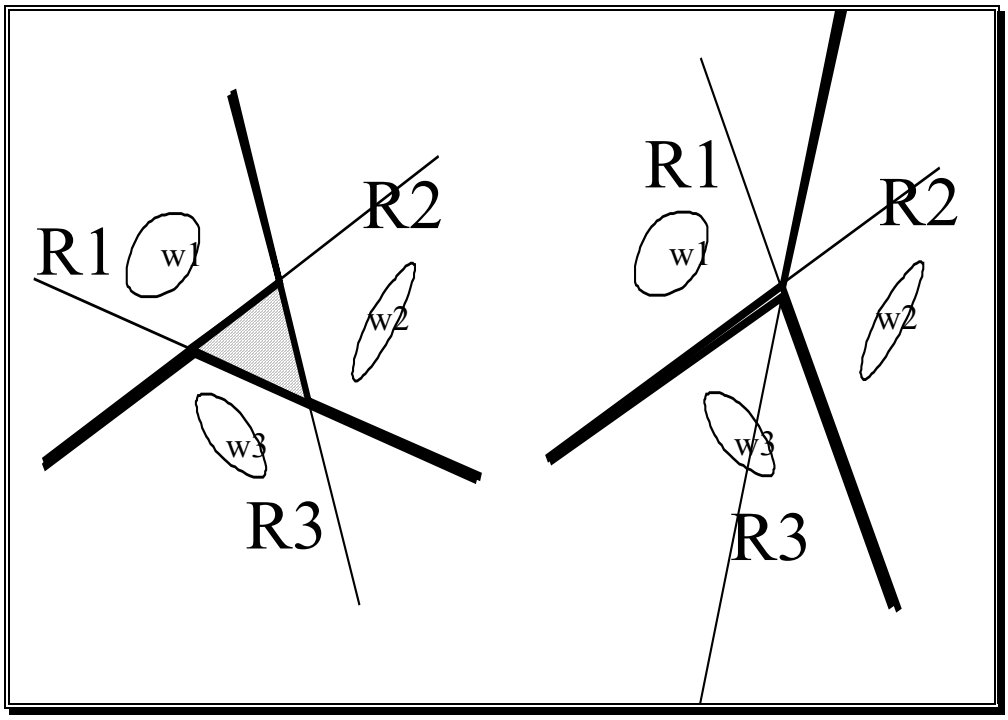


Figure 28 - choix de dichotomies pour un classifieur linéaire

4.3 MACHINE LINEAIRE

4.3.1 Introduction

On appelle machine linéaire un classifieur construit sur la base de fonctions discriminantes linéaires et sur des dichotomies de types « w_i - w_j ». Les surfaces de séparation sont donc des hyperplans.

4.3.2 Description de la machine linéaire

On suppose disposer de c fonctions discriminantes linéaires notées:

$$g_i(x) = \omega_i^t \times x + \omega_{i0} \quad pour\ i = 1,.....,c$$

Ces fonctions auront été obtenues par apprentissage sur les échantillons fournis pour les classes à rechercher (cet apprentissage est étudié plus loin).

La Machine linéaire est alors définie par le type de décision suivant:

$$\boxed{x \in w_i \quad si \quad g_i(x) > g_j(x) \quad \forall i \neq j}$$

Deux régions contiguës R_i et R_j sont séparées par une portion d'hyperplan H_{ij} défini par l'équation

$$g_{i(}x)=g_j(x)$$

soit

$$\boxed{\begin{array}{l} \left(\omega_i - \omega_j\right)^t \times x + \left(\omega_{i0} - \omega_{j0}\right) = 0 \\ \left(\omega_i - \omega_j\right) \perp H_{ij} \\ et \quad d(x, H_{ij}) = \dfrac{\left|g_i(x) - g_j(x)\right|}{\left\|\omega_i - \omega_j\right\|} \end{array}}$$

Notons que:

- Toutes les régions Ri (i=1,...,c) ne sont pas forcément adjacentes

- Les régions sont connexes et convexes (deux points quelconques d'une région définissent un segment qui appartient totalement à la région correspondante)

Ce classifieur est en fait adapté au cas où les lois de densités de probabilités $p(x/w_i)$ sont unimodales.

Exemples:

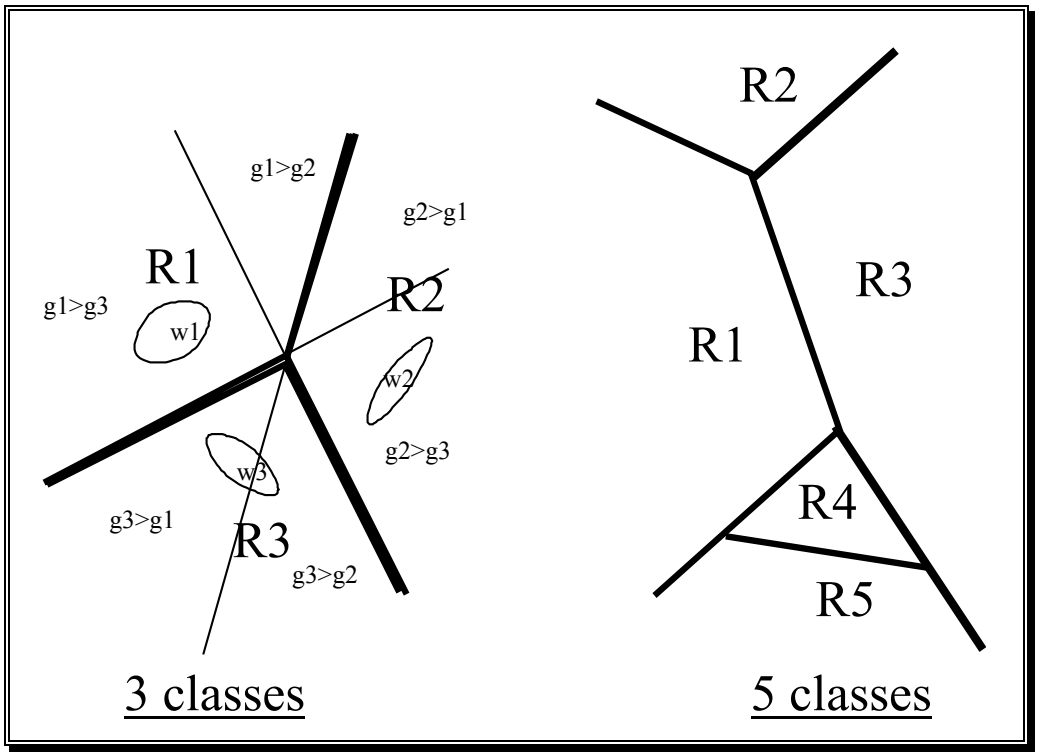


Figure 29 - Exemples de machines linéaires dans R^2

4.4 APPRENTISSAGE

4.4.1 Introduction

Les classes à rechercher sont donc définies par la fourniture d'échantillons de chacunes des classes. C'est à partir de ces ensembles d'échantillons qu'il faut « découvrir » les fonctions discriminantes caractéristiques de chacunes des classes. Dans la mesure où nous allons nous placer dans une approche où on cherche à séparer les classes deux à deux, nous allons étudier le problème de l'apprentissage dans le cas de deux classes et nous cherchons à déterminer la fonction g(x) telle que g(x)=0 définira l'hyperplan séparateur entre les classes.

4.4.2 Méthode « géométrique »

On cherche donc une fonction linéaire discriminante g(x) entre les deux classes w1 et w2 telle que

$$g(x) > 0 \Leftrightarrow x \in w1$$
$$g(x) < 0 \Leftrightarrow x \in w2$$

Cette fonction étant linéaire peut s'écrire :

$$g(x) = \omega^t x + \omega_0 \quad avec \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_d \end{bmatrix} \quad et \quad \omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \cdot \\ \cdot \\ \omega_d \end{bmatrix}$$

Passons artificiellement de l'espace des formes R^d à l'espace R^{d+1} en rajoutant une composante égale à 1 et notons

$$y = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_d \end{bmatrix} = \begin{bmatrix} 1 \\ x \end{bmatrix} \quad et \quad a = \begin{bmatrix} \omega_0 \\ \omega_1 \\ \omega_2 \\ \cdot \\ \cdot \\ \omega_d \end{bmatrix} = \begin{bmatrix} \omega_0 \\ \omega \end{bmatrix}$$

La fonction discriminante cherchée s'écrit donc maintenant

$$g(x)=g(y)=a^t y$$

La surface discriminante correspondante dans R^{d+1} est en fait un hyperplan passant par l'origine et dont l'équation est donnée par $a^t y=0$

On dispose donc de n échantillons définis dans R^{d+1} répartis entre les classes w_1 et w_2 que nous noterons y_1 y_2....y_n (ces échantillons sont obtenus simplement en rajoutant une composante constante égale à 1 aux échantillons fournis).

Faire l'apprentissage du classifieur va donc consister à rechercher g(y) et donc un vecteur a qui fasse que les n échantillons seront correctement classés. Il n'y a bien sûr une solution que si les échantillons des deux classes sont linéairement séparables.

Un échantillon y_i est classé correctement si l'une des deux conditions suivantes est vraie:

$$a^t \times y_i > 0 \quad et \quad y_i \in w_1$$
$$a^t \times y_i < 0 \quad et \quad y_i \in w2$$

Or

$$a^t \times y_i < 0 \quad \Leftrightarrow \quad a^t \times (-y_i) > 0$$

On peut donc simplifier le problème en remplaçant tous les échantillons y_i de la classe w_2 par leurs opposés ($-y_i$)

La recherche du vecteur a (permettant de définir l'hyperplan séparateur des deux classes) se trouve donc simplifiée de la façon suivante:

Rechercher un vecteur a de R^{d+1} tel que $a^t y_i > 0$ pour tous les échantillons (en prenant les opposés pour la classe w2)

Chaque échantillon y_i définit en fait une contrainte sur le vecteur solution a. Cette contrainte va en fait définir une zone de R^{d+1} dans laquelle le vecteur a ne peut pas se trouver (et de façon duale une zone dans laquelle il est permis qu'il se trouve!).

$a^t y_i = 0$ définit un hyperplan passant par l'origine de l'espace R^{d+1} ayant le vecteur y_i comme normale.

Le vecteur a solution doit donc être du coté « positif » de cet hyperplan (c'est à dire du coté du vecteur y_i)

Les n échantillons définissent en fait n demi-espaces , et c'est à l'intersection de ces n demi espaces que l'on peut trouver le vecteur a.

Voici un exemple de formes définies dans R, représentées dans R^{1+1}. On suppose avoir deux échantillons y_1 et y_2 représentant la classe w_1 et deux échantillons y_3 et y_4 représentant la classe w_2

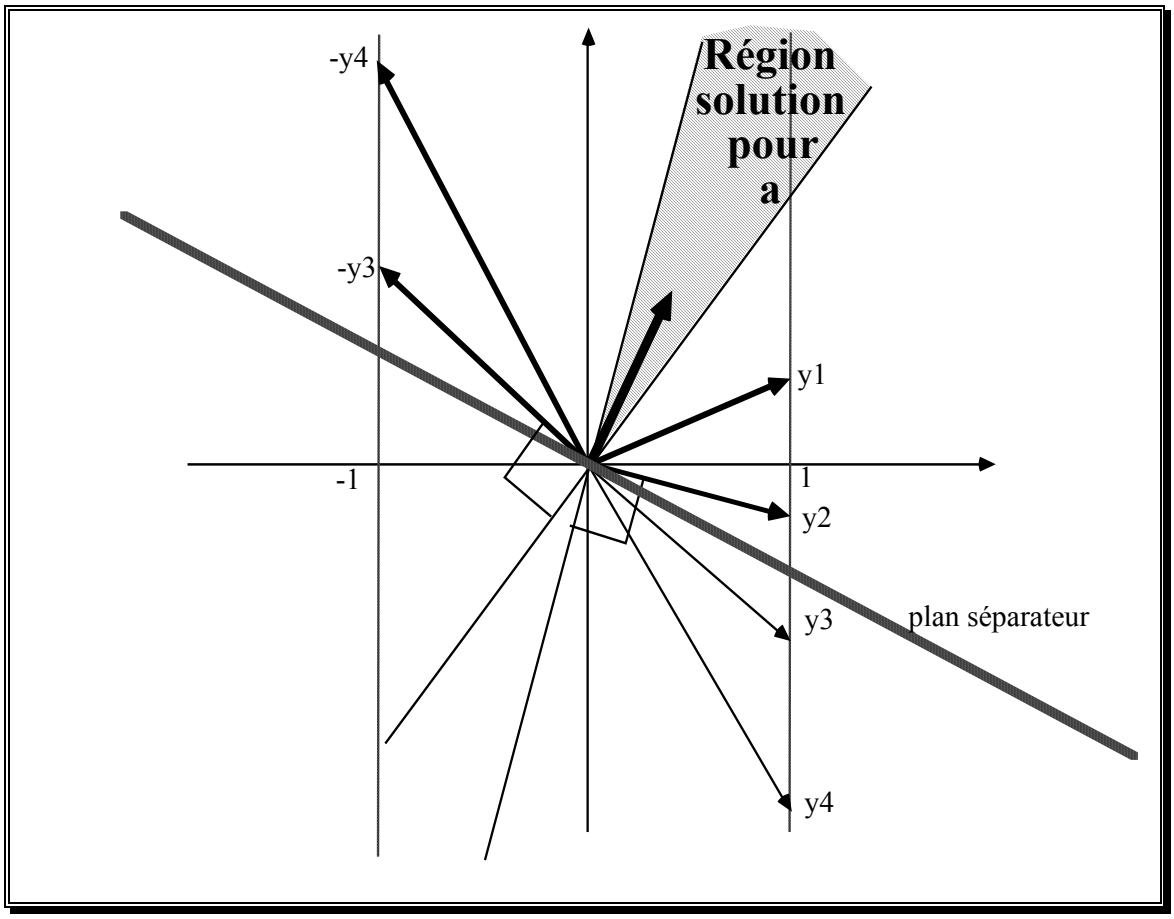


Figure 30 - apprentissage "géométrique" pour une machine linéaire

Il y a toujours la solution triviale a vecteur nul mais qui ne présente aucun intérêt bien sûr.

Si les deux classes sont linéairement séparables (suivant les n échantillons fournis !) alors on voit qu'il y a une infinité de vecteurs solutions a, on peut alors chercher une solution optimale.

On peut par exemple:

- choisir le vecteur a (définissant un hyperplan d'équation $a^t y=0$)
qui rende maximum la distance minimum des échantillons au plan séparateur.

- Chercher le vecteur a le plus petit satisfaisant

$$a^t \times y_i > b \quad \forall i$$

$$b \quad \textbf{Cte positive appelée marge}$$

Ce choix permet d'éviter les vecteurs solution limites.

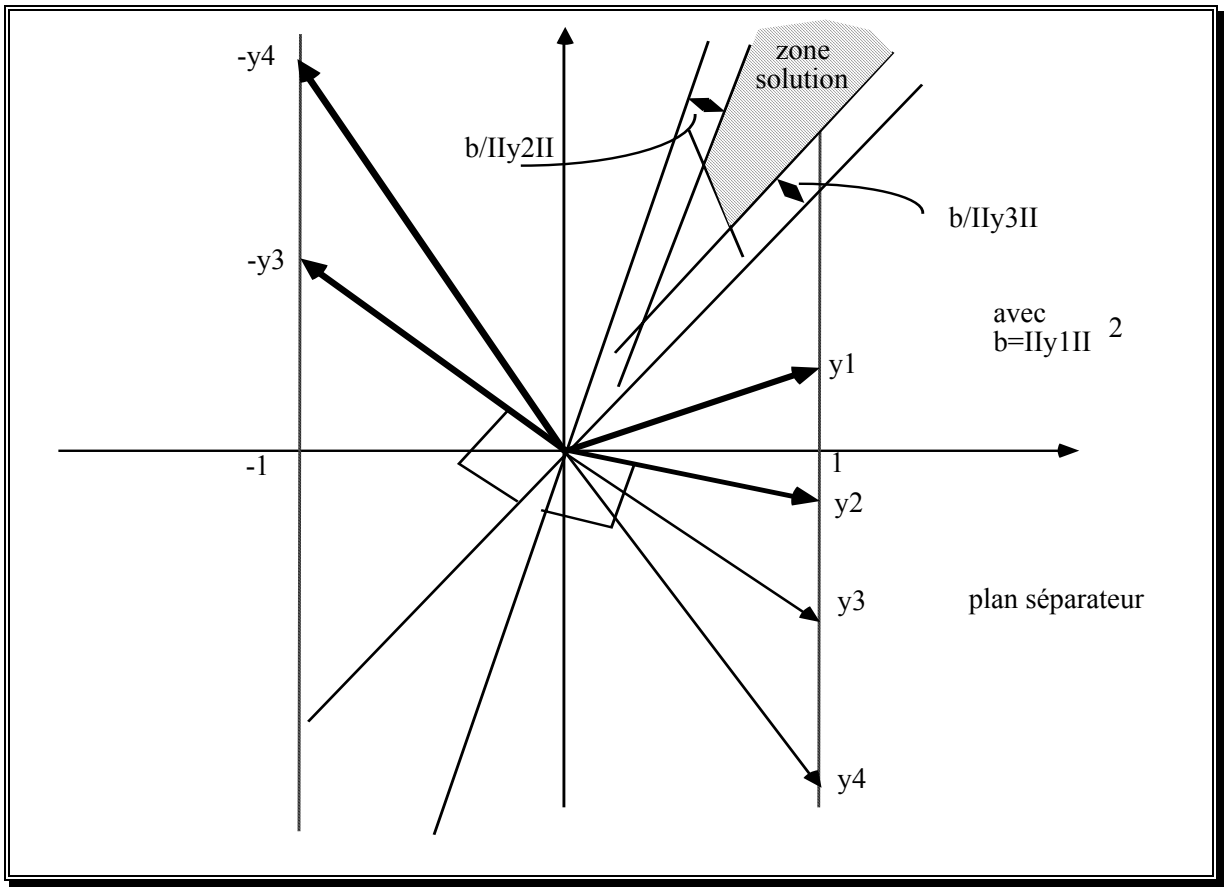


Figure 31 - Apprentissage géométrique avec marge

L'inconvénient de cette approche purement géométrique est qu'on ne peut pas trouver facilement de solution approchée si les échantillons des deux classes ne sont pas linéairement séparables.

4.4.3 methode de descente du gradient

4.4.3.1 Introduction

Le problème n'est plus abordé sous forme de n contraintes géométriques successives, mais sous la forme analytique suivante:

Trouver la solution à l'ensemble d'inégalités $\{ a^t y_i > 0 \}$ et pour cela définir un critère J(a) qui devra être minimum pour a solution.

on est alors confronté aux problèmes suivants:

- Trouver une fonction J(a) à minimiser

- Définir une méthode permettant de trouver le vecteur a qui minimise J(a)

4.4.3.2 Approche théorique

4.4.3.2.1 Choix du critère J(a)

Il y a plusieurs fonctions J(a) possibles qui sont basées sur la sommation des « contraintes » de la façon suivante:

Critère du perceptron:

$$J(a) = \sum_{y \in Y(a)} \left(-a^t y\right)$$

avec Y(a) : ensemble des échantillons mal classés si on choisit a

Procédure de relaxation:

$$J(a) = \sum_{y \in Y(a)} \left(a^t y\right)^2$$

autre...

$$J(a) = \#\left(Y(a)\right)$$

nombre d'échantillons mal classés si on choisit a

4.4.3.2.2 Détermination de a par « descente du gradient »

L'objectif est de trouver la valeur de a qui va minimiser la fonction J(a). On s'appuie sur la condition nécessaire qui est que la dérivée de J(a) doit être nulle. Le risque est cependant de tomber sur un minimum local de la fonction J(a) .

Le principe de la méthode de descente du gradient consiste à partir d'un vecteur arbitraire a_1 de calculer la dérivée en ce point et de descendre suivant la plus grande pente indiquée justement par la dérivée. En « descendant » ainsi sur l'hypersurface on finira par tomber dans un minimum de J(a) (trou!).

Voici l'algorithme de la méthode:

- Choisir un vecteur arbitraire a1
- Calculer le vecteur dérivée $\nabla J(a_1)$
- Calculer la nouvelle valeur a2 en se déplaçant à partir de a1 suivant la descente la plus grande

et itérer jusqu'à convergence en suivant la formule suivante:

$$\boxed{a_{k+1} = a_k - \rho_k \times \nabla J(a_k)}$$

avec $\rho_k > 0$ définissant l'échelle de descente.

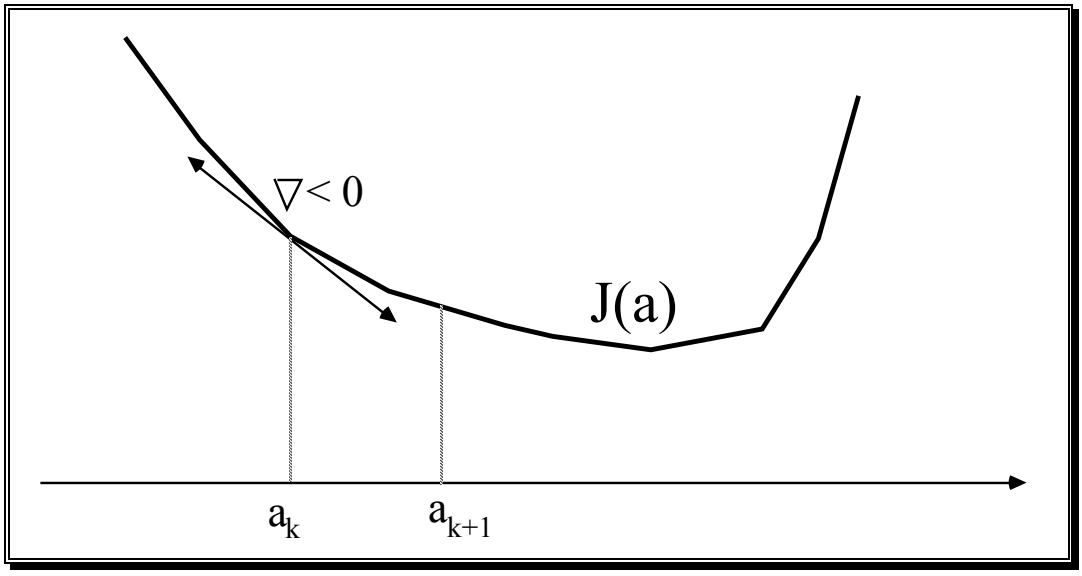


Figure 32 - Minimum d'une fonction J(a) par descente du gradient

On imagine facilement que le choix de ρ_k joue un rôle important dans la convergence de l'apprentissage et sur sa rapidité.

Ainsi si ρ_k est trop petit la convergence peut être extrêmement lente et d'autre part si on est prêt d'un minimum local on risque de s'y retrouver « piégé » !

Si ρ_k est trop grand on risque de diverger (en effectuant des déplacements trop grands)

Empiriquement on voit qu'il peut être intéressant d'avoir des valeurs de ρ_k élevées au début et qui vont diminuer au fur et à mesure (on peut ainsi éviter les piéges des minima locaux de la fonction J(a) à l'aide des valeurs élevées initiales et assurer une convergence fine sur le minimum global de la fonction J(a).

Il existe une méthode permettant de choisir une valeur optimum de ρ_k qui tienne compte de la dérivée et d'autre part de la courbure locale de J(a) par l'intermédiaire de la dérivée seconde.

En effet si l'on réalise le développement en série d'ordre 2 de la fonction J(a) on obtient:

$$J(a) \approx J(ak) + \nabla J^t(a_k) \times (a - a_k) + \frac{1}{2} \times (a - a_k)^t \times D \times (a - a_k)$$

$$avec \quad D = \left\{ \frac{\delta^2 J(a_k)}{\delta a_i \delta a_j} \right\}_{i=1,d+1 \quad j=1,d+1}$$

matrice des dérivées partielles secondes

Alors on obtient

$$J(a_{k+1}) \approx J(a_k) - \rho_k \|\nabla J\|^2 + \frac{1}{2} \times \rho_k^2 \times \nabla J^t \times D \times \nabla J$$

Si on cherche la valeur de ρ_k telle que $\frac{\delta J(a_{k+1})}{\delta \rho_k} = 0$

on obtient: $\rho_k = \frac{\|\nabla J\|^2}{\nabla J^t \times D \times \nabla J}$

Ce qui est une possibilité de choix optimale pour la valeur de ρ_k , mais on voit bien qu'à chaque itération le calcul de ρ_k va prendre beaucoup de temps calcul.

