NEURAL NETWORKS WITH FUNCTIONAL RESPONSES

Sidi Wu, Cédric Beaulac & Jiguo Cao

Department of Statistics and Actuarial Science, Simon Fraser University

Introduction

In functional data analysis (FDA), the regression of a functional response on a set of predictors can be a challenging task, especially if the relation between those predictors and the response is nonlinear. In this work, we adapt neural networks, a machine learning technique, to solve this problem.

We design a feed-forward neural network (NN) to predict functional curves with scalar inputs, using the following procedure:

1. Transform the functional response to a finite-dimension vector of coefficients;
2. Construct a NN with those coefficients as outputs and the scalar predictors as inputs;
3. Train the NN using proposed objective functions;
4. Predict the functional response using NN outputs.

Basic Assumptions

Suppose we have N subjects, and for the i-th subject, the input is a set of scalar variables $\boldsymbol{X}_i = \{X_{i1}, X_{i2}, ..., X_{iP}\}$, and the output is a functional variable $Y_i(t), t \in \mathcal{T}$ in the $L^2(t)$ space. **Note:** In reality, $Y_i(t)$ is usually measured in a discrete manner, for instance, at m_i time points or locations, with some observation error.

Representations of Functions (Dimension Reduction)

- Mapping to basis coefficients -

In FDA, it is common to represent functions using basis expansion, where the information of $Y_i(t)$ can be summarized into a set of finite-dimensional vector of basis coefficients as:

$$Y_i(t) = \sum_{k=1}^{K} c_{ik}\theta_k(t) = \boldsymbol{\theta}'\boldsymbol{C}_i \quad (1)$$

- $\boldsymbol{\theta}$: vector of the basis functions $\theta_1(t), ..., \theta_K(t)$ from a selected basis system, e.g. Fourier or B-spline
- $\boldsymbol{C_i}$: vector of the basis coefficients $\{c_{ik}\}_{k=1}^K$
- K: some pre-defined truncation integer

- Mapping to FPC scores -

The other popular method for dimension reduction is *functional principal component analysis* (FPCA). Let $\mu(t)$ and $K(t,t')$ be the mean and covariance functions of $Y(t)$, and accordingly, $K(t,t') = \sum_{k=1}^{\infty} \lambda_k\phi_k(t)\phi_k(t')$, where $\{\lambda_k, k \geq 1\}$ are the eigenvalues and ϕ_k's are the corresponding eigenfunctions satisfying $\int \phi_k^2(t)dt = 1$.

Denote $\tilde{Y}_i(t) = Y_i(t) - \mu(t)$ as the centered functional response, following the Karhunen-Loéve expansion, $\tilde{Y}_i$ can be approximated as:

$$\tilde{Y}_i(t) = \sum_{k=1}^{K} \xi_{ik}\phi_k(t) = \boldsymbol{\phi}'\boldsymbol{\xi}_i \quad (2)$$

- $\boldsymbol{\phi}$: vector of the first K functional principal components (FPCs)
- $\boldsymbol{\xi}_i$: vector of the FPC scores $\{\xi_{ik}\}_{k=1}^K$, where $\xi_{ik} = \int\{Y_i(t) - \mu(t)\}\phi_k(t)dt$
- K: the truncation integer determined by the desired proportion of variance explained

NNBB & NNSS

- NN for Basis Coefficients (NNBB) -

Given Eq.(1), learning how X's regress on $Y(t)$ can be naturally replaced with learning how X's regress on basis coefficients $\{c_k\}_{k=1}^K$. Hence, we set $\{c_k\}_{k=1}^K$ to be a function of X's, with a mapping function $F(\cdot)$ from $\mathbb{R}^P$ to $\mathbb{R}^K$, as:

$$\boldsymbol{C}_i = F(\mathbf{X}_i) \quad (3)$$

Eq. (3) can be extended to the mapping from $\boldsymbol{X}$ to the functional response $Y(t)$ as $Y_i(t) = \boldsymbol{\theta}'F(\boldsymbol{X}_i)$.

