5 UN NOUVEL ALGORITHME D'APPRENTISSAGE

Résumé

Le score de validation croisée E_p étudié dans le chapitre 4 peut être considéré comme la valeur prise - à la fin de l'apprentissage - par $\sqrt{J^* / N}$ où J^* est obtenue à partir du coût quadratique en pondérant chaque résidu en fonction de son influence sur les poids du modèle. Nous étudions dans ce chapitre la minimisation itérative de J^*, qui n'est définie que sur un domaine de l'espace des coefficients, dont nous ne connaissons pas les propriétés.

Nous montrons, dans un premier temps, que, moyennant l'approximation selon laquelle les $\{h_{ii}\}_{i=1, \ldots, N}$ ne dépendent que faiblement des coefficients du modèle, le gradient de J^* par rapport à ces derniers se calcule aisément. Ensuite, nous proposons une formule de modification itérative des coefficients directement inspirée de l'algorithme de Levenberg-Marquardt.

Puisque l'algorithme ne doit pas faire sortir les paramètres du domaine de définition de J^*, il faut considérer la minimisation de J^* comme une poursuite de la minimisation de J, sous réserve que celle-ci ait convergé vers un minimum de rang plein.

Les résultats montrent surtout un gain très important au niveau de la dispersion des scores E_p des minima atteints, ce qui permet d'augmenter la probabilité de trouver le vecteur des coefficients pour lequel E_p est le plus faible.

Cette méthode s'avère être une façon de régulariser les modèles a posteriori, c'est-à-dire après minimisation du coût quadratique. Il s'agit donc d'un complément très intéressant aux travaux présentés dans le chapitre 4.

5.1 Introduction

Dans le chapitre précédent, nous avons montré que, pour une architecture donnée, il était souhaitable de sélectionner le minimum $\boldsymbol{\theta}_{LS}$ de la fonction de coût quadratique $J(\boldsymbol{\theta}) = {}^t[\boldsymbol{y}_p - \boldsymbol{f}(X, \boldsymbol{\theta})]\,[\boldsymbol{y}_p - \boldsymbol{f}(X, \boldsymbol{\theta})]$ qui possède le plus petit score de validation croisée estimé $E_p = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{R_i}{1-h_{ii}}\right)^2}$, R_i et h_{ii} étant calculés à $\boldsymbol{\theta}_{LS}$. Ceci permet de limiter le surajustement, tout en utilisant tous les exemples disponibles.

Ceci revient à appliquer un algorithme d'apprentissage par rapport à une fonction J alors que l'on cherche en réalité le minimum d'une autre fonction. Nous montrons dans ce chapitre que l'on peut directement chercher à minimiser la fonction :

$$J^*(\boldsymbol{\theta}) = {}^t[\boldsymbol{y}_p - \boldsymbol{f}(X, \boldsymbol{\theta})]\,[I - H^*(\boldsymbol{\theta})]^{-2}\,[\boldsymbol{y}_p - \boldsymbol{f}(X, \boldsymbol{\theta})], \qquad (5.1)$$

dans laquelle H^* est la partie diagonale de la matrice de projection $H = Z\,({}^tZ\,Z)^{-1}\,{}^tZ$.

En notant $\boldsymbol{f}^*(X, \boldsymbol{\theta})$ le vecteur constitué par les N fonctions obtenues en éliminant l'influence de chaque exemple d'apprentissage $\left\{ f^{(-i)}(\boldsymbol{x}^i, \boldsymbol{\theta}) = y_{pi} - \frac{y_{pi} - f(\boldsymbol{x}^i, \boldsymbol{\theta})}{1 - h_{ii}} \right\}_{i = 1, \ldots, N}$, J^* peut également s'écrire [7] :

$$J^*(\boldsymbol{\theta}) = {}^t[\boldsymbol{y}_p - \boldsymbol{f}^*(X, \boldsymbol{\theta})]\,[\boldsymbol{y}_p - \boldsymbol{f}^*(X, \boldsymbol{\theta})]. \tag{5.2}$$

Une différence fondamentale entre ces deux fonctions de coût concerne leurs domaines de définition respectifs : contrairement à J, J^* n'est pas définie sur tout $\Re^q$ mais uniquement dans le domaine (noté Ω^*) où Z est de rang plein (égal à q). Il faudra par conséquent veiller, si l'on envisage une minimisation itérative de J^*, à rester dans ce domaine.

