Zeitschrift: Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association suisse des électriciens, de l'Association des entreprises électriques suisses

Herausgeber: Schweizerischer Elektrotechnischer Verein ; Verband Schweizerischer Elektrizitätsunternehmen

Band: 83 (1992)

Heft: 5

Artikel: Simulation et réseaux de neurones : un environnement de programmation pour l'élaboration d'algorithmes d'apprentissage

Autor: Mayoraz, Eddy / Guenin, Bertrand

DOI: https://doi.org/10.5169/seals-902801

Simulation et réseaux de neurones

Un environnement de programmation pour l'élaboration d'algorithmes d'apprentissage

Eddy Mayoraz et Bertrand Guenin

Cet article présente un environnement de programmation pour la simulation de réseaux de neurones artificiels dont l'architecture est soit fixée a priori, soit modifiée de manière dynamique. L'accent est mis sur la modularité (approche orientée objet), la transparence (interaction aisée à tous les niveaux du logiciel), la convivialité (utilisation par des menus interactifs ou depuis Mathematica). Cet outil est particulièrement adapté à la conception, à la mise au point et au test de nouvelles stratégies d'apprentissage.

Es wird eine Programmierumgebung zur Simulation von neuronalen Netzwerken mit fixer oder auch dynamisch modifizierbarer Architektur beschrieben. Bei der Entwicklung wurde der Akzent auf Modularität (objektorientierter Ansatz), Transparenz (interaktiver Zugriff auf alle Softwareniveaus) und Kompatibilität (Aufruf aus interaktiven Menüs oder aus dem Mathematica-Anwendungsprogramm) gelegt. Das Werkzeug ist speziell für den Entwurf, die Justierung und den Test von neuen Lernstrategien geeignet.

Adresse des auteurs

Eddy Mayoraz, ing. dipl. math. EPFL, et *Bertrand Guenin*, ing. dipl. info. EPFL, Département de Mathématiques, Recherche Opérationnelle, Ecole Polytechnique Fédérale de Lausanne, 1015 Lausanne.

Les réseaux de neurones artificiels sont proposés aujourd'hui comme solutions alternatives pour une grande variété de problèmes. Quelle que soit l'architecture (bouclée, en couches, etc.), la fonction de transfert des neurones (linéaire à seuil, sigmoïdale, etc.) et le mode d'adaptation (supervisé ou non), l'efficacité de ces modèles dépend fortement du processus d'apprentissage utilisé. Afin de surmonter des problèmes tels que l'optimisation de la taille des réseaux et la lenteur de convergence des algorithmes classiques (rétropropagation du gradient, algorithme de Kohonen), des recherches ont été récemment entamées pour concevoir des processus d'apprentissage, adaptant non seulement les poids liés aux connexions synaptiques mais également l'architecture du réseau (nombre de neurones et leurs interconnexions) [1; 2].

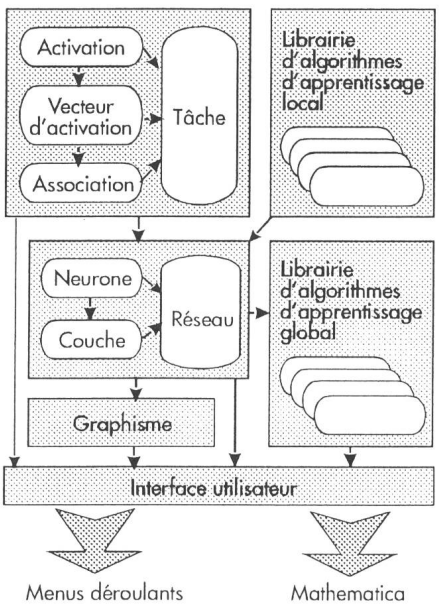

Figure 1 Structure générale de l'environnement

Il existe sur le marché du logiciel informatique un foisonnement de simulateurs de réseaux de neurones artificiels dédiés à l'enseignement ou à l'expérimentation. Dans la plupart de ces produits, les méthodes d'apprentissage font partie d'une boîte noire sur laquelle l'on ne peut agir que par l'intermédiaire de quelques paramètres de réglage. Dans le meilleur des cas l'utilisateur a accès au cœur de ces méthodes mais ne peut y apporter que des modifications limitées, de plus ces produits ne sont pas adaptables à des processus d'apprentissage modulant l'architecture.

