WELLS FARGO

Machine learning model validation

Risk Americas Workshop
New York, NY

Agus Sudjianto and Vijay Nair
Corporate Model Risk, Wells Fargo
May 9, 2022

Agenda

- **9:00 – 9:30: Introduction** – Agus Sudjianto
- **9:30-10:45: Machine Learning and Explainability**
  – Vijay Nair and Sri Krishnamurthy
- 10:45-11:00: **Break**
- **10:45-11:45: Unwrapping ReLU Networks**
  – Agus Sudjianto
- **11:45-12:45 Inherently Interpretable Models**
  – Vijay Nair and Sri Krishnamurthy
- **12:45-1:15: Lunch Break**

- **1:15-2:15: Outcome Testing**
  – Agus Sudjianto
- **2:15-3:15 Hands-on Exercises**
  – Sri Krishnamurthy
- **3:15-3:30: Break**
- **3:30-4:30 Bias and Fairness**
  – Nick Schimdt
- **4:30-5:00: ModelOp Presentation**
  – Jim Olsen

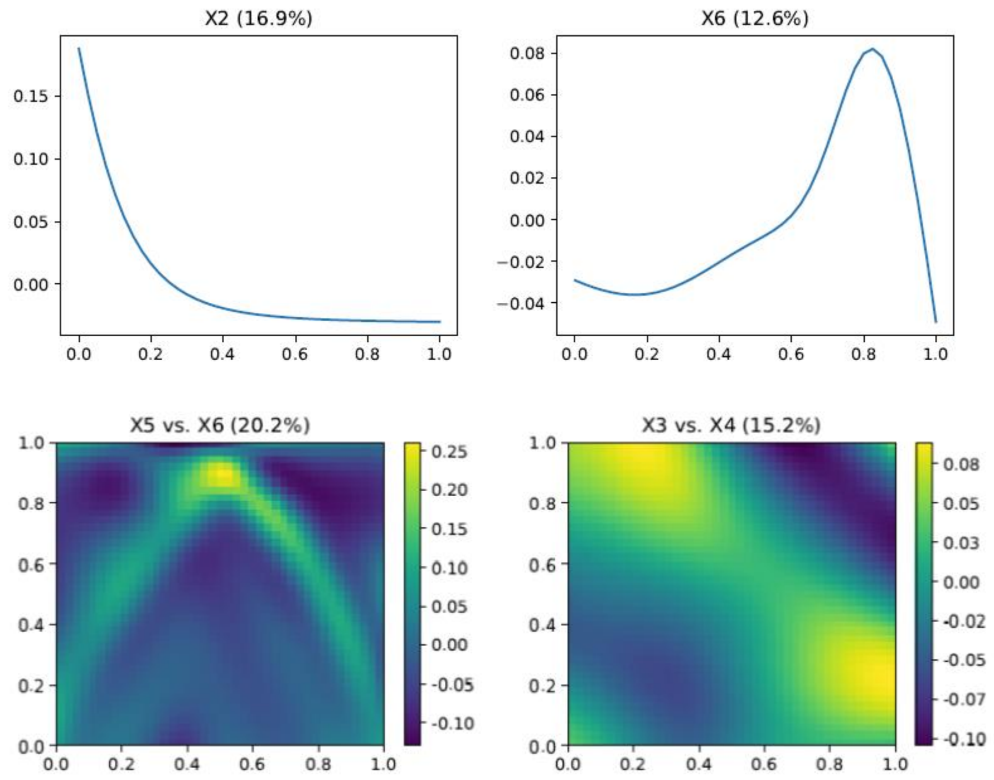
Overview

1. Introduction: Risk Dynamics, Conceptual Soundness and Outcome Testing
2. Supervised Machine Learning: Algorithms and Explainability
3. Deep ReLU Networks and Inherent Interpretation
4. **Inherently Interpretable Models**
5. Outcome Testing

Outline

- **Inherently interpretable models**
  - **Overview**
  - **FANOVA framework**
- Functional ANOVA Models
  - Explainable boosting machine
  - GAMI neural networks
- Comparisons
- PiML Demo

$$f(\boldsymbol{x}) = g_0 + \sum g_j(x_j) + \sum g_{jk}(x_j, x_k)$$



Inherently interpretable models

- Key characteristics of inherently interpretable models
  - **Parsimonious** models are easier to interpret, more robust and more likely to generalize
    - ✓ **Sparsity** → not too many active effects or complicated relationships
    - ✓ **Low-order main effects,** preferably linear
    - ✓ **Low-order interactions** → more than two hard to understands
  - **Analytic expression** for model form → use **regression coefficients** for interpretation

