Une nouvelle fonction de coût régularisante dans les réseaux de neurones artificiels

Application aux réseaux discriminants

Vincent Lemaire — Olivier Bernier — Daniel Collobert
Fabrice Clérot

France Télécom Recherche et Développement
DTL/DLI/TNT
Technopole Anticipa, 2 Avenue Pierre Marzin
F-22307 Lannion cedex

vincent.lemaire@francetelecom.com

RÉSUMÉ. *On présente une nouvelle méthode destinée à améliorer les performances en généralisation des perceptrons multicouches utilisés en tant que réseaux discriminants. On montre comment modifier le critère d'apprentissage afin de contrôler la distribution des erreurs au cours de l'apprentissage. Ce contrôle permet d'obtenir une meilleure marge dans les problèmes de classification. Des résultats améliorant notablement l'état de l'art sur deux différents problèmes sont présentés et valident la méthode.*

ABSTRACT. *A novel method is presented to enhance the generalisation performances of multilayer perceptron used as discriminant networks. We clearly show how to modify the learning criterium in order to control the error distribution. This control allows to obtain a better margin in classification problems. Results are given for two different problems. These results substantially improve state of the art results and validate the method.*

MOTS-CLÉS : *apprentissage, classification, régularisation, marge, réseaux de neurones.*

KEYWORDS: *learning, classification, regularisation, margin, neural networks.*

RIA – 14/2000 CAP'99, pages 287 à 311

1. Introduction

Le choix de la capacité d'un réseau de neurones est primordial pour ses capacités d'apprentissage et de généralisation. Si le modèle est trop simple, il sera incapable d'apprendre la fonction souhaitée, s'il est trop complexe; il sera incapable de généraliser, (bien que capable sans peine de passer par tous les points de l'ensemble d'apprentissage). En fait, dans le second cas, l'espace des solutions admissibles est beaucoup trop grand. La fonction de coût est aussi l'un des plus importants facteurs contrôlant les performances d'un perceptron multicouche.

La fonction de coût standard est la fonction quadratique [RUM 86] avec pour critère d'arrêt le passage à un seuil de l'erreur quadratique moyenne obtenue sur l'ensemble de validation. Nous appellerons cette méthode "MSE" dans tout ce qui suit.

De nombreux algorithmes ont été proposés pour accélérer l'apprentissage, pour trouver le "bon" pas d'apprentissage, ou trouver la bonne méthode d'arrêt de l'apprentissage, mais ils sont très souvent destinés à la minimisation de cette fonction de coût MSE. On peut noter cependant que bien qu'elle vise à réduire la norme des erreurs commises par le réseau, elle n'a aucun pouvoir de contrôle sur la distribution de ces dernières au cours de l'apprentissage. Or, contrôler la "forme" de la distribution des erreurs lors de la phase d'apprentissage nous a semblé être un problème de grande importance, car relié à la confiance de la prédiction effectuée, formalisée en termes de marge de classification. En d'autres termes, est-il possible de contrôler la convergence de l'algorithme d'apprentissage de manière à contrôler la forme de la distribution des erreurs et d'améliorer ainsi la robustesse de l'apprentissage ? Un contrôle de la forme de la distribution des erreurs permet-il un contrôle de la capacité du réseau de neurones et donc une meilleure généralisation ? Dans ce qui suit, une nouvelle fonction de coût est proposée dans ce but.

La deuxième section de cet article donne une brève introduction aux réseaux de neurones artificiels. Lors de la troisième section, la nouvelle fonction de coût et ses conditions d'utilisation seront décrites. La quatrième section est une comparaison entre la nouvelle fonction de coût et la fonction de coût MSE, dans le cadre d'un problème de détection de visages à l'aide d'un unique perceptron multicouche.

Dans la section 5, on s'attachera à réaliser une autre comparaison entre les deux fonctions de coût sur un autre problème de classification qualifié de benchmark. Le but sera de comparer les résultats obtenus par la technique de Bagging mise au point par Breiman [BRE 94, BIS 97] et plus particulièrement les résultats affichés dans [QUI 98] obtenus avec la fonction de coût MSE avec ceux obtenus par la nouvelle fonction de coût.

2. Une brève introduction aux réseaux de neurones

Les réseaux de neurones sont des approximateurs universels de fonctions [WHI 89]. Les réseaux de neurones, encore appelés *modèles connexionnistes*, sont constitués

d'un grand nombre de cellules (opérateurs, unités) simples et fortement interconnectées. Chaque opérateur effectue une transformation non linéaire (appelée *fonction de transfert* et qui est du genre tangente hyperbolique) de la somme pondérée de ses entrées. Les coefficients de pondération sont appelés *poids* (en référence à *poids synaptiques* pour des raisons historiques). De fait, toute la *connaissance* du réseau se trouve dans ces poids. Un réseau simple de neurones est constitué de couches d'unités (les neurones) où toute unité d'une couche est reliée à toutes les unités de la couche supérieure. La couche inférieure, appelée "couche d'entrée", reçoit les données, la couche supérieure, appelée "couche de sortie", donne le résultat. Les couches intermédiaires sont appelées "couches cachées". Cette architecture porte le nom de *perceptron multicouche*.

Connaissant la structure du réseau (les interconnexions), l'idée est de trouver les poids qui minimisent la différence entre les valeurs délivrées par le réseau et les valeurs désirées pour toutes les données.

Les données sont souvent divisées en deux parties appelées *ensemble d'apprentissage* et *ensemble de test*. L'ensemble d'apprentissage est utilisé pour déterminer les poids qui minimisent une certaine fonction de coût. La procédure généralement utilisée est une procédure itérative et la *rétropropagation du gradient d'erreur* en est la technique la plus populaire [RUM 86]. L'ensemble de test est utilisé pour vérifier les performances réelles du réseau sur des données "non apprises".

2.1. *Fonction de coût*

Une gamme de fonctions de coût peut être utilisée pour optimiser les performances d'un réseau de neurones. La fonction de coût la plus utilisée est celle qui mimimise l'erreur quadratique, elle peut être écrite :

$$C = \frac{1}{|P|} \sum_{x \in P} \sum_{i \in O} (d_i^x - \hat{s}_i^x)^2, \qquad [1]$$

où P est l'ensemble des exemples d'apprentissage, O l'ensemble des neurones de sortie, s_i^x la valeur du neurone de sortie i après la présentation de l'exemple x et d_i^x la valeur désirée pour le neurone correspondant.

2.2. *Procédure d'apprentissage*

Etant donné une fonction de coût, une architecture et une base de données, le but est ensuite de trouver les valeurs appropriées des poids qui minimisent la fonction de coût sur les exemples de l'ensemble d'apprentissage. Ceci est habituellement réalisé à l'aide d'une procédure itérative constituée des étapes suivantes : premièrement, initialiser les poids du réseau aléatoirement, puis pour chaque exemple de l'ensemble d'apprentissage, calculer la sortie du réseau lui correspondant, comparer le résultat

avec la valeur désirée et appliquer une correction de tous les poids afin de minimiser l'erreur. Une itération est la présentation de tous les exemples. La procédure complète peut représenter beaucoup d'itérations.

Pour l'apprentissage supervisé d'un perceptron multicouche, par correction d'erreur, l'algorithme le plus utilisé est l'algorithme de descente de gradient. Le calcul du gradient se fait en utilisant l'algorithme de la rétropropagation de l'erreur. L'algorithme d'apprentissage utilisant ce procédé a été découvert par [LEC 87, RUM 86] et reste encore aujourd'hui la méthode d'apprentissage la plus largement utilisée. La modification des poids du réseau de neurones est réalisée à l'aide d'un algorithme de gradient qui est de la forme :

$$\mathbf{W}^{t+1} = \mathbf{W}^{t} - \alpha \nabla C(\mathbf{W}^{t}) \qquad [2]$$

où la matrice **W** représente les poids du réseau, ∇ le gradient de la fonction de coût par rapport aux poids **W** et α un coefficient de modification des poids, appelé *pas d'apprentissage*.

Il existe deux méthodes principales de modification des poids du réseau liées à la manière de calculer le gradient: soit en utilisant un gradient total qui est une méthode globale encore appelée "batch", soit en utilisant un gradient partiel, qui est appelée "méthode stochastique". Bottou présente dans [BOT 91] une comparaison des deux méthodes et montre que la méthode stochastique est plus "rapide" que la méthode globale. Les propriétés de convergence de la rétropropagation "standard" (telle que proposée dans [LEC 87, RUM 86]), en version stochastique et en version "batch", sont discutées dans [BER 96]. Dans la suite de cet article, la méthode stochastique a été utilisée.

