Table des Matières

Chapitre 1

INTRODUCTION

La classification et la reconnaissance des formes sont des disciplines qui sont pratiquement nées, ou en tout cas qui ont pris leur vraie signification, avec le développement des ordinateurs. Elles traitent en effet de l'automatisation par ce moyen d'un certain nombre de tâches de perception artificielle réalisées usuellement par le cerveau humain. Dans cette tâche, il ne s'agira pas forcément d'imiter le fonctionnement de l'opérateur humain, mais plutôt de définir certaines méthodes permettant l'aide au diagnostic. Un grand nombre d'exemples, pour lesquels on est amené à utiliser certaines des multiples techniques de classification, sont disponibles dans la littérature. Nous en citerons deux qui ont été étudiés en détail par le groupe de traitement des signaux, des images et des communications du laboratoire d'Electronique de l'ENSEEIHT.

1 Exemples

1.1 Biomédical

Les techniques d'électrophysiologie sont développées et utilisées depuis plusieurs dizaines d'année, ceci dans des domaines aussi variés que l'électroencéphalographie, l'électrocardiographie, ... L'usage de l'électromyographie (EMG) en tant qu'indicateur de l'état du système neuromusculaire remonte à la fin du XIX^{ème} siècle. Cette technique a maintenant acquis une part importante dans la détection et le suivi d'atteintes des systèmes musculaires. Elle consiste à analyser le comportement du muscle lors d'une stimulation électrique, ou à observer son activité électrique lors d'une contraction naturelle. L'accès à l'activité électrique musculaire est réalisée au moyen de deux grandes catégories d'électrodes :

- les électrodes piquées dans la masse musculaire

- les électrodes de surface

L'inspection visuelle du signal recueilli à l'aide de la première catégorie d'électrodes permet de caractériser la pathologie neuromusculaire mais cette technique est très douloureuse pour le patient. Les électrodes en surface, quant à elles, ne permettent pas de distinguer les différentes pathologies avec une précision suffisante et donc il s'avère nécessaire de développer des outils de reconnaissance des formes permettant cette distinction [31]. La figure 1.1 montre une représentation de signaux électromyographiques (EMG) obtenues à l'aide d'électrodes piquées dans la masse musculaire (à gauche), d'électrodes de surface (à droite), pour des muscles sains (figures du haut) ou de pathologie 3A (figures du bas).

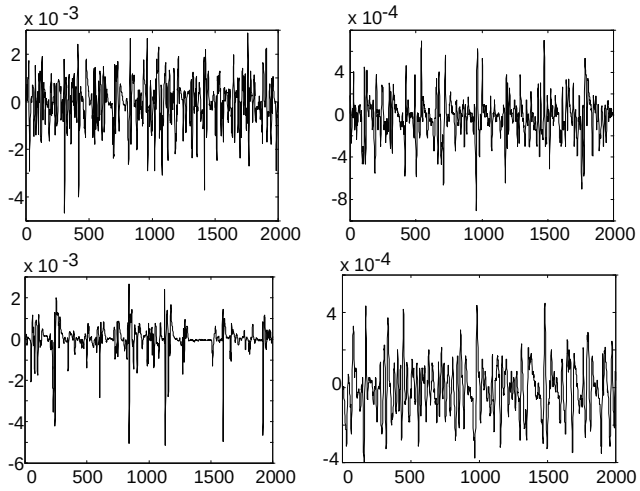


Fig. 1.1 Représentation Temporelle de Signaux Electromyographiques

1.2 Analyse des vibrations

On désire surveiller un réducteur à engrenages en détectant de façon précoce l'apparition d'un défaut qui peut être localisé sur une ou plusieurs dents. Divers types de défauts peuvent être pris en compte :

- les défauts localisés sur certaines dents de l'engrenage (écaillage, fissures, trous, ...)

- les défauts distribués sur toutes les dents de l'engrenage (usure uniforme, pitting, ...)

La distinction des signaux par simple observation de leur représentation temporelle ou spectrale n'est pas évidente :

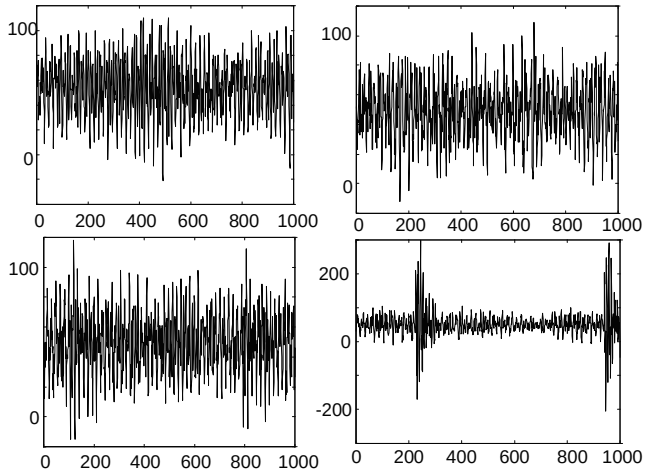


Fig. 1. 2 Représentation Temporelle de Signaux Vibratoires

Le but de l'étude est de développer des méthodes de surveillance et d'inspection spécifiques aux organes de transmission à engrenages à l'aide des techniques conventionnelles de reconnaissance des formes [22].

2 Chaîne de reconnaissance

Un système de reconnaissance des formes fait en général appel aux organes suivants:

- **un ensemble de capteurs** qui transmettent l'ensemble des informations sur les formes. Dans le premier exemple, les capteurs sont des électrodes qui convertissent l'information provenant des contractions musculaires en un signal appelé électromyographique. Dans le second exemple, les capteurs sont des accéléromètres qui transforment le signal vibratoire en un signal électrique.

- **une boite de prétraitement** dont le rôle est d'extraire de la masse souvent énorme des données transmises par les capteurs une série de paramètres pertinents pour le problème. Le prétraitement comporte la modélisation des signaux et la sélection des paramètres pertinents.

- **une boite d'apprentissage** qui permet de définir les techniques de décision. Cette étape se fonde sur l'expertise des signaux.

- **une boite de décision** dont le rôle est de situer la forme captée dans les classes de formes obtenues après la phase d'apprentissage et donc de fournir une reconnaissance. Le schéma général d'une chaîne de reconnaissance est donc le suivant :

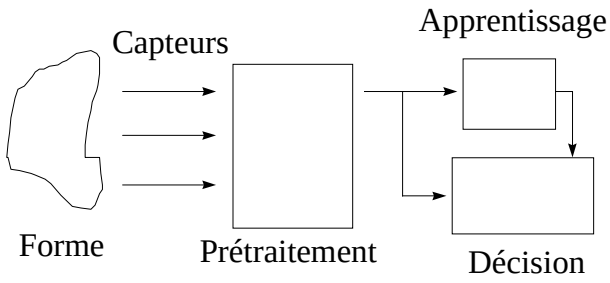


Fig. 1.3 Chaîne de Reconnaissance

Chapitre 2

PRETRAITEMENT

A la sortie des capteurs, on a des signaux analogiques que l'on désire répartir en différentes classes (i.e. classifier). Dans la plupart des applications, ces signaux analogiques sont échantillonnés à l'aide d'un convertisseur analogique-numérique. Le signal numérique obtenu est un vecteur de dimension p, où p représente le nombre de points ou variables de ce signal. On peut considérer ce vecteur paramètre comme la réalisation d'un vecteur aléatoire dans un espace à p dimensions. En analyse de données, on regroupe généralement ces divers paramètres dans un tableau. Un exemple d'un tel tableau est celui du poids (en kg), de la taille (en m), de l'age (en années) et de la note moyenne dans l'année (note sur 20) de dix individus [11]. Ce type de tableau est très fréquent dans toutes les sciences exactes où on peut effectuer une série de mesures physiques sur chaque individu :

	Poids	Taille	Age	Note
X_1	45	1.50	13	14
X_2	50	1.60	13	16
X_3	50	1.65	13	15
X_4	60	1.70	15	9
X_5	60	1.70	14	10
X_6	60	1.70	14	7
X_7	70	1.60	14	8
X_8	65	1.60	13	13
X_9	60	1.55	15	17
X_{10}	65	1.70	14	11

Dans l'exemple ci-dessus, on a $n = 10$ individus avec $p = 4$ variables et on note le $i^{ème}$individu

$$X_i = (X_i(1), ..., X_i(4))^t \qquad i \in \{1, ..., 10\} \tag{2.1}$$

En traitement du signal ou des images, les signaux ou images à classifier (individus) sont souvent constitués d'un nombre important de points (variables). De nombreux prétraitements ont été proposés dans la littérature pour réduire ce nombre de variables. Un des avantages de travailler avec un nombre réduit de variables est d'obtenir des règles de classification simples, rapides (avec une possible implantation temps-réel) et nécessitant peu de mémoire. Mais elle peut aussi réduire le problème connu sous le nom de "curse of dimensionaliy". Comme nous le verrons lorsque nous étudierons la règle de décision Bayésienne, le classifieur optimal (qui minimise par exemple la probabilité d'erreur) nécessite des connaissances a priori sur la loi des observations conditionnellement à chaque classe et les probabilités des différentes classes. Lorsque ces informations a priori ne sont pas disponibles, la plupart des règles de classification cherchent à estimer ces informations à partir d'un ensemble de signaux expertisés appelés *signaux d'apprentissage*. Lorsque le nombre de ces signaux d'apprentissage est important, les performances des classifieurs peuvent être très proches de la performance optimale du classifieur Bayésien et en général la probabilité d'erreur du classifieur diminue lorque le nombre de variables augmente. En revanche, plusieurs auteurs [33] [16], ont montré que ce résultat pouvait ne plus s'appliquer lorque le nombre de variables était trop important par rapport au nombre de vecteurs d'apprentissage. Les prétraitements permettant de réduire le nombre de variables peuvent éviter les problèmes de surdimensionnement ("curse of doimensionality"). On peut regrouper ces prétraitements en deux grandes classes : **les méthodes de sélection de données** et **les méthodes d'extraction de données**. Les méthodes de sélection de données cherchent à déterminer un sous-ensemble des données de départ afin d'obtenir la meilleure classification (i.e. la probabilité d'erreur la plus faible). Les méthodes d'extraction de données cherchent à représenter les données de départ éléments de $\mathbb{R}^p$ dans un autre espace de dimension inférieure, c'est-à-dire la transformation Φ de $\mathbb{R}^p$ dans $\mathbb{R}^q$, avec $q < p$, conduisant à une probabilité d'erreur minimale. Dans le premier cas, le prétraitement conserve certaines données de départ (avec leur sens physique si elles en ont) tandis que dans le second cas, le prétraitement peut transformer les données de départ par combinaisons linéaires ou application non-linéaires. Dans cette partie, nous nous limitons à certaines méthodes d'extraction de données qui nous paraissent intéressantes pour la classification en traitement du signal et des images.

1 Modélisation et représentation des signaux temporels

La modélisation des signaux temporels permet d'extraire l'information utile contenue dans ces signaux, en limitant le volume des données brutes. Plusieurs méthodes de modélisation peuvent être utilisées. On peut par exemple considérer que le signal temporel est la sortie d'un système linéaire ou non-linéaire et on identifie les paramètres de ce système (modélisation paramétrique ARMA, modélisation à l'aide des séries de Volterra, ...). On peut aussi utiliser certains jeux de coefficients liés aux paramètres de ces modèles (comme les coefficients de réflection ou les coefficients cepstraux qui ont montré des propriétés intéressantes pour la classification [22]).

L'utilisation de transformées comme la transformée de Fourier, les transformées temps-fréquence (Wigner Ville) ou les transformées temps-échelle (Ondelettes) s'est avérée très intéressante pour la classification [2],[15]. Dans certaines applications, ces transformées permettent de définir une signature propre à chaque classe. Les paramètres utilisés pour la classification sont alors dérivés des coefficients issus de ces transformées.

Une autre façon de diminuer le nombre de variables consiste à détecter des événements précis dans le signal (passages à zéro, extrema, ...) ou utiliser des paramètres statistiques (moyenne, médiane, énergie, moments d'ordre supérieur, ...).

A l'issue de cette première phase de prétraitement, on dispose d'un tableau de données constitué des paramètres (de modélisations, de transformées, ...) correspondant à chaque individu. Une des difficultés essentielles est de déterminer le type de modélisation ou de transformée le mieux adapté à un problème donné. A titre d'exemple, considérons le problème exposé précédemment de détection précoce des défauts sur une ou plusieurs dents d'un engrenage. De multiples techniques de modélisation ont été étudiées pour ce problème (AR, ARMA, MA, PRONY, ...). La représentation qui s'est révélée la plus efficace est celle constituée des coefficients cepstraux obtenus à partir d'une modélisation AR d'ordre 3. Un modèle AR d'ordre élevé (ordre 50) est nécessaire pour bien modéliser l'allure temporelle et spectrale du signal. Par contre, se limiter à trois coefficients cepstraux donne de meilleurs résultats de classification. Dans cette application, on disposait de dix signaux par jour et les enregistrements ont été effectués sur dix jours. Le tableau individus/variables était alors de dimension $n \times p = 100 \times 3$.

Les prétraitements présentés succintement dans ce paragraphe sont issus principalement du traitement du signal. Cependant, ce problème de réduction du volume des données a été étudié depuis de nombreuses années par les

statisticiens. L'objectif des prochains paragraphes est de présenter certaines méthodes statistiques d'analyse de données [7],[11],[35].

2 Analyse en composantes principales (ACP)

Soit le tableau de données défini précédemment (i.e. $X_i(j)$). On peut considérer que ce tableau contient n vecteurs de $\mathbb{R}^p$ (les vecteurs lignes) ou p vecteurs de $\mathbb{R}^n$ (les vecteurs colonnes). En fonction de la représentation choisie, on obtient deux types d'analyses appelées respectivement **ACP des individus** et **ACP des variables**. Le problème de l'analyse en composantes principales peut être présenté de plusieurs façons. Nous adoptons la démarche qui nous a paru la plus simple qui consiste à rechercher la meilleure représentation des individus (resp. variables) après projection sur un sous espace de $\mathbb{R}^p$ (resp. $\mathbb{R}^n$) (voir[8], [11], [28]).

2.1 ACP des Individus

2.1.1 Formulation du problème

L'ACP des individus consiste à représenter au mieux le nuage des n points de $\mathbb{R}^p$ en projetant ce nuage dans un espace de dimension $q < p$. Par exemple, pour $p = 2$ et $q = 1$, on cherche à projeter un nuage de points du plan sur une droite, de façon à déformer le moins possible le nuage de points. Plus précisément, le problème de l'ACP des individus consiste à rechercher un sous espace de $\mathbb{R}^p$ de dimension q noté $W_q(A)$ (défini par un point A et q vecteurs $u_1, ..., u_q$ appelés vecteurs principaux) tel que la moyenne des carrés des distances entre les individus et leurs projections sur $W_q(A)$ soit minimale. Ceci nécessite donc de définir une distance entre les individus et leurs projections sur $W_q(A)$. On munit $\mathbb{R}^p$ du produit scalaire $\langle x, y\rangle = x^t My$, où M est une matrice symétrique définie positive à p lignes et p colonnes et de la norme associée $\|x\|_M = \sqrt{\langle x, x\rangle}$ (M doit être symétrique pour avoir $\langle x, y\rangle = \langle y, x\rangle$, positive pour avoir $\langle x, x\rangle \geq 0$ et définie pour que $\|x\|_M = 0 \Longrightarrow x = 0$). Le choix de la matrice M sera discuté ultérieurement, mais on peut tout de suite dire qu'il n'y a aucune raison de choisir pour M la matrice identité. En effet, dans le cas où $M = I$, la distance entre deux individus x et y est définie par $d^2(x, y) = \sum_{j=1}^{p} (x_j - y_j)^2$. Ceci consiste à attribuer à chaque variable la même importance. En analyse de données, les variables correspondent souvent à des grandeurs qui ont des unités différentes et pour donner une importance différente à chacune de ces variables, il suffit de choisir une matrice M appropriée. Soient X_i le $i^{ème}$ individu et $\widetilde{X}_i$ sa projection M-orthogonale sur le sous espace $W_q(A)$ passant par A et engendré

par les q vecteurs principaux $u_1, ..., u_q$. La *moyenne des carrés des distances entre les individus et leurs projections sur* $W_q(A)$ est définie par :

$$I_{W_q(A)} = \frac{1}{n}\sum_{i=1}^{n} \left\| X_i - \widetilde{X}_i \right\|_M^2 \tag{2.2}$$

Le théorème suivant permet de déterminer le point A :

Théorème 1 ([8], p. 35) ([7], p. 172) : Le sous-espace $W_q(A)$ de dimension q qui minimise $I_{Wq(A)}$ contient nécessairement le barycentre du nuage de points noté m.

Preuve

On note $\widetilde{X}_i$ la projection de X_i sur le sous-espace $W_q(m)$ engendré par les vecteurs $u_1, ..., u_q$ contenant m et X_i^* la projection de X_i sur le sous espace $W_q(A)$ engendré par les vecteurs $u_1, ..., u_q$ contenant A (bien entendu $W_q(A)$ est parallèle à $W_q(m)$). On a

$$\begin{aligned} I_{W_q(A)} &= \frac{1}{n}\sum_{i=1}^{n} \|X_i - X_i^*\|_M^2 = \frac{1}{n}\sum_{i=1}^{n} \left\| X_i - \widetilde{X}_i + \widetilde{X}_i - X_i^* \right\|_M^2 \\ &= \frac{1}{n}\sum_{i=1}^{n} \left\| X_i - \widetilde{X}_i \right\|_M^2 + \frac{1}{n}\sum_{i=1}^{n} \left\| \widetilde{X}_i - X_i^* \right\|_M^2 + \frac{2}{n}\sum_{i=1}^{n} \left\langle X_i - \widetilde{X}_i, \widetilde{X}_i - X_i^* \right\rangle \end{aligned}$$

En notant m^* la projection de m sur $W_q(A)$ et en remarquant que $\widetilde{X}_i - X_i^* = m - m^*$, on obtient :

$$\begin{aligned} I_{W_q(A)} &= \frac{1}{n}\sum_{i=1}^{n} \left\| X_i - \widetilde{X}_i \right\|_M^2 + \frac{1}{n}\sum_{i=1}^{n} \left\| \widetilde{X}_i - X_i^* \right\|_M^2 + \frac{2}{n}\sum_{i=1}^{n} \left\langle X_i - \widetilde{X}_i, m - m^* \right\rangle \\ &= \frac{1}{n}\sum_{i=1}^{n} \left\| X_i - \widetilde{X}_i \right\|_M^2 + \frac{1}{n}\sum_{i=1}^{n} \left\| \widetilde{X}_i - X_i^* \right\|_M^2 + \frac{2}{n}\left\langle \sum_{i=1}^{n} \left(X_i - \widetilde{X}_i \right), m - m^* \right\rangle \\ &= \frac{1}{n}\sum_{i=1}^{n} \left\| X_i - \widetilde{X}_i \right\|_M^2 + \frac{1}{n}\sum_{i=1}^{n} \left\| \widetilde{X}_i - X_i^* \right\|_M^2 \end{aligned}$$

On remarque donc que I_{W_q} atteint son minimum lorsque $\widetilde{X}_i = X_i^*$ c'est-à-dire lorsqu'on projette sur $W_q(m)$.

Ce premier théorème indique qu'on peut centrer les données et choisir A comme étant le vecteur nul de $\mathbb{R}^p$. On notera alors par simplicité W_q au lieu de $W_q(0)$. Le problème se réduit alors à rechercher les q vecteurs principaux $u_1, ..., u_q$. En remarquant que $X_i - \widetilde{X}_i$ est M-orthogonal à $\widetilde{X}_i$, i.e.

$\left\langle X_i, \widetilde{X}_i \right\rangle = \left\langle \widetilde{X}_i, \widetilde{X}_i \right\rangle$, on obtient facilement :

$$I_{W_q} = \frac{1}{n} \sum_{i=1}^{n} \|X_i\|_M^2 - \left\| \widetilde{X}_i \right\|_M^2 \tag{2.3}$$

Il suffit donc de rechercher le sous-espace W_q qui maximise *l'inertie des projections* définie par :

$$J_{W_q} = \frac{1}{n} \sum_{i=1}^{n} \left\| \widetilde{X}_i \right\|_M^2 \tag{2.4}$$

Le théorème suivant permet de réduire considérablement la difficulté du problème :

Théorème 2 ([28], p. 167) : Soit W_k le sous-espace de dimension k portant l'inertie maximale, alors le sous-espace de dimension $k+1$ portant l'inertie maximale est la somme directe de W_k et du sous-espace de dimension 1 M-orthogonal à W_k portant l'inertie maximale : *les solutions sont "emboitées".*

Ce théorème indique que pour obtenir W_q, on peut procéder de proche en proche en cherchant d'abord le sous-espace de dimension 1 d'inertie maximale, puis le sous-espace de dimension 1 M-orthogonal au précédent d'inertie maximale, etc. L'inertie d'un sous-espace de dimension 1 engendré par le vecteur M-unitaire u (i.e. $\|u\|_M^2 = 1$) est définie par :

$$J(u) = \frac{1}{n} \sum_{i=1}^{n} \left\| \widetilde{X}_i \right\|_M^2 = \frac{1}{n} \sum_{i=1}^{n} u^t M X_i X_i^t M^t u = u^t M T M u \tag{2.5}$$

avec $\widetilde{X}_i = (u^t M X_i)\, u$ (puisque $\|u\|_M^2 = 1$) et où $T = \frac{1}{n} \sum_{i=1}^{n} X_i X_i^t$ est la matrice des covariances des individus.

2.1.2 Détermination du premier axe principal

Définition 1. On appelle premier axe principal des individus $X_1, ..., X_n$ le vecteur $u_1 \in \mathbb{R}^p$ qui maximise $J(u) = u^t M T M u$ sous la contrainte $\|u_1\|_M^2 = 1$. La combinaison linéaire $C_1 = u_1^t M X$, où X est un individu de $\mathbb{R}^p$, est appelée première composante principale de X.

Le lagrangien correspondant au problème d'optimisation sous contrainte précédent est défini par

$$L(u) = u^t M T M u - \lambda \left(u^t M u - 1 \right)$$

On a :

$$\frac{\partial L(u)}{\partial u} = 0 \Rightarrow MTMu = \lambda Mu \tag{2.6}$$

Puisque la matrice M est définie positive, elle est inversible. Donc :

$$\frac{\partial L(u)}{\partial u} = 0 \Rightarrow TMu = \lambda u$$

Le vecteur u est donc nécessairement un vecteur propre de TM (associé à la valeur propre λ). On a :

$$J(u) = u^t MTMu = u^t M\lambda u = \lambda \tag{2.7}$$

Donc, plus la valeur propre λ est grande, plus le critère $J(u)$ est important.

Conclusion. Le premier vecteur principal des individus $X_1, ..., X_n$ est le vecteur propre u_1 M-unitaire (i.e. $\|u_1\|_M = 1$) associé à la plus grande valeur propre λ_1 de la matrice TM (le vecteur u_1 est unique au signe près si λ_1 est unique).

2.1.3 Détermination du deuxième axe principal

Définition 2. On appelle deuxième axe principal des individus $X_1, ..., X_n$ le vecteur u_2 qui maximise $J(u)$ sous les contraintes $\|u\|_M^2 = 1$ et $\langle u, u_1 \rangle_M = 0$. La combinaison linéaire $C_2 = u_2^t MX$, où X est un individu de $\mathbb{R}^p$, est appelée deuxième composante principale de X.

La deuxième contrainte s'écrit $u^t Mu_1 = 0$ et donc le lagrangien associé à ce problème d'optimisation sous contraintes est

$$L(u) = u^t MTMu - \lambda \left(u^t Mu - 1\right) - \mu u^t Mu_1 \tag{2.8}$$

d'où :

$$\frac{\partial L\,(u)}{\partial u} = 0 \Rightarrow 2MTMu - 2\lambda Mu - \mu Mu_1 = 0 \tag{2.9}$$

En multipliant scalairement cette équation par le vecteur u_1, on obtient :

$$2u_1^t MTMu - 2\lambda u_1^t Mu - \mu u_1^t Mu_1 = 0 \tag{2.10}$$

Mais $u_1^t Mu_1 = 1, u_1^t Mu = 0$ et $u_1^t MT = \lambda_1 u_1^t$ (car $TMu_1 = \lambda_1 u_1$). Donc :

$$\mu = 0 \text{ et } TMu = \lambda u \tag{2.11}$$

Le vecteur u est donc nécessairement un vecteur propre de la matrice TM. Puisque $u^tMTMu = \lambda$, plus la valeur propre λ est grande, plus le critère $J(u)$ est grand.

Conclusion. Le deuxième vecteur principal de X est le vecteur propre u_2 M-unitaire (i.e. $\|u_2\|_M = 1$) associé à la plus grande valeur propre $\lambda_2 \leq \lambda_1$ de la matrice TM (le vecteur u_2 est unique au signe près si λ_2 est une valeur propre simple).

2.1.4 Détermination des axes principaux

En itérant le procédé présenté ci-dessus, on détermine les valeurs propres et les vecteurs propres de la matrice TM. **Les axes principaux successifs sont les vecteurs propres de la matrice de covariance** TM **rangés suivant l'ordre des valeurs propres décroissantes**. La représentation des individus se fait alors par projection de ces individus sur les axes principaux : la coordonnée d'un individu X sur le $l^{ème}$ axe principal est $C_l = u_l^tMX$.

2.1.5 Nombre d'axes principaux

La matrice M étant symétrique définie positive, elle est diagonalisable sur une base orthonormée de vecteurs propres avec des valeurs propres positives. On peut donc écrire $M = PDP^t$, où P est une matrice de passage unitaire. En utilisant cette décomposition, on obtient $TMu = \lambda u \Leftrightarrow \sqrt{D}P^tTPDP^tu = \lambda\sqrt{D}P^tu$ c'est-à-dire $\sqrt{D}P^tTP\sqrt{D}v = \lambda v$ avec $v = \sqrt{D}P^tu$. Le réel λ et le vecteur u sont donc éléments propres de TM si et seulement si λ et$\sqrt{D}P^tu$ sont éléments propres de $\sqrt{D}P^tTP\sqrt{D}$. Cette dernière matrice étant symétrique semi-définie positive, elle est diagonalisable sur une base orthonormée de vecteurs propres. La matrice $\sqrt{D}P^tTP\sqrt{D}$ admet donc p valeurs propres $\lambda_1 \geq ... \geq \lambda_p \geq 0$ et les vecteurs propres associés sont orthogonaux au sens de la matrice identité i.e. $u^tv = 0$. Dans la mesure où les variables ne sont pas liées par une relation affine, la matrice T est symétrique définie positive donc inversible. La matrice $\sqrt{D}P^tTP\sqrt{D}$ est donc inversible et toutes ses valeurs propres sont strictement positives. On en déduit que la matrice TM admet généralement p valeurs propres $\lambda_1 \geq ... \geq \lambda_p > 0$ et les vecteurs propres associés sont orthogonaux au sens de la matrice M i.e. $u^tMv = 0$. Dans tous les cas pratiques, on a donc p axes principaux (orthogonaux au sens de M) et p composantes principales.

Cependant, dans une volonté de réduire la taille des vecteurs de données, on ne conserve en pratique que certains axes principaux. Une valeur propre particulière rapportée à la somme des valeurs propres exprime la part de

l'inertie portée sur l'axe principal correspondant, c'est-à-dire la pertinence de cet axe pour la séparation des classes. En effet, puisque les vecteurs u_i engendrant W_q sont M-orthogonaux, la *moyenne des carrés des distances entre les individus et leurs projections* s'écrit :

$$I_{W_q} = \frac{1}{n}\sum_{i=1}^{n} \|X_i\|_M^2 \left[1 - \frac{\sum_{j=1}^{q}\lambda_j}{\sum_{j=1}^{p}\lambda_j}\right] \tag{2.12}$$

L'utilisation pratique de l'analyse en composantes principales des individus consiste à extraire les q vecteurs propres de T tels que le rapport $\sum_{j=1}^{q}\lambda_j / \sum_{j=1}^{p}\lambda_j$ soit suffisamment proche de 1 (on peut se fixer un seuil de qualité 80%, 90%, ... ou représenter sur une courbe le pourcentage et s'arrêter d'ajouter des axes lorsque cette courbe présente une rupture)

Preuve de (2.12) :

$$\begin{aligned}
I_{W_q} &= \frac{1}{n}\sum_{i=1}^{n}\|X_i\|_M^2 - \frac{1}{n}\sum_{i=1}^{n}\left\|\widetilde{X}_i\right\|_M^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\|X_i\|_M^2 - \frac{1}{n}\sum_{i=1}^{n}\left\|\sum_{j=1}^{q}\langle X_i, u_j\rangle u_j\right\|_M^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\|X_i\|_M^2 - \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{q}\langle X_i, u_j\rangle^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\|X_i\|_M^2 - \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{q} u_j^t M X_i X_i^t M u_j \\
&= \frac{1}{n}\sum_{i=1}^{n}\|X_i\|_M^2 - \sum_{j=1}^{q} u_j^t M T M u_j \\
&= \frac{1}{n}\sum_{i=1}^{n}\|X_i\|_M^2 - \sum_{j=1}^{q}\lambda_j \ (\text{car } TMu_j = \lambda_j u_j)
\end{aligned}$$

Si on conserve toutes les valeurs propres de TM, on a $\widetilde{X}_i = X_i$ et donc $I_{W_q} = 0$, c'est-à-dire :

$$\frac{1}{n}\sum_{i=1}^{n}\|X_i\|_M^2 = \sum_{j=1}^{p}\lambda_j$$

On en conclut le résultat annoncé :

$$I_{W_q} = \frac{1}{n}\sum_{i=1}^{n} \|X_i\|_M^2 \left[1 - \frac{\sum_{j=1}^{q}\lambda_j}{\sum_{j=1}^{p}\lambda_j}\right]$$

2.1.6 Choix de la métrique : ACP Normée

Le choix de la matrice M et de la métrique associée est très important pour l'ACP. **En général, on ne choisit pas pour M la matrice identité**. En effet, dans ce cas, l'ACP a l'inconvénient d'être sensible aux unités suivant lesquelles sont mesurées les variables. En effet, si les valeurs correspondant à la j^{eme} variable sont élevées, cette variable interviendra fortement dans la matrice T et prendra un rôle prépondérant (qui est artificiel) pour la direction des axes principaux. On préfère donc une variante de l'ACP des individus appelée **ACP normée**. Cette variante élimine cet inconvénient en choisissant comme métrique $M = diag\left(1/\sigma_1^2, ..., 1/\sigma_p^2\right)$. Cela revient à effectuer une ACP sur les données centrées réduites $\left(X_i\left(j\right) - m\left(j\right)\right)/\sigma\left(j\right)$ ($m\left(j\right)$ et $\sigma\left(j\right)$ étant la moyenne et l'écart type de la $j^{ème}$ variable), avec la métrique identité. En effet, la matrice des covariances des individus $\left(X_i\left(j\right) - m\left(j\right)\right)/\sigma\left(j\right)$ est TM avec $M = diag\left(1/\sigma_1^2, ..., 1/\sigma_p^2\right)$.

2.2 ACP des variables

On cherche à représenter au mieux le nuage des p points de $\mathbb{R}^n$ en ne conservant que certaines combinaisons linéaires des individus. La procédure très analogue à l'ACP des individus consiste à chercher les vecteurs propres et les valeurs propres d'une matrice de taille $N \times N$. On peut montrer que cette matrice de taille $N \times N$ admet autant de valeurs propres non nulles que la matrice TM de l'ACP des individus (de taille $p \times p$). L'ACP des variables peut également être normée en travaillant sur les données $\left(X_i\left(j\right) - m\left(j\right)\right)/\sqrt{n}\sigma\left(j\right)$. La distance d'un vecteur colonne à l'origine est alors égale à l'unité. Chaque point est représenté sur une hypersphère de rayon unité. On représente généralement l'ACP des variables dans les plans principaux définis à partir des axes principaux intéressants (correspondant aux valeurs propres les plus grandes. La projection de chaque variable sur un plan principal est située à l'intérieur du cercle unité. Ceci permet d'évaluer directement la qualité de la représentation de chaque variable. **Une variable est d'autant mieux représentée que sa projection est proche du cercle.**

Pour interprèter une ACP des variables, il faut noter que le cosinus de l'angle formé par deux variables est égal au coefficient de corrélation entre ces deux variables. Le nuage de points de l'ACP doit être interprété e**n terme de corrélation entre les variables (**voir annexe).

Il existe un lien entre les deux types d'ACP normées : on montre que les vecteurs propres et les valeurs propres de l'ACP des variables peuvent se déduire des vecteurs propres et des valeurs propres de l'ACP des individus. Ceci

permet de construire les projections des individus et des variables à partir d'une seule ACP. On pourra trouver certaines démonstrations en annexe.

2.3 Représentation Simultanée

On se limite dans cette partie à des données centrées et à une métrique $M = diag\left(1/\sigma_1^2, ..., 1/\sigma_p^2\right)$, ce qui correspond à effectuer une ACP normée. D'après [11], on appelle individu caractéristique de la variable j, l'individu défini par le vecteur $e_j = (0, ..., 0, 1, 0, ..., 0)^t$ de la base canonique de $\mathbb{R}^p$. Ce vecteur est caractéristique de la $j^{ème}$ variable car il ne se distingue du vecteur nul que par sa $j^{ème}$ composante. Le vecteur e_j admet pour norme $\|e_j\|_M^2 = e_j^t M e_j = 1/\sigma_j^2$. La projection d'un individu X sur le vecteur e_j est $P_j(X) = (e_j^t MX)e_j = 1/\sigma_j^2 X(j)e_j = \frac{X(j)}{\sigma_j}\frac{e_j}{\|e_j\|_M}$. Donc plus la $j^{ème}$ composante de X à savoir $X(j)$ est importante, plus la projection de X sur l'axe e_j est éloignée le long de l'axe.

Définition : On appelle représentation simultanée des individus et des variables la représentation des individus et des individus caractéristiques des variables**.** Pour une interprètation de cette représentation, on se reportera à l'exemple du tableau de données.

Remarque : la projection de l'axe e_j sur le $l^{ème}$ axe principal est $e_j^t M u_l$.

2.4 Exemple du tableau de données

Reprenons le tableau de données introduit au chapitre 2

	Poids	Taille	Age	Note		Poids	Taille	Age	Note
X_1	45	1.50	13	14	X_6	60	1.70	14	7
X_2	50	1.60	13	16	X_7	70	1.60	14	8
X_3	50	1.65	13	15	X_8	65	1.60	13	13
X_4	60	1.70	15	9	X_9	60	1.55	15	17
X_5	60	1.70	14	10	X_{10}	65	1.70	14	11

L'ACP normée des individus et des variables ainsi que la représentation simultanée sont représentées sur les figures 2.1, 2.2 et 2.3.

- **Interprétation de l'ACP des individus**

  On constate deux groupes d'individus : le groupe $1, 2, 3$ et le groupe $4, 5, 6, 7, 10$. Les individus 8 et 9 paraissent isolés. Les caractéristiques physiques des individus permettent d'expliquer ces deux groupes. A droite se trouvent les élèves physiquement développés (et mal notés). A gauche se trouvent les élèves bien notés (et mal développés physiquement). La première composante principale permet de distinguer ces deux groupes. L'individu 8 est au milieu des deux groupes. Cela s'explique par le fait que, sur le plan physique, il est aussi bien développé que les élèves $4, 5, 6, 7, 10$ et que, sur le plan scolaire, il réussit aussi bien que les élèves $1, 2, 3$.

- **Interprétation de l'ACP des variables**

  Toutes les variables sont bien représentées sur le cercle de corrélation puisqu'elles sont proches de la circonférence. Il apparaît une distinction évidente entre la variable note et les variables taille, poids et age (qui sont très corrélées par rapport à la variable note) : la caractéristique intellectuelle s'oppose aux caractéristiques physiques. La première composante principale est définie par cette opposition. La deuxième composante principale aurait plutôt tendance à opposer l'age de la taille.

- **Interprétation de la Représentation Simultanée**

  Dans la mesure où les individus sont bien représentés, les valeurs de $X_i(1)$ et de $X_i(2)$ sont données par les longueurs des segments joignant leurs projections sur l'axe à l'origine. Par exemple, l'individu 9 est petit, gros, vieux mais il travaille bien. L'origine des axes représente l'individu moyen dont les variables sont les moyennes calculées sur l'ensemble des individus.

