Calcul exact des dérivées d'un réseau de neurones multi-couches à propagation avant

Serge Iovleff

26 février 2001

Résumé

Dans cet article nous montrons comment calculer les dérivées premières et secondes d'un réseau de neurones à propagation avant général et les dérivées de tous les ordres d'un perceptron multi-couches. Notre approche est simple et reprend une idée de Vapnick [5]. Elle est basée sur les multiplicateurs de Lagrange et permet d'intégrer de manière évidente des contraintes supplémentaires sur les paramètres. Nous montrons en particulier que dans le cas du perceptron la matrice des dérivées secondes est bloc diagonale ce qui permet de réduire de manière très significative le nombre de données à stocker pour les méthodes d'estimation du second ordre.

Key Words : Réseaux de neurones ; Perceptron ; Optimisation.

1 Introduction

Les algorithmes standard d'apprentissage des perceptrons multi-couches utilisent l'algorithme de retro-propagation pour évaluer les dérivées premières de la fonction objectif par rapport aux paramètres du réseau (poids et biais). Le calcul de ces dérivées premières permet de mettre en oeuvre un algorithme de gradient (méthode au premier ordre) dont les défauts sont bien connus : lenteur de convergence et piège par des minima locaux ou par les plateaux de la fonction objectif.

Pour accélérer cette méthode il est courant d'utiliser des méthodes du second ordre : essentiellement les méthodes basées sur le gradient conjugué, ou de quasi-newton (BFGS) et ses variantes [4] qui approximent la matrice hessienne au cours des itérations. Rappelons que le calcul exact des dérivées secondes d'un perceptron multi-couches a été obtenu par Bishop [1] et que celles ci peuvent être évaluées par un algorithme de retro-propagation. Toutefois l'utilisation effective des dérivées secondes ne semblent pas être très répandue dans la pratique.

A l'heure actuelle, il n'existe pas à notre connaissance de formules explicites pour les dérivées d'ordre supérieure. Le but de cet article est dans un premier temps de proposer une méthode de calcul des dérivées premières et secondes d'un réseau de neurones à propagation avant général. Dans un deuxième temps, nous développons les formules pour les principaux réseaux de neurones à propagation avant utilisés : le perceptron multi-couches (pour lequel nous sommes capable d'obtenir les dérivées de tous les ordres) et les réseaux à bases radiales.

Notre approche est basée sur une idée de Vapnick [5] qui utilise les multiplicateurs de Lagrange. D'un point de vue pratique, la mise en oeuvre du calcul de ces dérivées peut être effectuée par un algorithme de rétro-propagation.

Ce document est organisé de la manière suivante. Dans un premier temps nous introduisons les notations que nous utilisons, décrivons le fonctionnement d'un réseau de neurones à propagation avant et donnons deux exemples de réseaux de

neurones. Nous présentons ensuite plusieurs exemples standard d'application de ces réseaux de neurones et nous décrivons le problème d'optimisation que l'on cherche à résoudre. Nous calculons de manière effective les dérivées première et seconde d'un réseau de neurones à propagation avant puis les dérivées de tous les ordres du perceptron multi-couches.

2 Notations, Fonctionnement et Exemples

Rappelons tout d'abord, le vocabulaire. Les réseaux de neurones sont caractérisés d'un point de vue fonctionnel par trois aspects :

1. Une *architecture* qui décrit les liaisons entre les neurones à l'aide d'un graphe orienté (les noeuds du graphe représente les neurones et les arcs les sens de communication entre neurones).
2. Une méthode de *propagation* qui décrit comment les neurones communiquent entre eux le long des arcs.
3. Une *activation* qui représente le traitement réalisé de manière individuelle par les neurones.

Les Réseaux de neurones à propagation avant sont caractérisés par une disposition en couche des neurones qui ne permet pas aux neurones d'une couche de communiquer avec les neurones des couches précédentes.

Un réseau de neurones est déterminé par un ensemble de paramètres inconnus qu'il faut déterminer (étape d'apprentissage) à l'aide d'un échantillon (les exemples). Ces paramètres doivent minimiser une fonction de coût (ou fonction objectif, ou fonction de perte, ou contraste).

