Enhancing Explainability of Neural Networks through Architecture Constraints

Zebin Yang[1], Aijun Zhang[1] and Agus Sudjianto[2]

[1]Department of Statistics and Actuarial Science, The University of Hong Kong
Pokfulam Road, Hong Kong

[2]Corporate Model Risk, Wells Fargo, USA

Abstract

Prediction accuracy and model explainability are the two most important objectives when developing machine learning algorithms to solve real-world problems. The neural networks are known to possess good prediction performance, but lack of sufficient model explainability. In this paper, we propose to enhance the explainability of neural networks through the following architecture constraints: a) sparse additive subnetworks; b) orthogonal projection pursuit; and c) smooth function approximation. It leads to a sparse, orthogonal and smooth explainable neural network (SOSxNN). The multiple parameters in the SOSxNN model are simultaneously estimated by a modified mini-batch gradient descent algorithm based on the backpropagation technique for calculating the derivatives and the Cayley transform for preserving the projection orthogonality. The hyperparameters controlling the sparse and smooth constraints are optimized by the grid search. Through simulation studies, we compare the SOSxNN method to several benchmark methods including least absolute shrinkage and selection operator, support vector machine, random forest, and multi-layer perceptron. It is shown that proposed model keeps the flexibility of pursuing prediction accuracy while attaining the improved interpretability, which can be therefore used as a promising surrogate model for complex model approximation. Finally, the real data example from the Lending Club is employed as a showcase of the SOSxNN application.

Keywords: Explainable neural network, additive decomposition, sparsity, orthogonal projection, smoothness, function approximation.

1 Introduction

The recent developments of neural network techniques offer tremendous breakthroughs in machine learning and artificial intelligence (AI). Substantial complicated network structures are designed and have brought great successes in areas like computer vision and natural language processing. Besides predictive performance, transparency and explainability are essential aspects of a trustful model; however, most of the neural networks remain black-box models, where the inner decision-making processes cannot be easily understood by human beings. Without sufficient explainability, their applications in specialized domain areas such as medicine and finance can be largely limited. For instance, a personal credit scoring model in the banking industry should be not only accurate but also convincing. The terminology "Explainable AI" advocated by the Defense Advanced Research Projects Agency (DARPA) draws the public attention (Gunning, 2017). Recently, the US Federal Reserve governor raised the regulatory use of AI in financial services (Brainard, 2018) and emphasized the development of explainable AI tools.

There has been a considerable amount of research works on explainability of machine learning algorithms, including the model-agnostic approach and the model distillation approach. The examples of the former approach are the partial dependence plot (Friedman, 2001), the individual conditional expectation plot (Goldstein et al., 2015), the locally interpretable model-agnostic explanation method (Ribeiro et al., 2016) and others (Apley, 2016, Liu et al., 2018). The examples of the latter approach are model compression and distillation (Bucilua et al., 2006, Ba and Caruana, 2014), network distilling (Hinton et al., 2015), network pruning (Wang et al., 2018), and tree-partitioned surrogate modeling (Hu et al., 2018). In constrast, most statistical models are intrinsically explainable, e.g., linear regression, logistic regression, and decision tree. These simplified models are however known to be less predictive than the black-box type of machine learning algorithms in dealing with large-scale complex data.

It is our goal to design highly predictive models that are fundamentally explainable. This is a challenging task as the two objectives (i.e., prediction accuracy and model explainability) are usually conflict with each other (Gunning, 2017). A linear regression can be easily explained, but it is limited to linear problems only. A deep neural network usually provides accurate prediction, but it can be hardly understood even by the model developers. To balance these two objectives, the explainable neural network (xNN) was recently proposed by Vaughan et al. (2018) based on the additive index model (AIM), where the AIM is also known as the projection pursuit regression (PPR) in statistical literature (Friedman and Stuetzle, 1981). Given the vector of features $\boldsymbol{x} \in \mathbb{R}^{n \times p}$ and the response $y \in \mathbb{R}^n$, the AIM

takes the following form to model the relationship between features and response,

$$f(\boldsymbol{x}) = \mu + h_1(\boldsymbol{w}_1^T \boldsymbol{x}) + ... + h_k(\boldsymbol{w}_k^T \boldsymbol{x}), \tag{1}$$

where μ is the intercept, $\boldsymbol{w}_j \in \mathbb{R}^p$ for $j = 1, \ldots, k$ are the projection indexes for each projection $z_j = \boldsymbol{w}_j^T \boldsymbol{x}$, and each $h_j(z_j)$ is the corresponding ridge function. The AIM decomposes a complex function into the linear combination of multiple uni-dimensional component functions. In the special case when $k = 1$, it reduces to the single-index model (Ichimura, 1993). In another special case when $\boldsymbol{w}_j = \boldsymbol{e}_j$ (the standard vector with 1 at the j-th position and 0 elsewhere) for $j = 1, \ldots, k$, the AIM reduces to the generalized additive model (GAM); see Hastie and Tibshirani (1990).

The AIM used to be estimated by alternating minimization between the ridge functions and the projection indexes. When $\boldsymbol{W} = \{\boldsymbol{w}_j\}_{j \in [k]}$ are fixed, the ridge functions are often estimated by nonparametric smoothers subject to backfitting, a common approach for the GAM. When $\{h_j\}_{j \in [k]}$ are fixed, the projection indexes can be iteratively estimated by Gauss-Newton algorithms. Such alternating estimation procedure may not guarantee the global optimum and it becomes time-consuming when dealing with large-sample observations. In contrast, the xNN approach by Vaughan et al. (2018) reformulates the AIM by a neural network architecture. It takes advantages of the additive decomposition as in (1), while modeling each ridge function as an independent subnetwork consisting of one input node, multiple hidden layers and one output node. The projection indexes $\boldsymbol{W}$ are represented by the projection layer right after the input layer. With such network architecture, the parameters can be simultaneously optimized by mini-batch backpropagation, which is especially effective for large-scale datasets by using the newly developed TensorFlow platform (Abadi et al., 2016). Moreover, the ℓ_1-shrinkage can be flexibly imposed onto the projection layer and the final combination layer, in order to enforce sparsity in the projection weights $\boldsymbol{W}$ and the set of ridge functions, respectively. Note that when $\boldsymbol{W}$ is fixed, the xNN corresponds to the sparse additive model studied by Ravikumar et al. (2009). It can also extended to the structured additive model (Fawzi et al., 2016) based on user-specified feature grouping.

There are some potential drawbacks of the original xNN architecture by Vaughan et al. (2018) in pursuit of model explainability. First, it was suggested to rewrite each ridge function as $\beta_j h_j(\boldsymbol{w}^T \boldsymbol{x})$ then apply the ℓ_1-shrinkage on $\{\beta_j\}_{j \in [k]}$ to enforce the sparsity. Such splitted ridge functions become unidentifiable unless the norm constraints are imposed, i.e. $\mathbb{E}[h_j^2] = 1$ for $j = 1, \ldots, k$. Second, the projection weights $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k$ in the original AIM can be correlated, hence add the difficulty for model interpretation. It is desirable that the resulting projection indexes are mutually orthogonal, just like that for the principal component analysis (PCA) for data rotation and dimension reduction. Third, the ridge functions as approximated by the neural networks may not be as smooth as the nonparametric methods,

e.g., the smoothing splines (Ruan and Yuan, 2010, Raskutti et al., 2012). The non-smooth ridge functions in the original xNN model add difficulty for model interpretation.

In this paper, we propose to enhance the explainability of neural networks through additive, sparse, orthogonal and smooth architecture constraints. Specifically, the xNN architecture is re-configured to take into account the sparse additive subnetworks with ℓ_1-sparsity constraints on both the projection layer and the final layer, together with an additional step of batch normalization for ensuring ridge function identifiability. The orthogonal constraint is imposed on the projection indexes such that $\boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}_k$, which is known as the Stiefel manifold $\mathrm{S}t(p, k)$ and can be treated by the Cayley transform (Wen and Yin, 2013). For each ridge function, the smooth constraint is imposed by considering an empirical version of the roughness penalty based on the squared second-order derivatives (Wahba, 1990). Combining all these constraints into the neural network architecture, we name it as SOSxNN with the emphasis on "x" (explainable).

