Designing neural network layers for functional data analysis

Cédric Beaulac

Université du Québec à Montréal

December the 16th 2023

The work presented here is the result of a collaboration with Ph.D candidate Sidi Wu and professor Jiguo Cao from Simon Fraser University and with research intern Valentin Larchevêque from Université Montpellier.

Designing neural network layers for functional data analysis

Functional data
Functional Output layer
Functional Input layer
Deep learning models and implementation

The talk

- Definition of functional data.
- Parametric representation of functional data.
- Regression of functional data.
- Functional layers.
- Proposed models.

Functional data

A brief introduction to functional data

- In functional data analysis (FDA), a replication $x_i(t)$ $t \in T$ is a function.
- The space over which the function is defined T can be time, spatial space, or higher-dimension spaces.
- A data set is a collection of such functions $S = \{x_i(t) | i \in (1, ..., n)\}$ over the same space.
- The data is collected as a collection of points over the space.

A simple functional data set.

- For the presentation, we suppose T is one-dimensional and is some measure of time.
- From now on, let us say the space-time is $[0, T]$.

A sample of points over T

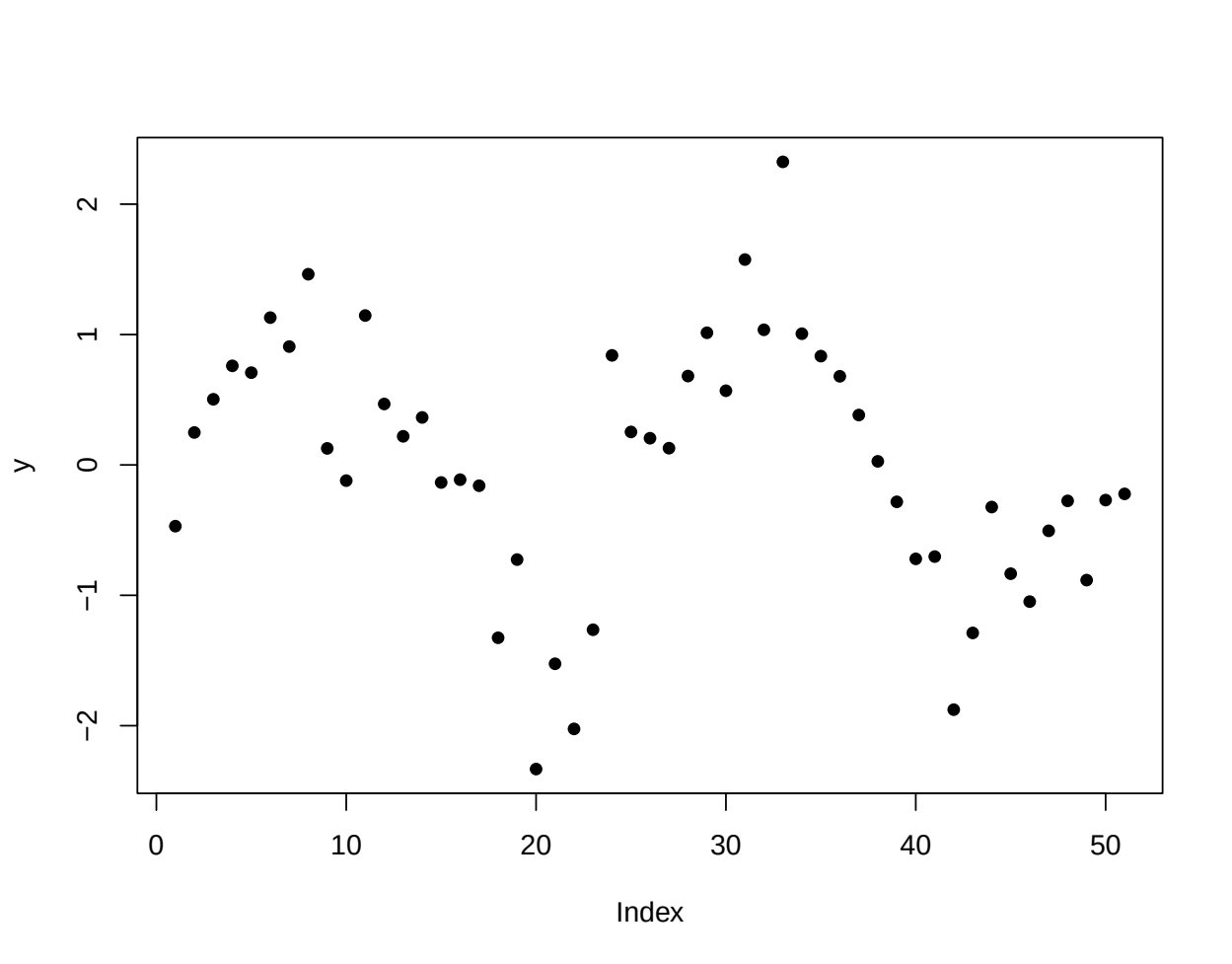

Figure: Sample from a functional data.

A simple functional data set.

This is different from time series:

- We are not trying to forecast the functional data.
- We must have multiple repetitions over T.

Multiple subjects: A sample of points over T

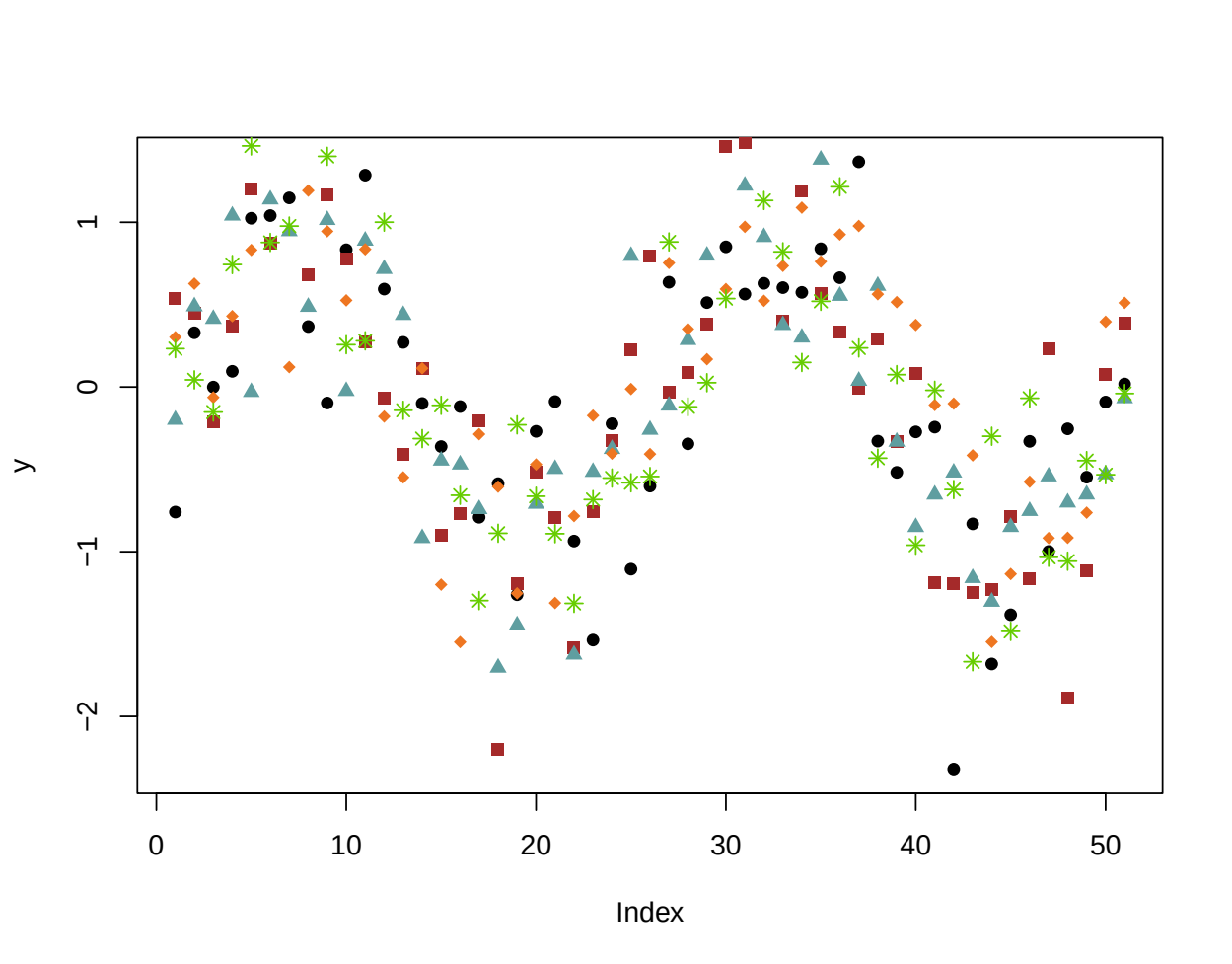

Figure: Sample from multiple functional data.

Exemple: real data set (El Nino)