2.1 Notations et Fonctionnement

Pour les indices, nous utilisons les conventions suivantes :

- Nous utilisons l'indice k pour indiquer le numéro de la couche du réseau, et nous supposerons que nous avons $m+1$ couches : lorsque $k=0$ il s'agit de la couche d'entrée, lorsque $k=1$ il s'agit de la première couche cachée, ..., lorsque $k=m$ il s'agit de la couche de sortie.
- Nous notons $s_0, s_1, \ldots, s_m$ le nombre d'unités par couche et nous supposons que $s_k > 0$ pour tous k.
- Nous utilisons l'indice j pour numéroter les neurones (ou unités) à l'intérieur d'une couche.
- Nous utilisons l'indice i pour noter le numéro de l'observation (exemple), et nous supposerons que nous avons n observations.
- Nous notons $t_1, \ldots, t_m$ le nombre de paramètres par couche.
- nous utilisons l'indice l pour numéroter les paramètres à l'intérieur d'une couche. l pourra varier de 1 à t_k.

Pour les fonctions d'activations et de propagation, nous utilisons les conventions suivantes :

- Nous notons h_j^k la fonction d'*activation* du j-ème neurones de la couche k ($k>0$) et nous supposons que h_j^k est de classe $\mathcal{C}^p$ où p est l'ordre de dérivation que nous souhaitons calculer. Les fonctions d'activations ne dépendent pas de paramètres.
  Etant donné un vecteur $X \in \mathbb{R}^{s_k}$, nous noterons $H^k(X)$ l'ensemble des fonctions d'activation de la couche k appliquées à chacune des coordonnées du vecteur $X = (x_1, \ldots, x_{s_k})$. On aura donc

$$H^k(X) = \left(h_1^k(x_1), \ldots, h_{s_k}^k(x_{s_k})\right)$$

– Nous notons $h_j^{k(p)}$ la dérivée p-ième de la fonction d'activation. De manière similaire nous notons $H^{k(p)}$ le vecteur des dérivées p-ième.
– Nous notons p_j^k la fonction de *propagation* du réseau de neurone des couches 0, 1,...,$k-1$ vers l'unité j de la couche k. Cette fonction dépend d'un ensemble de paramètres que nous notons W^k. Plus précisément, on aura

$$\begin{array}{rcccl} p_j^k & : & \mathbb{R}^{s_0} \times \mathbb{R}^{s_1} \times \ldots \times \mathbb{R}^{s_{k-1}} \times \mathbb{R}^{t_k} & \longrightarrow & \mathbb{R} \\ & & (X^0, X^1, \ldots, X^{k-1}, W^k) & \longrightarrow & p_j^k(X^0, X^1, \ldots, X^{k-1}; W^k) \end{array}$$

Nous supposons que p_j^k est de classe $\mathcal{C}^p$ par rapport aux paramètres W^k pour tout $(X^0, X^1, \ldots, X^{k-1})$ et de classe $\mathcal{C}^1$ par rapport à $(X^0, \ldots, X^{k-1})$ pour tout W^k.
Etant donné k vecteurs $X^0, \ldots, X^{k-1}$, et une valeur des paramètres W^k fixée, on note $P^k \equiv P^k(X^0, \ldots, X^{k-1}; W^k)$ le vecteur des s_k fonctions $(p_j^k)_{j=1}^{s_k}$.
– Nous notons $\mathbb{W} = (W^k)_{k=1}^m$ l'ensemble des paramètres du réseau.

Pour les exemples nous utilisons les conventions suivantes :
– Nous noterons (X_i^0, Y_i) les exemples de la base. L'indice 0 signifie qu'il s'agit de l'entrée du réseau. Y_i sera la sortie *attendue* du réseau.
– Etant donné un vecteur $X^0 \in \mathbb{R}^{s_0}$ en entrée et des valeurs des paramètres W^k fixées, on propapage le vecteur par la formule de récurrence

$$X^k = H^k(P^k(X^0, \ldots, X^{k-1}; W^k)) \tag{1}$$

Cette équation est appelée l'équation de propagation. Le vecteur X^m est appelé la sortie *calculée* du réseau.
Nous introduirons quelques notations supplémentaires dans le texte.