L'environnement de programmation présenté dans cet article permet la simulation de réseaux de neurones d'architectures diverses, statiques ou dynamiques, et est spécialement dédié à l'étude et à la comparaison de nouveaux algorithmes d'apprentissage.

Méthode de conception

Le développement d'un tel environnement de programmation est un travail de longue haleine, incluant la participation de nombreuses personnes intervenant dans un intervalle de temps étendu. Une conception orientée objet a été choisie afin de permettre un développement progressif du simulateur et de garantir un découplage optimal entre les divers éléments de ce logiciel.

Langage de programmation

Une importance toute particulière attribuée à la lisibilité, la modifiabilité et la réutilisabilité du code a nécessité le choix d'un langage de programmation permettant un haut niveau d'abstraction. Pour garantir une grande portabilité, ce langage devait être suf-

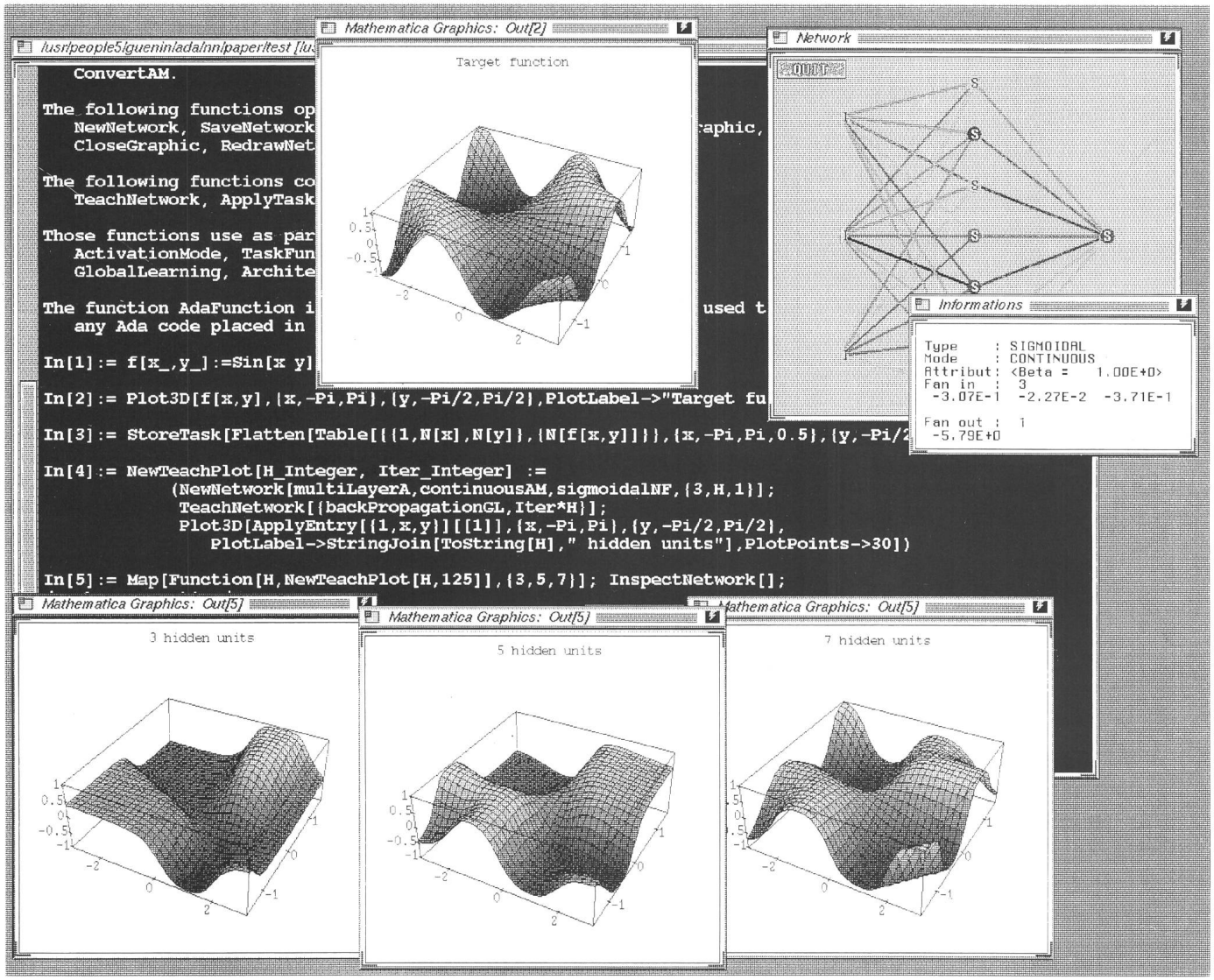

Figure 2 Exemple d'utilisation de l'environnement depuis Mathematica
Ces cinq lignes de commandes illustrent une interpolation de fonction à l'aide de réseaux de neurones en couches avec un nombre variable d'unités cachées. Les deux premières commandes (In[1], In[2]) définissent et affichent dans la fenêtre supérieure gauche la fonction *f* à approximer. L'instruction In[3] calcule un échantillonnage de la fonction *f*. La suivante définit une procédure qui regroupe la construction d'un réseau avec *H* unités cachées; son apprentissage avec une méthode de rétropropagation évoluée et l'affichage de l'approximation de *f* obtenue par le réseau. La dernière commande appelle cette procédure pour différentes valeurs de *H* et affiche le réseau final dans la fenêtre supérieure droite. Les trois approximations obtenues sont affichées dans les fenêtres du bas.

fisamment répandu; Ada a donc tout naturellement été retenu [3]. De plus, le choix d'un langage fournissant des outils pour la programmation concurrente facilite la simulation de réseaux dont la mise à jour des neurones est asynchrone.

Structure

Cet environnement se compose de plus de 60 modules répartis en 8 librairies et fournissant près de 900 routines. Il est clair qu'une description détaillée de ce logiciel ne rentre pas dans le cadre d'un tel article. Dans la structure générale du simulateur (fig. 1) on distingue deux types d'objets principaux, les *tâches* et les *réseaux*, chacun d'eux constitué d'objets plus simples tels que des associations ou des neurones. L'environnement permet la manipulation simultanée d'un grand nombre de ces objets.

