Réseaux d'yprels, classification et apprentissage incrémental

Yprel Networks, Classification and Incremental Learning

par Y. LECOURTIER, A. ENNAJI, E. STOCKER, F. GILLES
Université de Rouen, Laboratoire Images et Informatique Industrielle
UFR des Sciences et Techniques
F-76821 Mont Saint Aignan cedex

Résumé

L'article présente un type de réseaux neuro–mimétiques appelé « réseaux d'yprels ». Après avoir rappelé quelques caractéristiques essentielles de ces réseaux, on précise la méthode d'apprentissage incrémental permettant d'améliorer les performances par reprise des erreurs de classification. Les résultats obtenus pour un problème de reconnaissance de caractères sont alors présentés.

Mots clés : Réseaux de neurones, Classification, Apprentissage supervisé, Apprentissage incrémental, Reconnaissance de caractères.

Abstract

This paper presents a neural network methodology called « yprel networks ». After relating the main characteristics of the approach, we shall detail the incremental learning methodology used to improve the performances, which is based on the re–learning phases from the classification errors. The results obtained on a characters recognition problem are then discussed.

Key words : *Neural network, Classification, Supervised learning, Incremental learning, Character recognition.*

1. Introduction

Le problème de classification est un problème central de la reconnaissance de formes. De nombreux ouvrages y ont été consacrés, et des méthodes devenues classiques (arbres de décision, approche bayésienne, K plus proches voisins, méthodes d'analyse de données...) ont donné lieu au développement de nombreuses applications [Belaïd], [Dubuisson], [Gaillat]. Au cours de ces dernières années, le développement des méthodes de type neuro–mimétique a relancé les études sur la classification : voir par exemple [Rumelhart], [Kohonen], [Hecht]. Toutefois, certaines difficultés n'ont toujours pas reçu de solutions satisfaisantes. L'une d'elles concerne l'apprentissage incrémental, c'est à dire la possibilité d'améliorer les performances d'un classifieur en reprenant une phase d'apprentissage lorsque des exemples supplémentaires apportant un complément d'information deviennent disponibles. Les méthodes classiques prennent généralement pour hypothèse que la phase de recueil de données constituant la base d'apprentissage a lieu avant toute construction du classifieur et que la base constituée est « statistiquement significative », c'est à dire qu'elle contient un nombre suffisant d'exemples pour approcher les caractéristiques statistiques de la population à classer. Toutefois ces caractéristiques étant généralement inconnues, on se contente de recommander de prendre une base d'apprentissage « la plus grande possible » et on confirme la validité des résultats en utilisant une base de test contenant des données aux caractéristiques statistiques analogues (pouvant être obtenues par extraction aléatoire de la base d'apprentissage initiale avant apprentissage). Le problème de l'apprentissage incrémental est donc rarement abordé. Or, il présente l'avantage d'éliminer a priori le problème de la représentativité de la base d'apprentissage sous reserve que la méthode employée soit susceptible de générer en cours d'utilisation une aide à l'utilisateur pour compléter la base d'apprentissage et ainsi améliorer les performances du classifieur. De plus, une approche fournissant une aide à la constitution de base d'apprentissage aura pour intérêt de manipuler des bases ne contenant que des exemples pertinents pouvant être en nombre réduit, ce qui aura généralement pour effet d'accélérer les phases d'apprentissage.

Un autre problème important dans la constitution d'un classifieur basé sur un modèle neuro–mimétique est le choix a priori de la structure du réseau. En effet une amélioration du comportement du classifieur peut être obtenue non seulement par une modification des paramètres (cœfficients « synaptiques ») du réseau,

mais également par une remise en cause de sa structure. Or les méthodes classiques ne fournissent généralement aucune aide à l'utilisateur pour choisir une structure adaptée. Ce n'est qu'au cours des dernières années que certains auteurs ont identifié ce problème comme étant l'une des limitations des méthodes neuro–mimétiques et ont commencé à proposer des solutions [Fahlman], [Frean], [Gentric], [Hüning], [Knerr], [Perez].

Les réseaux d'yprels sont l'une des réponses possibles aux deux problèmes soulevés précédemment dans le cadre de l'apprentissage supervisé. Les principes retenus pour leur construction ont déjà fait l'objet de plusieurs communications [Lecourtier1-3]. Ils sont à rapprocher d'une part de ceux développés par Ivakhnenko et présentés dans [Hecht] sous le terme de GMDH (« Group Method of Data Handling »), et d'autre part de ceux proposés dans [Knerr]. Toutefois, ils utilisent une méthodologie différente pour élaborer le classifieur. Après avoir rappelé ces principes, cet article présentera une simulation d'apprentissage incrémental. Cette problématique nous amenera à préciser en particulier les choix faits pour répondre à deux questions : (i) comment peut–on décider de la nécessité d'une reprise d'apprentissage? (ii) quelles informations doit–on conserver pour pouvoir reprendre un apprentissage sans que cela corresponde à une reconstruction complète du classifieur?

Afin d'illustrer les possibilités de la méthodologie, on discutera dans une dernière partie des résultats obtenus sur le problème classique de la reconnaissance de caractères multi–fontes, et on donnera les résultats de tests préliminaires sur le problème de la reconnaissance de chiffres manuscrits.

2. La classification par réseaux d'yprels

Un classifieur par réseaux d'Yprels (**pr**ocesseur **el**émentaire « ***Y*** ») est constitué d'un ensemble de réseaux : chaque réseau du classifieur doit identifier les éléments d'une classe et rejeter ceux des autres classes.

Chacun des réseaux agit donc comme un « spécialiste » résolvant un sous–problème à deux classes : la classe qu'il doit identifier, et l'ensemble de toutes les autres classes. Une telle approche présente plusieurs avantages : d'une part le sous–problème à traiter par chaque réseau étant plus simple que le problème général, l'apprentissage de chaque réseau en sera simplifié; d'autre part les réseaux travaillant indépendamment les uns des autres, la procédure de supervision générale chargée de collecter les réponses des divers réseaux pourra détecter des situations de conflits entre ces réponses et donc préparer les phases de reprises d'apprentissage; enfin dans une optique d'implantation sur machine parallèle, la détermination de la réponse des réseaux pourra être facilement répartie sur plusieurs processeurs.

Un exemple de réseau d'yprels pour un problème à deux primitives est dessiné sur la figure [1].

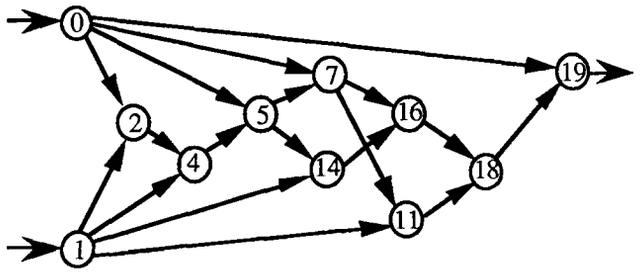


Figure 1. – Un exemple de réseau d'yprels.

On voit sur cette figure que ce réseau est constitué d'une part d'une couche d'yprels d'entrée (yprels 0 et 1), chacun de ces yprels étant associé à une primitive, et d'autre part d'un ensemble d'yprels combinant deux à deux les sorties d'yprels situés en amont dans le réseau.

L'élaboration de la décision d'un réseau est basée sur les principes suivants : (i) chaque yprel du réseau tente de prendre une décision de classification pour le prototype qui a été soumis au réseau, (ii) si un yprel amont a pris une décision, tous ses descendants dans le réseau ne feront que retransmettre cette décision. Le mécanisme de décision, qui sera détaillé en annexe, est donc partie intégrante du fonctionnement de l'yprel. Ceci diffère du fonctionnement de réseaux type Perceptron multicouches qui ne font que calculer une confiance dans l'appartenance à une classe, le mécanisme de décision étant un mécanisme externe, le plus simple est d'attribuer le prototype étudié à la classe de confiance maximale.