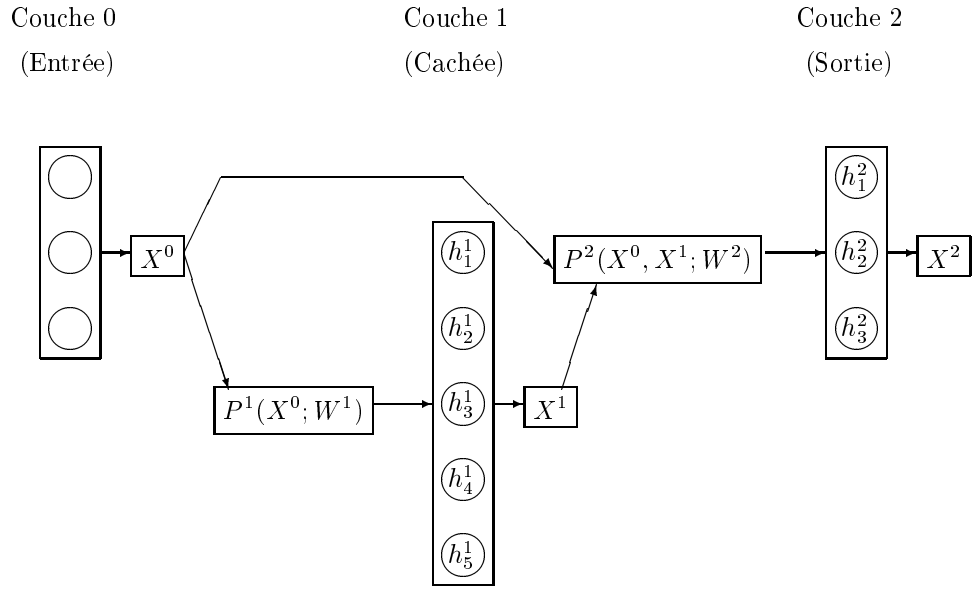

FIG. 1 – Un Réseau de Neurones à propagation avant avec une couche cachée

2.2 Exemple de réseaux de neurones à propagation avant

2.2.1 Le perceptron à m couches avec connexions directes vers la sortie

Pour les couches $k = 1$ à $m-1$, les fonctions de propagation p_j^k sont des fonctions affines qui effectuent les transformations suivantes :

$$P^k(X^0, \ldots, X^{k-1}; A^k, B^k) = A^k X^{k-1} + B^k$$

et pour la dernière couche la fonction de propagation effectue la transformation suivante

$$P^m(X^0, \ldots, X^{m-1}; A^0, A^m, B^m) = A^0 X^0 + A^m X^{m-1} + B^m$$

Les paramètres du percetron sont les matrices de poids $(A^0, A^1, \ldots, A^m)$ et les vecteurs de biais $(B^1, B^2, \ldots, B^m)$.

Les fonctions d'activation sont en général la fonction identité $(h_j^k(x) = x)$, la fonction logistique $(h_j^k(x) = e^x/(1+e^x))$ ou la tangeante hyperbolique $(h_j^k(x) = \tanh(x))$.

2.2.2 Les réseaux de neurones RBF (Radial Basis Function)

Pour la première couche, les fonctions de propagation p_j^1 sont des fonctions quadratiques

$$p_j^1(X^0; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = (X^0 - \boldsymbol{\mu}_j)' \boldsymbol{\Sigma}_j^{-1} (X^0 - \boldsymbol{\mu}_j)$$

Les paramètres du réseaux pour la première couche sont les centres et les matrices de variances covariances

$$W^1 = (\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)_{j=1}^{s_1}$$

où $\boldsymbol{\mu}_j \in \mathbb{R}^{s_0}$ pour tout j et $\boldsymbol{\Sigma}_j$ est une matrice de variance-covariance pour tout j. On prend souvent $\boldsymbol{\Sigma}_j$ diagonale.

Remarquons, que la propagation peut aussi s'écrire comme une fonction quadratique de l'entrée, on a

$$p_j^1(X^0) = X^{0\prime} \mathbf{M}_j X^0 + 2\mathbf{b}_j' X^0 + c_j$$

où $\mathbf{M}_j, \mathbf{b}_j, c_j$ sont respectivement une matrice, un vecteur et une constante. Cela fait un nombre important de paramètres, on peut le réduire en faisant l'hypothèse que $\mathbf{M}_j = \mathbf{M}, \forall j$ ou en faisant l'hypothèse que $\mathbf{M}_j = \sigma_j^2 Id$.