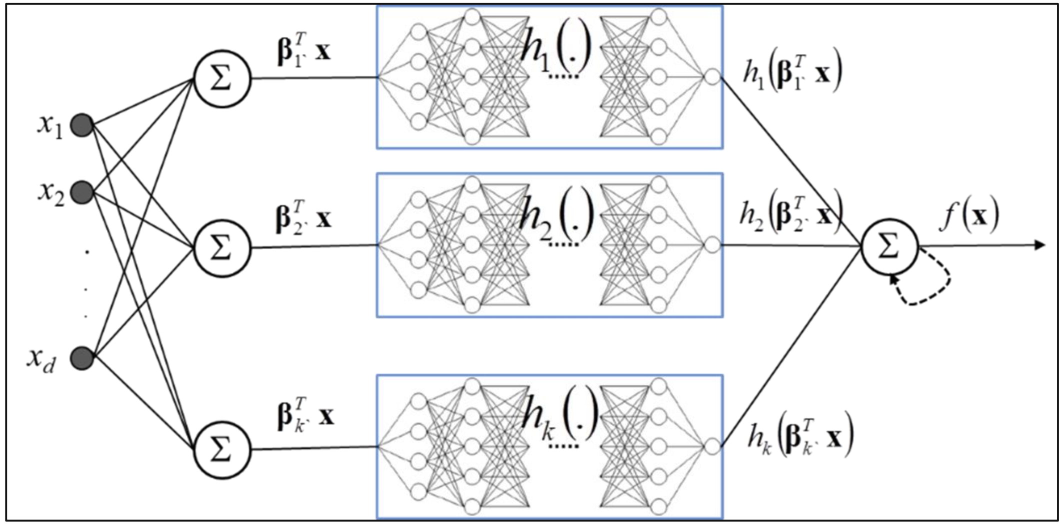
- Goals and challenges of complex models
  - **Goal** → extract as much **predictive performance** as possible
  - BUT … no analytic expressions → **rely on low dimensional summaries**
    - → don’t present the full picture
    - → can’t visualize it in low dimensions
  - **No parsimony** → lots of variables, complex relationships and interactions
  - **High correlations** which create lot of problems

- Emerging view:
  - **Low-order nonparametric models are adequate** in most of our applications → typical in **banking**
  - **Directly interpretable**
  - Reversing emphasis on complex modeling → **balance** potential small improvements for interpretation

Examples of "Low Order" Models

- **Additive Index Models:**

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



  – Generalization of Generalized Additive Models (GAMs): $f(\boldsymbol{x}) = g_1(x_1) + g_2(x_2) + \ldots + g_P(x_P)$
  – Incorporates certain types of interactions
  – **Projection pursuit regression** (Friedman and Stuetzle, 1981)
  – **Need for scalable algorithms** with large datasets and many predictors

  – Use specialized **neural network architecture and associated fast algorithms**
    – e**X**plainable **N**eural **N**etworks (xNNs) → Vaughan, Sudjianto, … Nair (2020)

Examples of “Low Order” Models

- **Functional ANOVA Models:**

$$f(\boldsymbol{x}) = g_0 + \sum_j g_j(x_j) + \sum_{j<k} g_{jk}(x_j, x_k) + \sum_{j<k<l} g_{jkl}(x_j, x_k, x_l) + \cdots$$

  – FANOVA models with low-order interactions are adequate for many of our applications
  – **Focus** on models with **functional main effects and second order interactions**
  – Stone (1994); Wahba and her students (see Gu, 2013)
    → use **splines** to estimate low-order functional effects non-parametrically
  – **Not scalable** to large numbers of observations and predictors
  – Recent approaches
    → use **ML architecture and optimization algorithms** to develop fast algorithms

FANOVA framework

$$f(\boldsymbol{x}) = g_0 + \sum_j g_j(x_j) + \sum_{j<k} g_{jk}(x_j, x_k)$$

- Model made up of mean g_0, **main effects $g_j(x_j)$, two-factor interactions $g_{jk}(x_j, x_k)$**
- **Interpretability**
  - Fitted model is **additive,** effects are enforced to be **orthogonal**
  - Components can be **easily visualized** and **interpreted directly**
  - Regularization or other techniques used to keep model parsimonious

- Two state-of-the-art ML algorithms for fitting these models:
  - **Explainable Boosting Machine** (Nori, et al. 2019) → boosted tress
  - **GAMI Neural Networks** (Yang, Zhang and Sudjianto, 2021) → specialized NNs

- GAMI-Tree and other inherently interpretable models under development

Nori, Jenkins, Koch and Caruana (2019). InterpretML: A Unified Framework for Machine Learning Interpretability. arXiv: 1909.09223
Yang, Zhang and Sudjianto (2021, Pattern Recognition): GAMI-Net. arXiv: 2003.07132

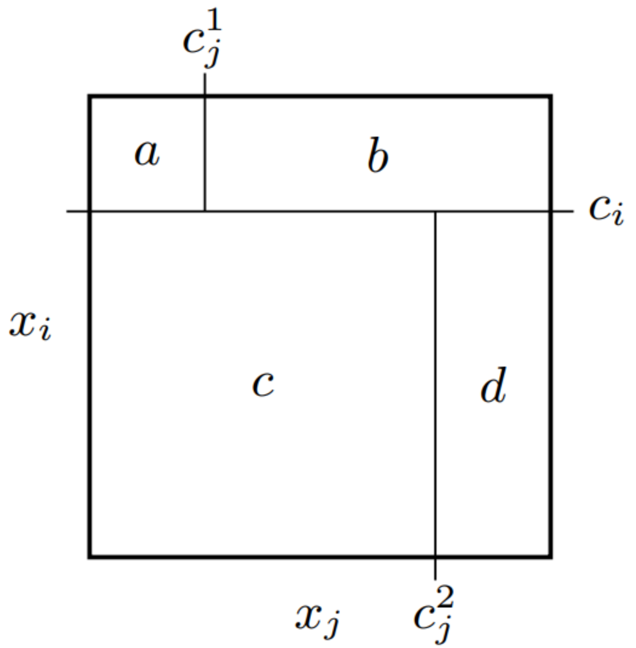
Explainable Boosting Machine

- **EBM** – Boosted-tree algorithm by Microsoft group (Lou, et al. 2013)

$$f(\boldsymbol{x}) = g_0 + \sum g_j(x_j) + \sum g_{jk}(x_j, x_k)$$

  - Microsoft InterpretML (Nori, et al. 2019)
  - fast implementation in C++ and Python