4.4.3.3 Méthode du perceptron

4.4.3.3.1 Méthode classique

Rappelons que le critère du perceptron (la fonction J(a)) est défini par:

$$J_p(a) = \sum_{y \in Y(a)} \left(-a^t y\right)$$

avec Y(a) étant l'ensemble des échantillons mal classés si on choisit a

Rappelons que y est mal classé ssi $a^t y<0$ et donc $J_p(a)>=0$
De plus $J_p(a)$ est « proportionnel » à la somme des distances des échantillons mal classés à la frontière de décision définie par le vecteur a.

Or $\nabla Jp = \begin{bmatrix} \frac{\delta J_p}{\delta a_1} \\ \frac{\delta J_p}{\delta a_2} \\ \cdot \\ \cdot \\ \frac{\delta J_p}{\delta a_{d+1}} \end{bmatrix}$ *alors* $\nabla Jp = \sum_{y \in Y(a} (-y)$

D'où la formule de l'algorithme de descente du gradient:

$$a_{k+1} = a_k + \rho_k \times \sum_{y \in Y(a)} y$$

avec Y(a) étant l'ensemble des échantillons mal classés pour a

Cette méthode converge à coup sûr vers une solution (pas forcément optimale) si les classes sont linéairement séparables.

4.4.3.3.2 Méthodes dérivées

Il est possible de définir des méthodes dérivées de la méthode précédente où la quantité de calculs sera diminuée. L'idée de base est de faire intervenir les échantillons mal classés par tirage aléatoire au lieu de faire intervenir l'ensemble des échantillons mal classés (pour un choix donné de a).

La méthode est la suivante:

on part de a1 arbitraire

$$a_{k+1} = a_k + \rho_k \times \tilde{y}_k \quad k \geq 1$$

$\tilde{y}_k$ est un vecteur y_k qui est mal classé par le choix a_k. Ce vecteur y_k est choisi de façon aléatoire dans $Y(a_k)$.

On peut de plus définir une marge . Ainsi un vecteur est mal classé si $a_k^t \times \tilde{y}_k \leq b$.

L'hyperplan séparateur obtenu est alors optimisé.

On démontre que

si

$$\rho_k \geq 0 \quad et \quad \lim_{n\to+\infty} \sum_{k=1}^{n} \rho_k = +\infty$$

$$et \quad \lim_{n\to+\infty} \frac{\sum_{k=1}^{n} \rho_k^2}{\left(\sum_{k=1}^{n} \rho_k\right)^2} = 0$$

alors

a_k converge vers une solution vérifiant $a^t \times y_i \succ b \quad \forall i$ si les deux classes sont linéairement séparables.

4.4.3.4 méthode par relaxation

4.4.3.4.1 Méthode classique

Dans cette méthode le critère du perceptron

$$J_p(a) = \sum_{y\in Y(a)} \left(-a^t y\right)$$

est remplacé par

$$J_r(a) = \sum_{y\in Y(a)} \left(a^t y\right)^2$$

Ici le gradient sera du « premier degré ».

La solution trouvée ne sera pas forcément optimale , on choisit donc ici aussi un critère avec « marge » qui prendra donc la forme suivante:

$$\boxed{J_r(a) = \frac{1}{2} \times \sum_{y\in Y(a)} \frac{\left(a^t y - b\right)^2}{\|y\|^2}}$$

avec Y(a) étant l'ensemble des échantillons mal classés si l'on choisit a et donc tels que $a^t y <= b$

alors

$$\nabla J_r = \sum_{y \in Y(a)} \frac{\left(a^t y - b\right)}{\|y\|^2} \times y$$

L'algorithme de descente du gradient prend alors la forme suivante:

$$\boxed{\begin{array}{l} a_1 \quad arbitraire \\ a_{k+1} = a_k + \rho_k \times \sum_{y \in Y(a)} \left[\frac{\left(b - a^t y\right)}{\|y\|^2} \times y \right] \end{array}}$$

4.4.3.4.2 Méthode dérivée

Comme pour la règle du perceptron il est possible de simplifier cet algorithme de relaxation en faisant intervenir les échantillons mal classés de Y(a) de façon aléatoire et individuellement.
en prenant de plus un ρ_k constant égal à ρ on obtient l'algorithme simplifié suivant (dit règle de relaxation) :

$$a_1 \ arbitraire$$

$$a_{k+1} = a_k + \rho \times \frac{b - a_k^t \times \tilde{y}_k}{\|\tilde{y}_k\|} \times \tilde{y}_k \quad avec \ \ a_k^t \times \tilde{y}_k \leq b$$

Expliquons l'origine du terme " relaxation " :

Notons $r_k = \frac{b - a_k^t \times \tilde{y}_k}{\|\tilde{y}_k\|}$ la distance de a_k à l'hyperplan $a^t y_k = b$

Comme $\frac{y_k}{\|y_k\|}$ est un vecteur unité normal à l'hyperplan $a^t y_k = b$

Passer de a_k à a_{k+1} consiste donc à "déplacer" le vecteur a_k d'une certaine fraction ρ de la distance à l'hyperplan ! Dans le cas extrême où ρ vaut 1 alors a_k est " mis " sur l'hyperplan.

On dit qu'il y a relaxation de la tension créée par l'inégalité $a^t y_k b$

4.5 EXERCICES

Soient les deux classes suivantes définies dans l'espace des caractéristiques $\mathbf{R^2}$

- classe 1 définie par **(0,1), (1,1), (2,1), (2,0)**
- classe 2 définie par **(-1,-1), (-1,-2), (0,-1), (0,-2)**

Appliquer une des méthodes d'apprentissage par « descente du gradient » puis représenter les échantillons des deux classes et la courbe de séparation obtenue entre les deux classes dans l'espace des caractéristiques

Remarque: *bien détailler et expliquer tout le cheminement suivi*

Soient les deux classes suivantes définies dans l'espace des caractéristiques $\mathbf{R^2}$

- classe 1 définie par **(1,0),(2,0),(2,1)**
- classe 2 définie par **(0,-2),(-1,-1), (-1,0),(0,1)**

Appliquer une des méthodes d'apprentissage par « descente du gradient » puis représenter les échantillons des deux classes et la courbe de séparation obtenue entre les deux classes dans l'espace des caractéristiques

Remarque: *bien détailler et expliquer tout le cheminement suivi*

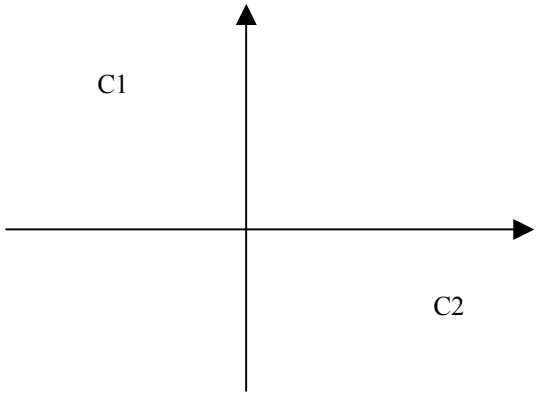
On suppose disposer d 'échantillons définissants deux classes C1 et C2.
On voudrait utiliser la séparation linéaire pour apprendre à reconnaître ces deux classes mais les deux classes ne sont pas linéairement séparables. Or on suppose savoir que les échantillons de la classe C2 forment en fait trois paquets bien distincts.

Une configuration des échantillons (avec formes supposées définies dans R^2 pourrait être la suivante :

Proposer une suite de d'opérations
de classification permettant d'utiliser la séparation linéaire

(attention on doit faire tout le travail automatiquement)

On donnera le maximum de détails.



Soient les deux classes suivantes définies dans l'espace des caractéristiques $\mathbf{R^2}$

- classe 1 définie par **(0,1), (1,1), (2,1), (2,0)**
- classe 2 définie par **(-1,-1), (-1,-2), (0,-1), (0,-2)**

Appliquer une des méthodes d'apprentissage par « descente du gradient » puis représenter les échantillons des deux classes et la courbe de séparation obtenue entre les deux classes dans l'espace des caractéristiques

Remarque: *bien détailler et expliquer tout le cheminement suivi*

Décrire la méthode d'apprentissage géométrique dans le cadre de la séparation linéaire (on supposera qu'il n'y a que deux classes définies par des échantillons) et appliquer au cas particulier suivant:

- classe 1 définie par (1,0) (0,1)
- classe 2 définie par (-1,0) (-1,-1)

Dans le cadre de la séparation linéaire, décrire la méthode d'apprentissage par « descente du gradient » (on supposera qu'il n'y a que deux classes définies par des échantillons) et appliquer au cas particulier suivant :

- classe 1 définie par : (1,0), (0,1) ;

classe 2 définie par : (-1,0), (-1,-1).

Décrire deux méthodes d'apprentissage (une géometrique et l'autre par « descente du gradient ») dans le cadre de la séparation linéaire (on supposera qu'il n'y a que deux classes définies par des échantillons) et appliquer au cas particulier suivant:

- classe 1 définie par (1,0) (0,1)

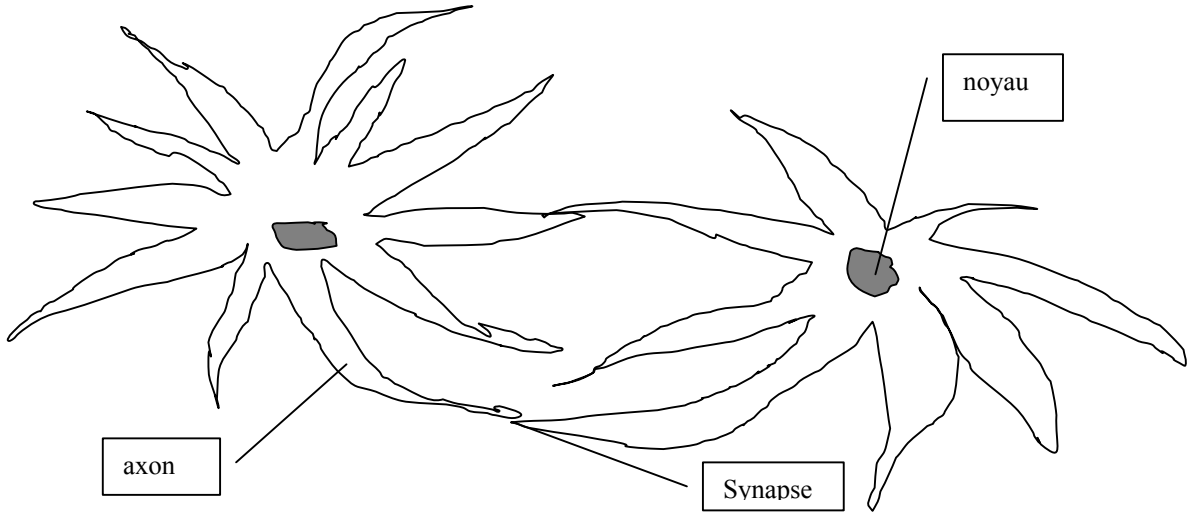
classe 2 définie par (-1,0) (-1,-1)

4.6 RECONNAISSANCE DES FORMES ET NEURONES

4.7 GENERALITES

Nous aborderons les réseaux de neurones dans la seule perspective de faire de la reconnaissance des formes et de la classification/segmentation. Notre objectif n'est donc pas celui des « neurosciences » consistant à chercher à utiliser la puissance des ordinateurs pour simuler l'intelligence humaine ou animale.

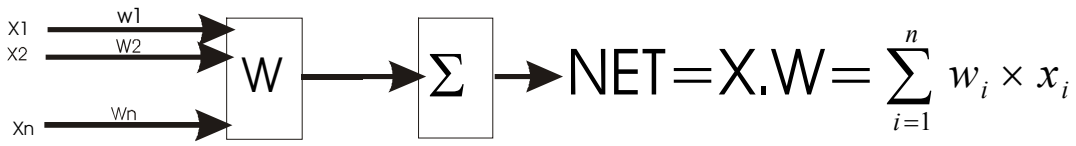
Rappelons quelques généralités sur ces cellules particulières que sont les neurones biologiques. Il s'agit donc de cellules comportant un noyau comme toutes les cellules vivantes mais comportant des branches appelées axons. Ces axons peuvent être reliés pour « connecter » des neurones.



Ces connexions se comptant par milliards et produisant ce qu'on appelle un réseau de neurones. Chacun de ces neurones peut être activé ou non par ce qu'on appelle la « force synaptique » et transmettre son « activation » par l'intermédiaire des synapses aux autres neurones. Chaque neurone pouvant être considéré comme comportant des entrées et des sorties.

Ce réseau de neurones peut recevoir des activations par l'intermédiaire des sens notamment et l'état d'excitation des neurones définit alors un état de la perception .

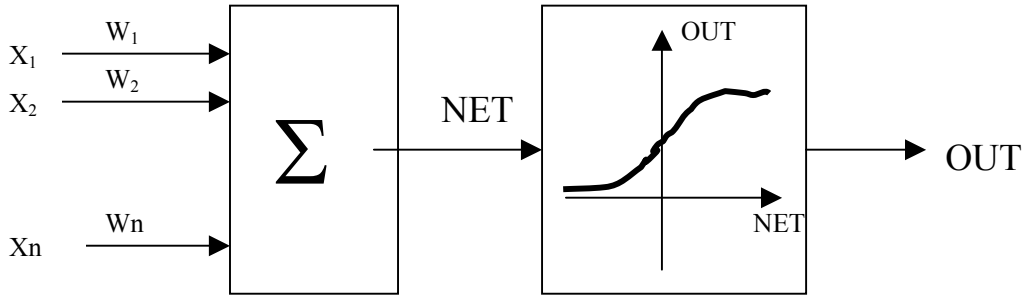
Nous allons maintenant étudier un type de neurone artificiel non utilisé dans les neurosciences mais utilisé en modélisation ou en reconnaissance des formes et voir quelles sont les analogies entre neurone biologique et neurone artificiel.



La somme pondérée des entrées, NET peut être considérée comme la « force synaptique » permettant d'activer le neurone biologique et donc de façon analogue au fonctionnement d'un neurone biologique il reste à définir une fonction d'activation qui définira l'activation du neurone en fonction des entrées.

Les caractéristiques du neurone artificiel étant définies par le nombre d'entrées possibles (dimension du vecteur X) les poids de connexion (vecteur W) et la fonction d'activation.

Nous pouvons représenter cela par le schéma suivant :



Avec par exemple comme fonction de sortie une fonction sigmoïde du type suivant :

$$OUT = \frac{1}{1+e^{-NET}}$$

Par exemple avec ce neurone particulier (pour l'instant nous n'avons pas encore de réseau de neurones composé de plusieurs neurones interconnnectés !) nous pouvons nous poser la question de ce que nous pouvons faire en terme de reconnaissance des formes.

A priori la « forme » c'est le vecteur d'entrée X ….C'est cette forme qui est présentée au neurone et ce neurone soumis à cette sollicitation répond OUT qui est une valeur dépendant bien sur de l'entrée mais aussi du vecteur poids W (c'est à dire des n poids scalaires x_1 x_2 …. x_n).

En faisant l'hypothése que ce neurone peut classer des formes dans deux classes C_1 et C_2 (et qu'une forme est classée par le neurone dans C_2 si et seulement si OUT > 0.5) on peut donc penser que si on choisit correctement le vecteur poids W on saura peut être classer des formes se répartissant dans deux classes.

On est donc confronté à un probléme classique en reconnaissance des formes. Si mes deux classes C_1 et C_2 sont représentées par quelques échantillons je peux envisager l'utilisation de mon neurone en deux phases :

Phase 1 : A l'aide des échantillons j'essaye de choisir des poids qui font que le neurone répond correctement dans tous les cas…il s'agit donc de la phase où on apprend quelque chose au neurone (reconnaître correctement les classes sur la base des échantillons)

Phase 2 : une fois que l'apprentissage est réalisé , je présente une forme inconnue et le neurone me reponds dans quelle classe il le range. Il s'agit en fait de la généralisation de ce que le neurone a appris lors de la phase 1

On fonctionne donc dans ce cas en classification supervisée (avec professeur).

En fait si je suppose que l'apprentissage a été fait avec les échantillons …
Si une forme en entrée est définie par le vecteur ${}^tX$= (x1 x2 x3 ….. xn) alors le neurone le classe dans C2 si et seulement si :

$$w_1.x_1 + w_2.x_2 + \ldots\ldots + w_n.x_n - 0.5 > 0.0$$

c'est à dire si le vecteur X est "au dessus" de l'hyperplan d'équation

$$w_1.x_1 + w_2.x_2 + \ldots\ldots + w_n.x_n - 0.5 = 0.0$$

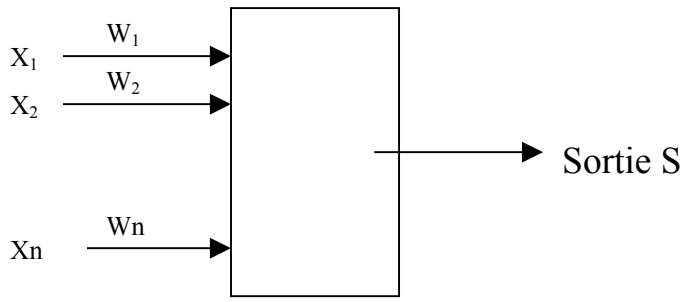
On en déduit donc qu'un neurone du type décrit ici est capable de faire une sorte de « séparation linéaire » de deux classes

4.8 HISTORIQUE

Les premiéres approches neuronales ont souvent été liées à la recherche de méthodes automatiques de perception ou de modélisation du systéme neuronal humain qui à cette époque était encore très mal connu.et nous nous bornerons pour l'essentiel à ce type de problématique.

4.8.1 1943 : premier modèle de neurone

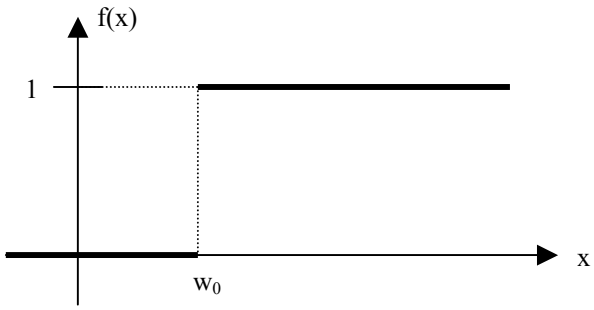
Ainsi le premier modèle de neurone a été défini en 1943 par Mac Culloch (un psychiatre) et Pitts (un mathématicen) comme étant un dispositif recevant un potentiel d'action et qui s'active ou non suivant l'intensité de ce potentiel d'action. Il s'agit donc d'un dispositif qui s'active (reponse **oui** ou **1**) ou non (réponse **non** ou **0**) suivant que le potentiel d'activation est supérieur à un seuil ou non.



Le potentiel d'activation est en fait ici la somme pondérée des entrées. Si cette somme pondérée dépasse un certain seuil w_0 alors le neurone fournit la valeur 1 sinon il fournit 0.

Ce neurone formel est donc défini par le seuil w_0 et les poids des connexions d'entrées les w_i et par une fonction de seuillage f( ).

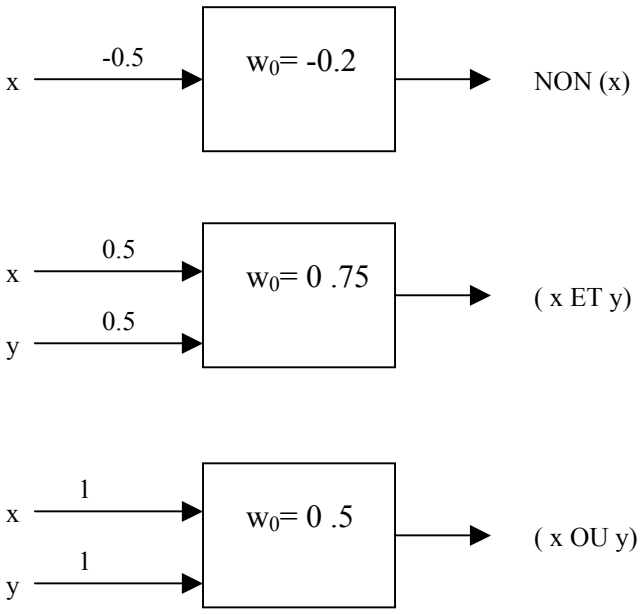
La fonction f( ) est définie de la façon suivante :



Et donc on a

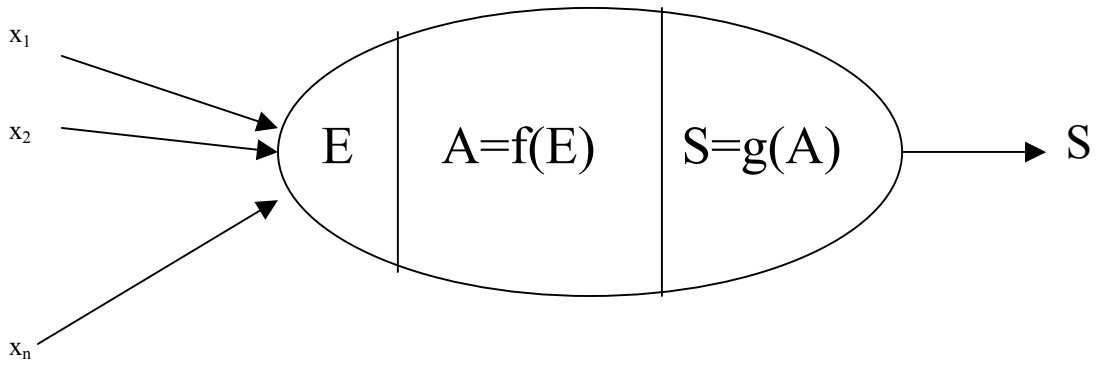
$$S = f\left(\sum_{i=1}^{n} w_i \times x_i\right)$$

En fait à l'aide de ce neurone Mac Culloch et Pitts ont conçu des réseaux de neurones permettant de calculer toutes les fonctions de l'algébre de Boole à partir de neurones calculant le ET, le OU et le NON.



Mais déjà avec cette approche on voit apparaître la possibilité de traiter des problémes de reconnaissance des formes. En effet, de par l'expression de la fonction f( ) on peut déduire que si on est confronté à un problème de reconnaissance de formes de deux classes linéairement séparables et qu'on dispose d'échantillons , il est alors certain qu'il existe une solution (en termes de w_i, i=0….n) telle que le neurone décrit sera une solution pour obtenir un classifieur.

A partir de ce modèle ont été définis divers modéles de neurones plus généraux qui satisfont aux critéres génériques suivants :



$\mathbf{x_i}$	entrées
$\mathbf{E= h(x_1,x_2,\ldots,x_n)}$	fonction entrée totale
A=f(E)	etat du neurone et fonction d'activation
S=g(A)	sortie

Dans ce cas de figure, les possibilités les plus courantes sont les suivantes :

Nature des entrées et des sorties :

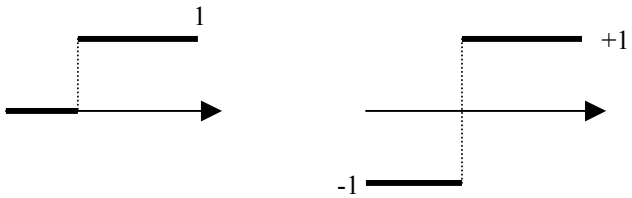
- binaires (0 , 1) ou (-1 , +1)
- réelles

Fonction d'entrée totale h ( ) :

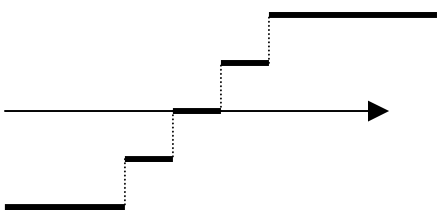
- booléenne
- linéaire $\sum w_i \times x_i$
- affine $\left(\sum w_i \times x_i\right) - a$

Fonction d'activation f( ) :

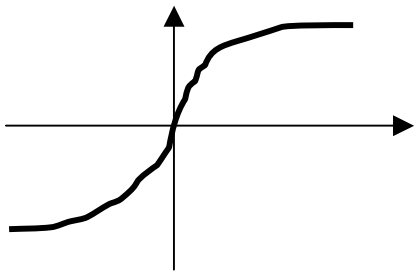
- binaire



- linéaire à seuils



- fonction sigmoïde



par exemple $f(x) = a \times \frac{\left(e^{kx} - 1\right)}{\left(e^{kx} + 1\right)}$

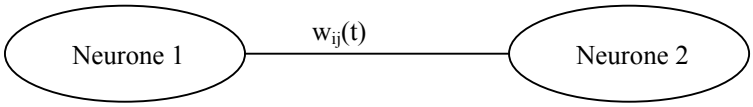
Fonction de sortie g( ) :

- en général l'identité

4.8.2 1949, les réseaux de Hebb

Toujours dans le domaine de la recherche sur le fonctionnement des neurones biologiques et sur les mécanismes d'apprentissage de l'intelligence humaine, Hebb a proposé un type de réseau de neurones totalement interconnecté (c'est à dire ou les neurones sont reliés par des connexions de type synapse fonctionnant à la fois en « entrée » et en « sortie ». Ces connexions sont affectées de poids qui évolue au cours du temps et en fonction de l'activation de chacun des deux neurones extrémités de cette connexion.