La structure d'un réseau est déterminée pas à pas durant l'apprentissage. L'algorithme d'apprentissage détaillé dans [Lecourtier3] est basé sur un schéma de coopération–compétition, illustré sur les figures [2] et [3], où chaque yprel est candidat à être la meilleure réponse au problème à deux classes que doit résoudre le réseau en construction. Une telle approche peut être rapprochée de celle utilisée dans les algorithmes génétiques [Golberg], mais sans nécessiter un codage explicite des gènes.

La liste L représentée sur la figure [3] est une liste des yprels ordonnés par performances. Un pas de l'algorithme d'apprentissage consiste alors à sélectionner dans cette liste deux yprels dont les

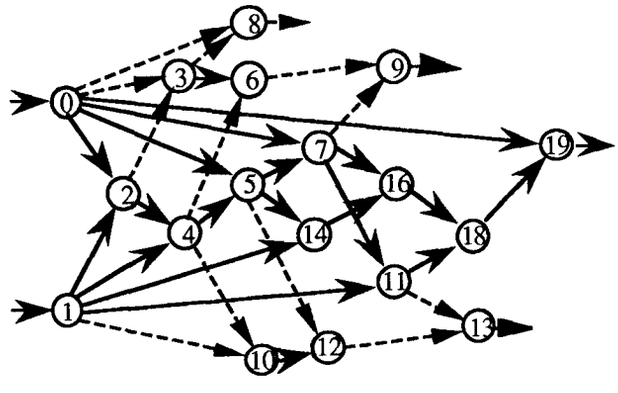


Figure 2. – Réseau en phase d'apprentissage.

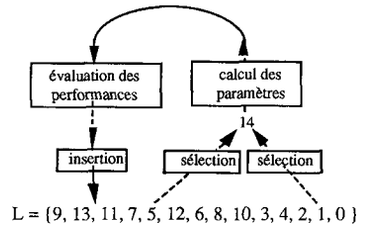


Figure 3. – Principe de l'apprentissage.

sorties vont devenir les entrées d'un nouvel yprel (coopération). Suite au calcul des paramètres de la fonction de transfert de cet yprel, ce dernier ne sera conservé que s'il a permis d'améliorer les performances vis à vis de celles de ses deux entrées.

Il faut noter que, avec l'algorithme d'apprentissage envisagé, les paramètres de chaque yprel sont calculés lorsque cet yprel est considéré comme un élément terminal possible pour le réseau, puis sont gelés et non modifiés ultérieurement. De ce fait, le calcul d'un nouveau réseau se limite au calcul des paramètres de son yprel terminal. Lorsque l'on décide l'arrêt d'un apprentissage, le premier élément de la liste L donne l'yprel terminal du réseau ayant les meilleures performances. On extrait alors l'ensemble des yprels constituant ce réseau (traits pleins sur la figure [2]), et on supprime tous ceux n'y participant pas (concurrence) (traits en pointillés sur la figure [2]).

La détermination des paramètres d'un yprel, basée sur la minimisation d'un critère de type moindres carrés tentant de séparer au mieux les éléments des deux classes du problème considéré non encore classifiés, est détaillé en annexe. En raisonnant dans un espace de caractéristiques donné, on peut fournir une interprétation géométrique du principe d'apprentissage d'un réseau. On peut montrer aisément [Lecourtier3] que tout yprel considéré comme yprel terminal d'un réseau délimite avec l'ensemble des yprels amont qui lui sont associés un domaine convexe qui contient tous les prototypes pour lesquels aucune décision de classification n'a été prise. Le but de l'apprentissage est de réduire le volume de ce domaine :

- soit par intersection des domaines convexes associés à deux yprels (qui seront les entrées du nouvel yprel),
- par création de plans séparateurs permettant d'isoler un sous–domaine convexe homogène (ne contenant soit que des prototypes de la classe à apprendre, soit que des prototypes des autres classes).

3. Apprentissage incrémental

Le principe de l'apprentissage incrémental est basé sur la remarque suivante : les performances d'un réseau dépendent fortement de la « qualité » de la base d'apprentissage qu'on lui fournit. Or, il est difficile de savoir *a priori* quels sont les exemples pertinents à mettre dans cette base puisque cela dépend à la fois du problème traité et de la méthode d'apprentissage utilisée. De ce fait, la réponse proposée ici à ce problème consiste à effectuer des reprises d'apprentissage sur erreurs.

On a montré [Lecourtier3] que les réseaux construits pouvaient détecter des erreurs de classification potentielles même sur une base non étiquetée du fait des redondances existant dans le processus de décision. On a donc un mécanisme permettant de sélectionner des prototypes qu'il faudra ensuite présenter à l'utilisateur pour qu'il les étiquette mais qui seront a priori des candidats utiles pour une reprise d'apprentissage. On pourrait donc travailler sur une suite d'exemples a priori en nombre quelconque au cours d'une utilisation, et améliorer le classifieur périodiquement pour lui éviter de commettre les erreurs précédemment identifiées. Bien évidemment, la mise à disposition d'une base étiquetée supplémentaire pourra également servir de support à une reprise d'apprentissage. Toutefois, pour les premiers tests de la méthodologie dont nous présentons les résultats au paragraphe 4, nous disposions d'un ensemble d'apprentissage fini, nous avons donc remplacé ce flot continu de données par un bouclage sur cet ensemble.

L'apprentissage incrémental pour le réseau de la classe k se fait donc selon l'algorithme suivant :

1. Choisir de manière aléatoire quelques exemples de chaque classe dans la base d'apprentissage (LS). Ce sous ensemble sera appelé LB_k : la base d'apprentissage du réseau associé à la classe k.
2. Utiliser l'ensemble d'apprentissage LB_k pour déterminer le meilleur réseau associé à la classe k en utilisant la méthodologie d'apprentissage présentée au paragraphe 2.
3. Tester la totalité de la base d'apprentissage LS et retenir toutes les erreurs commises par le réseau : c'est la base d'erreur associée au réseau de la classe k qu'on appellera EB_k.
4. Tant que les performances du réseau sont considérées insuffisantes,

4.1 – Choisir dans la base d'erreur EB_k un sous–ensemble d'éléments (appartenant aussi bien à la classe k qu'aux autres classes) et les rajouter dans la base d'apprentissage du réseau LB_k .

4.2 – Utiliser la base LB_k pour déterminer un nouveau réseau associé à la classe k par application de la méthodologie d'apprentissage et en réutilisant en priorité la structure du réseau obtenue à l'étape précédente.

4.3 – Tester la totalité de la base d'apprentissage LS, et construire une nouvelle base d'erreurs EB_k.

La figure [4] ci–dessous illustre le principe de l'apprentissage incrémental pour un réseau.