L'autre caractéristique de J^* est - compte tenu du fait que les $\{h_{ii}\}_{i=1, \ldots, N}$ sont tous compris entre 0 et 1 (cf. formule (2.2)) - d'être systématiquement supérieure à J.

Dans tout ce qui suit, afin de simplifier les notations, nous continuerons d'appeler h_{ii} les éléments diagonaux de H, tout en précisant qu'ils sont définis pour tout $\boldsymbol{\theta}$ et non plus seulement en $\boldsymbol{\theta}_{LS}$.

La suite de ce chapitre est consacrée à l'étude de la minimisation de la fonction J^* par adaptation de l'algorithme de Levenberg-Marquardt. Nous étudierons tout d'abord les calculs du coût et du gradient de cette fonction par rapport aux poids du modèle ainsi que la question de la modification itérative des coefficients. Ensuite, nous verrons comment mettre en œuvre cet algorithme de manière à en tirer profit.

5.2 Algorithme pour la minimisation de J^*

5.2.1 Calculs du coût et du gradient

La particularité de la minimisation de $J^*(\boldsymbol{\theta})$ réside dans le fait que les $\{h_{ii}\}_{i=1, \ldots, N}$ sont eux-mêmes fonctions des poids du modèle (sauf dans le cas d'un modèle linéaire). Cependant, nous allons montrer qu'en négligeant cette dépendance, le gradient de J^* par rapport aux coefficients se calcule simplement. En effet, notons :

$$J_i(\boldsymbol{\theta}) = \left(y_{pi} - f(\boldsymbol{x}^i, \boldsymbol{\theta})\right)^2 \tag{5.3}$$

et

$$J_i^*(\boldsymbol{\theta}) = \left(\frac{y_{pi} - f(\boldsymbol{x}^i, \boldsymbol{\theta})}{1 - h_{ii}(\boldsymbol{\theta})}\right)^2 \tag{5.4}$$

les contributions de l'exemple i respectivement à J et J^*. On a alors :

---

[7] Compte tenu de la définition (5.1), l'erreur E_p utilisée dans le chapitre 4 représente la valeur prise par $\sqrt{\frac{J^*}{N}}$ à la fin de la minimisation du coût quadratique J.

$$\frac{\partial J_i}{\partial \boldsymbol{\theta}} = -2\left(y_{pi} - f(\boldsymbol{x}^i,\boldsymbol{\theta})\right)\frac{\partial f(\boldsymbol{x}^i,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{5.5}$$

expression dans laquelle la dérivée de la fonction de régression par rapport au vecteur $\boldsymbol{\theta}$ s'obtient facilement, soit par calcul direct, soit - dans le cas d'un réseau de neurones - par rétropropagation.

En revanche, le calcul de $\frac{\partial J_i^*}{\partial \boldsymbol{\theta}}$ nécessiterait, en toute rigueur, celui du vecteur $\frac{\partial h_{ii}}{\partial \boldsymbol{\theta}}$. Dans tout ce qui précède, notamment dans le chapitre 3, nous avons toujours supposé qu'un développement au premier ordre dans l'espace des paramètres était valable ; en d'autres termes, nous avons supposé que les vecteurs qui définissent le sous-espace des solutions sont indépendants de $\boldsymbol{\theta}$. Or h_{ii} est la $i^{\text{ème}}$ composante de la projection, sur le sous-espace des solutions, du vecteur unité le long de l'axe i ; nous restons donc dans le même cadre d'approximation en supposant que les $\{h_{ii}\}_{i=1,\ldots,N}$ sont indépendants des paramètres :

$$\frac{\partial h_{ii}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cong 0 \tag{5.6}$$

L'approximation (5.6) permet ainsi d'obtenir :

$$\frac{\partial J_i^*}{\partial \boldsymbol{\theta}} \cong \frac{1}{(1-h_{ii})^2}\frac{\partial J_i}{\partial \boldsymbol{\theta}} \tag{5.7}$$

Nous disposons par conséquent d'une expression approchée du gradient de J^* par rapport aux poids du modèle, expression qui se calcule très facilement à partir du gradient de J et des valeurs $\{h_{ii}\}_{i=1,\ldots,N}$.