2.3. *Généralisation et sélection du modèle*

Le point important est la capacité de généralisation du réseau appris, *i.e.* la qualité des prédictions effectuées sur des données non apprises. Une estimation de l'erreur de généralisation est donnée par les performances du réseau sur des données test, représentatives du problème considéré et non étudiées lors de l'apprentissage. La différence entre l'erreur d'apprentissage et l'erreur de généralisation représente une mesure de la qualité de l'apprentissage effectué [BOU 92, VAP 82, VAP 95].

L'erreur de généralisation dépend avant tout de trois paramètres : le nombre d'exemples utilisés pour l'apprentissage, la complexité du problème sous-jacent et l'architecture du réseau. Les approches statistiques de la généralisation [VAP 71, VAP 82] sont un des domaines d'investigation majeur pour optimiser les performances de l'apprentissage des réseaux de neurones. Ce domaine est devenu, avec le temps, très riche et complexe, aussi le lecteur est invité à considérer par exemple l'article de synthèse de Wolpert [WOL 92].

Les methodes qui évaluent l'erreur de généralisation sont en général fondées sur la partition des données disponibles en trois sous-ensembles. Le premier sous-ensemble

appelé *ensemble d'apprentissage* est utilisé pour optimiser les poids du réseau correspondant à une architecture donnée. Le second sous-ensemble appelé *ensemble de validation* permet de comparer plusieurs architectures de réseaux et de retenir la meilleure. Le deuxième ensemble permet également de contrôler et de mesurer la généralisation du réseau au cours de l'apprentissage. Enfin, le troisième sous-ensemble appelé *ensemble de test* sert à estimer l'erreur de généralisation du réseau déterminé d'après les ensembles d'apprentissage et de validation. Le lecteur pourra trouver une comparaison de ces méthodes dans [TIB 96].

3. Présentation de la nouvelle méthode

3.1. *Description*

Une manière de contrôler la forme de la distribution des erreurs au cours de l'apprentissage est la prise en compte d'un moment d'ordre 4 des erreurs : la variance des erreurs quadratiques, en plus de l'erreur quadratique classique.

La fonction de coût à minimiser devient :

$$C^x = \sum_{i \in O} (d_i^x - s_i^x)^2 + \sum_{i \in O} \left(\frac{1}{P} \sum_{a=1}^{P} \left((d_i^a - s_i^a)^2 - \frac{1}{P} \sum_{b=1}^{P} (d_i^b - s_i^b)^2 \right)^2 \right) \quad [3]$$

qui peut s'écrire sous la forme de l'addition de deux coûts :

$$C^x = \sum_{i \in O} C_{quad}^x + \sum_{i \in O} C_{var}^x \quad [4]$$

où P est l'ensemble des exemples d'apprentissage, O l'ensemble des neurones de sortie, s_i^x la valeur du neurone de sortie i après la présentation de l'exemple x et d_i^x la valeur désirée pour le neurone correspondant.

Nous avons utilisé la méthode stochastique et dans ce cas, le gradient attaché à un neurone de sortie i pour une présentation d'un exemple x est :

$$G_i^x = \frac{\partial C_i^x}{\partial a_i^x} \quad [5]$$

$$= \frac{\partial}{\partial a_i^x} \left((d_i^x - s_i^x)^2 \right) + \frac{\partial}{\partial a_i^x} \left(\frac{1}{P} \sum_{a=1}^{P} \left((d_i^a - s_i^a)^2 - \frac{1}{P} \sum_{b=1}^{P} (d_i^b - s_i^b)^2 \right)^2 \right) \quad [6]$$

donc, si f est la fonction de transfert d'un neurone :

$$G_i^x = \frac{\partial C_i^x}{\partial a_i^x} \quad [7]$$

$$= \quad -2f'(a_i^x)(d_i^x - s_i^x) - \frac{4}{P} f'(a_i^x)(d_i^x - s_i^x) \left((d_i^x - s_i^x)^2 - \frac{1}{P} \sum_{e=1}^{P} (d_i^e - s_i^e)^2 \right) \quad [8]$$

Le gradient peut être écrit sous la forme :

$$G_i^x = -2f'(a_i^x)(d_i^x - s_i^x) - \frac{4}{P} f'(a_i^x)(d_i^x - s_i^x) \left((d_i^x - s_i^x)^2 - MSE \right) \quad [9]$$

ou encore :

$$G_i^x = G_{i_{quad}}^x (1 + \gamma) \quad [10]$$

avec :

$$\gamma = \frac{2}{P} \left((d_i^x - s_i^x)^2 - MSE \right) \quad [11]$$

où MSE représente l'erreur quadratique moyenne obtenue sur tous les exemples avant la présentation de l'exemple x.

On s'aperçoit que ce gradient peut être vu comme le gradient utilisé habituellement, dû à l'erreur quadratique, mais corrigé, pondéré, par γ qui représente une mesure entre l'erreur quadratique commise sur l'exemple x et l'erreur quadratique moyenne commise sur l'ensemble des exemples. Dans tout ce qui suivra, nous appellerons cette méthode d'apprentissage "VMSE" pour *Variance and Mean Squared Error*. Le principe du calcul du gradient d'un neurone caché est à l'identique de la rétropropagation de l'erreur quadratique (méthode MSE).

3.2. *La loi de modification des poids*

En reprenant l'équation [9] on pose :

$$G_i^x = G_{i_{quad}}^x + G_{i_{var}}^x \quad [12]$$

avec :

$$G_{i_{quad}}^x = -2f'(a_i^x)(d_i^x - s_i^x) \quad [13]$$

$$G_{i_{var}}^x = -\frac{4}{P} f'(a_i^x)(d_i^x - s_i^x) \left((d_i^x - s_i^x)^2 - MSE \right) \quad [14]$$

La loi de modification des poids devient alors :

$$\Delta w_{ij}^{t+1} = \alpha_{quad} G_{i_{quad}}^x s_j + \alpha_{var} G_{i_{var}}^x s_j + \beta \Delta w_{ij}^t \quad [15]$$

avec α_{quad} le pas d'apprentissage sur l'erreur quadratique, α_{var} le pas d'apprentissage sur la variance de l'erreur quadratique et β l'inertie (terme qui permet d'accélérer la convergence [PLA 86]).

3.3. *Implémentation de la rétropropagation*

Du fait de la nécessité de calculer l'erreur quadratique moyenne et la variance de l'erreur quadratique moyenne après chaque modification des poids, le temps de calcul peut devenir très important. En effet, pour chaque présentation d'un exemple, il faut réaliser P propagations et une rétropropagation. Aussi, afin de réduire les temps de calcul, nous utiliserons l'algorithme suivant qui, comme nous le verrons plus loin, permet d'introduire une contrainte plus forte sur le contrôle de la forme de la distribution des erreurs et d'ajouter de la stabilité au processus d'apprentissage :

– pour toutes les itérations

- pour tous les exemples x :

- calcul de $f_w(x)$,
- calcul de G^x_{quad} et G^x_{var} pour les neurones de sortie,
- calcul de G^x_{quad} et G^x_{var} pour les neurones cachés,
- modifications des poids;

- calcul de l'erreur quadratique moyenne et de la variance de l'erreur quadratique moyenne.

3.4. *Normalisation des pas d'apprentissage*

Dans les différentes études comparatives que nous allons présenter ci-après et pour étudier l'influence du terme ajouté dans la fonction de coût, on définit la variable ν par :

$$\nu = \frac{\alpha_{var} P}{\alpha_{quad}} = \frac{\alpha'_{var}}{\alpha_{quad}} \qquad [16]$$

où P représente la taille de l'ensemble utilisé dans le calcul de la MSE (le nombre d'exemples d'apprentissage).