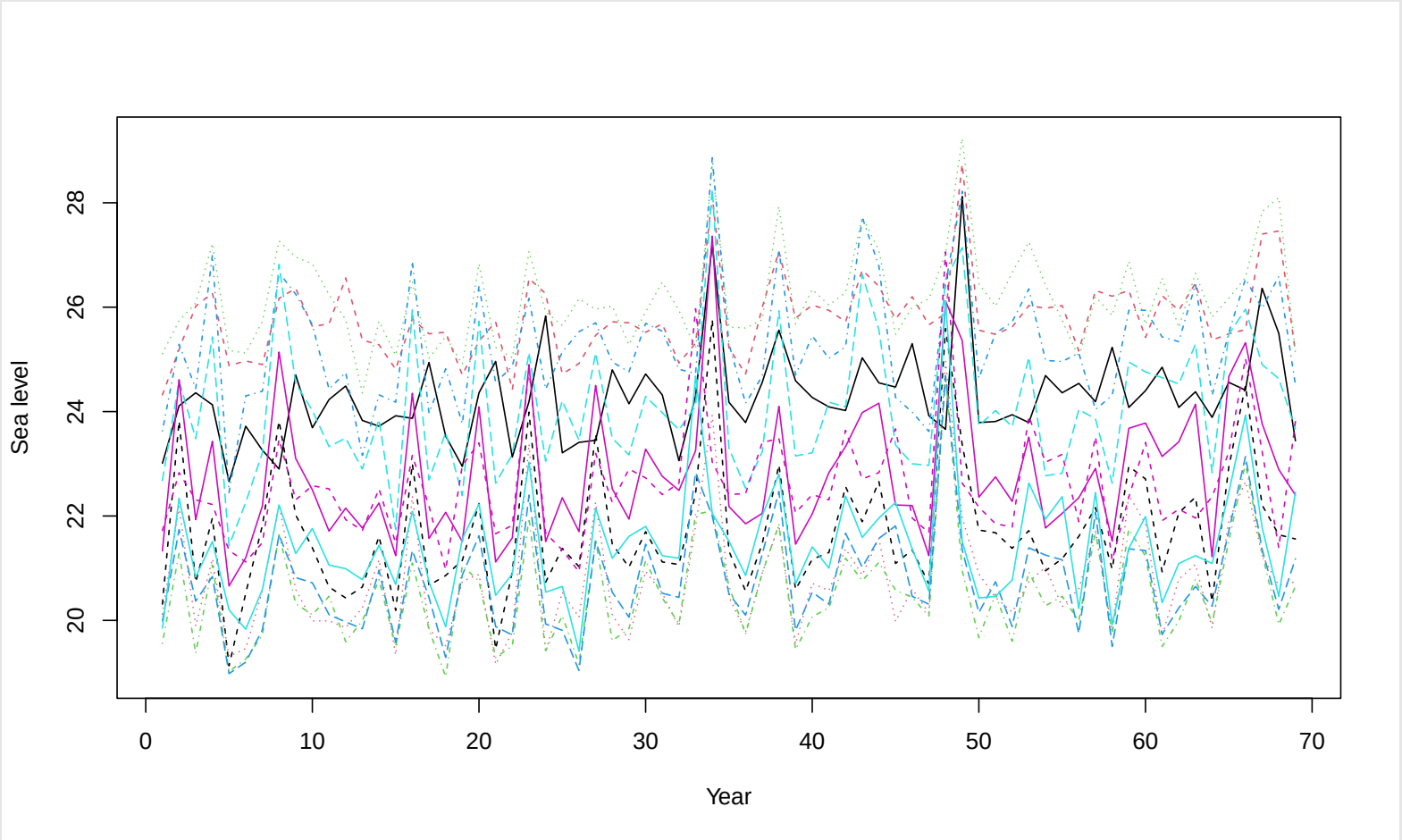

Figure: Yearly sea surface temperature.

Exemples

- Weight over time.
- Daily sugar level.
- Electroencephalography.

Functional data representation

- **It is assumed that the functional data is a realization of an underlying smooth stochastic process.**
- It is common to interpolate points and produce a smooth representation for $x_i(t)$
- This smooth representation is later used in the analysis.

Functional data representation

- The standard approach to do so is to use some basis expansion; a set of basis functions (that cover the space $[0, T]$) and a set of basis coefficients.
- Thus, the continuous and smooth curve is estimated as a linear combination of those functions and parameters.
- $x(t) = \sum_{k=1}^{K} c_k B_k(t)$

Functional data representation

- $x(t) = \sum_{k=1}^{K} c_k B_k(t)$
- We obtain a continuous representation of $x(t)$ (can be evaluated for any $t \in [0, T]$).
- We only need to *learn* a discrete number of parameters to produce a smooth and continuous representation of functional data.
- $\sum_j [x(t_j) - \sum_{k=1}^{K} c_k B_k(t_j)]^2$

B-Splines

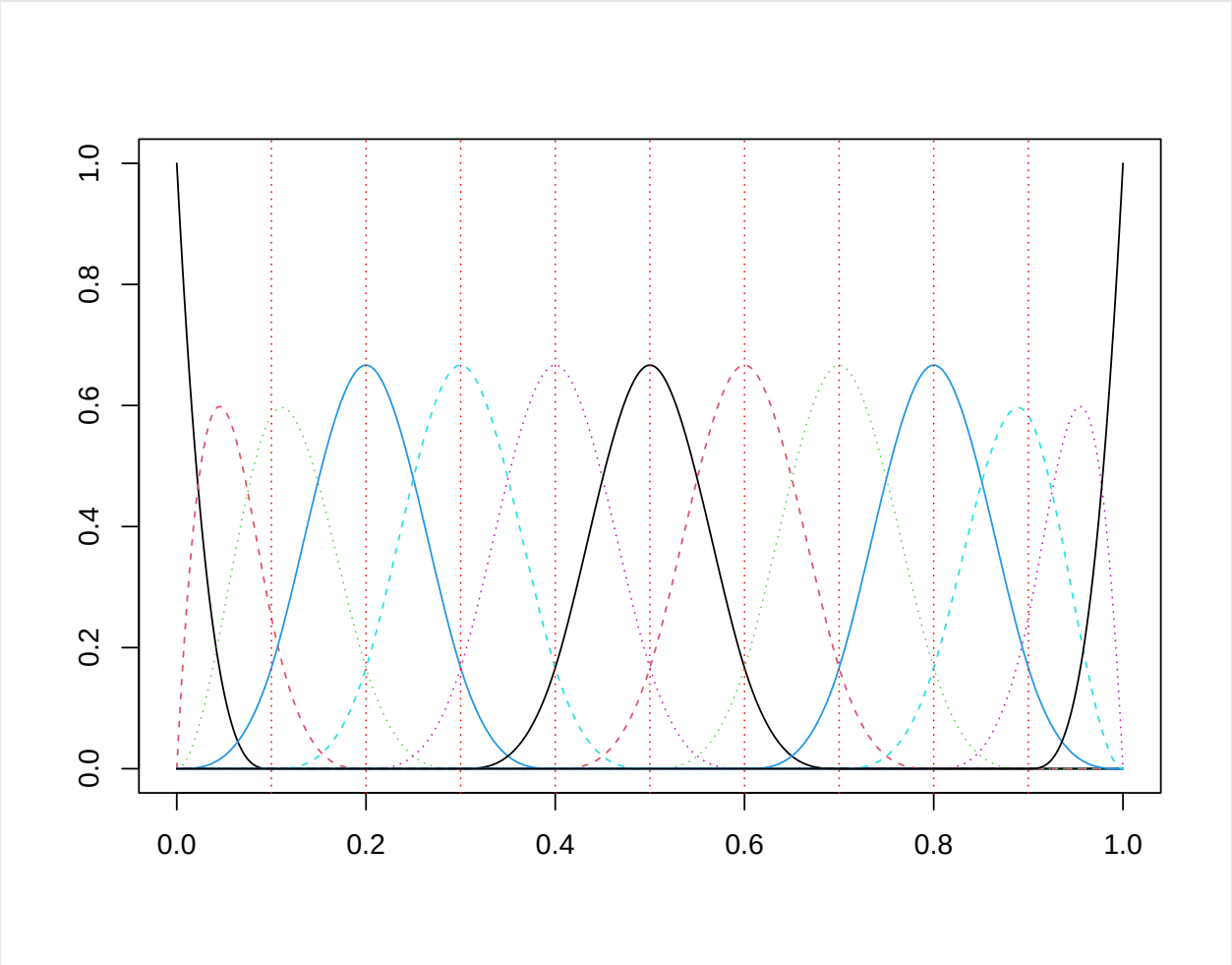

Figure: The thirteen basis functions defining and order four splines with nine interior knots.

A sample of points over T

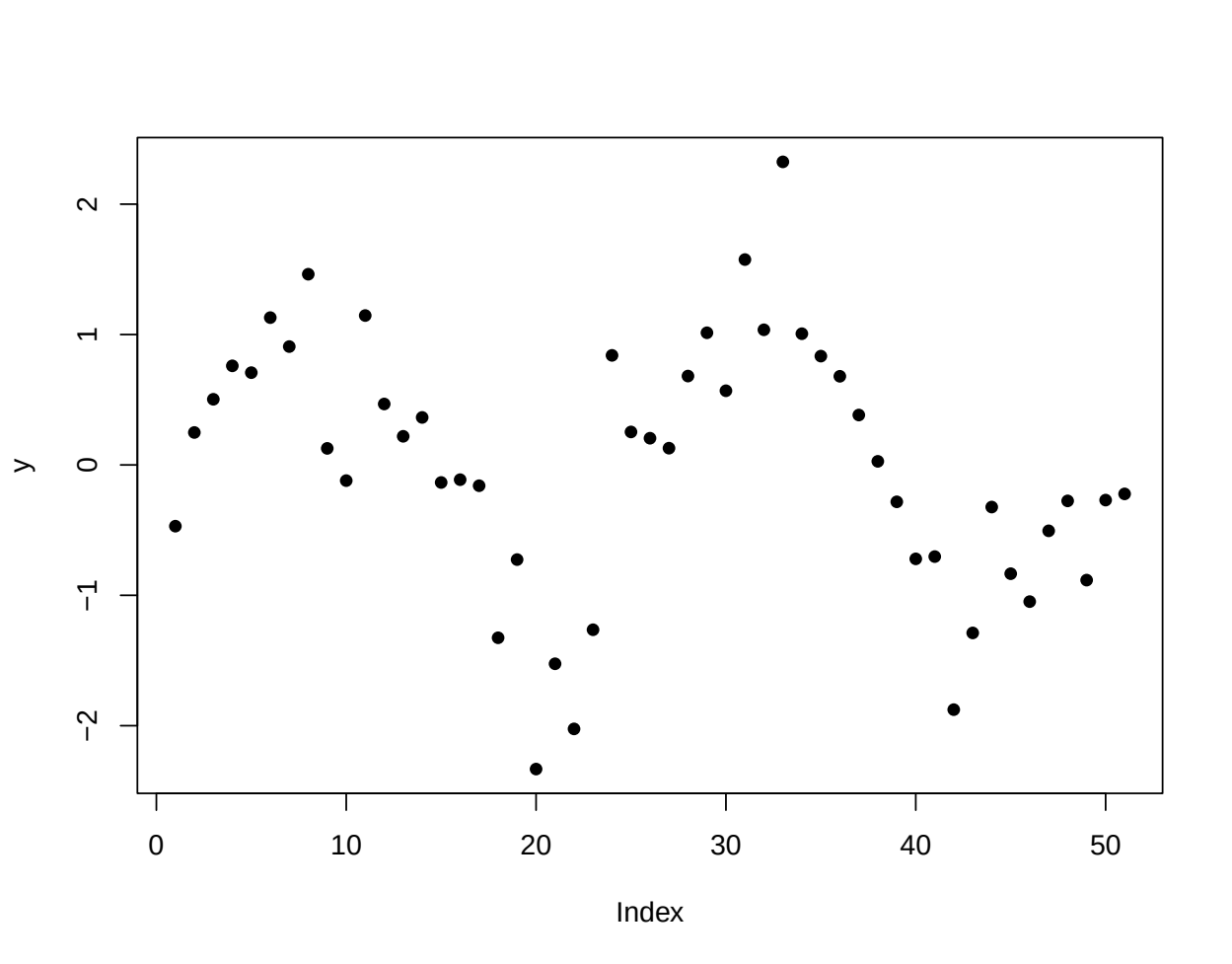

Figure: Sample from a functional data.

