Approximation d'un Réseau de Neurones ReLU interprétable par une Régression Logistique à Noyau

Marie Guyomard, Susana Barbosa, Lionel Fillatre

Approximation d'un Réseau de Neurones ReLU interprétable par une Régression Logistique à Noyau

Marie Guyomard[1] & Susana Barbosa[2] & Lionel Fillatre[1]

[1] *Université Côte d'Azur, CNRS, Laboratoire I3S, France*
Emails : guyomard@i3s.unice.fr, lionel.fillatre@i3s.unice.fr
[2] *Université Côte d'Azur, CNRS, Laboratoire IMPC, France*
Email : sudocarmo@gmail.com

Résumé. Cet article s'intéresse à un réseau de neurones interprétable dont la fonction de score se modélise comme une somme additive de fonctions splines univariées. Ce réseau est une extension des modèles interprétables usuels tels que les modèles additifs généralisés, les modèles à base de splines ainsi que les réseaux neuronaux additifs. Nous approchons le réseau de neurones par une régression logistique à noyau afin d'avoir des garanties quant à la convergence du processus d'entraînement. Ainsi, le modèle présenté est explicable puisque d'une part, la règle de décision estimée est interprétable et d'autre part, sa solution est unique conditionnellement à l'échantillon d'apprentissage.

Mots-clés. Régression Logistique à Noyau, Modèles Additifs Généralisés, Réseaux de Neurones ReLU

Abstract. This paper proposes an understandable neural network whose score function is modeled as an additive sum of univariate spline functions. It extends usual understandable models like generative additive models, spline-based models, and neural additive models. An approximation of this neural network by a kernel logistic regression provides garantees on the convergence of the optimization problem. Thus, the proposed method is explainable since on the one hand, the resulted decision rule is interpretable and on the second hand, its resulted estimated parameters are unique for a given training dataset.

Keywords. Kernel Logistic Regression, Generalized Additive Models, ReLU Neural Networks

1 Introduction

Les réseaux de neurones (RNs) sont largement utilisés tant leur performance sur les tâches de régression et de classification est significative (Meijering *et al.*, 2022). L'explicabilité des RNs et la possibilité de certifier leurs résultats sont souvent critiquées, c'est pourquoi ils sont souvent qualifiés de "boîtes noires" (Fel et Vigouroux, 2020). Ces dernières années, de nombreuses études ont été menées sur l'interprétabilité des RNs, mais aussi sur la convergence du processus d'apprentissage et donc l'unicité des paramètres estimés. Un pont rigoureux entre les RNs utilisant la fonction unitaire linéaire rectifiée (RNs ReLU) et l'approximation de fonctions

multidimensionnelles par des splines a été construit dans (Balestriero et Baraniuk, 2018). Les auteurs établissent que les RNs ReLU peuvent être interprétés comme des opérateurs splines partitionnant l'espace d'entrée en polyèdres. Malheureusement, le partitionnement induit par les couches cachées du réseau est trop complexe, les variables d'entrée étant mélangées par une transformation linéaire ou une convolution. D'autres méthodes comme les arbres de décision (DT) (Hastie *et al.*, 2009), les modèles MARS (Multivariate Adaptative Regression Splines) (Friedman, 1991) et les modèles additifs généralisés (GAMs) (Hastie, 2017) partitionnent également l'espace d'entrée. Leurs segmentations ne sont composées que d'orthotopes, les variables descriptives étant seuillées indépendamment les unes des autres et sont donc interprétables. Ces méthodes exploitent un algorithme glouton ("*greedy algorithm*") afin d'itérativement segmenter les variables. Par conséquent, tant l'optimalité du processus d'apprentissage que sa robustesse sont discutables.

Récemment, des études ont été menées dans le but de développer des RNs inspirés de ces méthodes interprétables tout en optimisant un critère global. Dans (Eckle *et al.*, 2019), il est théoriquement démontré qu'un RN peut approximer les modèles MARS, néanmoins aucun algorithme n'est proposé pour entraîner ce RN. Les modèles RNs additifs ("*Neural Additive Models*", NAM) présentés dans (Agarwal *et al.*, 2021) approximent les GAMs. Les auteurs proposent un RN convolutif dont les paramètres sont appris avec une Descente de Gradient Stochastique (DSG) (Boyd et Vandenberghe, 2004) sans aucune garantie de convergence vers une solution optimale.

Il a été démontré dans de très récents papiers, notamment dans (Jacot *et al.*, 2018) qu'il est possible de garantir la convergence du DSG pour un RN entièrement connecté lorsque le nombre de neurones composant sa couche cachée devient arbitrairement large. En effet, en supposant que les paramètres des RNs sont initialisés selon une distribution gaussienne, les RNs deviennent linéaires par rapport à leurs paramètres à mesure que le nombre de neurones composant la couche cachée augmente (Neal, 2012; Liu *et al.*, 2020). Cependant, il a été établit dans (Liu *et al.*, 2020) que cette approximation n'est plus vérifiée lorsque la couche de sortie est non-linéaire, ce qui est inévitable pour les tâches de classification impliquant une activation sigmoïde ou softmax en couche de sortie.

