Explainable Neural Networks (xNN) based on Additive Index Models: An overview

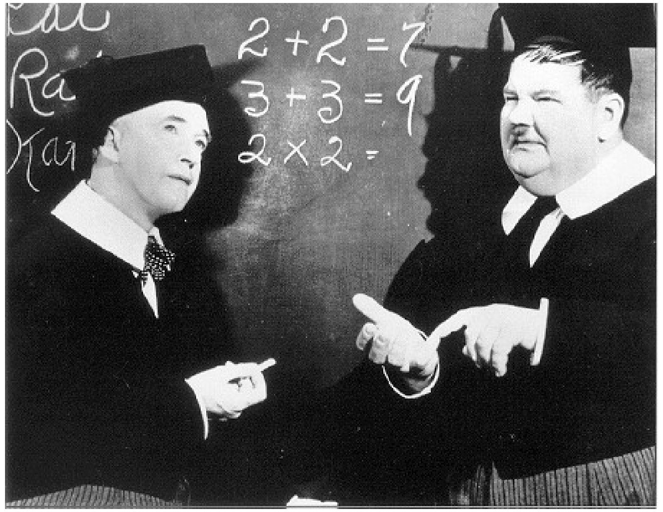
Hassan Omidi Firouzi

RBC Wholesale Credit Risk

July 23, 2018

This presentation is based on the article submitted on **arxiv.org** by Joel Vaughan et al. from **Wells Fargo Bank**.

“Some people are quite capable of seeing things they don’t understand and being okay with it. I wasn’t okay that something had violated my model of the world. I really am not okay with things that do that,” Geoff, Hinton



"Essentially, all models are wrong, but some are useful.“ Box, George E. P.

Should we care about explainability and interpretability of models?

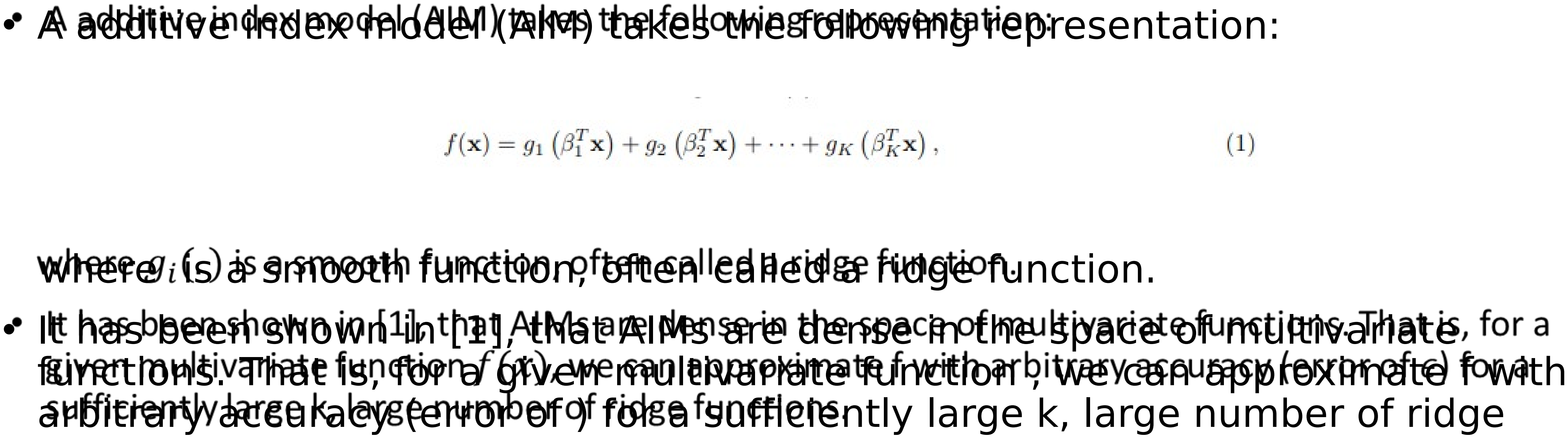
- Why interpretability/explainability are important characteristics of a fitted model?
- At what cost can we come up with an interpretable model? Trade-off between model predictive performance and model interpretability;
- Available approaches aiming at understanding the black-box models like NN models:
  - Additive feature attribution methods:
    - Local Interpretable Model-agnostic Explanation, LIME, (see [4]);
    - Layer-wise relevance propagation, LRP, (see [3]).
- How about introducing a NN model which has built-in interpretation mechanism (to some degree)?
  - Is xNN a class of NN models having a built-in interpretation mechanism?
  - Can xNN be used as a surrogate model?