5.2.3 Modification des coefficients

Après un rappel du principe et des fondements de l'algorithme de Levenberg-Marquardt, nous verrons comment adapter celui-ci à la minimisation de la fonction J^*.

5.2.3.a Rappel : l'algorithme de Levenberg-Marquardt

Cet algorithme est une méthode itérative de minimisation de fonctions. Supposons que l'on cherche à minimiser une fonction J par rapport à un vecteur de variables $\boldsymbol{\theta}$, que l'on se trouve, à l'étape k-1, au point $\boldsymbol{\theta}_{k-1}$ et que l'on cherche à se déplacer vers un point $\boldsymbol{\theta}_k$. Alors, si le déplacement $\boldsymbol{\theta}_{k-1} - \boldsymbol{\theta}_k$ est suffisamment petit, on peut approcher le vecteur des résidus par un développement de Taylor au premier ordre :

$$\boldsymbol{y}_p - \boldsymbol{f}(X, \boldsymbol{\theta}_k) \cong \boldsymbol{y}_p - \boldsymbol{f}(X, \boldsymbol{\theta}_{k-1}) + Z_{k-1}(\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}) \tag{5.8}$$

L'idée de l'algorithme de Levenberg-Marquardt est de choisir $\boldsymbol{\theta}_k$ de manière à minimiser $J(\boldsymbol{\theta}_k)$ tout en limitant la distance entre $\boldsymbol{\theta}_{k-1}$ et $\boldsymbol{\theta}_k$. On écrit donc :

$$E(\boldsymbol{\theta}_k) = {}^t[\boldsymbol{y}_p - \boldsymbol{f}(X, \boldsymbol{\theta}_k)]\,[\boldsymbol{y}_p - \boldsymbol{f}(X, \boldsymbol{\theta}_k)] + \lambda\,{}^t[\boldsymbol{\theta}_{k-1} - \boldsymbol{\theta}_k]\,[\boldsymbol{\theta}_{k-1} - \boldsymbol{\theta}_k] \tag{5.9}$$

En utilisant (5.8), on obtient l'expression des poids tels que $\frac{\partial E(\boldsymbol{\theta}_k)}{\partial \boldsymbol{\theta}_k} = 0$:

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \left(2\, {}^t Z_{k-1}\, Z_{k-1} + \lambda_{k-1}\, I \right)^{-1} \left. \frac{\partial\, J}{\partial\, \boldsymbol{\theta}} \right|_{\boldsymbol{\theta} = \boldsymbol{\theta}_{k-1}} \tag{5.10}$$

Pour cela, le gradient total $\left. \frac{\partial\, J}{\partial\, \boldsymbol{\theta}} \right|_{\boldsymbol{\theta} = \boldsymbol{\theta}_{k-1}}$ est calculé en sommant les contributions des N exemples d'apprentissages (formule (5.4)).

Cette méthode fait partie des algorithmes de second ordre. En effet, on reconnaît dans l'expression précédente une approximation classique du Hessien de la fonction de coût :

$$H = \left\{ \frac{\partial^2 J}{\partial\, \theta_i\, \partial\, \theta_j} \right\}_{i,\, j = 1,\, \ldots,\, N} \cong 2\, {}^t Z\, Z \tag{5.11}$$

Le paramètre λ, appelé pas de l'apprentissage, permet d'adapter l'algorithme à la forme de la fonction de coût et de réaliser un bon compromis entre la méthode de Newton (λ nul), qui converge très rapidement au voisinage d'un minimum, et la méthode du gradient simple (λ grand), efficace loin des minima.

Il existe plusieurs algorithmes d'asservissement automatique du pas ; nous avons mis en œuvre une technique simple et robuste :

- si la modification des paramètres provoque une diminution du coût, on accepte cette modification et on se rapproche de la direction de Newton en diminuant λ (par exemple par un facteur 10),
- si la modification des paramètres provoque une augmentation du coût, on rejette cette modification et on se rapproche de la direction du gradient en augmentant λ (par exemple par un facteur 10) ; on calcule une nouvelle modification des paramètres avec le nouveau pas.

