La Revue de MODULAD

MLP : PROGRAMME DE RESEAU DE NEURONES MULTICOUCHES

Joseph PRORIOL

Laboratoire de Physique Corpusculaire
de Clermont-Ferrand
IN2P3 - CNRS
Université Blaise Pascal
63177 AUBIERE CEDEX FRANCE

et

I.U.T.
Avenue A. Briand
03107 MONTLUÇON CEDEX FRANCE

(E-MAIL PRORIOL@FRCPN11 (BITNET))

RESUME

On présente un programme de réseaux de neurones multicouches écrit en FORTRAN.

Ce programme sert à la classification d'individus en diverses classes et accomplit un travail analogue à celui de l'analyse discriminante. Il est particulièrement adapté à la prise en compte de relations non linéraires.

Mots clés : Réseau de neurones, perceptron, classification.

1) INTRODUCTION

Ce programme a été écrit pour disposer d'une méthode supplémentaire pour discriminer entre plusieurs classes.

Il a été écrit en FORTRAN car l'environnement de travail pour lequel il a été conçu est un environnement FORTRAN. Le prototype a été écrit sans "open" car de nombreux fichiers de données sont sur des cartouches et les "open" gèrent difficilement ce type de données. Mais le programme écrit pour MODULAD contient des "open". Cependant pour faire fonctionner le programme, on n'a pas à toucher au code, la communication se fait par un fichier de paramètres qui est facile à modifier avec un traitement de texte.

L'apprentissage des réseaux de neurones est long. Le choix des paramètres est délicat.

Il n'existe pas de règle pour choisir les variables d'entrée. Cependant, les méthodes utilisées pour la discrimination linéaire restent valables et donnent de bons résultats.

Il n'existe pas de règle pour choisir le nombre de couches du réseau. On procède habituellement par essais. Il y a quelques tentatives pour essayer d'optimiser ce choix (Kayama et al (1990)).

On présente d'abord le réseau de neurones et les techniques utilisées. Un schéma donne ensuite la structure du programme MLP. On présente la structure des fichiers d'entrée pour le test et l'apprentissage.

2) LE RESEAU DE NEURONES (Rumelhart (1988), Hertz (1991))

On utilise un réseau de neurones multicouches, avec la rétropropagation de l'erreur pour le calcul des poids.

2.1 Les Relations.

On considère un perceptron à 2 couches. Sur la figure 1 on a 3 neurones d'entrée et 2 neurones de sortie. Les valeurs d'entrée dans le réseau de neurones sont ξ_i^μ. On suppose l'existence de poids W_{ji} entre les entrées et les sorties.

Les sorties O_i^μ sont calculées en utilisant les relations de la propagation vers l'avant. On utilise la fonction sigmoïde

$$g(x) = \left(1 + e^{-x/T}\right)^{-1} \qquad (1)$$

où T est une température.

A partir des entrées, on a

$$h_i^\mu = \sum_j W_{ij}\ \xi_j^\mu \tag{2}$$

Et les sorties sont calculées par la relation

$$O_i^\mu = g\ (h_i^\mu) \tag{3}$$

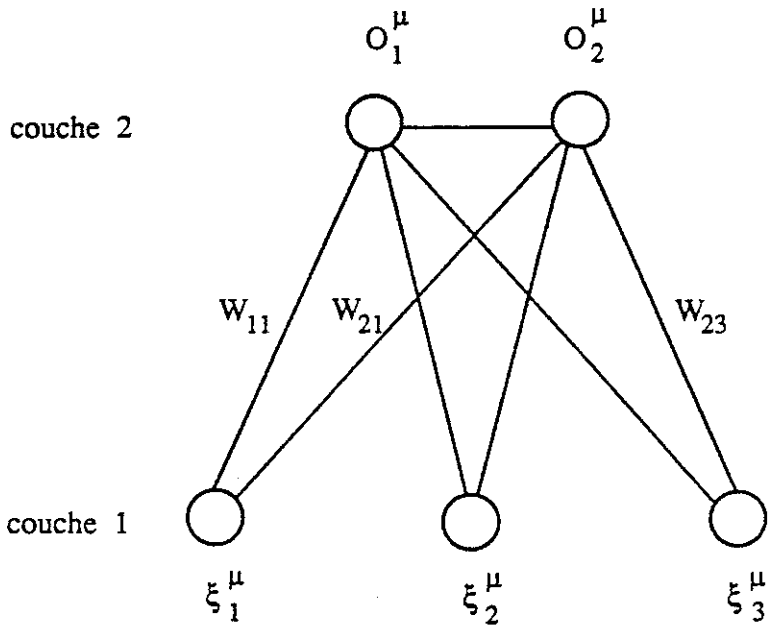


Figure 1

Si on connaît la classe de la donnée μ, on s'attend à une sortie ζ_i^μ. Cette valeur est en général différente de O_i^μ. On utilise cette différence pour calculer une correction sur les W_{ij} par la rétropropagation de l'erreur.