Nous pouvons faire quelques commentaires concernant l'algorithme ci–dessus :

– En fin d'apprentissage, chaque réseau construit est associé à une base d'apprentissage LB_k qui lui est propre même si la base d'apprentissage initiale LB_k est identique pour tous les réseaux du classifieur avant apprentissage. En effet, le but poursuivi par chaque réseau étant différent, les erreurs commises sont également différentes d'un réseau à l'autre. Les exemples rajoutés à LB_k

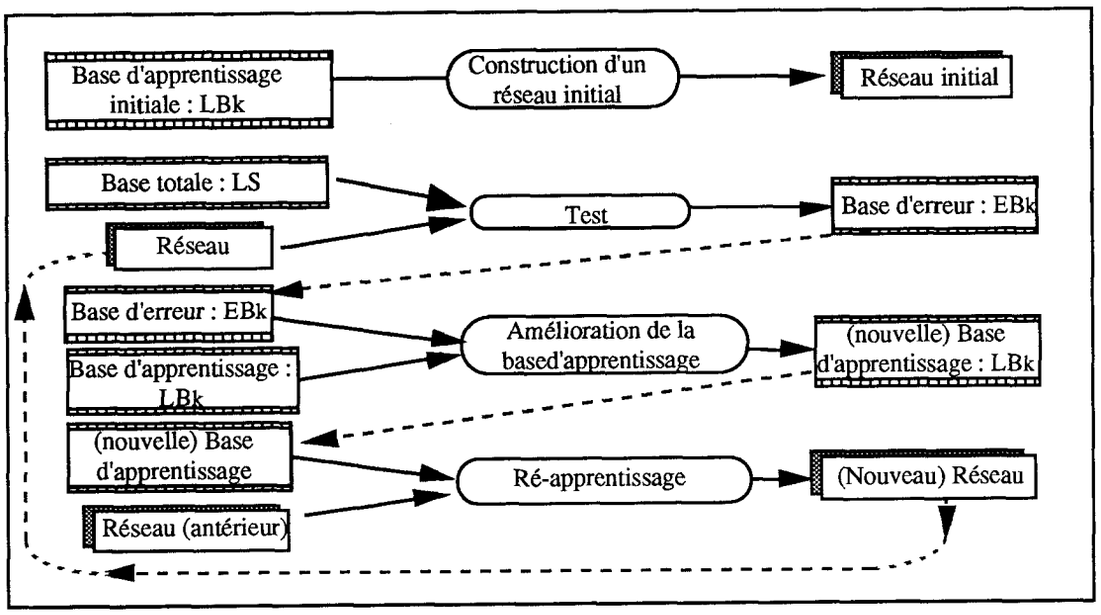


Figure 4. – Principe de l'apprentissage incrémental.

dépendent alors de la classe k et des performances obtenues par le réseau construit à l'étape précédente.

– Il est possible de changer complétement la base d'apprentissage utilisée, ou d'enrichir cette même base de nouveaux éléments. Les performances du réseau peuvent toujours être améliorées par de nouvelles phases d'apprentissage en enrichissant la base LB_k par des éléments choisis dans la nouvelle base d'apprentissage sans que cela se traduise par une remise en cause complète du réseau. De plus, puisque chaque réseau a ses propres données d'apprentissage, la remise en cause d'un réseau n'affecte pas le reste du classifieur.

– Les deux idées clé de la méthodologie sont : (i) obtenir pour chaque classe une base d'apprentissage aussi réduite que possible et, (ii) avoir dans cette base la meilleure représentation possible de la classe à reconnaître et des classes voisines de celle–ci. A cet effet, la base d'erreurs EB_k est reconstruite à chaque itération, et seul un sous ensemble de cette base est pris en compte. Le fait d'avoir des bases d'apprentissage LB_k de taille réduite permet des apprentissages plus rapides que s'ils étaient effectués sur la totalité de la base d'apprentissage. Mentionnons par ailleurs, à titre d'illustration, que pour le problème étudié au paragraphe 4.1, dès la première itération du processus d'apprentissage, la plupart des réseaux sont capables de rejeter plus de 98% des éléments des autres classes. De ce fait, on peut constater en fin de cycle d'apprentissage qu'ils ont généralement sélectionnés moins de 5% de l'ensemble des prototypes de la base d'apprentissage LS.

– Chaque réseau en apprentissage cherche à délimiter un domaine constitué d'un ensemble de sous–domaines convexes dans l'espace des caractéristiques pour la classe considérée [Lecourtier3]. Les meilleurs éléments pour l'apprentissage du réseau (et les plus difficiles à classer) sont donc ceux qui définissent les frontières de ce domaine. Ainsi, un processus d'apprentissage efficace doit permettre de choisir ces éléments. Pour cela, on associe à chaque élément de la base d'erreur une distance de Hamming calculée dans l'espace des primitives par rapport aux éléments de LB_k déjà appris. Deux types d'erreurs sont à considérer : les éléments appartenant à la classe k que le réseau a rejeté, et les éléments des autres classes que le réseau a accepté. Dans le premier cas, on sélectionne préférentiellement dans EB_k les éléments les plus éloignés des éléments déjà présents dans LB_k. A contrario, dans le deuxième cas, on sélectionne les éléments les plus proches des éléments déjà présents dans LB_k. On sélectionne au plus un nombre $N_{\max}$ d'éléments de la base d'erreurs ($N_{\max}$: paramètre de l'algorithme d'apprentissage) où chaque classe est représentée proportionnellement au nombre de représentants de cette même classe dans la base d'erreurs EB_k.

– A l'étape 4.2 de l'algorithme, la ré–initialisation du mécanisme de construction d'un réseau s'effectue en exploitant au mieux les performances et informations obtenues aux étapes antérieures. Dans ce but, les paramètres des yprels constituant l'ancien réseau sont recalculés sur la nouvelle base d'apprentissage. De ce fait, la structure de ce réseau est reconstruite en priorité, et c'est donc par rapport à cette structure qui s'était avérée la plus performante lors du ré–apprentissage précédent que se fait la recherche d'une meilleure solution.

– L'étape 4 de l'algorithme nécessite la définition d'un critère de performance. Le critère actuellement utilisé consiste à réaliser un apprentissage à 100% de la base d'apprentissage par le réseau. Un tel critère peut conduire dans certaines situations à un phénomène de sur–apprentissage. Ceci se traduit par une augmentation importante du nombre d'yprels du réseau sans amélioration significative des performances. Les travaux actuels

portent sur la définition d'un critère plus robuste, intégrant en particulier le paramètre évolution du nombre d'éléments bien classés de la base d'apprentissage par rapport au nombre d'yprels générés.

D'autres développements de la méthodologie sont en cours de test, en particulier, la mise en œuvre d'un processus de coopération entre réseaux [Stocker]. L'idée de base de cette coopération est de permettre au réseau de la classe k en construction d'exploiter le pouvoir discriminant des réseaux 0 à $k-1$ déjà construits. Ceci est réalisé en considérant les sorties des réseaux déjà construits comme primitives d'entrées supplémentaires pour le réseau en cours d'apprentissage.

4. Application en reconnaissance de caractères

4.1. APPLICATION LA CONNAISSANCE DE CARACTÈRES IMPRIMÉS MULTI-FONTES

Les résultats présentés dans ce paragraphe concernent un problème de reconnaissance de caractères imprimés multi-fontes. Les différents réseaux ont été construits à partir d'une base de 8800 exemples (majuscules, minuscules et chiffres) appartenant à onze fontes différentes de différents corps. De plus, la base comporte un ensemble de symboles (lettres grecques, triangles, carrés, ...) que tous les réseaux apprennent à rejeter comme des « non-caractères ». La figure [5] donne quelques échantillons des fontes utilisées.

A A A A D D D G G G F F R R R 6 6 a a a a a e e e e e

Figure 5. – Exemples de caractères des bases d'apprentissage et de test.

Pour chaque exemple, un ensemble de 50 primitives classiques en *OCR* (rapport hauteur/largeur du caractère, nombre d'intersections du tracé du caractère avec des sondes horizontales ou verticales, densité de pixels dans certaines zones, distance au rectangle englobant sur certaines sondes,...) ont été mesurées sur l'image du caractère. La taille des caractères n'ayant pas été retenue parmi les primitives, des lettres telles que v et V ou u et U ne pouvaient être distinguées, et ont été rassemblées dans la même classe. Au total, le problème considéré comporte 49 classes.