Computationally, the proposed SOSxNN model is estimated by modern neural network training techniques, including backpropagation, mini-batch gradient descent, batch normalization, and Adam for adaptive stochastic optimization (Goodfellow et al., 2016, Kingma and Ba, 2014). Unlike conventional network training without orthogonal and smooth constraints, we develop a new SOS-BP algorithm based on the backpropagation technique for calculating the derivatives and Cayley transform for preserving the projection orthogonality. Unlike the conventional backfitting algorithm, all the unknown parameters in the SOSxNN model are simultaneously estimated by mini-batch gradient descent. Owing to the modern machine learning platform (e.g., the TensorFlow used here) with the automatic differentiation technology, the empirical roughness penalty can be readily evaluated for each projected data point. The hyperparameters controlling the sparse and smooth constraints are optimized by the grid search, which is easily modified by the random search or more sophisticated automatic procedure. Moreover, the proposed SOS-BP algorithm based on mini-batch strategy is scalable to large-scale datasets, and it can also be implemented on GPU machines.

The proposed SOSxNN architecture keeps the flexibility of pursuing prediction accuracy while attaining the improved interpretability. It can be therefore used as a promising surrogate model for complex model approximation. Through simulation studies, the SOSxNN is shown competitive regarding the predictive accuracy as it can outperform or perform quite close to classical machine learning models, e.g., the support vector machine (SVM) and the multi-layer perceptron (MLP). On the other hand, the estimated SOSxNN model conveys the inherent explainability in terms of projection weights and corresponding subnetworks. As a showcase of potential applications, the SOSxNN is used to study a real data example from the Lending Cub for unveiling the multivariate underwriting features in the loan acquisition decision.

This paper is organized as follows. Section 2 provides the reformulation of the xNN subject to sparse, orthogonal and smooth constraints. Section 3 discusses the computational method through the new SOS-BP algorithm. Numerical experiments are conducted with both simulation studies and real data example, as presented in Section 4. Finally, Section 5 concludes and outlines the future study.

2 Explainable Neural Networks

Given the feature vector $\boldsymbol{x} \in \mathbb{R}^p$ and the response $y \in \mathbb{R}$, consider the additive index model with the matrix of projection indexes $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_j] \in \mathbb{R}^{p \times k}$ for $k \leq p$, and the corresponding ridge functions $h_j(z)$ of each projected data $z = \boldsymbol{w}_j^T \boldsymbol{x}$, for $j = 1, \ldots, k$. When each ridge function h_j is modeled by a feed-forward neural network, we reformulate the xNN model as

$$y = \mu + \sum_{j=1}^{k} \beta_j h_j(\boldsymbol{w}_j^T \boldsymbol{x}) + \varepsilon \tag{2}$$

subject to the following interpretability constraints:

$$\mathbb{E}[h_j] = 0, \ \mathbb{E}[h_j^2] = 1 \tag{2a}$$

$$\sum_{j=1}^{k} \left|\beta_j\right| \leq T_1 \tag{2b}$$

$$\sum_{i=1}^{p} \left|W_{ij}\right| \leq T_2 \tag{2c}$$

$$\boldsymbol{W}^T \boldsymbol{W} = \boldsymbol{I}_k \tag{2d}$$

$$\int \left[h_j^{(2)}(u)\right]^2 du \leq T_3, \tag{2e}$$

for $j = 1, \ldots, k$, where $T_1, T_2, T_3 > 0$ are the regularization parameters and ε is the random noise. The multiple constraints (2a-2e) are imposed by the interpretability considerations from the sparse, orthogonal and smooth perspectives.

a) **Sparse Additive Subnetworks.**
The ridge functions in (2) are modeled by the additive subnetworks, where each subnetwork corresponds to a feed-forward neural network for flexible function approximation. The neural network with one or multiple hidden layers is known to approximate a continuous function on a compact domain arbitrarily well. For each subnetwork, it is flexible to specify the number of layers and the number of nodes per layer. Upon different settings, the subnetwork can be either a wide network with few layers and many nodes per layer or a deep network with many layers and few nodes per layer.

Two ℓ_1-norm constraints are imposed for inducing the sparsity in both the scales of ridge functions and the projection weights, in order to make the xNN model parsimonious. By the ℓ_1 constraint (2b) upon suitable choice of T_1, a certain number of ridge functions $h_j(z)$ will have zero scales, i.e., $\hat{\beta}_j = 0$ for some j; such ridge functions and corresponding projection indexes will become inactive in the xNN model. By the ℓ_1 constraint (2c) upon the suitable choice of T_2, the sparsity in each projection index can be induced such that some negligible weights will be truncated to zero, similar to the sparse PCA method by Zou et al. (2006).

For the sake of subnetwork identifiability, the overall mean and variation of functions are constrained by the condition (2a). This is essentially performing normalization of each ridge function. It is a critical procedure for simultaneously estimating all the subnetworks while performing the ℓ_1-shrinkage on the ridge function scales.

b) **Orthogonal Projection Pursuit.**

The constraint (2d) is imposed to ensure that the projection indexes are mutually orthogonal, and it is sometimes denoted as the Stiefel manifold constraint $\boldsymbol{W} \in \mathrm{St}(p, k)$. Unlike the sparse and smooth constraints based on soft thresholding, the Stiefel manifold condition is a hard constraint, and it leads to a non-trivial problem for model estimation. In the next section, we develop an algorithm that preserves the projection orthogonality during the optimization process.

Without the orthogonality assumption, the traditional AIM or PPR model often results in correlated or even identical projection indexes, which makes it difficult to distinguish different additive components. It is a natural idea to consider orthogonal projection to avoid correlated ambiguity; see, e.g., the principal component regression (PCR) by Jolliffe (1982) and the supervised PCA by Bair et al. (2006), Barshan et al. (2011). The orthogonal constraint can also be understood from the linear algebra point of view, as it provides an orthogonal basis for data rotation. Compared to the original xNN model, the new SOSxNN model with orthogonal projection weights is easier to explain.

Moreover, the imposed constraint (2d) makes the xNN model more identifiable. As studied by Yuan (2011), some strong identifiability conditions are needed for estimating the AIM. Such identifiability conditions can be relaxed when the projection weights are assumed to be orthogonal. As in the context of neural network training by gradient-based methods, the orthogonal constraint is also helpful to avoid vanishing and exploding gradient problems (Wisdom et al., 2016).

c) **Smooth Function Approximation.**

The functional roughness penalty (2e) is imposed as a constraint in order to enforce each ridge function to be smooth. Following Wahba (1990), we formulate it by the integrated squared second-order derivatives over the entire range of projected data $z_j = \boldsymbol{w}_j^T \boldsymbol{x}$. In practice, the integral form in (2e) can be replaced by the following empirical roughness penalty,

$$\Omega(h_j) = \frac{1}{n} \sum_{i=1}^{n} \left[h_j^{(2)}(\boldsymbol{w}_j^T \boldsymbol{x}_i) \right]^2, \text{ for } j = 1, \ldots, k. \tag{3}$$

Thus, the neural network approximation of the ridge function subject to roughness penalty can be regarded as an alternative to classical nonparametric smoothers. As a benefit, the neural network training techniques can be directly used for fitting the smooth ridge functions. Compared with the original xNN model without smooth regularization, the new SOSxNN model can prevent non-smooth representation and achieve better explainability.

The above reformulated xNN model with sparse, orthogonal and smooth constraints is called SOSxNN in short. Fig. 1 presents the architecture of SOSxNN using neural network notations. In particular, each node $\boldsymbol{\Sigma}$ represents the linear combination with inputs from the previous layer and the coefficient arrows, where the arrows are drawn as dashed lines to indicate that the coefficients are subject to sparse constraints. The shaded area of $\mathrm{St}(p, k)$ indicates the orthogonal constraint for the projection indexes $\boldsymbol{W}^T \boldsymbol{W} = \boldsymbol{I}_k$. For each subnetwork, it consists of one input node for projected data, multiple hidden layers for the feed-forward neural network, and two output nodes $\mathbf{N}$ (normalization) and $\boldsymbol{\Omega}$ (roughness penalty). All the normalized ridge functions are then linearly combined to form the final model of the form (2), together with a bias node for capturing the overall mean μ. In model training, all the roughness penalty outputs are plugged into the final empirical loss based on the Lagrange formulation below.

