Deep Learning for Functional Data Analysis with Adaptive Basis Layers

Junwen Yao [1] Jonas Mueller [2] Jane-Ling Wang [1]

[1]University of California, Davis

[2]Amazon

Functional Data Analysis (FDA)

- Functional data are random functions defined on an interval or any k-dimensional domain.
- Example 1. Continuous stochastic processes, such as Gaussian processes on $[0, 1]$.
- Example 2. Household electricity consumption over a period.
- Functional data analysis (FDA) deals with the analysis of functional data.

Functional Data Analysis (FDA)

- Functional data are intrinsically infinite dimensional and generated by smooth underlying processes.
- The smoothness property is beneficial: the observed measurements at one location t_0 can inform us of $X(t)$ for t at nearby locations.
- Functional data are replicated trajectories, whereas time series data are usually repeated measurements of one subject.

Functional Data Analysis (FDA)

- Formally, let $X(t)$ denote a random function on $[0, 1]$.
- Assume $\mathcal{T} : X(t) \to Y$.
- Objective: Use $X(t)$ to infer/predict some response Y.
- **Goal**: estimate $\mathcal{T}$ from the data using neural networks.

  Data: i.i.d. copies of $(X(t), Y) = \{(X_i(t), Y_i)\}_{i=1}^{n}$.

Functional Neural Network (FNN)

- In reality, $X_i(t)$ are observed at discrete times $\{t_1, \cdots, t_{J+1}\}$.

  The observed data are

$$\left\{ \underbrace{[X_i(t_1), \cdots, X_i(t_{J+1})]}_{\text{high-dimensional data}} \right\}_{i=1}^{n}$$

Existing Methods

- Discretization: estimate an approximate relationship

$$\mathcal{T}_{\text{finite}} : [X_i(t_1), \ldots, X_i(t_{J+1})] \rightarrow Y_i.$$

  Use the vector of discrete observations as a network input.

- Basis representation/ dimension reduction:

$$X(t) \approx \sum_{k=1}^{K} a_k \phi_k(t)$$

  for a set of K continuous basis functions $\{\phi_k(t)\}_{k=1}^{K}$.

  Use $[a_1, \ldots, a_K]$ as a network input.

Drawbacks

- Functional data are typically high-dimensional.
- Discretization doesn't respect the continuity of functional covariates.
- The choice of the bases is often done manually without incorporating the information contained in Y.

Our Proposal: AdaFNN

Add a basis layer, which consists of a number of Basis Nodes, that computes a score c_i of $X(t)$ w.r.t. the basis $\beta_i(t)$,

$$c_i = \langle \beta_i, X \rangle = \int \beta_i(t) \cdot X(t)\, dt.$$

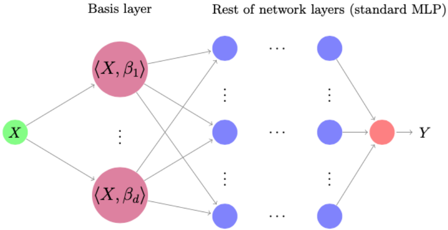


Figure: An overview of AdaFNN

Our Proposal: AdaFNN

Each basis function $\beta_i(t)$ can be approximated by a network [1] $\mathrm{nn}_{\Theta_i}(t)$ with weights Θ_i.

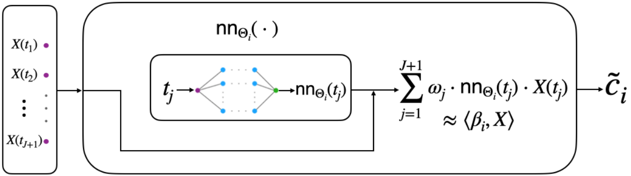


Figure: A basis node

[1] A similar idea was briefly mentioned in Rossi and Conan-Guez (2005) without actual implementation. It can also be approximated using a basis representation.

Our Proposal: AdaFNN

- Unlike previous two-step models (basis expansion), our model can be trained end-to-end.
- The dimension reduction step and the subsequent fitting step are synchronized in AdaFNN.

$\Rightarrow$ Learned basis functions are likely better suited for the desired task.

- The learned bases are continuous by construction.

Theoretical Results

Let $\mathcal{C}([0,1])$ denote the space of continuous functions defined on the compact interval $[0,1]$. Assume that the underlying mapping $\mathcal{T} : X \mapsto Y$ is a composite of a finite-dimensional linear transformation and a subsequent non-linear transformation.

That is, $\mathcal{T} = h \circ g$, where $g : \mathcal{C}([0,1]) \to \mathbb{R}^q$ is a linear continuous map, and $h : \mathbb{R}^q \to \mathbb{R}$ is a non-linear continuous map.

Theorem 1

There exists an AdaFNN that can achieve arbitrarily small error.

