Faculté Polytechnique de Mons

Dissertation originale présentée pour l'obtention du
grade de Docteur en Sciences Appliquées
par

Bernard GOSSELIN

APPLICATION DE RESEAUX DE NEURONES ARTIFICIELS A LA RECONNAISSANCE AUTOMATIQUE DE CARACTERES MANUSCRITS

Membres du jury:

Professeur J. TRECAT, FPMs, Président

Docteur C. PERNEEL, ERM
Professeur C. WELLEKENS, EURECOM

Professeur G. LIBERT, FPMs
Professeur H. LEICH, FPMs, Promoteur
Professeur A. PILATTE, FPMs, Doyen

Année académique 95-96

« Toute thèse s'offre aux rires des dieux »

Kierkegaard

Qu'il me soit permis d'adresser en premier lieu mes sincères remerciements à Monsieur le Professeur Leich, pour m'avoir permis de réaliser dans son laboratoire les recherches dont cette thèse est l'aboutissement, ainsi que pour la confiance qu'il m'a accordée tout au long de celles-ci. Je désire également remercier Monsieur le Professeur Boite, qui le premier éveilla mon intérêt pour ce sujet.

Je remercie les membres du jury, Messieurs les Professeurs Leich, Libert, Trécat, Monsieur le Docteur Perneel, ainsi que Monsieur le Professeur Wellekens, pour avoir accepté d'évaluer le présent travail.

Je tiens à témoigner ma reconnaissance à ceux, et en particulier mon père et mon beau-père, qui, par une lecture de la version préliminaire de mon manuscrit, ont contribué à en améliorer la qualité.

Je tiens également à exprimer toute ma gratitude à mon épouse, pour son soutien indéfectible et pour la tolérance dont elle a fait preuve à mon égard.

Enfin, il serait particulièrement ingrat de ne pas remercier Alfred, Christophe, Daniel, Fabrice, François, Georges, Hervé, Jean-François, Jean-Marc, Jean-Pierre, Nicolas, Olivier D., Olivier VDV., Sophie, Stéphane, Thierry, Vincent, et Xavier, qui, par leur exemple et leur soutien, m'ont permis d'arriver à mes fins.

Table des Matières

Notations

C	nombre total de classes
ω_i	une classe
$(\cdot)^T$	opérateur de transposition
$X = [x_1\ x_2\ \cdots\ x_d]^T$	vecteur de caractéristiques d'un objet à classifier
d	dimension du vecteur de caractéristiques
$p(X)$	probabilité d'occurence du vecteur de caractéristiques *X*
$p(\omega_i)$	probabilité *à priori* de la classe ω_i
$p(X\|\omega_i)$	fonction de répartition du vecteur *X*
$p(\omega_i\|X)$	probabilité *à posteriori* de la classe ω_i
$\underline{X} = [-1\ x_1\ x_2\ \cdots\ x_d]^T$	vecteur augmenté d'un objet à classifier
X_k	$k^{ème}$ vecteur de caractéristiques
N_i	nombre d'objets d'apprentissage de classe ω_i
N	nombre total d'objets d'apprentissage
$E\{\cdot\}$	opérateur d'espérance mathématique
$\mathbf{M}_i = E\{X\|\omega_i\} = [\mu_{i1}\ \mu_{i2}\ \cdots\ \mu_{id}]^T$	vecteur de caractéristiques moyen associé à la classe ω_i
$\mathbf{M} = E\{X\} = \sum_{i=1}^{C} p(\omega_i)\mathbf{M}_i = [\mu_1\ \mu_2\ \cdots\ \mu_d]^T$	vecteur de caractéristiques moyen

$\Sigma_i = E\left\{(X-\mathrm{M}_i)(X-\mathrm{M}_i)^T \mid \omega_i\right\}$	matrice de covariance associée à la classe ω_i
$\Sigma = E\left\{(X-\mathrm{M})(X-\mathrm{M})^T\right\} = \sum_{i=1}^{C}\left[p(\omega_i)\Sigma_i + p(\omega_i)(\mathrm{M}_i-\mathrm{M})(\mathrm{M}_i-\mathrm{M})^T\right]$	matrice de covariance globale
$\Gamma_i = \left[\sigma_{i1}^2\ \sigma_{i2}^2 \cdots \sigma_{id}^2\right]^T$	vecteur des éléments diagonaux de Σ_i
$\Gamma = \left[\sigma_1^2\ \sigma_2^2 \cdots \sigma_d^2\right]^T$	vecteur des variances générales des caractéristiques
$W = \left[w_1\ w_2 \cdots w_n\right]^T$	vecteur de poids d'un neurone à n entrées
$\underline{W} = \left[w_0\ w_1\ w_2 \cdots w_n\right]^T$	vecteur de poids augmenté d'un neurone, w_0 étant le biais
L	nombre de couches d'un perceptron multicouches
h_l	nombre de neurones de la couche l
$W_{l,i} = \left[w_{l,i1}\ w_{l,i2} \cdots w_{l,ih_{l-1}}\right]^T$	vecteur de poids du $i^{ème}$ neurone de la couche l
$Y_l^{(k)} = \left[y_{l,1}^{(k)}\ y_{l,2}^{(k)} \cdots y_{l,h_l}^{(k)}\right]^T$	vecteur des sorties des neurones de la couche l lorsque le $k^{ème}$ exemplaire d'apprentissage est présenté à l'entrée
$Y_0^{(k)} = X_k$	vecteur de caractéristiques du $k^{ème}$ exemplaire d'apprentissage
$\underline{Y}_l^{(k)} = \left[-1\ y_{l,1}^{(k)}\ y_{l,2}^{(k)} \cdots y_{l,h_l}^{(k)}\right]^T$	vecteur augmenté des sorties des neurones de la couche l
$T^{(k)} = \left[t_1^{(k)}\ t_2^{(k)} \cdots t_{h_L}^{(k)}\right]^T$	vecteur des sorties désirées pour la dernière couche L

$u_{l,i}^{(k)} = \underline{W}_{l,i}^{T} \underline{Y}_{l-1}^{(k)}$	entrée de la fonction d'activation du neurone *i* de la couche *l*
$\varphi(\cdot)$	fonction d'activation du neurone *i* de la couche *l*

Chapitre 1

Introduction

L'apparition des premières écritures, il y a près de 5500 ans, a révolutionné à ce point l'humanité, que celle-ci décida ultérieurement que son Histoire débuterait à ce moment. L'importance de l'écriture en tant que moyen de communication pour l'homme n'a fait que croître depuis. Aujourd'hui, le développement constant de l'outil informatique rend nécessaire une communication toujours plus étroite entre l'homme et la machine. C'est à celle-ci de s'adapter aux moyens de communication que l'homme maîtrise à présent et qu'il se complaît à utiliser, telle l'écriture.

La reconnaissance automatique de caractères manuscrits a pour objet de convertir des images qui sont compréhensibles par l'homme, en un code interprétable par un ordinateur. Deux disciplines peuvent être dégagées (*figure 1*): la reconnaissance *en-ligne* et la reconnaissance *hors-ligne*. Dans la reconnaissance en-ligne, les caractères sont reconnus au moment même où ils sont écrits. L'information est dynamique et monodimensionnelle, et consiste en une séquence de traits qui suivent une échelle temporelle. Dans la reconnaissance hors-ligne, au contraire, le processus d'écriture n'est pas accessible. L'information ne possède pas de nature temporelle; elle est bidimensionnelle, et consiste en l'image digitalisée des caractères. Nous ne nous intéresserons ici qu'à la reconnaissance hors-ligne.

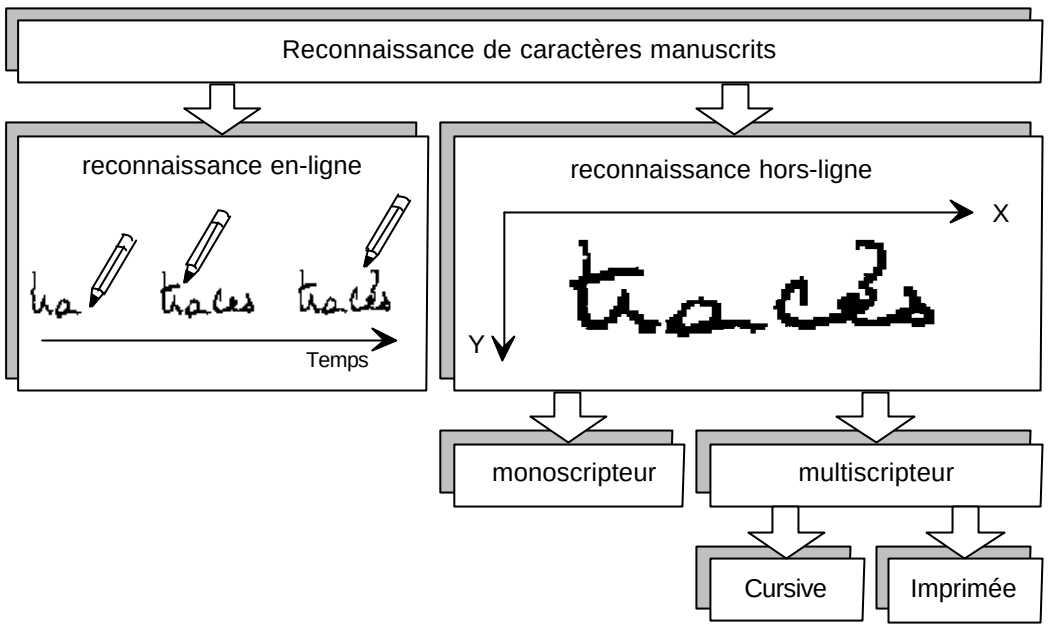


Figure 1 - Disciplines de la reconnaissance de caractères manuscrits.

La reconnaissance hors-ligne peut elle-même être scindée en deux domaines, en fonction du nombre de personnes dont l'écriture doit pouvoir être reconnue. Lorsque ce nombre est limité, le système de reconnaissance peut subir un entraînement spécifique sur base de l'écriture propre aux personnes concernées. On parle alors de reconnaissance *monoscripteur*. Lorsqu'un tel entraînement n'est pas possible, parce que les écritures peuvent provenir d'un très grand nombre de personnes distinctes, on parle de reconnaissance *multiscripteur*. C'est à ce niveau que s'inscrit ce travail.

Le style d'écriture lui-même peut en outre être réparti en deux catégories. L'écriture est dite *cursive* lorsque les lettres qui composent un mot sont constituées d'un seul trait continu, et est dite *imprimée* lorsque chaque caractère est constitué de traits qui lui sont propres. Bien que le premier mode soit la façon la plus naturelle pour l'homme d'écrire, c'est le second qui est le plus fréquemment rencontré en pratique, lorsque des documents provenant de nombreuses personnes distinctes doivent être gérés par l'homme lui-même. Cette contrainte est alors nécessaire pour limiter la dispersion d'aspect des caractères, et rendre ainsi ces derniers plus aisément exploitables par l'homme. C'est ce dernier mode qui constitue l'objet de notre étude.

La reconnaissance hors-ligne de caractères imprimés manuscrits dans un contexte multiscripteur est, depuis plusieurs années, l'objet d'intenses recherches, justifiées par l'importance de ses nombreuses applications potentielles. Ces dernières ont essentiellement pour but de faciliter la gestion de volumes de données toujours plus considérables.

L'automatisation du tri postal en est un exemple concret. Au vu de la quantité colossale de courrier à traiter chaque jour, un système de reconnaissance automatique de codes postaux, même partiellement efficace, permettrait de réduire avantageusement l'ampleur de cette tâche désagréable et fastidieuse. De même, la gestion des fréquents transferts financiers qui s'effectuent, actuellement encore, par mandats ou virements bancaires, serait-elle ainsi facilitée. Enfin, de nombreux documents administratifs doivent toujours être encodés à la main. Leur acquisition ainsi que leur archivage automatique, représenteraient également des gains considérables de temps.

La reconnaissance automatique de caractères manuscrits pourrait également s'intégrer dans des applications multimédia, domaine actuellement en plein essor s'il en est. Couplée à un système de synthèse vocale à partir du texte, elle serait alors à même de fournir une aide appréciable aux non-voyants, par exemple. Une autre application envisageable est la compression de documents, qui peut être réalisée en sauvegardant le texte qu'ils contiennent sous un format informatique, plutôt qu'en le conservant sous forme d'image. Cela rendrait en outre possible une indexation de ces documents, ainsi qu'une recherche aisée de sujets particuliers parmi ceux-ci. Appliqué à la lecture de fax, la compression de l'information qui serait obtenue permettrait de réduire les délais de transmission, tout en autorisant une excellente qualité d'impression à la réception. Cela rendrait également possible de communiquer l'information directement au travers d'une messagerie électronique.

Structure d'un Système de Reconnaissance Automatique

La reconnaissance automatique de l'écriture manuscrite « imprimée » s'effectue essentiellement en trois étapes (*figure 2*). La première d'entre elles consiste à analyser l'image brute fournie par un périphérique de digitalisation, de manière à localiser et isoler les caractères les uns des autres. C'est la segmentation du texte, qui décompose ce dernier en symboles élémentaires dont le nombre de classes distinctes possibles est limité, rendant ainsi plus aisée la tâche du système de classification. Malgré le style « imprimé » de l'écriture, cette étape est loin d'être triviale en pratique. Les contraintes d'écriture imposées sur les documents originaux, destinées à la faciliter, ne sont en effet pas toujours respectées. Elles demeurent en outre un frein à la popularisation des systèmes de reconnaissance automatique de caractères.

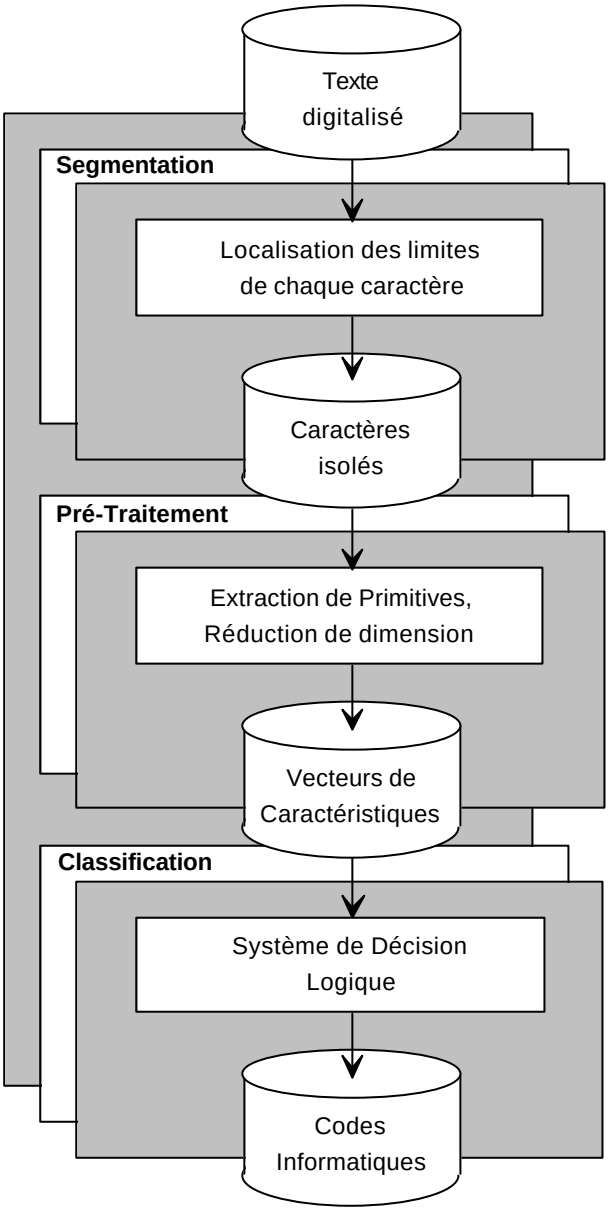


Figure 2 - Structure d'un système de reconnaissance automatique de caractères manuscrits « imprimés ».

Pour obtenir une bonne qualité de représentation des caractères et assurer ainsi qu'une procédure efficace de reconnaissance puisse être menée, la digitalisation doit s'effectuer à relativement haute résolution. Un caractère est alors représenté dans l'ordinateur par une matrice de grandes dimensions. Afin de réduire la dimension de représentation, et, ce faisant, de rendre plus facile la conception du système de classification, une étape d'extraction de primitives est

nécessaire. C'est le rôle du module de pré-traitement. Lorsque des caractéristiques discriminantes ont été extraites, elles sont soumises à un système de décision logique qui a pour tâche d'identifier le caractère qu'elles représentent, et de leur assigner le code informatique qui lui correspond. C'est la phase de classification proprement dite.

Depuis plusieurs années, les réseaux de neurones artificiels, et particulièrement les *perceptrons multicouches*, se sont montrés très efficaces dans le domaine de la reconnaissance statistique de formes. Ces systèmes sont composés d'un ensemble structuré d'unités de traitement, appelées *neurones*, qui fonctionnent en parallèle et qui sont fortement interconnectées. Ces neurones réalisent chacun une fonction non linéaire de leurs entrées, qui est déterminée par un ensemble de paramètres dont les valeurs sont établies à la suite d'un apprentissage par l'exemple. Les non-linéarités présentes dans ces systèmes connexionnistes, ainsi que l'apprentissage discriminant qu'ils subissent, rendent ces derniers particulièrement bien adaptés aux tâches de classification. Cependant, leurs performances sont fortement affectées par la qualité de représentation des objets à classifier. Pour des caractères, cela implique une grande dimension de représentation et, dans ce cas, les perceptrons multicouches peuvent devenir délicats à entraîner. Ils ne peuvent en outre être performants que si les caractéristiques qui leur sont fournies, sont pertinentes. Enfin, des réseaux de neurones de grande taille sont aussi très exigeants en puissance de calcul, ce qui restreint fortement leurs possibilités d'application.

Outre la disponibilité d'une procédure de segmentation qui permette d'assouplir les contraintes imposées à l'écriture, il apparaît dès lors également nécessaire, d'une part, de disposer d'algorithmes efficaces de pré-traitement des caractères, et d'autre part, d'optimiser les performances obtenues grâce à l'emploi de réseaux de neurones artificiels, à la fois en termes de qualité de classification et en termes de temps de calcul. C'est sur l'ensemble de ces points critiques que porte notre contribution personnelle.

Synopsis

Le chapitre 2 est entièrement consacré à la première des étapes de la reconnaissance automatique de caractères: la segmentation. Un algorithme classique, destiné à l'isolation de caractères suffisamment espacés, y est d'abord présenté. Un algorithme original est ensuite introduit, et permet la séparation de caractères qui se chevauchent. Nous décrivons ensuite deux méthodes qui permettent d'associer ces algorithmes afin de minimiser la durée moyenne de la procédure de segmentation.

Nous poserons au chapitre 3 le problème de la reconnaissance statistique de formes, et présenterons divers modèles de classificateurs, conventionnels et neuronaux. Leurs propriétés, ainsi que le degré de complexité des frontières de décision qu'ils sont à même de générer dans l'espace des variables, seront également décrits.

L'entraînement du perceptron multicouches, souvent considéré comme délicat, est l'objet d'une étude plus approfondie au chapitre 4. Plusieurs méthodes permettant d'accélérer sa phase d'apprentissage sont décrites, et un nouvel algorithme est présenté. Bien que ce dernier s'avère être très performant lorsque l'apprentissage du perceptron multicouches est contraint à s'effectuer hors ligne, ce sont d'autres méthodes qui se révèlent être globalement plus efficaces, lorsque des modes d'apprentissages alternatifs peuvent être envisagés. Les performances des divers classificateurs présentés au chapitre 3 sont ensuite comparées sur base d'un problème de reconnaissance de lettres manuscrites majuscules, dans un contexte multiscripteur. Le perceptron multicouches apparaît alors comme étant celui offrant la meilleure qualité de reconnaissance. L'importance de l'application, aux classificateurs, d'un apprentissage discriminant, est également mise en évidence.

Au cours du chapitre 5, nous analysons et améliorons plusieurs méthodes d'extraction de primitives. Ceci permet de dégager deux méthodes particulièrement efficaces pour la reconnaissance hors-ligne de caractères manuscrits. Une procédure d'analyse discriminante est également décrite et appliquée, et permet de réduire de manière très significative la dimension de représentation des caractères. Ces méthodes nous ont permis d'atteindre, pour des caractères n'ayant pas participé à l'entraînement des perceptrons multicouches, des taux de reconnaissance de 96,2% et de 98,7%, respectivement pour des lettres et des chiffres manuscrits extraits de la base de données NIST3 [Garris,92], très répandue et autorisant de ce fait des comparaisons objectives des performances. Ces taux de reconnaissance sont supérieurs à ceux rencontrés par ailleurs dans la littérature, ce qui prouve la qualité, tout autant que l'utilité, du pré-traitement des caractères effectué ici.

Nous consacrons ensuite le chapitre 6 au développement d'un filtre original, qui permet d'appliquer des distorsions diverses à des images de caractères. Ceci permet d'améliorer la capacité de généralisation du perceptron multicouches, en augmentant de manière artificielle la taille de la base de données destinée à son entraînement. Les paramètres du filtre autorisent un contrôle aisé des distorsions appliquées, ce qui limite le risque de génération de caractères trop ou pas assez déformés. Ceci a permis d'entraîner un système offrant un taux de reconnaissance

---

[Garris,92] **M.D. Garris & R.A. Wilkinson**
Handwritten segmented characters Database
Technical Report Special Database 3, HWSC, National Institute of Standards and Technology, Gaithersburg, Maryland, USA, Février 1992

de 91,2% sur des chiffres de formes telles que celles qui peuvent se présenter en nos contrées, présentant une dispersion d'aspect bien plus élevée que celle des chiffres qui constituent la base de données NIST3.

Enfin, le chapitre 7 propose diverses méthodes de mise en coopération de perceptrons multicouches. Nous montrons dans un premier temps que lorsque des réseaux de neurones entraînés sur base de primitives distinctes ne produisent pas les mêmes erreurs de classification, leur mise en collaboration, par l'intermédiaire de leur couche cachée ou par celui de leur couche de sortie, permet de réduire de manière significative le taux d'erreur de classification. Nous présentons ensuite plusieurs procédés nouveaux de coopération en cascade ou par substitution de perceptrons multicouches, basés sur le rejet de caractères par un réseau de neurones lorsque le niveau de ses sorties est inférieur à un seuil prédéfini. Ces modes de coopération permettent de conserver un taux de reconnaissance élevé, tout en minimisant le temps moyen nécessaire à la classification d'un caractère. Le système de reconnaissance final consiste ainsi en une association en cascade de quatre perceptrons multicouches de faibles dimensions, qui offre des taux de reconnaissance de 96,7% et 99,8%, respectivement pour des lettres et des chiffres provenant de la base de données NIST3, ainsi qu'une vitesse de reconnaissance qui est jusqu'à 2,7 fois plus élevée que celle obtenue à l'aide des meilleurs des perceptrons multicouches isolés. Appliquée aux chiffres rencontrés en nos pays, ces méthodes ont permis d'atteindre un taux de reconnaissance de 95,3%, à l'aide d'un système qui est près de 1,5 fois plus rapide que le meilleur des réseaux de neurones considérés isolément. Ce système de reconnaissance final est actuellement l'objet d'une série de tests objectifs, organisés, par un organisme indépendant, au niveau Européen. En outre, une application pratique en est également en cours de développement, avec le concours d'un partenaire industriel, et vise le domaine du traitement automatique de virements bancaires.

Chapitre 2

La Segmentation des Caractères

2.1 Introduction

Première étape du système de reconnaissance, la segmentation a pour tâche d'analyser l'image brute fournie à l'ordinateur par le périphérique de digitalisation, afin de localiser les caractères qu'elle contient. Outre les limites des caractères qui composent l'image, diverses informations peuvent également être extraites lors de la procédure de segmentation. Ces informations sont par exemple celles qui sont requises pour pouvoir reconstituer les mots à partir de leurs lettres isolées, ou pour pouvoir effectuer une remise en forme du document aussi semblable que possible à celle de l'original. Selon les cas, le rôle de l'étape de segmentation peut aussi inclure la recherche et l'élimination de l'image des graphiques, des photographies, des tableaux, et de manière plus générale, de tout ce qui ne constitue pas du texte. La procédure de segmentation s'avère être une phase critique car les erreurs commises ici seront difficilement corrigibles dans les étapes ultérieures, pouvant ainsi dégrader fortement les performances globales du système de reconnaissance.

Selon le type de périphérique utilisé, les points élémentaires de l'image, ou *pixels*[1], peuvent prendre une valeur binaire 0 ou 1, ou une valeur quantifiée représentant un niveau de gris ou une couleur. Dans ces derniers cas, l'image doit subir un processus de binarisation. Ce dernier permet de mettre en valeur les caractères, ou toute autre information utile, vis-à-vis du fond de l'image,

[1] terme issu de la contraction des mots anglais « ***pic****ture* » et « ***el****ement* ».

dont l'influence est ainsi éliminée. La détermination des limites des caractères est ainsi rendue plus aisée. Une technique de binarisation utilisée couramment consiste à assigner aux pixels une valeur binaire, en fonction du rapport entre leur intensité réelle et la valeur moyenne de l'intensité de l'image. Des algorithmes plus complexes de binarisation ont également étés élaborés afin de résoudre diverses situations délicates. Leur description sort cependant du cadre de l'étude menée ici, et nous renvoyons le lecteur intéressé à la littérature [Parker,93] [Pavlidis,93] [Liu,93]. Les images traitées dans la suite de ce travail seront supposées avoir été préalablement binarisées et ne contenir que des caractères.

La procédure de segmentation peut être facilitée lorsque le document original dispose de cases pré-imprimées, destinées à contraindre l'écriture. C'est le cas de beaucoup de documents administratifs, ou même de bulletins de virements bancaires, par exemple. La procédure de segmentation ne devient cependant pas triviale pour autant. En pratique, nombre de personnes ne respectent pas toujours correctement les limites indiquées, et surtout, l'imposition de contraintes d'écriture demeure un frein à la popularisation des systèmes de reconnaissance automatique de caractères. Enfin, ces mêmes contraintes d'écriture sont inexistantes sur certains documents ou dans certains environnements. Nous avons donc cherché à développer un algorithme de segmentation qui soit apte à traiter des documents qui ne comportent aucune contrainte d'écriture *a priori*.

2.2 La Segmentation Primaire

2.2.1 Les Principes de Base

[Parker,93] **J.R. Parker, C. Jennings, and A.G. Salkauskas**
Thresolding Using an Illumination Model
Actes de la 2nd Int. Conf. on Document Analysis and Recognition, pp 270-273, Tsukuba, Japon, Octobre 1993

[Pavlidis,93] **T. Pavlidis**
Threshold Selection Using Second Derivatives of the Gray Scale Image
Actes de la 2nd Int. Conf. on Document Analysis and Recognition, pp 274-277, Tsukuba, Japon, Octobre 1993

[Liu,93] **Y. Liu, R. Fenrich, and S.N. Srihari**
An Object Attribute Thresholding Algorithm for Document Image Binarization
Actes de la 2nd Int. Conf. on Document Analysis and Recognition, pp 278-281, Tsukuba, Japon, Octobre 1993

La segmentation primaire est une procédure de séparation de caractères qui se déroule en deux phases. Ces dernières sont basées sur les principes élémentaires suivants, qui dérivent directement de la structure même du processus d'écriture:

1. La détection des passages par zéro de l'histogramme de projection horizontale (*figure 2.1*), doit permettre d'isoler les lignes de texte les unes des autres.

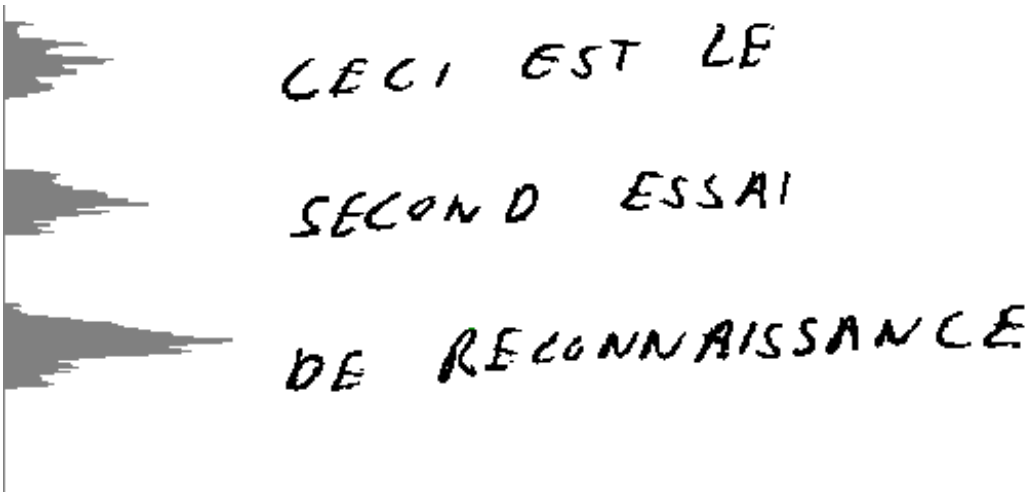


Figure 2.1 - Calcul de l'histogramme de projection horizontale.

2. La détection des passages par zéro de l'histogramme de projection verticale (*figure 2.2*), calculé sur une ligne de texte isolée et extraite de l'image, doit autoriser la séparation des caractères qui composent la ligne.

Figure 2.2 - Calcul de l'histogramme de projection verticale.

Ces deux principes très simples à mettre en oeuvre ne peuvent cependant pas être appliqués tels quels, et doivent être modifiés afin de pouvoir faire face aux divers problèmes qui se présentent en pratique.

2.2.2 La mise en Oeuvre de l'Algorithme

2.2.2.1 Le Problème des Pixels Parasites

Ce problème apparaît presque systématiquement sur tout document digitalisé. Ce sont des pixels bruités (*figure 2.3*), qui résultent soit de la présence de taches sur le document initial, soit d'un choix imparfait du seuil de binarisation. Un filtrage de l'image doit donc être effectué, de manière à réduire le nombre de ces pixels parasites.

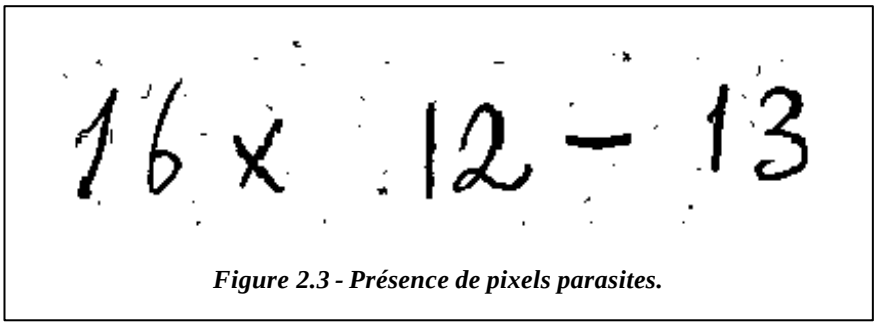


Figure 2.3 - Présence de pixels parasites.

Ce filtrage est effectué au cours de chacune des phases de la segmentation primaire. Lors de la première d'entre elles, l'image est décomposée en lignes de texte potentielles qui sont délimitées grâce au calcul de l'histogramme de projection horizontale. Le filtrage des pixels parasites s'effectue alors en ne considérant comme lignes de texte effectives que celles dont la hauteur en pixels est supérieure à un seuil prédéfini. La valeur de ce dernier doit être choisie de manière à obtenir le filtrage le plus complet possible, tout en évitant l'élimination de caractères ou de symboles significatifs de faible taille (le plus critique d'entre eux étant le signe de ponctuation « . »). Le seuil de validation des lignes de texte est donc dépendant de la taille absolue des caractères originaux, et doit en outre être adapté proportionnellement à la résolution utilisée par le périphérique de digitalisation lors de l'acquisition de l'image. Les essais expérimentaux que nous avons effectués nous ont conduits à utiliser un seuil de 9 pixels, lequel est valable pour une écriture courante, digitalisée avec une résolution de 300 Points par Pouce. Cette valeur qui correspond à une tache d'environ 0.75 mm sur le document original s'est révélée suffisante pour éliminer tous les pixels parasites dus au bruit de binarisation, ainsi que la plupart des taches causées par une qualité moyenne d'impression. Les taches de dimensions plus importantes sont rares, et n'apparaissent plus que dans des cas devenus exceptionnels de très mauvaise qualité d'impression.

Un filtrage similaire est effectué lors de la seconde phase de la segmentation primaire. Pour qu'une partie d'image délimitée par deux passages par zéro de l'histogramme de projection verticale soit considérée comme étant effectivement un caractère, il est obligatoire que sa largeur

soit de dimension suffisante (*figure 2.4*). La dimension minimale requise ici est, en toute logique, la même que celle exigée lors de l'étape de validation des lignes de textes.

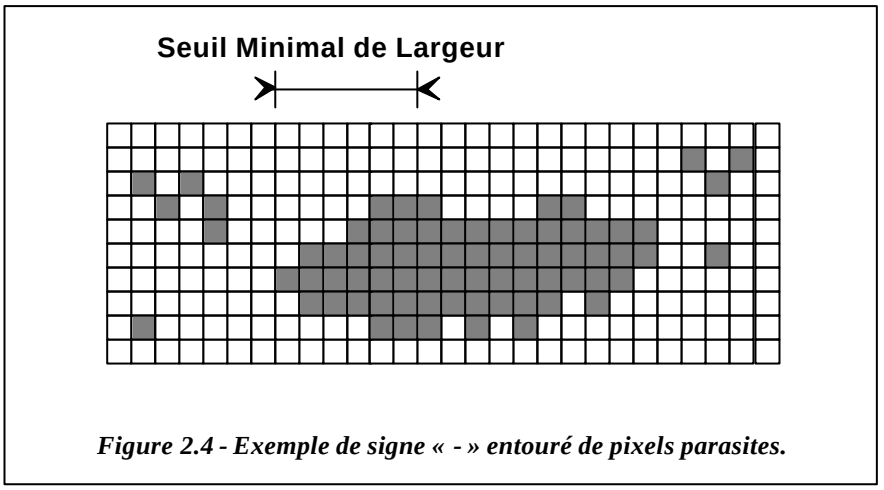


Figure 2.4 - Exemple de signe « - » entouré de pixels parasites.

La *figure 2.5* illustre un cas critique de segmentation qui peut apparaître en présence d'un bruit dense. Afin de pouvoir également procéder à l'élimination de ces pixels parasites, une condition supplémentaire de validation des caractères dut être introduite. Celle-ci stipule qu'une partie d'image susceptible de représenter un caractère doit, pour être effectivement considérée comme tel, posséder une densité suffisante de pixels d'intensité non nulle. Pour des raisons pratiques, nous avons voulu ce second seuil dépendant du carré de la valeur du premier, et nos essais nous ont conduits à utiliser la moitié de celui-ci.

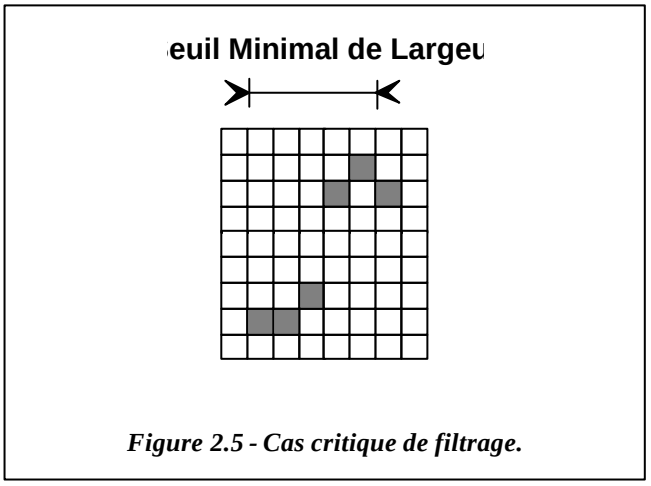


Figure 2.5 - Cas critique de filtrage.

2.2.2.2 Le Défaut d'Encrage Insuffisant

Deuxième type de défaut susceptible d'apparaître, l'encrage insuffisant peut, dans certains cas, provoquer la scission d'un caractère en deux ou plusieurs parties (*figure 2.6*). Une manière de lutter contre cet effet indésirable est de n'effectuer la segmentation des caractères que lorsqu'un nombre suffisant de zéros consécutifs de l'histogramme de projection verticale a été enregistré. A l'issue des essais expérimentaux menés, la même valeur de seuil que celle utilisée pour le filtrage des taches a été adoptée.

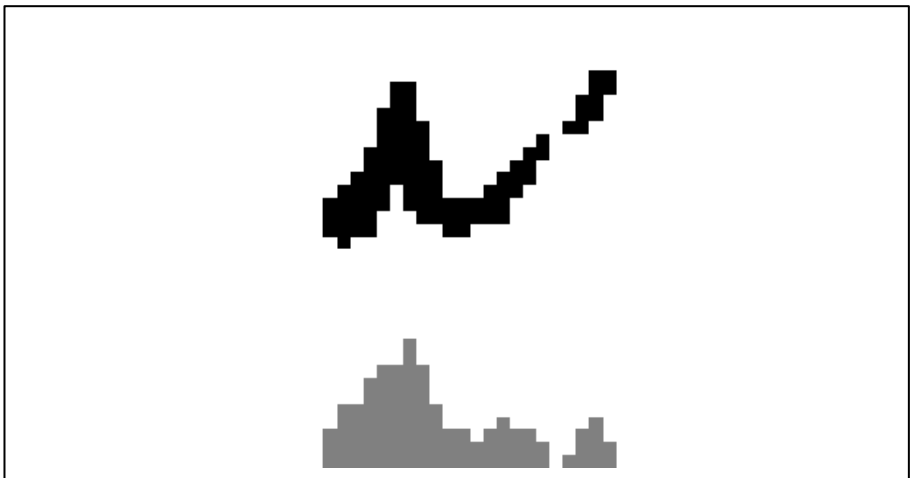


Figure 2.6 - Lettre « N » présentant un défaut d'encrage insuffisant.

2.2.2.3 Le Chevauchement de Caractères

Relativement fréquent en pratique, le chevauchement de deux ou de plusieurs caractères (*figure 2.7*) est un problème qui dépend en grande partie du soin apporté à l'écriture.

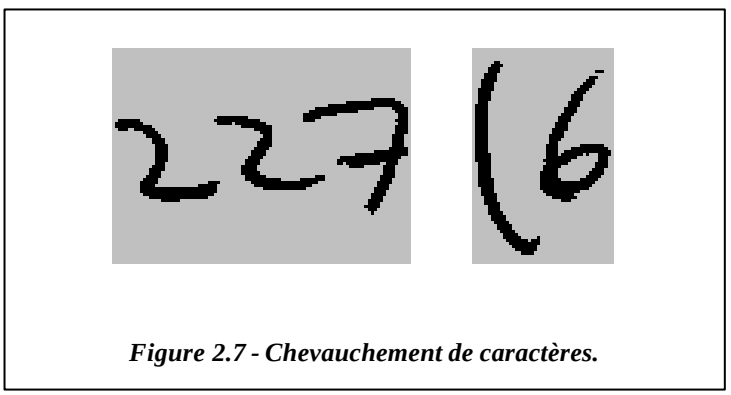


Figure 2.7 - Chevauchement de caractères.

Ce phénomène est souvent dû à une écriture trop inclinée, ou peut apparaître à cause d'un alignement imparfait entre l'axe de digitalisation et celui du texte écrit. Sous certaines conditions, il est possible d'appliquer des méthodes utilisées en écriture cursive afin d'estimer la valeur de la pente de l'inclinaison du texte. Lorsque cette valeur est déterminée, une rotation d'amplitude inverse est appliquée à l'image, ce qui peut faire apparaître des discontinuités du tracé qui étaient auparavant masquées par la projection des caractères inclinés (*figure 2.8*) [Bozinovic,89] [Schumacher,93]. L'efficacité de cette méthode n'est toutefois pas assurée, et son principal inconvénient est qu'elle requiert la présence dans l'image de caractères élancés.

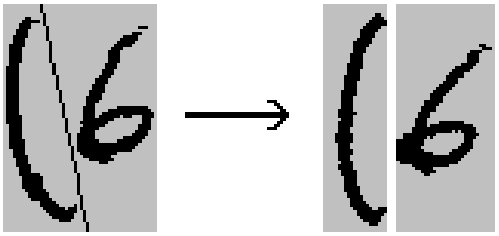


Figure 2.8 - Correction de pente et segmentation.

Cette possibilité trop restreinte d'application nous a conduit à développer un algorithme de segmentation dont les principes s'écartent assez nettement de ceux de la segmentation primaire, au point d'en faire une méthode de segmentation à part entière. En outre, à cause du volume de calcul relativement élevé requis par cet algorithme, son intégration à la procédure de segmentation primaire conduirait à un ralentissement considérable de la vitesse d'exécution de cette dernière. Afin de conserver une étape de segmentation primaire rapide, la résolution du problème du chevauchement de caractères est donc intégrée à une procédure de segmentation distincte, appelée *segmentation secondaire*, dont la description est l'objet de la section 3 du présent chapitre.

[Bozinovic,89] **R.M. Bozinovic and S.N. Srihari**
Off-line Cursive Word Recognition
IEEE Transactions on PAMI, Vol 11, n°1, pp. 68-83, 1989

[Schumacher,93] **L. Schumacher**
Segmentation de l'Ecriture Cursive
Travail de Fin d'Etudes, Faculté polytechnique de Mons, Juin 1993

2.2.2.4 Le Défaut d'Encrage Excessif

Le plus complexe des défauts à corriger est la concaténation de plusieurs caractères en un seul à la suite d'un encrage excessif de ceux-ci (*figure 2.9*).

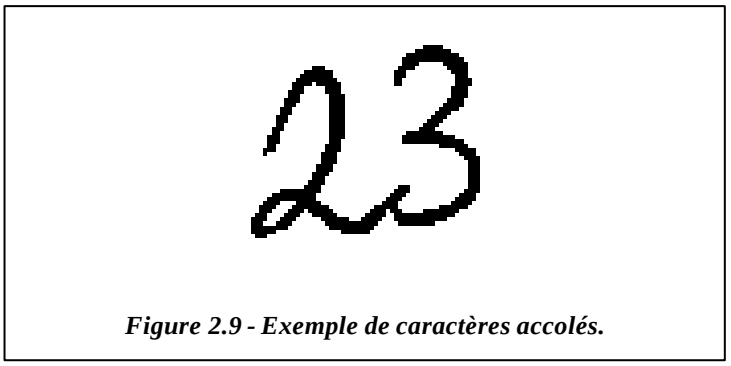


Figure 2.9 - Exemple de caractères accolés.

Une méthode de segmentation très simple à mettre en oeuvre a été proposée par Tsujimoto, dans le cadre de la segmentation de caractères typographiques [Tsujimoto,91]. L'algorithme, dit du «Break Cost », consiste à effectuer un ET logique entre chaque paire de colonnes contiguës de pixels avant de calculer l'histogramme de projection verticale. L'efficacité de cette méthode demeure malheureusement fort limitée, puisque celle-ci ne permet de faire apparaître de nouveaux passages par zéro de l'histogramme que là où les caractères n'étaient en contact que par deux pixels situés en diagonale (*figure 2.10*).

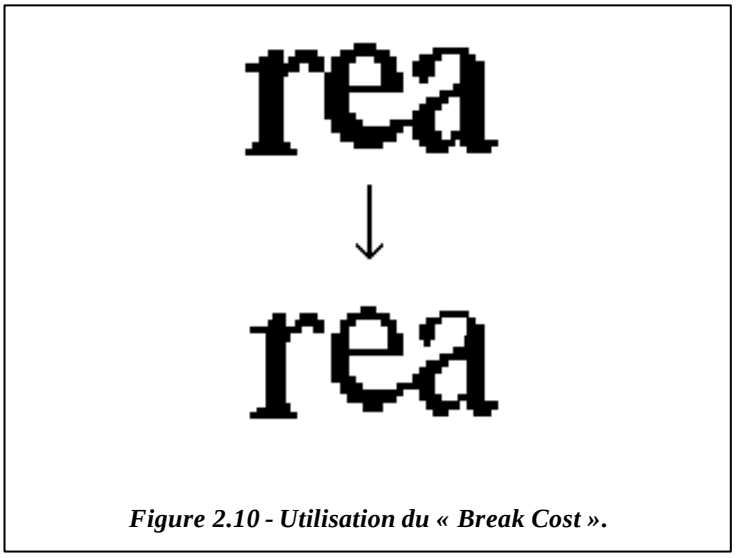


Figure 2.10 - Utilisation du « Break Cost ».

[Tsujimoto,91] **S. Tsujimoto and H. Asada**
Resolving Ambiguity in Segmenting Touching Characters
Proc. First Int. Conf. Document Analysis and Recognition, Saint-Mâlo, France, Vol2, pp 701-709, Octobre 1991

Fujisawa décrit dans [Fujisawa,92] une méthode plus performante de segmentation de caractères accolés, où l'épaisseur du tracé remplace l'histogramme de projection verticale. Cette manière distincte d'aborder le problème de la segmentation ainsi que l'augmentation significative du volume de calcul que cette méthode implique nous ont également conduits à inclure cette dernière à la procédure de segmentation secondaire.

2.2.3 Conclusions

La fréquence relative d'apparition de chaque type de défaut peut varier fortement d'une application à l'autre. Les types de défauts les plus fréquents demeurent toutefois la présence de pixels parasites et le chevauchement de caractères. Bien que ce dernier problème ne puisse pas être résolu à l'aide de la segmentation primaire, le très faible temps de calcul moyen requis par cette méthode peut en justifier l'utilisation.

L'ensemble de la procédure de segmentation primaire est résumée par l'organigramme ci-dessous.

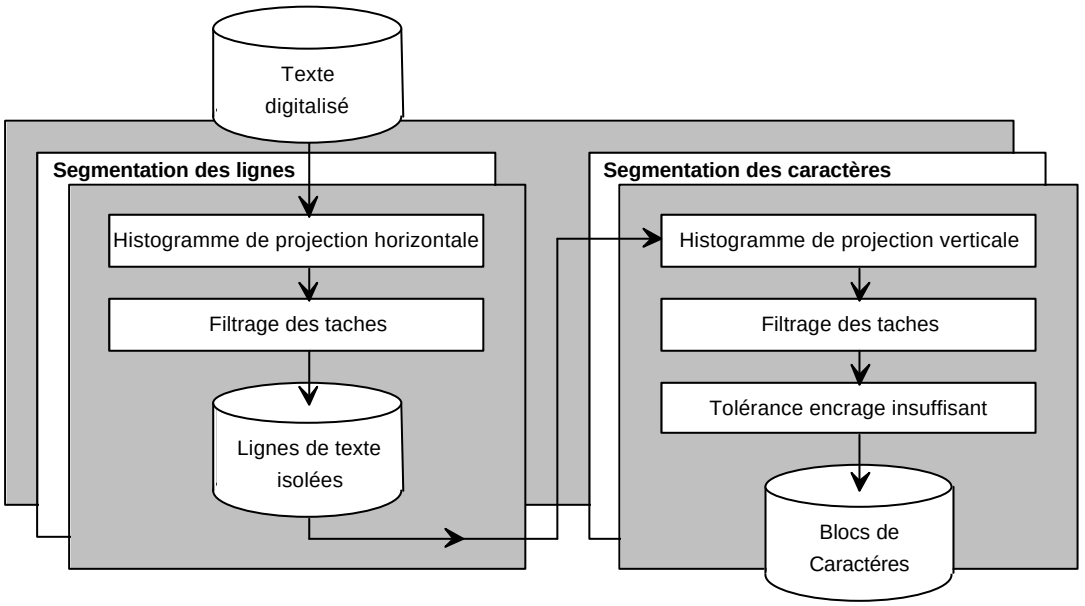


Figure 2.11 - La Procédure de Segmentation Primaire

[Fujisawa,92] **H. Fujisawa, Y. Nakano, and K. Kurino**
Segmentation Methods for Character Recognition: From Segmentation to Document Structure Analysis
Proc. of the IEEE, Vol 80, n°7, pp 1079-1091, Juillet 92.

2.3 La Segmentation Secondaire

2.3.1 Les Principes de Base

La procédure de segmentation secondaire a pour rôle d'analyser la portion d'image délimitée à l'issue de la segmentation primaire, afin d'y détecter la présence éventuelle de plusieurs caractères se chevauchant ou connectés entre eux, et, le cas échéant, d'établir des chemins possibles de séparation (*figure 2.12*).

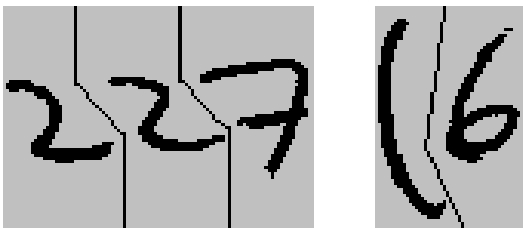


Figure 2.12 - Chevauchement de caractères et chemins de segmentation.

L'algorithme voué à la segmentation de caractères qui se chevauchent est basé sur la détection, pour chaque ligne de l'image et de la gauche vers la droite, de toutes les transitions allant de l'arrière-plan vers le corps d'un caractère. L'existence d'au moins deux transitions sur une même ligne indique alors soit la présence d'un second caractère, soit celle d'une boucle dans un caractère (*figure 2.13*). Le chaînage des transitions présentes sur plusieurs lignes consécutives permet ensuite de faire la distinction entre ces deux situations, ainsi que de construire les chemins de segmentation.

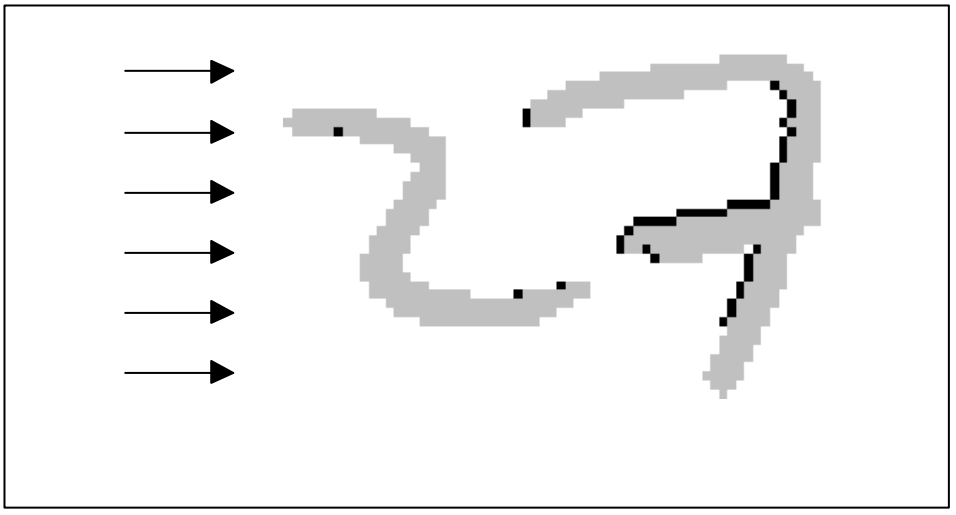


Figure 2.13 - Détection des transitions multiples.

Lorsqu'aucune segmentation ne peut être effectuée à l'aide de cette méthode, un algorithme destiné à segmenter des caractères connectés entre eux est utilisé. Comme brièvement cité précédemment, ce dernier se base sur une analyse de l'épaisseur du tracé des caractères afin de procéder à la séparation de ceux-ci.

La mise en oeuvre de ces deux algorithmes de segmentation est décrite dans les sections qui suivent.

2.3.2 Le Problème du Chevauchement de Caractères

2.3.2.1 Description de l'Algorithme Initial

Toutes les transitions entre l'arrière-plan et le corps même des caractères sont recherchées de haut en bas, et de gauche à droite. La première fois que deux transitions sont détectées sur une même ligne, il est vérifié qu'il est possible d'atteindre le bord gauche ou droit de l'image, au départ de la position de la seconde transition, au moyen d'une suite ininterrompue de pixels d'intensité nulle. Ceux-ci doivent se situer sur la ligne en cours jusqu'à la rencontre d'un pixel d'intensité non nulle, et sur la ligne précédente ensuite (*figure 2.14*). Ceci permet de s'assurer que les deux transitions n'appartiennent pas à un seul et même caractère, contenant une boucle fermée vers le haut. La vérification de l'appartenance des deux transitions à un seul caractère qui comporte une boucle fermée vers le bas ne peut pas encore être effectuée à ce niveau-ci, et est traitée ultérieurement. Lorsque la présence de plus de deux transitions est constatée sur une même ligne, la procédure de détection de boucle fermée vers le haut est effectuée pour chaque transition, à l'exception de la première. Les transitions qui s'avèrent être susceptibles d'appartenir à deux caractères distincts sont marquées et constituent le point de départ de chemins de segmentation possibles.

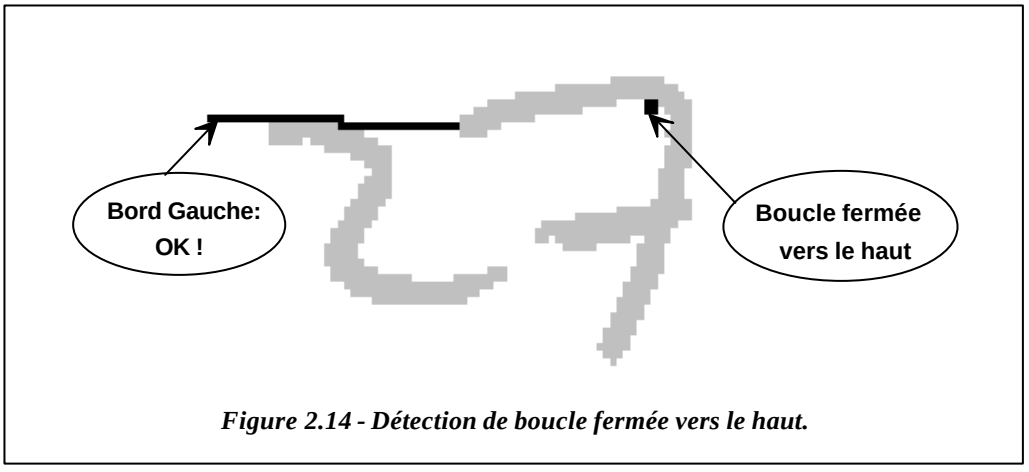


Figure 2.14 - Détection de boucle fermée vers le haut.

La construction des chemins de segmentation se poursuit par la détection de l'existence de transitions multiples sur les lignes ultérieures. Il est alors recherché à quel(s) chemin(s) de segmentation déjà créé(s) chacune de ces transitions, à l'exception de la première, peut être associée. Un pixel de l'arrière plan qui marque le début d'une transition est intégré à tous les chemins de segmentation auxquels il est possible de le relier par une suite continue de pixels d'intensité nulle, situés sur la ligne en cours (*figure 2.15*). Lorsqu'aucune connexion à un des chemins existants au moins n'est réalisable, un nouveau chemin de segmentation est créé, sous la condition décrite précédemment et destinée à exclure une transition due à l'existence dans un caractère d'une boucle fermée vers le haut.

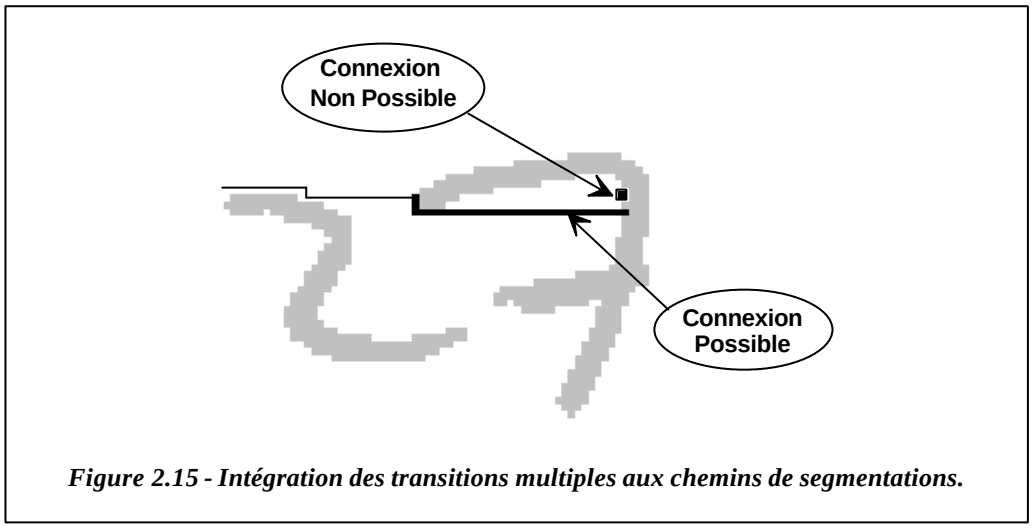


Figure 2.15 - Intégration des transitions multiples aux chemins de segmentations.

Lorsque l'ensemble de l'image a été analysée et que la liste de tous les chemins de segmentation possibles a été dressée, une phase de validation de ces chemins intervient. Le rôle de celle-ci est de détecter et d'écarter les chemins qui aboutissent dans une boucle fermée vers le bas d'un caractère. Un chemin de segmentation ne sera donc valable que s'il est possible, au départ de la position du dernier pixel incorporé à ce chemin, d'atteindre le bord gauche ou droit de l'image au moyen d'une suite ininterrompue de pixels d'intensité nulle. Ces derniers doivent appartenir à la même ligne que le dernier pixel du chemin jusqu'à la rencontre d'un pixel d'intensité non nulle, et à la ligne qui lui est immédiatement consécutive au delà (*figure 2.16*).

Bien que cet algorithme se soit révélé efficace dans de très nombreux cas, certaines situations particulières ont rendu nécessaire l'apport de légères modifications.

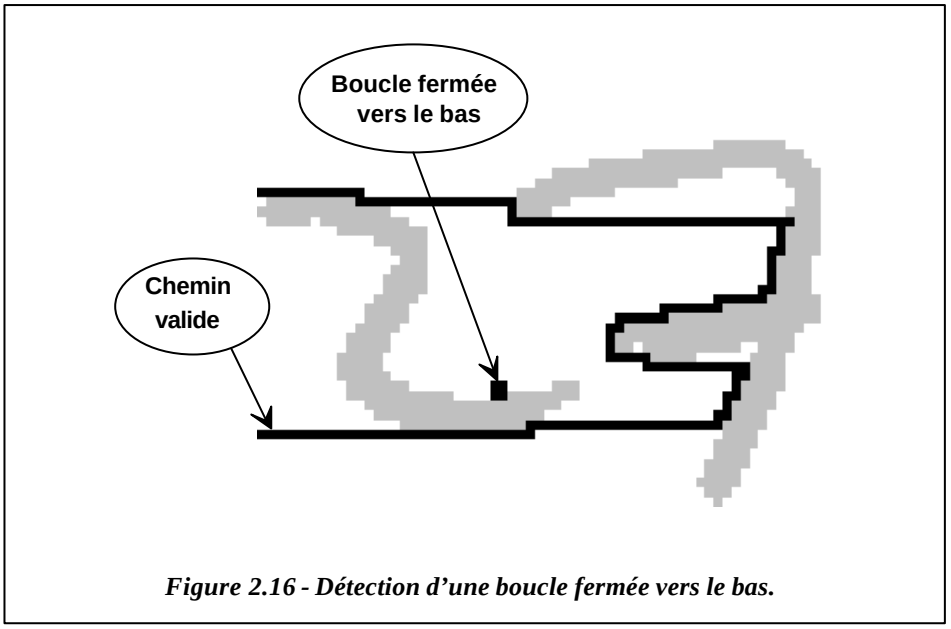


Figure 2.16 - Détection d'une boucle fermée vers le bas.

2.3.2.2 Le Problème des Pixels Parasites

L'image fournie par l'algorithme de segmentation primaire est souvent encore entachée de pixels parasites lorsqu'elle comprend des caractères qui se chevauchent. Dès que des chemins de segmentation sont validés et que les formes contenues dans l'image sont isolées, une procédure de filtrage des pixels parasites identique à celle décrite précédemment doit donc à nouveau être appliquée.

2.3.2.3 Le Défaut d'Encrage Insuffisant

Un caractère qui comporte un défaut d'encrage insuffisant est systématiquement scindé en plusieurs parties par l'algorithme de segmentation secondaire. L'utilisation d'un critère similaire à celui utilisé lors de la segmentation primaire est difficilement applicable ici: la distance la plus courte entre deux caractères qui se chevauchent peut être très faible, et il faut pouvoir les segmenter à coup sûr. Le cas de caractères qui présentent un défaut d'encrage insuffisant reste donc un problème pour la segmentation secondaire.

2.3.2.4 Les Situations Critiques

La *figure 2.17* illustre une situation où le bas d'un caractère est situé plus haut que le sommet d'un second caractère, de sorte qu'il n'apparaît pas de transitions multiples. Cette situation critique peut se produire lorsque l'écart entre les deux caractères est suffisamment faible que pour avoir été masqué par la présence voisine d'autres caractères lors du calcul de

l'histogramme de projection horizontale. Ce problème est résolu de manière simple, en procédant à nouveau à un calcul de l'histogramme de projection horizontale.

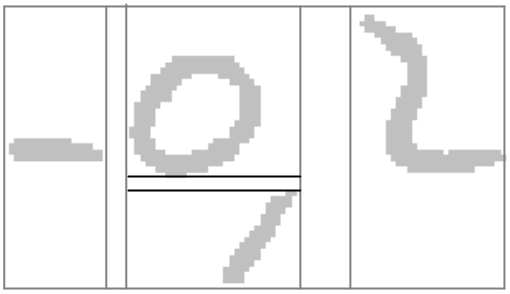


Figure 2.17 - Exemple de cas critique.

Un problème plus délicat se pose lorsque le bas d'un caractère est situé sur la ligne qui est juste au dessus du sommet d'un autre caractère (*figure 2.18*). Dans ce cas, ni l'histogramme de projection horizontale, ni l'algorithme de détection des transitions ne permettent la séparation des deux caractères.

Figure 2.18 - Autre cas critique.

Cette situation se résout en recherchant, pour chaque ligne de l'image et de haut en bas, la position du pixel d'intensité non nulle qui se situe le plus à gauche. Lorsque cette position se trouve à droite de celle déterminée sur la ligne immédiatement précédente, l'algorithme vérifie s'il est possible d'atteindre le bord droit de l'image à partir de cette position et au moyen d'une suite continue de pixels d'intensité nulle, situés sur la ligne précédente (*figure 2.19a*). Cette méthode permet de segmenter les caractères lorsque c'est celui qui est situé le plus à gauche qui est le plus haut. Si aucune segmentation ne peut être effectuée, l'analyse de l'image est reprise au moyen

d'un algorithme similaire qui permet la segmentation lorsque c'est le caractère situé le plus à droite qui est le plus haut (*figure 2.19b*).