Les fonctions d'activation sont pour la première couche des fonctions radiales, en général la gaussienne non normalisée. :

$$h_j^1(x) = e^{-\frac{1}{2}x}$$

Pour les couches suivantes $(k = 2, \ldots, m)$, les fonctions de propagation sont des fonctions affines comme pour un perceptron, les fonctions d'activation sont aussi les mêmes que celles utilisées pour un perceptron.

3 Estimation des paramètres (Apprentissage)

De manière à présenter le problème de l'estimation de manière très général, nous introduisons l'ensemble auxiliaire de paramètre $\mathbb{X} = (X_i^k)$ pour $i = 1, \ldots, n$ et $k = 1, \ldots, m$, nous notons $X^m = (X_1^m, \ldots, X_n^m)$ et nous nous donnons une fonction de coût $g(X^m, \mathbb{W})$. Nous supposerons que g est de classe $\mathcal{C}^p$ par rapport aux paramètres $\mathbb{W}$ pour tout X^m et de classe $\mathcal{C}^1$ par rapport aux paramètres X^m pour tout $\mathbb{W}$.

L'estimation des paramètres s'effectue en minimisant la fonction de coût

$$g(X^m, \mathbb{W}). \tag{2}$$

Remarquons que g dépend aussi des paramètres $\mathbb{W}$ à travers X^m (dépendance implicite) et que sous cette forme le problème de minimisation (2) n'est pas complètement spécifié. En fait, on peut le réécrire comme un problème de minimisation sous contraintes. En effet en considérant l'équation de propagation (1), on voit que l'on cherche en fait à résoudre le programme suivant

$$\begin{cases} \min\limits_{\mathbb{W},\mathbb{X}} g(X^m, \mathbb{W}) \\ \text{s.c. } X_i^k - H^k(P^k(X_i^0, \dots, X_i^{k-1}; W^k)) = 0 \quad i = 1, \dots, n, \quad k = 1, \dots, m \end{cases} \tag{3}$$

De manière usuelle, on transforme ce problème de minimisation sous contraintes en un problème de minimisation sans contraintes en introduisant les multiplicateurs de Lagrange. Pour cela on définit les vecteurs lignes $\mathbb{E} = (E_i^k)$ pour $i = 1, \dots, n$ et $k = 1, \dots, m$, et on cherche désormais à résoudre le programme

$$\begin{aligned} \min_{\mathbb{W},\mathbb{X},\mathbb{E}} \mathcal{L}(\mathbb{W}, \mathbb{X}, \mathbb{E}) \quad &= g(X^m, \mathbb{W}) \\ &+ \textstyle\sum_{i=1}^n \sum_{k=1}^m E_i^k (X_i^k - H^k(P^k(X_i^0, \dots, X_i^{k-1}; W^k)). \end{aligned} \tag{4}$$

Cette approche a été introduite par Vapnick [5].

Avant de passer au calcul des dérivées, donnons quelques exemples de fonctions de coût.

3.1 Exemple de fonctions de coût

3.1.1 Régression non linéaire avec bruit blanc

On suppose que l'on a le modèle

$$Y_i = F(X_i^0) + \epsilon_i$$

où ϵ_i est une suite de variable aléatoire iid de carré intégrable et où F est une fonction inconnue de $\mathbb{R}^n$ dans $\mathbb{R}$. On peut estimer F à l'aide d'un réseau de neurones à propagation avant en prenant comme fonction objectif

$$g(X^m, \mathbb{W}) = \sum_{i=1}^n \|Y_i - X_i^m\|^2.$$

On peut introduire une pénalisation sur les paramètres (régression ridge) en prenant comme fonction objectif

$$g(X^m, \mathbb{W}) = \sum_{i=1}^n \|Y_i - X_i^m\|^2 + \lambda \sum_{k=1}^m \left\|W^k\right\|^2$$

où λ est un paramètre fixé a priori.