Les figures [6], [7], [8] et [9] montrent l'évolution au cours des phases de ré-apprentissage successives de quatre paramètres importants liés à la méthodologie proposée pour deux des réseaux : ceux chargés d'identifier les lettres « T » (courbe en pointillé) et « V » (courbe en plein). En abscisses des courbes de ces figures est indiqué le nombre de ré-apprentissages effectués.

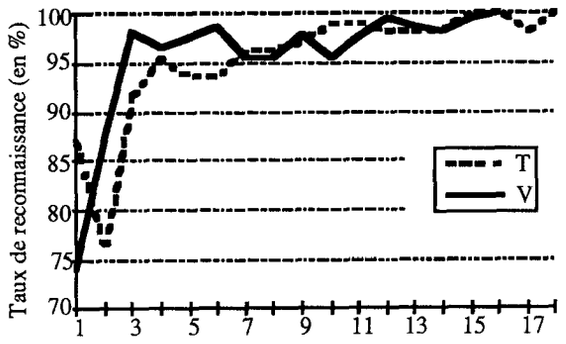


Figure 6. – Pourcentage des éléments bien reconnus.

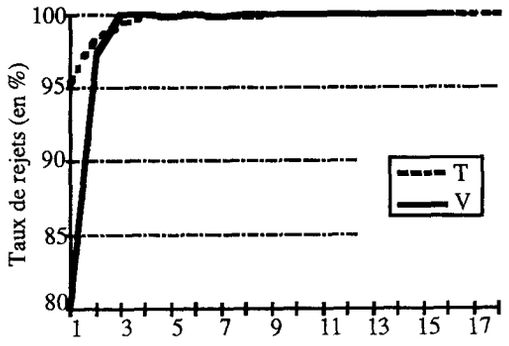


Figure 7. – Pourcentage des éléments bien rejetés.

Figure 8. – Taille des réseaux.

La courbe de la figure [6] montre que les réseaux apprennent peu à peu à reconnaître tous les éléments de la base proposée qui appartiennent à la classe qu'ils sont chargés d'identifier; la courbe de la figure [7] montre qu'ils apprennent à rejeter tous les éléments des autres classes. La courbe de la figure [8] montre l'évolution de la taille de ces réseaux au cours des ré-apprentissages successifs. On peut constater que ce nombre peut fluctuer de façon significative. Cela correspond à l'évolution possible de la structure : au cours d'un ré-apprentissage, le principe de concurrence peut conduire à explorer une nouvelle voie permettant une classification plus efficace en terme de nombre d'yprels nécessaires. En effet, le processus d'apprentissage permet « d'orienter » la recherche vers

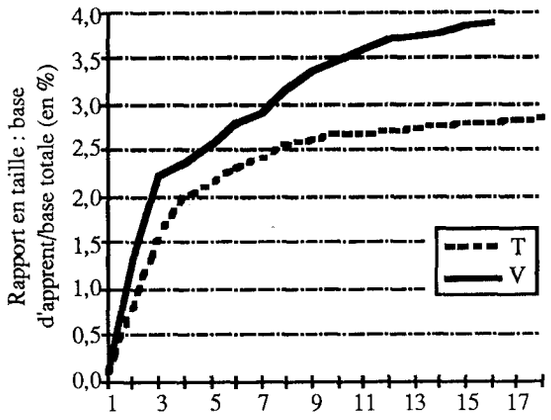


Figure 9. – Dimension base d'apprentissage/base totale.

des réseaux de taille inférieure à une taille donnée en augmentant la probabilité de construction de nouveaux réseaux à partir de réseaux de faible taille. La courbe de la figure [9] montre l'évolution en taille de la base d'apprentissage propre à chaque réseau au cours des ré–apprentissage. On voit que cette taille ne dépasse pas 4% de la base d'apprentissage totale pour les deux réseaux considérés.

Les 49 réseaux construits ont convergé vers un apprentissage à 100% tant en reconnaissance qu'en rejet sur la base d'apprentissage proposée. L'histogramme de la figure [10] précise le nombre de cycles de ré–apprentissage de chacun des réseaux pour atteindre ce but. On voit que plus des deux tiers des réseaux nécessitent moins de 20 cycles de ré–apprentissage.

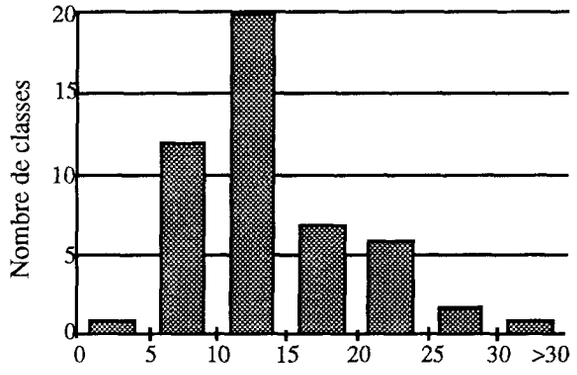


Figure 10. – Nombre de cycles de ré-apprentissage.

Les histogrammes des figures [11], [12] et [13] montrent les tendances observées pour les réseaux obtenus en fin de cycles de ré–apprentissages. L'histogramme de la figure [11] précise les tailles de ces réseaux (nombre d'yprels les composant). On voit que dans la plupart des cas, ce nombre reste relativement modeste, plus des deux tiers des classes étant identifiées par des réseaux comportant moins de 75 yprels.

Il est à remarquer par ailleurs que les réseaux n'utilisent que les primitives qui leur sont utiles pour atteindre leur but. Pour illustrer

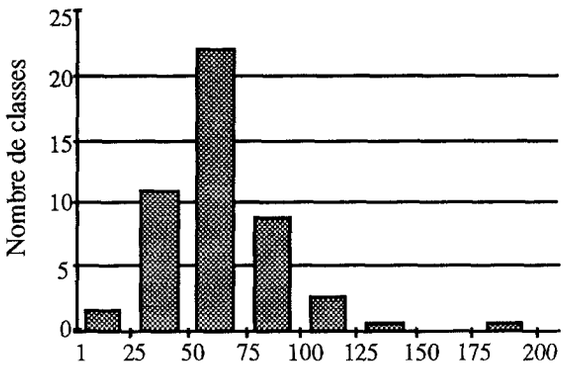


Figure 11. – Nombre d'yprels dans les réseaux.

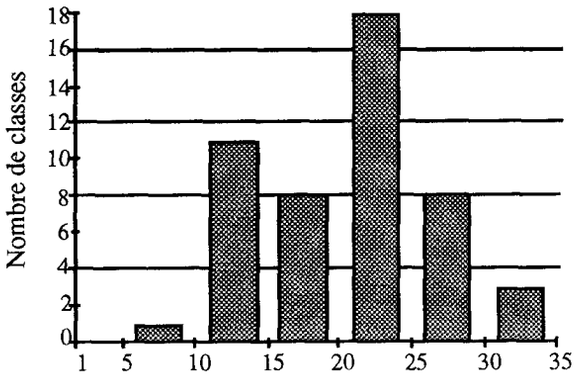


Figure 12. – Nombre de primitives utilisées.

