Régression Logistique à Noyau Équivalente à un Réseau de Neurones Interprétable

Marie GUYOMARD[1] Susana BARBOSA[2] Lionel FILLATRE[1]

[1]Laboratoire I3S, Université Côte d'Azur, France

[2]Laboratoire IPMC, Université Côte d'Azur, France

Résumé – Cet article s'intéresse à l'approximation par une régression logistique à noyau d'un réseau de neurones interprétable. Nous introduisons un nouveau noyau découlant directement de l'architecture du réseau de neurones. La règle de décision issue d'une régression logistique appliquée à ce noyau se modélise comme une décomposition additive de fonctions univariées et est donc interprétable. Enfin, cette méthode obtient des performances prédictives similaires à celles d'un réseau de neurones mais avec pour avantage d'offrir des garanties de convergence et donc d'unicité des résultats.

Abstract – This article focuses on approximating an interpretable neural network with kernel logistic regression. We introduce a new kernel that directly stems from the architecture of the neural network. The decision rule resulting from a logistic regression applied to this kernel is modeled as an additive decomposition of univariate functions and is therefore interpretable. Finally, this method achieves predictive performance similar to that of a neural network but with the advantage of offering convergence guarantees and thus result uniqueness.

1 Introduction

Les réseaux de neurones (RNs) sont largement utilisés tant leur performance sur les tâches de régression et de classification est significative. L'explicabilité des RNs et la possibilité de certifier leurs résultats sont souvent critiquées, c'est pourquoi ils sont souvent qualifiés de "boîtes noires". 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 [2]. 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), les modèles MARS (Multivariate Adaptative Regression Splines) et les modèles additifs généralisés (GAMs) [5] partitionnent également l'espace d'entrée. Contrairement aux RNs RELU leurs partitions sont interprétables car les variables descriptives sont seuillées indépendamment les unes des autres. Néanmoins, ces méthodes exploitent un algorithme glouton ("*greedy algorithm*") afin de segmenter itérativement 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. Les modèles RNs additifs ("*Neural Additive Models*", NAM) présentés dans [1] approximent les GAMs. En effet, les auteurs proposent un RN convolutif dont les paramètres sont appris avec une Descente de Gradient Stochastique (DSG), sans aucune garantie de convergence vers une solution optimale.

Il a été démontré dans de très récents articles, notamment dans [6] 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 [7]. Cependant, il a aussi été établi dans [7] 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 deux contributions principales. Le SATURNN (*Splines Approximation Throught Understandable ReLU Neural Network*) est un modèle interprétable combinant les avantages des MARS, GAMs, NAMs et RNs. Les nouveaux résultats de cet article découlent de nos premiers travaux théoriques présentés dans [4]. Dans un premier temps, nous établissons une équivalence entre le SATURNN et une Régression Logistique (RL) appliquée à une transformation non-linéaire des variables explicatives, résultant de la linéarisation partielle du SATURNN. Deuxièmement, nous montrons que cette transformation non-linéaire, dépendante des initialisations du SATURNN peut-être remplacée par une approche à noyau déterministe, découlant directement de l'architecture du SATURNN. Nous introduisons un nouveau noyau interprétable dont les performances sont équivalentes à celles du réseau de neurones. L'approximation du SATURNN par une RL à noyau permet 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 SATURNN et son approximation par une RL appliquée à une transformation non-linéaire des variables descriptives. La Section 3 établit une approximation de cette transformation non-linéaire par un nouveau noyau, indépendant des initialisations du SATURNN. Dans la Section 4, les performances et l'interprétabilité du SATURNN et de ces méthodes d'approximation sont comparées à des algorithmes de l'état de l'art. Enfin, la Section 5 conclut l'article.

2 Linéarisation du 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 la boule $\mathcal{B}_2^d(c, r) := \{x \in \mathbb{R}^d : \|x - c\|_2 \leq r\}$ 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 $x = (x_1, \ldots, x_d)$.

