Optimisation des réseaux de neurones de grande capacité

Francis Piéraut

Résumé- **Cet article présente les résultats de l'exploration de nouvelles techniques pour accélérer la convergence des réseaux de neurones ayant un nombre de neurones cachés élevés (grande capacité). La principale application visée par ce projet est le « data-mining ». À l'heure actuelle, deux nouvelles approches ont été explorées, soit une variante des réseaux incrémentaux et soit, des réseaux avec optimisation circulaire des paramètres. Les performances de ces techniques seront présentées et comparées à celles des réseaux de neurones conventionnels sur un ensemble de données de reconnaissance de caractères[9].**

Mots clés- **Réseaux de neurones, optimisation, réseaux incrémentaux, optimisation circulaire**

I. Introduction

Cet article présente les résultats de l'exploration de nouvelles techniques ayant pour but d'accélérer la convergence des réseaux de neurones ayant un nombre de neurones cachés élevés (grande capacité). La capacité d'un réseau de neurones est reliée à son potentiel d'extraire des relations complexes. La principale application visée par ce projet est le « data-mining », une catégorie de problème caractérisée par la complexité des relations à extraire. À l'heure actuelle, deux nouvelles approches ont été explorées, soit une variante des réseaux incrémentaux et soit, des réseaux avec optimisation circulaire des paramètres. Les performances de ces techniques seront présentées et comparées à celles des réseaux de neurones conventionnels sur un ensemble de données de reconnaissance de caractères[9]. La figure (1) représente le comportement typique de l'erreur en fonction du temps lorsque le nombre de paramètres à optimiser est élevé. L'apprentissage est très rapide au début mais ralentit très rapidement et devient de plus en plus difficile. Plus la capacité (le nombre de paramètres) du réseau est élevée, plus ce ralentissement apparaît tôt. De façon général, l'apprentissage demeure extrêmement lent rapidement. Une plus grande capacité permet d'avoir de meilleurs modèles et par conséquent de plus faibles taux d'erreur. Nous avons tenté d'accélérer cet apprentissage aux moyens de différentes techniques d'optimisation.

Francis Piéraut est un des membres du Laboratoire Informatique des Systèmes Adaptatifs (LISA[8]) du département d'informatique et de recherche opérationnelle de l'Université de Montréal, Canada. E-mail:pierautf@iro.umontreal.ca, http://www.iro.umontreal.ca/~pierautf/

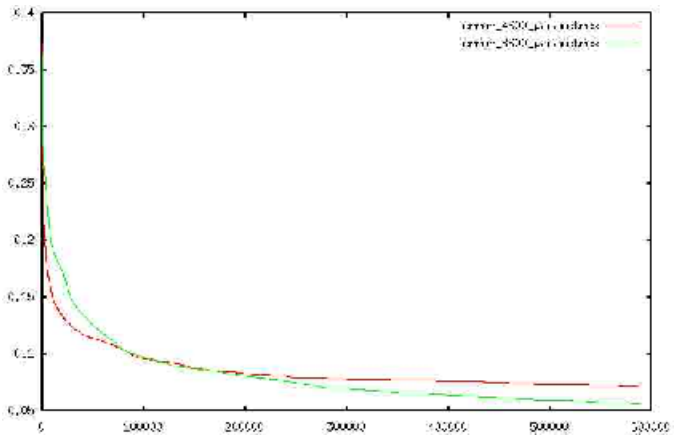


Fig. 1. Erreur de classification en fonction du temps sur la base de donnée « letters [9]» avec 26 classes. Nous avons tracé cette erreur pour 4300 et 8600 paramètres.

II. Différentes approches possibles

Les trois principales approches pour s'attaquer à ce problème, sont:

- Réduire le nombre de paramètres
- Optimiser une partie des paramètres
- Extrapoler les valeurs des paramètres