3.1.2 Régression logistique non-linéaire

On cherche à déterminer les probabilités d'appartenance dans deux sous-populations d'un échantillon d'individu. Si pour chaque individu, la probabilité d'appartenir au groupe 1 est $P(Y_i = 1) = \pi_i = \pi(X_i^0)$ où π est une probabilité inconnue. La vraisemblance de l'échantillon s'écrit

$$L(Y_1, \dots, Y_n) = \Pi_{i=1}^n \pi_i^{Y_i} (1 - \pi_i)^{1 - Y_i}.$$

On peut estimer π_i à l'aide d'un réseau de neurones à propagation avant en prenant comme fonction objectif

$$g(X^m, \mathbb{W}) = -\sum_{i=1}^{n} Y_i \log(X_i^m) + (1 - Y_i) \log(1 - X_i^m)$$

Dans ce cas la sortie de la dernière couche doit être un nombre compris dans l'intervalle $]0, 1[$ (ce qui peut être obtenu en prenant comme fonction d'activation pour la dernière couche la fonction logistique).

3.1.3 Séries Chronologiques Autorégressives

On considère une série chronologique $(X_t^0)_{t\in 0,\dots,T}$ dans $\mathbb{R}^n$ et on suppose que cette série a une dynamique autorégressive non-linéaire de la forme

$$X_{t+1}^0 = F(X_t^0, \dots, X_{t-p}^0) + \epsilon_{t+1}$$

où ϵ_t est une suite de variable aléatoire de matrice de variance covariance Γ inconnue mais supposée inversible.

Si on veut estimer F par un réseau de neurones à propagation avant, alors Rynkiewicz [3] a montré que la maximisation de la vraisemblance dans le cas gaussien était équivalente à la minimisation de la fonction de coût

$$g(X_t^m, \mathbb{W}) = \frac{1}{2} \ln \det(\hat{\Gamma})$$

où

$$\hat{\Gamma} = \frac{1}{n-p} \sum_{t=p}^{T} (X_t^0 - X_t^m)(X_t^0 - X_t^m)^T$$

3.2 Calcul des Dérivées

Pour résoudre le programme obtenu en (4) et appliquer les algorithmes standard d'optimisation, nous devons calculer les dérivées premières et secondes de $\mathcal{L}$ par rapport aux paramètres $\mathbb{W}$ et les dérivées premières par rapport aux paramètres $\mathbb{E}$ et $\mathbb{X}$. En effet nous verrons qu'il est possible d'obtenir une solution explicite pour annuler le gradient par rapport à ces derniers.

Pour ne pas surcharger les notations, nous allons supprimer l'indice i de nos notations. Le calcul des dérivées dans le cas général n'est pas difficile.

Les dérivées par rapport aux multiplicateurs de Lagrange sont les plus simples à obtenir puisque l'on a

$$\frac{\partial \mathcal{L}}{\partial e_j^k} = x_j^k - h_j^k(p_j^k)$$

En posant ces équations égales à zéro, on retrouve l'équation de propagation (1).

Considérons maintenant les dérivées par rapport aux exemples. Ici il faut faire la distinction lorsque $k = m$ et lorsque $k \neq m$. Dans le premier cas on a :

$$\frac{\partial \mathcal{L}}{\partial x_j^m} = \frac{\partial g}{\partial x_j^m} + e_j^m$$

et lorsque $1 \leq k < m$, on a :

$$\frac{\partial \mathcal{L}}{\partial x_j^k} = e_j^k - \sum_{k_1=k+1}^{m} \sum_{j_1=1}^{s_{k_1}} e_{j_1}^{k_1} h_{j_1}^{k_1(1)}(p_{j_1}^{k_1}) . \frac{\partial p_{j_1}^{k_1}}{\partial x_j^k}$$

En posant ces équations égales à zéro, on obtient les équations de rétro-propagation. On calcule en sortie

$$e_j^m = -\frac{\partial g}{\partial x_j^m}, \qquad j = 1, \ldots, s_m \tag{5}$$

puis on calcule les quantités $e_j^{m-1}, \ldots, e_j^1$ à l'aide des formules de récurrence suivantes :

$$\begin{cases} e_j^{k+1(1)} = e_j^{k+1} h_j^{k+1(1)}(p_j^{k+1}), & j = 1, \ldots, s_{k+1} \\ e_j^k = \sum_{k_1=k+1}^{m} \sum_{j_1=1}^{s_{k_1}} e_{j_1}^{k_1(1)} \frac{\partial p_{j_1}^{k_1}}{\partial x_j^k} & j = 1, \ldots, s_k \end{cases} \tag{6}$$

La quantité $e_j^{k(1)}$ est habituellement notée δ_j^k.

