Une contrainte globale perceptron multicouche pour l'utilisation de modèles de régression non linéaire en programmation par contraintes

Pierre-Alain Yvars[1], Martin Ghienne[1], Laurent Zimmer[2]
[1] ISAE-Supméca, Quartz, EA 7393, France
{pierre-alain.yvars,martin.ghienne}@isae-supmeca.f
[2] Dassault-Aviation, Direction de la prospective, France
laurent.zimmer@dassault-aviation.com

Mots-clés *: perceptron multicouche, contrainte globale, régression non linéaire, programmation par contraintes.*

1 Introduction

La programmation par contraintes a montré son efficacité dans de nombreux domaines tant pour la résolution de problèmes discrets que continus. A partir du moment où il est possible d'exprimer un problème à l'aide d'un ensemble de variables, d'un ensemble de domaines de définition pour ces variables et d'un ensemble de relations ou contraintes sur les variables, les outils de programmation par contraintes proposent des méthodes efficaces de résolution de ce type de problème. Les contraintes peuvent être formulées sous forme d'équations ou d'inéquations, linéaires ou non linéaires ou bien sous forme de contraintes globales. Nous nous intéressons ici à la possibilité d'intégrer dans un ensemble de contraintes un réseau de neurones de type perceptron multicouche (MLP). Le MLP est utilisé en régression et peut-être vu comme une boîte noire entrainée sur des données disponibles. [1] et [2] ont étudié la prise en compte de réseaux de neurones dans un réseau CSP (Constraint Satisfaction Problem) d'un point de vue conceptuel et applicatif mais pas sous la forme d'une contrainte globale dédiée à cet effet. A notre connaissance aucune implémentation de ce type n'a été proposée jusqu'ici.

2 Une contrainte globale perceptron multicouche

Un réseau de neurone de type perceptron multicouche utilisé en régression est une structure totalement algébrique. En notant p la dimension de la couche d'entrée , q la dimension de la couche de sortie, x_i la i^{ème} variables d'entrée, y_i la i^{ème} variable de sortie, K le nombre de couches cachées, 0 la couche d'entrée et $K+1$ la couche de sortie, on a la définition sous forme de suite récurrente:

$$\forall k \in \{1,..,K\}\ Z^k = f(W^k \times Z^{k-1} + B^k) \qquad (1)$$

Où W^k et B^k sont respectivement la matrice de poids et le vecteur de biais de la couche k et f la fonction d'activation. En notant m_k et n_k le nombre de neurones de la k^{ème} couche et le nombre de neurones de la couche amont, W^k et B^k s'écrivent :

$$\forall k \in \{1,..,K\} \quad W^k = \begin{pmatrix} w_{11}^k & \cdots & w_{1n_k}^k \\ \vdots & \ddots & \vdots \\ w_{m_k1}^k & \cdots & w_{m_kn_k}^k \end{pmatrix},\ B^k = \begin{pmatrix} b_1^k \\ \vdots \\ b_{m_k}^k \end{pmatrix},\ Z^k = \begin{pmatrix} z_1^k \\ \vdots \\ z_{n_k}^k \end{pmatrix} \qquad (2)$$

Les couches d'entrée (Z^0) et de sortie (Z^{K+1}) s'expriment alors comme suit :

$$Z^0 = \begin{pmatrix} x_1 \\ \vdots \\ x_p \end{pmatrix} \; et \qquad Z^{K+1} = \begin{pmatrix} y_1 \\ \vdots \\ y_q \end{pmatrix} = W^{K+1} \times Z^K + B^{K+1} \qquad (3)$$