Il a ainsi défini une régle d'apprentissage dite de Hebb :



Cette régle considére alors que toute connexion entre deux neurones se renforce si ces deux neurones sont actifs au même moment.

Si on note A1 et A2 l'activation des neurones 1 et 2 et si on suppose qu'un neurone actif à son activation qui vaut 1 et qu'un neurone inactif a son activation qui vaut 0 alors l'expression de la régle de Hebb est la suivante :

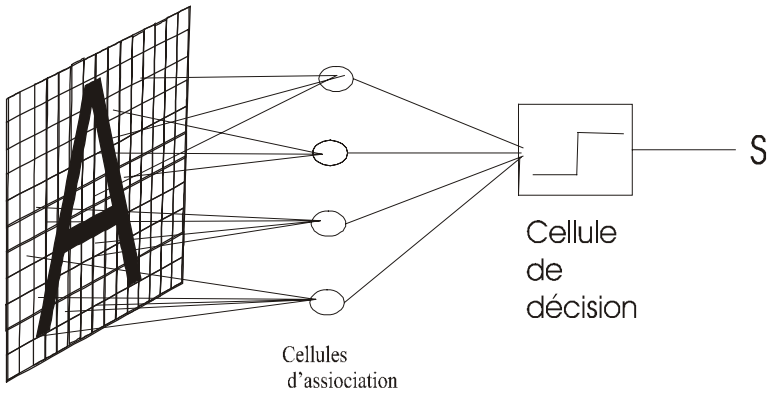
$$w_{ij}\left(t + \delta t\right) = w_{ij}\left(t\right) + \mu \times A_i \times A_j \qquad avec \qquad \mu \succ 0$$

au départ on a $w_{ij}(0) = 0$ pour tout i, j

Ce type de réseaux a surtout une vocation pour la modélisation des neurones biologiques et reste peu utilisé en reconnaissance des formes. Un des inconvénients majeurs de ce modéle vient du fait que les w_{ij} ne peuvent qu'augmentert au cours du temps

4.8.3 1958, Le PERCEPTRON de ROSENBLATT

Rosenblatt a conçu un réseau de neurones très simplifié dans le but de faire de la reconnaissance automatique de caractéres imprimés. Il aborde ce probléme sous la forme d'une perception visuelle simplifiée. Son approche consiste à concevoir autant de dispositifs spécialisés qu'il y a de lettres à reconnaître. Le schéma ci dessous décrit le dispositif possible pour la reconnaissance de la lettre « A »



La zone concernée par la perception est en fait considérée comme regardée par une rétine dont les cellules de reception forment une matrice.

Rosenblatt a alors conçu des dispositifs indépendants de reconnaissance de chaque lettre. Si l'on prend l'exemple de la lettre A représentée sur le schéma ci-dessus, Rosenblatt a choisi quelles cellules de reception de la rétine devaient être considérées, il les a regroupées sur des cellules d'association qui fournissent un résultat 0 ou 1 suivant que chaque cellule de reception vise une zone encrée ou non. Chacune de ces cellules d'association fournit donc alors une entrée (qui est pondérée par un poids w_i) à un neurone linéaire à seuil, qui fournit alors une sortie 0 ou 1 suivant qu'on n'a pas ou qu'on a reconnu la lettre « A ».

Chaque lettre qui « entre » est donc finalement codée sous forme d'un vecteur

$$^{t}e = \left(e_1 \quad e_2 \quad ..... \quad e_n\right)$$

Rosenblatt a lors élaboré une méthode d'apprentissage de chaque neurone. Si l'on reprend le dispositif pour la reconnaissance de la lettre 'A', alors cette méthode va consister à ajuster les poids de façon à ce que le dispositif réponde correctement à la présentation de lettre A (réponse désirée 1) et d'autres lettres (réponse désirée 0).

On dispose donc au départ de plusieurs exemples de lettre 'A' et de plusieurs exemples d'autres lettres. On présente alors plusieurs fois si nécessaire chacun de ces échantillons et contre-échantillons, en définissant un critère d'arrêt (par exemple « présenter chaque lettre 10 fois » ou bien « jusqu'à ce que la réponse obtenue soit juste pour chaque lettre »….Mais ici on ne se pose pas la question de la convergence).

A chaque lettre échantillon ou contre-échantillon présentée on applique alors la règle suivante d'évolution des pois w_i , dite « règle d'apprentissage du perceptron »

Rappelons que si les entrées sur ce neurone sont notée e_i , i=1,n et les poids des connexions w_i alors la sortie **s** du neurone est donnée par :

Si $\sum_{i=1}^{n} w_i \times e_i \succ w_0$ **alors s=1 sinon s=0**

Si l'on note **d** la réponse désirée à la présentation d'un échantillon ou d'un contre-échantillon alors on applique les modifications de poids suivantes

- **Si d==s alors**

  ne rien faire puisque le systéme répond alors correctement

- **Si d < s (soit d==0 et s==1) alors**

  il faut diminuer les poids des connexions recevant une entrée positive et augmenter les poids des connexions recevant une entrée négative car en effet il faut augmenter la valeur de $\sum_{i=1}^{n} w_i \times e_i$ de façon à se rapprocher de la réalisation de la condition $\sum_{i=1}^{n} w_i \times e_i \succ w_0$

- **Si d > s (soit d==1 et s==0) alors**

  il faut augmenter les poids des connexions recevant une entrée positive et diminuer les poids des connexions recevant une entrée négative car en effet il faut diminuer la valeur de $\sum_{i=1}^{n} w_i \times e_i$ de façon à se rapprocher de la réalisation de la condition $\sum_{i=1}^{n} w_i \times e_i \leq w_0$

ce qui peut être résumé par :

$$\mathbf{w_i = w_i + k \times (d-s) \times e_i \quad avec\ k > 0}$$

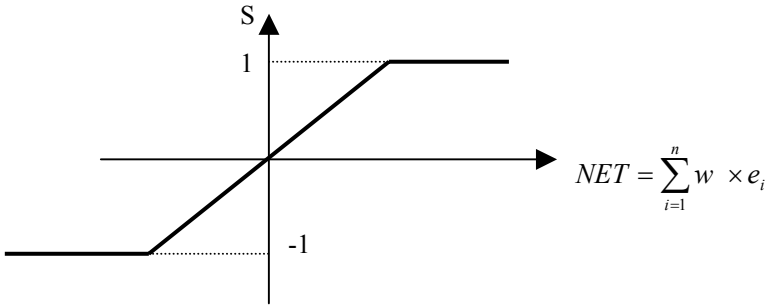
k : taux d'apprentissage

C'est la régle d'apprentissage du Perceptron

Notons ici comme nous l'avons vu précédemment, qu'une condition nécessaire pour qu'il y'ait convergence du processus d'apprentissage est que la classe des "A" soit séparable linéairement de la classe de "toutes les autres lettres"

4.8.4 1960, WIDROW et HOFF et régle δ

Widrow et Hoff reprenant l'idée du perceptron (mono-neurone) de Rosenblatt ont conçu ADALINE (ADAptative Linear Neurone). A la différence du neurone linéaire à seuil de Rosenblatt ils ont proposé une fonction d'activation linéaire du type suivant :



Les entrées peuvent donc être réelles comme pour le perceptron mais surtout la sortie est une sortie réelle au lieu d'être une sortie binaire.

Cela a amené les auteurs de ce nouveau neurone à définir une régle d'apprentisssage inspirée de celle de Rosenblatt qu'ils ont appelé régle d'apprentissage δ.

La formulation de cette rêgle δ dit aussi régle de Widrow Hoff est la suivante :

Soit S la sortie réelle du neurone

Soit T la sortie désirée sur présentation d'un échantillon,

Et soit **δ = (T – S)**

- **Si δ =0 alors**

  ne rien faire puisque le systéme répond alors correctement

- **Si δ >0 alors**

  il faut augmenter la valeur de $\sum_{i=1}^{n} w_i \times e_i$

- **Si δ <0 alors**

  il faut diminuer la valeur de $\sum_{i=1}^{n} w_i \times e_i$

  alors si l'on note

$$\Delta_i = \eta \times \delta \times e_i$$

avec $\eta > 0$ appelé taux d'apprentissage

alors on a

pour tout poids w_i $\boxed{w_i = w_i + \Delta_i}$

Nous verrons que cette régle δ sera utilisée ultérieurement lors de l'apprentissage de réseaux de neurones multicouches de type perceptron

4.8.5 1969 MINSKY et PAPERT

Jusqu'à cette époque (fin des années 60) deux approches sont en concurrence en ce qui concerne la recherche sur la perception et la cognition : les approches intelligence artificielle (logique et approches « problem-solving ») et les approches connectionnistes avec comme modèle dominant le perceptron.

A la fin des années 70 Marvin Minsky et Seymour Papert après avoir bien étudié le perceptron et notamment une evolution de réseau de neurones multicouche (le perceptron multi couche) indiquent qu'ils ne voient pas d'évolution possible pour résoudre les problémes non séparables linéairement . A partir de cette date la recherche sur les réseaux de neurones va subir une chute très importante.

Jusqu'à la découverte par Rumelhart et d'autres chercheurs de la technique dite de rétropropagation du gradient pour l'apprentissage du perceptron multicouche en 1986.

Mais jusqu'en 1986 seuls quelques chercheurs isolés ont continué dans la même voie.

Les années 1970 ont vu l'emergence des cartes auto-organisatrices et les mémoires associatives essentiellement sous l'impulsion de Kohonen , Grossberg et Hopfield.

Nous ne détaillerons pas plus l'histoire de la recherche en ce qui concerne les réseaux de neurones.

Mais nous allons présenter ci-dessous pourquoi il est interessant de passer aux réseaux de neurones plutot que de ne travailler qu'avec un neurone comme nous l'avons fait jusqu'à maintenant.

4.9 DU NEURONE AU RESEAU DE NEURONES

En fait Minsky et Papert ont mis en évidence les limites du neurone linéaire à seuil et de ses limitations. Ils ont notamment ouvert la voie aux réseaux de neurones et notamment au réseau de neurones dit « perceptron multicouche » mais dans la mesure ou aucune méthode rationnelle d'apprentissage de ce type de réseau n'était évidente la recherche en connexionnisme s'est restreinte à quelques courageux.

Nous allons voir maintenant comment on passe naturellement d'un neurone linéaire à seuil à un réseau de neurones du même type pour résoudre des problémes plus élaborés.

4.9.1 Les limites du neurone linéaire à seuil

Comme nous l'avons vu un neurone linéaire à seuil ne peut résoudre que des problemes de reconnaissance des formes à deux classes linéairement séparables.

En effet si on a un neurone à n entrées avec des poids $w_1, w_2, \ldots, w_n$ et un seuil w_0 et si on a une entrée ${}^t e = (e_1 \quad e_2 \quad \ldots\ldots\ldots \quad e_n)$ alors

Si $\sum_{i=1}^{n} w_i \times e_i \succ w_0$ **alors s=1 sinon s=0**

Dans l'espace R^n on peut alors définir un hyperplan d'équation

$$w_1 \times e_1 + w_2 \times e_2 + \ldots\ldots\ldots + w_n \times e_n - w_0 = 0$$

le demi espace defini par

$w_1 \times e_1 + w_2 \times e_2 + \ldots\ldots\ldots + w_n \times e_n - w_0 > 0$

contient tous les vecteurs e fournissant une sortie égale à 1

Le demi espace défini par

$w_1 \times e_1 + w_2 \times e_2 + \ldots\ldots\ldots + w_n \times e_n - w_0 < 0$

auquel on ajoute l'hyperplan contient tous les vecteurs fournissant une sortie à 0.

Si on part du point de vue que les sorties 0 et 1 signifient un numéro de classe il est alors immédiat que le neurone linéaire à seuil peut répresenter n'importe quel classifieur fonctionnant en séparateur linéaire pour deux classes linéairement séparables (c'est à dire qu'il existe des poids w1,w2,…,wn et un seuil w0 solutionnant le probléme…ce qui ne signifie pas qu'on sait forcément les trouver simplement).

Voyons maintenant un probléme simple de classification non linéairement séparable :

Soit deux classes définies par les échantillons suivants dans R2

Classe C_1 : (0,1) et (1,0)
Classe C_2 : (0,0) et (1,1)

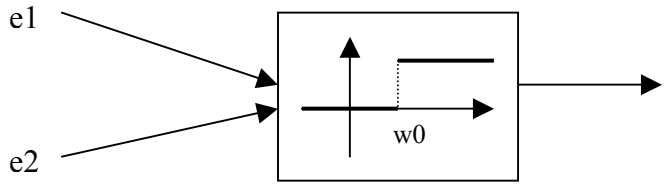
On aimerait donc avoir un neurone linéaire à seuil fournissant une sortie 1 pour C_1
Et fournissant 0 pour C_2

soit

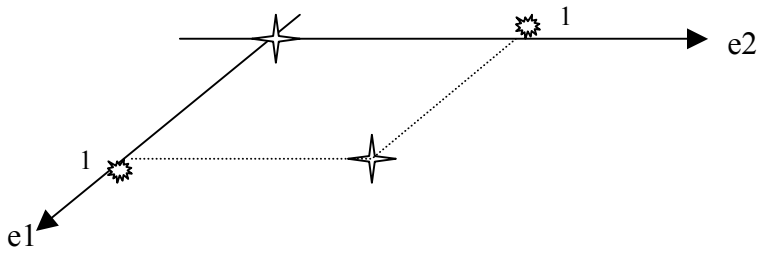
e1	**e2**	**Sortie désirée**	
0	0	0	C_2
0	1	1	C_1
1	0	1	C_1
1	1	0	C_2

En fait si l'on considére les entrées comme binaires il s'agit du ou exclusif en algébre de Boole c'est à ditre XOR (e_1,e_2).

On cherche donc un neurone linéaire à seuil satisfaisant ce probléme :



Si l'on représente les échantillons dans le plan on voit immédiatement que le probléme n'a pas de solution en w_1 w_2 et w_0

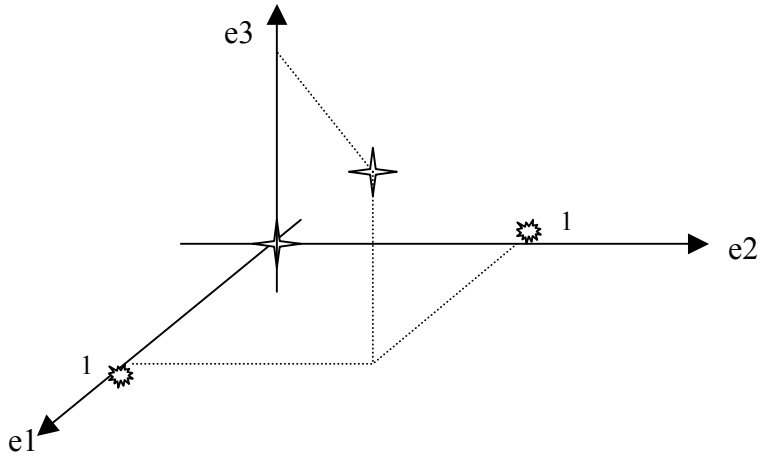


Par contre pour rendre le probléme linéairement séparable il suffit ici de rajouter une composante fictive e_3 à chaque vecteur

Par exemple,

Classe C_1 : (0,1,**0**) et (1,0,**0**)
Classe C_2 : (0,0,**0**) et (1,1,**1**)

On obtient alors un probléme linéairement séparable puisqu'on peut faire passer un plan entre les deux classes



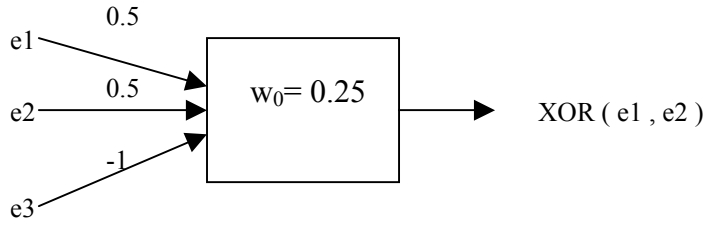
soit

e1	e2	e3	Sortie désirée
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

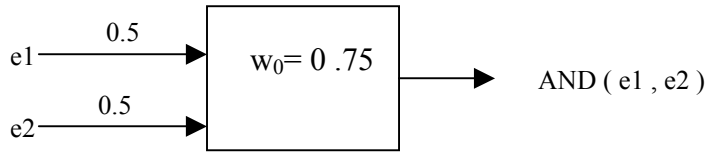
On remarque que e_3 est en fait le **ET logique** de e_1 et e_2

On en déduit que si on dispose d'un neurone linéaire à seuil à trois entrées et que ces trois entrées sont respectivement e1 , e2 et e3 alors on pourra trouver les poids permettant de résoudre le probléme.

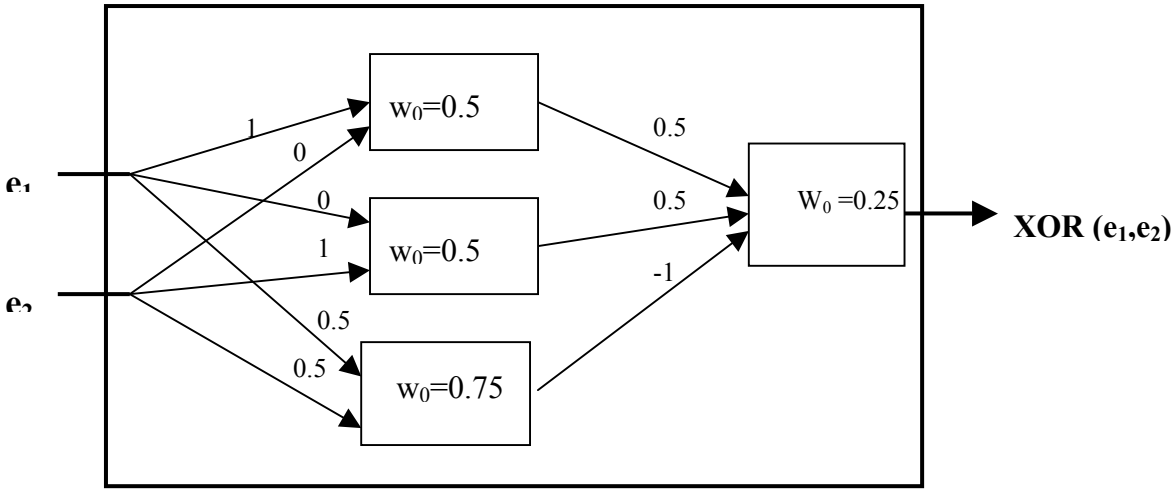
On peut choisir par exemple :



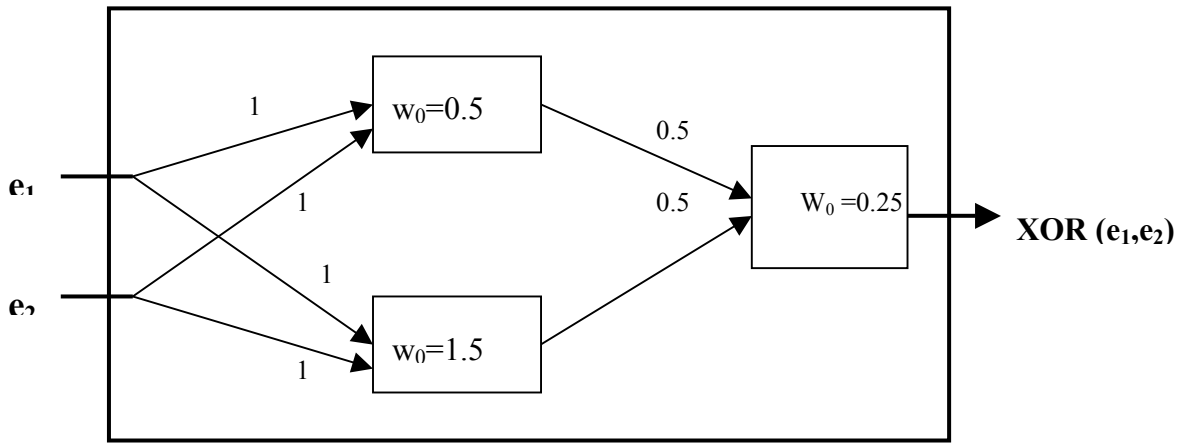
Il faut donc maintenant construire le dispositif permettant d'obtenir e_3 et en fait un neurone linéaire à seuil à deux entrées peut faire l'affaire comme nous l'avons déjà vu



En prenant deux neurones linéaires à seuil à deux entrées et une sortie on peut alors avoir le réseau de neurones à deux couches successives suivant :



Une autre possibilité plus simple et n'utilisant que des neurones de conception identique (deux entrées et une sortie) serait par exemple la suivante :



On remarque ainsi qu'avec des reseaux multicouches on peut résoudre des problémes non linéairement séparables. Mais le gros probléme est celui de l'apprentissage de tels réseaux multicouches et du contrôle de ce qu'ils peuvent produire. La recherche dans ce domaine a été quasiment stoppée pendant plus de 10 ans du fait de ce probléme de l'impossibilité de réaliser l'apprentissage d'un tel réseau.

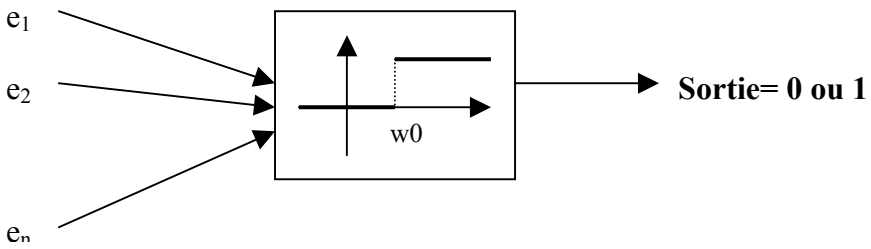
La régle δ de Widrow et Hoff puis la conception de neurones à fonction d'activation sigmoide ont permis la découverte d'une méthode mathématique permettant de rétropropager l'erreur de sortie dans le réseau et donc la conception d'une méthode d'apprentissage d'un tel réseau composé de ces nouveaux neurones (Rumelhart,Hinton et Williams en 1986). Cela a redonné un serieux regain d'activité à la recherche dans ces domaines.

4.9.2 Le perceptron multicouches

4.9.2.1 De la régle du perceptron à la régle δ

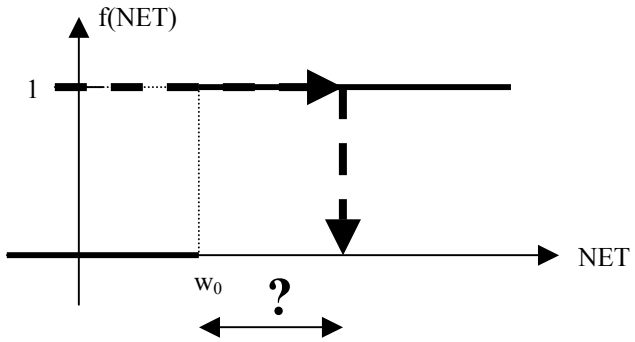
Nous allons comparer les deux régles d'apprentissage pour les deux neurones que nous avons déjà décrit, le neurone linéaire à seuil avec la régle du perceptron et le neurone à sortie réelle et fonction d'activation sigmoïde avec la régle δ.