A. Réduction du nombre de paramètres

La technique la plus intuitive est de réduire le problème en sous-problèmes moins complexes. Ceci permet de réduire le nombre de paramètres et d'éliminer les problèmes observés lorsque la capacité est très grande. Cette technique porte aussi le nom de « diviser pour régner ». Plusieurs approches utilisent directement ou indirectement cette technique dont les mixtures d'experts[5], le « boosting[4] » et les systèmes incrémentaux. Avec une mixture d'experts, on spécialise chaque expert sur une sous région de l'espace des caractéristiques. Avec cette technique, chaque expert possède moins de paramètres mais l'inconvénient est qu'elle nécessite une plus grande quantité d'exemples afin d'assurer une bonne généralisation. En ce qui concerne le « boosting », l'approche consiste à entraîner un réseau de faible capacité et d'utiliser les données qui n'ont pas bien été apprises par un autre réseau. Une récursivité est appliquée tant qu'on le désire. La pondération des résultats, des différents réseaux spécialisés, permet de mieux généraliser et de réduire le nombre de paramètres de chaque réseau. En ce qui concerne les réseaux

incrémentaux, l'approche consiste à commencer avec un réseau de faible capacité puis d'y ajouter des paramètres lorsque l'erreur d'apprentissage ne s'améliore plus. Cette approche permet de minimiser le nombre de paramètres à optimiser à chaque itération. Le « boosting » et les mixtures sont deux techniques qui sont assez bien étudiées à l'heure actuelle. Il existe plusieurs variantes de réseaux de neurones incrémentaux dont plusieurs n'ont jamais été expérimentées. Nous avons donc expérimenté deux nouvelles approches soit :

- Réseau incrémental avec optimisation de tous les paramètres où le nombre initial de neurones est égal à la taille des bloques.
- Réseau incrémental avec optimisation de tous les paramètres où le nombre initial de neurones est N fois la taille des bloques.

B. *Optimisation d'une partie des paramètres*

Cette approche consiste à n'optimiser que certains paramètres afin de concentrer l'optimisation sur les paramètres les plus importants ou à forcer la spécialisation de ces paramètres. Ceci permet de réduire le temps de calcul par l'élimination des calculs reliés à l'optimisation des paramètres de moindres importances ou les paramètres non sélectionnés.
Nous avons expérimenté deux variantes de système incrémentaux avec optimisation d'une partie des paramètres, soit :

- Réseau incrémental avec optimisation des paramètres ajoutés à chaque itération.
- Réseau avec optimisation circulaire par bloque des paramètres lorsque l'erreur ne s'améliore plus d'un certain facteur.

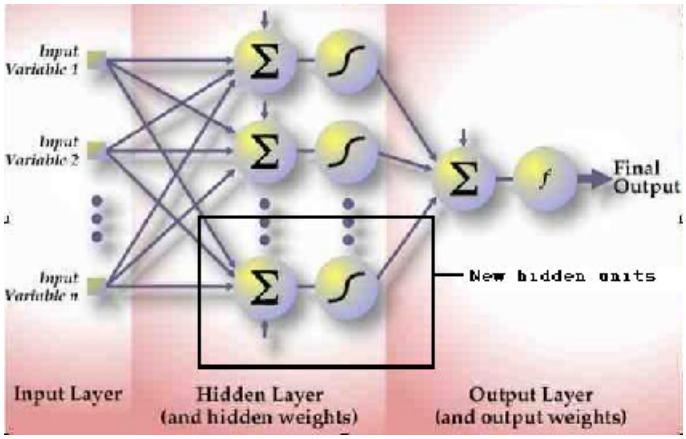


Fig. 2 : Architecture d'un réseau de neurones après l'ajout d'un neurone caché.

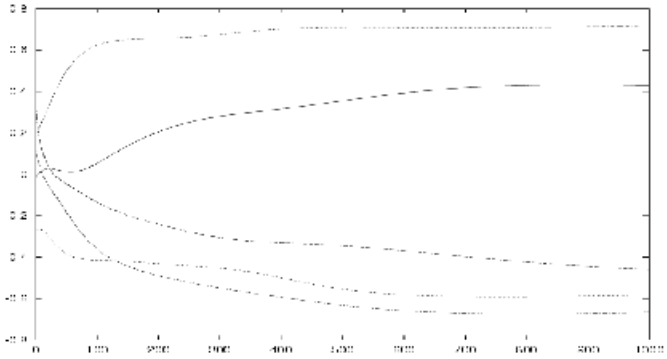
1) *Optimisation circulaire par bloque*

L'optimisation circulaire consiste à rétro-propager l'erreur sur les paramètres du sous bloque suivant. Après avoir choisi séquentiellement chaque bloque, nous continuons en sélectionnant le premier bloque, d'où le nom d'optimisation circulaire.