Then we propose to apply a dense feed-forward NN as the mapping function $F(\cdot)$, where the basis coefficients $[c_{i1}, c_{i2}, ..., c_{ik}] \in \mathbb{R}^k$ are the outputs of the NN. The model can be expressed as:

$$\boldsymbol{C}_i = \text{NN}_\eta(\boldsymbol{X}_i) = g_L\left(\cdots g_1\left(\sum_{p=1}^{P} w_{1p}X_{ip} + b_1\right)\right) \quad (4)$$

- $g_1, ..., g_L$: the activation functions at each layer
- η: NN parameter set consisting of weights $\{w_{\ell k}\}_{\ell=1}^L$ and bias $\{b_\ell\}_{\ell=1}^L$ of all hidden layers

$\text{NN}_\eta(\cdot)$ is optimized by minimizing the objective function:

$$L_{\boldsymbol{C}}(\eta) = \frac{1}{n_{\text{train}}}\sum_{i=1}^{n_{\text{train}}}\sum_{k=1}^{K}(\hat{c}_{ik} - c_{ik})^2 \quad (5)$$

where n_{train} is the no. of samples in the training set, and c_{ik}'s are obtained following Eq.(1).

- NN for FPC Scores (NNSS) -

Similarly, we can use FPC scores to represent $Y(t)$ and be the outputs of the NN, and we obtain:

$$\boldsymbol{\xi}_i = \text{NN}_\eta(\boldsymbol{X}_i) = g_L\left(\cdots g_1\left(\sum_{p=1}^{P} w_{1p}X_{ip} + b_1\right)\right). \quad (6)$$

and $\text{NN}_\eta(\cdot)$ is trained w.r.t. the objective function $L_{\boldsymbol{C}}(\eta) = \frac{1}{n_{\text{train}}}\sum_{i=1}^{n_{\text{train}}}\sum_{k=1}^{K}(\hat{\xi}_{ik} - \xi_{ik})^2$.

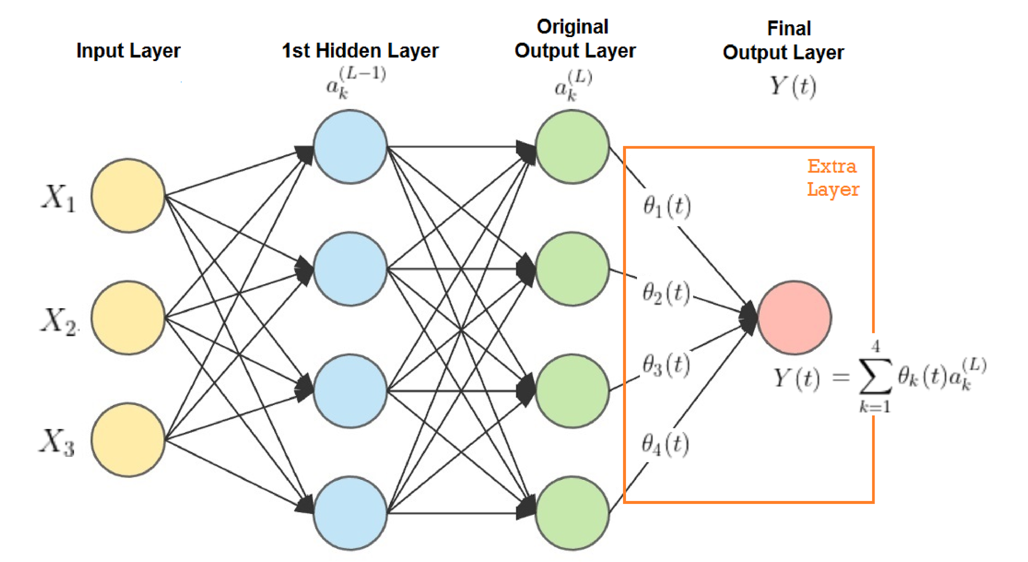
NNBR & NNSR

We further propose to modify the objective function to directly minimize the prediction error of the response variable:

$$L_{\boldsymbol{Y}}(\eta) = \frac{1}{n_{\text{train}}}\sum_{i=1}^{n_{\text{train}}}\sum_{j=1}^{m}(Y_i(t_j) - \hat{Y}_i(t_j))^2. \quad (7)$$

Note: Eq.(7) is implementable because the relation between $\hat{Y}_i(t)$ and $\hat{\boldsymbol{C}}$ (or $\hat{\boldsymbol{\xi}}$) is linear, thus we can easily compute the derivative of $\hat{Y}_i(t)$ as well as the gradient of $(Y_i(t) - \hat{Y}_i(t))^2$ with respect to $\hat{\boldsymbol{C}}$ (or $\hat{\boldsymbol{\xi}}$).

NNBB (or NNSS) trained by minimizing Eq.(7) is named NNBR (or NNSR), and can be treated as a NN with an extra output layer, where the final output is the weighted sum of the original outputs.



A graphical representation of the proposed neural network with an extra output layer ($L = 2, P = 3, K = 4$).