Additive Index Models (AIMs)

- A additive index model (AIM) takes the following representation:

  where is a smooth function, often called a ridge function.

- It has been shown in [1], that AIMs are dense in the space of multivariate functions. That is, for a given multivariate function , we can approximate f with arbitrary accuracy (error of) for a sufficiently large k, large number of ridge functions.



- A additive index model (AIM) takes the following representation:

$$f(\mathbf{x}) = g_1\left(\beta_1^T\mathbf{x}\right) + g_2\left(\beta_2^T\mathbf{x}\right) + \cdots + g_K\left(\beta_K^T\mathbf{x}\right), \quad (1)$$

where $g_i(.)$ is a smooth function, often called a ridge function.

- It has been shown in [1], that AIMs are dense in the space of multivariate functions. That is, for a given multivariate function $f(x)$, we can approximate f with arbitrary accuracy (error of ϵ) for a sufficiently large k, large number of ridge functions.

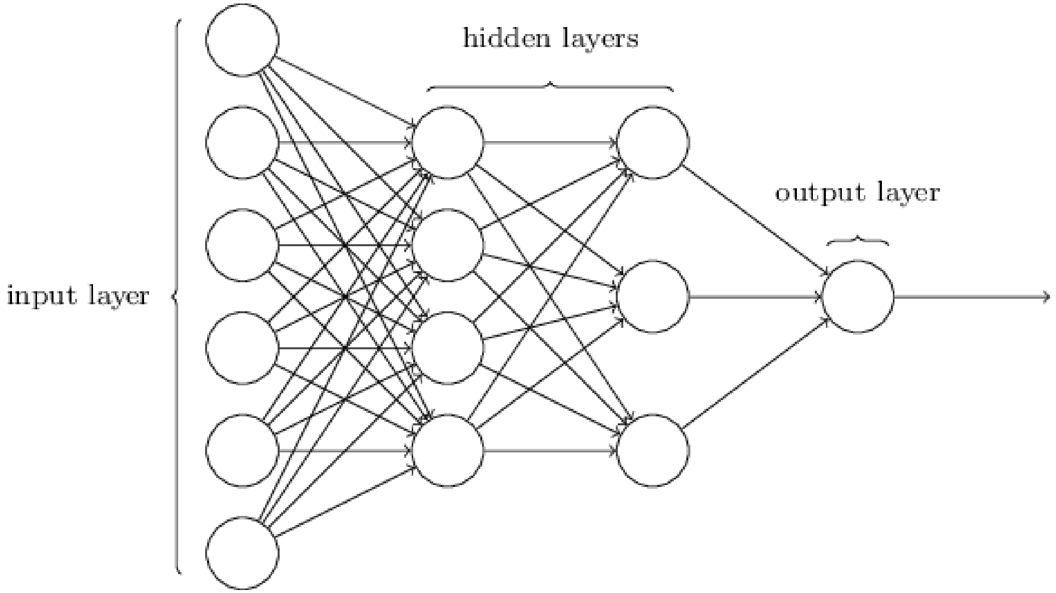
The underlying model for xNN fitting

- xNN is simply based on AIMs introduced in Equation (1). The goal is to fit the following modified version of Equation (1) via a feedforward NN.