Dans cet article nous présentons trois contributions principales. Premièrement nous introduisons le SATURNN (*Splines Approximation Throught Understandable ReLU Neural Network*) qui est un modèle interprétable combinant les avantages des MARS, GAMs, NAMs et RNs. Le SATURNN est intrinsèquement explicable et son processus d'entraînement implique la minimisation d'un critère global. Deuxièmement, nous établissons une équivalence entre le SATURNN et une Régression Logistique (RL) appliquée à une transformation non-linéaire des variables explicatives. Cette transformation est assimilable à un pré-traitement non-linéaire des données lié à l'architecture du SATURNN. En effet, la RL est appliquée à un modèle linéaire résultant de la linéarisation partielle du SATURNN par rapport à ses paramètres. Troisièmement, nous montrons que cette transformation non-linéaire, dépendante des initialisations du SATURNN peut-être remplacée par une approche à noyau déterministe. Bien que cette approximation ne dépende plus des initialisations du SATURNN, elle découle explicitement de l'architecture du réseau de neurones. L'approximation du SATURNN par une RL à noyau permet d'une part de retrouver une décomposition additive similaire à celle du

SATURNN et d'autre part de garantir la convergence du SATURNN et l'unicité du classifieur obtenu.

Cet article est structuré sous la forme suivante. La Section 2 introduit le modèle SATURNN. La Section 3 démontre l'équivalence entre le SATURNN et la RL appliquée à une transformation non-linéaire des variables descriptives. La Section 4 établit l'approximation du SATURNN par des méthodes à noyau. Dans la Section 5, les performances et l'interprétabilité du SATURNN et de ses méthodes d'approximation sont comparées à des algorithmes de l'état de l'art. Enfin, la Section 6 conclut le papier.

2 La méthode SATURNN

Nous disposons de N couples indépendants et identiquement distribués $\{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ où $x^{(i)} \in \mathbb{R}^d$ est le vecteur des variables explicatives et $y^{(i)} = \{0, 1\}$ est l'étiquette binaire à prédire. Nous supposons que les variables descriptives sont dans une boule ouverte $\mathcal{B}_2^d(0, r)$ telle que $\mathcal{B}_2^d(c, r) := \{x \in \mathbb{R}^d : \|x - c\|_2 \leq r\}$ est une boule dans $\mathbb{R}^d$ centrée en $c \in \mathbb{R}^d$ et de rayon $r > 0$ avec $\|x\|_2^2 = \sum_{i=1}^{d} x_i^2$ la norme euclidienne de $x = (x_1, \ldots, x_d)$.

L'utilisation des RNs (Goodfellow *et al.*, 2016) est très répandue pour les problèmes de classification non-linéaire, de part leur qualité d'approximateurs universels de n'importe quelle fonction (Hornik *et al.*, 1989; Leshno *et al.*, 1993). Considérons un RN ReLU à une couche cachée de p neurones et une sigmoïde en couche de sortie :

$$\Phi^{\mathrm{ReLU(p)}}(x) = \sigma(\psi^{\mathrm{ReLU(p)}}(x)), \tag{1}$$

$$\psi^{\mathrm{ReLU(p)}}(x) = \beta_0 + \sum_{k=1}^{p} \beta_k \phi\left(W_k x + b_k\right), \tag{2}$$

avec $\beta = (\beta_0, \beta_1, \ldots, \beta_p) \in \mathbb{R}^{p+1}$, $b = (b_1, \ldots, b_p) \in \mathbb{R}^p$ et $W_1, \ldots, W_p \in \mathbb{R}^{1\times d}$ les paramètres à estimer. Le RN (1) réalise une classification non-linéaire puisque la sigmoïde $\sigma(t) = 1/(1 + \exp(-t))$ est appliquée à une fonction de score non-linéaire (2). La fonction $\phi(\cdot) = \mathrm{ReLU}(\cdot) = \max\{0, \cdot\}$ est l'activation ReLU (Balestriero et Baraniuk, 2018). Les matrices de poids W_k dans l'équation (2) mélangent toutes les variables descriptives entre elles. De ce fait, la segmentation de l'espace d'entrée en régions obliques est très compliquée, ci ce n'est impossible à interpréter. La Figure 1 (gauche) illustre le partitionnement obtenu par un RN ReLU composé de 10 neurones.