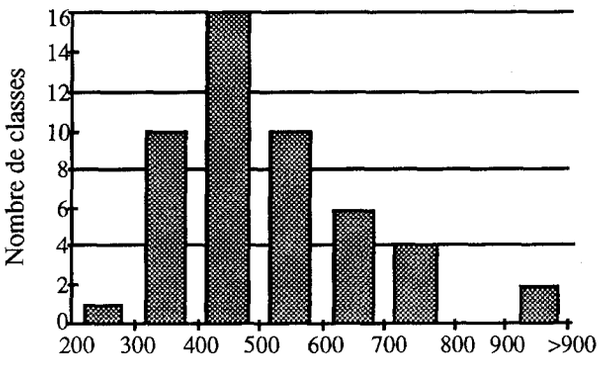


Figure 13. – Nombre de prototypes sélectionnés dans la base d'apprentissage.

ce fait, l'histogramme de la figure [12] montre qu'aucun réseau n'a eu à utiliser les 50 primitives proposées, et qu'en moyenne, il n'en utilise qu'une sur deux.

L'histogramme de la figure [13] concerne la taille des bases d'apprentissage propres à chaque réseau. On voit que la plupart de ces bases comportent entre 300 et 600 prototypes, ce qui correspond à un pourcentage compris entre 3,5 et 6,9% des 8800 prototypes de la base d'apprentissage. De plus, une analyse plus fine des prototypes contenus dans ces bases montre que les réseaux sélectionnent bien les prototypes utiles pour caractériser les frontières de la classe qu'ils sont chargés d'identifier. En

effet, on constate qu'à côté des prototypes de leur propre classe, ils sélectionnent préférentiellement les prototypes des classes « proches ». Par exemple, le réseau « 1 » sélectionne dans sa base 454 prototypes dont 126 « 1 » mais aussi 68 « I » et 39 « t », alors que 30 classes n'ont aucun représentant dans cette base; de même, le réseau « 5 » sélectionne 482 prototypes dont 114 « 5 », mais aussi 83 « S » et 77 « 6 » alors que 22 classes n'ont aucun représentant.

Les capacités de généralisation ont été validées sur une base de test supplémentaire d'environ 3900 exemples, avant que les erreurs issues de cette base ne soient utilisées en ré–apprentissage. En effet, le principe de la méthode ne différencie pas fondamentalement le rôle d'une base de test de celui de la base utilisée initialement pour la construction du classifieur, toute erreur issue d'une base de test pourra toujours venir enrichir les données des bases d'apprentissage des réseaux.

L'histogramme de la figure [15] montre que si les performances en rejet sont bonnes pour tous les réseaux, il n'en est pas de même pour les performances en reconnaissance illustrés sur la figure [14] (résultats obtenus avant ré-apprentissage sur la base de test). Ceci peut s'expliquer par le fait que, le problème comportant 49 classes, chacune des classes n'a qu'entre 100 et 200 prototypes dans la base initiale de 8800 prototypes, ce qui semble insuffisant pour avoir une description fiable des frontières de chacune des classes. L'analyse des performances du classifieur (ensemble des réseaux) sur la base de test peut alors se résumer dans le « tableau 1 » :

Tableau 1 – Performances globales du classifieur

Reconnaissance	3281	84,4 %
Ambigüité 2 réseaux	209	5,3 %
Ambigüité 3 réseaux	7	0,2 %
Autres ambigüités	4	0,1 %
Total reconnaissance	3501	90,1 %
Rejet	338	8,7 %
Confusions	49	1,3 %
Total base	3888	100 %

Une « ambiguité à n réseaux » signifie que le réseau correspondant à la classe du prototype analysé a bien répondu, mais que $n-1$ autres réseaux ont aussi considéré que ce prototype appartenait à la classe qu'ils étaient chargés d'identifier. La ligne « autres ambiguités » du tableau rassemble les ambiguités à plus de trois réseaux et les ambiguités par non décision. Une « ambiguité par non–décision » signifie qu'un ou plusieurs réseaux, dont celui correspondant à la classe du prototype analysé, n'ont pas conclu soit à l'appartenance soit au rejet. Une confusion est caractérisée

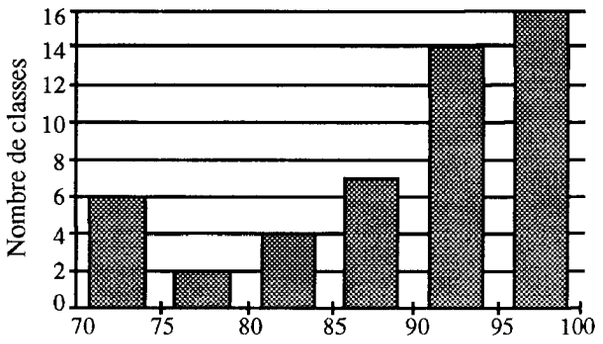


Figure 14. – Pourcentage de bonne reconnaissance.

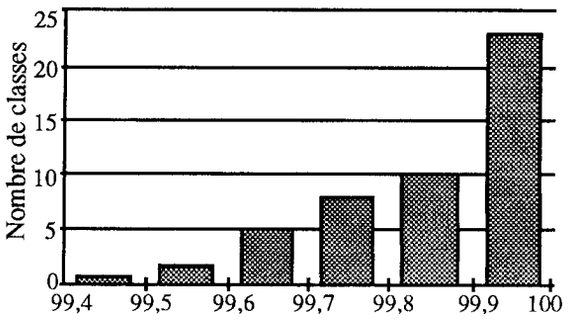


Figure 15. – Pourcentage de rejets corrects.

par le fait que le réseau correspondant à la classe du prototype analysé a rejeté ce prototype, et que un ou plusieurs autres réseaux ont, par contre, considéré que ce prototype appartenait à la classe qu'ils étaient chargés d'identifier.

Cette analyse globale montre que le principal effet des performances relativement moyennes de certains réseaux en reconnaissance est de générer des rejets : aucun réseau ne répond que le prototype appartient à la classe qu'il doit identifier pour 8,7 % des prototypes de la base. Par contre, les décisions prises peuvent être considérées comme fiables. Si l'on définit f, fiabilité de décision (hors rejet) par l'équation 1 :

$$f = 1 - \frac{nbc}{nbr + nbc} \qquad (1)$$

avec nbc = nombre de confusions, et nbr = nombre de prototypes reconnus (reconnaissance + ambiguité), on obtient une valeur de f égale à 98,6 %.

La génération d'ambiguités est un problème moins pénalisant que celui des confusions. En effet, cela correspond à générer plusieurs hypothèses parmi lesquelles figure l'hypothèse correcte : cela signifie que ces ambiguités pourront être levées éventuellement par un traitement ultérieur de nature contextuelle par exemple. Une analyse plus détaillée des principales erreurs commises par les réseaux montre que ces ambiguités proviennent essentiellement de formes proches telles que « I 1 » (18 cas) ou réciproquement « 1 I » (8 cas), ou « Q, O » (11 cas) ou réciproquement « O Q » (8 cas).