La seule grandeur restant à fixer avant l'apprentissage est donc la valeur initiale du pas ([Bishop, 95] conseille $\lambda_0 = 0.1$).

5.2.3.b Adaptation à la minimisation de J^*

Qualitativement, minimiser la fonction de coût J^* consiste à calculer la contribution de chaque exemple à la modification du vecteur $\boldsymbol{\theta}$ en faisant comme si cet exemple n'appartenait pas à la base d'apprentissage.

Pour ce faire, il faut donc reconsidérer l'expression (5.8) et l'écrire sous la forme :

$$\begin{aligned} \boldsymbol{y}_p - \boldsymbol{f}^*(X,\, \boldsymbol{\theta}_k) &\cong \boldsymbol{y}_p - \boldsymbol{f}^*(X,\, \boldsymbol{\theta}_{k-1}) + \sum_{i=1}^{N} \left. \frac{\partial\, f^{(-i)}(\boldsymbol{x}^i, \boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} \right|_{\boldsymbol{\theta} = \boldsymbol{\theta}_{k-1}} (\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}) \\ &\cong \boldsymbol{y}_p - \boldsymbol{f}^*(X,\, \boldsymbol{\theta}_{k-1}) + Z^*_{k-1}\, (\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}), \end{aligned} \tag{5.12}$$

Dans l'expression précédente, la matrice Z^*_{k-1} représente, à l'étape *k*-1, la matrice (*N*, *q*) dont les lignes sont égales aux dérivées partielles par rapport à $\boldsymbol{\theta}$ des fonctions $\left\{ f^{(-i)}(\boldsymbol{x}^i, \boldsymbol{\theta}) \right\}_{i=1,\ldots,N}$.

Par analogie avec l'expression (5.10), on obtient donc :

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \left(2\, {}^t Z^*_{k-1} Z^*_{k-1} + \lambda_{k-1} I \right)^{-1} \left. \frac{\partial J^*}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta} = \boldsymbol{\theta}_{k-1}} \tag{5.13}$$

Par ailleurs, par définition de $f^{(-i)}$ et d'après (5.6), on peut faire l'approximation :

$$Z^* = \left(I - H^* \right)^{-1} Z \tag{5.14}$$

La modification des paramètres à l'étape *k* s'écrit finalement :

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \left(2\, {}^t Z_{k-1} (I - H^*_{k-1})^{-2} Z_{k-1} + \lambda_{k-1} I \right)^{-1} \left. \frac{\partial J^*}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta} = \boldsymbol{\theta}_{k-1}} \tag{5.15}$$

De même que précédemment, le gradient total $\left. \frac{\partial J^*}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta} = \boldsymbol{\theta}_{k-1}}$ s'obtient en sommant les *N* contributions partielles calculées par (5.7).

En ce qui concerne l'asservissement du pas λ, il faut également adapter la méthode présentée précédemment, en intégrant la contrainte concernant le domaine de définition de J^*, domaine duquel on ne doit pas sortir. Nous proposons de l'adapter de la manière suivante :

- si la modification des paramètres provoque une diminution du score de validation croisé J^* **tout en conservant, numériquement, le rang de *Z***, on accepte cette modification et on se rapproche de la direction de Newton en diminuant λ,
- si la modification des paramètres provoque **soit une diminution du rang de *Z*, soit - si ce dernier est conservé -** une augmentation du score de validation croisée J^*, on rejette cette modification et on se rapproche de la direction du gradient en augmentant λ ; on calcule une nouvelle modification des paramètres **avec le nouveau gradient total équivalent**.

5.2.4 En résumé

Nous venons de voir qu'il est possible d'adapter l'algorithme de Levenberg-Marquardt de façon à minimiser la fonction de coût J^*. Les différences par rapport à la minimisation de *J* sont synthétisées dans le tableau 5.1.

Par conséquent, du point de vue du temps de calcul, la seule contrainte supplémentaire liée à la minimisation de J^* est celle du calcul des $\{h_{ii}\}_{i=1,\ \ldots,\ N}$. Ce calcul nécessite une décomposition de *Z* en valeurs singulières à chaque essai du pas λ (voir calcul des $\{h_{ii}\}_{i=1,\ldots,N}$ en Annexe 1). Rappelons néanmoins que la fonction J^* n'est définie que dans un domaine Ω^*.

