WELLS FARGO



# Adaptive Explainable Neural Networks (AxNN)

Jie Chen
Joint work with Joel Vaughan, Vijay Nair and Agus Sudjianto
Corporate Model Risk, Wells Fargo

June 5, 2020

# Importance of making ML transparent

- Supervised machine learning algorithms have very good predictive performance
- **But the biggest criticism is difficulty in interpretation … predictor $\hat{f}(x)$ is a `black box' – hard to interpret**
- True of all ensemble methods, SVM, neural network
- We need to be able to interpret and explain the results of ML algorithms:
  - Required by regulation
  - Get insights from the model and make scientific/business findings
- Some main questions to answer are
  - Which variables are important?
  - What is the input-output relationship for each important variable/a subset of important variables? Nonlinearity? Interaction?
  - How do correlations among variables impact the response surface?
  - How can we ensure the relationships from ML are consistent with historical and business understanding.
- Machine learning interpretation is an active research area now.

# Interpretation of Input-output relationship

**Existing tools for interpretation:**

- First order (main) effects
  - Partial dependence (PDP) plots (J. H. Friedman 2001)
  - Individual conditional expectation (ICE) plots (Goldstein, et al. 2015)
  - Accumulated local effects (ALE) plots (Apley 2016)
  - Accumulated total derivative effects (ATDEV) (Liu, et al. 2018)

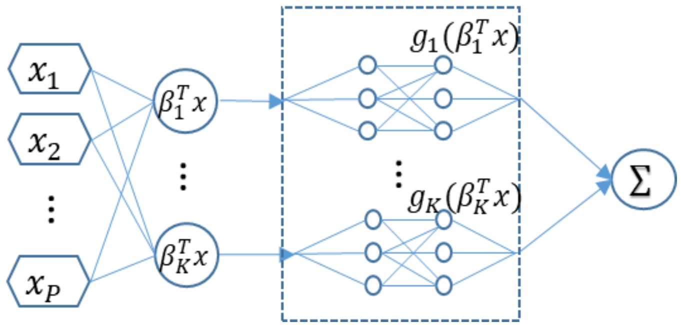
- Interaction detection
  - Post-hoc machine learning diagnostic tools
    - 2D PDA and H-statistics (J. H. Friedman 2001, Friedman and Popescu 2008)
  - Tree-based methods:
    - Additive groves of trees (Sorokina, et al. 2008)
    - GA2M, to estimate pairwise interactions (Lou, et al. 2013, Caruana, et al. 2015)
  - Neural network (NN) related methods
    - Interaction detection via MLP neural network learned weights (Tsang, Cheng and Liu 2017)
    - Disentangling Learned Interactions via NNs(Tsang, Liu, et al. 2018)

# Intrinsically Interpretable Models: Explainable Neural Network (xNN)

- Additive Index Model (AIM)

$$f(\boldsymbol{x}) = g_1(\boldsymbol{\beta}_1^T \boldsymbol{x}) + g_2(\boldsymbol{\beta}_2^T \boldsymbol{x}) + \ldots + g_K(\boldsymbol{\beta}_K^T \boldsymbol{x})$$

- XNN architecture



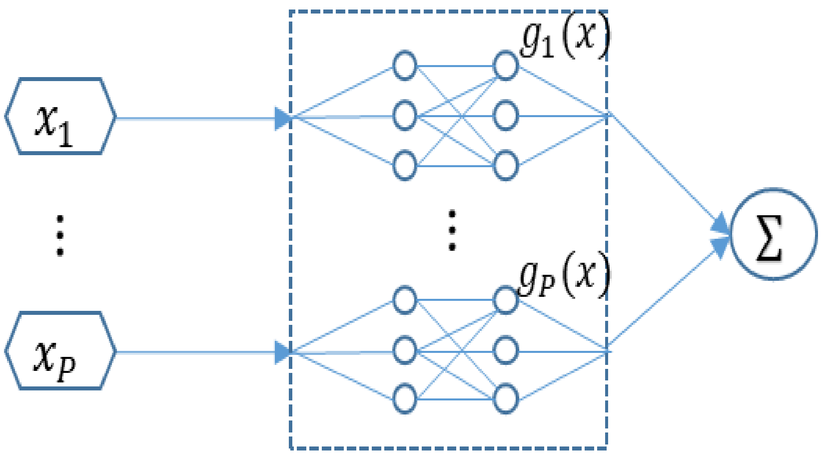

  - Structured network architecture designed to learn an AIM.
  - Network structure chosen to match features of AIM:
  - Two key structures:
    - **Projection Nodes**: Nodes with **linear** activation functions. Used for projections and final sums.
    - **Subnetwork**: Collection of nodes that:
      - Internally: fully connected, multi-layer, and use nonlinear activation functions.
      - Externally: only connected to rest of the network through a univariate input and univariate output.
      - Used to learn ridge functions, $g_k(\cdot)$