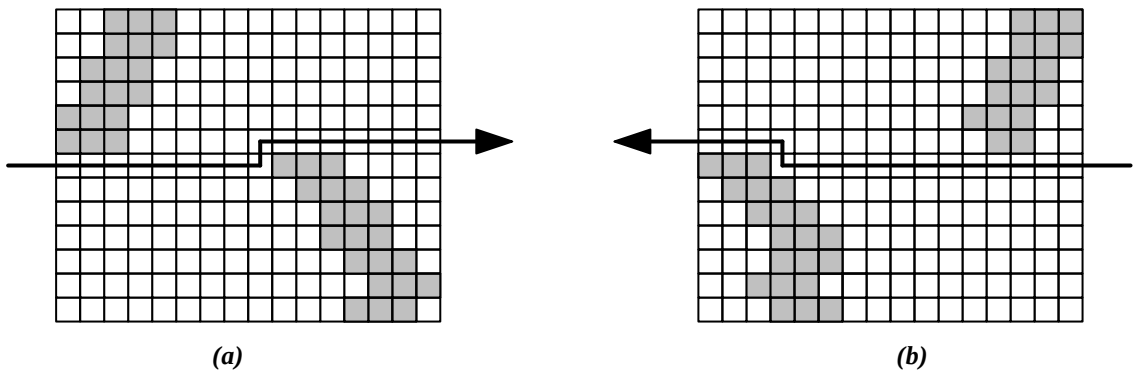


Figure 2.19 - Segmentation secondaire: résolution des cas critiques.

2.3.3 Le Problème des Caractères Connectés

Lorsqu'aucun chemin de segmentation ne peut être déterminé à l'aide de la méthode qui vient d'être décrite, une méthode de séparation de caractères accolés est utilisée. L'algorithme proposé par Fujisawa dans [Fujisawa,92] détermine l'épaisseur du tracé des caractères en recherchant la position du premier pixel d'intensité non nulle dans chaque colonne, à partir du bord supérieur de l'image d'abord, et à partir du bord inférieur ensuite. L'écart, en nombre de pixels, entre les deux positions trouvées est calculé et la segmentation a lieu au droit des minima de l'écart (*figure 2.20*). Bien que la séparation des caractères n'ait pas forcément lieu à l'endroit optimum, ce critère est celui qui permet, en l'absence d'autre information que celle de l'image brute, de résoudre au mieux les problèmes de segmentation de caractères accolés.

Nous avons adopté cette méthode, mais en imposant une contrainte supplémentaire, qui est de ne chercher à segmenter un caractère que lorsque sa largeur atteint une valeur limite. Cette condition est indispensable pour éviter une segmentation systématique, et donc pouvant être erronée, des caractères isolés soumis au processus de segmentation secondaire.

[Fujisawa,92] **H. Fujisawa, Y. Nakano, and K. Kurino**
Segmentation Methods for Character Recognition: From Segmentation to Document Structure Analysis
Proc. of the IEEE, Vol 80, n°7, pp 1079-1091, Juillet 92.

Figure 2.20 - Segmentation de caractères accolés.

Pour des raisons pratiques évidentes, la valeur de la largeur maximale autorisée pour un caractère doit être indépendante de la taille absolue de ce dernier sur le document original. De manière naturelle, c'est donc la valeur de la hauteur de la partie d'image analysée qui est utilisée. Cette valeur s'est révélée suffisante en pratique pour éviter de forcer la scission de caractères plus larges que la moyenne (les lettres «M » et «W », par exemple), tout en minimisant le nombre d'erreurs de segmentation de caractères accolés qui seraient plus étroits.

2.3.4 Conclusions

A la suite des modifications apportées, l'algorithme de segmentation secondaire est devenu extrêmement efficace pour la résolution des problèmes de chevauchement de caractères. Son inconvénient majeur est le temps de calcul très élevé qu'il requiert. Un second inconvénient est que les caractères qui comportent un défaut d'encrage insuffisant sont systématiquement scindés en plusieurs parties. Pour ces deux raisons, il est préférable d'utiliser conjointement les algorithmes de segmentation primaire et secondaire, en recherchant à restreindre l'utilisation de ce dernier aux situations qui le nécessitent.

L'ensemble de la procédure de segmentation secondaire est résumée par l'organigramme de la *figure 2.21*.

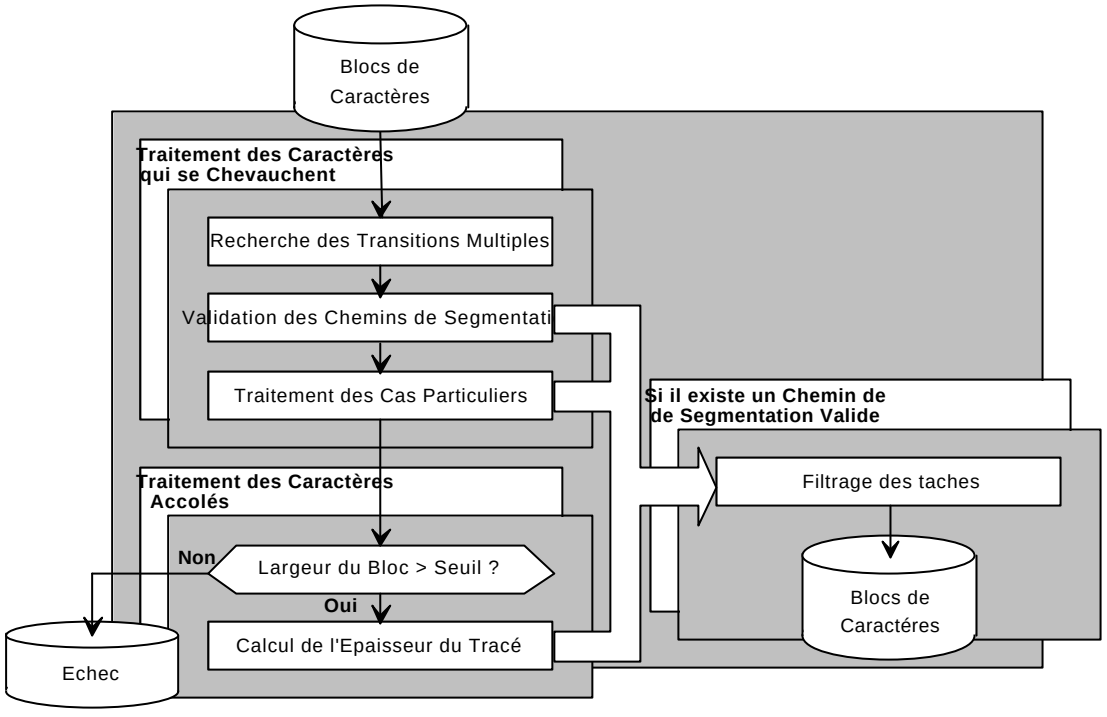


Figure 2.21 - Résumé de la Procédure de Segmentation Secondaire.

2.4 Optimisation de la Procédure de Segmentation

2.4.1 Introduction

L'intervention systématique de l'algorithme de segmentation secondaire implique un alourdissement considérable de la charge de calcul à effectuer en pratique. En outre, certains symboles, tel le signe « = », ou encore des caractères présentant un défaut d'encrage insuffisant, sont alors systématiquement scindés en plusieurs parties, ce qui est extrêmement difficile à corriger dans la suite du système de reconnaissance. Deux méthodes permettant de limiter la fréquence d'utilisation de l'algorithme de segmentation secondaire sont exposées dans les sections qui suivent.

2.4.2 Le Pipeline Direct

Cette méthode se base sur un critère qui permet d'estimer le nombre de caractères présents dans une image. Développé par L. Schumacher dans le cadre d'un travail consacré à la segmentation de l'écriture cursive [Schumacher,93], ce « Baromètre de Contenance » est défini comme suit:

$$\text{Densité de l'image} \times \frac{\text{Hauteur de l'image}}{\text{Largeur de l'image}}$$

où la densité d'une image vaut le rapport entre le nombre de pixels d'intensité non nulle et le nombre total de pixels.

Schumacher a constaté que la valeur de ce baromètre s'accroît lorsque le nombre de caractères contenus dans une image diminue. La *figure 2.22* illustre un relevé expérimental de la dispersion des valeurs du baromètre de contenance, en fonction du nombre de caractères contenus dans une image. Cette grandeur peut être utilisée afin *d'estimer* le nombre de caractères qu'une image est susceptible de contenir, et de juger alors de la nécessité de l'emploi de l'algorithme de segmentation secondaire (*figure 2.23*). La densité et les dimensions de l'image étant des valeurs préalablement déterminées lors de la procédure de segmentation primaire, la méthode est très aisée à mettre en oeuvre.

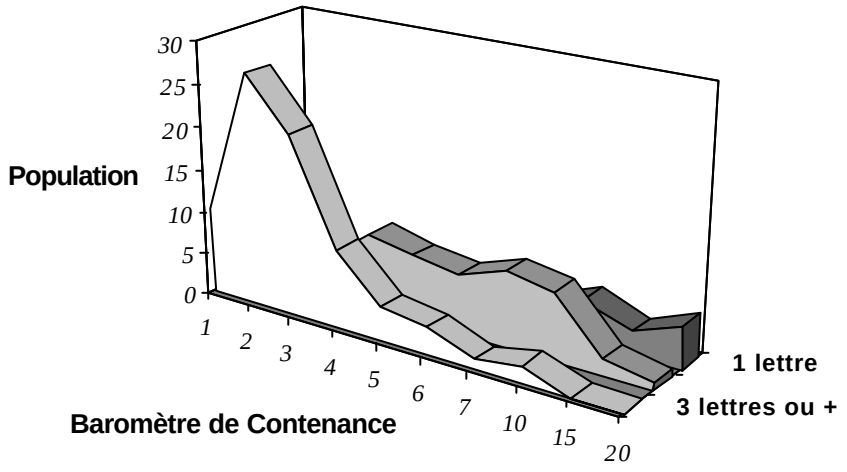


Figure 2.22 - Graphique de dispersion des valeurs du baromètre de contenance.

[Schumacher,93] **L. Schumacher**
Segmentation de l'Ecriture Cursive
Travail de Fin d'Etudes, Faculté polytechnique de Mons, Juin 1993

Les valeurs du baromètre de contenance varient toutefois fortement avec la qualité de l'impression des caractères (modèle gras ou non de crayon utilisé lors de l'écriture, par exemple), et les seuils de décision doivent donc être adaptés en conséquence. En outre, des essais expérimentaux ont montré que, si cette méthode permet d'assurer avec une confiance suffisante qu'un seul caractère est bien présent dans l'image, elle ne permet pas d'affirmer avec certitude l'existence de plusieurs caractères. Pour des raisons de sécurité, l'utilisation de l'algorithme de segmentation secondaire est par conséquent relativement fréquente. A titre d'exemple, dans le cadre d'une application de reconnaissance des 26 lettres latines majuscules, il s'est avéré très peu probable qu'une image contienne plus d'un caractère au delà d'une densité de 35%. Comme 50% des caractères isolés possédaient une densité inférieure à cette valeur, l'algorithme de segmentation secondaire intervenait souvent à mauvais escient. Cette méthode a cependant permis de réduire le taux moyen d'utilisation de ce dernier à 60%.

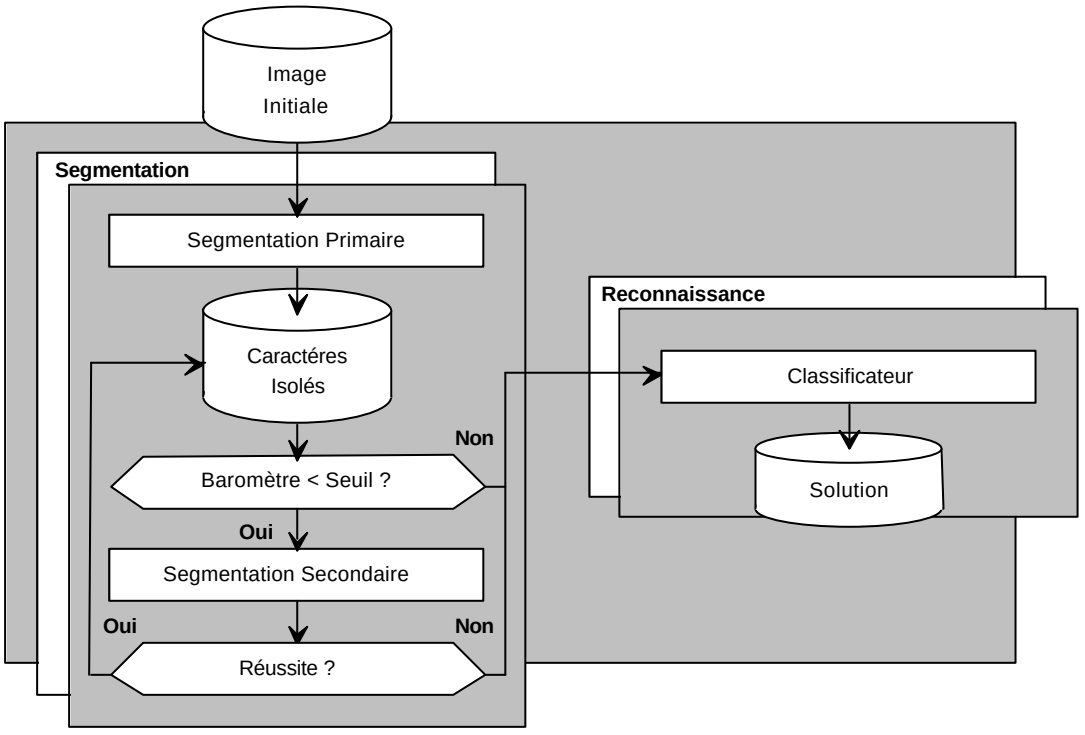


Figure 2.23 - Le Principe du pipeline direct.

2.4.2 Le Pipeline Indirect

Comme nous le verrons au chapitre 3, le principe de tout classificateur est d'attribuer un score à chaque classe, représentatif du degré de ressemblance entre celle-ci et le caractère inconnu. La classe reconnue est alors celle dont le score est le plus élevé. La valeur de celui-ci, ou encore celle de l'écart absolu ou même relatif entre les deux scores les plus élevés, peut être considérée comme un indice de confiance à accorder à la solution proposée par le classificateur. Un écart relatif important, par exemple, signifiera que la classe proposée par le système de reconnaissance se distingue assez nettement des autres. La « confiance » qui peut être accordée à la réponse est donc élevée. Si, au contraire, cet écart relatif est faible, la reconnaissance du caractère par le classificateur aura été délicate. C'est ce qui se produira très souvent dans le cas d'une segmentation primaire inefficace, qui aura concaténé plusieurs caractères en un seul et présenté celui-ci tel quel au système de reconnaissance.

Le pipe-line indirect consiste, sur base de cette constatation, à communiquer les caractères (ou supposés tels) au système de reconnaissance proprement dit dès la fin de la procédure de segmentation primaire. Ce n'est que lorsque l'indice de confiance mesuré aux sorties du classificateur est faible que l'algorithme de segmentation secondaire est sollicité (*figure 2.24*).

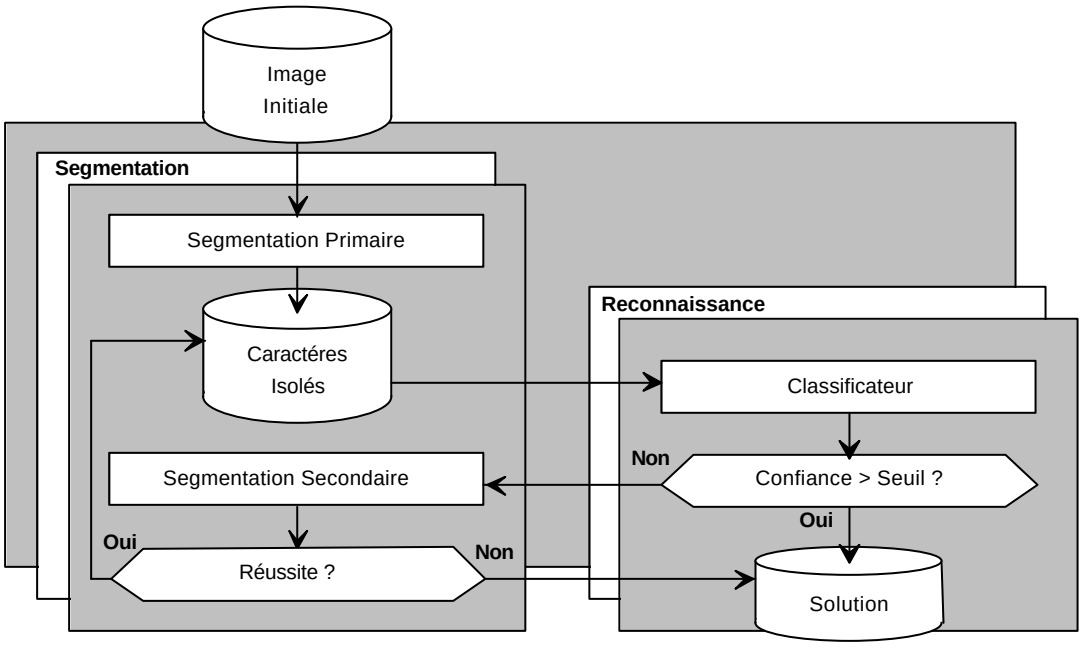


Figure 2.24 - Le Principe du Pipeline Indirect.

La valeur du seuil de confiance qui détermine l'intervention de l'algorithme de segmentation secondaire doit être choisie suffisamment élevée que pour pouvoir détecter un maximum d'erreurs de segmentation primaire. Des critères tels que la qualité d'impression du document original et la fréquence des défauts d'encrage interviennent dans la détermination de la valeur du seuil de confiance à utiliser. Celle-ci dépend également des performances du système de classification, et ce non seulement en termes de qualité, mais aussi en termes de rapidité. Le temps passé à tenter de reconnaître un ensemble de caractères concaténés est en effet perdu lorsque l'intervention de l'algorithme de segmentation secondaire est décidée.

2.4.4 Discussion: Choix d'une Méthode

Le choix entre les méthodes du pipe-line direct et du pipe-line indirect résulte essentiellement de la qualité des documents à traiter. Le pipe-line direct est préférentiellement utilisé lorsque l'impression des documents originaux est de bonne qualité et présente peu de défauts d'encrage insuffisant, et surtout lorsque l'on désire s'assurer de segmenter à coup sûr toute image contenant plus d'un caractère. Le pipe-line indirect, lui, est utilisé lorsque les documents sont susceptibles de présenter des défauts d'encrage et que l'on désire éviter de scinder un caractère en plusieurs parties, ce qui rendrait sa reconnaissance impossible. Le pipe-line indirect est également utilisé lorsque l'on désire privilégier la vitesse moyenne d'exécution de la procédure de reconnaissance.

L'efficacité du classificateur que nous avons développé, ainsi que la fréquence relativement élevée de la présence de défauts d'encrage insuffisant constatée au cours des essais expérimentaux nous ont conduits à adopter la méthode du pipe-line indirect. En outre, la pertinence de l'indice de confiance que le classificateur s'est révélé être à même de fournir, a permis de n'utiliser qu'à bon escient l'algorithme de segmentation secondaire, minimisant ainsi la durée moyenne de l'ensemble de la procédure de reconnaissance.

2.5 Résumé

Deux algorithmes pour la segmentation des caractères ont été présentés, et leur comportement vis-à-vis de problèmes typiques ont été décrits. La procédure finale de segmentation n'utilise systématiquement que l'algorithme de segmentation primaire, relativement facile à mettre en oeuvre et rapide à l'exécution. L'algorithme de segmentation secondaire, quant à lui, permet de segmenter de manière très efficace des caractères qui se chevauchent, problème qui ne peut être résolu à l'aide du premier algorithme. Il permet également d'effectuer une séparation de caractères accolés, bien que celle-ci ne soit pas toujours pertinente. Plus exigeant

en termes de ressources de calcul, l'algorithme de segmentation secondaire voit son utilisation conditionnée par la valeur d'un indice de confiance fourni par le système de classification proprement dit. Cette méthode a permis de minimiser le temps moyen nécessaire à l'ensemble du processus de reconnaissance.

Chapitre 3

La Reconnaissance Statistique de Formes

3.1 Notion de Classificateur

Soit la représentation d'un objet quelconque au moyen d'un vecteur de caractéristiques $X = [x_1\, x_2 \cdots x_d]^T$. Tous les vecteurs qui représentent l'ensemble des objets peuvent être positionnés dans l'espace Euclidien R^d, où ils correspondent chacun à un point. Ceux-ci peuvent alors être regroupés en amas, chacun de ces amas étant associé à une classe particulière. Un exemple pour un problème à deux classes est illustré à la *figure 3.1*.

Le rôle d'un classificateur est de déterminer, parmi un ensemble fini de classes, à laquelle appartient un objet donné.

Un classificateur doit être capable de modéliser au mieux les frontières qui séparent les classes les unes des autres. Cette modélisation fait appel à la notion de *fonction discriminante,* qui permet d'exprimer le critère de classification de la manière suivante:

« Assigner la classe ω_i à l'objet représenté par le vecteur X si, et seulement si, la valeur de la fonction discriminante de la classe ω_i est supérieure à celle de la fonction discriminante de n'importe quelle autre classe ω_j ».

Ou encore, sous forme mathématique:

$$X \in \omega_i \quad \Leftrightarrow \quad \Phi_i(X) \geq \Phi_j(X) \quad \forall j = 1,2,...,C; j \neq i. \text{ (3.1)}$$

où $\Phi_i(X)$ est appelé *fonction discriminante* de la classe ω_i, et C est le nombre total de classes.

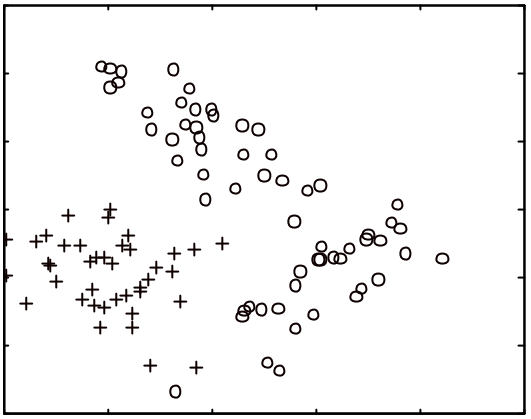


Figure 3.1 - Représentation d'objets appartenant à deux classes distinctes, dans un espace à deux dimensions.

Soit une fonction $\lambda(i|j)$, qui désigne le coût encouru lorsque la classe ω_i est assignée à un objet appartenant à la classe ω_j. Le classificateur optimal est celui qui minimise le coût total obtenu, étant donné une fonction de coût particulière. Une telle fonction peut être définie de manière élémentaire sur base de l'opérateur delta de Krönecker:

$$\lambda(i|j) = 1 - \delta_{ij} = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases} \text{ (3.2)}$$

Cette définition signifie que les classifications correctes n'introduisent aucune perte, et que les classifications incorrectes introduisent chacune un coût égal, de valeur unitaire. Dans ce cas, le coût global obtenu sur un ensemble fini d'objets vaut simplement le nombre d'erreurs de classification. Le classificateur optimal, également appelé « Bayesien », est alors celui qui minimise la probabilité d'erreur, c'est-à-dire la probabilité qu'une classe incorrecte soit assignée à un objet. Le critère de classification devient ainsi [Fukunaga,90]:

[Fukunaga, 90] **K. Fukunaga**
Introduction to Statistical Pattern Recognition
Academic Press, San Diego, 1990

$$X \in \omega_i \quad \Leftrightarrow \quad p(\omega_i|X) > p(\omega_j|X) \quad \forall j = 1,2,...,C;\ j \neq i. \quad (3.3)$$

où $p(\omega_i|X)$ est la probabilité *à posteriori* de la classe ω_i. La classe attribuée à l'objet représentée par le vecteur X est alors celle dont la probabilité étant donné X est supérieure à la probabilité de n'importe quelle autre classe, étant donné X.

Le calcul exact des probabilités *à posteriori* est cependant rarement possible, et des modèles de classificateurs ont été développés sur base d'autres fonctions discriminantes que la probabilité *à posteriori*. Ces classificateurs peuvent être séparés en trois catégories distinctes:

- les classificateurs paramétriques, qui sont entièrement définis par un ensemble fini de paramètres qu'il suffit de calculer,
- les classificateurs non paramétriques, qui ne dépendent d'aucun paramètre en particulier,
- les classificateurs dits « neuronaux », qui intègrent des fonctions discriminantes à la suite d'un apprentissage par l'exemple.

3.2 Les Classificateurs Paramétriques

3.2.1 Le Classificateur Euclidien

Il s'agit probablement de l'un des plus simples classificateurs qui puissent être conçus. La classe dont le vecteur de caractéristiques moyen est le plus proche, au sens de la distance Euclidienne, du vecteur de caractéristiques de l'objet à classifier est assignée à ce dernier. Les fonctions discriminantes utilisées sont donc de la forme suivante:

$$\Phi_i(X) = -\frac{1}{2}(X - \mathbf{M}_i)^T (X - \mathbf{M}_i) \quad (3.4)$$

où $\mathbf{M}_i = E\{X|\omega_i\}$ est le vecteur de caractéristiques moyen des éléments qui appartiennent à la classe ω_i, $E\{\cdot\}$ désignant l'opérateur d'espérance mathématique, et $(\cdot)^T$ celui de transposition.

Le terme quadratique $X^T X$ est indépendant de la classe de l'objet, et les fonctions discriminantes peuvent également s'écrire:

$$\Phi_i(X) = \mathrm{M}_i^T X - \frac{1}{2}\mathrm{M}_i^T \mathrm{M}_i \quad (3.5)$$

Les frontières qui séparent les classes dans l'espace R^d sont ici linéaires. Un exemple en est donné à la *figure 3.2*.

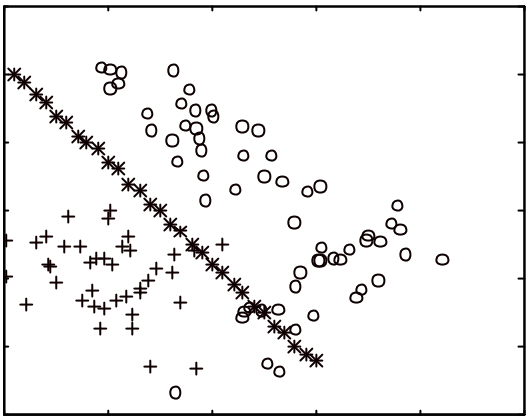


Figure 3.2 - Frontière fournie par le classificateur Euclidien dans le cas d'un problème à deux classes.

En pratique, les vecteurs de caractéristiques moyens ne sont pas disponibles, et doivent être estimés à partir d'un ensemble fini de prototypes de chaque classe:

$$\hat{\mathrm{M}}_i = \frac{1}{N_i} \sum_{X_k \in \omega_i} X_k \quad (3.6)$$

où N_i est le nombre total d'objets de classe ω_i qui sont à disposition, et X_k les vecteurs de caractéristiques qui représentent ces objets.

3.2.2 Le Classificateur Quadratique

Comme le nom l'indique, les frontières de décision fournies par ce modèle de classificateur sont quadratiques (*figure 3.3*). Les fonctions discriminantes s'expriment:

$$\Phi_i(X) = -\frac{1}{2}(X - \mathrm{M}_i)^T \Sigma_i^{-1}(X - \mathrm{M}_i) \quad (3.7)$$

où $\Sigma_i = E\left\{(X - \mathrm{M}_i)(X - \mathrm{M}_i)^T \middle| \omega_i\right\}$ est la matrice de covariance des vecteurs de caractéristiques de classe ω_i .

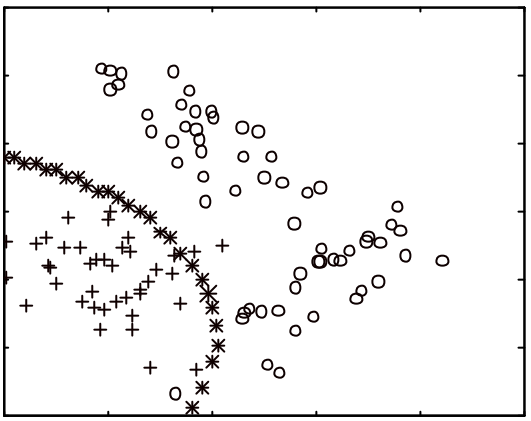


Figure 3.3 - Frontière obtenue à l'aide du classificateur quadratique.

Tout comme les vecteurs de caractéristiques moyens de chaque classe, les matrices de covariance ne peuvent qu'être estimées à partir des objets disponibles. Une estimation non biaisée en est donnée par [Lebart,82]:

$$\hat{\Sigma}_i = \frac{1}{N_i - 1} \sum_{X_k \in \omega_i} \left(X_k - \hat{\mathrm{M}}_i\right)\left(X_k - \hat{\mathrm{M}}_i\right)^T \quad (3.8)$$

[Lebart,82] **L. Lebart, A. Morineau, J.-P. Fénelon**
traitement des données statistiques
Ed. Dunod,Paris, 1982

Dans le cas particulier où les composantes des vecteurs de caractéristiques ne sont pas corrélées entre elles, les matrices de covariances expérimentales sont diagonales. L'expression des fonctions discriminantes se réduit alors à:

$$\Phi_i(x) = -\frac{1}{2}\sum_{j=1}^{d}\frac{\left(x_j - \hat{\mu}_{ij}\right)^2}{\hat{\sigma}_{ij}^2} \quad (3.9)$$

où $\hat{\mu}_{ij}$ et $\hat{\sigma}_{ij}^2$ représentent respectivement la moyenne et la variance expérimentales de la $j^{ème}$ composante du vecteur X, calculées sur les éléments de la classe ω_i.

3.2.3 Le Classificateur Gaussien

Les fonctions discriminantes utilisées ici sont basées sur une estimation paramétrique des fonctions de répartition des vecteurs de caractéristiques. Ce classificateur suppose que les éléments de chaque classe possèdent une distribution Gaussienne multivariable. Dans la mesure où cette hypothèse s'avère exacte, le classificateur Gaussien permet d'obtenir les frontières optimales de décision de Bayes. En effet, le théorème de Bayes permet de calculer les probabilités *à posteriori* $p(\omega_i|X)$ à partir des probabilités *à priori* $p(\omega_i)$ et des fonctions de répartition (ou vraisemblances) $p(X|\omega_i)$ selon:

$$p(\omega_i|X) = \frac{p(\omega_i)p(X|\omega_i)}{p(X)}, \quad (3.10)$$

et la règle de décision optimale (3.3) peut dès lors être reformulée comme suit:

$$X \in \omega_i \quad \Leftrightarrow \quad p(X|\omega_i)p(\omega_i) > p(X|\omega_j)p(\omega_j) \quad \forall j = 1,2,...C;\ j \neq i. \quad (3.11)$$

Lorsque les vecteurs de caractéristiques suivent une distribution Gaussienne, les vraisemblances sont estimées par:

$$p(X|\omega_i) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_i|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(X-\mathrm{M}_i)^T\Sigma_i^{-1}(X-\mathrm{M}_i)\right) \quad (3.12)$$

Le terme $(2\pi)^{-\frac{d}{2}}$, constant, peut être omis pour la classification. En prenant le logarithme, les fonctions discriminantes du classificateur Gaussien s'écrivent:

$$\Phi_i(X) = -\frac{1}{2}(X - \mathrm{M}_i)^T \Sigma_i^{-1}(X - \mathrm{M}_i) - \frac{1}{2}|\Sigma_i| + \ln(p(\omega_i)) \quad (3.13)$$

Les fonctions discriminantes du classificateur Gaussien ne diffèrent de celles du classificateur quadratique que par un biais, spécifique à chaque classe. Les frontières de décision entre les classes sont ici aussi de formes quadratiques (*figure 3.4*). En pratique, les probabilités *a priori* $p(\omega_i)$, les vecteurs de caractéristiques moyens M_i, et les matrices de covariances Σ_i, sont remplacés par leurs estimations expérimentales respectives $\hat{p}(\omega_i)$, $\hat{\mathrm{M}}_i$, et $\hat{\Sigma}_i$.

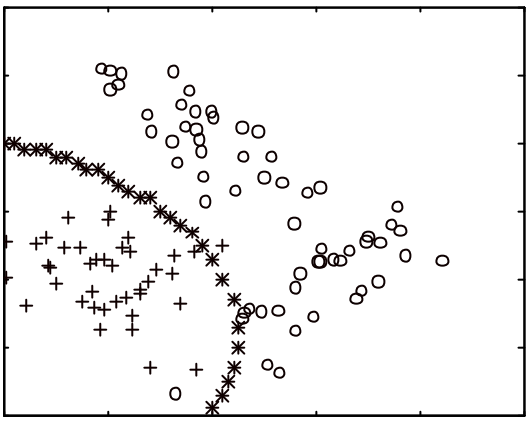


Figure 3.4 - Frontière générée par le classificateur Gaussien.

Lorsque les composantes des vecteurs de caractéristiques sont non corrélées, les matrices de covariances sont diagonales. L'expression (3.12) se réduit alors à un produit de *d* Gaussiennes indépendantes:

$$p(X|\omega_i) = \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} \exp\left(-\frac{(x_j - \mu_{ij})^2}{2\sigma_{ij}^2} \right) \quad (3.14)$$

Les fonctions discriminantes du classificateur Gaussien deviennent ainsi:

$$\Phi_i(X) = -\frac{1}{2}\sum_{j=1}^{d}\frac{\left(x_j - \mu_{ij}\right)^2}{\sigma_{ij}^2} - \frac{1}{2}\sum_{j=1}^{d}\sigma_{ij}^2 + \ln\left(p(\omega_i)\right) \quad (3.15)$$

L'hypothèse de non-corrélation des composantes des vecteurs de caractéristiques permet donc de simplifier fortement les calculs.

Une variante du classificateur Gaussien consiste à estimer la fonction de répartition par une somme pondérée de plusieurs Gaussiennes:

$$p(X|\omega_i) = \sum_{q=1}^{Q}\alpha_{qi}\frac{1}{(2\pi)^{\frac{d}{2}}\left|\sum_{qi}\right|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(X - \mathrm{M}_{qi}\right)^T\Sigma_{qi}^{-1}\left(X - \mathrm{M}_{qi}\right)\right) \quad (3.16)$$

où Q est le nombre total de Gaussiennes, et α_{qi} des coefficients de pondération devant être déterminés à partir des données, sous les contraintes:

$$\sum_{q=1}^{Q}\alpha_{qi} = 1, \forall i = 1,\ldots,C; \alpha_{qi} \geq 0. \quad (3.17)$$

Ce classificateur offre ainsi des possibilités plus complexes d'estimation des vraisemblances. La recherche de la valeur des paramètres (importance relative α_{qi} de chacune des Gaussiennes ainsi que leur vecteur de moyennes et matrice de covariance respectifs) doit cependant s'effectuer de manière itérative, dans la mesure où chaque Gaussienne n'est pas associée *à priori* à un groupe bien déterminé de vecteurs de caractéristiques.

3.3 Les Classificateurs Non Paramétriques

3.3.1 La Méthode du Plus Proche Voisin

Ce classificateur est une extrapolation du classificateur Euclidien décrit précédemment. Au lieu d'utiliser le vecteur de caractéristiques moyen M_i comme unique prototype d'une classe, la méthode du plus proche voisin fait intervenir tous les exemplaires des vecteurs de caractéristiques disponibles. La distance Euclidienne entre chacun de ceux-ci et celui de l'objet à classifier est

calculée, et la classe assignée à l'objet est celle du prototype le plus proche de celui-ci. Les fonctions discriminantes sont donc de la forme:

$$\Phi_i(X) = -\min_{X_k \in \omega_i} \frac{1}{2}(X - X_k)^T (X - X_k) \quad (3.18)$$

Le terme quadratique pouvant être omis, ces fonctions se réduisent à:

$$\Phi_i(X) = \min_{X_k \in \omega_i} \left(X_k^T X - \frac{1}{2} X_k^T X_k \right) \quad (3.19)$$

Les frontières de décision entre classes sont linéaires et constituées de nombreux petits polygones convexes, chacun contenant un seul prototype d'une seule classe. Chaque classe est alors délimitée par un polygone très complexe, qui n'est pas nécessairement convexe, ni même d'une seule pièce (*figure 3.5*). Ce classificateur permet ainsi d'établir des frontières de décision relativement complexes, lorsque suffisamment d'exemplaires de chaque classe sont disponibles. Ces performances sont toutefois atteintes au détriment du volume de calcul à effectuer et de la quantité de mémoire nécessaire, lesquels deviennent alors prohibitifs.

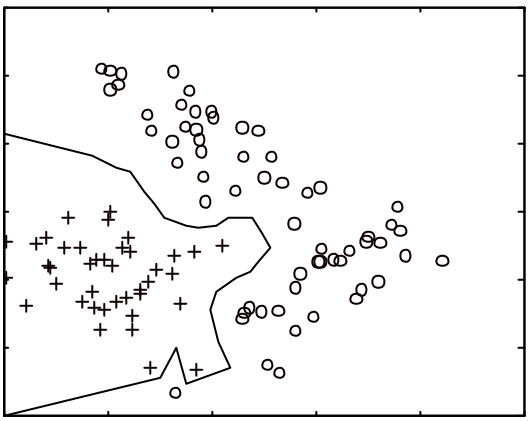


Figure 3.5 - Frontières fournies par le classificateur du Plus Proche Voisin.

Cover et Hart ont montré qu'il existe une relation entre le taux d'erreur minimal de Bayes et le taux d'erreur obtenu à l'aide de la règle de décision du Plus Proche Voisin. Cette relation n'est cependant valable qu'asymptotiquement, en considérant que le nombre de prototypes disponibles

pour chaque classe tend vers l'infini [Cover & Hart, 67]. Pour un problème à C classes, elle se définit comme suit:

$$P_e^* \le P_e \le P_e^*\left(2 - \frac{C}{C-1}P_e^*\right) \quad (3.20)$$

où P_e^* est le taux d'erreur de classification de Bayes, et P_e le taux d'erreur obtenu asymptotiquement par la règle du Plus Proche Voisin. En pratique, cette grandeur ne peut qu'être estimée pour un nombre de prototypes fini *N*, qui doit être suffisamment grand pour que l'estimation soit valable.

L'expression (3.20) permet d'obtenir une estimation de la borne inférieure de la probabilité d'erreur de Bayes:

$$P_e^* \ge \frac{C-1}{C}\left(1 - \sqrt{1 - \frac{C}{C-1}P_e}\right) \quad (3.21)$$

Ce résultat est très important, car il permet de comparer les performances d'un classificateur à une valeur mathématique théorique qui est une borne inférieure du taux d'erreur de Bayes.

3.3.2 La Méthode des k plus Proches Voisins

Un des inconvénients majeurs de la méthode du Plus Proche Voisin est que celle-ci présente une sensibilité élevée aux abords des frontières entre classes. Le plus proche voisin d'un objet peut être d'une classe incorrecte, alors que la majorité de ses voisins ne le sont pas. Afin de contrer cet effet, la classe assignée à un objet peut être celle qui est la plus représentée parmi les *k* plus proches prototypes trouvés. La méthode porte dans ce cas le nom de « *k* Plus Proches Voisins ». La fonction discriminante d'une classe est alors simplement le nombre de prototypes de cette classe qui se situent parmi les *k* plus proches voisins de l'objet à classifier:

$$\Phi_i(X) = \sum_{Xj \in \omega_i} \left| X_j \in \Psi_k(X) \right| \quad (3.22)$$

[Cover & Hart,67] **T. M. Cover & P. E. Hart**
Nearest Neighbor Pattern Classification
IEEE Transactions on Informations Theory, Vol. 13, n°1, janvier 1967

3.4 Les Classificateurs Neuronaux

3.4.1 Introduction

La reconnaissance du fait que le cerveau fonctionne de manière entièrement différente de celle d'un ordinateur conventionnel a joué un rôle très important dans le développement des réseaux de neurones artificiels. Les travaux effectués pour essayer de comprendre le comportement du cerveau humain ont menés à représenter celui-ci par un ensemble de composants structurels appelés neurones, massivement interconnectés entre eux. Le cerveau humain en contiendrait plusieurs centaines de milliards, et chacun de ceux-ci serait, en moyenne, connecté à dix mille autres. Le cerveau est capable d'organiser ces neurones, selon un assemblage complexe, non-linéaire et extrêmement parallèle, de manière à pouvoir accomplir des tâches très élaborées. Par exemple, n'importe qui est capable de reconnaître des visages, alors que c'est là une tâche quasiment impossible pour un ordinateur classique. C'est la tentative de donner à l'ordinateur les qualités de perception du cerveau humain qui a conduit à une modélisation électrique de celui-ci. C'est cette modélisation que tentent de réaliser les réseaux de neurones artificiels.

Haykin en propose la définition suivante [Haykin,94]:

« Un réseau de neurones est un processus distribué de manière massivement parallèle, qui a une propension naturelle à mémoriser des connaissances de façon expérimentale et de les rendre disponibles pour utilisation. Il ressemble au cerveau en deux points:

1. *la connaissance est acquise au travers d'un processus d'apprentissage;*
2. *les poids des connections entre les neurones sont utilisés pour mémoriser la connaissance. »*

C'est sur base de cette définition que repose l'élaboration des réseaux de neurones artificiels.

[Haykin,94] **S. Haykin**
Neural Networks - A comprehensive Foundation
Macmillan College Publishing Company, New York, 1994

Les sections qui suivent présentent plusieurs modèles de réseaux de neurones artificiels, ainsi qu'une analyse de leurs aptitudes à agir en tant que classificateur.

3.4.2 Du Neurone Biologique au Neurone Formel

Le neurone biologique est composé de quatre parties distinctes (*figure 3.6*) :

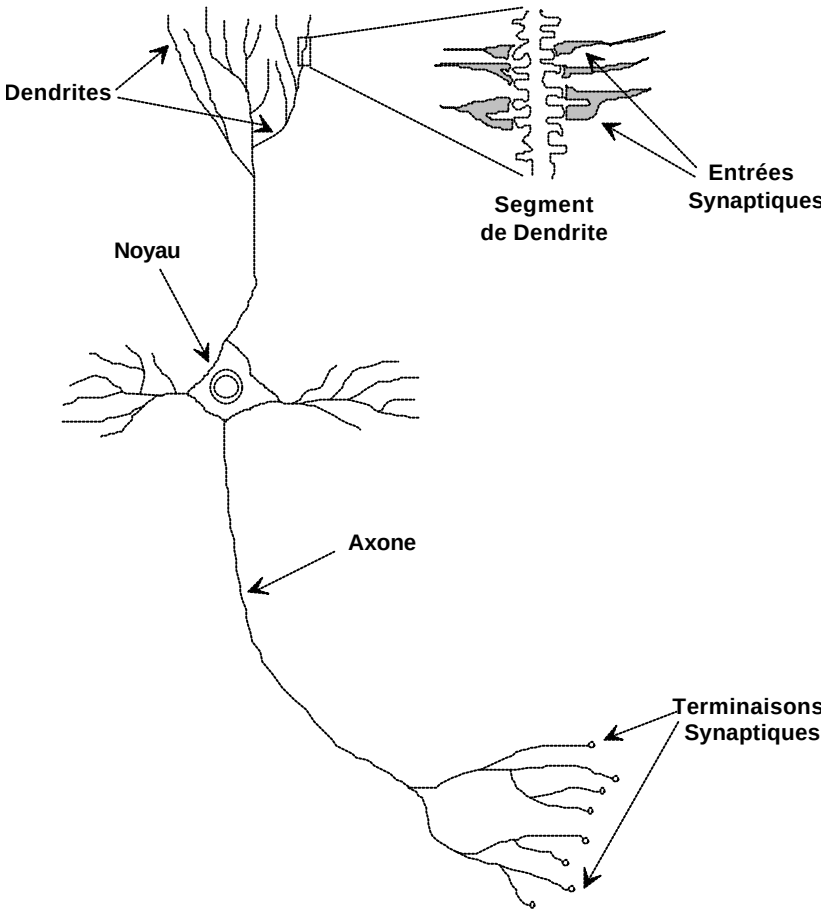


figure 3.6 - Schéma d'un neurone biologique.

- ***le corps cellulaire***, qui contient le noyau de la cellule nerveuse; c'est en cet endroit que prend naissance l'influx nerveux, qui représente l'état d'activité du neurone;
- ***les dendrites***, ramifications tubulaires courtes formant une espèce d'arborescence autour du corps cellulaire; ce sont les entrées principales du neurone, qui captent l'information venant d'autres neurones;

- ***l'axone***, longue fibre nerveuse qui se ramifie à son extrémité; c'est la sortie du neurone et le support de l'information vers les autres neurones;
- ***la synapse***, qui communique l'information, en la pondérant par un *poids synaptique*, à un autre neurone; elle est essentielle dans le fonctionnement du système nerveux.

Chaque neurone réalise une opération très simple, qui est en fait une somme pondérée de ses entrées. Le résultat est comparé à un seuil et le neurone devient excité si ce seuil est dépassé. L'information contenue dans le cerveau est représentée par les poids donnés aux entrées de chaque neurone. Du fait du grand nombre de neurones et de leurs interconnections, ce système possède une propriété de tolérance aux fautes. Ainsi, la défectuosité d'un élément mémoire (neurone) n'entraînera aucune perte réelle d'information, mais seulement une faible dégradation en qualité de toute l'information contenue dans le système. C'est pourquoi nous pouvons reconnaître le visage d'une personne, même si celle-ci a vieilli, par exemple.

La première étude systématique du neurone artificiel est due au neuropsychiatre McCulloch et au logicien Pitts [Lippmann,87] qui, s'inspirant de leurs travaux sur les neurones biologiques, proposèrent en 1943 le modèle suivant (*figure 3.7*):

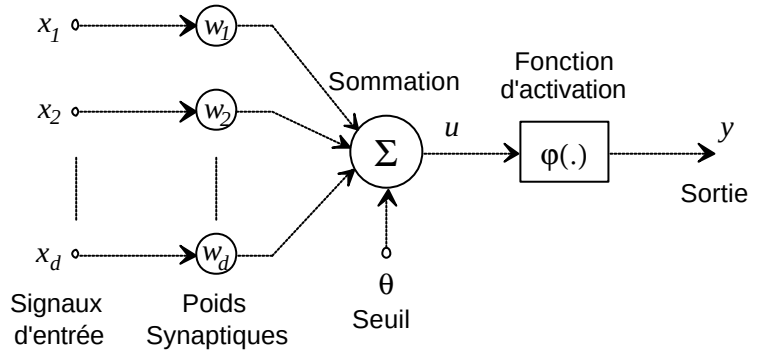


Figure 3.7 - Le modèle de neurone formel.

Ce neurone formel est un processeur élémentaire qui réalise une somme pondérée des signaux qui lui parviennent. La valeur de cette sommation est comparée à un seuil et la sortie du neurone est une fonction ***non linéaire*** du résultat:

[Lippmann,87] **R. Lippmann**
An Introduction To Computing with Neural Nets
IEEE ASSP Magazine, volume 4, n° 2, pp 4-22, Avril 1987

$$u = \sum_{j=1}^{d} w_j x_j - q \quad (3.23)$$

$$y = j(u) \quad (3.24)$$

En prenant la convention de noter par $\underline{X} = [-1\, x_1 \cdots x_d]^T$ le vecteur d'entrées augmenté, et par $\underline{W} = [\theta\ w_1 \cdots w_d]^T$ le vecteur de poids augmenté, il vient:

$$y = \varphi\left(\underline{W}^T \underline{X}\right) \quad (3.25)$$

Dans le modèle original de McCulloch et Pitts, la non-linéarité était assurée par la fonction seuil de Heaviside, définie par:

$$\mathbf{H}(u) \overset{\Delta}{=} \begin{cases} 1 & \textbf{si } u > 0 \\ 0 & \textbf{sinon} \end{cases} \quad (3.26)$$

En place de la fonction de Heaviside, il est également possible de choisir la fonction de signe (*figure 3.8*), définie comme suit:

$$\mathrm{sgn}(u) \overset{\Delta}{=} \begin{cases} 1 & \textbf{si } u > 0 \\ -1 & \textbf{sinon} \end{cases} \quad (3.27)$$

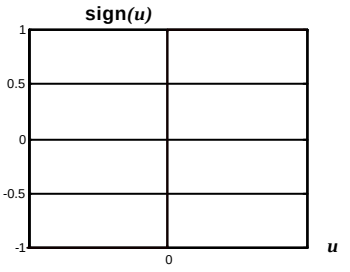


Figure 3.8 - fonction de signe.

3.4.3 La Carte Auto-Organisatrice

3.4.3.1 Architecture du Réseau

La carte auto-organisatrice est un ensemble structuré d'unités de traitement (neurones), qui sont disposées en une seule couche de laquelle émane une topologie définie par une notion de voisinage des cellules. Les principes de base de fonctionnement de ce réseau, inspirés des constatations biologiques relatives à *l'ordonnancement des neurones*, furent introduits en 1981 par Teuvo Kohonen [Kohonen,90].

La *figure 3.9* illustre un tel réseau dans sa configuration la plus courante, le cas bidimensionnel. L'entrée du réseau, unique et commune à tous les neurones, est le vecteur de caractéristiques *X* qui représente un objet. A chaque neurone sont associés des coordonnées (x,y) indiquant sa position sur la carte, ainsi qu'un vecteur de *poids synaptiques*, appelé ainsi par analogie avec les synapses rencontrées dans le cerveau.

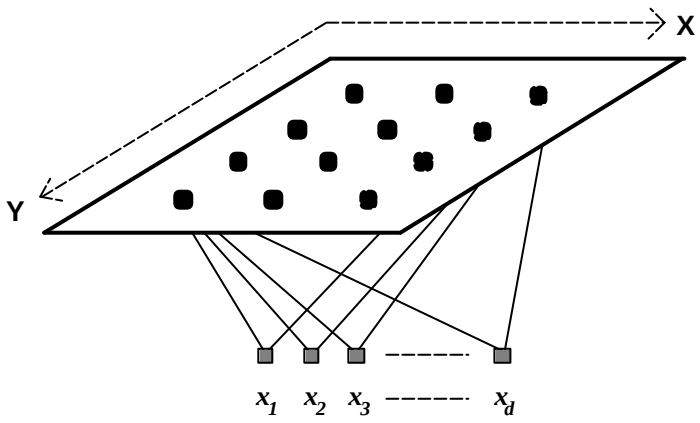


Figure 3.9 - Carte auto-organisatrice à deux dimensions.

3.4.3.2 La Phase d'Apprentissage

Tout réseau de neurones requiert une phase d'apprentissage, laquelle a pour but d'adapter les valeurs des poids synaptiques des cellules afin que le réseau soit à même de remplir la tâche que l'on désire lui attribuer en phase d'exploitation. Dans le cas de la carte auto-organisatrice,

[Kohonen,90] **T. Kohonen**
The Self-Organising Map
proc. of the IEEE, vol 78, n°9, pp 1464-1480, Septembre 1990

l'apprentissage est basé sur deux constatations biologiques essentielles, à savoir que, dans le cerveau:

(1) chaque cellule nerveuse correspond à un stimulus spécifique;
(2) il existe une région d'intense activité autour de la cellule la plus stimulée.

Le modèle d'entraînement de la carte consistera donc à :

(1) sélectionner le neurone correspondant le mieux à un signal d'entrée donné;
(2) induire dans un voisinage de l'élu une région d'intense activité.

Dans un premier temps, l'apprentissage est *non-supervisé* et consiste à répéter les étapes suivantes:

1. présenter un objet X à l'entrée du réseau, sans préciser la classe à laquelle il appartient;
2. rechercher le neurone q^* dont le vecteur de poids est le plus proche, au sens de la distance Euclidienne, du vecteur d'entrée:

$$q^*: \quad \min_q \frac{1}{2}\left(X - W_q\right)^T\left(X - W_q\right) \quad (3.28)$$

3. adapter le vecteur de poids de ce neurone *ainsi que ceux de ses voisins topologiques* de manière à ce qu'ils se rapprochent davantage du vecteur d'entrée:

$$\begin{aligned} &W_q(\tau+1) = W_q(\tau) + \eta\left(X - W_q(\tau)\right) && \text{si} \quad q \in \mathrm{V}\left(q^*\right) \quad (3.29)\\ &W_q(\tau+1) = W_q(\tau) && \text{sinon.} \end{aligned}$$

où $\mathrm{V}\left(q^*\right)$ désigne le voisinage du neurone q^* dans lequel les vecteurs de poids des cellules sont adaptés, et η, $(\eta > 0)$, est le *taux d'apprentissage*.

Le terme quadratique de la formule (3.28) peut être omis car il est constant pour tous les neurones. La recherche de celui dont le vecteur de poids synaptiques est le plus proche du vecteur d'entrée s'effectue alors selon:

$$q^*: \quad \max_q \left(W_q^T X - \frac{1}{2} W_q^T W_q \right) \quad (3.30)$$

L'étape 2 consiste donc simplement à rechercher le neurone dont la valeur de sortie est la plus élevée, sous la condition d'imposer que le seuil de chaque neurone soit tel que:

$$w_{q0} = \frac{1}{2} W_q^T W_q \quad (3.31)$$

Il vient ainsi:

$$q^*: \quad \max_q \left(\underline{W}_q^T \underline{X} \right) \quad (3.32)$$

Lorsque les poids d'un neurone sont modifiés, il convient d'adapter le seuil de manière à ce que la condition (3.31) soit toujours vérifiée.

Plusieurs méthodes peuvent être envisagées pour réaliser l'adaptation des poids synaptiques. Celle qui s'est avérée la plus performante pour organiser des cartes destinées à des tâches de reconnaissance [Knagenhjelm,90] porte le nom de *règle d'adaptation centrale*. Le taux d'apprentissage η de la formule (3.29) décroît à la fois avec le temps et avec la distance par rapport au neurone élu q^* (*figure 3.10*). La notion de voisinage est donc implicitement contenue dans la définition du taux d'apprentissage. Le fait de débuter l'adaptation des poids avec un taux d'apprentissage plus élevé permet d'obtenir assez rapidement une organisation rudimentaire générale des vecteurs de poids, tandis que sa décroissance progressive permet ensuite d'affiner la résolution de la carte. Les valeurs du taux d'apprentissage ainsi que celles de sa vitesse de décroissance dépendent du problème à traiter et doivent être déterminées empiriquement.

La règle d'adaptation centrale permet de résumer les équations (3.29) à une seule expression:

$$W_q(\tau+1) = W_q(\tau) + \eta(\tau, r)\left(X - W_q(\tau)\right) \quad \forall q \quad (3.33)$$

où *r* désigne la distance par rapport au neurone élu.

[Knagenhjelm,90] **P. Knagenhjelm, P. Bauer**
Infrastructure in Kohonen Maps
Proc. IEEE, Glasgow, Vol 1, mai 1990

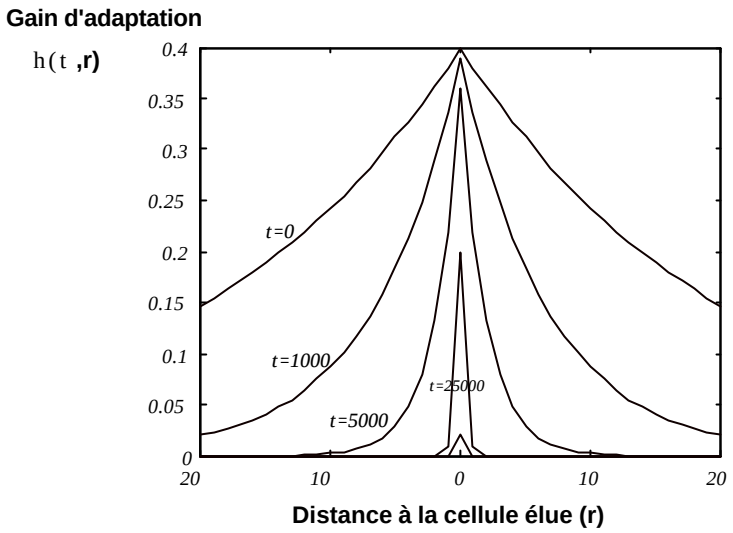


figure 3.10 - Evolution du taux d'apprentissage.

A l'issue d'un nombre suffisant d'itérations, les vecteurs de poids synaptiques se sont accordés spécifiquement sur les divers aspects des variables d'entrée. Un *quantificateur vectoriel*[1] a été créé: les vecteurs de poids, qui sont devenus les centroïdes des vecteurs d'entrée, ont migré dans l'espace de représentation R^d de ceux-ci, afin de représenter au mieux l'espace des signaux d'entrée. La carte auto-organisatrice présente cependant une différence par rapport à un quantificateur vectoriel conventionnel, puisque les cellules topologiquement proches sont ici sensibles à des signaux physiquement similaires.

Ce quantificateur n'est toutefois pas encore utilisable pour une tâche de reconnaissance. Le processus d'apprentissage n'ayant en effet pas été, jusqu'ici, supervisé, il convient à présent de déterminer quelles cellules se sont accordées sur tel ou tel type de stimulus: c'est l'objet de *l'étiquetage*. Cette procédure consiste à attribuer à chaque cellule une étiquette qui correspond à la classe d'entrée dont elle est la plus proche. A cette fin, l'ensemble des vecteurs de caractéristiques des objets d'apprentissage sont à nouveau présentés au système, et la fréquence avec laquelle chaque neurone est le plus proche d'une classe particulière est calculée. A la suite de cette étape, chaque neurone se voit associé à la classe pour laquelle il a été le plus fréquemment choisi. Il peut se produire que certains neurones ne soient jamais sélectionnés en tant que le plus proche d'un vecteur d'entrée d'un prototype d'apprentissage. Ils doivent, dans ce

[1] Développée à l'origine à des fins de codage de données, la quantification vectorielle consiste à substituer à un vecteur X, l'indice du vecteur le plus proche, au sens de la distance Euclidienne, déterminé parmi un ensemble restreint de prototypes possibles. Utilisée en mode de classification, la quantification consiste à rechercher la classe du prototype, plutôt que son indice.

cas, être écartés lors de l'utilisation de la carte en reconnaissance, afin d'éviter toute indétermination.

A l'issue de la procédure d'étiquetage, la carte auto-organisatrice peut enfin être utilisée en classificateur. Un vecteur de caractéristiques est présenté à l'entrée du réseau, et la classe reconnue est celle qui est associée au neurone dont le vecteur de poids synaptiques est le plus proche du vecteur d'entrée.

3.4.3.3 L'Algorithme « *Learning Vector Quantization* »

L'inconvénient de l'apprentissage non supervisé décrit précédemment est que les poids des neurones sont organisés de manière à minimiser une erreur de quantification, alors que l'objectif réel est de reconnaître la classe du vecteur d'entrée. Seules les frontières entre les classes importent donc vraiment. Une meilleure estimation de celles-ci peut être obtenue en introduisant une phase d'apprentissage supplémentaire, au cours de laquelle la classification des objets, ignorée jusqu'ici, est prise en considération. Cet apprentissage supervisé consiste à présenter à nouveau des vecteurs de caractéristiques au réseau, et à réajuster les vecteurs de poids de manière à minimiser le taux global d'erreurs de classification.

Cet algorithme, appelé *Learning Vector Quantization* (LVQ), a également été développé par Kohonen.

L'algorithme initial consistait à renforcer, à chaque itération, la ressemblance entre le vecteur de poids (centroïde) du neurone élu et celui du vecteur de caractéristiques présenté lorsque les classes de ceux-ci étaient équivalentes, et à éloigner les deux vecteurs dans le cas contraire. Le vecteur de poids du neurone le plus proche du vecteur d'entrée est alors adapté selon:

$$W_{q^*}(\tau+1) = W_{q^*}(\tau) + \eta(\tau)\left(X - W_{q^*}(\tau)\right) \quad \text{si} \quad \omega_{q^*} = \omega_X$$

$$W_{q^*}(\tau+1) = W_{q^*}(\tau) - \eta(\tau)\left(X - W_{q^*}(\tau)\right) \quad \text{si} \quad \omega_{q^*} \neq \omega_X \quad (3.34)$$

où:

- $\eta(\tau)$ est un gain scalaire qui décroît de manière monotone avec le temps
- ω_X et ω_{q^*} désignent respectivement la classe de l'objet représenté par le vecteur de caractéristiques X et celle associée à la cellule élue q^*.

L'algorithme connu sous le nom de *LVQ2* est une évolution de cet algorithme initial. Trois conditions doivent à présent être remplies pour qu'il y ait effectivement réajustement des centroïdes:

1. Le centroïde le plus proche du vecteur d'entrée doit être de classe différente de celui-ci;
2. Le centroïde le plus proche suivant doit appartenir à la même classe que le vecteur d'entrée;
3. Le vecteur d'entrée doit se trouver dans une fenêtre symétrique définie autour du plan médian entre ces deux centroïdes.

Si ces trois conditions sont simultanément vérifiées, le vecteur de référence incorrect est éloigné du vecteur d'entrée alors que le second vecteur en est rapproché. Les équations qui assurent un tel apprentissage sont les suivantes:

$$W_{q_1}(\tau+1) = W_{q_1}(\tau) - \eta(\tau)\left(X - W_{q_1}(\tau)\right)$$

$$W_{q_2}(\tau+1) = W_{q_2}(\tau) + \eta(\tau)\left(X - W_{q_2}(\tau)\right), \quad (3.35)$$

q_1 et q_2 désignant respectivement les neurones dont le vecteur de poids est le plus proche du vecteur d'entrée et le suivant le plus proche.

A titre d'exemple, la *figure 3.11* illustre une application de l'algorithme LVQ2 dans un cas monodimensionnel.

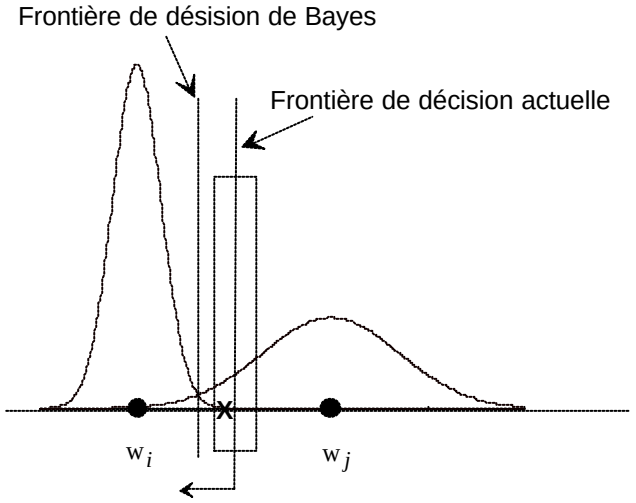


figure 3.11 - Adaptation des centroïdes par LVQ2.