The SOSxNN model estimation is carried out by the empirical risk minimization. Denote the list of unknowns in the proposed model by

$$\boldsymbol{\theta} = \Big\{ \mu; \beta_1, \ldots, \beta_k; h_1, \ldots, h_k; \boldsymbol{w}_1, \ldots, \boldsymbol{w}_k \Big\}, \tag{4}$$

consisting of scalars, vectors and functions. It is noted that the ridge function is approximated by the feed-forward neural network with finite parameters but for simplicity we denote the ridge function as unknown. For each feature vector $\boldsymbol{x}$, denote the SOSxNN prediction by $\hat{y} = \mathbb{E}[f(\boldsymbol{x}; \boldsymbol{\theta})] = \mu + \sum_{j=1}^{k} h_j(\boldsymbol{w}_j^T \boldsymbol{x})$. To measure the prediction accuracy, a convex loss $l(y, \hat{y})$ can be specified depending on the type of response y. Here, we give two typical examples of loss functions for regression and classification problems:

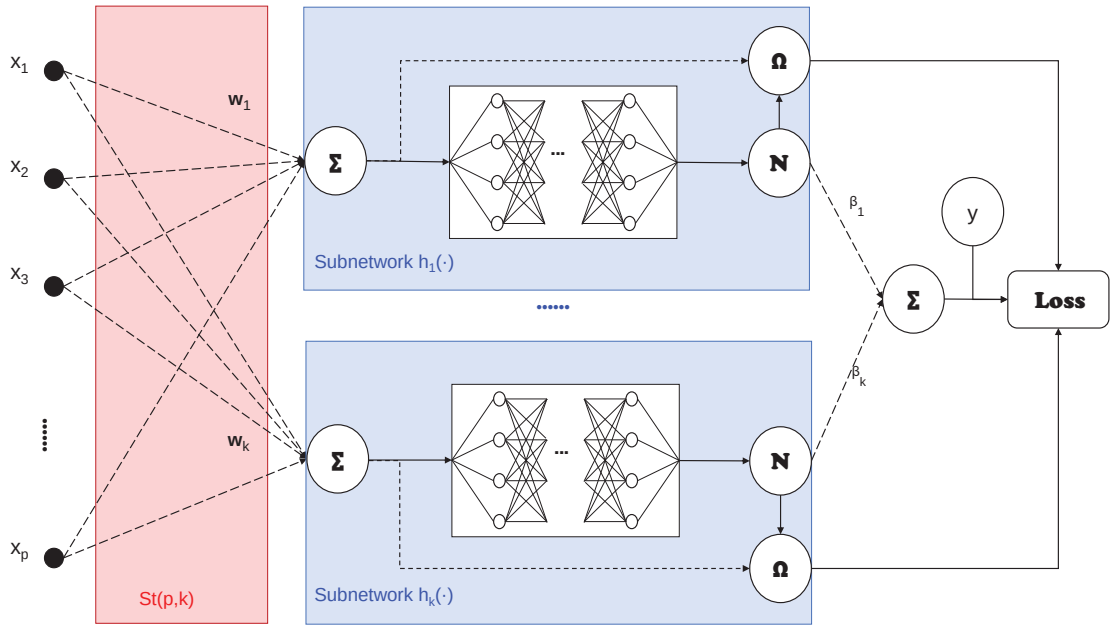


Figure 1: The SOSxNN architecture: the dashed arrows to the $\mathbf{\Sigma}$ nodes are subject to the sparse constraints on $\boldsymbol{W} = \{\boldsymbol{w}_j\}_{j\in[k]}$ and $\boldsymbol{\beta} = \{\beta_j\}_{j\in[k]}$, respectively; the red shaded $\mathrm{St}(p,k)$ area represents the Stiefel manifold constraint on the projection indexes ($\boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}_k$), and for each blue shaded subnetwork a smooth function approximation is performed to capture the normalized ridge function through a feed-forward neural network associated with a normalization node $\mathbf{N}$ and a roughness penalty node $\mathbf{\Omega}$.

- **Least squares loss.** For the regression problem with continuous responses,

$$l(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{5}$$

- **Cross entropy loss.** For the classification problem with binary responses $y \in \{0,1\}$,

$$l(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i), \tag{6}$$

  where $\hat{y}_i$ represents the predicted probability of $P(Y=1|\boldsymbol{x};\boldsymbol{\theta})$.

By the method of Lagrange multipliers, both the ℓ_1 penalty (2b, 2c) and the ℓ_2 roughness penalty (3) can be formulated as the soft regularizers, while the mean-variation condition

(2a) and the orthogonal condition (2d) stay as hard constraints. It leads to the following constrained optimization problem,

$$\begin{gathered}\min_{\boldsymbol{\theta}} \mathcal{L}_{\lambda_1,\lambda_2,\lambda_3}(\boldsymbol{\theta}) = l(\boldsymbol{\theta}) + \frac{\lambda_1}{kp}\|\boldsymbol{\beta}\|_{\ell_1} + \frac{\lambda_2}{kp}\sum_{j=1}^{k}\|\boldsymbol{w}_j\|_{\ell_1} + \frac{\lambda_3}{k}\sum_{j=1}^{k}\Omega(h_j) \\ \text{s.t. } \boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}_k, \\ \mathbb{E}[h_j] = 0, \mathbb{E}[h_j^2] = 1, \text{ for } j = 1, \ldots, k,\end{gathered} \tag{7}$$

where $\lambda_1, \lambda_2, \lambda_3 \geq 0$ are the regularization parameters for the ridge function scales, the projection indexes and the roughness penalties, respectively. Here for simplicity, we use the same λ_2 for all the projection indexes and the same λ_3 for all the ridge function roughness penalties, and they can be extended to be variable for specific projections, which is beyond the scope of the current paper.

3 Computational Method

In this section, we discuss the computational procedures for estimating the SOSxNN model by empirical risk minimization, i.e., solving the optimization problem (7). We develop a new SOS-BP algorithm based on modern neural network training techniques. The multiple unknown parameters in (4) are simultaneously optimized by mini-batch gradient descent. To deal with the hard constraints in (7), we employ the Cayley transform for preserving the projection indexes on the Stiefel manifold and employ the batch normalization for each estimated ridge function. The Adam technique for stochastic optimization is used to determine the adaptive learning rate. The grid search method for hyperparameter optimization is used to determine the regularization parameters.

3.1 SOS-BP Algorithm

The first-order gradient descent method is used for the empirical risk minimization problem (7). The backpropagation method is used to compute the derivatives based on the chain rule, which can be effectively conducted by automatic differentiation — a built-in function of modern neural computing platforms (e.g., TensorFlow used here). Such BP procedure is applied throughout the remaining discussions involving derivative and gradient evaluations.

One of the challenging tasks is to deal with the Stiefel manifold constraint $\boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}_k$. During the updates, the orthogonality of $\boldsymbol{W}$ needs to be preserved, for which we employ a fast and effective update scheme proposed by Wen and Yin (2013). It is based on the following Cayley transform along the gradient descent direction on Stiefel manifold

$$\boldsymbol{W}(\tau) = \left(\boldsymbol{I} + \frac{\tau}{2}\boldsymbol{A}\right)^{-1}\left(\boldsymbol{I} - \frac{\tau}{2}\boldsymbol{A}\right)\boldsymbol{W}, \tag{8}$$

Algorithm 1 The SOS-BP Algorithm

Require: $\{(\boldsymbol{x}_i, y_i\}_{i\in[n]}$ (Training data), k (Number of subnetowrks), λ_1, λ_2 (Sparsity), λ_3 (Smoothing), $\boldsymbol{H}$ (Subnetwork structure), η (Learning rate), τ (Step size for Cayley transform), n_b (Mini-batch size) and M (Number of epochs).