4. Etude comparative : classification

4.1. *Introduction*

Dans le cadre d'une classification par discrimination, le but recherché est la détermination d'un classifieur qui, à chaque observation x (vecteur de $\mathbb{R}^n$) associe une classe parmi un ensemble fixé de classes. Nous nous intéresserons particulièrement ici au cas où le classifieur est un perceptron multicouche et où l'on utilise q neurones de sortie (où q est le nombre de classes). Le codage habituellement utilisé consiste alors à attribuer le code (d_1, ..., d_q) à la classe h avec $d_i = d$ si $i = h$, $d_i = d'$ si $i \neq h$ et $d > d'$. Les valeurs habituellement utilisées sont $(d, d') = (+1, -1)$ ou $(1, 0)$ ou

encore $(0.9, 0.1)$. La sortie du réseau est un vecteur $\mathbf{y}$ de dimension q. Si $q = 2$, on peut se contenter d'utiliser un réseau à une seule sortie.

Pour les perceptrons multicouches qui utilisent des fonctions d'activation et des poids à valeurs continues, quelques valeurs particulières des sorties du réseau de neurones sont utilisées comme valeurs désirées pour les différentes classes du problème de classification à traiter. Par conséquent, on observe deux types d'erreurs : l'erreur d'apprentissage liée à la minimisation de la fonction de coût que nous appellerons dans ce qui suit l'*erreur d'estimation* et l'*erreur de classification*. L'erreur d'estimation est la différence entre ce que l'on souhaitait obtenir en sortie du réseau (pour un exemple donné) et ce que l'on obtient réellement à la fin de l'apprentissage, tandis que l'erreur de classification existe quand l'erreur d'estimation est supérieure à un seuil préétabli.

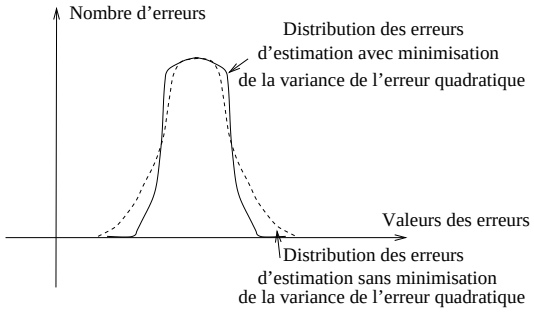

Figure 1. *Influence de la minimisation de la variance de l'erreur quadratique sur la distribution des erreurs d'estimations*

Les bornes de l'erreur de généralisation d'un système composite de classifieurs ont été précédemment établies [HOL 98, SCH 97] et sont basées sur la notion de marge de classification. La taille de la marge dépend à la fois de l'erreur quadratique moyenne obtenue et de la distribution des erreurs d'estimation, soit donc de la variance de l'erreur quadratique obtenue sur chaque classe (voir figure 1). Ainsi, la performance en termes de bonne classification dépend de la forme particulière de la distribution des erreurs d'estimation. Il s'ensuit que le choix d'une fonction de coût appropriée, de manière à contrôler la forme de la distribution des erreurs d'estimation peut être crucial pour obtenir une solution optimale au problème de classification posé.

Nous proposons ici d'utiliser la nouvelle fonction de coût décrite précédemment. En effet, cette méthode approche le problème en prenant en compte un moment d'ordre 4 (la variance de l'erreur quadratique) introduit dans la fonction de coût à minimiser lors de la phase d'apprentissage pour améliorer les résultats en généralisation. Cette approche peut être utilisée lorsque des méthodes [BRE 94, SCH 97] utilisant plusieurs réseaux de neurones afin d'améliorer la marge de classification nécessitent un temps de calcul trop important pour des applications industrielles. Nous étudierons son intérêt dans le cadre de la méthode du Bagging un peu plus loin (voir section 5).

Considérons un problème de classification à deux classes C_1 et C_2 classifiées à l'aide d'un réseau discriminant à une sortie. Le but de la phase d'apprentissage est d'obtenir les sorties suivantes pour le réseau :

– si $x \in C_1$ alors $f_w(x) = d_1$,
– si $x \in C_2$ alors $f_w(x) = d_2$.

avec x le vecteur d'entrée, d_1 et d_2 les sorties désirées respectivement pour un exemple de la classe C_1 et C_2 et $f_w(x)$ la réponse donnée par le réseau de neurones. A la fin de la phase d'apprentissage, ce réseau de neurones a une erreur quadratique moyenne m_1 sur la classe C_1 avec une variance σ_1^2 et une erreur quadratique moyenne m_2 sur la classe C_2 avec une variance σ_2^2 :

$$\sigma_1^2 = \frac{1}{n_1} \sum_{\substack{a=1 \\ a \in C_1}}^{n_1} \left[(d^a - s^a)^2 - \frac{1}{n_1} \sum_{\substack{b=1 \\ b \in C_1}}^{n_1} (d^b - s^b)^2 \right]^2 \quad [17]$$

$$\sigma_2^2 = \frac{1}{n_2} \sum_{\substack{c=1 \\ c \in C_2}}^{n_2} \left[(d^c - s^c)^2 - \frac{1}{n_2} \sum_{\substack{d=1 \\ d \in C_2}}^{n_2} (d^d - s^d)^2 \right]^2 \quad [18]$$

où n_p est le nombre d'exemples de la classe C_p, s^x la valeur de la sortie du réseau pour l'entrée x, d^x la sortie désirée pour l'entrée x.

Comme décrit au paragraphe 3.1, on peut prendre en compte la minimisation de la variance de l'erreur quadratique et, par extension, des variances σ_1^2 et σ_2^2, en ajoutant à l'habituelle fonction de coût, basée uniquement sur l'erreur quadratique, un terme associé à la variance de l'erreur quadratique observée sur chaque classe, le but étant ici d'augmenter la marge de classification.

L'expression de la nouvelle fonction de coût sur un problème de classification à deux classes devient :

$$\begin{aligned} C^x = & \quad (d^x - s^x)^2 \\ & + \tfrac{1}{n_1} \sum_{\substack{a=1 \\ a \in C_1}}^{n_1} \left[(d^a - s^a)^2 - \tfrac{1}{n_1} \sum_{\substack{b=1 \\ b \in C_1}}^{n_1} (d^b - s^b)^2 \right]^2 \\ & + \tfrac{1}{n_2} \sum_{\substack{c=1 \\ c \in C_2}}^{n_2} \left[(d^c - s^c)^2 - \tfrac{1}{n_2} \sum_{\substack{d=1 \\ d \in C_2}}^{n_2} (d^d - s^d)^2 \right]^2 \end{aligned} \quad [19]$$

L'expression du gradient de la fonction de coût C pour le neurone de sortie i et un exemple $x \in C_1$ est :

$$\begin{aligned} \left. \frac{\partial C^x}{\partial w_{ij}} \right|_{x \in C1} = & \quad \frac{\partial}{\partial w_{ij}} \left[(d^x - s^x)^2 \right] + \\ & \frac{\partial}{\partial w_{ij}} \left[\frac{1}{n_1} \sum_{\substack{a=1 \\ a \in C_1}}^{n_1} \left[(d^a - s^a)^2 - \frac{1}{n_1} \sum_{\substack{b=1 \\ b \in C_1}}^{n_1} (d^b - s^b)^2 \right]^2 \right] + \\ & \frac{\partial}{\partial w_{ij}} \left[\frac{1}{n_2} \sum_{\substack{c=1 \\ c \in C_2}}^{n_2} \left[(d^c - s^c)^2 - \frac{1}{n_2} \sum_{\substack{d=1 \\ d \in C_2}}^{n_2} (d^d - s^d)^2 \right]^2 \right] \end{aligned} \quad [20]$$

Le gradient, qui ne dépend pas de la variance observée sur la classe C_2, est par conséquent :

$$\left.\frac{\partial C^x}{\partial w_{ij}}\right|_{k \in C1} = s_j^x(-2f'(a^x)(d^x - s^x) - \left(s_j^x \frac{4}{n_1} f'(a^x)(d^x - s^x))\right) \left[(d^x - s^x)^2 - \frac{1}{n_1} \frac{\sum_{e=1}^{n_1}}{e \in C_1} (d^e - s^e)^2\right] \quad [21]$$

où a^x est la valeur de la somme pondérée à l'entrée du neurones i pour l'exemple x.