Nous avons introduit dans [4] le modèle SATURNN qui est un RN ReLU pour la classification composé d'une fonction de décision $\Phi(x)$ associée à une fonction de score spécifique $\psi(x, \theta)$:

$$\Phi(x) = \sigma(\psi(x, \theta)), \tag{1}$$

$$\psi(x, \theta) = \frac{1}{\sqrt{p}} \left[\beta_0 + \sum_{k=1}^{p} \beta_k \phi(s_k x_{\upsilon(k)} + b_k) \right], \tag{2}$$

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. 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 $\phi(\cdot) = \text{ReLU}(\cdot) = \max\{0, \cdot\}$ est l'activation ReLU [2]. Contrairement aux RNs ReLU traditionnels, dont les matrices de poids mélangent entre elles toutes les variables descriptives et rendent ainsi l'interprétation du partitionnement de l'espace d'entrée impossible, la matrice de poids est un paramètre fixé dans la modélisation du SATURNN directement inspiré des modèles MARS [5]. 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 $x_{\upsilon(k)}$. $\upsilon(k)$ est un sélecteur de variable aléatoirement initialisé tel que $\upsilon(k) = \{1, \ldots, d\}$ avec une probabilité $\frac{1}{d}$. Le paramètre $s_k \in \{-1, 1\}$ précise si $h_k(x)$ est une fonction non-décroissante ou non-croissante de $x_{\upsilon(k)}$ et est généré aléatoirement par une loi de Bernoulli avec une probabilité $\frac{1}{2}$. Une fois initialisés, les paramètres $\upsilon(k)$ et s_k sont fixés. 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, aboutissant à une règle de décision interprétable. Il est intéressant de constater que le SATURNN peut être aparenté à un cas spécial des GAMs [5]. En réécrivant (2), nous obtenons :

$$\psi(x, \theta) = \frac{1}{\sqrt{p}} [\beta_0 + \sum_{i=1}^{d} \underbrace{\sum_{1 \leq k \leq p : \upsilon(k) = i} \beta_k \phi\left(s_k x_i + b_k\right)}_{f_i(x_i)}], \tag{3}$$

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 [1] qui apprennent une combinaison de $d+1$ RNs, le SATURNN estime l'intégralité de la règle de décision avec un seul RN. Les paramètres du SATURNN sont estimés en minimisant une fonction de coût $\mathcal{L}^{\text{SATURNN}}(\theta)$:

$$\mathcal{L}^{\text{SATURNN}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} L\left(\sigma(\psi(x^{(i)}, \theta)), y^{(i)}\right), \tag{4}$$

où $L(\cdot)$ désigne l'Entropie Croisée Binaire [3] :

$$L(\hat{y}, y) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}). \tag{5}$$

Les paramètres $\hat{\theta}^{\text{SATURNN}}$ de SATURNN sont donnés par

$$\hat{\theta}^{\text{SATURNN}} = \underset{\theta \in \mathcal{B}_2^{2p+1}(\theta^{(0)}, R)}{\arg\min} \mathcal{L}^{\text{SATURNN}}(\theta). \tag{6}$$

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, avec $\beta_k^{(0)} \sim \mathcal{N}(0, 1)$ et $b_k^{(0)} \sim \mathcal{U}[-r, +r]$. 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.

Nous avons démontré que, quand le nombre p de neurones est suffisamment grand, la fonction de score du SATURNN (2) devient linéaire par rapport à θ (Th. 1 et 2 dans [4]). Nous considérons la RL appliquée à la fonction de score linéarisée du SATURNN suivante :

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

avec $g_0(x) = \nabla_\theta \psi(x, \theta^{(0)})$ le gradient de $\psi(x, \theta^{(0)})$ (2) par rapport à θ au point $\theta^{(0)}$. Le vecteur de paramètres $\eta = \theta - \theta^{(0)} \in \mathbb{R}^{2p+1}$ est estimé en minimisant $\mathcal{L}^{\text{RL}}(\eta)$:

$$\mathcal{L}^{\text{LR}}(\eta) = \frac{1}{N} \sum_{i=1}^{N} L\left(\delta_{LR}(x^{(i)}, \eta), y^{(i)}\right), \tag{8}$$

tel que l'estimation $\hat{\eta}^{\text{RL}}$ vérifie

$$\hat{\eta}^{\text{RL}} = \underset{\eta \in \mathcal{B}_2^{2p+1}(0, R)}{\arg\min} \mathcal{L}^{\text{RL}}(\eta). \tag{9}$$

Le nouveau théorème suivant établit qu'il est équivalent d'entraîner le SATURNN ou la RL non-linéaire (7) lorsque p est suffisamment grand :