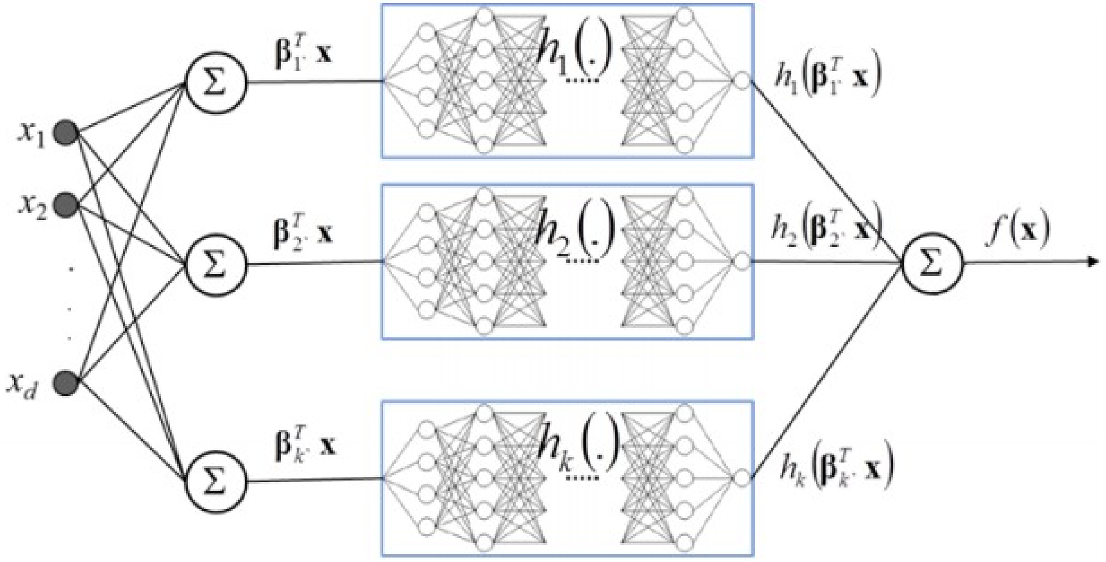
$$f(\mathbf{x}) = \mu + \gamma_1 h_1 \left(\beta_1^T \mathbf{x}\right) + \gamma_2 h_2 \left(\beta_1^T \mathbf{x}\right) + \cdots + \gamma_K h_K \left(\beta_K^T \mathbf{x}\right). \tag{2}$$

- The xNN fits Equation (2) using three components [5]:
  - The first hidden layer called the ***projection*** layer;
  - ***Subnetworks***; and
  - The ***combination*** layer.