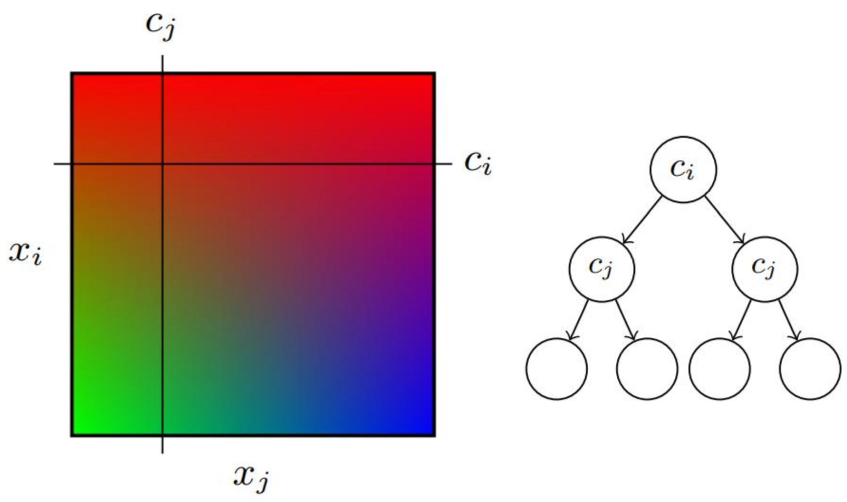
- **Multi-stage model training :**
  - 1: fit functional main effects non-parametrically
    - **Shallow tree boosting** with splits on the same variable for capturing a non-linear main effect
  - 2: fit pairwise interactions on residuals:
    a. Detect interactions using **FAST** algorithm (next page)
    b. For each interaction (x_j, x_k), fit function $g_{jk}(x_j, x_k)$ non-parametrically using a tree with depth two: 1 cut in x_j and 2 cuts in x_k, or 2 cuts in x_j and 1 cut in x_k (pick the better one)
    c. Iteratively fit all the detected interactions until convergence

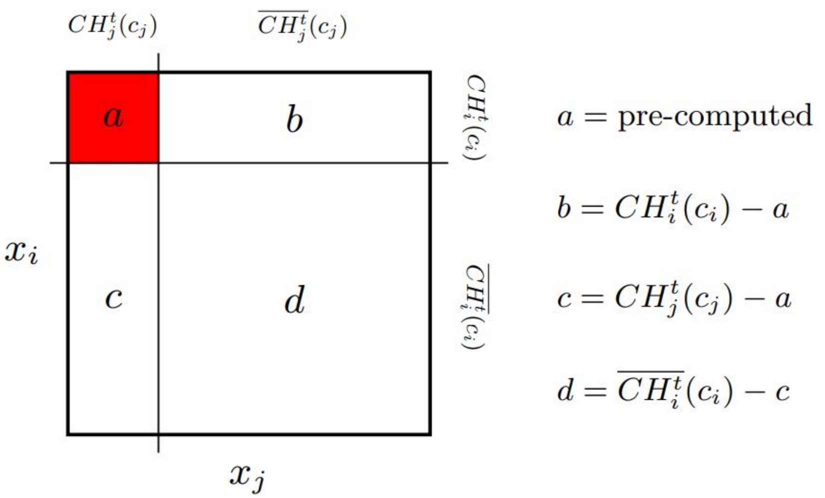


Lou, Caruana, Gehrke and Hooker (2013). Accurate Intelligible Models with Pairwise Interactions. Microsoft Research

Explainable boosting machine

- **FAST** algorithm for pairwise interaction detection
  - Obtain the residuals of fitted main effects;
  - Score each interaction (x_j, x_k) by simple depth-2 trees:
    - Place 1 cut on each of x_j and x_k, respectively;
    - Fast computation of target values for each quadrant;
    - Speedup by using bookkeeping data structures;
  - Select the top-K pairwise interactions.
- FAST algorithm has a **C++ implementation** in InterpretML.
- **Bagging** option for enhanced performance
  - Inner bag: fit individual effects on bagged sample;
  - Outer bag: fit individual EBMs on different subsamples of dataset.