Les deux classes ω_i et ω_j sont séparées par une frontière qui, à la suite de l'apprentissage non supervisé, se situe à mi-distance entre les deux centroïdes *i* et *j* associés à ces classes. La frontière optimale de décision se situe en réalité là où l'expression $p(x|\omega_i)p(\omega_i) = p(x|\omega_j)p(\omega_j)$ est vérifiée. Lorsqu'un objet *X* appartenant à la classe ω_j apparaît dans la partie gauche d'une fenêtre centrée sur la région de décision actuelle, donc mal placé, les trois conditions exigées par l'algorithme LVQ2 sont vérifiées, et il y aura donc réajustement des centroïdes. Le centroïde correspondant à la classe incorrecte sera éloigné du vecteur *X*, tandis que celui qui correspond à la classe correcte en sera rapproché. La frontière de décision entre les deux centroïdes se rapproche, dans ce cas, de la frontière optimale de Bayes. Remarquons que les formes mal classées qui apparaissent dans la partie droite de la fenêtre produiront l'effet contraire. Celui-ci demeure cependant négligeable car il y a beaucoup plus de classifications erronées de formes situées dans la partie gauche de la fenêtre et appartenant à la classe ω_j, que de classifications erronées de formes ω_i situées dans la partie droite de la fenêtre.

3.4.3.4 Propriétés de la Carte Auto-Organisatrice

Hormis le fait que les neurones topologiquement proches répondent à des stimuli physiquement semblables, la carte auto-organisatrice de Kohonen n'apporte rien de plus qu'un quantificateur vectoriel conventionnel. Bien que cette particularité puisse être intéressante à observer, elle est en soi inutile à la classification, puisque seule la classe associée au neurone dont le vecteur de poids synaptiques est le plus proche du vecteur d'entrée est alors requise.

Puisque la classe attribuée à un objet, présenté en entrée de la carte, est celle associée au neurone dont le vecteur de poids est le plus proche du vecteur de caractéristiques de l'objet, les fonctions discriminantes sont ici de la forme:

$$\Phi_i(X) = -\min_{W_q \in \omega_i} \frac{1}{2}(X - W_q)^T (X - W_q) \quad (3.36)$$

Comme vu précédemment, ces fonctions discriminantes peuvent également s'écrire:

$$\Phi_i(X) = \max_{\underline{W}_q \in \omega_i} \left(\underline{W}_q^T \underline{X}\right), \quad (3.37)$$

à condition que l'on ait posé:

$$w_{q0} = \frac{1}{2} W_q^T W_q \quad (3.38)$$

Les frontières de décision entre deux centroïdes sont linéaires (*figure 3.12*). Un tel classificateur, qui utilise un nombre multiple mais restreint, de vecteurs de prototypes pour chaque classe, peut être considéré comme un système intermédiaire entre le classificateur Euclidien, qui utilise un seul prototype pour chaque classe, et le classificateur du Plus Proche Voisin, qui utilise tous les vecteurs de caractéristiques des objets d'apprentissage comme prototype pour chaque classe. L'apprentissage discriminant, que subit dans un second temps la carte auto-organisatrice, lui confère toutefois un avantage certain. Le nombre optimal de cellules du réseau dépend essentiellement de l'application envisagée, et, comme dans le cas d'un quantificateur vectoriel conventionnel, doit être déterminé empiriquement.

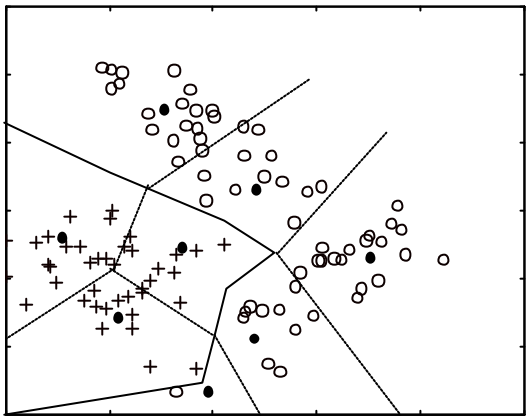


Figure 3.12 - Frontières de décisions fournies par la carte auto-organisatrice.

3.4.4 Le Perceptron

3.4.4.1 Architecture du Réseau

Présenté originellement par Rosenblatt, en 1958, le perceptron est la forme la plus simple de réseau de neurones, et permet de classifier correctement des objets appartenant à deux classes linéairement séparables. Il consiste en un seul neurone qui possède un seuil

ainsi qu'un vecteur de poids synaptiques ajustables, tout comme le modèle de neurone de McCulloch & Pitts (*figure 3.13*).

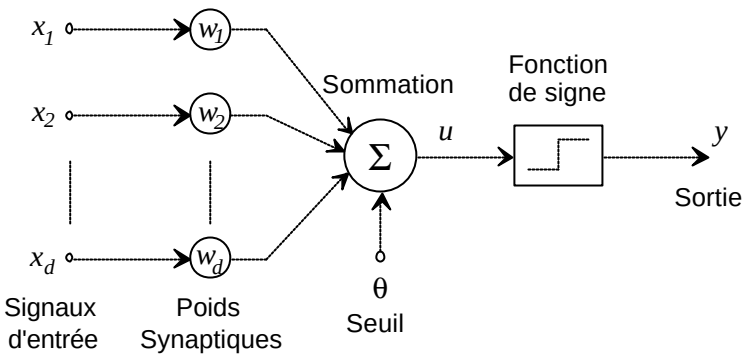


Figure 3.13 - Représentation du Perceptron.

Le perceptron associe à chaque classe une fonction discriminante linéaire qui s'exprime:

$$\Phi_i(X) = \underline{W_i}^T \underline{X}\,, \quad (3.39)$$

où:

- $\underline{W_i} = [w_0\ w_1\ w_2 \cdots w_n]^T$ est un vecteur de coefficients de pondération
- $\underline{X} = [-1\ x_1\ x_2 \cdots x_d]^T$ le vecteur de caractéristiques augmenté d'un objet à classifier.

Dans le cas d'un problème à deux classes, la règle de classification (3.1) s'écrit:

$$X \in \omega_1 \quad \text{ssi} \quad \Phi_1(X) \geq \Phi_2(X), \qquad X \in \omega_2 \text{ sinon. } (3.40)$$

Ce qui peut également s'exprimer selon:

$$X \in \omega_1 \quad \text{ssi} \quad \Phi_{12}(X) = \Phi_1(X) - \Phi_2(X) \geq 0, \qquad X \in \omega_2 \text{ sinon. } (3.41)$$

Les relations étant linéaires, il vient:

$$X \in \omega_1 \quad \text{ssi} \quad \Phi_{12}(X) = \underline{W_{12}}^T \underline{X} \geq 0, \quad X \in \omega_2 \text{ sinon. } (3.42)$$

C'est cette règle de décision qui est réalisée dans le perceptron, grâce à l'utilisation de la fonction de signe, définie par (3.27), comme fonction d'activation du neurone. La sortie du neurone vaut alors +1 si l'objet X appartient à la classe ω_1, et -1 dans le cas contraire.

Le perceptron illustré à la *figure 3.13* ne contient qu'un neurone. Celui-ci ne permet, dès lors, que d'effectuer la classification dans un problème à deux classes seulement. La reconnaissance de plusieurs classes est cependant rendue possible par la mise en parallèle de plusieurs perceptrons (*figure 3.14*). Le perceptron ainsi obtenu comporte un neurone par classe, chacun de ceux-ci réalisant une fonction discriminante linéaire de la classe à laquelle il est associé:

$$\Phi_i(X) = \underline{W}_i^T \underline{X} \quad (3.43)$$

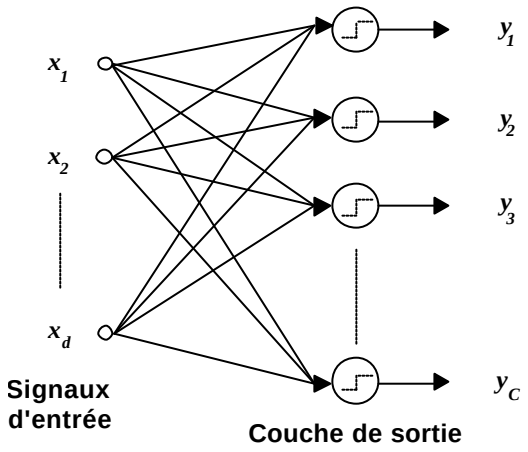


Figure 3.14 - Représentation d'un perceptron pour la reconnaissance de C classes.

Les paramètres du perceptron, c'est-à-dire les poids synaptiques des neurones, peuvent être déterminés grâce à un entraînement supervisé, effectué sur un ensemble de formes préclassifiées. La règle d'apprentissage du perceptron, développée originellement par Rosenblatt, est assurée de converger si les données sont linéairement séparables [Minsky,69]. Le perceptron partitionne l'espace des variables d'entrée en régions correspondant chacune à une classe, selon des frontières de décision linéaires, constituées de segments d'hyperplans, définis par $\Phi_i(X) - \Phi_j(X) = 0$. Le seuil des neurones permet de définir des hyperplans qui ne contiennent pas nécessairement l'origine de l'espace des paramètres.

[Minsky,69] **M. Minsky & S. Papert**
Perceptrons: A Introduction to Computational Geometry
The MIT Press, Cambridge, MA, 1969

La règle d'apprentissage du perceptron ne converge toutefois pas lorsque les objets à classifier ne sont pas linéairement séparables. Un autre algorithme peut alors être utilisé, tel le critère des moindres carrés de l'erreur.

3.4.4.2 Le Critère des Moindres Carrés de l'Erreur

Cette méthode d'apprentissage consiste à adapter les poids des neurones de manière à minimiser une fonction de coût, définie comme étant la somme, pour tous les objets d'apprentissage et pour toutes les sorties du réseau, du carré de l'écart entre la valeur réelle de chaque sortie et sa valeur désirée:

$$E = \sum_{k=1}^{N} \sum_{i=1}^{C} \left(\Phi_i(X_k) - t_i^{(k)} \right)^2 \quad (3.44)$$

où:

- $\Phi_i(X_k)$ représente la sortie du neurone *i* du perceptron lorsque le vecteur de caractéristiques X_k est présenté à l'entrée;
- $t_i^{(k)}$ est la valeur de sortie désirée, pour le neurone *i*, associée à ce même vecteur de caractéristiques.

Pour une tâche de classification, la sortie désirée vaut 1 pour le neurone qui correspond à la classe de l'objet présenté, et 0 pour tous les autres neurones. De plus, les sorties des neurones sont à présent des valeurs continues, qui approximent les valeurs 0 ou 1, plutôt que d'effectuer une simple décision logique entre ces deux alternatives. Ce n'est donc pas l'erreur de classification qui est minimisée (via une fonction de coût qui permettrait de compter simplement le nombre des erreurs), mais seulement une erreur moyenne.

L'expression (3.44) peut être réécrite:

$$E = \sum_{c=1}^{C} \sum_{X_k \in \omega_c} \sum_{i=1}^{C} \left(\underline{W}_i^T \underline{X}_k - \delta_{ic} \right)^2 \quad (3.45)$$

où: δ_{ic} = 1 si $i=c$
= 0 sinon.

Pour chaque vecteur d'apprentissage, une classification correcte est renforcée, alors qu'une classification incorrecte est pénalisée. L'apprentissage du perceptron est donc *discriminant*.

Les valeurs optimales des poids synaptiques sont obtenues en annulant le gradient de (3.45) par rapport aux paramètres du système, et il vient alors:

$$\underline{\mathbf{W}} = \left(\underline{\mathbf{X}}\underline{\mathbf{X}}^T\right)^{-1}\underline{\mathbf{M}}\,\mathbf{N} \quad (3.46)$$

où:

- $\underline{\mathbf{W}}$ est une matrice à (d+1) lignes et C colonnes, où chacune de ces dernières contient le vecteur de poids augmenté d'un neurone;
- $\underline{\mathbf{X}}$ est une matrice à (d+1) lignes et *N* colonnes contenant l'ensemble des vecteurs de caractéristiques augmentés des objets d'apprentissage;
- $\underline{\mathbf{M}}$ est une matrice à (d+1) lignes et C colonnes, dans laquelle chaque colonne est le vecteur de caractéristiques augmenté moyen d'une classe;
- **N** est une matrice diagonale à C lignes et C colonnes, dont les éléments diagonaux représentent le nombre total d'objets disponibles pour chaque classe.

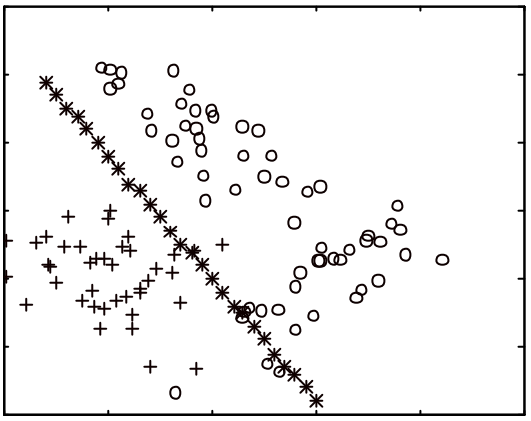


figure 3.15 - Frontière de décision générée par le perceptron.

Comme déjà précisé précédemment, les frontières de décision pour la classification offertes par le perceptron sont linéaires (*figure 3.15*). L'apprentissage discriminant, que subit celui-ci, lui confère toutefois un avantage certain vis-à-vis du classificateur Euclidien conventionnel, décrit à la section 2.1 du présent chapitre. Néanmoins, nombre de problèmes de classification nécessitent une partition non linéaire de l'espace d'entrée. Ces problèmes peuvent être résolus à l'aide du perceptron multicouches, qui consiste à mettre en cascade deux couches ou plus de perceptrons.

3.4.5 Le Perceptron Multicouches

3.4.5.1 Architecture du Réseau

La mise en cascade de perceptrons conduit à ce qu'on appelle le perceptron multicouches (*figure 3.16*). Les perceptrons employés ici diffèrent cependant de celui de Rosenblatt, par le fait que la non-linéarité utilisée est à présent une fonction continue, d'allure sigmoïdale par exemple, et non plus la fonction de signe. Lorsque le vecteur de caractéristiques d'un objet est présenté à l'entrée du réseau, il est communiqué à tous les neurones de la première couche. Les sorties des neurones de cette couche sont alors communiquées aux neurones de la couche suivante, et ainsi de suite. La dernière couche du réseau est appelée *couche de sortie*, les autres étant désignées sous le terme de *couches cachées* car les valeurs de sortie de leurs neurones ne sont pas accessibles de l'extérieur.

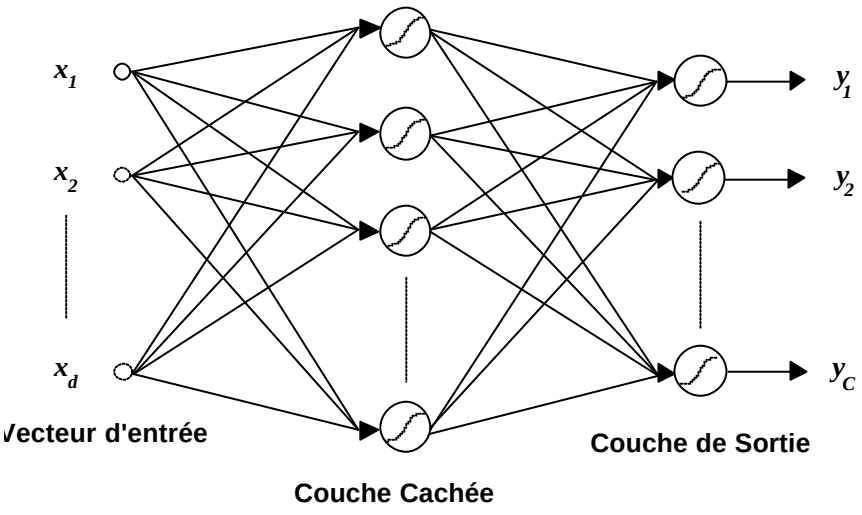


Figure 3.16 - Perceptron Multicouches à une couche cachée.

Dans le cas où une seule couche cachée est présente, les fonctions discriminantes réalisées par un tel réseau de neurones sont de la forme:

$$\Phi_i(X) = \varphi_{2,i}\left(-w_{2,i0} + \sum_{j=1}^{h_1} w_{2,ij}\varphi_{2,j}\left(-w_{1,i0} + \sum_{q=1}^{d} w_{1,jq}x_q\right)\right) \quad (3.47)$$

où:

- $\varphi_{l,i}(\cdot)$ représente la fonction d'activation du neurone *i* de la couche *l*;
- $\underline{W}_{l,i} = \left[w_{l,i0}\ w_{l,i1}\ w_{l,i2}\ \cdots\ w_{l,ih_{l-1}}\right]^T$ est le vecteur de poids augmenté de ce même neurone;
- h_l vaut le nombre de neurones de la couche *l*;
- $X = \left[x_1\ x_2\ \cdots\ x_d\right]^T$ est le vecteur de caractéristiques présenté à l'entrée du réseau.

La fonction d'activation des neurones doit absolument être non linéaire, car sans elle, le perceptron multicouches ne ferait qu'implanter une série de transformations linéaires consécutives, qui pourraient dès lors se réduire à une seule. Grâce à l'utilisation de fonctions d'activations non linéaires, le perceptron multicouches est à même de générer des fonctions discriminantes non linéaires. L'algorithme d'apprentissage du perceptron multicouches, connu sous le nom d'*algorithme de rétro-propagation*, nécessite toutefois que les fonctions d'activations des neurones soient continues et dérivables. Cette condition mise à part, celles-ci peuvent cependant être quelconques, offrant ainsi une grande liberté de modélisation. Les fonctions qui sont le plus couramment utilisées sont probablement les fonctions de type sigmoïdal (*figure 3.17*), qui peuvent être considérées comme des approximations continues de la fonction seuil de Heaviside.

$$\varphi(u) = \frac{1}{1+\exp(-\beta u)}$$

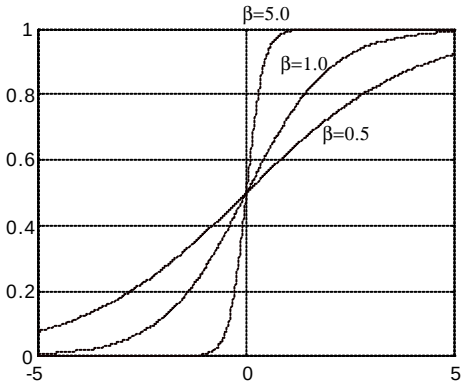


Figure 3.17 - Fonctions sigmoïdales.

3.4.5.2 La Phase d'Apprentissage

L'apprentissage du perceptron multicouches est supervisé, et consiste à adapter les poids des neurones de manière à ce que le réseau soit capable de réaliser une transformation donnée, représentée par un ensemble d'exemples constitué d'une suite de *N* vecteurs d'entrées $X_k = \left[x_{k1}\ x_{k2}\ \cdots x_{kd}\right]^T$ associée à une autre suite de vecteurs de sorties désirées $T^{(k)} = \left[t_1^{(k)}\ t_2^{(k)}\ \cdots t_{h_L}^{(k)}\right]^T$. Le critère des moindres carrés de l'erreur peut être utilisé pour définir la fonction de coût à minimiser, qui s'exprime alors:

$$E = \frac{1}{2}\sum_{k=1}^{N}\sum_{i=1}^{h_L}\left(y_{L,i}^{(k)} - t_i^{(k)}\right)^2 \quad (3.48)$$

où:

- N est le nombre d'exemples d'apprentissage;
- L est le nombre de couches du réseau;
- h_l vaut le nombre de neurones que contient la couche *l*;
- $y_{l,i}^{(k)}$ désigne la sortie du neurone i de la couche l lorsque le vecteur X_k est présenté à l'entrée du réseau;
- $t_i^{(k)}$ représente la valeur de sortie désirée pour le neurone *i* de la dernière couche lorsque le vecteur X_k est présenté à l'entrée du réseau;

La minimisation de cette fonction de coût non-linéaire se fait de manière itérative, en utilisant une méthode de gradient (*cfr. Annexe A*). La paternité de l'algorithme d'apprentissage sous sa forme actuelle peut être attribuée à Werbos, et date de 1974 [Werbos,74].

En prenant la convention de noter par $\underline{Y}_l^{(k)} = \left[-1\ y_{l,1}^{(k)}\ y_{l,2}^{(k)}\ \cdots y_{l,h_l}^{(k)}\right]^T$ le vecteur augmenté des sorties de la couche *l* lorsque le vecteur X_k est présenté à l'entrée du réseau et par $\underline{W}_{l,i} = \left[w_{l,i0}\ w_{l,i1}\ w_{l,i2}\ \cdots w_{l,ih_{l-1}}\right]^T$ le vecteur augmenté des poids du neurone *i* de la couche *l*, la règle d'apprentissage du perceptron multicouches peut se résumer comme suit:

[Werbos,74] **P. Werbos**
Beyond regression: New tools for prediction and analysis in the behavioral sciences
PhD thesis, Harvard University, Cambridge, MA, 1974

1. Initialiser l'ensemble des vecteurs de poids synaptiques $W_{l,i}$ à des valeurs aléatoires;
2. Appliquer un exemple d'apprentissage X_k à l'entrée du perceptron multicouches et évaluer l'erreur quadratique commise par le réseau sur cet exemple:

$$E^{(k)} = \sum_{i=1}^{h_L} \left(y_{L,i}^{(k)} - t_i^{(k)} \right)^2 \quad (3.49)$$

3. Corriger l'ensemble des poids synaptiques selon:

$$w_{l,ij}(\tau+1) = w_{l,ij}(\tau) - \alpha \frac{\partial E^{(k)}}{\partial w_{l,ij}} \quad (3.50)$$

où α est le taux d'apprentissage, $\alpha > 0$, et où:

$$\frac{\partial E^{(k)}}{\partial w_{l,ij}} = \delta_{l,i}^{(k)} y_{l-1,j}^{(k)} \quad (3.51)$$

avec, pour un neurone de la couche de sortie:

$$\delta_{L,i}^{(k)} = \dot{\varphi}_{L,i}\left(\underline{W}_{L,i}^T \underline{Y}_{L-1}^{(k)} \right) \left(y_{L,i}^{(k)} - t_i^{(k)} \right) \quad (3.52)$$

et pour un neurone d'une couche cachée:

$$\delta_{l,i}^{(k)} = \dot{\varphi}_{l,i}\left(\underline{W}_{l,i}^T \underline{Y}_{l-1}^{(k)} \right) \sum_{q=1}^{h_{l+1}} w_{l+1,qi} \delta_{l+1,q}^{(k)} \quad (3.53)$$

où:

$$\dot{\varphi}_{l,i}\left(u^{(k)} \right) = \frac{\partial \varphi}{\partial u} \Big| u = u^{(k)} \quad (3.54)$$

Lorsque la fonction d'activation des neurones est la fonction sigmoïde, il vient,
pour un neurone de la couche de sortie:

$$\delta_{L,i}^{(k)} = y_{L,i}^{(k)} \left(1 - y_{L,i}^{(k)} \right) \left(y_{L,i}^{(k)} - t_i^{(k)} \right) \quad (3.55)$$

pour un neurone d'une couche cachée:

$$\delta_{l,i}^{(k)} = y_{l,i}^{(k)} \left(1 - y_{l,i}^{(k)} \right) \sum_{q=1}^{h_{l+1}} w_{l+1,qi} \delta_{l+1,q}^{(k)} \quad (3.56)$$

4. Incrémenter τ et k, et reprendre au point 2.

Le processus de calcul du gradient de l'erreur et d'adaptation des poids est répété jusqu'au moment où un minimum de l'erreur, ou un point qui en est suffisamment proche, est trouvé. La fin de la phase d'apprentissage peut être décidée lorsque l'erreur moyenne observée à la sortie du réseau devient inférieure à un seuil prédéfini, ou encore lorsque l'amplitude moyenne du gradient de cette erreur devient très faible, puisque, par définition, le gradient est nul au minimum de la fonction de coût.

C'est l'expression (3.53) qui a donné son nom à l'algorithme d'apprentissage du perceptron multicouches: *rétro-propagation* du gradient de l'erreur. Le gradient local $\delta_{l,i}^{(k)}$ d'un neurone est en effet calculé à partir des gradients locaux $\delta_{l+1,q}^{(k)}$ des neurones de la couche ultérieure. Le calcul des gradients commence donc par la dernière couche, et est ensuite propagé progressivement de celle-ci vers la première couche du réseau.

Lorsque le perceptron multicouches est utilisé en tant que classificateur, la couche de sortie comporte un neurone par classe. En phase d'apprentissage, les valeurs désirées pour ces sorties sont alors:

- 1 pour le neurone qui correspond à la classe de l'objet présenté à l'entrée du réseau;
- 0 pour tous les autres neurones.

De la définition de la fonction de coût (3.49), l'apprentissage tend à augmenter la valeur de sortie du neurone qui correspond à la classe correcte, tout en cherchant à diminuer la valeur des sorties des neurones associés aux classes incorrectes. Il en résulte donc un apprentissage *discriminant*. La *figure 3.18* illustre un exemple de frontière de décision non linéaire générée par un perceptron multicouches.

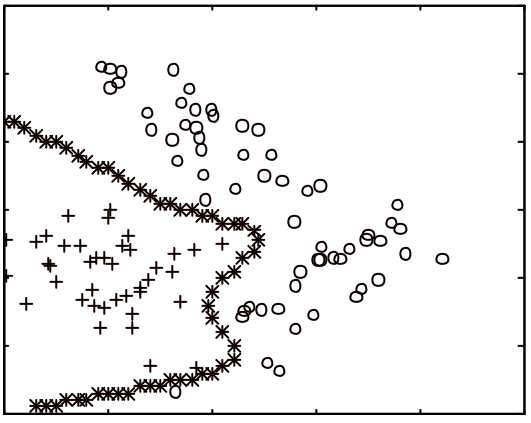


Figure 3.18 - Frontière de décision non linéaire obtenue à l'aide d'un perceptron multicouches. Ce dernier contenait 10 neurones en sa couche cachée.

3.4.5.3 Propriétés du Perceptron Multicouches

Cybenko est le premier à avoir démontré qu'un perceptron multicouches à d entrées et h_L sorties, comportant une seule couche cachée composée de neurones à fonction d'activation continue non linéaire, est suffisant pour approximer, au sens des moindres carrés, avec une erreur arbitrairement faible pour un ensemble donné d'objets d'apprentissage, n'importe quelle transformation continue représentée par un ensemble de vecteurs d'entrées $X_k = \left[x_{k1}\ x_{k2} \cdots x_{kd} \right]^T$ et un ensemble de vecteurs de sorties désirées $T^{(k)} = \left[t_1^{(k)}\ t_2^{(k)} \cdots t_{h_L}^{(k)} \right]^T$ [Cybenko,89] [Haykin,94]. La démonstration assume simplement que la non-linéarité est une fonction continue monotone croissante, qui est bornée inférieurement et supérieurement. Bien que ce soit une fonction d'allure sigmoïdale qui soit le plus couramment utilisée, il y a donc un grand nombre d'alternatives possibles à celle-ci. Il n'est en outre pas obligatoire qu'une non linéarité soit présente dans la couche de sortie.

Le théorème d'approximation prouve qu'un perceptron multicouches à une seule couche cachée est en théorie toujours suffisant. Toutefois, il ne préjuge en aucun cas du nombre d'unités cachées qui est nécessaire pour atteindre une qualité d'approximation suffisante. Ce nombre pouvant parfois être gigantesque, l'utilisation d'un perceptron multicouches à deux (ou plus) couches cachées ne comportant chacune qu'un nombre restreint de neurones peut parfois s'avérer être plus avantageuse.

Pour des problèmes de classification, il a été démontré [Bou & Wel,89] que, si un perceptron multicouches à une seule couche cachée, composée de neurones à fonction d'activation continue non linéaire, est entraîné sous les conditions suivantes:

[Cybenko,89] **G. Cybenko**
Approximation by Superpositions of a Sigmoïdal Function
Mathematics of Control, Signals, and Sytems, 2, pp 303-314, 1989

[Haykin,94] **S. Haykin**
Neural Networks - A comprehensive Foundation
Macmillan College Publishing Company, New York, 1994

[Bou & Wel,89] **H. Bourlard & C.J. Wellekens**
Links between markov models and multilayer perceptrons
D.S. Touretzki, editor, Advances in Neural Information Processing Systems,
Volume 1, pages 502-510, San Mateo, CA, 1989. IEEE, Morgan Kaufmann

1. le réseau comporte un neurone de sortie par classe, et est entraîné à produire 1 sur la sortie associée à la classe de l'objet présenté, et 0 sur toutes les autres sorties;
2. le critère d'optimisation est celui des moindres carrés de l'erreur;
3. le nombre d'unités cachées est suffisamment grand;
4. l'apprentissage ne converge pas vers un minimum local;

alors, les sorties du perceptron multicouches peuvent être interprétées comme de bonnes approximations, au sens des moindres carrés, des probabilités *à posteriori* des classes.

Le perceptron multicouches permet donc dans ce cas d'approximer la fonction discriminante optimale de Bayes, définie par la relation (3.3). Cette interprétation justifie l'utilisation du perceptron multicouches en tant que classificateur. La qualité de l'approximation n'est cependant pas garantie et dépend de l'apprentissage du réseau.

Comme pour le théorème d'approximation cité précédemment, il n'y a aucune obligation à ce que la non-linéarité des unités cachées soit réalisée par une fonction sigmoïdale. Cette non-linéarité a uniquement pour rôle de générer des moments d'ordres plus élevés des composantes du vecteur d'entrée, ce qui peut être obtenu à l'aide de bien d'autres fonctions que les sigmoïdes. D'autre part, la présence d'une non-linéarité dans la couche de sortie est à présent essentielle. Son rôle est ici de simuler une décision logique binaire, ce qui permet de minimiser le taux d'erreur de classification. Le choix de la non-linéarité se restreint par conséquent ici aux fonctions de type sigmoïdal.

Le rôle de la fonction d'activation non linéaire des unités cachées peut être mieux appréhendé grâce à l'interprétation suivante, proposée par H. Bourlard [Bourlard,94]. Soient $\underline{X}_k = \left[-1\ x_{k1}\ x_{k2}\ \cdots\ x_{kd}\right]^T$ un vecteur de caractéristiques augmenté présenté à l'entrée du réseau et $\underline{W}_{1,i} = \left[w_{1,i0}\ w_{1,i1}\ w_{1,i2}\ \cdots\ w_{1,id}\right]^T$ le vecteur de poids augmenté du neurone i de la couche cachée. Lorsque la fonction d'activation de ce neurone est une fonction sigmoïdale, la sortie de celui-ci vaut:

$$y_{1,i}^{(k)} = \frac{1}{1+\exp\left(-\sum_{j=0}^{d} w_{1,ij} x_{kj}\right)} \quad (3.57)$$

[Bourlard,94] **H. Bourlard, N. Morgan**
Connectionist Speech Recognition - A Hybrid Approach
Kluwer Academic Publishers, 1994

Le développement en série de Taylor de cette expression donne:

$$y_{1,i}^{(k)} = \sum_{n=0}^{\infty} a_n \left(\sum_{j=0}^{d} w_{1,ij} x_{kj} \right)^n \quad (3.58)$$

Dans le cas d'entrées binaires, on a $x_{kj}^n = x_{kj}, \forall n > 0$, et l'expression (3.58) peut être réécrite comme suit:

$$y_{1,i}^{(k)} = \alpha_0 + \sum_{j=0}^{d} \beta_j x_{kj} + \sum_{q \neq j}^{d} \gamma_{jq} x_{kj} x_{kq} + \sum_{p \neq q \neq j}^{d} \delta_{jqp} x_{kj} x_{kq} x_{kp} + \quad \cdots \quad (3.59)$$

Pour chaque neurone de la couche cachée, la fonction d'activation non linéaire génère une *combinaison linéaire* de tous les 2^d produits croisés possibles des *d* entrées binaires, permettant ainsi de prendre en considération des moments d'ordres élevés. Les coefficients $\alpha, \beta, \gamma, \ldots$ qui affectent ces produits croisés, dépendent des valeurs des poids qui relient les neurones entre eux. Ces coefficients sont adaptés lors de l'apprentissage du perceptron multicouches de manière à extraire les produits croisés qui sont nécessaires pour effectuer la tâche demandée, c'est-à-dire, pour une tâche de classification, ceux qui sont discriminants et insensibles au bruit.

Lors de la phase d'utilisation, un vecteur d'entrée donné activera certains produits croisés, et les valeurs des sorties seront calculées sur base de ceux-ci. Si le vecteur d'entrée comporte du bruit, certains produits croisés en seront également affectés. Cependant, si l'apprentissage s'est déroulé correctement, ces produits là ne seront plus pris en compte, et des valeurs de sortie correctes pourront être obtenues sur base des produits croisés insensibles au bruit extraits lors de la phase d'apprentissage.

3.4.6 Autres Modèles de Réseaux de Neurones

3.4.6.1 Réseaux de Neurones à Poids Partagés

La fascination de certains chercheurs pour les réseaux de neurones artificiels les a conduits à utiliser ceux-ci non plus seulement pour la classification, mais aussi pour des procédures qui faisaient jusque là partie d'une étape distincte des systèmes de reconnaissance de formes. Un des exemples les plus connus est probablement le perceptron multicouches à poids partagés, où l'étape d'extraction de primitives est traitée par le réseau de neurones lui-même.

Développé initialement par Y. Lecun, spécialement dans le cadre de la reconnaissance de codes postaux aux Etats-Unis [LeCun,90], il s'agit d'un perceptron multicouches à quatre couches cachées dont les trois premières ne sont pas complètement interconnectées (*figure 3.19*). Les neurones de ces couches sont regroupés en sous-couches, où tous les neurones partagent les mêmes valeurs de poids synaptiques (d'où le nom de ce modèle de réseau), mais prennent en compte une partie distincte des sorties de la couche précédente. Les neurones d'une même sous-couche effectuent donc un même traitement, c'est-à-dire qu'ils extraient les mêmes caractéristiques locales, mais sur des entrées distinctes. Celles-ci peuvent être totalement distinctes ou se chevaucher partiellement. Le nombre de neurones d'une sous-couche est celui nécessaire pour recouvrir l'ensemble des sorties de la couche précédente.

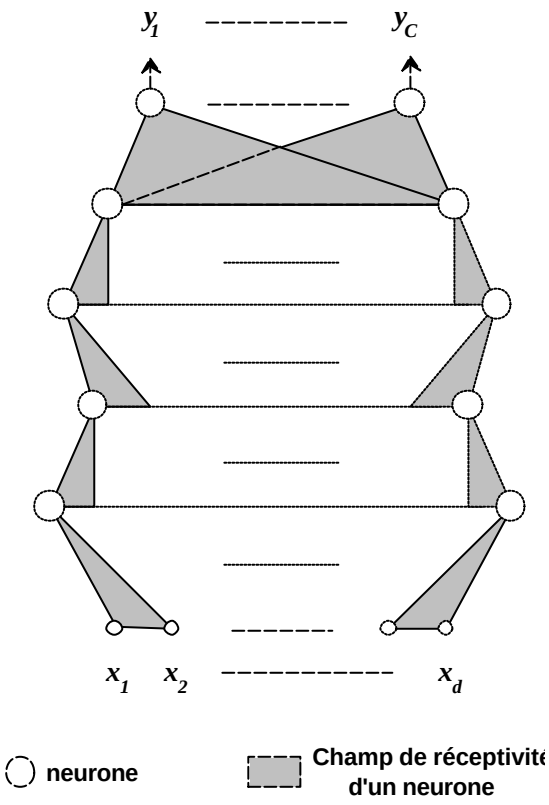


Figure 3.19 - Perceptron multicouches à poids partagés.

[LeCun,90] **Y. LeCun**
Constrained Neural Network for Unconstrained Handwritten Digit Recognition
Proc. of International Workshop on Frontiers in Handwritting Recognition,
pp 145-154, Montreal, Canada,Avril 1990

Le vecteur d'entrée de ce réseau est directement l'image même d'un caractère, codée sur une matrice de 20 par 20 pixels. La première couche est subdivisée en quatre sous-couches dont les neurones sont connectés à une zone de 5 par 5 pixels de l'image d'entrée, décalée chaque fois de un pixel d'un neurone à l'autre (*figure 3.20*). L'opération effectuée par chacune des sous-couches peut être vue comme une convolution bidimensionnelle non-linéaire, entre les vecteurs de poids synaptiques des neurones qu'elle contient et l'image d'entrée.

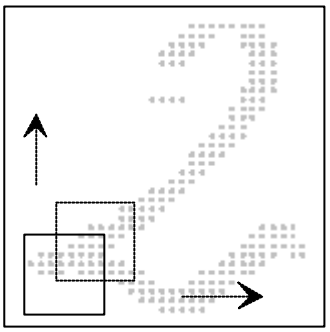


Figure 3.20 - Prélèvement des pixels de l'image.

La deuxième couche cachée a pour rôle de réduire la dimension de représentation des caractéristiques extraites par la première couche. Elle comporte également quatre sous-couches, de taille quatre fois plus faible que celle des sous-couches précédentes. Chaque neurone n'effectue ici que le calcul de la valeur moyenne de quatre sorties connexes de la couche précédente.

Les neurones de la troisième couche cachée sont répartis en 12 sous-couches, et possèdent chacun deux groupes de 5×5 entrées prises de manière similaire à celles de la première couche. Les deux groupes d'entrées sont toutefois issus d'une sous-couche distincte de la couche précédente. Le rôle des neurones de la troisième couche est de combiner les caractéristiques locales extraites à l'aide des deux premières couches, en caractéristiques d'ordres plus élevés.

La quatrième couche cachée comporte également 12 sous-couches, dont les neurones effectuent la même opération de calcul de valeur moyenne que ceux de la deuxième couche.

La dernière couche est la seule à être complètement interconnectée et comporte 300 entrées et 10 sorties, une pour chaque chiffre.

Au total, ce réseau comporte 5430 neurones et 135 736 poids synaptiques, dont 3600 seulement sont des paramètres libres [Haykin,94].

L'algorithme d'apprentissage d'un tel réseau de neurones est similaire à celui de rétro-propagation. Le principal avantage de ce réseau de neurones est que, grâce au partage des valeurs des poids synaptiques, le nombre de paramètres indépendants dans l'ensemble du système est relativement limité, réduisant ainsi le temps d'apprentissage de manière significative. De plus, le partage des mêmes valeurs de poids synaptiques d'un neurone à l'autre, entraîne des traitements équivalents de parties d'image distinctes, offrant alors une grande tolérance vis-à-vis des translations et/ou rotations éventuelles du caractère. Ce modèle de réseau a permis d'atteindre, sur des chiffres manuscrits présegmentés extraits de codes postaux aux Etats-Unis, un taux de reconnaissance absolu de 95,1%, et un taux de rejet de 9,1% pour un taux d'erreur mesuré de 1% [Haykin,94].

L'inconvénient majeur de ce système est cependant que le nombre de neurones et de poids synaptiques qu'il contient est très élevé, impliquant ainsi un temps de reconnaissance considérable. Ce problème s'accentue encore lorsque le nombre de classes augmente (reconnaissance des 26 lettres de l'alphabet latin, voire des 26 lettres et des 10 chiffres simultanément). Le temps de calcul nécessaire à la reconnaissance devient alors prohibitif, nécessitant par conséquent, pour une application pratique, la disponibilité de machines puissantes.

3.4.6.2 Réseaux à Fonction Radiale de Base

Le réseau à Fonction Radiale de Base comporte deux couches de neurones (*figure 3.21*). Les cellules de sortie effectuent une combinaison linéaire de fonctions de base non linéaires, fournies par les neurones de la couche cachée. Ces fonctions de base produisent une réponse différente de zéro seulement lorsque l'entrée se situe dans une petite région bien localisée de l'espace des variables. Bien que plusieurs modèles de fonctions de base existent, le plus courant est de type Gaussien:

$$y_{1,i}(X) = \exp\left[-\frac{(X - W_{1,i})^T (X - W_{1,j})}{2\sigma_i^2}\right] \quad (3.60)$$

où:

[Haykin,94] **S. Haykin**
Neural Networks - A comprehensive Foundation
Macmillan College Publishing Company, New York, 1994

- X est le vecteur d'entrée du réseau
- $y_{1,i}$ est la sortie du neurone i de la première couche
- W et σ_i^2 sont respectivement le vecteur de poids synaptiques et le paramètre de normalisation de ce neurone (W correspond ici aux coordonnées du centre de la Gaussienne).

La sortie d'un neurone de la seconde couche est simplement donnée par:

$$y_{2,i} = W_{2,i}^T Y_1 \quad (3.61)$$

où Y_1 est le vecteur des sorties des neurones de la première couche.

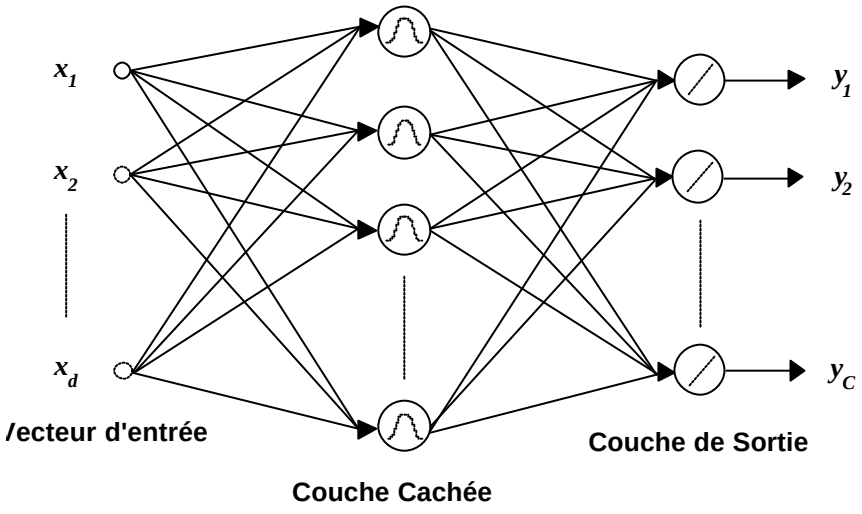


Figure 3.21 - Réseau de neurones à fonction radiale de base.

Plus le vecteur d'entrée est proche du centre d'une Gaussienne, plus la sortie du neurone de la première couche qui lui correspond est élevée. Le terme « Fonction Radiale de Base » vient du fait que la Gaussienne est symétrique radialement, c'est-à-dire que la valeur de sortie obtenue est la même pour toutes les entrées situées à une même distance du centre de la Gaussienne.

L'apprentissage du réseau à fonction radiale de base est généralement scindé en deux parties [Hush,93]. Dans un premier temps, les poids des neurones de la couche cachée sont

[Hush,93] **D. R. Hush & B. G. Horne**
Progress in Supervised Neural Networks
IEEE Signal Processing Magazine, pp 8-39, Janvier 1993

adaptés au moyen d'une quantification vectorielle. Il existe plusieurs manières d'effectuer cette dernière, mais celle qui est probablement la plus efficace et la plus simple à mettre en oeuvre est l'algorithme *K-moyennes* [Deketelaere,94]. Lorsque les poids des cellules cachées sont fixés, les paramètres de normalisation σ_i^2 sont déterminés en calculant la dispersion des données d'apprentissage associées à chaque centroïde. La seconde couche du réseau peut alors être entraînée. L'apprentissage est cette fois supervisé (les valeurs de sortie désirées sont fournies), et s'effectue typiquement à l'aide d'un algorithme basé sur un critère des Moindres Carrés de l'Erreur. L'entraînement de la seconde couche est très rapide, car, d'une part, les sorties de la couche cachée peuvent être calculées une fois pour toutes pour tous les exemples d'apprentissage, et d'autre part, les sorties des neurones de la seconde couche sont linéaires. Les méthodes classiques d'apprentissage de transformations linéaires, tel le critère des moindres carrés du perceptron (*cfr. § 3.4.4.2*), peuvent donc être appliquées.

Le fait que l'apprentissage de la couche cachée soit non supervisé est toutefois un inconvénient de ce modèle de réseau vis-à-vis d'un perceptron multicouches. Pour palier à ce problème, des méthodes d'apprentissage supervisé de réseaux à fonction radiale de base ont également été développées [Musavi et al,92] [Wettschereck,92]. Un avantage du réseau à fonction radiale de base est que sa phase d'apprentissage est plus rapide que celle du perceptron multicouches. Ceci ne constitue toutefois qu'un avantage utile à l'étude et à la mise au point d'un système (de classification ou autre). Dans l'optique d'une utilisation pratique, c'est en effet le volume de calcul à effectuer en phase d'exploitation qu'il est préférable d'optimiser.

En théorie, le réseau à fonctions radiales de base est capable, tout comme le perceptron multicouches, d'effectuer une approximation arbitrairement proche de n'importe quelle transformation non linéaire [Hartman,90]. La principale différence entre les deux est la nature de

[Deketelaere,94] **S. Deketelaere**
Application de la quantification vectorielle au codage bas débit d'un signal de parole
Thèse de doctorat, Faculté Polytechnique de Mons, 1994

[Musavi et al,92] **M. T. Musavi, W. Ahmed, K. H. Chan, K. B. Faris, & D. M. Hummels**
On the Training of the Radial Basis Function Classifiers
Neural Networks, vol 5, pp 595-603, 1992

[Wettschereck,92] **D. Wettschereck & T. Dietterich**
Improving the Performance of Radial Basis Function Networks by Learning Center Locations
J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, Advances in Neural Information Processing Systems 4, pp 1133-1140, 1992

[Hartman,90] **E.J Hartman, J. D. Keeler, & J. M. Kowalski**
Layered Neural Networks with Gaussien Hidden Units as Universal Approximations
Neural Computation, vol 2, n°2, pp 210-215, 1990

la fonction d'activation des neurones de la couche cachée. La non-linéarité sigmoïdale utilisée dans le perceptron multicouches fournit une sortie différente de zéro pour une région infiniment grande de l'espace d'entrée, ce qui lui confère un certain pouvoir de généralisation dans des régions où peu de données d'apprentissage, voire aucune, ne sont disponibles. Au contraire, la non-linéarité radiale utilisée ici, ne fournit une réponse différente de zéro que localement. Le nombre d'unités cachées nécessaire pour permettre au réseau de recouvrir l'ensemble de l'espace d'entrée peut dès lors s'avérer parfois très élevé [Haykin,94].

Une autre différence entre ces deux modèles de réseau est que la non-linéarité, présente dans la couche de sortie du perceptron multicouches, est inexistante dans le réseau à fonction radiale de base, ce qui constitue un désavantage de ce dernier vis-à-vis du premier. L'importance de cette non-linéarité, pour un apprentissage minimisant le taux global d'erreur de classification, a en effet été mise en évidence au *§ 3.4.5.3*.

La caractéristique linéaire de la couche de sortie d'un réseau à fonction radiale de base rend ce dernier plus proche du perceptron original de Rosenblatt que ne l'est le perceptron multicouches. Le réseau à fonction radiale de base diffère cependant du perceptron, dans le sens où il est capable de réaliser des transformations non linéaires de l'espace d'entrée. La *figure 3.22* illustre un exemple de frontières de décision générées par le réseau à fonction radiale de base.

[Haykin,94] **S. Haykin**
Neural Networks - A comprehensive Foundation
Macmillan College Publishing Company, New York, 1994

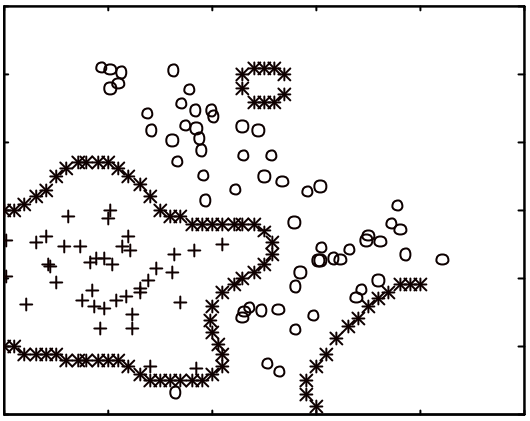


Figure 3.22 - Frontières de décision obtenues à l'aide d'un réseau à fonction radiale de base. 64 neurones furent nécessaires en couche cachée (contre 10 seulement pour le perceptron multicouches).

3.5 Evaluation des Performances

Les performances d'un classificateur quelconque peuvent être représentées par sa matrice de confusion, qui donne, pour chaque classe, la probabilité que ses éléments soient reconnus comme appartenant à cette classe ou à chacune des autres classes possibles. Ces performances peuvent également être résumées par les probabilités d'erreur de classification, calculées pour chaque classe, ou encore, par la valeur moyenne de ces mêmes probabilités, calculée sur l'ensemble des classes. En pratique, toutes ces probabilités ne peuvent qu'être estimées, en comptant le nombre d'erreurs de classification commises sur un ensemble fini d'objets. Le nombre de données disponibles est généralement restreint, car celles-ci sont fastidieuses à collecter et à traiter. Plusieurs méthodes d'estimation de l'erreur de classification existent, et visent à optimiser l'utilisation de ces données.

- ***La méthode de Resubstitution*** consiste à entraîner le classificateur avec toute la base de données, et à le tester ensuite avec les mêmes échantillons qui ont servis à l'entraîner. Cette méthode permet de mieux construire le classificateur, puisque le nombre de données d'entraînement est maximal. L'erreur de classification mesurée dans ce cas est cependant biaisée et généralement trop optimiste, à cause d'un phénomène de sur-entraînement, qui peut être considéré comme l'analogue d'un phénomène de Gibbs en approximation polynomiale.

- ***La méthode du « Leave-One-Out »*** consiste, elle, à entraîner le classificateur avec toutes les données, sauf une, qui sert à le tester ensuite. Cette méthode fourni une estimation quasiment non biaisée du taux d'erreur, mais est extrêmement coûteuse en volume de calculs. Il est en effet nécessaire de réentraîner le classificateur autant de fois qu'il y a d'éléments dans la base de données, afin de pouvoir calculer le taux d'erreur moyen.

- ***La méthode du « Holdout »*** consiste à partager aléatoirement l'ensemble total de données en un ensemble d'entraînement et un ensemble de tests, ne comportant aucun élément en commun. L'inconvénient de cette méthode est que le nombre de données disponibles pour l'apprentissage du classificateur se trouve réduit. L'estimation obtenue du taux d'erreur est alors également biaisée, et donne une borne supérieure de la valeur de ce dernier.

- ***La méthode de Rotation*** est un compromis des deux méthodes précédentes. L'ensemble total de données est partagé aléatoirement en k partitions distinctes, dont une sert à tester la classificateur entraîné à l'aide des $(k-1)$ autres. Une rotation des partitions est ensuite réalisée, et de nouveaux tests effectués. Lorsque chacune des k partitions a été utilisée pour tester le classificateur, le taux d'erreur moyen peut être calculé. Cette méthode permet d'obtenir une estimation du taux d'erreur moins biaisée que celle fournie par la méthode du Holdout, tout en réduisant le volume de calcul requis par la méthode du Leave-One-Out.

3.6 Résumé

Plusieurs modèles de classificateurs ont été présentés. Ceux-ci peuvent être répartis en trois catégories: les classificateurs paramétriques, non-paramétriques, et neuronaux. Les frontières de décision que ces classificateurs sont à même de construire ont été décrites. La complexité de celles-ci va de la séparation linéaire unique offerte par le classificateur Euclidien, à la séparation non linéaire d'ordre élevé offerte par le perceptron multicouches.

Les classificateurs paramétriques peuvent s'avérer très efficaces sous certaines hypothèses de distribution des variables. Les classificateurs non paramétriques ainsi que les réseaux de neurones ne font, par contre, aucune hypothèse sur l'allure de cette distribution, qu'ils sont capables de modéliser à l'aide d'un apprentissage par l'exemple. Les réseaux de neurones peuvent en outre subir un apprentissage discriminant, ce qui constitue un avantage considérable pour des tâches de classification.

Chapitre 4

De l'Utilisation du Perceptron Multicouches

4.1 Introduction

Nous avons vu que, sous certaines conditions, les sorties d'un perceptron multicouches constituent une bonne estimation des probabilités *à posteriori* de chaque classe (*§ 3.4.5.3*). Cette propriété fait donc potentiellement de ce modèle de réseau de neurones, un des meilleurs classificateurs qui puisse être construit. En pratique, cependant, la qualité de l'approximation n'est pas garantie; elle est limitée par le nombre de paramètres disponibles, et dépend surtout de l'apprentissage du réseau de neurones, qui consiste en la minimisation d'une fonction de coût à très grand nombre de variables, et qui est dès lors souvent délicat. Les principaux problèmes qui se posent, sont la rencontre de minima locaux de la fonction de coût, ce qui conclut prématurément la phase d'apprentissage, ainsi qu'une convergence relativement lente de l'algorithme de rétro-propagation. L'objet de ce chapitre est de proposer diverses méthodes pouvant permettre d'accomplir plus aisément cette phase d'apprentissage.

4.2 Quelques Astuces d'Apprentissage

4.2.1 Utilisation d'un Terme de Moment

La formule d'adaptation des poids synaptiques, fournie par l'algorithme de rétro-propagation, est souvent modifiée par l'ajout d'un terme de moment. Dans ce cas, la valeur d'un poids synaptique n'est plus seulement adaptée proportionnellement à la dérivée de la fonction de coût par rapport à ce poids, mais est également modifiée en fonction de la correction appliquée à l'instant précédent. Sous forme mathématique, la formule d'adaptation des poids synaptiques s'écrit alors:

$$w_{l,ij}(\tau+1) - w_{l,ij}(\tau) = -\alpha \frac{\partial E^{(k)}}{\partial w_{l,ij}} + \beta\left(w_{l,ij}(\tau) - w_{l,ij}(\tau-1)\right) \quad (4.1)$$

où β, $0 \le \beta \le 1$, est un paramètre qui est appelé *terme de moment*.

Cette expression peut être réécrite:

$$w_{l,ij}(\tau+1) - w_{l,ij}(\tau) = -\alpha \sum_{q=0}^{\tau} \beta^q \frac{\partial E^{(k-q)}}{\partial w_{l,ij}} \quad (4.2)$$

Grâce à l'utilisation d'un terme de moment, la direction de recherche du minimum à un instant donné, est une somme pondérée des gradients actuel et précédents. La pondération qui intervient, est telle que l'importance relative d'un gradient décroît exponentiellement au fur et à mesure que ce dernier est éloigné dans le temps. Le fait de prendre en compte plusieurs gradients consécutifs aide les poids synaptiques à traverser les sections plates de la surface de la fonction de coût, après qu'ils en aient descendu des sections abruptes [Fombellida,93]. Ceci permet en outre de modifier les poids synaptiques, non plus à l'aide d'un même taux d'adaptation pour l'ensemble de ceux-ci, mais selon un taux qui est propre à chaque poids, et qui est dépendant de son histoire particulière.

[Fombellida,93] **M. Fombellida**
Architectures connexionnistes évolutives: algorithmes d'apprentissage et heuristiques
Thèse de Doctorat, Université de Liège, octobre 1993

Les valeurs des paramètres α et β doivent être déterminées empiriquement, de manière à limiter la fréquence d'apparition de deux phénomènes qui sont opposés, mais qui conduisent tous deux à un net ralentissement de l'évolution de l'apprentissage.

- D'une part, lorsque les valeurs de ces paramètres sont faibles, les poids synaptiques sont peu modifiés quand une région de faible pente de la fonction de coût est rencontrée; l'ensemble du système n'évolue alors que lentement.

- D'autre part, lorsque les valeurs de ces paramètres sont élevées, des corrections d'amplitude importante sont appliquées aux poids synaptiques dans les régions de forte pente de la fonction de coût, ce qui peut entraîner une augmentation de la valeur de cette dernière. Il en résulte ainsi également un ralentissement de l'évolution de l'apprentissage.

L'apprentissage du perceptron multicouches nécessite donc une phase de méta-optimisation, au cours de laquelle des valeurs des coefficients α et β, conduisant au meilleur compromis entre ces deux situations extrêmes, doivent être recherchées.

4.2.2 Les Minima Locaux

Lorsque le critère d'apprentissage du perceptron multicouches est celui des Moindres Carrés de l'Erreur, et que la non-linéarité des neurones est réalisée au moyen d'une fonction sigmoïdale, le gradient local observé pour un neurone *i* vaut (*§ 3.4.5.2*):

si *i* est un neurone de la couche de sortie:

$$\delta_{L,i}^{(k)} = y_{L,i}^{(k)}\left(1-y_{L,i}^{(k)}\right)\left(y_{L,i}^{(k)}-t_i^{(k)}\right) \quad (4.3)$$

Si *i* est un neurone d'une couche interne:

$$\delta_{l,i}^{(k)} = y_{l,i}^{(k)}\left(1-y_{l,i}^{(k)}\right)\sum_{q=1}^{h_{l+1}} w_{l+1,qi}\delta_{l+1,q}^{(k)} \quad (4.4)$$

Lorsque la sortie d'un neurone est proche d'une des deux valeurs de saturation de la fonction sigmoïdale, le gradient local est donc très faible. Les corrections appliquées aux poids synaptiques de ce neurone deviennent alors insignifiantes, conduisant ainsi à une stagnation de l'apprentissage du perceptron multicouches. Ce phénomène se produit quelle que soit la valeur de saturation obtenue, et donc également lorsque celle-ci est à l'opposé de celle que l'on désire atteindre! Ainsi, par exemple, un neurone appartenant à la dernière couche, et dont la valeur de

sortie est presque nulle, verra ses poids synaptiques pratiquement inchangés, même si la valeur désirée de sa sortie est l'unité. A la limite, si une saturation absolue est atteinte, plus aucune adaptation des poids synaptiques de ce neurone n'est possible, et l'apprentissage reste bloqué dans un minimum local.

Un autre critère de minimisation que celui des Moindres Carrés de l'Erreur peut cependant être utilisé, et permet d'éviter le ralentissement de l'apprentissage à cause d'une saturation de neurones appartenant à la couche de sortie. Ce critère est connu sous le nom de *critère de l'Entropie Relative*, et consiste à utiliser la fonction de coût définie comme suit [Kara & Venet,93]:

$$E^{(k)} = \sum_{i=1}^{h_L} \left[t_i^{(k)} \ln\left(\frac{t_i^{(k)}}{y_{L,i}^{(k)}} \right) + \left(1 - t_i^{(k)}\right) \ln\left(\frac{1 - t_i^{(k)}}{1 - y_{L,i}^{(k)}} \right) \right] \quad (4.5)$$

La valeur minimale de cette fonction de coût est également atteinte lorsque les sorties réelles des neurones valent les sorties désirées. On peut montrer (*cfr. Annexe A*) que le gradient local d'un neurone de la couche de sortie s'exprime à présent:

$$\delta_{L,i}^{(k)} = \frac{\left(y_{L,i}^{(k)} - t_i^{(k)}\right)}{y_{L,i}^{(k)}\left(1 - y_{L,i}^{(k)}\right)} \dot{\varphi}_{L,i}\left(u_{L,i}^{(k)}\right) \quad (4.6)$$

Soit, lorsque la fonction d'activation de ces neurones est la fonction sigmoïde:

$$\delta_{L,i}^{(k)} = y_{L,i}^{(k)} - t_i^{(k)} \quad (4.7)$$

L'utilisation du critère de l'Entropie Relative permet donc de corriger les poids synaptiques d'un neurone de la couche de sortie, avec une amplitude d'autant plus élevée que la valeur réelle de sortie de ce neurone est éloignée de celle désirée. Les quelques essais d'apprentissage que nous avons effectués, ont rapidement confirmé que cela permettait une accélération significative de la convergence de l'algorithme de rétro-propagation.

L'expression (4.4), qui fournit la valeur du gradient local d'un neurone d'une couche cachée, à partir des valeurs des gradients locaux des neurones de la couche qui lui est ultérieure, n'est cependant aucunement modifiée. Des minima locaux peuvent donc toujours apparaître à

[Kara & Venet,93] **N.B. Karayiannis & A. N. Venetsanopoulos**
Artificial Neural Networks - Learning Algorithms, Performance Evaluation, and Applications
Kluwer Academic Publishers, 1993

cause d'une saturation de neurones appartenant à une couche cachée. Le risque d'apparition d'un tel phénomène peut toutefois être réduit en initialisant l'ensemble des vecteurs de poids synaptiques du perceptron multicouches à des valeurs aléatoires de faible amplitude.

4.2.3 La Validation Croisée

Lorsque, au cours de l'apprentissage, l'évolution du taux de reconnaissance est relevée pour des objets qui participent à l'entraînement du perceptron multicouches d'une part, et pour des objets qui n'y participent pas d'autre part, des courbes semblables à celles illustrées à la *figure 4.1* sont obtenues. Alors que le taux de reconnaissance observé sur l'ensemble des objets d'apprentissage ne cesse de croître, celui mesuré sur l'ensemble de test diminue lorsque la procédure d'apprentissage se prolonge au delà d'un certain point. Ce phénomène s'explique par le fait que, si le perceptron multicouches apprend effectivement les caractéristiques propres à chaque classe au début de son entraînement, il commence ensuite à modéliser trop fidèlement les objets de l'ensemble d'apprentissage eux-mêmes, ainsi que le bruit que ces derniers comportent. Cela peut être interprété comme l'analogue d'un phénomène de Gibbs en approximation polynomiale.

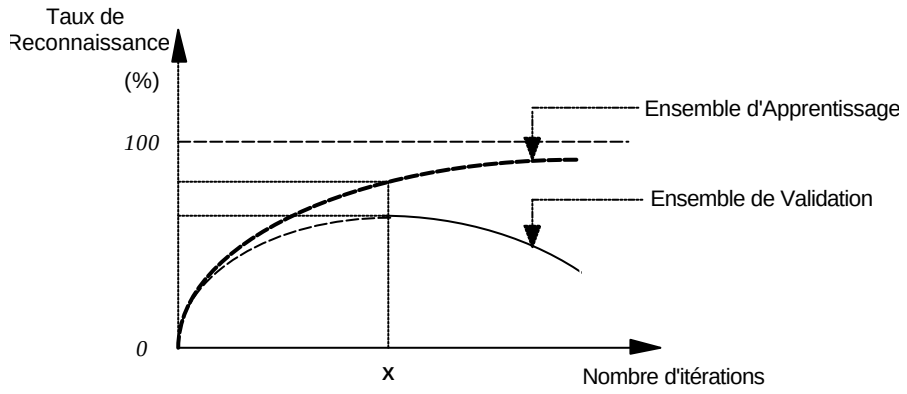


Figure 4.1 - Principe de la Validation Croisée.