Le calcul est identique pour un exemple appartenant à la classe C_2. On retrouve donc la formulation proposée au paragraphe 3.2 et le calcul du gradient attaché aux neurones cachés ne change pas dans sa forme, sauf qu'il comporte deux termes. La règle de modification des poids est :

$$\Delta w_{ij}^{t+1} = \alpha_{quad} G_{quad}^x + \alpha_{var} G_{var}^x + \beta \Delta w_{ij}^t \quad [22]$$

avec α_{quad} le pas d'apprentissage sur l'erreur quadratique, α_{var} le pas d'apprentissage sur la variance de l'erreur quadratique et β l'inertie. On peut noter ici qu'il est possible de différencier le pas d'apprentissage sur la variance de l'erreur quadratique de la classe C_1 par rapport à la classe C_2. Il est aussi possible de ne calculer qu'une seule variance, indépendamment de la classe, et chercher à ne minimiser que cette variance "globale".

4.2. *Conditions expérimentales*

Le problème de classification à deux classes que nous allons utiliser pour notre étude comparative est un problème de détection de visages. La nouvelle fonction de coût est testée sur le "préréseau" de l'application Multrak [BER 98], qui est un système temps réel pour la détection et le suivi automatique de personnes lors d'une visioconférence. Ce système est capable de détecter et de suivre continuellement la position des visages présents dans le champ de vision de la caméra. Le cœur du système est un réseau de neurones modulaire [FER 97] qui détecte les visages avec précision. Le préréseau est utilisé comme filtre, c'est-à-dire que ne seront présentées au réseau de neurones modulaire que les images détectées par le préréseau comme étant des visages. Il doit être beaucoup plus rapide que le réseau modulaire sans dégrader le taux de détection des visages pour l'ensemble du système. Pour des questions de performance temps réel, la vitesse du préréseau est critique et impose de n'utiliser qu'un seul réseau de neurones discriminant.

Nous avons donc entraîné deux préréseaux en tant que détecteurs de visages : un en utilisant la nouvelle fonction de coût décrite précédemment et l'autre uniquement à l'aide de l'erreur quadratique. Chaque réseau de neurones est un perceptron multicouche, utilisant des fonctions d'activation sigmoïdales, possédant 300 entrées (cor-

respondant à une image de 15 x 20 pixels), une couche cachée de huit neurones et une sortie. La base de données utilisée est constituée de trois parties :

– ensemble d'apprentissage : 7 000 visages de face ou tournés et 7 000 non-visages ;

– ensemble de validation : 7 000 visages de face ou tournés et 7 000 non-visages ;

– ensemble de test : 7 000 visages de face ou tournés et 7 000 non-visages.

sachant qu'un non-visage est une image quelconque autre qu'un visage.

Pour comparer les deux fonctions de coût utilisées, différentes expérimentations ont été réalisées. Pour chaque expérimentation, cinquante apprentissages ont été réalisés avec différentes initialisations des poids. Ceci permet d'obtenir, pour chaque condition expérimentale, la moyenne et l'intervalle de confiance de chaque résultat obtenu. Chaque apprentissage a été stoppé quand le coût sur l'ensemble de validation ne diminuait plus depuis 200 itérations. Dans ce cas, l'erreur quadratique moyenne, la variance de l'erreur quadratique sur chaque classe, la marge et le taux de détection sont calculés sur la meilleure configuration des poids (celle pour laquelle l'erreur de généralisation est minimale, voir paragraphe 2.3) pour cet apprentissage sur chaque ensemble (apprentissage, validation, test) et sont détaillés ci-après.

Dans la suite, nous étudions et comparons les deux fonctions de coût. Nous montrons que si le pas d'apprentissage sur le terme de variance ajouté est bien choisi :

1. la variance sur l'ensemble d'apprentissage diminue ;
2. la marge sur l'ensemble d'apprentissage s'accroît ;
3. les performances en classification sur l'ensemble de test augmentent.

Dans les figures qui vont suivre, les résultats provenants de la nouvelle fonction de coût sont étiquetés “VMSE” et ceux découlant uniquement de l'erreur quadratique “MSE”.

4.3. *L'influence du terme sur la variance*

Dans ce paragraphe le paramètre α_{quad} relié à l'erreur quadratique possède une valeur constante de 10^{-2}. L'influence de ν ($\nu = \frac{\alpha'_{var}}{\alpha_{quad}}$, voir paragraphe 3.4) est examinée sur l'intervalle $[10^{-4} : 10^{2}]$ pour évaluer comment le gradient ajouté interagit avec le gradient de l'erreur quadratique. La valeur de l'inertie β a elle été fixée à 0.9. Les comparaisons sont réalisées pour l'erreur quadratique moyenne globale (pour tous les exemples quelles que soient leurs classes d'appartenance) et pour la variance de l'erreur quadratique sur chaque classe. Les résultats pour le préréseau entraîné avec la fonction de coût MSE sont constants, puisque α_{quad} est constant.

La figure 2 montre les résultats obtenus sur l'erreur quadratique moyenne globale avec les deux fonctions de coût sur l'ensemble d'apprentissage. Pour $\nu \in [10^{-4} : 10]$, les deux fonctions de coût fournissent à peu près les mêmes résultats avec le même

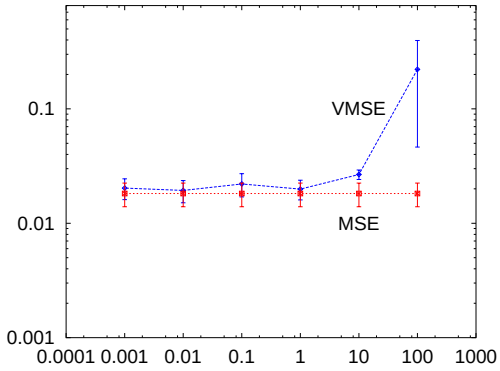

Figure 2. *L'erreur quadratique moyenne globale sur l'ensemble d'apprentissage avec les deux fonctions de coût en fonction de ν*

intervalle de confiance. En revanche, pour $\nu = 100$, l'erreur quadratique moyenne globale augmente beaucoup avec la nouvelle fonction de coût. On voit ici que dans ce cas, α'_{var} est trop grand, comparé à α_{quad}. La minimisation de la variance de l'erreur quadratique au cours de l'apprentissage empêche la minimisation de l'erreur quadratique et le réseau de neurones renvoie toujours la même valeur de sortie, pour laquelle la variance de l'erreur est alors minimale.

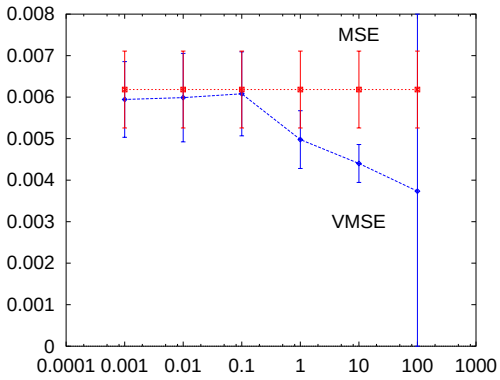

Figure 3. *La variance de l'erreur quadratique pour la première classe (visages) sur l'ensemble d'apprentissage avec les deux fonctions de coût en fonction de ν*

Les figures 3 et 4 montrent les résultats obtenus sur la variance de l'erreur quadratique associée à chaque classe de l'ensemble d'apprentissage. Pour $\nu \in [10^{-4} : 10^{-1}]$, les deux fonctions de coût présentent des performances similaires. Pour $\nu \in [10^{-1} : 10]$ la nouvelle fonction de coût réduit les variances jusqu'à obtenir un gain de 37 % par rapport à la fonction de coût standard et ce avec un intervalle de confiance du même

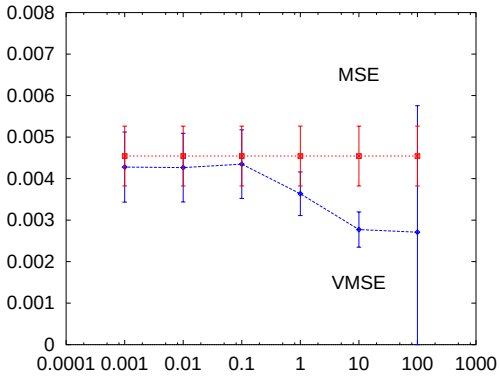

Figure 4. *La variance de l'erreur quadratique pour la deuxième classe (non-visages) sur l'ensemble d'apprentissage avec les deux fonctions de coût en fonction de ν*

ordre. Par contre, pour $\nu = 100$, les variances sont aussi améliorées mais avec un intervalle de confiance insuffisant.