Dans cet article, nous introduisons le modèle SATURNN qui est un RN ReLU pour la classification (1) composé d'une fonction de score spécifique :

$$\psi(x, \theta) = \frac{1}{\sqrt{p}} \left[\beta_0 + \sum_{k=1}^{p} \beta_k \phi(s_k x_{\upsilon(k)} + b_k)\right], \tag{3}$$

où $\theta = [\beta^T, b^T]^T \in \mathbb{R}^{2p+1}$ est le vecteur des paramètres à estimer et x^T est la transposée de x. La matrice de poids W composant les RNs ReLU (2) est un paramètre fixé dans la modélisation du SATURNN, directement inspiré des modèles MARS (Friedman, 1991). Chacun des p

neurones composant la couche cachée $h_k(x) = \phi(s_k x_{\upsilon(k)} + b_k)$ prend en entrée une variable individuelle, plus précisément le k-ème neurone $h_k(x)$ traite $x_{\upsilon(k)}$ où $\upsilon : \{1, \ldots, p\} \mapsto \{1, \ldots, d\}$ est un sélecteur indiquant quelle variable d'entrée est transformée par ce neurone. Puisque la fonction ReLU est non-décroissante, le signe $s_k \in \{-1, 1\}$ précise si $h_k(x)$ est une fonction non-décroissante ou non-croissante de $x_{\upsilon(k)}$. Les paramètres s_k sont fixés et distribués aléatoirement selon une loi de Bernoulli de paramètre $1/2$. De la même manière, la fonction de sélection est aléatoirement initialisée tel que $\upsilon(k)$ prend la valeur $i \in \{1, \ldots, d\}$ avec une probabilité de $1/d$. Ainsi, toutes les variables descriptives sont sélectionnées le même nombre de fois en moyenne. Enfin, le biais b_k indique le noeud, c'est à dire le point à partir duquel $h_k(x)$ devient linéaire : $h_k(x) = s_k x_{\upsilon(k)} + b_k$ quand $s_k x_{\upsilon(k)} > -b_k$. Cette construction de la couche cachée réalise un partitionnement de l'espace d'entrée avec des orthotopes (Fig.1 droite), aboutissant à une règle de décision interprétable. Puisqu'un orthotope est le produit d'intervalles fermés, il est facile de savoir quelles variables sont impliquées dans cette orthotope et quels sont leurs domaines de définition.

Il est intéressant de constater que le SATURNN peut être aparenté à un cas spécial des GAMs (Hastie, 2017). En réécrivant (3), nous obtenons :

$$\psi(x, \theta) = \frac{1}{\sqrt{p}} \left[\beta_0 + \sum_{i=1}^{d} \underbrace{\sum_{1 \leq k \leq p : \upsilon(k) = i} \beta_k \phi\left(s_k x_i + b_k\right)}_{f_i(x_i)} \right], \tag{4}$$

où $f_i(x_i)$ est une fonction de spline univariée de x_i, composée de la somme de fonctions linéaires ReLU.

Contrairement aux NAMs (Agarwal *et al.*, 2021) qui apprennent une combinaison de $d+1$ RNs, le SATURNN estime l'intégralité de la règle de décision avec un seul RN.

$$\hat{\theta}^{\text{SATURNN}} = \underset{\theta \in \mathcal{B}_2^{2p+1}(\theta^{(0)}, R)}{\arg\min} \mathcal{L}^{\text{SATURNN}}(\theta) = \underset{\theta \in \mathcal{B}_2^{2p+1}(\theta^{(0)}, R)}{\arg\min} \frac{1}{N} \sum_{i=1}^{N} L\left(\sigma(\psi(x^{(i)}, \theta)), y^{(i)}\right), \tag{5}$$

avec $L(\cdot)$ la fonction de perte, par exemple la Cross-Entropie Binaire très répandue pour les problèmes de classification binaire (Goodfellow *et al.*, 2016). Comme pour tout processus d'entraînement de RNs, une initialisation des paramètres θ est nécessaire ; $\theta^{(0)} = [\beta^{(0)^T}, b^{(0)^T}]^T$ désigne le vecteur des paramètres initialisés. Les $\beta_k^{(0)}$ satisfont $\beta_k^{(0)} \sim \mathcal{N}(0, 1)$ pour $k = \{0, \ldots, p\}$. Les seuils $b_k^{(0)}$ sont initialisés selon une distribution uniforme sur l'intervalle $[-r, r]$ puisque $|x_i|_2 \leq r : b_k^{(0)} \sim \mathcal{U}[-r, +r]$. Une régularisation ℓ_2 est ajoutée afin de garantir que les paramètres estimés $\hat{\theta}^{\text{SATURNN}}$ après l'entraînement ne soient pas trop éloignés de ceux initialisés, soit à une distance $R > 0$ près. Étant donné que $\psi(x, \theta)$ est non-linéaire à la fois en x et en θ, le problème d'optimisation (5) n'est pas convexe. Ainsi, nous n'avons aucune garantie quant à sa convergence et à l'unicité de ces paramètres estimés $\hat{\theta}^{\text{SATURNN}}$. Dans la prochaine section, nous allons montrer que pour un grand p la fonction de score $\psi(x, \theta)$ devient linéaire par rapport aux paramètres θ et de ce fait le problème d'optimisation (5) devient convexe.