Finalement il nous reste à calculer les dérivées par rapport aux paramètres $\mathbb{W}$, pour les dérivées premières on a :

$$\frac{\partial \mathcal{L}}{\partial w_l^k} = -\sum_{j=1}^{s_k} e_j^k h_j^{k(1)}(p_j^k) \frac{\partial p_j^k}{\partial w_l^k} + \frac{\partial g}{\partial w_l^k} = -\sum_{j=1}^{s_k} e_j^{k(1)} \frac{\partial p_j^k}{\partial w_l^k} + \frac{\partial g}{\partial w_l^k} \tag{7}$$

pour $l = 1, \ldots, t_k$. Pour les dérivées secondes on a :

$$\frac{\partial^2 \mathcal{L}}{\partial w_{l_1}^{k_1} \partial w_{l_2}^{k_2}} = \frac{\partial^2 g}{\partial w_{l_1}^{k_1} \partial w_{l_2}^{k_2}} \text{ si } k_1 \neq k2$$

et

$$\frac{\partial^2 \mathcal{L}}{\partial w_{l_1}^k \partial w_{l_2}^k} = -\sum_{j=1}^{s_k} e_j^{k(1)} \frac{\partial^2 p_j^k}{\partial w_{l_1}^k \partial w_{l_2}^k} - \sum_{j=1}^{s_k} e_j^{k(2)} \frac{\partial p_j^k}{\partial w_{l_1}^k} \frac{\partial p_j^k}{\partial w_{l_2}^k} + \frac{\partial^2 g}{\partial w_{l_1}^k \partial w_{l_2}^k} \tag{8}$$

où on a posé $e_j^{k(2)} = e_j^k h_j^{k(2)}(p_j^k)$.

3.3 Mise en oeuvre

Pour calculer les dérivées premières et secondes d'un réseau de neurones multicouches à propagation avant, il faut donc réaliser les étapes suivantes

1. `Propagation : Pour` $i = 1, \ldots, n$
   (a) `présenter` X_i^0 `au réseau`
   (b) `calculer` $X_i^1, \ldots, X_i^m$ `et les quantités` $h_j^{k(1)}(p_j^k(X_i^0, X_i^1, \ldots, X_i^k; W^k))$ `et` $h_j^{k(2)}(p_j^k(X_i^0, X_i^1, \ldots, X_i^k; W^k))$ `pour` $k = 1, \ldots, m$.
2. `Rétro-propagation : Pour` $i = 1, \ldots, n$
   (a) `calculer` e_{ij}^m .
   (b) `Calcul des quantités` $e_{ij}^k, e_{ij}^{k(1)}, e_{ij}^{k(2)}$ `pour` $j = 1, \ldots, s_k$, $k = 1, \ldots, m-1$.
   (c) `Calcul des dérivées à l'aide des formules (7) et (8).`

Les formules (7) et (8) sont des dérivées locales (pour un exemple), elles se calculent en faisant une somme sur i.

En général, la quantité $\partial^2 g / \partial w_{l_1}^{k_1} \partial w_{l_2}^{k_2}$ est nulle. Dans ce cas, pour tous les réseaux de neurones à propagation avant, le nombre d'éléments à stocker pour calculer les dérivées secondes est $t_1^2 + t_2^2 +, \ldots, t_k^2$.

4 Applications

4.1 Le perceptron à m couches avec connexions directes vers la sortie

Pour ne pas avoir à faire la distinction entre les poids et les biais il est commode de transformer l'ensemble des paramètres $\mathbb{X}$ en ajoutant à chacun des vecteurs X^k, une coordonnée qui vaut toujours 1. Dans ce cas la fonction de propagation est linéaire et les équations de propagation s'écrivent très simplement

$$X^k = H^k(W^k X^{k-1})$$

pour $k = 1, \ldots, m-1$ et

$$X^m = H^m(W^m X^{m-1} + W^0 X^0)$$

la dernière coordonnées de la fonction H^k est donc la fonction qui vaut toujours 1.