Reprenons la régle d'apprentissage du perceptron avec un neurone linéaire à seuil.

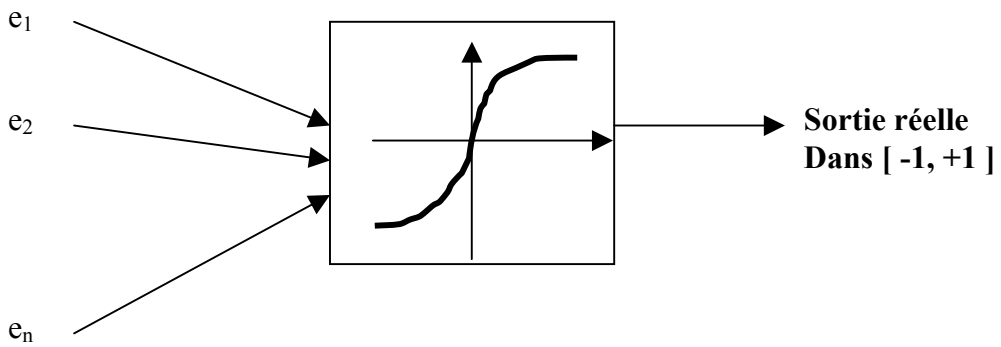


On voit ici que si l'on a un échantillon avec sa réponse désirée d égale à 0 par exemple et que le neurone soumis à cet échantillon réponde 1.

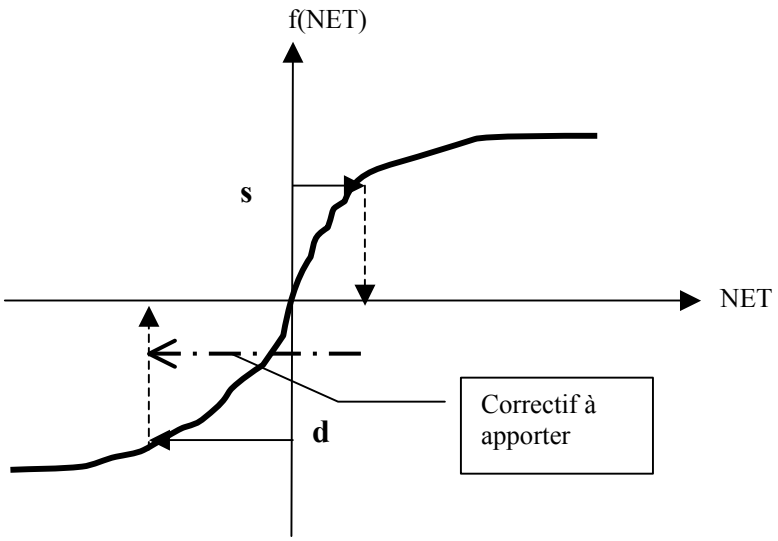
Cela signifie que $\sum_{i=1}^{n} w_i \times e_i \succ w_0$ mais on ne sait pas à quelle valeur de NET = $\sum_{i=1}^{n} w_i \times e_i$ correspond cette sortie à 1. Cela peut être toute valeur de NET entre w_0 et + ∞, d'où la difficulté à décider quelle type de correction on doit apporter à NET (doit elle être forte ou faible ?).



Considérons maintenant un neurone à sortie réelle et à fonction d'activation sigmoïde du type suivant :



Ici si l'on dispose d'un échantillon c'est à dire d'une entrée pour laquelle on connaît la sortie désirée réelle d et qu'on obtienne en fait s alors on saura quelle somme pondérée des entrées NET a fourni cette sortie s et quel NET il faudrait pour obtenir d. Comme la fonction sigmoide est croissante monotone , on voit que l'on a une idée bien plus précise sur la correction nécessaire à apporter à NET.

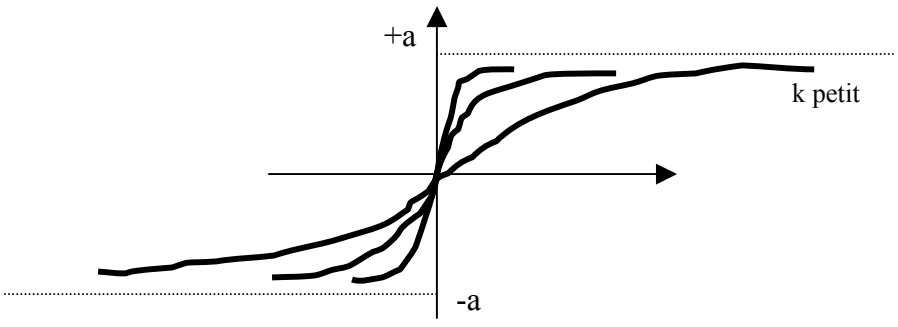


Bien sûr cela ne résoud pas pour l'instant le problème de l'apprentissage mais l'on sent bien que l'on a progressé. Effectivement comme on le verra par la suite c'est avec ce type de neurone que l'on a pu construire des réseaux de neurones et surtout définir une méthode d'apprentissage pour ce type de réseau.
Nous allons donc dans la suite détailler un peu plus ces neurones à fonctions d'activation sigmoïde.

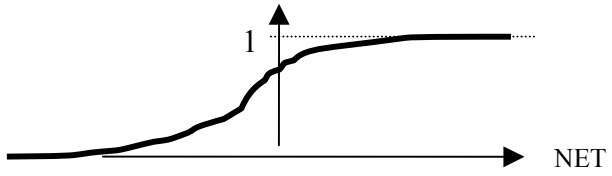
4.9.2.2 Quelques exemples de neurones à fonction d'activation sigmoïde

Nous allons présenter quelques exemples de fonctions d'activation sigmoïde utilisées couramment et leur intérêt respectif.

- $f(NET) = a \times \dfrac{\left(e^{k \times NET} - 1\right)}{\left(e^{k \times NET} + 1\right)}$ avec k et a positifs



- $f(NET) = \frac{1}{1 + e^{-NET}}$



L'intérêt majeur de cette fonction sigmoïde particuliére vient du fait que l'on a la relation suivante :

$$\frac{\partial f}{\partial NET} = f(NET) \times (1 - f(NET))$$

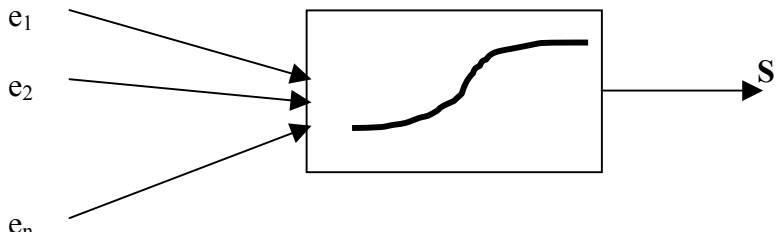
or comme f(NET) est la sortie du neurone, si je note cette sortie S

Alors $f'(NET) = S \times (1 - S)$

Nous verrons que cette remarque facilitera le calcul, lors de l'apprentissage d'un réseau multicouche composé de neurones de ce type puisque dans ce calcul nous aurons à calculer la dérivée de la fonction d'activation pour un NET donné et que cela s'obtiendra à partir de la sortie du neurone concerné.

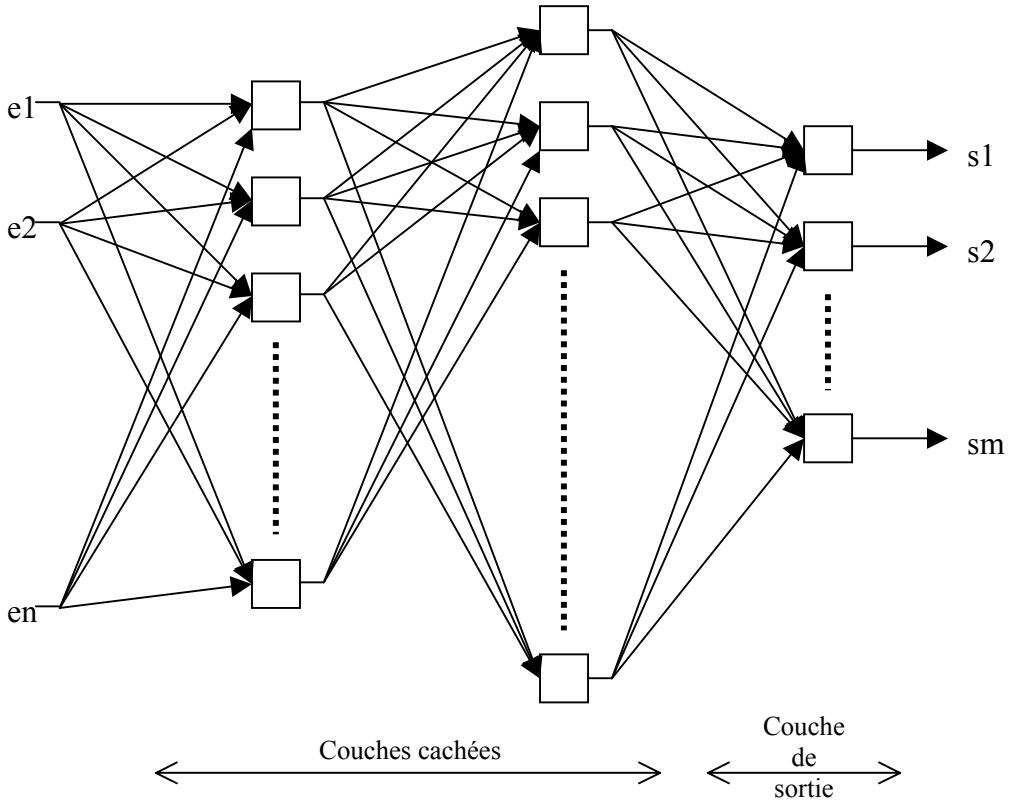
4.9.2.3 Le perceptron multicouche et son fonctionnement

Nous allons considérer un réseau de neurones organisé en couches successives construit avec des neurones du type suivant :



Notons que l'on dispose d'un vecteur d'entrée envoyé sur chaque neurone de la couche d'entrée, toutes les sorties de cette couche fournisssant l'entrée de la couche suivante…jusqu'à la **couche de sortie** qui produit donc un vecteur de sortie. Cette couche de sortie joue un rôle particulier dans l 'apprentissage dans la mesure où pour un échantillon donné on connait la sortie désirée par contre pour les neurones des

autres couches on ne peut pas connaître la sortie désirée et pour cette raison ces couches seront appelées **les couches cachées** du réseau.



Comment fonctionne un tel réseau dit « Perceptron multicouche » ?

En fait chaque connexion (une entrée vers un neurone ou la sortie d'un neurone vers l'entrée d'un autre neurone) marquée par une fléche sur ce schéma est affectée d'un poids. De ce fait le calcul s'effectue séquentiellement à partir de l'entrée soumise au réseau de la gauche vers la droite, chaque couche étant traitée entiérement avant de passer à la suivante.

4.9.2.4 Apprentissage du perceptron multicouche

L'apprentissage d'un réseau de neurones consiste à configurer les poids de toutes les connexions de façon à ce que le réseau réponde correctement à la présentation d'un ensemble d'échantillons.

Qu'est ce qu'un échantillon ?

C'est en fait l'ensemble constitué d'une entrée et de la sortie désirée correspondante.

Ce qu'on veut apprendre au réseau de neurones est rassemblé dans un ensemble d'échantillons. En supposant que le réseau de neurones fournit la réponse

désirée pour chaque échantillon on dira que le réseau a appris. La suite d'opérations consistant à arriver aux poids des connexions permettant une réponse correcte à toute sollicitation d'un échantillon s'appelle l'apprentissage.

Une fois que le réseau a appris ce qu'on voulait à partir des échantillons, le réseau va pouvoir maintenant appliquer et généraliser en fait ce qu'il a appris.

A une entrée quelconque il fournira une réponse.

Si la sortie est par exemple un numéro de classe dans une classification, alors chaque classe aura été définie par des échantillons et en généralisation le travail du réseau de neurones sera de fournir un numéro de classe à une forme inconnue qui se présente.

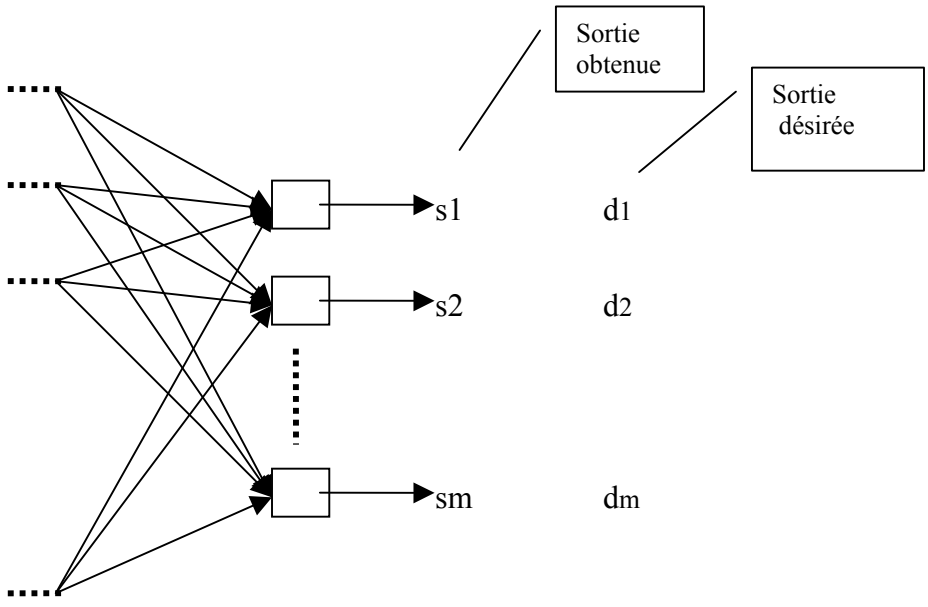
Quand considére t'on que la réponse pour un échantillon est correcte ? En fait il s'agit de comparer le vecteur de sortie désiré et le vecteur de sortie obtenu.

Ceci est quantifié classiquement par l'erreur quadratique

$$E^2 = \sum_{i=1}^{m} (s_i - d_i)^2$$

et on considérera donc que la réponse est correcte à un ε prés

réponse correcte $\Leftrightarrow$ $\sum_{i=1}^{m} (s_i - d_i)^2 \leq \varepsilon$ ε étant à choisir



On voit ici que si la réponse n'est pas correcte pour un échantillon donné il faudra corriger les poids des neurones de la couche cachée en utisant le fait qu'on connaît les m sorties désirées et les m sorties obtenues.

Mais pour les couches cachées on dispose des sorties mais pas des sorties désirées qu'il faudra donc estimer.

La méthode d'estimation consistera à rétropropager les erreurs en sortie du réseau vers la gauche pour permettre l'évolution des poids de toutes les connexions.

Il y aura donc en fait deux méthodes légérement différentes suivant que l'on traite des neurones de la couche de sortie ou des neurones des couches cachées.

Nous ne donnerons pas ici de démonstrations ou des justifications élaborées de la méthode d'apprentissage du perceptron multicouche mais simplement une description détaillée permettant de concevoir aisément les algorithmes correspondants

4.9.2.4.1 Algorithme général d'apprentissage

Nous allons décrire ici l'algorithme général puis dans les paragraphes suivants nous passerons à la description en détail sur la méthode d'évolution des poids d'un réseau sur présentation d'un échantillon.

TANTQUE

Condition arrêt non satisfaite

FAIRE

Choisir un échantillon ;
Calculer sortie de réseau ;
Calculer l'erreur quadratique E^2 ;

SI

$E^2 > \varepsilon$

ALORS

Corriger les poids de la couche de sortie
Corriger les poids des couches cachées

FSI

FINTANT

- La condition d'arrêt pouvant être
  le nombre d'itérations fixé est atteint
  Ou bien
  le réseau répond correctement à tous les échantillons

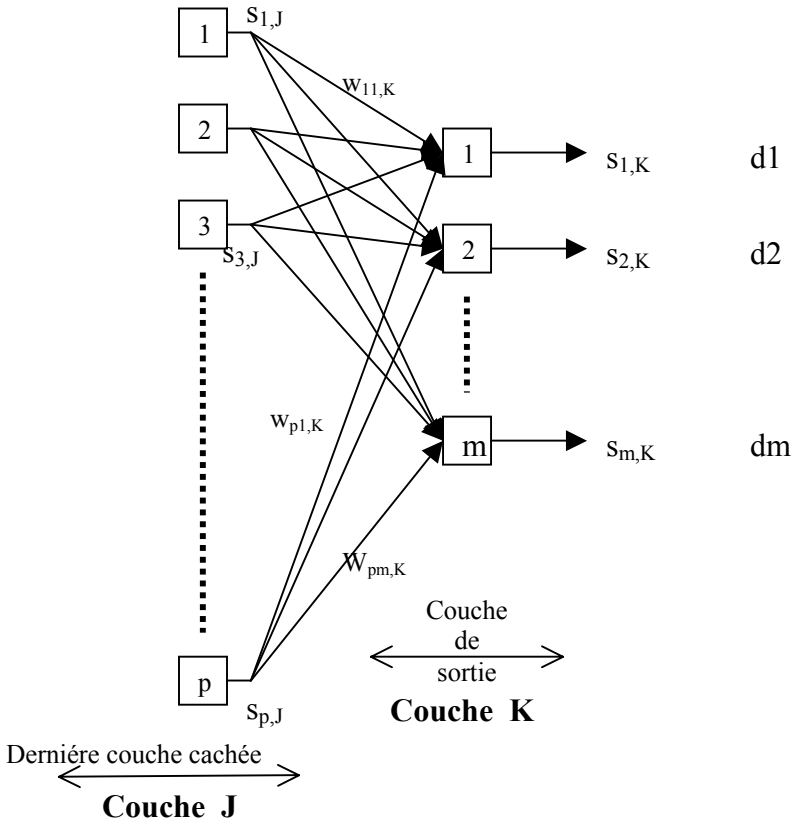
- Le choix d'un échantillon pouvant être :
  chaque échantillon chacun son tour en séquence
  ou
  tirage aléatoire d'un échantillon

4.9.2.4.2 Apprentissage de la couche de sortie

Tout d'abord nous supposons que nous avons mis une entrée échantillon, que nous avons calculé le vecteur de sortie, qu'on connaît bien sûr la sortie désirée et que l'erreur quadratique n'est pas acceptable. On doit donc procéder à la modification des poids des connexions et on commence donc par les neurones de la couche de sortie.

L'apprentissage de la couche de sortie va consister à calculer à partir des erreurs en sortie une modificarion des poids des neurones de cette couche.

Nous allons choisir les notations sur le schéma suivant pour décrire la méthode :



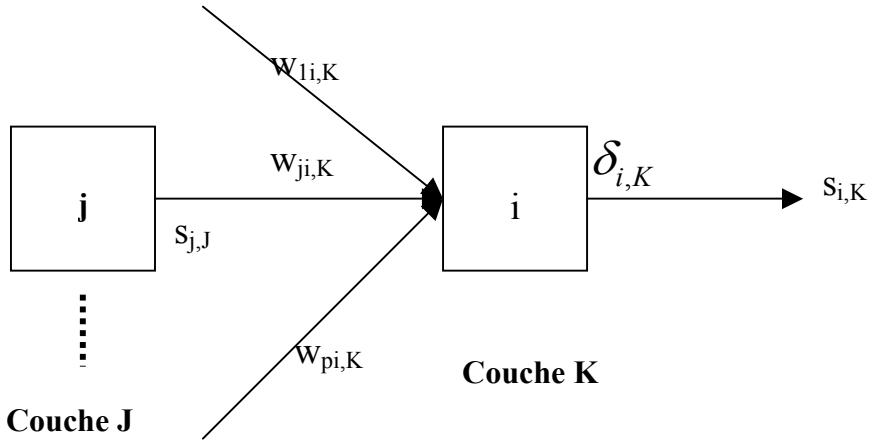
Pour chaque neurone i de la couche de sortie K on calcule un facteur de correction

$$\delta_{i,K} = s_{i,K} \times \left(1 - s_{i,K}\right) \times \left(d_i - s_{i,K}\right)$$

cela n'est valide que si $f(NET) = \dfrac{1}{1 + e^{-NET}}$

car $f'(NET) = f(NET) \times (1 - f(NET))$

sinon $\delta_{i,K} = f'(NET_{i,K}) \times \left(d_i - s_{i,K}\right)$



Si l'on note $w_{ji,K}$ le poids de la connexion reliant la sortie j de la couche J au neurone i de la couche de sortie K,

Alors on calcule le correctif à apporter à ce poids de la façon suivante :

$$\Delta w_{ji,K} = \eta \times \delta_{i,K} \times s_{i,J}$$

avec $\eta > 0$ *taux d'apprentissage*

et

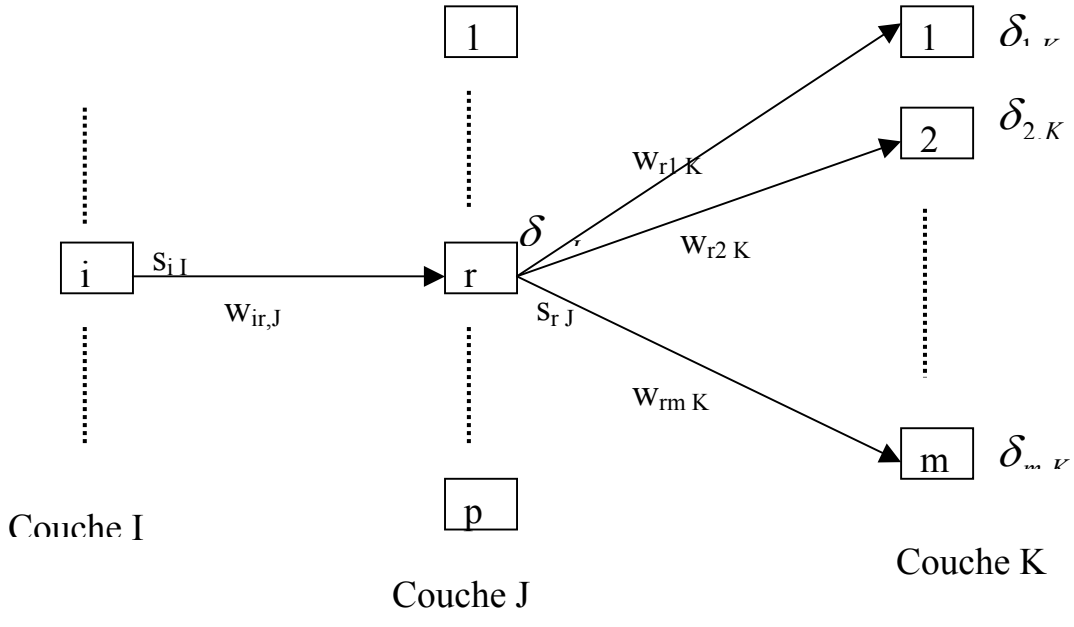
$$w_{ji,K} \leftarrow w_{ij,K} + \Delta w_{ji,K}$$

Ce travail est réalisé pour i variant de 1 à m et j de 1 à p

4.9.2.4.3 Apprentissage des couches cachées

L'étape de correction au niveau de la couche de sortie étant réalisée, on passe au traitement de la derniére couche cachée.

Nous allons indiquer dans le schéma ci-dessous les notations utilisées puis nous donnerons les formules à appliquer pour la correction des poids des neurones de la couche cachée en cours.



Pour mettre à jour le poids $w_{ir,J}$ on calcule un facteur correctif qui ne peut plus être proportionnel à la différence entre une sortie désirée sur le neurone r de la couche J et la sortie obtenue $s_{r,J}$ puisque la sortie désirée n'existe pas ! Cette différence sera remplaçée par une somme pondérée des $\delta_{i,K}$, ces valeurs répercutant les erreurs du niveau précédent que nous avons calculée à l 'étape précédente.

La formule est la suivante en supposant toujours que l'on utilise dans les neurones , la sigmoïde $f(NET) = \dfrac{1}{1+e^{-NET}}$

$$\delta_{r,J} = s_{r,J} \times (1 - s_{r,J}) \times \left(\sum_{q=1}^{m} w_{rq,K} \times \delta_{q,K} \right)$$

si on a une autre fonction sigmoïde il faudra calculer :

$$\delta_{r,J} = f'(NET_{r,J}) \times \left(\sum_{q=1}^{m} w_{rq,K} \times \delta_{q,K} \right)$$

on calcule alors

$$\Delta w_{ir,J} = \eta \times \delta_{r,J} \times s_{i,J}$$

puis

$$w_{ir,J} \leftarrow w_{ir,J} + \Delta w_{ir,J}$$

En remontant ainsi le réseau jusqu'à arriver aux entrées on aura mis à jour toutes les connexions.