# Generalized additive model network (GAMnet)

- Generalized additive models (GAM)

$$f(\boldsymbol{x}) = g_1(x_1) + g_2(x_2) + \ldots + g_P\,(x_P)$$

- A special case of xNN: Unlike AIM, each ridge function has only a single dimensional input and captures just the main effects of the corresponding predictors:



# Adanet

- The AdaNet algorithm (Cortes, Gonzalvo, et al. 2017) uses NN architecture to directly minimize

$$F(w) = \frac{1}{N}\sum_{i=1}^{N} \Phi(\sum_{j=1}^{J} w_j\, h_j(\boldsymbol{x_i})\,, y_i) + \sum_{j=1}^{J}(\lambda r(h_j) + \beta)\,|w_j|$$

  - $h_j$ is the base learner
  - $J$ is the number of iterations
  - $w_j$ is the mixture weight for each base learner
  - $\boldsymbol{x}_i = (x_{i,1}, \dots, x_{i,P})^T$
  - $N$ is number of sample size
  - $r(h_j)$ is complexity measurement
  - $\Phi$ is the loss function

- AdaNet is a lightweight TensorFlow-based framework for automatically learning high-quality models with minimal expert intervention.
  - AdaNet builds on recent AutoML efforts to be fast and flexible while providing learning guarantees.
  - AdaNet provides a general framework for not only learning a neural network architecture, but also for learning to ensemble to obtain even better models.

- AdaNet grows the ensemble of NNs adaptively.
  - At each iteration, it measures the ensemble loss for multiple candidates and selects the best one to move onto for the next iteration.
  - At subsequent iterations, the previous subnetworks are frozen, and only newly added subnetworks are trained.

- The main challenge is that it is difficult to interpret the results.

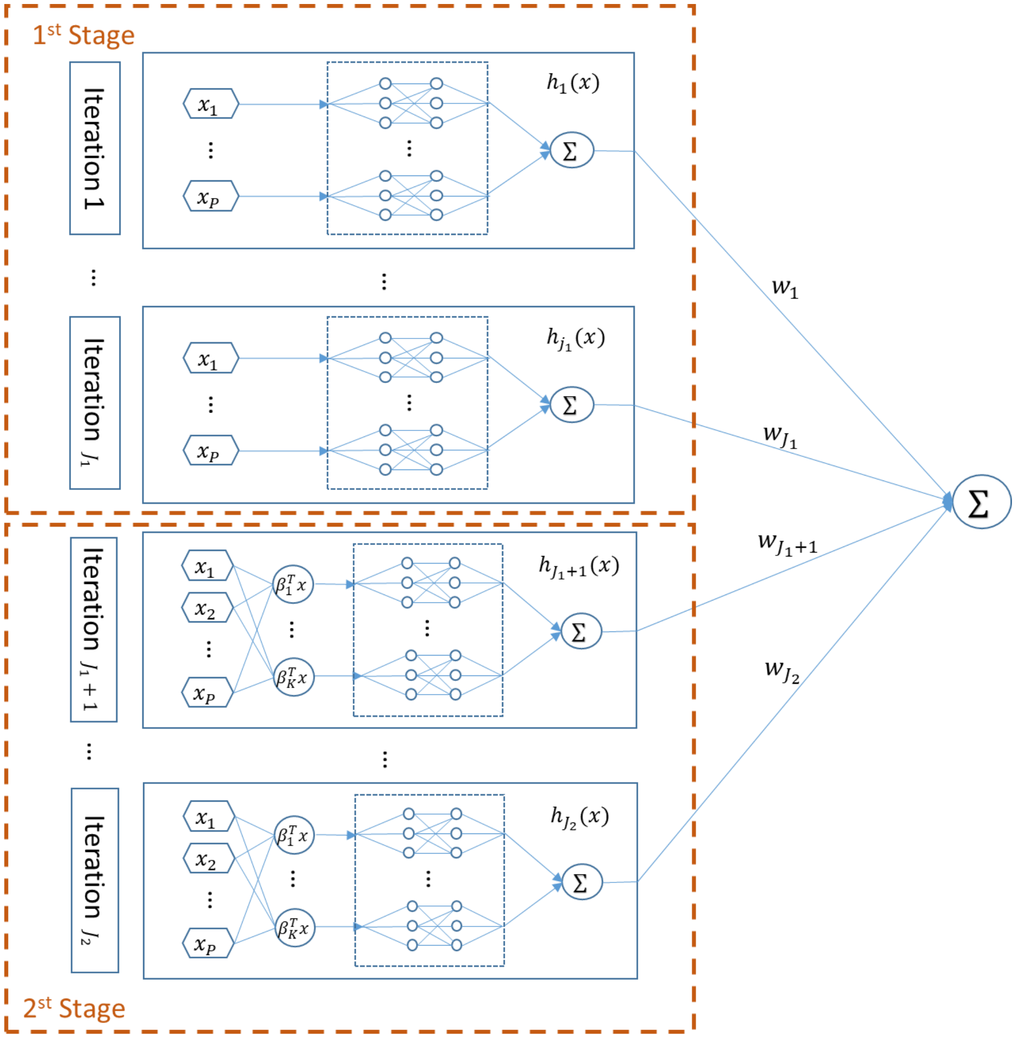
# Focus here: Adaptive explainable neural networks (AxNNs)