Afin de simplifier, nous supposerons que la fonction de coût ne dépend que des sorties calculées du réseau et ne dépend pas directement des paramètres.

4.1.1 Calcul des dérivées premières

L'équation de rétro-propagation (6) bien connue devient :

$$\left\{ \begin{array}{ll} e_j^{k+1(1)} = e_j^{k+1} h_j^{k+1(1)} (\sum_{j_2=1}^{s_k} w_{jj_2}^{k+1} x_{j_2}^k), & j = 1, \ldots, s_{k+1} \\ e_j^k = \sum_{j_1=1}^{s_{k+1}} e_{j_1}^{k+1(1)} w_{j_1 j} & j = 1, \ldots, s_k \end{array} \right. \tag{9}$$

pour $k = m-1$ jusqu'à $k = 1$. La dérivée par rapport au poids $w_{j_1 j_2}^k$ ($k > 0$ s'écrit :

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w_{j_1 j_2}^k} &= -e_{j_1}^k x_{j_2}^{k-1} h_{j_1}^{k(1)} \left(\sum_{j=1}^{s_{k-1}} w_{j_1 j}^k x_j^{k-1} \right) \\ &= -e_{j_1}^{k(1)} x_{j_2}^{k-1} \end{aligned}$$

et la dérivée par rapport au poids $w_{j_1 j_2}^0$ s'écrit

$$\frac{\partial \mathcal{L}}{\partial w_{j_1 j_2}^0} = -e_{j_1}^m x_{j_2}^0$$

On peut observer que le vecteur gradient correspondant à la j-ième ligne de la matrice W^k ($k > 0$) s'écrit :

$$\frac{\partial \mathcal{L}}{\partial W_j^k} = -e_j^{k(1)} X^{k-1}$$

et le vecteur gradient correspondant à la j-ième ligne de la matrice W^0 s'écrit :

$$\frac{\partial \mathcal{L}}{\partial W_j^0} = -e_j^{m(1)} X^0$$

4.1.2 Calcul des dérivées secondes

Pour les dérivées secondes, on a

$$\frac{\partial^2 \mathcal{L}}{\partial w_{j_1 j_2}^{k_1} \partial w_{j_3 j_4}^{k_2}} = 0 \qquad \text{si } k_1 \neq k_2, \text{ ou si } j_1 \neq j_3$$

et

$$\begin{aligned}\frac{\partial^2 \mathcal{L}}{\partial w^k_{j_1 j_2} \partial w^k_{j_1 j_4}} &= -e^k_{j_1} x^{k-1}_{j_2} x^{k-1}_{j_4} h^{k(2)}_{j_1} \left(\sum_{j=1}^{s_{k-1}} w^k_{j_1 j} x^{k-1}_j \right) \\ &= -e^{k(2)}_{j_1} x^{k-1}_{j_2} x^{k-1}_{j_4} \text{ lorsque } k > 0 \end{aligned} \tag{10}$$

et

$$\frac{\partial^2 \mathcal{L}}{\partial w^0_{j_1 j_2} \partial w^0_{j_1 j_4}} = -e^{m(2)}_{j_1} x^0_{j_2} x^0_{j_4} \tag{11}$$

On peut observer que les dérivées secondes peuvent être calculer pour chacune des lignes de la matrice W^k indépendemment les unes des autres. Pour la ligne j, la matrice des dérivées secondes s'écrit

$$\frac{\partial^2 \mathcal{L}}{\partial {W^k_j}^2} = e^{k(2)}_j X^{k-1} X^{k-1\prime}. \tag{12}$$

Le nombre d'éléments à stocker pour calculer les dérivées secondes d'un perceptron avec connexions directes vers la couche de sortie est donc $s_0^2 + s_1^2 + s_2^2 + \ldots + s_m^2$.