3 Approximation avec une Régression Logistique

Liu *et al.* (2020) ont démontré que la linéarisation globale d'un RN composé d'une couche de sortie Sigmoïde comme défini à l'équation (1) n'est pas possible, le noyau tangent n'étant pas constant. Ainsi, nous linéarisons partiellement le SATURNN à travers la linéarisation de sa fonction de score $\psi(x, \theta)$ par rapport à θ. Dans un second temps, nous établissons l'équivalence entre entraîner le SATURNN et une RL sur sa fonction de score linéarisée. Les résultats théoriques concernant la linéarisation partielle de la fonction de score du SATURNN sont présentés dans (Guyomard *et al.*, 2023). L'approximation par RL découle de ces travaux.

Nous pouvons démontrer que la fonction de score du SATURNN $\psi(x, \theta)$ peut être correctement approximée par le modèle linéaire suivant :

$$\psi^{\text{lin}}(x, \theta, \theta^{(0)}) = \psi(x, \theta^{(0)}) + \nabla_\theta \psi(x, \theta^{(0)})^T (\theta - \theta^{(0)}). \tag{6}$$

Il s'agit d'une linéarisation partielle, bien que la fonction ne dépende plus de θ, elle reste néanmoins non-linéaire en x à travers le gradient $g_0(x) = \nabla_\theta \psi(x, \theta^{(0)})$ de $\psi(x, \theta^{(0)})$ par rapport à θ au point $\theta^{(0)}$. Ainsi, quand le nombre p de neurones est suffisamment grand, le SATURNN devient équivalent à une RL appliquée à la fonction de score linéarisée du SATURNN, définie comme suivant :

$$\delta_{RL}(x, \eta) = \sigma\left(c_0(x) + g_0(x)^T \eta\right) \approx \sigma\left(g_0(x)^T \eta\right), \tag{7}$$

avec $\eta = \theta - \theta^{(0)} \in \mathbb{R}^{2p+1}$ le vecteur de paramètres à apprendre quand on entraîne la RL. $g_0(x)$ peut être considéré comme un pré-traitement non-linéaire des variables d'entrée x. Le terme $c_0(x) = \psi(x, \theta^{(0)})$ est constant par rapport à η. Il dépend seulement des initialisations $\theta^{(0)}$ du RN. Un bref calcul montre que l'espérance de $c_0(x)$ est nulle et sa variance est bornée par $4r^2 + 1/p$. Ainsi, cette variance est négligeable par rapport à $g_0(x)^T \eta$, justifiant l'approximation de la partie droite dans (7). Le vecteur de paramètres η est définit par $\hat{\eta}^{\text{RL}}$ et estimé en minimisant $\mathcal{L}^{\text{RL}}(\eta)$:

$$\hat{\eta}^{\text{RL}} = \underset{\eta \in \mathcal{B}_2^{2p+1}(0,R)}{\arg\min}\ \mathcal{L}^{\text{RL}}(\eta) = \underset{\eta \in \mathcal{B}_2^{2p+1}(0,R)}{\arg\min}\ \frac{1}{N} \sum_{i=1}^{N} L\left(\delta_{RL}(x^{(i)}, \eta), y^{(i)}\right). \tag{8}$$

Puisque $\delta_{RL}(x, \eta)$ dans (7) est un modèle classique de RL estimé par la minimisation de la Cross-Entropie avec une régularisation ℓ_2 sur η, le problème d'optimisation (8) est fortement convexe. Ainsi la convergence est garantie et la solution $\hat{\eta}^{\text{RL}}$ est unique. Quelque soit l'initialisation du SATURNN $\theta^{(0)}$, nous obtenons une solution unique $\hat{\eta}^{\text{RL}}\left(\theta^{(0)}\right)$. Comme établit dans le Théorème suivant, il est équivalent d'optimiser (5) ou (8) quand le nombre de neurones p composant le SATURNN est suffisamment grand.

Théorème 1 (Équivalence entre $\mathcal{L}^{\text{SATURNN}}$ (5) et $\mathcal{L}^{\text{RL}}$ (8))**.**
Soit $\theta^{(0)} = [\beta_0^{(0)}, \ldots, \beta_p^{(0)}, b_1^{(0)}, \ldots, b_p^{(0)}]$ *et* $r, R > 0$ *tel que* $\beta_k^{(0)} \sim \mathcal{N}(0, 1)$ *et* $b_k^{(0)} \sim \mathcal{U}[-r, +r]$. *Soit* $\mathcal{L}^{SATURNN}$ *définie à l'équation (5) et* $\mathcal{L}^{RL}$ *définie à l'équation (8). Nous avons*

$$\sup_{\substack{\theta \in \mathcal{B}_2^{2p+1}(\theta^{(0)}, R) \\ \eta \in \mathcal{B}_2^{2p+1}(0,R)}} \left|\mathcal{L}^{SATURNN}(\theta) - \mathcal{L}^{RL}(\eta)\right| \leq \frac{R^2}{2\sqrt{p}}. \tag{9}$$

La régularisation ℓ_2 ajoutée dans la fonction de coût (5) par la contrainte $\theta \in \mathcal{B}_2^{2p+1}(\theta^{(0)}, R)$ pour une valeur raisonnable de R garantie que l'erreur d'approximation reste négligeable pour un p suffisamment large. Il est important de remarquer que notre RL reste dépendante de $\theta^{(0)}$ par le biais de $g_0(x)$. Dans la prochaine section, nous introduisons une approximation de $\delta_{\text{RL}}(x)$ en utilisant une méthode à noyau totalement indépendante des initialisations $\theta^{(0)}$.