Initialize the network with $\boldsymbol{W}$ satisfying $\boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}_k$.
for Epoch $m = 1, ..., M$ **do**
    Split the reshuffled data into $B = \lfloor \frac{n}{n_b} \rfloor$ mimi-batches, each with n_b samples.
    for Batch $b = 1, ..., B$ **do**
        Select the j-th mini-batch and set $t = (m-1)B + b$.
        Update $\boldsymbol{W}$ by $\boldsymbol{W}^{(t+1)} = \boldsymbol{W}^{(t)}(\tau)$.
        Update $\tilde{\boldsymbol{\theta}}^{(t+1)} = \tilde{\boldsymbol{\theta}}^{(t)} - \eta_t \cdot \nabla^{(t)}_{\tilde{\boldsymbol{\theta}}}$, where $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta} \setminus \boldsymbol{W}$.
        Perform batch normalization for h_j, $j = 1, \ldots, k$.
        Update η_t adjusted by Adam optimizer.
    end for
    if Validation score is not improving **then**
        Stop training.
    end if
end for
Prune the subnetworks if the corresponding $\beta_j's$ are close to zero.
Fine-tune the remaining subnetworks.

where τ is a step size, $\boldsymbol{A} = \boldsymbol{G}_{\boldsymbol{W}}\boldsymbol{W}^T - \boldsymbol{W}\boldsymbol{G}_{\boldsymbol{W}}^T$ is a skew-symmetric matrix with $\boldsymbol{G}_{\boldsymbol{W}}$ being the partial gradient $\partial\mathcal{L}/\partial\boldsymbol{W}$. It can be verified that $\boldsymbol{W}(\tau)^T\boldsymbol{W}(\tau) = \boldsymbol{W}^T\boldsymbol{W}$ for $\tau \in \mathbb{R}$, therefore the orthogonality can be preserved. It is also justified that when the step size τ is small enough, the value of the objective function will get improved.

The multiple parameters in (4) can be simultaneously optimized by mini-batch gradient descent together with Cayley transform. It is an iterative procedure with flexibility in specifying the mini-batch size n_b and the number of epochs M. The total number of gradient descent iterations is controlled by M multiplied by $\lfloor n/n_b \rfloor$. Within each iteration, the parameters $\boldsymbol{W} = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k\}$ are updated separately by Cayley transform, while the remaining parameters $\boldsymbol{\theta} \setminus \boldsymbol{W}$ are updated by gradient descent with adaptive learning rates determined by the adaptive moment estimation (Adam) method (Kingma and Ba, 2014). The Adam optimizer utilizes the past optimization information and has been shown capable of preventing from being trapping into local minima.

The new SOS-BP algorithm for estimating the SOSxNN model is presented in Algorithm 1. We discuss several other computational aspects before addressing the problem of

hyperparameter selection in the next subsection.

a) For initialization, a simple strategy can be used to generate an orthogonal matrix $\boldsymbol{W}$ by singular value decomposition (SVD). We can randomly generate a matrix from $\mathbb{R}^{p\times k}$, then run SVD to generate $\boldsymbol{W}$ by collecting the singular vectors.

b) Each subnetwork modeled by the feed-forward neural network can be parametrized by $\boldsymbol{H} = [n_1, n_2, \ldots;$ "act-type"], where n_j stands for the number of nodes for the jth hidden layer and "act-type" is the type of activation function used by each node. A deeper network could be more expressive but is more difficult to be trained. Empirically, it is found that a two-hidden-layer network with nonlinear activation has been demonstrated to be sufficient.

c) The roughness penalty (3) for each ridge function is evaluated empirically for each mini-batch data, where the second-order derivatives for each data point are readily obtainable by automatic differentiation. This procedure corresponds to the dashed input arrow of the $\boldsymbol{\Omega}$ node within each subnetwork; see Fig. 1.

d) The three soft regularization terms in (7) are automatically taken into account by Cayley transform and the gradient descent. More specifically, $\sum_{j=1}^{k} \|\boldsymbol{w}_j\|_{\ell_1}$ takes effect when computing the partial gradient $\boldsymbol{G}_{\boldsymbol{W}}$ for Cayley transform, while $\|\boldsymbol{\beta}\|_{\ell_1}$ and $\sum_{j=1}^{k} \Omega(h_j)$ take effect when computing the partial gradient $\nabla_{\tilde{\boldsymbol{\theta}}}^{(t)}$ in Line 7 of Algorithm 1.

e) Note the the ℓ_1-shrinkage in neural networks may not shrinkage the weights exactly to zero, we perform a network pruning step to delete all the subnetworks with sufficiently small $|\beta_j|$ estimates. The remaining subnetworks are then fine-tuned to reduce bias.

f) To deal with the mean-variation constraint for each ridge function, a normalization procedure is required. We adopt the popular batch normalization strategy in neural network training. Referring to Fig. 1, the batch normalization is performed by the **N** node within each subnetwork.

g) The proposed SOS-BP algorithm adopts the mini-batch gradient descent strategy, and it utilizes some of the latest developments of neural network training techniques. It is capable of handling very big dataset.

3.2 Hyperparameter Selection

In Algorithm 1, there exist multiple hyperparameters that need to be tuned. First of all, the parameter k controls the maximal number of subnetworks, and it is set to be $\min\{p, 10\}$

since a small number of subnetworks is preferable for high-dimensional data. The selection of maximal training epochs and mini-batch is also important for the optimization results. In general, the choice of these values is related to the sample sizes. The mini-batch size is set to $\min\{1000, 0.2n\}$. For small datasets, e.g., fewer than 10000 samples, we can set the maximal training epochs to 10000 with early stop strategy. Based on enormous numerical experiments, we suggest to determine the other hyperparameters empirically by the following configurations:

a) The feed-forward neural network structure is fixed to be $[10, 6;$ "tanh"$]$;

b) The initial learning rate η is fixed at $\eta = 0.001$;

c) The step size τ for Cayley transform is fixed at $\tau = 0.1$;

d) The sparsity parameters λ_1, tuned at log-scale with $[10^{-3}, 10^{-2}, 10^{-1}]$;

e) The sparsity parameters λ_2, tuned at log-scale with $[10^{-3}, 10^{-2}, 10^{-1}]$;

f) The smoothing parameter λ_3, tuned at log-scale within range $[10^{-5}, 10^{-6}, 10^{-7}]$.

The first three hyperparameters were tested to have relatively stable performances. For the architecture of subnetworks, a deeper network could be more expressive. However, at the same time, the training for the deep neural network is more difficult and may be more easily overfitted. Empirically, a two-hidden-layer network with nonlinear activations would be sufficient. We fix the subnetwork structure to $[10, 6]$ with hyperbolic tangent activations. The initial learning rate η is not sensitive to the performance as it is sufficiently small, so we fix it to 0.001. It is worth mentioning that for Cayley transform, an adaptive scheme for the step size τ was proposed by Wen and Yin (2013), while here we fix $\tau = 0.1$ to be a relatively small value for the guaranteed gradient descent, which makes the network training relatively straightforward.

For the last three hyperparameters $(\lambda_1, \lambda_2, \lambda_3)$, we simply employ the grid search method and use the hold-out validation to choose the optimal parameter configurations. Other hyperparameter tuning methods with better global optimization performances are also being investigated, including random search, Bayesian optimization method that maximizes the expected improvement, and the sequential space-filling design as in the context of automated machine learning. These results will be reported in our future work.

4 Numerical Experiments

In this section, we compare the proposed SOSxNN method to several benchmark models through simulation studies under multiple scenarios. It is shown that the intrinsically in-

terpretable SOSxNN is flexible enough to approximate complex functions and achieve high prediction accuracy. We also include a real case study based on the Lending Club dataset as a showcase of the SOSxNN application.

4.1 Experimental Setting

Several benchmark models are considered for comparison, including the original xNN model by Vaughan et al. (2018), the multi-layer perceptron (MLP), the support vector machine (SVM), the random forest (RF), the least absolute shrinkage and selection operator (LASSO).