Il est clair que le temps de calcul pour l'apprentissage d'un réseau de neurones sera en général énorme.

4.9.2.5 Choix d'un réseau en fonction du probléme à résoudre

Pour un probléme donné et nous prendrons un probléme de classification il n'existe pas de régle généréle permettant de concevoir facilement son réseau de neurones en fonction de la complexité du probléme à résoudre.

Bien sur, on conçoit aisément qu'un réseau complexe puisse résoudre des problémes complexes…mais comment choisir le bon nombre de couches et combien faut-il prendre de neurones sur chaque couche ?

Voici une régle basée sur l'expérience pour réaliser à l'aide d'un perceptron multicouches un classifieur pour n classes :

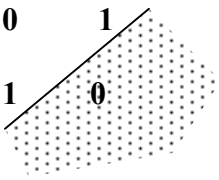
Il faut sur l'une des couches au moins E(n+n/2) neurones.

Sur le nombre de couches et le type d'hypersurfaces de séparation possibles entre classes, nous pouvons donner les éléments subjectifs suivants basés sur l'expérimentation :

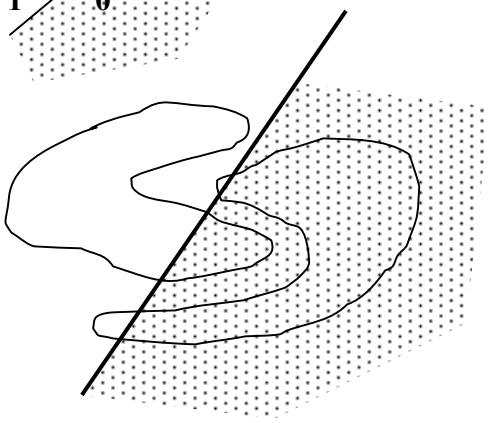
- **<u>Réseau monocouche</u>**

  Région de décision : hyperplan

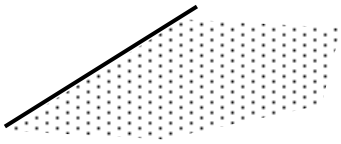
  Probleme du XOR :

  classes imbriquées :

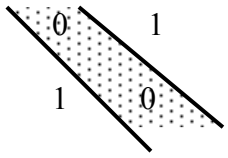
  Forme des régions :

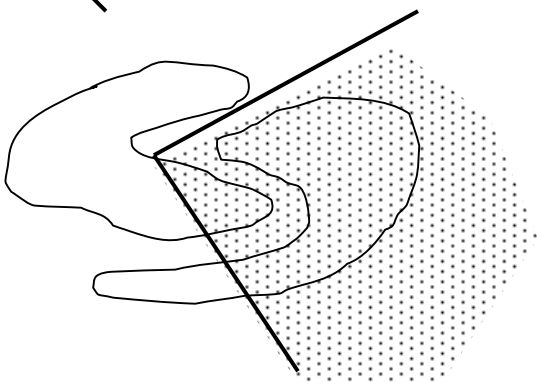
- **<u>Réseau à deux couches</u>**

  Région de décision : régions convexes ouvertes ou fermées

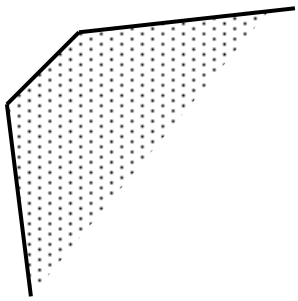
Probleme du XOR :



classes imbriquées :



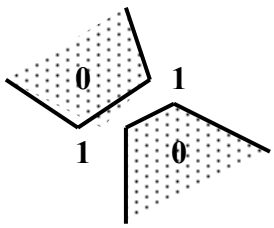
Forme des régions :



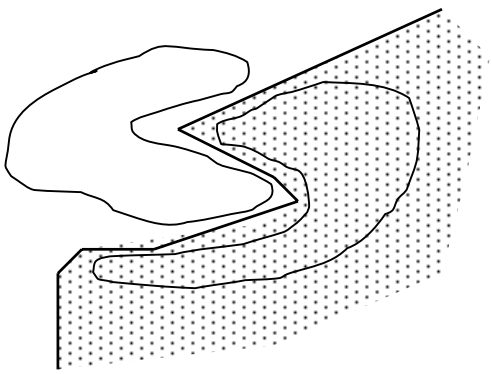
- **<u>Réseau à trois couches</u>**

  Région de décision : région arbitraire limitée par le nombre de noeuds

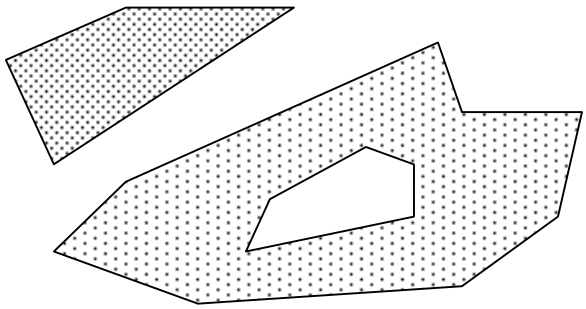
  Probleme du XOR :



  classes imbriquées :



Forme des régions :



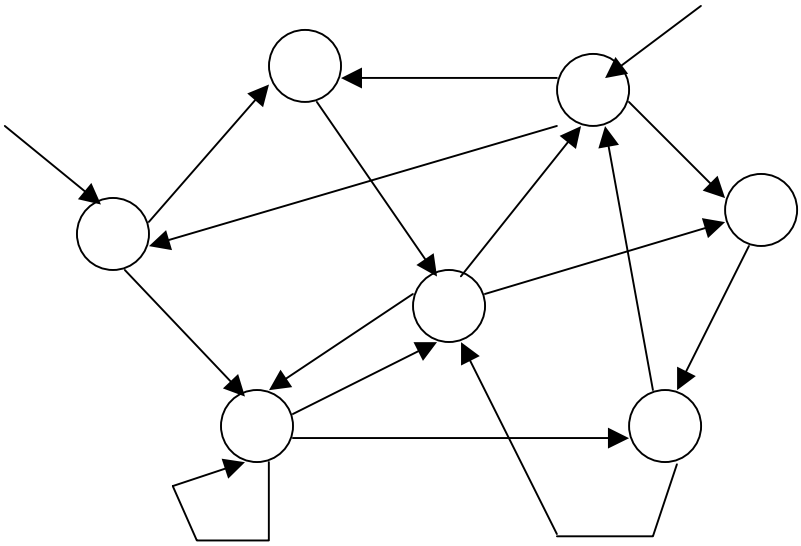
4.9.3 Réseaux totalement interconnectés

Nous resterons ici dans la généralité et donnerons des éléments sommaires d'information. Disons que dans ce type de réseau de neurones tous les neurones sont interconnectés à la façon des neurones biologiques. Il y a bien sur des entrées qui déclenchent une activation du neurone, mais à la différence des réseaux multicouches, il n'y a pas un fonctionnement séquentiel. Les sorties de neurones deviennent des entrées d'autres neurones et même la sortie d'un neurone peut être renvoyée en entrée sur le même neurone. Toutes les connexions sont pondérées.

Le fonctionnement d'un tel réseau est en général réalisé sur une base ou on fait évoluer tous les neurones en même temps jusqu'à convergence c'est à dire qu'aucun neurone ne change plus d 'état. Ce genre de réseau utilise bien sur des neurones qui ont pour sortie ACTIF ou NON ACTIF (en général 0 ou 1).

La réponse du réseau est alors l'état d'activation atteint par tous les neurones au lieu d'être la sortie de la couche de sortie du perceptron multicouches.

C 'est une approche analogue aux mémoires associatives.



Ce type de réseau de neurones comme tout réseau de neurones doit pouvoir subir un apprentissage. On dispose donc d'échantillons qui sont en fait des couples « entrées et liste de neurones actifs ». L'apprentissage consiste à modifier les poids des connexions jusqu'à obtenir la configuration désirée des neurones actifs.

Les poids initiaux d'un réseau n'ayant encore « rien appris » pourront par exemple être choisis de façon aléatoire.

Un exemple de réseaux de ce type est donné par les réseaux de Hopfield que nous verrons ultérieurement.

4.9.4 Reseaux à contrepropagation

Ces réseaux ont été conçus par Hecht et Nielsen en 1987.

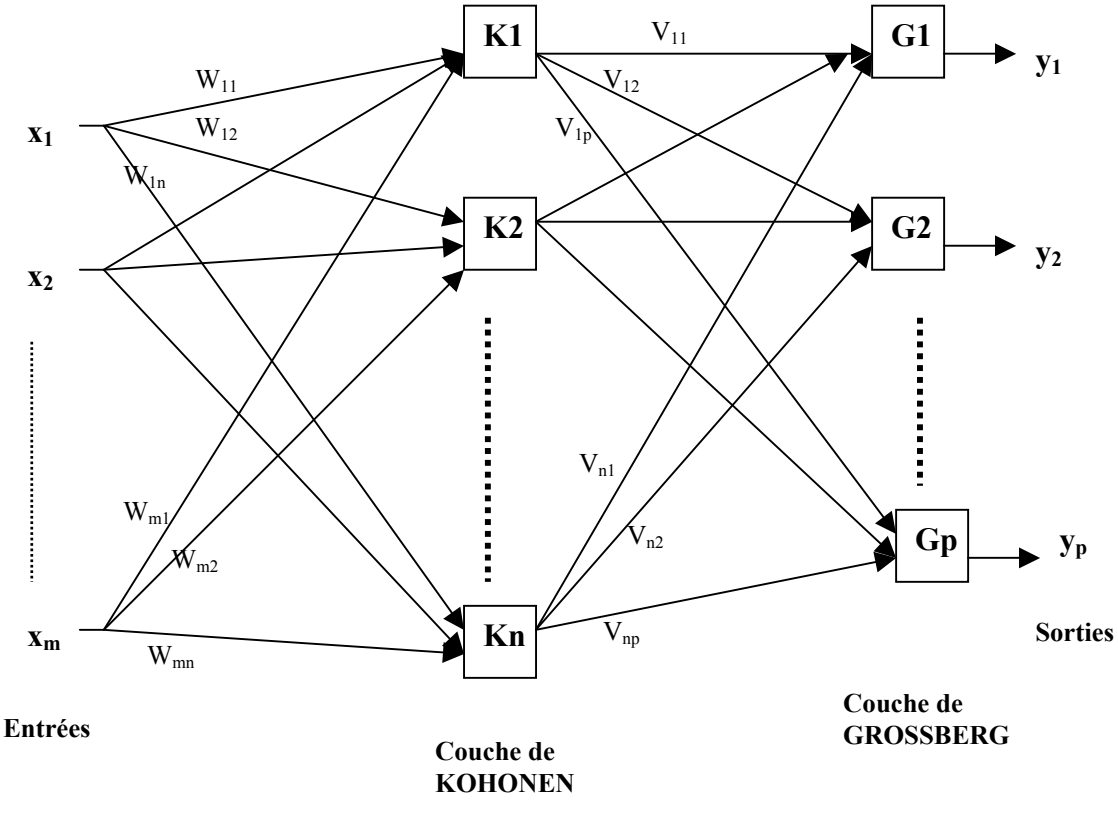
Leur temps d'apprentissage est réduit d'un facteur de 100 par rapport à un perceptron multicouche classique mais sa conception est moins générale que celle d'un réseau classique à rétropropagation du gradient.

4.9.4.1 Structure du réseau de contrepropagation

Ce réseau est composé de deux couches de neurones connectés séquentiellement.

La premiére couche est formée de neurones de « Kohonen » et la deuxiéme couche est formée de neurones de grossberg. Ces neurones ne fonctionnent pas de la même façon et leurs apprentissages sont différent.

Voici un exemple général de réseau de ce type :



Chaque entrée xj est connectée à chaque neurone de Kohonen Ki avec un poids ce Wji.

Chaque neurone de Kohonen Ki est donc affecté d'un vecteur poids Wi avec

$${}^{t}W_i = \left(W_{1i} \quad .W_{2i} ..... \quad W_{mi}\right)$$

L'ensemble des poids de toutes les connexions de la couche de Kohonen peut donc être représentés par une matrice de n lignes et m colonnes

$$W = \begin{Bmatrix} W_{11} & W_{12} & . & . & W_{1n} \\ W_{21} & W_{22} & . & . & W_{2n} \\ . & . & . & . & . \\ . & . & . & . & . \\ W_{m1} & W_{m2} & . & . & W_{mn} \end{Bmatrix} = \{W_1 \quad W_2 \quad . \quad . \quad W_m\}$$

De la même façon l'ensemble de tous les poids de la couche de Grossberg peut être représenté par une matrice de n lignes et p colonnes :

$$V = \begin{Bmatrix} V_{11} & V_{12} & . & . & V_{1p} \\ V_{21} & V_{22} & . & . & V_{2p} \\ . & . & . & . & . \\ . & . & . & . & . \\ V_{n1} & V_{n2} & . & . & V_{np} \end{Bmatrix} = \{V_1 \quad V_2 \quad . \quad . \quad V_p\}$$

Ce réseau de neurones comme tous les réseaux doit être capable d'apprendre par apprentissage puis ensuite d'être utilisé en « généralisation ».

Si l'on regarde ce réseau comme devant résoudre des problémes rencontrés en reconnaissance des formes et classification, nous pouvons dire qu'il est équivalent à un algorithme de nuées dynamiques qui regrouperait des entrées par affinité et donnerait une sortie (donc un label) sur la sortie de Grossberg. Mais ce réseau peut résoudre bien des problémes différents. L'apprentissage consistera bien sûr comme tout réseau de neurone, a mettre à jour les poids des connexions.

4.9.4.2 Fonctionnement du réseau en « génèralisation »

On suppose donc que le réseau de neurone a déjà subi son étape d'apprentissage et que donc tous les poids des connexions sont figés.

Le fonctionnement de ce réseau est assez original, puisqu'en fait pour les neurones de la couche de Kohonen, la réponse de chaque neurone dépend de l'activation de tous les neurones de la couche. Le fonctionnement est séquentiel

puisqu'on calcule d'abord les sorties des neurones de Kohonen puis les sorties des neurones de grossberg

- **calcul des sorties de la couche de Kohonen :**

Le fonctionnement pour cette couche est qualifié de tout pour le gagnant (« winner takes all »)

Considérons le neurone de Kohonen Kj et calculer son activation NETj :

$$NET_j = W_{1j} \times x_1 + W_{2j} \times x_2 + ... + W_{mj} \times x_m = \sum_{i=1}^{m} W_{ij} \times x_i$$

Le neurone de Kohonen K_l pour lequel l'activation est la plus grande aura sa sortie égale à 1 et tous les autres neurones auront leur sortie à 0

donc

<u>Si</u> on a $NET_l > NETi$ pour tout $i \neq l$

<u>Alors</u> $k_l=1$ et $k_i=0$ pour tout $i \neq l$

Notons ici qu'il existe des variantes du type « les premiers se partagent les gains »

Voici un exemple qui considére que deux neurones de Kohonen se partagent les gains : les deux neurones ayant les activations les plus elevées auront dans l'ordre les sorties 1 et 0.5 par exemple, les autres neurones ayant leur sortie à 0.

- **calcul de la couche de Grossberg :**

Le calcul de la couche de grossberg est plus « classique »

En fait chaque neurone de Grossberg produit une sortie qui est la somme pondérée des entrées, sachant que ces entrées sont les sorties k_1 k_2 … k_n des neurones de Kohonen.

Donc la sortie y_j du neurone G_j est en fait égale à l'activation de ce neurone, soit :

$$y_j = NET_j = \sum_{i=1}^{n} V_{ij} \times k_i$$

et comme seul le neurone n° l produit une sortie à 1

le vecteur de sortie de la couche de Grossberg Y vaut :

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \\ \\ y_p \end{pmatrix} = \begin{pmatrix} V_{l1} \\ V_{l2} \\ \\ \\ V_{lp} \end{pmatrix}$$ et donc $Y= V_l$

4.9.4.3 Apprentissage du reseau

L'apprentissage de ce type de réseau est original car il procéde en deux temps.

Premier temps :

On réalise l'apprentissage de la couche de Kohonen à l'aide d'un ensemble d'échantillons qui ne sont en fait que des vecteurs d'entrée sans définir de sorties attendues sur la couche de Grossberg.

Cet apprentissage consiste à ajuster les poids de la couche de Kohonen de façon que des vecteurs d'entrée voisins activent le même neurone.

En fait à la fin de cette phase d'apprentissage on aura réalisé un travail de type nuées dynamiques où on regroupe des formes qui se ressemblent (mais ici on ne définit pas le nombre de classes à priori. Sur les n neurones de Kohonen seuls k d'entre eux peuvent être au moins une fois activés et dans ce cas k est le nombre de classes trouvées.

Deuxiéme temps :

L'ensemble des échantillons de départ a donc été partagé en k paquets « plus ou moins homogénes » et donc k neurones de la couche de Kohonen pourront au moins une fois fournir une sortie à 1. L'apprentissage de la couche de Grossberg consiste à définir le vecteur poids Vi correspondant qui sera la sortie définie pour ce paquet ;

Si par exemple il s'agit du neurone K_3 par exemple alors la sortie sera le vecteur V_3 qu'on pourra fixer.

On voit donc qu'avec ce type de réseau on peut faire des nuées dynamiques mais sans avoir à fixer de façon atrbitraire le nombre de classes à trouver. Les classes étant trouvées la couche de Grossberg permet de donner un « label » à chaque classe trouvée.

4.9.4.3.1 Apprentissage de la couche de Kohonen

Revenons sur le fonctionnement de la couche de Kohonen et voyons quelle interprétation vectorielle on peut donner à la régle « tout pour le gagnant »

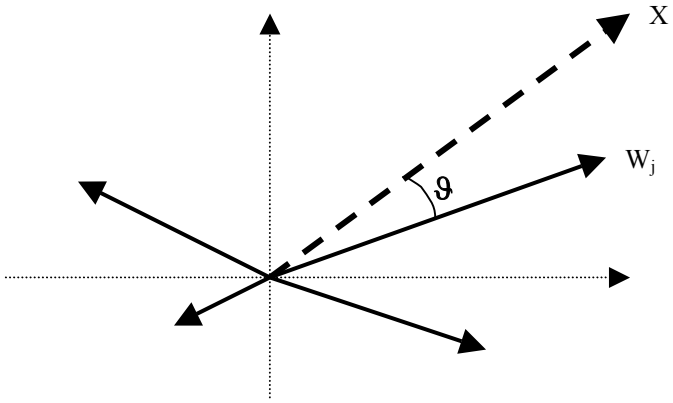
Rappelons que :

$$NET_j = W_{1j} \times x_1 + W_{2j} \times x_2 + ... + W_{mj} \times x_m = \sum_{i=1}^{m} W_{ij} \times x_i$$

Et le neurone de Kohonen K_l pour lequel l'activation est la plus grande aura sa sortie égale à 1 et tous les autres neurones auront leur sortie à 0

Si on a $NET_l > NETi$ pour tout $i \neq l$
Alors $k_l=1$ et $k_i=0$ pour tout $i \neq l$

En fait NET_j représente le produit scalaire du vecteur poids W_j et du vecteur d'entrée X. Pour fixer les idées supposons que ces vecteurs sont des vecteurs de R^2 et représentons ces vecteurs graphiquement.



$$NET_j = \|W_j\| \times \|X\| \times \cos(\vartheta)$$

En fait on peut interpréter ce produit scalaire comme une mesure de la « ressemblance de X et de W_j» en terme de direction puisque ce produit scalaire est d'autant plus élevé que ϑ est proche de 0.

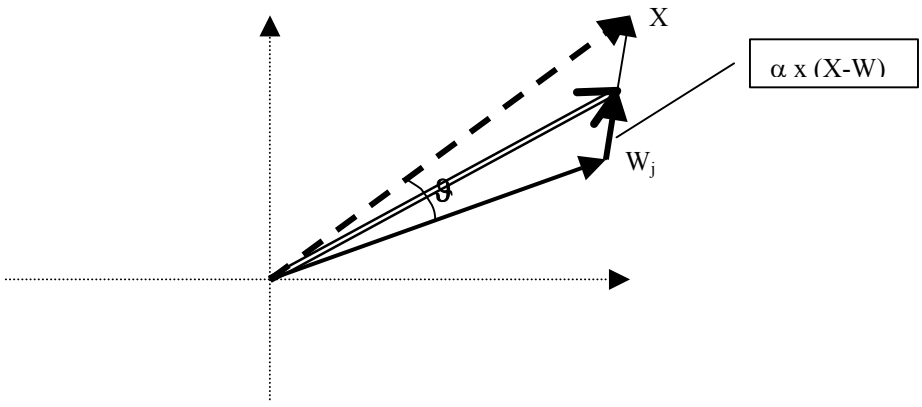
Lors de la phase d'apprentisssage on présente donc successivement les diverses entrées (choix aléatoire ou pas) et on va modifier les poids du neurone gagnant (donc les composantes du vecteur W_j) de façon à « rapprocher » le vecteur poids de l'entrée qui l'a activé.

Seul donc le vecteur poids qui a été activé va être modifié suivant la rêgle suivante :

$$W_j \leftarrow W_j + \alpha \times \left(X - W_j\right) \quad \text{avec } 0 < \alpha < 1$$

α *étant un coefficient définissant classiquement le taux d'apprentissage*

exemple α =0.5

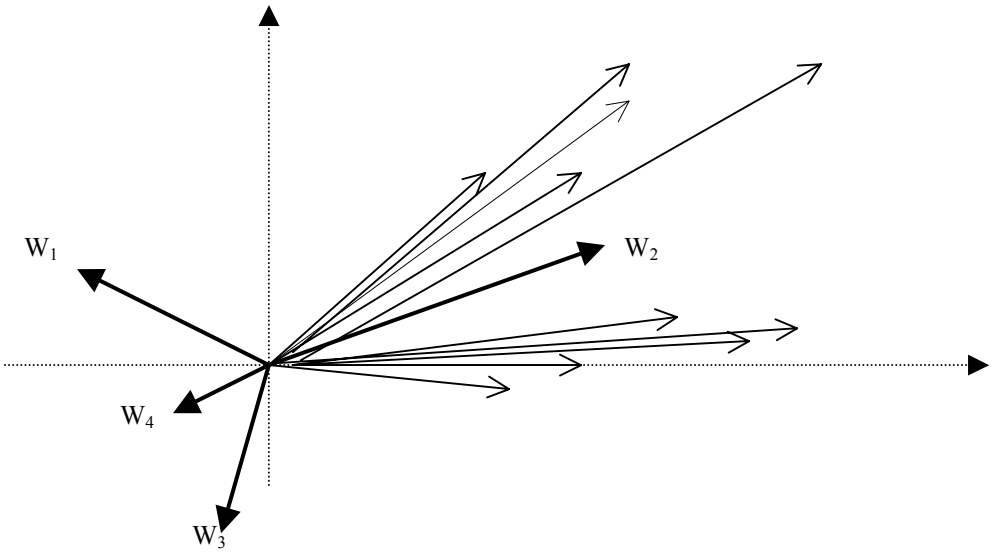


Exemples de valeurs pour α:

Si on a beaucoup d'entrées, on peut choisir une valeur de α petite, on peut aussi choisir des valeurs de α qui décroissent au cours des présentations des entrées. D'autre part on peut décider de présenter chaque entrée plusieurs fois (bien sûr pas les unes derriére les autres) auquel cas on pourra prendre des α très petits.

Un probleme important dans cette technique d'apprentissage est celui de l'initialisation des poids de la couche de Kohonen et ici il est en général conseillé d'avoir une poliktique d'initialisation des poids liée aux entrées à traiter.

Voyons ce probléme sur un exemple où les entrées sont les vecteurs représentés en trait fin et où on aurait quatre neurones sur la couche de Kohonen:



On voit bien que les entrées peuvent être divisées en deux groupes distincts et pourtant il est clair que dans le cas de figure représenté seul le neurone K2 par l'intermédiaire de son vecteur poids W2 va capter toutes les entrées et donc conclure qu'il n'y a qu'un seul groupe homogéne pour les entrées.