4 Approximation avec une Régression Logistique à Noyau Déterministe

Jacot *et al.* (2018) construisent un pont rigoureux entre les RNs et les méthodes à noyau. Néanmoins, toute cette théorie et celles qui ont suivi (Liu *et al.*, 2020) ne peuvent s'appliquer au SATURNN. En effet, Jacot *et al.* (2018); Neal (2012); Lee *et al.* (2017) démontrent que les RNs dont les paramètres sont initialisés selon une distribution gaussienne sont équivalents à des processus gaussiens lorsque la profondeur des RNs tend à l'infini. De plus, Liu *et al.* (2020) met en avant que le noyau tangent ne reste pas constant quand les RNs sont composés d'une couche de sortie non-linéaire. Dans cette section, nous allons démontrer que malgré son architecture particulière comprenant la matrice de poids fixés, les initialisations des paramètres non-gaussiennes mais aussi sa couche de sortie non-linéaire, le SATURNN peut être correctement approximé par une RL à noyau ("*Kernel Logistic Regression*", KLR). Premièrement nous introduisons un noyau exact dépendant des initialisations $\theta^{(0)}$. Ensuite, nous établissons que ce noyau peut être approximé par un noyau déterministe, c-à-d., indépendant des initialisations, lorsque le nombre de neurones composant la couche cachée est suffisamment grand.

Nous considérons dans cette section le modèle de RL $\delta_{\text{RL}}(x, \eta)$ définie à l'équation (7). D'après le Théorème de Représentation (Schölkopf *et al.*, 2001), le vecteur de paramètres estimé $\hat{\eta}^{\text{RL}}$ dans (8) peut être exprimé comme une combinaison linéaire du vecteur d'entrée, c-à-d., il existe $\{\alpha_j\}_{j=1}^N \subset \mathbb{R}$ tel que $\hat{\eta}^{\text{RL}} = \sum_{j=1}^N \alpha_j g_0(x^{(j)})$. Soit $x, \tilde{x} \in \mathbb{R}^d$, la fonction noyau $\kappa_0(x, \tilde{x}) = g_0(x)^T g_0(\tilde{x})$ est définie par

$$\kappa_0(x, \tilde{x}) = \frac{1}{p}\left[1 + \sum_{k=1}^{p} \phi\left(s_k x_{\upsilon(k)} + b_k^{(0)}\right) \phi\left(s_k \tilde{x}_{\upsilon(k)} + b_k^{(0)}\right) + {\beta_k^{(0)}}^2 \mathbb{1}_{\{s_k x_{\upsilon(k)} + b_k^{(0)} > 0\}} \mathbb{1}_{\{s_k \tilde{x}_{\upsilon(k)} + b_k^{(0)} > 0\}}\right]. \tag{10}$$

Il en découle que l'équation (7) peut se réécrire comme un problème de KLR dont le vecteur d'entrée $K_0(x) \in \mathbb{R}^N$ est définie par $K_0(x) = \left(\kappa_0\left(x^{(1)}, x\right), \ldots, \kappa_0\left(x^{(N)}, x\right)\right)^T$:

$$\delta_{KLR}(x, \alpha) = \sigma\left(\sum_{j=1}^{N} \alpha_j \kappa_0\left(x^{(j)}, x\right)\right) = \sigma\left(K_0(x)^T \alpha\right). \tag{11}$$

Tout comme pour la modélisation $\delta_{\text{LR}}(x, \eta)$, la KLR est appliquée à une transformation non-linéaire des variables descriptives découlant du noyau. Pour $\delta_{\text{LR}}(x, \eta)$ le pré-traitement de chaque échantillon $x \mapsto g_0(x) = \nabla_\theta \psi(x, \theta^{(0)})$ est indépendant de la base de données d'apprentissage. Tandis que quand nous considérons l'approche par noyau $\delta_{\text{KLR}}(x, \alpha)$ dans

(11), la transformation de chaque échantillon dépend directement de la distribution de l'ensemble de la base de données d'entraînement : $x \mapsto \left(\kappa_0(x^{(i)}, x)\right)_{i=1}^N$. De plus, nous savons de part le théorème de représentation que le pré-traitement est équivalent à une transformation optimale, c-à-d, une transformation résultant de l'apprentissage de la RL appliquée à la fonction de score linéarisée (7). Les fonctions de splines univariées apprises par SATURNN, et donc la segmentation univariée, peuvent être retrouvées en calculant $\tilde{\theta}^{\text{KLR}} = \sum_{j=1}^N \hat{\alpha}_j^{\text{KLR}} g_0(x^{(j)}) + \theta^{(0)}$, avec $\hat{\alpha}^{\text{KLR}}$ la solution qui minimise le problème d'optimisation défini par (8) appliqué à $\delta_{KLR}(x, \alpha)$. Puisque la KLR est fondée sur le Théorème de la Représentation appliquée à la RL, les deux modèles sont évidemment équivalents. Par définition, le noyau $\kappa_0(x, \tilde{x})$ dans (10) est toujours dépendant de $\theta^{(0)}$. Cependant, la proposition suivante montre que ce noyau peut être correctement approximé par son espérance.