Explainable boosting machine: Example

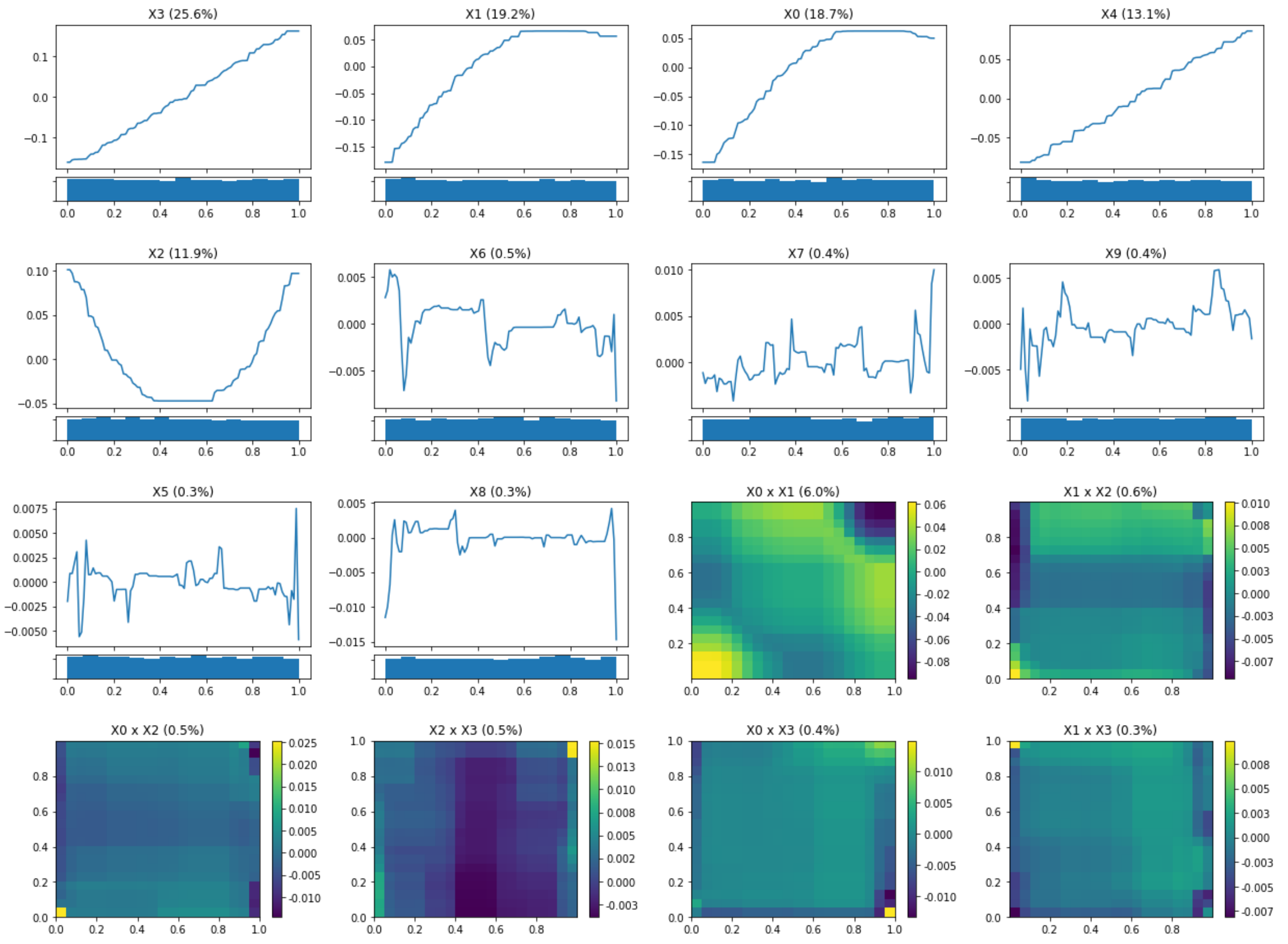
Friedman1 simulated data:

- sklearn.datasets.make_friedman1 n_samples=10000, n_features=10, and noise=0.1.
- Multivariate independent features $\boldsymbol{x}$ uniformly distributed on [0,1]
- Continuous response generated by

$$y(\boldsymbol{x}) = 10\sin(\pi x_0 x_1) + 20(x_2 - 0.5)^2$$
$$+20x_3 + 10x_4 + \epsilon$$