- Achieve the dual goals of good predictive performance and model interpretability
- For achieving good predictive performance:
  - We build a ensemble of a series of GAMnets and a series of xNNs using a two-stage process.
  - This can be done using either boosting or stacking ensemble.
- For interpretability:
  - The main and interaction effects from AxNN are obtained directly from decomposing the ridge functions of the AxNN algorithm
  - There is no extrapolation issue as in PDP
  - The main and (lower-order) interaction effects from AxNN can be easily visualized.
  - An importance measure for ranking the significance of all the detected main and interaction effects is provided.
- For computation and tuning:
  - Use Google's open source tool Adanet that can be efficiently accelerated by training with distributed computing.
  - It borrows strengths of AdaNet and does efficient NN architecture search and requires less tuning.

# Formulation of AxNN

- The notions of main effects and higher-order interactions have been extended from two-way ANOVA to more complex models with many different definitions
- The formulation and underlying architecture of AxNN is described below. Let
$$h\big(E(Y|x)\big) = f(\boldsymbol{x}) = f(x_1, \dots, x_P)$$
- Main effects
  - The effect associated with an individual predictor obtained by projecting the original model onto the space spanned by GAMs
  - AxNN first fits the main effects using GAMnet
- Interaction effects:
  - Any embedded main effects may distort the magnitude of the interaction effects.
    - For example, for $f(x) = \log(x_1 + x_2)$ where $x_1 \sim U(0,1), x_2 \sim U(0.6,1)$, the R square for the linear regression between $f$ and $x_i's$ is more than 0.98.
  - Use xNN to fit an AIM to capture the remaining structure in $I(\boldsymbol{x})$
$$I(\boldsymbol{x}) = f(\boldsymbol{x}) - [g_1(x_1) + \cdots + g_P\,(x_P)].$$
    where interactions can be captured xNN with nonlinear ridge functions

# AxNN flow chat




- In the first stage, an ensemble with base learners of GAM networks (GAMnet) are used to capture the main effects.
- In the second stage, an ensemble of explainable neural networks (xNN), that are incremental to the first stage, adaptively fit additive index models.
- The incremental part from the second stage can be interpreted as interaction effects, allowing for direct interpretation of the fitted model.

# AxNN boosting ensemble

---

**Algorithm 1: AxNN with boosting ensemble**

---

1) For the first stage

For $k = 1, \dots, J_1$

a. Train $h_k(\boldsymbol{x})$ by $\min_{h_k} \frac{1}{N} \sum_{i=1}^{N} \Phi\left(\sum_{j=1}^{k-1} w_j h_j(\boldsymbol{x_i}) + h_k(\boldsymbol{x_i}), y_i\right)$ with $\sum_{j=1}^{k-1} w_j h_j(\boldsymbol{x_i})$ fixed, where $h_k(\boldsymbol{x})$ is GAMnet.

b. Train $w_1, \dots w_k$ by $\min_{w_1,\dots,w_k} \frac{1}{N} \sum_{i=1}^{N} \Phi\left(\sum_{j=1}^{k-1} w_j h_j(\boldsymbol{x_i}) + w_k h_k(\boldsymbol{x_i}), y_i\right)$ with $h_1, \dots, h_k$ fixed.