Proposition 1 (Convergence du noyau)**.**
Soit $\theta^{(0)} = [\beta_0^{(0)}, \ldots, \beta_p^{(0)}, b_1^{(0)}, \ldots, b_p^{(0)}]$ *et* $r, R > 0$ *tel que* $\beta_k^{(0)} \sim \mathcal{N}(0, 1)$ *et* $b_k^{(0)} \sim \mathcal{U}[-r, +r]$. *L'espérance de* $\kappa_0(x, \tilde{x})$ *par rapport à* $\theta^{(0)}$*, pour n'importe quelle paire* $(x, \tilde{x}) \in \mathbb{R}^d \times \mathbb{R}^d$*, est définie par*

$$\kappa(x, \tilde{x}) = \frac{1}{p} + \frac{r^2}{6} + \frac{1}{4rd} \sum_{i=1}^d \varrho(x_i, \tilde{x}_i), \quad \varrho(x_i, \tilde{x}_i) = 2r(x_i \tilde{x}_i + 1) - |x_i - \tilde{x}_i| + \frac{1}{6}|x_i - \tilde{x}_i|^3. \tag{12}$$

De plus, la variance $\mathbb{V}(\kappa_0(x, \tilde{x}))$ *de* $\kappa_0(x, \tilde{x})$ *satisfait* $\mathbb{V}(\kappa_0(x, \tilde{x})) = O\left(\frac{1}{p^2}\right)$.

De la Loi des Grands Nombres, nous en déduisons que le noyau $\kappa_0(x, \tilde{x})$ converge en probabilité vers la valeur de son espérance $\kappa(x, \tilde{x})$. Ainsi, lorsque le nombre de neurones composant la couche cachée du SATURNN est suffisamment grand, nous proposons d'approximer $\delta_{KLR}(x, \alpha)$ par $\delta_{EKLR}(x, \alpha)$ définie par

$$\delta_{EKLR}(x, \alpha) = \sigma\left(\sum_{j=1}^N \alpha_j \kappa(x^{(j)}, x)\right) = \sigma\left(K(x)^T \alpha\right), \tag{13}$$

où le vecteur de variables descriptives $K(x) \in \mathbb{R}^N$ est $K(x) = \left(\kappa\left(x^{(1)}, x\right), \ldots, \kappa\left(x^{(N)}, x\right)\right)^T$. L'EKLR définie par (13) ne dépend plus de $\theta^{(0)}$. Malgré que cette méthode (13) semble totalement indépendante du SATURNN, il est possible de retrouver la règle de décision estimée par le SATURNN. Soit $\hat{\alpha}^{\text{EKLR}}$ le vecteur de paramètres estimé par $\delta_{EKLR}(x, \alpha)$. Nous avons

$$\delta_{EKLR}(x, \hat{\alpha}^{\text{EKLR}}) \;=\; \sigma\left(\sum_{j=1}^N \hat{\alpha}_j^{\text{EKLR}} \kappa\left(x^{(j)}, x\right)\right) = \sigma\left(\hat{\beta}_0 + \frac{1}{4rd} \sum_{i=1}^d \hat{\beta}_i(x_i)\right), \tag{14}$$

avec $\hat{\beta}_0 = \left(\frac{1}{p} + \frac{r^2}{d}\right) \sum_{j=1}^N \hat{\alpha}_j^{\text{EKLR}}$ et $\hat{\beta}_i(x_i) = \sum_{j=1}^N \hat{\alpha}_j^{\text{EKLR}} \varrho(x_i^{(j)}, x_i)$. D'après l'équation (14), nous retrouvons une décomposition additive similaire à celle du SATURNN définie à l'équation (4). Cette décomposition ne dépend plus de l'initialisation $\theta^{(0)}$. Chaque fonction $\hat{\beta}_i(x_i)$ dépend de l'intégralité de l'échantillon d'apprentissage $x^{(j)}$. La régularisation ℓ_2 permet de garantir dans l'entraînement des poids de la RL, une estimation unique des paramètres $\hat{\alpha}_j^{\text{EKLR}}$. Ainsi, les fonctions $\hat{\beta}_i(x_i)$ sont elles aussi uniques pour un échantillon d'apprentissage donné.