Contribution de l'exemple *i*…	J	J^*
…à la fonction de coût	$J_i(\boldsymbol{\theta}) = \left(y_{pi} - f(\boldsymbol{x}^i,\boldsymbol{\theta})\right)^2$	$J_i^*(\boldsymbol{\theta}) = \left(\frac{y_{pi} - f(\boldsymbol{x}^i,\boldsymbol{\theta})}{1 - h_{ii}(\boldsymbol{\theta})}\right)^2$
…au gradient	$\frac{\partial J_i}{\partial \boldsymbol{\theta}} = -2\left(y_{pi} - f(\boldsymbol{x}^i,\boldsymbol{\theta})\right)\boldsymbol{z}^i$	$\frac{\partial J_i^*}{\partial \boldsymbol{\theta}} \cong -2\frac{y_{pi} - f(\boldsymbol{x}^i,\boldsymbol{\theta})}{\left(1 - h_{ii}(\boldsymbol{\theta})\right)^2}\boldsymbol{z}^i$
…à la modification des poids	$\left(2\,{}^tZ\,Z + \lambda\, I\right)^{-1}\frac{\partial J_i}{\partial \boldsymbol{\theta}}$	$\left(2\,{}^tZ\,(I - H^*)^{-2}\,Z + \lambda\, I\right)^{-1}\frac{\partial J_i^*}{\partial \boldsymbol{\theta}}$

Tableau 5.1 : Différences entre la minimisation de J et de J^*

5.3 Mise en œuvre de l'algorithme

Nous venons de montrer au paragraphe précédent qu'il était possible d'adapter l'algorithme de Levenberg-Marquardt à la minimisation de la fonction de coût J^*. Or, celle-ci n'étant définie que sur un domaine Ω^* de $\Re^q$, il convient, d'une part, d'initialiser les poids à l'intérieur de ce domaine, et, d'autre part, de s'assurer, à chaque modification des coefficients, que l'on reste bien à l'intérieur de Ω^*.

Ce paragraphe débute par l'étude de ces contraintes et conduit à définir la manière dont nous avons mis en œuvre ce nouvel algorithme d'apprentissage. Ensuite, les résultats ainsi obtenus seront commentés.

5.3.1 Etude des contraintes

Le domaine Ω^* est défini par les exemples d'apprentissage, et, bien entendu, par la famille de fonctions paramétrées considérée. Nous ne connaissons pas ses propriétés mathématiques, en particulier en termes de compacité. On peut néanmoins prévoir une difficulté pratique, résultant de l'initialisation des poids du réseau avant apprentissage : en effet, la procédure habituelle d'initialisation des poids d'un réseau consiste à choisir pour ceux-ci des valeurs aléatoires voisines de zéro, afin de ne pas saturer les fonctions de transfert sigmoïdales en début d'apprentissage. Or, il est facile de vérifier que l'origine de $\Re^q$ ne fait pas partie de Ω^*. Il est donc très probable que ce type d'initialisation fasse démarrer l'apprentissage en-dehors du domaine recherché.

La figure 5.1 illustre le fait que, durant la minimisation de la fonction J lors d'un apprentissage conventionnel, le point représentatif du réseau dans l'espace des paramètres parcourt successivement des zones où la matrice Z est de rang plein, et des zones où elle n'est pas de rang plein, ce qui explique la discontinuité de J^*. Il s'agit d'un apprentissage réalisé sur les données issues de la fonction de régression $\frac{\sin(x)}{}$ avec une architecture à 2 neurones cachés avec terme direct, pourtant peu susceptible de présenter un surajustement. Le minimum atteint est de rang plein, mais l'apprentissage a traversé des zones avec déficience

de rang. Par ailleurs, on constate que bien que J^* - lorsqu'il est défini - n'est soumis à aucune contrainte sinon à celle d'être supérieure à J.