Smooth and continuous function over T

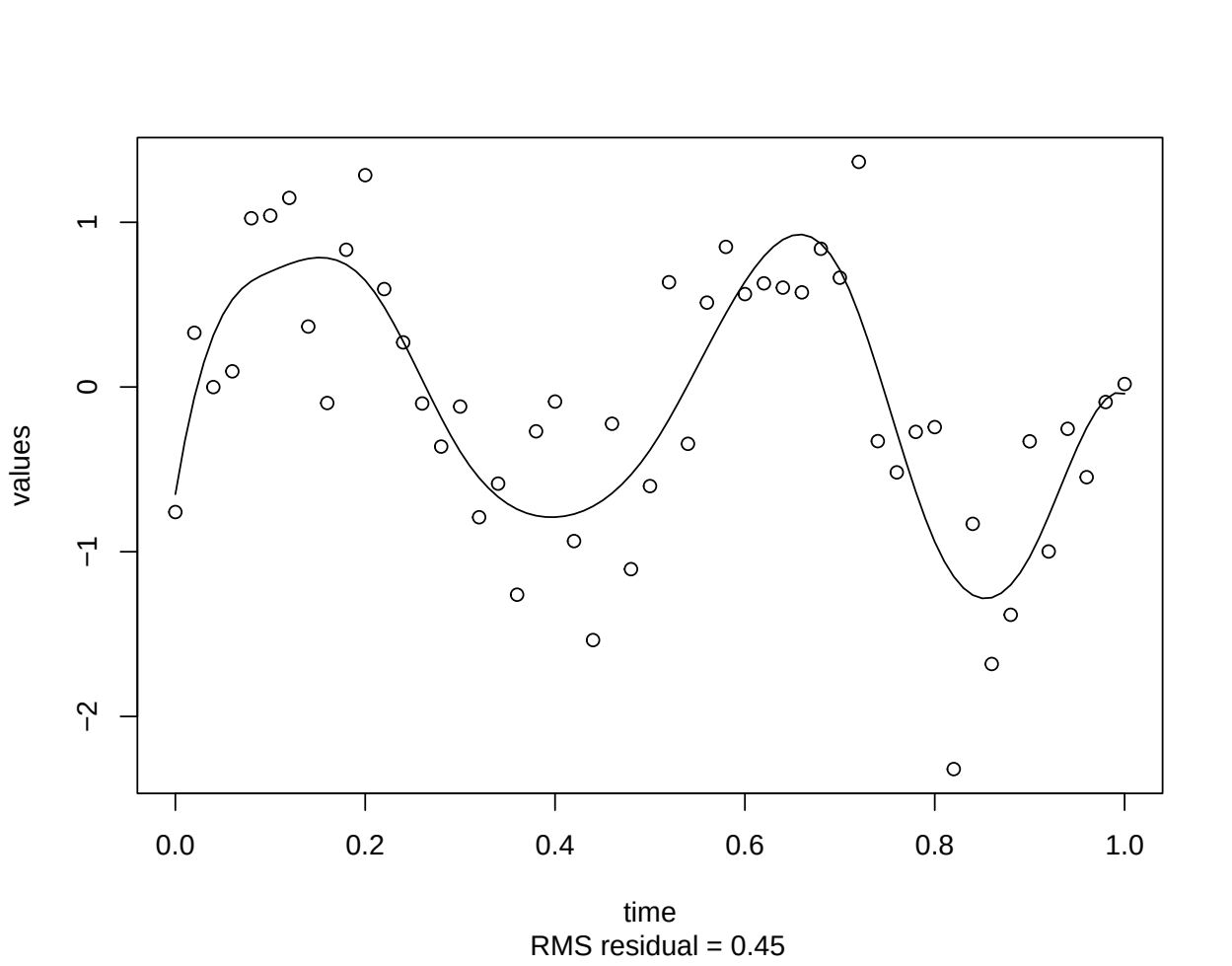

Figure: Smoothing the sample from a functional data.

Smooth and continuous function over T

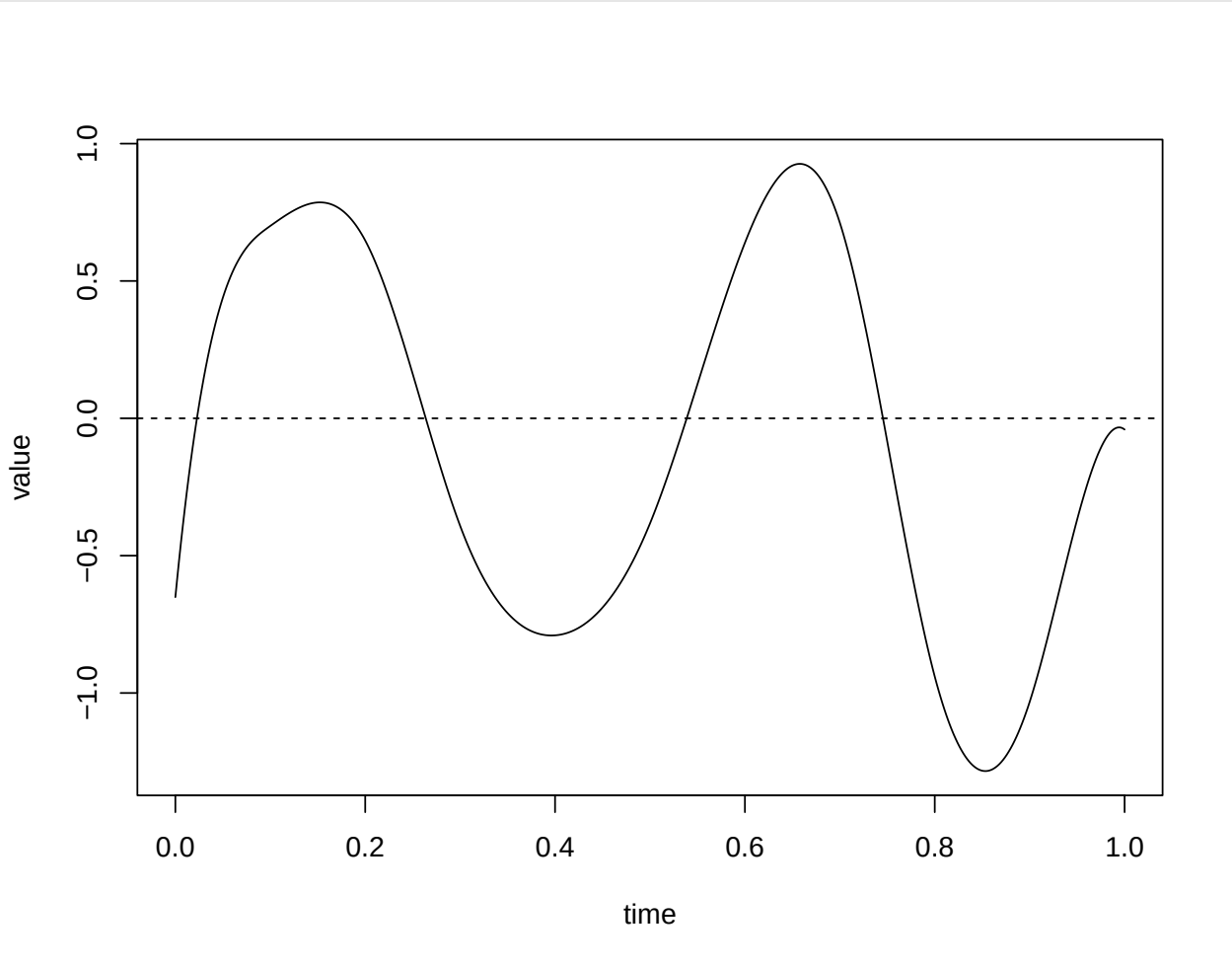

Figure: Keep the curve (smooth and continuous representation)

Functional data representation

- The data is now represented using the B-Spline basis functions (for instance with order 4 and 9 interior knots)
- This observation is later used in the analysis.
- With the following coefficients: [-0.65115973, 0.53578405, 0.70073762, 0.94566931, -0.59899313, -0.89844612, -0.54772926, 0.79263628, 1.21055785, -1.36245376, -1.40150503, 0.05021092, -0.04074864]

Functional data representation

Smoothing has multiple benefits:

- Dimension reduction
- Easy-to-compute derivatives
- Manages irregularly-spaced data
- *Smooth out* measurement errors

Irregularly-space data

What is irregularly-spaced data:

- Irregularly-collected data.
- Different subject/repetitions i are collected at different times $t \in [0, T]$
- The can also be collected a different number of times.