La diminution des performances de reconnaissance, à cause d'un entraînement trop prolongé du perceptron multicouches, peut être évité en procédant, tout au long de celui-ci, à une *validation croisée* [Bourlard,94]. Cette procédure consiste à mesurer régulièrement, au cours de

[Bourlard,94] **H. Bourlard, N. Morgan**
Connectionist Speech Recognition - A Hybrid Approach

la phase d'apprentissage, le taux de reconnaissance atteint sur un ensemble d'objets ne participant pas à l'adaptation des poids synaptiques du réseau de neurones. L'entraînement du perceptron multicouches peut alors être interrompu lorsque le taux de reconnaissance mesuré en validation est maximum. Cette méthode présente cependant l'inconvénient d'être plus coûteuse en temps de calcul, ou, lorsque la quantité de données est limitée, l'inconvénient de réduire encore la quantité disponible pour effectuer l'apprentissage du réseau de neurones. Malgré ces désavantages, le recours à une procédure de validation croisée s'avère indispensable en pratique, afin de pouvoir arrêter la phase d'apprentissage au moment le plus opportun.

4.3 Les Différents Modes d'Apprentissage

4.3.1 l'Apprentissage En-Ligne

Il existe deux modes principaux d'apprentissage, selon la façon dont les vecteurs de poids synaptiques sont adaptés. Le premier, dit apprentissage « en-ligne », consiste à modifier les valeurs de ces poids synaptiques immédiatement après la présentation d'un objet. Dans ce cas, seul le gradient instantané de la fonction de coût est utilisé pour l'adaptation des paramètres du système. Sous la condition que les objets soient présentés au réseau de neurones de manière aléatoire, l'apprentissage en-ligne rend la recherche du minimum de la fonction de coût stochastique en nature, ce qui rend moins probable, pour l'algorithme de rétro-propagation, de tomber dans un minimum local.

4.3.2 l'Apprentissage Hors-Ligne

Le second mode principal d'apprentissage consiste, quant à lui, à accumuler les gradients instantanés consécutifs, et à n'effectuer l'adaptation des poids synaptiques que lorsque l'ensemble des objets d'apprentissage ont été présentés au perceptron multicouches. On parle alors d'apprentissage «hors-ligne ». Cette dernière méthode permet de mieux estimer le gradient réel de la fonction de coût, puisqu'il est à présent calculé à partir d'un ensemble d'objets, plutôt qu'à partir d'un seul.

L'efficacité relative des modes d'apprentissage en-ligne et hors-ligne dépend essentiellement du problème considéré. L'apprentissage en-ligne présente cependant l'avantage que, pour une seule présentation de l'ensemble de la base de données, il implique de multiples

phases d'adaptations des poids synaptiques lorsque des données similaires se représentent, ce qui se produit fréquemment pour des bases de données très étendues.

4.3.3 l'Apprentissage en Demi-Ligne

Sous l'hypothèse, non restrictive lorsque le but de l'apprentissage n'est que de construire un classificateur, que chacune des classes possède la même probabilité d'occurrence *à priori*, un troisième type d'apprentissage peut être introduit. Celui-ci consiste alors à présenter successivement au réseau de neurones un seul exemplaire de chaque classe, d'accumuler les gradients instantanés, et d'effectuer l'adaptation des poids synaptiques lorsque, pour chaque classe, ***un*** exemplaire aura été présenté. La procédure est alors répétée pour un autre exemplaire de chaque classe. Ce mode d'apprentissage apparaît comme une solution de compromis entre les modes d'apprentissage en-ligne et hors-ligne, et nous lui avons dès lors donné le nom d'apprentissage en *demi-ligne*.

L'hypothèse que chacune des classes possède la même probabilité d'occurrence *à priori* peut sembler fort restrictive au premier abord. Elle ne l'est cependant en aucune manière lorsque l'entraînement du perceptron multicouches n'est effectué à d'autres fins que la classification, comme nous allons le démontrer dans les lignes qui suivent.

A l'issue d'un entraînement optimal, les sorties du perceptron multicouches constituent une bonne estimation des probabilités *à posteriori* que possède chaque classe d'être celle de l'objet présenté à l'entrée du réseau de neurones (*§ 3.4.5.3*). Ces probabilités *à posteriori* peuvent s'exprimer, au moyen de la formule de Bayes:

$$p(\omega_i|x) = \frac{p(x|\omega_i)\,p(\omega_i)}{p(x)} \quad (4.8)$$

où:

$$p(x) = \sum_{k=1}^{C} p(x|\omega_k)\,p(\omega_k) \quad (4.9)$$

C étant le nombre de classes.

L'hypothèse d'une équiprobabilité *à priori* de chaque classe revient donc à entraîner le perceptron multicouches de manière telle que ses sorties soient une bonne estimation de:

$$\tilde{p}(\omega_i|x) = \frac{p(x|\omega_i)\frac{1}{C}}{\tilde{p}(x)} \quad (4.10)$$

où:

$$\tilde{p}(x) = \frac{1}{C}\sum_{k=1}^{C} p(x|\omega_k) \quad (4.11)$$

Lors de l'utilisation du perceptron multicouches en phase de reconnaissance, la probabilité *à priori* de chaque classe peut être aisément prise en compte, en multipliant simplement par celle-ci, la valeur de chaque sortie du réseau de neurones. En phase de reconnaissance, les valeurs de $p(x)$ et de $\tilde{p}(x)$ sont en outre des constantes quelle que soit la classe, et le fait d'utiliser l'un ou l'autre de ces termes ne modifie donc pas les valeurs *relatives* des fonctions discriminantes réalisées. La qualité de la classification qui est ainsi accomplie n'est dès lors aucunement altérée.

L'hypothèse d'équiprobabilité effectuée lors de l'apprentissage du perceptron multicouches ne requiert donc, afin d'obtenir une qualité de classification comparable à celle atteinte en l'absence de cette hypothèse, que de multiplier, en phase de reconnaissance, chaque valeur de sortie du réseau de neurones par la probabilité *à priori* de la classe qui lui est associée. Ce procédé offre en outre l'avantage de ne devoir entraîner le réseau de neurones qu'une et une seule fois pour toutes, lorsqu'il est destiné à être utilisé dans des applications où les probabilités *à priori* sont susceptibles d'évoluer. Ainsi, par exemple, un système de reconnaissance de caractères peut-il être aisément adapté pour une langue particulière, ou un système de reconnaissance de chiffres peut-il être modifié, suivant que l'on désire traiter des codes postaux ou des montants de transferts bancaires, où la probabilité *à priori* du chiffre « 0 » est plus élevée.

Lorsque l'on désire obtenir, en couche de sortie, une image la plus fidèle possible des probabilités *à posteriori*, le mode d'apprentissage en demi-ligne est inapplicable. En outre, il est alors souhaitable d'assurer que la somme des valeurs des sorties soit égale à l'unité, afin de respecter la définition de la probabilité. A cet effet, la fonction d'activation sigmoïdale des neurones de la couche de sortie peut être remplacée par la fonction *softmax*, définie comme suit [Bourlard,94]:

$$\varphi(u_i) = \frac{e^{u_i}}{\sum_{k=1}^{h_L} e^{u_k}} \quad (4.12)$$

[Bourlard,94] **H. Bourlard, N. Morgan**
Connectionist Speech Recognition - A Hybrid Approach
Kluwer Academic Publishers, 1994

Nous avons cependant constaté qu'en pratique, même lorsque seule la fonction sigmoïdale est utilisée, la somme des sorties des neurones possède une tendance naturelle à s'approcher assez nettement de l'unité. Cette constatation a également été observée fréquemment par ailleurs [Hush,93] [Bourlard,94]. Notre but n'étant pas d'obtenir absolument une image fidèle des probabilités *à posteriori*, mais bien d'effectuer une classification performante, nous continuerons à employer la fonction sigmoïdale classique.

4.4 Méthodes d'Apprentissage Accéléré

4.4.1 Introduction

Malgré les diverses astuces d'apprentissage exposées aux sections précédentes, l'entraînement d'un perceptron multicouches demeure souvent une tâche très longue et très fastidieuse. De nombreuses modifications de l'algorithme de rétro-propagation du gradient de l'erreur ont donc été proposées, afin d'en accélérer la vitesse de convergence. Des méthodes de minimisation de *Newton*, ou de *Quasi-Newton*, ont même parfois été envisagées [Fombellida,93]. Ces algorithmes font intervenir les dérivées partielles de second ordre de la fonction de coût, et sont donc *à priori* plus performants qu'une « simple » méthode de gradient. D'un point de vue pratique, cependant, les réseaux de neurones sont rarement de dimension suffisamment faible que pour pouvoir procéder au calcul des multiples dérivées partielles secondes en un laps de temps raisonnable. Ainsi, si le nombre d'itérations à effectuer pour atteindre le minimum est réduit, la durée de chacune d'entre elles se voit augmentée de façon telle que l'ensemble de la procédure d'apprentissage n'est accélérée en aucune manière. Les méthodes d'apprentissage accéléré qui sont exposées dans les sections qui suivent, sont dès lors des méthodes qui ne nécessitent que le calcul des dérivées du premier ordre de la fonction de coût, et qui peuvent ainsi être appliquées à l'entraînement de perceptrons multicouches de grande taille.

[Hush,93] **D. R. Hush & B. G. Horne**
Progress in Supervised Neural Networks
IEEE Signal Processing Magazine, pp 8-39, Janvier 1993

[Bourlard,94] **H. Bourlard, N. Morgan**
Connectionist Speech Recognition - A Hybrid Approach
Kluwer Academic Publishers, 1994

[Fombellida,93] **M. Fombellida**
Architectures connexionnistes évolutives: algorithmes d'apprentissage et heuristiques
Thèse de Doctorat, Université de Liège, octobre 1993

4.4.2 Description des Problèmes de Test

4.4.2.1 Le « OU » Exclusif

L'apprentissage de la fonction logique « OU » exclusif entre deux variables est un problème classique de test du perceptron multicouches. Il s'agit d'un problème de classification, dont les deux classes « 0 » ou « 1 » ne sont pas linéairement séparables. Ce problème peut être résolu à l'aide d'un perceptron multicouches comportant une seule couche cachée, de faible taille. Ceci rend ce dernier très aisé à mettre en oeuvre et rapide à apprendre, ce qui permet d'effectuer de nombreux essais et d'assurer ainsi une mise au point optimale des divers paramètres d'apprentissage.

Les essais d'apprentissage ont été effectués à l'aide d'une centaine d'exemplaires de chacune des classes, bruités de manière aléatoire. L'amplitude du bruit généré était limitée à 0,25 , de sorte que deux droites sont suffisantes pour séparer correctement les classes (*figure 4.2*). Une couche cachée ne comportant que deux neurones est alors suffisante pour permettre une classification parfaite de tous les objets d'apprentissage.

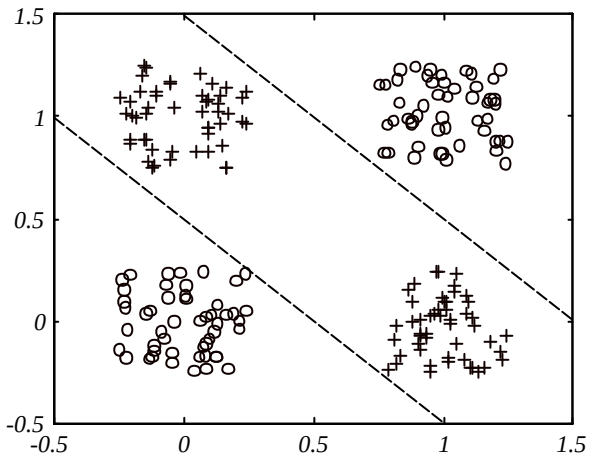


Figure 4.2 - Ensemble d'apprentissage pour le problème du « OU » exclusif, et frontières de décision optimales .

Les trois modes d'apprentissage (en-ligne, en demi-ligne, et hors-ligne) ont systématiquement été testés, sans et avec utilisation d'un terme de moment. Dans chaque cas, de très nombreux essais ont été effectués, de manière à parfaire la phase de méta-optimisation des

paramètres d'apprentissage. Le but de ce problème n'étant que de permettre une comparaison des vitesses relatives de convergence de divers algorithmes d'apprentissage, aucune validation croisée n'a été effectuée ici, et l'entraînement était considéré comme terminé lorsque l'ensemble de tous les objets était correctement classifié.

4.4.2.2 La Reconnaissance de Caractères Manuscrits

Le second problème destiné aux tests des divers algorithmes d'apprentissage du perceptron multicouches est la reconnaissance des 26 lettres majuscules latines, dans un contexte manuscrit et multiscripteur. Un perceptron à une seule couche cachée a été entraîné, sur base de 220 exemplaires manuscrits de chaque lettre, écrits par 50 personnes distinctes. Une cinquantaine d'autres exemplaires, ayant été écrits par 20 autres personnes, ont été utilisés pour la validation. L'apprentissage a été ici interrompu lorsque le taux de reconnaissance mesuré sur l'ensemble de validation était maximal.

Après leur digitalisation, les images des lettres ont été traduites en un vecteur de caractéristiques de 64 composantes. Ces dernières sont obtenues en superposant, à l'image d'un caractère, une grille de 8 x 8 mailles, et en calculant ensuite la valeur arithmétique moyenne des pixels contenus dans chacune de ces mailles (*figure 4.3*).

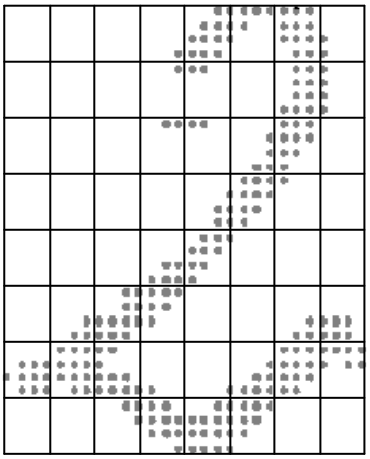


Figure 4.3 - Codage de l'image d'un caractère: la valeur arithmétique moyenne des pixels de chaque maille est calculée.

Le perceptron multicouches employé comportait donc 64 entrées, et, le nombre total de classes s'élevant à 26, la couche de sortie contenait 26 neurones. Comme le nombre d'unités

cachées nécessaires ne peut être déterminé *à priori*, plusieurs essais ont dû être effectués pour fixer celui permettant d'atteindre les meilleurs résultats de reconnaissance. Un nombre de neurones en couche cachée égal à 30 a été adopté à l'issue de ces essais, et a été conservé pour tous les perceptrons multicouches qui ont été entraînés à l'aide des divers algorithmes développés, afin de pouvoir comparer objectivement ces derniers entre eux.

4.4.3 Résultats de Référence

Les résultats qui servent de référence, sont les meilleurs ayant été obtenus en appliquant l'algorithme de rétro-propagation de base, sans et avec terme de moment. Ils ne tiennent pas compte des diverses tentatives d'apprentissage qui ont été nécessaires pour déterminer les valeurs optimales du ou des paramètres de la méthode. Les tableaux 4.I et 4.II résument respectivement les résultats obtenus pour le problème du « OU » exclusif et celui de la reconnaissance de caractères manuscrits. Par convention, une *itération* représente un cycle complet au cours duquel toutes les données de l'ensemble d'apprentissage sont présentées une fois au réseau de neurones.

Mode d'apprentissage	Taux d'adaptation	Terme de moment	Nombre d'itérations
En-Ligne	1,1	-	8
Demi-Ligne	1,0	-	4
Hors-Ligne	0,03	-	140
En-Ligne	1,1	0,8	5
Demi-Ligne	1,0	0,95	3
Hors-Ligne	0,07	0,7	47

Table 4.I - Application de l'algorithme de rétro-propagation au problème du « OU » exclusif.

Pour les deux problèmes étudiés, le mode d'apprentissage hors-ligne apparaît comme étant, et de loin, le mode le plus défavorable. Bien qu'il implique moins de calculs par itération, puisque les poids sont alors moins fréquemment modifiés qu'à l'aide des deux autres méthodes, le nombre d'itérations nécessaires est tel que la durée globale de l'apprentissage est bien plus

élevée. C'est le mode d'apprentissage en demi-ligne qui semble ici être le plus efficace, en réalisant un bon compromis entre la qualité obtenue de l'estimation du gradient de l'erreur et le nombre de modifications apportées par itération aux poids synaptiques du réseau de neurones. L'utilisation d'un terme de moment, quant à elle, a permis d'accélérer l'apprentissage dans tous les cas.

Mode d'apprentissage	Taux d'adaptation	Terme de moment	Taux de Reconnaissance	Nombre d'itérations
En-Ligne	0,05	-	87,9 %	31
Demi-Ligne	0,05	-	87,7 %	25
Hors-Ligne	0,001	-	88,2 %	1061
En-Ligne	0,05	0,9	87,6 %	21
Demi-Ligne	0,05	0,5	87,2 %	18
Hors-Ligne	0,0005	0,9	87,7 %	366

Table 4.II - Application de l'algorithme de rétro-propagation au problème de la reconnaissance de caractères manuscrits.

Pour le problème de la reconnaissance de caractères manuscrits, aucune convergence de l'algorithme n'a pu être constatée dans le mode d'apprentissage hors-ligne, au départ de poids synaptiques initialisés à des valeurs aléatoires, en deçà de trois milliers d'itérations. Une phase préliminaire d'apprentissage des vecteurs de caractéristiques moyens de chaque classe a cependant permis de réduire fortement la durée de l'apprentissage dans ce cas. Cette phase préliminaire, très rapide puisque chaque classe n'est représentée que par un seul objet, a été systématiquement effectuée lors des tests des autres méthodes d'apprentissage hors-ligne, afin de pouvoir les comparer objectivement entre elles. L'application de cette phase préliminaire aux modes d'apprentissage en-ligne et en demi-ligne, n'a pas permis de constater une différence significative en termes de vitesse d'apprentissage, et s'avère donc inutile pour ces deux modes.

4.4.4 L'Algorithme de Sanossian & Evans

A l'issue de la phase de méta-optimisation, les valeurs des paramètres de l'algorithme de rétro-propagation sont celles qui conduisent au meilleur compromis entre les deux situations extrêmes suivantes, déjà évoquées précédemment:

- Lorsque, selon un poids synaptique, la pente de la fonction de coût, donnée par l'amplitude de la dérivée de cette dernière par rapport au poids considéré, est élevée, l'erreur est fort sensible à toute variation de ce poids. Il est alors judicieux de ne modifier celui-ci que légèrement, et donc d'utiliser une faible valeur du taux d'adaptation α.

- Si, au contraire, la pente de la fonction de coût est faible, l'erreur est peu sensible à une variation du poids concerné. Une correction importante pourrait alors lui être apportée, afin d'accélérer la convergence de l'algorithme d'apprentissage, ce qui conduit à utiliser dans ce cas une valeur élevée du taux d'apprentissage α.

Comme l'algorithme de rétro-propagation classique utilise une même valeur du taux d'apprentissage pour tous les poids synaptiques, ce dernier n'est optimal dans aucun des cas, et la convergence de l'algorithme demeure relativement lente. La solution proposée par Sanossian et Evans est de choisir une valeur du taux d'adaptation α, indépendamment pour chaque poids synaptique, et qui décroît lorsque l'amplitude du module de la dérivée de l'erreur par rapport au poids considéré augmente [Sanossian,91]. Ces valeurs du taux d'adaptation sont déterminées à l'avance et tabulées (*figure 4.4*).

Le nombre de valeurs distinctes que peut prendre le taux d'adaptation est *à priori* arbitraire, mais il ne faut pas perdre de vue qu'une phase de méta-optimisation demeure nécessaire pour chacune d'entre elles. Trois valeurs distinctes seulement ont donc été employées pour effectuer les tests, dont les résultats pour le problème du « OU » exclusif sont résumés au tableau 4.III.

Par rapport à l'algorithme de rétro-propagation classique, il apparaît une diminution du nombre d'itérations nécessaires. La durée globale de l'apprentissage se trouve également réduite, du fait que l'algorithme de Sanossian & Evans ne requiert pratiquement pas de calculs supplémentaires par itération. Les modes d'apprentissage en-ligne et en demi-ligne apparaissent

---

[Sanossian,91] **H. Sanossian & D. Evans**
An Acceleration Method for the Backpropagation Learning Algorithm
Proc. of the 4th International Conference On Neural Networks & Their Apllications, Nimes, France, novembre 1991

ici comme étant équivalents, et demeurent tous deux nettement plus rapides que le mode d'apprentissage hors-ligne. L'utilisation d'un terme de moment permet toujours d'accélérer la vitesse de convergence.

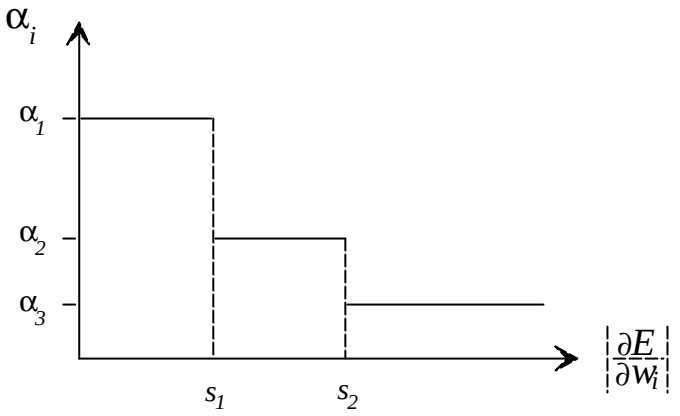


Figure 4.4 - Evolution du taux d'apprentissage en fonction du module de la dérivée de l'erreur.

Mode d'apprentissage	Taux d'adaptation (seuils d'utilisation)	Terme de moment	Nombre d'itérations
En-Ligne	2 / 1 / 0,5 (0,2 / 0,5)	-	3
Demi-Ligne	4 / 2 / 1 (0,5 / 1)	-	4
Hors-Ligne	0,65 / 0,2 / 0,02 (0,5 / 2)	-	83
En-Ligne	2 / 1 / 0,5 (0,1 / 0,5)	0,8	2
Demi-Ligne	5 / 3 / 2 (0,05 / 0,1)	0,9	2
Hors-Ligne	0,6 / 0,5 / 0,07 (0,2 / 0,5)	0,8	33

Table 4.III - Application de l'algorithme de Sanossian & Evans au problème du « OU » exclusif.

L'inconvénient majeur de cette méthode d'apprentissage, est que la phase de méta-optimisation des nombreux paramètres de l'algorithme requiert beaucoup de temps lorsque le problème étudié est de grandes dimensions. Ceci nous a empêché de l'appliquer au problème de la reconnaissance de caractères manuscrits.

4.4.5 La Méthode de l'Annulation de l'Erreur

Cette méthode d'apprentissage est basée sur le fait que la fonction de coût qui régit l'entraînement du réseau de neurones, définie aussi bien selon le critère des Moindres Carrés de l'Erreur que selon celui de l'Entropie Relative, s'annule en son minimum global. Ce dernier pourrait donc être déterminé en recherchant à annuler cette fonction de coût.

Soit l'équation différentielle suivante:

$$\frac{\partial x}{\partial \tau} = -x^{\nu} \quad (4.13)$$

où $0 < \nu < 1$.

Cette équation possède une solution unique[1], qui est:

$$x = \left(x_0^{1-\nu} - (1-\nu)(\tau - \tau_0)\right)^{\frac{1}{1-\nu}} \quad (4.14)$$

Le point d'équilibre $x = 0$ peut ainsi être atteint, au départ de n'importe quelles conditions initiales (x_0, τ_0), en un nombre fini d'itérations. L'apprentissage du réseau de neurones pourrait donc également s'effectuer en faisant de sorte que l'évolution de la fonction de coût suive une loi déterminée par l'équation (4.13) [Gosselin,93].

Lorsque les poids synaptiques sont modifiés, l'évolution que subit l'erreur est donnée, en se limitant aux termes du premier ordre, par:

[1] Cette solution peut aisément être déterminée en introduisant le changement de variable: $y = x^{1-\nu}$

[Gosselin, 93] **B. Gosselin**
Multilayer Perceptron and Uppercase Handwritten Characters Recognition
Proc. of the 2nd International Conference on Document Analysis and Recognition, pp 935-938, Tsukuba, Japon, octobre 1993

$$\Delta E = E(\tau+1) - E(\tau) = \nabla E^T \cdot \Delta W \quad (4.15)$$

où $\nabla E = \left[\frac{\partial E}{\partial w_1} \cdots \frac{\partial E}{\partial w_n}\right]^T$ est le vecteur des dérivées partielles premières de l'erreur par rapport à chacun des *n* poids synaptiques du réseau de neurones, et $\Delta W = \left[\Delta w_1 \cdots \Delta w_n\right]^T$ le vecteur des modifications apportées à ceux-ci.

Lorsque le taux d'adaptation utilisé est particulier à chaque poids synaptique, on a $\Delta w_i = -\alpha_i \frac{\partial E}{\partial w_i}$, et l'expression (4.15) devient:

$$\Delta E = -\sum_{i=1}^{n} \alpha_i \left(\frac{\partial E}{\partial w_i}\right)^2 \quad (4.16)$$

En posant:

$$\alpha_i = \frac{K}{\left(\frac{\partial E}{\partial w_i}\right)^2} \quad (4.17)$$

où K est une constante, il vient:

$$\Delta E = -n \cdot K \quad (4.18)$$

De manière à ce que l'évolution de l'erreur soit régie par une équation semblable à (4.13), la constante K doit valoir:

$$K = \frac{E^{\nu}}{n} \quad (4.19)$$

Il en résulte donc:

$$\alpha_i = \frac{E^{\nu}}{n\left(\frac{\partial E}{\partial w_i}\right)^2} \quad (4.20)$$

et:

$$\Delta w_i = \frac{-E^{\nu}}{n\left(\frac{\partial E}{\partial w_i}\right)} \quad (4.21)$$

Comme dans l'algorithme développé par Sanossian & Evans, un poids synaptique sera ici d'autant peu modifié que la sensibilité de la fonction de coût vis-à-vis de celui-ci sera élevée.

Toutefois, lorsque la pente devient extrêmement faible, la correction appliquée au poids synaptique devient très importante, et les termes de second ordre, négligés dans l'expression (4.15), ne sont plus négligeables. Ceci peut parfois provoquer une augmentation importante de la valeur de la fonction de coût, et une limitation doit donc être introduite. L'évolution des taux d'adaptation α_i suit alors une courbe semblable à celle illustrée à la *figure 4.5*.

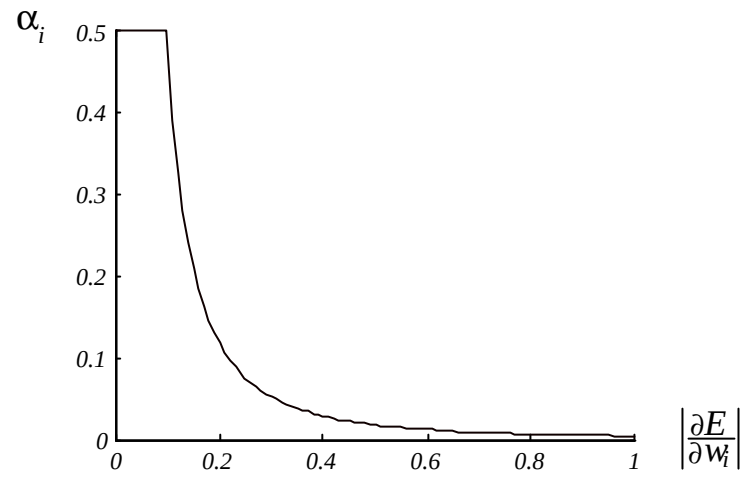


Figure 4.5 - Evolution du taux d'adaptation en fonction du module de la dérivée première de l'erreur.

La phase de méta-optimisation se résume, dans cette nouvelle méthode, à rechercher des valeurs opportunes pour deux paramètres seulement, qui sont: le coefficient de décroissance ν et la valeur maximale α_{max} du taux d'adaptation. La phase de méta-optimisation est ainsi nettement plus aisée que pour la méthode précédente, ce qui confère à l'algorithme développé ici un avantage non négligeable vis-à-vis de cette dernière. Ce nouvel algorithme a ainsi pu être appliqué aux deux problèmes de tests, dont les tableaux 4.IV et 4.V résument les résultats.

Aucune convergence n'a ici été constatée, en un nombre suffisamment restreint d'itérations, sans l'utilisation d'un terme de moment. En l'absence de ce dernier, en effet, de nombreuses oscillations de l'erreur se produisent, et conduisent à un très net ralentissement de l'apprentissage. Dans le cas du mode d'apprentissage en-ligne, l'utilisation d'un terme de moment n'a pas permis de limiter suffisamment ces oscillations, empêchant ainsi l'algorithme de converger. En outre, la convergence obtenue dans le mode d'apprentissage en demi-ligne a été relativement délicate à atteindre. En ce qui concerne le mode d'apprentissage hors-ligne, par contre, la vitesse de convergence s'avère être la plus grande de celles obtenues jusqu'à présent, pour le problème de

la reconnaissance de caractères manuscrits. Cette amélioration progressive du comportement de l'algorithme d'apprentissage, au fur et à mesure que les gradients instantanés de l'erreur sont de plus en plus cumulés, semble indiquer que cette méthode requiert la meilleure estimation possible du gradient ***global*** de l'erreur.

Mode d'apprentissage	Taux d'adaptation maximal	Taux de décroissance	Terme de moment	Nombre d'itérations
En-Ligne	-	-	-	non convergence
Demi-Ligne	10	0,5	0,95	12
Hors-Ligne	0,5	0,2	0,8	33

Table 4.IV - Application de l'algorithme de l'Annulation de l'Erreur au problème du « OU » exclusif.

Mode d'apprentissage	Taux d'adap-tation maximal	Taux de décroissance	Terme de moment	Taux de Reconnaissance	Nombre d'itérations
En-Ligne	-	-	-	-	non convergence
Demi-Ligne	5	0,3	0,8	82,2 %	19
Hors-Ligne	0,5	0,3	0,95	87,9 %	46

Table 4.V - Application de l'algorithme de l'Annulation de l'Erreur au problème de la reconnaissance de caractères manuscrits.

La *figure 4.6* représente l'évolution du taux d'adaptation d'un poids synaptique, en fonction du module de la dérivée de l'erreur par rapport à ce poids, dans le cas du problème du « OU » exclusif. La courbe supérieure n'est valable qu'en début d'apprentissage, lorsque l'erreur est encore importante, alors que la courbe inférieure est celle suivie en fin d'apprentissage, lorsque l'erreur devient faible. Sur ce graphique, ont également été représentées les valeurs expérimentales des taux d'adaptations déterminés lors de l'application de la méthode de Sanossian & Evans. L'existence d'une nette corrélation entre les valeurs des taux d'adaptation

employés pour les deux méthodes d'apprentissage peut être constatée. Celles que génère la méthode de l'Annulation de l'Erreur sont toutefois supérieures, pour la plupart des poids synaptiques.

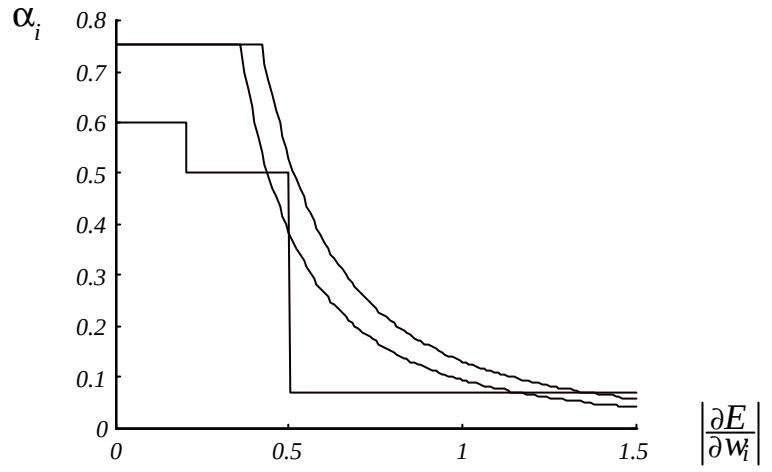


Figure 4.6 - Comparaison des valeurs des taux d'adaptation utilisées lors de l'application de la méthode de l'Annulation de l'Erreur et de celle de Sanossian & Evans au problème du « OU » exclusif.

4.4.6 L'Algorithme de Vogl

La méthode d'apprentissage développée par Vogl se rapproche plus de l'algorithme de rétro-propagation initial. Le taux d'adaptation des poids synaptiques est identique pour tous ceux-ci, mais peut lui-même être modifié d'une itération à l'autre [Vogl et al.,88]:

- tant que la valeur de la fonction de coût décroît, le taux d'adaptation est augmenté:

$$E(\tau) < E(\tau - 1) \qquad \alpha(\tau + 1) = \alpha(\tau) \cdot (1 + \varepsilon_a) \quad (4.22)$$

[Vogl et al.,88] **T.P. Vogl, J.K. Mangis, A.K. Rigler, W.T. Zink, & D.L. Alkon**
Accelerating the convergence of the backpropagation method
Biological Cybernetics, volume 59, pp 257-263, 1988

- il demeure par contre inchangé lorsque l'erreur n'augmente que légèrement:

$$E(\tau-1) \le E(\tau) < (1+\rho)\cdot E(\tau-1) \qquad \alpha(\tau+1) = \alpha(\tau) \quad (4.23)$$

- et il est diminué si l'erreur augmente de manière significative:

$$E(\tau-1)\cdot(1+\rho) \le E(\tau) \qquad \alpha(\tau+1) = \alpha(\tau)\cdot(1-\varepsilon_d) \quad (4.24)$$

Dans ce dernier cas, les poids synaptiques sont en outre réinitialisés à leur dernière valeur ayant permis d'obtenir une erreur qui n'a pas nécessité de diminution du taux d'apprentissage.

Cette méthode apparaît surtout intéressante par le fait que la valeur du taux d'adaptation se modifie automatiquement, tout au long de l'apprentissage, en fonction des circonstances rencontrées. Le choix de la valeur initiale du taux d'adaptation apparaît dès lors moins critique que précédemment, et une phase de méta-optimisation n'est *à priori* vraiment nécessaire que pour les trois paramètres principaux de cet algorithme, à savoir:

- le taux d'accroissement ε_a ;
- le taux de décroissance ε_d ;
- le seuil de tolérance de dépassement de l'erreur ρ.

Tous ces facteurs sont des réels positifs inférieurs à l'unité.

Mode d'apprentissage	Taux d'adaptation initial	$1+\varepsilon_a$	$1-\varepsilon_d$	$1+\rho$	Nombre d'itérations
En-Ligne	1,1	1,5	0,5	1,04	7
Demi-Ligne	1,0	1,05	0,5	1,04	4
Hors-Ligne	0,03	1,1	0,1	1,04	116

Table 4.VI - Application de l'algorithme de Vogl au problème du « OU » exclusif.

Mode d'apprentissage	Taux d'adaptation initial	$1+\varepsilon_a$	$1-\varepsilon_d$	$1+\rho$	Taux de Reconnaissance	Nombre d'itérations
En-Ligne	0,02	1,1	0,7	1,04	88,2 %	12
Demi-Ligne	0.03	1,05	0,5	1,04	88,2 %	14
Hors-Ligne	0,001	1,05	0,7	1,04	87,9 %	494

Table 4.VII - Application de l'algorithme de Vogl au problème de la reconnaissance de caractères manuscrits.

C'est le mode d'utilisation en-ligne de l'algorithme d'apprentissage qui apparaît à nouveau comme étant le plus intéressant. En outre, la modification du taux d'adaptation au fur et à mesure de l'apprentissage permet de mieux suivre l'évolution de ce dernier, et d'atteindre ainsi la vitesse de convergence la plus élevée de toutes celles relevées au cours de ces tests. Le recours à un terme de moment n'a permis, dans aucun des cas, d'améliorer la rapidité de convergence de l'algorithme, et s'avère donc ici inutile. Les paramètres de l'algorithme ayant un effet « exponentiel », une valeur non adéquate de ceux-ci est rapidement détectée, et la phase de méta-optimisation est donc ici relativement aisée à mener.

4.4.7 Conclusions

La méthode qui se présente comme étant la meilleure solution potentielle, pour des problèmes qui ne pourraient être résolus qu'à l'aide d'un mode d'apprentissage hors-ligne, est celle de l'Annulation de l'Erreur. Au cours de tous les essais effectués ici, ce sont toutefois les modes d'apprentissage en-ligne ou en demi-ligne qui se sont révélés être les plus performants. Bien que les poids synaptiques soient moins fréquemment modifiés, au cours d'une itération, au moyen du mode d'apprentissage en demi-ligne qu'au moyen du mode d'apprentissage en-ligne, l'écart en volume de calcul à effectuer est peu significatif, et ces deux modes peuvent être considérés comme étant équivalents. L'algorithme qui apparaît alors être le plus avantageux est celui proposé par Vogl. Il permet une mise au point automatique du taux d'adaptation tout au long de la phase d'apprentissage, et est en outre l'algorithme dont la phase de méta-optimisation s'avère la moins critique pour les performances de l'algorithme.

4.5 Comparaisons des Performances de Divers Classificateurs

Les divers classificateurs présentés au chapitre précédent ont été appliqués au même problème de reconnaissance de caractères que le perceptron multicouches. Afin de pouvoir considérer leurs performances de la manière la plus objective possible, les caractères de l'ensemble de validation ont été soumis, épurés de toute information contextuelle, à des tests de reconnaissance visuels auprès de multiples personnes. Le taux de reconnaissance humain observé sur ces caractères a alors été, en moyenne, de 96 %. Le tableau 4.VIII reprend les meilleures qualités de reconnaissance offertes par chaque modèle de classificateur. Le temps moyen de reconnaissance annoncé l'est à titre indicatif, et est celui qui a été observé lorsque chaque classificateur est implanté de manière à minimiser le volume de calcul requis en phase de reconnaissance[2]. La *figure 4.7* situe les différents classificateurs, en fonction de leurs performances respectives.

Les performances atteintes par le perceptron à simple couche montrent toute l'importance d'un apprentissage discriminant, puisque ce modèle de classificateur, qui ne peut générer que des frontières de décision linéaires, atteint pourtant de meilleurs résultats que les classificateurs Quadratique et Gaussien. La même remarque peut être formulée pour la carte auto-organisatrice de Kohonen. Tout comme le perceptron multicouches, cette dernière a subi plusieurs entraînements, de manière à déterminer le nombre le plus adéquat de cellules à utiliser [Accaino,92]. La meilleure des cartes ayant subi un apprentissage supervisé comportant, en fin de compte, moins de cellules que la meilleure de celles n'ayant pas subi d'apprentissage supervisé, elle offre une vitesse de reconnaissance supérieure.

[2] Le volume de calcul pourrait toutefois être plus réduit pour les classificateurs de type «k Plus Proches Voisins », ainsi que pour ceux basés sur une carte auto-organisatrice, en tenant compte, par exemple, des propriétés triangulaires de la distance Euclidienne. Ceci permettrait d'éviter de procéder à une recherche exhaustive de la plus courte distance entre le vecteur d'entrée et tous ceux des prototypes construits à la suite de l'apprentissage.

[Accaino,92] **S. Accaino**
Reconnaissance de Caractères Manuscrits Majuscules par Réseau de Kohonen
Travail de Fin d'Etudes, Faculté Polytechnique de Mons, juin 1992

Classificateur	Taux de Reconnaissance	Temps de Reconnaissance Moyen
Euclidien	71,3 %	2,74 ms
Quadratique	72,0 %	5,37 ms
Gaussien	73,3 %	5,41 ms
1 Plus Proche Voisin	80,9 %	590,3 ms
3 Plus Proches Voisins	80,0 %	590,4 ms
5 Plus Proches Voisins	81,0 %	590,4 ms
8 Plus Proches Voisins	81,4 %	590,5 ms
Carte Auto-Organisatrice (non supervisé)	76,7 %	26,6 ms
Carte Auto-Organisatrice (supervisé)	84,3 %	23,3 ms
Perceptron	77,4 %	2,81 ms
Perceptron Multicouches	88,2 %	5,06 ms
Homme	96,0 %	-

Table 4.VIII - Comparaison des performances de divers classificateurs.
La durée moyenne de reconnaissance a été mesurée, à titre de comparaison uniquement,
sur un P.C. équipé d'un processeur 486DX porté à 66 MHz.

Le temps de calcul élevé ainsi que la grande quantité de mémoire exigés par les classificateurs du type « k Plus Proche Voisin » apparaissent, par contre, comme pouvant compromettre leur mise en application pratique. La carte auto-organisatrice de Kohonen, lorsqu'elle subit un apprentissage supervisé, permet un taux de reconnaissance supérieur à celui qu'offrent ces derniers, tout en permettant de réduire le nombre de prototypes de référence pour chaque classe, et apparaît donc comme une alternative intéressante.

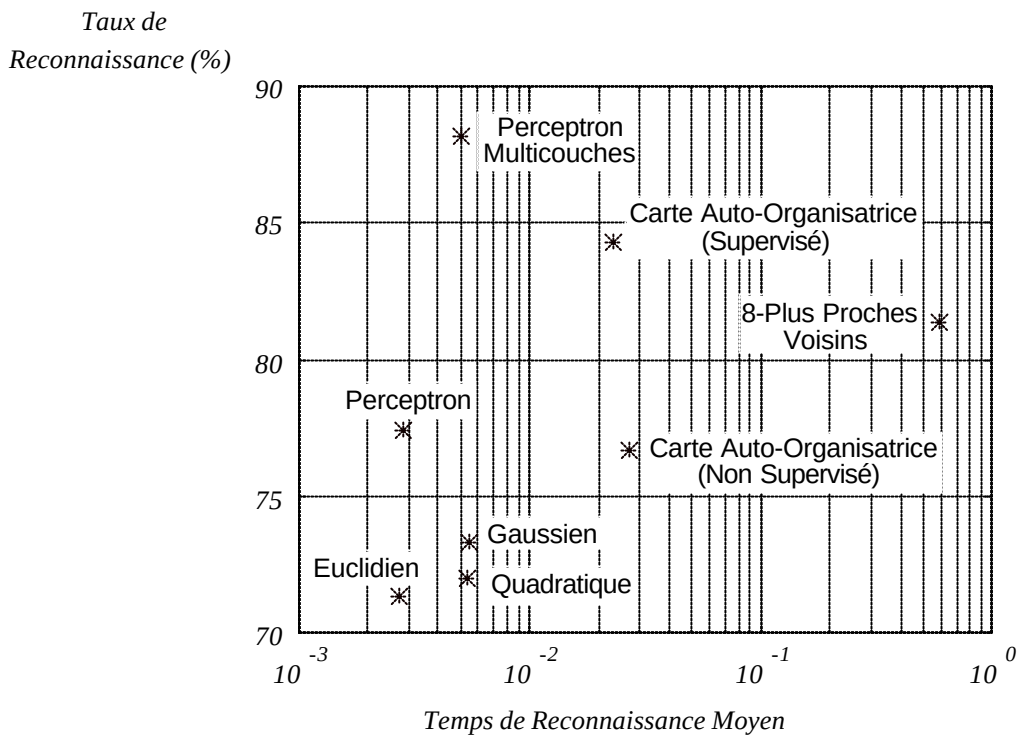


Figure 4.7 - Positionnement des différents classificateurs, en fonction de leurs performances respectives. La durée moyenne de reconnaissance est tracée selon une échelle logarithmique.

Le meilleur résultat de reconnaissance a été atteint à l'aide d'un perceptron multicouches, ce qui fait plus que confirmer l'aptitude de ce dernier à agir en tant que classificateur. La puissance de calcul qu'il requiert en phase de reconnaissance, apparaît en outre relativement restreinte, puisque la vitesse de reconnaissance obtenue ici est comparable à celle mesurée du classificateur Gaussien. De manière à atteindre une vitesse de reconnaissance aussi élevée que celle requise par le simple perceptron, un perceptron multicouches ne comportant que 15 unités cachées a également été entraîné, et a permis d'obtenir un taux de reconnaissance moyen de **84,1%** sur l'ensemble des caractères de test. Cet excellent résultat, obtenu pour un volume de calcul des plus réduit, met également en évidence l'intérêt du perceptron multicouches vis-à-vis des autres modèles de classificateurs.

L'inconvénient principal du perceptron multicouches demeure que le temps nécessaire à son entraînement est, malgré l'utilisation d'algorithmes d'apprentissage accéléré, relativement élevé, car il nécessite plusieurs phases d'apprentissage afin de déterminer le nombre le plus adéquat de ses unités cachées. D'un point de vue pratique cependant, la reconnaissance de caractères manuscrits dans un contexte multiscripteur ne requiert l'entraînement d'un perceptron

multicouches qu'une et une seule fois pour toutes, et cette phase ne peut dès lors pas être considérée comme étant critique.

4.6 Résumé

Divers conseils destinés à faciliter l'apprentissage du perceptron multicouches ont d'abord été présentés. Cette phase demeurant relativement lente, plusieurs méthodes visant à l'accélérer ont été ensuite décrites et testées. Les essais d'entraînement du perceptron multicouches à la classification, dans un contexte multiscripteur et en l'absence de contraintes d'écriture, des 26 lettres latines majuscules montrent que les modes d'apprentissage en-ligne ou en demi-ligne sont bien plus rapides que le mode d'apprentissage hors-ligne. L'algorithme proposé par Vogl s'avère dans ces cas très efficace: le taux d'adaptation est lui-même modifié automatiquement au cours de l'apprentissage, et prend ainsi une valeur optimale quelles que soient les circonstances. En outre, la phase de méta-optimisation est alors fortement facilitée, ce qui confère un avantage non négligeable à cet algorithme vis-à-vis des autres.

D'autres modèles de classificateurs ont finalement été comparés au perceptron multicouches, sur base du même problème de reconnaissance de caractères. Les capacités du perceptron multicouches à générer des frontières de décision complexes entre les classes se sont avérées lui donner un avantage considérable, et lui ont permis d'offrir le taux de reconnaissance le plus élevé. Cette performance est en outre atteinte au prix d'un temps de calcul, en phase de reconnaissance, bien plus faible que celui requis par le classificateur qui offre la qualité de reconnaissance la plus proche, à savoir la carte Auto-Organisatrice de Kohonen.

Chapitre 5

L'Extraction de Caractéristiques

5.1 Introduction

Afin d'obtenir une bonne qualité de représentation des caractères, et donc de garantir qu'une reconnaissance efficace puisse être menée, l'acquisition des caractères doit s'effectuer à relativement haute[1] résolution. Un caractère digitalisé est alors représenté par une matrice de dimensions élevées, ce qui peut rendre complexe l'entraînement des paramètres du réseau de neurones. En outre, bien que les réseaux de neurones artificiels de type perceptron multicouches, possèdent la propriété de pouvoir générer des régions de décision complexes, la façon dont les données leur sont présentées, peut affecter leurs performances de manière significative. Une procédure efficace de pré-traitement des caractères, capable d'extraire des primitives robustes et résistantes au bruit, est souvent requise pour un système de reconnaissance basé sur un classificateur conventionnel, mais est donc également souhaitable lorsque le classificateur est un réseau de neurones.

L'extraction de primitives est une étape de grande importance: si elle est mal conçue, il sera difficile, voire impossible, d'effectuer une reconnaissance efficace. Pour la reconnaissance de caractères manuscrits, les primitives recherchées doivent présenter une certaine invariance *géométrique* d'une part, et une certaine invariance *statistique* d'autre part. L'invariance géométrique signifie une tolérance vis-à-vis des opérateurs de translation, de rotation, et de

[1] « haute » signifie ici de l'ordre de deux à trois cents points par pouce, pour une écriture courante.

changement d'échelle, tandis que l'invariance statistique signifie une tolérance vis-à-vis du bruit inévitablement présent sur chaque caractère.

La section 5.2 décrit quelques pré-traitements qui peuvent être appliqués aux caractères, indépendamment ou successivement, afin d'en améliorer la qualité de représentation. Plusieurs méthodes d'extraction de primitives sont ensuite présentées et discutées à la section 5.3. La section 5.4, enfin, propose une méthode de sélection de caractéristiques, qui permet d'élaguer le vecteur de primitives de base afin de n'en conserver que les composantes les plus discriminantes. La section 5.5 présente alors les résultats obtenus.

5.2 Opérations de Pré-Traitements

5.2.1 Le Filtrage Bidimensionnel

Un premier traitement qui peut être effectué, en vue d'améliorer la qualité de l'image qui représente un caractère, est un filtrage bidimensionnel. Un filtre passe-bas, par exemple, permettra de limiter les discontinuités du trait d'un caractère, dues à la présence d'un défaut d'encrage insuffisant (*figure 5.1*). Un tel traitement ne peut toutefois être appliqué que restrictivement, et le choix des caractéristiques de celui-ci dépend aussi bien de la qualité d'impression du document original (type de défauts d'encrage, couleur de ce dernier), que des conditions d'acquisition (quantité, orientation, et couleur de l'éclairage).

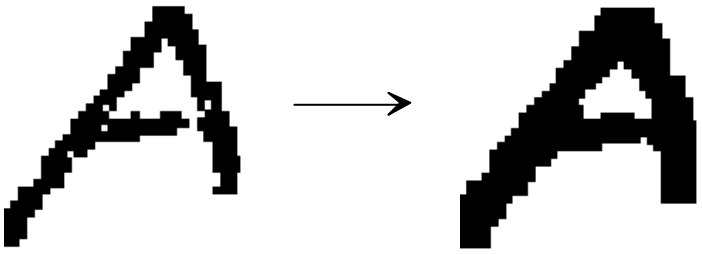


Figure 5.1 - Restauration de la continuité du trait d'un caractère, grâce à l'utilisation d'un filtre passe-bas.

L'action d'un filtre dans le domaine des fréquences spatiales est donnée par le produit du signal et de la réponse fréquencielle du filtre:

$$Y(f,g) = H(f,g) \cdot X(f,g), \text{ (5.1)}$$

la forme particulière de la réponse fréquencielle utilisée $H(f,g)$ déterminant les atténuations ou les amplifications des composantes aux diverses fréquences f et g. La relation équivalente d'un filtre numérique dans le domaine des signaux est le produit de convolution [Kunt,93]:

$$y(k,l) = \sum_{k'=-\infty}^{+\infty} \sum_{l'=-\infty}^{+\infty} h(k-k', l-l') \cdot x(k', l') \text{ (5.2)}$$

Très souvent, la réponse fréquencielle idéale $H_i(f,g)$ contient des transitions rapides entre les bandes passantes et les bandes bloquées. La réponse impulsionnelle correspondante $h_i(k,l)$, obtenue par transformation de Fourier inverse, est alors forcément à étendue infinie, et doit donc être tronquée en pratique. La réponse impulsionnelle idéale est alors multipliée par une fonction fenêtre appropriée afin de limiter l'étendue de la réponse:

$$h(k,l) = h_i(k,l) \cdot w(k,l) \text{ (5.3)}$$

où $w(k,l)$ est la fonction fenêtre.

La recherche de la fenêtre qui minimise l'écart entre $H_i(f,g)$ et $H(f,g)$ peut se faire en se basant sur de « bonnes » fenêtres unidimensionnelles [Huang,78]. Le changement de variable $\omega = 2\pi\sqrt{f^2 + g^2}$ permet par exemple de réduire un filtre bidimensionnel à symétrie circulaire à un filtre unidimensionnel. Ayant déterminé le filtre pour la variable ω, il suffit d'effectuer la transformation inverse pour obtenir le filtre bidimensionnel.

[Kunt,93] **M.Kunt, G. Granlund, & M. Kocher**
Traitement Numérique des Images
Presses Polytechniques et Universitaires Romandes, Collection Electricité, 1993

[Huang,78] **T.S. Huang**
Two-Dimensional Windows
Two-Dimensional Digital Signal Processing, pp 136-138, S.K. Mitra & M.P. Ekstrom Edts, Dowden, Hutchingon & Ross, Inc., 1978

La fenêtre la plus aisée à mettre en pratique demeure toutefois la fenêtre carrée. Le filtrage revient dans ce cas à effectuer le produit de convolution suivant:

$$y(k,l) = \sum_{k'=-a}^{+a} \sum_{l'=-a}^{+a} h(k-k', l-l') x(k', l') \quad (5.4)$$

où a est la demi-largeur de la fenêtre.

La réponse impulsionnelle tronquée du filtre fournit une matrice de pondération, avec laquelle il suffit de balayer l'image point par point pour en effectuer le filtrage (*figure 5.2*). Même lorsque l'image initiale est binaire, l'image résultante contient souvent plusieurs niveaux de gris, et une procédure de binarisation s'avère nécessaire chaque fois que l'on désire retrouver une représentation binaire du caractère. Une grande attention doit alors être apportée au choix de la valeur du seuil de binarisation, afin d'éviter l'élimination des améliorations apportées par le filtrage.

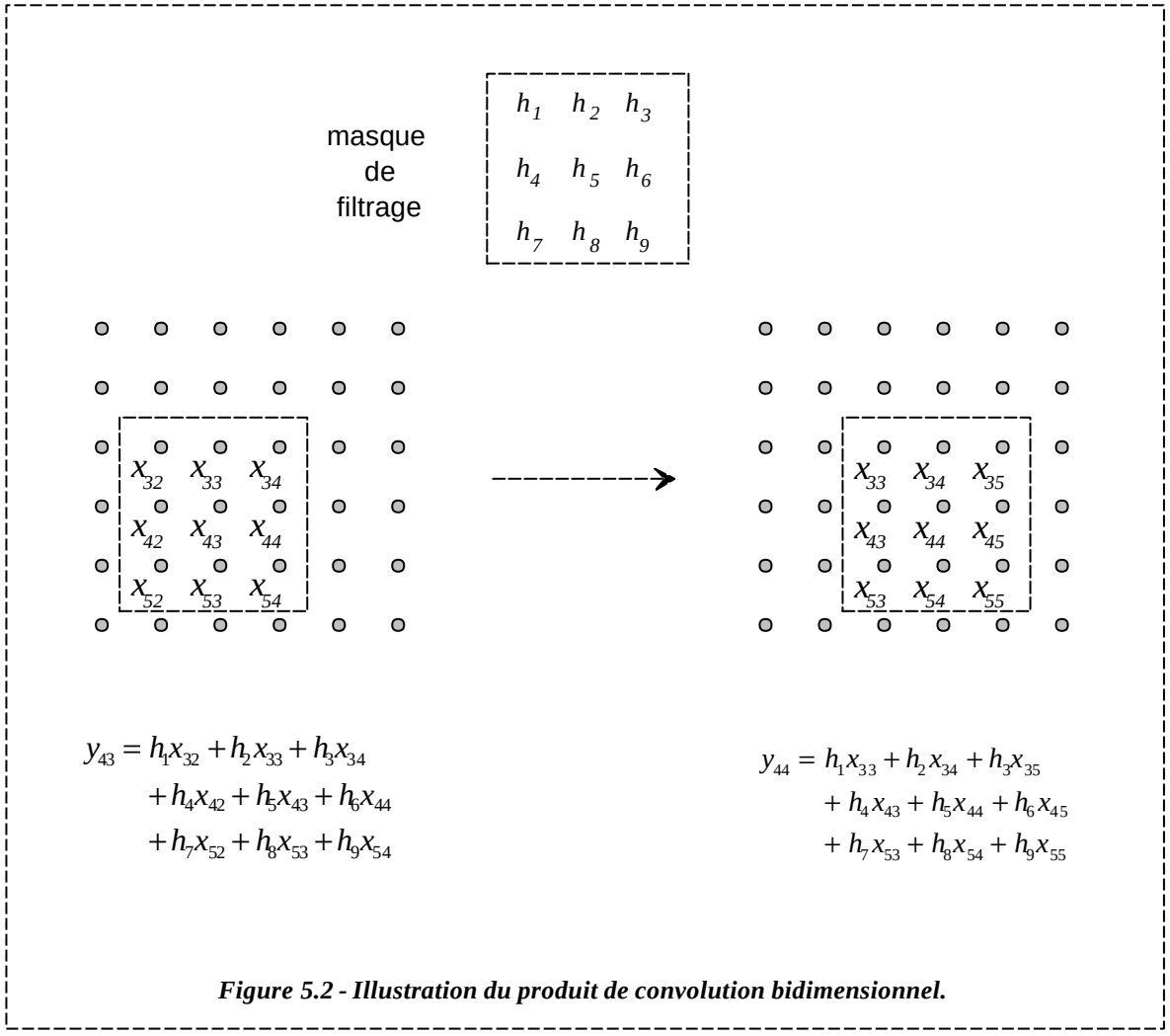


Figure 5.2 - Illustration du produit de convolution bidimensionnel.

Pour des raisons de temps de calcul, les dimensions de la fenêtre sont souvent réduites à 3×3 pixels. Des exemples de filtres passe-bas et passe-haut typiques sont repris à la *figure 5.3*. Ces filtres sont normalisés de manière à ce que la somme de tous les coefficients de pondération soit égale à l'unité.

1/9 1/9 1/9	1/10 1/10 1/10	1/16 1/8 1/16
1/9 1/9 1/9	1/10 1/5 1/10	1/8 1/4 1/8
1/9 1/9 1/9	1/10 1/10 1/10	1/16 1/8 1/16

(a) filtres passe-bas

–1 –1 –1	0 –1 0	1 –2 1
–1 9 –1	–1 5 –1	–2 5 –2
–1 –1 –1	0 –1 0	1 –2 1

(b) filtres passe-haut

Figure 5.3 - Exemples de filtres bidimensionnels.

Un problème de filtrage se pose pour les pixels qui sont situés sur le pourtour de l'image à traiter. En effet, la fenêtre de pondération centrée sur ceux-ci fait alors intervenir des valeurs inconnues, de pixels situés en dehors des limites l'image (*figure 5.4*). Cette dernière peut cependant être considérée comme étant entourée de pixels fictifs d'intensité nulle, puisqu'un caractère est censé être entièrement inclus dans l'image représentée dans l'ordinateur. Grâce à cette simple convention, le filtrage d'un caractère est effectué avec une perte minimale d'information.

Si les méthodes de filtrage bidimensionnel permettent parfois d'améliorer la qualité de représentation des caractères, un soin suffisant apporté à l'écriture ainsi que de bonnes conditions d'acquisition, lorsqu'il est possible de les réaliser, demeurent malgré tout, les meilleurs garants de l'obtention de bonnes performances de reconnaissance.

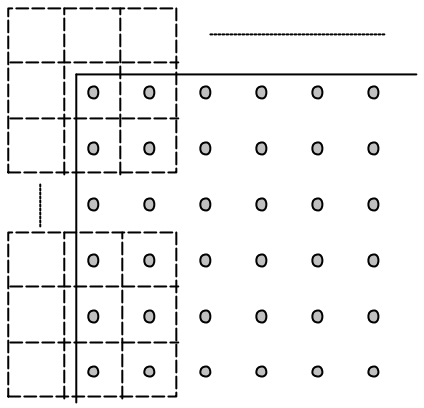


Figure 5.4 - Le problème du filtrage aux abords de l'image.

5.2.2 La Squelettisation

La procédure de squelettisation s'effectue sur une image binaire, et a pour but de réduire l'épaisseur du tracé d'un caractère à un pixel seulement, tout en conservant la continuité de celui-ci. Le principe de cette procédure est d'effectuer une succession d'opérations d'érosion conditionnelle, jusqu'à ce que le but recherché soit atteint [Impedevo,91] . Une opération d'érosion consiste à éliminer dans une image tous les pixels d'intensité non nulle qui sont adjacents aux pixels de l'arrière-plan (*figure 5.5*) [Niblack,86]. L'érosion conditionnelle signifie qu'un pixel d'intensité non nulle ne peut être éliminé que si, une fonction logique déterminée des valeurs des pixels qui l'entourent, est vérifiée. Cette fonction logique a pour rôle de vérifier la participation ou non du pixel central au maintien de la continuité du tracé.

La procédure de squelettisation consiste, sur base de ces principes, à balayer l'image, en passes successives, au moyen d'une fenêtre de dimension 3×3 (*figure 5.6*). Le pixel du centre de cette fenêtre prend alors une valeur déterminée par une fonction logique de l'ensemble des 9 points de la fenêtre. Le balayage de l'image s'effectue de haut en bas et de gauche à droite, et le processus se répète autant de fois que nécessaire. L'arrêt se produit lorsque l'ensemble de la matrice qui contient le caractère a été balayée sans qu'aucun nouveau pixel n'ait pu être éliminé.

[Impedevo,91] **S.Impedevo, L.Ottaviano, & S. Occhingro**
Optical Character Recognition - A Survey
Int. Journal of Pattern Recognition and Artificial Intelligence, 1991

[Niblack,86] **W. Niblack**
An Introduction to Digital Image Processing
Prentice-Hall International (UK) Ltd, 1986

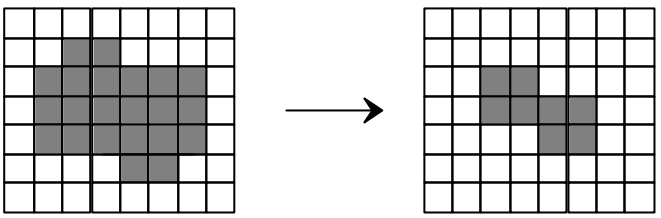


Figure 5.5 - Erosion d'une image.

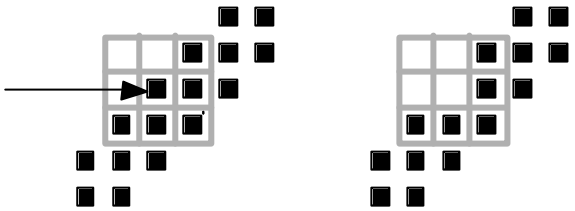


Figure 5.6 - Fenêtre d'érosion conditionnelle.

La fonction logique d'élimination des pixels doit être établie en examinant cas par cas les 256 (2^8) situations distinctes qui peuvent se présenter, en fonction des valeurs binaires des 8 pixels qui entourent le pixel central de la fenêtre. Ce dernier doit évidemment être d'intensité non nulle pour pouvoir être éventuellement éliminé, et la fonction logique n'est simplement pas évaluée dans le cas contraire. Diverses contraintes régissent la conception de cette fonction logique. Comme l'objectif est d'obtenir un tracé final d'épaisseur unique aussi fidèle que possible au tracé initial, il est souhaitable que l'élimination des pixels se fasse de manière symétrique, de façon à déterminer la ligne médiane de ce dernier. De plus, le résultat de la procédure de squelettisation doit être isotropique, c'est-à-dire invariant aux rotations. Enfin, la vitesse d'exécution de l'ensemble de la procédure doit être la plus élevée possible.