Fully connected Feedforward NN architecture



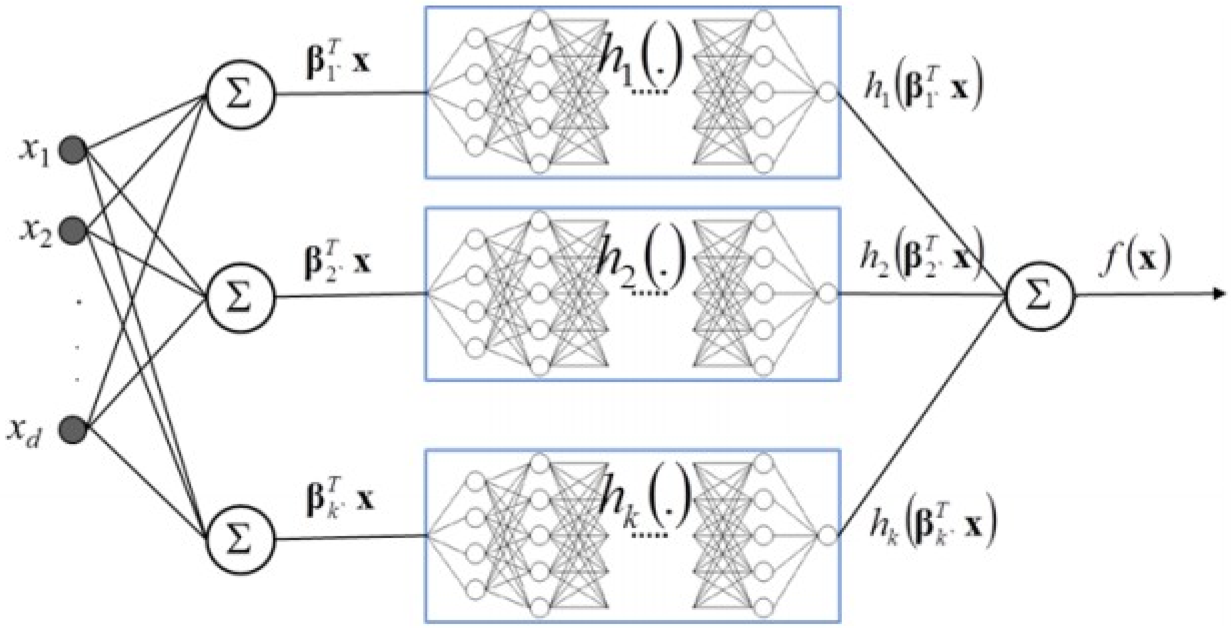
xNN model architecture


x_1
x_2
x_d
Σ
$\boldsymbol{\beta}_{1\cdot}^T \mathbf{x}$
$\boldsymbol{\beta}_{2\cdot}^T \mathbf{x}$
$\boldsymbol{\beta}_{k\cdot}^T \mathbf{x}$
$h_1(.)$
$h_2(.)$
$h_k(.)$
$h_1(\boldsymbol{\beta}_{1\cdot}^T \mathbf{x})$
$h_2(\boldsymbol{\beta}_{2\cdot}^T \mathbf{x})$
$h_k(\boldsymbol{\beta}_{k\cdot}^T \mathbf{x})$
Σ
$f(\mathbf{x})$

Explainable Neural Network Architecture (xNN)

- Projection layer uses linear activation function;
- The weights of the node i in the projection layer corresponds to the coefficient in Equation (2);
- The output of each node in the projection layer is used as the input to exactly one subnetwork;
- Each subnetwork is used to learn one of the ridge functions introduced in Equation (2);
- Subnetworks typically consist of multiple fully-connected layers and use nonlinear activation functions;
- The inputs of the final node, combination layer, are the univariate activations of all of the subnetworks; and
- The weights learned in the final layer correspond to the ’s in Equation (2).

- Projection layer uses linear activation function;
- The weights of the node i in the projection layer corresponds to the coefficient β_i in Equation (2);
- The output of each node in the projection layer is used as the input to exactly one subnetwork;
- Each subnetwork is used to learn one of the ridge functions introduced in Equation (2);
- Subnetworks typically consist of multiple fully-connected layers and use nonlinear activation functions;
- The inputs of the final node, combination layer, are the univariate activations of all of the subnetworks; and
- The weights learned in the final layer correspond to the γ_i’s in Equation (2).



Example 1: The First Three Legendre Polynomials

- These polynomials are orthogonal on the interval [−1, 1] and have a range of [−1, 1] over the same interval.

$$f_1(x) = x; \qquad f_2(x) = \frac{1}{2}\left(3x^2 - 1\right); \qquad f_3(x) = \frac{1}{2}\left(5x^3 - 3x\right) \qquad (3)$$

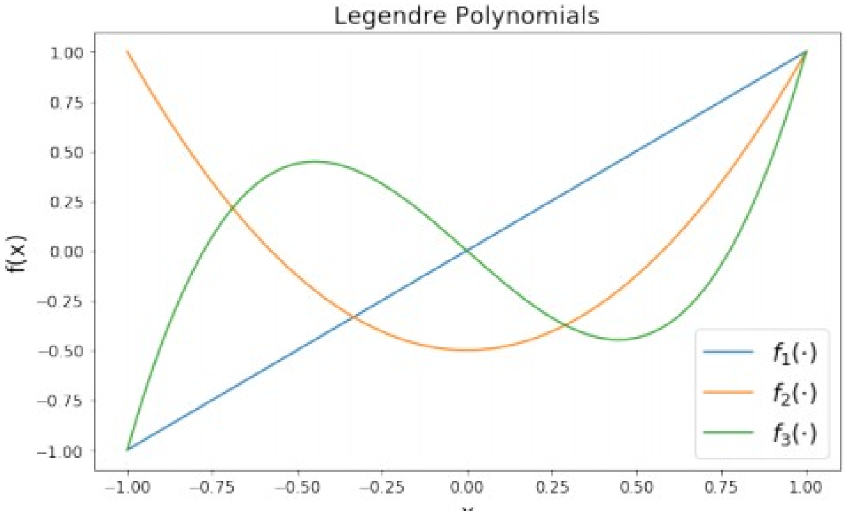
- Simulate five independent variables, from a Uniform distribution on [−1, 1].
- Define the function f as the summation of the three polynomials:

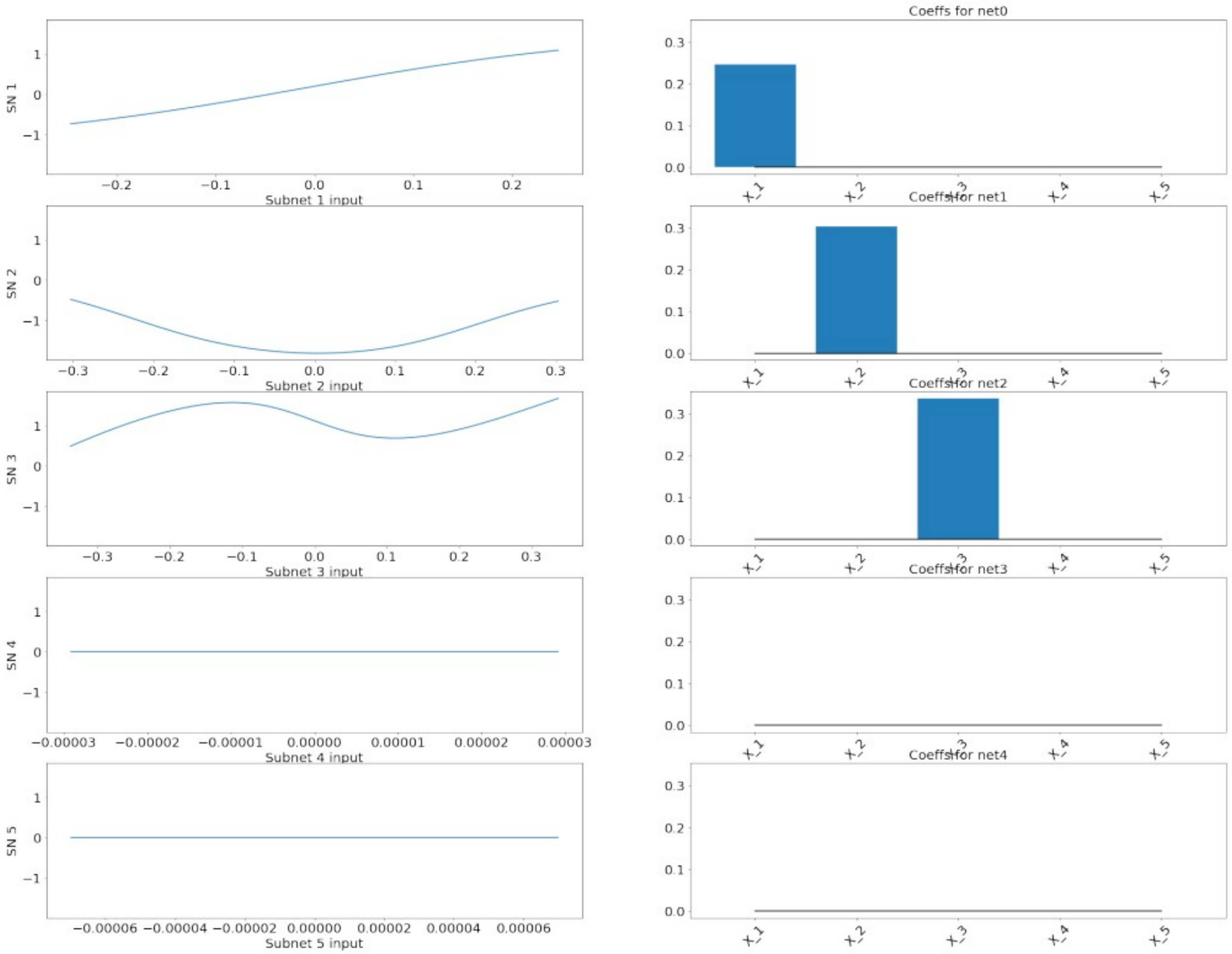
$$y = f_1(x_1) + f_2(x_2) + f_3(x_3) \qquad (4)$$