4.2. APPLICATION À LA RECONNAISSANCE DE CHIFFRES MANUSCRITS (BASE NIST)

Afin de poursuivre la validation de la méthodologie proposée, nous nous intéressons actuellement au problème de la reconnaissance de chiffres manuscrits issus de la base NIST. Les résultats présentés ici sont des résultats préliminaires puisque seule une fraction de 20000 chiffres de la base NIST a été utilisée. Ce problème a fait l'objet d'une abondante littérature, et a en particulier reçu une grande attention dans le cadre des applications des réseaux de neurones [Lecun], [Lee], [Martin]...etc. Par comparaison au problème analysé précédemment, les différences essentielles proviennent (i) d'un nombre de classes plus faible (10), (ii) d'une variabilité plus grande des formes de prototypes à l'intérieur de ces classes, (iii) d'un nombre accru de caractéristiques d'entrées : en effet, nous utilisons un vecteur d'entrée comportant 124 primitives [Heuttel–2] (intersections avec des sondes, moments invariants de Hu, profils, projections horizontales et verticales, extrema, primitives structurelles, primitives basées sur la polygonalisation du tracé...). Un premier apprentissage a été réalisé sur une base équilibrée de 10000 prototypes (1000 prototypes de chaque classe). Les fractions de la base retenues dans les bases d'apprentissage de chaque réseau sont plus importantes que dans l'application précédente (de 13,5 à 32,6 %) montrant ainsi la plus grande variabilité des prototypes. La taille des réseaux se trouve également augmentée (de 217 à 727 yprels). Les réseaux sélectionnent cependant les primitives qui leur sont utiles (de 48 à 91 primitives utilisées sur 124 disponibles). Les performances obtenues sur une autre base de 10000 prototypes sont résumées sur le tableau 2 :

Tableau 2 – Performances obtenues pour la reconnaissance de chiffres manuscrits

Reconnaissance	7660	76,60 %
Ambigüité 2 réseaux	791	7,91 %
Ambigüité 3 réseaux	64	0,64 %
Autres ambigüités	3	0,03 %
Total reconnaissance	8518	85,18 %
Rejet	1097	10,97 %
Confusions	385	3,85 %
Total base	10000	100 %

Dans ce cas, le facteur de qualité de l'équation [1] vaut 95,67 %. Ces résultats sont à rapprocher des résultats obtenus individuellement pour les dix réseaux constituant le classifieur, résumés sur le tableau [3] :

Tableau 3 – Résultats (en %) obtenus pour chaque classe

Réseau	Reco.	Rejet	Non Décision
0	93,50	99,46	0,00
1	89,60	99,60	0,00
2	76,50	96,43	0,00
3	78,81	97,79	0,01
4	82,70	98,60	0,02
5	83,50	97,94	0,00
6	93,90	99,35	0,01
7	91,10	99,31	0,01
8	77,40	97,39	0,00
9	84,80	98,90	0,00

Ces premiers résultats sont à améliorer en poursuivant les apprentissages à partir d'autres bases de prototypes. A cet effet une amélioration du réseau du « 2 » a par exemple été obtenue en reprenant un apprentissage sur une autre fraction de la base NIST de 10000 prototypes. Cependant plusieurs points sont actuellement en cours d'étude. En effet, d'une part, le grand nombre de primitives d'entrée induit une combinatoire très importante des combinaisons possibles, et d'autre part la taille des réseaux et la taille des bases d'apprentissage augmentent; ces trois faits se conjuguent pour rallentir sensiblement les apprentissages successifs malgré la ré–initialisation par le réseau de l'étape précédente. Les tests en cours sont réalisés en utilisant le principe de coopération des réseaux, ce qui devrait permettre d'améliorer sensiblement les performances des réseaux.

Les réflexions actuelles portent (i) sur la mise au point d'un test d'arrêt permettant d'éviter les phénomènes de sur–apprentissage des réseaux, (ii) sur la possibilité de simplifier un réseau par analyse des parties du réseau impliquées majoritairement lors des décisions de classification ou de rejet, par exemple par des techniques analogues à celles utilisées pour les arbres de décision [Breiman], (iii) sur la possibilité de simplifier une base après un certain nombre de cycles en fonction de l'évolution possible de la structure de ce réseau, et (iv) sur la possibilité de guider plus efficacement les combinaisons à tester lors de l'apprentissage.

5. Conclusion et perspectives

L'article montre que de bonnes performances peuvent être obtenues pour réaliser une tâche de classification supervisée en utilisant l'approche des réseaux d'yprels. Ces performances peuvent toujours être améliorées si des situations différentes de celles étudiées lors de l'apprentissage se présentent grâce à l'apprentissage incrémental. Le traitement par classe permet de ne reprendre qu'un seul réseau défaillant en laissant inchangé le reste du classifieur. Remarquons que la méthodologie permet sans difficulté de traiter des problèmes pour lesquels certaines classes ont un recouvrement partiel. Actuellement la suite du travail porte sur l'optimisation de la taille des réseaux construits et sur la mise en oeuvre d'un processus de coopération entre plusieurs réseaux, la sortie de certains réseaux pouvant être considérée comme des primitives auxiliaires pour d'autres.

6. Annexes

A.1. CALCUL DES PARAMÈTRES D'UN YPREL

Cette annexe précise la méthode de détermination des paramètres d'un yprel dont une version antérieure a été présentée dans [Lecourtier3]. Comme indiqué au paragraphe 2, chaque yprel tente de prendre une décision. Trois sortes de décisions sont possibles pour le prototype proposé : l'yprel peut soit conduire à une décision d'appartenance à la classe considérée, soit rejeter le prototype comme appartenant à une autre classe, ou encore conduire à une non décision. Un tel fonctionnement nécessite un codage particulier de l'information transmise sur la sortie d'un yprel. D'autre part, deux types d'yprels sont à considérer : ceux constituant la couche d'entrée et ayant une seule entrée, et les yprels généraux à deux entrées. Nous présentons dans la première partie de cette annexe le codage de l'information utilisé dans la méthodologie, nous préciserons ensuite les caractéristiques de chaque type d'yprels et la règle de calcul de leurs paramètres.

1) Codage de l'information

Le codage de l'information transmise par un yprel sur sa sortie est basé sur la règle suivante :

- une valeur de sortie négative est interprétée comme une décision de rejet : le prototype présenté n'appartient pas à la classe que doit identifier le réseau considéré.
- une valeur de sortie appartenant à l'intervalle $[0, 1]$ est interprétée comme une non décision. L'yprel n'a pas été capable de décider s'il fallait rejeter ou accepter le prototype considéré.
- une valeur de sortie supérieure à 1 sera interprétée comme correspondant à une prise de décision autre que le rejet. Une valeur supérieure à 1 sera interprétée comme un codage d'une décision et pourra donc être utilisée pour la construction d'une explication. Les valeurs supérieures à 1 doivent être considérées comme de nature discrète, alors que les valeurs inférieures à 1 sont à variation continue. Dans la version actuelle du logiciel seule la valeur logique arbitraire 2 est utilisée et est interprétée comme une décision d'appartenance à la classe identifiée par le réseau. D'autres valeurs logiques identifiants des ambiguïtés entre classes sont actuellement en cours d'implémentation.

2) Yprels de la couche d'entrée

Chaque yprel de la couche d'entrée est associé à une primitive et joue un rôle de normalisation de la valeur des primitives pour la base considérée : tous les éléments de la base d'apprentissage appartenant à la classe que doit identifier le réseau doivent conduire à une valeur de sortie de l'yprel appartenant à l'intervalle $[0, 1]$ de non décision. Un yprel de la couche d'entrée ne prendra donc aucune décision d'appartenance à la classe, mais pourra prendre des décisions de rejet.

Le calcul de la valeur y de sortie de l'yprel de la couche d'entrée associé à la primitive x est donc le suivant :

- calculer $y(x) = h \cdot x + k$,
- $y < 1$, retourner $y(x)$, sinon retourner $1 - y(x)$

L'apprentissage des paramètres h et k de l'yprel s'effectue selon le schéma suivant :

- sur l'ensemble des valeurs de la primitive x correspondant aux prototypes de la classe à identifier, calculer les valeurs minimales et maximales de x, soient $Min(x)$ et $Max(x)$ ces valeurs.
- déterminer h et k tels que $y(Min(x)) = 0$ et $y(Max(x)) = 1$.