Given a dataset, we split it into training, validation and testing sets, where the validation set is used to determine the hyperparameters based on the simple grid search method. We specify the configuration of tuning hyperparameters for each benchmark model as follows. For LASSO, the shrinkage parameter is tuned within $\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$. For SVM based on the radial basis function (RBF) kernel, the best regularization parameter and kernel parameter are searched within the grid $\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\} \times \{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$. For MLP, we use one hidden layer with ten hyperbolic tangent nodes. For fair comparison, the optimization setting for MLP is specified the same as the proposed SOSxNN model. For RF with 100 base trees, the maximum tree depth and the minimum leaf number required for tree splitting are selected from $\{3, 4, 5, 6, 7, 8\} \times \{20, 50, 100\}$. Both SOSxNN and xNN use the same neural network structure for subnetwork specification, as discussed in Section 3.2, while all the other hyperparameters are optimized by the grid search method. For fair comparison, the hyperparameter tuning and network training schemes are kept the same for SOSxNN and xNN.

All the experiments are implemented using Python environment on a server with 64 Intel Xeon 2.60G CPUs. In particular, both SOSxNN and xNN are implemented using the "TensorFlow" package, while all the other benchmark models (LASSO, LogR, SVM, RF, and MLP) are implemented using the "Scikit-Learn" package.

4.2 Simulation Study

We consider four different modeling scenarios under the regression settings. In all the examples, the multivariate features are generated by the following mechanism:

a) Generate the 10-dimensional $\boldsymbol{z}$ randomly from Unif$(-1, 1)$;

b) Generate pairwise correlated features by $x_j = \frac{z_j + tu}{1+t}$ for $j = 1, 2, \ldots, 10$, where t is chosen such that the correlation coefficient $\rho = \frac{t^2}{1-t^2} = 0.5$;

Then, in each example, the response y is generated according to different functional relationships as specified below.

Scenario 1: Additive function with orthogonal projection. This is a simple case that follows the SOSxNN model assumption. It consists of four additive components and their projection indexes are mutually orthogonal.

$$y = h_1(\boldsymbol{w}_1^T\boldsymbol{x}) + h_2(\boldsymbol{w}_2^T\boldsymbol{x}) + h_3(\boldsymbol{w}_3^T\boldsymbol{x}) + h_4(\boldsymbol{w}_4^T\boldsymbol{x}) + \varepsilon, \tag{9}$$

in which the weights and ridge functions are given by

$$\boldsymbol{W}^T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0.3 & 0.5 & 0 & 0 & 0 \end{bmatrix},$$

$$h_1(z) = 2z,\ h_2(z) = 0.2e^{-4z},$$
$$h_3(z) = 3z^2,\ h_4(z) = 2.5\sin(\pi z).$$

It is noted that the last three features are treated as nuisance variables.

Scenario 2: Additive function with near-orthogonal projection. In this scenario, the function is also in additive form but the projection indexes are not orthogonal. The response is generated by

$$y = 3 + h_1(\boldsymbol{w}_1^T\boldsymbol{x}) + h_2(\boldsymbol{w}_2^T\boldsymbol{x}) + h_3(\boldsymbol{w}_3^T\boldsymbol{x}) + \varepsilon, \tag{10}$$

and the weights and ridge functions given by

$$\boldsymbol{W}^T = \begin{bmatrix} 0.1 & 0.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0.9 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$h_1(z) = 0.5z,\ h_2(z) = \frac{4\sin(\pi z)}{2 - \sin(\pi z)},\ h_3(z) = -4\exp(-z^2).$$

Note that such functional setting is adopted from Ruan and Yuan (2010) and Xia (2008).

Scenario 3: Non-additive function with orthogonal projection. Consider the multi-index example adopted from Xia (2008) with the following functional relationship

$$y = \boldsymbol{w}_1^T\boldsymbol{x} + \frac{4\boldsymbol{w}_2^T\boldsymbol{x}}{0.5 + (1.5 + \boldsymbol{w}_3^T\boldsymbol{x})^2} + \varepsilon, \tag{11}$$

$$\boldsymbol{W}^T = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & -0.5 & 0.5 & -0.5 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

In this case, it does not follows the AIM model as not all the components are additive.

Scenario 4: Non-additive function with non-orthogonal projection. Consider both non-orthogonal projection indexes and non-additive ridge functions. The response is generated with

$$y = \exp(\boldsymbol{w}_1^T \boldsymbol{x}) \sin(\pi \boldsymbol{w}_2^T \boldsymbol{x}) + \varepsilon, \tag{12}$$

$$\boldsymbol{W}^T = \begin{bmatrix} 0 & 0.5 & 0.5 & -0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.5 & 0 & 1 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

In all the four scenarios, we set the white noise $\varepsilon \sim N(0, 1)$. Four different sample sizes are considered, including, $n = \{1000, 2000, 5000, 10000\}$. Each data is further split into two parts with 80% for training and 20% for validation. Moreover, a separate testing set with 10000 samples is generated following the same data generation mechanism.

Tables 1–4 summarize the results of the compared models under Scenario 1 to 4, respectively. The experiments are repeated for 10 times with averaged prediction accuracy being reported, where the prediction accuracy is measured on the test set by the mean square error (MSE). In particular, the best results are highlighted in bold. Even though Scenario 2, 3, 4 are generated by functions that do not follow the assumption of additive function with orthogonal projection, the SOSxNN can still show its strong approximation ability. In most cases, it ranks the best or nearly the best. Therefore, we can conclude that the proposed SOSxNN model is competitive regarding the prediction accuracy. On the other hand, the LASSO model is relatively weak in terms of prediction accuracy, although the fitted sparse model by LASSO can be easily interpreted.

The main advantage of the proposed SOSxNN over the compared models lies in its ability in balancing prediction accuracy and model interpretability, as the proposed SOSxNN is intrinsically explainable. It takes the form of additive decomposition, while each ridge function and the corresponding projection index can be easily interpreted. For illustration, we draw the estimated ridge functions and projection indexes by SOSxNN vs. xNN in the case of sample size being 10000, as shown in Figs. 2–5. For both Scenarios 1 and 2, the left, middle and right sub-figures of Figs. 2–3 represent the ground truth, the SOSxNN estimation, and the xNN estimation, respectively. For Scenario 3 and 4 with non-additive underlying functions, we cannot visualize their ground truth as for Scenarios 1 and 2. For each sub-figure, the left panel shows the ridge functions and the right panel is the corresponding projection indexes. Vertically, the ridge functions and corresponding projection indexes are sorted in the descending order by their percentage scales, where the scale reflects the captured amount of functional variation per each additive component.

In Scenario 1, the sine curve $h_4(z)$ is the dominant component, followed by the exponential curve $h_2(z)$ and the linear curve $h_1(z)$. The quadratic term $h_3(z)$ takes the minimal

Table 1: Testing MSE comparison under Scenario 1.

Sample	SOSxNN	xNN	SVM	RF	MLP	LASSO
1000	**1.100**	1.213	1.266	1.723	1.285	2.619
2000	**1.058**	1.163	1.184	1.446	1.115	2.603
5000	**1.034**	1.128	1.100	1.313	1.042	2.586
10000	**1.018**	1.089	1.071	1.264	1.021	2.604

Table 2: Testing MSE comparison under Scenario 2.

Sample	SOSxNN	xNN	SVM	RF	MLP	LASSO
1000	**1.095**	1.154	1.436	1.212	1.250	2.118
2000	**1.024**	1.107	1.292	1.120	1.116	2.132
5000	**1.022**	1.087	1.170	1.077	1.045	2.152
10000	**1.009**	1.085	1.114	1.053	1.022	2.129