  depending only $\boldsymbol{x_0} \sim \boldsymbol{x_4}$

EBM Output with Test RMSE = 0.0284 and R2 = 97.39%



GAMI-Net

- NN-based algorithm for non-parametrically fitting

$$f(\boldsymbol{x}) = g_0 + \sum g_j(x_j) + \sum g_{jk}(x_j, x_k)$$

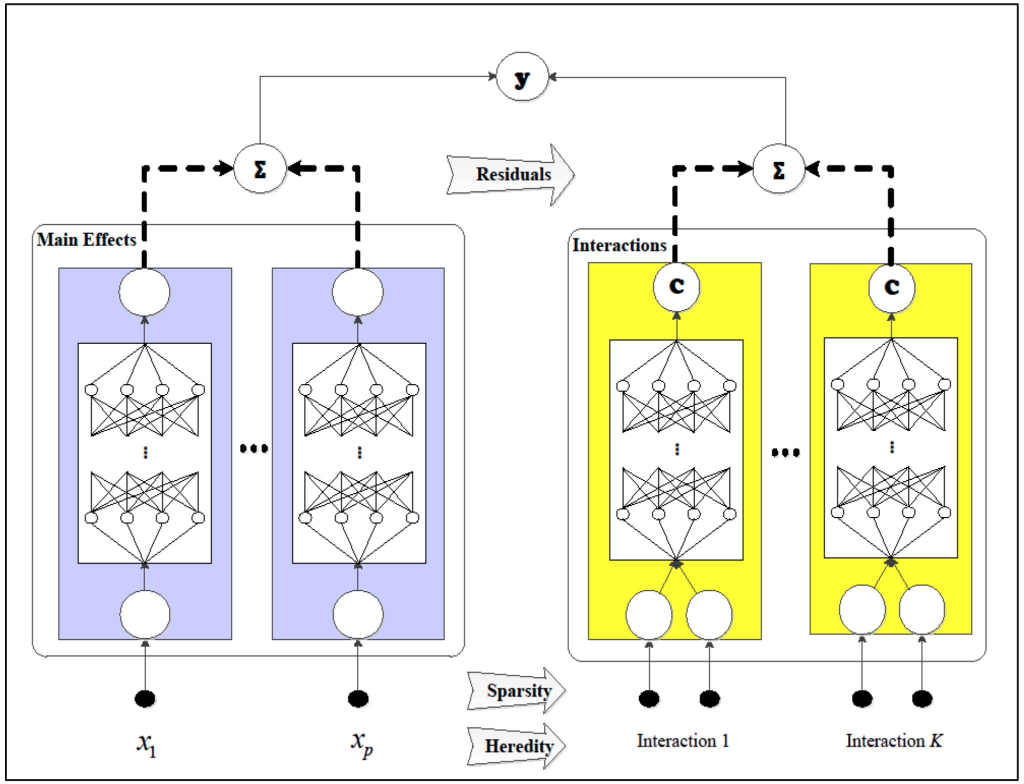
- **Multi-stage training algorithm:**

  1: estimate $\{g_j(x_j)\}$ → train main-effect subnets and **prune** small main effects

  2: estimate $\{g_{jk}(x_j, x_k)\}$ → compute residuals from main effects and train pairwise interaction nets

  - Select candidate interactions using heredity constraint
  - Evaluate their scores (by FAST) and select top-K interactions;
  - Train the selected two-way interaction subnets;
  - Prune small interactions

  3: retrain main effects and interactions simultaneously



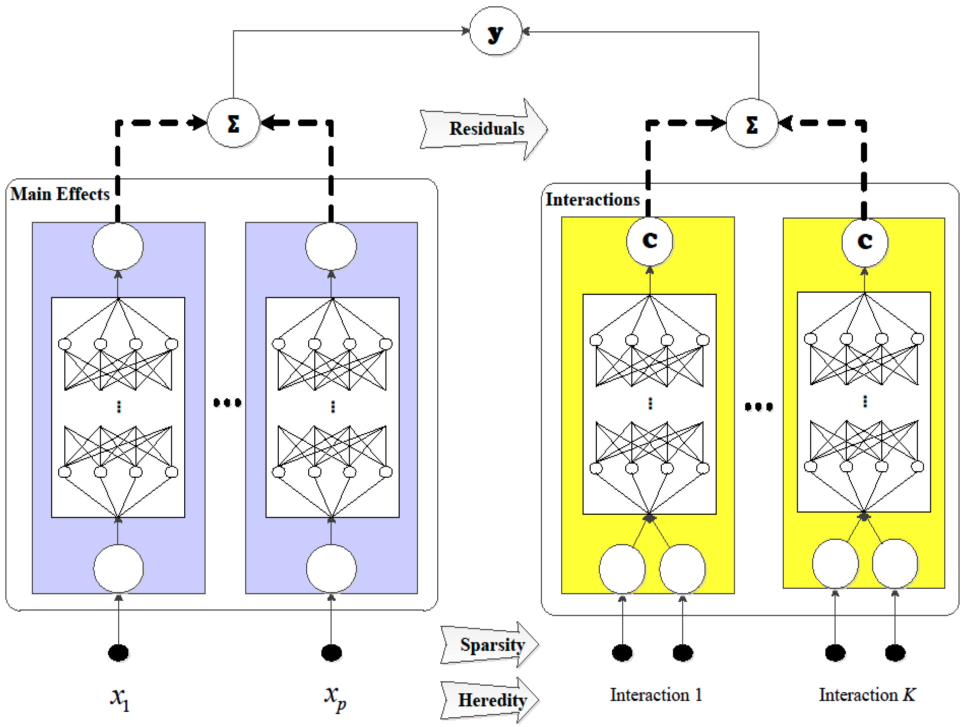
Yang, Zhang and Sudjianto (2021, Pattern Recognition): GAMI-Net. arXiv: 2003.07132

GAMI-Net and interpretability constraints

Incorporates the following constraints :

- **Sparsity**: select only the most important main effects and pairwise interactions
- **Heredity**: a pairwise interaction is selected only if at least one of its parent main effects has been selected
- **Marginal Clarity**: enforce pairwise interactions to be nearly orthogonal to the main effects, by imposing penalty
- **Potential monotonicity**: certain features can be constrained to be monotonic increasing or decreasing

$$f(\boldsymbol{x}) = g_0 + \sum g_j(x_j) + \sum g_{jk}(x_j, x_k)$$



Diagnostics: Effect importance and feature importance

- Each **effect importance** (before normalization) is given by

$$D(h_j) = \frac{1}{n-1}\sum_{i=1}^{n} g_j^2(x_{ij}), \qquad D(f_{jk}) = \frac{1}{n-1}\sum_{i=1}^{n} g_{jk}^2(x_{ij}, x_{ik})$$

- For prediction at $\boldsymbol{x}_i$, the **local feature importance** is given by

$$\phi_j(x_{ij}) = g_j(x_{ij}) + \frac{1}{2}\sum_{j \neq k} g_{jk}(x_{ij}, x_{ik})$$