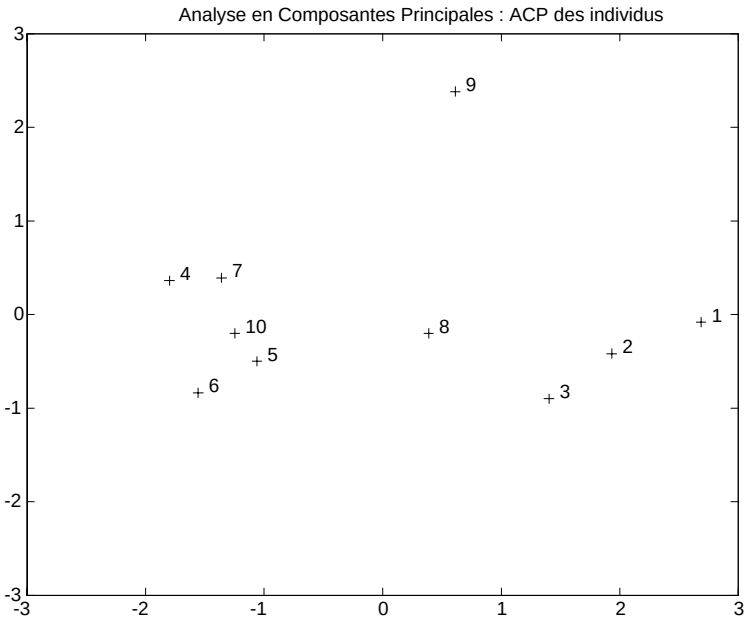


Fig. 2.1. ACP des Individus

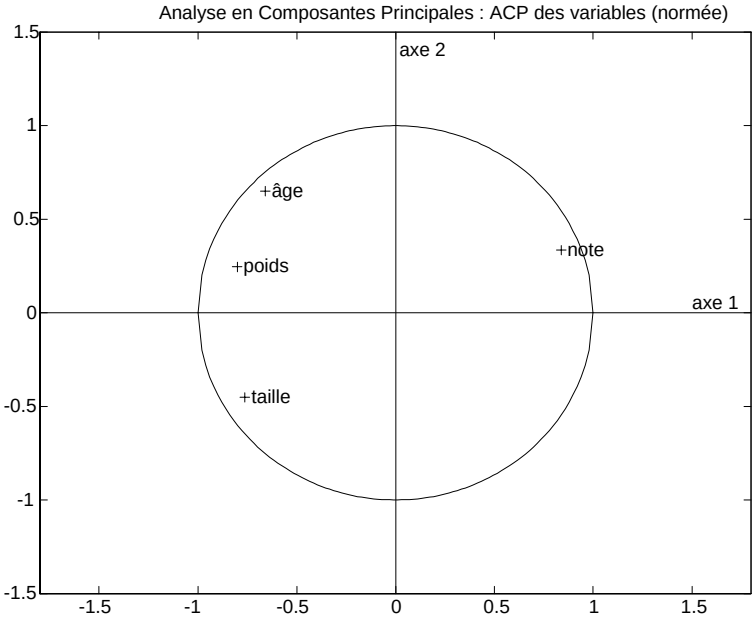


Fig. 2.2 ACP des Variables

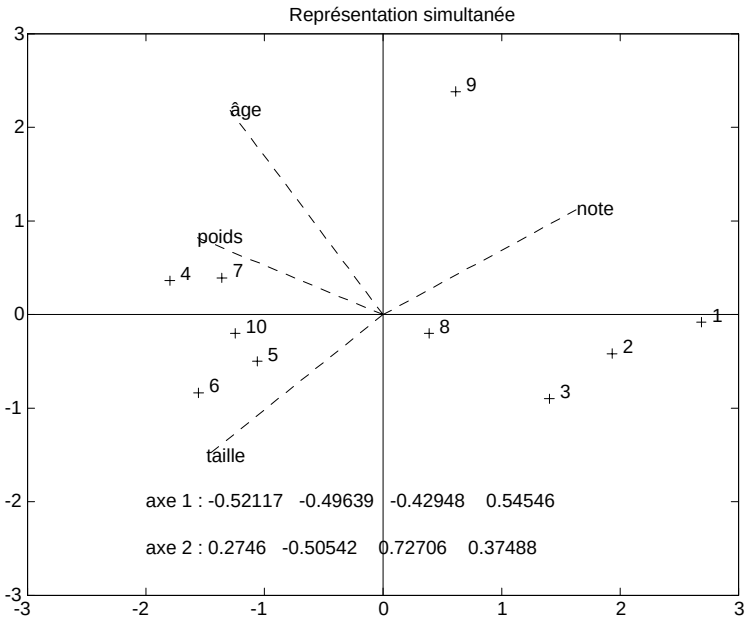


Fig. 2.3 Représentation Simultanée

3 Critère de Fisher et Analyse Factorielle Discriminante

Parmi les p paramètres obtenus sans modélisation (cas des étudiants) ou avec modélisation (cas des signaux temporels issus de l'analyse vibratoire), on cherche à déterminer les plus pertinents dans un but de classification. L'idée de Fisher [9] consiste à déterminer des axes (appelés axes factoriels discriminants) tels que les projections des divers points sur ces axes permettent de séparer les classes. L'exemple ci dessous montre qu'un axe judicieusement choisi peut séparer les diverses classes.

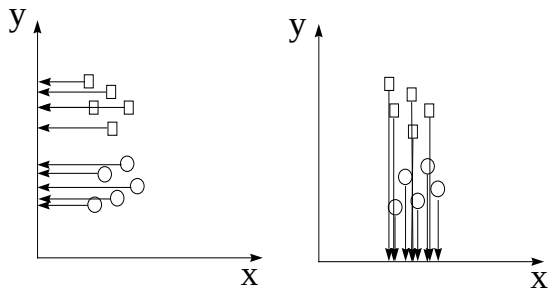


Fig. 2.4 Séparation des Classes après Projection

A la différence de l'ACP, l'analyse factorielle discriminante suppose une analyse en mode supervisé. On dispose alors d'une base d'apprentissage constituée de vecteurs expertisés répartis dans les différentes classes.

3.1 Premier axe factoriel discriminant (deux classes)

Matrices de dispersion interclasse et intraclasse et critère de Fisher

On dispose de n vecteurs $\mathbb{R}^p$ répartis en deux classes W_1 et W_2 (analyse supervisée). Si x désigne un tel vecteur, sa projection sur l'axe de vecteur directeur u est notée $\widetilde{x} = u^t x$. Soient m_1 et m_2 les moyennes des deux classes W_1 et W_2 et $\widetilde{m}_1$ et $\widetilde{m}_2$ leurs projections sur l'axe u. On peut définir la dispersion de la classe W_i après projection par :

$$\widetilde{s}_i^2 = \sum_{x \in W_i} (\widetilde{x} - \widetilde{m}_i)^2 \tag{2.13}$$

Le critère de Fisher se définit alors de la façon suivante :

$$J(u) = \frac{(\widetilde{m}_1 - \widetilde{m}_2)^2}{\widetilde{s}_1^2 + \widetilde{s}_2^2} \tag{2.14}$$

Fisher a proposé d'évaluer la pertinence d'un axe u pour la classification à l'aide du critère $J(u)$. Plus $J(u)$ est élevé, plus l'axe u est pertinent. On montre aisément que le critère $J(u)$ vérifie :

$$J(u) = \frac{u^t B u}{u^t S u} \tag{2.15}$$

B et S étant deux matrices appelées respectivement **matrice de dispersion interclasse** et **matrice de dispersion intraclasse** définies par :

$$B = (m_1 - m_2)(m_1 - m_2)^t \tag{2.16}$$

$$\begin{aligned} S &= S_1 + S_2 \\ &= \sum_{x \in W_1} (x - m_1)(x - m_1)^t + \sum_{x \in W_2} (x - m_2)(x - m_2)^t \end{aligned} \tag{2.17}$$

Remarque 1

Dans tous les cas, la matrice de dispersion intraclasse S est une matrice symétrique positive. Pour, $n > p$ elle est généralement définie et donc inversible. La matrice B est également symétrique positive mais elle n'est jamais inversible (elle est semidéfinie positive). En effet, d'après (2.16), le rang de B est au plus 1.

Remarque 2

On vérifiera que la **matrice de dispersion totale**

$$T = \sum_{x \in W_1 \cup W_2} (x - m)(x - m)^t \text{ avec } m = \frac{1}{n} \sum_{x \in W_1 \cup W_2} x \tag{2.18}$$

(notons que $\frac{1}{n}T$ est la matrice des covariances) est liée aux matrices de dispersion interclasse et intraclasse par la relation $T = S + \frac{n_1 n_2}{n} B$ avec $n_1 = card(W_1)$ et $n_2 = card(W_2)$.

Détermination du premier axe factoriel discriminant

Lorsque l'on veut déterminer l'axe le plus discriminant appelé premier axe factoriel discriminant, il suffit de déterminer le vecteur u_1 qui maximise $J(u) = \frac{u^t B u}{u^t S u}$. On remarque que $J(au) = J(u)$ pour tout $a \in \mathbb{R}$. On peut alors effectuer une maximisation de $u^t B u$ sous la contrainte $u^t S u = 1$ (pour $u^t S u = a^2$, on a $v^t S v = 1$ avec $v = au$).

Définition. On appelle premier axe factoriel discriminant du tableau de données $X_i(j)$ le vecteur u_1 qui maximise le critère $u^t B u$ sous la contrainte $u^t S u = 1$.

Le Lagrangien associé à ce problème d'optimisation est :

$$L(u) = u^t B u - \lambda \left(u^t S u - 1 \right) \tag{2.19}$$

qui admet pour dérivée

$$L'(u) = 2Bu - 2\lambda S u \tag{2.20}$$

L'annulation de $L'(u)$ conduit alors à

$$Bu = \lambda S u \tag{2.21}$$

Mais $Bu = \alpha (m_1 - m_2)$ d'où :

$$u = kS^{-1}(m_1 - m_2) \tag{2.22}$$

Comme précisé précédemment, la matrice S est en général inversible pour $n > p$.

3.2 Premier axe factoriel discriminant ($K \geq 2$)

Matrices de dispersion interclasse et intraclasse et critère de Fisher

L'expression (2.17) de la matrice de dispersion intraclasse se généralise aisément au cas de K classes :

$$S = \sum_{i=1}^{K} S_i = \sum_{i=1}^{K} \sum_{x \in W_i} (x - m_i)(x - m_i)^t \tag{2.23}$$

La généralisation de la matrice B n'est pas si évidente. Dans le cas de K classes, la matrice de dispersion T s'écrit :

$$T = \sum_{i=1}^{K} \sum_{x \in W_i} (x - m)(x - m)^t \tag{2.24}$$

En décomposant $x - m$ sous la forme $x - m_i + m_i - m$, on obtient:

$$T = \sum_{i=1}^{K} \sum_{x \in W_i} (x - m_i)(x - m_i)^t + \sum_{i=1}^{K} \sum_{x \in W_i} (m_i - m)(m_i - m)^t \tag{2.25}$$

c'est-à-dire :

$$T = S + \sum_{i=1}^{K} n_i (m_i - m)(m_i - m)^t \tag{2.26}$$

Il parait alors naturel de définir la matrice de dispersion interclasse par:

$$B = \sum_{i=1}^{K} n_i (m_i - m)(m_i - m)^t \tag{2.27}$$

Remarque

Dans le cas $K = 2$, la matrice B ainsi définie est égale à $\frac{n_1 n_2}{n}$fois celle définie précédemment. Pour avoir des définitions analogues, il suffirait de définir le critère de Fisher de la façon suivante $J(u) = \frac{n_1 n_2}{n} \frac{(\tilde{m}_1 - \tilde{m}_2)^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$, ce qui revient évidemment à la même chose que ce qui a été proposé.

Détermination du premier axe factoriel discriminant

Puisque $I(au) = I(u), \forall a \in \mathbb{R}$, on peut effectuer une maximisation de $u^t B u$ sous la contrainte $u^t S u = 1$. Comme précédemment, l'annulation du Lagrangien associé à ce problème d'optimisation conduit à:

$$Bu = \lambda S u \tag{2.28}$$

Pour $n > p$, la matrice S est en général inversible, donc u est nécessairement un vecteur propre de $S^{-1}B$. Si λ est la valeur propre de $S^{-1}B$ associée à u, on a $I(u) = \lambda$. Le premier axe discriminant est donc le vecteur propre u_1 associé à la plus grande valeur propre de $S^{-1}B$ notée λ_1(unique au signe près si la valeur propre λ_1 est simple). La valeur propre λ_1 représente le pouvoir discriminant du premier axe discriminant u_1. Plus cette valeur propre est grande, plus l'axe est discriminant.

Remarque 1

B et S sont des matrices symétriques positives ($u^tBu \geq 0$ et $u^tSu \geq 0, \forall u$). **Les valeurs propres de $S^{-1}B$ sont donc toutes positives ou nulles**. En effet si $S^{-1}Bu = \lambda u$, alors $u^tBu = \lambda u^tSu$ et donc $\lambda \geq 0$.

Remarque 2

La maximisation de u^tBu sous la contrainte $u^tSu = 1$ correspond à la première étape de l'ACP avec une matrice de covariance qui serait $S^{-1}BS^{-1}$ (en effet $u^tBu = u^tS(S^{-1}BS^{-1})Su$) et une matrice $M = S$ (qui est bien symétrique définie positive).

Remarque 3

La maximisation de $I(u) = \frac{u^tBu}{u^tSu}$ est équivalente à la maximisation de $J(u) = \frac{u^tBu}{u^tTu}$ en vertu du théorème de Huyghens $u^tTu = u^tBu + u^tSu$. La maximisation de $J(u)$ conduit à $T^{-1}Bu = \mu u$ ($\frac{1}{n}T$ est une estimation de la matrice de covariance ; elle est donc symétrique positive et inversible dans les mesure où les variables ne sont pas liées par une relation affine). **Les vecteurs propres de $S^{-1}B$ et de $T^{-1}B$ sont donc identiques.** On peut d'ailleurs vérifier cette propriété très rapidement. En effet, soit u un vecteur propre de $S^{-1}B$ associé à la valeur propre $\lambda \geq 0$ i.e. $Bu = \lambda Su$. Si $\lambda = 0$, alors $Bu = 0$ et donc $T^{-1}Bu = 0$, ce qui signifie que u est un vecteur propre de $T^{-1}B$ avec la valeur propre $\lambda = 0$. Si $\lambda > 0$, puisque $T = B + S$, on a $Tu = \frac{1+\lambda}{\lambda}Bu$ d'où $T^{-1}Bu = \frac{\lambda}{1+\lambda}u$, ce qui signifie que u est un vecteur propre de $T^{-1}B$ avec la valeur propre $\mu = \frac{\lambda}{1+\lambda}$ (avec $0 < \frac{\lambda}{1+\lambda} < 1$). **Ces résultats indiquent également que les valeurs propres de $T^{-1}B$ sont dans l'intervalle** $[0, 1]$.

Remarque 3

Les cas limites $\mu_1 = 0$ et $\mu_1 = 1$ (ou $\lambda_1 = 0$ et $\lambda_1 = +\infty$) peuvent s'expliquer de la façon suivante (voir [28]):

* pour $\mu_1 = 0$, le meilleur axe discriminant a un pouvoir nul. Aucune séparation linéaire n'est possible. Cela correspond par exemple au cas où les nuages sont concentriques de même centre de gravité. Cependant, dans certains cas, il existe une possibilité de discrimination non-linéaire.

* pour $\mu_1 = 1$, les dispersions intraclasses après projection sur u sont nulles. Les nuages de points sont donc chacun dans un hyperplan orthog-

onal à u. Si les centres de gravité se projettent en des points différents, la discrimination est parfaite.

On peut noter que la valeur propre μ (ou λ) est une mesure pessimiste du pouvoir discriminant d'un axe. On peut parfois discriminer parfaitement avec $\mu < 1$.

3.3 Deuxième axe factoriel discriminant

On opère comme pour l'analyse en composantes principales avec $T = S^{-1}BS^{-1}$ et $M = S$.

Définition.

On appelle deuxième axe factoriel discriminant du tableau $X_i(j)$, le vecteur u_2 qui maximise $I(u)$ sous les contraintes $u^tSu = 1$ et $u^tSu_1 = 0$.

Détermination du deuxième axe principal

Le lagrangien associé au problème précédent est

$$L(u) = u^tBu - \lambda(u^tSu - 1) - \mu u^tSu_1 \tag{2.29}$$

qui est très similaire au lagrangien de l'analyse en composantes principales défini par l'eq. (2.8). On obtient successivement $\mu = 0$ et $Bu = \lambda Su$. Le vecteur u est donc un vecteur propre de $S^{-1}B$. Puisque $u^tBu = \lambda$, plus la valeur propre λ est grande, plus le critère $I(u)$ est grand.

Conclusion. Le deuxième axe factoriel discriminant du tableau $X_i\,(j)$ est le vecteur propre u_2 unitaire au sens de S (i.e. $\|u_2\|_S = 1$) associé à la plus grande valeur propre $\lambda_2 \leq \lambda_1$ de la matrice $S^{-1}B$ (le vecteur u_2 est unique au signe près si λ_2 est une valeur propre simple).

Remarque 1

En itérant le procédé, on détermine les valeurs propres et les vecteurs propres de la matrice $S^{-1}B$. **Les axes factoriels discriminants successifs sont les vecteurs propres de la matrice $S^{-1}B$ rangés suivant l'ordre des valeurs propres décroissantes** (ces vecteurs propres sont orthogonaux au sens de la matrice S i.e. $u^tSv = 0$).

Remarque 2

Comme on l'a précisé ci-dessus, il est équivalent de réaliser une analyse factorielle discriminante sur le tableau $X_i(j)$ ou d'effectuer une ACP sur des données dont la matrice de covariance serait $T = S^{-1}BS^{-1}$ avec une métrique $M = S$.

Remarque 3

On peut voir l'analyse factorielle discriminante de la façon suivante [9] : déterminer q vecteurs notés $u_1, ..., u_q$ engendrant un sous espace de dimension $q < p$ noté S_q tel que le rapport de la dispersion interclasse et de la

dispersion intraclasse des projections soit maximal. Il suffit alors de déterminer la dispersion interclasse et la dispersion intraclasse des projections. Pour cela, on définit la projection d'un vecteur X_i sur u_k par $u_k^t X_i$. Le vecteur $\widetilde{X_i} = \left(u_1^t X_i, ..., u_q^t X_i\right)^t$ qui correspond à la projection de X_i sur S_q s'écrit alors $\widetilde{X_i} = U^t X_i$ où U est une matrice à p lignes et q colonnes. Les matrices de covariance interclasse et intraclasse des projections $\widetilde{X_i}$ s'écrivent alors $\widetilde{B} = U^t B U$ et $\widetilde{S} = U^t S U$. Le problème consiste alors à déterminer la matrice U qui maximise le rapport de la variance interclasse et de la variance intraclasse des projections défini par $I(U) = \frac{\det(U^t B U)}{\det(U^t S U)}$ (le déterminant d'une matrice est le produit des valeurs propres de cette matrice et correspond donc à une mesure de dispersion). On montre alors que la matrice U maximisant $I(U)$ est constituée des q vecteurs propres associés aux q plus grandes valeurs propres de $S^{-1}B$.

Remarque 4

Lorsque l'on veut déterminer la pertinence des diverses variables, il suffit de déterminer $I(e_i)$ ou $J(e_i)$ avec $i \in \{1, ..., p\}$, e_i étant le $i^{ème}$ axe de coordonnées. Dans l'exemple d'analyse des vibrations cité précédemment, on obtient, en considérant les 15 premiers coefficients cepstraux les résultats suivants :

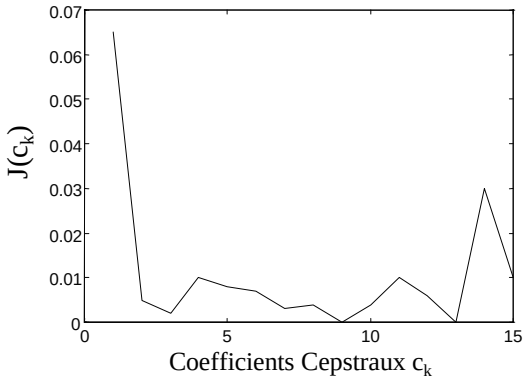


Fig. 2.2 Critère de Fisher pour les coefficients cepstraux

Nombre d'axes factoriels discriminants

Dans la quasi totalité des cas pratiques, on a $n > p \geq K$. Seuls les vecteurs propres de $S^{-1}B$ correspondant à des valeurs propres non nulles sont intéressants. Or :

$$S^{-1}Bu = 0 \Rightarrow Bu = 0 \Rightarrow u^t B u = 0 \tag{2.30}$$

d'où

$$u^t B u = \sum_{i=1}^{K} n_i \left[u^t (m_i - m)\right]^2 = 0 \Rightarrow u^t (m_i - m) = 0 \quad \forall i \in \{1, ..., K\} \tag{2.31}$$

L'ensemble des vecteurs $m_i - m$ est en général de dimension $K - 1$ (on a $\sum_{i=1}^{K} n_i (m_i - m) = 0$). Son orthogonal est de dimension $p - K + 1$. On peut donc trouver $p - K + 1$ vecteurs associés à la valeur propre $\lambda = 0$ et $K - 1$ vecteurs propres associés à des valeurs propres non nulles. **On peut donc déterminer $K - 1$ axes factoriels discriminants.**

Remarque

Dans le cas de deux classes, le problème de l'analyse factorielle discriminante consiste à projeter les données sur une droite, c'est-à-dire déterminer un hyperplan qui sépare au mieux les deux classes (droite si $p = 2$, plan si $p = 3$, etc ...). Dans le cas de K classes, on détermine un espace de dimension $K - 1$ dans lequel les projections sont séparées en K nuages.

Chapitre 3

METHODES STATISTIQUES

1 Décision Bayésienne

La décision bayésienne est l'approche probabiliste fondamentale du problème de la reconnaissance des formes. Elle suppose que le problème admet un modèle probabiliste c'est-à-dire que les lois de probabilité de chaque classe sont connues. Cette hypothèse n'est pas toujours réaliste et sera supprimée dans les parties *Apprentissage supervisé et apprentissage non supervisé*.

1.1 Etude théorique

On suppose défini un espace des formes Ω et une partition P de cet espace en K classes $\omega_1, ..., \omega_K$ que l'on identifie aux K formes possibles. On suppose également défini un ensemble A constitué des diverses actions possibles $a_1, ..., a_q$. On peut parfois ne pas identifier une forme dans un cas litigieux (décision avec rejet). Le nombre d'actions n'est donc pas toujours égal au nombre de formes possibles. On effectue sur chaque forme p mesures $x(1), ..., x(p)$ et on appelle X l'espace des mesures. Dans la suite, on suppose sans perte de généralité que $X = \mathbb{R}^p$.

Définir une règle de classification, c'est définir une fonction de décision d définie de X dans A, qui associe à toute mesure x une action a_i. On pose :

Probabilité d'apparition à priori de la classe ω_i

$$P(\omega_i) \tag{3.1}$$

Densité de probabilité du vecteur x conditionnelle à son appartenance à la classe ω_i

$$f(x|\omega_i) \tag{3.2}$$

Probabilité que la forme représentée par x appartienne à la classe ω_i

$$P(\omega_i|x) \tag{3.3}$$

Densité de probabilité du vecteur x

$$f(x) \tag{3.4}$$

Pour développer la règle de décision Bayésienne, on doit connaître $P(\omega_i)$ et $f(x|\omega_i)$. En utilisant la règle de Bayes et le théorème des probabilités totales, on en déduit alors :

$$P(\omega_i|x) = \frac{f(x|\omega_i)P(\omega_i)}{f(x)} \tag{3.5}$$

$$f(x) = \sum_{i=1}^{K} f(x|\omega_i)P(\omega_i) \tag{3.6}$$

Pour la règle de décision Bayésienne, on définit une fonction de coût $c(a_j, \omega_i) \geq 0$ qui est le coût de prendre la décision a_j alors que la forme représentée par x appartient à la classe ω_i. Le coût moyen de prendre la décision a_j ayant observé x est alors défini par :

$$R(a_j|x) = \sum_{i=1}^{K} c(a_j, \omega_i) P(\omega_i|x) \tag{3.7}$$

$R(a_i|x)$ est appelé risque conditionnel. Le coût moyen de prendre la décision $d(x)$ ayant observé x est défini de la même façon par

$$R_d(x) = \sum_{i=1}^{K} c(d(x), \omega_i) P(\omega_i|x) \tag{3.8}$$

Le coût moyen total associé à la fonction de décision d est alors défini par

$$R_d = \int_{\mathbb{R}^p} R_d(x) f(x) dx \tag{3.9}$$

Le problème de la décision Bayésienne consiste à déterminer une fonction de décision d qui minimise R_d. Puisque $f(x) \geq 0$, la fonction d qui minimise R_d notée d^* vérifie :

$$R_{d^*}(x) \leq R_d(x) \qquad \forall x \in \mathbb{R}^p$$

On considère maintenant le cas habituel de la décision Bayésienne sans rejet pour lequel l'espace des décisions ne comporte que les décisions $a_1, ..., a_K$ correspondant aux K classes $\omega_1, ..., \omega_K$ (on se reportera aux exercices pour des exemples de décision Bayésienne avec rejet). On note $c_{ji} = c(a_j, \omega_i)$ et $R_j(x) = R(a_j|x) = \sum_{j=1}^{K} c_{ji} P(\omega_i|x)$. La règle de décision Bayésienne s'énonce alors ainsi :

$$d^*(x) = a_j \Longleftrightarrow R_j(x) \leq R_k(x) \qquad \forall k$$

1.2 Cas particulier de deux classes

On a alors :

$$\begin{aligned} R_1(x) &= c_{11} P(\omega_1|x) + c_{12} P(\omega_2|x) \\ R_2(x) &= c_{21} P(\omega_1|x) + c_{22} P(\omega_2|x) \end{aligned} \tag{3.10}$$

La règle de décision bayésienne est alors définie par :

$$d^*(x) = a_1 \Leftrightarrow R_1(x) \leq R_2(x) \tag{3.11}$$

Il est raisonnable de supposer que le coût d'une décision correcte est inférieur au coût d'une erreur. On a alors :

$$c_{11} < c_{12} \text{ et } c_{22} < c_{21} \tag{3.12}$$

d'où :

$$d^*(x) = a_1 \Leftrightarrow \frac{f(x|\omega_1)}{f(x|\omega_2)} \geq \frac{c_{12} - c_{22}}{c_{21} - c_{11}} \frac{P(\omega_2)}{P(\omega_1)} \tag{3.13}$$

Le rapport $\frac{f(x|\omega_1)}{f(x|\omega_2)}$ est souvent appelé rapport de vraisemblance. La règle de décision bayésienne consiste à comparer le rapport de vraisemblance à une quantité indépendante de x notée :

$$Q = \frac{c_{12} - c_{22}}{c_{21} - c_{11}} \frac{P(\omega_2)}{P(\omega_1)} = \alpha\beta \text{ avec } \alpha = \frac{c_{12} - c_{22}}{c_{21} - c_{11}} \text{ et } \beta = \frac{P(\omega_2)}{P(\omega_1)} \tag{3.14}$$

1.3 Cas d'une fonction de coût non informative

Lorsqu'on a aucune information a priori, on choisit souvent :

$$c_{ji} = 1 - \delta_{ij} = \begin{cases} 0 & \text{si } i = j \\ 1 & \text{si } i \neq j \end{cases} \tag{3.15}$$

On a alors

$$R_j(x) = \sum_{i=1}^{K} c_{ji} P(\omega_i | x) = \sum_{i \neq j}^{K} P(\omega_i | x) = 1 - P(\omega_j | x) \tag{3.16}$$

d'où la règle de décision bayésienne

$$d^*(x) = a_j \Leftrightarrow P(\omega_j | x) \geq P(\omega_k | x), \forall k \in \{1, ..., K\} \tag{3.17}$$

Cas de deux classes

Dans le cas de deux classes, la règle devient :

$$d^*(x) = a_1 \Leftrightarrow P(\omega_1 | x) \geq P(\omega_2 | x) \Leftrightarrow \frac{f(x | \omega_1)}{f(x | \omega_2)} \geq \frac{P(\omega_2)}{P(\omega_1)} \tag{3.18}$$

1.4 Interprétation en terme de probabilité d'erreur

On définit la probabilité d'erreur de la règle de décision bayésienne par

$$P_e = \sum_{i=1}^{K} P[d(x) = a_i \cap x \notin \omega_i] \tag{3.19}$$

Lorsque la fonction de coût est $c_{ji} = 1 - \delta_{ij}$, on montre que la **règle de Bayes qui minimise le coût moyen total R_d minimise également la probabilité d'erreur P_e. Dans ce cas, la règle de Bayes est donc optimale au sens du coût moyen total R_d défini précédemment mais aussi au sens d'une probabilité d'erreur minimale.**

Preuve :

On étudie tout d'abord le cas de deux classes ω_1 et ω_2 pour lequel

$$P_e = P[d(x) = a_1 \cap x \in \omega_2] + P[d(x) = a_2 \cap x \in \omega_1]$$

On a alors

$$P_e = P[d(x) = a_1 \mid x \in \omega_2] P(\omega_2) + P[d(x) = a_2 \mid x \in \omega_1] P(\omega_1) \tag{3.20}$$

Notons $R_1 = \{x \in \mathbb{R}^p / d(x) = a_1\}$ et $R_2 = \{x \in \mathbb{R}^p / d(x) = a_2\}$ les régions

d'acceptation pour les classes ω_1 et ω_2. On a alors :

$$\begin{aligned} P_e &= \int_{R_2} P(\omega_1) f(x|\omega_1) dx + \int_{R_1} P(\omega_2) f(x|\omega_2) dx \\ &= P(\omega_2) \left[1 - \int_{R_2} f(x|\omega_2) dx\right] + \int_{R_2} P(\omega_1) f(x|\omega_1) dx \\ &= P(\omega_2) + \int_{R_2} [P(\omega_1) f(x|\omega_1) - P(\omega_2) f(x|\omega_2)] dx \\ &= P(\omega_2) - \int_{R_2} [P(\omega_2|x) - P(\omega_1|x)] f(x) dx \end{aligned}$$

La probabilité d'erreur est donc minimale lorsque R_2 est définie comme suit :

$$R_2 = \{x / P(\omega_2|x) > P(\omega_1|x)\} \tag{3.21}$$

On retrouve la règle de Bayes définie ci-dessus (voir eq. (3.18)).

On peut aisément illustrer graphiquement ce résultat dans le cas de deux classes avec une variable x monodimensionnelle :

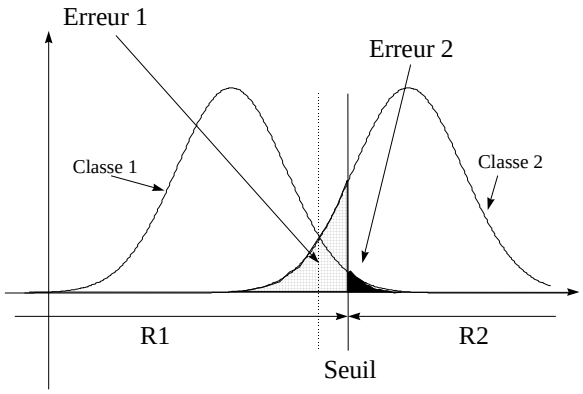


Fig 4.1 : Probabilité d'erreur de la décision bayésienne

On suppose que l'on s'est fixé un seuil représenté par la ligne verticale en trait continu. Lorsque x est supérieur à ce seuil ($x \in R_1$), on prend la décision $d(x) = a_2$ et vice-versa. Il y a deux cas d'erreur :

1) x appartient à ω_2 et x est inférieur au seuil (Probabilité de Fausse Alarme)

2) x appartient à ω_1 et x est supérieur au seuil (Probabilité de Non Détection)

La probabilité d'erreur correspondant à la règle de décision bayésienne est la somme des aires noircies, soit :

$$P_e = \int_{R_2} f\left(x\left|\omega_1\right.\right) P\left(\omega_1\right) dx + \int_{R_1} f\left(x\left|\omega_2\right.\right) P\left(\omega_2\right) dx \tag{3.22}$$

Lorsqu'on déplace le seuil jusqu'au trait en pointillé, on réduit au minimum la surface représentant la probabilité d'erreur. C'est exactement ce que dit la règle de Bayes : "tant que $f\left(x\left|\omega_1\right.\right) P\left(\omega_1\right) < f\left(x\left|\omega_2\right.\right) P\left(\omega_2\right)$, il est avantageux (moins couteux) de prendre la décision $d\left(x\right) = \omega_2$". La généralisation au cas de K classes est immédiate. On montre que la probabilité d'erreur est minimale lorsque

$$R_i = \left\{x / P\left(\omega_i\left|x\right.\right) > P\left(\omega_j\left|x\right.\right), \forall j\right\} \tag{3.23}$$

Influence des probabilités a priori

On suppose toujours $\alpha = 1$. Lorsque les deux classes sont a-priori équiprobables ($P\left(\omega_1\right) = P\left(\omega_2\right)$), le seuil optimal est x_0. Lorsque $P\left(\omega_1\right) > P\left(\omega_2\right)$, on a $\beta < 1$ et donc $Q = \alpha\beta < 1$. Le seuil est alors à droite de x_0. Cela revient à dire qu'on choisit plus souvent l'hypothèse ω_1 pour $P\left(\omega_1\right) > P\left(\omega_2\right)$ que pour $P\left(\omega_1\right) = P\left(\omega_2\right)$, ce qui est tout à fait normal.

Influence des coûts de décision

On peut faire le même type d'analyse que précédemment pour une fonction de coût différente de $c_{ji} = 1 - \delta_{ij}$. Dans la pratique, on définit des coût différents pour les diverses hypothèses lorsque les erreurs n'ont pas la même importance. On peut citer, à titre d'exemple, les cas de la détection de cibles ou du dépistage de maladies. Soit ω_1 la classe "présence d'une cible" ou "patient malade" et ω_2 la classe "absence d'une cible" ou "patient non malade". Prendre la décision $d\left(x\right) = a_2$ alors que x appartient à la classe ω_1 a des conséquences graves (cible non détectée, maladie non dépistée). Cette erreur est **l'erreur de non détection** (risque α en probabilités). Prendre la décision $d\left(x\right) = \omega_1$ alors que x appartient à la classe ω_2, est une erreur moins grave que la précédente car des tests supplémentaires permettent sa correction (par exemples d'autres examens ou d'autres radars). Cette erreur est **l'erreur de fausse alarme** (risque β en probabilités) :

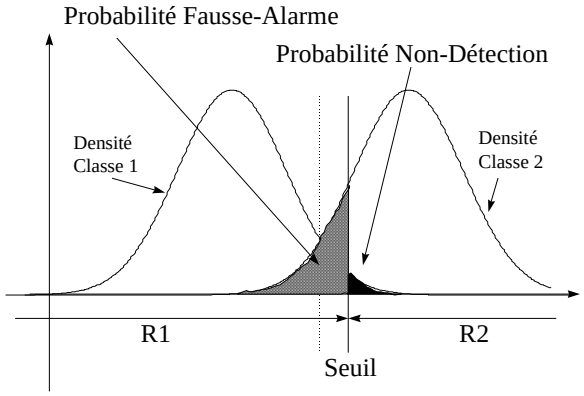


Fig. 4.2 Probabilité de Fausse-Alarme et de Non-Détection

Supposons donc que les coûts c_{ji} soient différents de $1-\delta_{ij}$, par exemple $c_{11} = c_{22} = 0$ et $c_{21} > c_{12}$ (le coût de non détection est plus important que le coût de fausse alarme). Le paramètre α est alors inférieur à 1 d'où $Q < 1$. Comme précédemment (influence des probabilités a priori), Le seuil est à droite de x_0. Cela revient à dire qu'on choisit plus souvent l'hypothèse $\omega_1$que l'hypothèse ω_2, ce qui est logique puisque le choix de l'hypothèse ω_2 est très risqué.

1.5 Densités Gaussiennes

Supposons que le vecteur x admette une densité de probabilité gaussienne, conditionnellement à chaque classe. On a alors :

$$f\left(x\left|\omega_i\right.\right) = \frac{1}{\left(2\pi\right)^{p/2}\sqrt{\det\Sigma_i}}\exp -\frac{1}{2}\left(x-m_i\right)^t\Sigma_i^{-1}\left(x-m_i\right) \qquad x\in\mathbb{R}^p \tag{3.24}$$

$\det\Sigma_i$ étant le déterminant de la matrice de covariance de la $i^{ème}$ classe Σ_i. La règle de décision bayésienne, pour une fonction de coût $c_{ji} = 1-\delta_{ji}$, s'écrit :

$$d^*\left(x\right) = a_i \Leftrightarrow P\left(\omega_i\left|x\right.\right) \geq P\left(\omega_k\left|x\right.\right) \qquad \forall k \tag{3.25}$$

c'est-à-dire :

$$d^*\left(x\right) = a_i \Leftrightarrow g_i\left(x\right) \geq g_k\left(k\right) \qquad \forall k \tag{3.26}$$

$$\begin{aligned} g_i\left(x\right) = -\left(x-m_i\right)^t\Sigma_i^{-1}\left(x-m_i\right) - \ln\det\Sigma_i \\ +2\ln P\left(\omega_i\right) \end{aligned} \tag{3.27}$$

La règle de décision se fait en déterminant le maximum de fonctions $g_i(x)$, appelées **fonctions discriminantes**. Dans le cas Gaussien, les fonctions discriminantes sont des formes quadratiques par rapport à x.

1.5.1 Matrices de Covariance Identiques

Lorsque les matrices de covariances des différentes classes sont identiques ($\Sigma_i = \Sigma$), on a la règle de décision suivante :

$$\begin{array}{c} d^*(x) = a_i \\ \Updownarrow \\ (x-m_i)^t \Sigma^{-1}(x-m_i) - (x-m_k)^t \Sigma^{-1}(x-m_k) \leq \ln \frac{P(\omega_i)}{P(\omega_k)} \qquad \forall k \end{array} \tag{3.28}$$

c'est-à-dire :

$$d^*(x) = a_i \Leftrightarrow \left(x - \frac{1}{2}(m_i + m_k)\right)^t \Sigma^{-1}(m_i - m_k) \geq \ln \frac{P(\omega_k)}{P(\omega_i)} \qquad \forall k \tag{3.29}$$

Dans ce cas, les fonctions discriminantes sont affines par rapport à x (c'est-à-dire de la forme $Ax + b$).

1.5.2 Règle de la distance aux barycentres

Lorsque les matrices de covariance sont identiques et que les classes sont équiprobables, la règle de décision bayésienne s'écrit :

$$d^*(x) = a_i \Leftrightarrow (x-m_i)^t \Sigma^{-1}(x-m_i) \leq (x-m_k)^t \Sigma^{-1}(x-m_k) \qquad \forall k \tag{3.30}$$

- Pour $\Sigma = \sigma^2 I_d$, la règle de décision Bayésienne se réduit à la règle très simple suivante :

  $$d^*(x) = a_i \Leftrightarrow d_2(x, m_i) \leq d_2(x, m_k) \qquad \forall k$$

  où d_2 est la distance euclidienne usuelle. On affecte donc x à la classe dont le barycentre est le plus proche de x au sens de la distance euclidienne : c'est **la règle de la distance aux barycentres**.

- Pour $\Sigma \neq \sigma^2 I_d$, on définit une distance entre x et m_k appellée **distance de Mahalanobis** par

  $$d_M(x, m_k) = \sqrt{(x-m_k)^t \Sigma^{-1}(x-m_k)}$$

  La règle de décision bayésienne devient alors la **règle de la distance aux barycentres** au sens de la distance de Mahalanobis.

Dans le cas gaussien général, on a toujours une règle de distance aux barycentres avec

$$d^2(x, m_k) = (x - m_k)^t \Sigma_k^{-1} (x - m_k) + \ln \det \Sigma_k - 2 \ln P(\omega_k)$$

1.6 Classifieur et fonctions discriminantes

Dans la plupart des cas, on peut représenter un classifieur sous la forme d'un ensemble de fonctions discriminantes $g_i(x)$, $i \in \{1, ..., K\}$. Le classifieur attribue à x la classe ω_i telle que :

$$g_i(x) \geq g_k(x) \qquad \forall k \tag{3.31}$$

Pour le classifieur bayésien, $g_i(x) = -R_i(x)$. On peut alors représenter le classifieur de la façon suivante :

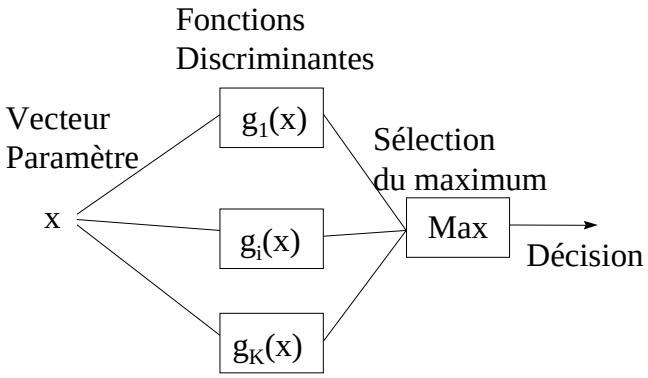


Fig. 4.3 Classifieur sous la forme de Fonctions Discriminantes

L'effet de la règle de décision est de séparer l'espace en K régions de décisions (une par classe). Les surfaces qui séparent ces régions sont appelées surfaces de décision.