On calcule une fonction coût

$$E = \frac{1}{2} \sum_{\mu,i} C_i\ (\zeta_i^\mu - O_i^\mu)^2 \tag{4}$$

où C_i est un coefficient qui permet de varier le coût pour chaque classe comme dans CART (Breiman et al (1984)).

Si on choisit de réactualiser W_{ij} par la relation suivante où ε est une constante inférieure à 1 :

$$\Delta W_{ij} = -\varepsilon \frac{\partial E}{\partial W_{ij}} \tag{5}$$

la valeur E décroît au fur et à mesure de la présentation des événements au réseau de neurones :

Alors

$$\Delta W_{ij} = \varepsilon \sum_{\mu} \delta_i^{\mu} \, g\left(h_j^{\mu}\right) \tag{6}$$

Pour la couche de sortie la valeur de δ_i^{μ} est

$$\delta_i^{\mu} = C_i \, g'\left(h_i^{\mu}\right) \left(\zeta_i^{\mu} - O_i^{\mu}\right) \tag{7}$$

Si on introduit des couches cachées entre la couche d'entrée et la couche de sortie la valeur de δ_i^{μ} est

$$\delta_j^{\mu} = g'\left(h_i^{\mu}\right) \sum_{i} W_{ij} \, \delta_i^{\mu} \tag{8}$$

où le δ_i^{μ} est celui de la couche suivante : on utilise la rétropropagation de l'erreur pour calculer ΔW_{ij}.

Mais l'actualisation de W_{ij} est faite à l'aide de la relation

$$\Delta W_{ij}(t+1) = \eta \, \Delta W_{ij}(t) + \varepsilon \sum_{\mu} \delta_i^{\mu} \, g\left(h_i^{\mu}\right) \tag{9}$$

le paramètre η est choisi de l'ordre de 0.5 et le paramètre ε dans la gamme de valeurs comprises entre 0.5 et 0.01.

Le réseau sera construit avec un certain nombre de couches, à chaque neurone de sortie on affectera une classe : donc le nombre de neurones de sortie est égal au nombre de classes. Le nombre de neurones d'entrée est égal au nombre de variables du problème.

2.2 L'apprentissage.

Au début, on initialise les W_{ij} par de petites valeurs aléatoires.

On présente un individu au réseau, on fait le calcul en avant avec les relations (1), (2), (3), que l'on applique aux différentes couches. On arrive à la couche de sortie.

On calcule les diverses valeurs δ_i^{μ} à partir des relations (7) et (8) et on peut calculer les ΔW_{ij} à partir de la relation (9) en descendant les différentes couches (rétropropagation de l'erreur). On réactualise les valeurs de W_{ij}.

On recommence ainsi avec un autre individu. Si le nombre d'individus du lot d'apprentissage est trop faible, on recommence jusqu'à ce que le coût E soit petit.

La phase d'apprentissage est lente et longue. Il faut faire 10^6 présentations au moins pour que E converge vers une valeur minimum. Si l'apprentissage converge mal, il faut diminuer ε et augmenter le nombre de présentations.

Quand l'apprentissage est terminé, les poids W_{ij} sont fixés, on les stocke dans un fichier.

2.3 Test

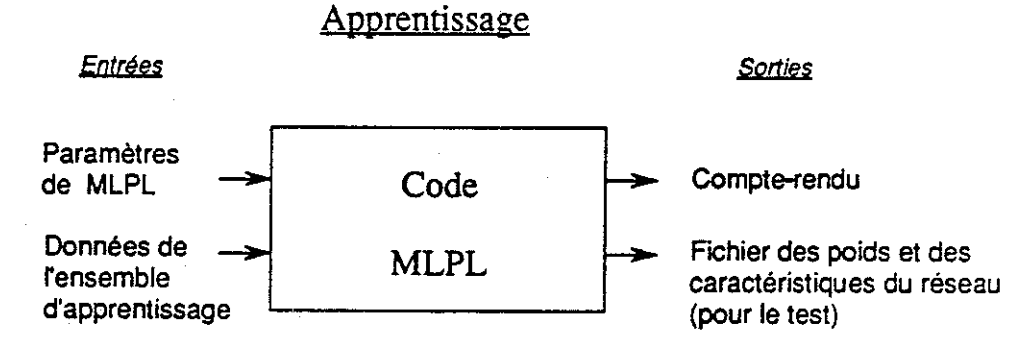
On présente alors au réseau un lot d'individus-test différent du lot d'apprentissage.