- For GAMI-Net (or EBM), the **global feature importance** is given by

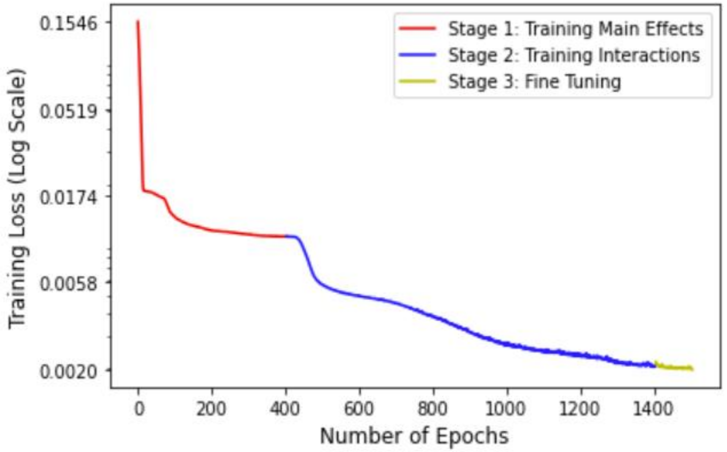
$$\mathrm{FI}(x_j) = \frac{1}{n-1}\sum_{i=1}^{n}\left(\phi_j(x_{ij}) - \overline{\phi_j}\,\right)^2$$

- The effects can be visualized by a line plot (for main effect) or heatmap (for pairwise interaction).

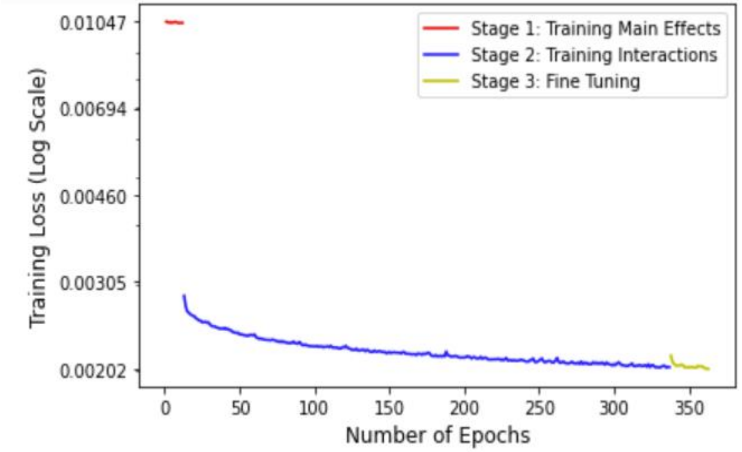
GAMI-Net implementation

- GAMI-Net is implemented in both TensorFlow[9] and PyTorch[10]
- Some implementation details:
  - **Warm initialization for subnetworks:**
    1. Fit a fast GAM with B-splines using subsampled training data;
    2. Generate 1D or 2D random sample X and compute $Y = \hat{s}(X)$ by the fitted B-spline component;
    3. Initialize the main effect or interaction subnetwork by training it to the generated data (X, Y).
  - **Other tricks** …

Without warm initialization



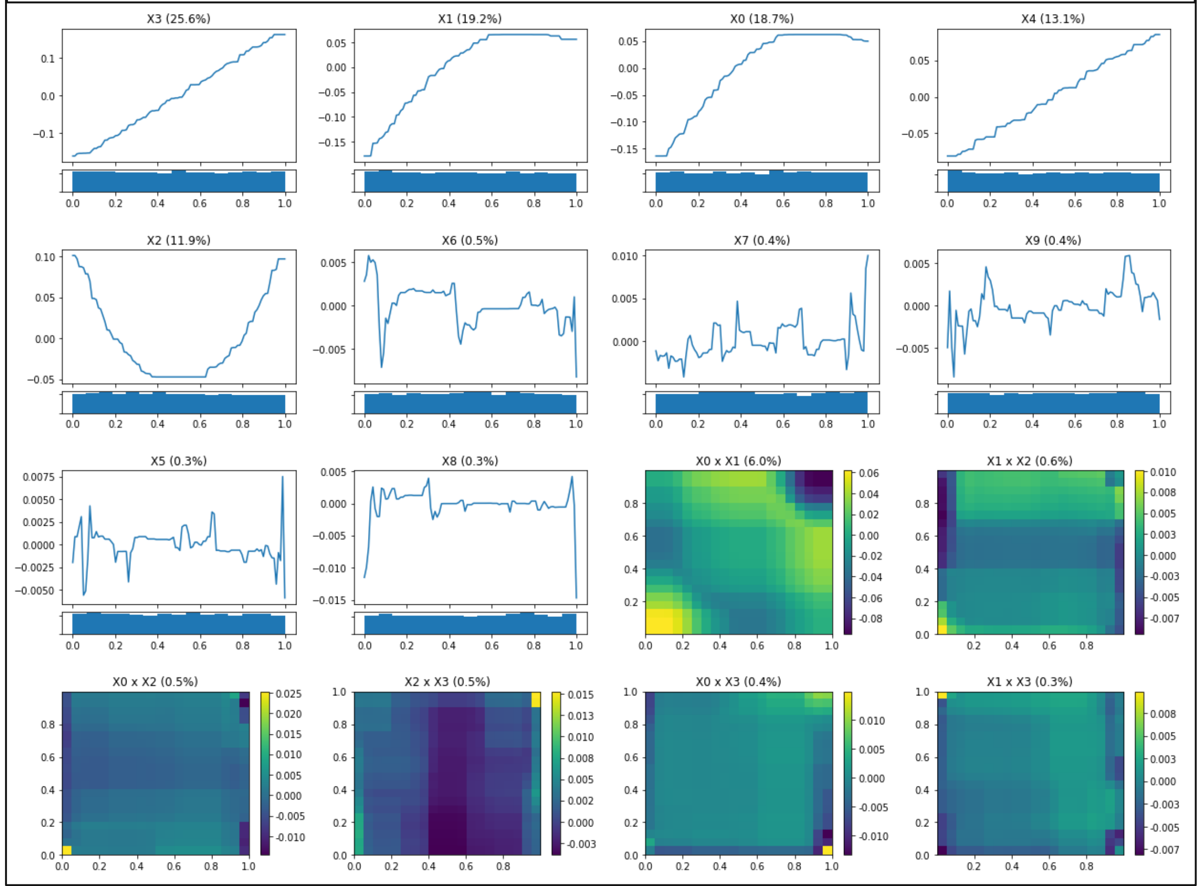
With warm initialization (~3 times faster)