4.1.3 Calcul des dérivées d'ordre supérieur

Le calcul des dérivées supérieures est évident, nous donnons à titre d'exemple, la dérivée d'ordre p, on a

$$\frac{\partial^p \mathcal{L}}{\partial w^k_{j_1 j_{l_1}} \ldots \partial w^k_{j_1 j_{l_p}}} = -e^{(p)k}_{j_1} x^{k-1}_{j_{l_1}} \ldots x^{k-1}_{j_{l_p}} \tag{13}$$

Les dérivées sont nulles dés que l'indice de ligne de deux poids est différent, ou dés que l'indice du numéro de couche est différent.

4.2 Les réseaux de neurones RBF (Radial Basis Function)

Pour les réseaux de Neurones RBF, seule la première couche intervient avec une propagation quadratique et l'équation de retro-propagation est la même que pour un perceptron. Seul les calculs de la dérivée des paramètres intervenant dans la propagation de la couche 0 vers la couche 1 sont différentes.

Comme pour le perceptron, il est commode de rajouter aux vecteurs $X^0, \ldots, X^{m-1}$ une coordonnée qui vaut toujours 1. Dans ce cas la fonction de propagation p^1_j s'écrit

$$p^1_j(X^0; W^1) = X^{0\prime} M_j X^0 = \sum_{j_1=1}^{s_0} \sum_{j_2=1}^{s_0} m_{j_1 j_2 j} x^0_{j_1} x^0_{j_2}$$

Un calcul rapide montre alors que les équations de rétropropagation sont les mêmes que pour le perceptron (les couches 2 à m sont les mêmes) et que la dérivée première par rapport au paramètre $m_{j_1 j_2 j}$ (l'élément de la ligne j_1 et de la colonne j_2 de la j-ième matrice) s'écrit

$$\frac{\partial \mathcal{L}}{\partial m_{j_1 j_2 j}} = -e^{1(1)}_j x^0_{j_1} x^0_{j_2}$$

tandis que la dérivée seconde par rapport aux paramètres $m_{j_1 j_2 j}$ et $m_{j_3 j_4 j}$ s'écrit

$$\frac{\partial^2 \mathcal{L}}{\partial m_{j_1 j_2 j} \partial m_{j_3 j_4 j}} = -e^{1(2)}_j x^0_{j_1} x^0_{j_2} x^0_{j_3} x^0_{j_4}$$

Si pour l'un des paramètres, on a $m_{j_1 j_2} \equiv m_{j_1 j_2 j}$ pour tout j, la dérivée première devient

$$\frac{\partial \mathcal{L}}{\partial m_{j_1 j_2}} = -x^0_{j_1} x^0_{j_2} \sum_{j=1}^{s_1} e^{1(1)}_j$$

et les dérivées secondes deviennent

$$\frac{\partial^2 \mathcal{L}}{\partial m_{j_1 j_2} \partial m_{j_3 j_4 j}} = -x^0_{j_1} x^0_{j_2} x^0_{j_3} x^0_{j_4} \sum_{j_5=1}^{s_1} e^{1(2)}_{j_5}$$

5 Conclusion

Nous avons développer un algorithme pour évaluer de manière exacte les dérivées premières et secondes d'un réseau de propagation avant très général : propagation arbitraire, connexions directes entre couches non contigues et fonction de coût très générale pouvant inclure un terme de régularisation sur les paramètres.

Une implémentation d'une méthode utilisant le calcul effectif des dérivées secondes pour estimer les paramètres du perceptron et des réseaux RBF devrait permettre d'accélérer de manière significative les méthodes d'estimation des paramètres actuellement utilisées.

Références

[1] Bishop C., (1992) *Exact calculation of the hessian matrix for the multilayer Perceptron*, Neural Computation, Vol 2, pp 494-501.

[2] Buntine W.L., Weigend A.S. (1995) *Computing 2nd derivatives in feed-forward networks. A review.* , IEEE transactions on Neural Networks, 5 (3), pp 480-488.

[3] Rynkiewicz J. (2000) *Estimation et identification de modèles autorégressifs non-linéaires multidimensionnels*, Prépublication du Samos 125.

[4] Shepherd A. J. (1997) *Second-Order Methods for Neural Networks*, Springer Verlag.

[5] Vapnick V. N. (1995) *The Nature of Statistical Learning Theory*, Springer Verlag.