C. *Extrapolation des valeurs des paramètres*

Lors des recherches, nous avons observé l'évolution des valeurs des paramètres. Nous avons constaté que ceux-ci ne varient que très peu et qu'il serait intéressant d'exploiter cette propriété en extrapolant ces valeurs. Cette approximation permettrait de réduire considérablement le temps de calcul. Cette approche n'a pas été expérimentée.

.



D. *Algorithme « cascade-correlation[1]»*

Les dernières recherches concluantes sur les réseaux incrémentaux avec optimisation d'une partie des paramètres ont été réalisées en 1991 sous le nom de « cascade-correlation ».

E. *Explications*

Les auteurs, présentent deux raisons pour lesquelles l'apprentissage est si lent.

1. « step size problem » : Dans l'algorithme standard, on n'utilise que l'information de la première dérivée. Sachant que lorsque le « learning rate » est trop grand, l'algorithme divergera et sachant que l'on utilise que l'information de la dérivée première, on ne peut avoir le « learning rate » optimale. Par conséquent, le nombre d'itérations doit être grand car nous choisirons un « learning rate » sous optimal.

2. « moving target problem » : Durant l'apprentissage en utilisant la technique de "backpropagation", les poids ne tiennent pas compte des changements des autres paramètres. Le problème est que les poids essaient d'apprendre un problème qui change constamment dû aux autres poids. Ceci explique le temps important avant lequel les poids se spécialisent de façon constante sur certaines caractéristiques. Ce qui explique le comportement observé en pratique, la vitesse d'apprentissage décroît de façon significative plus le nombre de neurones cachés est important. Une manifestation du "moving target problem" est ce qu'on appelle l'effet de troupeau. Supposons que nous ayons deux classes, chaque unité décide de façon indépendante sur quel problème elle va réduire l'erreur.

1) Description de l'algorithme « Cascade-correlation »

Les deux idées principales de cet algorithme sont l'architecture en cascade et une maximisation de la corrélation entre les nouveaux neurones et l'erreur résiduelle. L'architecture en cascade permet d'ajouter récursivement un neurone lorsque le poids du neurone courant ne change plus. L'algorithme d'apprentissage permet de maximiser la corrélation entre les nouveaux neurones et l'erreur résiduelle.

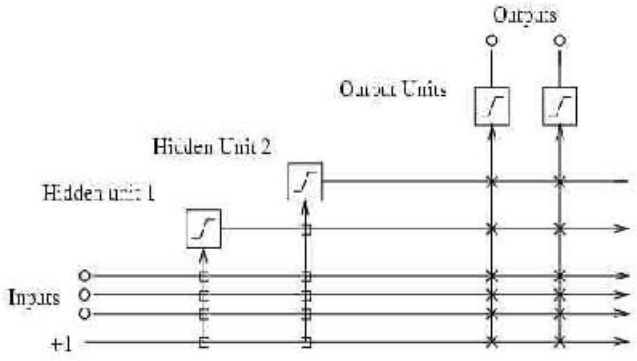


Fig. 3. Architecture (cascade-correlation) après deux neurones cachés ajoutés.

Solutions

Les auteurs proposent une solution à chacun de ces problèmes:

1. « step size problem » : Utiliser les informations des dérivées supérieures. La technique la plus efficace semble être QuickProp. Il est important de mentionner que cette technique calcule un « learning rate » pour chacun des paramètres.
2. « moving target problem » : Il ne faut permettre qu'à un seul groupe d'unités de changer à chaque itération. « Cascade correlation » utilise la version la plus restrictive de cette idée, soit d'optimiser un seul neurone à la fois.

III. RÉSULTATS

A. *Réduction du nombre de paramètres*

1) Voici une comparaison de l'erreur en fonction du temps entre un réseau conventionnel de 100 neurones cachés et un réseau incrémental avec optimisation de tous les paramètres où le nombre initial de neurones est égal à la taille des bloques, soit 10 unités cachées.