Ces résultats montrent que le terme de variance ajouté interagit avec le terme sur l'erreur quadratique. S'il est du même ordre, il permet d'améliorer les résultats obtenus sur la variance de l'erreur quadratique de chaque classe. Une valeur bien choisie du pas d'apprentissage α'_{var} améliore les résultats sur la variance de l'erreur quadratique associée à chaque classe sans dégrader l'erreur quadratique moyenne globale.

4.4. *Augmentation de la marge*

Une seconde expérimentation montre la relation entre la minimisation de la variance de l'erreur quadratique et la maximisation de la marge pour $\nu = 1$ (α_{quad} = 0.01; α'_{var} = 0.01).

Pour quantifier le pourcentage de la population à l'intérieur de la marge, proche de la frontière de décision, et par conséquent le taux de biens classés à l'extérieur de la marge, nous avons déterminé un seuil θ (voir figure 5) pour différentes valeurs de la marge. Ce seuil est réglé de manière à obtenir le meilleur taux de détection en fonction de la marge sur l'ensemble d'apprentissage, sachant qu'un exemple est considéré bien classé si :

– $f_w(k) \leq \theta$ et k $\in C_1$,
– $f_w(k) \geq \theta$ + Marge et k $\in C_2$.

La figure 6 montre que pour une marge donnée, le taux de détection (sur l'ensemble d'apprentissage) avec la nouvelle fonction de coût est meilleur qu'avec la fonction de coût standard. Par conséquent, pour un même taux de détection, la marge est augmentée.

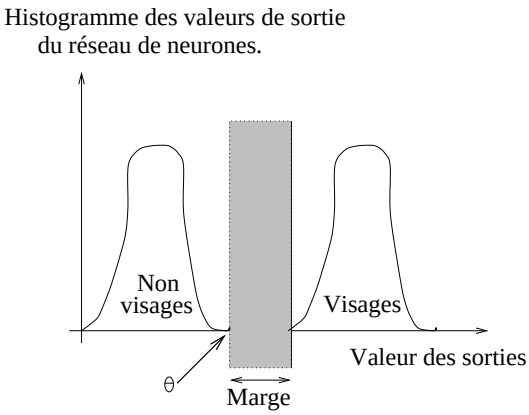

Figure 5. *Explication du taux de biens classés en fonction de la marge*

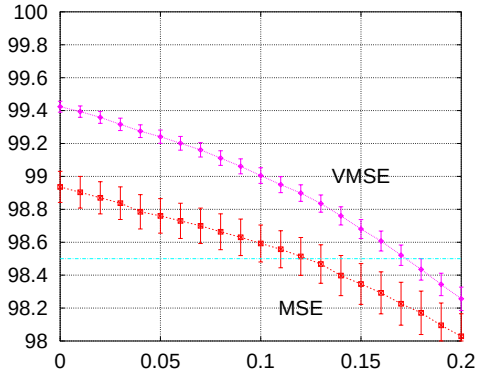

Figure 6. *Le taux de biens classés en fonction de la marge pour l'ensemble d'apprentissage*

Par exemple, pour un taux de détection de 98.5 % (voir figure 6), la nouvelle fonction de coût fournit une marge de 0.17 ± 0.01, alors que la fonction de coût standard ne fournit que 0.12 ± 0.04. L'effet de la minimisation des variances est clairement visible ici. Il est de pousser la distribution des erreurs d'estimation sur les visages vers la droite et celle des non-visages vers la gauche (voir figure 5).

Le deuxième but est donc atteint : la minimisation de la variance de l'erreur quadratique sur chaque classe permet d'améliorer la marge de classification.

4.5. *Une meilleure marge accroît les performances en généralisation*

Ce paragraphe montre que la maximisation de la marge sur l'ensemble d'apprentissage améliore les performances en généralisation (sur l'ensemble de test).

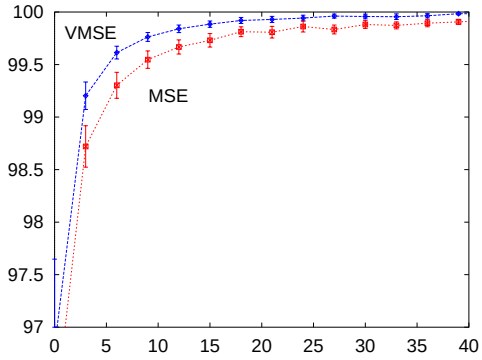

Figure 7. *Le taux de détection pour les visages sur l'ensemble de test en fonction du taux de fausse alarme (non-visages classés comme visages) pour les deux fonctions de coût.*

La figure 7 montre l'effet d'une marge améliorée : la différence entre les deux courbes représente l'amélioration obtenue pour la marge. Avec la fonction de coût MSE et pour un taux de détection de 99.5 %, le taux de fausse alarme est de 8 %, alors qu'avec la nouvelle fonction de coût, ce taux est de seulement 5 %, ce qui représente une amélioration de 37 %.

4.6. *Discussion*

Une nouvelle méthode a été proposée et testée pour améliorer les performances en généralisation sur un problème de classification à l'aide de perceptrons multicouches. Cette méthode est utilisée dans le réseau détecteur de visages appelé *préréseau de l'application Multrak* [BER 98]. L'utilisation de notre nouvelle fonction de coût a permis d'améliorer les performances du préréseau comparé à la fonction de coût basée uniquement sur l'erreur quadratique : le taux de fausse alarme est réduit de 20 % (sur l'ensemble de test A de la base de données du CMU [HUN 94, ROW 95]) pour le même taux de détection.

Cette méthode a été appliquée à un problème de classification à deux classes, mais peut être étendue à des problèmes de classification possédant plus de classes. Elle peut aussi être utilisée dans les méthodes qui utilisent plusieurs réseaux de neurones pour améliorer la généralisation, comme nous allons le voir dans le problème suivant.

5. Etude comparative : *Bagging*

5.1. *Introduction*

Bagging, acronyme de *Bootstrap Aggregating*, est une procédure de stabilisation ([BIS 97] p. 55) qui émule la possession de répliques indépendantes d'un ensemble d'apprentissage. C'est une méthode assez récente [BRE 94] destinée à améliorer les performances des systèmes classifieurs parmi d'autres méthodes existantes, [HOL 98, KOH 95, RAG 93, SCH 97, UTG 90]. Ces différentes méthodes ont montré l'intérêt qu'il y a à générer et à combiner plusieurs classifieurs afin d'augmenter la précision de l'ensemble et sont théoriquement justifiées par le dilemme biais-variance [BRE 94, GEM 92, HAN 90, PER 93, Rav 96, WOL 92].

Supposons que l'on possède un ensemble de données d'apprentissage, de taille N, où chacune de ces données appartient à une classe dont le nombre est K, et une *machine à apprendre* à l'aide de laquelle on veut construire un classifieur compte tenu des données d'apprentissage. Le classifieur obtenu aura alors une précision inhérente à la représentativité des données apprises. En effet, pour un problème de régression (au sens des moindres carrés), l'évaluation d'un estimateur $f_{\mathcal{D}}$ entraîné sur un ensemble de données $\mathcal{D}$ de taille fixée, est faite en prenant l'erreur moyenne sur tous les exemples appartenant à $\mathcal{D}$ du critère des moindres carrés :

$$E_{\mathcal{D}}\left[(y - f_{\mathcal{D}})^2\right] = E_{\mathcal{D}}\left[(E[y|x] - f_{\mathcal{D}})^2\right] + E_{\mathcal{D}}\left[(y - E[y|x])^2\right] \qquad [23]$$

Le second terme ne dépend que du bruit contenu dans les données. La performance d'un estimateur est donnée par :

$$E_{\mathcal{D}}\left[(E[y|x] - f_{\mathcal{D}})^2\right] = (E_{\mathcal{D}}[f_{\mathcal{D}}] - E[y|x])^2 + E_{\mathcal{D}}\left[(f_{\mathcal{D}} - E_{\mathcal{D}}[f_{\mathcal{D}}])^2\right] \qquad [24]$$