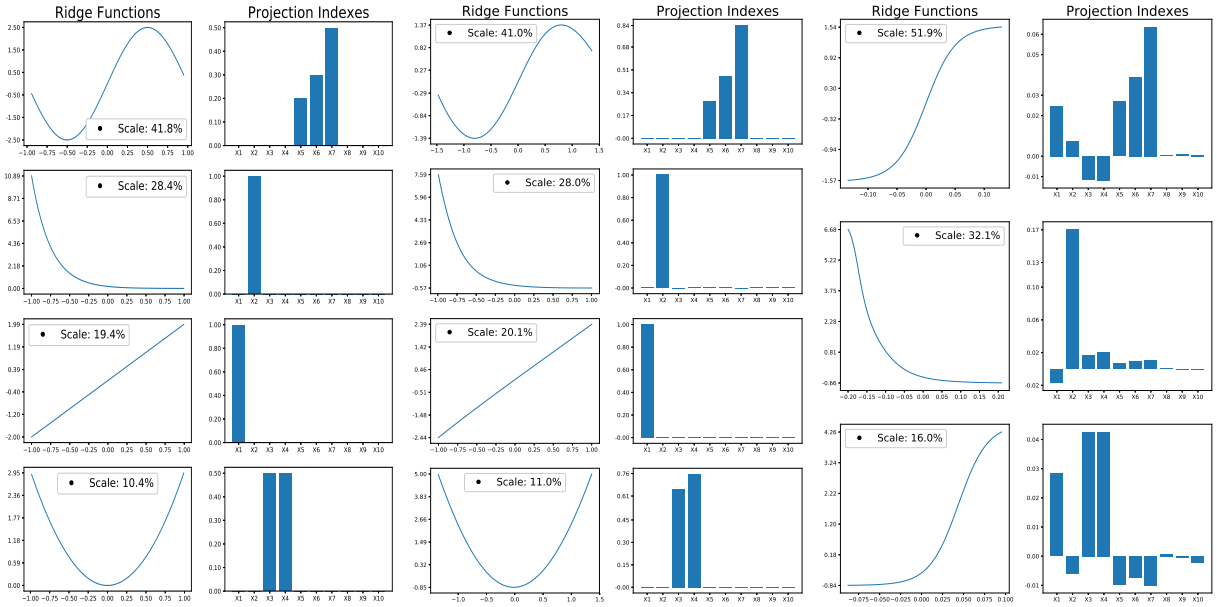
Table 3: Testing MSE comparison under Scenario 3.

Sample	SOSxNN	xNN	SVM	RF	MLP	LASSO
1000	**1.049**	1.081	1.056	1.140	1.272	1.243
2000	**1.033**	1.064	1.037	1.100	1.120	1.235
5000	**1.020**	1.023	1.023	1.071	1.045	1.240
10000	1.019	**1.010**	1.016	1.050	1.022	1.248

Table 4: Testing MSE comparison under Scenario 4.

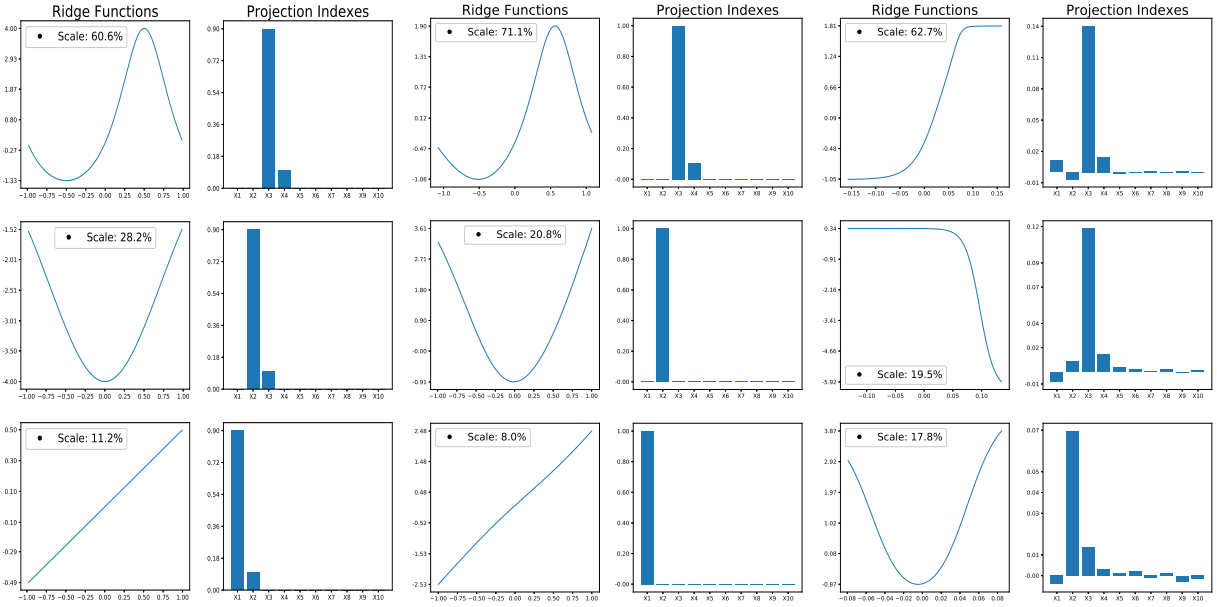
Sample	SOSxNN	xNN	SVM	RF	MLP	LASSO
1000	**1.100**	1.118	1.195	1.175	1.288	1.307
2000	1.065	**1.064**	1.145	1.108	1.144	1.297
5000	1.057	1.041	1.079	1.062	**1.038**	1.290
10000	1.032	1.041	1.055	1.049	**1.019**	1.288

variation, which is around 10% of the total variation. It can be seen in Fig. 2 that the four ridge functions and corresponding projections are all successfully recovered by SOSxNN, with approximately correct scale estimates. However, the estimated effects by xNN are hard to explain. They suffer from dense and correlated projection indexes as well as incorrect ridge functions. For instance, the linear component gets mixed with the sine curve in both projection indexes and ridge functions. This is because the linear component $h_1(z)$ can be



(a) Ground Truth (b) SOSxNN (c) xNN

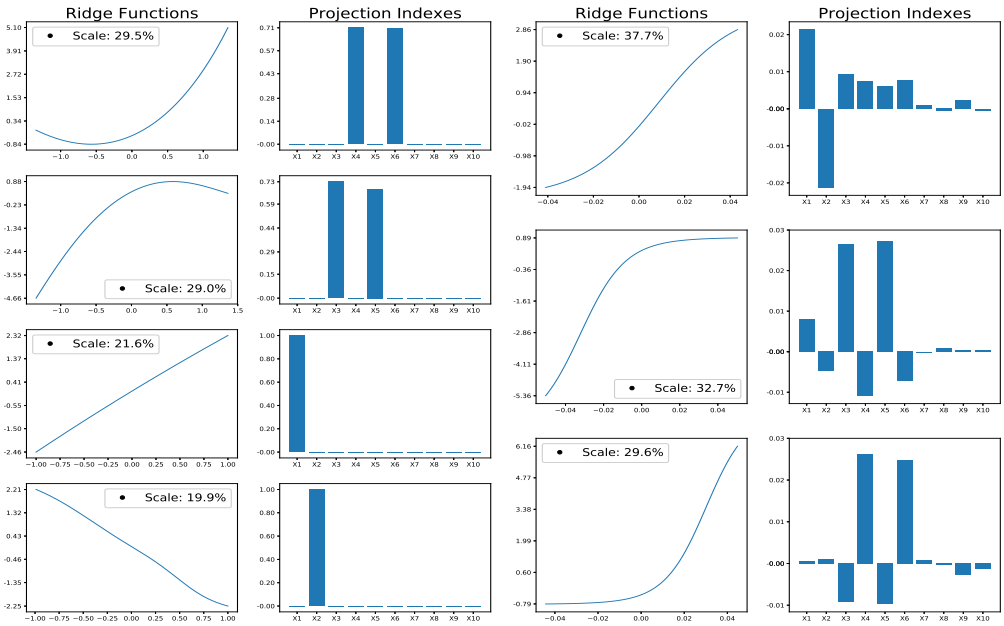
Figure 2: Visualized model fits (vs. the ground truth) for Scenario 1.