Théorème 1. *Soit $\theta^{(0)} = [\beta_0^{(0)}, \ldots, \beta_p^{(0)}, b_1^{(0)}, \ldots, b_p^{(0)}]$ et $r > 0$ tel que $\beta_k^{(0)} \sim \mathcal{N}(0, 1)$ et $b_k^{(0)} \sim \mathcal{U}[-r, +r]$. Soit $\mathcal{L}^{SATURNN}$ définie par (6) et $\mathcal{L}^{RL}$ par (9). Nous avons presque partout*

$$\sup_{\substack{\theta \in \mathcal{B}_2^{2p+1}(\theta^{(0)}, R) \\ \eta \in \mathcal{B}_2^{2p+1}(0, R)}} \left| \mathcal{L}^{SATURNN}(\theta) - \mathcal{L}^{RL}(\eta) \right| \leq \frac{R^2}{2\sqrt{p}}. \tag{10}$$

Pour une valeur raisonnable de R, l'erreur d'approximation reste négligeable pour p suffisamment large. Il est alors équivalent d'entraîner le SATURNN ou la RL appliquée à la transformation non-linéaire des variables d'entrée : $x \mapsto g_0(x)$.

3 Régression Logistique à Noyau

Dans [6] un pont rigoureux entre les RNs et les méthodes à noyau est construit. Néanmoins, cette théorie et celles qui ont suivi ne peuvent s'appliquer au SATURNN. Il est démontré dans [6, 7] 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, [7] 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).

D'après le Théorème de Représentation [10], le vecteur de paramètres estimé $\hat{\eta}^{\text{RL}}$ par (9) 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 $\hat{\eta}^{\text{RL}} = \sum_{j=1}^N \alpha_j g_0(x^{(j)})$. Il en découle que (7) peut se réécrire comme une KLR :

$$\delta_{KLR}(x, \alpha) = \sigma\left(K_0(x)^T \alpha\right), \tag{11}$$

avec $K_0(x) = \left(\kappa_0\left(x^{(1)}, x\right), \ldots, \kappa_0\left(x^{(N)}, x\right)\right)^T \in \mathbb{R}^N$ le vecteur d'entrée tel que, pour tout $x, \tilde{x} \in \mathbb{R}^d$, la fonction noyau $\kappa_0(x, \tilde{x})$ est définie par :

$$\kappa_0(x, \tilde{x}) = \frac{1}{p}\left[1 + \sum_{k=1}^{p} \phi\left(s_k x_{\upsilon(k)} + b_k^{(0)}\right) \phi\left(s_k \tilde{x}_{\upsilon(k)} + b_k^{(0)}\right) + {\beta_k^{(0)}}^2 \mathbb{1}_{\{s_k x_{\upsilon(k)} + b_k^{(0)} > 0\}} \mathbb{1}_{\{s_k \tilde{x}_{\upsilon(k)} + b_k^{(0)} > 0\}}\right]. \tag{12}$$

Tout comme pour la modélisation $\delta_{\text{LR}}(x, \eta)$, la KLR est appliquée à une transformation non-linéaire $K_0(x)$ des variables descriptives découlant du noyau. La transformation de chaque échantillon dépend directement de la distribution de l'ensemble de la base de données d'entraînement : $x \mapsto \left(\kappa_0(x^{(i)}, x)\right)_{i=1}^N$. 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 de $\delta_{KLR}(x, \alpha)$. Par définition, le noyau $\kappa_0(x, \tilde{x})$ dans (12) est toujours dépendant de $\theta^{(0)}$. D'après la Loi des Grands Nombres, 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 $\delta_{EKLR}(x, \alpha)$, une LR à noyau indépendante de $\theta^{(0)}$.

Proposition 1. *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 $b_k^{(0)} \sim \mathcal{U}[-r, +r]$. D'après la loi forte des grands nombres et puisque σ est continue, le kernel K_0 (12) converge asymptotiquement vers K (15). Ainsi, lorsque p est suffisamment grand, la LR à noyau (11) peut être approximée par :*

$$\delta_{EKLR}(x, \alpha) = \sigma\left(K(x)^T \alpha\right), \tag{13}$$

avec $K(x) = \left(\kappa\left(x^{(1)}, x\right), \ldots, \kappa\left(x^{(N)}, x\right)\right)^T \in \mathbb{R}^N$ le vecteur d'entrée tel que pour tout $x, \tilde{x} \in \mathbb{R}^d$, la fonction noyau $\kappa(x, \tilde{x})$ est définie par :

$$\kappa(x, \tilde{x}) = \mathbb{E}(\kappa_0(x, \tilde{x})) = \frac{1}{p} + \frac{r^2}{6} + \frac{1}{4rd} \sum_{i=1}^{d} \varrho(x_i, \tilde{x}_i), \tag{14}$$

$$\varrho(x_i, \tilde{x}_i) = 2r(x_i \tilde{x}_i + 1) - |x_i - \tilde{x}_i| + \frac{1}{6}|x_i - \tilde{x}_i|^3. \tag{15}$$