Une *activation* caractérise l'état d'un neurone. Suivant la nature de la fonction de transfert, celle-ci peut prendre des valeurs continues, discrètes ou encore binaires. Un vecteur d'activation représente l'état d'un réseau en décrivant l'activité de chacun de ses neurones. Dans un contexte d'apprentissage supervisé, l'on désire que le réseau associe à un ensemble d'états initiaux choisis des états finaux correspondants. On peut donc décrire un tel problème par une *tâche* définie comme un ensemble d'associations (couple de vecteurs d'activations). Une approche orientée objet a été choisie pour représenter les différentes entités précitées, ce qui favorise leur gestion dynamique.

L'objet *neurone* met à disposition de l'utilisateur des outils permettant entre autres d'établir ou de supprimer des connexions avec d'autres neurones; de mettre à jour les poids synaptiques par apprentissage et d'évaluer la fonction de transfert. Il existe différents types de neurones caractérisés par la nature de leur activation, le type de leurs connexions et la

fonction de transfert utilisée. L'objet *réseau* est une liste dynamique de couches, elles-mêmes listes dynamiques de neurones. La notion de couche n'est qu'une structure intermédiaire entre le neurone et le réseau; elle facilite l'expression d'actions parallèles mais n'impose aucune restriction sur l'architecture réalisable.

Deux librairies extensibles, l'une regroupant des algorithmes d'apprentissage local et l'autre des méthodes d'apprentissage global, mettent à disposition de l'utilisateur une foule d'algorithmes permettant de réaliser d'une part l'apprentissage d'une sous-tâche à l'aide d'un neurone et d'autre part l'apprentissage d'une tâche à l'aide d'un réseau.

Graphisme

La visualisation graphique des réseaux fournit une information qualitative immédiate. Les différentes grandeurs numériques (p.ex. poids synaptique, seuils, activations) caractérisant le réseau sont symbolisées par des couleurs et leurs valeurs exactes peuvent être obtenues en cliquant sur le neurone considéré. Ce module est l'unique élément du logiciel utilisant la librairie graphique GL disponible sur les stations Silicon Graphics, il n'est donc pas portable sur des systèmes non munis de cette librairie [4].

Interface utilisateur

La totalité des éléments du simulateur peuvent être utilisés par l'intermédiaire d'une interface composée de menus déroulant, spécialement adaptés pour des essais ponctuels ou des démonstrations. Cependant, lors d'une étude d'algorithmes, la phase de développement de ceux-ci est inévitablement suivie d'une phase de tests souvent fastidieuse, caractérisée par les étapes suivantes:

- expérimentations répétées des algorithmes pour différents échantillons de données,
- gestion d'une grande masse de résultats provenant des expérimentations,
- traitement de ces résultats pour en extraire l'information pertinente à l'aide d'outils statistiques ou graphiques.