2 Apprentissage Supervisé

Le gros inconvénient de la règle de décision bayésienne est de supposer connues les densités de probabilités conditionnellement à chaque classe (soit $f(x|\omega_i)$) et les probabilités d'appartenance à chaque classe $P(\omega_i)$. Dans la plupart des applications, $f(x|\omega_i)$ et $P(\omega_i)$ sont inconnues. Mais on dispose, en mode supervisé, d'une base d'apprentissage qui va nous permettre

de compenser ce manque de connaissance a priori. L'estimation des probabilités $P(\omega_i)$ est relativement simple, soit par estimation statistique, soit par la connaissance même du problème. L'estimation des densités $f(x|\omega_i)$ est un problème plus délicat. Deux approches sont envisageables :

- les **méthodes paramétriques** supposent que les densités $f(x|\omega_i)$ ont une forme paramétrique connue (par exemple Gaussienne) et le problème devient alors celui de l'estimation d'un vecteur paramètre à partir d'observations issues de chaque densité $f(x|\omega_i)$ (par exemple dans le cas où les observations sont issues de variables aléatoires réelles gaussiennes indépendantes de moyenne m et de variance σ^2, on a $\theta = (m, \sigma^2)^t$)
- les **méthodes non paramétriques** ne font aucune hypothèse sur les densités conditionnelles $f(x|\omega_i)$.

Etant en mode supervisé, le problème peut se résoudre classe par classe. Dans cette partie, on notera $f(x)$ la densité conditionnelle à la classe ω et $x_1, ..., x_n$ les n vecteurs d'apprentissage (éléments de $\mathbb{R}^p$) associés à cette classe ω.

2.1 Méthodes paramétriques

Nous nous limitons aux estimateurs très classiques obtenus à l'aide de la méthode du maximum de vraisemblance ou dérivés de l'inférence bayésienne.

2.1.1 Estimateur du Maximum de Vraisemblance

On suppose que le vecteur paramètre θ est déterministe. L'estimateur du maximum de vraisemblance s'obtient par maximisation de la fonction de vraisemblance des n vecteurs x_i, en supposant que ces vecteurs sont indépendants. Sous certaines hypothèses de régularité (voir par exemple [17]), l'estimateur du maximum de vraisemblance s'obtient en annulant les dérivées partielles de la fonction de vraisemblance par rapport à chaque paramètre θ_i.

2.2.2 Estimateurs Bayésiens

A la différence de l'estimation par maximum de vraisemblance, les estimateurs bayésiens supposent qu'on dispose d'une information a priori sur le vecteur inconnu θ résumée dans une loi a priori $f(\theta)$. Les estimateurs de Bayes les plus couramment utilisés sont l'estimateur de la moyenne a Posteriori et l'estimateur du maximum a Posteriori.

Estimateur de la moyenne a Posteriori.

Cas où $\theta \in \mathbb{R}$

L'estimateur de la moyenne a Posteriori du paramètre θ est une fonction de $x = (x_1, ..., x_n)^t$ notée $\widehat{\theta}_{MOAP}(x)$ qui minimise $E\left[(\theta - \pi(x))^2\right]$ c'est-à-dire :

$$E\left[\left(\theta - \widehat{\theta}_{MOAP}(x)\right)^2\right] = \min_{\pi} E\left[(\theta - \pi(x))^2\right] \tag{3.32}$$

On montre aisément [24] que :

$$\widehat{\theta}_{MOAP}(x) = E[\theta \,|x] \tag{3.33}$$

Pour déterminer $E[\theta \,|x]$, on opère de la façon suivante :

1) Détermination de la loi de $x_1, ..., x_n \mid \theta$
2) Détermination de la loi de $x_1, ..., x_n, \theta$
3) Détermination de la loi de $x_1, ..., x_n$
4) Détermination de la loi de $\theta \,|x_1, ..., x_n$
5) Détermination de $E[\theta \,|x_1, ..., x_n]$

Cas où $\theta \in \mathbb{R}^p$

On montre que l'estimateur $\widehat{\theta}_{MOAP}(x) = E[\theta \,|x]$ minimise le coût quadratique

$$E\left[(\theta - \pi(x))^t Q (\theta - \pi(x))\right] \tag{3.34}$$

pour toute matrice symétrique définie positive Q (et donc en particulier pour $Q = I_p$, matrice identité d'ordre p).

Estimateur du Maximum A Posteriori (estimateur MAP).

Cas où $\theta \in \mathbb{R}$

L'estimateur du maximum a Posteriori du paramètre θ est une fonction de $x = (x_1, ..., x_n)^t$ notée $\widehat{\theta}_{MAP}(x)$ qui minimise une fonction de coût "uniforme" :

$$c(\theta - \pi(x)) = \begin{cases} 0 \text{ si } |\theta - \pi(x)| \leq \frac{\Delta}{2} \\ 1 \text{ si } |\theta - \pi(x)| > \frac{\Delta}{2} \end{cases}$$

c'est-à-dire :

$$c\left(\theta - \widehat{\theta}_{MAP}(x)\right) = \min_{\pi} c(\theta - \pi(x)) \tag{3.35}$$

Un cas particulièrement intéressant est le cas où Δ est un nombre arbitrairement petit non nul. L'estimateur $\widehat{\theta}_{MAP}(x)$ vérifiant (3.35) correspond à la

valeur de $\pi(x)$ qui maximise la loi a posteriori $p(\theta|\, x)$ d'où le nom de **maximum a Posteriori**. L'estimateur du maximum a posteriori s'obtient alors en annulant la dérivée de $p(\theta|\, x)$ ou de son logarithme par rapport à θ.

Cas où $\theta \in \mathbb{R}^p$

Comme précédemment, on cherche le vecteur θ qui maximise la loi a posteriori $p(\theta|\, x)$. La recherche pratique de cet estimateur consiste à rechercher les valeurs de θ_i qui annulent les dérivées partielles de $p(\theta|\, x)$ ou de son logarithme par rapport à θ_i.

2.2 Méthodes non-paramétriques

Plusieurs méthodes d'estimation non-paramétriques sont utiles en classification. Nous présentons tout d'abord les méthodes des fenêtres de Parzen et des k plus proches voisins qui cherchent à estimer les densités $f(x)$ à l'aide des échantillons $x_1, ..., x_n$. Nous nous intéressons ensuite aux méthodes basées sur les fonctions discriminantes linéaires.

2.2.1 Estimation des densités

La probabilité pour qu'un vecteur x_i appartienne à une région R inclue dans $\mathbb{R}^p$ est

$$P_R = \int_R f(u)\, du \tag{3.36}$$

Supposons que les n vecteurs x_i soient indépendants. Le nombre de vecteurs x_i appartenant à la région R noté N_R est alors une variable aléatoire de loi Binomiale telle que :

$$P[N_R = k] = \mathcal{C}_n^k P_R^k (1 - P_R)^{n-k} \qquad k \in \{0, ..., n\} \tag{3.37}$$

On en déduit

$$\begin{array}{l} E\left(\frac{N_R}{n}\right) = P_R \\ Var\left(\frac{N_R}{n}\right) = \frac{P_R(1-P_R)}{n} \end{array} \tag{3.38}$$

N_R/n est donc un estimateur non biaisé et convergent du paramètre P_R. Supposons que $f(u)$ soit une fonction continue sur R contenant x et que R soit une région suffisamment petite pour que l'on puisse considérer que f est constante sur R. On a alors :

$$\int_R f(u)\, du \simeq f(x)\, V_R \tag{3.39}$$

V_R étant la longueur ($p = 1$), la surface ($p = 2$), le volume ($p = 3$) ... de la région R. On peut alors définir un estimateur de la densité de probabilité $f(x)$ par :

$$\widehat{f}(x) = \frac{N_R}{nV_R} \tag{3.40}$$

On notera que dans le cas $p = 1$, on retrouve la définition de l'histogramme en choisissant pour R un intervalle centré autour de x.

Remarques

- Si on fixe le volume V_R et que le nombre de données n augmente, N_R/n converge vers P_R mais $\frac{N_R}{nV_R}$ converge vers la valeur moyenne de f sur R et non pas vers $f(x)$.
- Si on diminue le volume V (à n fixé), la région R peut ne contenir aucun échantillon, ce qui donne $\widehat{f}(x) \simeq 0$.

Une idée naturelle est alors de faire varier n et R simultanément. On considère alors une suite de régions R_k contenant x et possèdant N_k échantillons. Soit V_k le "volume" de la région R_k et $\widehat{f_k}(x) = \frac{N_k}{kV_k}$ la $k^{ème}$ estimation de $f(x)$. Comment définir les régions R_k pour que $\widehat{f_k}(x)$ soit un bon estimateur de $f(x)$? Pour assurer la convergence de $\widehat{f_k}(x)$ vers $f(x)$ lorsque $k \to \infty$, il suffit de satisfaire les trois propriétés suivantes ([8], p. 115) :

$$\begin{aligned} \lim_{k\to\infty} V_k &= 0 \\ \lim_{k\to\infty} N_k &= \infty \\ \lim_{k\to\infty} \frac{N_k}{k} &= 0 \end{aligned}$$

Le problème est maintenant de fixer la suite (N_k, V_k). On trouve essentiellement deux méthodes dans la littérature : la **méthode des fenêtres de Parzen** et **la méthode des k plus proches voisins**.

2.2.2 Méthode des fenêtres de Parzen

L'idée des fenêtres de Parzen est d'obtenir une estimée de la densité de probabilité f par la superposition d'une fonction appelée noyau centrée sur x. En choisissant comme région R_k un hypercube de coté h_k et de dimension p, on obtient $V_k = h_k^p$ et :

$$N_k = \sum_{i=1}^{k} \phi\left(\frac{x - x_i}{h_k}\right) \tag{3.41}$$

avec

$$\begin{array}{l} \phi(u) = 1 \text{ si } |u_j| \leq 1/2 \qquad \forall j \in \{1, ..., p\} \\ \phi(u) = 0 \text{ sinon} \end{array} \tag{3.42}$$

En effet, $\phi\left(\frac{x-x_i}{h_k}\right) = 1$ si x_i est dans la région R_k. On a alors :

$$\widehat{f}_k(x) = \frac{1}{kh_k^p} \sum_{i=1}^{k} \phi\left(\frac{x - x_i}{h_k}\right) \tag{3.43}$$

Examinons l'effet du paramètre h_k

On pose :

$$\delta_k(x) = \frac{1}{V_k} \phi\left(\frac{x}{h_k}\right) \tag{3.44}$$

- Lorsque h_k est grand, $\delta_k(x)$ est une fonction variant lentement. Il faut que x_i soit très éloigné de x pour que $\delta_k(x - x_i)$ différe de $\delta_k(0)$. Dans ce cas, $\widehat{f}_k(x)$ est la superposition de k fonctions à variations lentes : $\widehat{f}_k(x)$ estime $f(x)$ avec peu de résolution.
- Lorsque h_k est petit, $\delta_k(x)$ est une fonction variant rapidement. Dans ce cas, $\widehat{f}_k(x)$ est la superposition de k fonctions étroites centrées sur les échantillons : $\widehat{f}_k(x)$ est une estimation bruitée de $f(x)$.

On peut généraliser cette approche en prenant d'autres noyaux ϕ et d'autres volumes V_k. Pour satisfaire les propriétés On montre que $\widehat{f}_k(x)$ est une densité de probabilité si ϕ vérifie les propriétés suivantes :

$$\begin{array}{l} \phi(u) \geq 0 \qquad \forall u \\ \int_{\mathbb{R}^p} \phi(u)\, du = 1 \end{array} \tag{3.45}$$

Pour que $\widehat{f}_k(x)$ ainsi défini soit un estimateur non biaisé et convergent de $f(x)$, il faut et il suffit que ϕ vérifie les propriétés suivantes [9][8] :

$$\sup_u \phi(u) < +\infty \tag{3.46}$$

$$\lim_{h_k \to \infty} \frac{1}{V_k} \phi\left(\frac{u}{h_k}\right) = \delta(u) \tag{3.47}$$

$$\lim_{k \to \infty} V_k = 0 \tag{3.48}$$

$$\lim_{k \to \infty} kV_k = +\infty \tag{3.49}$$

Pour satisfaire les deux dernières propriétés, on peut choisir

$$V_k = \frac{V_1}{\sqrt{k}} \text{ ou } V_k = \frac{V_1}{Lnk}$$

Le choix d'un noyau ϕ se fait généralement parmi un catalogue de fonctions prédéfinies (voir [8], p. 120 pour quelques exemples). La méthode des fenêtres de Parzen s'applique bien lorsque le nombre de variables p est faible. Par contre, lorsque p est important, les calculs deviennent vite très importants.

2.2.3 Méthode des k plus proches voisins

Nous avons vu l'importance et la difficulté du choix des régions R_k dans la méthode des fenêtres de Parzen. Une autre méthode consiste à laisser les régions R_k être fonction des données plutôt que des fonctions arbitraires du nombre d'échantillons. Par exemple, on peut centrer la région sur x et augmenter le volume jusqu'à ce qu'il contienne N_k échantillons, N_k étant une fonction spécifique des données (par exemple on peut choisir N_k comme la partie entière de $\sqrt{k}$). Les N_k échantillons sont alors les N_k plus proches voisins de x. La région R_k s'ajuste à la valeur de la densité de probabilité au point x considéré. Comme pour la méthode des fenêtres de Parzen, l'estimateur de $f(x)$ est défini par :

$$\widehat{f}_k(x) = \frac{N_k}{kV_k} \tag{3.50}$$

On montre qu'une condition nécessaire et suffisante pour que $\widehat{f}_k(x)$ converge vers $f(x)$ est :

$$\lim_{k\to\infty} N_k = +\infty \tag{3.51}$$

$$\lim_{k\to\infty} \frac{N_k}{k} = 0 \tag{3.52}$$

2.2.4 Règle du plus proche voisin (1PPV)

Définition

Soit $X_n = \{x_1, ..., x_n\}$ l'ensemble des vecteurs dont les classes sont connues. Si x est l'élément à classer (de classe inconnue), x'_n son plus proche voisin dans X_n au sens d'une certaine distance et $\omega'_n \in \{\omega_1, ..., \omega_K\}$ la classe de x'_n. La règle du plus proche voisin consiste à classer x dans la classe ω'_n. **L'élément x est affecté à la classe de son plus proche voisin au sens d'une certaine métrique.**

*) *Justification*

Considérons la règle de Bayes pour une fonction de coût $1-\delta_{ij}$:

$$d(x)=a_j \Leftrightarrow P(\omega_j|x) \geq P(\omega_k|x) \qquad \forall k \in \{1,...,K\} \tag{3.53}$$

c'est-à-dire :

$$d(x)=a_j \Leftrightarrow f(x|\omega_j)\,P(\omega_j) \geq f(x|\omega_k)\,P(\omega_k) \qquad \forall k \in \{1,...,K\} \tag{3.54}$$

Si on estime les densités $f(x|\omega_j)$ à l'aide de la méthode du plus proche voisin et que les probabilités $P(\omega_j)$ sont estimées par la proportion d'éléments de la base d'apprentissage appartenant à ω_j, on a :

$$d(x)=a_j \Leftrightarrow \frac{N_j}{n_j V_j}\frac{n_j}{n} \geq \frac{N_k}{n_k V_k}\frac{n_k}{n} \qquad \forall k \in \{1,...,K\} \tag{3.55}$$

Pour la règle du plus proche voisin, on a $N_j=N_k=1$, soit :

$$d(x)=a_j \Leftrightarrow V_k \geq V_j \qquad \forall k \in \{1,...,K\} \tag{3.56}$$

ce qui signifie que le plus proche voisin de x est dans la classe ω_j.

*) *Convergence*

On suppose que la densité $f(x)$ est continue au point x. La probabilité d'avoir un vecteur x_i dans une région R centrée sur x est $P_R=\int_R f(u)\,du$. La probabilité pour que les n vecteurs $x_1,...,x_n$ soient à l'extérieur de S est $(1-P_R)^n$ qui tend vers zéro lorsque $n \to +\infty$. Donc

$$\lim_{n\to\infty} P[x'_n \notin R] = \lim_{n\to\infty} P[x_1 \notin R,...,x_n \notin R] = 0 \tag{3.57}$$

Par conséquent :

$$\forall \varepsilon > 0 \qquad \lim_{n\to\infty} P[\|x'_n - x\| > \varepsilon] = 0 \tag{3.58}$$

L'équation (3.58) montre que x'_n converge en probabilité vers x.

Soit $f_{x'_n}(u|x)$ la densité de probabilité de x'_n conditionnellement à x. D'après (3.57), quelle que soit la région R entourant x :

$$\lim_{n\to\infty} \int_R f_{x'_n}(u|x)\,du = 1 = \lim_{n\to\infty} \int_R \delta(u-x)\,du \tag{3.59}$$

c'est-à-dire :

$$\lim_{n\to\infty} \int_R \left[f_{x'_n}(u|x) - \delta(u-x)\right] du = 0 \tag{3.60}$$

On en déduit :

$$\lim_{n\to\infty} f_{x'_n}(u|x) = \delta(u-x) \tag{3.61}$$

ce qui signifie que **la densité du plus proche voisin de x converge vers** $\delta(u-x)$.

*) *Erreur de la règle du PPV*

On montre en annexe que la probabilité asymptotique ($n \to \infty$) de la règle du 1PPV est définie par :

$$P_1 = \lim_{n\to\infty} P_n = \int \left[1 - \sum_{i=1}^{K} P^2\left(\omega = \omega_i | \, x\right)\right] f\left(x\right) dx \tag{3.62}$$

Cette égalité peut aussi s'écrire :

$$P_1 = E\left[1 - \sum_{i=1}^{K} P^2\left(\omega = \omega_i | \, X\right)\right] \tag{3.63}$$

*) *Bornes de l'erreur [3]*

L'expression (3.62) permet de borner l'erreur de la règle du plus proche voisin. Si P^* désigne la probabilité d'erreur minimale obtenue à l'aide de la règle de décision Bayésienne (dans le cas d'une fonction de coût $c_{ij} = 1 - \delta_{ij}$), on montre en annexe l'inégalité de Cover et Hart :

$$P^* \leq P_1 \leq P^*\left(2 - \frac{K}{K-1}P^*\right) \tag{3.64}$$

Puisque $P^*\left(2 - \frac{K}{K-1}P^*\right) \leq 2P^*$, l'inégalité de Cover et Hart montre qur la probabilité asymptotique d'erreur de la règle du 1PPV est au pire deux fois la probabilité d'erreur optimale de la règle de Bayes. Cette inégalité permet aussi de conclure que $P_1 = P^*$ dans les deux cas limites $P^* = 0$, et $P^* = \frac{K-1}{K}$ (voir partiel du 26 Mars 1999).

2.2.5 Règle des k plus proches voisins

Définition

La règle de classification du plus proche voisin peut se généraliser à plusieurs voisins. **On affectera alors x à la classe majoritairement représentée par ses k plus proches voisins, au sens d'une certaine distance.**

*) *Choix de k*

Lorsque k est grand, on considère un grand nombre de voisins et donc intuitivement la classification semble meilleure. En fait, ceci n'est vrai que pour une base d'apprentissage assez importante autour du point x à classer. En effet, si la base d'apprentissage n'est pas suffisamment importante, certains voisins de x appartiendront à des classes différentes de celle de x et donc la classification sera moins bonne. On peut montrer que la performance **asymptotique** (en terme de probabilité d'erreur) de la règle des $kPPV$ est

meilleure que celle obtenue avec la règle du *1PPV [6]*. De nombreux auteurs se sont intéressés à déterminer des bornes sur la probabilité d'erreur asymptotique de la règle des $kPPV$ notée P_k. Par exemple, dans le cas de deux classes, on a [6] :

$$P^* \leq P_k \leq P^* + \frac{1}{\sqrt{ke}} \text{ ou } P^* \leq P_k \leq P^* + \sqrt{\frac{2P_1}{k}}$$

La première inégalité montre que la probabilité d'erreur asymptotique de la règle des $kPPV$ est en général proche de la probabilité d'erreur du classifieur Bayésien et que

$$\lim_{k \to \infty} P_k = P^*$$

Lorsque P^* est faible, on a les approximations suivantes :

$$P_1 \approx 2P^* \text{ et } P_3 \approx P^* + 3\left(P^*\right)^2$$

Les résultats précédents concernent la probabilité d'erreur asymptotique de la règle des $kPPV$. En pratique, lorsque le nombre d'échantillons n est fini, le choix de k reste souvent arbitraire et on prend en général k relativement faible par rapport au nombre de points N.

Remarques

- pour la règle du *1PPV* ($k = 1$), les n vecteurs x_i définissent une partition de l'espace en N régions R_i telles que dans R_i, le plus proche voisin de x est x_i. Cette partition porte le nom de partition de Voronoi.

- il existe de nombreuses règles de calcul rapide des plus proches voisins dont certaines sont détaillées dans [8].

2) Ambiguïté

Lorsque k est multiple du nombre de classes, il peut y avoir ambiguïté sur le choix de la classe. Par exemple, comment procède-t-on lorsque $k = 2$, que l'un des deux voisins est dans la classe ω_1 et que l'autre est dans la classe ω_2 ? On peut lever l'indétermination en calculant les distances entre x et ses plus proches voisins. On choisira la classe pour laquelle la distance (ou la somme des distances) est minimale. Lorsqu'une classe est plus largement représentée dans la base d'apprentissage, cette classe sera plus souvent choisie avec les méthode des $kPPV$. On peut alors pondérer la distance utilisée en tenant compte de cette disproportion.

3 Apprentissage Non Supervisé

On suppose dans cette partie qu'une base d'apprentissage constituée de N vecteurs de données de $\mathbb{R}^p$ est disponible mais qu'on ne connaît pas la classe associée à ces vecteurs. Le problème de classification en mode non supervisé consiste à regrouper ces vecteurs en nuages de points (clusters), chaque nuage de points correspondant à une classe. Les méthodes de classification effectuant ce regroupement sont connues sous le nom de méthodes de coalescence (clustering methods). On peut se demander le véritable intérêt de ces méthodes dépourvues d'apprentissage supervisé. Duda [9] justifie leur utilité de la façon suivante :

- l'apprentissage supervisé est un processus couteux puisqu'on doit étiqueter et stocker tous les vecteurs de données. Dans certaines applications ou la base d'apprentissage est trop importante, on peut imaginer de construire le classifieur à l'aide d'un apprentissage supervisé sur une partie des données, puis de continuer l'apprentissage avec des données non étiquetées en espérant que ces données diffèrent peu de celles de l'apprentissage supervisé. Un exemple d'application est la reconnnaissance de caractères. On peut imaginer de construire le classifieur à l'aide d'une séquence d'apprentissage étiquetée (la classe associée à chaque caractère est connue) puis de modifier le classifieur à l'aide des décisions prises sur les caractères de classe inconnue. Cette procédure évite de faire un apprentissage supervisé avec un trop grand nombre de caractères et le fait de prendre en compte les caractères de classe inconnue rend le classifieur plus robuste,

- l'apprentissage non supervisé peut être également très utile dans les applications ou les formes évoluent au cours du temps. Par exemple, si on effectue un apprentissage supervisé avec deux classes, les décisions seront prises en faveur de l'une de ces deux classes, même si les formes d'une troisième classe apparaissent. Un apprentissage non supervisé permet de résoudre ce problème,

- enfin, un apprentissage non supervisé peut nous informer sur la structure des données. Une tel apprentissage peut par exemple nous aider à reconnaître des vecteurs possèdant des caractéristiques très éloignées de la plupart des vecteurs de données.

Etant donné un ensemble d'apprentissage $X = \{x_1, ..., x_N\}$, le problème des méthodes de coalescence consiste à répartir les N vecteurs de données en K groupes de points ω_i, c'est-à-dire à réaliser une partition de X en K

sous-ensembles ω_i. On distingue plusieurs catégories de méthodes de coalescence parmi lesquelles les méthodes d'optimisation et les méthodes hiérarchiques. Les méthodes d'optimisation constituent la partition de X en minimisant un critère de coût approprié. Les méthodes hiérarchiques réalisent une arborescence dont les sommets sont les vecteurs observés et la base est l'ensemble complet des observations (hièrarchie ascendante) ou dont le sommet est l'ensemble complet des observations et la base est constituée des observations (hiérarchie descendante).

3.1 Méthodes d'optimisation

Le problème peut paraître simple puisqu'il suffit de définir un critère de dispersion et de déterminer la partition de X qui minimise ce critère. On montre que le nombre de partitions de X en K sous ensembles vérifie (**??**, p. 384) :

$$P\left(N,K\right)=\frac{1}{K!}\sum_{i=0}^{K}i^{N}\left(-1\right)^{K-i}C_{K}^{i}\qquad K<N$$

avec $P(N,1)=1, P(N,N)=1$ et $P(N,K)=0$ pour $K>N$. Une recherche exhaustive de la partition optimale pose vite problème car par exemple $P(100,5)\approx 10^{68}$. Les méthodes de coalescence recherchent donc une partition intéressante parmi un sous ensemble de toutes les partitions possibles.

3.1.1 Partition d'erreur quadratique minimale

Erreur quadratique d'une Partition

Définir une partition de $X=\{x_1,...,x_N\}$ en K classes $\omega_1,...,\omega_K$ consiste à affecter chaque vecteur x_k à une classe ω_i. Si g_i est le centre de gravité de la $i^{ème}$ classe ω_i contenant N_i vecteurs, on définit l'erreur quadratique de la $i^{ème}$ classe par

$$E_i^2=\sum_{k=1}^{N_i}d^2\left(x_k,g_i\right) \tag{3.65}$$

L'erreur quadratique intraclasse de la partition de $X=\{x_1,...,x_N\}$ en K classes est alors définie par :

$$E^2=\sum_{i=1}^{K}E_i^2$$

On vérifie aisément que pour tout point y on a :

$$\sum_{k=1}^{N_i} d^2(x_k, y) = \sum_{k=1}^{N_i} d^2(x_k, g_i) + N_i d^2(g_i, y) \tag{3.66}$$

En particulier, l'erreur quadratique des données calculée par rapport au centre de gravité g vérifie :

$$\sum_{i=1}^{K} \sum_{k=1}^{N_i} d^2(x_k, g) = E^2 + \sum_{i=1}^{K} N_i d^2(g_i, g) \tag{3.67}$$

On dit que l'erreur quadratique de X est la somme de l'erreur quadratique intraclasse et de l'erreur quadratique interclasse. Une bonne classification doit minimiser l'erreur quadratique intraclasse E^2 ou de manière équivalente maximiser le moment interclasse $\sum_{i=1}^{K} N_i d^2(g_i, g)$. Comme nous l'avons souligné précédemment, une recherche exhaustive de la partition qui minimise E^2 n'est pas possible. En pratique, on recherche donc une partition associée à un minimum local de E^2. L'algorithme le plus connu permettant de déterminer cette partition est l'algorithme ISODATA (appelé également K-means algorithm).

Algorithme ISODATA

L'algorithme consiste à construire une partition de l'ensemble X en un certain nombre de classes. On effectue les opérations suivantes :

1) on se fixe au départ le nombre de classes et les barycentres de ces classes. Ce choix initial peut se faire de manière aléatoire ou en s'inspirant d'une connaissance a priori des objets à classer,

2) on affecte le vecteur x_i à la classe ω_j telle que $d(x_i, g_j) = \inf_k d(x_i, g_k)$ au sens d'une distance à définir. En général, on utilise la distance euclidienne ou la distance de Mahalanobis (mais cette dernière engendre un coût calculatoire plus important puisqu'on doit calculer la matrice de covariance à chaque étape),

3) on détermine le centre de gravité g_k^* de chaque nouvelle classe ainsi formée ω_k^*,

4) on répète les étapes 2) et 3) tant que surviennent des modifications dans la composition des classes.

Dans une version améliorée de l'algorithme ISODATA, on rajoute une cinquième étape permettant de modifier le nombre de classes de manière à l'adapter aux données, de façon à ne pas avoir de classes trop vides ou trop riches :

5) deux classes sont regroupées si leurs centres de gravité sont proches, une classe est éclatée si elle contient un nombre trop important de vecteurs x_i ou si l'erreur quadratique associée est trop grande.

Convergence de l'algorithme

Examinons ce que devient l'erreur quadratique

$$E^2 = \sum_{i=1}^{K} E_i^2 = \sum_{i=1}^{K} \sum_{k=1}^{N_i} d^2 (x_k, g_i) \tag{3.68}$$

au cours de l'algorithme. Après la phase de réaffectation des vecteurs (étape 2), la classe ω_i qui possèdait N_i vecteurs est devenue une classe ω_i^* possédant N_i^* vecteurs. L'erreur quadratique de la classe ω_i^* calculée par rapport à l'ancien centre de gravité g_i est :

$$F^2 = \sum_{i=1}^{K} \sum_{k=1}^{N_i^*} d^2 (x_k, g_i) \tag{3.69}$$

Soit x_k un élément de ω_i^*. Si x_k n'a pas changé de classe, sa contribution à l'erreur quadratique reste la même. Mais si x_k a changé de classe et qu'après réaffectation il passe dans ω_i^*, c'est que :

$$d^2 (x_k, g_i) < d^2 (x_k, g_j) \qquad \forall j \tag{3.70}$$

La contribution de l'élément x_k à F_K^2 est donc plus faible que la contribution de x_k à E_K^2. D'où

$$F^2 \leq E^2 \tag{3.71}$$

Considérons les classes ω_k^* avec leurs nouveaux centres de gravité g_k^*. D'après l' équation (3.66), les erreurs quadratiques calculées par rapport au centre de gravité g_k^* sont inférieures aux erreurs calculées par rapport à g_k. Donc :

$$E_*^2 = \sum_{i=1}^{K} \sum_{k=1}^{N_i^*} d^2 (x_k, g_i^*) \leq F^2 \tag{3.72}$$

On voit donc qu'à chaque itération, l'erreur quadratique diminue. La suite des erreurs quadratique est une suite décroissante de nombres positifs qui est donc convergente. Puisque le nombre de partitions possibles est fini, la suite atteint sa limite après un nombre fini d'itérations. Un critère d'arrêt de l'algorithme peut être la comparaison à un seuil de :

$$\left| \frac{E_{n+1}^2 - E_n^2}{E_{n+1}^2} \right|$$

Remarques

- L'inconvénient majeur de la méthode est qu'on n'a pas la certitude d'obtenir la meilleure solution. En effet, la partition obtenue après convergence de l'algorithme dépend des K centres choisis initialement. L'un des moyens pour avoir des résultats valables est d'exécuter l'algorithme plusieurs fois avec différentes partitions initiales. On retient alors la partition qui minimise l'erreur quadratique. On peut aussi constituer des classes avec les vecteurs qui restent groupés quelles que soient les valeurs initiales de centres (Dubuisson ([8]) parle alors de formes fortes). Certains vecteurs peuvent ne pas être classés suite à une indécision,

- Le critère de l'erreur quadratique est bien adapté aux nuages de points compacts. L'algorithme ISODATA pose donc problème pour des nuages de points circulaires ([32], p. 449), rectilignes ou plus généralement non compacts. On peut alors dans ce cas définir des paramètres et des mesures de dispersion adaptés à la forme des nuages de points. L'expérience montre que l'algorithme pose également problème lorsque le nombre de vecteurs d'apprentissage diffère fortement d'une classe à une autre ([9], p. 220), ([32], p. 484),

- Comme variante de cet algorithme, on peut calculer les distances à plusieurs représentants de chaque classe (et pas uniquement au centre de gravité). Cette variante a un avantage lorsque le centre de gravité d'une classe est dans une zone de faible densité intermédiaire entre deux zones denses ou lorsque les classes sont non-convexes (cf l'avantage de la règle des kPPV par rapport à la règle de la distance au barycentre).

3.1.2 Apprentissage d'un mélange de lois

On suppose que chaque classe est décrite par une loi conditionnelle notée $f\left(x|\,\theta_i\right)$ dont la forme est connue et dépend d'un vecteur paramètre inconnu θ_i. On suppose que les probabilités a priori de chaque classe $P\left(\omega_i\right)$ sont inconnues et que le nombre de classes K est connu. La loi d'un vecteur x de l'apprentissage est alors définie par :

$$f(x|\,\theta,P)=\sum_{i=1}^{K}P\left(\omega_i\right)f\left(x|\,\theta_i\right)$$

avec $\theta=\left(\theta_1^t,...,\theta_K^t\right)^t$ et $P=\left(P\left(\omega_1\right),...,P\left(\omega_K\right)\right)^t$. La densité $f(x|\,\theta,P)$ est donc un mélange de densités dont les paramètres du mélange sont $P\left(\omega_i\right)$.

L'apprentissage de ce mélange de lois consiste à utiliser les vecteurs de données x_k distribués suivant $f(x|\,\theta, P)$ pour estimer les vecteurs θ et P. On suppose que la densité du mélange est identifiable (on pourra par exemple se réfèrer à ([9], p. 191) pour un exemple de mélange de lois non-identifiable) c'est-à-dire que

$$\theta \neq \theta' \Rightarrow \exists x/ \quad f(x|\,\theta) \neq f(x|\,\theta')$$

En supposant que les vecteurs de l'apprentissage sont indépendants, les estimateurs du maximum de vraisemblance de θ et de P se déterminent en maximisant le logarithme de la fonction de vraisemblance :

$$L(\theta, P) = \sum_{k=1}^{N} \ln \left[\sum_{i=1}^{K} P(\omega_i)\, f(x_k|\,\theta_i) \right]$$

La maximisation de $L(\theta, P)$ par rapport à θ et P conduit généralement à un système d'équations non-linéaires difficiles à étudier même dans le cas gaussien. Une solution itérative basée sur l'algorithme EM (Expectation/Maximization) peut alors être envisagée ([32], p. 444). Cette solution coincide avec l'algorithme ISODATA dans le cas gaussien où toutes les matrices de covariance sont égales et connues ou en en utilisant l'approximation

$$P(\omega_i|\,x_k) \approx \begin{cases} 1 \text{ si } i = p \\ 0 \text{ sinon} \end{cases}$$

où $\widehat{g}_p$ est le barycentre le plus proche de x_k ([9], p. 201).

Remarque

L'algorithme d'apprentissage tel qu'il a été présenté est un algorithme non-supervisé paramétrique. On peut également mettre en oeuvre des algorithmes non-supervisés non-paramétriques qui, par exemple, estiment la densité du mélange à l'aide de la méthode des fenêtres de Parzen. Ces algorithmes sont cependant peu utilisés car ils engendrent un fort coût calculatoire.

3.1.3 Algorithmes de coalescence floue

Une partition floue de $X = \{x_1, ..., x_N\}$ en K classes $\omega_1, ..., \omega_K$ est définie par un ensemble de K fonctions u_j définies de X dans $A = [0, 1]$. $u_j(x_i)$ représente le degré d'appartenance de x_i à la classe ω_j de sorte que

$$\sum_{j=1}^{K} u_{ij} = 1, \quad \forall i = 1, ..., N \tag{3.73}$$

On note U la matrice à N lignes et K colonnes constituée des éléments u_{ij}. On suppose que chaque classe est paramètrée par un vecteur de $\mathbb{R}^p$ noté θ_i et on pose $\theta = \left(\theta_1^t, ..., \theta_K^t\right)^t$. La plupart des algorithmes de coalescence floue cherchent à minimiser la fonction de coût

$$J_q(\theta, U) = \sum_{i=1}^{N} \sum_{j=1}^{K} u_{ij}^q d(x_i, \theta_j)$$

sous les N contraintes définies par (8.1), d étant un indice de proximité (généralement une distance) défini sur $\mathbb{R}^p$.

Remarques

- **Classifieur flou dégénéré** : lorsque $A = \{0, 1\}$, la contrainte (8.1) implique que pour chaque indice i, un seul des termes u_{ij} vaut 1 et tous les autres valent 0. En d'autres termes, x_i est affecté à une classe ω_k telle que $u_{ik} = 1$ et $u_{ij} = 0, \forall j \neq k$. Dans ce cas, la fonction de coût $J_q(\theta, U)$ est indépendante de q,

- lorsque $q = 1$, le classifieur dégénéré est optimal par rapport à tous les classifieurs flous. Par contre, lorsque $q \neq 1$, le classifieur dégénéré n'est généralement pas optimal ([32], p. 454).

Minimisation de $J_q(\theta, U)$

Le lagrangien associé à la minimisation de $J_q(\theta, U)$ sous les contraintes (8.1) est défini par :

$$L_q(\theta, U) = J_q(\theta, U) - \sum_{i=1}^{N} \lambda_i \left(\sum_{j=1}^{K} u_{ij} - 1 \right)$$

En annulant la dérivée partielle de $L_q(\theta, U)$ par rapport à u_{ij}, on obtient :

$$u_{ij} = \left[\frac{\lambda_i}{q d(x_i, \theta_j)} \right]^{\frac{1}{q-1}}, \quad \forall j = 1, ..., K \tag{3.74}$$

La contrainte $\sum_{j=1}^{K} u_{ij} = 1$ conduit alors à :

$$\lambda_i = q \left\{ \sum_{k=1}^{K} \left(\frac{1}{d(x_i, \theta_k)} \right)^{\frac{1}{q-1}} \right\}^{1-q} \tag{3.75}$$

En combinant les équations (3.74) et (3.75, on obtient :

$$u_{ij} = \left[\sum_{k=1}^{K} \left(\frac{d\left(x_i, \theta_j\right)}{d\left(x_i, \theta_k\right)} \right)^{\frac{1}{q-1}} \right]^{-1} \tag{3.76}$$

L'annulation de la dérivée partielle de $L_q\left(\theta, U\right)$ par rapport à θ_j conduit à :

$$\sum_{i=1}^{N} u_{ij}^q \frac{\partial d\left(x_i, \theta_j\right)}{\partial \theta_j} = 0, \quad \forall j = 1, ..., K$$

Dans le cas $d(x_i, \theta_j)$ est le carré de la distance euclidienne, i.e. $d(x_i, \theta_j) = \left(x_i - \theta_j\right)^t \left(x_i - \theta_j\right)$, on obtient

$$\frac{\partial d\left(x_i, \theta_j\right)}{\partial \theta_j} = 2\left(\theta_j - x_i\right)$$

d'où

$$\theta_j = \frac{\sum_{i=1}^{N} u_{ij}^q x_i}{\sum_{i=1}^{N} u_{ij}^q} \tag{3.77}$$

Malheureusement, le système d'équations non-linéaires (8.2) et (3.77) ne conduit pas à une expression explicite de u_{ij} et θ_j. En pratique, on utilise l'algorithme itératif suivant :

- Initialisation de θ_j, pour $j = 1, ..., K$,
- $n = 0$
- Répéter
  - pour $i = 1$ à N
    - pour $j = 1$ à K
      $u_{ij}(n) = \left[\sum_{k=1}^{K} \left(\frac{d(x_i, \theta_j(n))}{d(x_i, \theta_k(n))} \right)^{\frac{1}{q-1}} \right]^{-1}$
    - fin
  - fin
  - $n = n + 1$
  - pour $j = 1$ à K
    $\theta_j(n) = \frac{\sum_{i=1}^{N} u_{ij}^q (n-1) x_i}{\sum_{i=1}^{N} u_{ij}^q (n-1)}$

- tant que $\|\theta(n) - \theta(n-1)\| < \varepsilon$.