Irregularly-space data

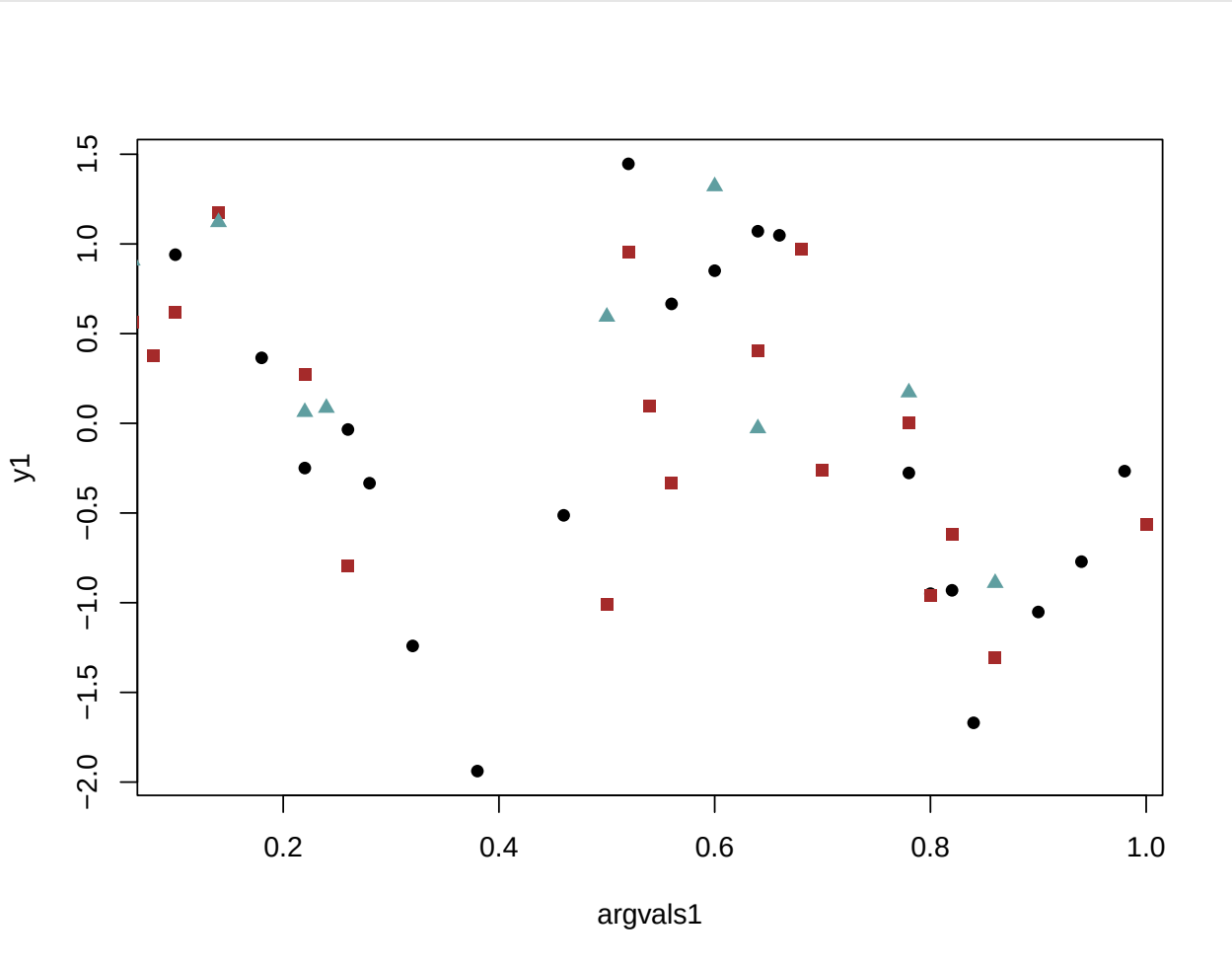

Figure: Irregularly-space functional data.

Problems in functional data analysis

1. Represent the data in a way that aid further analysis.
2. Display the data so as to highlight various characteristics.
3. Study important sources of variation among the data. (Functional principal component analysis)
4. **The regressions of functional data onto scalar variable and vice versa.**

Functional regression

- We cannot simply treat the observed points over $[0, T]$ as scalar and use regular regression.
- Different observations might observe data at different moments.
- Different observations might be observed a different number of times.
- Observations at time points close to another are related.

Traditional Regression cannot be directly applied.

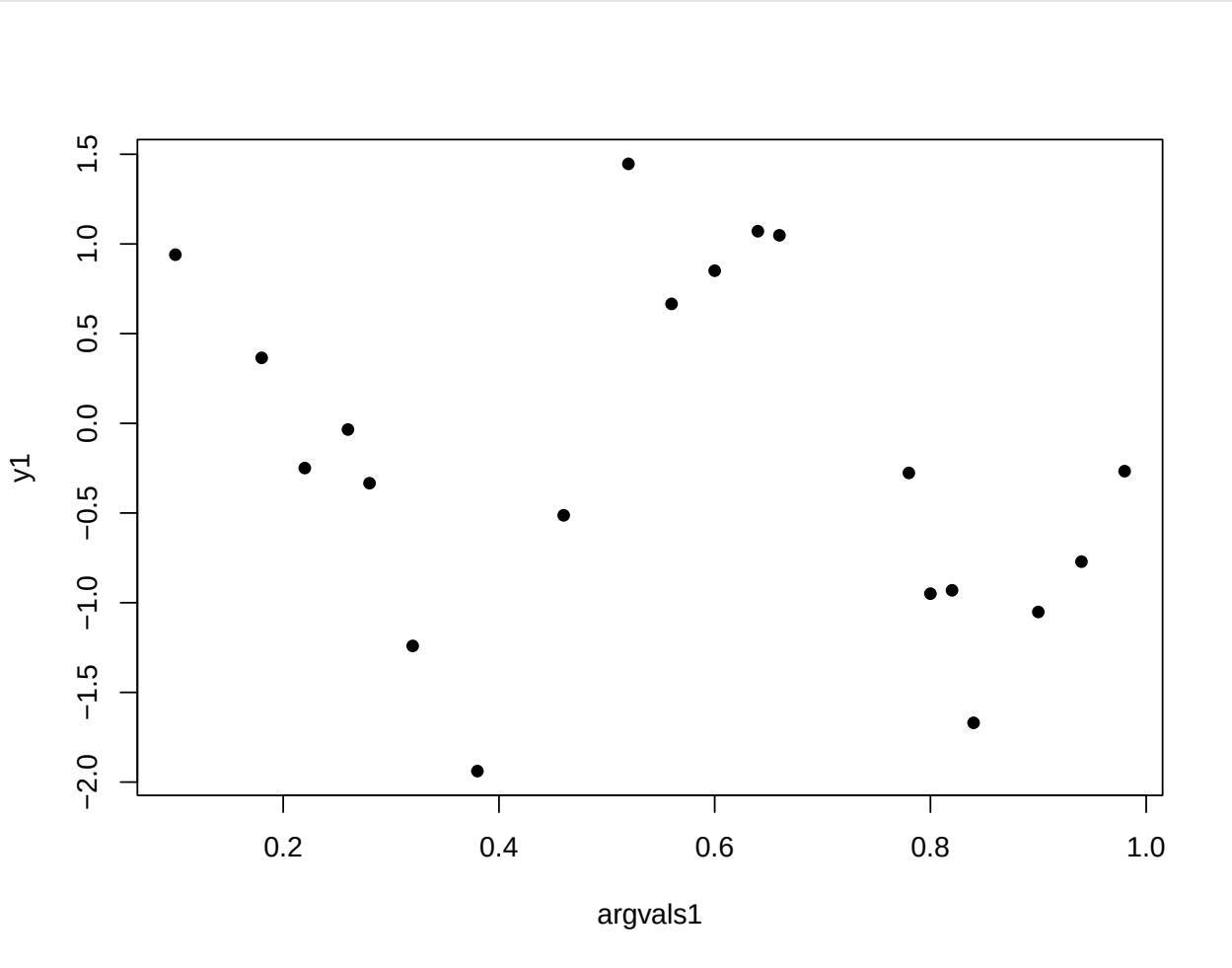

Figure: Sample from a functional data.

Traditional Regression cannot be directly applied.

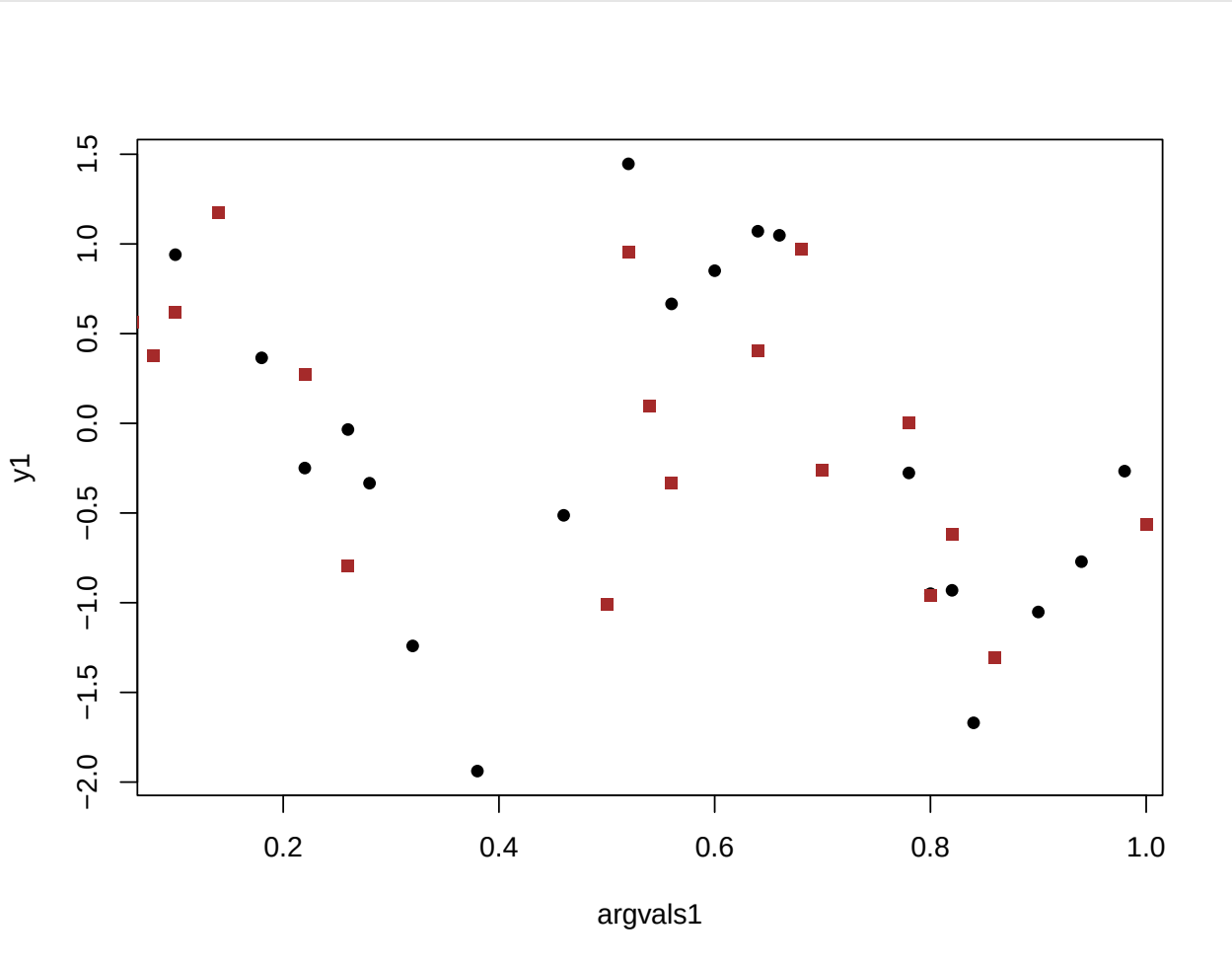

Figure: Sample from a functional data.

Traditional Regression cannot be directly applied.