2) For the second stage

Assume $L = \sum_{j=1}^{J_1} w_j h_j(\boldsymbol{x_i})$ are obtained from the first stage, and fix it.

For $k = J_1 + 1, \dots, J_2$

a. Train $h_k(\boldsymbol{x})$ by $\min_{h_k} \frac{1}{N} \sum_{i=1}^{N} \Phi\left(L + \sum_{j=J_1+1}^{k-1} w_j h_j(\boldsymbol{x_i}) + h_k(\boldsymbol{x_i}), y_i\right)$ with $\sum_{j=J_1+1}^{k-1} w_j h_j(\boldsymbol{x_i})$ fixed, where $h_k(\boldsymbol{x})$ is xNN.

b. Train $w_{J_1+1}, \dots w_k$ by $\min_{w_{J_1+1},\dots w_k} \frac{1}{N} \sum_{i=1}^{N} \Phi\left(L + \sum_{j=J_1+1}^{k-1} w_j h_j(\boldsymbol{x_i}) + w_k h_k(\boldsymbol{x_i}), y_i\right)$ with $h_{J_1+1}, \dots, h_k$ fixed.

---

*All the penalty terms are ignored in algorithm 1 and 2 for simplicity.

# AxNN stacking ensemble

- Stacking ensemble approach
  - It `stacks' with each base learner trained using the original response variable rather than "residuals" .
  - For each iteration, AdaNet selects the best subsets among multiple candidate subsets.
  - The candidate subnetworks can vary with different depths and over iterations, usually with increasing complexity manner, so the base learners are different for different iterations.
- The rationale here is model (weighted) averaging and stacking, <u>similar to random forest:</u>
  - The base learner from each iteration is unbiased but has high variance.
  - The variance is reduced through weighted averaging/stacking.
  - This method requires strong base learners: deeper or wider NN architecture.
- In contrast, the rationale behind boosting is similar to gradient boosting machine (GBM):
  - It starts with weak learners.
  - It boosts performance over the iterations by fitting to "residuals" and removing bias.
- Both boosting and stack approaches have similar performance.

# Ridge Function decomposition for interpretability

- We do ridge function decomposition by grouping the ridge functions with the same projection coefficient patterns:
  - First stage: ridge functions with the same covariate are grouped together to account for the main effect of the corresponding covariate.
  - Second stage: apply a coefficient threshold value to the projection layer coefficients of each ridge function:
    - Projection coefficients bigger than threshold value are considered active;
    - Ridge functions with the same set of active projection coefficients are aggregated.
    - Different sets of active projection coefficients account for different interaction patterns.
- The fitted response $\hat{f}$ can be decomposed into

$$\hat{f} = \sum_{j=1}^{J_1} w_j h_j(\boldsymbol{x_i}) + \sum_{j=J_1+1}^{J_2} w_j h_j(\boldsymbol{x}_i) = \sum_{p=1}^{P} M\left(x_{i,p}\right) + \sum_{q \in S} I_q(\boldsymbol{x}_i)$$

  - $M\left(x_{i,p}\right)$ is the main effect for $x_p$ and i-th observation;
  - $I_q(\boldsymbol{x}_i)$ is the interaction effect of the active projection coefficient set $q$.
- The importance of main effects and interaction effects can be measured by the standardized variance of each effect over the overall response variance.

# Simple synthetic example

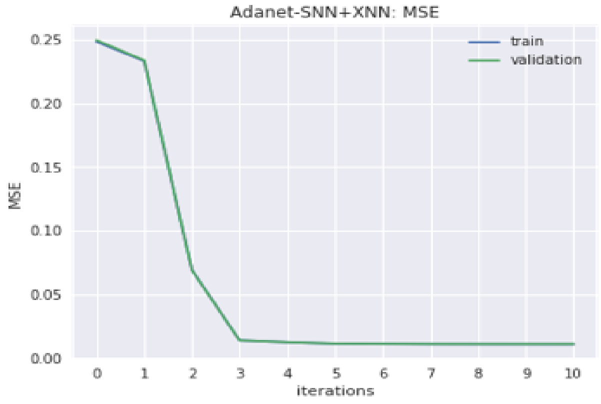
- A simple synthetic example

$$y = x_1 + x_2^2 + x_3^3 + e^{x_4} + x_1 x_2 + x_3 x_4 + \epsilon,$$

  – $x_i \sim U(-1,1), \epsilon \sim N(0, 0.1)$.