Cet algorithme est une version floue de l'algorithme ISODATA appelée *fuzzy K-means algorithm* ou *fuzzy c-means algorithm*.

Remarques

- L'algorithme ISODATA flou est bien adapté aux nuages de points compacts. Cependant, on trouve des versions modifiées de cet algorithme adaptées à d'autres formes de nuages de points comme des éllipsoides ou des hyperplans ([32]),
- Dans le cas où la distance d est appropriée (par exemple la distance de Mahalanobis), soit l'algorithme converge vers un minimum local de $J_q(\theta, U)$ en un nombre fini d'itérations, soit une sous suite d'itérées de l'algorithme converge vers un minimum local de $J_q(\theta, U)$.

Un cas particulier important : le classifieur flou dégénéré

Dans cette partie, on suppose que chaque élément x_i est affecté à une seule classe et une seule c'est-à-dire :

$$\begin{aligned} u_{ij} &= 1 \\ u_{ik} &= 0, \forall k \neq j \end{aligned}$$

Ce problème peut être vu comme un cas particulier de l'algorithme flou. Rappelons la forme du critère à minimiser :

$$J(\theta, U) = \sum_{i=1}^{N} \left[\sum_{j=1}^{K} u_{ij} d(x_i, \theta_j) \right] = \sum_{i=1}^{N} J(\theta, U_i)$$

Pour minimiser $J(\theta, U_i)$, il suffit de choisir pour k la valeur de l'indice j qui minimise $d(x_i, \theta_j)$, c'est-à-dire le numéro de la classe dont le barycentre est le plus proche de x_i :

$$u_{ij} = \begin{cases} 1 \text{ si } d(x_i, \theta_j) \leq d(x_i, \theta_k) & \forall k = 1, ..., K \\ 0 \text{ sinon} \end{cases}$$

L'application de l'algorithme ISODATA flou conduit à :

- Initialisation de θ_j, pour $j = 1, ..., K$,
- $n = 0$

- Répéter
  - pour $i = 1$ à N
    - pour $j = 1$ à K
      $u_{ij}(n) = \begin{cases} 1 \text{ si } d\left(x_i, \theta_j(n)\right) \leq d\left(x_i, \theta_k(n)\right) & \forall k = 1, ..., K \\ 0 \text{ sinon} \end{cases}$
    - fin
  - fin
  - $n = n + 1$
  - pour $j = 1$ à K
    calculer $\theta_j(n)$ comme la moyenne des éléments x_i tels que $u_{ij}(n-1) = 1$
- tant que $\theta(n) \neq \theta(n-1)$.

On retrouve l'algorithme ISODATA exposé précédemment qui recherche la partition d'erreur quadratique minimale.

3.2 Classification Hiérarchique

3.2.1 Hiérarchie ascendante basée sur des mesures de dissimilarité

Méthode des distances

On calcule les distances entre les vecteurs de données pris deux à deux. On procède alors par étapes successives, chacune d'elles consistant à regrouper les vecteurs les plus proches au sens d'une certaine distance. A la fin de chaque étape, on calcule les distances entre chaque groupe qui vient d'être créé et les autres objets (vecteurs ou groupes). On itère le processus jusqu'à ce que tous les vecteurs soient réunis dans un seul groupe. La seule difficulté de ce processus réside dans le choix d'une formule pour le calcul des distances entre groupes. On peut utiliser les distances inter-groupes suivantes :

- Distance *Min* (algorithme du minimum ou single linkage algorithm)

$$d(X_i, X_j) = \min d(x, y) \qquad x \in X_i, y \in X_j$$

  Cette distance favorise la formation de classes allongées ([9], p. 233).

- Distance *Max* (algorithme du maximum ou complete linkage algorithm)

$$d(X_i, X_j) = \max d(x, y) \qquad x \in X_i, y \in X_j$$

- Distance *Moyenne* (average linkage algorithm)

$$d(X_i, X_j) = \frac{1}{N_i N_j} \sum_{x \in X_i, y \in X_j} d(x, y)$$

- Distance entre les moyennes

$$d(X_i, X_j) = d\left(g_i, g_j\right)$$

où X_i et X_j sont deux groupes de cardinaux N_i et N_j, de centres de gravité g_i et g_j.

On donne alors une représentation sous forme d'arbre dont les noeuds représentent les fusions successives. Lorsque la longueur des branches de l'arbre est la distance entre les deux groupes fusionnés, on parle d'**hiérarchie indicée**.

Méthode des moments d'ordre 2

Dans le cas de deux classes, l'équation (3.67) s'écrit :

$$\sum_{i=1}^{2} \sum_{k=1}^{N_i} d^2\left(x_k, g\right) = E^2 + N_1 d^2\left(g_1, g\right) + N_2 d^2\left(g_2, g\right) \tag{3.78}$$

avec $E^2 = E_1^2 + E_2^2$. On montre aisément que

$$N_1 d^2\left(g, g_1\right) + N_2 d^2\left(g, g_2\right) = \frac{N_1 N_2}{N_1 + N_2} d^2\left(g_1, g_2\right) \tag{3.79}$$

Cette expression représente l'augmentation du moment intraclasse lorsqu'on fusionne les classes ω_1 et ω_2. A chaque étape de la méthode des moments d'ordre 2, on fusionne les classes qui provoquent la plus faible augmentation du moment intraclasse. On peut se ramener à l'algorithme de la méthode des distances en remplaçant la distance par la pseudo-distance suivante (qui ne vérifie pas l'inégalité triangulaire) :

$$pd\left(\omega_1, \omega_2\right) = \frac{N_1 N_2}{N_1 + N_2} d^2\left(g_1, g_2\right) \tag{3.80}$$

Cette pseudo-distance fait intervenir les centres de gravité de chacune des classes. On ne peut donc pas aisément calculer les pseudo-distances d'une itération en fonction de celles des itérations précédentes. Pour éviter ce problème, on peut utiliser la formule suivante :

$$pd\left(i \cup j, k\right) = \frac{\left(N_i + N_k\right) pd(i,k) + \left(N_j + N_k\right) pd(j,k) - N_k pd(i,j)}{N_i + N_j + N_k} \tag{3.81}$$

3.2.2 Hiérarchie descendante

A l'inverse de la classification ascendante, la classification descendante procède par dichotomies ou scissions successives. A chaque étape de l'algorithme, il y a deux règles à appliquer pour déterminer

* le choix de la classe à scinder

* le mode d'affectation des vecteurs à chacune des classes

La solution qui semble la plus justifiée est celle de Fages [25] :

* on scinde la classe de dispersion maximum

* pour réaliser les sous-classes, on choisit un objet périphérique de la classe à scinder pour en faire le noyau d'une nouvelle classe. On révise alors l'affectation de tous les vecteurs avec la règle de la distance au barycentre.

M. Roux [25] propose un algorithme inspiré de la méthode de Fages :

1) On choisit deux points i_1 et i_2 d'une classe ω_i. Les vecteurs les plus proches de i_1 (resp. i_2) sont affectés à la classe de i_1 (resp. i_2),

2) On détermine les centres de gravités g_1 et g_2 des classes de i_1 et de i_2 et les masses M_1 et M_2 de ces deux classes. On détermine alors la dispersion de ces deux classes définie par :

$$\frac{N_1 N_2}{N_1 + N_2} d^2 (g_1, g_2)$$

On réitère l'opération à l'intérieur de la classe ω_i pour toutes les paires de points i_1 et i_2. On retient i_1 et i_2 et la classe ω_i qui maximisent

$$\frac{N_1 N_2}{N_1 + N_2} d^2 (g_1, g_2)$$

On affecte alors définitivement les points de la classe ω_i en fonction de leur proximité aux points i_1 et i_2.

3.2.3 Hiérarchie basée sur la théorie des graphes

à voir

Chapitre 4

Fonctions discriminantes linéaires et réseaux neuronaux

La règle de classification Bayésienne ainsi que ses règles dérivées provenant de l'apprentissage supervisé ou non supervisé sont basées sur les lois des paramètres de l'espace de représentation choisi, conditionnellement à chaque classe. Une autre démarche consiste à déterminer directement une ou plusieurs fonctions discriminantes et d'utiliser la règle de classification décrite au chapitre précédent (voir paragraphe 5). Les méthodes obtenues sont alors non-paramétriques puisqu'elles ne supposent pas connues la forme des densités de chaque classe. On distingue deux classes de méthodes suivant que ces fonctions discriminantes sont linéaires ou non-linéaires

1 Fonctions Discriminantes Linéaires

Dans le chapitre précédent, les méthodes de classification étaient basées sur une connaissance a priori des densités conditionnelles $f(x|\omega_j)$ ou à leur estimation à l'aide de données de chacune des classes ω_j. Le but recherché était de remplacer les densités $f(x|\omega_j)$ par leurs estimations dans la règle de classification bayésienne. Par exemple, dans le cas de deux classes, la règle de classification bayésienne après estimation des densités s'écrit :

$$d(x) = a_1 \Leftrightarrow \widehat{f}(x|\omega_1) P(\omega_1) \geq \widehat{f}(x|\omega_2) P(\omega_2)$$

c'est-à-dire

$$d(x) = a_1 \Leftrightarrow g(x) > 0 \tag{4.1}$$

où $g(x) = \widehat{f}(x|\omega_1) P(\omega_1) - \widehat{f}(x|\omega_2) P(\omega_2)$ est une fonction appelée fonction discriminante. Cette fonction est généralement non-linéaire par rapport à x.

Le classifieur ainsi obtenu n'admet pas une probabilité d'erreur minimale dans la mesure où on utilise les estimations des densités et non pas les densités elles-mêmes (on dit parfois que c'est un classifieur sous-optimal). Une autre façon de construire un classifieur sous-optimal consiste à estimer une fonction discriminante de décision $g(x)$ à l'aide des données de l'apprentissage. Par souci de simplicité, on peut imposer à $g(x)$ d'être une fonction linéaire de x, ce qui fait l'objet de ce paragraphe.

On se propose donc de déterminer une fonction discriminante linéaire de la forme :

$$g\left(x\right) = w^t x$$

avec $w = \left(w_1, ..., w_{p-1}, w_p\right)^t$ et $x = \left(x_1, ..., x_{p-1}, 1\right)^t$ telle que la règle de classification associée

$$\begin{array}{rcl} d\left(x\right) & = & a_1 \Leftrightarrow g(x) > 0 \\ d\left(x\right) & = & a_2 \Leftrightarrow g(x) < 0 \end{array} \tag{4.2}$$

donne des résultats acceptables. Il existe plusieurs méthodes permettant de déterminer la fonction discriminante linéaire g, c'est-à-dire le vecteur inconnu w.

1.1 Algorithme du Perceptron

Cas de deux classes

L'algorithme du perceptron suppose que les deux classes ω_1 et ω_2 sont linéairement séparables, c'est-à-dire qu'il existe un vecteur w^* (il en existe alors en général plusieurs) tel que

$$\begin{array}{rcll} \left(w^*\right)^t x & > & 0 & \forall x \in \omega_1 \\ \left(w^*\right)^t x & < & 0 & \forall x \in \omega_2 \end{array}$$

En d'autres termes, il existe un hyperplan d'équation $\left(w^*\right)^t x = 0$ séparant les vecteurs x de la classe ω_1 des vecteurs x de la classe ω_2. L'idée de l'algorithme du perceptron est de chercher le vecteur w en minimisant une fonction de cout $J(w)$ appropriée. Une idée naturelle consisterait à définir $J(w)$ comme le nombre de vecteurs x de la base d'apprentissage mal classés par la règle (4.2). Cette fonction de coût est nulle lorsque tous les éléments de la base d'apprentissage sont bien classés et strictement positive lorsqu'au moins un des vecteurs de cette base est mal classés. Cette fonction $J(w)$ est constante

par morceaux et se prète donc mal aux algorithmes standards d'optimisation. La fonction de coût de l'algorithme du perceptron est définie comme suit :

$$J(w) = \sum_{x \in Y} \left(-\delta_x w^t x\right) \tag{4.3}$$

où δ_x représente la sortie désirée :

$$\begin{aligned} \delta_x &= 1 \text{ si } x \in \omega_1 \\ \delta_x &= -1 \text{ si } x \in \omega_2 \end{aligned}$$

et Y est le nombre de vecteurs de la base d'apprentissage mal classés par la règle (4.2). La fonction de coût (4.3) est continue par morceaux et linéaire par morceaux. Par conséquent, l'optimisation de cette fonction peut se faire à l'aide d'un algorithme de descente. L'algorithme habituel de descente est un algorithme itératif défini comme suit : à l'itération n, on présente un vecteur $x(n) = (x_1(n), ..., x_p(n))^t$ dont la classe est connue (i.e. δ_x est connu) et on met à jour la valeur du poids w_k à l'aide de la règle suivante :

$$w_k(n+1) = w_k(n) - \mu \left.\frac{\partial J(w)}{\partial w_k}\right|_{w_k = w_k(n)}$$

où $w_k(n)$ et $w_k(n+1)$ représentent les valeurs du poids w_k à la $n^{ème}$ et à la $(n+1)^{ème}$ itérations et où $\mu > 0$ est un paramètre d'apprentissage. Des calculs élémentaires montrent que l'équation précédente s'écrit :

Mise à jour des poids w_k

$$w_k(n+1) = w_k(n) + \mu \sum_{x \in Y} (\delta_x x)$$

La règle de mise à jour des poids w_k est donc très simple. On notera que le terme $\Delta w_k(n) = w_k(n+1) - w_k(n) = \mu \sum_{x \in Y} (\delta_x x)$ est nul dès que tous les vecteurs x de la base d'apprentissage sont bien classés. On montre que dans le cas de deux classes d'apprentissage ω_1 et ω_2 linéairement séparables (séparées par hyperplan) avec $\mu \in]0, 2]$, cette règle de mise à jour est convergente après un nombre fini d'itérations n_0, c'est-à-dire :

$$\widehat{w}_k(n_0) = \widehat{w}_k(n_0+1) = \widehat{w}_k(n_0+2) = ...$$

Afin d'accélerer la convergence, on peut utiliser un paramètre d'apprentissage variant dans le temps $\mu(n)$. La règle de mise à jour précédente est alors

convergente si les deux conditions suivantes sont satisfaites :

$$\begin{aligned} \sum_{n=0}^{+\infty} \mu(n) &= +\infty \\ \sum_{n=0}^{+\infty} \mu^2(n) &< +\infty \end{aligned}$$

Un exemple classique de paramètre d'apprentissage vérifiant ces deux hypothèses est $\mu(n) = \frac{c}{n}$.

Variante de l'algorithme du perceptron

Dans l'algorithme du perceptron présenté ci-dessus, lea règle de mise à jour des poids nécessite à chaque itération de calculer la fonction discriminante pour tous les éléments de la base d'apprentissage, ce qui représente un coût calculatoire important. Une variante de cet algorithme consiste à présenter les vecteurs de la base d'apprentissage séquentiellement et de ne corriger les poids du réseau à l'itération n qu'à l'aide de l'erreur "instantanée" $e(n) = y(n) - d(n)$. Plus précisément, cet algorithme proposé par Rosenblatt en 1958 [27], [19] est défini de la façon suivante :

1) *Initialisation des poids*

$w_k(0) = 0$ (l'algorithme du perceptron converge quelle que soit la valeur initiale des poids ; cependant, le nombre d'itérations nécessaire pour que l'algorithme converge dépend des valeurs initiales des poids)

2) *Présentation d'une entrée de la base d'apprentissage*

On présente une entrée $x(n)$ dont la classe est connue. On définit la sortie désirée $d(n)$ de la façon suivante :

$$\begin{aligned} d(n) &= 1 \text{ si } x(n) \in \omega_1 \\ d(n) &= 0 \text{ si } x(n) \in \omega_2 \end{aligned}$$

3) *Calcul de la sortie du réseau*

$$y(n) = f_h\left(\sum_{i=1}^{p} w_i(n) x_i(n)\right)$$

où la non-linéarité $f_h(x)$ est définie par

$$\begin{aligned} f_h(x) &= 1 \qquad x > 0 \\ f_h(x) &= -1 \qquad x \leq 0 \end{aligned} \tag{4.4}$$

4) *Mise à jour des poids*

$$\widehat{w}_k(n+1) = \widehat{w}_k(n) + \mu x_k(n)\left[d(n) - y(n)\right]$$

5) *Retour à 2) jusqu'à convergence*

Le gain μ doit appartenir à l'intervalle $]0, 2]$. Il peut évoluer au cours du temps (on note alors $\mu(n)$). Le règlage de ce gain se fait en tenant compte des deux remarques suivante :

* la convergence de l'algorithme est d'autant plus rapide que le gain μ est proche de 2.

* l'erreur résiduelle sur les poids est d'autant plus faible que le gain μ est proche de 0.

A titre d'exemple, la figure suivante présente les résultats obtenus avec un perceptron à une couche avec 80 entrées et $\mu = 0.01$:

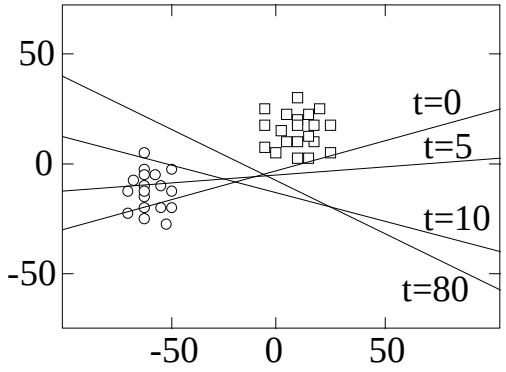


Fig. 4.9 : exemple de convergence du perceptron à une couche

Il est important de noter qu'il n'y a pas unicité de la valeur vers laquelle converge $w(n)$ (la valeur limite dépend des valeurs initiales des poids et de l'ordre dans lequel on présente les vecteurs de la base d'apprentissage).

Remarque

Dans le cas où $\mu(n)$ est un paramètre variable, cette variante de l'algorithme du perceptron converge également en un nombre fini d'itérations ([32], p. 60) si les conditions suivantes sont vérifiées :

$$\lim_{n\to\infty} \mu(n) = 0, \sum_{n=0}^{\infty} \mu(n) = \infty \text{ et } \sum_{n=0}^{\infty} \mu^2(n) < \infty$$

On peut par exemple choisir $\mu(n) = \frac{c}{n}$ (notons que les corrections deviennent de plus en plus faibles lorsque n augmente).

Cas de $K > 2$ classes

L'algorithme du perceptron se généralise au cas de K classes. On trouvrea par exemple une généralisation connue sous le nom d'algorithme de Kesler pour le cas de K classes linéairement séparables dans ([32], p. 63).

1.2 Méthodes des Moindres Carrés

La détermination d'un hyperplan séparateur dans le cas de classes linéairement séparables peut se faire à l'aide de l'algorithme du perceptron. Lorsque cette condition de séparabilité n'est pas satisfaite, on peut essayer de déterminer un hyperplan séparateur qui dans certaines situations donnera des résultats (sous-optimaux) suffisants. Ceci est l'objet de cette partie.

Filtre Optimal de Wiener-Hopf

Le filtre optimal de Wiener Hopf est défini par le vecteur w qui minimise la moyenne de l'erreur quadratique instantanée entre les valeurs réelle et désirée de la fonction de coût :

$$J(w) = \frac{1}{2}E\left[e^2(n)\right] = \frac{1}{2}E\left[\left(d(n) - w^t x(n)\right)^2\right]$$

Puisque J est une forme quadratique par rapport au vecteur w dont le Hessien est une matrice symétrique définie positive, J admet un minimum global unique défini par :

$$\frac{\partial J}{\partial w_k} = 0 \qquad \forall k \in \{1, ..., p\} \tag{4.5}$$

c'est-à-dire

$$E\left[e(n)x_k(n)\right] = 0 \qquad \forall k \in \{1, ..., p\} \tag{4.6}$$

ou :

$$E\left[d(n)x_k(n)\right] = \sum_{i=1}^{p} w_i E\left[x_i(n)x_k(n)\right] \qquad \forall k \in \{1, ..., p\} \tag{4.7}$$

$r_{dx}(k) = E\left[d(n)x_k(n)\right]$ est la fonction d'intercorrélation entre d et x tandis que $r_x(i,k) = E\left[x_i(n)x_k(n)\right]$ est la fonction d'autocorrélation de x. Le système d'équations (4.7) est appelé **système d'équations de Wiener-Hopf** et le filtre dont les poids (notés w_i^{opt}) vérifient les équations de Wiener-Hopf est appelé **filtre de Wiener**. Pour résoudre les équations de Wiener-Hopf,

on doit résoudre un système de p équations à p inconnues, ce qui revient à inverser une matrice $p \times p$. On peut éviter cette inversion matricielle en utilisant un algorithme itératif. Les poids sont supposés variables et leurs valeurs sont mises à jour de façon itérative, dans le but qu'ils convergent vers la solution optimale.

Méthode de la plus profonde descente

Soit $w_k(n)$ la valeur du poids w_k à la $n^{ème}$ itération. La méthode de la plus profonde descente consiste à mettre à jour les poids $w_k(n)$ de façon à se déplacer dans la direction du vecteur gradient mais dans le sens inverse soit :

$$\begin{aligned} w_k(n+1) &= w_k(n) + \Delta w_k(n) \\ w_k(n+1) &= w_k(n) - \mu \left.\frac{\partial J(w)}{\partial w_k}\right|_{w_k = w_k(n)} \end{aligned} \tag{4.8}$$

où $\Delta w_k(n)$ est la variation du poids $w_k(n)$ et où μ est un paramètre > 0 appelé paramètre d'apprentissage (learning rate). On obtient alors la règle de mise à jour suivante :

$$w_k(n+1) = w_k(n) + \mu \left[r_{dx}(k) - \sum_{j=1}^{p} w_j(n)\, r_x(j,k) \right] \qquad k = 1, ..., p \tag{4.9}$$

La méthode de la plus profonde descente est exacte en ce sens qu'aucune approximation n'est faite.

Remarques

- La méthode de la plus profonde descente exposée ci-dessus cherche à minimiser $J(w) = \frac{1}{2} E[e^2(n)]$. Dans le cas général où la fonction de coüt $J(w)$ admet plusieurs minima locaux, l'algorithme converge vers l'un de ces minima locaux (dans la mesure oi le paramètre μ_k est convenablement choisi) mais pas forcément vers le mnimum global de $J(w)$. Dans notre cas d'étude, le critère $J(w)$ est une forme quadratique par rapport à w qui admet un seul minimum global unique. L'algorithme de la plus profonde descente convergera donc vers ce minimum,

- Il existe également une méthode de plus profonde descente qui cherche à minimiser $E_{totale}(n) = \frac{1}{2} \sum_{i=1}^{n} e^2(i)$. Cette seconde approche donne un résultat similaire à (4.9), obtenu en remplaçant les espérances mathématiques par des moyennes temporelles (ce qui est se comprend aisément si les processus physiques étudiés (entrée du filtre et réponse désirée) sont ergodiques,

- Un des problèmes de l'algorithme de plus profonde descente défini par (4.9) est la connaissance a priori des quantités $r_{dx}(k)$ et $r_x(j,k)$. Dans certaines applications, ces quantités sont inconnues. On peut alors utiliser l'algorithme qui minimise $E_{totale}(n)$. Mais cet algorithme nécessite la mise en mémoire de toutes les données au fur et à mesure des itérations (entrées, erreurs, poids, ...). Pour résoudre ce problème, on peut utiliser des estimations instantanées des fonctions de corrélation. On obtient alors l'algorithme LMS (Least Mean Square).

Algorithme LMS

L'algorithme LMS est basé sur des estimées instantanées de $r_{dx}(k)$ et de $r_x(j,k)$:

$$\begin{array}{l} \widehat{r}_{dx}(k,n) = d(n)\, x_k(n) \\ \widehat{r}_x(j,k,n) = x_j(n) x_k(n) \end{array} \tag{4.10}$$

Lorsqu'on remplace $r_{dx}(k)$ et $r_x(j,k)$ par leurs estimations instantanées dans (4.9), on obtient :

$$\widehat{w}_k(n+1) = \widehat{w}_k(n) + \mu \left[d(n)\, x_k(n) - \sum_{j=1}^{p} \widehat{w}_j(n)\, x_j(n) x_k(n) \right] \qquad k = 1, ..., p \tag{4.11}$$

c'est-à-dire :

$$\widehat{w}_k(n+1) = \widehat{w}_k(n) + \mu x_k(n) \left[d(n) - y(n) \right] \qquad k = 1, ..., p \tag{4.12}$$

où $y(n) = \sum_{j=1}^{p} \widehat{w}_j(n)\, x_j(n)$ est la valeur de la fonction discriminante à la $n^{ème}$ itération. On notera l'utilisation de $\widehat{w}_k$ au lieu de w_k qui montre le fait qu'on travaille avec des estimations des fonctions de corrélation. Dans la méthode de la plus profonde descente, les poids partent d'une valeur initiale $w(0)$ et suivent une trajectoire bien précise qui se termine sur la solution optimale de Wiener Hopf w^{opt}. Au contraire, dans l'algorithme LMS, les poids $\widehat{w}$ suivent une trajectoire aléatoire. Pour cette raison, l'algorithme LMS est communément appelé l'algorithme du gradient stochastique. Il est à noter qu'habituellement, l'initialisation de l'algorithme LMS se fait par $\widehat{w}(0) = 0$.

Convergence

- **Dans le cas où le pas d'adaptation μ est fixe**, sous certaines conditions ($0 < \mu < \frac{2}{Trace(R_x)}$ où R_x est la matrice des autocorrélations de x ([32], p. 69), ([13], p. 131)), la moyenne de $w(n)$ converge :

$$\lim_{n\to\infty} E\left[w\left(n\right)\right] = w^{opt}$$

et :

$$\lim_{n\to\infty} E\left[\left\|w\left(n\right) - w^{opt}\right\|^2\right] = \text{constante}$$

La dernière égalité montre qu'on a dans ce cas une erreur résiduelle.

- **Dans le cas où le pas d'adaptation est variable** (on le note alors $\mu\left(n\right)$) et vérifie ([12], p. 377) :

$$\lim_{n\to\infty} \mu\left(n\right) = 0, \sum_{n=0}^{\infty} \mu\left(n\right) = \infty \text{ et } \sum_{n=0}^{\infty} \mu^2\left(n\right) < \infty$$

la règle de mise à jour (4.9) assure la convergence avec probabilité 1 de $w(n)$ vers la solution de Wiener :

$$\lim_{n\to\infty} P\left[w\left(n\right) = w^{opt}\right] = 1 \tag{4.13}$$

On a également la convergence au sens de la moyenne quadratique :

$$\lim_{n\to\infty} E\left[\left\|w\left(n\right) - w^{opt}\right\|^2\right] = 0$$

Remarque

La mise à jour des poids w_k grâce à l'algorithme du perceptron définie par l'équation (**??**) est exactement identique à celle décrite par l'algorithme LMS (4.12). Cependant, on peut noter que dans l'algorithme du perceptron, la sortie du système $y(n)$ est quantifiée ($y(n) = 1$ ou $y(n) = -1$) ce qui n'est pas le cas dans l'algorithme LMS. De plus, la convergence de l'algorithme du perceptron est assurée après un nombre fini d'itérations, ce qui n'est pas le cas avec l'algorithme LMS ([32], p. 59).

1.3 Ho-Kashyap

p. 159 Duda, p. 218 Dubuisson, SVMs

2 Machines à vecteurs supports (SVMs)

2.1 Introduction

Le problème habituel de classification dans le cas de deux classes peut-être décrit comme suit : on dispose d'une base d'apprentissage $\mathcal{B} = \{(x_1, y_1), ..., (x_n, y_n)\}$ où $x_1, ..., x_n$ sont n vecteurs de $\mathbb{R}^p$ et $y_1, ..., y_n$ sont des variables binaires telles que

$$\begin{aligned} y_i &= 1 \text{ si } x_i \in \omega_1 \\ y_i &= -1 \text{ si } x_i \in \omega_2 \end{aligned}$$

et on cherche une règle de classification associant à tout vecteur $x \in \mathbb{R}^p$ une décision $d(x)$ de façon à minimiser une fonction de coût appropriée comme la probabilité d'erreur du classifieur. On s'intéresse dans cette partie aux classifieurs (appelés parfois machines d'apprentissage) tels que $d(x)$ est une fonction de $\mathbb{R}^p$ dans $\{-1, +1\}$ paramétrée par un vecteur inconnu α. On notera alors naturellement la règle de décision $d(x, \alpha)$. Sans perte de généralité, on choisira la fonction de coût habituelle

$$c(d(x, \alpha), y) = \begin{cases} 1 \text{ si } y \neq d(x, \alpha) \\ 0 \text{ si } y = d(x, \alpha) \end{cases}$$

On sait que dans ce cas, le coût moyen

$$R(\alpha) = E[c(d(x, \alpha), y)]$$

est la probabilité d'erreur du classifieur. En effet, on a

$$\begin{aligned} R(\alpha) &= E[c(d(x, \alpha), y)] \\ &= E[E[c(d(x, \alpha), y) | x]] \\ &= \int E[c(d(x, \alpha), y) | x] f(x) dx \\ &= \int P[y \neq d(x, \alpha) | x] f(x) dx \end{aligned}$$

et $P[y \neq d(x, \alpha) | x]$ est le coût moyen conditionnel à x noté $R_d(x)$ au chapitre 3.

Minimisation du risque empirique

Dans la plupart des applications pratiques, on ne connait pas la loi du vecteur (x, y). En revanche, on dispose de la base d'apprentissage $\mathcal{B}$ telle que le vecteur $((x_1, y_1), ..., (x_n, y_n))$ est un échantillon de la loi de (x, y) (les vecteurs (x_i, y_i) sont indépendants et de même loi que (x, y)). On utilise

dans ce cas le principe de minimisation du risque empirique (empirical risk minimization principle ou ERM principle) qui consiste à minimiser le risque empirique défini par

$$R_{emp}(\alpha) = \frac{1}{n}\sum_{i=1}^{n} c\left(d(x_i,\alpha), y_i\right)$$

On peut de manière plus générale définir d'autres fonctions de coût qui permettent de résoudre des problèmes comme l'estimation de densités ou la régression (voir [34], p. 19). Vapnik et Chervonenkis ont étudié diverses relations entre $R_{emp}(\alpha)$ et $R(\alpha)$. En particulier, ils ont montré le résultat suivant ([34], p. 84) :

$$P\left\{R(\alpha) \leq R_{emp}(\alpha) + \frac{\varepsilon}{2}\left(1 + \sqrt{1 + \frac{4R_{emp}(\alpha)}{\varepsilon}}\right)\right\} > 1 - \eta \qquad (4.14)$$

où ε est défini comme suit :

- Si l'ensemble de fonctions indicatrices $S = \{c\left(d(x,\alpha), y\right), \alpha \in \Lambda\}$ contient un nombre fini d'éléments N, alors

$$\varepsilon = 2\frac{\ln N - \ln \eta}{n}$$

- Si l'ensemble de fonctions indicatrices $S = \{c\left(d(x,\alpha), y\right), \alpha \in \Lambda\}$ contient un nombre infini d'éléments mais possède une dimension de Vapnik-Chervonenkis finie est égale à h, alors

$$\varepsilon = 4\frac{h\left(\ln\frac{2n}{h} + 1\right) - \ln(\eta/4)}{n}$$

L' inégalité (4.14) est fondamentale et montre que lorsque $\frac{n}{h}$ est grand, ε est petit et donc le risque empirique $R_{emp}(\alpha)$ est proche du risque réel $R(\alpha)$. Dans ce cas on peut appliquer le principe de minimisation du risque empirique puisque minimizer $R_{emp}(\alpha)$ revient à minimizer $R(\alpha)$. Par contre, dans le cas où $\frac{n}{h}$ est petit, une faible valeur de $R_{emp}(\alpha)$ ne garantit pas une faible valeur de $R(\alpha)$. Dans ce cas, on doit minimiser le terme de droite de l'inégalité (4.14), ce qui conduit au principe de de minimisation du risque structurel (Structural Risk Minimization principle ou SRM principle).

Minimisation du risque structurel

Le terme de droite de l'inégalité (4.14) dépend du risque empirique lié à la base d'apprentissage mais aussi de la dimension de Vapnik-Chervonenkis

de l'ensemble des fonctions indicatrices notée $\dim_{VC}() = h$. Le principe de minimisation du risque structurel consiste à faire de la dimension de Vapnik-Chervonenkis h une variable de contrôle de l'algorithme de classification. Notons que le second terme du membre de droite de l'inégalité (4.14) règle l'intervalle de confiance de l'estimateur $R_{emp}(\alpha)$ de $R(\alpha)$ et en conséquence ce terme est parfois appelé intervalle de confiance. Le principe de minimisation du risque structurel consiste à définir une structure de fonctions indicatrices $S = \cup_k S_k$ où $S_k = \{c(d(x,\alpha),y), \alpha \in \Lambda_k\}$ est une suite d'ensembles telle que

$$S_1 \subset S_2 \subset ... \subset S_n ...$$

avec $\dim_{VC}(S_k) = h_k < +\infty$ et $h_1 \leq h_2 \leq ... \leq h_n ...$. On dit qu'une telle structure est *admissible*. Le principe de minimisation du risque structurel consiste à choisir la fonction indicatrice $c\left(d(x,\alpha_n^k),y\right)$ qui minimise la borne

$$R_{emp}(\alpha) + \frac{\varepsilon}{2}\left(1 + \sqrt{1 + \frac{4R_{emp}(\alpha)}{\varepsilon}}\right) \tag{4.15}$$

parmi toutes les fonctions indicatrices de S_k. Ce principe réalise un compromis entre le risque empirique et l'intervalle de confiance (sorte de dilemme biais-variance). En pratique, on cherche le risque optimal correspondant à la valeur optimale de h notée h^*, comme l'illustre la figure ci-dessous (issue de [34], p. 96) :

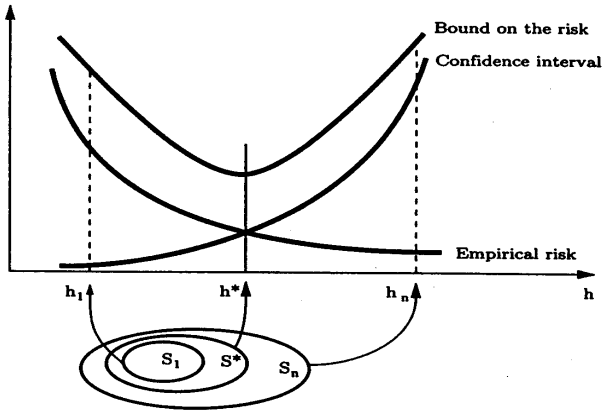


FIGURE 4.2. The bound on the risk is the sum of the empirical risk and the confidence interval. The empirical risk decreases with the index of the element of the structure, while the confidence interval increases. The smallest bound of the risk is achieved on some appropriate element of the structure.

La borne (4.15) peut être exprimée sous la forme $R_{emp}(\alpha) + \Phi\left(\frac{n}{h_k}\right)$, où $R_{emp}(\alpha)$ est le risque empirique et $\Phi\left(\frac{n}{h_k}\right)$ l'intervalle de confiance. Pour minimiser cette borne, il y a deux approches :

- la première approche consiste à fixer l'intervalle de confiance en choisissant convenablement la structure de la machine qui va être utilisée pour l'apprentissage. Lorsque cette structure est choisie (pas trop compliquée pour éviter le phénomène d'overfitting et pas trop simple pour avoir un classifieur performant), on cherche la machine possédant cette structure minimisant le risque empirique. Une telle approche est suivie lorsque le classifieur est un réseau de neurones. Dans un premier temps, on choisit la structure du réseau (nombre de couches et nombre de noeuds par couche) et ensuite on cherche les poids du réseau qui minimisent une fonction de coût appropriée. Cependant, la capacité de contrôle d'un réseau de neurones est faible ([34], p. 130) car la fonction de coût possède en général plusieurs minima locaux et l'algorithme de rétropropagation du gradient converge vers l'un de ces minima (pas forcément le minimu global). De plus, la valeur des poids après convergence dépend des valeurs initiales de ces poids.

- La seconde approche consiste à fixer la valeur du risque empirique $R_{emp}(\alpha)$ et à optimiser la structure de la machine de façon à minimiser l'intervalle de confiance. Une telle approche est suivie par les machines à vecteurs supports dont le fonctionnement est décrit dans ce qui suit.