Le premier terme est le biais, il ne dépend que de l'ensemble de fonctions Γ, où est choisie $f_{\mathcal{D}}$: si l'ensemble de fonctions Γ est suffisamment grand ou bien choisi, il peut contenir une fonction suffisamment proche de la fonction de régression (la fonction de régression est ici la fonction qui minimise l'erreur quadratique). Néanmoins, en choisissant un ensemble de fonctions de grande taille, on risque d'augmenter la variance : la distance entre la fonction $f_{\mathcal{D}}$ obtenue et la meilleure fonction que l'on peut obtenir compte tenu de l'utilisation d'un ensemble de données de taille fixé $E_{\mathcal{D}}[f_{\mathcal{D}}]$. C'est ce que l'on appelle le *dilemme biais-variance*. L'utilisation d'un ensemble de réseaux de neurones permet de réduire la variance, quand les estimateurs $f_{\mathcal{D}}$ sont identiquement et indépendamment distribués. Pour simplifier les notations, on notera f_i pour $f_{\mathcal{D}}$ et E pour $E_{\mathcal{D}}$:

$$\bar{f} = \frac{1}{N}\sum_{i=1}^{N} f_i \qquad [25]$$

La variance de la moyenne des estimateurs s'exprime par :

$$E\left[\left(\bar{f}-E\left[\bar{f}\right]\right)^2\right]=E\left[\left(\frac{1}{N}\sum_{i=1}^{N}f_i-E\left[\frac{1}{N}\sum_{i=1}^{N}f_i\right]\right)^2\right] \quad [26]$$

$$=E\left[\left(\frac{1}{N}\sum_{i=1}^{N}f_i\right)^2\right]+\left(E\left[\frac{1}{N}\sum_{i=1}^{N}f_i\right]\right)^2-2E\left[\frac{1}{N}\sum_{i=1}^{N}f_iE\left[\frac{1}{N}\sum_{i=1}^{N}f_i\right]\right] \quad [27]$$

$$\Leftrightarrow E\left[\left(\bar{f}-E\left[\bar{f}\right]\right)^2\right]=E\left[\left(\frac{1}{N}\sum_{i=1}^{N}f_i\right)^2\right]-\left(E\left[\frac{1}{N}\sum_{i=1}^{N}f_i\right]\right)^2 \quad [28]$$

Exprimons les deux termes de l'équation :

$$E\left[\left(\frac{1}{N}\sum_{i=1}^{N}f_i\right)^2\right]=\frac{1}{N^2}\sum_{i=1}^{N}E\left[f_i^2\right]+\frac{2}{N^2}\sum_{i<j}E\left[f_if_j\right] \quad [29]$$

et :

$$\left(E\left[\frac{1}{N}\sum_{i=1}^{N}f_i\right]\right)^2=\frac{1}{N^2}\sum_{i=1}^{N}\left(E\left[f_i\right]\right)^2+\frac{2}{N^2}\sum_{i<j}E\left[f_i\right]E\left[f_j\right] \quad [30]$$

On obtient alors :

$$\Leftrightarrow E\left[\left(\bar{f}-E\left[\bar{f}\right]\right)^2\right]=$$

$$\frac{1}{N^2}\sum_{i=1}^{N}\left(E\left[f_i^2\right]-\left(E\left[f_i\right]\right)^2\right)+\frac{2}{N^2}\sum_{i<j}\left(E\left[f_if_j\right]-E\left[f_i\right]E\left[f_j\right]\right) \quad [31]$$

$$=\frac{1}{N^2}\sum_{i=1}^{N}\text{ Var }\left(f_i\right)+\frac{2}{N^2}\sum_{i<j}\left(E\left[f_if_j\right]-E\left[f_i\right]E\left[f_j\right]\right) \quad [32]$$

Si les estimateurs sont tous égaux, $\forall i,\ f_i=f$, on a :

$$E\left[\left(\bar{f}-E\left[\bar{f}\right]\right)^2\right]=E\left[\left(f-E\left[f\right]\right)^2\right]=\text{ Var }\left(f\right) \quad [33]$$

Si les estimateurs sont tous identiquement et indépendamment distribués, on a :

$$\forall i,j \text{ on a Var }\left(f_i\right)=\text{ Var }\left(f_j\right) \quad [34]$$

$$\Rightarrow E\left[\left(\bar{f} - E\left[\bar{f}\right]\right)^2\right] = \frac{1}{N}\ \mathrm{Var}\ (f_i) \qquad [35]$$

On admet que l'hypothèse d'indépendance et de distribution identique des estimateurs est vérifiée quand ils sont entraînés sur des ensembles de données différents mais provenant de la même distribution.

Malheureusement, dans de nombreux problèmes, la taille de l'ensemble d'apprentissage est limitée et le découper pour construire d'autres classifieurs peut diminuer la précision de chacun. On peut essayer alors de "créer" de nouveaux ensembles d'apprentissage à l'aide d'une approximation *bootstrap* et de l'ensemble d'apprentissage que nous qualifierons d'*original*. Pour construire un deuxième ensemble d'apprentissage de la même taille que l'ensemble original, on réalise N tirages indépendants avec remise dans l'original. Ce deuxième ensemble sera donc différent de l'ensemble original puisque chaque exemple qu'il contient peut apparaître plusieurs fois. Notons cependant que le deuxième ensemble provient de la même distribution de probabilité que le premier, la fréquence d'apparition des exemples étant juste différente. A présent, à l'aide du nouvel ensemble, on construit un deuxième classifieur. L'opération complète pouvant être réitérée autant de fois que l'on veut, notons ce nombre T. Finalement, on définit le classifieur global comme étant la combinaison des T classifieurs ainsi construits. Sa réponse étant, pour un exemple présenté simultanément à tous les classifieurs construits, la réponse moyenne de ces derniers.

Notons que la nouvelle fonction de coût VMSE ne minimise pas la variance telle que considérée dans l'équation [35]. Dans cette étude comparative, le but est de rechercher si la méthode VMSE peut permettre, après avoir amélioré le pouvoir de classification d'un unique classifieur, d'améliorer celui d'une combinaison de classifieurs ou si au contraire elle perd de son intérêt. Autrement dit, l'intérêt apporté par l'utilisation des ensembles de réseaux de neurones [BRE 94] peut-il être augmenté en lui conjuguant la nouvelle fonction de coût proposée?

5.2. *Conditions expérimentales*

Le problème qui nous intéresse ici est un problème de classification à deux classes sur un problème lié au crédit [1]. La base de données disponible possède 690 exemples comprenant chacun quinze attributs auxquels s'ajoute l'étiquette d'appartenance à la classe 1 ou 2. Les véritables noms des attributs ont été effacés pour devenir anonymes, on ne peut donc savoir à quoi ils correspondent. Cette base de données est intéressante, parce qu'elle comprend un bon mélange d'attributs continus et discrets. Nous présentons la répartition des attributs et la normalisation que nous en avons effectuée dans le tableau 1. Il y a aussi des données manquantes : 37 exemples (5 %) ont un ou plusieurs attributs manquants (voir tableau 2).

1. On peut trouver ce fichier et d'autres informations sur les conditions de comparaison sur le site : ftp://ftp.ics.uci.edu/pub/machine-learning-databases/credit-screening/

Rappelons que le but ici est de comparer les résultats obtenus par la technique de Bagging mise au point par Breiman [BIS 97, BRE 94] et plus particulièrement les résultats affichés dans [QUI 98] avec la nouvelle fonction de coût et la fonction de coût MSE. La répartition des exemples suivant les deux classes est : 307 exemples appartiennent à la classe 1, soit 44.5 % et 383 à la classe 2, soit 55.5 %.

Nous avons utilisé la même procédure d'apprentissage que celle énoncée dans [QUI 98], qui est :

– création de duplications de la base de données de départ par tirage aléatoire uniforme avec remise (la duplication a le même nombre d'éléments que la base de départ) ;

– pour chaque nouvelle base ainsi créée (que l'on appelle D_i) :

- on subdivise la base en dix blocs de taille égale, que l'on appelle B_j ;

- on réalise dix apprentissages en utilisant neuf des blocs comme ensemble d'apprentissage et le dixième comme ensemble de "test" ; on appelle RB_j le résultat (par exemple la MSE) obtenu en utilisant le bloc j comme ensemble de test ;

- le résultat final pour cette base est le résultat moyen obtenu sur les dix apprentissages, que l'on appelle RD_i :

$$RD_i = \frac{1}{10} \sum_{j=1}^{10} RB_j \qquad [36]$$

– le résultat final obtenu R_f est fonction du nombre (N) de répliques de la base d'origine utilisées :

$$R_f = \frac{1}{N} \sum_{i=1}^{N} RD_i \qquad [37]$$

Chaque réseau de neurones possède quinze neurones en entrée, six cachés et un en sortie. Nous précisons ici que le critère d'arrêt des différents apprentissages est basé uniquement sur l'erreur quadratique (tout comme les auteurs de [QUI 98]). Le terme ajouté dans la nouvelle fonction de coût est donc utilisé comme un facteur de régularisation, ce qui ne change en rien la loi de modification des poids énoncé au paragraphe 3.2. Son influence a été mesurée pour $\nu = 1$ et $\nu = 10$ (α_{quad} = 0.01, β = 0.9).