Cette règle de calcul de la sortie de l'yprel et ce schéma d'apprentissage assurent bien le respect du codage de l'information définie précédemment pour tous les éléments de la base d'apprentissage :

- tous les prototypes de la classe à apprendre conduiront à une valeur de sortie de l'yprel appartenant à l'intervalle $[0, 1]$ par choix de h et k.
- tous les prototypes des autres classes ne pourront que conduire à une valeur inférieure à 1. Si cette valeur appartient à $[0, 1]$, aucune décision ne sera prise par l'yprel, alors qu'une valeur inférieure à 0 sera interprétée comme un rejet.

3) Yprels généraux

Un yprel général admet deux entrées qui sont des sorties d'autres yprels selectionnés comme indiqué au paragraphe 3. Soient $n1$ et $n2$ les yprels dont les sorties y_{n1} et y_{n2} seront utilisées comme entrées de l'yprel n. Rappelons que si une décision de classification (appartenance ou rejet) est prise par au moins une des deux entrées, cette décision est transmise à la sortie y_n de l'yprel n. Ceci est valable sauf quand les deux entrées aboutissent à deux décisions incompatibles (appartenance et rejet).Une telle situation étant impossible en cours d'apprentissage, elle est codée avec une valeur particulière supérieure à 1 en phase de test et un traitement particulier et non présenté ici est opéré. Dans le cas où aucune décision de classification n'est prise par les deux entrées, la sortie de l'yprel « n » obeit à la règle suivante :

– calculer $y_n = h\,(a_1 y_{n1} + a_2 y_{n2}) + k$

– si $0 \leq y_n \leq 1$, retourner y_n ;

– si $y_n < 0$, alors $y_n = S_0$;

– si $y_n > 1$, alors $y_n = S_1$.

Les paramètres a_1, a_2, h, k, S_0 et S_1 sont déterminés en cours d'apprentissage. Du fait des valeurs possibles des entrées de l'yprel n, l'ensemble des prototypes de la base d'apprentissage se répartit dans l'espace de coordonnées (y_{n1}, y_{n2}) comme illustré sur la figure [16].

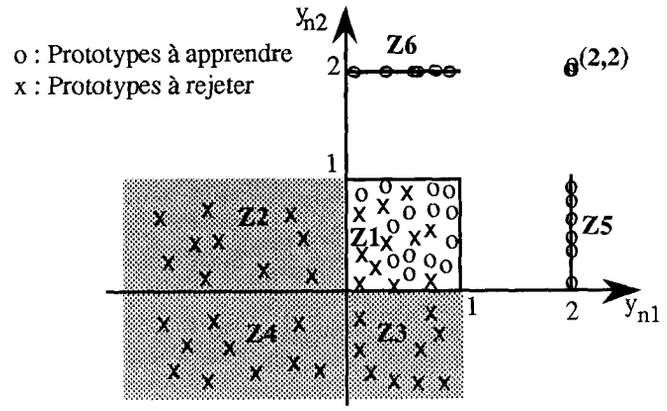


Figure 16. – Répartition des prototypes d'apprentissage dans l'espace des deux entrées d'un yprel.

La figure [16] fait apparaître sept zones différentes :

– $Z1$ contient les éléments que ni y_{n1} ni y_{n2} ne sont capables de classer,

– $Z2$ (resp. $Z3$) contient les éléments que y_{n1} (resp. y_{n2}) rejette, mais que y_{n2} (resp. y_{n1}) n'est pas capable de classer,

– $Z4$ contient les éléments rejetés à la fois par les deux entrées,

– $Z5$ et $Z6$ sont des zones où une seule des deux entrées est capable de prendre une décision d'appartenance à la classe considérée, alors que l'autre entrée conduit à une non–décision,

– le point (2,2) contient tous les éléments identifiés comme appartenant à la classe considérée par les deux entrées à la fois.

Les paramètres a_1, a_2, h, k, S_0 et S_1 de l'yprel « n » dépendent uniquement des éléments appartenant à $Z1$ et leur détermination se fait selon l'algorithme suivant :

1 – déterminer la liste des éléments de la base d'apprentissage de la zone $Z1$,

2 – calculer la combinaison linéaire des entrées $a_1 y_{n1} + a_2 y_{n2}$ tel que les éléments de la classe à identifier soient regroupés dans un intervalle le plus court possible, et que les éléments des autres classes soient rejetés le plus possible à l'extérieur de cet intervalle.

3 – déterminer l'intervalle $[f_{o\min}, f_{o\max}]$ contenant tous les éléments appartenant à la classe à identifier, et l'intervalle $[f_{x}\min, f_{x}\max]$ contenant les éléments des autres classes. A partir des 4 bornes de ces intervalles, déduire les deux paramètres h et k tel que si $y_n = h(a_1 y_{n1} + a_2 y_{n2}) + k$ est à l'extérieur de l'intervalle $[0, 1]$, une décision de rejet ou d'appartenance à la classe est prise.

A l'étape 2, les paramètres a_1 et a_2 sont déterminés par la minimisation d'un critère classique de moindres carrés avec contrainte. A l'étape 3, si on note $[f_{\min}, f_{\max}]$ l'intervalle contenant aussi bien les éléments de la classe à apprendre que les éléments des autres classes pour lesquels aucune décision n'est possible, les seules situations possibles sont illustrées sur la figure [17]. Les valeurs correspondantes de S_0 et S_1 sont alors déduites. Les paramètres h et k sont déterminés de sorte que l'intervalle $[f_{\min}, f_{\max}]$ soit appliqué sur $[0, 1]$.

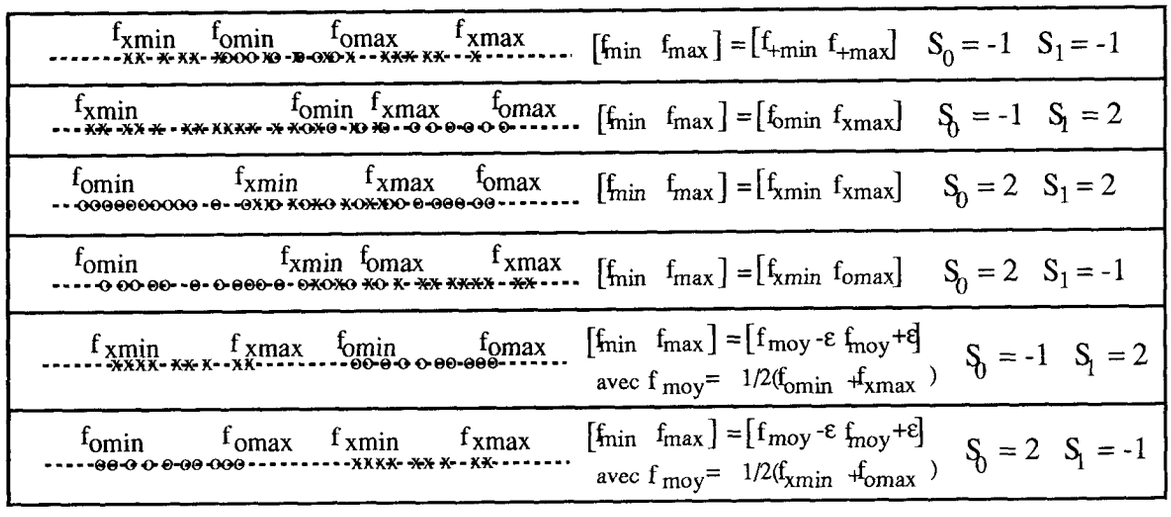


Figure 17 – Les différentes situations possibles pour la détermination des paramètres de l'yprel.

BIBLIOGRAPHIE

[Belaïd] A. Belaïd, Y. Belaïd, *Reconnaissances des formes, méthodes et application*, InterEdition, 1992.

[Booker] L.B. Booker, D.E. Goldberg, J.H. Holland, « Classifier Systems and Genetic Algorithms », *Artificial Intelligence*, n°40, 1989, pp. 235-282.