L'EKLR définie par (13) ne dépend plus de $\theta^{(0)}$. Bien 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

$$\delta_{EKLR}(x, \hat{\alpha}^{\text{EKLR}}) = \sigma\left(\hat{\beta}_0 + \frac{1}{4rd} \sum_{i=1}^{d} \hat{\beta}_i(x_i)\right), \tag{16}$$

$$\hat{\beta}_0 = \left(\frac{1}{p} + \frac{r^2}{d}\right) \sum_{j=1}^{N} \hat{\alpha}_j^{\text{EKLR}}, \tag{17}$$

$$\hat{\beta}_i(x_i) = \sum_{j=1}^{N} \hat{\alpha}_j^{\text{EKLR}} \varrho(x_i^{(j)}, x_i). \tag{18}$$

D'après (16), nous retrouvons une décomposition additive similaire à celle du SATURNN définie dans (3). 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.

4 Expériences Numériques

Nous comparons la performance du SATURNN et de ses méthodes d'approximation, $\delta_{\text{LR}}(x, \eta)$ dans (7), $\delta_{\text{KLR}}(x, \alpha)$ dans (11) et $\delta_{\text{EKLR}}(x, \alpha)$ dans (13), à diverses méthodes de l'état de l'art. Nous considérons la base de données réelle Framingham [9] dont le but est de prédire un évènement cardio-vasculaire à partir de 15 variables descriptives tel que le genre, le taux de cholestérol ou encore l'indice de masse corporelle. Les proportions de la base de données étant fortement déséquilibrées (85% - 15%) nous avons choisi de la rééquilibrer. Nous réalisons ainsi les expériences sur 1114 patients.

Nous comparons la performance des méthodes proposées à de nombreux autres modèles non-linéaires tels que les Forêts Aléatoires (RF), MARS, GAMs [5], Explainable Boosting Machine (EBM) [8], RNs ReLU [3] et NAMs [1]. 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 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. De plus, le SATURNN et le NAM sont entraînés avec le même nombre de neurones, à savoir $p = 40$. Afin d'éviter le sur-apprentissage, le RN ReLU quant à lui contient $p = 10$ neurones. Le tableau 1 résume les performances moyennes obtenues par chaque méthode par validation-croisée 5-folds sur les échantillons 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 voire même supérieures aux autres méthodes : le SATURNN composé de $p = 40$ neurones obtient 72% d'AUC sur l'échantillon de validation, le GAM 69% et le NAM 70%. En mélangeant les variables entre elles, le RN ReLU ne permet pas de visualiser graphiquement le découpage de l'espace d'entrée. Sur la figure 1, nous pouvons analyser les segmentations induites par les méthodes interprétables. Le NAM (rose) et le SATURNN

TABLE 1 : Performance moyenne et AUC (écart-type) sur les échantillons de validation et temps d'entraînement moyen (en secondes). Les méthodes présentées dans l'article sont en bleu. La KLR, l'EKLR et la RL PSI LIN (la LR appliquée à la fonction de score linéarisée du SATURNN$_\infty$) ont $p = 50\,000$ neurones.

	Test		Computation
Méthodes	**Acc**	**AUC**	**Time**
RF	0.64 (0.02)	0.70 (0.02)	0.1
MARS	0.66 (0.02)	0.71 (0.01)	0.1
GAM	0.63 (0.02)	0.69 (0.02)	0.65
EBM	0.65 (0.01)	0.72 (0.01)	7.8
NAM	0.65 (0.01)	0.70 (0.02)	694
RN ReLU	0.61 (0.03)	0.66 (0.03)	483
SATURNN	**0.67 (0.03)**	0.72 (0.02)	735
SATURNN$_\infty$	0.62 (0.02)	0.71 (0.01)	591
RL PSI LIN	0.64 (0.02)	0.69 (0.02)	52
KLR	0.66 (0.02)	**0.73 (0.02)**	0.32
EKLR	0.66 (0.01)	**0.73 (0.02)**	0.35