Chaque apprentissage a été stoppé quand le coût sur l'ensemble de test ne diminuait plus depuis 200 itérations. Dans ce cas, l'erreur quadratique moyenne, la variance de l'erreur quadratique globale et le pourcentage d'erreur sont calculés sur la meilleure configuration des poids pour cet apprentissage sur l'ensemble de test et sont détaillés ci-après.

Les valeurs manquantes ont été remplacées, arbitrairement par la valeur -1.0, donc au mimimum de la distribution de la variable considérée. D'autres méthodes consistent à remplacer les valeurs manquantes d'une variable par la moyenne globale de cette variable ou encore dans le cas de variable modale par la modalité majoritaire. Le lecteur

Attribut	Valeur de départ	Normalisation effectuée
A1	a, b	0.5, 1.0
A2	continue	[0.0,1.0]
A3	continue	[0.0,1.0]
A4	u, y, l, t	0.25, 0.50, 0.75, 1.0
A5	g, p, gg	0.3, 0.6, 0.9
A6	c, d, cc, i, j, k, m	0.07, 0.14, 0.28, 0.35, 0.42, 0.49, 0.56
	v, q, w, x, e, aa, ff	0.63, 0.70, 0.84, 0.91, 0.98
A7	v, h, bb, j, n, z, dd	0.11, 0.22, 0.33, 0.44, 0.55, 0.66, 0.77
	ff, o	0.88, 0.99
A8	continue	[0.0,1.0]
A9	t, f	0.3, 0.7
A10	t, f	0.3, 0.7
A11	continue	[0.0,1.0]
A12	t, f	0.3, 0.7
A13	g, p, s	0.2, 0.5, 0.8
A14	continue	[0.0,1.0]
A15	continue	[0.0,1.0]
A16	attribut de classe	0.1, 0.9
Ax	donnée manquante	-1.0

Tableau 1. *La composition du fichier de données et la normalisation associée*

Attribut	Nombre d'attributs manquants
A1	12
A2	12
A4	6
A5	6
A6	9
A7	9
A14	13

Tableau 2. *Répartition des attributs manquants*

pourra trouver dans [FES 99, LAK 96, VAM 96] des informations complémentaires sur l'apprentissage avec des valeurs manquantes et sur les techniques de remplacement des valeurs manquantes.

5.3. *Résultats*

Les résultats présentés dans les figures 8, 9, 10, 11 sont étiquetés MSE pour la fonction de coût n'utilisant que l'erreur quadratique, VMSE 1 et VMSE 10 respectivement pour $\nu = 1$ et $\nu = 10$.

On précise que pour $\nu = 0.1$ (ou inférieur), les résultats avec la méthode VMSE sont identiques à ceux obtenus avec la méthode MSE. Les valeurs de ν ont été choisies en fonction de l'expérience acquise sur le problème précédent de détection de visages.

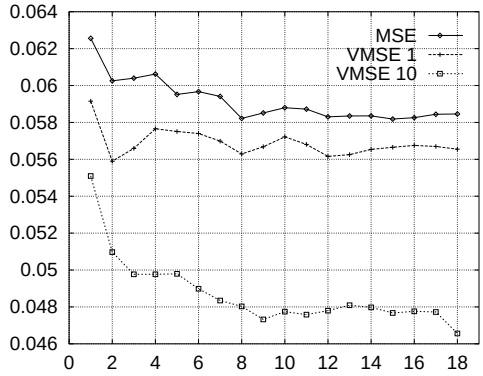

Figure 8. *Résultats obtenus sur l'erreur quadratique moyenne globale en fonction du nombre de répliques*

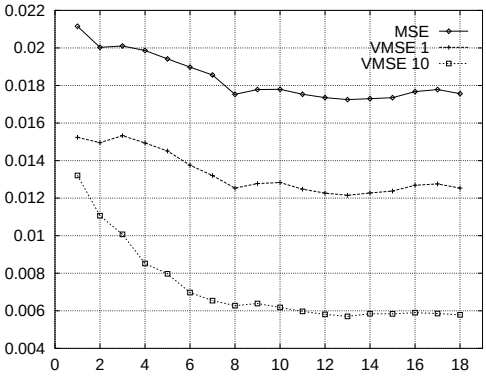

Figure 9. *Résultats obtenus sur la variance de l'erreur quadratique en fonction du nombre de répliques pour la classe 1*

Plus précisément, on trouve figure 8 le résultat obtenu sur l'erreur quadratique moyenne globale, figures 9, 10 le résultat respectivement sur la variance de l'erreur quadratique pour la classe 1, 2 et figure 11 le pourcentage de mal classé en fonction du nombre de répliques utilisées. Sur cette dernière figure, on indique le meilleur résultat obtenu par Quinlan [QUI 98] (ce dont on ne dispose pas pour les trois premiers résultats affichés). Dans le cas de l'utilisation de la fonction de coût MSE ($\nu = 0$), nous n'avons pas exactement retrouvé les résultats de Quinlan, ce qui peut provenir d'une erreur de notre part quant à la compréhension des conditions expérimentales.

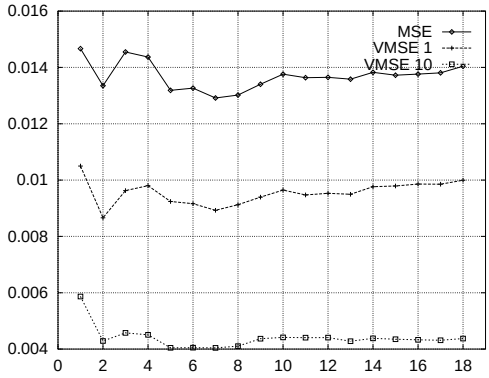

Figure 10. *Résultats obtenus sur la variance de l'erreur quadratique en fonction du nombre de répliques pour la classe 2*

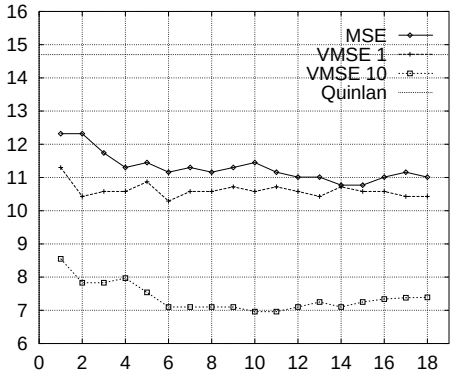

Figure 11. *Résultats obtenus sur le pourcentage de mal classé en fonction du nombre de répliques*

Les résultats trouvés ici avec la méthode MSE sont néanmoins meilleurs et constituent donc de bonnes valeurs de comparaison.

On peut aussi noter qu'un des résultats de Breiman [BRE 94] montrant de façon expérimentale que le nombre de répliques nécessaire mais suffisant est de l'ordre de 10, après quoi les performances ne sont que très peu améliorées, est retrouvé. Ceci est plus particulièrement visible sur les figures 8, 9.

5.4. *Discussion*

On s'aperçoit que les résultats obtenus avec la méthode VMSE sont meilleurs que ceux obtenus avec la fonction MSE tant sur l'erreur quadratique moyenne globale que

sur sa variance et sur le pourcentage de mal classé. On peut conclure que la nouvelle fonction de coût se conjugue très bien avec la technique du *Bagging*. Le contrôle de la forme de la distribution des erreurs, réalisé grâce à une minimisation de la variance de l'erreur quadratique, permet une meilleure généralisation. Cette amélioration déjà constatée sur la première étude comparative n'utilisant qu'un seul réseau de neurones est conservée au fur et à mesure que le nombre de réseaux est augmenté. On voit bien sur les différentes figures de résultats que l'amélioration apportée en fonction du nombre de répliques utilisées est à peu près identique pour les trois courbes. L'amélioration des performances obtenues avec une seule base est conservée lorsque le nombre de répliques utilisées augmente.