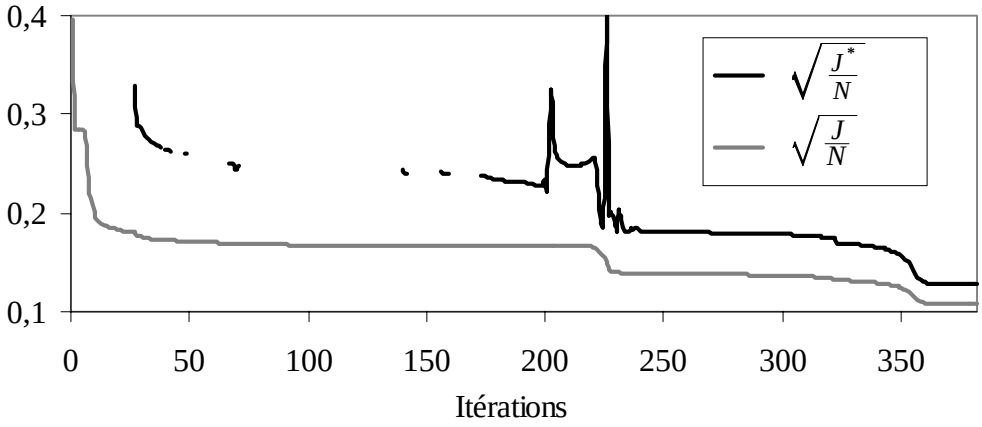

Figure 5.1 : Exemple d'évolution, discontinue, de J^* lors d'une minimisation itérative de J par l'algorithme de Levenberg-Marquardt

De plus nous avons montré au paragraphe 3.3.3 que, pour un exemple à forte influence, le résidu R_i tend en général vers 0 **moins rapidement** que h_{ii} tend vers 1. En d'autres termes, il y a de fortes chances pour que la fonction de coût J^* prenne de grandes valeurs, voire tende vers l'infini, à la frontière du domaine Ω^*. Dans l'hypothèse où Ω^* serait formé d'un ou plusieurs domaines compacts, la surface représentée par J^* serait constituée de "cuvettes" à bord fini ou infini, desquelles il serait donc difficile de sortir.

Dans ces conditions, il apparaît préférable d'utiliser la méthode de minimisation de J^* comme une amélioration itérative de l'apprentissage classique, en laissant l'apprentissage sur J se poursuivre jusqu'à son terme[8], et commencer celui sur J^* à partir du vecteur final des coefficients, sous réserve que celui-ci se trouve dans Ω^*.

5.3.2 Mise en œuvre de l'algorithme

Afin de rendre compte des performances que l'on obtient par la méthode décrite précédemment, nous avons choisi de représenter les solutions atteintes (minima de J ou de J^*) - pour diverses initialisations des poids - par le couple ($EQMA$, E_p) dont les coordonnées représentent les valeurs prises à la fin de l'apprentissage respectivement par $\sqrt{\frac{J}{N}}$ et $\sqrt{\frac{J^*}{N}}$.

La figure 5.2 présente dans le plan ($EQMA$, E_p) les résultats obtenus en poursuivant la minimisation du coût quadratique par la minimisation de J^*. Il s'agit ici d'une architecture à 4 neurones cachés plus terme direct, ajustée sur les données issues de la fonction de régression $\frac{\sin(x)}{x}$. Ces résultats sont comparés à ceux obtenus avec les mêmes initialisations aléatoires des poids (en l'occurrence 500) sans poursuite de l'apprentissage.

---

[8] C'est-à-dire jusqu'à satisfaction de différents critères d'arrêts portant sur la norme du gradient, le pas d'apprentissage, voire le nombre d'itérations.

Certains exemples d'évolution de la solution obtenue, entre la fin de l'apprentissage sur J et la fin de celui sur J^*, sont présentés sur la figure 5.2 sous forme de flèches.