5 Expériences Numériques

Dans cette section, nous comparons la performance du SATURNN et de ses méthodes d'approximation ($\delta_{\text{LR}}(x, \eta)$ dans (7), et $\delta_{\text{KLR}}(x, \alpha)$ dans (11)) à diverses méthodes de l'état de l'art. Nous considérons seulement deux variables explicatives afin de pouvoir visualiser les règles de décision estimées mais aussi le partitionnement résultant des différentes méthodes de classification non-linéaires. Nous simulons 400 échantillons $x^{(i)} = (x_1^{(i)}, x_2^{(i)}) \in \mathbb{R}^2$ selon une loi normale. Puisqu'en réalité la règle de décision est bruitée, nous définissons les étiquettes $y^{(i)}$ à partir d'une distribution de Bernoulli :

$$y^{(i)} \sim \mathcal{B}(p(x^{(i)})) \text{ avec } p(x^{(i)}) = \sigma(f_1(x_1^{(i)}) + f_2(x_2^{(i)})).$$

La probabilité générant la distribution de Bernoulli est construite à partir de la sigmoïde appliquée à une fonction de score non-linéaire. Les fonctions f_1 et f_2 sont définies de manière à obtenir les régions de la Figure 1. Nous comparons la performance des méthodes proposées à de nombreux autres modèles non-linéaires tels que les Forêts Aléatoires (RF) (Breiman, 2001), MARS (Friedman, 1991), GAMs (Hastie, 2017), Explainable Boosting Machine (EBM) (Lou *et al.*, 2012), RNs ReLU (Goodfellow *et al.*, 2016) et NAMs (Agarwal *et al.*, 2021). Pour les algorithmes gloutons, un algorithme de *tuning* est nécessaire afin d'apprendre les modèles avec des hyperparamètres optimaux. Le nombre optimal de bases de splines pour les MARS, le nombre d'arbres ainsi que leur profondeur composant le RF et l'EBM sont établis par *gridsearch*. Afin d'obtenir une comparaison équitable, nous n'intégrons pas d'effets d'intéraction pour les MARS, GAM, EBM et NAMs. Le tableau 1 résume les performances moyennes obtenues par chaque méthode par validation-croisée 5-folds sur les échantillons d'apprentissage et de validation. Enfin, le temps de calcul ne prend pas en compte le temps nécessaire pour optimiser les hyperparamètres mais seulement le temps d'apprentissage de la méthode avec des hyperparamètres optimaux.

Bien que le SATURNN ne mélange pas les variables explicatives mais traite chacune d'entre elles séparément, la méthode proposée obtient des performances similaires aux autres méthodes. Le SATURNN réalise un partitionnement interprétable de l'espace d'entrée (Fig.1 droite) avec des orthotopes contrairement aux RNs ReLU (Fig.1 gauche) qui eux produisent des régions obliques impossible à interpréter. De plus, contrairement aux RNs ReLU, le SATURNN peut être considéré comme un GAM spécifique (4) puisque les splines estimées sont univariées et peuvent donc être analysées (Fig. 2). Les modèles NAM (courbes roses) et SATURNN (courbes oranges) estiment des splines davantage cohérentes que le MARS ou le GAM. En effet, sur la Figure 1, nous pouvons constater que lorsque X_2 est grand, la plupart des échantillons appartiennent à la classe 1, ce qui justifie que les splines estimées par le NAM et le SATURNN pour $X2$ sont décroissantes à partir d'un certain seuil. Paradoxalement, la spline estimée pour $X2$ par le GAM (Fig. 2-droite, courbe verte) est toujours croissante. Ainsi, l'interprétation faite des splines estimées pas le GAM ne peut être fiable, remettant en question la convergence de cet algorithme glouton. Enfin, l'approximation du SATURNN par la KLR est bien plus rapide à entraîner que les RNs et obtient des performances similaires, si ce n'est meilleures sur l'échantillon de validation (78%) que les autres méthodes (76% pour le NAM et le RN ReLU).

Méthodes	Performance Apprentissage	Performance Validation	Temps de Calcul
RF	0.97 (0.01)	0.76 (0.01)	0.64
MARS	0.82 (0.01)	0.80 (0.01)	0.29
GAM	0.81 (0.01)	0.78 (0.01)	0.11
EBM	0.82 (0.01)	0.78 (0.02)	0.30
NAM	0.77 (0.02)	0.76 (0.01)	227
RN ReLU	0.78 (0.02)	0.76 (0.02)	208
SATURNN	0.77 (0.02)	0.76 (0.03)	233
SATURNN_∞	0.81 (0.01)	0.79 (0.03)	386
RL PSI LIN	0.81 (0.01)	0.80 (0.01)	19
KLR	0.74 (0.01)	0.78 (0.03)	0.05

Table 1: Performance moyenne (écart-type) sur les échantillons d'apprentissage et de validation et temps d'entraînement moyen (en secondes). Les méthodes présentées dans l'article sont en bleu. Le RN ReLU et le SATURNN sont entraînés avec $p = 10$. La KLR et la RL PSI LIN (la LR appliquée à la fonction de score linéarisée du SATURNN_∞) sont composées de $p = 30000$.