Il est donc important que ce type de situation soit évitée.

Cela pourra être réalisé par une heuristique d'initialisation des poids mais aussi en modifiant la technique d'apprentisssage.

Quelques techniques d'initialisation des vecteurs poids :

- Prendre tous les vecteurs poids identiques par exemple

$$\left(\frac{1}{\sqrt{m}} \quad \frac{1}{\sqrt{m}} \quad \ldots\ldots\ldots \quad \ldots\ldots\ldots \quad \frac{1}{\sqrt{m}}\right)$$

puis réaliser l'apprentissage avec des entrées modifiées qui au départ iront chercher des vecteurs poids .

si une entrée est x1 x2 …… xm

alors l'entrée modifiée sera :

$$\left(\beta \times x_1 + \frac{1-\beta}{\sqrt{m}} \quad \beta \times x_2 + \frac{1-\beta}{\sqrt{m}} \quad \ldots\ldots\ldots \quad \ldots\ldots\ldots \quad \beta \times x_m + \frac{1-\beta}{\sqrt{m}}\right)$$

On part avec $\beta = 1$ puis petit à petit au cours de l'apprentissage on fait tendre β vers 1

De cette façon on verra les vecteurs poids s'adapter aux données en entrée.

- Ajouter un bruit blanc gaussien aux entrées puis au cours de l'apprentisssage diminuer petit à petit l'intensité de ce bruit.

- On prend des vecteurs poids initiaux aléatoires et lors de l'apprentissage au lieu d'ajuster les poids du seul neurone gagnant on ajuste les poids de tous les neurones.

Un autre probléme important est celui de l'ordre avec lequel on va présenter les entrées. Puisque chaque entrtée présentée modifie un vecteur poids. Ici en général on fera des tirages aléatoire sur l'ensemble des entrées , une entrée déjà présentée étant retirée du lot. Une autre possibilité est de décider de pouvoir tirer chaque entrée non plus une fois mais n fois.

4.9.4.3.2 Apprentissage de la couche de Kohonen

Le principe consiste à modifier les poids des connexions reliant le neurone de Kohonen gagnant (sortie ==1) à la couche de Grossberg.

Sachant que l'on connaît quel vecteur de sortie Y on désire suivant le neurone de Kohonen qui est activé.

Si le neurone de Kohonen qui fournit 1 est le neurone K_i , alors l'apprentissage qui est appliqué est le suivant :

$$\forall j = 1 \ldots p \qquad V_{ij} \leftarrow V_{ij} + \beta \times \left(y_j - V_{ij} \right) \times k_i$$

avec k_i sortie du neurone de kohonen n° i fournissant 1

y_j composante j du vecteur de sortie désiré sur Grossberg

V_{ij} poids connexion du neurone K_i au neurone G_j

Au départ β est proche de 0.1 par exemple puis on le diminue progressivement au cours de l'apprentisssage.

4.9.4.4 Exemples d'applications

Nous donnerons deux exemples , un exemple en reconnaissance de formes (classification non supervisée) et un exemple en interpolation de fonction mathématique.

4.9.4.4.1 Contrepropagation et classification automatique

Un des exemples d'utilisation de ce réseau de neurones est de réaliser une sorte de fonctionnnement en k-moyennes (nuéesz dynamiques : ISODATA1) ou le nombre de classes k est en fait découvert automatiquement par le réseau. Ce réseau dans un ensemble d'entrées va donc réaliser des regroupements en paquets (classes) « homogénes ».

En fait on présente toutes les formes pour la phase d'apprentissage qui va permettre que des neurones de la couche de Kohonen regrouperont chacun un paquet d'entrées proches. En fait un vecetur poids sera « piégé » par chaque paquet «homogéne».

L'intérêt ici par rapport aux nuées dynamiques est que la méthode découvre elle-même le nombre de classes pourvu qu'on ait prévu au départ un nombre suffisant de neurones de Kohonen.

La couche de Grossberg ne sert plus qu'à une chose c'est de donner un « label » (sous forme d'un nombre) à chaque classe trouvée. De ce fait un seul neurone suffit pour la couche de Grossberg, les poids étant par exemple : 1, 2, 3, …..

Chaque neurone de la couche de Kohonen sortira un et un seul de ces nombres, donc on est sûr que chaque classe trouvée donnera un label (nombre) différent.

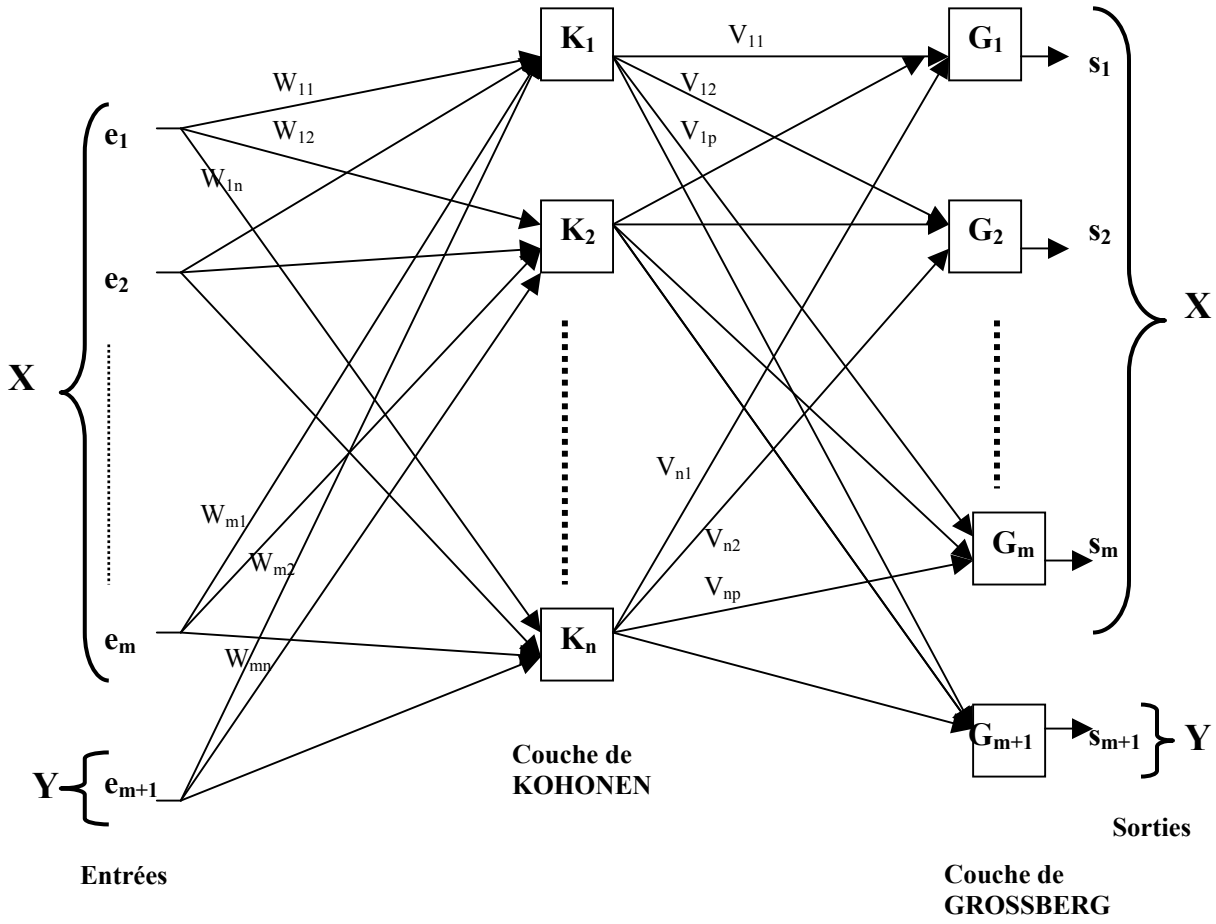
4.9.4.4.2 Approximation de fonction mathématique

Soit à approximer une fonction réelle Y=F(X) et sa fonction inverse F^{-1}, X étant un vecteur de R^d et supposons que l'on dispose d'exemples, c'est à dire de couples (Y, X).

$$^{t}X = (x_1, x_2, ..., x_m) \in R^m \quad et \quad Y \in R$$

On dispose donc d'exemples (Y_1,X_1), (Y_2,X_2), ……..(Y_k,X_k)

On considére alors le réseau de Kohonen-Grossberg suivant :



Il y a m+1 entrées et m+1 sorties.

Pour l'apprentissage :

on procéde en présentant tous les exemples (X_i, Y_i) de la façon suivante :

D'abord En entrée X_i et $e_{m+1}=Y_i$ et on désire en sortie X_i, Y_i
Puis en entrée X_i et $e_{m+1}=0$ et en sortie on désire X_i, Y_i
Puis en entrée (0 0 ….0) et $e_{m+1}=Y_i$ et en sortie on désire X_i, Y_i

En fonctionnement en généralisation :

Si on donne en entrée un vecteur X et $e_{m+1}=0$
on obtiendra Y approximation de F(X)
Si on donne en entrée un X= vecteur nul et un $e_{m+1}=Y$
en sortie, on aura X qui sera une approximation de $F^{-1}(Y)$

4.10 EXERCICES

1) Décrire un neurone linéaire à seuil (sortie 0 où 1) à entrées réelles et expliquer son fonctionnement.

2) Prouver que ce neurone est incapable de distinguer les deux classes suivantes:
- w1 définie par des entrées égales à (1,1) et (0,0)
- w2 définie par des entrées égales à (1,0) et (0,1)

expliquer pourquoi.

3) Proposer un réseau de neurones linéaires à seuil, capable de résoudre le problème précédent de séparation en deux classes. Commenter.

---

1) Proposer un neurone linéaire à seuil permettant de réaliser le ET logique de trois valeurs booléennes (description des poids et du seuil), et un neurone linéaire à seuil donnant le OU logique de deux valeurs booléennes (description des poids et du seuil).
2) Donner alors la représentation du réseaux de neurones permettant de calculer $\bigcup_{i=1}^{2}(\bigcap_{j=1}^{3} A_{ij})$, Aij valant 0 ou 1.

---

1) Décrire un neurone linéaire à seuil (sortie 0 où 1) à entrées réelles et expliquer son fonctionnement.

2) Proposer un neurone linéaire à seuil, capable de résoudre le problème de séparation des deux classes suivantes définies dans R^2 par les échantillons :
- classe 1 : (-1,-1)
- classe 2 : (-1,1) (1,1) (1,-1)

Pour cela,
représenter graphiquement les formes dans R^2 et tracer une droite séparant linéairement les deux classes
donner son équation et en déduire le neurone capable de reconnaître ces deux classes

---

1) Décrire un neurone linéaire à seuil (sortie 0 où 1) à entrées réelles et expliquer son fonctionnement.
2) Prouver que ce neurone est incapable de distinguer les deux classes suivantes:
- w1 définie par des entrées égales à (1,1) et (0,0)
- w2 définie par des entrées égales à (1,0) et (0,1)

expliquer pourquoi.
3) Proposer un réseau formé de neurones linéaires à seuil, capable de résoudre le problème de séparation des deux classes suivantes définies dans R^2 par les échantillons :
classe 1 : (-1,-1)
classe 2 : (-1,1) (1,1) (1,-1)

On veut utiliser un réseau de Kohonen Grossberg pour réaliser un classifieur supervisé.

On suppose disposer d'un certain nombre d'échantillons (vecteurs de R^n) en vrac représentant toutes les classes recherchées (en nombre k supposé inférieur à 6).

1) Décrire le réseau que vous proposez
2) Expliquer les problèmes posés par l'apprentissage

3) Décrire le fonctionnement du classifieur obtenu

---

1) proposer un neurone linéaire à seuil permettant de réaliser le ET logique de trois valeurs booléennes (description des poids et du seuil) , et un neurone linéaire à seuil donnant le OU logique de deux valeurs booléennes(description des poids et du seuil). Donner alors la représentation du réseaux de neurones permettant de calculer $\bigcup_{i=1}^{2}\left(\bigcap_{j=1}^{3} A_{ij}\right)$

2) Définir ce qu'est un perceptron multicouches et comment l'utiliser en reconnaissance des formes. Détailler la méthode d'apprentissage (donner si possible une formulation détaillée sur un exemple)

.

Comment utiliser un réseau de Kohonen-Grossberg pour faire de la classification supervisée (échantillons de chacune des classes fournis).

---

5 METHODES STRUCTURELLES EN RECONNAISSANCE DES FORMES

Avant de passer aux approches structurelles en reconnaissance des formes nous allons étudier rapidement les bases fondamentales de ces approches.

5.1 LANGAGES ET GRAMMAIRES

5.1.1 Quelques définitions

5.1.1.1 Langages

Soit X un alphabet, par exemple X= {A,B,C,…..,X,Y,Z,a,b,c,d,e….y,z}

Alors on note X* l'ensemble des mots qu'on peut écrire avec X

Un langage L est un sous ensemble de X*

Exemples de langages :

L={ x ∈ X* / nombre de lettres de x est pair}

L= { a,aa,aaa, …..a^n pour n>0….}

L=Φ

5.1.1.2 Produit de deux langages

Soient L_1 et L_2 deux langages définis sur deux alphabets quelconques X_1 et X_2

Alors le produit de L_1 et L_2 est défini sur l'alphabet $X_1 \cup X_2$ par

$$L_1L_2 = \{ xy / x \in L1 \text{ et } y \in L2 \}$$

Exemple :

L_1= { a, abc } et L_2 = {bcd,d}

alors

L_1L_2 = { abcd, ad, abcbcd, abcd}

5.1.1.3 Fermeture d'un langage

La fermeture d'un langage L , notée L^* est définie de la façon suivante:

$$L^* = \bigcup_{n \geq 0} L^n$$ avec $L^0=\{\lambda\}$ λ est le mot vide

et $L^n=LL^{n-1}$ pour n >= 1

5.1.1.4 Grammaire et langage généré

Une grammaire permet de représenter des langages infinis (à nombre infini de mots) par un nombre fini de symboles et de régles.

On peut définir une grammaire de la façon suivante

G = (X , V , S , P)

Avec

- **X** : alphabet terminal (les "lettres")
- **V** : alphabet auxiliaire (ou non terminal)
- **S** : élément particulier de V appelé axiome permettant comme son nom l'indique le démarrage de la génération de mots
- **P** : ensemble de régles de production qui permettront le déroulement du processus de génération de mots

Définissons une régle de production :

Une régle de production peut être définie par le passage d'une chaine α non vide formée de lettres de l'alphabet terminal et d'éléments de l'alphabet auxiliaire à une nouvelle chaine β du même type pouvant être vide.

$$\alpha \rightarrow \beta$$

avec $\alpha \in (V \cup X)^* \times V \times (V \cup X)^*$

et $\beta \in (V \cup X)^*$

Cela signifie que α et β sont des chaines formées en concarténant un nombre quelconque d'élément de V ou X mais α contient au moins une lettre de l'alphabet et β peut être le mot vide.

Utilisation d'une règle de production :

Soit $a \in (V \cup X)^* \times V \times (V \cup X)^*$

Alors on dit qu'on peut reécrire **a** en **b** par la régle de production $\alpha \rightarrow \beta$

Si $a = a_1 \alpha a_2$ et que $b = a_1 \beta a_2$

on écrit alors que $a \Rightarrow b$

Une régle de production permet donc de transformer une chaine de $(V \cup X)^* \times V \times (V \cup X)^*$ en une nouvelle chaine (qui peut éventuellement être vide).

Décrivons maintenant le processus génératif associé à la grammaire G :

On part de l'axiome S qui peut être considéré comme une régle de production particuliére permettant d'initialiser le processus des reécritures successives puis par application successives de régles de production éligibles on procéde à des reécritures successives a_1 a_2 ……

Si a_n n'est composé que de symboles du vocabulaire terminal (alphabet X) alors on ne peut plus faire de reécritures et a_n est donc un des mots générés par la grammaire G.

L'ensemble des mots qui peuvent être générés par la grammaire forme le langage généré par G que nous noterons L(G).

Exemples de grammaires et langages générés correspondants :

- Exemple 1 :

Grammaire G_1 **X={a,b}**
V={S,A}
P= $\begin{cases} S \rightarrow aS \\ S \rightarrow bA \\ bA \rightarrow bbA \\ A \rightarrow a \end{cases}$
S axiome

Exemple de reécritures :

$S \Rightarrow aS \Rightarrow aaS \Rightarrow aabA \Rightarrow aabbA \Rightarrow aabbbA \Rightarrow aabbba$

et donc aabbba $\in L(G_1)$

en fait on voit que

$L(G_1)$ ={ $a^n b^m a$ avec n>=0 et m>=1}

Notons qu'on peut donner une forme très simplifiée de la grammaire G1 résumant toute la définition de la grammaire.

Grammaire G_1

$$\begin{cases} S \to aS \mid bA \\ bA \to bbA \\ A \to a \end{cases}$$

- Exemple 2 :

Donnons directement la grammaire G2 sous sa forme simplifiée

Grammaire G_2

$$\begin{cases} S \to aSBA \mid abA \\ AB \to BA \\ bB \to bb \\ bA \to bc \\ cA \to cc \\ aB \to ab \end{cases}$$

Exemple de reécritures:

$S \Rightarrow aSBA \Rightarrow aabABA \Rightarrow aabBAA \Rightarrow aabbAA \Rightarrow aabbAA \Rightarrow aabbcA \Rightarrow aabbcc$

et on remarque que

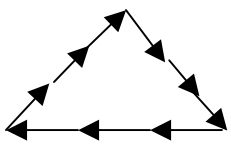
$L(G_2) = \{ a^n b^n c^n$ avec n >= 1 $\}$

Application :

Si a est ← b est ↗ et c est ↘

Et supposons que tous ces vecteurs sont de longueur 1

Alors $L(G_2)$ est l'ensemble des triangles équilatéraux à une base horizontale et de coté de longueur un entier positif quelconque.



5.2 GRAMMAIRES REGULIERES ET AUTOMATES FINIS

5.2.1 Grammaire réguliére

Une grammaire réguliére est une grammaire dont les régles de production ne peuvent prendre que deux formes très simples :

$$A \to aB \quad avec \quad A\,et\,B \in V$$
$$ou \quad A \to a \quad avec \quad a \in X$$

exemple :

Grammaire G_R

$$\begin{cases} S \to aS \mid bA \\ A \to bA \mid a \end{cases}$$

alors on a ,

$L(G_R)$ ={ $a^n b^m a$ avec n>=0 et m>=1}

Nous allons maintenant donner la définition d'un automate fini et voir que toute grammaire réguliére peut être reprtésentée par un automate fini et vice versa.

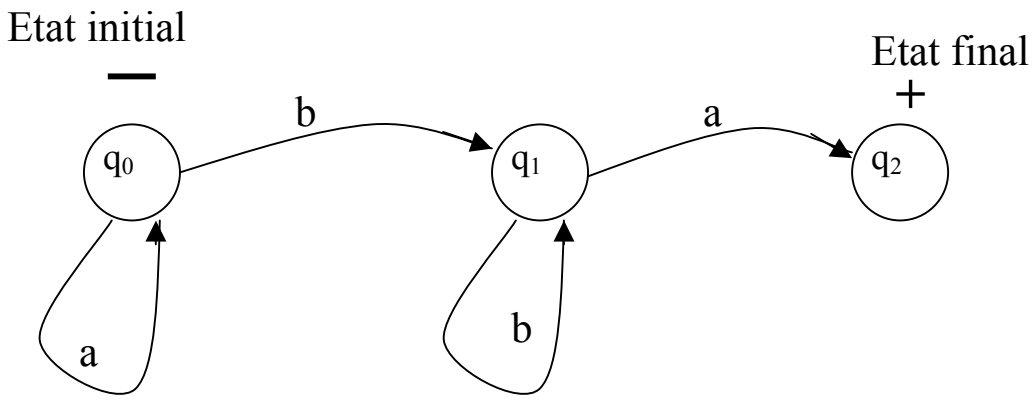
5.2.2 Automate fini

Un automate fini A est défini par le formalisme suivant:

A = (X , Q , δ , q_0 , Q')

avec **X :** alphabet terminal
Q : ensemble fini d'états
q0 ∈ Q : état initial
Q' ⊂ Q : ensemble des états finals
δ : une application de l'ensemble **Q.X** dans **Q**
appelée fonction de transition

Voici par exemple une représentation graphique de l'automate fini qui correspond à la grammaire réguliére G_R vue précedemment.



Le langage généré par l'automate A L(A) est identique au langage généré par la grammaire réguliére $L(G_R)$

On remarque que,

- l'état initial q_0 et ses transitions correspond à l'axiome S
- Les états finals et leurs transitions correspondent aux régles qui produisent une lettre de V
- Les états intermédiaires et leurs régles de transition correspondent aux autres régles de production ;

Dans le cas présent

q_0 et ses deux transitions correspond à $S \to aS|bA$

q_1 et ses deux transitions correspond à $A \to bA|a$

la régle $A \to a$ permettant ici d'atteindre un état final.

Remarque :

La fonction de transition δ qui est une application de l'ensemble **Q.X** dans Q n'est pas totalement définie dans notre exemple.

En effet cette application peut se formaliser de la façon suivante

$$q_j = \delta(q_i, a_k) \quad \forall q_j, q_i \in Q \; et \; \forall a_k \in X$$

Or dans notre automate on voit que l'application δ n'est pas totalement définie puisqu'on ne définit pas les valeurs suivantes :

$$\delta(q_2, a) \; et \; \delta(q_2, b)$$

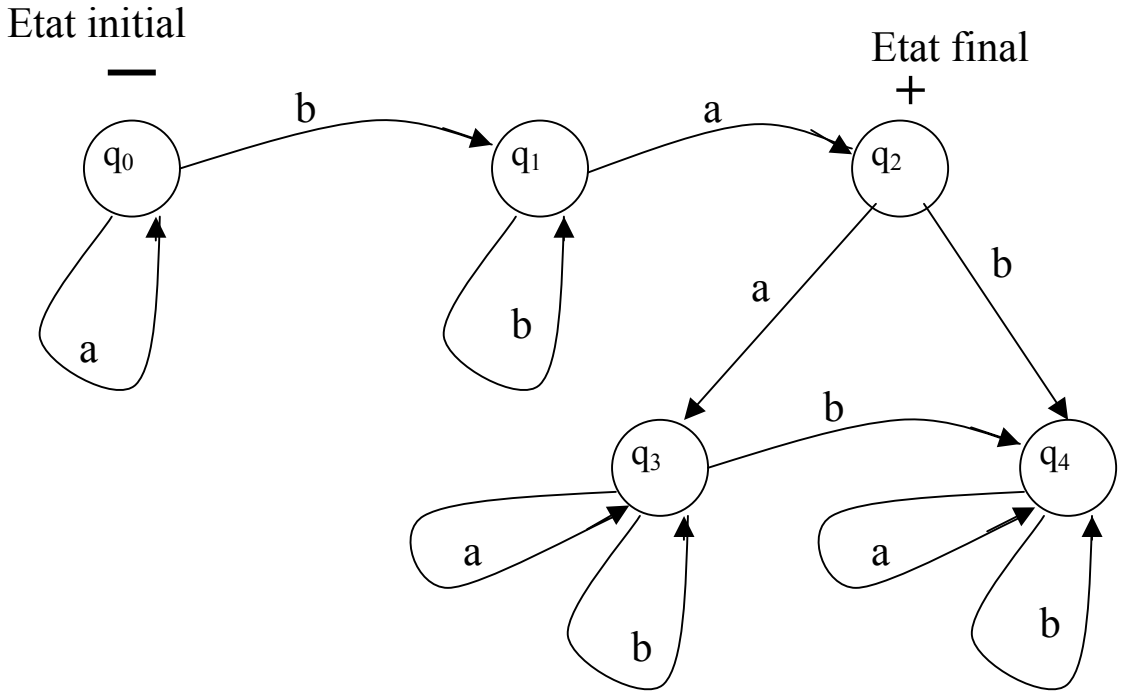
Cela généré deux états qui ne peuvent être des états finals . Ces deux états seront des états « poubelles » que nous noterons par exemple q_3 et q_4.