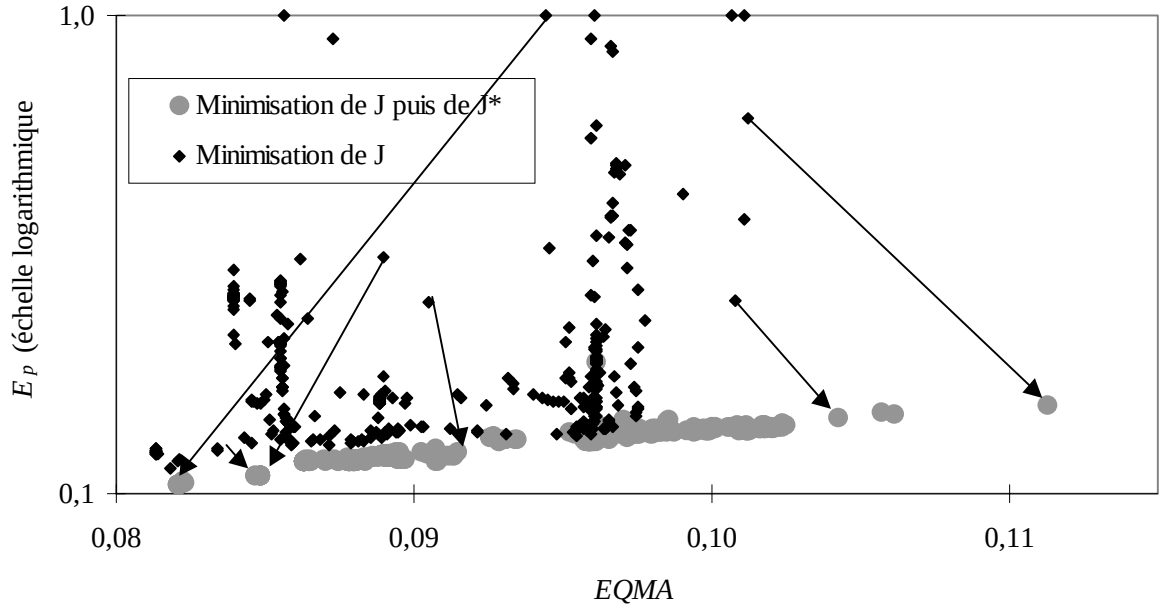

Figure 5.2 : Exemple d'évolution des solutions atteintes grâce à la poursuite de l'apprentissage par la minimisation itérative de J^*

L'analyse de cet exemple amène plusieurs commentaires :

1. comme nous l'avons montré dans le chapitre 4, beaucoup de minima de J présentent - malgré une matrice Z de rang plein - un surajustement certain. Ceci se traduit, dans le plan précédent, par une dispersion des performances des minima beaucoup plus importante sur E_p que sur l'*EQMA*, ce qui nous a conduit à utiliser une échelle logarithmique pour l'axe des ordonnées[9],
2. cette dispersion a été considérablement réduite par le fait de poursuivre, par la minimisation de J^*, chaque apprentissage qui avait convergé vers un minimum de rang plein. A nombre d'initialisations aléatoires fixé, cette stratégie augmente donc la probabilité de détecter le minimum global de J^*,
3. la poursuite de l'apprentissage avec J^* comme fonction de coût conduit à des réductions effective de E_p. Ceci est logique car il n'y aucune raison - surtout pour des "grandes" architectures - que les minima globaux de J et de J^* coïncident. Cependant, pour les minima présentant la plus petite valeur de l'erreur E_p, la réduction de cette dernière n'est pas forcément significative (sur cet exemple le meilleur minimum atteint est passé de $E_p = 0{,}113$ à $E_p = 0.104$),
4. la poursuite de l'apprentissage avec J^* comme fonction de coût ne se traduit pas forcément par une augmentation de l'*EQMA*. La figure 5.2 présente deux exemples pour lesquels

[9] Pour des raisons de lisibilité du graphique, nous avons choisi de représenter les solutions telles que $E_p > 1{,}0$ comme si leur E_p était égale à 1,0.

cette poursuite de l'apprentissage s'est traduite par la diminution conjointe de l'*EQMA* et de E_p.

Dans le cas du problème maître-élève, cette stratégie ne modifie pas les minima sélectionnés sur la base de E_p jusqu'à 5 neurones cachés (figure 5.3). Néanmoins, même pour des architectures de taille supérieure, les erreurs E_p des minima atteints ne sont que légèrement inférieures à celles atteintes par l'apprentissage classique (figure 4.8).

Dans ce cas, la principale amélioration apportée par la minimisation de J^* consiste en une dispersion moindre des solutions atteintes dans le plan (*EQMA*, E_p), ce qui - à nombre d'initialisations égal - augmente la probabilité de trouver le minimum global de J^*.