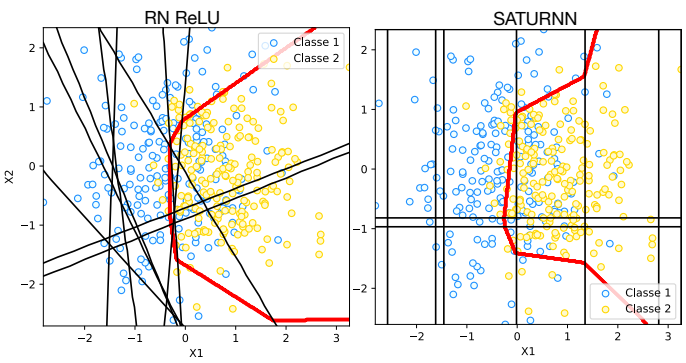

Figure 1: Partition du RN ReLU $p = 10$ (gauche) et du SATURNN $p = 10$ (droite). La règle de décision estimée est en rouge et les seuils en noir.

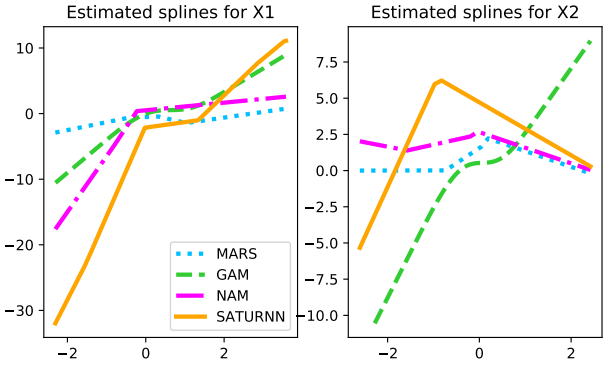

Figure 2: Splines estimées pour $X1$ (gauche) et $X2$ (droite).

6 Conclusion

Ce papier propose un Réseau de Neurones ReLU interprétable. L'architecture de ce réseau est contrainte et repose sur une somme de fonctions splines univariées. Il est démontré que ce réseau est asymptotiquement équivalent à une Régression Logistique à Noyau, lorsque le nombre de neurones composant la couche cachée tend à être infini. Nous obtenons un modèle additif composé d'une somme de fonctions univariées dépendant uniquement de l'échantillon d'apprentissage. La convergence est garantie et la décomposition de l'espace d'entrée obtenue est ainsi unique. Les prochains travaux porteront sur l'application de cette méthode sur des données réelles, notamment médicales puisque les médecins sont particulièrement intéressés par les modèles additifs avec des fonctions univariées.

Bibliographie

Agarwal, R., Melnick, L., Frosst, N., Zhang, X., Lengerich, B., Caruana, R. et Hinton, G. E. (2021). Neural additive models: Interpretable machine learning with neural nets. *Advances in Neural Information Processing Systems*, 34:4699–4711.

Balestriero, R. et Baraniuk, R. (2018). A spline theory of deep learning. *In International Conference on Machine Learning*, pages 374–383. PMLR.

Boyd, S. et Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Eckle, K. *et al.* (2019). A comparison of deep networks with relu activation function and linear spline-type methods. *Neural Networks*, 110:232–242.

Fel, T. et Vigouroux, D. (2020). Representativity and consistency measures for deep neural network explanations. *arXiv preprint arXiv:2009.04521*.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The annals of statistics*.

Goodfellow, I. J., Bengio, Y. et Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, MA, USA.

Guyomard, M., Barbosa, S. et Fillatre, L. (2023). Understandable relu neural network for signal classification. *IEEE International Conference on Acoustics, Speech and Signal Processing*.

Hastie, T., Tibshirani, R., Friedman, J. H. et Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.

Hastie, T. J. (2017). Generalized additive models. *In Statistical models in S*, pages 249–307. Routledge.

Hornik, K., Stinchcombe, M. et White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Jacot, A., Gabriel, F. et Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J. et Sohl-Dickstein, J. (2017). Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*.

Leshno, M., Lin, V. Y., Pinkus, A. et Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867.

Liu, C., Zhu, L. et Belkin, M. (2020). On the linearity of large non-linear models: when and why the tangent kernel is constant. *Advances in Neural Information Processing Systems*.

Lou, Y., Caruana, R. et Gehrke, J. (2012). Intelligible models for classification and regression. *In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158.

Meijering, E., Calhoun, V. D., Menegaz, G., Miller, D. J. et Ye, J. C. (2022). Deep learning in biological image and signal processing. *IEEE Signal Processing Magazine*.

Neal, R. M. (2012). *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

Schölkopf, B., Herbrich, R. et Smola, A. J. (2001). A generalized representer theorem. *In Computational Learning Theory*, pages 416–426. Springer Berlin Heidelberg.