(orange) estiment des splines similaires pour le nombre de cigarettes par jour et le glucose. Tandis que le GAM (vert) et le MARS (bleu) introduisent peu de non-linéarités, alors que leurs performances auraient pu être améliorées comme le démontrent l'AUC du SATURNN. Ces difficultés résident dans le fait que ces méthodes n'optimisent pas un critère global mais utilisent une méthode itérative pour ajouter des effets de seuil. Nous pouvons constater que la spline induite par le SATURNN pour la variable diabète (Fig. 1-droite) est nulle. Ce comportement peut être apparenté à une sélection de variables : le SATURNN estime que les variables diabète, nombre de battements de coeur par minute et présence de toux n'ont pas d'effets significatifs sur le développement de pathologie. Bien qu'en écartant ces variables de la règle de décision, le SATURNN a de meilleures performances que les méthodes NAM et GAM qui ont ajouté de la complexité en seuillant ces dernières. Finalement, les méthodes d'approximation du SATURNN sont bien plus rapides que les RNs (resp. 320 et 350 millisecondes pour la KLR et l'EKLR) et obtiennent de meilleures performances prédictives (73% d'AUC sur l'échantillon de validation pour la KLR et l'EKLR).

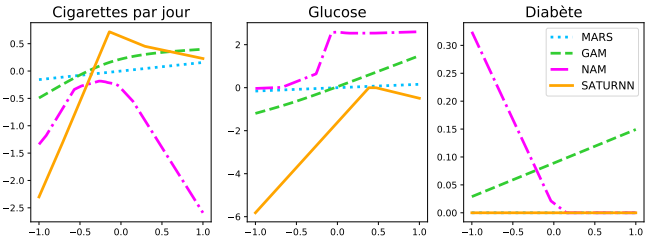

FIGURE 1 : Splines estimées par le MARS (bleu), le GAM (vert), le NAM (rose) et le SATURNN avec $p = 40$ neurones (orange).

5 Conclusion

Ce papier introduit une régression logistique à noyau qui est équivalente au réseau de neurones interprétable SATURNN mais qui est plus facile à estimer. Le noyau proposé découle directement de l'architecture du SATURNN. Il introduit ainsi un partitionnement de l'espace d'entrée interprétable qui se modélise comme une décomposition additive de fonctions splines univariées. Enfin, nous avons la garantie que l'apprentissage de la régression logistique à noyau converge vers une solution unique qui rend la règle de décision estimée unique conditionnellement à l'échantillon d'apprentissage. Les prochains travaux porteront sur la généralisation de ces méthodes aux tâches de classification multiclasses.

Références

[1] Rishabh AGARWAL, Levi MELNICK, Nicholas FROSST, Xuezhou ZHANG, Ben LENGERICH, Rich CARUANA et Geoffrey E HINTON : Neural additive models : Interpretable machine learning with neural nets. *Advances in Neural Information Processing Systems*, 34, 2021.

[2] Randall BALESTRIERO *et al.* : A spline theory of deep learning. *In International Conference on Machine Learning*, pages 374–383. PMLR, 2018.

[3] Ian J. GOODFELLOW, Yoshua BENGIO et Aaron COURVILLE : *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016.

[4] Marie GUYOMARD, Susana BARBOSA et Lionel FILLATRE : Understandable relu neural network for signal classification. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2023.

[5] Trevor HASTIE, Robert TIBSHIRANI, Jerome H FRIEDMAN et Jerome H FRIEDMAN : *The elements of statistical learning : data mining, inference, and prediction*, volume 2. Springer, 2009.

[6] Arthur JACOT, Franck GABRIEL et Clément HONGLER : Neural tangent kernel : Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[7] Chaoyue LIU, Libin ZHU et Misha BELKIN : On the linearity of large non-linear models : when and why the tangent kernel is constant. *Advances in Neural Information Processing Systems*, 33:15954–15964, 2020.

[8] Yin LOU, Rich CARUANA et Johannes GEHRKE : Intelligible models for classification and regression. *In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012.

[9] Syed S MAHMOOD, Daniel LEVY, Ramachandran S VASAN et Thomas J WANG : The framingham heart study and the epidemiology of cardiovascular disease : a historical perspective. *The lancet*, 383(9921):999–1008, 2014.

[10] Bernhard SCHÖLKOPF, Ralf HERBRICH et Alex J. SMOLA : A generalized representer theorem. *In* David HELMBOLD et Bob WILLIAMSON, éditeurs : *Computational Learning Theory*, pages 416–426. Springer Berlin Heidelberg, 2001.