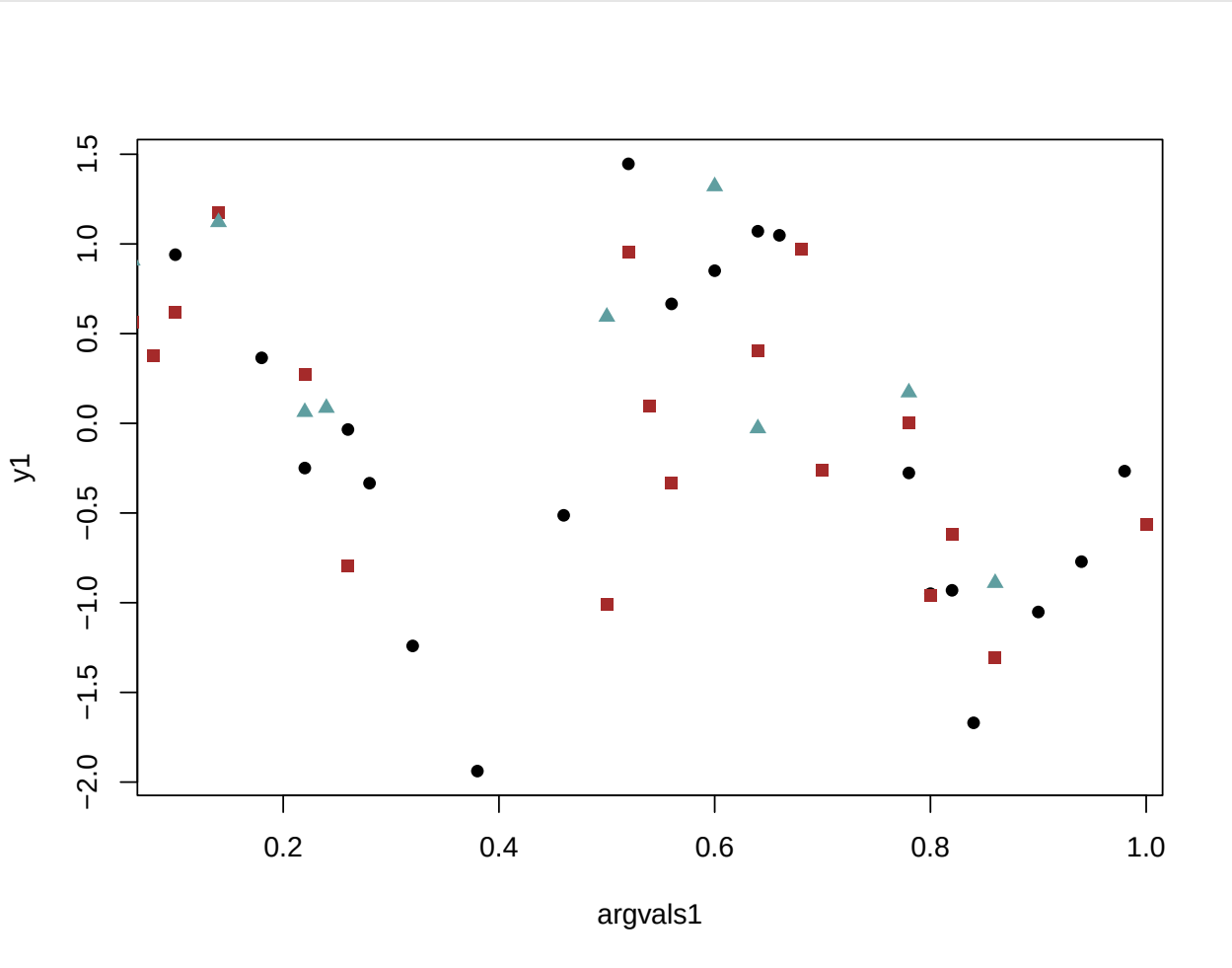

Figure: Sample from a functional data.

Functional regression problems

- Function on scalar (FoS) regression.
- Scalar on functional (SoF) regression.
- Function on function (Fof) regression.

Function on scalar regression (FoS)

- Scalar predictors, functional response.
- Parameters are now functions that must be estimated for $t \in [0, T]$.
- The model takes the form $y(t) = \beta_0 + \sum_{j=1}^{p} x_j \beta_j(t) + \varepsilon(t)$

Function on scalar regression (FoS)

- A common approach is to smooth the response, $y_i(t) = \sum_{k=1}^{K} c_k^i B_k(t)$
- The we regress the coefficients c_k onto the predictors x.
- Thus we learn a matrix of parameters B such that $\mathbf{c} = \mathbf{x}B$ using least square.
- This means, doing two steps of least square sequentially.

Scalar on function regression (SoF)

- Functional predictors, scalar response.
- We take the inner product between the functional predictor and a functional weight.
- The model takes the form: $y = \beta_0 + \int_T x(t)\beta(t)dt + \varepsilon$
- A bit more complicated than FoS regression.

Scalar on function regression (SoF)

- A simple solution is to express the parameter $\beta(t)$ using a basis expansion: $\beta(t) = \sum_k c_k B_K(t)$
- $Y = \beta_0 + \sum_k c_k \int_T x(t) B_K(t) dt + \varepsilon$
- The parameters are now the basis coefficients (c_k) and they can be estimated provided a numerical solution for $\int_T x(t) B_k(t)$ for all k.

Functional regression

- Those are not the only way to proceed.
- Our proposed method are inspired by these.

Neural network architectures for functional data analysis

Integrating deep learning into functional data analysis

- We seek solution to solve all three regression problems described.
- We seek ways that functional data can be integrated, either as input or output, in deep learning models.
- We focus on designing input and output layers that can be connected to already existing deep learning architecture (CNN, LSTM, RNN, etc...)
- Using the functional data as it is collected,
- but considering the smooth and continuous assumption.

Functional Output Layer

- We want to design a functional output layer for neural networks (NN).
- To integrate our model into currently existing deep learning architecture, we have to make sure it can be trained using back-propagation.
- The predictors entering the model can be of any form.
- We propose to first output k basis coefficients, and then through a deterministic layer we reconstruct the functional response.

Functional Output Layer

- The functional data that we have is the following:
- We know $\mathbf{t}^i = [t_1^i, t_2^i, ..., t_m^i]$ the vector of time points when the ith subject has been observed,
- and we have $\mathbf{y}_i = [y_i(t_1), y_i(t_2), ...y_i(t_m)]$.

Functional Output Layer

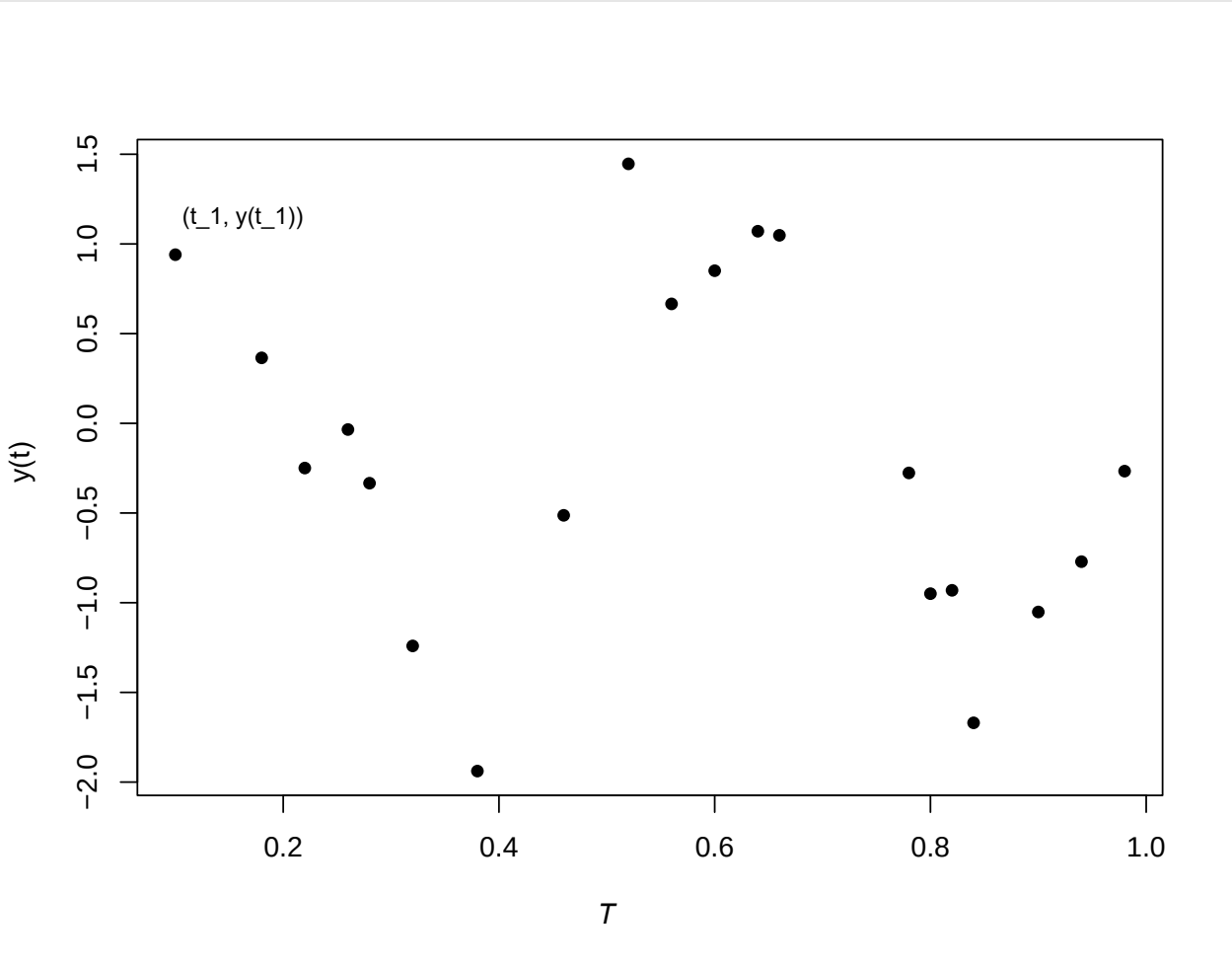

Figure: Sample from a functional data.