Ces nouveaux états se rajoutent à l'ensemble Q et on doit donc à nouveau compléter l'application d en ajoutant :

$$q_3 = \delta(q_3, a) \ et \ q_3 = \delta(q_3, b)$$
$$q_4 = \delta(q_4, a) \ et \ q_4 = \delta(q_4, b)$$

On obtient alors ce qu'on appelle un auttomate fini complet où toutes les transitions sont définies. Dés qu'on tombe sur un état final on a défini un mot du langage mais si on tombe dans un état « poubelle » on n'en sortira plus.

Voici la représentation graphique de cet automate :



5.3 FORMES ET MOTS, DECISION ET GRAMMAIRES REGULIERES

Jusqu'à maintenant nous n'avons travaillé qu'avec des formezs qui étaient dféfinies par un vecteur de valeurs, mais comme nous l'avons vu des formes peuvent être définies comme des mots d'un langage (par exemple l'ensemble des triangles équilatéraux à base horizontale et de coté de longueur un entier positif quelconque).

En fait nous traitons ici maintenaht des formes au sens de la morphologie.

Un exemple concret est le codage de frontiéres de régions connexes en traitement d'images par codage de Freeman (voir cours de traitement d'images numériques). Les lettres de l'alphabet etant tous les vecteurs élementaires définis par

les composantes suivantes (0,1),(1,1),(0,1),(-1,1),(-1,0),(-1,-1),(0,-1), le contour d'une région connexe est alorsle « mot » écrit avec une suite de ces vecteurs.

Des formes qui se ressemblent et qu'on considérera comme appartenant à la même classe sont en fait un ensemble de mots écrits avec un alphabet particulier. Si l'on peut considérer l'ensemble des mots de cette classe comme un langage généré par une grammaire G, alors G sera la caractérisation de cette classe particuliére.

On aura alors la correspondance suivante :

$$classe\, C_1 \;\; de \;\; formes \Leftrightarrow L(G_1)$$

et donc $\boxed{C_1 \Leftrightarrow L(G_1)}$

Une grammaire génére une classe et un mot inconnu ne pourra être considéré comme de la même classe que si c'est un mot généré par cette grammaire.

Dans ce formalisme n classes différentes seront donc en fait n langages $L(G_1)$, $L(G_2)$, ….$L(G_n)$ générés par les grammaires G_1, G_2, … G_n

On en déduit que la caractérisation d'une classe est alors une grammaire !!

Idéalement si chaque classe est définie par une grammaire connue et si une forme inconnue arrive et que l'on veut savoir à quelle classe l'affecter alors on va regarder quelle grammaire peut générer cette forme , on pourra alors décider à quelle classe elle sera affectée.

Formalisons ce probléme de décision et voyons les problémes posés ;

On suppose qu'on a n classes C_1, C_2, … C_n exhaustives ou non,
générées par n grammaires G_1, G_2, ….G_n

soit un mot x inconnu à classer dans une des classes

On regarde pour chaque grammaire G_i si x peut être engendré soit si $x \in L(G_i)$
Il s'agit là d'un probléme d'analyse syntaxique.

- **Si on a aucune solution alors x n'appartient à aucune des classes définies**
- **Si on a une réponse alors x est mis dans la classe correspondante**
- **Si on a plus de une réponse positive alors il y a indécision (recouvrement de classes)**

Quels sont les problémes posés par cette approche :

Si les classes sont exhaustives et que x n'est généré par aucune grammaire, on doit alors chercher à estimer le degré de ressemblance du mot avec chaque langage $L(G_i)$ Pour lui affecter une classe.
Si x peut être généré par plusieurs grammaires il est alors probable que les langages générés par les grammaires correspondantes ont trop d'intersections, c'est à dire que

chacune de ces grammaires n'est pas assez restrictive. Il est alors peut être possible de revoir le processus qui a généré ces grammaires (lors d'un apprentissage)

Le probléme de savoir si un mot appartient à un langage généré par une grammaire est un probléme d'analyse syntaxique. Nous abordons ce probléme dans la suite dans le cas particulier des grammaires réguliéres et donc des automates finis.

5.3.1 Analyse syntaxique et grammaires réguliéres

Nous allons traiter ce probléme avec l'automate fini correspondant à la grammaire réguliére. Notre probléme est donc le suivant :

Soit $x \in X^*$ et soit un automate **A**

Comment décider si x est un mot de **L(A)**

Notons $x = a_1 a_2 \ldots\ldots a_n$ avec $a_i \in X$

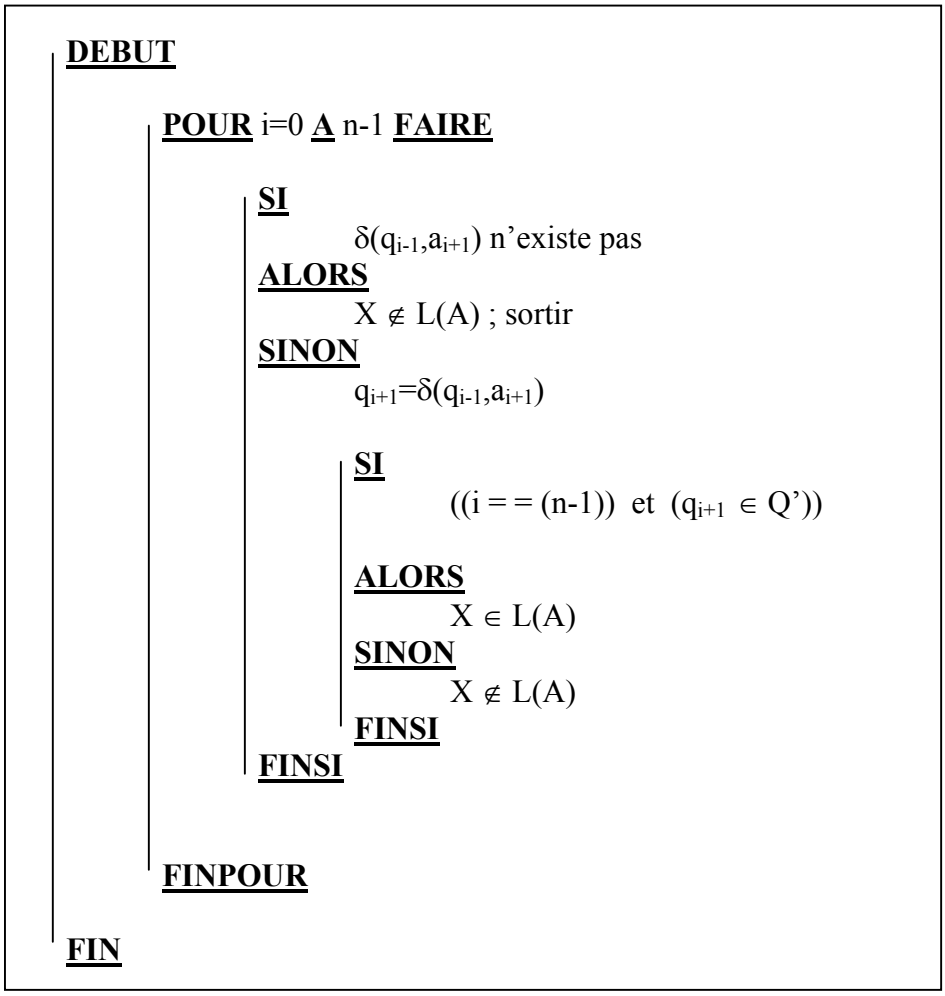
Et soit l'automate $\mathbf{A = (X , Q , \delta , q_0 , Q')}$

On suposera que cet automate A est complétement spécifié avec ses états poubelles

L'algorithme d'analyse syntaxique est alors tout simplement le suivant :

```
DEBUT
    POUR i=0 A n-1 FAIRE
        qi+1=δ(qi-1,ai+1)
    FINPOUR
    SI
        qn ∈ Q'
    ALORS
        x ∈ L(A)
    SINON
        x ∉ L(A)
    FINSI
FIN
```

Si l'automate n'est pas complétement spécifié alors l'algorithme pourrait s'écrire de la façon suivante :



Maintenant si pour un x donné on trouve qu'il ne fait pas partie des mots générés par l'automate A , nous allons essayer de voir si en modifiant le mot x par suppression, ou ajouts ou substitutions de lettres on peut arriver à trouver un mot qui pourrait être généré par A. La complexité plus ou moins importante des modifications donnera une « distance » de x à L(A).

5.3.2 Distance d'un mot à un langage régulier

Nous allons chercher le nombre minimal de transformations à faire subir à un mot x pour qu'il devienne un mot de L(A) et ces transformations fourniront un critére de ressemblance du mot avec le langage L(A).

Nous ne décrirons qu'un algorithme de ce type :

Algorithme de Wagner :

Cet algorithme consiste à calculer une distance d'un mot x à un langage L(A) en s'appuyant sur les transformations necessaires pour passer de x à un mot de L(A).

Cette distance s'exprime donc en un nombre minimal d'opérations à faire subir à un mot x (insertion, substitution, élimination de lettre) pour arriver à un mot généré par l'automate et de ce fait pour atteindre un état final de l'automate.

Nous noterons S T ou R un des états de l'automate A , en supposant que ces états sont q_0 q_1 q_2….q_m et on supposera que q_0 est l'état initial.

Soit le mot $x = a_1a_2$ …..a_n avec $a_i \in X$

Définissons :

F(j,S) j=1….n nombre minimum d'opérations transformant $a_1a_2…a_j$ en un mot qui serait généré par chemin dans l'automate allant de l'etat initial à l'état S

V(T,S,a_j) coût pour transformer aj en unje chaine générée par un chemin de l'état T à l'état S

Alors on a

$$F(j,S) = \min_T \{F(j-1,T) + V(T,S,a_j)\}$$

Voyons maintenant comment évaluer V(T,S,a_j)

Soit,

P(T,S) longueur d'un des plus courts chemins (en nombre d'arcs) entre T et S

et

L(T,S,a) valant 1 si la lettre a appartient à l'un des ces chemins sinon valant 0

alors

```
SI
        (T = = S)
ALORS
        V(T,S,a) = 1
SINON
        V(T,S,a) = P(T,S)-L(T,S,a)
FINSI
```

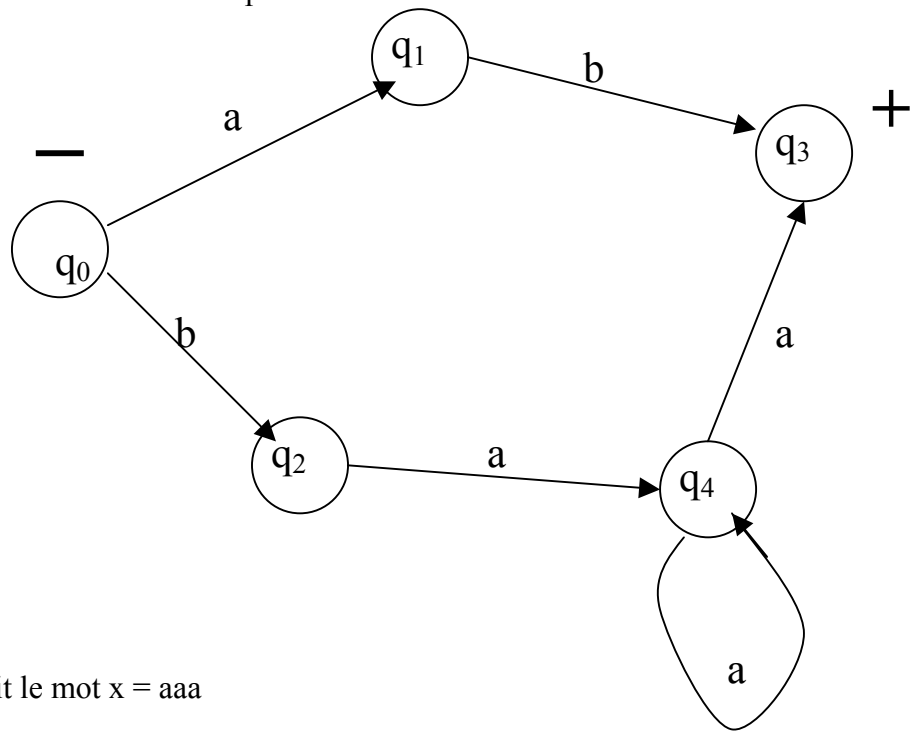
Enfin

Si Q' représente l'ensemble des états finals

Alors

$$d(x, L(A)) = \min_{S \in Q'} \{F(n, S)\}$$

Prenons un exemple :



Et soit le mot x = aaa

Quelle la distance de x à L(A) ?

Sous mot	F(j,q_0)	F(j,q_1)	F(j,q_2)	F(j,q_3)	F(j,q_4)
j=1 a	1	0	1	1	1
j=2 aa	2	1	2	1	1
j=3 aaa	3	2	3	1	1

Et donc **d(x,L(A)) = 1**

5.3.3 Apprentissage : inférence grammaticale

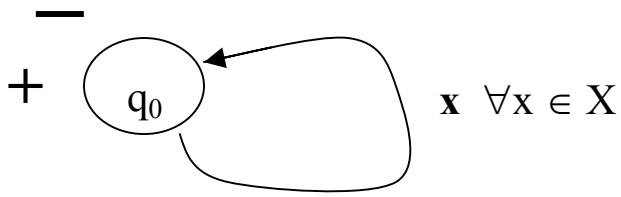
Maintenant nous allons supposer que les formes que l'on gére sont en fait des mots d'un langage particulier et nous supposons que les classes sont en fait définies par des échantillons (et donc des mots « échantillons »).

Chaque classe est donc définie par un petit paquet de mots. C'est à partir de ce paquet de mots qu'il faut trouver une technique de discrimination. Ici il s'agit de découvrir quelle est la grammaire réguliére (ou l'automate fini, c'est la même chose) qui généré la classe des échantillons.

Notons I l'ensemble des échantillons définissant une classe alors il faut trouver l'automate ou la grammaire qui génére ces mots.

Il y a bien sûr une infinité de solutions

Notamment on peut choisir la grammaire générant X* , c'est à dire tous lmes mots sur l'alphabet terminal X. Cela inclut bien sûr les échantillons de I ….mais cela ne sera pas du tout discriminant par rapport aux autres classes.



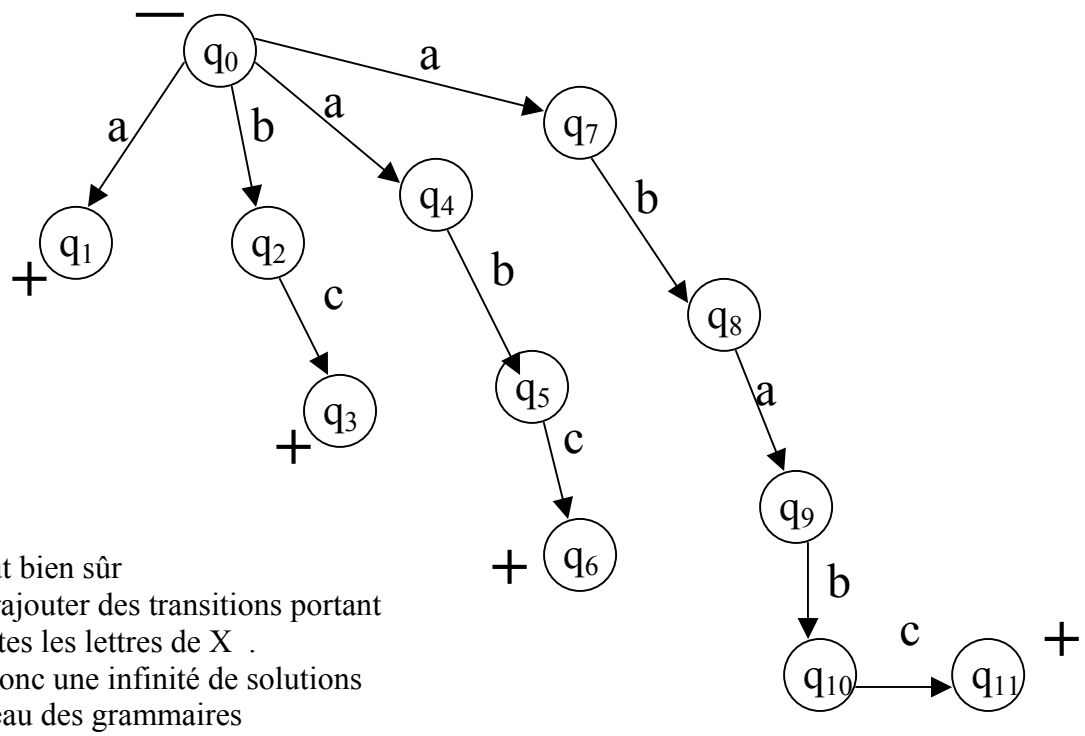
et c'est là tout le probléme…comment trouver un automate pour une classe de façon à ce que le langage généré se recoupe le moins possible avec ceux des autres classes.

Une possibilité (mais pas la meilleure c'est évident) est de choisir la grammaire canonique maximale qui ne généré que les échan,tillons de I.

Voyons cela sur un exemple :

Soit I= {a, bc, abc, ababc}

A lors la grammaire canonique maximale GCN(I) serait représentée par l'automate suivant :



On peut bien sûr
Aussi rajouter des transitions portant sur toutes les lettres de X .
Il y a donc une infinité de solutions au niveau des grammaires

Pour éviter cette infinité de solutions par ajout de transitions sans grand intérêt on va imposer une condition sur l'ensemble I des échantillons , c'est que cet ensemble d'échantillons soit complet par rapport à la grammaire G =(X,V,P,S)

Un ensemble d'échantillons I est complet par rapport à la grammaire G =(X,V,P,S) si et seulement si

1. $I \subset L(G)$
2. l'alphabet d'écriture de I est X
3. Toutes les régles de P sont utilisées au moins une fois dans une génération des mots de I

Le probléme de l'inférence grammaticale se pose alors maintenant comme suit :

Pour un ensemble I d'échantillons définissant une classe il faut trouver la grammaire réguliére G telle que,

1. **$I \subset L(G)$**
2. **I est complet par rapport à G**

exemple algorithme d'inférence réguliére :

L'algorithme u v^k w :

Le principe de cet algorithme est de rechercher dans l'ensemble de mots échantillons I , des traces de régles récursivces caractéristiques de la grammaire qui a engendré cet ensemble.

Nous allons expliquer cet algorithme sur la base d'un exemple concret :

$$\text{Soit} \quad I = \begin{cases} x_1 = aabaaababcabc \\ x_2 = abcabaabcbc \\ x_3 = aaaaabc \end{cases}$$

Etape 1 « recherche des répétitions »

$x_1 = (a)^2baaababcabc$
$x_1 = aab(a)^3babcabc$
$x_1 = aabaa(ab)^2cabc$
$x_1 = aabaaab(abc)^2$
$x_2 = abcab(a)^2bcbc$
$x_2 = abcabaa(bc)^2$
$x_3 = (a)^5bc$

Les régles récursives éligibles sont donc celles qui concernent a, ab, abc, bc

On choisit évidemment ici la régle recursive sur **a** puisque c'est celle qui est la plus fréquente

En considérant que z est n'importe quel mot non vide composé de séquence de la lettre **a**, c'est à dire $\mathbf{z \in \{a\}^+}$

L'ensemble I des échantillons peut alors être reécrit en :

$$I = \begin{cases} x_1 = zbzbzbczbc \\ x_2 = zbczbzbcbc \\ x_3 = zbc \end{cases}$$

Etape 2 « recherche des répétitions »

$x_1 = (zb)^3czbc$
$x_1 = zbzb(zbc)^2$
$x_2 = zbc(zb)^2cbc$
$x_2 = zbczbz(bc)^2$

Les régles récursives éligibles sont donc celles qui concernent zb, zbc,bc
On choisit bien sûr zb

On peut donc reécrire l'ensemble I pour $\mathbf{y \in \{zb\}^+}$

$$I = \begin{cases} x_1 = ycyc \\ x_2 = ycycbc \\ x_3 = yc \end{cases}$$

Etape 3 « recherche des répétitions »

$x_1 = (yc)^2$
$x_2 = (yc)^2bc$
$x_3 = yc$

Une seule régle recursive eligible: yc

On peut donc reécrire l'ensemble I pour $\mathbf{x \in \{yc\}^+}$

$x_1 = x$
$x_2 = xbc$
$x_3 = x$

Il est alors evident qu'on peut terminer en considérant

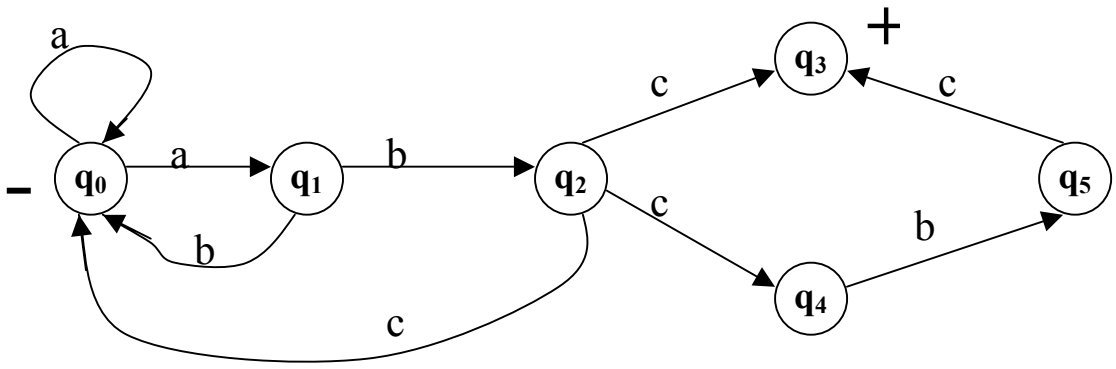
$\mathbf{\{x\}^+}$ **et** $\mathbf{\{x\}^+bc}$

On peut en déduire l'expression réguliére inférée permettant de représentant sous une forme simplifiée l'ensemble des mots et donc le langage obtenu à parti des échantillons de I , on en déduira la forme de l'automate fini associé.

$\mathbf{x^+ + x^+bc}$
soit $\mathbf{(yc)^+ + (yc)^+bc}$
soit $\mathbf{((zb)^+c)^+ + ((zb)^+c)^+bc}$
soit $\mathbf{((a^+b)^+c)^+ + ((a^+b)^+c)^+bc}$

Expression réguliére inférée : $\mathbf{((a^{+}b)^{+}c)^{+} + ((a^{+}b)^{+}c)^{+}bc}$

Automate fini associé :



Cette méthode est bien adaptée aux formes (mots) qui ont des structures répétitives.

5.4 GRAMMAIRES ET AUTOMATES FINIS STOCHASTIQUES

Certains mots dans une classe peuvent être plus probables que d'autres, ou bien certains types de mots peuvent être plus courants…On peut aussi considérer que si un mot donné peut être généré par plusieurs grammaires, il peut se produire qu'une grammaire soit plus plausible que d'autres pour générer ce mot.

Supposons donc avoir n classes de mots définies par n grammaires génératrices $G_1, G_2 \ldots G_n$.

En raisonnant dans le cadre d'une approche bayesienne et pour un mot x inconnu à classer , on devra donc être capable de calculer $P(G_i / x)\ \forall i$.

$P(G_i / x)$ est la probabilité que G_i soit la grammaire qui génére x.

La décision s'obtiendra donc ainsi $x \in$ classe Ci / $P(G_i / x) > P(G_j / x)\ \forall i \mathrel{!}= j$

Pour appliquer cette décision bayésienne, il faut que les classes soient exhaustives et il faut connaître

$P(G_i)\ \forall i$	probabilité à priori de la classe Ci
$P(x/G_i)\ \forall i$	probabilité que x soit engendré par Gi
$P(x) = \sum_{i=1}^{n} P(x/G_i) \times P(G_i)$	probabilité du mot x

5.4.1 Grammaires stochastiques

Nous ne donnerons ici que quelques définitions de base permettant de comprendre cette approche.

Nous nous placerons dans le cas de figure simplificateur où la probabilité d'application d'une régle de production de la grammaire est indépendante de la séquence de régles appliquées précedemment.