Un MLP utilisé en régression non linéaire est donc une fonction de $\mathbb{R}^p$ dans $\mathbb{R}^q$, combinaison d'opérateurs d'addition de multiplication et de fonctions d'activation possédant une description analytique complète. Ainsi en notant $\mathbb{IR}$ l'ensemble des intervalles de $\mathbb{R}$, un MLP peut être transformé en une fonction de $\mathbb{IR}^p$ dans $\mathbb{IR}^q$. Les définitions formelles précédentes restent valables en remplaçant, Z^k par $[Z^k]$, vecteur d'intervalles tel que $[Z^k] = \begin{pmatrix} [z_1^k] \\ \vdots \\ [z_{n_k}^k] \end{pmatrix}$, $[Z^0] = \begin{pmatrix} [x_1] \\ \vdots \\ [x_p] \end{pmatrix}$ et $[Z^{K+1}] = \begin{pmatrix} [y_1] \\ \vdots \\ [y_q] \end{pmatrix}$.

Nous proposons de définir la contrainte globale MLP telle que :

$$MLP([Z^0], [Z^{K+1}], W, B, f) \text{ avec } W = (W^1, \dots, W^{K+1}) \; et \; B = (B^1, \dots, B^{K+1}) \qquad (4)$$

W, B et f représentent les constantes associées aux poids, biais et fonction d'activation alors que $[Z^0] \; et \; [Z^{K+1}]$ sont les variables ensemblistes d'entrées et de sorties de la contrainte.

A la création de la contrainte, un ensemble d'arbres syntaxiques dont les feuilles sont représentées par les $[x_i]$ est généré. Il forme un graphe acyclique direct (DAG) et permet d'évaluer efficacement les $[y_i]$ en fonction des $[x_i]$. Un algorithme de contraction de type HC4Rev [3] suivi d'une méthode de 3B Consistance [4] est alors applicable sur la contrainte MLP et permet de réduire les $[x_i]$ et les $[y_i]$.

La contrainte MLP a été implémentée dans le solveur de programmation par contraintes en domaines mixtes de l'environnement DEPS Studio [5] et a été mise en œuvre sur des données issues d'un cas d'étude de synthèse en aéronautique décrit dans [6]. Certaines parties du modèle analytique décrit dans [6] sont obtenues à partir de données expérimentales approximées sous forme de modèles polynomiaux de $\mathbb{R}^n$ dans $\mathbb{R}$ avec $n \in [\![1, 4]\!]$. Sur la base de ces modèles polynomiaux nous avons entrainé plusieurs perceptrons multicouche pour utiliser les poids et biais générés comme paramètres de la contrainte MLP.

3 Conclusions et perspectives

Ce travail représente une première étape vers la possibilité de manipuler des modèles d'apprentissage comme des contraintes au sens de la programmation par contraintes en contractant efficacement les domaines des variables d'un perceptron multicouche via la contrainte MLP que nous avons développée.

Références

[1] Xavier Fischer. *Stratégie de conduite du calcul pour l'aide à la décision en conception mécanique intégrée. Application aux appareils de pression*, Thèse de Doctorat ENSAM, Décembre 2000.

[2] Arnaud Lallouet and A. Legtchenko. Two contributions of constraint programming to machine learning, *ECML 2005, 16th European Conference on Machine Learning*, Porto, Portugal, October 3-7, 2005

[3] Frédéric Benhamou, Frédéric Goualard and Laurent Granvilliers, Jean-François Puget. Revising Hull and Box consistency,*16th International Conference on Logic Programming, 1993*

[4] Olivier Lhomme. Consistency techniques for numeric CSPs. *International Joint Conference on Artificial Intelligence*, pp 232–238, Chambéry(France), 1993.

[5] Pierre-Alain Yvars and Laurent Zimmer, DEPS Studio : Un environnement intégré de modélisation et de résolution pour la synthèse de système. *20èmes journées Approches Formelles dans l'Assistance au Développement de Logiciels – AFADL*, 2021

[6] J. Sobieszczanski, J.S. Agte and R.R. Sandusky. *Bi-Level Integrated System Synthesis (BLISS)*, National Aeronautic and Space administration (NASA), Langley Research Center, August 1998