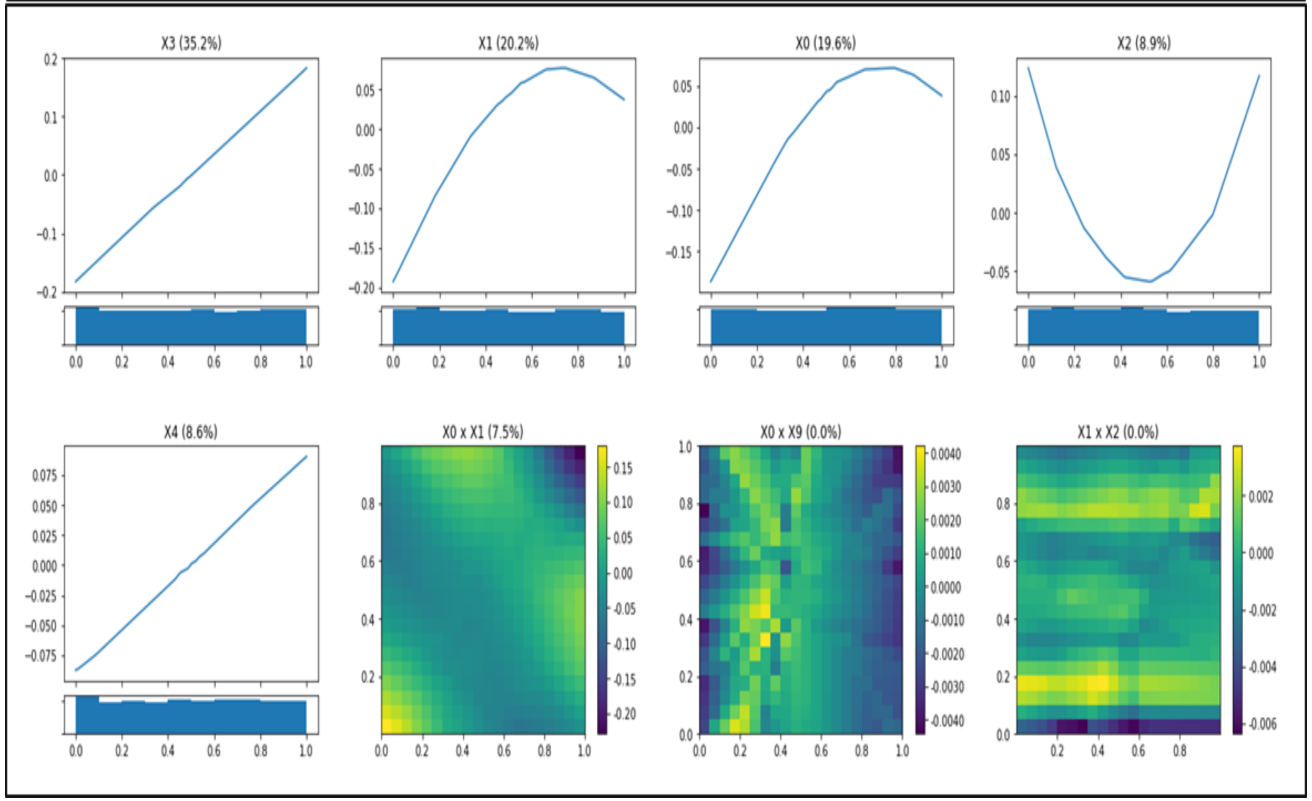
[9] GAMI-Net in TensorFlow 2.0: https://github.com/ZebinYang/gaminet
[10] GAMI-Net in PyTorch 1.10: https://github.com/ZebinYang/GAMINet-PyTorch