Par rapport à une définition classique d'une grammaire voici la définition qu'on peut avoir pour une grammaire stochastique G_s:

G_s = (X , V , S , P)

- **X** : alphabet terminal (les "lettres")
- **V** : alphabet auxiliaire (ou non terminal)
- **S** : élément particulier de V appelé axiome permettant comme son nom l'indique le démarrage de la génération de mots
- **P** : ensemble de régles de production qui permettront le déroulement du processus de génération de mots

Mais une régle de production peut être définie par le passage d'une chaine α non vide formée de lettres de l'alphabet terminal et d'éléments de l'alphabet auxiliaire à une nouvelle chaine β du même type pouvant être vide. Cette régle ayant une probabilité p de se produire

$$\alpha \xrightarrow{p} \beta$$

avec $\alpha \in (V \cup X)^* \times V \times (V \cup X)^*$

et $\beta \in (V \cup X)^*$

Une telle grammaire stochastique peut en plus être réguliére si et seulement si on a :

avec $\alpha \in V$

et $\beta \in X.V \text{ ou à } X$

Détaillons comment on peut procéder au calcul de $P(x/G_s)$:

Soit donc x un mot généré par la grammaire G_s à la suite de l'application des régles de production P_1 P_2 …. P_m et supposons que ces régles ont des probabilités p_1 p_2 … p_m

Notons $$q(x) = \prod_{i=1}^{m} p_i$$

Si on a une grammaire ambiguë , cela signifie qu'il y a plusieurs possibilités pour générer x (et chaque possibilité fournit évidemment une valeur $q_j(x)$)

S'il y a nx possibilités au total alors on a :

$$p(x / G_s) = \sum_{j=1}^{n_x} q_j(x)$$

Si on a une grammaire non ambiguë alors

$$p(x / G_s) = q(x)$$

5.4.1.1 Quelques compléments sur les grammaires stochastiques

Nous allons donner ici quelques définitions complétant les éléments déjà donnés sur les grammaires stochastiques :

Grammaire caractéristique d'une grammaire stochastique G_s

C'est en fait la grammaire G en enlevant les probabilités affectées aux régles de production.

Définition du langage stochastique généré par G_s :

$L(Gs) = \{ (x,p(x)) / x \in L(G)\}$

Définition d'une grammaire stochastique propre :

Si la somme des « probabilités » associées aux régles ayant même partie gauche est égale à 1

Définition d'une grammaire stochastique consistante :

Si $\sum_{x \in L(G)} p(x) = 1$

Exemple :

Gs = (V, X, P, S)

X= {a, b} **V= {S}** $P : \begin{cases} S \xrightarrow{p} aSb \\ S \xrightarrow{1-p} ab \end{cases}$

$L(G_s)$ = { ($a^n b^n$, $p^{n-1}(1-p)$) / x ∈ L(G) et n >=1 }

On remarque que Gs est non ambiguë et consistante

Car $\lim_{n\to+\infty}\sum_{n=1}^{N} p^{n-1}(1-p)=1$

5.4.2 Automates finis stochastiques

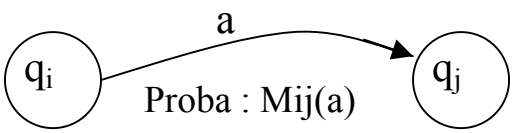
Un automate fini stochastique correspondant à une grammaire réguliére stochastique est défini de la façon suivante :

A_s = (X , Q , M , Π_0 , F)

Avec **X : alphabet terminal**
Q : ensemble de n états $q_1, q_2, \dots, q_n$
M : application de X dans l'ensemble des {matrices stochastiques n x n }
Π_0 : vecteur ligne de dimension n représentant la distribution initiale (c'est à dire les états initiaux)
Π_F : vectur colonne reprsentant l' ensemble des états finals

Une matrice stochastique est une matrice carrée dont tous les éléments sont positifs ou nuls et telle que la somme des éléments de chaque ligne vaut 1.

En fait une transition de qi à qj à l'aide de la letttre a par exemple aura une probabilité Mij (a). Cette probabilité est l'élément ligne i colonne j de la matrice stochastique M(a).



On peut étendre la définition de M à X^* de la façon suivante :

Si $a_1, a_2, \dots, a_n$ sont des lettres de X alors pour $a_1a_2\dots a_n \in X^*$ on définit $M(a_1a_2\dots a_n) = M(a_1) \times M(a_2) \times \dots \times M(a_n)$

Pour le mot vide λ de X^*on définit $M(\lambda) = I_n = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

Le langage stochastique généré est alors défini par :

$$L(A_S) = \{(x, p(x)) / \, x \in X^* \;\; et \;\; p(x) = \Pi_0 \times M(x) \times \Pi_F \geq 0\}$$

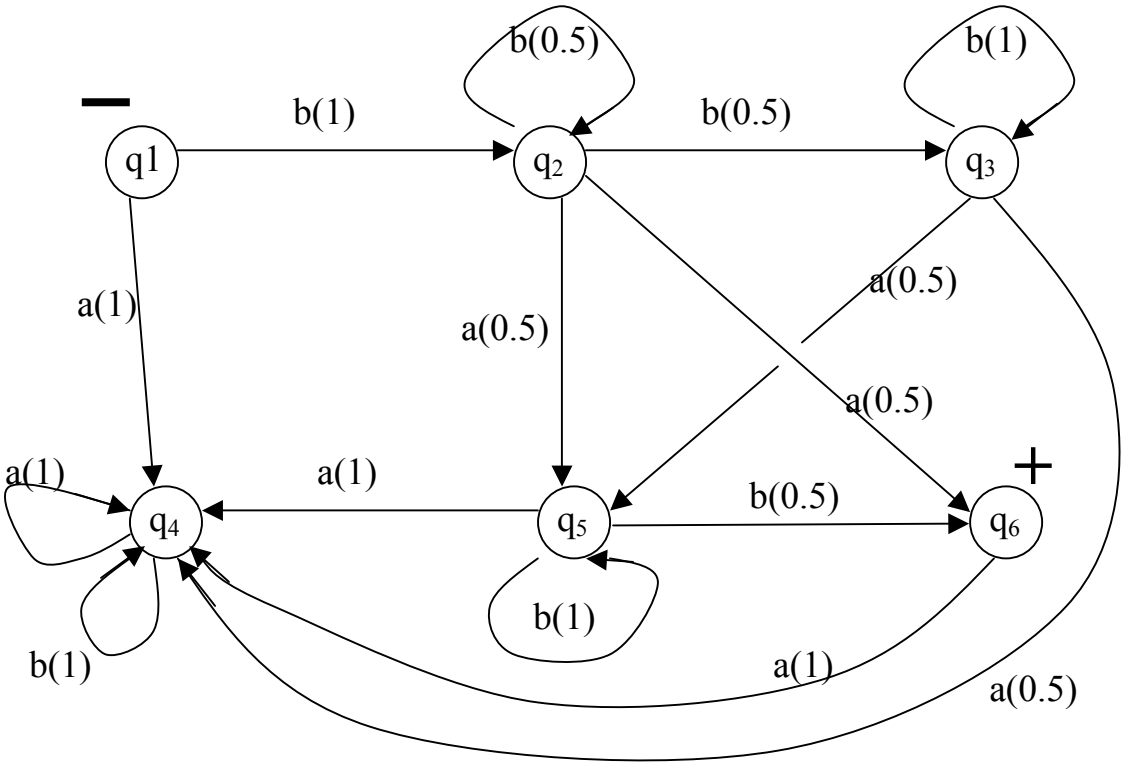
exemple :

$$\mathbf{A_s = (X , Q , M , \Pi_0 , \Pi_F)}$$

Avec **X : {a,b}**
Q : ensemble de 6 états $q_1, q_2, \ldots, q_6$
Π_0 : (1 0 0 0 0 0)
Π_F : (0 0 0 0 1 0)^t

$$M(a) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad M(b) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

L'automate stochastique complet correspondant est le suivant :



Les mots générés sont les suivants :

$$x = b^m a b^n \text{ avec } m > 0 \text{ et } n > 0$$

$$\text{avec } p(x) = 0.5 + 0.5^m - 0.5^n$$

Le langage généré est donc

$$L(A_S) = \{ (x = b^m a b^n , p(x)) / m > 0 \text{ et } n > 0 \}$$

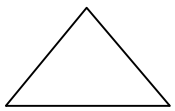
5.5 METHODES STRUCTURELLES POUR LA RECONNAISSANCE DES FORMES

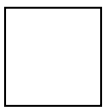
5.5.1 Généralités

Nous allons aborder ce probléme par un exemple :

Supposons que nous voulons distinguer les deux classes de formes suivantes :

Classe des triangles équilatéraux et classe des carrés.
Ces formes ayant une base horizontale , une taille quelconque et une position quelconque dans le plan.





Si l'on décide de choisir une approche bayésienne classique alors il faut définir n mesures (par exemple surface/périmétre, surface….) et dans ce cas chaque forme sera un vecteur de R^n et on peut alors appliquer les diverses approches vues au début de ce cours.

On peut aussi choisir une approche « structurelle » basée sur une description de la forme des objets traités. Ainsi ici notre triangle particulier pourrait ^tre décrit par :

« Un segment horizontal de longueur l suivi (sens trigonométrique direct) d'un segment oblique montant de longueur l, suivi d'un segment oblique descendant de longueur l qui rejoint le segment initial »

Dans cette représentation on considére :

- Un ensemble de formes élémentaires (primitives)
  Ici : nos divers segments
- Une description de l'assemblage de ces primitives
  Ici : la suite ordonnée de segments

Dans notre espace de représentation une forme sera décrite par une suite de segments relevés le long du contour de l'objet considéré, l'ordre pour ces segments étant fondamental.

Dans le cadre de cette approche structurelle nous serons confrontés aux problémes habituels en reconnaissance des formes.

Si la classification est supervisée chaque classe est définie par un ensemble d'échantillons et donc de mots et il y aura donc une phase d'apprentissage (permettant d'obtenir un automate régulier par exemple). Puis pour une forme inconnue (un mot x) il faut décider si la forme appartient ou non à lma classe (par exemple verifier si le mot peut être généré par l'automate trouvé lors de la phase d'apprentissage).

L'approche apprentissage donnant une grammaire générant un langage est en fait très sensible au « bruit » (par exemple une lettre remplacée par une autre) lors de la phase de décision.

Nous n'allons donc pas approfondir plus cette voie mais nous allons plutot nous orienter vers des mesures de ressemblance entre formes pour la décision.

5.5.2 Ressemblance entre mots

Nous partons donc du point de vue qu'une forme est représentée par une suite de symboles qu'on peut considérer comme les lettres d'un alphabet terminal. Une forme est donc considérée comme un mot écrit avec un alphabet particulier.

Nous allons alors aborder la phase de décision de classification à l'aide d'une mesure de ressemblance entre mots écrits sur le même alphabet.

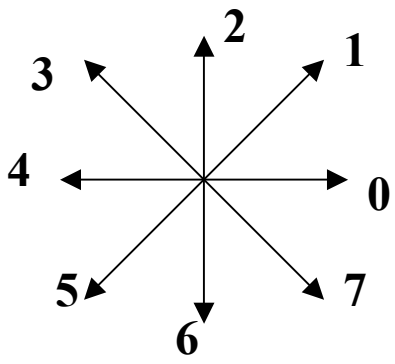
- Alphabet : nombre fini de composants élémentaires (lettres)
- Mot x : suite ordonnée de lettres
- Le mot vide est noté λ
- $|x|$ nombre de lettres de x
- X^* ensemnble des mots possibles dont λ
- X^+ ensemble des mots non vides
- Concatenation $x=x_1x_2\ldots x_n$ et $y=y_1y_2\ldots y_m$
  $xy = x_1x_2\ldots x_n\, y_1y_2\ldots y_m$
- Sous mot : $y \in X^*$ est un sous mot de $x \in X^*$
  Ssi $\exists$ u et $v \in X^*$ / x ==uyv (u :préfixe, v suffixe)

Exemple :

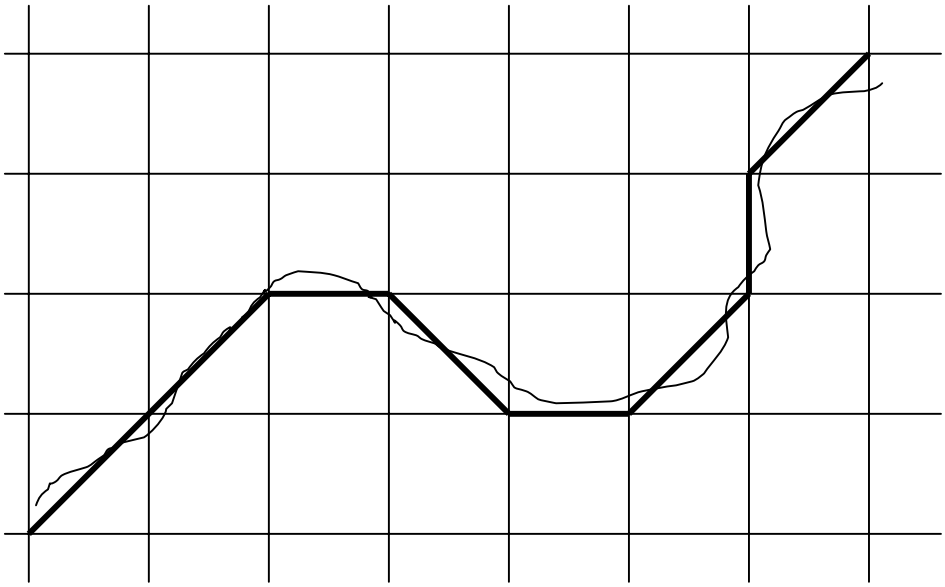
Représentation d'un contour par codage de Freeman dans une image

Un contour d'objet va être caractérisé par une suite de vecteurs élémentaires appelés codes de Freeman

Codage de Freeman :



Forme dans une image :



L'objet est ici codé x = 11070121 dans l'alphabet { 0,1,2,3,4,5,6,7}

Si une classe d'objets est représentée par un ensemble d'échantillons , nptre méthode va consister pour une forme inconnue x à mesurer sa ressemblance avec l'ensemble d'échantillons et donc il faut être capable de mesurer la ressemblance ou la distance entre x et un objet quelconque.

Cette ressemblance estime donc la « distance » entre deux chaines.

Un premier aspect simple peut ëtre de chercher si une chaine est sous chaine d'une autre, nous verrons d'abord ce probléme classique en édition de textes

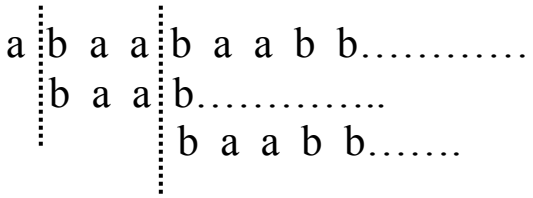
Puis nous estimerons la distance entre deux chaines en estimant la quantité de transformations à faire subir à une chaine pour atteindre l'autre

5.5.2.1 Inclusion de chaines

Soient deux chaines écrites avec le même alphabet,
y de longueur l et x de longueur n
comment savoir si y est inclus dans x.

Nous allons donner sommairement sur un exemple la description d'un algorithme classique, celui de Morris et Pratt :

Soit x = abaabaabb……..
et y = baabb……..



5.5.2.2 Distance entre chaines

L'idée est d'estimer la distance entre deux chaines en prenant en compte les opérations (substitution, suppression,ajout) pour passer d'une chaine à l'autre. Chacune de ces opérations peut être définie par un coût. La distance est en fait le coût de la suite d'opération de coût minimum pour passer d'une chaine à l'autre.

Si l'on note λ le mot vide alors,

Opérations élémentaires	**coût**
Substitution $a \to b$	C(a,b)
Insertion $\lambda \to a$	C(λ,a)
Destruction $a \to \lambda$	C(a,λ)

Exemple de chemin (non minimal) passant de ab à bac :

$$ab \to \boxed{\lambda \to c \qquad b \to \lambda \qquad c \to b \qquad \lambda \to c} \to bac$$

cab ca ba bac

Le coût de ces transformations est donc : **C(λ,c)+ C(b,λ)+ C(c,b)+ C(λ,c)**

La distance entre chaînes que nous venons de suggérer est en fait une distance d' « édition » dite de Levenstein

D_L(x,y) : coût de la suite d'opérations la moins coûteuse permettant de passer deu mot **x** au mot **y**.

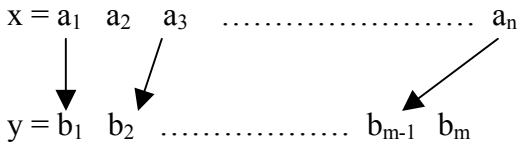
La condition imposée sur la fonction de coût est que C(,) soit une distance, car si a et b sont des lettres alors DL(a,b) = C(a,b).

On doit donc avoir, si a b, c sont des lettres

C(a,b) = = C(b,a)
C(a,a) = = 0
C(a,b)+C(b,c) >= C(a,c)

Défintion d'une TRACE:

Le passage d'un mot à un autre peut être représenté par une trace qui matérialise toutes les opérations réalisées.
Voici un exemple :



Les lettres reliées correspondent à des substitutions
Les lettres non reliées correspondent à des suppressions ou des ajouts

TRACE : $(a_1 \to b_1)\ (a_2 \to \lambda)\ (a_3 \to b_2)$ .....$(a_n \to b_{m-1})\ (\lambda \to b_m)$

La distance entre deux mots est aussi donc le coût d'une trace de coût minimal.

5.5.2.2.1 Algorithme de calcul de la distance de Levenstein

Soient **$x = a_1a_2a_3$ ……a_n**
et **$y = b_1b_2$ …… $b_{m-1}\,b_m$**

et notons **$x(i) = a_1a_2$ …a_i**
et **$D(i,j) = D_L(x(i),y(j))$**

alors,

$$D(i,j) = \min\begin{cases} D(i-1,j-1) + C(a_i,b_j) \\ D(i-1,j) + C(a_i,\lambda) \\ D(i,j-1) + C(\lambda,b_j) \end{cases}$$

finalement,

DL(x,y) =D(n,m)

ALGORITHME de CALCUL :

```
DEBUT
        D(0,0) = 0 ;

        POUR   i =1   à   n
               D(i,0) = D(i-1,0)+C(a_i,λ);
        FINPOUR

        POUR   j =1   à   m
               D(0,j) = D(0,j-1)+C(λ,b_j);
        FINPOUR

        POUR   i =1   à   n
        POUR   j =1   à   m
               M1  = D(i-1,j-1)+C(a_i,b_j);
               M2  = D(i-1,j)+C(a_i,λ);
               M3  = D(i,j-1)+C(λ,b_j);
               D(i,j) = min ( M1 , M2 , M3 );

        FINPOUR
        FINPOUR

        D_L (x,y) =D(n,m) ;
FIN
```

Exemple :

X= { a , b , c , d }

x = aabacd et et   y = abd

$C(\alpha,\alpha) = = 0$ pour tout α de X

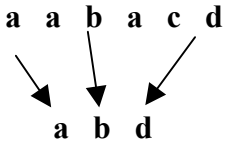
$C(\alpha,\lambda) = = C(\lambda,\alpha) = = C(\alpha,\beta)$ pour tout α de X et pour tout α et β de X avec $\alpha \neq \beta$

Le calcul peut être résumé dans le tableau suivant :

D(i,j)	i=0 λ	i=1 a	i=2 a	i=3 b	i=4 a	i=5 c	i=6 d
j=0 λ	0	1	2	3	4	5	6
j=1 a	1	0	1	2	3	4	5
j=2 b	2	1	1	1	2	3	4
j=3 d	3	2	2	2	2	3	3

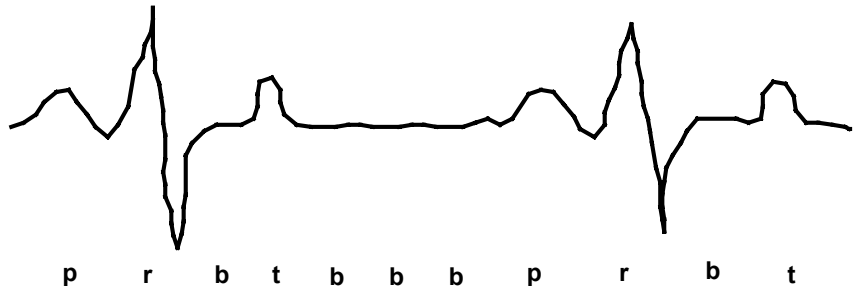
$$\mathbf{D_L(x,y) = D(3,6) = 3}$$

Trace correspondante :



5.6 EXERCICES

Soit le signal suivant :



En supposant disposer de l'alphabet **X = { p,r,b,t }**
et que les formes de signal acceptables appartiennent au langage
L = (prbt (b+bb+bbb))*

1) Donner la grammaire **G = { X,V,S,P }** générant le langage **L .**
2) Décrire l'automate non complet correspondant (on suppose qu'un signal étudié commence obligatoirement avec **p** mais qu'il peut être de longueur quelconque. D'autre part un état particulier d'un automate peut-être à la fois initial et final).

3) En représentant un automate complet avec ses états poubelle, décrire une technique pour déterminer si un signal est normal ou pas.

So*lution possible :*

1) X = { p , r , b , t }

V= { S , A , B , C , D , E , F }

$$P \begin{cases} S \to pA \\ A \to rB \\ B \to bC \\ C \to tD \\ D \to bE/b \\ E \to pA/bF/b \\ F \to pA/bS/b \end{cases}$$

2)

Soient les échantillons suivants décrivant une classe de formes :

X1 = aabcabcababaaa
X2 = aaaaccbbabc
X3 = abcaaab
X4 = aaaaab

Donner l'expression réguliére inférée par cet ensemble d'échantillons et l'automate associé.

Donner la grammaire G={X,V,S,P} correspondante.

Donner une méthode permettant de savoir si une suite de caractéres peut être générée par l'automate inféré par les échantillons.

On suppose qu'une classe est définie par les échantillons suivants:

X1=aabaaababcabc
X2=abcabaabcbc
X3=aaaaabc

Donner l'automate généré par l'algorithme uvkw
Donner la grammaire G={X,V,S,P} correspondante.

Soient les échantillons suivants décrivant une classe de formes :

X1 = abcabcabcaaa X3 = aaaababab
X2 = aaaaabcaa X4 = aaaaab

Donner l'expression réguliére inférée par cet ensemble d'échantillons et l'automate associé.

Donner la grammaire G={X,V,S,P} correspondante.

Donner une méthode permettant de savoir si une suite de caractéres peut être générée par l'automate inféré par les échantillons.

Soit un signal de type plateau supposé décrit par une chaine de symboles
(M: montée, P: plateau , D: descente) . Chaque symbole correspondant à une durée constante.

1) Représenter l'automate permettant de reconnaitre un plateau de hauteur 1 (une montée, un plateau de largeur quelconque, une descente) et donner l'expression réguliére correspondante.

2) Donner l'automate permettant de représenter une suite de plateaux de hauteur 1 (on suppose qu'une descente est séparée d'une montée par au moins un plat).

3) Donner la grammaire réguliére permettant de génèrer les signaux de la question 2.

4) Comment concevoir un automate capable de reconnaitre un plateau de hauteur quelconque (nombre de montées égal au nombre de descentes)

Soient les échantillons suivants décrivant une classe de formes :

X1 = ababaaabcc
X2 = aabccaabc
X3 = aabc
X4 = aaaabbbab

Donner l'expression réguliére inférée par cet ensemble d'échantillons et l'automate associé en utilisant la méthode uv^kw.

Donner la grammaire G={X,V,S,P} correspondante.

Donner une méthode permettant de savoir si une suite de caractéres peut être générée par l'automate inféré par les échantillons.

On suppose qu'une classe est définie par les échantillons suivants:

X1=aabaaababcabc
X2=abcabaabcbc
X3=aaaaabc

Donner l'automate généré par l'algorithme uv^kw
Donner la grammaire G={X,V,S,P} correspondante.

---

6 REFERENCES

1. **Pattern classification and scene analysis**
   Richard O. DUDA , Peter E. HART
   Ed; John Wiley & sons - 1973

2. **Méthodes structurelles pour la reconnaissance des formes**
   Laurent MICLET - Ed. Eyrolles
   collection technique et scientifique des télécommunications 1984

3. **Pattern recognition principles**
   Julius T. TOU , Rafael C. GONZALEZ
   Addison-wesley publishing company - 1974

4. **Exercices de reconnaissance des formes par ordinateur**
   Philippe FABRE
   Ed. Masson 1989

5. **Neural networks – a systematic introduction**
   Raul ROJAS
   Ed; Springer - 1996

6. **Les réseaux de neurones**
   Jean-François JODOUIN
   Ed. Hermés - 1994

7. **Introduction to statistical pattern recognition**
   Keinosuke FUKUNAGA
   Academlic press - 1972