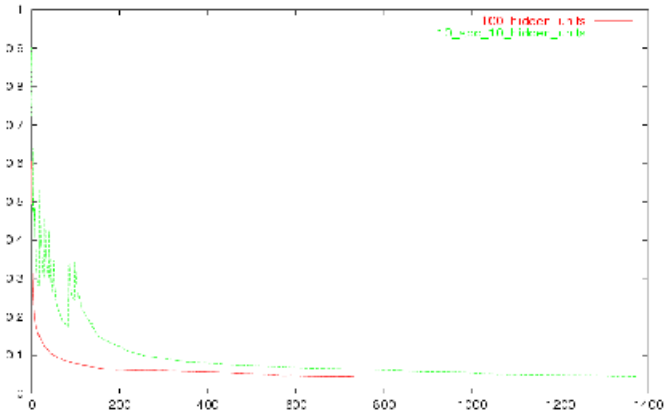


Fig. 3 Erreur de classification en fonction du temps sur la base de donnée « letters » avec 26 classes. Nous avons tracé cette erreur pour un réseau de neurones de 100 neurones cachés et un réseau incrémental (10 + i *10 neurones, max=100).

Nous constatons que cette approche n'est pas concluante, l'erreur reste toujours supérieure à celle d'un réseau conventionnel. L'ajout de neurones semble déstabiliser le réseau et le faible nombre de neurones, au départ, semble être la cause de ce comportement. Ces résultats nous ont amené à faire l'expérience subséquente.

2) Voici une comparaison de l'erreur en fonction du temps entre un réseau conventionnel de 100 neurones cachés et un réseau incrémental avec optimisation de tous les paramètres où le nombre initial de neurones est 10 fois la taille des bloques (5 unités cachées) et où le système incrémental est activé après 50 itérations.

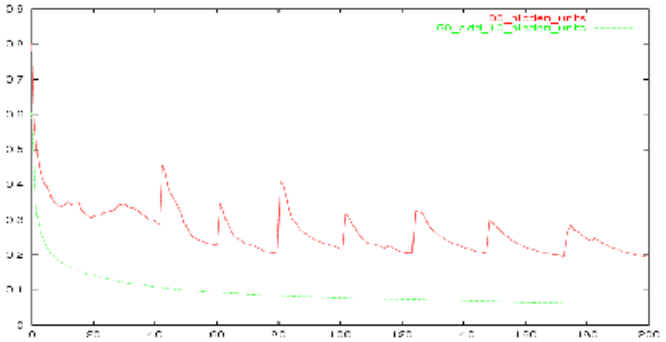


Fig. 4 Erreur de classification en fonction du temps sur la base de donnée « letters » avec 26 classes. Nous avons tracé cette erreur pour un réseau de neurones de 100 neurones cachés et un réseau incrémental (50 + i * 10 neurones, max=100).

Cette approche n'est pas tellement plus concluante que la précédente et nous amène à croire que l'optimisation de tous les paramètres après avoir ajouté des neurones n'est pas une bonne idée. Une autre explication pourrait être que la proportion du nombre de neurones ajoutés par rapport au nombre de neurones est trop importante et que cela a pour conséquence de déstabiliser le réseau. Suite à ces résultats, nous avons délaissé cette approche.

B. *Optimisation d'une partie des paramètres*

*1) Voici une comparaison de l'erreur en fonction du temps entre un réseau conventionnel de 100 neurones cachés et un réseau incrémental avec optimisation des paramètres ajoutés à chaque itération (10 neurones + i * 10 neurones).*

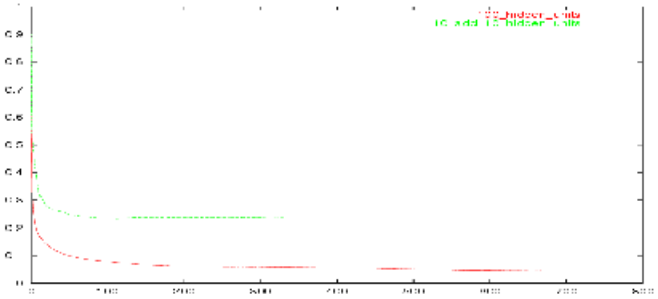
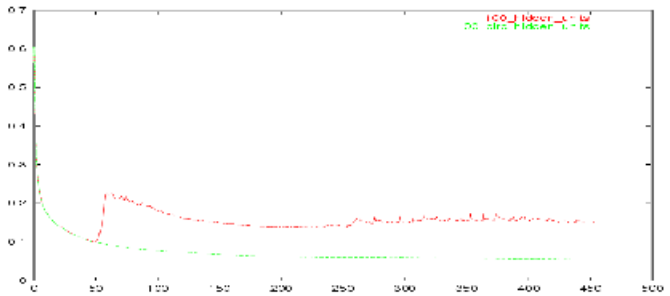


Fig. 4 Erreur de classification en fonction du temps sur la base de donnée « letters » avec 26 classes. Nous avons tracé cette erreur pour un réseau de neurones de 100 neurones cachés et un réseau incrémental (10 + i * 10 neurones, max=1000).