C'est sur base de l'ensemble de ces critères que la fonction logique de squelettisation a été élaborée. Elle est décrite à l'annexe B.

Le problème qui se pose aux abords d'une image lors d'une opération de filtrage apparaît ici également, puisque la fonction logique fait alors intervenir des valeurs de pixels inexistants.

L'adoption de la même convention que précédemment permet toutefois d'effectuer la squelettisation du caractère sans que la dimension de l'image ne soit réduite.

La *figure 5.7* illustre un exemple de résultat de l'opération de squelettisation.

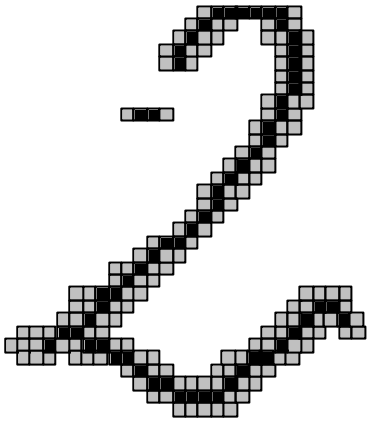


Figure 5.7 - Chiffre « 2 » et son squelette.

5.2.3 La Normalisation des Caractères

En fonction de la taille initiale des caractères sur le document original d'une part, et de la résolution employée par le périphérique de digitalisation d'autre part, les dimensions de l'image qui représente un caractère dans l'ordinateur sont fortement variables. Afin d'éviter que le système de reconnaissance soit perturbé par cette dispersion de grandeurs, l'image d'un caractère peut être ramenée à des dimensions normalisées.

Cette normalisation se fait en ré-échantillonnant l'image des caractères. Les points d'échantillonnage correspondent chacun à un pixel de l'image normalisée; leurs coordonnées dans l'image initiale du caractère sont régulièrement espacées, et calculées de manière à recouvrir celle-ci entièrement selon sa plus grande dimension (*figure 5.8*). Le calcul de la valeur que prend un point d'échantillonnage diffère toutefois suivant que les dimensions initiales du caractère, soit sa largeur et sa hauteur, sont supérieures ou inférieures aux dimensions de normalisation.

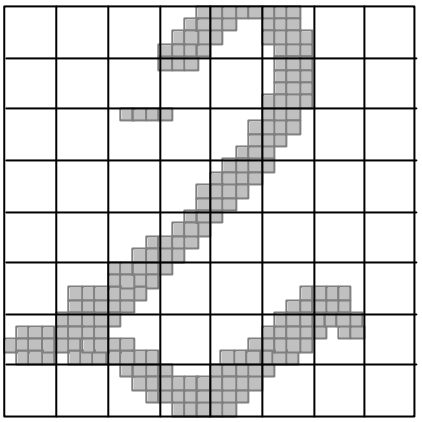


Figure 5.8 - Illustration de la procédure de normalisation d'un caractère.

Lorsqu'au moins une des deux dimensions de l'image qui représente un caractère, est supérieure à la dimension standard, celle pour laquelle l'écart est le plus élevé détermine l'amplitude de la contraction à effectuer. L'espace moyen entre deux points d'échantillonnage est calculé de manière à ce que, selon cette direction, la grille recouvre entièrement l'image du caractère. Les coordonnées calculées des points d'échantillonnage sont pratiquement toujours des valeurs réelles, et doivent donc être arrondies aux valeurs entières les plus proches (*figure 5.8*). Afin de conserver l'aspect initial du caractère, la même valeur d'intervalle d'échantillonnage est utilisée selon l'autre direction.

La valeur arithmétique moyenne de l'intensité de tous les pixels de l'image initiale, qui sont compris dans chaque zone ainsi délimitée, est calculée. Lorsque l'on travaille en niveaux de gris, cette valeur moyenne est attribuée au point d'échantillonnage correspondant. Si, par contre, l'on désire obtenir une image binaire, le point d'échantillonnage prend une valeur d'intensité unitaire dès que la valeur moyenne calculée est différente de zéro.

Lorsque les deux dimensions du caractère sont inférieures aux dimensions de normalisation, c'est celle pour laquelle l'écart est le plus faible qui détermine l'amplitude de la dilatation à effectuer. La largeur moyenne de l'intervalle d'échantillonnage est alors calculée de manière à recouvrir entièrement le caractère selon cette direction. Toujours de manière à conserver l'aspect initial du caractère, le même rapport de dilatation sera ensuite utilisé selon chacune des directions. Comme l'espace qui sépare deux points d'échantillonnage est ici, en moyenne, inférieur à un pixel, ces derniers prennent simplement la valeur du pixel dont les coordonnées entières sont les plus proches de ses coordonnées réelles calculées.

L'inconvénient de cette méthode de normalisation est que beaucoup de pixels de l'image normalisée n'apportent parfois aucune information (*figure 5.9*).

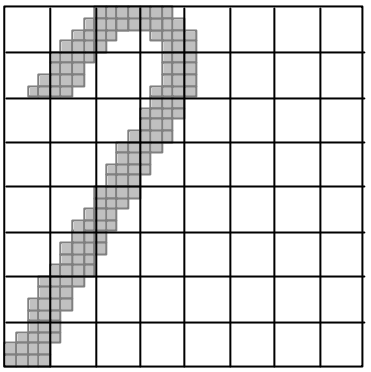


Figure 5.9 - Mise en évidence d'un défaut de la première méthode de normalisation: beaucoup de variables n'apportent aucune information.

Une autre méthode de normalisation qui peut être plus efficace consiste à utiliser une valeur distincte de l'intervalle d'échantillonnage pour chaque direction, de manière à ce que le caractère soit entièrement recouvert par la grille d'échantillonnage selon chacune d'entre elles (*figure 5.10*). Cette manière de procéder implique, toutefois, une perte d'information quant à l'aspect initial des caractères, ce qui pourrait compromettre la reconnaissance de certains d'entre eux. Pour éviter ce problème, le rapport d'aspect initial du caractère, c'est-à-dire le rapport entre sa largeur et sa hauteur, peut être calculé et inclus dans le vecteur de primitives.

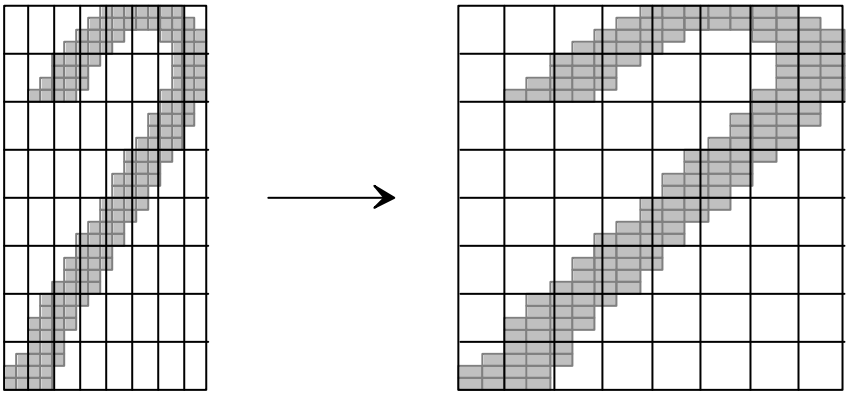


Figure 5.10 - Seconde méthode de normalisation.

5.3 Méthodes d'Extraction de Primitives

5.3.1 Introduction

Afin de pouvoir comparer entre eux les résultats que permettent d'atteindre les diverses méthodes d'extraction de primitives présentées ici, les tests ont été effectués sur des caractères manuscrits extraits d'une même base de données, référencée ETL-3 [ETL-3,93]. Celle-ci contient 200 séries de caractères manuscrits, écrites par autant de personnes distinctes, et comprenant 48 classes: les 26 lettres latines de A à Z, les 10 chiffres 0 à 9, ainsi que 12 symboles tels que « - », « (» , «) », « * », « + », « = », etc. La taille originale de la matrice qui représente un caractère, est de 64×64 pixels, soit 4096 valeurs. La moitié de la base de données a été utilisée pour effectuer l'entraînement d'un perceptron multicouches en classificateur, alors que l'autre moitié a servi pour effectuer les tests.

Bien que manuscrits, les caractères de cette base de données sont issus d'écritures très appliquées et présentant très peu de dispersion de styles. Ceci en fait une base de données qui ne permet pas d'entraîner efficacement un système de reconnaissance en vue d'une utilisation pratique courante, mais, dans la mesure où cette même base de données a également été utilisée par d'autres auteurs, elle nous permettra d'effectuer une comparaison objective entre les procédures d'extraction de primitives développées ici et d'autres méthodes.

5.3.2 La Méthode des Pixels Moyennés

La procédure de normalisation des caractères permet très facilement de réduire la dimension de représentation de ceux-ci, puisqu'il suffit de diminuer les valeurs des dimensions de normalisation. Les caractères sont alors presque systématiquement contractés, plusieurs pixels de l'image originale étant remplacés par leur valeur arithmétique moyenne. Lorsque les dimensions de normalisation deviennent relativement faibles, il est préférable, afin de minimiser la perte d'information, de conserver, après réduction, une image en niveaux de gris, plutôt que binaire. Les dimensions de normalisation doivent dans tous les cas demeurer suffisamment élevées, pour éviter la perte d'information discriminante.

[ETL-3,93] **ETL-3**
Character Database
Image Understanding Section, Electrotechnical Laboratory 1-1-4, Umezono, Tsukuba, Ibaraki, 305, Japan

La table 5.I reprend les résultats de reconnaissance obtenus sur l'ensemble de test pour chacune des méthodes de normalisation décrites à la section 5.2.3, et pour différentes valeurs des dimensions de représentation.

Méthode de Normalisation	Dimensions de Normalisation: Largeur/Hauteur	Nombre Total de Primitives	Taux de Reconnaissance
Première	10/10	100	91,4 %
Première	8/8	64	93,2 %
Première	7/7	49	92,5 %
Seconde	10/10	101	93,3 %
Seconde	8/8	65	94,1 %
Seconde	7/7	50	94,2 %

Table 5.I - Résultats de reconnaissance obtenus pour la méthode des Pixels Moyennés.

Dans tous les cas, c'est un perceptron multicouches comprenant 30 neurones en couche cachée, qui a permis d'obtenir les meilleurs taux de reconnaissance. Ces derniers sont en outre systématiquement plus élevés lorsque la seconde méthode de normalisation est utilisée, ce qui confirme que la seconde méthode de normalisation permet une meilleure représentation des caractères.

5.3.3 La Vectorisation

5.3.3.1 Principe de la Méthode

La vectorisation d'un caractère est une procédure visant à représenter celui-ci par un ensemble de segments de droite plutôt que par un ensemble de pixels. Cette représentation vectorielle, qui substitue deux paires de coordonnées à un ensemble de pixels, permet des gains considérables d'espace et de temps.

Un algorithme de vectorisation a été développé par d'Acierno et ses coéquipiers pour la reconnaissance de caractères typographiques [d'Acierno,91]. Le codage des segments s'effectuait selon leur taille (grand, moyen, petit), leur orientation (est, sud-est, ...), ainsi que leur position relative les uns par rapport aux autres (gauche, droit, haut, bas, croisement). Si cette méthode très simple s'est révélée être suffisante pour représenter des caractères typographiques, cela n'a pas été le cas pour des caractères manuscrits. La dispersion des styles d'écriture de ces derniers est en effet élevée, et leur représentation vectorielle doit, par conséquent, être beaucoup plus fidèle à la représentation originale pour permettre une reconnaissance efficace.

Kondo a, quant à lui, proposé de représenter des chiffres manuscrits par un nombre constant de segments de droite de mêmes dimensions [Kondo,86]. Seul l'angle que fait chacun des segments avec l'horizontale est alors codé. Cette méthode a toutefois été établie pour des chiffres manuscrits provenant des Etats-Unis d'Amérique, où la dispersion de styles d'écriture est beaucoup plus faible que celle constatée dans nos contrées. La même remarque que précédemment se révèle ainsi être également d'application ici, d'autant plus que l'extension de cette méthode aux lettres latines paraît être encore plus délicate.

La méthode de vectorisation développée par Vallée consiste à calculer des droites de régression à partir des pixels qui représentent le squelette des caractères [Vallée,93]. Les segments sont codés au moyen des points d'intersection des droites de régression, ainsi que par la valeur des courbures successives de la suite de segments. C'est cette dernière approche qui nous a semblé être la plus intéressante.

Un premier problème qui se pose cependant ici, est l'indisponibilité de l'historique du tracé de l'écriture, et une estimation de celui-ci devra donc être effectuée à partir de l'image brute des caractères. Le second problème qui se pose, est que les points d'intersection des droites de régression peuvent parfois se situer relativement loin du tracé d'un caractère, provoquant alors la perte de certaines caractéristiques essentielles à la reconnaissance. La méthode mise au point par

---

[d'Acierno,91] **A.d'Acierno, C.De Stefano, & M. Vento**
A Structural Character Recognition Method Using Neural Network
Proc. First Int. Conf. Document Analysis and Recognition, pp 803-811, Saint-Malo, France, octobre 1991

[Kondo,86] **S. Kondo & B. Attachoo**
Model of Handwriting Process and its Analysis
Proc. of the 8th Int. Conf. on Pattern Recognition, Vol1., pp 562-565, Paris, octobre 1986

[Vallée,93] **T. Vallée, R. Mullot, & J. Labiche**
Polygonalisation de tracé et extraction de primitives pour la reconnaissance en ligne de mots manuscrits
Proc. 14ème colloque GRETSI, pp 671-674, Juan-les-Pins, Septembre 1993

Vallée a donc été remaniée, afin d'être mieux adaptée à la reconnaissance hors-ligne de caractères.

5.3.3.2 La Construction des Segments

La procédure de vectorisation requiert l'extraction préliminaire du squelette du caractère. Les segments de droite sont alors construits en suivant le tracé du squelette au moyen d'une méthode de recherche orientée, et sont validés à l'aide d'un critère des moindres carrés de l'erreur de représentation.

Le point de départ du premier segment est le premier pixel, appartenant au squelette, qui est trouvé en effectuant un balayage de l'image de haut en bas et de gauche à droite. Le deuxième point de ce segment est ensuite recherché dans le voisinage immédiat du premier. Cette recherche débute selon une direction horizontale, et se poursuit en éventail, selon des directions qui s'écartent progressivement de l'horizontale, si aucun pixel ne peut être trouvé (*figure 5.11*). Dès qu'un segment contient au moins deux points, un pixel connexe au dernier point validé est recherché en premier lieu, selon la direction actuelle du segment, et selon des directions qui s'en écartent progressivement ensuite (*figure 5.12*). La recherche orientée permet d'estimer, dans une certaine mesure, l'historique du tracé qui a été suivie lors du processus d'écriture du caractère.

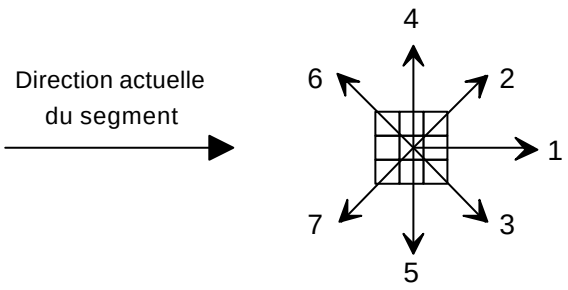


Figure 5.11 - Ordre préférentiel des directions de recherche de points adjacents.

Dès qu'un nouveau pixel du squelette est repéré, le segment en cours de construction est redéfini comme étant celui dont les extrémités sont le premier point du segment et le nouveau pixel trouvé. La valeur de la distance moyenne entre ce segment de droite et tous les pixels du squelette qui lui sont incorporés, est alors adaptée (*Annexe C*).

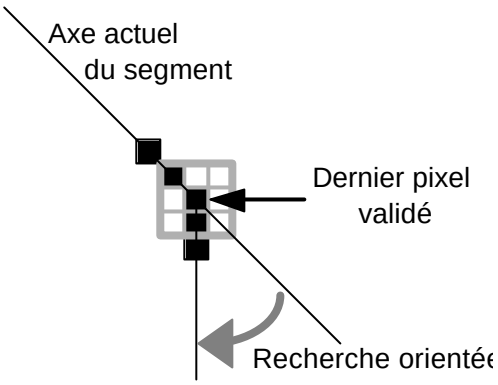


Figure 5.12 - Illustration de la procédure de Recherche Orientée.

Tant que cette distance, qui est un indice de l'erreur de représentation commise en remplaçant les pixels du squelette par un segment de droite, demeure inférieure à un seuil déterminé, le nouveau pixel trouvé est accepté dans le segment. Dans le cas contraire, le segment est clôturé au dernier pixel ayant été validé. La construction d'un nouveau segment est alors entamée, et débute selon une direction de recherche préférentielle des pixels, équivalente à celle du segment qui vient d'être clôturé. Afin d'assurer le maintien de la continuité du tracé, le dernier pixel validé d'un segment constitue toujours le premier point du segment suivant.

Dès qu'un pixel du squelette a été traité, une intensité de valeur nulle lui est assignée, de manière à éviter qu'il soit à nouveau pris en considération ultérieurement. La procédure de recherche continue jusqu'à ce que plus aucun pixel du squelette ne soit trouvé dans le voisinage immédiat du dernier point du segment en cours de construction. La procédure de balayage de l'image est alors reprise afin de s'assurer qu'elle ne contient plus que des pixels d'intensité nulle. Lorsque ce n'est pas le cas, le premier pixel trouvé constitue le point de départ d'un nouveau segment, dont la construction s'entreprend comme décrite précédemment. Lorsque tous les pixels de l'image sont d'intensité nulle, cela signifie que l'opération de vectorisation du squelette du caractère est achevée.

Afin de garantir une qualité suffisante de représentation des caractères, il s'est avéré indispensable de conclure systématiquement un segment dès la rencontre d'un pixel de croisement, et de maintenir ce dernier pour pouvoir effectuer la construction des autres segments au départ de celui-ci (*figure 5.13*). Par définition, un pixel de croisement est un pixel du squelette ayant au moins trois pixels allumés dans son voisinage immédiat. Comme un pixel de croisement doit rester présent dans l'image jusqu'à ce que tous les segments qui y aboutissent aient été construits, l'intensité de ce pixel ne peut pas être mise à une valeur nulle dès qu'il aura été

incorporé à l'un des segments. Ultérieurement, lorsqu'un segment sera clôturé faute de pixels, la construction de segments reprendra au départ du dernier pixel de croisement rencontré, tant que tous les pixels voisins de celui-ci n'ont pas été traités. Ce n'est qu'alors seulement, que l'intensité du pixel de croisement prendra une valeur nulle.

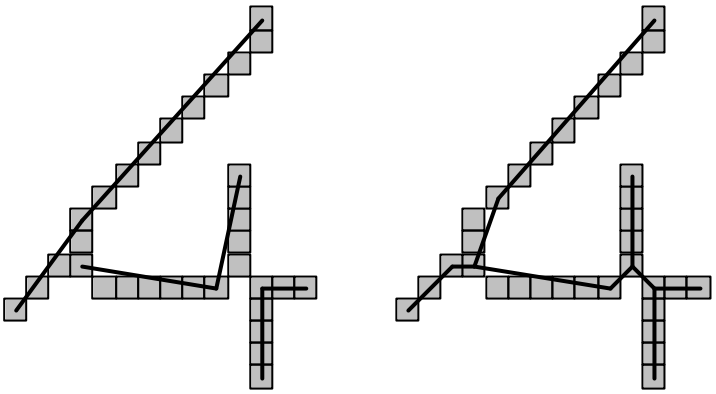


Figure 5.13 - Comparaison des résultats de la vectorisation sans maintien des points de croisement (à gauche), et avec maintien (à droite).

Afin d'assurer également le maintien de la continuité du tracé, le premier pixel trouvé lors d'un balayage de la matrice ne peut pas être immédiatement effacé et est considéré d'office comme un point de croisement, s'il existe au moins deux pixels d'intensité non nulle dans son voisinage immédiat (*figure 5.14*).

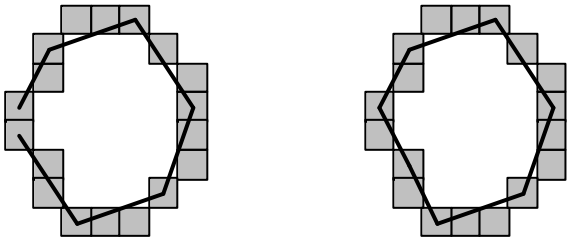


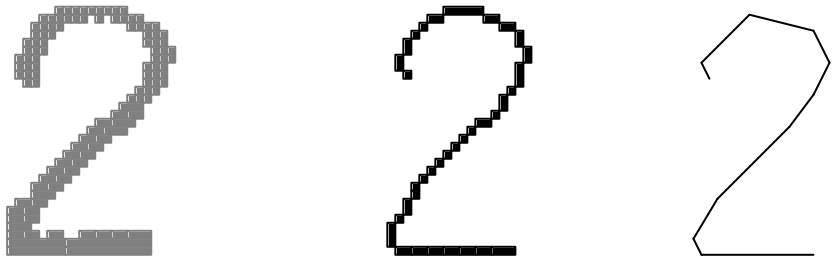
Figure 5.14 - Comparaison des résultats de vectorisation, sans que le premier pixel soit considéré comme un point de croisement (à gauche), et en le considérant comme tel (à droite).

Une modification de l'algorithme de construction des segments a été proposée par Schumacher, et consiste à limiter la longueur des segments construits [Schumacher,93]. Cette modification permet de mieux prendre en compte les subtilités du tracé lorsque celui-ci s'avère plus sinueux, et en améliore ainsi la qualité de représentation.

Les valeurs du seuil de distance moyenne utilisé pour la validation des segments, ainsi que celle de la longueur maximale tolérée, doivent être déterminées empiriquement. Au terme de nombreux essais, des valeurs de 0.2 pour la distance et de 10 pour la longueur maximale se sont avérées suffisantes pour offrir une bonne qualité de représentation, tout en évitant qu'une prolifération du nombre de segments apparaisse. Ces valeurs ont été établies pour des caractères normalisés à des dimensions de 32×32 pixels, et doivent bien évidemment être adaptées en fonction de la taille réelle des caractères.

L'opération de vectorisation achevée, un caractère est représenté par une liste de paires de coordonnées, qui représentent les extrémités de chaque segment. Ces coordonnées sont alors divisées par la plus grande des dimensions de l'image initiale, afin d'obtenir une invariance vis-à-vis de l'échelle du tracé et des valeurs normalisées comprises entre 0 et 1.

Un exemple de vectorisation d'un caractère est illustré à la *figure 5.15*. La *figure 5.16*, quant à elle, résume la procédure suivie pour la construction des segments.



(a) ***(b)*** ***(c)***

Figure 5.15 - Exemple de vectorisation d'un caractère:
le caractère original (a), son squelette (b), et le caractère vectorisé (c).

[Schumacher,93] **L. Schumacher**
Segmentation de l'Ecriture Cursive
Travail de Fin d'Etudes, Faculté Polytechnique de Mons, juin 1993

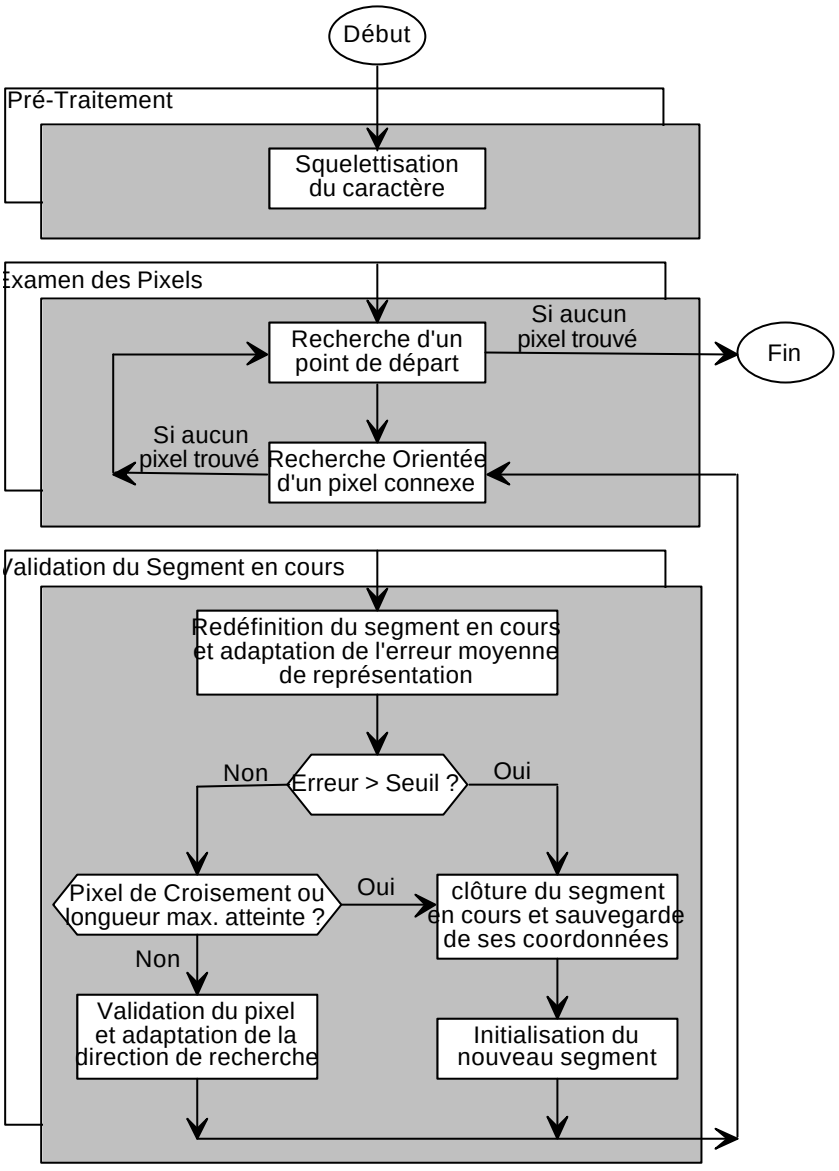


Figure 5.16 - Résumé de la procédure de vectorisation d'un caractère.

5.3.3.3 Discussion et Résultats

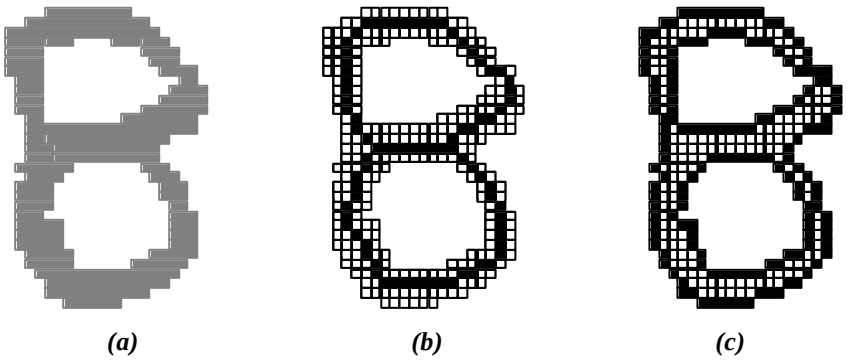
A la suite de multiples expériences, il s'est avéré, pour la plupart des caractères de la base de données ETL-3, que 16 segments sont suffisants pour coder l'ensemble des pixels qui représentent un caractère. La longueur du vecteur de primitives est ainsi réduit à $4 \times 16 = 64$ composantes. Lorsque moins de 16 segments sont nécessaires pour coder un caractère, le vecteur de primitives est complété par des zéros, afin de maintenir sa longueur constante.

Pour une dimension du vecteur de primitives de 64 composantes, le meilleur taux de reconnaissance atteint sur l'ensemble de test fût de **82,7%**, à l'aide d'un perceptron multicouches comportant 30 neurones en sa couche cachée. L'existence de plusieurs problèmes explique la valeur particulièrement faible, au regard de celles obtenues au moyen de la méthode des Pixels Moyennés, de ce taux de reconnaissance.

Le premier de ces problèmes est que la procédure de squelettisation des caractères, requise au préalable, introduit systématiquement, dans la représentation de ceux-ci, des distorsions qui peuvent parfois s'avérer critiques. Un exemple en est illustré aux *figures 5.17a* et *b*, où le squelette d'une lettre «B » devient délicat à distinguer de celui d'un chiffre «8 ». Un problème similaire peut également se poser pour des caractères tels que « 5 » et « S », ou encore « D » et « O », par exemple.

Les effets de ce premier problème peuvent être limités, en procédant à la vectorisation du contour des caractères, plutôt que de leur squelette (*figure 5.17c*). L'image des caractères étant binaire, l'extraction de leur contour est une procédure très aisée à mettre en oeuvre. Le même algorithme que précédemment peut ensuite être utilisé pour effectuer la vectorisation du contour. Les essais que nous avons effectués, ont montré que le nombre de segments nécessaire pour obtenir une bonne qualité de représentation des caractères doit être doublé, conduisant ainsi à des vecteurs de primitives de 128 composantes.

La vectorisation du contour au lieu de celle du squelette a permis d'augmenter le taux de reconnaissance à **85,6%**, ce qui demeure malgré tout relativement faible.



(a) ***(b)*** ***(c)***

Figure 5.17 - caractère « B » original (a), son squelette (b), et son contour (c).

Un deuxième problème qui est apparu est que, malgré l'utilisation d'une procédure de recherche orientée lors de la construction des segments, la suite de segments obtenue dépend

fortement de la position relative du pixel de départ du premier segment (*figure 5.18*). De même, la présence de certains défauts dans l'image initiale des caractères, influence-t-elle fortement le nombre et l'historique des segments construits (*figure 5.19*). Le résultat de la vectorisation de caractères peut ainsi être extrêmement variable, dans le sens où les segments qui se correspondent pour plusieurs caractères d'une même classe, ne sont pas toujours situés à une même position relative dans le vecteur de primitives fourni au système de classification.

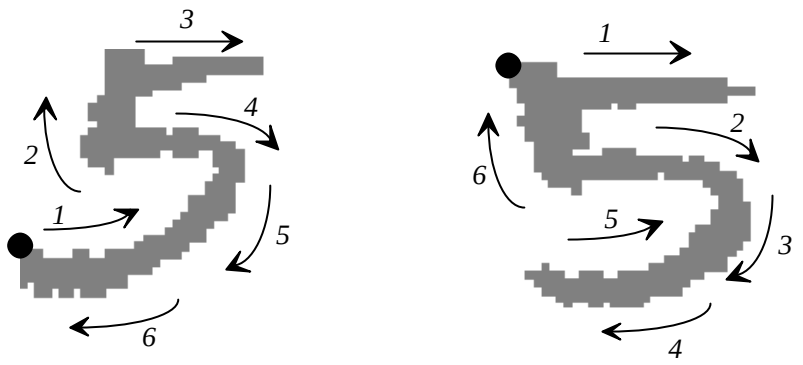


Figure 5.18 - Dépendance de la suite de segments construits, vis-à-vis de la position relative du pixel de départ du premier segment.

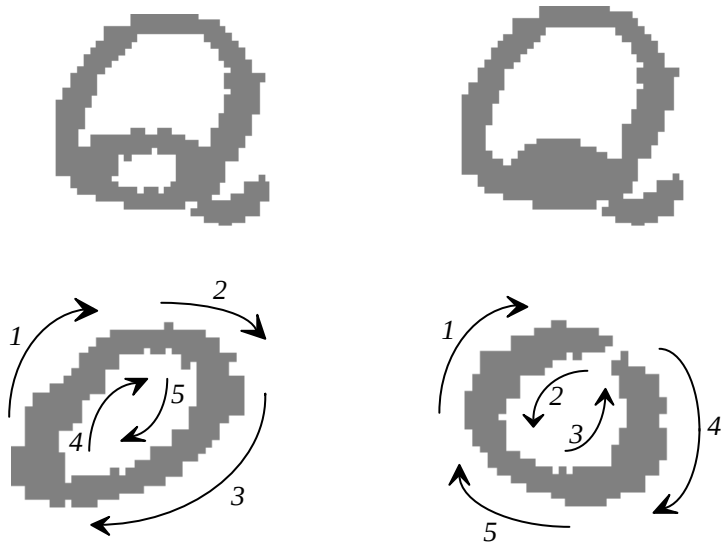


Figure 5.19 - Variation du nombre (lettre « Q ») et de l'ordre (lettre « O ») des segments construits, en cas de présence de défauts d'écriture.

Un troisième problème se pose en cas de présence d'un défaut d'encrage insuffisant du caractère. Les pixels d'intensité nulle, qui perturbent le tracé du squelette du caractère, vont

générer un grand nombre de points de croisement, ce qui augmente considérablement le nombre total de segments créés (*figure 5.20*). Outre la dispersion plus élevée des styles d'écriture qui en résulte, les derniers segments trouvés ne pourront être inclus dans le vecteur de primitives, puisque sa dimension est limitée. Certaines informations, peut-être discriminantes, seront alors indisponibles pour le système de classification, entraînant ainsi systématiquement une erreur de reconnaissance.

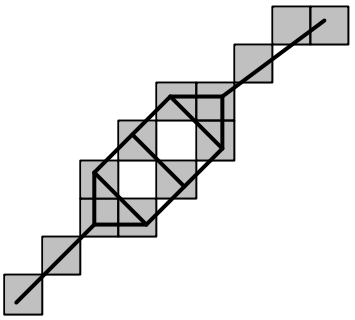


Figure 5.20 - Prolifération du nombre de segments en cas de défaut d'encrage insuffisant.

5.3.4 L'Analyse Normalisée du Contour

5.3.4.1 La Détection des Concavités

La présence, dans le vecteur de primitives, d'une information topologique de plus haut niveau que l'image brute du caractère, peut faciliter grandement la tâche du système de classification. Ainsi Philipski propose-t-il d'extraire une information quant aux diverses concavités présentées par le contour des caractères [Philipski,92]. La méthode proposée est très simple, puisqu'elle consiste à envoyer un certain nombre de « sondes » vers le caractère, selon diverses directions (*figure 5.21*). La longueur mesurée de chaque sonde est l'ordonnée, selon l'axe de recherche, du premier pixel d'intensité non nulle rencontré.

[Philipski,92] **A. Filipski & R. Flandrena**
Automated Conversion of Engineering Drawings to CAD Form
Proc. of the IEEE, vol. 80, n°7, pp 1195-1209, juillet 1992

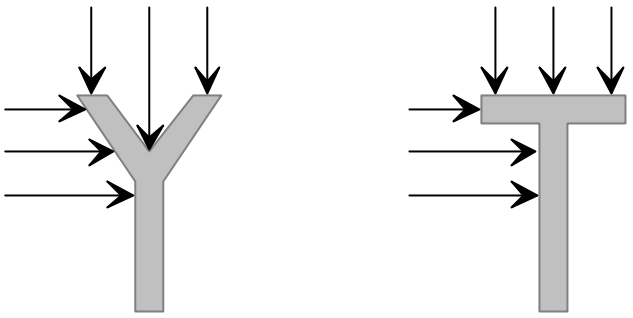


Figure 5.21 - Exemple d'utilisation de sondes:
celles-ci permettent d'effectuer une discrimination entre les lettres « Y » et « T ».

Dans le cas de caractères typographiques, les positions et les orientations des caractéristiques topologiques discriminantes demeurent pratiquement fixes dans l'image d'un caractère. L'emplacement ainsi que le nombre de sondes à utiliser peuvent alors être déterminés *à priori*. Cette situation idéale n'est évidemment pas vérifiée dans le cas de caractères manuscrits, et les sondes utilisées doivent alors être uniformément réparties le long du pourtour de l'image du caractère. Pour des raisons de facilité de calcul, il est aussi préférable que la détection des concavités ait lieu sur des caractères préalablement ramenés à des dimensions normalisées (*figure 5.22*). Le signal fourni par les sondes peut en outre être ainsi également normalisé, en divisant la valeur mesurée par celle de la dimension standard des caractères.

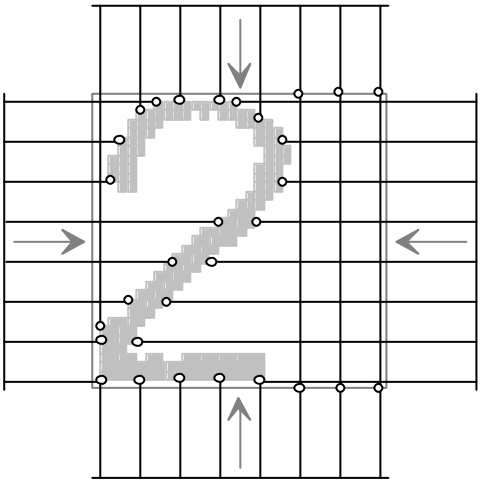


Figure 5.22 - Détection des concavités d'un caractère normalisé.

Des primitives semblables ont également étés utilisées par Suzuki [Susuki,92]. L'utilisation d'une sonde pour chaque ligne et chaque colonne de l'image des caractères conduit cependant à un vecteur de primitives de dimension relativement élevée. Afin de compresser les signaux des sondes, Suzuki a utilisé une méthode de codage qui, bien qu'offrant un taux de compression élevé, entraîne la perte d'une information quant à la position des concavités détectées. Seule était en effet conservée une information relative au nombre de concavités présentes selon chaque direction. Jugeant l'information de localisation des concavités utile à la reconnaissance, nous avons préféré employer une autre méthode pour coder les signaux des sondes.

L'information contenue dans le vecteur de primitives, provenant de l'analyse du contour, est compressée en remplaçant simplement plusieurs signaux de sondes consécutives par leur valeur moyenne. Cette opération peut être considérée comme un filtrage « passe-bas » du contour du caractère, et permet de masquer les éventuels accidents de contour (*figure 5.23*). Une valeur trop élevée du taux de compression risque toutefois de ne plus permettre la détection de concavités plus importantes, qui constituent souvent des caractéristiques topologiques discriminantes. C'est donc de manière empirique que le taux de compression des signaux des sondes doit être déterminé.

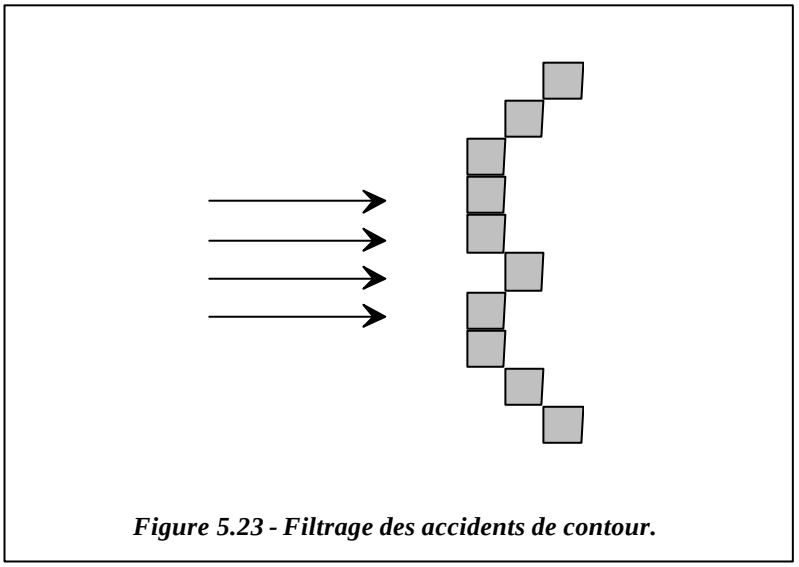


Figure 5.23 - Filtrage des accidents de contour.

Pour les caractères extraits de la base de données ETL-3, normalisés dans une matrice de 32×32 pixels au moyen de la première des méthodes décrites à la section 2.3 du présent

[Susuki,92] **H. Susuki, L.O. Chua, & T. Matsumoto**
A CNN Handwritten Character Recogniser
Int. Journal of Circuit Theory and Applications, vol. 20, n°5 pp 601-612, octobre 1992

chapitre, nos essais ont montré que, jusqu'à quatre signaux de sondes pouvaient être moyennés en un seul, tout en conservant une qualité de représentation suffisante (c'est-à-dire en ne provoquant pas d'augmentation du taux global d'erreur de classification mesuré). Selon chaque direction de recherche de concavités, 8 primitives étaient alors extraites, et le vecteur de caractéristiques global se réduisait donc à **32** composantes seulement. Ces dernières ont cependant permis d'atteindre un taux de reconnaissance de **94,8%** sur l'ensemble de test, à l'aide d'un perceptron multicouches comprenant 60 unités cachées [Gosselin,94].

5.3.4.2 La Détection des Intersections

Bien que le taux de reconnaissance soit le plus élevé atteint jusqu'ici, l'information de concavité n'est pas suffisante pour permettre une discrimination efficace entre toutes les classes de caractères. Certaines des confusions de reconnaissance qui se produisent, semblent en effet être peu logiques, et devraient donc de ce fait, être aisément corrigibles. C'est pourquoi le vecteur de primitives a été complété en y incluant les valeurs du nombre d'intersections entre le corps du caractère lui-même et des droites d'orientation déterminée.

Mori, dans [Mori,92], propose d'utiliser 6 droites parallèles et régulièrement espacées, selon chacune des quatre directions de recherche envisagées (verticale, horizontale, et les deux diagonales). Ce nombre apparaît trop peu élevé pour la reconnaissance de caractères manuscrits, et Suzuki propose lui, d'utiliser une droite pour chaque ligne, colonne, ou diagonale de l'image [Susuki,92]. Dans un premier temps, nous avons cependant opté de n'utiliser que des droites d'orientation verticale ou horizontale (*figure 5.24*), afin de conserver un vecteur de primitives de faible dimension.

[Gosselin,94] **B. Gosselin**
A Comparison of Different Feature Extractors for Handwritten Characters Recognition
Proc. of the ProRISC/IEEE Workshop on Circuits, Systems and Signal Proccessing, Papendael, Hollande, Mars 1994.

[Mori,92] **Mori**
Historical Review of OCR Research and Development
Proc. of the IEEE, Vol 80, n°7, pp. 1025-1216, Juillet 1992

[Susuki,92] **H. Susuki, L.O. Chua, & T. Matsumoto**
A CNN Handwritten Character Recogniser
Int. Journal of Circuit Theory and Applications, vol. 20, n°5 pp 601-612, octobre 1992

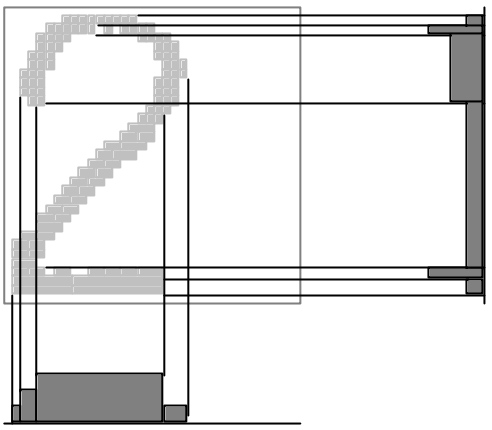


Figure 5.24 - Détection des intersections.

Une compression de la dimension du vecteur de primitives est réalisée de manière similaire à ce qui était effectué au point précédent, simplement en moyennant les valeurs de plusieurs composantes consécutives en une seule. Afin d'obtenir des valeurs normalisées comprises entre 0 et 1, le nombre d'intersections trouvées selon chaque direction peut, en outre, être divisé par le nombre maximal détectable, soit *à priori* la moitié des dimensions de normalisation des caractères.

Il apparaît cependant que le nombre moyen de ces intersections est très faible, et qu'il ne dépasse que rarement une valeur de quatre. Quand cette situation exceptionnelle se produit, c'est encore le plus souvent dû à la présence d'un défaut d'encrage insuffisant du caractère, qui augmente artificiellement le nombre d'intersections. Ne pas pouvoir prendre celles-ci en considération constitue, dans ce cas, un avantage non négligeable. Le nombre d'intersections a donc été saturé à une valeur de quatre, cette dernière étant également utilisée pour normaliser les primitives extraites. Ceci permet d'obtenir pour les primitives d'intersection une dynamique équivalente à celle des primitives de concavité, leur accordant ainsi à chacune une même importance relative dans le vecteur de caractéristiques global.

Pour les caractères extraits de la base de données ETL-3 et normalisés dans une matrice de 32×32 pixels, il est apparu, à la suite des essais que nous avons effectués, qu'un taux de compression supérieur à 2 pouvait entraîner une augmentation du taux global d'erreur de classification. Le vecteur de primitives extraites ici comportait donc finalement **32** composantes également.

Le vecteur de primitives global, constitué de la réunion des signaux de concavité et de ceux des intersections, était ainsi constitué de **64** composantes. Le taux de reconnaissance mesuré sur l'ensemble de test a alors atteint **96,1%**, ceci à l'aide d'un perceptron multicouches comprenant 30 unités cachées seulement.

5.3.4.3 Les Modifications Apportées

Lorsque la première méthode de normalisation des caractères est utilisée, il apparaît fréquemment que nombre de sondes ne peuvent extraire d'information utile (*figures 5.22* et *5.24*). La seconde méthode de normalisation peut alors être appliquée en lieu et place de la première, afin de forcer un caractère à occuper l'entièreté de la matrice standardisée (*figure 5.25*). Cette méthode implique cependant d'inclure obligatoirement dans le vecteur de primitives la valeur du rapport entre la largeur et la hauteur du caractère original. Cette information, qui traduit l'aspect initial des caractères, s'est en effet révélée être d'une importance capitale pour la discrimination entre certains d'entre eux. La longueur totale du vecteur de primitives était alors portée à **65** composantes. La qualité des primitives ainsi extraites a été améliorée, puisqu'un taux de reconnaissance de **96,9%** a pu être atteint, à l'aide d'un perceptron multicouches comprenant 60 unités cachées.

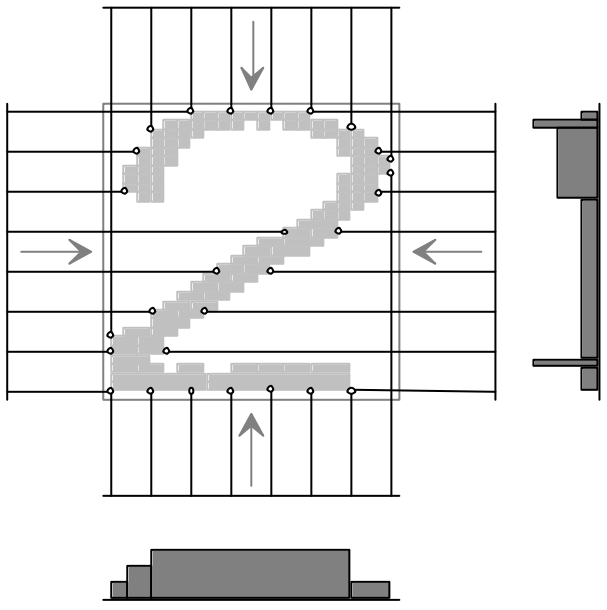


Figure 5.25 - Utilisation de la seconde méthode de normalisation des caractères.

Un deuxième problème qui est apparu, est que l'inclinaison présentée par certains caractères se montre parfois suffisante pour empêcher la détection de concavités discriminantes au moyen des seules sondes horizontales ou verticales, compromettant ainsi la reconnaissance (*figure 5.26*). Cet effet peut être contré par l'utilisation de sondes verticales partant du milieu du caractère et dirigées vers l'extérieur de celui-ci, ainsi que par l'utilisation de sondes diagonales également (*figure 5.27*). Cela porte à dix le nombre total de directions de recherche de concavités.

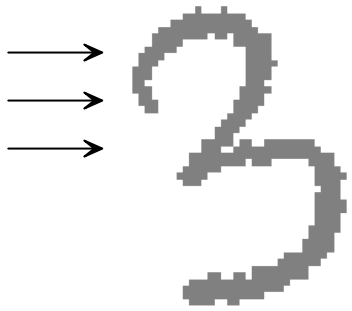


Figure 5.26 - Masquage partiel d'une concavité, du à une inclinaison trop importante du caractère.

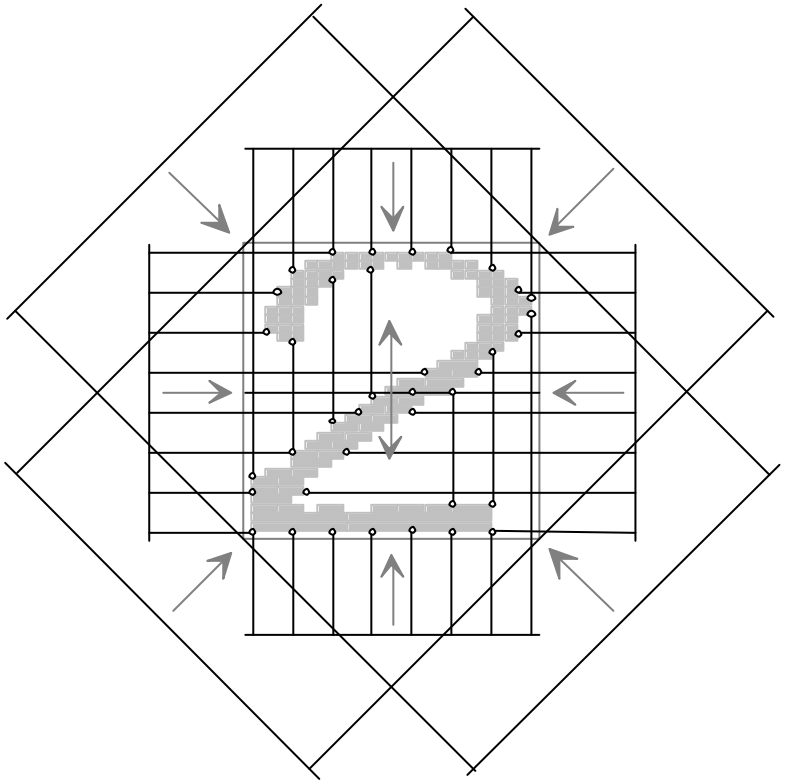


Figure 5.27 - Les dix directions de recherche utilisées pour la détection de concavités.

Les sondes diagonales sont quelque peu plus délicates à traiter que les sondes horizontales ou verticales. En premier lieu, une image normalisée à une taille de 32×32 pixels, nécessite l'utilisation de $(2 \times 32 - 1) = 63$ sondes selon chacune des directions diagonales, au lieu de 32 pour les directions horizontales ou verticales. Afin d'accorder à chaque direction une même importance relative au sein du vecteur de primitives global, les sondes prélevées diagonalement seront toutefois codées au moyen d'un même nombre de valeurs que les sondes prélevées horizontalement ou verticalement. Cette opération s'effectue de façon semblable à celle précédemment appliquée, c'est-à-dire que les valeurs de plusieurs sondes consécutives sont moyennées en une seule.

Toutefois, alors que les caractères sont assurés d'occuper l'entièreté de l'image selon les directions verticales et horizontales à la suite de l'opération de normalisation, ce n'est pas le cas selon les directions diagonales. Quelques sondes situées aux extrémités fournissent donc toujours une réponse nulle. Afin de conserver la meilleure qualité d'information possible, celles-ci sont décomptées, et seules les sondes restantes d'une diagonale sont prises en considération pour le calcul des valeurs qui lui sont attribuées dans le vecteur de primitives global. Selon chaque diagonale, les valeurs minimale et maximale des sondes sont ensuite recherchées, et servent à normaliser les signaux des sondes à des valeurs comprises entre 0 et 1. Ceci a toujours pour but d'accorder une même importance relative à toutes les composantes du vecteur de primitives global.

Le vecteur de primitives global atteint à présent une taille de 113 composantes, nombre qui est relativement élevé. L'emploi de méthodes de sélection de caractéristiques discriminantes pourrait dès lors s'avérer utile, et permettrait d'élaguer le vecteur de primitives afin de n'en conserver que les composantes les plus utiles à la classification. Ce point constitue l'objet de la section 5.4.

5.3.5 Résumé: Comparaison des Résultats

Le tableau 5.II résume les performances de reconnaissance atteintes au moyen de chacune des méthodes d'extraction de primitives qui viennent d'être exposées. Lors des tests effectués par Suzuki, également sur des caractères extraits de la base de données ETL-3, un taux de reconnaissance de **94,8%** avait été atteint, pour un vecteur de primitives réduit à **66** composantes. Le classificateur utilisé par Suzuki était également un réseau de neurones de type perceptron multicouches, dont le nombre d'unités cachées était fixé à 48 [Susuki,92].

[Susuki,92] **H. Susuki, L.O. Chua, & T. Matsumoto**
A CNN Handwritten Character Recogniser

De la comparaison de ces résultats, il ressort que les méthodes des Pixels Moyennés et surtout de l'Analyse Normalisée du Contour, apparaissent les mieux adaptées à la reconnaissance de caractères manuscrits. Les trop nombreux problèmes rencontrés lors de la vectorisation des caractères, font que nous abandonnerons définitivement cette approche par la suite. Dans tous les cas, la seconde méthode de normalisation s'est révélée bien plus efficace que la première et sera donc la seule à être encore utilisée par la suite.

Catégorie de Primitives	Dimension du Vecteur de Caractéristiques	Taux de Reconnaissance
Pixels Moyennés ($1^{ère}$ méthode de normalisation)	64	93,8 %
Pixels Moyennés (2^{nde} méthode de normalisation)	50	94,2 %
Vectorisation du Squelette	64	82,7 %
Vectorisation du Contour	128	85,6 %
Signaux de Concavités	32	94,8 %
Analyse Normalisée du Contour ($1^{ère}$ méthode de normalisation)	64	96,1 %
Analyse Normalisée du Contour (2^{nde} méthode de normalisation)	65	96,9 %

Table 5.II - Résumé des performances obtenues à partir des différentes catégories de primitives, pour des caractères issus de la base de données ETL-3.

L'examen des erreurs de classification commises montre que chacune des méthodes d'extraction de primitives possède quelques lacunes qui empêchent une reconnaissance correcte de caractères qui, à nos yeux d'humains, semblent pourtant faciles à reconnaître. Ces erreurs « simples » devraient pouvoir être aisément évitées en augmentant le nombre de primitives de base prélevées sur les caractères, ce qui rend nécessaire l'intervention ultérieure d'une procédure d'analyse discriminante, afin de limiter la taille finale du vecteur de caractéristiques.

5.4 La Sélection de Caractéristiques Discriminantes

5.4.1 Introduction

Les performances d'un classificateur dépendent fortement de la qualité de représentation des formes à reconnaître, ce qui implique généralement l'obligation de représenter ces dernières au moyen d'un nombre élevé de primitives. Il est alors fréquent qu'une partie de celles-ci ne contienne que des informations redondantes ou inutiles à la classification, rendant ainsi l'apprentissage du système de reconnaissance plus complexe.

Le processus connu sous le nom de « Sélection de Caractéristiques » a pour but de filtrer le vecteur des primitives de base, de manière à en extraire l'information discriminante et à présenter celle-ci au classificateur de manière pertinente.

5.4.2 L'Analyse en Composantes Principales

L'analyse en composantes principales est une procédure qui peut être menée dans le but de présenter plus efficacement les données au système de classification. Pour rappel, il s'agit d'une transformation linéaire qui effectue une rotation du système d'axes de représentation des données, afin d'obtenir la meilleure représentation possible, au sens des moindres carrés, des vecteurs de la base de données (*figure 5.28*). Cette méthode d'analyse est également connue sous le nom d'*expansion de Karhunen & Loève* [Fukunaga,90]. Il est aisé de montrer que minimiser la somme des carrés des distances des points qui représentent les données par rapport aux nouveaux axes est équivalent à maximiser la dispersion des données selon chacun d'entre eux. Les axes selon lesquels cette dispersion est la plus élevée sont donc ceux qui représentent le mieux les données.

[Fukunaga,90] **K. Fukunaga**
Introduction to Statistical Pattern Recognition
Academic Press, 1990

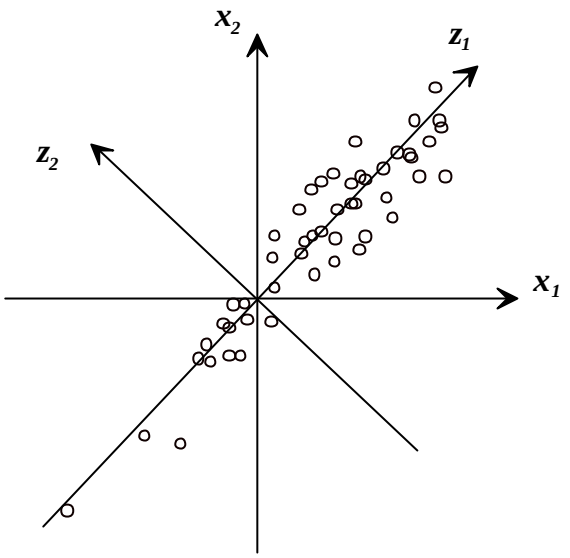


Figure 5.28 - Axes de représentation des données avant (x_1, x_2) et après (z_1, z_2) une procédure d'Analyse en Composantes Principales.

Les nouveaux vecteurs de caractéristiques sont obtenus par la transformation linéaire:

$$Z = U^T X \quad (5.5)$$

où X est une matrice à d lignes et à N colonnes qui contient l'ensemble de tous les vecteurs de primitives disponibles, et U une matrice de transformation d x d, réalisant la projection des données selon un nouveau système d'axes orthonormés ($U^T U = I$).

La matrice U qui permet d'obtenir la meilleure représentation possible des données est la matrice des vecteurs propres de la matrice d'autocorrélation des variables $X^T X$, chaque vecteur propre correspondant à un axe de projection [Lebart,82]. La valeur propre qui lui est associée est en outre représentative de l'amplitude de la dispersion des données selon cet axe.

Un pré-traitement des caractéristiques est le plus souvent nécessaire avant d'effectuer l'analyse en composantes principales. Il est ainsi préférable de centrer les caractéristiques, afin d'éliminer l'influence du niveau général de celles-ci. La *figure 5.29* illustre une situation où une

[Lebart,82] **L. Lebart, A. Morineau, & J.-P. Fénelon**
traitement des données statistiques
Ed. Dunod,Paris, 1982

telle opération est indispensable: l'espace centré (z_1, z_2) rend, en effet, mieux compte de la dispersion des données que l'espace vectoriel (t_1, t_2) . Un second problème est que les axes recherchés lors de l'analyse en composantes principales s'orientent dans les directions qui présentent les plus grandes dispersions. Afin d'accorder à toutes les primitives de base la même importance *à priori*, il convient donc de normaliser celles-ci avant d'effectuer l'analyse en composantes principales. Cette dernière aura donc lieu sur les caractéristiques centrées et réduites, et porte alors le nom d'*analyse en composantes principales normée* [Lebart,82].

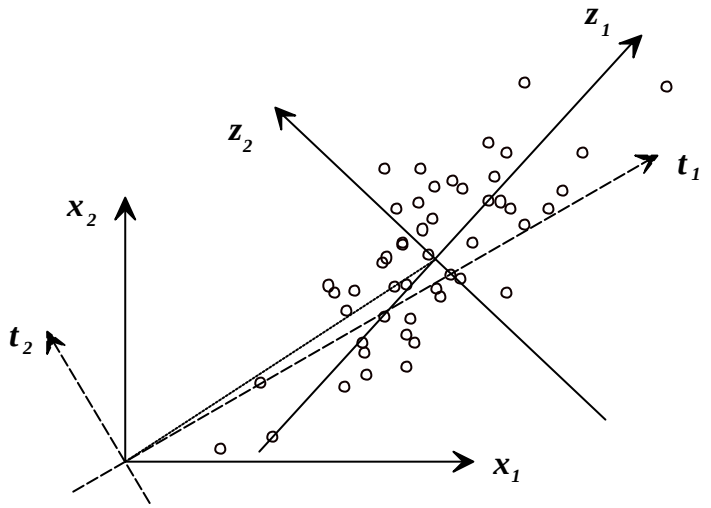


Figure 5.29 - Illustration de la nécessité de centrer les variables avant de procéder à l'analyse en composantes principales.

La matrice de transformation U est à présent obtenue par la recherche des vecteurs propres de $Y^T Y$, où Y est la matrice des caractéristiques centrées, réduites, et dont les éléments sont calculés selon:

$$y_{ij} = \frac{\left(x_{ij} - \mu_j\right)}{\sigma_j \sqrt{N}} \quad (5.6)$$

où x_{ij} représente la valeur de la caractéristique j pour le $i^{ème}$ objet disponible, alors que μ_j et σ_j en sont respectivement la moyenne et l'écart-type, estimés à partir de l'ensemble de tous les vecteurs de caractéristiques. Comme il n'y a pas de risque de phénomène de surentraînement ici,

[Lebart,82] **L. Lebart, A. Morineau, & J.-P. Fénelon**
traitement des données statistiques
Ed. Dunod,Paris, 1982

toute la base de données peut être utilisée pour calculer ces valeurs. Celles-ci sont donc données par:

$$\hat{\mu}_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij} \text{ , (5.7)}$$

et:

$$\hat{\sigma}_j^2 = \frac{1}{N}\sum_{i=1}^{N}\left(x_{ij} - \hat{\mu}_j\right)^2 \text{ (5.8)}$$

Les coefficients $\frac{1}{\sqrt{N}}$ et $\frac{1}{N}$, qui apparaissent dans les expressions (5.6) et (5.8) respectivement, ne sont introduits que pour faire coïncider la matrice à diagonaliser $\mathsf{Y}^T\mathsf{Y}$ avec la matrice des covariances expérimentales des variables, conformément à un usage répandu.

Comme déjà précisé, l'amplitude de la dispersion des données selon chaque nouvel axe de représentation est fournie par la valeur propre associée à ce dernier. Une autre pratique courante consiste à pondérer chaque vecteur de la matrice de projection U par l'inverse de la racine carrée de la valeur propre qui lui correspond, ce qui permet d'obtenir des nouveaux vecteurs de caractéristiques ayant une matrice de covariance unitaire. Les nouveaux vecteurs de caractéristiques sont dans ce cas obtenus au moyen de:

$$\mathsf{Z} = \mathsf{D}^{-\frac{1}{2}}\mathsf{U}^T\mathsf{Y} \text{ (5.9)}$$

où D est la matrice (diagonale) des valeurs propres de $\mathsf{Y}^T\mathsf{Y}$, et U en est celle des vecteurs propres.

A l'issue de l'analyse en composantes principales, les meilleurs axes de représentation sont ceux selon lesquels la dispersion des données est la plus grande[1]. Une réduction de la dimension de l'espace de représentation est donc réalisée, avec le moins d'erreur possible, en éliminant les axes dont les valeurs propres associées sont les plus faibles.