[Breiman] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, *Classification and Regression Trees*, Wadsworth&Brooks, Pacific Grove, CA, 1984.

[Dubuisson] B. Dubuisson, *Diagnostic et reconnaissance des formes*, Hermes, 1990.

[Fahlman] S.E. Fahlman, C. Lebiere, « The Cascade-Correlation Learning Architecture », in *Advances in Neural Information Processing Systems*, D.S. Touretsky Ed., M. Kauffmann, n°2, 1990, pp. 524-532.

[Frean] M. Frean, « The Upstart algorithm : a method for constructing and training feed-forward neural networks », *Neural Computation*, n°2, 1990, pp. 198-209.

[Gentric] P. Gentric, H. Withagen, « Constructive Methods for a new Classifier Based on a Radial-basis-function Neural Network accelerated by a Tree », in *New Trends in Neural Computation*, J. Mira, J. Cabestani, A. Prieto Eds, Springer-Verlag, Berlin, 1993, pp. 125-130.

[Gaillat] G. Gaillat, *Méthodes statistiques en reconnaissance des formes*, Ed ENSTA, Paris, 1983.

[Golberg] D.E. Golberg, *Algorithmes génétiques*, Addison-Wesley, France, 1994.

[Hecht] R. Hecht-Nielsen, *Neurocomputing*, Addison-Wesley, Reading, MA., 1990.

[Heutte1] L. Heutte, J.V. Moreau, Y. Lecourtier, « A new feature extraction strategy for handwritten pseudo-character recognition based on multiple feature extractors », in *Proc. 6th IGS International Conference on Handwriting and Drawing*, ICOHD'93, 1993, pp. 186-188.

[Heutte2] L. Heutte, J.V. Moreau, B. Plessis, J.L. Plagnaud, Y. Lecourtier, « Handwritten numeral recognition based on multiple feature extractors », *Proc. 2nd IAPR International Conference on Document Analysis and Recognition*, ICDAR'93, Tsukuba, Japan, 1993, pp. 167-170.

[Hüning] H. Hüning, « A node Splitting Algorithm that reduces the Number of Connections in a Hamming Distance Classifying Network », in *New Trends in Neural Computation*, J. Mira, J. Cabestani, A. Prieto Eds., Springer-Verlag, Berlin, 1993, pp. 102-107.

[Kohonen] T. Kohonen, *Self organisation and associative memory*, Springer-Verlag, Berlin, 1986.

[Knerr] S. Knerr, L. Personnaz, G. Dreyfus, « Une nouvelle approche de la reconnaissance de chiffres manuscrits par réseaux de neurones », BIGRE , n°80, 1992, p. 325-332.

[Lecourtier1] Y. Lecourtier, A. Ennaji, J. Le Bas, « Réseaux d'yprels et classification », *Bigre*, n°80, 1992, p. 286-294.

[Lecourtier2] Y. Lecourtier, B. Dorizzi, P. Sebire, A. Ennaji, « MLP Modular versus Yprel Classifiers », in *New Trends in Neural Computation*, J. Mira, J. Cabestani, A. Prieto Eds., Springer-Verlag, Berlin, 1993, pp. 569-574.

[Lecourtier3] Y. Lecourtier, A. Ennaji, F. Gilles, P. Chavy, « Yprel nets and Classification », *IEEE SMC Conference*, n°3, Le Touquet, 1993, pp. 463-468.

[Lecun] Y. Le Cun, B. Boser, J.S. Denkere, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, « Backpropagation applied to Handwritten Zip Code Recognition », *Neural Computation*, n°1, 1989, pp. 541-551.

[Lee] Y. Lee, « Handwritten digit recognition using K nearest neighbor, radial basis function and backpropagation neural networks », *Neural Computation*, n°3, 1991, pp. 440-449.

[Martin] G.L. Martin, J. A. Pittman, « Recognizing hand printed letters and digits using backpropagation learning », *Neural Computation*, n°3, 1991, pp. 258-267.

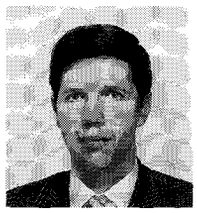
[Perez] J.C. Perez, E. Vidal, « Constructive Design of LVQ and DSM Classifiers », in *New Trends in Neural Computation*, J. Mira, J. Cabestani, A. Prieto Eds., Springer-Verlag, Berlin, 1993, pp. 334-339.

[Rumelhart] J. Rumelhart, J. McClelland, *Parallel Distributed Processing*, MIT Press, Cambridge MA, 1986.

[Stocker] E. Stocker, Y. Lecourtier, A. Ennaji, « A distributed classifier based on Yprel networks cooperation », *From natural to artificial neural computation*, J. Mira, F. Sandoval Eds, Springer-Verlag, Berlin, 1995, pp. 330-337.

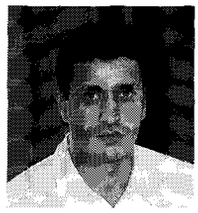
LES AUTEURS

Yves LECOURTIER



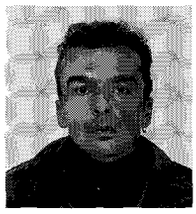
Yves Lecourtier a soutenu un doctorat de 3ème cycle en traitement du signal en 1978 puis un doctorat d'état en automatique théorique en 1985 à l'Université Paris-Sud. Son domaine de recherche portait alors principalement sur l'étude des propriétés structurelles des modèles dynamiques décrits par équations d'état. Professeur à l'Université de Rouen depuis 1987, il a réorienté ses activités vers la reconnaissance de formes et les réseaux neuro-mimétiques avec applications principales à l'analyse de documents et la reconnaissance de l'écrit. Il préside actuellement le GRCE, groupe de recherche en communication écrite, qui rassemble les chercheurs français du domaine.

Abdel ENNAJI



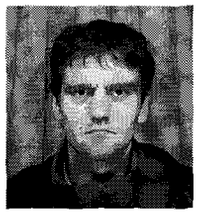
Abdel Ennaji est Maître de conférence à l'Université de Rouen depuis 1993. Il a obtenu un DEA Instrumentation et Commande en 1989 puis a soutenu un doctorat de l'Université de Rouen en 1993. Ses activités de recherche portent essentiellement sur les techniques de classification et en particulier les réseaux neuro-mimétiques. Ces techniques sont utilisées dans le domaine de la reconnaissance de formes et en particulier pour le développement de stratégies de reconnaissance en lecture optique de texte.

Emmanuel STOCKER



Emmanuel Stocker a suivi des études d'informatique spécialisées en Intelligence Artificielle. Il est titulaire du DEA I.A.R.F.A. (Intelligence Artificielle, Reconnaissanes des Formes et Applications) de l'Université Pierre et Marie Curie (Paris VI). Il termine actuellement une thèse de doctorat au laboratoire LA3i de l'Université de Rouen dont le but est d'utiliser des principes issus de l'intelligence artificielle distribuée et des systèmes multi-agents pour améliorer la stratégie d'apprentissage des réseaux neuro-mimétiques d'Yprels.

Frédéric GILLES



Frédéric Gilles a suivi des études d'électronique, d'électrotechnique et d'automatique. Il est titulaire du DEA Instrumentation et Commande option traitement d'image de l'Université de Rouen. Il termine actuellement une thèse de doctorat au laboratoire LA3i/LCIA de l'Université de Rouen dont le but est d'étudier des systèmes neuro-mimétiques à évolution autonome dans des boucles sensori-motrices en s'inspirant de modèles neuronaux proches des concepts de la biologie et de la physiologie.

Manuscrit reçu le 7 Novembre 1995.