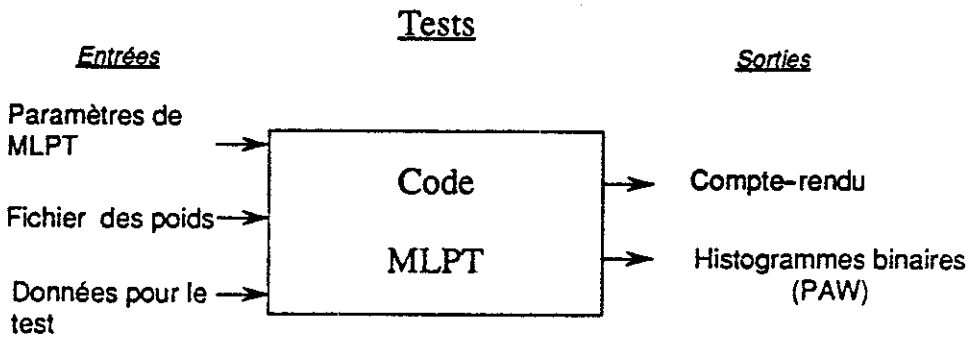
A partir des variables d'entrées ξ_i^μ, on calcule les sorties O_j^μ à l'aide des relations (1), (2), (3). Les valeurs O_j^μ servent à la classification de l'individu.

3) LE PROGRAMME MLP

Nous avons écrit 2 programmes pour MLP : on a séparé la phase d'apprentissage de la phase du test.

L'organisation des programmes est donnée ci-dessous.





A la sortie de l'apprentissage, un fichier conserve les poids et les caractéristiques du réseau. Il est réutilisé pour tous les tests.

A la sortie des tests, on conserve des histogrammes binaires qui peuvent être retraités par un programme spécifique PAW (Brun (1989)).

4) LE PROGRAMME D'APPRENTISSAGE MLPL

Le programme d'apprentissage MLPL fonctionne avec un fichier paramètres et un fichier de données sur les divers individus.

Le fichier paramètres a été écrit pour pouvoir être facilement modifié par un éditeur classique.

Les valeurs du fichier de données pour les individus du lot d'apprentissage ne sont pas nécessairement normalisées ; un sous-programme les ramène à l'intervalle $[-1, +1]$.

Le fichier de sortie contient les résultats de l'apprentissage : les différentes valeurs des poids W_{ij} et les caractéristiques du réseau.

Le fichier résultat contient un compte-rendu de l'apprentissage. On imprime la valeur du coût et pour chaque classe, la moyenne des valeurs du neurone de sortie pour les individus de la même classe et pour les individus des classes différentes, ainsi que la moyenne de ces deux valeurs. Pour un apprentissage parfait, le coût doit être faible, et les valeurs voisines de 1, 0 et 0.5. On a ainsi un indice de la valeur de l'apprentissage. On regarde également la convergence des divers paramètres.

Si l'apprentissage se fait mal, on diminue la valeur de ϵ et on augmente le nombre de présentations.

5) LE PROGRAMME TEST MLPT

Le programme test fonctionne de la même manière que le programme d'apprentissage.

Le fichier résultat contient d'abord une matrice de confusion pour les individus du lot test.

La sortie suivante est un balayage de l'histogramme de chaque neurone de sortie pour les événements de même classe ou de classe différente. La valeur de la sortie est comprise entre 0 et 1.

On fait varier une valeur de coupure de 0 à 1 (cut) et on regarde le nombre d'individus de la même classe ou de classe différente du neurone dont la valeur de sortie est au-dessus de la coupure.

6) REMERCIEMENTS

J'ai été aidé au début de mon apprentissage aux réseaux de neurones par G. Barrand du LAL d'Orsay, qui m'a donné quelques procédures en C.
Monsieur Lechevallier de l'INRA m'a aidé à présenter mon programme.

Références

Breiman L., Friedman J.H., Ohlsen R.A., Stone C.J. (1984)
Classification and Regression trees
Wadworth

Brun R., Conet O., Vandoni C., Zanarini P. (1989)
PAW (Physics Analysis Workstation)
CERN Computer Center, Program Library

Chabanon C., Dubuisson B. (1991)
Methodes non probabilistes in
Analyses dicriminantes sur variables continues,
éditeur Celeux G.
INRIA Coll. Didactique N°7

Hertz J., Krogh A., Palmer R.G. (1991)
Introduction to the theory of neural computation
Addison-Wesley

Kayama M., Abe S., Takenaga H., Morouka Y. (1990)
Troisièmes Journées Internationales :
Les réseaux neuro-mimétiques
et leurs applications - p. 363
Neuro-Nimes 90.