Afin d'optimiser ce travail de test, une interface entre le simulateur et le logiciel Mathematica a été réalisée [5]. Elle se présente comme une extension du langage de Mathematica comprenant une vingtaine de routines de bases qui permettent la gestion de diverses tâches et différents réseaux. Le langage de programmation interprété (inspiré de Lisp) intégré à Mathematica permet une mise en œuvre aisée des programmes de tests; et leurs résultats produits sous forme d'objets Mathematica (listes) peuvent être traités directement par les puissants outils analytiques et graphiques de ce logiciel. La figure 2 illustre un petit exemple d'utilisation de l'environnement depuis Mathematica.

Conclusion

Cet article présente un environnement de simulation de réseaux de neurones. Ses originalités résident essentiellement dans sa possibilité de modéliser aussi bien des architectures statiques que dynamiques et dans son utilisation depuis Mathematica. Cet outil de recherche orienté vers l'élaboration et l'expérimentation de nouveaux modèles comble une lacune importante dans le domaine des simulateurs de réseaux de neurones artificiels dont la plupart sont avant tout à vocation didactique.

Remerciement

Les auteurs tiennent à remercier E. Amaldi pour ses nombreuses suggestions quant à la conception du logiciel ainsi que R. P. Hüsler et S. Gerber pour leur contribution au développement informatique. Ce travail a été en partie réalisé par des travaux de semestres et de diplôme d'étudiants de l'Ecole Polytechnique Fédérale de Lausanne et a été supervisé grâce au support du Fonds National Suisse de la Recherche Scientifique (n° 20-5637.88).

Bibliographie

[1] *Mézard M.* and *J.-P. Nadal*: Learning in feedforward layered networks: the tiling algorithm. J. Phys. A: Math. Gen. (1989) pp. 2191–2203.

[2] *Frean M.*: The Upstart algorithm: A method for constructing and training feedforward neural networks. Neural Computation, 2(1990)2, pp. 198–209.

[3] *United States DoD*: Reference Manual for the Ada Programming Language, ANSI/MIL-STD-1815A, 1983.

[4] *Graphics Library*: Reference Manual, Ada Edition, IRIS-4D Series, 1990.

[5] *S. Wolfram*: Mathematica: a System For Doing Mathematics by Computer. Addison-Wesley, 1988.

IBZ

Schulen für Technik und Informatik

Im Zuge der Erweiterung unserer Engagements im Bereich der technischen Weiterbildung wie auch zur Neubesetzung von Vakanzen in unserem Lehrkörper suchen wir laufend neue Dozenten. Gut ausgewiesenen Ingenieuren und Technikern mit Freude an der Wissensvermittlung und am Umgang mit Menschen bietet sich damit Gelegenheit zu einer

nebenamtlichen Lehrtätigkeit

an unseren Schulen in Zürich, Bern, Basel, Frauenfeld, Sursee und Thun. Die IBZ-Schulen für Technik und Informatik zählen zu den führenden schweizerischen Privatinstituten für berufsbegleitende Aus- und Weiterbildung.

Für die IBZ-Schulen in Bern und Zürich suchen wir für den Fachbereich Elektrotechnik-Meisterkurse

Elektro-Ingenieure HTL

und

Eidg. dipl. Elektro-Installateure/ Kontrolleure

denen wir die Möglichkeit praxisbezogener Wissensvermittlung in den Fächern Elektrotechnik, Mathematik, Messkunde, Schemakenntnisse, Telefonie und Vorschriften bieten.

Im Fachbereich Elektrotechnik und Elektronik bietet sich an den IBZ-Schulen von Zürich und Bern

Elektro-Ingenieuren HTL

Gelegenheit zur nebenberuflichen Wissensvermittlung in den Fächern Bauelemente, Digitaltechnik, Messkunde, Regelungstechnik, Projektierung, Elektrotechnik, SPS, CAE und Programmiertechnik (Programmierung in Pascal).

Initiative, ausgewiesene Fachleute, die diese Herausforderung annehmen und junge, aktive Berufsleute in deren beruflichem Weiterkommen unterstützen wollen, werden gebeten, sich mit unserem Herrn H.P. Ruggli in Verbindung zu setzen.

IBZ Schulen für Technik und Informatik Brugg AG
Zentralsekretariat, Wildischachen, 5200 Brugg
Telefon 056/41 46 47, Fax 056/41 48 21

D2/92