2.2 L'hyperplan séparateur optimal

On dispose d'une base d'apprentissage $\mathcal{B} = \{(x_1, y_1), ..., (x_n, y_n)\}$ et on cherche un hyperplan $\mathcal{H}$ d'équation

$$g_{w,b}(x) = w^T x - b = 0$$

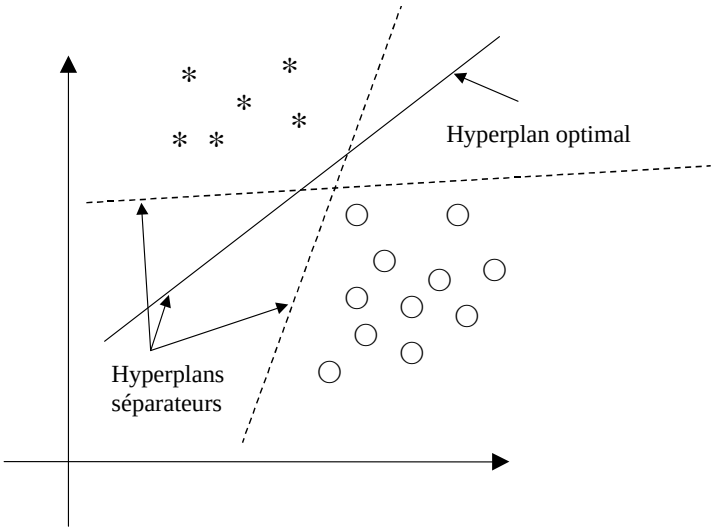
qui sépare les données des deux classes ω_1 et ω_2, c'est-à-dire tel que

$$\begin{aligned} g_{w,b}(x_i) &> 0 \text{ si } x_i \in \omega_1 \\ g_{w,b}(x_i) &< 0 \text{ si } x_i \in \omega_2 \end{aligned} \tag{4.16}$$

Lorsque cet hyperplan séparateur $\mathcal{H}$ sera déterminé, pour un vecteur $x \in \mathbb{R}^p$ de classe inconuue, on utilisera la règle de classification suivante

$$f(x) = \text{sign}\left(g_{w,b}(x)\right)$$

Il existe en général une infinité d'hyperplan vérifiant les conditions précédentes pour les n vecteurs x_i de la base d'apprentissage, comme l'illustre la figure suivante :



Les machines à vecteurs supports cherchent à déterminer l'hyperplan maximisant la marge de la base d'apprentissage $\mathcal{B}$. La marge (géométrique) du couple (x_i, y_i) est définie par

$$\gamma_i(\widetilde{w}) = \frac{y_i\left(w^T x_i - b\right)}{\|\widetilde{w}\|}$$

avec $\widetilde{w} = (w, b)$. La marge de la base d'apprentissage est définie par

$$\gamma_{\mathcal{B}}(\widetilde{w}) = \min_{i\in\{1,\ldots,n\}} \frac{y_i\left(w^T x_i - b\right)}{\|\widetilde{w}\|}$$

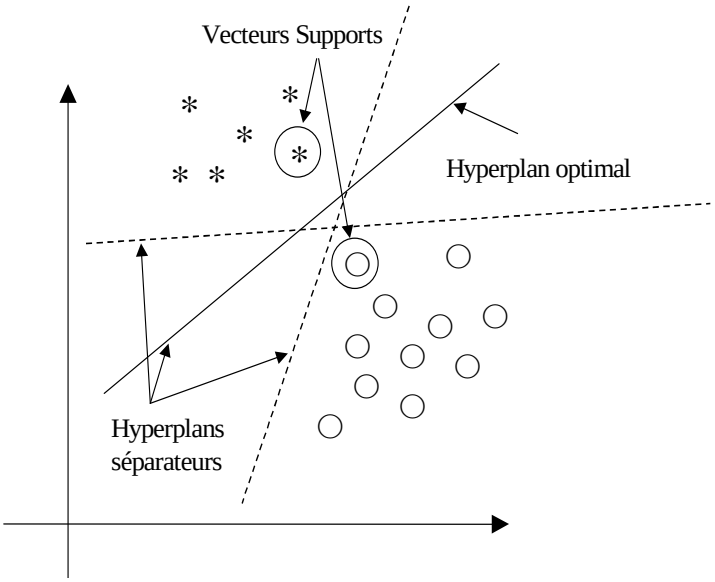
On remarquera que le vecteur x_i est bien classé par la règle de décision (décision) lorsque $\gamma_i(\widetilde{w}) > 0$. On peut noter que $\gamma_{\mathcal{B}}(a\widetilde{w}) = \gamma_{\mathcal{B}}(\widetilde{w})$, $\forall a$, ce qui traduit la non unicité du vecteur $\widetilde{w}$ définissant l'hyperplan optimal séparateur. Pour déterminer cet hyperplan séparateur, on doit se fixer des contraintes sur $\widetilde{w}$:

- Une solution consisterait à chercher $\widetilde{w}$ sous la contrainte $\|\widetilde{w}\| = 1$, c'est-à-dire à chercher le vecteur $\widetilde{w}$ minimisant $\min_{i\in\{1,\ldots,n\}} y_i\left(w^T x_i - b\right)$ sous la contrainte $\|\widetilde{w}\| = 1$. Malheureusement, un tel problème est compliqué puisque la fonction à optimiser est non-linéaire et n'est pas quadratique. De plus la contrainte $\|\widetilde{w}\| = 1$ est aussi non-linéaire.

- La solution retenue consiste à minimiser la marge du classifieur de façon à ce que les points les plus proches de l'hyperplan vérifient $y_i\left(w^T x_i - b\right) = 1$, c'est-à-dire déterminer l'hyperplan tel que

$$\min_{i\in\{1,\ldots,n\}} y_i\left(w^T x_i - b\right) = 1 \tag{4.17}$$

On peut toujours satisfaire une telle condition car si $y_i\left(w^T x_i - b\right) = a$ en changeant $\widetilde{w}$ en $\widetilde{w}/a$, on aura $y_i\left(w^T x_i - b\right) = 1$. Le vecteurs x_i vérifiant $y_i\left(w^T x_i - b\right) = 1$ sont appelés *vecteurs supports*. Pour l'exemple précédent, ces vecteurs supports sont encerclés sur la figure ci-dessous :



- L'hyperplan $\mathcal{H}$ vérifiant la contrainte (4.17) est appelé hyperplan *canonique*. Un tel hyperplan vérifie

$$y_i\left(w^T x_i - b\right) \geq 1, \quad \forall i = 1, ..., n$$

Dans ces conditions, la marge du classifieur pour un hyperplan canonique est

$$\gamma_{\mathcal{B}}\left(\widetilde{w}\right) = \frac{1}{\|\widetilde{w}\|}$$

Le problème consistant à maximiser la marge du classifieur peut alors se formuler comme suit :

$$\boxed{\text{Minimiser } \tfrac{1}{2}\|w\|^2 \text{ sous les contraintes } y_i\left(w^T x_i - b\right) \geq 1, \forall i}$$

Ce problème est relativement simple car bien que la fonction à optimiser soit non-linéaire (quadratique), les contraintes sont linéaires.

2.3 Détermination de l'hyperplan séparateur optimal

On construit le Lagrangien associé au problème d'optimisation sous contraintes précédent :

$$L\left(\widetilde{w},\alpha\right)=\frac{1}{2}w^{T}w-\sum_{i=1}^{n}\alpha_{i}\left[y_{i}\left(w^{T}x_{i}-b\right)-1\right]$$

En annulant les dérivées partielles de $L\left(\widetilde{w},\alpha\right)$ par rapport à b et w, on obtient :

$$\sum_{i=1}^{n}\alpha_{i}y_{i}=0$$
$$w=\sum_{i=1}^{n}\alpha_{i}y_{i}x_{i}$$

D'après la théorie des multiplicateurs de Kühn et Tucker rappelée ci-dessous : **Résumé : Lorsqu'on a un problème d'optimisation convexe (fonction $f(x)$ à optimiser convexe et contraintes $G_i\left(x\right)\leq 0$ convexes), une condition nécessaire et suffisante d'optimalité est l'existence de paramètres α_i tels que**

$$L'(x)=f'\left(x\right)+\sum_{i=1}^{n}\alpha_{i}G_{i}'\left(x\right)=0$$

avec $\alpha_i = 0$ si $G_i\left(x\right) < 0$ (en d'autres termes $\alpha_i G_i\left(x\right) = 0$). La fonction $L(x)$ est appelée Lagrangien du problème d'optimisation.

On en déduit :

$$w=\sum_{i=1}^{n}\alpha_{i}y_{i}x_{i}\text{ avec }\left\{\begin{array}{l}\alpha_{i}=0\text{ si la contrainte est une inégalité stricte}\\ \alpha_{i}\geq 0\text{ si la contrainte est une égalité}\end{array}\right.$$

d'où

$$w=\sum_{\text{vecteurs supports}}\alpha_{i}y_{i}x_{i} \tag{4.18}$$

On remarquera que la condition $\alpha_i G_i\left(x\right) = 0$ s'écrit ici $\alpha_i\left[y_i\left(w^T x_i - b\right) - 1\right] = 0$. On pose alors $\alpha = \left(\alpha_1, ..., \alpha_n\right)^T$, $x = \left(x_1, ..., x_n\right)^T$ et $Y =$diag$(y_1, ..., y_n)$, où

diag$(y_1, ..., y_n)$ est une matrice diagonale de taille $n \times n$ dont les éléments diagonaux sont $y_1, ..., y_n$. En remplaçant l'expression de w dans le Lagrangien et en prenant en compte les conditions de Kühn et Tucker, on obtient

$$\begin{aligned} U(\alpha) &= \frac{1}{2}\left(\alpha^T Y x\right) x^T Y^T \alpha - \sum_{i,j=1}^{n} \alpha_i y_i \alpha_j y_j x_i^T x_j + b \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i \\ &= -\frac{1}{2}\alpha^T Y \left(x x^T\right) Y^T \alpha + \sum_{i=1}^{n} \alpha_i \end{aligned}$$

que l'on doit maximiser dans le domaine $\alpha_i \geq 0, \forall i$ sous les contraintes $\sum_{i=1}^{n} \alpha_i y_i = 0$. Ce problème d'optimisation est simple car la fonction à optimiser est quadratique (donc convexe) et les contraintes associées sont simples (linéaires).

Après avoir déterminé la solution de ce problème notée $\alpha_0 = (\alpha_1^0, ..., \alpha_n^0)$, on peut déterminer le vecteur w à l'aide de (4.18). Alors la norme du vecteur w_0 correspondant est (voir [34], p. 135)

$$\|w_0\|^2 = w_0^T w_0 = \sum_{\text{vecteurs supports}} \alpha_i^0 \alpha_j^0 y_i y_j x_i^T x_j$$

Mais on a aussi

$$\|w_0\|^2 = w_0^T \sum_{\text{vecteurs supports}} \alpha_i^0 y_i x_i$$

Puisque $y_i\left(w^T x_i - b\right) - 1 = 0$ pour un vecteur support, on a $y_i w^T x_i = 1 + y_i b$, d'où ([4], p. 98) :

$$\begin{aligned} \|w_0\|^2 &= \sum_{\text{vecteurs supports}} \alpha_i^0 \left(1 + y_i b\right) \\ &= \sum_{\text{vecteurs supports}} \alpha_i^0 \text{ (à cause de la contrainte } \sum_{i=1}^{n} \alpha_i y_i = 0) \end{aligned}$$

La marge du classifieur s'exprime donc sous la forme ([4], p. 98) :

$$\boxed{\gamma = \frac{2}{\|w_0\|} = 2\left(\sum_{\text{vecteurs supports}} \alpha_i^0\right)^{-1/2}}$$

La règle de classification pour un vecteur $x$devient

$$f(x) = \text{sign}\left(\sum_{z_i \text{ vecteurs supports}} \alpha_i^0 y_i x_i^T x - b_0\right)$$

Il reste à déterminer le vecteur b_0 comme suit :

$$b_0 = \frac{1}{2}\left(w_0^T z^+ + w_0^T z^-\right)$$

où z^+ est un vecteur support de la première classe et z^- un vecteur support de la seconde classe.

2.4 Cas non séparable : le classifieur "soft-margin SVM"

La théorie précédente ne fonctionne que si les données des deux classes sont linéairement séparables. Dans le cas non séparable, on peut introduire des variables $\xi_i \geq 0$ qui autorisent certaines erreurs tout en les pénalisant. Une approche classique consiste à introduire un terme de pénalisation linéaire, c'est-à-dire à considérer le problème suivant :

$$\boxed{\begin{array}{l}\text{minimiser } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{sous les contraintes } y_i\left(w^T x_i - b\right) \geq 1 - \xi_i, \forall i\end{array}} \tag{4.19}$$

Les contraintes $y_i\left(w^T x_i - b\right) \geq 1 - \xi_i$ sont plus douces que dans le cas précédent où on impose $y_i\left(w^T x_i - b\right) \geq 1$, d'où le nom de l'algorithme (on parle de classifieur "hard-margin" dans le cas de contraintes de la forme $y_i\left(w^T x_i - b\right) \geq 1$). Le Lagrangien associé à ce problème d'optimisation s'écrit :

$$L\left(\widetilde{w}, \alpha, \beta, \xi\right) = \frac{1}{2}w^T w + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i\left(w^T x_i - b\right) - 1 + \xi_i\right] - \sum_{i=1}^{n}\beta_i\xi_i$$

dans lequel le terme $\frac{1}{2}w^T w + C\sum_{i=1}^{n}\xi_i$ représente la fonction à minimiser ($C\sum_{i=1}^{n}\xi_i$ est le terme de pénalisation), le terme $\sum_{i=1}^{n}\alpha_i\left[y_i\left(w^T x_i - b\right) - 1 + \xi_i\right]$ traduit la contrainte sur les marges et $\sum_{i=1}^{n}\beta_i\xi_i$ traduit la contrainte de positivité des variables ξ_i. Il est important de noter que les multiplicateurs de Lagrange α_i et β_i sont positifs ou nuls. En annulant les dérivées partielles de $L\left(\widetilde{w}, \alpha, \beta, \xi\right)$ par rapport à b, w et ξ, on obtient :

$$\sum_{i=1}^{n}\alpha_i y_i = 0$$

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

$$\alpha = C\mathbb{I} - \beta$$

avec les notations évidentes $\alpha = (\alpha_1, ..., \alpha_n)^T$, $\beta = (\beta_1, ..., \beta_n)^T$ et $\mathbb{I} = (1, ..., 1)^T$. Puisque les multiplicateurs de Lagrange β_i vérifient $\beta_i \geq 0$, la dernière égalité donne les contraintes suivantes

$$\alpha_i \leq C, \qquad \forall i = 1, ..., n$$

Comme pour le classifieur "hard-margin", en remplaçant l'expression de w dans le Lagrangien et en prenant en compte les conditions de Kühn et Tucker, on obtient le problème suivant

$$\boxed{\begin{array}{l} \text{Maximiser } U(\alpha) = -\frac{1}{2}\alpha^T Y \left(xx^T\right) Y^T \alpha + \sum_{i=1}^{n} \alpha_i \\ \text{sous les contraintes } 0 \leq \alpha_i \leq C, \forall i = 1, ..., n \end{array}}$$

Parler de la résolution de ce problème en décomposant en problèmes à 2 dimensions

Remarque 1 **:** on peut aussi introduire un terme de pénalisation quadratique qui conduit au problème d'optimisation suivant :

$$\boxed{\begin{array}{l} \text{Minimiser } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i^2 \\ \text{sous les contraintes } y_i\left(w^T x_i - b\right) \geq 1 - \xi_i, \forall i \end{array}}$$

Un tel problème est par exemple résolu dans ([14], p. 55).

Remarque 2 : Un algorithme d'optimisation permettant de résoudre ce problème s'appelle algorithme SMO (Sequential Minimal Optimisation) et a été développé par John Platt [21].

2.5 Cas non séparable : Classifieur ν-SVM

Un inconvénient majeur du classifieur "soft margin SVM" est le manque de contrôle du nombre de points qui sont considérés comme des points aberrants et qui sont à l'intérieur de la zone délimitée par la marge (outliers) (ces points aberrants vérifient $y_i\left(w^T x_i - b\right) < 1$ et il y a égalité si x_i est un vecteur support). Une façon de contrôler le nombre de points aberrants consiste à pénaliser le critère à optimiser par un terme fonction de la marge γ, ce qui peut se traduire par le problème suivant :

$$\boxed{\begin{array}{l} \text{Minimiser } \frac{1}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i - \nu\gamma \\ \text{sous les contraintes } y_i\left(w^T x_i - b\right) \geq \gamma - \xi_i, \forall i \end{array}} \tag{4.20}$$

On montre que dans ce problème, le paramètre ν est une borne supérieure du nombre de points aberrants donné par $\frac{1}{n}\text{Card}\left\{(x_i, y_i) \left| y_i \left(w^T x_i - b\right) < 1 \right.\right\}$. Pour chaque valeur de ν, il existe une valeur de C telle que la solution du problème (4.19) et la solution du problème (4.20) conduisent à la même marge. On pourrait donc utiliser plusieurs valeurs de C dans (4.19) pour obtenir une valeur adéquate du nombre de points aberrants. L'avantage du problème (4.20) est que la variable de contrôle ν est optimisée directement dans le programme d'optimisation avec une simplicité remarquable. Le Lagrangien associé au problème (4.20) s'écrit :

$$\begin{aligned} L\left(\widetilde{w}, \alpha, \beta, \xi, \gamma, \delta\right) &= \frac{1}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n}\xi_i - \nu\gamma \\ &\quad - \sum_{i=1}^{n}\alpha_i\left[y_i\left(w^T x_i - b\right) - \gamma + \xi_i\right] - \sum_{i=1}^{n}\beta_i\xi_i - \delta\gamma \end{aligned}$$

En annulant les dérivées partielle de ce lagrangien par rapport à $\widetilde{w}, \xi, \gamma$, on obtient

$$\begin{aligned} \frac{\partial L}{\partial w} &= 0 \Rightarrow \boxed{w = \textstyle\sum_{i=1}^{n}\alpha_i y_i x_i} \\ \frac{\partial L}{\partial b} &= 0 \Rightarrow \sum_{i=1}^{n}\alpha_i y_i = 0 \\ \frac{\partial L}{\partial \xi} &= 0 \Rightarrow \frac{1}{n}\mathbb{I} - \alpha - \beta = 0 \Rightarrow \boxed{\alpha = \tfrac{1}{n}\mathbb{I} - \beta} \\ \frac{\partial L}{\partial \gamma} &= 0 \Rightarrow \sum_{i=1}^{n}\alpha_i - \nu - \delta = 0 \Rightarrow \boxed{\textstyle\sum_{i=1}^{n}\alpha_i = \nu + \delta} \end{aligned}$$

On remplace ces variables dans le lagrangien pour obtenir

$$\boxed{U(\alpha) = -\tfrac{1}{2}\alpha^T Y\left(xx^T\right)Y^T\alpha + \textstyle\sum_{i=1}^{n}\alpha_i}$$

que l'on doit maximiser sous les contraintes

$$\begin{aligned} \boxed{0 \leq \alpha_i \leq \tfrac{1}{n},\ \forall i} \ (\text{car } \beta_i &\geq 0) \\ \boxed{\textstyle\sum_{i=1}^{n}\alpha_i \geq \nu} \ (\text{car } \delta &\geq 0) \end{aligned}$$

Pour plus de détails, on pourra se référer à [29].

2.6 Cas non séparable : prétraitement non-linéaire

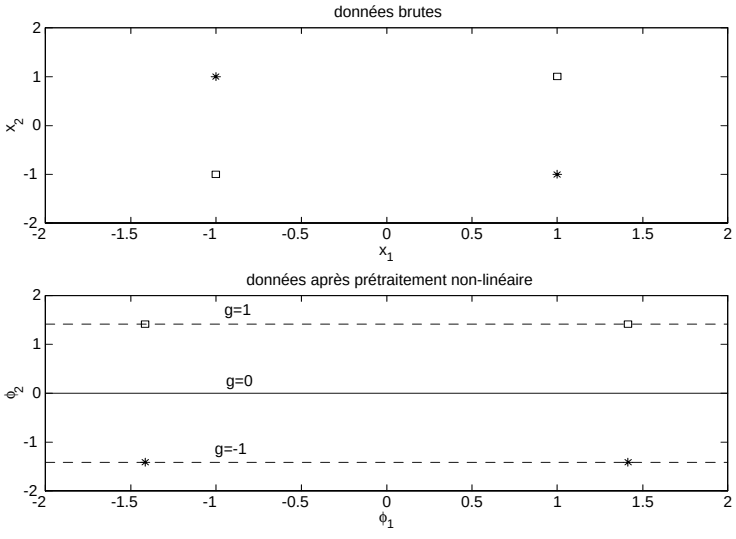
Comme nous l'avons vu précédemment, la théorie de base des machines à vecteurs supports ne fonctionne que si les données des deux classes sont linéairement séparables. Dans le cas non séparable, on peut utiliser les machines à vecteurs supports mais en prenant soin transformer les données x_i à l'aide d'un prétraitement non linéaire $\phi : \mathbb{R}^p \to \mathbb{R}^q$. Un exemple classique appelé problème XOR consiste à séparer les deux classes ω_1 et ω_2 à partir des données expertisées $\chi_1 = \{(1,1),(-1,-1)\}$ et $\chi_2 = \{(1,-1),(-1,1)\}$. Il est clair que χ_1 et χ_2 ne sont pas linéairement séparables. Considérons l'application ϕ définie par :

$$\phi : \begin{array}{l} \mathbb{R}^2 \to \mathbb{R}^6 \\ (x_1, x_2)^T \mapsto \left(\sqrt{2}x_1, \sqrt{2}x_1x_2, 1, \sqrt{2}x_2, x_1^2, x_2^2\right)^T \end{array}$$

Pour cet exemple, l'hyperplan maximisant la marge dans l'espace des variables est d'équation

$$g(x_1, x_2) = x_1x_2 = 0$$

et la marge correspondante est $\gamma = \sqrt{2}$. Une projection des données de $\mathbb{R}^6$ dans le plan $(\phi_1, \phi_2) = \left(\sqrt{2}x_1, \sqrt{2}x_1x_2\right)$ ainsi que les données d'origine sont représentées ci-dessous :



Dans l'espace des variables, les ensembles χ_1 et χ_2 sont clairement linéairement séparables. On peut donc appliquer l'algorithme du perceptron sur les

données $\phi(x_i)$ ou l'algorithme des machines à vecteurs supports. Dans le cas des machines à vecteurs supports, la règle de classification s'écrit :

$$\begin{aligned} f(\tilde{x}) &= \text{sign}\left(g_{w,b}(\tilde{x})\right) = \text{sign}\left(w^T\phi(x) - b\right) \\ &= \text{sign}\left(\sum_{\text{vecteurs supports}} \alpha_i^0 y_i \phi(x_i)^T \phi(x) - b_0\right) \end{aligned}$$

avec $\tilde{x} = \phi(x)$. Comme on peut le remarquer, la règle de décision est une fonction des produits scalaires $\phi(x_i)^T \phi(x)$. La fonction à optimiser pour le classifieur ν-SVM dans le cas d'un prétraitement non-linéaire s'écrit

$$\boxed{U(\alpha) = -\tfrac{1}{2}\alpha^T Y G Y^T \alpha + \textstyle\sum_{i=1}^n \alpha_i}$$

où G est une matrice appelée parfois matrice de Gram définie par $G_{ij} = \phi(x_i)^T \phi(x_j)$ que l'on doit maximiser sous les contraintes

$$\begin{aligned} \boxed{0 \leq \alpha_i \leq \tfrac{1}{n}, \forall i} \ (\text{car } \beta_i &\geq 0) \\ \boxed{\textstyle\sum_{i=1}^n \alpha_i \geq \nu} \ (\text{car } \delta &\geq 0) \end{aligned}$$

Donc, la fonction à optimiser dépend des produits scalaires $\phi(x_i)^T \phi(x_j)$. Dans le cas d'un prétraitement non-linéaire, les remarques précédentes indiquent qu'il n'est pas nécessaire de calculer explicitement les variables $\phi(x_i)$ et qu'il suffit de déterminer les produits scalaires $\phi(x_i)^T \phi(x_j)$ et $\phi(x_i)^T \phi(x)$, x étant un vecteur à classifier. On introduit alors un noyau défini comme suit

$$k(x, y) = \phi(x)^T \phi(y) \tag{4.21}$$

qui permet de calculer les produits scalaires $\phi(x_i)^T \phi(x)$ et $\phi(x_i)^T \phi(x)$. A titre d'exemple, si $x = (x_1, x_2)^T$ et $\phi(x) = \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right)$, un calcul élémentaire permet d'obtenir

$$\phi(x)^T \phi(y) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} . \begin{pmatrix} y_1^2 \\ y_2^2 \\ \sqrt{2}y_1y_2 \end{pmatrix} = x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2$$

et alors on peut poser

$$k(x, y) = \left(x^T y\right)^2$$

La question fondamentale que l'on est amené à se poser est alors la suivante : "quelles sont les conditions que doit vérifier k pour qu'il existe ϕ tel que

(4.21) soit vérifié. La réponse à cette question est donnée dans le théorème suivant (appelé théorème de Mercer) :
Théorème ([14], p. 35) : *un noyau k défini de $\chi \times \chi \rightarrow R$ est un noyau de Mercer si et seulement si $\forall r \in N$ et $x = (x_1, ..., x_r) \in \chi^r$, la matrice $K = (k(x_i, x_j))_{i,j=1}^{r}$ est semi-définie positive.*

Les noyaux habituellement utilisés sont rappelés ci-dessous dans le cas $\dim(\chi) = p$ (voir ([34], p. 149) ou ([14], p. 38):

Nom du noyau	**Expression**	$\dim(\phi(x))$
Polynomial (de degré p)	$k(x,y) = (\langle x, y\rangle)^q$ $q \in \mathbb{N}^+$	$\mathcal{C}_{p+q-1}^q$
Polynomial complet	$k(x,y) = (\langle x, y\rangle + c)^q$ $c \in \mathbb{R}^+$, $q \in \mathbb{N}^+$	$\mathcal{C}_{p+q}^q$
RBF	$k(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$ $\sigma \in \mathbb{R}^+$	∞
Mahalanobis	$k(x,y) = \exp\left(-(x-y)^T \Sigma (x-y)\right.$ $\Sigma = \text{diag}\left(\frac{1}{\sigma_1^2}, ..., \frac{1}{\sigma_p^2}\right)$, $\sigma_i \in \mathbb{R}^+$	∞

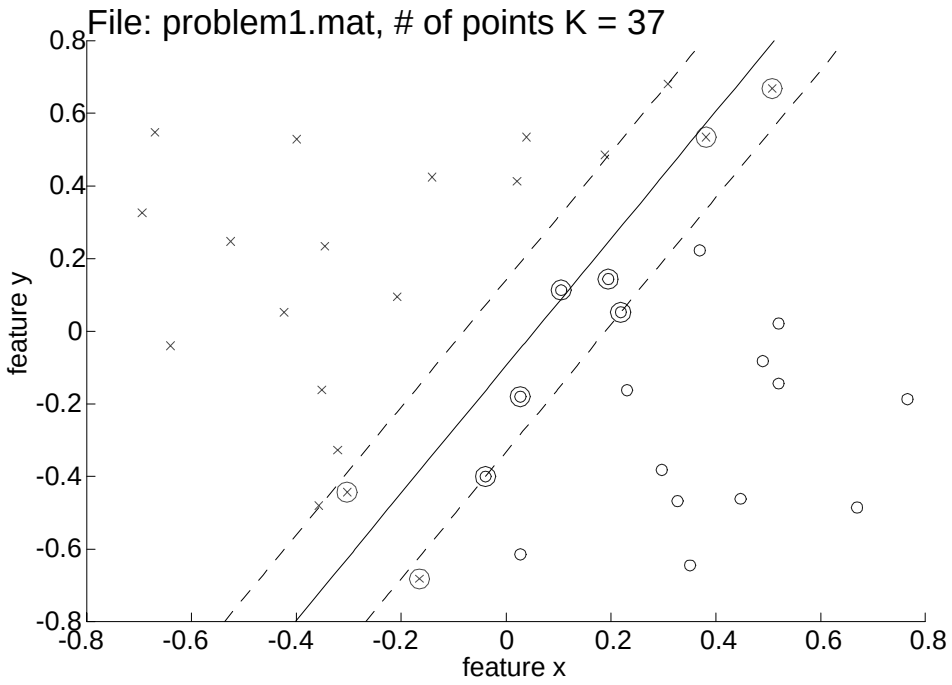
Une fois que le noyau k est choisi et que le problème d'optimisation est résolu, on peut déterminer la frontière de décision entre les deux classes qui est définie par

$$\sum_{\text{vecteurs supports}} \alpha_i^0 y_i \phi(x_i)^T \phi(x) - b_0 = 0$$

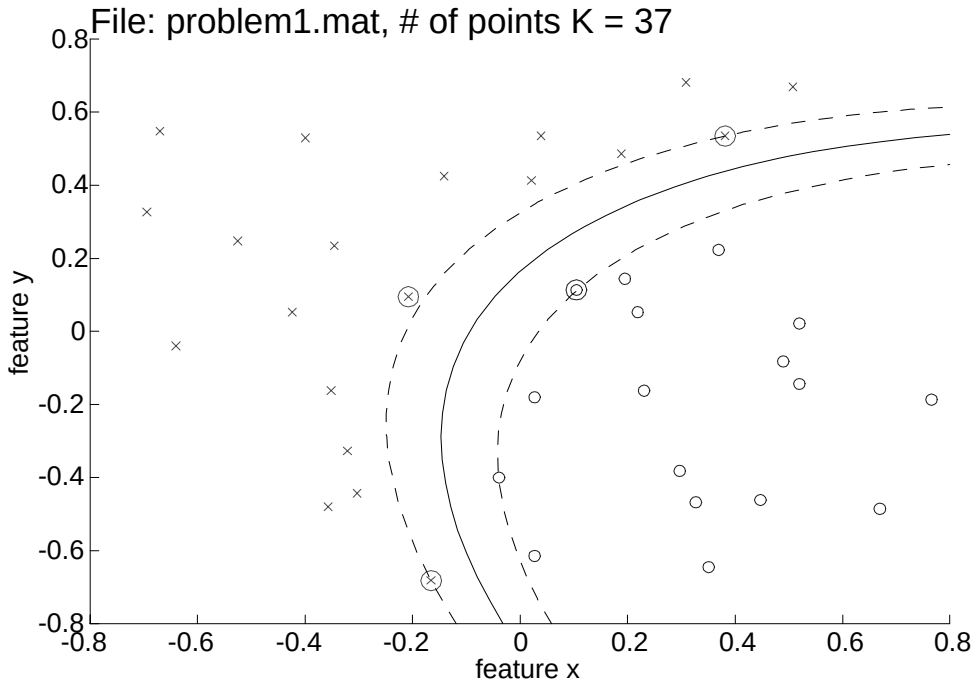
ou

$$\sum_{\text{vecteurs supports}} \alpha_i^0 y_i k(x_i, x) - b_0 = 0$$

Deux exemples de frontières de décisions avec leurs marges sont représentés ci-dessous. Le premier exemple correspond à l'application de l'algorithme de base sans prétraitement tandis que le second exemple montre l'amélioration obtenue en considérant un noyau polynomial de degré 2.



Représentation de la frontière de décision et de la marge
dans le cas d'un classifieur SVM linéaire (absence de prétraitement).



Représentation de la frontière de décision et de la marge
dans le cas d'un classifieur SVM avec prétraitement polynomial d'ordre 2.

Remarque : Soit $f \in \mathcal{F} = \left\{ g : \begin{array}{l} \chi \to \mathbb{R} \\ x \mapsto g(x) = \langle \phi(x), \tilde{w} \rangle \end{array} \text{ avec } \|\tilde{w}\| = 1 \right\}$, ensemble des fonctions discriminantes linéaires de χ dans $\mathbb{R}$ (avec éventuel prétraitement ϕ). Le théorème de représentation de Mercer ([14], p. 48) indique que toute fonction f minimisant le risque régularisé

$$R_{reg}(f) = g_{emp}\left(\{x_i, y_i, f(x_i)\}_{i\in\{1,\ldots,n\}}\right) + g_{reg}(\|f\|)$$

(où x_i sont les vecteurs de la base d'apprentissage et $y_i \in \{-1, +1\}$ leurs classes) admet une représentation de la forme

$$f(.) = \sum_{i=1}^{n} \alpha_i k(x_i, .), \qquad \alpha = (\alpha_1, ..., \alpha_n) \in \mathbb{R}^n \tag{4.22}$$

Donc, tout algorithme d'apprentissage $\mathcal{A}$ (qui est est rappelons le une application définie de $\mathcal{B}$ (base d'apprentissage) dans $\mathcal{F}$ (ensemble des fonctions discriminantes)) cherchant une fonction discriminante minimisant un risque régularisé aura une solution possédant la forme (4.22). Le classifieur associé aura alors une règle de décision définie par $d(x) = sign(f(x))$. Le théorème de représentation de Mercer indique que dans la mesure où l'apprentissage se fait par minimisation d'un coût régularisé, on a une solution qui se décompose sous la forme d'une somme pondérée de noyaux. Les coefficients α_i seront alors déterminés à l'aide du problème dual (voir ce qui précède).

2.7 Capacités de généralisation du classifieur SVM

Dans le cas séparable, le classifieur SVM se fixe l'erreur empirique $R_{emp}(\alpha) = 0$. On a dans ce cas

$$\begin{aligned} R_{emp}(\alpha) + \frac{\varepsilon}{2}\left(1 + \sqrt{1 + \frac{4R_{emp}(\alpha)}{\varepsilon}}\right) &= \varepsilon \\ &= 4\frac{h\left(\ln\frac{2n}{h} + 1\right) - \ln(\eta/4)}{n} \end{aligned}$$

et donc

$$R(\alpha) \leq 4\frac{h\left(\ln\frac{2n}{h} + 1\right) - \ln(\eta/4)}{n}$$

Pour avoir des bonnes propriétés de généralisation, il faut que cette borne soit faible. Puisque n est fixé, on doit essayer de minimiser la dimension de VC h. C'est ce que fait le classifieur SVM qui détermine l'hyperplan de marge

maximale. En effet, alors que la dimension de l'ensemble des hyperplans séparateurs est $n+1$, la dimension de VC de l'ensemble des hyperplans séparateurs de marge Δ dans le cas où le vecteur d'observation x appartient à une sphère de rayon R est telle que

$$h \leq \min\left(\text{ent}\left(\frac{R^2}{\Delta^2}\right), n\right) + 1 \tag{4.23}$$

où $\text{ent}\left(\frac{R^2}{\Delta^2}\right)$ désigne la partie entière de $\frac{R^2}{\Delta^2}$. Donc, une fois qu'on a déterminé l'hyperplan séparateur, on peut calculer Δ et en déduire une borne sur h. En maximisant la marge, on minimise $\frac{1}{\Delta^2}$ et donc on a tendance à minimiser h.

Dans le cas non-séparable, on utilise un prétraitement $\phi(x)$ et on cherche à nouveau à obtenir un risque empirique nul. Dans ce cas, le choix de ϕ peut se faire par minimisation d'une estimation de la dimension de VC h. Un exemple de reconnaissance de caractères est traité dans (voir ([34], p. 151)) à l'aide de prétraitements polynomiaux. Les données sont séparables dès que l'ordre du polynome dépasse 3. Dans ce cas, le classifieur de dimension h minimale donne les meilleures performances. Notons au passage que dans cet exemple, la dimension de VC de la structure est bien inférieure au nombre de paramètres libres de cette structure (on peut parfois avoir l'inverse comme pour l'ensemble des fonctions $f(z,\alpha) = \theta \sin(\alpha z), \alpha \in \Lambda$ qui est de dimension de VC infinie).

3 Réseaux Neuronaux

Le cerveau humain est une structure composée d'un nombre très important de cellules appelées neurones. Ces neurones sont organisés en un immense réseau dont les fonctionnalités sont extrêmement compliquées et diverses. Toute cette structure pense, apprend et mémorise. Un réseau de neurones artificiel est un modèle du cerveau humain. Les composantes élémentaires de ce réseau sont des unités de calcul qui reçoivent en permanence des entrées et qui produisent des sorties. Les composantes élémentaires appelées neurones calculent leur propre état en fonction des autres composantes auxquelles elles sont connectées. Un réseau de neurones est défini par 1) **sa structure** (nombre de noeuds, de couches, ...), 2) **les règles d'apprentissage (**ces règles spécifient les valeurs initiales des poids et leur évolution) et 3) **les règles de classification**.

3.1 Définition d'un réseau de neurones

Cellule Elémentaire

la cellule élémentaire d'un réseau de neurones est représentée sur la figure 4.4. Elle est constituée de $p-1$ entrées $x_1, ..., x_{p-1}$, d'une sortie y, de $p-1$ poids synaptiques $w_1, ..., w_{p-1}$ et d'un biais a (a est parfois appelé offset). La sortie de la cellule élémentaire est définie par :

$$y = f\left(\sum_{i=1}^{p-1} w_i x_i - a\right) \tag{4.24}$$

f étant généralement choisie parmi les trois non-linéarités de la figure 4.5.

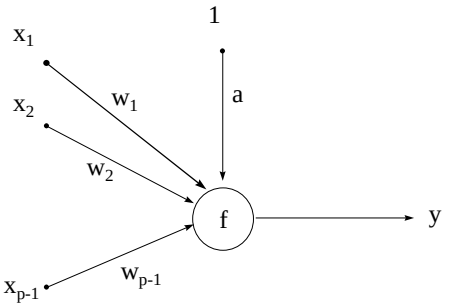


Fig. 4.4 Cellule élémentaire

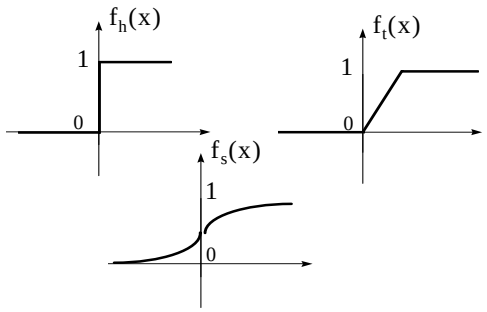


Fig 4.5 Non linéarités usuelles a) Echelon b) Seuil logique c) Sigmoide

La fonction sigmoïde peut être définie de différentes façons comme $f_s(x) = \frac{1}{1+e^{-\alpha x}}$ (fonction logistique) ou $f_s(x) = \frac{1+th(\alpha x)}{2}$. On remarquera que l'équation (4.24) peut s'écrire

$$y = f\left(\sum_{i=1}^{p} w_i x_i\right) \tag{4.25}$$

avec $x_p = -1$ et $w_p = a$.

Perceptron à une couche

Le perceptron à une couche [19] est constitué de la mise en parallèle de cellules élémentaires :

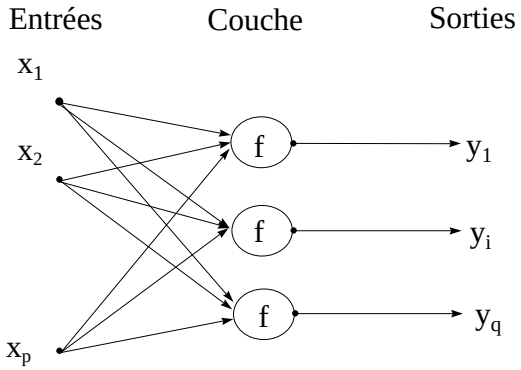


Fig. 4.6 Perceptron à une couche

Il est défini par q sorties $y_1, ..., y_q$ reliées aux entrées par l'intermédiaire de poids et de non-linéarités. On a dans ce cas :

$$y_i = f\left(\sum_{j=1}^{p} w_{ji} x_j\right) \tag{4.26}$$

avec $x_p = -1$ et $w_{pi} = a_i$. Dans le cas d'une seule sortie ($q = 1$), le perceptron à une couche est la cellule élémentaire.

Perceptron Multicouches

Le perceptron multicouches est un réseau avec une ou plusieurs couches reliant les noeuds d'entrée et de sortie. Il a la particularité d'avoir des couches cachées qui ne sont pas reliées aux sorties. Un exemple de perceptron possédant deux couches cachées et une couche de sortie est représenté sur la figure suivante :

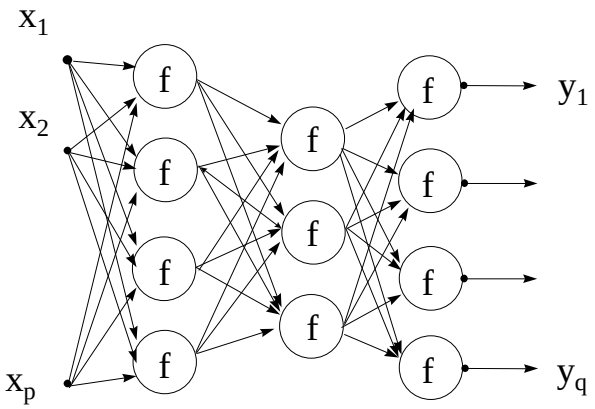


Fig. 4.8 Perceptron Multicouches

3.2 Choix de la structure du réseau

Définir la structure d'un réseau de neurones, c'est choisir le nombre de couches, le nombre de noeuds par couche et le nombre de sorties. Pour définir cette structure, il est indispensable d'avoir une idée des régions de décision générées par un réseau c'est-à-dire des régions définies par $d(x) = a_i$. Dans le cas d'une classification sans rejet, on a K régions de décisions partitionnant l'espace de mesure $X = \mathbb{R}^p$ en K parties correspondant à $d(x) = a_1$ (on décide que la forme représentée par x est la classe ω_1), $d(x) = a_2$ (on décide que la forme représentée par x est la classe ω_2), ..., $d(x) = a_K$ (on décide que la forme représentée par x est la classe ω_K).