More Extensions

Eq.(7) can be further modified for different needs:

- **Irregularly-spaced functional data**

$$L_{\boldsymbol{Y}_{irr}}(\eta) = L_{\boldsymbol{Y}}(\eta) \cdot 1\,(Y_i(t_j) \text{ is observed}) \quad (8)$$

- **Smoothness control for $\hat{Y}(t)$**

$$L_{\boldsymbol{pen}}(\eta) = L_{\boldsymbol{Y}}(\eta) + \lambda\sum_{j=3}^{K}(\Delta c_k)^2 \quad (9)$$

where $\Delta^2 c_k = c_k - 2c_{k-1} + c_{k-2}$ is the difference of a set of consecutive basis coefficients.

Implementation

- Data & Models for Comparison -

- **Data**: generated by

$$Y(t_j) = \sum_{k=1}^{10} \xi_k(\mathbf{X})\phi_k(t_j) + \epsilon(t_j), j = 1, ..., 40$$

  - $\mathbf{X} = \{X_1, ..., X_{10}\}$: vector of random predictors
  - $\xi_k(\cdot)$: nonlinear functions for some k
  - $\phi_k(\cdot)$: B-spline basis functions
  - $\epsilon(\cdot)$: random noise function
- **Models**: Function-on-scalar regression model (FoS), NNBB, NNSS, NNBR & NNSR

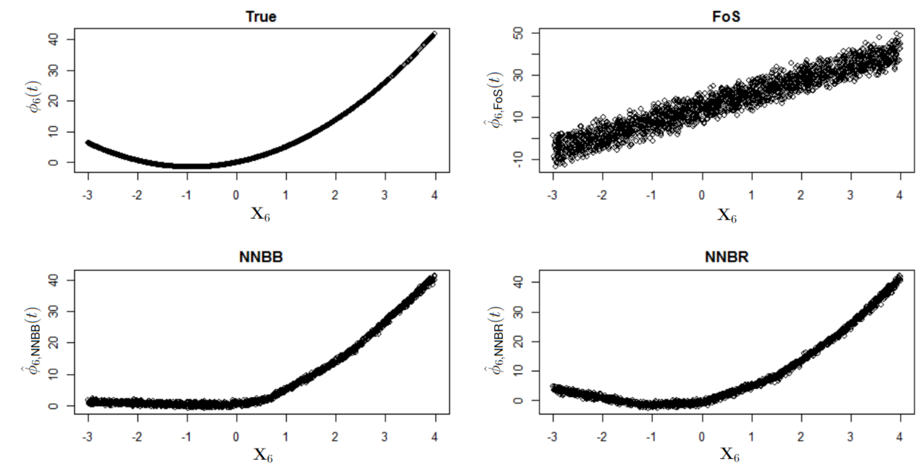
- Results -

- **Prediction Accuracy**

Methods	FoS	NNBB	NNSS	NNBR	NNSR
Mean	24.5373	3.8478	5.7422	1.1548	1.7862
Std. Dev.	0.7632	0.7914	0.2055	0.0958	0.0810
p-value	-	<2.2e-16	<2.2e-16	<2.2e-16	<2.2e-16

Table of Mean(SD) of MSE between $Y(t)$ and $\hat{Y}(t)$ for various models in test sets (20%), along with the p-values of the two-sided paired t-test of MSE of NN-based model comparing that of FoS, given 20 different training iterations.

- **Relation Reconstruction**



Visualizations of true $\phi_6(t)$ (top left), $\hat{\phi}_{6,\text{FoS}}(t)$(top right), $\hat{\phi}_{6,\text{NNBB}}(t)$ (bottom left), and $\hat{\phi}_{6,\text{NNBR}}(t)$ against $\mathbf{X}_6$ (bottom right), respectively.

Summary

- Highlights-

- Have superior predictive power, especially when the relation between the predictors and the response are non-linear.
- Flexible for both regularly or irregularly spaced functional data.
- Can handle a large number of predictors.

- Limitations -

- Contain many hyper-parameters and the tuning process is time-consuming.

- Potentials -

- Extend to predict a multi-dimensional (mainly two-dimensional) functional response.
- Combine with existing NN with functional inputs to construct NN architectures for both functional predictors and functional responses.

References

[1] J. O. Ramsay and B. W. Silverman. *Functional Data Analysis (Second Edition).* Springer, 2005.

[2] Fabrice Rossi and Brieuc Conan-Guez. Functional multi-layer perceptron: a non-linear tool for functional data analysis. *Neural Networks*, 18(1):45–60, 2005.

[3] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Contact information:
Sidi Wu
sidi_wu@sfu.ca