6. Conclusion

On a présenté dans cette section une nouvelle méthode destinée à améliorer les performances en généralisation des perceptrons multicouches utilisés en tant que réseaux discriminants et approximateurs de fonctions. On a clairement montré comment modifier le critère d'apprentissage afin de contrôler la distribution des erreurs au cours de l'apprentissage. Cette méthode permet de minimiser à la fois les erreurs de classification et celles d'estimation par une minimisation de la variance de l'erreur quadratique.

Cette méthode utilisée en tant que fonction de régularisation est très facilement implémentable, même dans les cas où il y a plus d'un neurone de sortie. En effet, il suffit de calculer la MSE sur chaque neurone de sortie du réseau. Par exemple, pour un problème à cinq classes où l'on utilise cinq neurones de sortie, il faut calculer cinq MSE. Ces MSE ayant été calculées, le calcul des cinq gradients G_{var} est alors immédiat, conformément à l'équation [14]. Les modifications à apporter dans un programme d'apprentissage où la méthode MSE était utilisée sont alors minimes.

Reste à savoir comment choisir de façon simple et efficace les paramètres α_{quad}, α_{var} et β. Pour β, nous renvoyons le lecteur à [PLA 86] du fait que nous avons remarqué que de nombreux utilisateurs de réseaux de neurones introduisent ce terme dans la règle de modification des poids alors que d'autres n'y trouvent pas un grand intérêt, expérimentalement parlant. Pour ce qui est de α_{quad} et de α_{var}, le critère de choix, comme nous l'avons montré, est imposé par leur ratio illustré dans cette section par la variable ν.

Deux études comparatives ont été présentées sur deux problèmes différents. Les résultats expérimentaux obtenus améliorent sensiblement l'état de l'art :

– pour le problème de détection de visages : baisse de 37 % du taux de fausse alarme (pour un taux de détection de 99.5 %) pour $\nu = 1$;

– pour le problème de crédit : amélioration du taux de bien classé de 30.4 % pour $\nu = 10$.

L'idée de contrôler la forme de la distribution des erreurs au cours de l'apprentissage afin d'obtenir une meilleure généralisation se trouve validée de manière ex-

périmentale. Enfin, pour obtenir la réponse à la question de savoir si cette nouvelle fonction de coût permet d'obtenir de bons résultats sur des problèmes autres que ceux de classification par discrimination, le lecteur est invité à consulter [LEM 99].

7. Bibliographie

[BER 96] BERTSEKAS D. P., TSITSIKLIS J. N., *Neuro-Dynamic Programming*, MA: Athena Scientific, Belmont, 1996, ISBN 1-886529-10-8.

[BER 98] BERNIER O., COLLOBERT M., FÉRAUD R., LEMAIRE V., VIALLET J., COLLOBERT D., « MULTRAK : a system for Automatic Multiperson Localization and tracking in real-Time », *International Conference on Image Processing*, 1998.

[BIS 97] BISHOP C. M., Ed., « *Neural Networks and Machine Learning* », vol. 168 de *F*, Chapitre I-11, NATO ASI Series, 1997.

[BOT 91] BOTTOU L., « Une approche théorique de l'apprentissage connectioniste; applications à la reconnaissance de la parole. », PhD thesis, Université de Paris 11, Orsay, France, 1991.

[BOU 92] BOUCHERON S., *Théorie de l'apprentissage*, Hermès, Paris, 1992.

[BRE 94] BREIMAN L., « Bagging Predictors », rapport n° TR-421, 1994, University of California, Berkley.

[FER 97] FERAUD R., BERNIER O., « Ensemble and Modular approaches for Face Detection: a Comparison », *Neural Information Processing System*, vol. 10, december 1997, p. 472-478.

[FES 99] FESSANT F., MIDENET S., « A knowledge-based parser : Neural Network bases approaches .Development of a neural network based imputation system for travel diqry data », rapport, 1999, Test : Technologies for European Surveys of Travel Behaviour, Deliverable D5-B.

[GEM 92] GEMAN S., BIENENSTOCK E., DOURSAT R., « Neural Networks and the bias-variance dilemma », *Neural Computation*, vol. 4, 1992, p. 1-58.

[HAN 90] HANSEN L. K., SALAMON P., « Neural Networks ensembles », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, 1990, p. 993-1001.

[HOL 98] HOLGER S., BENGIO Y., « Training method for adaptative boosting of neural networks », *Neural Information Processing System*, 1998.

[HUN 94] HUNKE H. M., « Locating and Tracking of Human Faces with Neural Network », rapport n° CS-94-155, 1994, CMU.

[KOH 95] KOHAVI R., JOHN G. H., « Automatic parameter selection by minimizing estimated error. », KAUFFMAN M., Ed., *International Conference on Machine Learning*, vol. 12, San Francisco, 1995, p. 304-311.

[LAK 96] LAKSHMINARAYAN K., HARP S., GOLDMAN R., SAMAD T., « Imputation of missing data using machine learning techniques », *KDD*, 1996.

[LEC 87] LE CUN Y., « Modèles connexionnistes de l'Apprentissage », PhD thesis, Université Paris 6, 1987.

[LEM 99] LEMAIRE V., « Une nouvelle fonction de coût régularisante dans les réseaux de neurones artificiels : Application à l'estimation des temps de blocage dans un nœud ATM. », PhD thesis, Université de Paris 6 , Pierre et Marie Curie, 1999, http://www.robo.jussieu.fr/lemaire/.

[PER 93] PERRONE M. P., « Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Measure Optimization », PhD thesis, Brown University, Institute - Brain and Neural Systems, 1993.

[PLA 86] PLAUT D., NOWLAN S., HINTON G., « Experiments on learning by backpropagation. », rapport n° CMU-CS-86-126, 1986, Department of computer science, Carnegie Mellon University.

[QUI 98] QUINLAN J. R., « Bagging, Boosting and C4.5 », rapport, 1998, University of Sydney.

[RAG 93] RAGAVAN H., RENDELL L., « Lokkahead feature construction for learning hard concepts. », KAUFFMAN M., Ed., *International Conference on Machine Learning*, vol. 10, San Francisco, 1993, p. 252-259.

[Rav 96] RAVIV, Y., INTRATOR, N., « Bootstrapping with noise: An effective Regularization Technique », *Connection Science*, vol. 8, 1996, p. 355-372.

[ROW 95] ROWLEY H., BALUJA S., KANADE T., « Human Face Detection in Visual Scenes », rapport, 1995, School of Computer Science Carnegie Mellon University, Pittsburgh, PA 15213, CMU-CS-95-158R.

[RUM 86] RUMELHART D. E., HINTON G. E., WILLIAMS, R. J., « Learning internal representations by error propagation », *Parallel Distribued Processing: Explorations in the Microstructures of Cognition*, vol. 1, 1986, p. 318-362.

[SCH 97] SCHAPIRE R. E., FREUND Y., BARTLETT P., LEE W., « Boosting the margin : A new explanation for the effectiveness of voting methods », *Machines That Learn*, 1997, http://www.research.att.com/ schapire/.

[TIB 96] TIBSHIRANI R., « A comparison of some error estimates for neural network models », *Neural Computation*, vol. 8, 1996, p. 152-163.

[UTG 90] UTGOGG P. E., BRODLEY C., « An incremental method for finding multivariate splits for decision trees. », KAUFFMAN M., Ed., *International Conference on Machine Learning*, vol. 7, San Francisco, 1990, p. 58-65.

[VAM 96] VAMPLEW P., CLARK D., ADAMS A., MUENCH J., « Techniques for Dealing with Missing Values in Feedforward Networks », *ACNN*, 1996, p. 250-254.

[VAP 71] VAPNIK V. N., CHERVONENKIS A. Y., « On the uniform Convergence of relative Frequencies of Events to their Probabilities », *Theory of Probability and its applications*, vol. 16, 1971, p. 264-271.

[VAP 82] VAPNIK V., *Estimation of Dependences Based on Empirical Data*, Springer Verlag New York Heidelberg Berlin, 1982.

[VAP 95] VAPNIK V., *The Nature of Statistical Learning Theory*, Springer Verlag New York Heidelberg Berlin, 1995.

[WHI 89] WHITE H., HORNIK K., « Mutilayer feedforward networks are universal approximators », *Neural Network*, vol. 2, 1989, p. 359-366.

[WOL 92] WOLPERT D. H., « Stacked Generalization », *Neural Networks*, vol. 5, 1992, p. 241-259.