Functional Output Layer

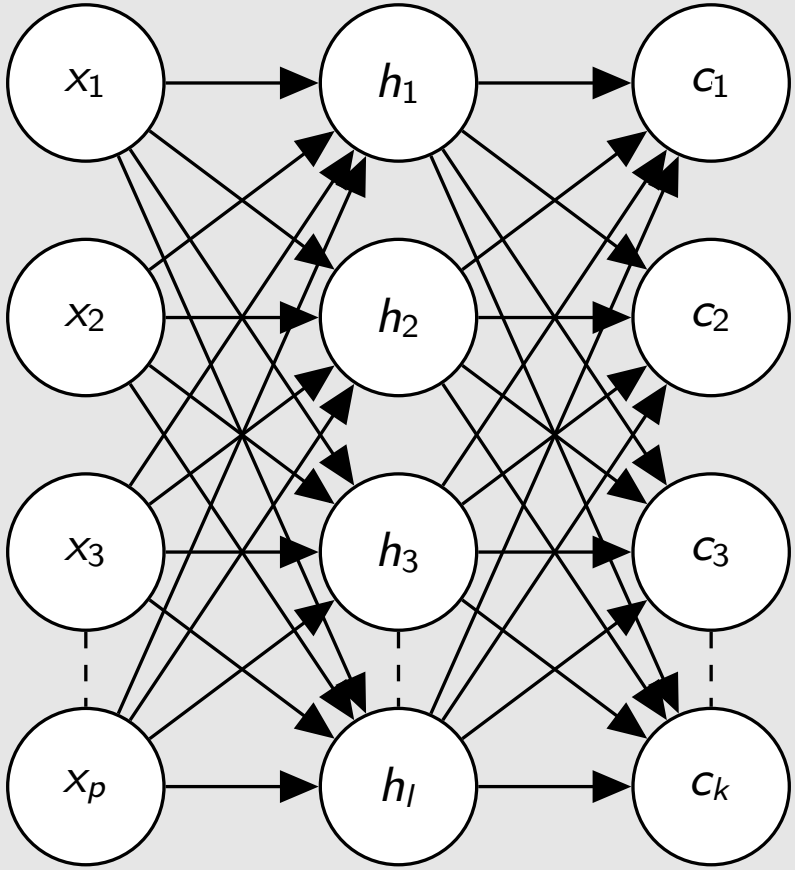

Figure: A 1-hidden layer NN to predict coefficients.

Functional Output Layer

- We want to train the model using the data $\mathbf{y}_i$,
- but we have a NN that outputs $\hat{\mathbf{c}}$: $\text{NN}(\mathbf{x}_i) = \hat{\mathbf{c}}^i$.
- However, we can construct a predicted response $\hat{y}_i(t) = \sum_{k=1}^{K} \hat{c}_k^i B_k(t)$.

Functional Output Layer

- We can evaluate $\hat{y}_i(t)$ at $\mathbf{t}^i$: $\hat{y}_i(\mathbf{t^i}) = [\hat{y}_i(t_1^i), \hat{y}_i(t_2^i), ...\hat{y}_i(t_m^i)]$ $= [\sum_{k=1}^{K} \hat{c}_k^i B_k(t_1^i), \sum_{k=1}^{K} \hat{c}_k^i B_k(t_2^i), ..., \sum_{k=1}^{K} \hat{c}_k^i B_k(t_m^i)]$
- We propose to construct $\mathbf{B}_{K\times m}$ where $B_{k,j} = B_k(t_j)$.
- Thus $\hat{y}_i(\mathbf{t^i}) = \widehat{\mathbf{c}}_{1\times K}^i \times \mathbf{B}_{K\times m}$.
- That way we can train the model using the following objective function : $L_Y = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} [y_i(t_j) - \hat{y}_i(t_j)]^2$

How can you train the model with $\hat{y}$? It is not the output of the NN!

Functional Output Layer

We can use custom operations that are not part of typical NN model, such as the matrix multiplication $\widehat{\mathbf{c}}^i_{1\times K} \times \mathbf{B}_{K\times m}$ because this operation is differentiable.

We only need to use the pytorch or keras *matmult* operator in this case.

Functional Output Layer

- By doing this 2-layers process we smooth the response and regress the coefficients onto x jointly. (a single optimization problem)
- We can train the model using the response variable collected $\mathbf{y}_i$
- $L_Y = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} [y_i(t_j) - \hat{y}_i(t_j)]^2$
- This can be thought of as adding a deterministic layer after the coefficient layer.

Functional Output Layer

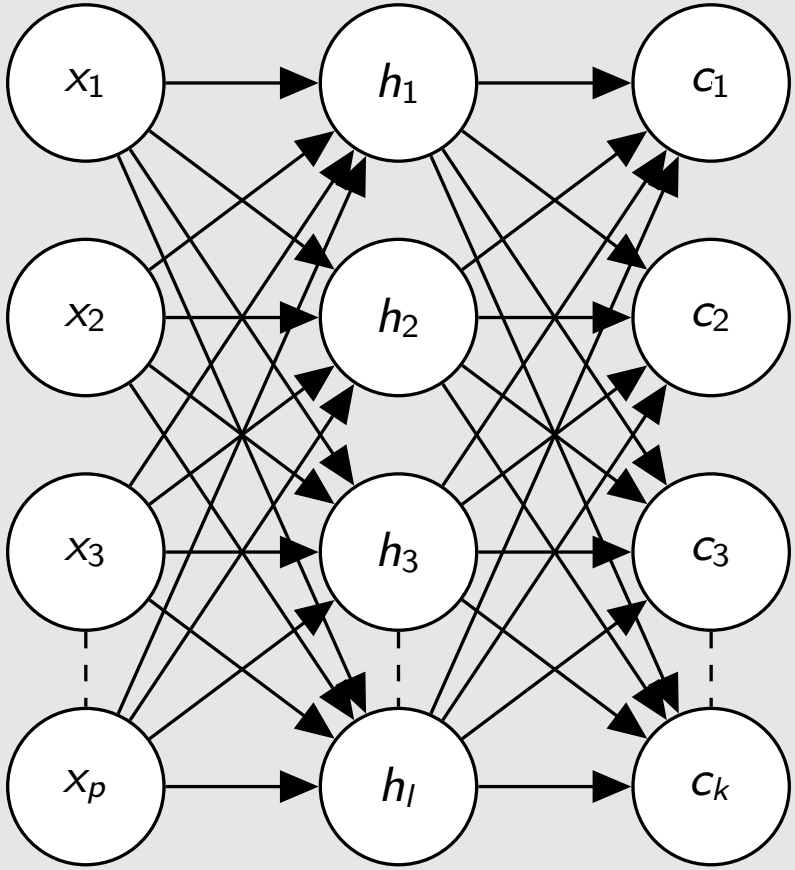

Figure: A 1-hidden layer NN to predict coefficients.

Functional Output Layer

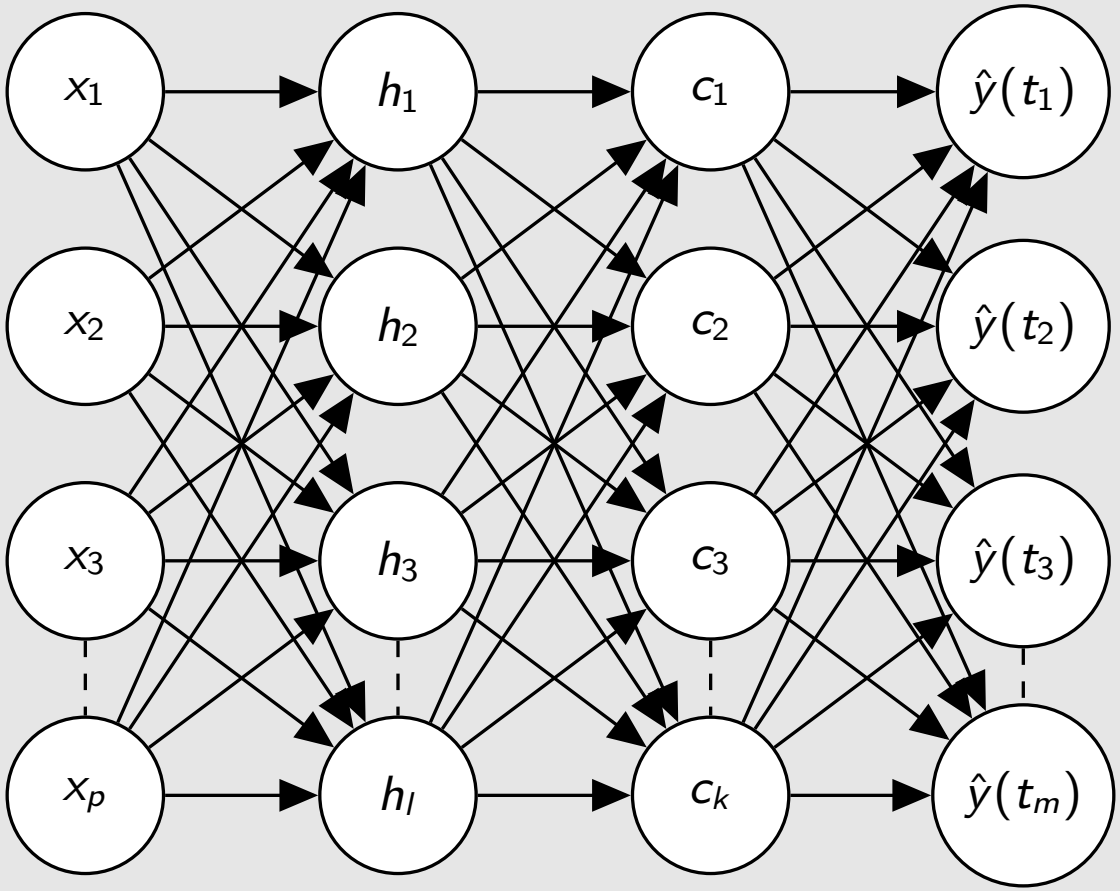

Figure: The model proposed can be perceived as adding a deterministic layer to the model.