- Training/validation/testing data: 50K/25K /25K
  – Testing MSE: 0.0107
  – Testing R square(R2): 0.9913

- There is a steep decrease of training and validation errors after two GAMnet weak learners.




- We can automatically select the architecture from previous iterations and other candidate networks with the same number of layers but with one more unit

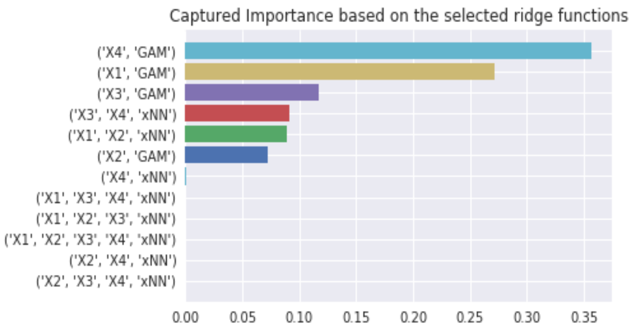
**Table 1: Neural network type and architectures over the iterations**

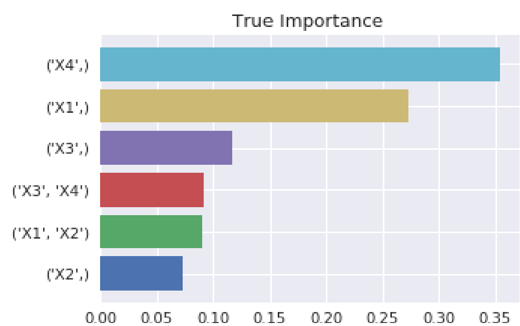
| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| stage | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| iteration | 1 | 2 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| weak learner type | GAMnet | GAMnet | xNN | xNN | xNN | xNN | xNN | xNN | xNN | xNN | xNN | xNN |
| # of layer | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| # of units | 5 | 6 | 6 | 7 | 8 | 8 | 8 | 9 | 9 | 9 | 10 | 11 |

# Simple synthetic example – Ridge function decomposition

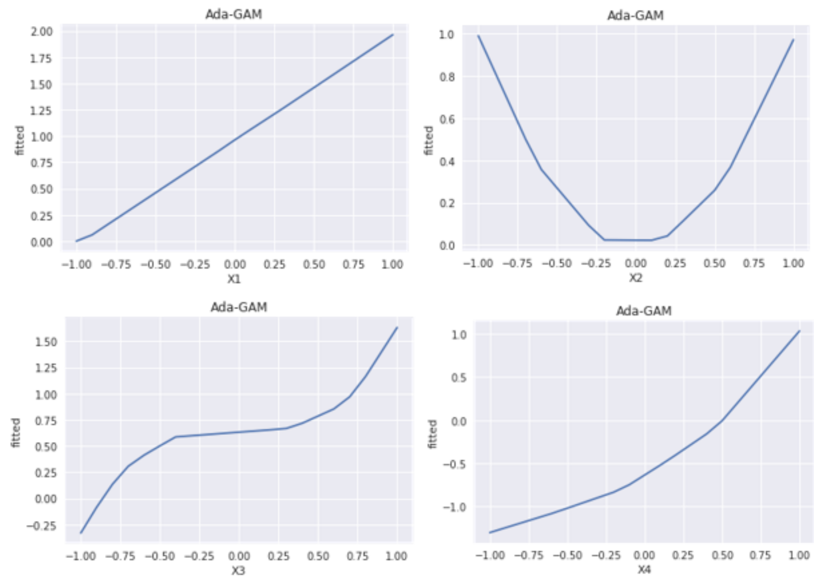
$$y = x_1 + x_2^2 + x_3^3 + e^{x_4} + x_1 x_2 + x_3 x_4 + \epsilon,$$