Avec cette approche, l'erreur atteint un plateau malgré le fait que la capacité augmente. Ce comportement s'explique par le fait que les nouveaux paramètres ne sont pas en mesure de compenser pour les paramètres immuables.

2) Voici une comparaison de l'erreur en fonction du temps entre un réseau conventionnel de 100 neurones cachés et un réseau avec optimisation circulaire des paramètres lorsque l'erreur ne s'améliore plus d'un certain facteur.



Dans l'implantation actuelle, nous rétro-propageons l'erreur globale sur un sous-ensemble des paramètres. Malgré le fait que ces résultats semblent être catastrophiques, il faut mentionner qu'ils nous amènent à nous demander si le fait de rétro-propager l'erreur globale (MSE[1]) est approprié ou que le choix des paramètres par l'optimisation circulaire est adéquat.

IV. Travaux futurs

L'approche consistant à optimiser une partie des paramètres n'a pas été assez approfondit et l'extrapolation des valeurs des paramètres n'a pas été expérimentée. Tel que mentionné dans la section précédente, il serait intéressant de faire une étude du meilleur choix d'erreur à rétro-propager sur un sous bloque de paramètres et du critère de choix des paramètres. Il est clair que le choix au moyen de l'optimisation circulaire ne semble pas être la bonne approche.

V. Conclusion

Nous avons proposé 3 nouvelles approches pour accélérer la convergence des réseaux de neurones ayant un nombre de neurones cachés élevés. À l'heure actuelle, seul les deux premières approches ont été expérimentées soit, une variante des réseaux incrémentaux et soit, des réseaux avec optimisation circulaire des paramètres. Nos expérimentations montrent que ces deux approches ne répondent pas à leur objectif et donnent de pires résultats que les réseaux de neurones conventionnels. Tel que mentionné dans la section « travaux futurs », l'optimisation d'une partie des paramètres n'a pas encore été approfondie suffisamment et l'extrapolation des valeurs des paramètres n'a pas été expérimentée. Ces travaux m'ont permis de développer une librairie de réseaux de neurones spécialisée pour les expérimentations et celle-ci me permettra d'explorer très facilement de nouvelles approches.

VI. Remerciements

Je voudrais remercier mon directeur de recherche, Jean-Jules Brault, mon co-directeur Yoshua Bengio, mon collaborateur Gilles Caporossi et tous les membres du LISA[9].

VII. Références

[1] "Cascade-Correlation Learning Architecture", Scott E. Fahlman and Christian Lebiere August 1991. 13 pages

[2] "On the optimality of incremental neural network", Ron Meir and Vitaly Maioro, NIPS volume 11, 1998 (http://nips.djvuzone.org) 6 pages

[3] "Learning Sequential Task by Incrementally Adding Higher Orders", Mark Ring

[4] "Boosting Neural Networks", H. Schwenk and Y. Bengio, Neural Computation, vol. 12, no. 8, pp. 1869--1887, 2000. 19 pages

[5] "Parallel Mixture of SVMs for Very Large Scale Problem," Ronan Collobert, Samy Bengio, and Yoshua Bengio, Neural Computation, 2002. 10 pages.

[6] "The sample complexity of pattern classification with neural networks: the size of the weights os more important than the size of the network", Peter L. Bartlet April 1997

[7] "Successes and failures of backpropagation: a theorical investigation", P. Fransconi, M. Gori, and A. Tesi

[8] LISA, Laboratoire d'Informatique des Systèmes Adaptatifs http://www.iro.umontreal.ca/~lisa/

[9] letters, base de données de reconnaissance de caractères (26 classes, 16 entrées, 20000 exemples) http://www1.ics.uci.edu/~mlearn/MLRepository.html

VIII. Biographie

Francis Piéraut est diplômé de l'école polytechnique de Montréal en décembre 99, en génie informatique. Il a fait ses stages dans le groupe de recherche Microcell-Labs (algorithme d'apprentissage) et chez Locus-Dialgue (reconnaissance de la parole). Il fait présentement sa maîtrise en collaboration avec le groupe du LISA[8]

[1] Mean square error