Toutefois, un axe qui offre une bonne représentation des données n'est pas forcément un axe qui en permet une bonne classification. La *figure 5.30* en illustre un exemple extrême pour un problème à deux dimensions: l'axe z_2 permet une classification linéaire parfaite, au contraire de l'axe z_1, qui offre pourtant une bien meilleure qualité de représentation des données. Afin de permettre la meilleure classification possible, un autre critère que celui de la valeur propre doit donc être utilisé pour sélectionner les nouveaux axes de représentation des données.

[1] cette dispersion est évidemment celle mesurée avant qu'il y ait normalisation par (5.9).

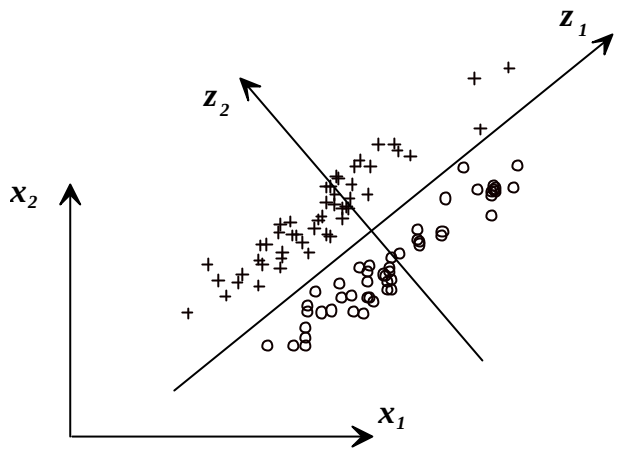


Figure 5.30 - Illustration du problème de la réduction adéquate de la dimension de représentation: l'axe z_1 est celui qui offre la meilleure qualité de représentation des variables, mais c'est l'axe z_2 qui en permet la classification.

5.4.3 Estimation de la Puissance de Discrimination

5.4.3.1 Le Critère de Fisher

Le critère de Fisher consiste à calculer la distance entre les valeurs moyennes d'une caractéristique établies pour deux classes données, et de la normaliser par la moyenne des variances, afin d'estimer le pouvoir discriminant de la caractéristique considérée entre ces deux classes.

Ce critère s'écrit, pour une caractéristique *k* déterminée [Fukunaga,90]:

$$D_k(\omega_i,\omega_j) = \frac{(\mu_{ik} - \mu_{jk})^2}{\sigma_{ik}^2 + \sigma_{jk}^2} \quad (5.10)$$

[Fukunaga,90] **K. Fukunaga**
Introduction to Statistical Pattern Recognition
Academic Press, 1990

où $\mu_{ik} = E\{x_k|\omega_i\}$ et $\sigma_{ik}^2 = E\{(x_k - \mu_{ik})^2|\omega_i\}$ représentent respectivement la moyenne et la variance de la caractéristique *k*, calculées sur l'ensemble des éléments de la classe ω_i.

Plus ce coefficient est élevé, plus la caractéristique est discriminante pour les deux classes considérées. Cependant, classer toutes les caractéristiques par ordre de puissance discriminante décroissante en fonction de toutes les paires de classes (ω_i, ω_j) est une tâche très fastidieuse. En outre, elle devient rapidement complexe lorsque le nombre de classes augmente, et rien n'assure qu'il sera possible de n'en retenir qu'un nombre limité par la suite. Il est donc préférable d'utiliser comme critère un pouvoir discriminant « global », calculé directement sur l'ensemble des classes.

5.4.3.2 Le Critère de Fisher Généralisé

Le critère de Fisher généralisé permet d'estimer la puissance moyenne de discrimination d'une caractéristique. Il s'exprime [Fukunaga,90]:

$$D_k = \frac{\sum_{i=1}^{C} p(\omega_i)(\mu_{ik} - \mu_k)^2}{\sum_{i=1}^{C} p(\omega_i)\sigma_{ik}^2} \quad (5.11)$$

où $\mu_k = \sum_{i=1}^{C} p(\omega_i)\mu_{ik}$ est la valeur moyenne de la caractéristique *k* calculée sur l'ensemble des éléments de toutes les classes, $p(\omega_i)$ étant la probabilité *à priori* de la classe ω_i.

En posant:

$$\overline{\mu_k^2} = \sum_{i=1}^{C} p(\omega_i)\mu_{ik}^2 \quad (5.12)$$

ainsi que:

$$\overline{\sigma_k^2} = \sum_{i=1}^{C} p(\omega_i)\sigma_{ik}^2 \text{ , (5.13)}$$

l'expression (5.11) peut s'écrire:

$$D_k = \frac{\overline{\mu_k^2} - \mu_k^2}{\overline{\sigma_k^2}} \text{ (5.14)}$$

Ce pouvoir discriminant global peut être calculé pour chacune des composantes du vecteur de primitives, et seules celles pour lesquelles il est supérieur à un seuil qui est à définir seront conservées.

Afin de pouvoir estimer de manière significative la puissance de discrimination propre à chaque caractéristique, il est essentiel que chacune apporte sa propre information. A cet effet, une procédure d'analyse en composantes principales normée peut être effectuée avant le calcul du pouvoir discriminant global. Les caractéristiques sont alors non seulement décorrélées entre elles, mais deviennent également centrées et réduites, c'est-à-dire de moyenne générale nulle et de variance unitaire:

$$\begin{cases} \mu_k = 0 \\ \sigma_k^2 = 1 \end{cases} \text{ , } \forall\ k \qquad \text{(5.15)}$$

La variance de la caractéristique k, calculée sur l'ensemble des éléments de toutes les classes, vaut:

$$\sigma_k^2 = E\left\{\left(x_k - \mu_k\right)^2\right\} \text{ (5.16)}$$

Cette expression peut être réécrite:

$$\begin{aligned} \sigma_k^2 &= E\left\{x_k^2\right\} - \mu_k^2 \\ \sigma_k^2 &= \sum_{i=1}^{C} p(\omega_i) E\left\{x_k^2 | \omega_i\right\} - \mu_k^2 \\ \sigma_k^2 &= \sum_{i=1}^{C} p(\omega_i)\sigma_{ik}^2 + \sum_{i=1}^{C} p(\omega_i)\mu_{ik}^2 - \mu_k^2 \\ \sigma_k^2 &= \overline{\sigma_k^2} + \overline{\mu_k^2} - \mu_k^2 \text{ (5.17)} \end{aligned}$$

En tenant compte des conditions (5.15), l'expression (5.14) s'écrit:

$$D_k = \frac{1-\overline{\sigma_k^2}}{\overline{\sigma_k^2}} = \frac{1}{\overline{\sigma_k^2}} - 1 \quad (5.18)$$

L'ajout ou le retrait d'une constante ne modifiant en rien les valeurs relatives des pouvoirs de discrimination, ces dernier peuvent s'exprimer:

$$D_k = \frac{1}{\overline{\sigma_k^2}} \quad (5.19)$$

La puissance de discrimination d'une caractéristique est alors tout simplement inversement proportionnelle à la moyenne de sa variance selon chaque classe.

L'expression (5.19) peut également s'écrire:

$$D_k = \frac{\overline{\mu_k^2}}{1-\overline{\mu_k^2}}, \quad (5.20)$$

et le pouvoir discriminant peut donc se calculer sur base des moyennes uniquement.

L'expression (5.20) permet de calculer très aisément le pouvoir discriminant global d'une caractéristique, dès lors qu'une procédure d'analyse en composantes principales normée a été préalablement effectuée. En pratique, les valeurs de $\overline{\mu_k^2}$ ne sont pas à disposition, et doivent être estimées à partir des exemplaires disponibles des objets. Cette estimation se fait selon:

$$\overline{\hat{\mu}_k^2} = \sum_{i=1}^{C} \hat{p}(\omega_i)\hat{\mu}_{ik}^2 \quad (5.21)$$

où:

$$\hat{\mu}_{ik} = \frac{1}{N_i} \sum_{x_k \in \omega_i} x_k \quad (5.22)$$

Bien que le critère établi permette de classer les caractéristiques par ordre décroissant de puissance de discrimination, le nombre qui doit en être conservé ne peut être déterminé qu'à la suite d'essais pratiques.

5.5 Application de l'Analyse Discriminante

5.5.1 Introduction

La méthode de sélection de caractéristiques discriminantes développée au point précédent a été appliquée aux deux méthodes d'extraction de primitives qui se sont précédemment révélées les plus efficaces, à savoir celle des *Pixels Moyennés* et celle de l'*Analyse Normalisée du Contour*.

La *figure 5.31* illustre une partie commune des histogrammes des valeurs propres et des puissances de discrimination, obtenus sur des primitives de type Pixels Moyennés après une procédure d'analyse en composantes principales normée. Chaque courbe a été normalisée de manière telle que la somme de ses éléments vaille l'unité, ce qui permet de les comparer sur un même graphique. Comme suggéré à la section 5.4.2, il apparaît clairement ici que certaines caractéristiques, malgré la valeur propre relativement importante qui leur est associée, ne sont en fait que très peu discriminantes. Ceci confirme que les meilleurs axes de représentation ne sont pas systématiquement les meilleurs axes de classification, et qu'une procédure d'analyse discriminante est effectivement nécessaire.

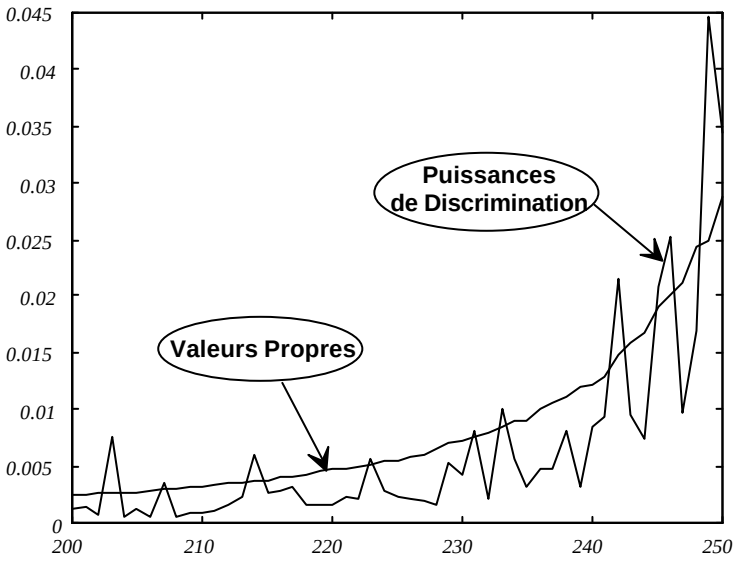


Figure 5.31 - Comparaison des Valeurs Propres et des Puissances de Discrimination, pour des primitives de type Pixels Moyennés.

Comme précisé précédemment, la valeur du seuil de puissance de discrimination à utiliser pour sélectionner les primitives doit être déterminé à l'aide d'essais pratiques. Lorsque les caractéristiques ont été classées par ordre décroissant de leur pouvoir discriminant, une courbe d'allure semblable à celle de la courbe illustrée à la *figure 5.32* a été observée, et ce pour chacune des deux catégories de primitives retenues.

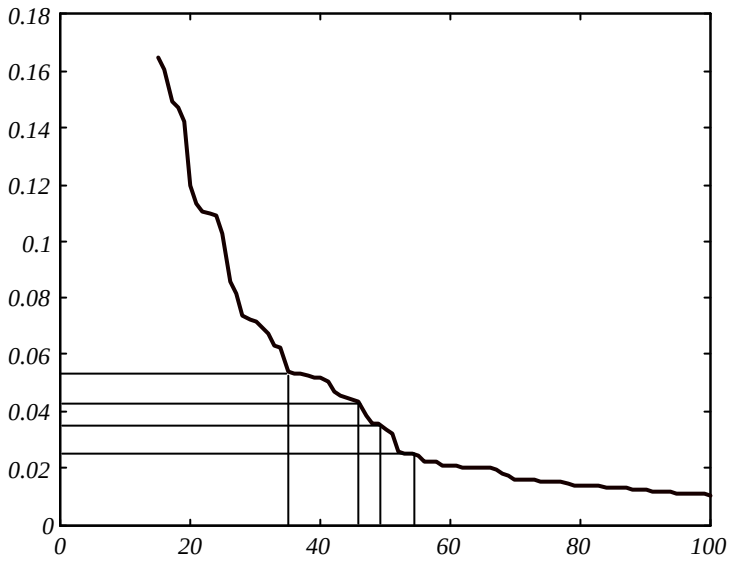


Figure 5.32 - Extrait de l'histogramme des pouvoirs discriminants.

Pour la seconde moitié des caractéristiques environ, le pouvoir discriminant calculé s'est révélé être très faible. Il est dès lors fort probable que ces dernières caractéristiques représentent en réalité plus du bruit que quelque chose d'autre, et les éliminer équivaut dans ce cas à filtrer le signal que représente le vecteur de primitives de manière à n'en conserver que l'information utile.

En ce qui concerne la première moitié des caractéristiques, la courbe du pouvoir discriminant présentait une évolution par paliers. De ce fait, le nombre d'essais auquel il a fallu procéder en pratique pour déterminer la valeur optimale du seuil de sélection fut relativement restreint (2 ou 3 dans notre cas).

5.5.2 Résultats pour la Base de Données ETL-3

Grâce à l'efficacité de la méthode de sélection de caractéristiques discriminantes, le codage initial des primitives de base pourra s'effectuer avec un taux de compression moins élevé que

précédemment. Ceci permet de maintenir une qualité de représentation des caractères plus élevée, la procédure d'analyse discriminante se chargeant automatiquement de réduire la dimension finale du vecteur de caractéristiques, en éliminant toute redondance éventuelle. La dimension du vecteur de primitives de base doit toutefois demeurer suffisamment faible pour pouvoir procéder au calcul de la matrice d'autocorrélation des variables, nécessaire à la procédure d'analyse en composantes principales.

Pour la méthode des Pixels Moyennés, cela se traduit par une normalisation des caractères à une dimension de 16 par 16 pixels, ce qui fait un vecteur de caractéristiques de 257 composantes, en y incluant le rapport d'aspect initial du caractère.

Le tableau 5.III reprend les meilleurs résultats obtenus, pour différents nombres de composantes sélectionnées. Le plus haut taux de reconnaissance atteint ici est de **95,1%**, alors qu'il était limité à 94,2% sans l'utilisation de la procédure d'analyse discriminante. Le nombre d'unités cachées nécessaires est toutefois plus élevé ici, et augmente lorsque le nombre de caractéristiques sélectionnées diminue. La phase d'apprentissage des réseaux de neurones apparaît cependant plus rapide que précédemment, grâce à la décorrélation des composantes des vecteurs d'entrée, réalisée par la procédure d'analyse en composantes principales.

Nombre de caractéristiques sélectionnées	Nombre d'unités cachées	Taux de Reconnaissance
64	60	94,8 %
48	60	95,1 %
29	90	95,1 %

Table 5.III - Résultats de l'application de l'Analyse Discriminante aux primitives Pixels Moyennés, pour des caractères extraits de la base de données ETL-3.

Dans le cas des primitives issues de l'Analyse Normalisée du Contour, les signaux des sondes sont à présent codés sur 16 valeurs au lieu de 8 précédemment. Comme 10 directions sont utilisées pour la recherche des concavités et 2 pour celle des intersections, le nombre de composantes du vecteur de primitives de base est porté à 193, en tenant compte du rapport d'aspect initial du caractère.

Les résultats obtenus sont repris au tableau 5.IV. Le meilleur taux de reconnaissance atteint fut de **97,3%**, alors qu'il avait été de 96.9% sans utilisation de la méthode d'analyse discriminante. Le nombre d'unités cachées nécessaire s'est avéré ici aussi être supérieur à celui obtenu précédemment. L'accélération de l'entraînement des perceptrons multicouches grâce à une décorrélation des composantes des vecteurs d'entrées, déjà constatée pour les primitives de la catégorie précédente, a été également observée pour les primitives issues de l'Analyse Normalisée du Contour.

Nombre de composantes sélectionnées	Nombre d'unités cachées	Taux de Reconnaissance
71	60	97,3 %
60	60	97,1 %
49	60	97,1 %
32	60	97,1 %
21	60	96,9 %

Table 5.IV - Résultats de l'application de l'Analyse Discriminante aux primitives issues de l'Analyse Normalisée du Contour, pour des caractères extraits de la base de données ETL-3.

5.5.3 Résultats pour la Base de Données NIST3

Afin de pouvoir comparer les performances des systèmes de classification développés ici avec celles de systèmes mis au point par d'autres auteurs, les deux méthodes qui se sont révélées les plus efficaces ont été également appliquées sur la base de données NIST3. Celle-ci provient des Etats-Unis d'Amérique et contient des exemplaires manuscrits des 10 chiffres ainsi que des 26 lettres latines majuscules, ayant étés écrits par plus de 2000 personnes distinctes.

Les tests furent effectués sur les chiffres d'une part, et sur les lettres d'autre part. L'ensemble d'entraînement était constitué dans le premier cas de 5000 exemplaires de chacun des 10 chiffres manuscrits, alors que l'ensemble de test contenait, lui, 2000 autres exemplaires.

Dans le second cas, l'ensemble d'entraînement comprenait 1324 exemplaires de chacune des 26 lettres, alors que l'ensemble de test en contenait 235 autres.

Le tableau 5.V résume les résultats obtenus sur les chiffres à l'aide de chacune des méthodes d'extraction de primitives. Les résultats obtenus sur les lettres, quant à eux, sont résumés au tableau 5.VI.

Catégorie de Primitives	Nombre de Composantes Sélectionnées	Nombre d'Unités Cachées	Taux de Reconnaissance
PM	60	120	97,5 %
PM	46	120	97.8 %
PM	33	90	97.6 %
ANC	83	50	98.5 %
ANC	69	60	98.7 %

Table 5.V - Résultats de reconnaissance de chiffres extraits de la base de données NIST3.

L'analyse discriminante permet une plus grande diminution du nombre de caractéristiques pour les primitives de type Pixels Moyennés que pour les primitives issues de l'Analyse Normalisée du Contour. Par contre, ces dernières étant des primitives de plus haut niveau, moins d'unités cachées se sont avérées nécessaires dans le perceptron multicouches. Elles offrent en outre une meilleure qualité de reconnaissance globale.

En utilisant la méthode des Pixels Moyennés, seules 46 des 257 caractéristiques initiales ont été retenues. Un taux de reconnaissance de **97,8%** a été atteint sur l'ensemble de test, à l'aide d'un perceptron multicouches comprenant 120 unités en sa couche cachée. Lors de l'utilisation de la méthode d'Analyse Normalisée du Contour, 69 des 193 caractéristiques initiales ont été sélectionnées. Celles-ci ont permis d'obtenir un taux de reconnaissance de **98,7%** sur l'ensemble de test, grâce à un perceptron multicouches contenant 60 unités cachées.

La notion de rejet de caractères par le réseau de neurones peut être introduite en exigeant un niveau minimal des valeurs de sortie de celui-ci. Lorsque ce niveau n'est pas atteint, le

caractère présenté à l'entrée du système de reconnaissance est rejeté, et aucune décision n'est prise quant à son appartenance à l'une ou l'autre classe. Ceci a pour conséquence que les caractères acceptés par le perceptron multicouches possèdent alors une probabilité plus élevée d'être correctement reconnus. La *figure 5.33* illustre, pour chacune des catégories de primitives, l'évolution du taux d'erreur observé sur les caractères acceptés par le système, en fonction du taux de rejet qui est toléré.

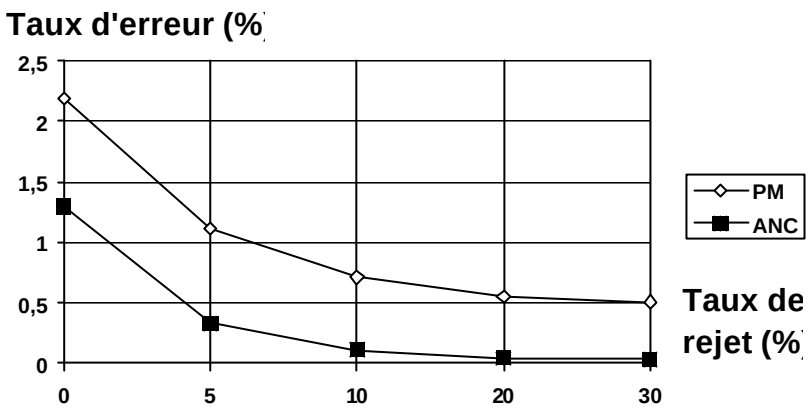


Figure 5.33 - Evolution du taux d'erreur en fonction du taux de rejet, pour des chiffres extraits de la base de données NIST3.

En ce qui concerne les lettres, le meilleur résultat atteint à l'aide de la méthode des Pixels Moyennés a été de **94,5%**, pour un vecteur de caractéristiques réduit à 43 composantes, et pour un perceptron multicouches comprenant 120 unités cachées. La méthode de l'Analyse Normalisée du Contour, quant à elle, a permis d'atteindre un taux de reconnaissance de **96,2%**, à partir de 72 caractéristiques sélectionnées, et à l'aide d'un perceptron multicouches contenant 90 neurones en sa couche cachée. La *figure 5.34* reprend l'évolution du taux d'erreur en fonction du taux de rejet toléré, pour chacune des catégories de primitives.

Catégorie de Primitives	Nombre de Composantes Sélectionnées	Nombre d'Unités Cachées	Taux de Reconnaissance
PM	70	60	93,4 %
PM	56	90	94,1 %
PM	43	120	94,5 %
ANC	86	60	95,8 %
ANC	79	60	95,9 %
ANC	72	90	96,2 %

Table 5.VI - Résultats de reconnaissance de lettres majuscules extraites de la base de données NIST3.

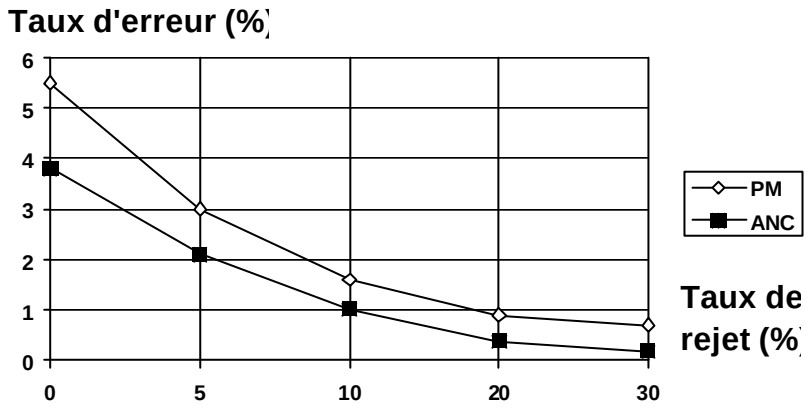


Figure 5.34 - Evolution du taux d'erreur en fonction du taux de rejet, pour des lettres majuscules extraites de la base de données NIST3.

Une comparaison de différents classificateurs appliqués à la reconnaissance de chiffres manuscrits également extraits de la base de données NIST3 a été effectuée par Blue et son équipe [Blue et al.,94]. Les primitives de base utilisées était alors semblables aux primitives de type *Pixels Moyennés,* obtenues à l'aide de la première méthode de normalisation de caractères

[Blue et al.,94] **J.L. Blue, G.T. Candela, P.J. Grotter, R. Chellappa, & C.L. Wilson**
Evaluation of Pattern Classifiers for Fingerprint and OCR applications
Pattern Recognition, Vol.27, n°4, pp 485-501, 1994

que nous avons décrite à la section 5.2.3. Une procédure d'analyse en composantes principales non normée fût menée, et les composantes sélectionnées par Blue le furent simplement sur base des valeurs propres associées aux nouveaux axes de représentation. En utilisant également un perceptron multicouches comme classificateur, le meilleur taux de reconnaissance obtenu fut de **95,7%**. Le réseau comprenait alors 64 unités cachées, et le nombre de caractéristiques était réduit à 52. Le meilleur résultat absolu obtenu par Blue a été de **97,4%**, à l'aide d'un classificateur de type «Plus Proche Voisin », pour un vecteur de caractéristiques réduit à 40 composantes.

Dans [Heutte,93], Heutte propose une méthode de reconnaissance de caractères basée sur une combinaison de plusieurs catégories de primitives. Les plus discriminantes d'entre elles sont sélectionnées sur base du critère de Fisher généralisé, sans toutefois qu'une procédure d'analyse en composantes principales ne soit préalablement effectuée. A l'aide d'un classificateur de type « Plus Proche Voisin », un taux de reconnaissance de **98,1%** a pu être atteint, toujours pour des chiffres extraits de la base de données NIST3. Le nombre de caractéristiques sélectionnées n'avait cependant pu être réduit qu'à 157.

Chhabra a, lui, extrait des caractéristiques géométriques, et en a obtenu des primitives de plus haut niveau en multipliant deux à deux toutes les composantes du vecteur de caractéristiques de base [Chhabra,93]. Des primitives résultantes, 200 ont été sélectionnées et ont servis à entraîner un perceptron à simple couche. Les taux de reconnaissance atteints ont été de **95,7%** pour les chiffres et de **95,1%** pour les lettres majuscules. Afin d'obtenir un taux d'erreur inférieur à **1%**, le taux de rejet avait alors dû être fixé à **12%** et **20%** respectivement.

En utilisant une combinaison de plusieurs quantificateurs vectoriels entraînés indépendamment, Ho a atteint un taux de reconnaissance sur les chiffres de **96,4%**. Le

[Heutte,93] **Heutte**
Handwritten Numeral Recognition Based on Multiple Feature Extractors
Proc. 2nd Int.Conf. on Document Analysis and Recognition, pp167-170,
Tsukuba, Japon, octobre 1993

[Chhabra,93] **A.K. Chhabra, Z. An, D. Balick, G. Cerf, K. Loris, P. Sheppard, R. Smith, & B. Wittner**
High-Order Statistically Derived Combinations of Geometric Features for Handprinted Character Recognition
Proc of 2nd Int. Conf. on Document Analysis and Recognition, pp 397-401,
Tsukuba, japon, octobre 1993

vecteur de caractéristiques était alors de dimension relativement élevée, puisqu'il comprenait 512 composantes distinctes [Ho,93].

Le tableau 5.VII résume tous ces résultats, pour la reconnaissance de chiffres manuscrits. La comparaison de ceux-ci avec les performances réalisées ici prouve la pertinence des primitives de type Pixels Moyennés, et surtout de celles extraites à l'aide de l'Analyse Normalisée du Contour, ainsi que l'efficacité du critère de sélection de caractéristiques utilisé. En particulier, la comparaison des résultats atteints ici pour les primitives de type Pixels Moyennés, pratiquement identiques à celles utilisées par Blue, avec ceux obtenus par ce dernier confirme à nouveau l'importance d'une sélection de primitives sur base d'un critère discriminant, plutôt que sur base d'un critère de qualité de représentation des données.

Type de Primitives (auteur)	Dimension du vecteur de caractéristiques	Classificateur Utilisé	Taux de Reconnaissance
pixels (Blue)	52	Perceptron Multicouches	95,7 %
pixels (Blue)	40	Plus Proche Voisin	97,4 %
combinaison (Heutte)	157	Plus Proche Voisin	98,1 %
géométriques (Chhabra)	200	Perceptron	95,7 %
pixels (Ho)	512	Quantificateurs Vectoriels	96,4 %
pixels (Gosselin)	46	Perceptron Multicouches	97,8 %
géométriques (Gosselin)	69	Perceptron Multicouches	98,7 %

Table 5.VII - Comparaison de différents systèmes de reconnaissance appliqués aux chiffres de la base de données NIST3.

[Ho,93] **T.K. Ho**
Recognition of Handwritten Digits by Combining Independent Learning Vector Quantizations
Proc of 2nd Int. Conf. on Document Analysis and Recognition, pp 818-821, Tsukuba, Japon, octobre 1993

5.5.4 Intégration de l'Analyse en Composantes Principales au Perceptron Multicouches

Si elles permettent de faciliter la phase d'apprentissage des réseaux de neurones, les procédures d'analyse en composantes principales et de sélection de caractéristiques entraînent toutefois la nécessité d'effectuer systématiquement, en phase de reconnaissance, une projection du vecteur de primitives initial sur le vecteur de caractéristiques réduit. Il est cependant possible, dès que l'apprentissage des perceptrons multicouches est achevé, d'intégrer cette opération aux calculs effectués par la première couche des réseaux de neurones. Tous les calculs effectués jusqu'à ce point ne font en effet qu'appliquer des transformations linéaires consécutives, lesquelles peuvent dès lors se réduire à une seule. La taille du réseau qui est obtenu est, dans ce cas, la même que celle d'un réseau qui aurait été entraîné directement sur base du vecteur de primitives initial.

Cette intégration de la procédure de projection au réseau de neurones n'est cependant pertinente que lorsque le nombre d'unités cachées du perceptron multicouches est faible par rapport au nombre de primitives initiales et au nombre de caractéristiques sélectionnées.

En effet, si:

- N_i est le nombre de primitives initiales;
- N_s représente le nombre de caractéristiques discriminantes sélectionnées;
- N_c est le nombre d'unités cachées du perceptron multicouches;

le nombre de multiplications à effectuer par la première couche du réseau de neurones, lorsque celle-ci intègre la procédure de projection, vaut:

$$N(o) = N_i \cdot N_c \quad (5.23)$$

Par contre, dans le cas où la procédure de projection n'est pas intégrée au perceptron multicouches, le nombre de multiplications à effectuer pour réaliser l'opération équivalente, est de:

$$N(o) = N_i \cdot N_s + N_s \cdot N_c \quad (5.24)$$

L'opération d'intégration de la procédure de projection au réseau de neurones n'est donc intéressante à réaliser que lorsque le nombre d'unités contenues dans la première couche cachée est tel que:

$$N_c < \frac{N_i \cdot N_s}{N_i - N_s} \quad (5.25)$$

Dans le cas où cette condition n'est pas vérifiée, le fait d'exécuter la procédure de projection du vecteur de primitives initial sur le vecteur de caractéristiques réduit, extérieurement au perceptron multicouches, permet d'augmenter la vitesse relative de reconnaissance par rapport à un réseau qui aurait été entraîné directement sur base du vecteur de primitives initiales.

En ce qui concerne les réseaux de neurones obtenus dans le cadre de l'application de reconnaissance de chiffres rapportée à la section précédente, l'opération d'intégration ne s'est justifiée que pour le perceptron multicouches entraîné sur base des primitives issues de l'Analyse Normalisée du Contour. Le second réseau, entraîné sur base des primitives de type Pixels Moyennés, malgré la non intégration de la procédure de projection, comportait deux fois plus d'unités cachées que le premier, et s'est finalement révélé être près de 50% plus lent que celui-ci.

Dans le cas du problème de la reconnaissance de lettres, ce n'est également que pour le perceptron multicouches entraîné à l'aide des primitives fournies par l'Analyse Normalisée du Contour que l'intégration de la procédure de projection s'est avérée pertinente. L'écart entre le nombre d'unités cachées des deux réseaux finalement obtenus étant toutefois moins important que dans le cas du problème de la reconnaissance de chiffres, ces derniers impliquent ici un volume de calcul équivalent.

5.6 Résumé

Diverses méthodes d'extraction de primitives pour la reconnaissance hors-ligne de caractères manuscrits ont étés développées. Deux d'entre elles, la méthode des Pixels Moyennés et celle de l'analyse Normalisée du Contour, se sont montrées particulièrement efficaces. La tentative de vectorisation des caractères s'est par contre révélée être un échec, principalement à cause de l'impossibilité de reconstituer correctement l'historique du tracé en écriture hors-ligne.

Une méthode d'analyse discriminante a également été présentée. Son efficacité de haut niveau, ainsi que sa facilité de mise en oeuvre, en font une étape indispensable pour

l'apprentissage de n'importe quel classificateur en général, et du perceptron multicouches en particulier. Tant la puissance de discrimination des caractéristiques qui lui sont alors fournies, que le fait qu'elles soient décorrélées grâce à la procédure d'analyse en composantes principales, facilitent sa phase d'apprentissage de manière significative. Appliquée aux deux meilleures catégories de primitives qui ont été développées, l'analyse discriminante a permis la conception de systèmes de reconnaissance de caractères manuscrits de très haute qualité.

Chapitre 6

La Génération Artificielle de Caractères Manuscrits

6.1 Introduction

L'image d'un caractère manuscrit, ou même imprimé, qui appartient à une classe déterminée peut être considérée comme étant issu d'un archétype de caractère de cette classe, auquel différentes distorsions, linéaires ou non, auraient étés appliquées. La mise en pratique de ce principe conduit à deux voies distinctes qui peuvent être envisagées afin d'augmenter la capacité de généralisation d'un système de reconnaissance.

La première de ces voies consiste à appliquer des distorsions locales à l'image du caractère à reconnaître, de manière à tenter de retrouver le modèle de la classe dont il est issu, ou, à défaut, à s'en rapprocher le plus possible. L'image qui représente un caractère est donc passée au travers d'un filtre, avant d'être communiquée au système de reconnaissance proprement dit. Des valeurs déterminées des paramètres de ce filtre pourraient par exemple permettre de réaliser une adaptation rapide d'un système de classification à un scripteur particulier. La méthode dite de *comparaison non linéaire*

[Tsukomo,84], celle faisant appel à la notion de *distance tangente* [Simard et al.,93], ou encore l'approche basée sur le calcul d'une *distance élastique* [Bertille,93], ont permis d'atteindre des résultats très positifs, en termes de capacité de généralisation d'un classificateur. Ces méthodes rendent toutefois nécessaire, lors de la phase de reconnaissance, la résolution systématique d'un problème d'optimisation non linéaire pour calculer la distance entre le caractère d'entrée et un prototype déterminé d'une classe. Malgré l'utilisation de contraintes globales et locales, destinées à restreindre les possibilités de distorsions, ces méthodes exigent énormément de ressources de calcul et/ou de mémoire vive disponible.

La seconde voie qui peut être envisagée pour augmenter le pouvoir de généralisation d'un système de reconnaissance, consiste à utiliser également un algorithme de distorsion de caractères, mais à présent afin de générer artificiellement des prototypes des différentes classes, et d'étendre ainsi le plus possible la diversité de la base de données destinée à l'entraînement du système de classification. Ce dernier reçoit alors la tâche d'assimiler ces déformations, et voit ses frontières de décision, séparant les diverses classes entre elles, modifiées. Cette manière de procéder semble particulièrement bien adaptée aux réseaux de neurones artificiels, au vu de leur grande aptitude à apprendre par l'exemple. Le principal avantage de cette méthode est que la distorsion de caractères n'est plus nécessaire lors de la phase de reconnaissance, et cette dernière n'est donc plus pénalisée. La phase d'apprentissage, par contre, peut voir sa durée considérablement augmentée car la quantité de données utilisée pour l'entraînement du réseau de neurones est supérieure. D'un point de vue pratique, cependant, la phase d'apprentissage n'est habituellement effectuée qu'une et une seule fois pour toutes, alors que l'objectif courant de l'application demeure l'utilisation du système en phase de reconnaissance. Il est donc essentiel que ce soit cette dernière qui puisse se dérouler de la manière la plus rapide possible, et la seconde approche semble dès lors préférable à la première.

[Tsukomo,84] **J.Tsukomo & K. Asai**
Non-linear matching method for handprinted character recognition
Proc. 11th Int. Conf Pattern Recognition, pp 770-773, 1984

[Simard et al.,93] **P.Simard, Y. Le Cun, J. Denker**
Efficient pattern recognition using a new transformation distance
Neural Information Processing Systems 5, S. J. Hanson, J.D. Cowan, and C.L. Giles (eds.), pp 50-58, Morgan Kaufmann Publishers, San Mateo, CA, 1993

[Bertille,93] **J.M. Bertille**
An elastic matching approach applied to digit recognition
Proc. 2nd Int. Conf. on Document Analysis and Recognition, pp 82-85, Tsukuba, Japon, octobre 1993

6.2 Commentaires sur la Génération de Caractères

La génération de caractères manuscrits « artificiels » s'effectue à partir d'un ou de plusieurs prototypes de caractères réels ou idéalisés, auxquels sont appliquées des déformations d'amplitudes aléatoires mais limitées. Diverses méthodes de distorsion de caractères ont été développées sur base de ce principe, et sont commentées dans les lignes qui suivent.

- La méthode mise au point par Kita requiert l'élaboration d'un archétype de caractère de chaque classe, sur base d'une suite hiérarchique de traits qui sont supposés être autant de mouvements d'un stylo imaginaire [Kita,93]. Chacun de ces mouvements peut alors subir des distorsions, lesquelles sont gérées par un ensemble de «règles » d'écriture, établies d'après des observations visuelles. Cette méthode implique d'étudier une à une chacune des classes de caractères, afin d'établir la ou les listes hiérarchiques des traits qui peuvent être utilisés par l'homme afin de former ces caractères. Les archétypes de caractère ainsi obtenus sont malheureusement fort semblables à des caractères provenant d'une frappe typographique. En outre, cette méthode n'a été développée que pour des caractères issus de l'écriture *kanji*, qui est une catégorie d'écriture Japonaise ne présentant pratiquement aucune caractéristique topologique courbe, au contraire des caractères latins. Cette méthode, en ne permettant pas la représentation de ces derniers avec une qualité suffisante, s'avère leur être difficilement applicable.

- La méthode développée par Tung est basée sur un principe semblable à celui de la méthode de Kita, mais les déformations que subissent les traits sont cette fois régies par des équations mathématiques [Tung et al.,93]. Cela rend cette méthode plus simple à mettre en oeuvre, mais celle-ci possède cependant toujours les mêmes inconvénients vis-à-vis des caractéristiques topologiques courbes.

- Kondo a quant à lui, proposé de représenter l'archétype d'un caractère au moyen d'une suite continue de segments de dimension égale, en utilisant un même nombre de segments pour

[Kita,93] **N. Kita**
Making a personal recognition dictionnary from characters automaticly generated by using handwriting model
Proc. 2nd Int. Conf. on Document Analysis and Recognition, pp76-81, Tsukuba, Japon, octobre 1993

[Tung et al,93] **C.H. Tung, Y.J. Chen, H.S. Lee**
Performance analysis of an OCR system via an Artificial handwritten Chinese character generator
Proc. 2nd Int. Conf. on Document analysis and recognition, pp 315-318, Tsukuba, Japon, octobre 1993

chaque archétype [Kondo,86]. Des caractères sont alors générés en apportant de légères perturbations aléatoires, à la fois à la dimension et l'orientation des segments, sous la contrainte que le tracé final formé par ceux-ci reste continu. La construction des archétypes de caractères requiert la détermination d'un modèle moyen de caractère de chaque classe, ainsi que la squelettisation de celui-ci. Les inconvénients de cette dernière procédure, ainsi que les difficultés d'application de la méthode de représentation de caractères établie par Kondo, ont toutefois été précédemment décrits, à la section du chapitre 5 consacrée à la vectorisation de caractères.

Le principal défaut de toutes ces méthodes est justement qu'elles nécessitent une *représentation des caractères sous forme vectorielle*, ce qui ne les rend applicables presque exclusivement qu'aux systèmes de reconnaissance « en ligne » de l'écriture. La difficulté d'obtenir une bonne qualité de représentation vectorielle en reconnaissance « hors ligne » a, en effet, déjà été montrée au chapitre précédent.

Un autre défaut de ces méthodes est l'utilisation d'un seul archétype de caractère par classe, ou d'un nombre très limité d'entre eux. Ceci implique l'obligation, afin de générer un nombre suffisant de caractères, de faire varier les divers paramètres de l'algorithme selon une échelle très large. Le nombre de ces paramètres étant en outre relativement élevé, une partie non négligeable des caractères générés peut présenter des déformations trop importantes, ou, à l'opposé, nulles, les effets de certains paramètres pouvant s'additionner ou se compenser. Ceci entraîne donc la nécessité d'examiner visuellement l'ensemble des images des caractères générés, afin d'éliminer celles qui seraient trop déformées pour être encore considérées comme représentant un caractère. Celles qui seraient trop semblables à l'archétype initial du caractère doivent également être éliminées, car leur ajout à l'ensemble d'apprentissage du classificateur n'apporterait aucune information utile supplémentaire à ce dernier.

Ces deux inconvénients majeurs ont fait que nous avons préféré développer une méthode originale de génération artificielle de caractères manuscrits. Cette méthode se base sur deux catégories de distorsion, appelées respectivement ***distorsion linéaire*** et ***distorsion géométrique***. Celles-ci peuvent être directement appliquées à chacun des caractères réels issus de l'ensemble d'apprentissage initial, et ne nécessitent pas l'élaboration d'un archétype de caractère de chaque classe. Ceci permet de générer un grand nombre de caractères pertinents à

[Kondo,86] **S. Kondo & B. Attachoo**
Model of Handwriting Process and its Analysis
Proc. of the 8th Int. Conf. on Pattern Recognition, Vol. 1, pp 562-565, Paris, Octobre 1986

l'aide d'algorithmes qui n'utilisent qu'un nombre restreint de paramètres, et qui limitent ainsi tout recours fastidieux à une procédure de sélection des caractères générés.

6.3 La Distorsion Linéaire

6.3.1 Principe de la Méthode

Cette méthode de déformation de caractères est basée sur un principe extrêmement simple. Celui-ci repose sur le fait que, lorsque les quatre sommets de l'image qui représente un caractère subissent des déplacements relatifs d'amplitudes et de directions différentes, il y correspond une distorsion de l'ensemble du caractère (*figure 6.1*). Cette distorsion est entièrement déterminée par les valeurs de huit paramètres élémentaires, qui sont l'amplitude des déplacements verticaux et horizontaux appliqués aux quatre sommets de l'image qui représente le caractère.

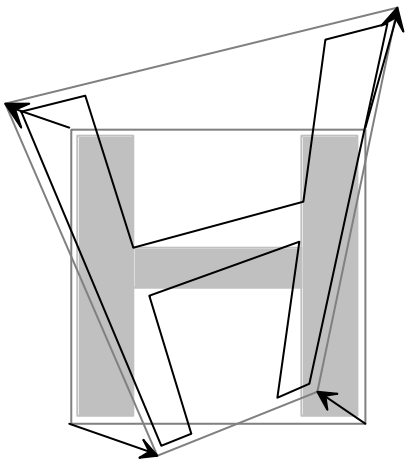


Figure 6.1 - Principe de la distorsion linéaire d'un caractère.

L'image qui représente un caractère dans l'ordinateur est le résultat de la discrétisation du caractère au moyen d'une grille d'échantillonnage régulière (*figure 6.2a*), cette dernière étant désignée par la suite par le terme de *grille de référence*. Une image déformée du caractère peut être facilement obtenue en appliquant la distorsion désirée à la grille d'échantillonnage elle-même (*figure 6.2b*), avec laquelle il suffira alors d'échantillonner à nouveau l'image qui représente le caractère.

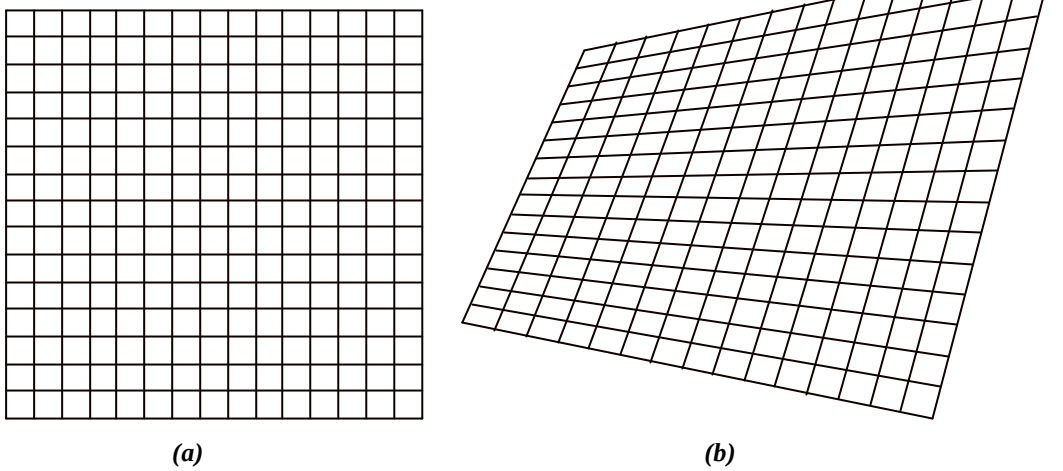


Figure 6.2 - Déformation linéaire de la grille d'échantillonnage d'un caractère.

6.3.2 Le Ré-Echantillonnage des Caractères

La procédure de ré-échantillonnage d'un caractère consiste à superposer, à l'image qui représente ce dernier, une grille d'échantillonnage déformée, dont la position de chacun des points doit être calculée par rapport au système d'axes formé par la grille de référence. Chaque point d'échantillonnage se voit alors attribuer la valeur du pixel le plus proche de la grille de référence (*figure 6.3*).

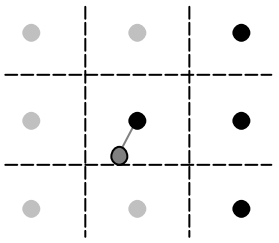


Figure 6.3 - Le ré-échantillonnage bidimensionnel.

Soit une grille d'échantillonnage initiale carrée. Après déformation, le contour de cette grille est représenté par quatre segments de droite qui ne sont plus ni parallèles ni perpendiculaires

deux à deux, et la position d'un point d'échantillonnage déterminé doit alors être calculée à partir des équations des deux droites correspondant respectivement à la valeur de son indice vertical et à celle de son indice horizontal (*figure 6.4*).

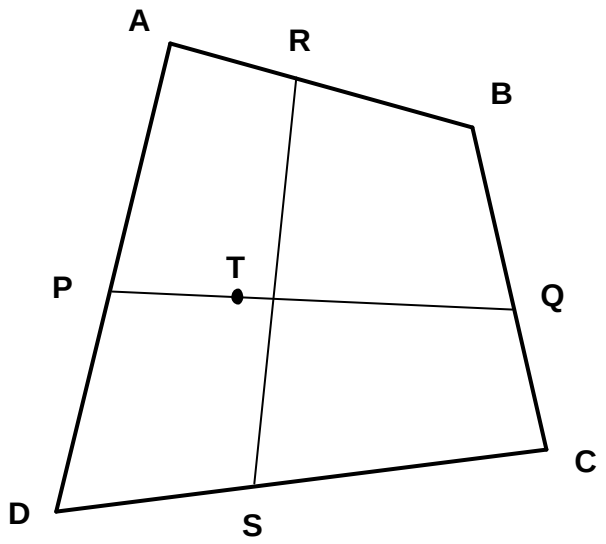


Figure 6.4 - Contour de la grille d'échantillonnage après sa distorsion (ABCD), et les droites PQ et RS, dont l'intersection correspond à un point d'échantillonnage déterminé.

Les coordonnées des points A, B, C, et D, sont directement déterminées à partir des valeurs des huit paramètres de déformation, qui, pour rappel, sont l'amplitude des déplacements verticaux et horizontaux relatifs des quatre sommets de la grille d'échantillonnage.

La droite qui correspond à un indice i de position verticale coupe les segments opposés AD et BC en des points P et Q qui sont tels que:

$$\frac{\|AP\|}{\|AD\|} = \frac{\|BQ\|}{\|BC\|} = \lambda_i \text{, (6.1)}$$

où λ_i est fonction de l'indice i.

Si N_e est le nombre de points d'échantillonnage prélevés selon chaque axe, le rapport λ_i est donné par la simple relation linéaire:

$$\lambda_i = \frac{i}{N_e - 1}, \qquad \forall i = 0, \cdots, N_e - 1. \quad (6.2)$$

Par convention, les indices des points d'échantillonnage vont croissant du haut vers le bas de l'image, et de la gauche vers la droite, comme il est d'usage courant en traitement d'image.

De manière similaire, la droite qui correspond à un indice *j* de position horizontale coupe les segments opposés AB et DC en des points R et S qui sont tels que:

$$\frac{\|AR\|}{\|AB\|} = \frac{\|DS\|}{\|DC\|} = \lambda_j, \quad (6.3)$$

où:

$$\lambda_j = \frac{j}{N_e - 1}, \qquad \forall j = 0, \cdots, N_e - 1. \quad (6.4)$$

Les coordonnées du point d'échantillonnage qui se situe à l'intersection des deux segments de droites PQ et RS peuvent être déterminées en exprimant leurs équations par rapport au système d'axes formé par la grille de référence, et en résolvant ensuite un système de deux équations à deux inconnues. Une autre méthode permet cependant d'isoler les contributions de chacun des indices *i* et *j*, et de calculer ainsi plus facilement et plus rapidement la position des points d'échantillonnage.

En adoptant une notation vectorielle, il vient:

$$\begin{cases} \overrightarrow{AP} = l_i \overrightarrow{AD} \\ \overrightarrow{BQ} = l_i \overrightarrow{BC} \end{cases} \quad (6.5)$$

ainsi que:

$$\begin{cases} \overrightarrow{AR} = l_j \overrightarrow{AB} \\ \overrightarrow{DS} = l_j \overrightarrow{DC} \end{cases} \quad (6.6)$$

Soit le point T appartenant au segment |PQ| tel que:

$$\overrightarrow{PT} = l_j \overrightarrow{PQ} \quad (6.7)$$

Nous allons démontrer que ce point T est bien le point d'intersection des segments de droite |PQ| et |RS|.

De (6.5), on peut écrire:

$$\begin{aligned} \overrightarrow{PQ} &= \overrightarrow{PA} + \overrightarrow{AB} + \overrightarrow{BQ} \\ &= \overrightarrow{AB} + l_i\left(\overrightarrow{BC} + \overrightarrow{DA}\right) \\ &= \overrightarrow{AB} + l_i\left(\overrightarrow{BC} + \overrightarrow{DC} + \overrightarrow{CA}\right) \\ &= \overrightarrow{AB} + l_i\left(\overrightarrow{BA} + \overrightarrow{DC}\right) \end{aligned} \quad (6.8)$$

D'où:

$$\overrightarrow{PT} = l_j \overrightarrow{AB} + l_i\left(l_j \overrightarrow{BA} + l_j \overrightarrow{DC}\right) \quad (6.9)$$

En prenant (6.6) en considération, il vient:

$$\begin{aligned} \overrightarrow{PT} &= \overrightarrow{AR} + l_i\left(\overrightarrow{RA} + \overrightarrow{DS}\right) \\ &= \overrightarrow{AR} + l_i\left(\overrightarrow{RA} + \overrightarrow{DA} + \overrightarrow{AS}\right) \\ &= \overrightarrow{AR} + l_i\left(\overrightarrow{RS} + \overrightarrow{DA}\right) \\ &= \overrightarrow{AR} + \overrightarrow{PA} + l_i \overrightarrow{RS} \end{aligned} \quad (6.10)$$

Et donc:

$$\overrightarrow{RA} + \overrightarrow{AP} + \overrightarrow{PT} = l_i \overrightarrow{RS} \quad (6.11)$$

Soit:

$$\overrightarrow{RT} = l_i \overrightarrow{RS} \quad (6.12)$$

Ce qui prouve que le point T appartient au segment |RS| également, et est donc bien le point d'échantillonnage recherché.

Les expressions (6.5) et (6.7) rendent ainsi possible, pour le calcul de la position des points d'échantillonnage, la séparation des contributions de chacun des indices *i* et *j*. L'équation de la droite qui passe par les points P et Q, déterminés pour une valeur donnée du rapport λ_i, peut en effet être calculée dans un premier temps, la position du point d'échantillonnage étant alors déterminée dans un second temps, en comptant un déplacement relatif λ_j le long de cette droite. Cette manière de procéder permet d'éviter d'avoir à résoudre systématiquement un système de deux équations à deux inconnues pour déterminer la position de chaque point d'échantillonnage.

Quelles que soient les déformations appliquées à la grille d'échantillonnage, il est obligatoire que celle-ci recouvre toujours l'entièreté de la grille de référence qui représente un caractère. En outre, il est également indispensable d'éviter de sous-échantillonner un caractère, afin d'en conserver une qualité de représentation suffisante. En pratique, cela se traduit par l'existence de deux relations entre la largeur de la grille d'échantillonnage, mesurée relativement à celle de la grille de référence, et le nombre de points de chacune de ces deux grilles.

Soient L_e la largeur relative de la grille d'échantillonnage avant toute déformation, et δ_{max} l'amplitude maximale relative du déplacement que peut subir un sommet de cette grille, selon l'axe vertical ou horizontal.

La largeur relative de la grille d'échantillonnage après distorsion varie entre $(L_e - 2\delta_{max})$ au minimum, et $(L_e + 2\delta_{max})$ au maximum (*figure 6.5*). Afin que la grille d'échantillonnage recouvre toujours l'entièreté de la grille de référence, sa largeur doit donc être telle que:

$$L_e - 2\delta_{max} \geq 1 \quad (6.13)$$

Soit:

$$L_e \geq 1 + 2\delta_{max} \quad (6.14)$$

Le sous-échantillonnage du caractère, quant à lui, sera évité si l'intervalle maximum qui sépare deux points de la grille d'échantillonnage est inférieur ou égal à celui qui sépare deux points de la grille de référence. Cet intervalle maximum est atteint lorsque la largeur de la grille d'échantillonnage atteint sa valeur la plus grande, soit $(L_e + 2\delta_{max})$, ou encore $(1 + 4\delta_{max})$ au minimum, d'après (6.14).

Le nombre de points d'échantillonnage prélevés selon chaque axe doit donc être tel que:

$$\frac{1+4\delta_{max}}{N_e} \leq \frac{1}{N_r} \quad (6.15)$$

où N_e est le nombre de points d'échantillonnage à prélever, et N_r celui de la grille de référence.

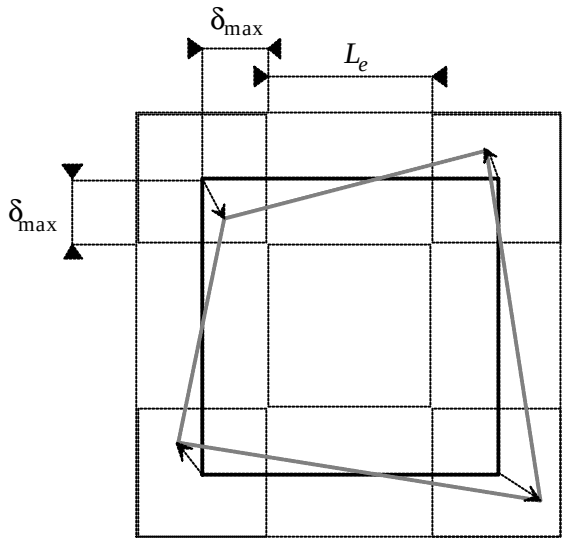


Figure 6.5 - Contour de la grille d'échantillonnage, et limites de son domaine de déformation.

Dans le but de limiter autant que possible le volume de calcul à effectuer, les limites inférieures des expressions (6.14) et (6.15) seront utilisées, et il vient alors:

$$\begin{cases} L_e = 1+2\delta_{max} \\ N_e = (1+4\delta_{max})N_r \end{cases} \quad (6.16)$$

A l'issue de la procédure de ré-échantillonnage, la taille de l'image qui représente le caractère déformé doit être ramenée à celle de la matrice de référence. Une étape de normalisation du caractère, en tous points semblable à celle décrite à la section 2.3 du chapitre 5, est donc introduite à la suite du processus de ré-échantillonnage.

6.3.3 Le Filtrage des Caractères

Malgré le respect de la condition (6.15), nécessaire pour éviter tout phénomène de sous-échantillonnage, de trop nombreuses discontinuités apparaissent dans l'image d'un caractère ré-échantillonné (*figure 6.6*). Ces discontinuités peuvent être considérées comme résultant de la superposition, à l'image du caractère, d'un bruit à haute fréquence. Un filtrage de l'image au moyen d'un filtre passe-bas bidimensionnel devrait dès lors permettre d'en améliorer la qualité.

(a) image du caractère initial

(b) image du caractère ré-échantillonné

Figure 6.6 - Images d'un caractère avant (a) et après (b) déformation.

Les filtres passe-bas qui ont été appliqués sont les filtres typiques introduits à la section 5.2.1, et repris ici à la *figure 6.7*. La meilleure qualité d'image a été obtenue à l'aide du filtre illustré à la *figure 6.7a*, qui est celui dont les fréquences de coupures sont les plus basses.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

(a)

1/10	1/10	1/10
1/10	1/5	1/10
1/10	1/10	1/10

(b)

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

(c)

Figure 6.7 - Les filtres passe-bas appliqués.

Cependant, si le filtrage du caractère permet de réduire le bruit à haute fréquence, il a également pour conséquence d'élargir excessivement l'épaisseur du tracé du caractère (*figure 6.8a*). Dans la plupart des cas, ce problème peut être résolu en effectuant une procédure d'érosion de l'image qui représente le caractère. Pour rappel, il s'agit d'une opération qui consiste à éliminer dans une image tous les pixels d'intensité non nulle qui sont adjacents aux

pixels de l'arrière-plan (*cfr. §5.2.2*). Cette procédure peut être appliquée plusieurs fois consécutivement, et l'on parle alors d'érosion à *n* pas, où *n* est le nombre de fois que la procédure est répétée. Dans ce cas, seuls les pixels dont la distance par rapport à l'arrière-plan est supérieure à *n* pixels seront conservés. Le résultat d'une opération d'érosion à deux pas est illustré à la *figure 6.8b*.

(a) résultat du filtrage passe-bas

(b) image du caractère après filtrage passe-bas et érosion à deux pas

Figure 6.8 - Illustration des opérations de filtrage et d'érosion d'un caractère.

L'épaississement du tracé des caractères, dû au filtrage passe-bas de leur image, a parfois pour conséquence de faire disparaître certaines caractéristiques topologiques, qui peuvent pourtant être essentielles. L'image de nombreux caractères latins, tels que 6, 8, 9, B, P, ou R, comporte en effet une ou plusieurs boucles de faible diamètre moyen, dont la perte lors de l'opération de filtrage est une conséquence fortement indésirable (*figure 6.9*). La procédure d'érosion ne permet pas dans ce cas de retrouver une image de bonne qualité.

(a) caractère à l'issue du filtrage passe-bas

(b) résultat de l'opération d'érosion

Figure 6.9 - Illustration d'un cas critique de filtrage passe-bas et d'érosion.

Le principe d'un filtrage passe-bas des images des caractères ré-échantillonnés a donc été abandonné, au profit de celui du procédé de sur-échantillonnage.

6.3.4 Le Sur-Echantillonnage des Caractères

Bien que nettement plus exigeant en termes de ressources de calcul, le sur-échantillonnage permet d'améliorer la qualité des images des caractères générés de manière très significative (*figure 6.10*). A l'issue de quelques essais visuels, le nombre de points d'échantillonnage a été fixé à 1,75 fois la valeur minimale exigée par la condition (6.15). Cette valeur a permis d'atteindre une très bonne qualité de représentation des caractères, tout en limitant le volume de calcul à effectuer.

Figure 6.10 - Exemple de caractère généré, avec sur-échantillonnage (à comparer aux figures 6.6b et 6.8b).

La *figure 6.11* illustre des exemples de caractères générés à partir d'un même caractère original, et pour différentes valeurs des huit paramètres de déformation. Le caractère initial est celui qui se situe en haut, à gauche, de l'image.

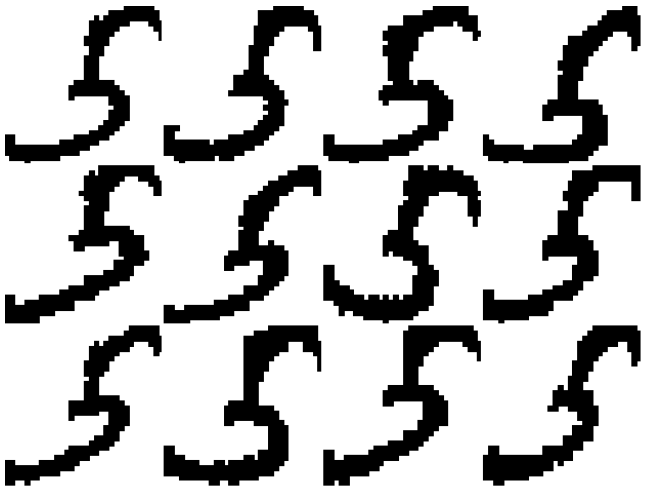


Figure 6.11 - Exemples de caractères générés à l'aide de la méthode de distorsion linéaire.

6.3.5 Résultats

La méthode de distorsion linéaire de caractères a été appliquée à des chiffres manuscrits extraits de la base de données NIST3, déjà décrite précédemment. Les caractères étaient normalisés dans une matrice de 32×32 pixels, et l'amplitude relative maximale du déplacement que pouvaient subir les sommets a été de 25%. Des déplacements d'amplitude relative supérieure à cette valeur ont en effet conduit à des images de caractères devant être rejetées, car n'étant plus suffisamment représentatives de la classe qui leur est associée. De (6.15), le nombre de points d'échantillonnage requis au minimum était donc de $(1+4\times0{,}25)\times32=64$. La procédure de sur-échantillonnage des caractères, requise pour en assurer une qualité de représentation suffisante, nous a donc conduit à utiliser une grille d'échantillonnage de $1{,}75\times64=112$ points. Afin de respecter la condition (6.14), ces 112 points étaient répartis sur une distance initiale de $(1+2\times0{,}25)\times32=48$ pixels, relativement à la grille de référence.

L'ensemble d'apprentissage initial contenait 800 exemplaires de chacun des 10 chiffres, et l'ensemble de test contenait, lui, 200 autres exemplaires. La méthode de distorsion linéaire a été appliquée sur chaque forme d'apprentissage, de manière à générer trois caractères à partir d'un seul. La taille globale de l'ensemble d'apprentissage était ainsi multipliée par un facteur quatre. Les primitives de l'Analyse Normalisée du Contour ont ensuite été extraites, et une procédure d'analyse discriminante appliquée. Un perceptron multicouches a alors été entraîné sur base des caractéristiques sélectionnées.

Le tableau 6.I compare les taux de reconnaissance qui ont étés obtenus pour différents domaines de variation des amplitudes des paramètres de déformation. Lorsque ces amplitudes sont systématiquement imposées à une valeur maximale, le caractère aléatoire de la distorsion subie par la grille d'échantillonnage se retrouve dans les directions des déplacements subits.

Valeurs des paramètres de déformation	Taille du perceptron multicouches	Taux de Reconnaissance
sans déformation	60 x 60 x 10	97,0 %
$0 < \delta < 10\%$	62 x 60 x 10	97,2 %
$0 < \delta < 25\%$	64 x 60 x 10	97,3 %
$10\% < \delta < 25\%$	64 x 60 x 10	97,6 %
δ = +/- 10%	64 x 60 x 10	97,6 %
δ = +/- 25%	64 x 60 x 10	97,8 %

Table 6.I - Comparaison des taux de reconnaissance obtenus pour différents domaines de variation de l'amplitude des déplacements appliqués aux sommets de la grille d'échantillonnage.