- Importance of main/interaction effects

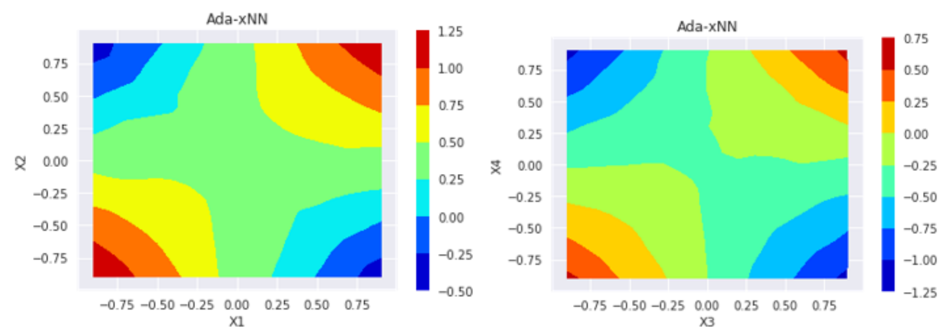



- Main effect from the first stage




- Interaction effect from the second stage




- Decomposition results are consistent with the true model form

# Complicated synthetic examples: Models

- **Example 1**

$$f(x) = \pi^{x_1 x_2}\sqrt{2x_3} - \sin^{-1} x_4 + \log(x_3 + x_5) - \frac{x_9}{x_{10}}\sqrt{\frac{x_7}{x_8}} - x_2 x_7,$$

  – where $x_1, x_2, x_3, x_6, x_7, x_9 \sim U(0,1), x_4, x_5, x_8, x_{10} \sim U(0.6,1)$.

- **Example 2:**

$$f(x) = x_1^2 + x_2^2 + x_3^2 + x_3 x_4 + 2x_4 x_5 x_6 + x_4^3 x_7 + x_5 x_6 x_7 + x_7 x_8 x_9 x_{10},$$

  – where $x_1, \ldots, x_{10} \sim U(-1,1)$

- **Example 3:**

$$f(x) = x_1 x_2 + 2^{x_3 + x_5 + x_6} + 2^{x_3 + x_4 + x_5 + x_7} + sin\big(x_7 sin(x_8 + x_9)\big) + arccos(0.9 x_{10}),$$

  – where $x_1, \ldots, x_{10} \sim U(-1,1)$

- **Example 4:**

$$f(x) = \frac{1}{1 + x_1^2 + x_2^2 + x_3^2} + \sqrt{\exp(x_4 + x_5)} + |x_6 + x_7| + x_8 x_9 x_{10},$$

  – where $x_1, \ldots, x_{10} \sim U(-1,1)$

# Complicated synthetic examples: Performance

- For both boosting and stacking, we considered only one layer for the ridge function subnetworks:
  - AxNN boosting starts with weak GAMnet and xNN networks: xNN with 2 subnets and each ridge subnetwork with 3 or 5 units.
  - The stacking AxNN starts with stronger GAMnet and xNN networks: xNN with 15 or 20 subnets and each ridge subnetwork with 10 units.
- AxNN stacking has the best performance over all the four examples; AxNN boosting and FFNN are close; RF has the worst performance.

**Table 2: test performance for the complicated synthetic examples (no error)**

| No | metric | ground truth | AxNN boosting | AxNN stacking | RF | XGB | FFNN |
|---|---|---|---|---|---|---|---|
| Example 1 | MSE | 0 | 0.0013 | **0.0004** | 0.0110 | 0.0018 | 0.0016 |
| | R2 score | 1 | 0.9984 | **0.9995** | 0.9866 | 0.9978 | 0.9980 |
| Example 2 | MSE | 0 | 0.0027 | **0.0012** | 0.1654 | 0.0138 | 0.0204 |
| | R2 score | 1 | 0.9956 | **0.9981** | 0.7351 | 0.9779 | 0.9673 |
| Example 3 | MSE | 0 | 0.0027 | **0.0007** | 0.1880 | 0.0150 | 0.0085 |
| | R2 score | 1 | 0.9993 | **0.9998** | 0.9539 | 0.9963 | 0.9979 |
| Example 4 | MSE | 0 | 0.0010 | **0.0008** | 0.0515 | 0.0033 | 0.0019 |
| | R2 score | 1 | 0.9980 | **0.9983** | 0.8926 | 0.9931 | 0.9960 |

# Complicated synthetic examples—Ridge function decomposition

- For all four examples, almost all the main effects from the first stage correctly capture the true projected main effects (correlation close to 1).
- The second stage also detects and captures significant high order interactions correctly (high correlations with the true pure interaction terms). Estimation of insignificant interactions are less accurate and unstable
- When the interactions have a big overlap, the union of the interactions (with higher order) can be detected instead.