Functional Output Layer

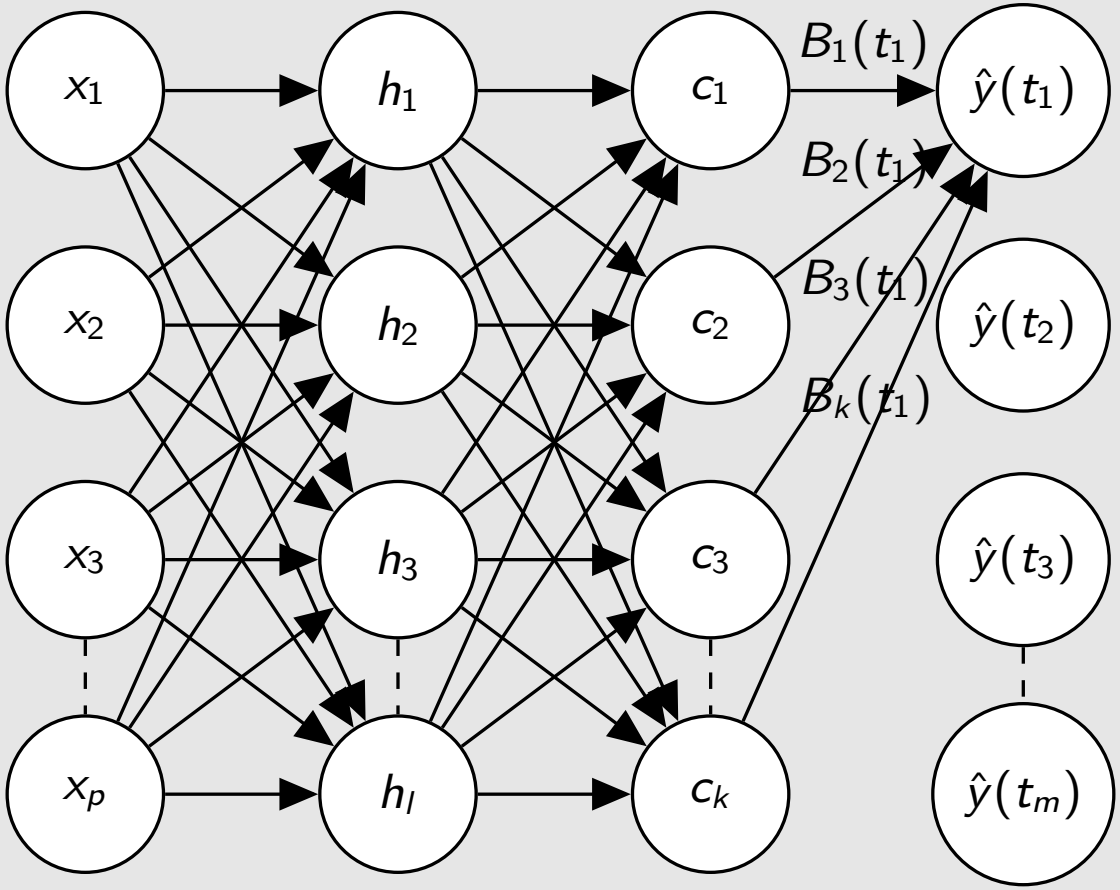

Figure: The model proposed can be perceived as adding a deterministic layer to the model.

Functional Output Layer

- This creates a continuous and smooth prediction ($\hat{y}(t)$ exists for every $t \in [0, T]$/interpolates)
- No need to first smooth the data.
- Learning a basis representation of the functional data is beneficial to further address common FDA concerns:
- Irregularly spaced data and smoothness regularization (coherent with literature).

Functional Input Layer

- To process functional input we need to learn a functional weight/parameter $\beta(t)$.
- $y = \beta_0 + \int_T x(t)\beta(t)dt + \varepsilon$
- Our concept is similar to the functional output layer.

Functional Input Layer

- Elements of the first hidden layer are of the form: $h_l = g(\int_T x(t)\beta_l(t))$
- with $\beta_l(t) = \sum_{k=1}^{K} c_{l,k} B_l(t)$
- This means the coefficients $\mathbf{c}$ are the parameters we are trying to learn in this layer through back-propagation (learning the functional weight).
- As long as all the operation involving $\mathbf{c}$ are differentiable, we should be able to train such model.

Functional Input Layer

$$h_l = g(\int_T x(t)\beta_l(t)) \tag{1}$$

$$= g(\int_T x(t)\sum_{k=1}^{K} c_{l,k}B_k(t)) \tag{2}$$

$$= g(\sum_{k=1}^{K} c_{l,k}\int_T x(t)B_k(t)) \tag{3}$$

$$= g(\sum_{k=1}^{K} c_{l,k}f_k) \tag{4}$$

where $f_k = \int_T x(t)B_k(t)$. We can learn c, by constructing a simple fully connected layer mapping $\mathbf{f}$ to $\mathbf{h}$:
$\mathbf{h} = g(\mathbf{Cf})$, where $\mathbf{C}$ is a l by K matrix of coefficients.

Simple MLP with features f as input

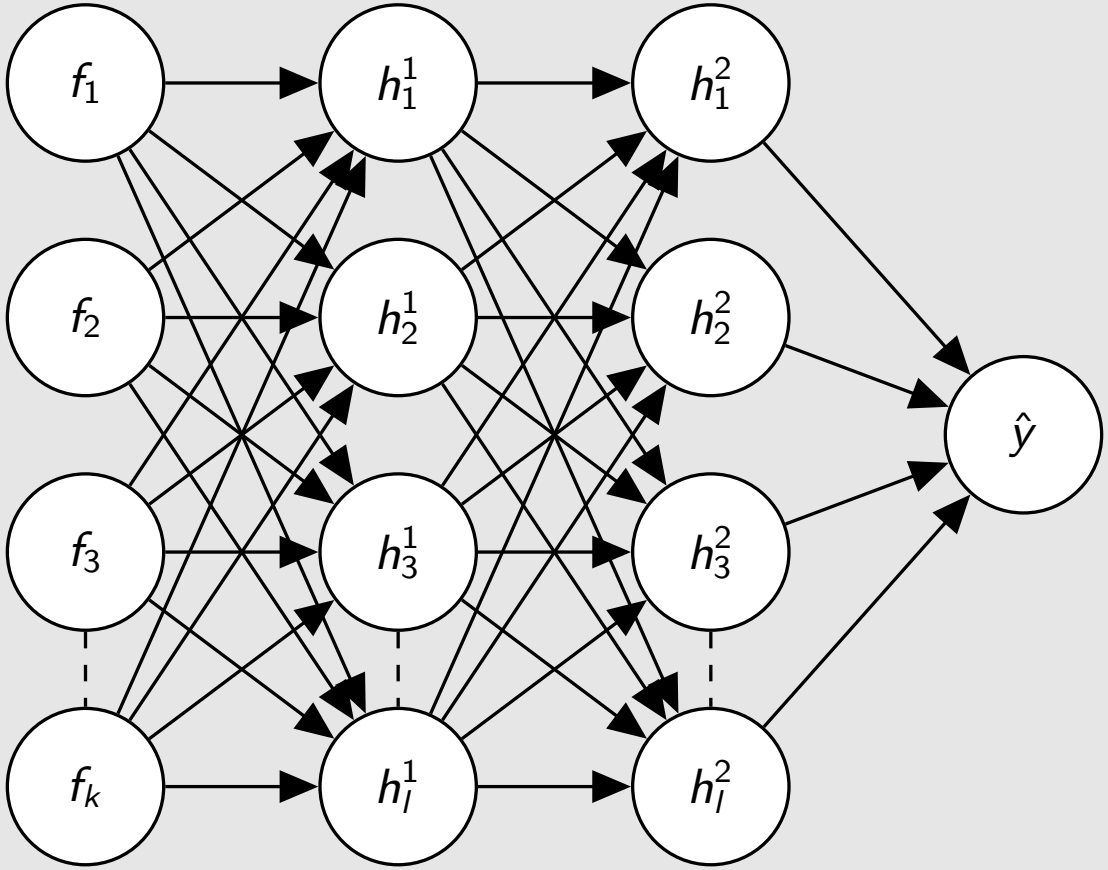

Figure: Simple MLP with scalar input and output.

Simple MLP with features f as input

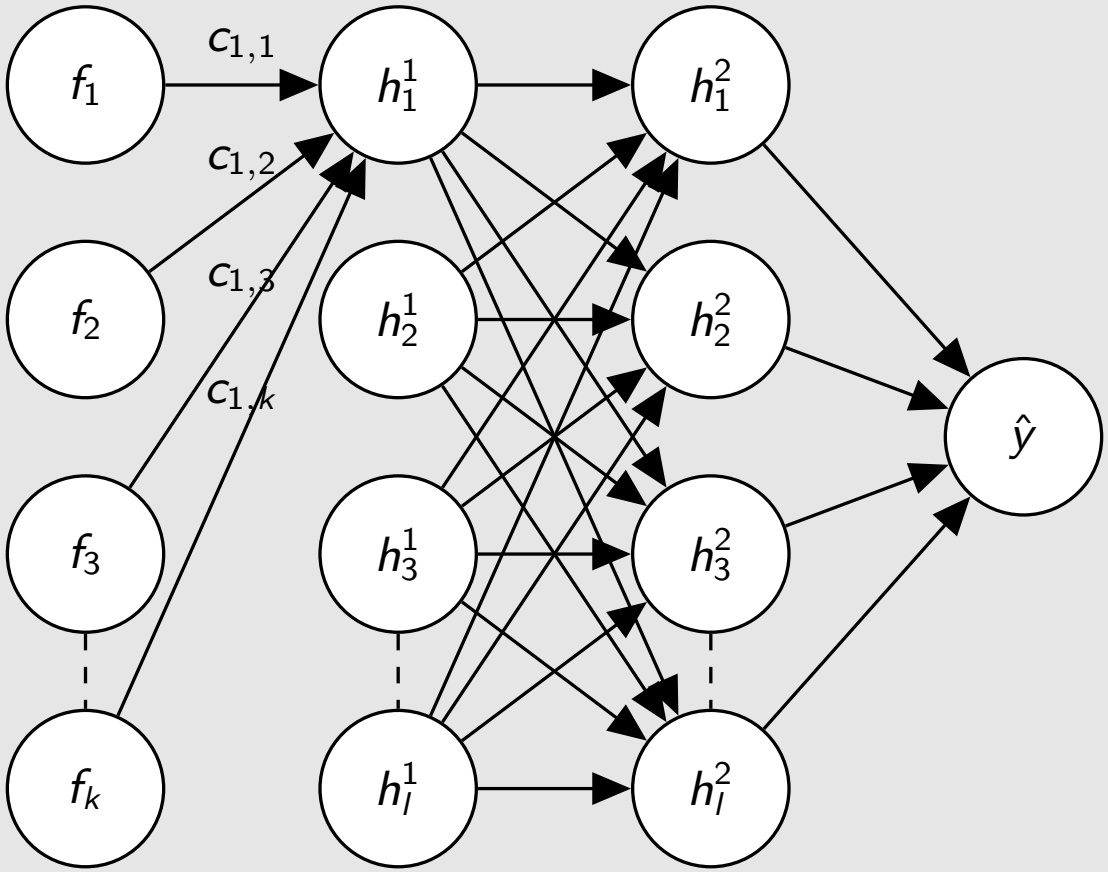

Figure: Simple MLP with scalar input and output.