Le meilleur résultat fut atteint en utilisant systématiquement une valeur maximale des paramètres de déformation. Ceci s'explique par le fait que, lorsque les déplacements appliqués aux sommets de la grille d'échantillonnage sont d'amplitude trop faible, les caractères générés demeurent trop semblables aux caractères initiaux pour apporter suffisamment d'information supplémentaire.

Du fait que l'amplitude des paramètres de déformation est constante, le nombre total de distorsions distinctes que peut subir un caractère est relativement limité. Pour chacun des quatre sommets de la grille d'échantillonnage, quatre nouvelles positions sont possibles à l'issue de la procédure de distorsion (*figure 6.12*). Le nombre total de déformations que peut subir la grille d'échantillonnage s'élève donc à 4^4, soit *256*. Parmi celles-ci, *16* ne génèrent aucune distorsion du caractère car les déplacements subits par les sommets de la grille sont de sens identique ou opposés deux à deux, et conduisent ainsi à des grilles d'échantillonnage identiques à une

translation et/ou à un facteur d'échelle près. En outre, parmi les autres déformations que peut subir la grille d'échantillonnage, quatre sont équivalentes à d'autres, ce qui fait qu'il n'existe réellement que *236* distorsions distinctes. Les *20* autres distorsions peuvent être aisément évitées, en introduisant des relations conditionnelles entre les huit paramètres de déformation, ce qui rend non nécessaire toute procédure de validation manuelle *à posteriori* des caractères générés. Quelques essais visuels doivent toutefois être menés au préalable, afin de fixer la valeur de l'amplitude maximale des paramètres de déformation.

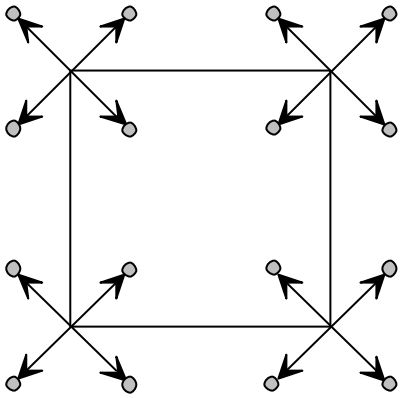


Figure 6.12 - Représentation des quatre nouvelles positions possibles de chaque sommet de la grille d'échantillonnage.

Une lacune de la méthode de distorsion linéaire des caractères est que la position relative des différentes caractéristiques topologiques qui constituent un caractère n'est jamais modifiée. Ceci est illustré par les trois exemplaires de la lettre « E », représentés à la *figure 6.13*. Les traits horizontaux constituent des éléments topologiques essentiels, et le fait que leur position relative les uns par rapport aux autres demeure identique, quelle que soit la distorsion appliquée, restreint la quantité d'information nouvelle apportée. Une seconde procédure de distorsion, appelée ***distorsion géométrique***, a donc été développée, et permet de remédier à ce problème.

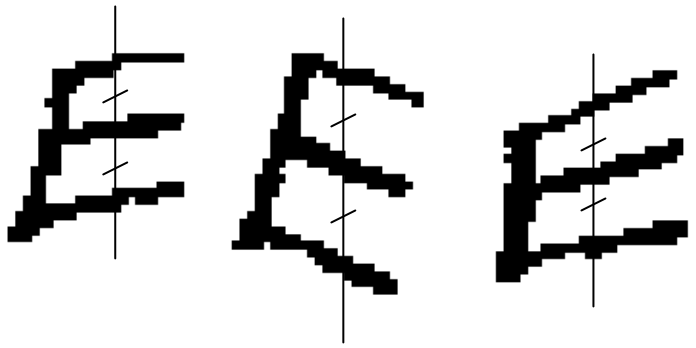
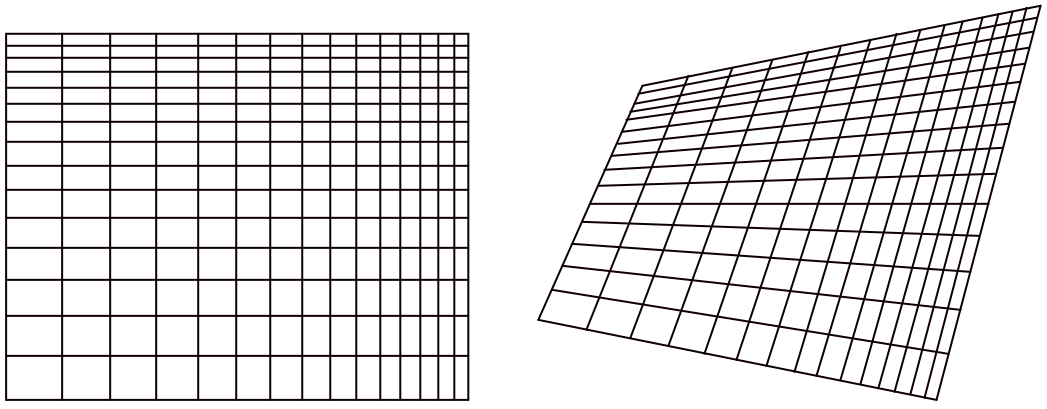


Figure 6.13 - Lacune de la méthode de distorsion linéaire:
les positions relatives des traits horizontaux demeurent constantes.

6.4 La Distorsion Géométrique

6.4.1 Principe de la Méthode

Le nom de cette procédure de distorsion provient du fait que le pas qui sépare deux points d'échantillonnage n'est plus constant, mais suit une progression géométrique (*figure 6.14a*). Ceci permet de dilater ou de contracter différentes parties d'un caractère, suivant que la raison de la série géométrique utilisée selon chaque axe est supérieure ou inférieure à l'unité. Lorsque les méthodes de distorsions arithmétique et géométrique sont conjointement appliquées, une grille d'échantillonnage d'allure semblable à celle illustrée à la *figure 6.14b* est obtenue.



(a) ***(b)***

Figure 6.14 - Exemples de grilles d'échantillonnage obtenues à l'aide de la distorsion géométrique.

6.4.2 Le Ré-Echantillonnage du Caractère

Lorsque les deux méthodes de distorsion sont simultanément appliquées, les formules qui permettent de calculer la position des points d'échantillonnage, établies à la section 6.3.2, doivent être modifiées afin de prendre en compte la distorsion géométrique également. Seuls sont cependant affectés les coefficients λ_i et λ_j, qui indiquent le rapport selon lequel les segments opposés du contour de la grille d'échantillonnage sont coupés en fonction des valeurs des indices *i* et *j*. Ces coefficients ne varient plus proportionnellement à ces dernières, mais dépendent à présent des raisons des progressions géométriques.

Soient γ_v et γ_h respectivement les raisons utilisées selon l'axe vertical et horizontal. Les coefficients λ_i et λ_j s'expriment à présent:

$$\lambda_i = \Delta_v\left(1+\gamma_v+\gamma_v^2+\cdots+\gamma_v^{i-1}\right) = \Delta_v\frac{\gamma_v^i-1}{\gamma_v-1} \quad (6.17)$$

et:

$$\lambda_j = \Delta_h\left(1+\gamma_h+\gamma_h^2+\cdots+\gamma_h^{j-1}\right) = \Delta_h\frac{\gamma_h^j-1}{\gamma_h-1} \quad (6.18)$$

où Δ_v et Δ_h désignent respectivement la largeur de l'intervalle d'échantillonnage initial selon l'axe vertical et horizontal, mesurée relativement à celle de la grille d'échantillonnage, et où $0 \le i, j \le N_e - 1$.

Chacun des côtés de la grille d'échantillonnage devant être entièrement couvert par l'ensemble des N_e points d'échantillonnage, il vient:

$$\lambda_{i=N_e-1} = \Delta_v\frac{\gamma_v^{N_e-1}-1}{\gamma_v-1} = 1 \qquad \text{et} \qquad \lambda_{j=N_e-1} = \Delta_h\frac{\gamma_h^{N_e-1}-1}{\gamma_h-1} = 1 \quad (6.19)$$

La valeur initiale relative du pas entre deux points d'échantillonnage en est aisément déduite, et on obtient alors:

$$\Delta_v = \frac{\gamma_v-1}{\gamma_v^{N_e-1}-1} \qquad \text{et} \qquad \Delta_h = \frac{\gamma_h-1}{\gamma_h^{N_e-1}-1} \quad (6.20)$$

ainsi que:

$$\lambda_i = \frac{\gamma_v^i - 1}{\gamma_v^{N_e-1} - 1} \qquad \text{et} \qquad \lambda_j = \frac{\gamma_h^j - 1}{\gamma_h^{N_e-1} - 1} \quad (6.21)$$

où $0 \le i, j \le N_e - 1$.

Les seuls paramètres de la méthode de distorsion géométrique sont les deux raisons γ_v et γ_h. Une valeur de ces paramètres supérieure à l'unité entraîne une dilatation progressive de l'intervalle d'échantillonnage, et résulte donc en une contraction progressive de l'image du caractère. Afin d'obtenir une amplitude maximale de contraction équivalente à celle de dilatation, les raisons géométriques seront de valeur aléatoirement choisie dans l'intervalle $\left[\frac{1}{\gamma_{\max}} \cdots \gamma_{\max}\right]$, où $\gamma_{\max}$ est supérieur à l'unité et reflète l'amplitude maximale de dilatation de l'intervalle d'échantillonnage. La largeur maximale de ce dernier, relativement à celle de l'ensemble de la grille d'échantillonnage, peut être obtenue en calculant la valeur initiale du pas, lorsqu'il y a contraction maximale:

$$\Delta_{\max} = \frac{1 - \frac{1}{\gamma_{\max}}}{1 - \frac{1}{\gamma_{\max}^{N_e-1}}} \quad (6.22)$$

Afin d'obtenir, lorsque la méthode de distorsion arithmétique est appliquée conjointement, une valeur relative à la largeur de la grille de référence, celle-ci doit être multipliée par un facteur qui vaut la largeur maximale relative de la grille d'échantillonnage, soit $(1 + 4\delta_{\max})$, d'après (6.16). Pour éviter de sous-échantillonner un caractère, l'espace maximal qui sépare deux points d'échantillonnage consécutifs doit toujours être inférieur ou égal à celui qui sépare deux points de la grille de référence, ce qui donne donc:

$$\frac{1 - \frac{1}{\gamma_{\max}}}{1 - \frac{1}{\gamma_{\max}^{N_e-1}}}(1 + 4\delta_{\max}) \le \frac{1}{N_r} \quad (6.23)$$

La valeur du nombre minimum de points d'échantillonnage à prélever en est déduite à l'issue de quelques manipulations mathématiques, et il vient:

$$N_e \ge 2 + \frac{-\log\left(\gamma_{\max} - N_r(1 + 4\delta_{\max})(\gamma_{\max} - 1)\right)}{\log(\gamma_{\max})} \quad (6.24)$$

L'algorithme de l'ensemble de la procédure de génération de caractères se résume finalement comme suit:

1. Fixer les valeurs maximales δ_{max} et γ_{max} tolérées pour les paramètres de distorsion linéaire et géométrique; En déduire la valeur de la largeur relative de la grille d'échantillonnage, ainsi que le nombre minimum de points de celle-ci, au moyen de:

$$\begin{cases} L_e = 1 + 2\delta_{max} \\ N_e \geq 2 + \dfrac{-\log\left(\gamma_{max} - N_r\left(1 + 4\delta_{max}\right)\left(\gamma_{max} - 1\right)\right)}{\log\left(\gamma_{max}\right)} \end{cases}$$

2. Appliquer des déplacements d'amplitudes et de directions aléatoires aux sommets A, B, C, et D de la grille d'échantillonnage, et calculer leurs nouvelles coordonnées par rapport à la grille de référence. Choisir aléatoirement les valeurs des raisons γ_v et γ_h des progressions géométriques que suivent respectivement les échelles verticales et horizontales.

3. Pour une valeur donnée de l'indice *i*, calculer les coordonnées des points P et Q tels que:

$$\begin{cases} \overrightarrow{AP} = l_i \overrightarrow{AD}, \\ \overrightarrow{BQ} = l_i \overrightarrow{BC} \end{cases} \qquad \text{avec:} \qquad \lambda_i = \frac{\gamma_v^i - 1}{\gamma_v^{N_e - 1} - 1}$$

4. Pour une valeur donnée de l'indice *j*, calculer les coordonnées du point d'échantillonnage T correspondant, tel que:

$$\overrightarrow{PT} = l_j \overrightarrow{PQ} \qquad \text{où:} \qquad \lambda_j = \frac{\gamma_h^j - 1}{\gamma_h^{N_e - 1} - 1}$$

5. Attribuer au point d'échantillonnage considéré la valeur du pixel de la grille de référence dont les coordonnées entières sont les plus proches de celles calculées au point 4;

6. Si $j < N_e$ incrémenter *j*, et retourner au point 4;
   Sinon tant que $i < N_e$, incrémenter *i*, remettre *j* à *0*, et aller au point 3.

Des exemples de caractères manuscrits générés grâce à l'utilisation conjointe des méthodes de distorsion linéaire et géométrique sont illustrés à la *figure 6.15*.

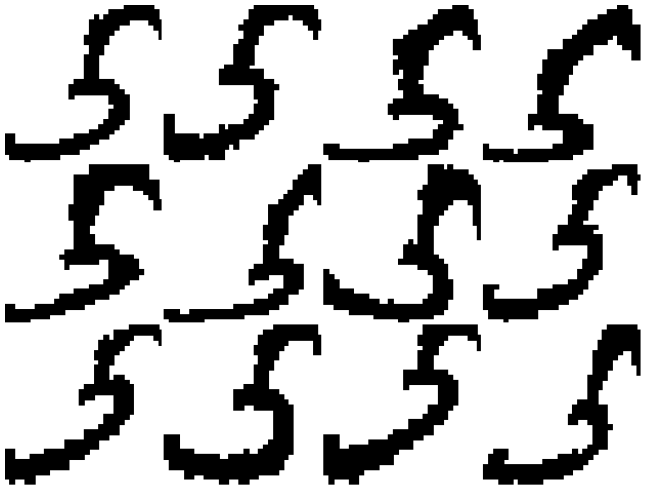


Figure 6.15 - Exemples de caractères manuscrits générés à l'aide de l'utilisation conjointe des deux méthodes de distorsion. Le caractère initial est celui situé en haut, à gauche.

6.4.3 Résultats

De manière similaire à ce qu'il était advenu lors des essais d'application de la procédure de distorsion arithmétique, il est apparu plus avantageux d'utiliser systématiquement une valeur maximale des raisons des progressions géométriques, plutôt que de faire varier l'amplitude de ces dernières aléatoirement. Le caractère aléatoire de la distorsion subie par la grille d'échantillonnage se retrouve alors dans le sens, contraction ou dilatation, de l'échelle géométrique. Ceci permet d'assurer qu'une distorsion suffisante soit toujours appliquée aux caractères.

La valeur maximale γ_{max} des raisons des progressions géométriques a été limitée à 1,0092, à l'issue d'examens visuels des déformations obtenues. La valeur de l'amplitude maximale des déplacements appliqués aux sommets de la grille d 'échantillonnage, quant à elle, a pu être maintenue à 25% lors de l'utilisation conjointe des deux méthodes de distorsion, sans qu'une dégradation excessive des caractères n'apparaisse. Pour ces valeurs, et pour des caractères normalisés à une dimension de 32×32 pixels, le nombre minimal de points d'échantillonnage à prélever vaut 97, d'après (6.23). La valeur de 112, adoptée pour effectuer le sur-échantillonnage requis par la méthode de distorsion arithmétique, s'avère donc toujours (théoriquement) suffisante. En pratique, cette valeur a permis de conserver une bonne qualité de représentation des caractères, même lorsque les deux méthodes de distorsion ont été appliquées simultanément, et n'a donc pas dû être augmentée. En utilisant ce nombre de points

d'échantillonnage, une valeur de γ_{max} égale à 1,0092 correspond à positionner la moitié d'entre eux dans les premiers 3/8 de la largeur de la grille de référence, ce qui revient à dilater cette partie de $\frac{1/2}{3/8} = 33\%$. Une valeur égale à 1,0040 correspond, elle, à positionner la moitié des points d'échantillonnage dans les premiers 4/9 de la largeur de la grille de référence, ce qui est équivalent à effectuer une dilatation de cette partie de 12,5%. A titre de comparaison, une valeur de γ_{max} de 1,0125 entraînerait le recouvrement du premier 1/3 de cette largeur par la moitié des points d'échantillonnage, soit une dilatation partielle de 50%. Le tableau 6.II résume les résultats obtenus

Paramètres de distorsion linéaire	Paramètres de distorsion géométrique	Taille du perceptron multicouches	Taux de Reconnaissance
-	-	60 x 60 x 10	97,0 %
-	γ = 1,0040	63 x 60 x 10	97,3 %
δ = +/- 10%	γ = 1,0040	66 x 60 x 10	97,8 %
δ = +/- 25%	γ = 1,0040	67 x 75 x 10	97,9 %
-	γ = 1,0092	64 x 75 x 10	97,5 %
δ = +/- 10%	γ = 1,0092	68 x 90 x 10	98,1 %
δ = +/- 25%	γ = 1,0092	68 x 90 x 10	98,0 %

Table 6.II - Performances obtenues grâce à l'utilisation conjointe des deux méthodes de distorsion de caractères.

L'utilisation conjointe des deux méthodes de distorsion a permis de porter le taux de reconnaissance à 98,1%, au départ d'un taux de 97,0% sans génération de caractères, et d'un taux de 97,8% en utilisant la méthode de distorsion linéaire seule. Plus l'amplitude des paramètres de déformation augmente, plus le nombre de caractéristiques à prendre en considération, ainsi que le nombre d'unités cachées du perceptron multicouches, deviennent élevés, ce qui, en soit, demeure parfaitement logique. La diversité des formes à classifier devient simplement telle que,

plus de caractéristiques, linéaires ou non, doivent être prises en considération pour pouvoir en effectuer la reconnaissance.

L'utilisation systématique d'une amplitude maximale des raisons géométriques limite à quatre le nombre de déformations distinctes que peut subir la grille d'échantillonnage au moyen de la méthode de distorsion géométrique. En utilisant cette dernière conjointement à celle de la distorsion arithmétique, le nombre total de déformations distinctes possibles est porté à 4×236, soit 944. La combinaison des deux méthodes de distorsion permet donc de multiplier, si nécessaire, par un facteur allant jusqu'à près de 1000 la taille de la base de données destinée à l'entraînement d'un classificateur, tout en évitant de devoir procéder à une validation manuelle des caractères générés. Quelques essais visuels préalables demeurent toutefois nécessaires, afin de déterminer l'amplitude maximale acceptable des paramètres de déformation.

6.5 Applications Pratiques

Les chiffres manuscrits contenus dans la base de données NIST3 proviennent, comme déjà précisé précédemment, des Etats-Unis d'Amérique, où la dispersion de styles d'écriture est bien moindre que celle rencontrée en pays européens. L'absence complète de certaines formes de chiffres (*figure 6.16*), très distinctes, d'un point de vue topologique, de celles rencontrées dans la base de données NIST3, rend les caractères contenus dans celle-ci insuffisants pour entraîner un système de reconnaissance destiné à être utilisé en pratique en nos contrées. Si aucun exemple de ces formes particulières n'a en effet été présenté au réseau de neurones lors de sa phase d'apprentissage, ce dernier sera fort probablement incapable de les reconnaître ultérieurement, tant il est vrai qu'un entraînement adéquat de ce type de classificateur est une procédure cruciale.

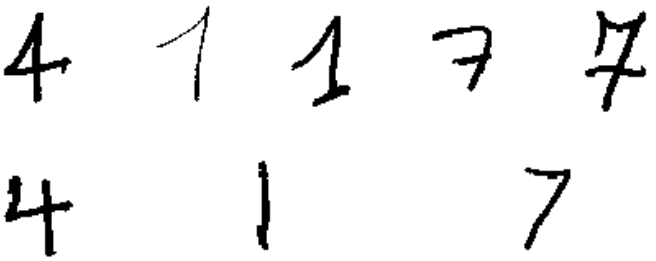


Figure 6.16 - Exemples de formes de chiffres européens (en haut), et leurs homologues de la base de donnée NIST3 (en bas).

Une acquisition manuelle de ces formes particulières de chiffres a donc été effectuée, en l'absence de contrainte d'écriture, auprès de 110 personnes distinctes. Cela a permis de recueillir 500 exemplaires de chaque chiffre, qui ont étés partagé par moitié pour l'apprentissage d'un perceptron multicouches et pour les tests de reconnaissance. Les caractères destinés à l'apprentissage ont ensuite subit chacun plusieurs séries de distorsions linéaires et géométriques. Les essais visuels préliminaires nous ont ici contraints à réduire la valeur du paramètre de distorsion linéaire à 20%, alors que celle du paramètre de distorsion géométrique a pu être maintenu à 1,0092. Les résultats de reconnaissance obtenus sur les caractères de test, pour différents nombres de caractères générés, et pour les deux méthodes d'extraction de primitives *Pixels Moyennés* et *Analyse Normalisée du Contour*, sont repris au tableau 6.III. L'augmentation significative du taux de reconnaissance observé confirme à nouveau l'efficacité, et surtout l'utilité, de la méthode de génération de caractères qui a été développée.

Afin d'obtenir un système de reconnaissance mixte, c'est-à-dire capable de reconnaître aussi bien des chiffres typés «Américains » que des chiffres «Européens », le plus étendu des ensembles d'entraînements générés a été complété d'une proportion équivalente de caractères issus de la base de données NIST3. L'ensemble d'apprentissage contenait ainsi 5000 exemplaires de chaque chiffre. Les taux de reconnaissance mesurés sur un ensemble de test comprenant des caractères des deux catégories sont repris au tableau 6.IV. Malgré la faible diminution des valeurs des taux de reconnaissance absolus, la diversité des écritures qui peuvent être reconnues à présent (*figure 6.17*) en font un système qui est tout à fait susceptible d'être utilisé dans des applications pratiques. La *figure 6.18* illustre des exemples de caractères qui ne sont toujours pas reconnus.

Catégorie de primitives	Nombre de prototypes de chaque classe	Taille du perceptron multicouches	Taux de reconnaissance
PM	250	38 x 60 x 10	86,0 %
PM	500	40 x 60 x 10	87,6 %
PM	1250	39 x 90 x 10	89,2 %
PM	2500	40 x 120 x 10	90,1 %
ANC	250	67 x 30 x 10	87,9 %
ANC	500	65 x 30 x 10	89,4 %
ANC	1250	66 x 60 x 10	90,8 %
ANC	2500	67 x 60 x 10	91,9 %

Table 6.III - Application de la génération de caractères pour des chiffres « Européens »; la deuxième colonne indique le nombre de prototypes de chaque classe que contenait l'ensemble d'apprentissage, les 250 premiers étant les exemplaires originaux, et les autres étant des exemplaires générés à partir de ceux-ci.

Primitives	Taille du perceptron multicouches	Taux de reconnaissance
PM	40 x 150 x 10	89,7 %
ANC	67 x 60 x 10	91,2 %

Table 6.IV - Résultats obtenus pour des systèmes entraînés sur base d'un ensemble d'apprentissage équilibré en caractères typés « Européens » et « Américains ».

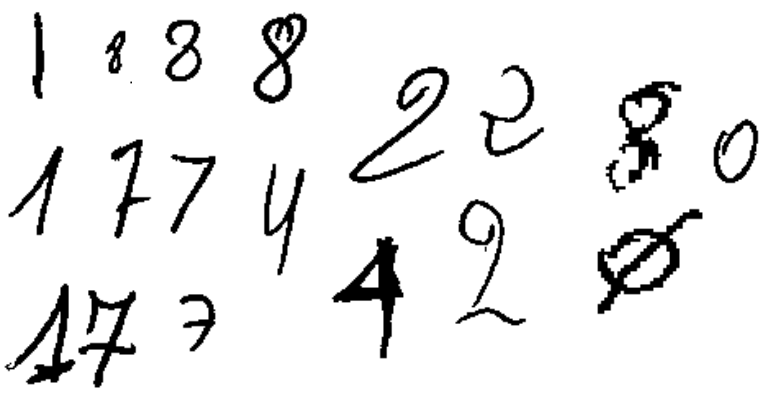


Figure 6.17 - Exemples de caractères correctement reconnus à l'aide d'un perceptron multicouches, entraîné à l'aide des primitives ANC, sur base d'un ensemble d'apprentissage contenant à la fois des caractères typés « Européens » et « Américains ».

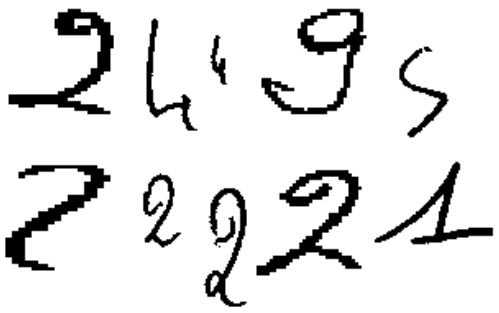


Figure 6.18 - Exemples de caractères qui ne sont toujours pas correctement reconnus.

La méthode de génération de caractères qui a été développée a également permis d'entraîner efficacement un système de reconnaissance de lettres majuscules manuscrites. L'ensemble d'apprentissage était alors constitué de 250 prototypes réels de chaque classe, qui ont chacun permis de générer quatre exemplaires artificiels, et de 1250 autres exemplaires provenant de la base de données NIST3. Le nombre total de prototypes de chaque lettre que contenait l'ensemble d'apprentissage s'élevait donc à 2500. L'ensemble de tests contenait, quant à lui, 950 autres exemplaires de chaque lettre, sur lesquels un taux de reconnaissance de **93,1%** a pu être mesuré, sur base des primitives issues de *l'Analyse Normalisée du Contour*.

Un dernier système de reconnaissance, destiné à la reconnaissance simultanée des 26 lettres majuscules et des 10 chiffres, a également été entraîné. L'ensemble d'apprentissage contenait alors 2500 exemplaires de chaque classe, dont la moitié provenait de la base de donnée NIST3.

La méthode de *L'Analyse Normalisée du Contour* a été utilisée pour l'extraction de caractéristiques, et les résultats ainsi obtenus sont résumés au tableau 6.V. Ce dernier reprend également, à titre comparatif, les meilleurs résultats obtenus à l'aide des systèmes entraînés spécifiquement sur des lettres ou sur des chiffres. La dernière colonne de ce tableau indique en outre le taux de reconnaissance mesuré lorsque la catégorie *lettre* ou *chiffre* du caractère d'entrée n'est pas précisée au réseau de neurones. L'effondrement des taux de reconnaissance qui en résulte alors est simplement dû à l'apparition de nombreuses confusions, dont les principales sont reprises au tableau 6.VI. Il est important de remarquer que ces confusions sont très souvent inévitables pour l'homme également, en l'absence de toute information contextuelle.

Ensemble d'entraînement	Ensemble de Test	Taille du perceptron Multicouches	Taux de reconnaissance	Taux de reconnaissance si catégorie non précisée
lettres+chiffres	lettres	71 x 90 x 36	92,0 %	84,5 %
lettres+chiffres	chiffres	71 x 90 x 36	88,9 %	78,9 %
lettres	lettres	72 x 90 x 26	93,1 %	-
chiffres	chiffres	67 x 60 x 10	91,2 %	-

Table 6.V - Résumé des performances de reconnaissance des différents systèmes développés.

Caractère d'entrée	Taux de reconnaissance	Confusion principale
chiffre 0	56 %	lettre O, 32 %
chiffre 5	68 %	lettre S, 19 %
lettre O	73 %	chiffre 0, 15 %
chiffre 6	77 %	lettre G, 12 %
chiffre 1	74 %	lettre I, 10 %
chiffre 2	70 %	lettre Z, 9 %
lettre S	84 %	chiffre 5, 8 %
lettre Z	86 %	chiffre 2, 8 %

Table 6.VI - Confusions les plus fréquentes se produisant lors de la reconnaissance simultanée de chiffres et de lettres.

6.6 Résumé

Deux méthodes de distorsion d'images de caractères manuscrits ont été présentées. Utilisées conjointement, les méthodes de distorsion linéaire et géométrique permettent, à partir d'exemplaires réels de caractères manuscrits, de générer d'autres prototypes, et d'augmenter ainsi la taille de la base de données destinée à l'entraînement d'un réseau de neurones en classificateur. Les résultats des tests effectués ont confirmé que les capacités de généralisation de ce dernier étaient ainsi améliorées.

En utilisant systématiquement une amplitude maximale des paramètres de déformation, ces méthodes assurent en outre la création de caractères suffisamment distincts des caractères initiaux, pour apporter une information utile supplémentaire. Quelques essais visuels doivent toutefois être préalablement menés, afin de fixer la valeur maximale des paramètres de

déformation, et d'éviter la création de caractères n'étant plus assez représentatifs de la classe dont ils sont issus.

Cette méthode de génération artificielle de caractères a permis d'entraîner efficacement divers systèmes de reconnaissance de caractères manuscrits non contraints, qui, en étant capables de traiter correctement une diversité très étendue de styles d'écritures, possèdent de bonnes potentialités d'utilisation pratique.

Chapitre 7

La Mise en Collaboration de Perceptrons Multicouches

7.1 Introduction

Des deux méthodes d'extraction de primitives qui se sont révélées les plus efficaces pour la reconnaissance hors-ligne de lettres et de chiffres manuscrits extraits de la base de données NIST3 (*§5.5.3*), celle de l'Analyse Normalisée du Contour permet d'atteindre un taux de reconnaissance plus élevé, pour un réseau de neurones de dimensions plus réduites (*Table 7.I*). Tant en termes de qualité que de vitesse de reconnaissance, cette méthode s'avère donc être plus performante. Cependant, lors des tests de reconnaissance, il est apparu que les deux réseaux de neurones produisent rarement les mêmes erreurs. Ainsi, lorsque l'un des perceptrons multicouches est incapable de classifier correctement un caractère, le second d'entre eux propose souvent la solution correcte.

A titre d'exemple, la *figure 7.1* reprend les matrices de confusion relevées pour chacun de ces systèmes de reconnaissance. Ces matrices de confusion contiennent, pour chaque classe possible du caractère d'entrée, la fréquence avec laquelle chaque autre classe est proposée, en sortie du classificateur, comme étant celle du caractère d'entrée. La comparaison de ces matrices montre que les confusions commises par les deux systèmes de reconnaissance n'apparaissent pas forcément entre les mêmes classes, et qu'il est dès lors possible qu'une mise en collaboration des deux réseaux de neurones puisse conduire à une réduction du taux global d'erreur. Diverses

méthodes visant à permettre aux perceptrons multicouches de collaborer entre eux, ont donc été développées en ce sens, et constituent l'objet de ce chapitre.

Base de données	Catégorie de primitives	Taille du perceptron multicouches	Durée de reconnaissance	Taux de reconnaissance
chiffres	PM	46 x 120 x 10	32,7 ms	97,8 %
chiffres	ANC	69 x 60 x 10	22,2 ms	98,7 %
lettres	PM	43 x 120 x 26	34,4 ms	94,5 %
lettres	ANC	72 x 90 x 26	34,2 ms	96,2 %

Table 7.I - Résumé des performances atteintes sur des chiffres extraits de la base de données NIST3. Afin de pouvoir comparer ces systèmes à ceux développés dans les sections qui suivent, le temps de reconnaissance a chaque fois été mesuré à l'aide d'un P.C. équipé d'un processeur 486DX porté à 66 MHz.

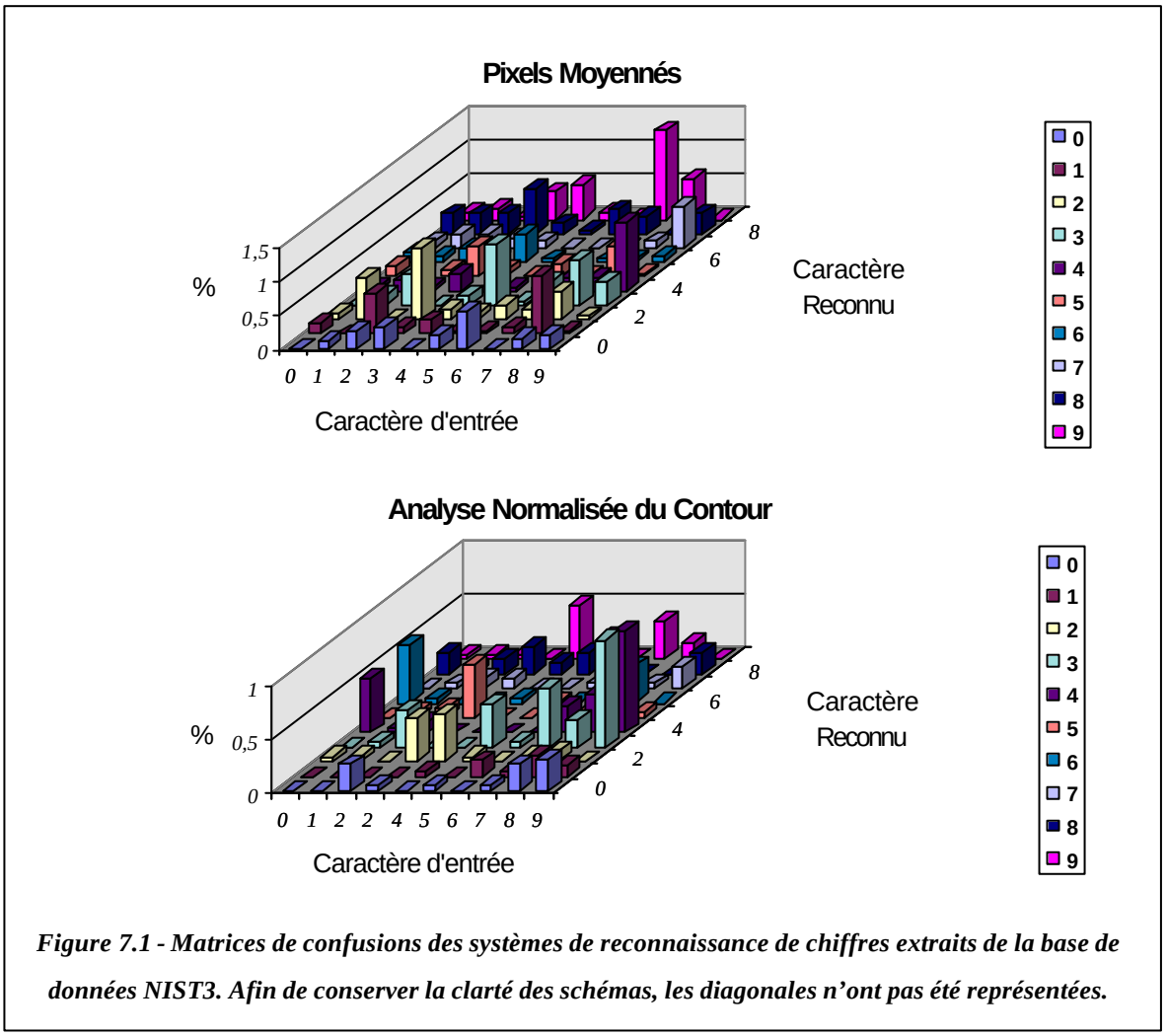


Figure 7.1 - Matrices de confusions des systèmes de reconnaissance de chiffres extraits de la base de données NIST3. Afin de conserver la clarté des schémas, les diagonales n'ont pas été représentées.

7.2 La Coopération en Parallèle

7.2.1 Coopération par la Couche Cachée

La première des méthodes de mise en collaboration consiste à réunir les deux réseaux de neurones en un seul, en les combinant par l'intermédiaire de leur couche cachée, puisque c'est celle-ci qui extrait toute l'information discriminante nécessaire à la reconnaissance (*figure 7.2*). Une nouvelle phase d'apprentissage doit alors être menée, au cours de laquelle seuls les poids des neurones de la nouvelle couche de sortie sont modifiés. Les poids synaptiques des neurones des couches cachées restent fixés à leur valeur, de manière à conserver les mêmes informations discriminantes que précédemment. Cette seconde procédure d'apprentissage est donc relativement rapide et ne requiert qu'un nombre restreint d'itérations [Gosselin,95].

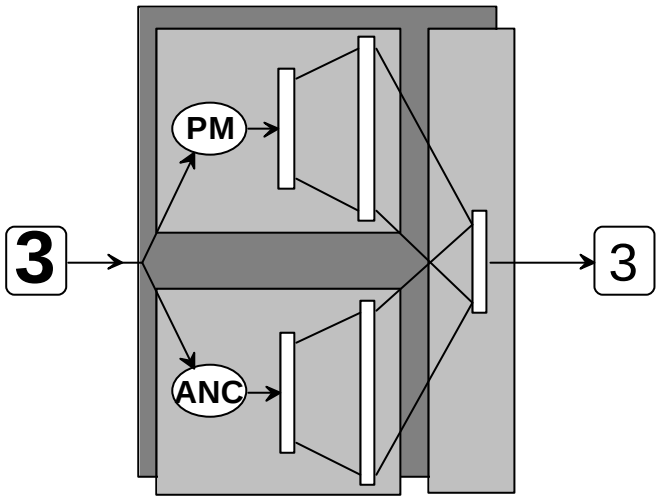


Figure 7.2 - Coopération par l'intermédiaire de la couche cachée.

Le *tableau 7.II* reprend les performances obtenues pour les réseaux de neurones entraînés à la reconnaissance des 10 chiffres d'une part, et à la reconnaissance des 26 lettres latines

[Gosselin,95] **B. Gosselin**
Neural Networks Combination for Improved Handwritten Characters Recognition
Proc. of Int. Conf. on Signal and Image Processing, pp 144-146, Las Vegas, Nevada, Novembre 1995.

majuscules d'autre part. Dans les deux cas, une diminution du taux d'erreur de classification a été obtenue.

Base de Données	Taille du perceptron multicouches	Durée de reconnaissance	Taux de reconnaissance
Chiffres	(46 x 120 + 69 x 60) x 10	53,8 ms	99,8 %
Lettres	(43 x 120 + 69 x 90) x 26	68,6 ms	96,6 %

Table 7.II - Performances de reconnaissance obtenues lors de la mise en coopération de perceptrons multicouches par l'intermédiaire de leur couche cachée.

7.2.2 Coopération par la Couche de Sortie

Une deuxième méthode de mise en coopération de perceptrons multicouches se base uniquement sur les vecteurs de sortie des réseaux de neurones, et consiste à en effectuer le produit terme à terme (*figure 7.3*). Cette méthode est extrêmement simple et rapide à mettre en oeuvre, puisqu'aucune nouvelle procédure d'apprentissage n'est à présent requise. Les résultats atteints sont résumés au *tableau 7.III*.

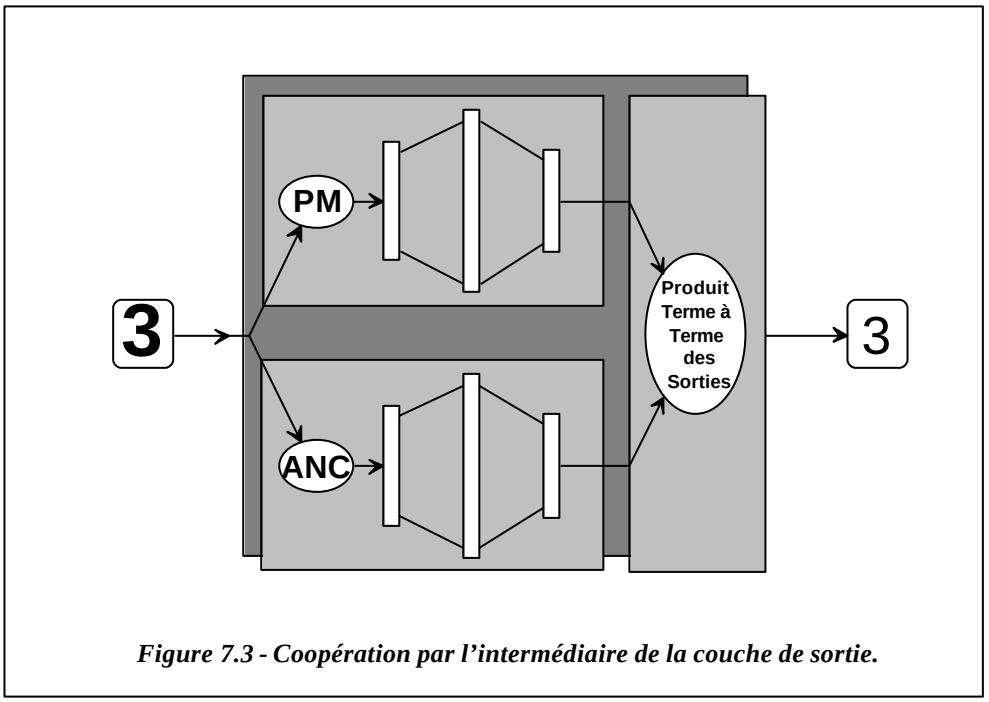


Figure 7.3 - Coopération par l'intermédiaire de la couche de sortie.

Base de Données	Taille du perceptron multicouches	Durée de reconnaissance	Taux de reconnaissance
Chiffres	46 x 120 x 10 + 69 x 60 x 10	53,8 ms	99,6 %
Lettres	43 x 120 x 26 + 72 x 90 x 26	68,7 ms	96,7 %

Table 7.III - Performances de reconnaissance obtenues lors de la mise en coopération de perceptrons multicouches par l'intermédiaire de leur couche de sortie.

La différence entre les taux de reconnaissance obtenus ici et ceux atteints au moyen de la première méthode de coopération est peu significative, et les deux manières de procéder peuvent être considérées comme étant équivalentes.

7.2.3 Optimisation de la Coopération

Si elle permet de diminuer le nombre d'erreurs de classification, la coopération en parallèle de perceptrons multicouches est aussi très coûteuse en termes de temps de calcul. Le système de reconnaissance illustré à la *figure 7.3* se révèle ainsi être plus lent que le plus efficace des réseaux de neurones individuels d'un facteur supérieur à **2,5** dans le cas des chiffres, et d'un facteur supérieur à **2** dans le cas des lettres.

Un premier processus qui peut être mené, afin de diminuer le temps nécessaire à la reconnaissance, est de tenter de réduire les dimensions des réseaux de neurones utilisés. Lors de l'apprentissage des perceptrons multicouches de manière individuelle, une croissance excessive du nombre de leurs unités cachées peut en effet s'être produite, afin de compenser les lacunes des primitives propres à chacun des systèmes. Il est dès lors probable que la mise en coopération de deux réseaux de neurones de plus faible taille permette d'obtenir également un taux de reconnaissance élevé.

Ces perceptrons multicouches de plus faibles dimensions sont en outre déjà disponibles, et résultent des multiples entraînements auxquels il a été nécessaire de procéder antérieurement, à la fois afin de sélectionner un nombre opportun de caractéristiques discriminantes à prendre en considération, et afin de déterminer le nombre d'unités cachées le plus adéquat. La recherche de la combinaison optimale de deux perceptrons multicouches est donc très aisée à mener, puisqu'aucune nouvelle phase d'apprentissage supplémentaire n'est requise. Le tableau 7.IV reprend les meilleurs résultats qui ont alors été obtenus.

Base de Données	Type de coopération	Taille du perceptron multicouches	Durée de reconnaissance	Taux de reconnaissance
Chiffres	couche cachée	46 x 60 x 10 + 48 x 30 x 10	36,6 ms	99,8 %
Chiffres	couche de sortie	46 x 60 x 10 + 48 x 30 x 10	36,6 ms	99,8 %
Lettres	couche cachée	43 x 90 x 26 +72 x 60 x 26	53,3 ms	96,6 %
Lettres	couche de sortie	43 x 90 x 26 + 72 x 60 x 26	53,4 ms	96,7 %

Table 7.IV - Maintien des taux de reconnaissance et diminution du volume de calcul, grâce à une mise en coopération de réseaux de neurones de plus faibles dimensions.

Pour des taux de reconnaissance équivalents à ceux atteints au moyen des systèmes précédents, le temps de reconnaissance a été divisé d'un facteur **1,5** dans le cas des chiffres, et d'un facteur **1,3** dans le cas des lettres. Par rapport au meilleur des perceptrons multicouches pris isolément, le temps de reconnaissance n'est plus, respectivement, que **70%** supérieur, et **60%** supérieur.

7.3 La Coopération en Cascade

Lors de la coopération en parallèle de réseaux de neurones entraînés sur base de catégories distinctes de primitives, ceux-ci sont tous deux utilisés systématiquement en phase de reconnaissance, ce qui demeure exigeant en termes de ressources de calcul. L'évolution du taux d'erreur de chacun des deux réseaux de neurones pris isolément indique cependant que celui-ci décroît très rapidement en fonction du taux de rejet toléré (*figure 7.4*). Dans le cas du problème de la reconnaissance de chiffres, par exemple, le seul réseau de neurones qui utilise les primitives issues de l'Analyse Normalisée du Contour permet ainsi d'atteindre un taux d'erreur équivalent à celui observé lorsque les deux perceptrons multicouches coopèrent en parallèle, pour un taux de rejet de **10%** seulement.

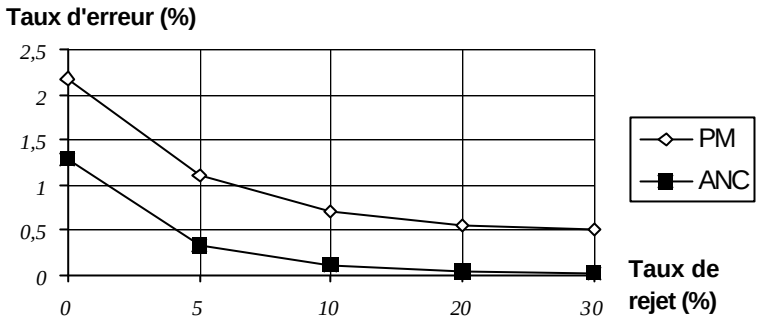


Figure 7.4 - Evolution du Taux de Reconnaissance en Fonction du Taux de Rejet toléré, dans le cas du problème de la reconnaissance de chiffres.

Une collaboration *en cascade* des réseaux de neurones peut alors être envisagée: seul le perceptron multicouches entraîné au moyen des primitives fournies par l'Analyse Normalisée du Contour est utilisé dans un premier temps, et ce n'est que lorsque le niveau de ses sorties s'avère insuffisant qu'une mise en collaboration des deux réseaux de neurones, par l'intermédiaire de leur couche cachée ou par celui de leur couche de sortie, est appliquée pour achever la reconnaissance (*figure 7.5*) [Gosselin,96a].

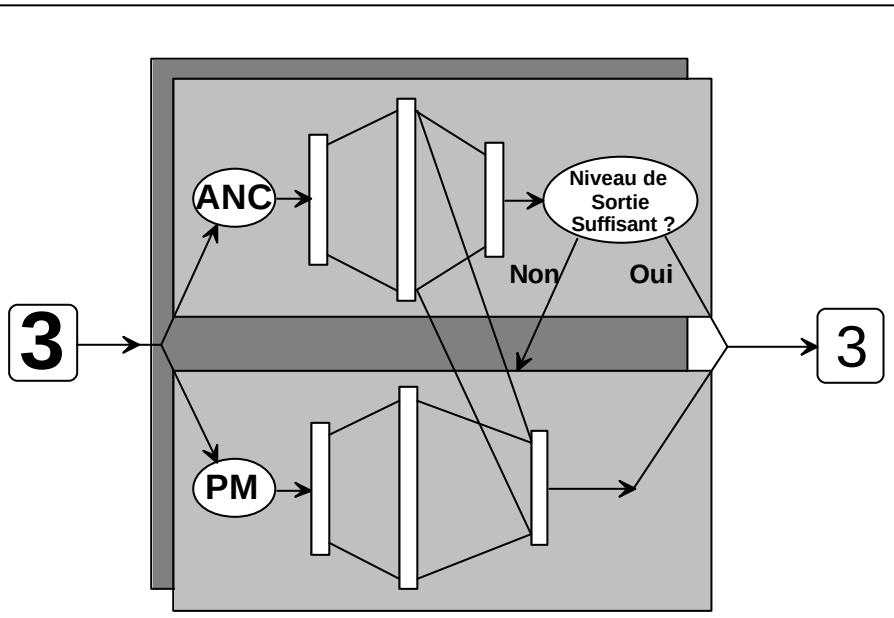


Figure 7.5 - Principe de la coopération en cascade de perceptrons multicouches.

[Gosselin,96a] **B. Gosselin**
Coopération Optimisée de Réseaux de Neurones pour la Reconnaissance de Caractères Manuscrits
Actes du 10ème Congrès Reconnaissance des Formes et Intelligence Artificielle, vol 2, pp 659-664, Rennes, France, Janvier 1996

Le taux de reconnaissance observé, sur les caractères qui doivent à présent être reconnus au moyen des deux perceptrons multicouches simultanément, est plus faible que celui établi à la section précédente, puisque ces caractères, ayant été rejetés par un des réseaux de neurones, sont ceux les plus critiques à reconnaître. De manière à maintenir un taux de reconnaissance global aussi élevé que celui atteint précédemment, le taux de rejet du premier réseau de neurones doit donc être tel que l'augmentation du taux de reconnaissance mesuré à la sortie de celui-ci compense la diminution du taux de reconnaissance obtenu lorsque les deux perceptrons multicouches sont utilisés simultanément. Une première conséquence de ceci est qu'il n'y a aucune assurance que le taux de reconnaissance global puisse être maintenu à sa plus haute valeur, et que la valeur optimale du seuil de rejet du premier réseau de neurones doit être déterminée empiriquement. La seconde conséquence de l'écart qui apparaît entre le taux de reconnaissance mesuré à la sortie du premier perceptron multicouches et celui relevé lors de la collaboration de l'ensemble des deux réseaux de neurones est, qu'en pratique, plus un caractère sera reconnu rapidement, et plus la probabilité sera élevée que la classe proposée soit correcte.

Quel que soit le mode de coopération des perceptrons multicouches lors de la reconnaissance de caractères rejetés par le premier d'entre eux, le meilleur résultat a été chaque fois obtenu à l'aide de la même paire de réseaux de neurones que celle déterminée à l'issue de la section précédente [Gosselin,96b]. Ces résultats sont repris au tableau 7.V, ainsi qu'aux *figures 7.6* et *7.7*.

Base de Données	Type de coopération	Taux d'accès au second réseau de neurones	Durée de reconnaissance	Taux de reconnaissance
Chiffres	couche cachée	15,3 %	14,7 ms	99,8 %
Chiffres	couche de sortie	15,3 %	14,7 ms	99,8 %
Lettres	couche cachée	14,9 %	27,6 ms	96,7 %
Lettres	couche de sortie	14,9 %	27,7 ms	96,7 %

Table 7.V - Maintien des taux de reconnaissance et diminution du volume de calcul, grâce à une mise en coopération de réseaux de neurones de plus faibles dimensions.

[Gosselin,96b] **B. Gosselin**
Optimal Neural Networks Combination for Handwritten Character Recognition
Proc. EUSIPCO'96, Trieste, Italie, Septembre 1996 (à paraître)

Pour un taux de reconnaissance équivalent à celui obtenu en utilisant systématiquement les deux réseaux de neurones, un seul d'entre eux suffit, pendant près de **85%** du temps en moyenne. Dans le cas des chiffres, la vitesse moyenne de reconnaissance a ainsi été augmentée d'un facteur **2,5** par rapport au système mis au point à la section 7.2.3, et est à présent **50%** plus élevée que celle atteinte par le meilleur des perceptrons multicouches pris individuellement! En ce qui concerne le problème de la reconnaissance des lettres, le système ici conçu est **2** fois plus rapide que celui qui utilise systématiquement les deux réseaux de neurones, et près de **25%** plus rapide que le plus efficace des perceptrons multicouches considéré isolément.

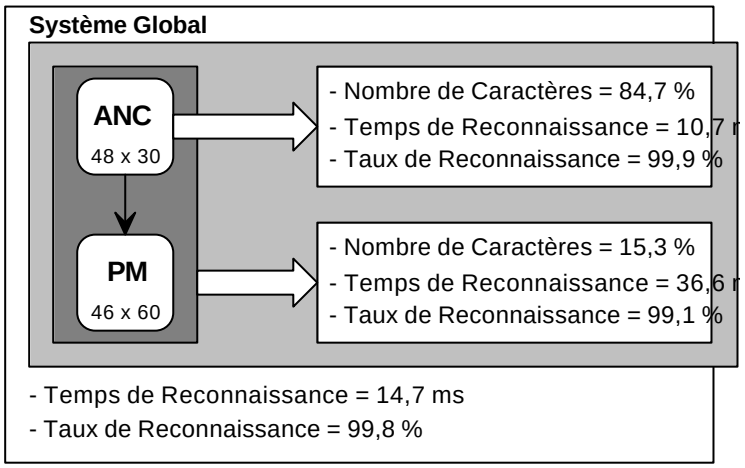


Figure 7.6 - Résultats de la mise en coopération en cascade de perceptrons multicouches pour la reconnaissance des chiffres.

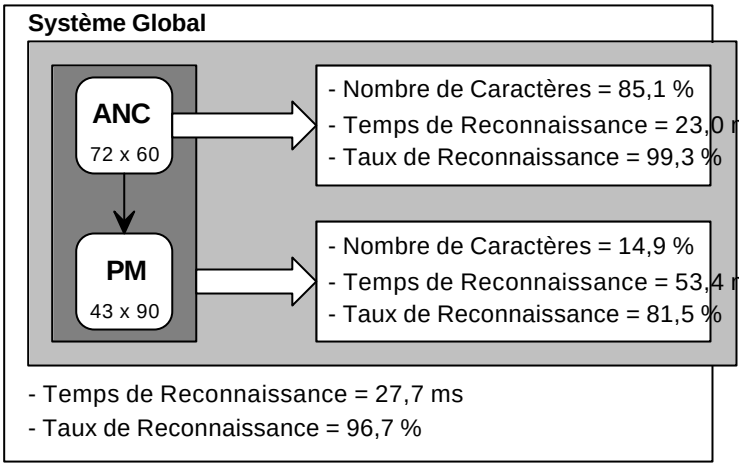


Figure 7.7 - Résultats de la mise en coopération en cascade de perceptrons multicouches pour la reconnaissance des lettres.

7.4 La Substitution de Réseaux de Neurones

La combinaison par substitution de plusieurs systèmes de reconnaissance est une méthode qui peut également être envisagée en vue d'obtenir une réduction supplémentaire de la durée moyenne de reconnaissance. Cette procédure consiste à essayer d'utiliser, aussi souvent que possible, un système de reconnaissance extrêmement rapide, destiné à la classification des caractères les mieux formés, auquel est substitué, en cas de rejet d'un caractère, un autre système de reconnaissance, offrant une meilleure qualité de classification de manière à conserver un taux de reconnaissance global élevé. Le temps consacré par le système rapide à essayer de reconnaître le caractère, est cependant perdu en cas de rejet de celui-ci, et il n'y a donc aucune assurance que la durée moyenne de reconnaissance puisse être diminuée. Les performances globales dépendent alors essentiellement du rapport entre la vitesse et la qualité (taux de rejet) du système de reconnaissance rapide [Gosselin,96c].

[Gosselin,96c] **B. Gosselin**
Multilayer Perceptrons Combination Applied to Handwritten Characters Recognition
in Neural Processing Letters (à paraître)

Quelques essais ayant rapidement montré qu'à volume de calcul égal, un classificateur conventionnel offre une moins bonne qualité de reconnaissance qu'un perceptron multicouches de faibles dimensions, le système rapide de reconnaissance est également basé sur un tel réseau de neurones. En outre, les primitives issues de l'Analyse Normalisée du Contour étant celles permettant d'atteindre les taux de reconnaissance les plus élevés, c'est un perceptron multicouches entraîné sur base de celles-ci qui est utilisé. C'est naturellement au système mis au point à la section précédente que revient la tâche d'assurer le maintien d'un taux de reconnaissance global élevé. Comme le mode de coopération par l'intermédiaire de la couche cachée s'est précédemment révélé être équivalent à celui de coopération par l'intermédiaire de la couche de sortie, c'est ce dernier, plus simple à mettre en oeuvre, qui a été appliqué. Les meilleurs résultats alors obtenus sont repris au tableau 7.VI.

Base de données	Perceptron multi-couches rapide	Taux d'accès au $2^{ème}$ réseau	Taux d'accès au $3^{ème}$ réseau	Durée moyenne de reconnaissance	Taux de reconnaissance
chiffres	69 x 10 x 10	43,5 %	16,4 %	12,7 ms	99,8 %
lettres	72 x 30 x 26	23,0 %	11,8 %	20,7 ms	96,7 %

Table 7.VI - Coopération par substitution de systèmes de reconnaissance.

Une diminution relative de **14%** de la durée moyenne de reconnaissance a pu être obtenue dans le cas du problème de la reconnaissance de chiffres. Cette réduction s'élève à **25%** dans le cas du problème de la reconnaissance de lettres. La taille plus importante du réseau de neurones préliminaire qui est alors utilisé, est largement compensée par la fréquence plus limitée des accès aux deuxième et troisième perceptrons multicouches.

Les *figures 7.8* et *7.9* illustrent la structure réactualisée des deux systèmes de reconnaissance. La même remarque que précédemment peut à nouveau être formulée ici: la classification de caractères rejetés par un système de reconnaissance étant plus critique, le taux de reconnaissance observé sur ceux-ci diminue. De manière à compenser cette diminution et à maintenir le taux de reconnaissance global à sa plus grande valeur, le taux de reconnaissance relevé sur les caractères non rejetés doit être supérieur à cette dernière. Il en résulte donc que plus un caractère est reconnu rapidement, plus la probabilité est élevée que la classe proposée soit correcte.

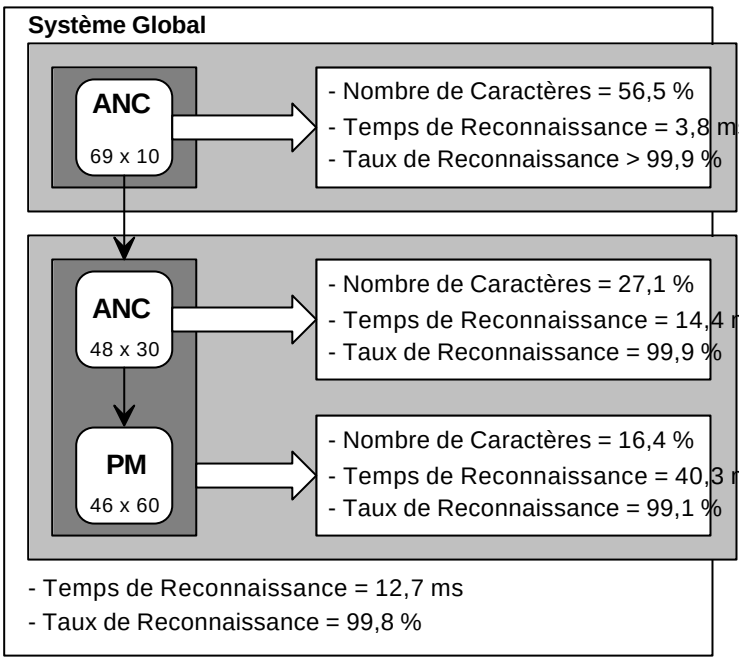


Figure 7.8 - Substitution de réseaux de neurones pour la reconnaissance de chiffres.

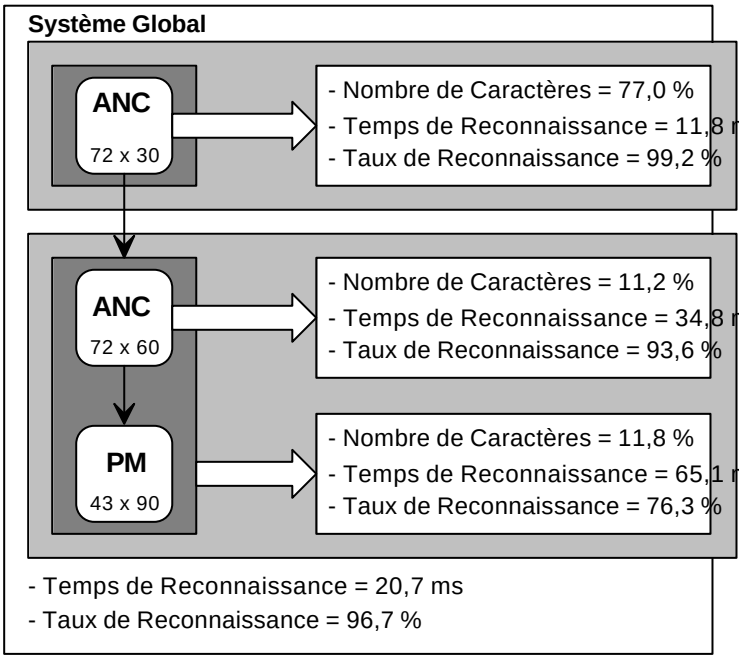


Figure 7.9 - Substitution de réseaux de neurones pour la reconnaissance de lettres.

Une amélioration supplémentaire de la vitesse de reconnaissance peut être obtenue en appliquant le principe d'une coopération en cascade au système de reconnaissance rapide également. Une telle association de deux perceptrons multicouches de faible taille, entraînés sur base de primitives distinctes, devrait permettre de réduire de manière significative le taux de rejet de ce système de reconnaissance, au prix d'un faible accroissement de son temps de calcul.

Le tableau 7.VII ne reprend que le meilleur résultat atteint dans le cas du problème de la reconnaissance de chiffres, aucune amélioration n'ayant pu être obtenue dans le cas du problème de la reconnaissance de lettres.

Système rapide de reconnaissance	Taux d'accès au $2^{ème}$ réseau	Taux d'accès au $3^{ème}$ réseau	Taux d'accès au $4^{ème}$ réseau	Durée moyenne de reconnaissance	Taux de reconnaissance
69 x 10 x 10 + 46 x 10 x 10	43,5 %	8,5 %	5,5 %	8,2 ms	99,8 %

Table 7.VII - Second mode de coopération par substitution de systèmes de reconnaissance.

La coopération en cascade de deux perceptrons multicouches, contenant chacun 10 unités cachées, a ainsi permis de réduire à **8,5%** seulement le taux d'accès au système de reconnaissance élaboré à la section 7.3. Le système de reconnaissance ici conçu est alors **80%** plus rapide que celui-ci, et **2,7** fois plus rapide que le plus efficace des perceptrons multicouches individuels, tout en offrant un taux de reconnaissance global de **99.8%**. Sa structure est illustrée à la *figure 7.10*.