Comparison of Results on Friedman1 Simulated Data

EBM Output with Test RMSE = 0.0284 and R2 = 97.39%



GAMI-Net Output with Test RMSE = 0.0058 and R2 = 99.89%

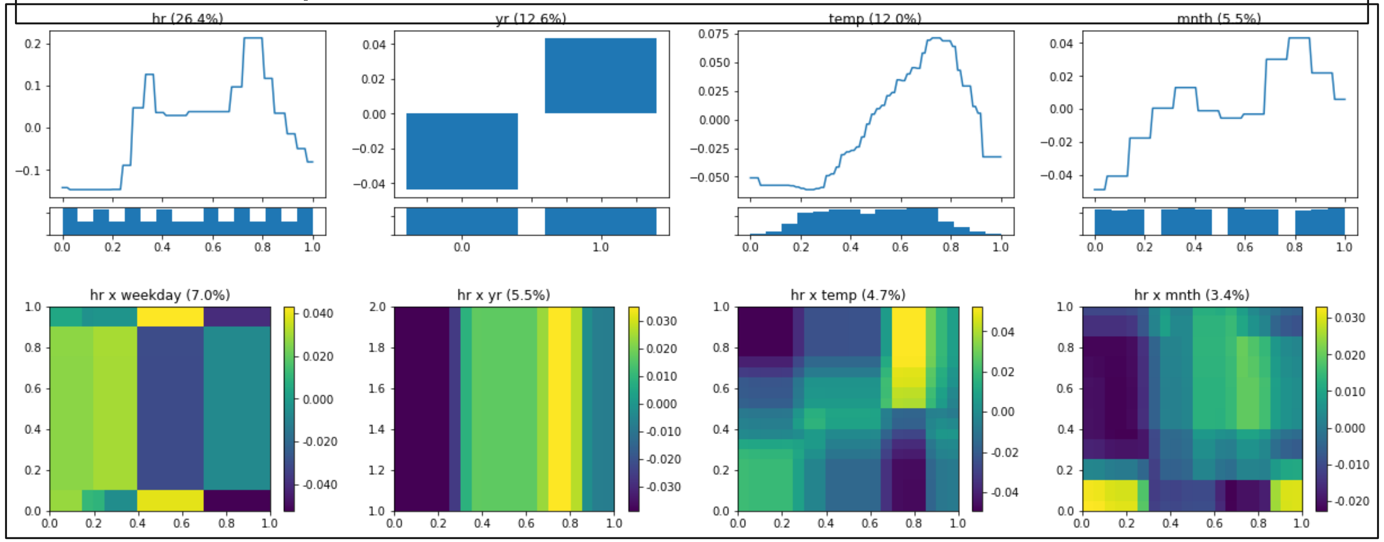


Comparisons

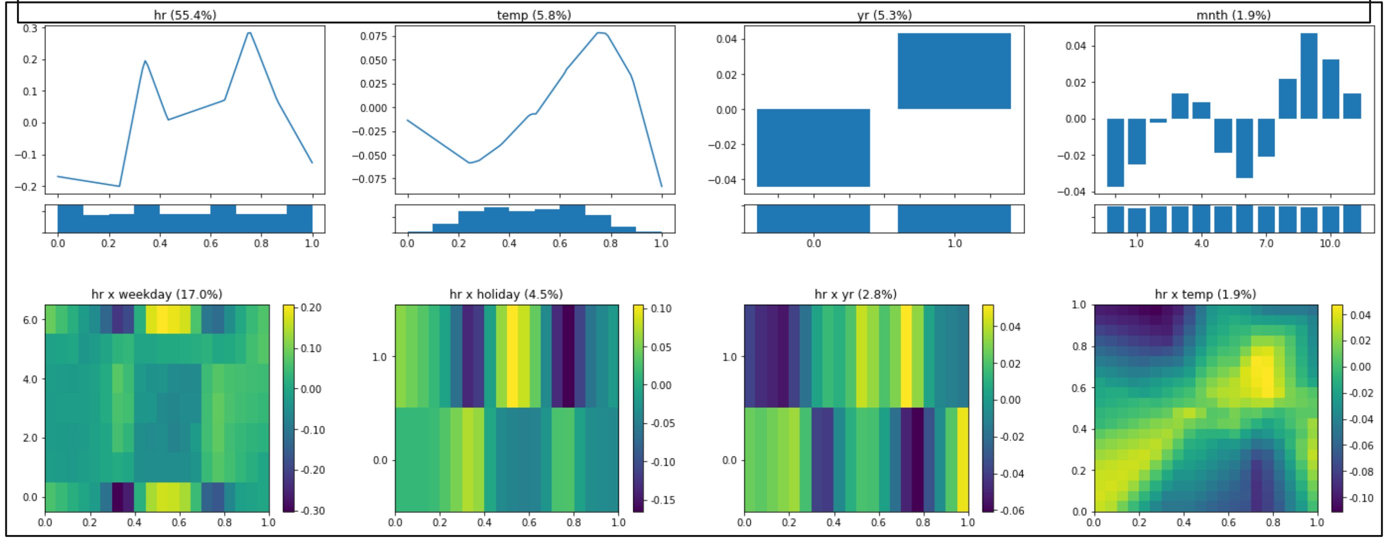
Bike sharing data:

- Hourly count of rental bikes between years 2011 and 2012 in Capital bikeshare system
- Sample size: 17,379
- Features include weather conditions, precipitation, day of week, season, hour of the day, etc.
- Response is count of total rental bikes

EBM Output with test RMSE = 0.0825 and R2 = 80.58%



GAMI-Net Output with test RMSE = 0.0595 and R2 = 89.89%



Discussion: EBM vs. GAMI-Net

- Both ML algorithms to fit low-order FANOVA models $f(\boldsymbol{x}) = g_0 + \sum g_j(x_j) + \sum g_{jk}(x_j, x_k)$
- Exploit ML architectures → fast and scalable → can handle large datasets with many predictors
- Inherently interpretable models: additive, orthogonal effects, can be directly visualized
- Direct techniques for capturing feature and effects importance → no need to rely on post hoc tools

- **EBM:**
  - Tree-based →piecewise constant models
  - Good for fitting non-smooth response surfaces
  - Introduced FAST algorithm for identifying important interactions

- **GAMI-Net:**
  - NN based → piecewise linear models
  - Good for fitting smooth models
  - Has sparsity constraints
  - Can incorporate monotonicity

PiML Demo