- The generated values from Equations (3) and (4) are then fitted to a xNN model with the following architecture:
  - Use all five simulated variables as features;
  - Each subnetwork consists of two hidden layers with structures such as [25, 10] or even [12,6] and nonlinear activation functions like **tanh**, sigmoid, ReLu, among others;
  - penalty has been used on the first and last hidden layers to enhance model explainability and reduce overfitting.

- These polynomials are orthogonal on the interval [−1, 1] and have a range of [−1, 1] over the same interval.
- Simulate five independent variables, $x_1, \ldots, x_5$ from a Uniform distribution on [−1, 1].
- Define the function f as the summation of the three polynomials:
- The generated values from Equations (3) and (4) are then fitted to a xNN model with the following architecture:
  - Use all five simulated variables as features;
  - Each subnetwork consists of two hidden layers with structures [25, 10] or even [12,6] and nonlinear activation function like **tanh**, sigmoid, ReLu, among others;
  - L_1 penalty has been used on the first and last hidden layers to enhance model explainability and reduce overfitting.

Example 1: The First Three Legendre Polynomials cont’d





- The first column illustrates the univariate functions learned by subnetwork i, scaled by
- . The second column displays the values of , the projection coefficient;

The first column illustrates the univariate functions learned by subnetwork i, scaled by γ_i;

. The second column displays the values of β_i, the projection coefficient;

Example 2: *Non-Linear Model*

- Simulate four independent variables, from a Uniform distribution on [−1, 1].
- Data generating process takes the following from:

$$y = \exp(x_1) \cdot \sin(x_2) + \epsilon \qquad \text{where } \epsilon \sim N(0, 0.1).$$

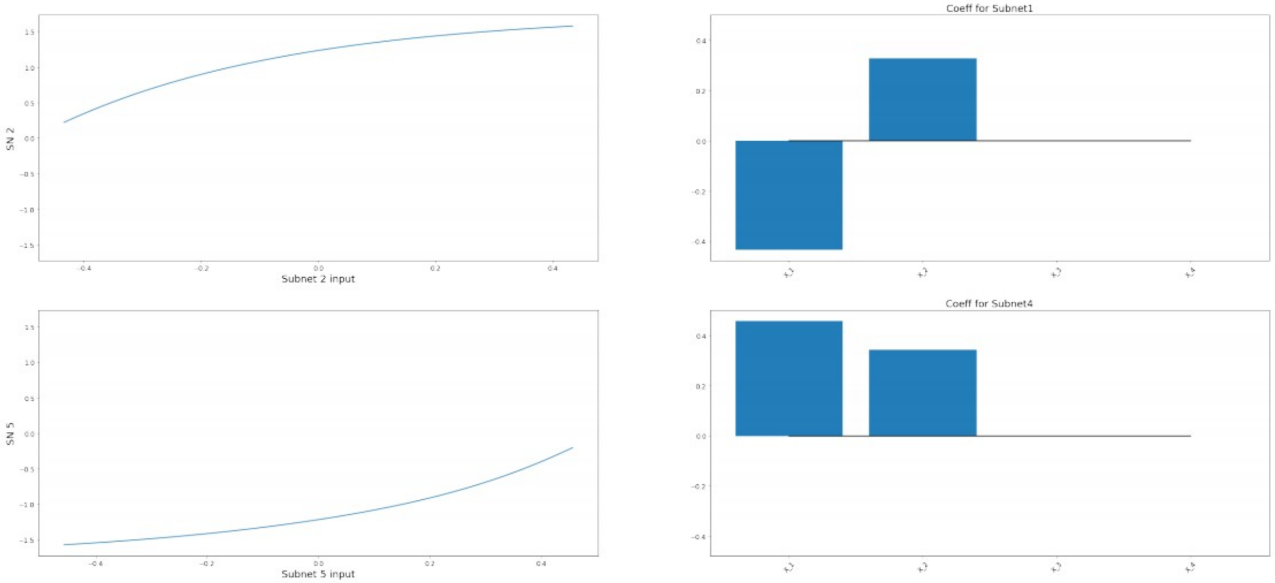
- To fit a xNN model:
  - All four simulated variables are used as features (and as noise variables);
  - 10 subnetworks and a subnet structure of [12,6] with tanh activation have been employed;
  - penalty has been used on the first and last hidden layers to enhance model explainability and reduce overfitting.