Le processus de substitution de systèmes de reconnaissance peut être répété tant qu'il permet de réduire la durée moyenne de reconnaissance et de maintenir le taux de reconnaissance à sa plus haute valeur. La forte dégradation de la qualité de reconnaissance, qui s'est ici produite pour des perceptrons multicouches comportant moins de 10 unités cachées n'a toutefois pas autorisé de réduction supplémentaire de la durée moyenne de reconnaissance. De même, aucune amélioration n'a pu être constatée en insérant, dans la chaîne de substitution, des réseaux de neurones comportant entre 10 et 30 unités cachées, eu égard au temps de calcul élevé requis par ceux-ci, relativement à la qualité de reconnaissance qu'ils permettent d'atteindre. La substitution en cascade de plus de deux systèmes de reconnaissance n'a donc pas ici rendu possible une nouvelle augmentation de la vitesse moyenne de reconnaissance.

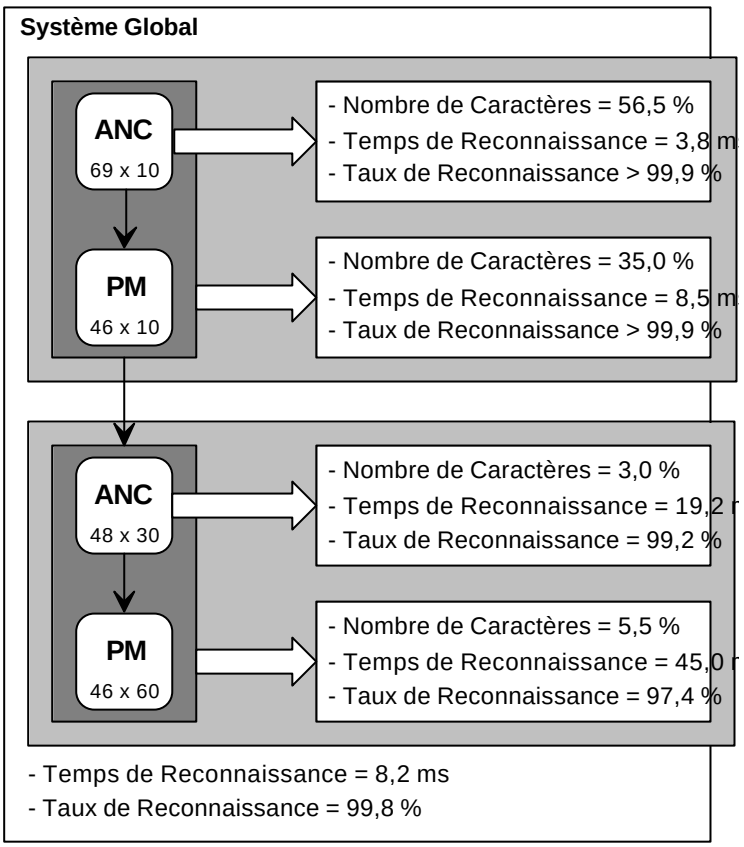


Figure 7.10 - Mise en coopération de quatre perceptrons multicouches pour la reconnaissance de chiffres manuscrits extraits de la base de données NIST3.

Les *figures 7.11* et *7.12* illustrent respectivement des exemples de chiffres et de lettres qui n'étaient pas reconnus à l'aide du meilleur des perceptrons multicouches isolés, soit celui entraîné sur base des primitives issues de l'Analyse Normalisée du Contour, mais qui le sont correctement à l'aide des systèmes de reconnaissance combinés. La coopération avec des réseaux de neurones qui utilisent les primitives de type Pixels Moyennés rend à présent possible la reconnaissance de ces caractères, qui présentent un contour particulièrement déformé. Les *figures 7.13* et *7.14* illustrent, elles, des exemples de caractères qui ne sont toujours pas reconnus grâce à la mise en coopération de perceptrons multicouches.

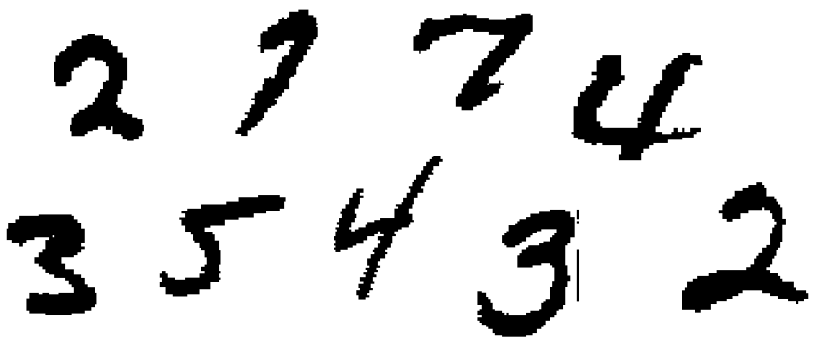


Figure 7.11 - Exemples de chiffres non reconnus à l'aide d'un seul perceptron multicouches, et reconnus correctement grâce à la mise en collaboration de plusieurs d'entre eux. Les classes correctes sont, de gauche à droite et de haut en bas: 2, 7, 7, 4, 3, 5, 4, 3, 2.

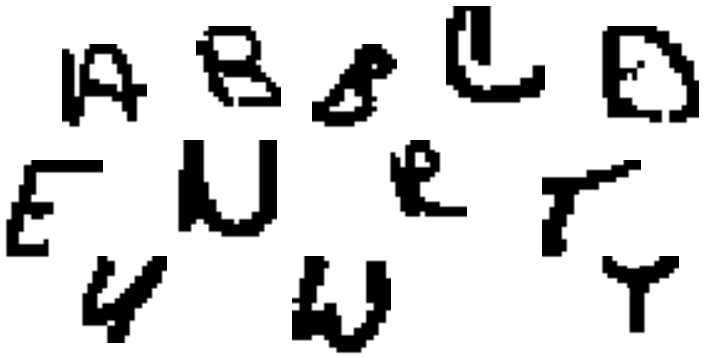


Figure 7.12 - Exemples de lettres non reconnues à l'aide d'un seul perceptron multicouches, et reconnues correctement grâce au système final de reconnaissance. Les classes correctes sont, de gauche à droite et de haut en bas: A, B, B, C, D, E, N, R, T, U, W, Y.

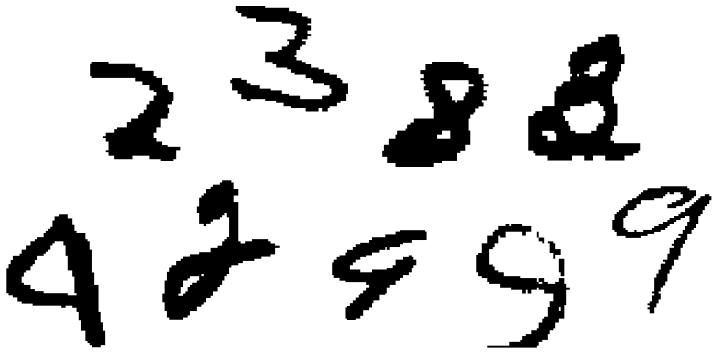


Figure 7.13 - Exemples de chiffres non reconnus à l'aide du système final.
Les classes proposées sont, de gauche à droite et de haut en bas (classes réelles entre parenthèses):
3 (2), 5 (3), 2 (8), 2 (3), 4 (9), 8 (2), 4 (9), 4 (9), et 4 (9).

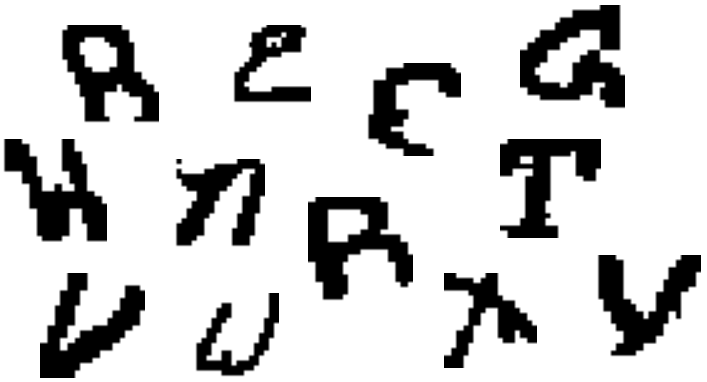


Figure 7.14 - Exemples de lettres non reconnues à l'aide du système final.
Les classes proposées sont, de gauche à droite et de haut en bas (classes réelles entre parenthèses):
A (B), C (E), C (E), Q (G), X (H), A (N), A (R), I (T), V (U), U (W), D (X), V (Y).

7.5 Applications Pratiques

7.5.1 Comportement vis-à-vis de Caractères Typographiques

En pratique, de nombreux documents peuvent comporter des caractères typographiques pré-imprimés, nécessitant ainsi un système de reconnaissance apte à traiter simultanément des caractères typographiques et manuscrits. Les deux systèmes de reconnaissance obtenus à l'issue du chapitre 6, et entraînés à la classification des diverses formes de chiffres qui peuvent se rencontrer en Europe, ont ainsi été testés pour la reconnaissance de chiffres typographiques imprimés dans la fonte OCR-B[1]. Le tableau 7.VIII compare les performances obtenues à l'aide de chacun de ces systèmes, pour la reconnaissance de caractères manuscrits et typographiques. Les taux de reconnaissance particulièrement faibles mesurés sur ces derniers confirment l'importance de l'ensemble d'apprentissage des réseaux de neurones, et l'intérêt qu'il y a à ce que ses éléments soient suffisamment représentatifs des formes à classifier en phase d'exploitation.

Type de chiffres	Catégorie de primitives	Taux de reconnaissance
Manuscrits	PM	89,7 %
Typographiques	PM	71,8 %
Manuscrits	ANC	91,2 %
Typographiques	ANC	82,5 %

Table 7.VIII - résultats de reconnaissance de chiffres manuscrits et typographiques à l'aide des systèmes entraînés à la classification de chiffres « Européens ».

Eu égard à la dispersion extrêmement faible de formes que présentent les caractères typographiques, l'apprentissage de perceptrons multicouches à la classification de ces derniers uniquement est bien moins coûteuse en temps de calcul qu'un nouvel entraînement de réseaux de neurones à la reconnaissance simultanée de caractères manuscrits et typographiques. La mise en coopération de perceptrons multicouches nouvellement entraînés à la reconnaissance de chiffres

[1] Cette fonte est utilisée pour pré-imprimer les bulletins de virements bancaires, par exemple.

typographiques et de ceux déjà mis au point pour la classification de chiffres manuscrits, pourrait alors permettre d'atteindre de bonnes performances de reconnaissance pour les deux types de caractères, au prix d'une plus grande rapidité de mise en oeuvre.

La meilleure qualité de reconnaissance étant obtenue au moyen de réseau de neurones exploitant les primitives fournies par la méthode de l'Analyse Normalisée du Contour, c'est sur base de celles-ci qu'un perceptron multicouches a été entraîné spécifiquement à la reconnaissance des chiffres typographiques. Les caractéristiques sélectionnées étaient, dans ce cas, les mêmes que celles prises en considération par le réseau de neurones entraîné à la classification de caractères manuscrits, et un perceptron multicouches ne comportant que 30 unités cachées, a ainsi permis d'observer un taux de reconnaissance de 100% sur un ensemble de test constitué de 200 exemplaires typographiques de chacun des dix chiffres.

Les résultats, atteints lors de la mise en coopération en parallèle de ce réseau avec celui entraîné à la reconnaissance de caractères manuscrits sur base de la même catégorie de primitives, sont repris au tableau 7.IX.

Type de Chiffres	Mode de Coopération	Taux de Reconnaissance
manuscrits	couche de sortie	82,0 %
typographiques	couche de sortie	100 %
manuscrits	couche cachée	89,3 %
typographiques	couche cachée	99,3 %

Table 7.IX - Coopération en parallèle de perceptrons multicouches entraînés indépendamment à la reconnaissance de caractères typographiques et manuscrits.

La décroissance des taux de reconnaissance qui se produit pour les chiffres manuscrits est due à une coopération systématique avec un réseau de neurones qui est surentraîné à la classification de caractères typographiques. Bien que, grâce à un ré-apprentissage des poids synaptiques des neurones de la dernière couche, cet effet soit moins marqué dans le cas de la mise en collaboration des perceptrons multicouches par l'intermédiaire de leur couche cachée, le taux de reconnaissance atteint demeure insuffisant, et une coopération par substitution des deux réseaux de neurones semble donc être plus pertinente.

Le tableau 7.X contient les résultats qui ont été obtenus lorsque c'est le perceptron multicouches entraîné à la classification de chiffres typographiques qui est utilisé en premier lieu. La valeur du seuil de rejet du premier réseau a alors été fixée de manière à maintenir le taux de reconnaissance obtenu sur les caractères manuscrits à sa plus haute valeur. Lorsque la coopération par substitution des deux réseaux de neurones a lieu en utilisant prioritairement celui entraîné à la classification de chiffres manuscrits, la valeur maximale du taux de reconnaissance obtenu pour ces derniers n'a pu être conservée qu'en évitant systématiquement d'accéder au second perceptron multicouches, ce qui ne présente aucun intérêt.

Type de chiffres	Taux d'accès au second réseau	Taux de reconnaissance
manuscrits	99,8 %	91,2 %
typographiques	70,7 %	90,6 %

Table 7.X - Coopération par substitution de perceptrons multicouches entraînés indépendamment à la reconnaissance de caractères typographiques et manuscrits.

Bien que les performances de reconnaissance atteintes sur les chiffres typographiques soient à présent plus élevées que sans coopération aucune des deux réseaux de neurones, elles demeurent très largement insuffisantes, compte tenu de la dispersion d'aspect extrêmement faible que présente ce type de caractères, et ne permettent pas d'envisager une application pratique de ce système de reconnaissance. Ceci nous a donc contraint à procéder à l'entraînement d'un seul perceptron multicouches, à la reconnaissance simultanée de caractères typographiques et manuscrits. Le tableau 7.XI reprend les meilleurs résultats obtenus pour chacune des méthodes d'extraction de primitives.

Les chiffres typographiques sont à présent correctement reconnus avec une probabilité très élevée, et le taux de reconnaissance observé pour les chiffres manuscrits a même été légèrement amélioré. Le fait que cette dernière augmentation se soit produite, indique que la base de données utilisée pour effectuer l'apprentissage des perceptrons multicouches n'est pas encore suffisamment étendue pour représenter l'ensemble des chiffres manuscrits de manière significative, et il est donc fortement probable que les performances de reconnaissance puissent encore être améliorées en augmentant la taille de cet ensemble d'apprentissage.

Type de chiffres	Catégorie de primitives	Dimensions du perceptron multicouches	Taux de reconnaissance
manuscrits	PM	40 x 150 x 10	90,3 %
typographiques	PM	40 x 150 x 10	98,9 %
manuscrits	ANC	67 x 150 x 10	91,9 %
typographiques	ANC	67 x 150 x 10	100 %

Table 7.XI - Résultats de l'entraînement de perceptrons multicouches à la reconnaissance simultanée de caractères typographiques et manuscrits.

7.5.2 Mises en Coopération de Perceptrons Multicouches

Comme pour les caractères extraits de la base de données NIST3, il apparaît ici encore que les erreurs de classification commises par les deux réseaux de neurones obtenus ne se produisent pas systématiquement pour les mêmes caractères, malgré l'existence d'une corrélation plus importante entre les confusions qui ont lieu (*figure 7.15*). Les diverses méthodes de mise en collaboration de perceptrons multicouches exposées aux sections 7.2 à 7.4 ont donc été appliquées, en vue de tenter de réduire ici également le taux global d'erreur. Une coopération par l'intermédiaire de la couche cachée s'étant toutefois révélée, dans le cas de réseaux de neurones entraînés à partir de caractères d'aspects comparables, équivalente à une coopération par l'intermédiaire de la couche de sortie, seul ce dernier mode a été envisagé. Les résultats obtenus pour des chiffres manuscrits sont repris au tableau 7.XII. Le taux de reconnaissance relevé pour des chiffres typographiques est, lui, demeuré supérieur à **99,9%** dans tous les modes de coopération.

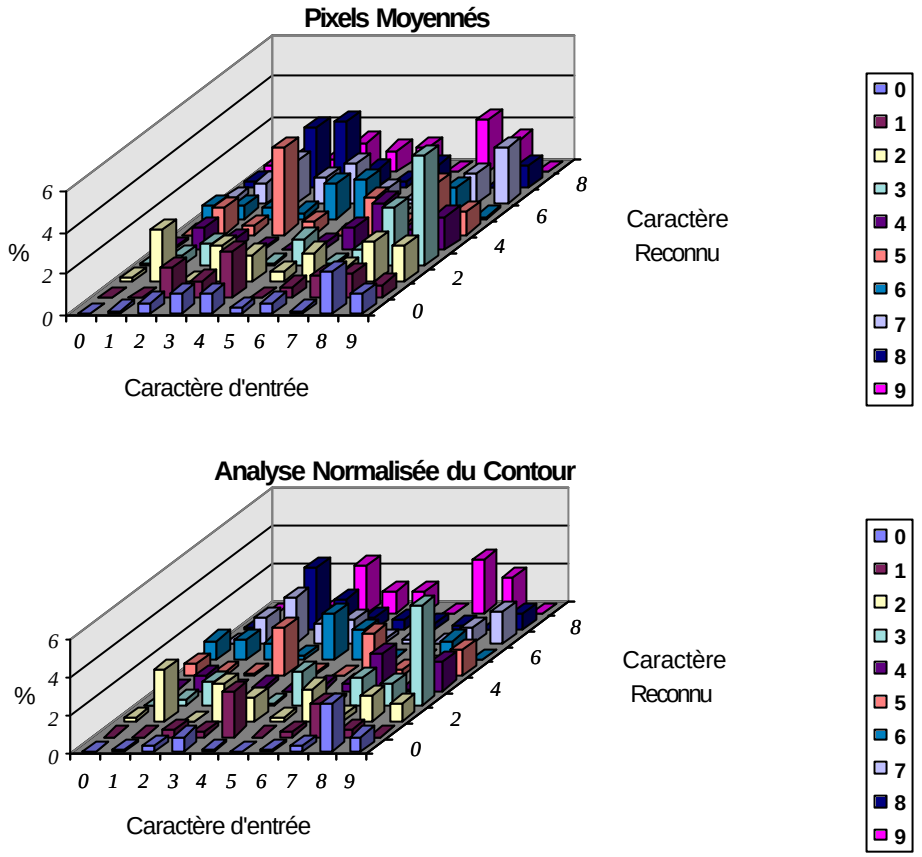


Figure 7.15 - Matrices de confusions des systèmes de reconnaissance de chiffres « Européens ». Afin de conserver la clarté des schémas, les diagonales n'ont pas été représentées.

Mode de coopération	Dimensions des perceptrons multicouches	Durée moyenne de reconnaissance	Taux de reconnaissance
PM seul	40 x 150 x 10	32,2 ms	90,3 %
ANC seul	67 x 150 x 10	42,9 ms	91,9 %
Parallèle	67 x 60 x 10 + 40 x 90 x 10	53,3 ms	95,1 %
Cascade	67 x 60 x 10 + 40 x 90 x 10	33,6 ms	95,3 %

Table 7.XII - Résultats de la mise en coopération de deux perceptrons multicouches pour la reconnaissance de chiffres manuscrits « Européens ».

La mise en collaboration des deux perceptrons multicouches a rendu possible une réduction des dimensions des réseaux de neurones utilisés, et la coopération en cascade a en outre permis de limiter à **38,7%** la fréquence d'utilisation du second réseau. Par rapport au meilleur des réseaux de neurones isolés, ce système de reconnaissance commet ainsi plus de **40%** d'erreurs de classification en moins, pour une vitesse moyenne de reconnaissance qui est de **28%** supérieure.

La légère augmentation du taux de reconnaissance qui apparaît en appliquant le mode de coopération en cascade, plutôt que celui de coopération en parallèle, s'explique par le fait que les valeurs de sortie du second perceptron multicouches peuvent parfois être telles que la collaboration des deux réseaux de neurones ne permet plus de conserver la solution pourtant correcte que proposait le premier d'entre eux. Ce phénomène se produit plus rarement lors d'une coopération en cascade des deux perceptrons multicouches, grâce à la restriction du nombre d'accès au second d'entre eux.

L'utilisation par substitution d'un système de reconnaissance rapide, constitué d'un seul réseau de neurones, préliminairement au système qui vient d'être conçu, n'a pas permis ici de réduire la durée moyenne de reconnaissance. Un temps moyen de 29,4 ms par caractère a par contre pu être obtenu, grâce à l'utilisation préliminaire d'un système de reconnaissance constitué de deux perceptrons multicouches de faibles dimensions, coopérant en cascade. La vitesse moyenne de reconnaissance est ainsi augmentée de **46%** par rapport à celle offerte par le meilleur des réseaux de neurones isolés, pour un taux de reconnaissance global de **95,3%**. Ces performances s'avèrent suffisantes pour envisager une mise en application pratique de ce système. Une telle application est d'ailleurs en cours de développement avec l'aide d'un partenaire industriel, et vise le domaine du traitement automatique de virements bancaires.

Dès qu'une structure de coopération de perceptrons multicouches est définie, chacun de ces derniers peut subir un entraînement spécifique, uniquement à partir des caractères de l'ensemble d'apprentissage initial qui sont rejetés par le système de reconnaissance qui le précède dans la chaîne de coopération. Ceci devrait permettre d'améliorer encore les performances, en termes de qualité et/ou de vitesse de reconnaissance.

La *figure 7.16* illustre des exemples de chiffres qui n'étaient pas correctement classifiés à l'aide du meilleur des perceptrons multicouches considérés isolément, mais qui le sont grâce à la mise en collaboration des réseaux de neurones. Des exemples de caractères qui ne peuvent toujours pas être correctement reconnus sont illustrés à la *figure 7.17*.

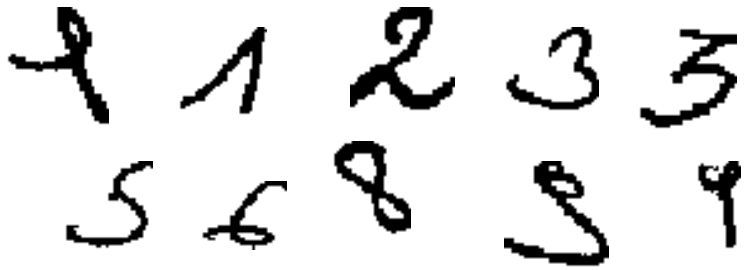


Figure 7.16 - Exemples de chiffres non correctement classifiés à l'aide du meilleur des perceptrons multicouches isolés, mais qui le deviennent grâce à la mise en coopération de réseaux de neurones. Les classes correctes sont (de gauche à droite et de haut en bas): 1, 1, 2,3 3, 5, 6, 8, 9, 9.

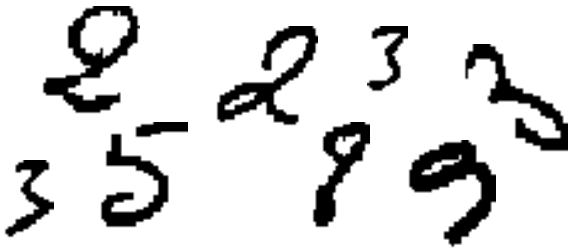


Figure 7.17 - Exemples de chiffres non correctement classifiés à l'aide du système final de reconnaissance.

La structure finale du système de reconnaissance ici conçu est illustrée à la *figure 7.18*. La répartition des taux d'accès aux divers réseaux de neurones, en fonction de la classe des caractères d'entrée, est représentée à la *figure 7.19*. La *figure 7.20* illustre la répartition des taux de reconnaissance par classe, pour les deux meilleurs perceptrons multicouches isolés, et pour la plus performante des méthodes de coopération. La *figure 7.21* compare l'évolution des taux de reconnaissance en fonction du taux de rejet pour ces trois systèmes. Enfin, la matrice de confusion de l'ultime système de reconnaissance est reprise à la *figure 7.22*.

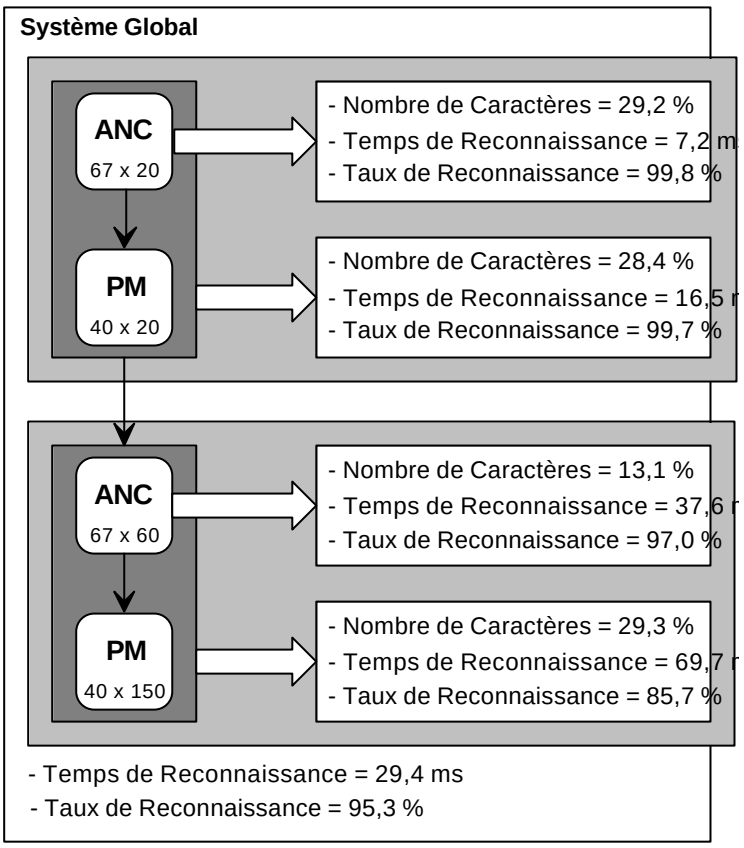


Figure 7.18 - Structure du système final de reconnaissance de chiffres manuscrits « Européens ».

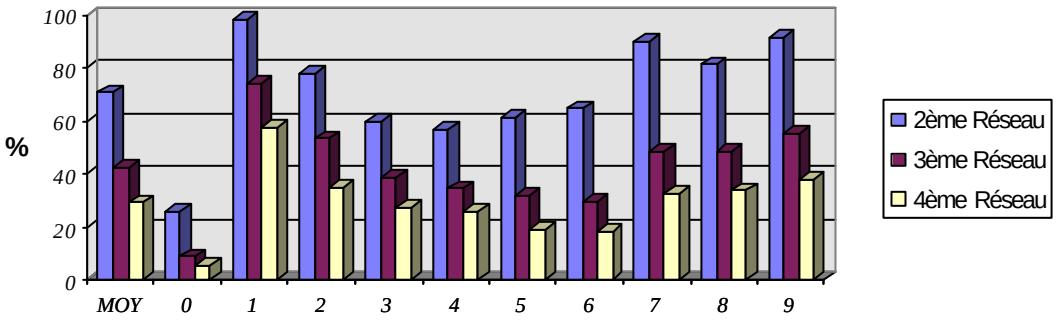


Figure 7.19 - Répartition des fréquences d'accès aux divers réseaux de neurones, en fonction de la classe des caractères d'entrée. Les chiffres « 1 », « 2 », « 7 », « 8 », et « 9 » sont les plus délicats à reconnaître, le chiffre « 0 » est le plus aisé.

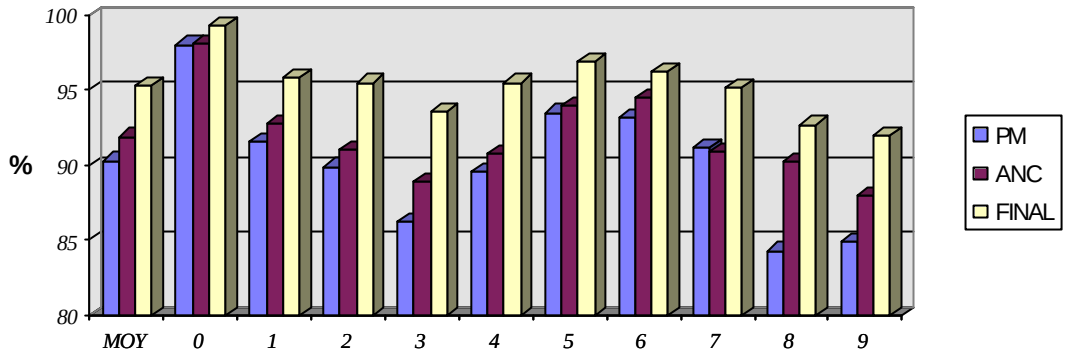


Figure 7.20 - Répartition des taux de reconnaissance par classe, pour les deux meilleurs réseaux de neurones isolés, et pour le système final de reconnaissance.
Ce dernier permet d'augmenter la qualité de reconnaissance pour toutes les classes.

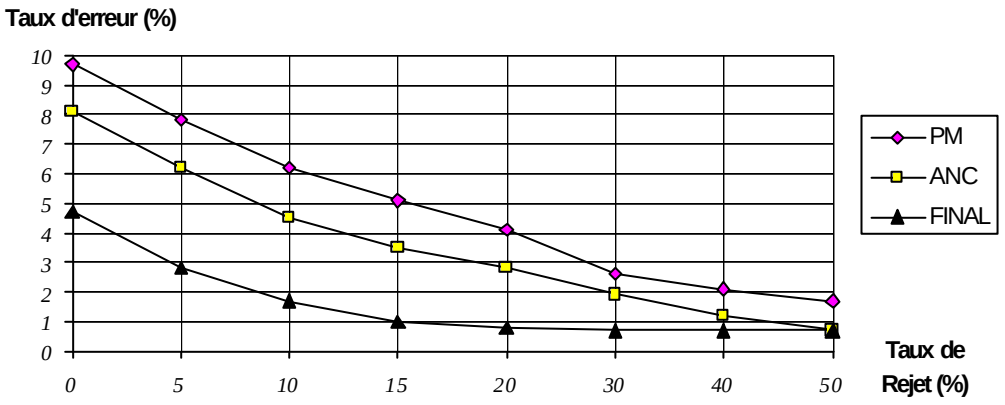


Figure 7.21 - Evolution du taux de reconnaissance en fonction du taux de rejet, pour les deux meilleurs réseaux de neurones isolés, et pour le système final de reconnaissance.

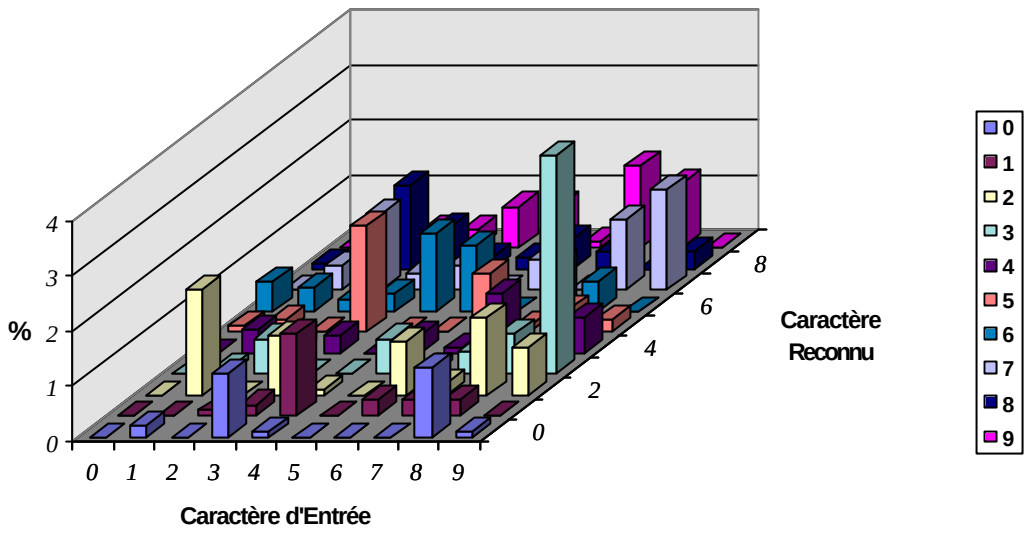


Figure 7.22 - Matrice de confusion du système final de reconnaissance de chiffres « Européens ». Afin de conserver la clarté des schémas, les diagonales n'ont pas été représentées.

7.6 Résumé

Plusieurs méthodes de mise en collaboration de perceptrons multicouches entraînés au moyen de catégories distinctes de primitives ont été présentées, et se sont révélées capables de réduire le taux global d'erreur de manière significative. La plupart des réseaux de neurones qui sont combinés, résultent en outre des tentatives d'apprentissages qui ont dû être effectuées précédemment, lors de l'optimisation du processus de reconnaissance utilisant un seul réseau de neurones. Toutes les méthodes de collaboration proposées sont donc ainsi extrêmement aisées à mettre en oeuvre.

Les diverses méthodes de coopération conçues, résultent d'une étude menée de manière à réduire le plus possible la durée moyenne nécessaire à la reconnaissance. La coopération en parallèle de perceptrons multicouches a montré que la taille de chacun des réseaux de neurones pouvait être réduite sans que le taux global de reconnaissance ne soit affecté. La coopération en cascade des perceptrons multicouches a ensuite rendu possible une réduction plus significative du temps moyen de reconnaissance, en limitant la fréquence d'utilisation du second réseau. Enfin, la

substitution en cascade de deux systèmes de reconnaissance a permis d'obtenir un processus de reconnaissance encore plus rapide.

Le système final de reconnaissance consiste en une combinaison de quatre perceptrons multicouches, dans laquelle les deux plus petits sont utilisés durant près de **60%** du temps en moyenne. Cela a conduit à un processus de reconnaissance qui est **1,5** fois plus rapide que celui obtenu lorsque seul le plus performant des perceptrons multicouches individuels est utilisé, pour un taux d'erreurs de classification qui est **40%** moins important.

Chapitre 8

Conclusion Générale

Un système de reconnaissance hors-ligne de caractères manuscrits a été présenté. Il est constitué d'une association de plusieurs modules.

- Le premier d'entre eux analyse l'image fournie par un périphérique de digitalisation et en extrait la suite de caractères à reconnaître. Il dispose pour cela de deux algorithmes de segmentation. Le premier est très simple et rapide à l'exécution, et suffit à résoudre une grande majorité des situations qui peuvent se présenter. Le second algorithme, plus complexe et plus coûteux en temps de calcul, est destiné à la séparation de caractères qui se chevauchent. Son emploi est contrôlé par le module de classification, ce qui permet d'en limiter la fréquence et de réduire ainsi la durée moyenne de cette étape.

- Le deuxième élément de la structure code les caractères en vue d'en extraire une information discriminante, tout en réduisant leur dimension de représentation. Cette réduction est nécessaire pour rendre les images de caractères exploitables par le module de classification, alors que l'extraction de caractéristiques discriminantes permet de présenter les caractères à ce dernier de manière plus pertinente. La méthode de *l'Analyse Normalisée du Contour* s'est révélée être la plus efficace, et est celle qui est prioritairement employée. Utilisée en collaboration avec celle-ci, la méthode des *Pixels Moyennés* permet d'augmenter de manière significative la qualité de la reconnaissance effectuée. Dans le but d'optimiser l'ensemble du processus de reconnaissance, son emploi est conditionné par le résultat de la classification obtenue à l'aide de la première méthode.

- Le troisième module de la chaîne effectue la classification proprement dite. Il réalise à cet effet un ensemble de fonctions discriminantes, qui associent à chaque classe un score représentatif de sa probabilité d'être celle du caractère présenté. Deux catégories de perceptrons multicouches sont disponibles pour obtenir ces valeurs, chacune d'entre elles étant associée à un des vecteurs de caractéristiques que peut fournir le module qui précède. Lorsque le plus haut score obtenu à l'aide du premier réseau de neurones est jugé insuffisant, le deuxième d'entre eux est sollicité pour achever la reconnaissance du caractère critique. Une autre condition sur les valeurs de sortie intervient également, et permet d'avoir recours à la seconde procédure de segmentation, le cas échéant. Suivant les applications, plusieurs perceptrons multicouches peuvent collaborer en cascade ou par substitution. Ceci permet de minimiser le temps moyen de traitement d'un caractère, tout en conservant un taux de reconnaissance élevé.

La majeure partie de la thèse a porté sur le développement de méthodes performantes d'extraction de primitives d'une part, et sur la mise en oeuvre de perceptrons multicouches pour la classification d'autre part. L'association de ces techniques a permis d'obtenir des taux de reconnaissance de 96,7% et 99,8%, respectivement pour des lettres et des chiffres manuscrits extraits de la base de données NIST3. Ces performances s'avèrent supérieures à celles publiées par ailleurs dans la littérature. Appliquées à des chiffres manuscrits de type « Européen », dont les formes présentent une dispersion d'aspect nettement plus élevée, ces méthodes ont permis d'atteindre un taux de reconnaissance de 95,3%. Par rapport au meilleur des perceptrons multicouches considéré isolément, cela représente une diminution de 40% du nombre d'erreurs de classification, pour une vitesse de traitement qui est 1,5 fois plus élevée.

L'étude des problèmes de la reconnaissance automatique de caractères abordés dans cette thèse a porté principalement sur cinq points:

1. En premier lieu, un nouvel algorithme de segmentation de caractères a été conçu. Il permet de procéder à une séparation efficace de caractères qui se chevauchent. Cet algorithme est nécessaire, et suffisant, pour assurer le maintien des performances de reconnaissance face à cette situation, qui se présente souvent en pratique en l'absence de contraintes d'écriture.

2. Deuxièmement, plusieurs méthodes d'apprentissage du perceptron multicouches ont été analysées. Bien que ce modèle de réseau de neurones soit un classificateur potentiellement très performant, son entraînement est souvent délicat. Une procédure originale a été développée, et s'est avérée être celle permettant d'obtenir la vitesse de convergence la plus élevée lorsque

le mode d'apprentissage hors-ligne est appliqué. Sa phase de méta-optimisation très simple la rend en outre aisée à mettre en oeuvre, y compris pour des problèmes de grandes dimensions. Ce sont toutefois des méthodes utilisant les modes d'apprentissage en-ligne ou en demi-ligne, dont la notion a également été introduite ici, qui se sont révélées être globalement les plus performantes.

3. Plusieurs méthodes d'extraction de primitives ont ensuite été envisagées. Les plus efficaces d'entre elles ont encore été améliorées par l'utilisation de techniques complémentaires. L'ensemble de celles-ci constitue des procédures de traitement d'images aisées à implanter matériellement. Une procédure d'analyse discriminante, très simple à mettre en oeuvre, a en outre été appliquée. La qualité des résultats obtenus a confirmé sa pertinence, et en a montré l'intérêt pour un classificateur de type perceptron multicouches.

4. Un procédé original permettant de générer de manière artificielle des images de caractères manuscrits a été introduit. Celui-ci permet, en augmentant la taille de la base de données destinée à l'entraînement du perceptron multicouches, d'en améliorer les performances de classification, sans en augmenter le temps de calcul en phase d'exploitation. Basé sur deux algorithmes de distorsions, linéaires et géométriques, il permet de créer un grand nombre de caractères à partir de l'image originale d'un seul. Cette génération, à partir d'un caractère manuscrit réel, permet d'éviter tout aspect «mécanique » des formes créées. Le nombre restreint de paramètres, qui régissent les distorsions produites, rend en outre ces dernières aisément contrôlables, ce qui réduit considérablement le risque de création de caractères trop ou insuffisamment déformés.

5. Enfin, diverses structures de mise en coopération de perceptrons multicouches ont été proposées. La plupart d'entre elles prennent en considération des réseaux de neurones précédemment entraînés, et donc déjà à disposition. La collaboration de perceptrons multicouches, entraînés sur base de catégories distinctes de primitives, a permis d'améliorer la qualité de la reconnaissance effectuée. Le taux d'erreur de classification a ainsi été réduit de plus de 40% par rapport à celui offert par le meilleur des réseaux de neurones isolés, dans le cas de chiffres manuscrits présentant une dispersion de styles d'écriture aussi élevée que celle existant en nos contrées. Des perceptrons multicouches de très faibles dimensions ont ensuite été entraînés, et sont destinés à effectuer une reconnaissance rapide des caractères les plus faciles. La mise en coopération en cascade, ainsi que l'utilisation par substitution, de ces réseaux de neurones, ont alors permis d'obtenir un système de reconnaissance de chiffres qui est près de 50% plus rapide que le plus efficace des réseaux de neurones isolés, tout en maintenant la qualité de reconnaissance à son niveau le plus élevé. Une nouvelle phase d'entraînement des perceptrons multicouches de la chaîne de reconnaissance, spécifiquement

sur base des caractères rejetés par le classificateur constitué des réseaux de neurones qui précèdent chacun d'entre eux, devrait permettre d'améliorer encore la qualité et/ou la vitesse moyenne de reconnaissance de l'ensemble du système. Les excellents résultats que ce dernier a permis d'atteindre sur les caractères de la base de données NIST3, lui donnent en outre accès à une série de tests objectifs, actuellement organisés, au niveau Européen, par un organisme indépendant. Une application pratique en est également en cours de développement, grâce au concours d'un partenaire industriel, et vise le marché du traitement automatique de virement bancaires.

Annexe A

L'Algorithme de Rétro-Propagation

L'apprentissage du perceptron multicouches consiste à adapter les poids synaptiques des neurones, de manière à ce que le réseau soit capable de réaliser une transformation donnée, représentée par un ensemble d'exemples constitué d'une suite de N vecteurs d'entrées $X_k = \left[x_{k1}\ x_{k2} \cdots x_{kd} \right]^T$ à laquelle est associée une suite de vecteurs de sorties désirées $T^{(k)} = \left[t_1^{(k)}\ t_2^{(k)} \cdots t_{h_L}^{(k)} \right]^T$.

Lorsque le critère des Moindres Carrés de l'Erreur est utilisé pour définir la fonction de coût à minimiser, celle-ci s'exprime:

$$E = \frac{1}{2} \sum_{k=1}^{N} \sum_{i=1}^{h_L} \left(y_{L,i}^{(k)} - t_i^{(k)} \right)^2 \quad \text{(A.1)}$$

où:

- N est le nombre d'exemples d'apprentissage;
- L est le nombre de couches du réseau;
- h_l vaut le nombre de neurones que contient la couche l;
- $y_{l,i}^{(k)}$ désigne la sortie du neurone i de la couche l lorsque le vecteur X_k est présenté à l'entrée du réseau;

- $t_i^{(k)}$ représente la valeur de sortie désirée pour le neurone i de la dernière couche lorsque le vecteur X_k est présenté à l'entrée du réseau;

La minimisation de cette fonction de coût non linéaire se fait de manière itérative, en utilisant une méthode de gradient [Luenberger,89].

A chaque itération, un vecteur d'entrée $X_k = \left[x_{k1}\ x_{k2}\ \cdots\ x_{kd} \right]^T$ ainsi que le vecteur associé de sorties désirées $T^{(k)} = \left[t_1^{(k)}\ t_2^{(k)}\ \cdots\ t_{h_L}^{(k)} \right]^T$ sont présentés au système. L'erreur localement effectuée est alors calculée selon:

$$E^{(k)} = \frac{1}{2} \sum_{i=1}^{h_L} \left(y_{L,i}^{(k)} - t_i^{(k)} \right)^2 \quad \text{(A.2)}$$

Les poids synaptiques qui relient les neurones entre eux peuvent ensuite être adaptés en fonction de la relation:

$$w_{l,ij}(\tau+1) = w_{l,ij}(\tau) - \alpha \frac{\partial E^{(k)}}{\partial w_{l,ij}} \quad \text{(A.3)}$$

où α est le taux d'apprentissage, et où $\dfrac{\partial E^{(k)}}{\partial w_{l,ij}}$ est appelé *gradient instantané* de l'erreur.

Soient:

- $\varphi_{l,i}(\cdot)$: la fonction d'activation du neurone i de la couche l;
- $\underline{W}_{l,i} = \left[w_{l,i0}\ w_{l,i1}\ w_{l,i2}\ \cdots\ w_{l,ih_{l-1}} \right]^T$: le vecteur de poids augmenté de ce même neurone;
- h_l : le nombre de neurones que contient la couche l;
- $X_k = \left[-1\ x_{k1}\ x_{k2}\ \cdots\ x_{kd} \right]^T$: le vecteur de caractéristiques augmenté présenté à l'entrée du réseau;
- $\underline{Y}_l^{(k)} = \left[-1\ y_{l,1}^{(k)}\ y_{l,2}^{(k)}\ \cdots\ y_{l,h_l}^{(k)} \right]^T$: le vecteur augmenté des sorties de la couche l lorsque le vecteur X_k est présenté à l'entrée du réseau.

[Luenberger,89] **D.G. Luenberger**
Linear and Nonlinear Programming
Addison-Wesley Publishing Company, 1989

Ces dernières valeurs sont déterminées à partir des sorties des neurones de la couche précédente par $(i > 0)$:

$$y_{l,i}^{(k)} = \varphi\left(\underline{W}_{l,i}^{T}\underline{Y}_{l-1}^{(k)}\right) \text{ (A.4)}$$

ce qui s'écrit simplement:

$$y_{l,i}^{(k)} = \varphi\left(u_{l,i}^{(k)}\right) \text{ (A.5)}$$

à condition que l'on ait posé:

$$u_{l,i}^{(k)} = \underline{W}_{l,i}^{T}\underline{Y}_{l-1}^{(k)} \text{ (A.6)}$$

Le processus de calcul du perceptron multicouches est initialisé en posant $\underline{Y}_0^{(k)} = \underline{X}_k$.

Le gradient instantané de l'erreur peut s'exprimer:

$$\frac{\partial E^{(k)}}{\partial w_{l,ij}} = \frac{\partial E^{(k)}}{\partial u_{l,i}^{(k)}}\frac{\partial u_{l,i}^{(k)}}{\partial w_{l,ij}} \text{ (A.7)}$$

En posant:

$$\frac{\partial E^{(k)}}{\partial u_{l,i}^{(k)}} = \delta_{l,i}^{(k)} \text{ (A.8)}$$

où $\delta_{l,i}^{(k)}$ est appelé *gradient local* de l'erreur, et en tenant compte que de (A.6) on a:

$$\frac{\partial u_{l,i}^{(k)}}{\partial w_{l,ij}} = y_{l-1,j}^{(k)} \text{ (A.9)}$$

Il vient ainsi:

$$\frac{\partial E^{(k)}}{\partial w_{l,ij}} = \delta_{l,i}^{(k)} y_{l-1,j}^{(k)} \text{ (A.10)}$$

Le gradient local (A.8) peut se développer selon:

$$\frac{\partial E^{(k)}}{\partial u_{l,i}^{(k)}} = \sum_{q=1}^{h_{l+1}} \frac{\partial E^{(k)}}{\partial u_{l+1,q}^{(k)}}\frac{\partial u_{l+1,q}^{(k)}}{\partial u_{l,i}^{(k)}} \text{ (A.11)}$$

Ou encore:

$$\frac{\partial E^{(k)}}{\partial u_{l,i}^{(k)}} = \sum_{q=1}^{h_{l+1}} \frac{\partial E^{(k)}}{\partial u_{l+1,q}^{(k)}} \frac{\partial u_{l+1,q}^{(k)}}{\partial y_{l,i}^{(k)}} \frac{\partial y_{l,i}^{(k)}}{\partial u_{l,i}^{(k)}} \quad \text{(A.12)}$$

Comme on a:

- de (A.8): $\frac{\partial E^{(k)}}{\partial u_{l+1,q}^{(k)}} = \delta_{l+1,q}^{(k)}$,
- de (A.6): $\frac{\partial u_{l+1,q}^{(k)}}{\partial y_{l,i}^{(k)}} = w_{l+1,qi}$,
- de (A.5): $\frac{\partial y_{l,i}^{(k)}}{\partial u_{l,i}^{(k)}} = \dot{\varphi}_{l,i}\left(u_{l,i}^{(k)}\right)$,

il vient:

$$\delta_{l,i}^{(k)} = \dot{\varphi}_{l,i}\left(u_{l,i}^{(k)}\right) \sum_{q=1}^{h_{l+1}} w_{l+1,qi} \delta_{l+1,q}^{(k)} \quad \text{(A.13)}$$

C'est l'expression (A.13) qui a donné son nom à l'algorithme d'apprentissage du perceptron multicouches: *rétro-propagation* du gradient de l'erreur. En effet, le gradient local $\delta_{l,i}^{(k)}$ d'un neurone est calculé à partir des gradients locaux $\delta_{l+1,q}^{(k)}$ des neurones de la couche ultérieure. Le calcul des gradients commence donc par la dernière couche, et est ensuite propagé de celle-ci vers la première couche du réseau.

Le gradient local doit être calculé en premier lieu pour les neurones de la couche de sortie. De l'expression (A.8), il vient:

$$\delta_{L,i}^{(k)} = \frac{\partial E^{(k)}}{\partial u_{L,i}^{(k)}} = \frac{\partial E^{(k)}}{\partial y_{L,i}^{(k)}} \frac{\partial y_{L,i}^{(k)}}{\partial u_{L,i}^{(k)}} \quad \text{(A.14)}$$

Dans le cas où le critère de minimisation est celui des moindres carrés de l'erreur, cette dernière est calculée par (A.2), et on a donc:

$$\frac{\partial E^{(k)}}{\partial y_{L,i}^{(k)}} = y_{L,i}^{(k)} - t_i^{(k)} \quad \text{(A.15)}$$

Il vient finalement:

$$\delta_{L,i}^{(k)} = \left(y_{L,i}^{(k)} - t_i^{(k)}\right)\dot{\varphi}_{L,i}\left(u_{L,i}^{(k)}\right) \quad \text{(A.16)}$$

Les formules (A.10), (A.13) et (A.16) permettent de calculer facilement la valeur de la modification qui doit être apportée à chaque poids du réseau.

Lorsque la fonction de coût régissant l'apprentissage n'est pas le critère des Moindres Carrés de l'Erreur, seule l'expression (A.15) doit être recalculée, et l'expression (A.16) adaptée en conséquence. A titre d'exemple, la fonction de coût définie au moyen du critère de l'Entropie Relative s'exprime:

$$E^{(k)} = \sum_{i=1}^{h_L}\left[t_i^{(k)} \ln\left(\frac{t_i^{(k)}}{y_{L,i}^{(k)}}\right) + \left(1 - t_i^{(k)}\right)\ln\left(\frac{1-t_i^{(k)}}{1-y_{L,i}^{(k)}}\right)\right] \quad \text{(A.17)}$$

L'expression (A.15) devient alors dans ce cas:

$$\frac{\partial E^{(k)}}{\partial y_{L,i}^{(k)}} = \frac{-t_i^{(k)}}{y_{L,i}^{(k)}} + \frac{1-t_i^{(k)}}{1-y_{L,i}^{(k)}}, \quad \text{(A.18)}$$

et le gradient local pour un neurone de la couche de sortie est ainsi donné par:

$$\delta_{L,i}^{(k)} = \frac{\left(y_{L,i}^{(k)} - t_i^{(k)}\right)}{y_{L,i}^{(k)}\left(1-y_{L,i}^{(k)}\right)}\dot{\varphi}_{L,i}\left(u_{L,i}^{(k)}\right) \quad \text{(A.19)}$$

Lorsque la fonction d'activation des neurones est la fonction sigmoïde, on a:

$$\varphi_{l,i}\left(u_{l,i}^{(k)}\right) = \frac{1}{1+\exp\left(-u_{l,i}^{(k)}\right)} \quad \text{(A.20)}$$

Il vient donc:

$$\begin{aligned}\dot{\varphi}_{l,i}\left(u_{l,i}^{(k)}\right) &= \frac{\exp\left(-u_{l,i}^{(k)}\right)}{\left(1+\exp\left(-u_{l,i}^{(k)}\right)\right)^2} \\ &= \varphi_{l,i}\left(u_{l,i}^{(k)}\right)\left(1-\varphi_{l,i}\left(u_{l,i}^{(k)}\right)\right) \\ &= y_{l,i}^{(k)}\left(1-y_{l,i}^{(k)}\right) \qquad \text{(A.21)}\end{aligned}$$

Lorsque le critère des Moindres Carrés de l'Erreur est utilisé, on obtient ainsi:

- pour un neurone de la couche de sortie:

$$\delta_{L,i}^{(k)} = y_{L,i}^{(k)}\left(1-y_{L,i}^{(k)}\right)\left(y_{L,i}^{(k)}-t_i^{(k)}\right) \quad \text{(A.22)}$$

- pour un neurone d'une couche cachée:

$$\delta_{l,i}^{(k)} = y_{l,i}^{(k)}\left(1-y_{l,i}^{(k)}\right)\sum_{q=1}^{h_{l+1}} w_{l+1,qi}\delta_{l+1,q}^{(k)} \quad \text{(A.23)}$$

Lorsque la fonction de coût est définie sur base du critère de l'Entropie Relative, l'expression du gradient local pour un neurone de la couche de sortie s'exprime simplement:

$$\delta_{L,i}^{(k)} = y_{L,i}^{(k)} - t_i^{(k)} \quad \text{(A.24)}$$

L'expression du gradient local pour un neurone d'une couche cachée demeure, elle, inchangée.

Annexe B

La Fonction Logique de Squelettisation

La recherche du squelette d'un caractère s'effectue en éliminant des pixels qui appartiennent au corps même du caractère de manière itérative, tant que cela ne rompt pas la continuité du tracé. Cette procédure requiert un balayage de l'image en deux passes alternatives, de gauche à droite et de haut en bas d'abord, et de haut en bas et de gauche à droite ensuite (*figure B.1*), à chacune desquelles correspond une fonction logique des valeurs des neuf pixels situés dans une fenêtre 3×3 centrée sur le pixel susceptible d'être éliminé (*figure B.2*), ayant pour rôle d'assurer le maintien de la continuité du tracé.

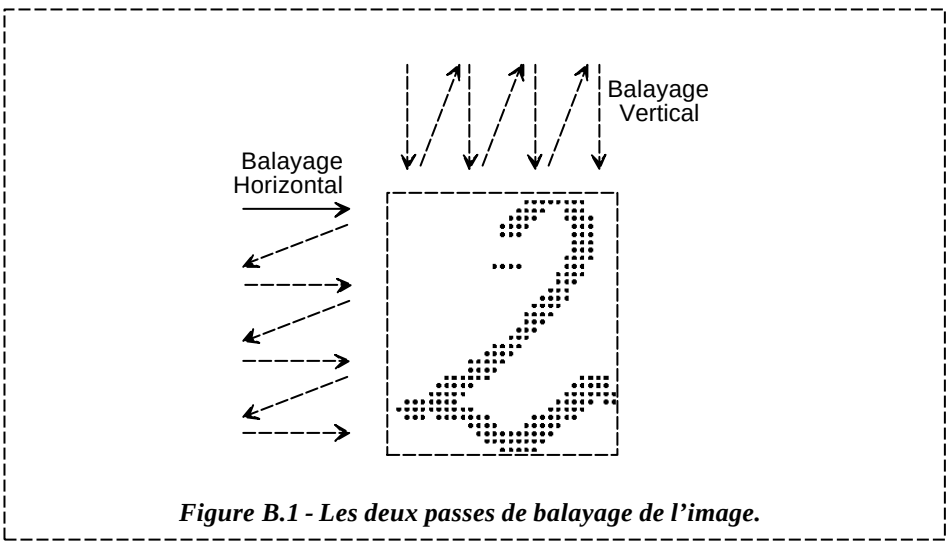


Figure B.1 - Les deux passes de balayage de l'image.

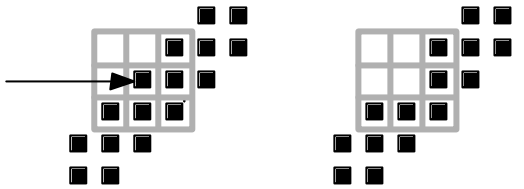


Figure B.2 - Fenêtre de prélèvement des variables de la fonction logique de squelettisation.

La *figure B.3* illustre l'ordre de prélèvement des valeurs des pixels dans la fenêtre d'analyse. Les fonctions logiques ont été établies en examinant cas par cas les diverses situations qui peuvent se présenter, et en tenant compte des éventuelles symétries. Elles s'expriment,

pour la passe de balayage horizontal:

$$F_h = p_O\left[\tilde{p}_8' \cdot p_4 \cdot (p_1' + p_2) \cdot (p_6 + p_7') + p_8 \cdot p_4' \cdot (p_2 + p_3') \cdot (p_5' + p_6)\right] \cdot (p_1 + p_2 + p_3 + p_5 + p_6 + p_7) \quad \text{(B.1)}$$

pour la passe de balayage vertical:

$$F_v = p_O\left[\tilde{p}_2' \cdot p_6 \cdot (p_3' + p_4) \cdot (p_8 + p_1') + p_2 \cdot p_6' \cdot (p_4 + p_5') \cdot (p_7' + p_8)\right] \cdot (p_3 + p_4 + p_5 + p_7 + p_8 + p_1) \quad \text{(B.2)}$$

Le symbole « ' » appliqué à une variable signifie qu'il faut complémenter la valeur du pixel correspondant. Le symbole «~ » signifie, lui, que la valeur à prendre en considération est celle ***avant*** élimination éventuelle, lors de la passe en cours, du pixel concerné.

P_1	P_2	P_3
P_8	P_0	P_4
P_7	P_6	P_5

Figure B.3 - Ordre de prélèvement des pixels dans la fenêtre d'analyse.

La fonction logique de balayage vertical est simplement obtenue à partir de celle du balayage horizontal, en lui faisant subir une rotation de +90°, suivie d'une symétrie d'axe vertical. En pratique, ces fonctions logiques ne sont évaluées que lorsque le pixel central P_0 n'est pas d'intensité nulle, et, pour des raisons de vitesse d'exécution, les 2^8=256 valeurs que chacune d'elles peuvent prendre sont calculées une fois pour toutes, et tabulées.

Annexe C

Vectorisation de Caractères: La Validation des Segments

Le principe de la vectorisation des caractères, c'est-à-dire de leur représentation par un ensemble de segments de droites structurés au lieu d'un ensemble de points, a été décrit à la section 3.3.2 du chapitre 5. Nous ne décrivons ici que le calcul du critère de validation utilisé lors de la construction des segments.

Ce critère est une Erreur Moyenne de Représentation, définie comme étant la valeur moyenne des carrés des distances entre un segment et tous les pixels qu'il remplace. Un segment en cours de construction est défini par les coordonnées (x_0, y_0) du premier pixel qui lui a été associé, ainsi que par les coordonnées (x_n, y_n) du dernier pixel susceptible de lui être incorporé (*figure C.1*).

La droite qui passe par ces deux points a pour équation:

$$S \equiv ax - by + c = 0 \quad \text{(C.1)}$$

où:

$$\begin{cases} a = y_n - y_0 \\ b = x_n - x_0 \\ c = x_n y_0 - y_n x_0 \end{cases} \quad \text{(C.2)}$$

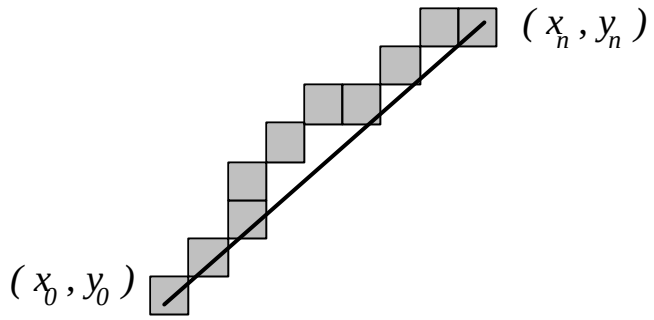


Figure C.1 - Segment en cours de construction et ensemble des pixels qu'il remplace.

Le carré de la distance entre un point p_i de coordonnées (x_i, y_i) et le segment *S* se calcule selon:

$$d^2(S, p_i) = \frac{(ax_i - by_i + c)^2}{a^2 + b^2} \qquad \text{(C.3)}$$

Le critère de validation du segment s'exprime alors:

$$\rho = \frac{1}{n}\sum_{i \in S} d^2(S, p_i) < \rho_{\max} \quad \text{(C.4)}$$

où r est *l'Erreur Moyenne de Représentation*, calculée sur les n points que remplace le segment, et où $r_{\max}$ est un paramètre qui permet de contrôler la qualité de représentation obtenue.

Les essais que nous avons effectués nous ont conduit à utiliser une valeur de $\rho_{\max} = 0{,}25$. Au delà de celle-ci, la distorsion introduite devient trop importante pour maintenir une qualité suffisante de représentation des caractères, et en deçà, les segments sont prématurément clôturés, ce qui augmente alors considérablement le nombre d'entre eux requis pour coder un caractère entier, sans pour autant en améliorer la qualité de représentation de manière significative.

La formule (C.3) peut être réécrite sous la forme:

$$d^2(S, p_i) = \frac{a^2 x_i^2 + b^2 y_i^2 - 2abx_i y_i + 2acx_i - 2bcy_i + c^2}{a^2 + b^2}, \text{ (C.5)}$$

et dès lors (C.4) se calculer selon:

$$\rho = \frac{a^2 \sum_{i=1}^{n-1} x_i^2 + b^2 \sum_{i=1}^{n-1} y_i^2 - 2ab \sum_{i=1}^{n-1} x_i y_i + 2ac \sum_{i=1}^{n-1} x_i - 2bc \sum_{i=1}^{n-1} y_i + nc^2}{n\left(a^2 + b^2\right)} \text{ (C.6)}$$

Le premier et le dernier point appartenant au segment même, ils peuvent être omis lors du calcul de l'Erreur Moyenne de Représentation.

La formule (C.6) permet une mise à jour aisée de l'Erreur Moyenne de Représentation, chaque fois qu'un pixel supplémentaire est inclus au segment. Il suffit en effet de conserver les valeurs de $\sum_{i=1}^{n-1} y_i^2$, $\sum_{i=1}^{n-1} x_i^2$, $\sum_{i=1}^{n-1} x_i y_i$, $\sum_{i=1}^{n-1} x_i$, et $\sum_{i=1}^{n-1} y_i$ en mémoire et de les actualiser lorsqu'un nouveau pixel est introduit dans le segment, et de recalculer ainsi seulement les nouvelles valeurs des paramètres a, b, c, et n. Cette manière de procéder permet d'obtenir rapidement l'Erreur Moyenne de Distorsion, sans avoir à recalculer systématiquement toutes les distances entre le segment redéfini et l'ensemble des pixels qu'il remplace.