Cellule élémentaire avec une non linéarité de type échelon

La sortie de la cellule est $y = 0$ ou $y = 1$. Une telle cellule permet donc de construire un classifieur pour un problème à deux classes ω_1 et ω_2 selon

la règle suivante :

$$d(x) = a_1 \Longleftrightarrow y = 1 \Longleftrightarrow \sum_{i=1}^{p-1} w_i x_i - a > 0 \tag{4.27}$$

$$d(x) = a_2 \Longleftrightarrow y = 0 \Longleftrightarrow \sum_{i=1}^{p-1} w_i x_i - a < 0 \tag{4.28}$$

La frontière entre les deux régions de décision $d(x) = a_1$ et $d(x) = a_2$ est un hyperplan d'équation $\sum_{i=1}^{p-1} w_i x_i - a = 0$. Par exemple dans le cas de vecteurs de $\mathbb{R}^2$, la frontière entre les deux régions de décision est une droite d'équation $w_1 x_1 + w_2 x_2 - a = 0$. Cette droite sépare le plan en deux hyperplans correspondant aux classes ω_1 et ω_2. Cette étude montre que lorsque les deux classes peuvent être séparées par un hyperplan la cellule élémentaire est suffisante pour la classification.

Perceptron à deux couches et une sortie avec des non-linéarité de type échelon

La cellule élémentaire avec une non linéarité de type échelon génère des frontières de décision qui sont des hyperplans. Dans le cas de deux couches, le problème est plus compliqué. Tout d'abord toute région de décision de type **polygone convexe** délimité par des hyperplans peut-être générée. En effet, il suffit de construire un réseau (en choisissant convenablement les poids et offsets) pour lequel 1) chaque noeud de la première couche se comporte comme une cellule élémentaire dont la sortie est haute (+1) lorsque les entrées sont situées d'un coté d'un hyperplan délimitant ce polygone 2) et d'effectuer ensuite une opération "ET Logique" définie par :

$$y = f\left(\sum_{i=1}^{N_1} w_i x_i\right) = 1 \Longleftrightarrow x_i = 1 \qquad \forall i \in \{1, ..., N_1\} \tag{4.29}$$

Cette opération peut être effectuée en prenant des poids constants égaux à 1 et un offset égal à $N_1 - \varepsilon$ avec $0 < \varepsilon < 1$. La région de décision est alors un polygone convexe qui a autant de cotés que de noeuds dans la seconde couche. Cette analyse montre qu'on peut à l'aide d'un perceptron à deux couches avec des non-linéarités de type échelon générer toute région de décision polygonale convexe.

Mais on peut également générer d'autres régions de décision que des polygones convexes avec un perceptron à deux couches (voir [1], p. 125). Par exemple, on peut obtenir des régions de décisions qui ont la forme d'un polygone non convexe ou même qui sont constituées de la réunion de plusieurs

polygones. Malheureusement, on ne peut générer des régions de décisions arbitraires avec un tel réseau lorsque les non-linéarités sont des échelons (voir [1], p. 126). Ce résultat est à prendre avec précautions car lorsqu'on remplace les non-linéarités par des sigmoides, on peut approcher avec la précision voulue toute région de décision (voir [1], p. 126).

Perceptron à trois couches et une sortie avec des non-linéarités de type échelon

Le perceptron à trois couches et une sortie avec des non-linéarités de type échelon permet d'obtenir des régions de décision plus complexes. Par exemple, toute région de décision pouvant être partitionnée en petits hypercubes (carrés (n=2), cubes (n=3), ...) peut être générée. Chaque hypercube nécessite 2N noeuds dans la première couche (4 noeuds pour obtenir un carré) à savoir un pour chaque coté de l'hypercube et un noeud dans la seconde couche pour le ET logique. Une sortie de la seconde couche est haute lorsque les entrées sont dans l'hypercube associé. Ensuite, on effectue un "OU logique" entre les différents hypercubes :

$$y = f\left(\sum_{i=1}^{N_1} w_i x_i\right) = 1 \Longleftrightarrow \exists i \in \{1, ..., N_2\} \text{ tel que } x_i = 1 \qquad (4.30)$$

Cette opération OU logique peut être effectuée en prenant tous les poids égaux à 1 et un offset égal à ε avec $0 < \varepsilon < 1$. L'analyse exposée ci-dessus montre qu'il n'est pas nécessaire en général d'utiliser un nombre de couches supérieur à 3. En effet, toute région de décision peut être approchée avec la précision voulue par une réunion de petits hypercubes. De plus, toute région de décision définie par une réunion de polygones convexes peut être construite de la façon suivante :

1) le nombre de noeuds de la première couche fixe le nombre de cotés de chaque polygone convexe de la région de décision (on effectue ensuite un ET logique pour constuire chaque polygone),

2) le nombre de noeuds de la seconde couche est égal au nombre de polygones convexes qui constituent la région de décision (on effectue ensuite un OU logique pour constuire la réunion des polygones)

Remarques

1) L'analyse exposée ci-dessus concerne le perceptron multicouches avec des nonlinéarités de type échelon. La même analyse peut être effectuée avec des non linéarités de type sigmoïde. On utilise alors généralement la règle de

classification suivante :

$$\begin{array}{l} d(x) = a_1 \Longleftrightarrow y < \frac{1}{2} \\ d(x) = a_2 \Longleftrightarrow y > \frac{1}{2} \end{array} \tag{4.31}$$

Les régions de décision sont alors limitées par des lignes quelconques et non plus par des lignes brisées. Dans ce cas, toute région de décision peut être approchée par un perceptron à deux couches. Cependant, ce perceptron à deux couches n'est pas toujours optimal au sens de la rapidité de convergence (voi règles d'apprentissage) ou de la facilité d'implantation (il est souvent préférable d'utiliser un perceptron constitué d'un faible nombre de noeuds).

2) L'analyse exposée ci-dessus permet de construire un classifieur à deux classes à l'aide d'une sortie. Dans le cas général de $K > 2$ classes, on utilise la plupart du temps un nombre de sorties égal au nombre de classes et des non-linéarités de type sigmoïde. La classe choisie est alors celle qui correspond à la sortie maximale. Cependant, on peut imaginer d'autres stratégies comme par exemple utiliser une sortie unique avec une non-linéarité de type sigmoïde et choisir la classe à partir de la valeur de la sortie dans l'intervalle $]0, 1[$. Par exemple, pour trois classes, on peut adopter la règle suivante :

$$\begin{array}{rcl} d(x) & = & a_1 \Longleftrightarrow y < 1/3 \\ d(x) & = & a_2 \Longleftrightarrow y \in]1/3, 2/3[\\ d(x) & = & a_3 \Longleftrightarrow y > 2/3 \end{array}$$

3.3 Comparaison avec le classifieur Bayésien

Considérons le problème à deux classes pour lequel les deux densités de probabilité conditionnelles sont gaussiennes de moyennes respectives m_1 et m_2 et de même matrice de covariance Σ. Dans le cas de classes équiprobables, le problème se réduit à (voir eq. 3.29) :

$$d^*(x) = a_1 \Leftrightarrow \left(x - \frac{1}{2}(m_1 + m_2)\right)^t \Sigma^{-1}(m_1 - m_2) \geq \ln \frac{P(\omega_2)}{P(\omega_1)} \tag{4.32}$$

Cette règle de décision s'écrit :

$$d(x) = a_1 \Leftrightarrow \sum_{j=1}^{p-1} w_j x_j - a \geq 0 \tag{4.33}$$

avec $a = \frac{1}{2}(m_1 + m_2)^t \Sigma^{-1}(m_1 - m_2) + \ln \frac{P(\omega_2)}{P(\omega_1)}$ et $w = \Sigma^{-1}(m_1 - m_2)$. On voit donc que le classifieur Bayésien est équivalent dans ce cas à un classifieur construit à partir d'un perceptron à une couche et une sortie.

De manière plus générale, le classifieur Bayésien pour deux classes est défini par

$$d(x) = a_1 \Leftrightarrow P(\omega_1|\, x) - P(\omega_2|\, x) \geq 0$$

où $g(x) = P(\omega_1|\, x) - P(\omega_2|\, x)$ est une fonction généralement non-linéaire par rapport à x. Le classifieur neuronal cherche à approximer cette fonction $g(x)$ par la sortie d'un réseau de neurones. Cette démarche peut être justifiée théoriquement par le théorème d'approximation universelle (voir p. 53).

3.4 Règles d'Apprentissage

On appelle règle d'apprentissage d'un réseau de neurones, la règle permettant de mettre à jour les poids du réseau à l'itération $n+1$ à partir des valeurs de l'instant n. Une telle règle a déjà été présentée pour le perceptron à une couche avec une non-linéarité $f_h(x)$. Malheureusement, on ne peut généraliser cette règle pour des réseaux multi-couches. Cette partie étudie les règles d'apprentissage dans le cas où les non-linéarités sont des sigmoides.

Perceptron à une couche avec non-linéarité $f_s(x)$

On se limite dans cette partie à la fonction logistique définie par :

$$f_s(x) = \frac{1}{1+\exp(-\alpha x)}$$

dont la dérivée vérifie $f_s'(x) = \alpha f_s(x)\left[1 - f_s(x)\right]$. La minimisation de la fonction de coût $J(w) = \frac{1}{2}E\left[e^2(n)\right] = \frac{1}{2}E\left[\left(d(n) - y(n)\right)^2\right]$ conduit à un système d'équations non-linéaires qui n'admet pas de solution analytique explicite (en théorie, on n'est même pas sûr de l'existence d'un minimum de $J(w)$, un contre-exemple étant $J(w) = e^{-w}$). Dans les applications pratiques, on espère que le critère $J(w)$ admet un ou plusieurs minima globaux et on cherche à estimer ce ou ces minima. On utilise alors généralement l'algorithme LMS pour lequel la mise à jour des poids $w_i(n)$ s'écrit :

$$\widehat{w}_k(n+1) = \widehat{w}_k(n) + \mu e(n) x_k(n) y(n)\left[1 - y(n)\right]$$

Il est important de noter que, même si le critère possède un ou plusieurs minima globaux, l'algorithme LMS peut ne pas converger vers ce ou ces minima et rester bloqué autour d'un minimum local de $J(w)$. On peut se demander si les performances de l'algorithme du perceptron sont meilleures lorsqu'on utilise une non-linéarité sigmoidale $f_s(x)$ par rapport à un seuil logique $f_h(x)$.

J. Shynk and N. Bershad [30] ont démontré que les performances des algorithmes étaient similaires quelle que soit la non-linéarité utilisée. Donc, tant qu'on se limite à une cellule élémentaire, on ne peut espèrer de bons résultats de classification que dans le cas où les classes sont linéairement séparables ([13], p. 149).

Algorithme de Rétropropagation du Gradient

L'algorithme de rétropropagation du gradient est une généralisation de l'algorithme LMS présenté précédemment. Dans le cas d'une seule sortie, l'erreur instantanée du réseau $\frac{1}{2}e^2(n) = \frac{1}{2}[d(n) - y(n)]^2$ est fonction de tous les poids du réseau. L'algorithme de rétropropagation du gradient consiste à mettre à jour les poids du réseau de façon à minimiser la fonction de coût $\frac{1}{2}E[e^2(n)]$, à l'aide de l'algorithme LMS. L'algorithme utilise des non-linéarités différentiables comme par exemple la fonction logistique $f_s(x) = \frac{1}{1+\exp(-\alpha x)}$ et peut se résumer de la façon suivante :

1) *Initialisation des poids du réseau*

Les poids $w_{ij}^{(l)}(0)$ liant la $(l-1)^{ème}$et la $l^{ème}$ couche sont choisis aléatoirement de faible valeur,

2) *Présentation d'une entrée de la base d'apprentissage*

On présente une entrée $x(n)$ dont la classe est connue. On définit la sortie désirée $d(n)$ de la façon suivante :

$$\begin{array}{l} d(n) = 1 \text{ si } x(n) \in \omega_1 \\ d(n) = 0 \text{ si } x(n) \in \omega_2 \end{array} \tag{4.34}$$

3) *Calcul de la sortie du réseau*

Supposons que le réseau comporte L couches et que la $l^{ème}$ couche contienne N_l noeuds. Soit $y_j^{(l)}$ la $j^{ème}$ sortie de la $l^{ème}$ couche telle que :

$$\begin{array}{l} y^{(L)}(n) = y(n) \\ y_j^{(0)}(n) = x_j(n) \end{array} \tag{4.35}$$

On a alors :

$$y_j^{(l)}(n) = f_s\left(\sum_{i=1}^{N_{l-1}} w_{ij}^{(l)}(n)\, y_i^{(l-1)}(n)\right) \tag{4.36}$$

et en particulier pour la sortie de la dernière couche

$$y(n) = f_s\left(\sum_{i=1}^{N_{L-1}} w_{i1}^{(L)}(n)\, y_i^{(L-1)}(n)\right) \tag{4.37}$$

4) *Mise à jour des poids*

$$\widehat{w}_{ij}^{(l)}(n+1) = \widehat{w}_{ij}^{(l)}(n) + \mu\delta_j^{(l)}(n)\, y_i^{(l-1)}(n) \tag{4.38}$$

avec :

$$\delta^{(L)}(n) = e(n)y(n)[1-y(n)] \tag{4.39}$$

$$\delta_j^{(l)}(n) = y_j^{(l)}(n)\left[1-y_j^{(l)}(n)\right]\sum_k \delta_k^{(l+1)}(n)\, w_{jk}^{(l+1)}(n) \tag{4.40}$$

La sommation se faisant sur tous les termes d'erreur de la $(l+1)^{ème}$ couche $e(n) = d(n) - y(n)$ est l'erreur en sortie du réseau.

5) *Retour à 2) jusqu'à convergence.*

A titre d'exemple, on donne les régions de décision obtenues avec un perceptron à deux couches après 50,100, 150 et 200 mises à jour réalisées à l'aide de l'algorithme de rétropropagation du gradient. Les entrées de la première classe sont uniformément réparties dans un disque de rayon 1 centré sur l'origine. Les entrées de la seconde classe sont uniformément réparties dans un disque de rayon 5 centré sur l'origine privé du disque précédent :

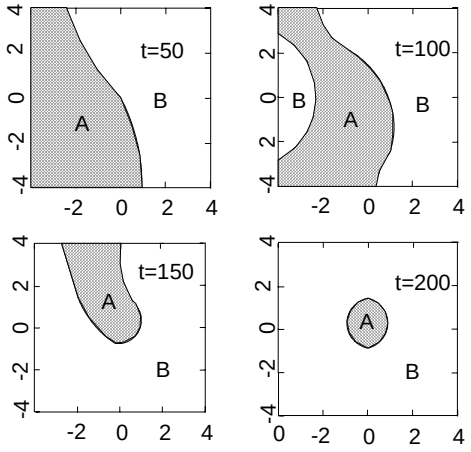


Fig. 4.10 Exemple d'évolution des frontières de décision pour le perceptron multicouches

Une difficulté rencontrée avec cet algorithme est qu'il est parfois nécessaire de présenter un grand nombre de fois la séquence d'apprentissage pour avoir convergence (dans les logiciels comme Matlab, une présentation complète de la base d'apprentissage s'appelle *epoch*).

Remarques

- le choix du paramètre μ est primordial et doit se faire en faisant un compromis :

  * pour de faibles valeurs de μ, la convergence est lente mais l'erreur résiduelle est faible,

  * pour de fortes valeurs de μ, la convergence est rapide mais l'erreur résiduelle peut être importante, ce qui peut se traduire par des oscillations autour du minimum.

  Pour résoudre ce problème, on peut utiliser la règle de mise à jour suivante qui fait intervenir un terme appelé *momentum constant* noté $\beta \in [0, 1[$:

$$\begin{aligned} \Delta\widehat{w}_{ij}^{(l)}(n) &= \widehat{w}_{ij}^{(l)}(n+1) - \widehat{w}_{ij}^{(l)}(n+1) \\ &= \beta\Delta\widehat{w}_{ij}^{(l)}(n-1) + \mu\delta_j^{(l)}(n)\, y_i^{(l-1)}(n) \end{aligned}$$

  Pour $\beta = 0$, on retrouve la règle classique de la rétropropagation du gradient. Pour $\beta > 0$, on obtient :

$$\Delta\widehat{w}_{ij}^{(l)}(n) = \mu \sum_{k=0}^{n} \beta^{n-k} \delta_j^{(l)}(n)\, y_i^{(l-1)}(n),$$

  ce qui a pour effet 1) d'augmenter la rapidité de la descente lorsque deux erreurs consécutives sont de même signe et 2) de ralentir la descente lorsque deux erreurs consécutives sont de signes différents. L'introduction de β a en général un effet stabilisateur pour l'algorithme,

- la convergence de l'algorithme de la rétropropagation du gradient peut être problématique ; en effet, rien n'empêche l'algorithme d'être piégé dans des minima locaux. Les règles d'arrêt de l'algorithme dépendent de l'application. Si, par exemple, on cherche à identifier un système non-linéaire, on pourra s'arrêter lorsque l'erreur entre la sortie du réseau et du système à identifier est inférieure à un seuil prédéfini. Si on cherche à faire de l'égalisation, on pourra s'arrêter lorsque l'erreur entre la sortie du réseau et une séquence d'apprentissage est suffisamment faible.

3.5 Justifications Théoriques

Le perceptron peut être vu comme une structure permettant de modéliser un système non-linéaire. Plus précisément, si p est le nombre d'entrées et q le nombre de sorties, le perceptron peut être vu comme une application de $\mathbb{R}^p$ dans $\mathbb{R}^q$ qui est indéfiniment dérivable (lorsqu'on utilise des non-linéarités indéfiniment dérivables comme la fonction logistique). La question fondamentale suivante se pose alors :

Quel est le nombre minimal de couches nécessaire pour l'approximation d'une fonction de $\mathbb{R}^p$ dans $\mathbb{R}^q$?

La réponse à cette question est donné par le théorème d'approximation universelle suivant démontré par Cybenko en 1988 [5][13] :

Théorème : *Soit f une fonction continue croissante bornée vérifiant $\lim_{x\to-\infty} f(x) = 0$ et $\lim_{x\to\infty} f(x) = 1$ (une fonction satisfaisant ces deux conditions est appelée fonction saturante) et I_p l'hypercube $[0,1]^p$. L'espace des fonctions continues sur I_p est noté $C(I_p)$. Alors, pour toute fonction $g \in C(I_p)$, $\forall \varepsilon > 0$, il existe un entier M et un ensemble de constantes réelles $\alpha_i, \theta_i, w_{ij}$ avec $i = 1, ..., M$ et $j = 1, ..., p$ tel que la fonction*

$$F(x_1, ..., x_p) = \sum_{i=1}^{M} \alpha_i f\left(\sum_{j=1}^{p} w_{ij} x_j\right)$$

soit une approximation de g c'est-à-dire :

$$|F(x_1, ..., x_p) - g(x_1, ..., x_p)| < \varepsilon$$

pour tout $x = (x_1, ..., x_p)^t \in I_p$.

Ce théorème approximation universelle donne une justification mathématique de l'approximation d'une fonction arbitraire continue par un perceptron avec une unique couche cachée. Mais ce théorème ne dit pas qu'un perceptron à une couche est optimal au sens de rapidité de convergence ou de facilité d'implantation. En pratique, il est souvent préférable d'utiliser un perceptron avec deux ou trois couches constituées d'un faible nombre de noeuds.

4 Apprentissage non supervisé : cartes de Kohonen

Un important principe d'organisation du cerveau est que le placement des neurones est ordonné et permet de reconnaître certaines caractéristiques physiques des stimulants extèrieurs. Par exemple, à chaque niveau de la région auditive du cerveau, les cellules et fibres nerveuses sont rangées anatomiquement en fonction de la fréquence des divers sons perçus par un individu. Bien

que la plupart de l'organisation des cellules et des fibres soit d'origine génétique, il est probable que certaines organisations aux niveaux les plus élevés soient crées durant l'apprentissage par des algorithmes qui encouragent une auto-organisation. Kohonen [18] présente un tel algorithme qui produit des cartes s'organisant elles-mêmes comme le font les neurones du cerveau.

Algorithme de Kohonen

1) on définit une carte composée de K neurones possédant chacun un certain nombre de voisins. Ces neurones sont numérotés et sont connectés à leur(s) voisin(s) tout au long de l'algorithme. On désignera par N_i le voisinage du $i^{ème}$ neurone. Une telle carte est représentée ci-dessous :

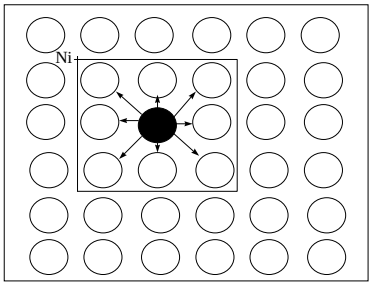


Fig. 5.1 Exemple de Voisinage pour une carte de Kohonen

2) A chaque neurone i est associé un vecteur poids $w_i = (w_{i1}, ..., w_{ip})^t$de même dimension que le vecteur x à classifier. Les composantes w_{ij} sont initialisées aléatoirement à de faibles valeurs ($w_i(0)$ proche de l'origine).

3) on présente un vecteur x de la base d'apprentissage. On compare ce vecteur à tous les vecteurs poids w_i à l'aide d'une distance comme la distance euclidienne. On détermine le plus proche voisin de x noté w_c vérifiant :

$$d(x, w_c) = \inf_i d(x, w_i) \tag{4.41}$$

4) le vecteur w_c et ses voisins sont mis à jour à l'aide de l'algorithme du gradient :

$$\begin{array}{ll} w_i(n+1) = w_i(n) - \frac{\mu}{2}\frac{\partial e^2(n)}{\partial w_i(n)} & i \in N_c \\ w_i(n+1) = w_i(n) & i \notin N_c \end{array} \tag{4.42}$$

avec $e(n) = x(n) - w_i(n)$. On obtient donc :

$$\begin{array}{ll} w_i(n+1) = w_i(n) + \mu(x(n) - w_i(n)) & i \in N_c \\ w_i(n+1) = w_i(n) & i \notin N_c \end{array} \tag{4.43}$$

5) retour à 1)

Lorsque tous les vecteurs de la base d'apprentissage (non expertisée) ont été présenté, la phase de classification proprement dite peut commencer. Deux approches sont possibles :

* on regroupe certains vecteurs w_i de façon à constituer le nombre de classes désiré. On présente un vecteur z qui est comparé à tous les vecteurs w_i. On affecte z grâce à la règle de la distance au barycentre ou à la règle des kPPV.

* on peut choisir une carte comportant un nombre de neurones égal au nombre de classes désiré. On classe alors le vecteur z dans la classe minimisant $d(z, w_i)$.

On peut faire les remarques suivantes :

1) la densité ponctuelle locale asymptotique des vecteurs w_i (c'est-à-dire le nombre de vecteurs w_i par unité de volume de l'espace constitué par les vecteurs d'entrée) est une fonction croissante de la densité de probabilité des vecteurs d'entrée $f(x)$.

2) le facteur $\mu \in [0, 1]$ peut évoluer au cours du temps : on le note alors $\mu(n)$. Plus $\mu(n)$ est proche de 1, plus la carte évolue rapidement. Plus $\mu(n)$ est proche de 0, plus la carte est précise. En général on prend pour $\mu(n)$ une fonction décroissante proche de 1 pour les premières mises à jour (par exemple $\mu(n) = 0.9(1 - 10^{-3}n)$). Ensuite $\mu(n)$ garde une valeur constante (par exemple $\mu(n) = 0.01$). La phase de rangement des cellules se fait durant les premières mises à jour. Les autres mises à jour permettent d'ajuster la carte.

3) Le voisinage N_i peut évoluer au cours du temps. Lorsque N_i est "grand", la carte évolue rapidement. L'évolution est beaucoup plus lente lorsque N_i est "petit".

4) la précision de la carte dépend du nombre de mises à jour qui doit être relativement important. Une règle est de prendre au moins un nombre d'étapes supérieur à $500N_n$, N_n étant le nombre de neurones de la carte. Si le nombre d'éléments de la base d'apprentissage est trop faible, on peut les présenter plusieurs fois.

Apprentissage Supervisé

Lorsqu'on dispose d'une base d'apprentissage expertisée, on peut modifier l'algorithme de Kohonen qui porte alors le nom de LVQ (Learning Vector Quantization). On suppose que les vecteurs de la base d'apprentissage sont répartis en K classes $\omega_1, ..., \omega_K$. On définit la classe d'un vecteur w_i par la classe majoritaire des vecteurs x de la base d'apprentissage ayant pour plus proche voisin w_i. Soit x un vecteur de la base d'apprentissage, C_x la classe

de x et C_{w_c} la classe du plus proche voisin de x noté w_c. L'algorithme LVQ consiste à remplacer l'étape de mise à jour (4.43) par :

$$\begin{array}{ll} w_i(n+1) = w_i(n) + \mu(x(n) - w_i(n)) & i \in N_c, C_x = C_{w_c} \\ w_i(n+1) = w_i(n) - \mu(x(n) - w_i(n)) & i \in N_c, C_x \neq C_{w_c} \\ w_i(n+1) = w_i(n) & i \notin N_c \end{array} \tag{4.44}$$

L'apprentissage supervisé récompense la bonne classification et punit la mauvaise. D'autres algorithmes de quantification vectorielle plus élaborés sont disponibles dans la littérature [18].

5 Comparaison des différentes techniques de classification

Les techniques de classification les plus couramment employées sont

* la classification par la règle de la distance aux barycentres (qui découle de la décision Bayésienne en supposant que les densités conditionnelles sont normales et que les classes sont équiprobables)

* la classification par la règle des kppv (qui découle de la décision Bayésienne lorsqu'on estime les densités de chaque classe à l'aide d'une méthode non-paramétrique)

* la classification par perceptron multi-couches (qui ne fait aucune hypothèse sur les densités conditionnelles de chaque classe)

La règle de de décision Bayésienne est optimale (au sens de la probabilité d'erreur) lorsque les densités conditionelles de chaque classe sont connues. Lorsque ce n'est pas le cas, il n'y a pas de méthode optimale. On peut néanmoins faire les remarques suivantes :

* la méthode des kppv demande une mémoire importante puisqu'il faut stocker tous les paramètres correspondant à chaque élément de la base d'apprentissage. Le coût calculatoire que nécessite cette méthode est assez élevé car il faut calculer la distance à tous les éléments de la base d'apprentissage. Cependant elle donne souvent de meilleurs résultats que la règle de la distance aux barycentres particulièrement dans le cas de classes non-convexes. Cette dernière remarque est illustrée sur la figure 4.11. Clairement, lorsqu'on utilise la règle de la distance au barycentre, les éléments de la première classe représentés par des étoiles sont mal classés dans le cas des classes non-convexes et bien classés dans le cas de classes convexes.

* les classifieurs basés sur les réseaux neuronaux font des hypothèses très faibles sur les lois des différentes classes. Les résultats obtenus avec cette méthode sont asymptotiquement (asymptotiquement signifiant lorsque

le nombre d'éléments de la base d'apprentissage tend vers l'infini) équivalents aux résultats obtenus avec la règle de décision Bayésienne. Cependant, elles engendrent un coût calculatoire très important.

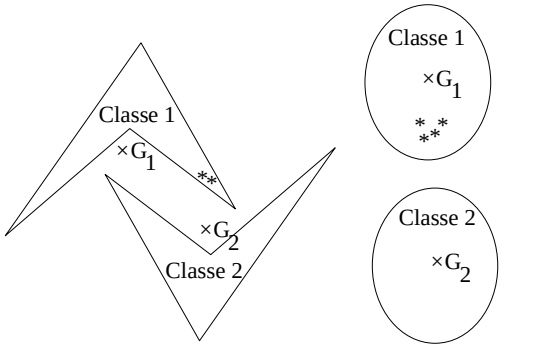


Fig. 4.11 Modèles de classes convexes et non-convexes

Chapitre 5

Classification Structurelle

1 Introduction

Une approche classique en reconnaissance des formes est fondée sur l'étude statistique des mesures qui ont été effectuées sur des objets à reconnaître. L'étude de la statistique de ces mesures permet, à l'aide de règles de décision souvent élémentaires (règle de la distance au barycentre, règle des k plus proches voisins,...), d'affecter à chaque objet une classe. Ces méthodes de classification, qualifiées souvent de **méthodes statistiques**, peuvent s'avérer inefficaces dans certaines applications. En effet, ces méthodes ont le défaut de considérer indifféremment les différentes composantes des vecteurs de l'espace de représentation. Par exemple, imaginons le problème (illustré en cours) de la classification de différentes espèces de poisson. Une approche statistique consiste à effectuer des mesures sur des images de poissons. Toutes les mesures liées à la taille de ces poissons (périmètre, surface, ...) sont mal adaptées au problème car elles ne sont pas caractéristiques d'une espèce de poisson (deux poissons peuvent appartenir à la même espèce sans pour autant avoir la même taille !!). C'est pour cette raison que certaines méthodes de classification, appelées **méthodes structurelles ou méthodes syntaxiques**, sont basées sur l'analyse de paramètres liés à la nature même des formes étudiées. Par exemple, à l'aide d'un algorithme de détection de contours, on peut extraire d'une image, une succession de segments de droite caractéristiques d'une image. Un rectangle pourra alors être représenté de la façon suivante : "un segment de droite de longueur L suivi d'un segment de droite de longueur l qui lui est orthogonal, suivi d'un segment de droite de de longueur proche de L paralléle au premier segment, suivi d'un segment de droite de longueur proche de l qui se termine au point de départ du premier segment. Cette description est naturellement identique pour tous les rectangles quelle que soit leur orientation. Le but de cette par-

tie est de présenter quelles techniques structurelles de classification qui sont essentiellement développées dans [20].

2 Structures de chaines

2.1 Généralités

Une description très simple d'un objet consiste à représenter cet objet comme une suite ordonnée de symboles appelés **lettres** et notées $a, b, c....$ L'ensemble de ces lettres constitue un **alphabet** noté A. Une suite ordonnée d'éléments de A est une **phrase** ou une **chaine**. La **longueur d'une chaine** notée $|x|$ est le nombre de lettres qui la constitue. La chaine vide notée $\emptyset$ est par convention la suite de lettres de longueur nulle. La chaine constituée de la répétition de n lettres identiques a est notée $a^n = a.a....a$ (n fois). L'ensemble des chaines sur A (resp. chaines non vides) est noté A^+ (resp. $A^* = A^+ \backslash \emptyset$).

Exemple

ALPHABET

$$A = \{a, b, c, d\} \tag{5.1}$$

CHAINE

$$x = abbcad \qquad \text{avec} \qquad |x| = 6 \tag{5.2}$$

On peut définir sur A^* une opération interne, en général non commutative, dite de concaténation telle que, $\forall\,(x, y) \in A^*$ avec $x = x_1...x_n$ et $y = y_1...y_m$

$$z = xy = x_1...x_n y_1...y_m \tag{5.3}$$

Cette opération possède $\emptyset$ comme élément neutre.

2.2 Exemples

1) **Codage de Freeman**

Un exemple fondamental en traitement d'images est donné par le codage de Freeman. Chaque ligne d'une image peut être approchée de façon discrète par une suite de vecteurs de taille élémentaire et de direction choisie dans un ensemble fini.

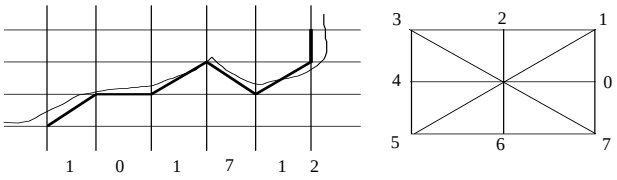


Fig. 6.1 Codage de Freeman pour une ligne

L'alphabet correspondant au codage de Freeman est $A = \{0, ..., 7\}$.

2) **Généralisation au codage par changement de directions**

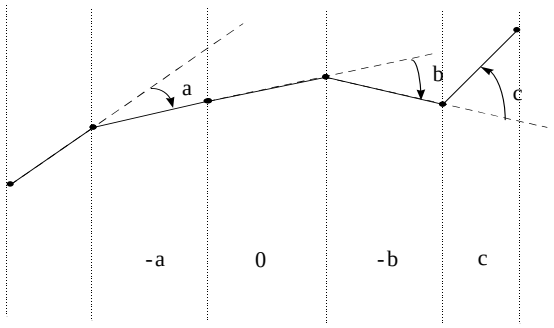


Fig. 6.2 Codage par Changement de Directions

L'alphabet correspondant au codage par changement de directions est

$$A = \{-180, -179, ..., -1, 0, 1, ..., 179, 180\} \tag{5.4}$$

3) **Codage des parties concaves, convexes et plates**

A un niveau plus élevé de traitement, la frontière d'un objet dans une image peut être représentée par une successions de parties concaves (notées *cc*), convexes (notées *cv*) et à peu près plates (notées *pl*).

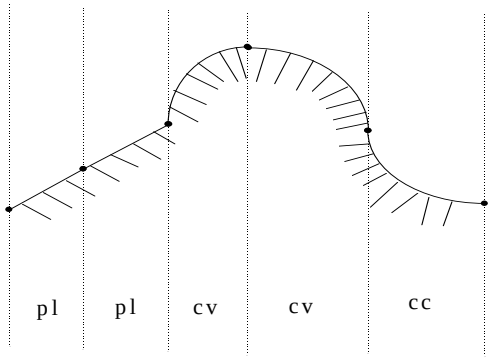


Fig. 6.3 Codage des parties concaves, convexes et plates

L'alphabet correspondant au codage des parties concaves, convexes et plates est $A = \{pl, cv, cc\}$.

4) **Codage des ECG**

Codage des pics positifs, négatifs et des lignes.

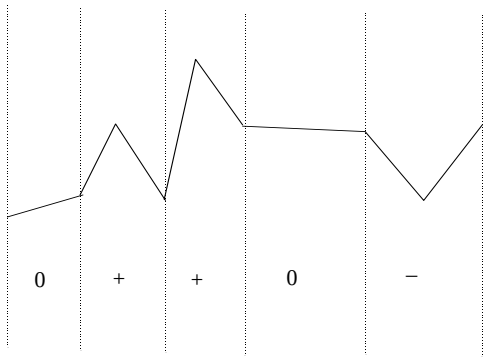


Fig. 6.4 Codage des pics positifs, des pics négatifs et des lignes

2.3 Comparaison des Chaînes

Il existe des algorithmes qui permettent de tester si une chaine est incluse dans une autre (algorithmes d'inclusion de chaines [20]). On se limitera ici à présenter un algorithme qui calcule la distance entre deux chaines par l'algorithme de Wagner et Fisher.

Distance de Levenstein

On définit trois opérations baptisées "substitution", "insertion" et "destruction" :

- **Substitution**

  Elle consiste à changer une lettre a en une lettre b, est notée $a \rightarrow b$ et coûte $S(a, b)$.

- **Insertion**

  Elle consiste à insérer une lettre a, est notée $\emptyset \rightarrow a$ et coûte $S(\emptyset, a)$.

- **Destruction**

  Elle consiste à détruire une lettre a, est notée $a \rightarrow \emptyset$ et coûte $S(a, \emptyset)$.

Définition *: La* **distance de Levenstein** *entre deux chaines x et y , notée $d(x, y)$ est le coût de la transformation la moins coûteuse permettant de passer de x à y.*

Pour calculer $d(x,y)$ en évitant un coût calculatoire énorme, on impose $d(a,b) = S(a,b)$, ce qui revient à supposer que $d(a,b)$ est une distance c'est-à-dire vérifie l'inégalité triangulaire $d(a,c) \leq d(a,b) + d(b,c)$.

Trace de coût minimal

Définition *: On définit un type de transformation particulier, appelé* ***Trace****, permettant de passer de x à y par une succession d'opérations décrites par le schéma suivant :*

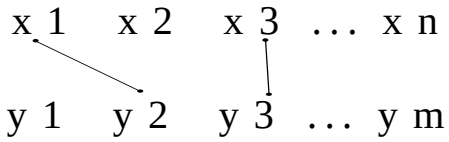


Fig. 6.5 Modèle de Trace

Un trait correspond à une substitution tandis que les autres opérations sont des destructions et des insertions. On impose que dans une trace deux traits ne se coupent pas, ne partent pas (ou n'arrivent pas) à une même lettre. On peut alors démontrer le théorème suivant :

Théorème *: Le coût de la transformation de optimale entre x et y est la coût de la trace optimale entre x et y.*

Autrement dit, pour chercher la transformation optimale, il suffit de chercher parmi les traces permettant de passer de x à y.

Algorithme de Wagner and Fisher

Une trace peut être considérée comme une structure hiérarchique qui nous permet d'avoir un algorithme de calcul récurrent. Si $x = x_1...x_n$ et $y = y_1...y_m$, on note $x(i) = x_1...x_i$ et $d_{ij} = d[x(i), y(j)]$ de sorte que $d_{nm} = d(x,y)$. En utilisant le fait que la distance entre $x(i)$ et $y(j)$ est nécessairement optimale, on a :

$$d_{ij} = Min \left\{ \begin{array}{c} d_{i-1,j-1} + S(x_i, y_j) \\ d_{i,j-1} + S(\emptyset, y_j) \\ d_{i-1,j} + S(x_i, \emptyset) \end{array} \right\} \tag{5.5}$$

Ceci vient du fait de la nature des traces qui nous autorise à travailler de gauche à droite puisque les traces ne se coupent pas. De plus, on pose :

$$\begin{aligned} d_{00} &= 0 \\ d_{i0} &= d_{i-10} + S(x_i, \emptyset) \\ d_{0j} &= d_{0j-1} + S(\emptyset, y_j) \end{aligned} \tag{5.6}$$

On obtient alors l'algorithme de Wagner et Fisher.

3 Structures d'arbres et de graphes

Le principe de coder un objet sous la forme d'une chaine est tres satisfaisant car il permet des traitements efficaces et rapides mais il est limité dans la mesure où il ne donne qu'une représentation monodimensionnelle. Pour résoudre ce problème, il est préfèrable dans certaines applications de coder un objet sous la forme d'un arbre.

3.1 Les arbres comme graphes particuliers

Un **graphe orienté** est un couple d'ensembles (X, U) défini par :

- X est un ensemble de **noeuds** (ou sommets).
- $U \subset X \times X$ est un ensemble de couples ordonnés de sommets appelés **arcs** dont les éléments sont appelés origine et extrémité.