(a) Ground Truth (b) SOSxNN (c) xNN

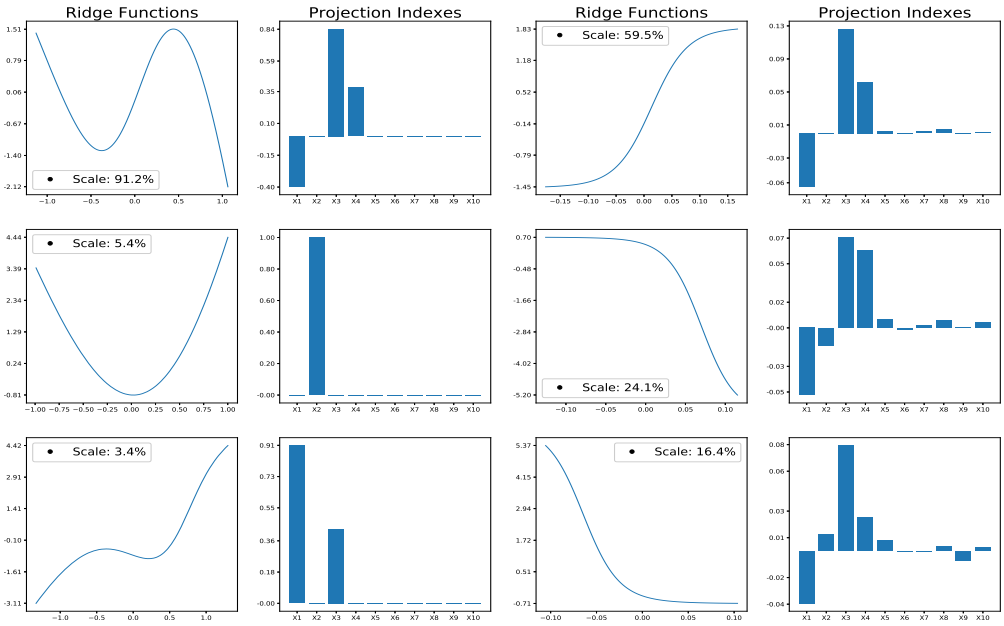
Figure 3: Visualized model fits (vs. the ground truth) for Scenario 2.

easily confounded with other components due to the identifiability issue. With the orthogonality constraint, such confounding problem can be effectively avoided. Moreover, it can be seen that the projection indexes obtained by xNN are less sparse and mutually correlated,



(a) SOSxNN (b) xNN

Figure 4: Visualized model fits for Scenario 3.



(a) SOSxNN (b) xNN

Figure 5: Visualized model fits for Scenario 4.

which makes the interpretation difficult.

Scenario 2 is designed with near-orthogonal projection indexes. Interestingly, the SOSxNN can still find an orthogonal approximation to the ground truth model. As shown in Fig. 3, the

main effect $h_2(z)$ is captured with almost correct ridge function and projection indexes. The less important $h_1(z)$ and $h_3(z)$ are also approximated with univariate coefficients. Referring to Table 2, we can see such an approximation generalizes well on the testing set compared to the benchmark models. In contrast, the estimation results by xNN are less explainable. For example, the projection indexes of the first two components are highly correlated, and their ridge functions looks totally different. The linear term takes 10% of the total variation in the ground truth, but xNN fails to detect such an effect.

Although Scenario 3 and 4 do not take the additive functional form, both xNN and SOSxNN can find good approximations to such complex functions, with relatively high accuracy as compared to other benchmark models. Such approximations can be easily visualized and interpreted. As shown in Fig. 4a, SOSxNN reveals that (x_3, x_5) and (x_4, x_6) are two different groups of variables with nonlinear ridge functions, and each group takes about 30% of the total variation; the variables x_1 and x_2 are instead linearly correlated to the response. In Fig. 5a, it can be seen that 91.2% of the total variation is explained by a sine-like curve, subject to the projection of (x_1, x_3, x_4). The rest 8.8% of variation is explained by another two components with orthogonal projection indexes. In contrast, the xNN model tends to estimate highly correlated projections. The estimated projection weights for each component are less sparse, which makes the xNN result hard to interpret.

4.3 Lending Club Dataset

The peer-to-peer (P2P) lending is a method of lending money through online services by matching individual lenders and borrowers. It has been one of the hottest FinTech applications. The Lending Club dataset is obtained from (`https://www.lendingclub.com/info/download-data.action`), including all the issued loans and declined loan applications that do not meet Lending Club credit underwriting policy, from Jan. 2015 to Jun. 2018. Each sample represents a loan application with six features and the binary flag indicating the approval result (approved or declined). In particular, a data cleaning procedure is implemented to remove samples with missing values or outliers. We delete the cases whose risk score is greater than 850, as that is out of the range of FICO score. Moreover, the samples where the debt-income-ratio is within the range of 0 – 200% are kept. As a result, the cleaned dataset has 1,433,570 accepted applications, and 5,611,316 declined cases. Such dataset is then split into three parts, with 40% for training, 10% for validation and the rest 50% for testing.

The loan purpose is a categorical variable with multiple categories, and we group these various purposes into five categories, including "Credit Card", "Debt Consolidation", "Housing", "Purchase" and "Others". A summary of the dataset is given in Table 5. Through one-hot-encoding, the categorical variable "Loan Purpose" is represented by five dummy

Table 5: Description of the Lending Club Dataset.

No.	Variables	Range
X1	Application Date	Jan. 2015 - Jun. 2018
X2	Amount Requested	150 - 166650
X3	Risk Score	300 - 850
X4	Debt Income Ratio	0% - 200%
X5	Employment Length	0 - 10 & 11 (more than 10)
X6	Load Purpose	"Debt Consolidation", "Credit Card", "Housing", "Purchases", and "Others"

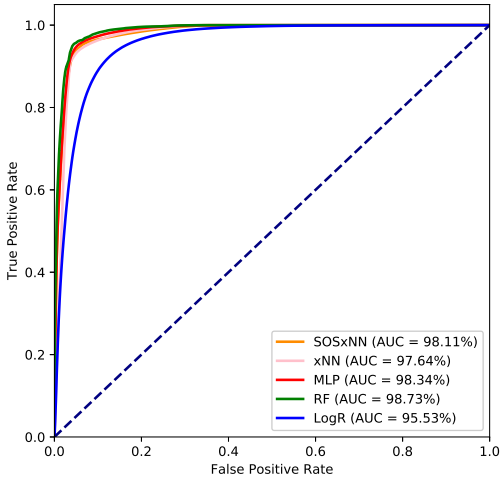
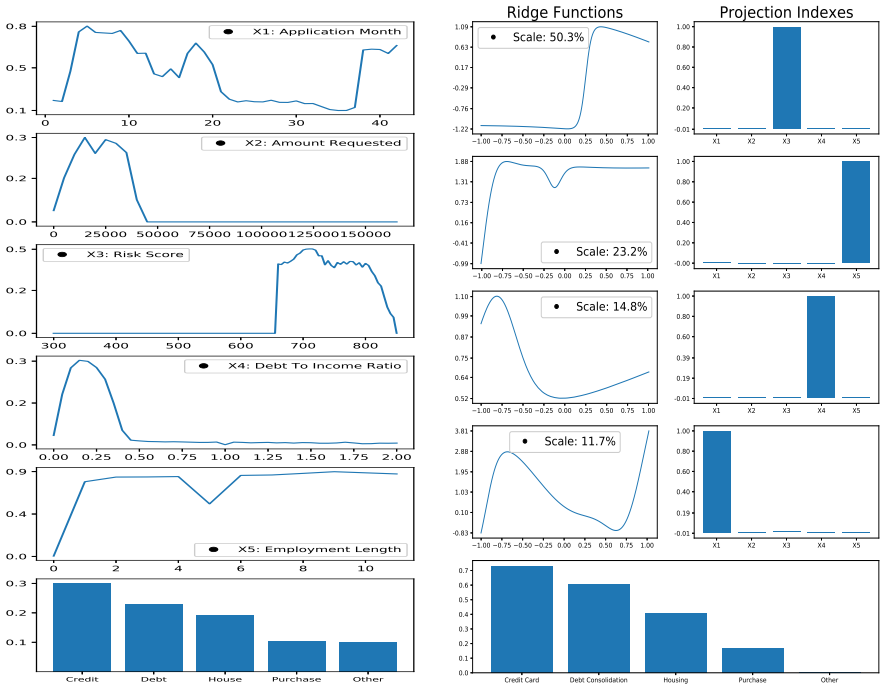


Figure 6: The ROC curves of compared methods for the Lending Club dataset.

variables with values between 0 and 1. In the SOSxNN and xNN model, these dummy variables can be treated as the bias terms and we use the category "Others" as the baseline. Due to the huge sample size, the SVM model is not applicable, and we compare the proposed SOSxNN model with the LogR (logistic regression), the RF, the MLP, and the xNN model.