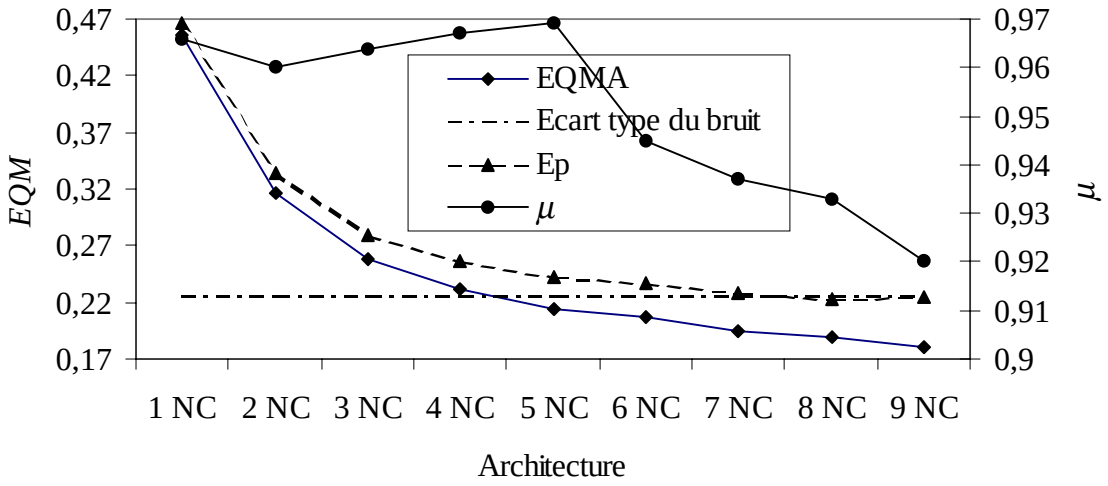

Figure 5.3 : Performances des minima sélectionnés sur E_p suite à la poursuite de l'apprentissage par la minimisation itérative de J^* : cas du problème maître-élève

Finalement, cette stratégie s'avère donc intéressante à plusieurs titres. Même si le gain - en termes de E_p atteint - n'est pas toujours significatif, elle permet de diminuer largement la dispersion et la performance moyenne des minima atteints, quel que soit le nombre d'initialisations aléatoires utilisées. Lors du choix du ou des modèles définitifs dans le plan (E_p, μ), ceci permet d'augmenter le nombre de modèles à la disposition du concepteur.

5.4 Conclusion

Nous avons montré qu'il est possible de considérer directement la fonction J^* comme une fonction de coût, et d'adapter l'algorithme de Levenberg-Marquardt pour la minimiser. Compte tenu des caractéristiques de cette fonction, il est cependant conseillé de considérer l'apprentissage sur J^* comme un prolongement de l'apprentissage sur J.

D'autres méthodes pourraient consister à "changer de fonction de coût" dès que la minimisation de J est rentrée dans le domaine Ω^*, ou à reprendre a posteriori l'apprentissage sur J^* à partir des poids pour lesquels, lors de la minimisation de J, la valeur prise par J^* était la plus faible. Outre le fait de nécessiter le calcul de J^* et donc des $\{h_{ii}\}_{i=1, \ldots, N}$ lors de la première phase de l'apprentissage, ce qui ralentit significativement les calculs, ces autres

méthodes donnent - sur les deux exemples étudiés - de mauvais résultats (erreurs E_p plus élevées et dispersion des minima, dans le plan ($EQMA$, E_p), plus grande qu'après minimisation de J).

Le fait de poursuivre l'apprentissage sur J par la minimisation de J^* peut s'interpréter comme une forme de régularisation a posteriori des solutions obtenues. En effet, on peut considérer J^* comme la somme de J et d'un terme de pénalisation. Cependant, ce type de pénalisation est différent de ceux couramment utilisés car, contrairement à - par exemple - la norme du vecteur des poids, le terme de pénalisation utilisé ici dépend explicitement des données utilisées. On ne pénalise donc pas a priori les solutions à forte variance, mais celles présentent un surajustement par rapport à une certaine base d'apprentissage.

Finalement, les travaux présentés dans ce mémoire montrent que, face au dilemme biais / variance, les approches a priori (par régularisation) et a posteriori (validation croisée) peuvent être utilisées dans un même contexte, à savoir le leave-one-out.