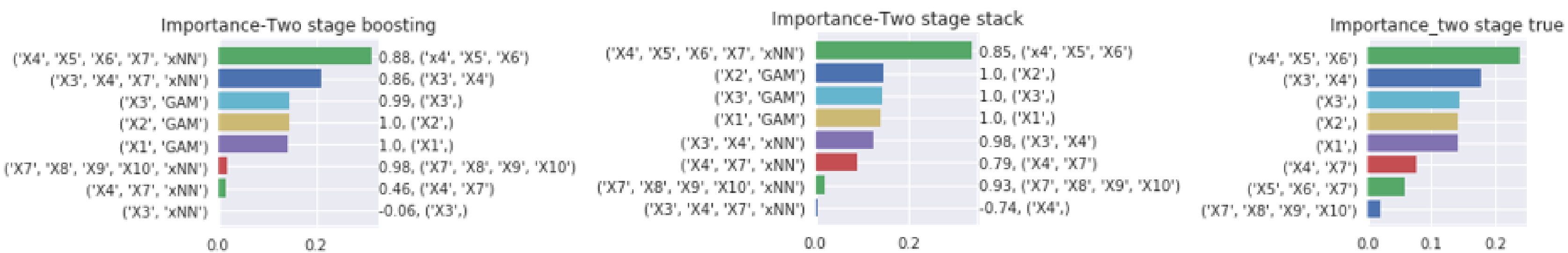



**Figure 9: ridge function decomposition for Synthetic Example 2**

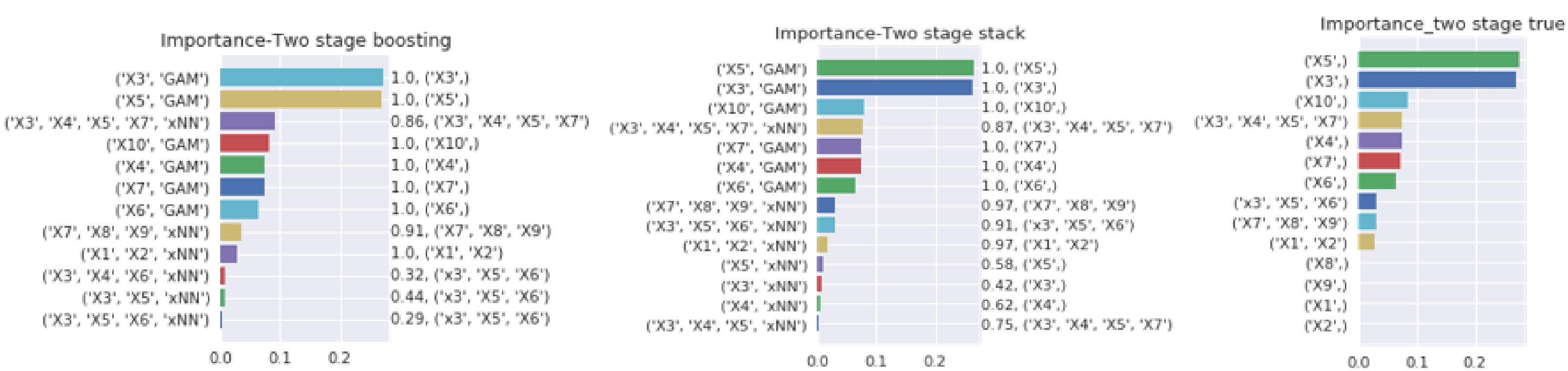



**Figure 10: ridge function decomposition for Synthetic Example 3**

# Conclusions

- AxNN is a new machine learning framework that achieves the dual goals of predictive performance and model interpretability.
- We have introduced and studied the properties of two-stage approaches, with GAMnet base learners to capture the main effects and xNN base learners to capture the interactions.
- The stacking and boosting algorithms have comparable performances. Both decompose the fitted responses into main effects and higher-order interaction effects through ridge function decomposition.
- AxNN borrows strength of AdaNet and does efficient NN architecture search and requires less tuning.
- Paper link: https://arxiv.org/abs/2004.02353