On représente graphiquement un noeud par un chiffre ou une lettre à l'intérieur d'un cercle et un arc par un trait portant une flèche :

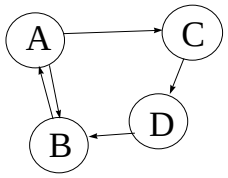


Fig. 6.6 Exemple de graphe

Un graphe est **connexe** si pour tout couple (A, B) de sommets distincts, il existe au moins un chemin d'origine A et d'extrémité B.

Dans le cas général, un graphe est **orienté**. Si U possède toujours l'élément (A, B) lorsqu'il possède l'élément (B, A), le graphe est dit **non orienté**. On adoptera alors la représentation graphique suivante :

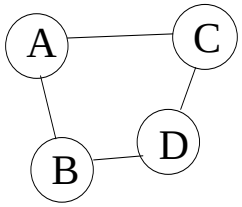


Fig. 6.7 Exemple de graphe non orienté

Dans ces conditions, on appelle **arbre**, un graphe non orienté tel que l'une des propriétés suivantes soit satisfaite :

- il est connexe, mais perd cette propriété dès qu'on lui enlève un arc quelconque
- il est connexe, possède n sommets et $n-1$ arcs
- il existe un chemin et un seul entre tout couple de sommets différents

Exemple d'arbre

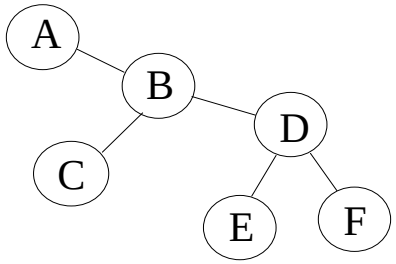


Fig. 6.8 Exemple d'arbre

3.2 Les arbres comme structure hiérarchique

On peut considérer un arbre comme une succession de sous arbres partant d'un même noeud appelé sommet ou racine. Dans ces conditions, on peut adopter la définition suivante :

Définition : *Un **arbre** T est un ensemble fini X de noeuds tels que*

- *il existe dans X un éléments particulier noté $rac(T)$ qui est le sommet ou la racine de l'arbre T.*

- *les autres noeuds, s'il en existe, sont regroupés en ensembles disjoints $X_1, ..., X_n$ qui sont eux-mêmes des arbres notés $T_1, ..., T_n$ dont la racine est reliée à $rac(T)$ par un arc.*

Dans l'exemple précédent, si on prend D comme racine, les sous arbres sont $(A, B, C), (E)$ et (F). On peut alors repésenter cet arbre sous la forme suivante :

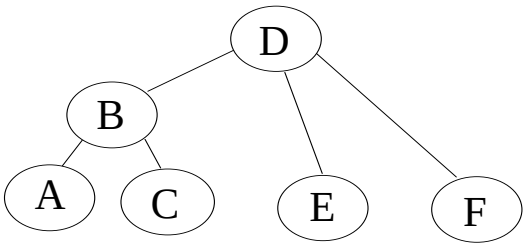


Fig. 6.9 Arbre comme structure hiérarchique

Définition *: Un arbre est dit étiqueté quand il existe une application de l'ensemble de ses noeuds dans un alphabet A. Autrement dit, chacun de ses noeuds possède une étiquette appartenant à A.*

3.3 Distance entre arbres-Algorithme de Selkow

On considère l'ensemble des arbres étiquetés sur un alphabet A défini comme précédemment (voir définition 2). On note $Eti(a)$ l'étiquette de chaque noeud $a \in T$ et $Eti(T)$ l'étiquette de sa racine. On note $T(i)$ l'arbre obtenu en enlevant les sous-arbres $T_{i+1}, ..., T_n$. En particulier, on a $T(n) = T$.

Définition *: On dit que deux arbres T et U sont égaux s'ils ont même étiquette ($Eti(T) = Eti(U)$) et mêmes sous-arbres.*

Sur l'ensemble des arbres étiquetés, on définit trois opérations appelées "Changement d'étiquette", "Insertion d'un sous-arbre" et "Destruction d'un sous-arbre". Soit T un arbre étiqueté de sous arbres $T_1, ..., T_n$ dont les étiquettes appartiennent à l'alphabet $A = \{a_1, ..., a_q\}$ avec $Eti(T) = a_j$.

- Changement d'étiquette

L'opération "Changement d'étiquette" notée $L(a_j, a_i)$ a pour effet de transformer l'arbre T en un arbre T^* qui possède les mêmes sous-arbres que T mais dont l'étiquette est a_i.

- Insertion d'un sous-arbre

  L'opération "Insertion d'un sous-arbre" notée $I_i(A)$ transforme T en T^* qui possède la même étiquette que T mais dont les sous arbres sont $T_1, ..., T_i, A, T_{i+1}, ..., T_n$.

- Destruction d'un sous-arbre

  L'opération "Destruction d'un sous-arbre" notée $D(T_i)$ transforme T en T^* qui possède la même étiquette que T mais dont les sous arbres sont $T_1, ..., T_{i-1}, T_{i+1}, ..., T_n$.

Pour chaque opération, on se fixe un coût positif ou nul. Pour le changement d'étiquette, on se fixe la matrice des coûts $C_L(k, l)$ (qui correspond à $L(a_k, a_l)$). Pour l'insertion et la destruction d'un sous-arbre, on se fixe le coût de l'insertion et de la destruction d'un arbre réduit à un noeud étiqueté par a_k, notés respectivement $C_I(a_k)$ et $C_D(a_k)$. Le coût des opérations $I_i(A)$ et $D(T_i)$ sera alors :

$$C_I(A) = \sum_{k \in A} C_I(a_k) \tag{5.7}$$

$$C_D(T_i) = \sum_{k \in T_i} C_I(a_k) \tag{5.8}$$

Il existe de multiples transformations permettant de passer d'un arbre à un autre. La distance $d(T, U)$ entre deux arbres étiquetés T et U est définie par le coût de la transformation la moins couteuse permettant de passer de T à U. De la même façon que pour l'algorithme de Wagner and Fisher, on a un algorithme récursif qui a été démontré par Selkow :

Théorème : Soient deux arbres T et U (de sous-arbres $T_1, ..., T_n$ et $U_1, ..., U_m$). On a :

$$d[T(0), U(j)] = C_L(Eti(T), Eti(U)) + \sum_{k=1}^{j} C_I(U_k) \tag{5.9}$$

$$d[T(i), U(0)] = C_L(Eti(T), Eti(U)) + \sum_{k=1}^{i} C_D(T_k) \tag{5.10}$$

$$d[T(i), U(j)] = Min \left\{ \begin{array}{c} d[T(i-1), U(j-1)] + d[T_i, U_j] \\ d[T(i), U(j-1)] + C_I[U_j] \\ d[T(i-1), U(j)] + C_D[T_i] \end{array} \right\} \tag{5.11}$$

Chapitre 6

VALIDATION DU CLASSIFIEUR

1 Introduction

Lorsqu'un classifieur a été choisi pour un problème donné, à partir d'un ensemble d'apprentissage (mode supervisé) ou sans (mode non supervisé), il est important de déterminer la performance de ce classifieur en terme de sa probabilité d'erreur. On suppose dans ce chapitre que toutes les erreurs ont la même importance, ce qui correspond à choisir la fonction de coût $c_{ij} = 1 - \delta_{ij}$ dans l'approche Bayésienne. Bien évidemment, le classifieur est d'autant meilleur que sa probabilité d'erreur est faible. Ce chapitre s'intéresse tout d'abord à l'estimation de la probabilité d'erreur à l'aide d'un ensemble de vecteurs de test dont les classes sont connues. Dans une seconde partie, nous nous intéresserons à la répartition de l'ensemble des données expertisées en deux sous-ensembles : l'ensemble d'apprentissage et l'ensemble de test.

2 Estimation de la probabilité d'erreur

L'estimation de la probabilité d'erreur associée à la classe ω_i suppose qu'on dispose de N_i vecteurs x_j, $j = 1, ..., N_i$, appartenant à la classe ω_i et à chacun de ces vecteurs est associée une décision $d(x_i) \in \{a_1, ..., a_k\}$ où K est le nombre de classes. Pour chaque vecteur x_j, on construit la variable aléatoire binaire Y_j définit par :

$$Y_j = \begin{cases} 0 \text{ si } d(x_j) = a_j \\ 1 \text{ sinon} \end{cases}$$

L'**estimateur du maximum de vraisemblance** de la probabilité d'erreur

associée à la classe ω_i notée P_i est défini par

$$\widehat{P}_i = \text{argmax}_{P_i} P\left[Y_1 = y_1, ..., Y_{N_i} = y_{N_i}; P_i\right]$$

En supposant que tous les vecteurs de l'ensemble de test sont indépendants, on en déduit aisément

$$\widehat{P}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} Y_j \triangleq \frac{K_i}{N_i}$$

où $K_i = \sum_{j=1}^{N_i} Y_j$ est une variable aléatoire de loi binomiale $\mathcal{B}(N_i, P_i)$. On en déduit

$$E\left[\widehat{P}_i\right] = P_i \text{ et } Var\left[\widehat{P}_i\right] = \frac{P_i(1-P_i)}{N_i}$$

L'estimateur $\widehat{P}_i$ du paramètre P_i est donc sans biais et convergent (en moyenne quadratique).

On peut préférer à une estimation ponctuelle $\widehat{P}_i$ une **estimation par intervalle de confiance** qui consiste à déterminer un intervalle IC tel que $P\left[P_i \in IC\right] = \alpha$. On dit alors que IC est un intervalle de confiance pour $\widehat{P}_i$ avec un seuil de confiance α (souvent on choisit $\alpha = 0.90$ ou 0.95). Dans le cas où N_i est suffisamment grand pour pouvoir appliquer le théorème de la limite centrale, on peut utiliser l'approximation

$$\widehat{P}_i \simeq \mathcal{N}\left(P_i, \frac{P_i(1-P_i)}{N_i}\right) \Longleftrightarrow \frac{\widehat{P}_i - P_i}{\sqrt{\frac{P_i(1-P_i)}{N_i}}} \simeq \mathcal{N}(0,1)$$

Les tables de la loi normale centrée réduite donnent le paramètre a_α tel que

$$P\left[\left|\frac{\widehat{P}_i - P_i}{\sqrt{\frac{P_i(1-P_i)}{N_i}}}\right| \leq a_\alpha\right] = \alpha$$

Par exemple pour $\alpha = 0.95$, on trouve $a_\alpha = 1.96$. On a alors

$$\left(\widehat{P}_i - P_i\right)^2 - a_\alpha^2 \frac{P_i(1-P_i)}{N_i} \leq 0 \tag{6.1}$$

avec la probabilité α. La résolution de cette équation du second degré fournit un intervalle de confiance de la forme

$$\left[\frac{\widehat{P}_i + a_\alpha^2 - \sqrt{\Delta}}{1 + \frac{a_\alpha^2}{N_i}}, \frac{\widehat{P}_i + a_\alpha^2 + \sqrt{\Delta}}{1 + \frac{a_\alpha^2}{N_i}}\right]$$

où Δ est le discriminant de l'équation du second degré (6.1). Des intervalles de confiance pour la probabilité d'erreur sont représentés dans ([9], p. 75) et permettent par exemple de conclure

$$\begin{aligned} \alpha &= 0.95, N_i = 50, \widehat{P}_i = 0 \Longrightarrow P_i \in [0, 0.08] \text{ ([9], p.75)} \\ \alpha &= 0.95, N_i = 250, \widehat{P}_i = 0.2 \Longrightarrow P_i \in [0.15, 0.25] \text{ [16]} \end{aligned}$$

On peut de la même façon déterminer des estimations ponctuelles et par intervalle de confiance de la probabilité d'erreur d'un classifieur P_e ([32], p. 340). Certains auteurs ont déterminé des inégalités reliant $\widehat{P}_e$ et P_e. A titre d'exemple, nous donnons une telle inégalité indépendante de la loi des données de l'apprentissage et même de la fonction de décision du classifieur ([6], p. 122) :

$$P\left[\left|\widehat{P}_e - P_e\right| > \epsilon\right] \leq 2e^{-2N\epsilon^2} \tag{6.2}$$

où $N = \sum_{i=1}^{K} N_i$ est le nombre total de vecteurs de test. Cette inégalité suppose que les vecteurs de test sont indépendants et il n'existe pas, à notre connaissance, d'inégalité aussi générale lorsque les vecteurs ne sont pas indépendants.

3 Disjonction des ensembles d'apprentissage et de test

Les données expertisées étant en nombre fini, un problème fondamental pour l'évaluation des performances d'un classifieur est de constituer l'ensemble d'apprentissage et l'ensemble de test. Il existe plusieurs méthodes permettant d'effectuer cette répartition :

Méthode de Resubstitution

Cette méthode consiste à utiliser les mêmes données pour l'apprentissage et le test. Elle fournit une estimation optimiste de la probabilité d'erreur du classifieur. Par exemple, si on utilise la règle du 1PPV, et qu'on utilise les mêmes données pour l'apprentissage et le test, le plus proche voisin d'un vecteur x_i est lui même et donc $\widehat{P}_e = 0$. Il existe des inégalités entre $\widehat{P}_e$ et P_e, mais ces inégalités ne sont valables que pour certaines règles de classification, par exemple pour les règles des kPPV.

Méthode "Holdout"

Cette méthode consiste à séparer toutes les données expertisées en deux ensembles disjoints de cardinaux N_a et N_t tels que $N_a + N_t = N$. Il n'existe

malheureusement pas de méthode permettant de faire cette répartition et cette méthode d'avoir beaucoup de données expertisées.

Méthode "Leave one out" (appelée aussi méthode de validation croisée) :

Cette méthode consiste à effectuer la procédure de test à partir d'un seul vecteur de données et de construire le classifieur à l'aide des $N-1$ vecteurs restants (constituant l'ensemble d'apprentissage). On répète cette opération N fois jusqu'à ce que tous les vecteurs aient été une fois vecteur de test. On estime la probabilité d'erreur par la moyenne des erreurs au cours de ces N opérations. Cette méthode fournit une estimation pessimiste de la probabilité d'erreur.

D'autres méthodes plus élaborées comme la méthode de rotation (qui est une variante de la méthode de validation croisée) ou une méthode utilisant la technique du bootstrap sont disponibles dans la littérature. On se reportera à [6] [16] pour une étude complète de ce problème.

CHAPITRE 7

ANNEXES

1 ACP des Individus et ACP des Variables

Dans cet annexe, nous allons sans cesse passer de l'ACP des individus à l'ACP des variables. Dans un but de simplification, on peut adopter les notations suivantes :

ACP des Individus

n individus $X_1, ..., X_n$ à p variables tels que $X_i^t = (X_i(1), ..., X_i(p))$. Pour l'ACP normée, on considère :

$$X_i' = \begin{pmatrix} \frac{X_i(1)-m(1)}{\sigma(1)} \\ . \\ \frac{X_i(p)-m(p)}{\sigma(p)} \end{pmatrix}$$

avec

$$m(j) = \frac{1}{n}\sum_{i=1}^{n} X_i(j) \text{ et } \sigma^2(j) = \frac{1}{n}\sum_{i=1}^{n} [X_i(j) - m(j)]^2$$

ACP des Variables

p variables $V_1, ..., V_p$ telles que $V_j^t = (X_1(j), ..., Xn(j))$. Pour l'ACP normée, on considère :

$$V_j' = \begin{pmatrix} \frac{X_1(j)-m(j)}{\sigma(j)\sqrt{n}} \\ . \\ \frac{X_n(j)-m(j)}{\sigma(j)\sqrt{n}} \end{pmatrix}$$

La norme du vecteur V_j' est :

$$\sum_{i=1}^{n} \left[\frac{X_i(j) - m(j)}{\sigma(j)\sqrt{n}}\right]^2 = \frac{1}{n\sigma^2(j)}\sum_{i=1}^{n} [X_i(j) - m(j)]^2 = 1$$

Tous les vecteurs V_j' sont situés sur une hypersphère de rayon unité.

Interprètation de l'ACP des Variables

Le coefficient de corrélation entre les variables V_j et V_k s'écrit :

$$r_{jk} = \frac{cov\,(V_j, V_k)}{\sigma\,(j)\,\sigma\,(k)} = \frac{1}{\sigma\,(j)\,\sigma\,(k)\,n} \sum_{i=1}^{n} \left[X_i\,(j) - m\,(j)\right]\left[X_i\,(k) - m\,(k)\right]$$

Le coefficient de corrélation entre les variables V_j et V_k est le produit scalaire entre les variables réduites V_j' et V_k'. En tenant compte de l'égalité :

$$\left\|V_k' - V_j'\right\|^2 = \left\|V_k'\right\|^2 + \left\|V_j'\right\|^2 - 2\left\langle V_k', V_j'\right\rangle$$

on obtient :

$$\left\|V_k' - V_j'\right\|^2 = 2\,(1 - r_{jk})$$

Si le coefficient de corrélation entre les variables V_j' et V_k' est égal à 1, les deux points V_j' et V_k' sont confondus dans l'ACP des variables. Si ce coefficient est égal à -1, on a $\left\|V_k' - V_j'\right\| = 2$ et les deux points V_j' et V_k' sont diamétralement opposés sur l'hypersphère unité. Lorsque le coefficient de corrélation est nul (par exemple variables indépendantes), on a $\left\|V_k' - V_j'\right\| = \sqrt{2}$ et les deux points V_j' et V_k' sont sur deux rayons orthogonaux ([35], pp. 116 et 117).

Lien entre les deux types d'ACP

Il existe un lien entre l'ACP normée des variables et l'ACP normée des individus, ce qui permet d'effectuer une seule analyse.

On note $X = [X_1', ..., X_n']$ et $V = \left[V_1', ..., V_p'\right]$ les matrices de tailles respectives $p \times n$ et $n \times p$ obtenues par concaténation des vecteurs lignes et colonnes du tableau de données (on a $V = \frac{1}{\sqrt{n}} X^t$). Les matrices de covariances des ACP des individus et des variables s'écrivent alors :

$$T_I = \frac{1}{n} X X^t \text{ et } T_V = V V^t = \frac{1}{n} X^t X$$

Soit x un vecteur propre de T_I associé à la valeur propre λ. On a alors :

$$T_I x = \lambda x \Leftrightarrow \frac{1}{n} X X^t x = \lambda x$$

Soit $y = \frac{1}{\sqrt{\lambda}} X^t x$. On a :

$$T_V y = \frac{1}{n} X^t X \frac{1}{\sqrt{\lambda}} X^t x = \sqrt{\lambda} X^t x = \lambda y$$

Donc y est un vecteur propre de T_V associé à la valeur propre λ.

Réciproquement, **si y est un vecteur propre de T_V associé à la valeur propre** μ, alors $x = \frac{1}{\sqrt{\lambda}} Xy$ est un vecteur propre de T_I associé à la même valeur propre μ.

Lien entre les projections des différents points

Soit x un vecteur propre de T_I avec la valeur propre λ. Dans l'ACP des individus, la projection de X_i sur l'axe x est $X_i^t x$ qui est la $i^{ème}$ composante de $X^t x$. Le vecteur $X^t x$ représente donc les projections de tous les individus sur l'axe x.

De la même façon, soit y un vecteur de T_V avec la valeur propre λ. Dans l'ACP des variables, la projection de la variable V_j sur l'axe y est $V_j^t y$ qui est la $j^{ème}$ composante de $V^t y = \frac{1}{\sqrt{n}} Xy = \frac{1}{\sqrt{n}} X \frac{1}{\sqrt{\lambda}} X^t x = \sqrt{n\lambda} x$.

En pratique, on effectue les deux étapes suivantes :

1) *ACP des Individus*

En général $p < n$. L'ACP des individus est donc la plus simple à effectuer car elle se fait à partir d'une matrice de dimensions $p \times p$ (pour l'ACP des variables la matrice de covariance est de dimensions $n \times n$). On obtient alors p vecteurs propres v_k et p valeurs propres λ_k ($k \in \{1, ..., p\}$).

On détermine ensuite les projections des individus X_i sur les axes principaux (en calculant les produits scalaires correspondants).

2) *ACP des Variables*

On détermine les p vecteurs propres de T_V associés aux valeurs propres non nulles par :

$$y_k = \frac{1}{\sqrt{\lambda_k}} X^t x_k$$

La projection de la variable V_j sur l'axe y_k est alors $V_j^t y_k$ qui est la $j^{ème}$ composante de $V^t y_k = \frac{1}{\sqrt{n}} X y_k = \frac{1}{\sqrt{n}} X \frac{1}{\sqrt{\lambda_k}} X^t x_k = \sqrt{n\lambda_k} x_k$.

2 Erreur de la règle du 1PPV

*) *Erreur de la règle du PPV*

On note $P_n(x, x'_n)$ la probabilité d'erreur sachant que x et x'_n sont connus. La probabilité d'erreur connaissant x est alors définie par :

$$P_n(x) = \int P_n(x, u) \, f_{x'_n}(u|\, x) \, du \tag{7.1}$$

Soient ω et ω'_n les classes de x et de son plus proche voisin x'_n. On commet une erreur si $\omega \neq \omega'_n$ donc :

$$P_n(x, x'_n) = 1 - \sum_{i=1}^{K} P(\omega = \omega_i, \omega'_n = \omega_i |\, x, x'_n) \tag{7.2}$$

Les vecteurs x et x'_n sont différents et sont supposés indépendants. Donc :

$$P\left(\omega=\omega_i,\omega'_n=\omega_i|\,x,x'_n\right)=P\left(\omega=\omega_i|\,x\right)P\left(\omega'_n=\omega_i|\,x'_n\right) \tag{7.3}$$

On en déduit :

$$P_n\left(x\right)=\int\left[1-\sum_{i=1}^{K}P\left(\omega=\omega_i|\,x\right)P\left(\omega'_n=\omega_i|\,u\right)\right]f_{x'_n}\left(u|\,x\right)du \tag{7.4}$$

$f_{x'_n}\left(u|\,x\right)$ converge vers $\delta\left(u-x\right)$ lorsque n tend vers l'infini, donc :

$$\lim_{n\to\infty}P_n\left(x\right)=1-\sum_{i=1}^{K}P\left(\omega=\omega_i|\,x\right)^2 \tag{7.5}$$

La probabilité de faire une erreur est définie par :

$$P_n=\int P_n\left(x\right)f\left(x\right)dx \tag{7.6}$$

Donc, en admettant qu'on peut intervertir l'intégrale et la limite :

$$P_1=\lim_{n\to\infty}P_n=\int\left[1-\sum_{i=1}^{K}P^2\left(\omega=\omega_i|\,x\right)\right]f\left(x\right)dx \tag{7.7}$$

Cette égalité peut aussi s'écrire :

$$P_1=E\left[1-\sum_{i=1}^{K}P^2\left(\omega=\omega_i|\,X\right)\right] \tag{7.8}$$

*) *Bornes de l'erreur [3]*

L'expression (3.62) permet de borner l'erreur de la règle du plus proche voisin. Si P^* désigne la probabilité d'erreur minimale obtenue à l'aide de la règle de décision Bayésienne (dans le cas d'une fonction de coût $c_{ij}=1-\delta_{ij}$), nous allons montrer :

$$P^*\leq P_1\leq P^*\left(2-\frac{K}{K-1}P^*\right) \tag{7.9}$$

Dans le cas d'une fonction de coût $c_{ij}=1-\delta_{ij}$, la règle de Bayes est définie par :

$$d\left(x\right)=\omega_j\Leftrightarrow P\left(\omega_j\,|x\right)\geq P\left(\omega_k\,|x\right),\qquad\forall k\in\{1,...,K\} \tag{7.10}$$

L'erreur correspondant à cette règle est donnée par (3.62). Le terme $\sum_{i=1}^{K} P^2(\omega = \omega_i | x)$ peut se décomposer de la façon suivante :

$$\sum_{i=1}^{K} P^2(\omega = \omega_i | x) = \sum_{i \neq j} P^2(\omega = \omega_i | x) + P^2(\omega = \omega_j | x) \tag{7.11}$$

En utilisant l'inégalité de Cauchy Schwartz, on montre :

$$(K-1) \sum_{i \neq j} P^2(\omega = \omega_i | x) \geq \left(\sum_{i \neq j} P(\omega = \omega_i | x) \right)^2$$

On pose $A_j(x) = \sum_{i \neq j} P(\omega = \omega_i | x) = 1 - P(\omega = \omega_j | x)$, ce qui permet d'obtenir :

$$\begin{aligned} 1 - \sum_{i=1}^{K} P^2(\omega = \omega_i | x) &\leq 1 - P^2(\omega = \omega_j | x) - \frac{1}{K-1} \left(\sum_{i \neq j} P(\omega = \omega_i | x) \right)^2 \\ &\leq 1 - (1 - A_j(x))^2 - \frac{A_j(x)^2}{K-1} \\ &\leq 2A_j(x) - \frac{K}{K-1} A_j(x)^2 \end{aligned} \tag{7.12}$$

Donc :

$$P_1 = E\left[1 - \sum_{i=1}^{K} P^2(\omega = \omega_i | X) \right] \leq 2E[A_j(X)] - \frac{K}{K-1} E\left[A_j(X)^2 \right] \tag{7.13}$$

On a $E\left[A_j(X)^2\right] = VarA_j(X) + E[A_j(X)]^2 \geq E[A_j(X)]^2$ soit :

$$P_e \leq P^* \left(2 - \frac{K}{K-1} P^* \right) \tag{7.14}$$

La règle de décision Bayésienne étant optimale au sens de la probabilité d'erreur, on a $P_e \geq P^*$ ce qui achève la démonstration.

Chapitre 8

EXERCICES

1 Exercices Divers

1.1 Exercice 1

Extrait de [10]

En communications numériques, on veut transmettre un symbole x binaire défini par :

$$\text{Classe } 1 : x = 0$$
$$\text{Classe } 2 : x = 1$$

Le symbole émis x passe par un canal de transmission, où il est perturbé par un bruit n supposé blanc Gaussien centré de variance σ^2. Le signal reçu est alors $z = x + n$. Le problème est de retrouver le symbole émis à partir du signal reçu. On suppose que les deux valeurs 0 et 1 ont la même probabilité d'apparition, ce qui est le cas en pratique.

1. Enoncez la règle de décision Bayésienne.
2. Calculer la probabilité d'erreur correspondante

1.2 Exercice 2

Deux classes équiprobables sont définies par des lois normales de moyennes et matrices de covariance :

$$M_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, M_2 = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \Sigma_1 = \Sigma_2 = \begin{pmatrix} 1 & 1/2 \\ 1/2 & 1 \end{pmatrix}$$

1. Enoncer la règle de classification qui semble appropriée à ce problème. Déterminer la frontière des zones de décision correspondant à cette règle.

2. On se définit une fonction de coût de la façon suivante :

$$c_{11} = c_{22} = 0 \text{ et } c_{12} = 2c_{21}$$

Que devient la frontière déterminée à la question précédente ? Commentaires

1.3 Exercice 3

Extrait de [10]

On veut reconnaître des signaux Morse, c'est-à-dire des impulsions de durée t variable qui sont, soit des points (classe 1), soit des traits (classe 2). On admet que les densités de probabilité conditionnelles associées aux deux classes sont :

$$f(t \mid \omega_i) = \frac{\frac{1}{\pi b}}{1 + \left(\frac{t-a_i}{b}\right)^2}$$

i étant l'indice de la classe, t étant la durée de l'impulsion mesurée par le chronomètre numérique et a_i la valeur probable de la classe i.

1. Déterminer la règle de décision Bayésienne permettant de reconnaître les points et les traits.
2. On suppose $a_1 = 3, a_2 = 5$ et $b = 5$. Quel est le comportement de $P(\omega_1 \mid t)$ quand $t \to^{+}_{-} \infty$?
3. Calculer la probabilité d'erreur de la règle de Bayes.

2 Exercice 4

On considère dans le plan muni de la distance euclidienne, les cinq points suivants :

$$x_1 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}, x_2 = \begin{pmatrix} 2 \\ -2 \end{pmatrix}, x_3 = \begin{pmatrix} -2 \\ -2 \end{pmatrix}, x_4 = \begin{pmatrix} -2 \\ 2 \end{pmatrix}, x_5 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Les trois premiers points sont éléments de la classe ω_1 et les deux derniers de la classe ω_2.

1. Tracer les frontières de décision obtenues en utilisant la règle du plus proche voisin.
2. Préciser les zones de décision obtenues avec la règle de la distance au barycentre.
3. Conclusions

3 Exercice 5

On dispose d'une base de données expertisées qui servira à l'apprentissage :
Classe 1 :

$$x_1 = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}, x_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, x_3 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, x_4 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, x_5 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

Classe 2 :

$$x_6 = \begin{pmatrix} 3 \\ 3 \end{pmatrix}, x_7 = \begin{pmatrix} 4 \\ 3 \end{pmatrix}, x_8 = \begin{pmatrix} 3.5 \\ 3.5 \end{pmatrix}, x_9 = \begin{pmatrix} 4 \\ 4 \end{pmatrix}, x_{10} = \begin{pmatrix} 4 \\ 4.5 \end{pmatrix}$$

On désire utiliser l'algorithme des kPPV avec $k = 3$ pour les points suivants :

$$\begin{aligned} A &= \begin{pmatrix} 0.5 \\ 1 \end{pmatrix}, B = \begin{pmatrix} 3.5 \\ 2.5 \end{pmatrix}, C = \begin{pmatrix} 2 \\ 3 \end{pmatrix}, D = \begin{pmatrix} 4.5 \\ 4 \end{pmatrix} \\ E &= \begin{pmatrix} 1.5 \\ 0.5 \end{pmatrix}, F = \begin{pmatrix} 3 \\ 3 \end{pmatrix}, G = \begin{pmatrix} 1.5 \\ 1.5 \end{pmatrix}, H = \begin{pmatrix} 3 \\ 1 \end{pmatrix} \end{aligned}$$

1. Donner les résultats intermédiaires permettant la classification de ces points puis déterminer leur classe. Peut-il y avoir ambiguité ? Qu'auraient été les conclusions avec la règle du 1PPV ?

2. Déterminer le meilleur axe discriminant correspondant à l'analyse factorielle discriminante et les projections des points A à H sur cet axe. Donner une représentation graphique et commenter.

4 Exercice 6

Tracer sous forme indicée une représentation hiérarchique ascendante, à l'aide de la méthode des distances, des éléments suivants :

$$\begin{aligned} x_1 &= 112.5, x_2 = 124, x_3 = 120, x_4 = 103, x_5 = 121 \\ x_6 &= 115, x_7 = 123, x_8 = 116, x_9 = 104, x_{10} = 106 \end{aligned}$$

Donner quelques partitions possibles

5 Exercice 7

On donne les 10 éléments suivants qui sont répartis dans deux classes

Classe 1

$$x_1 = \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}, x_2 = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix}, x_3 = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix}, x4 = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}, x_5 = \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}$$

Classe 2

$$x_6 = \begin{pmatrix} 4 \\ 4 \\ 4 \end{pmatrix}, x_7 = \begin{pmatrix} 4 \\ 4 \\ 3 \end{pmatrix}, x_8 = \begin{pmatrix} 4 \\ 4 \\ 4 \end{pmatrix}, x_9 = \begin{pmatrix} 5 \\ 3 \\ 3 \end{pmatrix}, x_{10} = \begin{pmatrix} 4 \\ 3 \\ 2 \end{pmatrix}$$

On désire réaliser l'analyse factorielle discriminante de ces éléments

1. Calculer la matrice interclasse.
2. Déterminer les axes discriminants et les projections des divers éléments. Les deux classes sont-elles bien séparées ? On donne :

$$T^{-1} = \begin{pmatrix} 1.41 & -2.231 & 0.490 \\ & 11.2129 & -6.25 \\ & & 4.943 \end{pmatrix}$$

On pourra utiliser, après quelques explications, le fait que la matrice $T^{-1}B$ admet une seule valeur propre non nulle.

6 Examen du 18 Mars 1993

Documents autorisés, les deux exercices sont indépendants.

Exercice 1

On dispose d'une base de données expertisée constituée des points suivants :

Classe 1 :

$$x_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, x_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, x_3 = \begin{pmatrix} -1 \\ -1 \end{pmatrix}, x_4 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, x_5 = \begin{pmatrix} 0 \\ -1 \end{pmatrix}, x_6 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Classe 2 :

$$x_7 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, x_8 = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, x_9 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, x_{10} = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$$

1. Déterminer le meilleur axe discriminant au sens de l'analyse factorielle discriminante. Représenter les projections des différents points sur cet axe discriminant. Que peut-on en conclure ?

2. Déterminer les axes correspondant à l'analyse en composantes principales. Qu'en déduisez-vous ?

3. Afin de déterminer le paramètre le plus pertinent et donc de confirmer les résultats de la question précédente, on décide d'utiliser le critère de Fisher. Les résultats obtenus sont-ils en accord avec la question précédente ? On donne :

$$S_w = S_1 + S_2 = \frac{1}{9}\begin{pmatrix} 39 & 26 \\ 26 & 48 \end{pmatrix}$$

Exercice 2

Lorsqu'il y a fort recouvrement des classes, on préfère ajouter à la règle de décision Bayésienne classique une option de rejet, ce qui se traduit par l'introduction d'une fonction de coût $(0, 1, \mu)$ définie par :

$$\begin{aligned} c_{ij} &= 1 - \delta_{ij} \qquad i, j = 1, ..., K \\ c_{0j} &= \mu \qquad j = 1, ..., K \end{aligned}$$

La décision $d(x) = \omega_0$ correspond au rejet, c'est-à-dire au cas où on ne classe pas le point x.

1. En vous inspirant de ce qui a été fait en cours, montrez que la règle de décision Bayésienne avec rejet est définie par :

$$\begin{aligned} d(x) &= \omega_0 \Longleftrightarrow \mu \leq 1 - \max_{i=1,...,K} P(\omega_i \mid x) \\ d(x) &= \omega_i \Longleftrightarrow \begin{cases} \mu > 1 - \max_{i=1,...,K} P(\omega_i \mid x) \\ P(\omega_i \mid x) = \mu \leq 1 - \max_{j=1,...,K} P(\omega_j \mid x) \end{cases} \end{aligned}$$

2. En utilisant la relation $\sum_{i=1}^{K} P(\omega_i \mid x) = 1$, donner un minorant de $\max_{j=1,...,K} P(\omega_j \mid x)$ puis un majorant de μ pour que la procédure de rejet soit réalisable. On supposera que cette condition est réalisée.

3. Dans le cas deux classes avec $\mu \leq \frac{1}{2}$, la règle s'écrit :

$$\begin{aligned} d(x) &= \omega_1 \Longleftrightarrow P(\omega_1 \mid x) \geq 1 - \mu \\ d(x) &= \omega_2 \Longleftrightarrow P(\omega_2 \mid x) \geq 1 - \mu \\ d(x) &= \omega_1 \Longleftrightarrow \text{sinon} \end{aligned}$$

On suppose que les densités des deux classes sont Gaussiennes :

$$x \mid \omega_1 \sim N(m_1, \Sigma) \text{ et } x \mid \omega_2 \sim N(m_2, \Sigma)$$

Montrer que la règle consiste alors à comparer une fonction linéaire $D_L(x)$ à deux seuils S_1 et S_2 que l'on précisera?

4. Sous l'hypothèse Gaussienne, dans le cas d'une variable unidimensionnelle x, déterminer la loi de $D_L(x)$. Tracer les zones de rejet et d'acceptation.

5. Déterminer les probabilités d'erreur, de décision correcte et de rejet notées respectivement $E(\mu) = P(\widehat{\omega} \neq \omega, \widehat{\omega} \neq \omega_0), C(\mu) = P(\widehat{\omega} = \omega)$ et $R(\mu) = P(\widehat{\omega} = \omega_0)$, toujours dans le cas unidimensionnel. Comment évoluent ces probabilités lorsque μ parcoure l'intervalle $\left[0, \frac{1}{2}\right]$? Que se passe-t-il pour $\mu = 0$ et $\mu = \frac{1}{2}$?

7 Examen du 22 Mars 1994

Documents autorisés, les deux exercices sont indépendants.

Exercice 1

1. Une analyse détaillée de la loi de probabilité de trois classes ω_1, ω_2 et ω_3 permet de supposer qu'elles sont définies par la densité :

$$\begin{array}{ll} f_\theta(x) = \frac{1}{\theta} x^{\frac{1}{\theta}-1} & x \in [0,1] \\ f_\theta(x) = 0 & \text{sinon} \end{array}$$

avec $\theta = 1$ pour la première classe, $\theta = 2$ pour la seconde classe et $\theta = \frac{1}{2}$ pour la troisième classe. Donner les régions de décision découlant de la règle de classification Bayésienne.

2. On décide de regrouper les deux premières classes et on suppose que la densité correspondant à cette réunion est de la forme précédente, soit

$$\begin{array}{ll} f_\theta(x) = \frac{1}{\theta} x^{\frac{1}{\theta}-1} & x \in [0,1] \\ f_\theta(x) = 0 & \text{sinon} \end{array}$$

avec un paramètre θ inconnu. On garde la même densité pour l'autre classe soit

$$\begin{array}{ll} f_{1/2}(x) = 2x & x \in [0,1] \\ f_{1/2}(x) = 0 & \text{sinon} \end{array}$$

Proposer un estimateur du paramètre θ en fonction des données de la réunion $\omega_1 \cup \omega_2$ notées $x_1, ..., x_{n_1}$ et donner la règle de classification correspondante. Application numérique : $n_1 = 30$ et $\prod x_i = 45.10^{-7}$.

3. On désire travailler sur les observations $y_i = -Lnx_i$. La densité de la seconde classe est alors définie par :

$$\begin{array}{ll} g_2(y) = 2e^{-2y} & y > 0 \\ g_2(y) = 0 & \text{sinon} \end{array}$$

On modèlise la densité de la première classe (après regroupement) par une loi exponentielle dont le paramètre α est aléatoire de densité $h(\alpha)$:

$$\begin{array}{ll} g_1(y \mid \alpha) = \alpha e^{-\alpha y} & y > 0 \\ h(\alpha) = \frac{\delta^r}{(r-1)!}\alpha^{r-1}e^{-\delta\alpha} & \alpha > 0 \quad (\delta > 0) \end{array}$$

Donner un estimateur Bayésien du paramètre α et la règle de classification qui en découle. Application numérique : $\delta = 20, r = 2, n_1 = 30$ et $\prod x_i = 45.10^{-7}$.