The receiver operating characteristic (ROC) curves of different models are plotted in Fig. 6, together with the area under curve (AUC) scores shown in the bottom right corner. In terms of the AUC scores, the RF model performs the best, followed by MLP, SOSxNN, xNN and LogR. Despite the high predictive performance, both the blackbox types of RF and MLP are too complex to interpret. On the contrary, the LogR performs the worst, but the estimated model can be easily interpreted. Compared to these benchmark models, the



(a) Marginal Plot (b) SOSxNN

Figure 7: Visualized SOSxNN model fits for the Lending Club dataset. The left sub-figure shows the marginal rates with respect to both numerical and categorical variables. The right sub-figure shows the SOSxNN model estimates with the component functions sorted in the descending order of functional variation scales.

proposed SOSxNN method achieves the relatively high accuracy, while the estimated model is essentially interpretable.

The estimated subnetworks and projection indexes of SOSxNN are visualized in Fig. 7. Without any prior knowledge, SOSxNN automatically selects a sparse AIM that follows the approximately GAM structure. Compared to marginal rates on the left sub-figure of Fig. 7, the estimated ridge functions are more smooth and can provide quantitative ordering of different variables in terms of explaination power. More specifically, we have the following findings. First, the risk score (X3) is the most important feature for a loan application, as it is estimated to explain 50.3% of the total variation. It is found that the risk scores around 700 are generally preferred, but extremely high scores may negatively impact the application. Second, the employment length (X5) accounts for the second important feature (with scale of 23.2%), and it will impact applicants who have fewer than two years of working experience, although it also shows a small drop around 5 years of working experience. Third, the model suggests that a moderate debt-income-ratio (X4) is preferable. The application date (X1) will also influence the application result, as the loan acquisition policy may change over

time (one possible reason can be referred to the shortage of funds in the markets). Among the numerical variables, the loan amount (X2) turns out not a significant factor and is not included in the fitted SOSxNN model. Finally, the bar plot shows the intercepts for different loan purposes (X6). Loans with the purpose of "Credit card" and "Debt consolidation" are shown to enjoy high approval rates than the other three categories.

4.4 Summary of Results

We summarize the above experimental results from the following two perspectives.

a) The proposed SOSxNN model is flexible enough to decompose a complex relationship into several additive components. Compared with the popularly used machine learning algorithms, the proposed model is competitive in the sense of prediction accuracy. The numerical results show that when the true underlying model satisfies the additive model assumption with orthogonal or near-orthogonal projection indexes, the SOSxNN outperforms its counterparts. In more general settings with highly complicated (known or unknown) functions, the proposed model can be a promising surrogate model for complex model approximation.

b) The proposed SOSxNN model is inherently explainable with additive, sparsity, orthogonal and smoothness considerations. As shown in the numerical experiments, most of the main effects can be nicely captured or well explained. Compared to the original xNN or AIM model, the SOSxNN leads to a more identifiable model with sparse and uncorrelated projection weights, which enhances the explainability of neural networks for modeling complex relationships.

Therefore, the proposed SOSxNN method provides an effective approach that balances between predictive accuracy and model explainability. It is justified that the SOSxNN is a promising tool for interpretable machine learning.

5 Conclusion

This paper proposes a new neural network model with enhanced explainability in terms of additive decomposition, sparsity, orthogonal projection, and smooth function approximation. First, a complex function can be decomposed into sparse additive univariate functions, which is straightforward for model interpretation. Second, the projection indexes are enforced to be orthogonal and sparse, which makes it ease for model understanding. Lastly, we also consider the smoothness of each ridge function as an additional constraint. Numerical experiments

based on simulation and real-world datasets demonstrate that the proposed SOSxNN has competitive prediction performance compared to existing machine learning models. More importantly, it is designed to be intrinsically explainable.

Some research topics are promising for future study. It is demonstrated in this paper that the classical statistical models can be reformulated by neural network architectures subject to flexible constraints. It is desirable to consider a broader class of statistical models such that the new developments of neural network training techniques can benefit especially for large-scale statistical modeling problems. Second, it is interesting the study in-depth the statistical properties of the SOSxNN model regarding the model identifiability conditions and optimality conditions. Third, it is also of our interest to investigate the use of SOSxNN to model the main effects and interaction effects in functional analysis of variance.

Acknowledgment

We thank Vijay Nair and Joel Vaughan from Wells Fargo for helpful discussions. This research project was partially supported by the Big Data Project Fund of The University of Hong Kong from Dr Patrick Poon's donation.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Apley, D. W. (2016). Visualizing the effects of predictor variables in black box supervised learning models. *arXiv:1612.08468*.

Ba, J. and Caruana, R. (2014). Do deep nets really need to be deep? In *Advances in Neural Information Processing Systems (NIPS)*, pages 2654–2662.

Bair, E., Hastie, T., Paul, D., and Tibshirani, R. (2006). Prediction by supervised principal components. *Journal of the American Statistical Association*, 101(473):119–137.

Barshan, E., Ghodsi, A., Azimifar, Z., and Jahromi, M. Z. (2011). Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds. *Pattern Recognition*, 44(7):1357–1371.

Brainard, L. (2018). What are we learning about artificial intelligence in financial services? `https://www.federalreserve.gov/newsevents/speech/brainard20181113a.htm`.

Bucilua, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 535–541. ACM.

Fawzi, A., Fiot, J. B., Chen, B., Sinn, M., and Frossard, P. (2016). Structured dimensionality reduction for additive model regression. *IEEE Transactions on Knowledge and Data Engineering*, 28(6):1589–1601.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232.

Friedman, J. H. and Stuetzle, W. (1981). Projection pursuit regression. *Journal of the American Statistical Association*, 76(376):817–823.

Goldstein, A., Kapelner, A., Bleich, J., and Pitkin, E. (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, 24(1):44–65.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Gunning, D. (2017). Explainable artificial intelligence (XAI). *Defense Advanced Research Projects Agency (DARPA)*.

Hastie, T. and Tibshirani, R. (1990). *Generalized Additive Models*. Chapman & Hall, London.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv:1503.02531*.

Hu, L., Chen, J., Nair, V. N., and Sudjianto, A. (2018). Locally interpretable models and effects based on supervised partitioning (LIME-SUP). *arXiv:1806.00663*.

Ichimura, H. (1993). Semiparametric least squares (sls) and weighted sls estimation of single-index models. *Journal of Econometrics*, 58(1-2):71–120.

Jolliffe, I. T. (1982). A note on the use of principal components in regression. *Applied Statistics*, pages 300–303.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv:1412.6980*.

Liu, X., Chen, J., Nair, V., and Sudjianto, A. (2018). Model interpretation: A unified derivative-based framework for nonparametric regression and supervised machine learning. *arXiv:1808.07216*.

Raskutti, G., Wainwright, M. J., and Yu, B. (2012). Minimax-optimal rates for sparse additive models over kernel classes via convex programming. *Journal of Machine Learning Research*, 13(Feb):389–427.

Ravikumar, P., Lafferty, J., Liu, H., and Wasserman, L. (2009). Sparse additive models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(5):1009–1030.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). Why should I trust you? Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM.

Ruan, L. and Yuan, M. (2010). Dimension reduction and parameter estimation for additive index models. *Statistics and Its Interface*, 3(4):493–499.

Vaughan, J., Sudjianto, A., Brahimi, E., Chen, J., and Nair, V. N. (October 2018). Explainable neural networks based on additive index models. *The RMA Journal*, pages 40–49.

Wahba, G. (1990). *Spline Models for Observational Data*, volume 59. SIAM.

Wang, J., Xu, C., Yang, X., and Zurada, J. M. (2018). A novel pruning algorithm for smoothing feedforward neural networks based on group lasso method. *IEEE transactions on Neural Networks and Learning Systems*, 29(5):2012–2024.

Wen, Z. and Yin, W. (2013). A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434.

Wisdom, S., Powers, T., Hershey, J., Le Roux, J., and Atlas, L. (2016). Full-capacity unitary recurrent neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4880–4888.

Xia, Y. (2008). A multiple-index model and dimension reduction. *Journal of the American Statistical Association*, 103(484):1631–1640.

Yuan, M. (2011). On the identifiability of additive index models. *Statistica Sinica*, 21:1901–1911.

Zou, H., Hastie, T., and Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286.