- Simulate four independent variables, $x_1, \ldots, x_4$ from a Uniform distribution on [−1, 1].
- Data generating process takes the following from:
- To fit a xNN model:
  - All four simulated variables are used as features (x_3 and x_4 as noise variables);
  - 10 subnetworks and a subnet structure of [12,6] with tanh activation have been employed;
  - l_1 penalty has been used on the first and last hidden layers to enhance model explainability and reduce overfitting.

Example 2: *Non-Linear Model cont’d*

- The xNN model cannot recover the data generating process; however, it still fits the data well;
- Only two out of ten subnetworks have at least one non-zero input coefficient, β_{ij}.
- The xNN approximates the simulated non-linear function with $f_2(-0.43\ x_1 + 0.33\ x_2) + f_5(0.46\ x_1 + 0.34\ x_2)$.

- The xNN model cannot recover the data generating process; however, it still fits the data well;
- Only two out of ten subnetworks have at least one non-zero input coefficient, .
- The xNN approximates the simulated non-linear function with + .



https://www.shutterstock.com

Thoughts/Questions on xNN model and Concluding remarks

- Is xNN a class of NN models with a built-in interpretation mechanism?
  - Answer: Yes, but partially!
- Can xNN be used as a surrogate model?
  - Answer: Theoretically xNN should provide a good approximation for any given function (based on AIMs models) if the architecture is chosen properly. However, in practice further investigations are needed!
- Does xNN say anything about the closed form of each trained ridge function?
  - Answer: Mostly No! since each subnetwork is again a black box, the problem of interpretability still applies to the ridge functions. Similar to non-parametric models, AIMs.
- How can one perform sensitivity analysis of ridge functions to the features? Which factors do contribute to the shape of each ridge function?
  - Answer: Since xNN provides the overall shape of each ridge function, one can change the input slightly to see the impact on the subnetworks' outputs. However, since there is no closed form for each ridge function we can’t **globally** draw a firm conclusion on the sensitivity analysis of ridge functions and consequently the whole xNN model to the input features;
  - Answer: It’s not clear!
- What is the impact of different non-linear activation functions and other predetermined parameters on the final model performance?
  - Answer: Not clear. Needs to be verified at least for cases with at most one linear and one quadratic ridge functions (in line with Theorem 2 in [2]).

References

1. Diaconis, P., et al. *“On nonlinear functions of linear combinations”,* SIAM J. Sci. and Stat. Comput 5(1), 175–191, 1984.
2. *Chen, Y., et al. “Generalized additive and index models with shape constraints”.* 2015.
3. Lundberg, S. M., et al. “*A Unified Approach to Interpreting Model Predictions”.* Nov., 2017*.*
4. *Ribeiro, M., et al. “Why Should I Trust You?” Explaining the Predictions of Any Classifier.* Aug., 2016.
5. Vaughan J., et al. “*Explainable Neural Networks based on Additive Index Models*”**.** June, 2018.