4. Afin de confirmer les résultats obtenus à la question précédente, on effectue des mesures concernant un deuxième caractère. Après expertise, on obtient les résultats suivants :

Classe 2 :

$$\begin{aligned} m_1 &= \begin{pmatrix} 0.3 \\ 0.1 \end{pmatrix}, m_2 = \begin{pmatrix} 0.2 \\ 0.0 \end{pmatrix}, m_3 = \begin{pmatrix} 0.0 \\ 0.1 \end{pmatrix} \\ m_4 &= \begin{pmatrix} 0.4 \\ -0.2 \end{pmatrix}, m_5 = \begin{pmatrix} 0.1 \\ -0.2 \end{pmatrix}, m_6 = \begin{pmatrix} 0.5 \\ -0.4 \end{pmatrix} \end{aligned}$$

Classe 1 :

$$m_7 = \begin{pmatrix} 0.7 \\ -0.3 \end{pmatrix}, m_8 = \begin{pmatrix} 0.8 \\ -0.3 \end{pmatrix}, m_9 = \begin{pmatrix} 0.7 \\ 0.2 \end{pmatrix}, m_{10} = \begin{pmatrix} 0.8 \\ 0.3 \end{pmatrix}$$

Expliciter la règle de la distance aux barycentres et comparer les résultats obtenus avec la question 3. Commentaires.

Exercice 2

Considérons une population de N individus et supposons que $p+q$ variables aient été observées pour chacun de ces individus. On note ces variables $X_1, ..., X_p$ et $Y_1, ..., Y_q$. Nous utilisons pour repérer individus et variables les trois indices $i \in \{1, ..., N\}, j \in \{1, ..., p\}$ et $k \in \{1, ..., q\}$. Nous notons x_{ij} (resp. y_{ik}) la valeur de X_j (resp. de Y_k) pour l'individu i. On peut représenter les données par le tableau suivant :

	X_1	...	X_j	...	X_p	Y_1	...	Y_k	...	Y_q
1										
.										
.										
N										

A chaque variable X_j (resp. Y_k), on peut associer un vecteur de $\mathbb{R}^N$ que nous noterons également X_j (resp. Y_k) et qui a pour composantes les x_{ij} (resp. y_{ik}). $\mathbb{R}^N$ est muni du produit scalaire habituel $\langle Z, T \rangle = Z^t T$. Les vecteurs X_i et Y_k définissent des sous espaces de $\mathbb{R}^N$ notés E_x et E_y que nous supposons de dimensions respectives p et q. Le problème de l'ANALYSE CANONIQUE s'énonce en ces termes :

Quels sont les deux vecteurs $Z \in E_x$ et $T \in E_y$ tels que $\cos(Z,T) = \frac{\langle Z,T \rangle}{\|Z\|\|T\|}$ *soit maximum ?*

Plus le cosinus entre ces deux vecteurs est grand, plus l'angle correspondant est petit. Il s'agit donc de trouver le couple des directions de E_x et de E_y qui sont les plus proches. On note $Z = \sum_{i=1}^{p} u_i X_i = XU$ et $T = \sum_{j=1}^{q} v_j Y_j = YV$ avec $U = (u_1, ..., u_p)$ et $V = (v_1, ..., v_q)$. Les matrices X et Y sont de tailles respectives $N \times p$ et $N \times q$.

En utilisant la même démarche que celle étudiée en cours pour l'ACP, on désire déterminer les deux vecteurs U_1 et V_1 de normes unité qui maximisent $\cos(Z,T)$. Pour cela :

a) Ecrire le Lagrangien L correspondant à la maximisation de $\langle Z, T \rangle$ avec les contraintes $\|Z\| = \|T\| = 1$ en fonction de U et V. En déduire les deux équations conduisant à la maximisation de L.

b) En multipliant ces deux équations par des vecteurs convenablement choisis, montrer que les deux facteurs λ et μ du Lagrangien sont égaux.

c) On suppose que les matrices $X^t X$ et $Y^t Y$ sont inversibles et on pose $P_X = (X^t X)^{-1} X^t$ et $P_Y = (Y^t Y)^{-1} Y^t$. Montrer que U_1 et V_1 sont des vecteurs propres des opérateurs $P_X Y P_Y X$ et $P_Y X P_X Y$.

d) conclure en calculant $\cos(Z_1, T_1)$.

8 Examen du 22 Mars 1995

Documents autorisés, les deux exercices sont indépendants.

Exercice 1

Deux classes C_1 et C_2 a priori équiprobables sont définies par des distributions normales caractérisées par les moyennes et matrices de covariance suivantes :

$$\begin{array}{ll} \text{Classe 1 :} & M_1 = 0 \quad \sum_1 = I_p \\ \text{Classe 2 :} & M_2 = (m_1, ..., m_p)^t \quad \sum_2 = I_p \end{array}$$

I_p étant la matrice identité de dimensions $p \times p$.

1. On suppose dans une première partie que le vecteur M_2 est connu. Déterminer la règle de classification appropriée à ce problème.

2. Déterminer la probabilité d'erreur correspondant à la décision précédente à l'aide de la fonction

$$\Phi(x) = \int_x^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-u^2}\, du$$

(On rappelle que si $X = (X_1, ..., X_p)^t$ est un vecteur Gaussien, toute combinaison linéaire $\sum_{i=1}^{p} a_i X_i$ est une variable aléatoire Gaussienne)

3. On désire maintenant estimer le vecteur M_2 à partir de N vecteurs $V_1, ..., V_N$ éléments de la classe C_2. Quel est l'estimateur du maximum de vraisemblance de M_2 ? Donner la règle de décision obtenue à l'aide de cet estimateur.

Exercice 2

On se donne 7 points de $\mathbb{R}^2$ que l'on cherche à regrouper en deux classes C_1 et C_2 :

$$O = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad A = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad B = \begin{pmatrix} 1 \\ -1.5 \end{pmatrix} \quad C = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \quad D = \begin{pmatrix} -3 \\ -1 \end{pmatrix}$$

$$E = \begin{pmatrix} -2 \\ 1 \end{pmatrix} \quad F = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$$

1. Tracer, sous forme d'arbre indicé, une représentation hiérarchique ascendante par la méthode des distances. On utilisera la distance entre groupes suivante

$$d\left(i \cup j, k\right) = Min\left\{d\left(i,k\right), d\left(j,k\right)\right\}$$

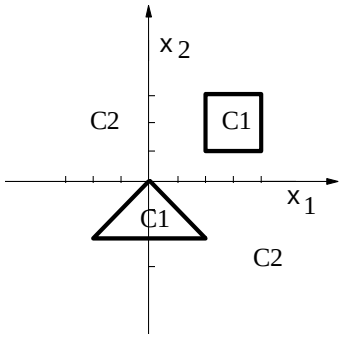
(L'évaluation des distances peut se faire aisément à l'aide de considérations géométriques)

2. Que donne la méthode d'agrégation autour des centres mobiles pour la partition initiale suivante

Classe 1 : A, B
Classe 2 : O, C, D, E, F

Exercice 3

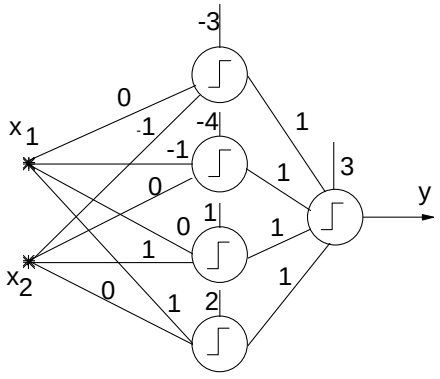
On désire construire un perceptron multi-couches permettant d'obtenir les régions de décision suivantes :



(C_1 : classe 1 et C_2 : classe 2).

1. Déterminer un ensemble de poids et offsets d'un perceptron tel que la sortie soit égale à $+1$ lorsque $x = {}^t(x_1, x_2)$ est à l'intérieur du triangle ci-dessus et égale à -1 lorsque x est à l'extérieur du triangle.

2. Expliciter les régions de décisions correspondant au réseau suivant :



3. Donner alors la structure d'un réseau répondant au problème posé.

9 Examen du 22 Mars 1996

Documents autorisés, les deux exercices sont indépendants.

Exercice 1

On considère dans le plan $\mathbb{R}^2$ sept points définis par :

$$x_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, x_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, x_3 = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$$
$$x_4 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, x_5 = \begin{pmatrix} 0 \\ 2 \end{pmatrix}, x_6 = \begin{pmatrix} 0 \\ -2 \end{pmatrix}, x_7 = \begin{pmatrix} -2 \\ 0 \end{pmatrix}$$

Les trois premiers points sont éléments d'une classe notée C_1 et les quatre derniers appartiennent à une classe C_2.

1. Tracer les zones de décision correspondant à la règle de la distance au barycentre. On précisera les coordonnées des barycentres et l'équation de la frontière entre les deux classes.

2. Soit la fonction *sign* définie par :

$$\begin{aligned} sign\,(x) &= 1 \qquad x > 0 \\ sign(x) &= -1 \qquad x < 0 \end{aligned}$$

   Quelle est la structure la plus simple d'un perceptron, avec non linéarités du type *sign*, permettant d'obtenir les zones de décision de la question précédente ? On précisera la valeur des poids et des offsets.

3. On remplace la non linéarité *sign* par la fonction :

$$f_\alpha(x) = \frac{1 - e^{-(x-\alpha)}}{1 + e^{-(x-\alpha)}} \qquad x \in \mathbb{R}$$

   qui vérifie $f'_\alpha(x) = 2\left[1 - f_\alpha(x)^2\right]$. Donner le critère à mimiser et la règle de mise à jour des poids et offset(s) pour le perceptron de la question précédente.

4. Représenter graphiquement les zones de décision correspondant à la règle du plus proche voisin ? Comparer les résultats obtenus avec ceux de la question 1 et commenter.

Exercice 2

Dans un certain nombre de problèmes de reconnaissance des formes, on a le choix entre classer une forme inconnue parmi K classes notées $\omega_1, ..., \omega_K$ expertisées ou rejeter cette forme ce qui correspond à créer une classe de rejet notée ω_{K+1}. Si le coût associé au rejet n'est pas trop élevé, on obtient des règles de classification très satisfaisantes. Considérons la fonction de coût suivante :

$$\begin{array}{ll} c_{ij} = 0 & i = j \in \{1, ..., K\} \\ c_{K+1j} = \lambda & j \in \{1, ..., K\} \\ c_{ij} = \mu & (i, j) \in \{1, ..., K\}^2, i \neq j \end{array}$$

1. Déterminer les coûts moyens $R_i(x)$ pour $i \in \{1, ..., K+1\}$. On distinguera le cas $i = K+1$ du cas $i \in \{1, ..., K\}$ et on remarquera que :

$$\sum_{k=1}^{K} P[\omega_k \mid x] = 1$$

(x appartient toujours à l'une des classes ω_i, $i \in \{1, ..., K\}$).

2. Montrer que la règle de classification Bayésienne avec rejet est définie par :

$$fd^*(x) = \omega_{K+1} \qquad \text{si} \qquad P[\omega_i \mid x] < 1 - \frac{\lambda}{\mu} \quad \forall i \in \{1, ..., K\}$$

et pour $j \in \{1, ..., K\}$

$$fd^*(x) = \omega_j \qquad \text{si} \qquad \left\{ \begin{array}{c} P[\omega_j \mid x] \geq 1 - \frac{\lambda}{\mu} \\ P[\omega_j \mid x] = \max_i P[\omega_i \mid x] \end{array} \right.$$

3. A quoi correspondent les cas $\lambda = 0$ et $\lambda > \mu$? Que se passe-t-il lorsque $\frac{\lambda}{\mu}$ évolue de 0 à 1.

4. On considère le cas de deux classes équiprobables définies par des densités normales (à une dimension) de même variance $\sigma^2 = 1$ et de moyennes :

$$Classe\ 1 : m_1 = 1$$
$$Classe\ 2 : m_2 = -1$$

Donner les zones d'acceptation des deux classes et la zone de rejet pour $\frac{\lambda}{\mu} = \frac{1}{4}$.

5. Déterminer a pour que les fonctions discriminantes suivantes donnent la même règle de décision que la décision bayésienne avec rejet :

$$g_{K+1}(x) = a\sum_{k=1}^{K} p(x \mid \omega_k) P(\omega_k)$$
$$g_i(x) = p(x \mid \omega_i) P(\omega_i) \quad i \in \{1, ..., K\}$$

Tracer ces fonctions discriminantes pour le problème unidimensionnel de la question 4. Décrire qualitativement ce qui se passe lorsque $\frac{\lambda}{\mu}$ évolue de 0 à 1.

10 Examen du 24 Mars 1997

Documents autorisés, les trois exercices sont indépendants.

Exercice 1 : Décision Bayésienne

1. Rappeler la règle de décision Bayésienne pour deux classes ω_1 et ω_2 de probabilités P_1 et P_2 avec des coûts c_{ij}.

2. On suppose que les vecteurs à classer sont éléments de $\mathbb{R}^n$. Soient D_1 et D_2 les domaines de $\mathbb{R}^n$ définis par :

$$D_i = \{x \in \mathbb{R}^n \quad | \quad fd(x) = \omega_i\} \quad i = 1, 2$$

On note $q(i,j)$ la probabilité de l'événement "$fd(x) = \omega_i$ " $\cap$ "x est élément de la classe ω_j" avec $(i,j) \in \{1,2\}^2$. Montrer que :

$$q(i,j) = P_j \int_{D_i} p(x|\omega_j)\, dx$$

3. On définit le risque global R de la façon suivante :

$$R = \sum_{i=1}^{2}\sum_{j=1}^{2} c_{ij} q(i,j)$$

Montrer que :

$$R = P_1 c_{21} + P_2 c_{22} + \int_{D_1} g(x)\, dx$$

avec $g(x) = P_1(c_{11} - c_{21}) p(x|\omega_1) + P_2(c_{12} - c_{22}) p(x|\omega_2)$.

4. On désire minimiser R par rapport au domaine D_1.

   a) considérer tout d'abord le cas particulier $P_1 = P_2 = 1/2, c_{ij} = 1-\delta_{ij}$, $p(x|\omega_1)$ et $p(x|\omega_2)$ étant les densités des lois uniformes respectivement sur $[-1,+1]$ et $[-1/2,+1/2]$. Représenter la fonction g et déterminer le domaine D_1 qui minimise R.

   b) Plus généralement, quel est le domaine D_1 (fonction de g) qui minimise R ?

5. En déduire que la règle de décision Bayésienne minimise le risque global R.

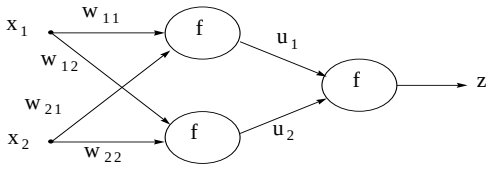
6. On suppose que $c_{ij} = 1 - \delta_{ij}$. Montrer que le risque global R est la probabilité d'erreur de la règle de décision Bayésienne. Conclusion.

Exercice 2 : Perceptrons Multi-couches

On considère un réseau de neurones à deux couches et deux entrées défini de la façon suivante :

* la première couche contient deux neurones (sans biais)
* la deuxième couche contient un seul neurone (sans biais)
* les non-linéarités sont de la forme $f(x) = \frac{1}{1+e^{-x}}$.

On notera y_1 et y_2 les sorties de la première couche et on note $e^2(n) = (d(n) - z(n))^2$ l'erreur instantanée en sortie du réseau. Un tel réseau est représenté sur la figure suivante :



1. Déterminer les dérivées partielles de l'erreur $e^2(n)$ par rapport aux poids du réseau.

2. Donner les règles de mise à jour des poids du réseau en fonction des dérivées partielles déterminées à la question précédente.

3. Expliciter les relations de récurrence qui permettent de mettre à jour les poids de la première couche en fonction des mises à jour des autres couches.

4. On adopte la règle de classification suivante :

$$Classe\ 1 \text{ si } z \geq 1/2$$
$$Classe\ 2 \text{ si } z \leq 1/2$$

Représenter les deux régions de décision (on pourra déterminer l'équation de la frontière) lorsque $w_{11} = w_{22} = 1, w_{12} = w_{21} = 0, u_1 = 1, u_2 = -2$

Comparer ces régions à celles obtenues lorsque les non linéarités sont des échelons ($f(x) = 1$ si $x > 0$ et $f(x) = 0$ si $x \leq 0$).

Exercice 3 : Classification structurelle

Deux classes C_1 et C_2 sont constituées respectivement de carrés et de triangles équilatéraux qui sont tous orientés comme le précise la figure 1.

1. Donner le code de Freeman d'un représentant de chacune de ces classes (On prendra A comme origine et on parcourera la forme dans le sens indiqué).

2. A l'aide de l'algorithme de Wagner et Fisher, déterminer la distance entre le carré privé de l'une de ses branches (représenté sur la figure 2) et les deux représentants des classes C_1 et C_2. En déduire la classe du carré privé de l'une de ses branches.

3. On décide de représenter les formes carré et triangle à l'aide des arbres de la figure 3. On veut déterminer la classe du carré privé de l'une de ses branches représenté par l'arbre de la figure 4. Donner les étapes de cette classification (on donnera un exemple de calcul de distance).

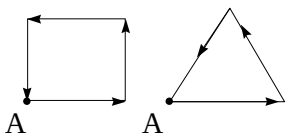


Figure 1

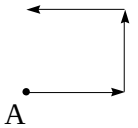


Figure 2

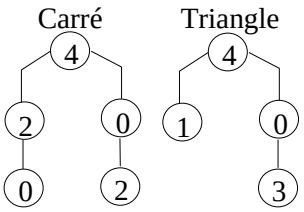


Figure 3

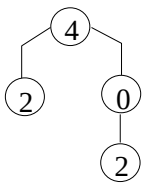


Figure 4

11 Examen du 24 Mars 1998

Documents autorisés, les trois exercices sont indépendants.

Exercice 1 : Décision en mode non supervisé

On considère les 5 observations suivantes $x_1 = 1, x_2 = 7, x_3 = 15, x_4 = 5, x_5 = 2$.

1) Effectuer une classification hiérarchique ascendante (sous forme d'arbre indicé) par la méthode des distances. On utilisera la distance entre groupes suivante :

$$d(i \cup j, k) = \min \{d(i,k), d(j,k)\}$$

2) Que donne la méthode d'agrégation autour des centres mobiles pour la partition initiale suivante ?

$$\begin{array}{lcl} \text{Classe 1} & : & x_1, x_2, x_3 \\ \text{Classe 2} & : & x_4, x_5 \end{array}$$

3) Effectuer une classification hiérarchique ascendante à l'aide de la méthode des moments d'ordre 2.

Exercice 2 : Décision Bayésienne

On définit deux classes ω_0 et ω_1 de probabilités a priori $P_1 = P_2 = \frac{1}{2}$ de la façon suivante :

$$\begin{array}{ll} \omega_0 : & Y = 0 \text{ si l'étudiant canadien EN98 n'a pas la moyenne} \\ & \text{a son examen de classification} \\ \omega_1 : & Y = 1 \text{ sinon} \end{array}$$

On suppose que la seule observation X concernant cet étudiant est le nombre d'heures de révision pour cet examen. On suppose que les densités conditionnelles de chaque classe sont :

$$\omega_0 : \begin{array}{ll} f_0(x) = \frac{1}{a} & x \in [0,a] \\ f_0(x) = 0 & \text{sinon} \end{array} \qquad \omega_1 : \begin{array}{ll} f_1(x) = \frac{2x}{a^2} & x \in [0,a] \\ f_1(x) = 0 & \text{sinon} \end{array}$$

1) Représenter $f_0(x)$ et $f_1(x)$ et interprèter ces densités conditionnelles.

2) Donner la règle de décision Bayésienne associée au problème posé.

3) Donner la probabilité d'erreur correspondante.

4) Montrer que dans le cas de deux classes avec une fonction de coût $c_{ij} = 1 - \delta_{ij}$, la règle de Bayes est équivalente à :

$$\begin{array}{c} d^*(x) = \omega_0 \text{ si } \eta(x) < \frac{1}{2} \\ d^*(x) = \omega_1 \text{ sinon} \end{array}$$

avec $\eta(x) = P[Y = 1 | X = x]$. **Indication** : on pourra utiliser la formule de Bayes et on en déduira $\eta(x) = \frac{P_1 f_1(x)}{P_1 f_1(x) + P_0 f_0(x)}$.

5) Tracer $\eta(x)$ pour le problème précédent. Commentaires.

6) Déterminer $\min(\eta(x), 1 - \eta(x))$ puis $E[\min(\eta(X), 1 - \eta(X))]$. Conclusion

Exercice 3 : Discrimination Linéaire

On considère un problème de décision à deux classes :

$$\begin{aligned} \omega_0 &: Y = 0 \\ \omega_1 &: Y = 1 \end{aligned}$$

et on considère la règle de discrimination linéaire basée sur une observation X à valeurs réelles :

$$\begin{array}{ll} d(x) = y' \text{ si } x \leq x' & \\ d(x) = 1 - y' \text{ sinon} & y' \in \{0, 1\} \end{array}$$

Les densités conditionnelles des deux classes sont notées $f_0(x)$ et $f_1(x)$ et on suppose que les deux classes sont équiprobables.

1) Considérer le cas $y' = 1$ et déterminer l'erreur de la règle de discrimination linéaire notée $P_e^1(x')$.

2) Considérer le cas $y' = 0$ et déterminer l'erreur de la règle de discrimination linéaire notée $P_e^0(x')$.

3) La règle de discrimination linéaire optimale consiste à choisir x' et y' qui minimisent $P_e^{y'}(x')$. La probabilité d'erreur optimale est donc :

$$\begin{aligned} P_{opt} &= \min\left\{\inf_{x'} P_e^0(x'), \inf_{x'} P_e^1(x')\right\} \\ &= \inf_{x'}\left\{\min\left(P_e^0(x'), P_e^1(x')\right)\right\} \end{aligned}$$

Montrer que :

$$P_{opt} = \frac{1}{2} - \frac{1}{2}\sup_{x'} |F_0(x') - F_1(x')| \qquad (E)$$

où $F_0(x)$ et $F_1(x)$ sont les fonctions de répartition associées à $f_0(x)$ et $f_1(x)$. **Indication :** on pourra utiliser l'égalité $\min\{a, b\} = \frac{a+b-|a-b|}{2}$. Interprèter le résultat de l'équation (E).

4) **Application**.

On suppose que

* la loi de X conditionnellement à ω_0 est uniformément répartie sur $[0, a]$ $(a \geq \frac{1}{2})$

* la loi de X conditionnellement à ω_1 est uniformément répartie sur $[b, 1]$ $(b \leq \frac{1}{2})$

Déterminer P_{opt} en fonction de a et b. En déduire que $P_{opt} = 0$ pour $a = b = \frac{1}{2}$.

12 Examen du 26 Mars 1999

Documents autorisés, les deux exercices sont indépendants.

Exercice 1

1) On suppose que les vecteurs à classer sont éléments de $\mathbb{R}^n$ et que l'expertise a donné n classes notées c_i avec $i \in \{1, ..., n\}$. On suppose de plus que la densité de probabilité conditionnellement à la classe c_i est gaussienne de moyenne $m_i = (0, ..., 0, 1, 0, ..., 0)^t$ (le 1 est à la $i^{ème}$ position) et de matrice de covariance l'identité. Donner la règle de classification bayésienne associée à ce problème. Représenter la frontière de décision dans le cas $n = 2$ et les régions correspondant à c_1 et à c_2.

2) Soient $X_1, ..., X_N$ des vecteurs de $\mathbb{R}^n$ de loi normale à n dimensions de moyenne m et de matrice de covariance égale à l'identité. Déterminer la fonction de vraisemblance associée aux N vecteurs $X_1, ..., X_N$. Montrer que maximiser cette fonction de vraisemblance revient à minimiser une forme quadratique par rapport au vecteur m définie par $Q(m) = a^t mm + {}^t bm$ où a est un réel et b est un vecteur que l'on précisera. Quelle est la dérivée de $Q(m)$ par rapport au vecteur m ? En déduire l'estimateur du maximum de vraisemblance de m.

2) On suppose désormais que les classes sont gaussiennes de moyennes m_i inconnues (les matrices de covariance sont toujours égales à l'identité). L'expert fournit N_i vecteurs indépendants correspondant à la classe c_i notés V_j^i, ce qui permet de déterminer l'estimateur du maximum de vraisemblance de chaque vecteur moyenne m_i. Quelle est la règle de décision bayésienne associée ? Considérer le cas suivant ($n = 2$ et $p = 2$) :

$$\text{classe 1 } (c_1) \qquad V_1^1 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}, V_2^1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, V_3^1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, V_4^1 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

$$\text{classe 2 } (c_2) \qquad V_1^2 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, V_2^2 = \begin{pmatrix} -1 \\ -1 \end{pmatrix}, V_3^2 = \begin{pmatrix} -2 \\ 1 \end{pmatrix}, V_4^2 = \begin{pmatrix} -2 \\ -1 \end{pmatrix}$$

3) Que donnent la règle de la distance au barycentre (avec distance euclidienne) et la règle du plus proche voisin pour le problème ci-dessus ($n = 2$) ? On prendra soin de représenter les régions de décisions correspondant aux classes c_1 et c_2.

Exercice 2

Etant donné un problème de classification à K classes **équiprobables**, on se propose de comparer la probabilité d'erreur de la règle de Bayes notée P^* et la probabilité asymptotique de la règle du plus proche voisin notée P_e.

1) Rappeler l'inégalité de Cover et Hart reliant P^* et P_e. En déduire la relation existant entre P^* et P_e lorsque $P^* = 0$ et $P^* = \frac{K-1}{K}$.

2) On se propose d'étudier d'autres cas pour lesquels il existe une relation simple entre P^* et P_e. Etant donné un réel $r \in \left]0, \frac{1}{2}\right[$, on suppose que les densités de probabilité conditionnelles de deux classes ω_1 et ω_2 sont définies par :

$$\text{Classe 1} \quad : \quad f_1(x) = \begin{cases} 1 & \text{si } 0 \leq x \leq 2r \\ 1 & \text{si } 1 \leq x \leq 2-2r \\ 0 & \text{sinon} \end{cases}$$

$$\text{Classe 2} \quad : \quad f_2(x) = \begin{cases} 1 & \text{si } 0 \leq x \leq 2r \\ 1 & \text{si } 2 \leq x \leq 3-2r \\ 0 & \text{sinon} \end{cases}$$

a) Déterminer $P(\omega_1|\, x)$ et $P(\omega_2|\, x)$ dans le cas de classes équiprobables avec des coûts $c_{ij} = 1 - \delta_{ij}$.

b) En déduire la règle de Bayes pour $x \notin [0, 2r]$. Que se passe-t-il pour $x \in [0, 2r]$?

c) Pour $x \notin [0, 2r]$, on adopte la règle de décision Bayésienne et pour $x \in [0, 2r]$, on propose la stratégie suivante :

$$\begin{aligned} d(x) &= \omega_1 \text{ si } x \in [0, r[\\ d(x) &= \omega_2 \text{ si } x \in [r, 2r[\end{aligned}$$

Quelle est la probabilité d'erreur correspondant à cette stratégie ?

d) Pour $x \in [0, 2r]$, on propose alors la stratégie suivante :

$$\begin{aligned} d(x) &= \omega_1 \text{ si } x \in E \\ d(x) &= \omega_2 \text{ si } x \in F \end{aligned}$$

E et F étant deux ensembles quelconques tels que $E \cup F = [0, 2r]$. Calculer la probabilité d'erreur associée à cette règle de décision et en déduire P^*.

e) On rappelle que la probabilité d'erreur asymptotique de la règle du proche voisin est (voir eq. (3.61) p. 33 du poly) :

$$P_e = \int_{\mathbb{R}} (1 - \sum_{i=1}^{K} P^2(\omega_i|\, x)) f(x) dx$$

Calculer P_e et comparer avec P^*

3) Généraliser le cas précédent au problème à K classes équiprobables de densités de probabilité conditionnelles :

$$\text{Classe } i: \ f_i(x) = \begin{cases} 1 & \text{si } 0 \leq x \leq \frac{K}{K-1} r \\ 1 & \text{si } i \leq x \leq i+1-\frac{K}{K-1} r \\ 0 & \text{sinon} \end{cases}$$

13 Examen du 27 mars 2000

Exercice 1

1) Donner la règle de classification bayésienne correspondant au problème suivant :

$$C_1 : f_1(x) \sim N\left(m, \sigma_1^2\right) \qquad C_2 : f_2(x) \sim N\left(m, \sigma_2^2\right)$$

où $\sigma_1 < \sigma_2$ et où les probabilités a priori des deux classes C_1 et C_2 sont P_1 et P_2. Représenter la frontière de décision et les régions d'acceptation des deux classes C_1 et C_2. Déterminer la probabilité d'erreur de cette règle de décision à l'aide de la fonction :

$$\phi(u) = \int_{-\infty}^{u} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} \, dt$$

Commenter succintement (sans calcul) le résultat obtenu.

2) Donner la règle de classification bayésienne correspondant au problème suivant :

$$C_1 : f_1(x,y) \sim N_2\left(m \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \sigma_1^2 I_2\right) \qquad C_2 : f_2(x,y) \sim N_2\left(m \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \sigma_2^2 I_2\right)$$

où $I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. On rappelle que si X et Y sont deux variables aléatoires indépendantes de loi normale $\mathcal{N}(0,1)$, $U = X^2 + Y^2$ suit une loi du χ_2^2 dont la densité est :

$$\begin{aligned} f(u) &= \frac{1}{2} \exp\left(-\frac{u}{2}\right) \qquad u \geq 0 \\ &= 0 \qquad\qquad\qquad\quad u < 0 \end{aligned}$$

En déduire la probabilité d'erreur de la règle de décision bayésienne.

3) On suppose qu'on dispose de n données (x_i, y_i), $i = 1, ..., n$ correspondant à la classe C_i. Quels sont les estimateurs du maximum de vraisemblance de m et de σ_i^2 ?

Exercice 2 : Coalescence Floue

Une partition floue de $X = \{x_1, ..., x_N\}$ (où $x_i \in \mathbb{R}^p$) en K classes $\omega_1, ..., \omega_K$ est définie par un ensemble de K fonctions u_j définies de X dans $A = [0, 1]$ telles que $u_{ij} = u_j(x_i)$ représente le degré d'appartenance de x_i à la classe ω_j avec

$$\sum_{j=1}^{K} u_{ij} = 1, \quad \forall i = 1, ..., N \tag{8.1}$$

On note U la matrice à N lignes et K colonnes constituée des éléments u_{ij}. On suppose que chaque classe est caractérisée par un vecteur de $\mathbb{R}^p$ noté θ_i (qui représente par exemple son centre de gravité) et on pose $\theta = \left(\theta_1^t, ..., \theta_K^t\right)^t$. La plupart des algorithmes de coalescence floue cherchent à minimiser la fonction de coût

$$J_q(\theta, U) = \sum_{i=1}^{N} \sum_{j=1}^{K} u_{ij}^q d(x_i, \theta_j)$$

par rapport à θ et U sous les N contraintes définies par (8.1), d étant une distance définie sur $\mathbb{R}^p$.

1) Déterminer le lagrangien $L_q(\theta, U)$ associé à la minimisation de $J_q(\theta, U)$ sous les contraintes (8.1). En annulant la dérivée partielle de $L_q(\theta, U)$ par rapport à u_{ij}, donner une expression de u_{ij} en fonction des paramètres λ_i $(i = 1, ..., N)$ de ce lagrangien. En utilisant cette expression de u_{ij} et la contrainte $\sum_{j=1}^{K} u_{ij} = 1$, déterminer λ_i en fonction de q, K, θ et des données x_i. En déduire

$$u_{ij} = \left[\sum_{k=1}^{K} \left(\frac{d(x_i, \theta_j)}{d(x_i, \theta_k)}\right)^{\frac{1}{q-1}}\right]^{-1} \tag{8.2}$$

2) On suppose que $d(x_i, \theta_j)$ est le carré de la distance euclidienne, i.e. $d(x_i, \theta_j) = (x_i - \theta_j)^t (x_i - \theta_j)$. En annulant la dérivée partielle de $L_q(\theta, U)$ par rapport à θ_j, donner une expression de θ_j en fonction de q, u_{ij} et des données x_i.

3) Malheureusement, le système d'équations non-linéaires donnant u_{ij} et θ_j ne conduit généralement pas à une expression explicite de u_{ij} et θ_j. Afin d'obtenir une telle expression, on se propose d'étudier un classifieur simplifié appelé classifieur flou dégénéré pour lequel un seul des éléments de la suite $(u_{i1}, ..., u_{iK})$ vaut 1 et tous les autres valent 0. Montrer que la matrice U qui

minimise $J_q(\theta, U)$ vérifie alors :

$$u_{ij} = 1 \text{ si } 1 \text{ si } d(x_i, \theta_j) = \min_{k=1,...,K} d(x_i, \theta_k) \quad (8.3)$$

$$u_{ij} = 0 \text{ sinon} \quad (8.4)$$

On notera que dans ce cas $J_q(\theta, U)$ est indépendant de q.

4) Application numérique : on pose $X = \{x_1, x_2, x_3, x_4\}$ avec

$$x_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, x_2 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}, x_3 = \begin{pmatrix} 0 \\ 3 \end{pmatrix}, x_4 = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

et $K = 2$,$\theta_1 = (1,0)^t$ et $\theta_2 = (1,3)^t$ et on choisit $q = 1$. Déterminer la matrice U associée au classifieur flou dégénéré i.e. conformément à (8.3). En déduire $J_1(\theta, U)$. Dans le cas général (classifieur flou non dégénéré), on pose

$$U = \begin{pmatrix} a & 1-a \\ b & 1-b \\ c & 1-c \\ d & 1-d \end{pmatrix}$$

où a, b, c, d sont quatres réels de l'intervalle $[0, 1]$. Déterminer $J_1(\theta, U)$ et comparer sa valeur avec la valeur obtenue pour le classifieur flou dégénéré (on pourra utiliser une minoration adéquate du style $10 > 1$). Commenter le résultat obtenu.

5) en pratique, on détermine la matrice U et le vecteur θ à l'aide de l'algorithme proposé sur la feuille suivante. Sous quel nom connaissez vous cet algorithme lorsque les termes u_{ij} sont déterminés conformément à (8.3). Rappeler certains inconvénients de cet algorithme.

Algorithme de Coalescence Flou

- Initialisation de θ_j, pour $j = 1, ..., K$,
- $n = 0$
- Répéter
  - * pour $i = 1$ à N
    - ◎ pour $j = 1$ à K

$$u_{ij}(n) = \left[\sum_{k=1}^{K} \left(\frac{d(x_i, \theta_j(n))}{d(x_i, \theta_k(n))} \right)^{\frac{1}{q-1}} \right]^{-1}$$

◎ fin

* fin
* $n = n + 1$
* pour $j = 1$ à K

$$\theta_j(n) = \frac{\sum_{i=1}^{N} u_{ij}^q(n-1)x_i}{\sum_{i=1}^{N} u_{ij}^q(n-1)}$$

- tant que $\|\theta(n) - \theta(n-1)\| < \varepsilon$.

Bibliography

[1] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press, 1995.

[2] M. Chabert, "Détection et estimation de ruptures noyées dans un bruit multiplicatif. Approches classiques et temps-échelle," Ph. D. dissertation, INP of Toulouse, n° 1395, France, 1997.

[3] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification," *IEEE Trans. Inform. Theory*, Vol. IT-13, N°1, pp. 21-27, Jan. 1967.

[4] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-bases Learning Methods*, Cambridge, UK: Cambridge University Press, 2000.

[5] G. Cybenko , "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, 2, pp. 303-314, 1989.

[6] L. Devroye, L. Gyorfi and G. Lugosi, *A Probabilistic Theory of Pattern Recognition.* Berlin: Springer Verlag, 1996.

[7] E. Diday, J. Lemaire, J. Pouget et F. Testu, *Eléments d'Analyse de Données*, Dunod, 1982.

[8] B. Dubuisson, *Diagnostic et Reconnaissance des Formes*, Hermès, 1990.

[9] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis,* Wiley, New York, 1973.

[10] P. Fabre, *Exercices de Reconnaissance des Formes par Ordinateur*, Masson, 1989.

[11] T. Foucart, *Analyse Factorielle-Programmation sur Micros-Ordinateurs*, Masson, 1985.

[12] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, 1972.

[13] S. Haykin, *Neural Networks A comprehensive Foundation*, IEEE Computer Society Press, 1994.

[14] R. Herbrich, *Learning Kernel Classifiers,* MIT Press, 2002.

[15] F. Hlawatsch, *Time-Frequency Analysis and Synthesis of Linear Signal Spaces: Time-Frequency Filters, Signal Detection and Estimation and Range-Doppler Estimation,* Kluwer, Boston, 1998.

[16] A. K. Jain, R. P. W. Duin and J. Mao, "Statistical Pattern Recognition: A Review," *IEEE Trans. PAMI*, Vol. 22, *no*°1, pp. 4-37, Jan. 2000.

[17] M. G. Kendall and A. Stuart, *The Advanced Theory of Statistics*, Griffin, London, 1966.

[18] T. Kohonen, *Self-Organization and Associative Memory*, Third Edition, Springer Verlag, New York, 1988.

[19] R. P. Lippmann, "An introduction to Computing with Neural Nets," *IEEE ASSP Magazine 4*, pp. 4-22, 1987.

[20] L. Miclet, *Méthodes Structurelles pour la Reconnaissance des Formes*, Eyrolles, Paris, 1984.

[21] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in B. Schölkopf, C. J. C. Burges, and A. J. Smola (Eds.), *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, pp. 185-208, MIT Press.

[22] H. Poublan, "Reconnaissance Automatique de Signaux à Evolution Continue", Thèse de l'INP de Toulouse, n°518, France, 1991.

[23] H. Poublan and F. Castanié, "k-Nearest-Neighbor Distribution with Application to Class Likeness Measurement," *IEEE Int. Symp. Inform. Theory*, Budapest, June 1991.

[24] Christian Robert, *L'analyse Statistique Bayésienne*, Economica, 1992.

[25] M. Roux, *Algorithmes de Classification*, Masson, 1985.

[26] F. Rosenblatt, *Principles of Neurodynamics*, Spartan Books, Washington DC, 1962.

[27] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, Vol. 65, pp. 386-408, 1958.

[28] G. Saporta, *Probabilités, Analyse des Données et Statistique*, Technip, 1990.

[29] B. Schölkopf, A. Smola, R. C. Williamson and P. L. Bartlett, "New support vector algorithms," *Neural Computation*, Vol. 12, pp. 1207-1245, 2000.

[30] J. J. Shynk and N. J. Bershad, "Steady-state Analysis of a Single Layer Perceptron Based on a System Identification Model with Bias Terms," IEEE Trans. on Circuit and Systems, vol. CAS-38, pp. 1030-1042, 1991.

[31] S. Sitbon, "Analyse du couple optimal Modélisation Paramétrique / Classification Automatique. Application à l'aide au diagnostic en électromyographie", Thèse de l'INP de Toulouse, n°ordre 281, 1989.

[32] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, London, 1999.

[33] G. V. Trunk, "A problem of dimensionality: a simple example," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 1, no 3, pp. 306-307, July 1979.

[34] V. Vapnik, *The Nature of Statistical Learning Theory*, New York: Springer, 1995.

[35] M. Volle, *Analyse des Données*, Economica, $3^{ème}$ édition, Paris, 1985.