Simulation

Model

$$X(t) = \sum_{k=1}^{50} c_k \phi_k(t), \quad t \in [0,1],$$

where terms on the right hand are defined as:

1. $\phi_1(t) = 1$ and $\phi_k(t) = \sqrt{2}\cos((k-1)\pi t),\ k = 2, \ldots, 50;$
2. $c_k = z_k r_k$, and r_k are i.i.d. uniform random variables on $[-\sqrt{3}, \sqrt{3}]$.

Simulation

Case 1: $z_1 = 20, z_2 = z_3 = 5$, and $z_k = 1$ for $k \geq 4$.

The response $y = (\langle \phi_3, X \rangle)^2$.

Use AdaFNN$(0, 0)$ with 2 bases:

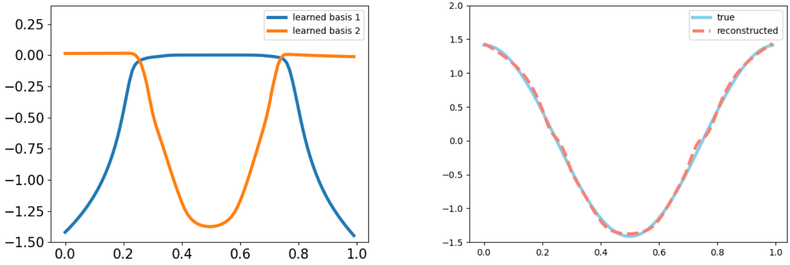


Figure: $\phi_3 \approx \hat{\phi}_3 = \hat{\beta}_2 - \hat{\beta}_1$

Simulation

Case 4: $z_k = 1$ for all k. The response is $y = \langle \beta_2, X \rangle + (\langle \beta_1, X \rangle)^2$, where

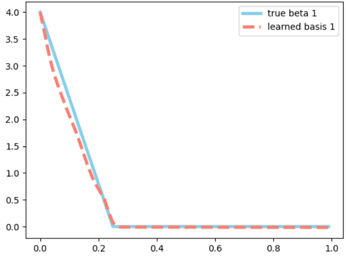
$$\beta_1(t) = (4 - 16t) \cdot 1\{0 \leq t \leq 1/4\}$$

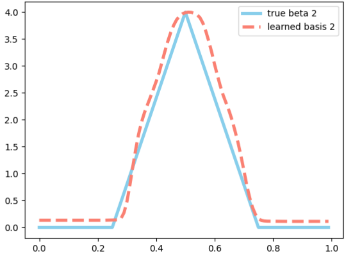
and

$$\beta_2(t) = (4 - 16|1/2 - t|) \cdot 1\{1/4 \leq t \leq 3/4\}.$$

Centered Gaussian noise is added to Y, and $X(t)$ is also contaminated by measurement error.

Use AdaFNN$(0, 0)$ with 2 bases:





Real Data Experiments

In 9 regression/classification tasks over four different datasets [2], AdaFNN empirically outperforms all baseline models.

METHOD	TASK 1	TASK 2	TASK 3	TASK 4	TASK 5	TASK 6	TASK 7	TASK 8	TASK 9
RAW DATA (48) + NN	0.099	0.284	0.124	0.296	0.380	0.488	0.472	0.406	0.373
B-SPLINE (15) + NN	0.094	0.306	0.137	0.326	0.335	**0.477**	0.429	0.413	0.387
$\mathrm{FPCA}_{0.99}$ + NN	0.119	0.339	0.143	0.306	0.363	0.493	0.431	0.429	0.378
ADAFNN (0.0, 0.0)	**0.084***	0.290*	0.129*	0.311	0.365	**0.477**	**0.410***	0.377*	0.375
ADAFNN (0.0, 1.0)	0.094	0.276	0.126	0.327	0.561	0.479*	0.498	0.374	0.392
ADAFNN (0.0, 2.0)	0.097	0.276	0.129	0.324	0.596	0.481	0.473	0.381	0.445
ADAFNN (0.5, 0.0)	0.108	**0.260**	0.130	0.310*	0.380*	0.490	**0.410**	0.376	**0.368***
ADAFNN (0.5, 1.0)	0.089	0.279	0.126	0.324	0.616	0.486	0.494	**0.362**	0.413
ADAFNN (0.5, 2.0)	0.098	0.280	0.128	0.345	0.392	0.509	0.444	0.373	0.450
ADAFNN (1.0, 0.0)	**0.084**	0.288	**0.118**	**0.294**	**0.339**	0.485	0.413	0.378	0.406
ADAFNN (1.0, 1.0)	0.097	0.282	0.133	0.320	0.651	0.502	0.456	0.371	0.394
ADAFNN (1.0, 2.0)	0.092	0.279	0.127	0.326	0.371	0.510	0.414	0.374	0.416

Figure: For each task, the asterisk indicates which AdaFNN hyperparameters performed best on the validation set, and the best performing method on the test data is indicated in bold.

[2] UK Power electricity data, wearable device data, Medfly and Mexfly data

Thank you!