Functional Input Layer

- $f_k = \int_T x(t)B_k(t)$.
- $x(t)$ is not observed as a function, but as a collection of points over $[0, T]$.
- We propose to directly estimate the integral with a summation.
- We propose $f_k = \sum_{j=1}^{m} w_j x(t_j)B_k(t_j)$ (numerical approximation).
- We can perceive this first step as a deterministic layer since $\mathbf{f}$ is a just a linear combination of $x(t)$

Functional Input Layer

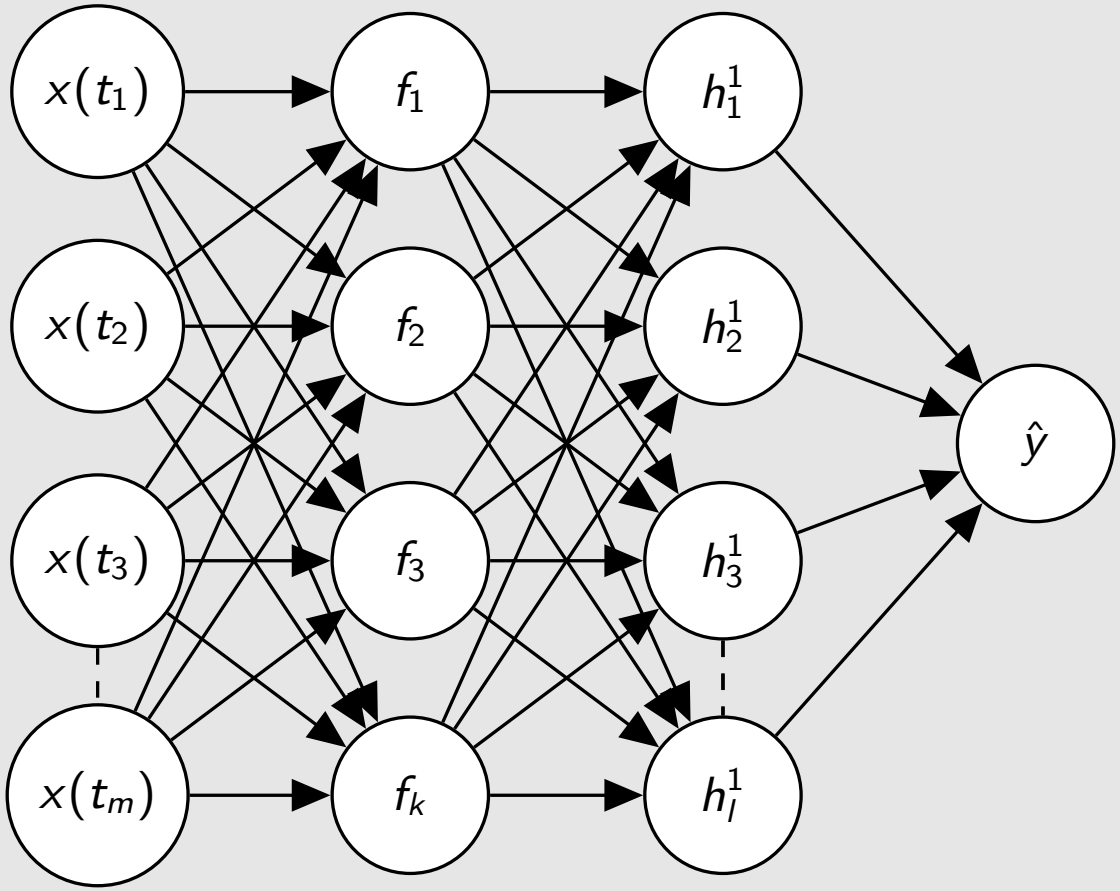

Figure: Simple MLP with scalar input and output.

Functional Input Layer

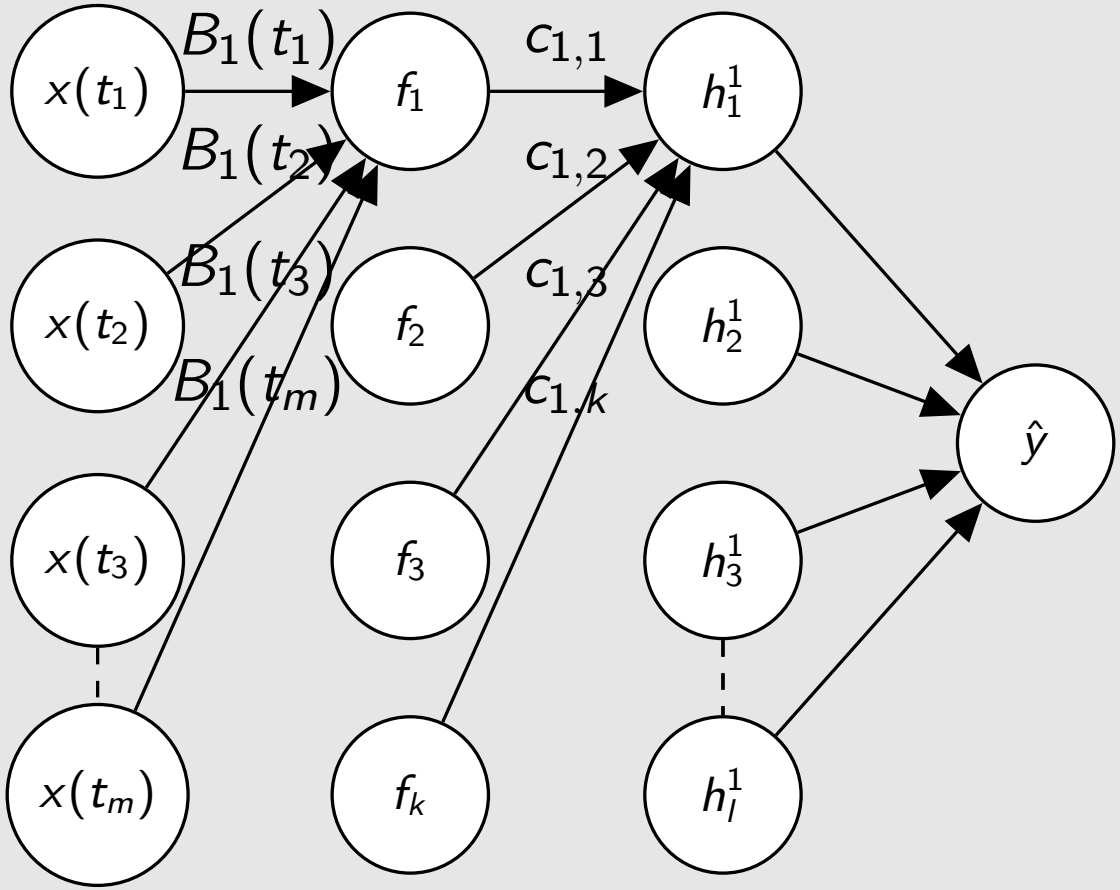

Figure: Simple MLP with scalar input and output.

Proposed models and results

Function on scalar regression

- We implemented FoS models for simple regression problems.
- Its strength lies in the ability of neural networks to capture non-linear relations.
- We tested our model on multiple simulated data sets and a real data set.

Wu, S., Beaulac, C. and Cao, J. Neural Networks for Scalar Input and Functional Output, Statistics and Computing, (33), 118, 2023.

Function on scalar regression : results

- To enforce non-linearity between scalar predictors $\mathbf{x}$ and functional response $y(t)$, we made the relation between x and basis coefficients of $y(t)$ non-linear.
- We are able to retrieve the polynomial relation with our proposed model:

Function on scalar regression : results

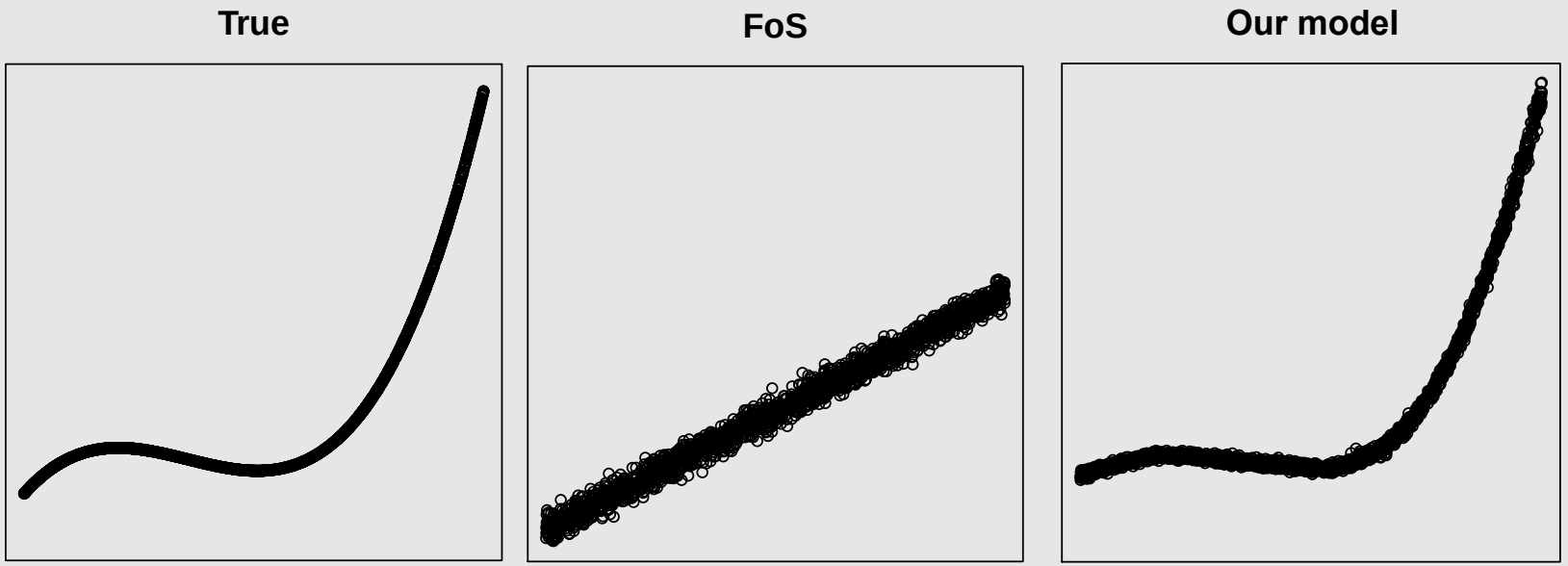

Figure: Basis coefficient as a function predictor X_1

Function on scalar regression : results

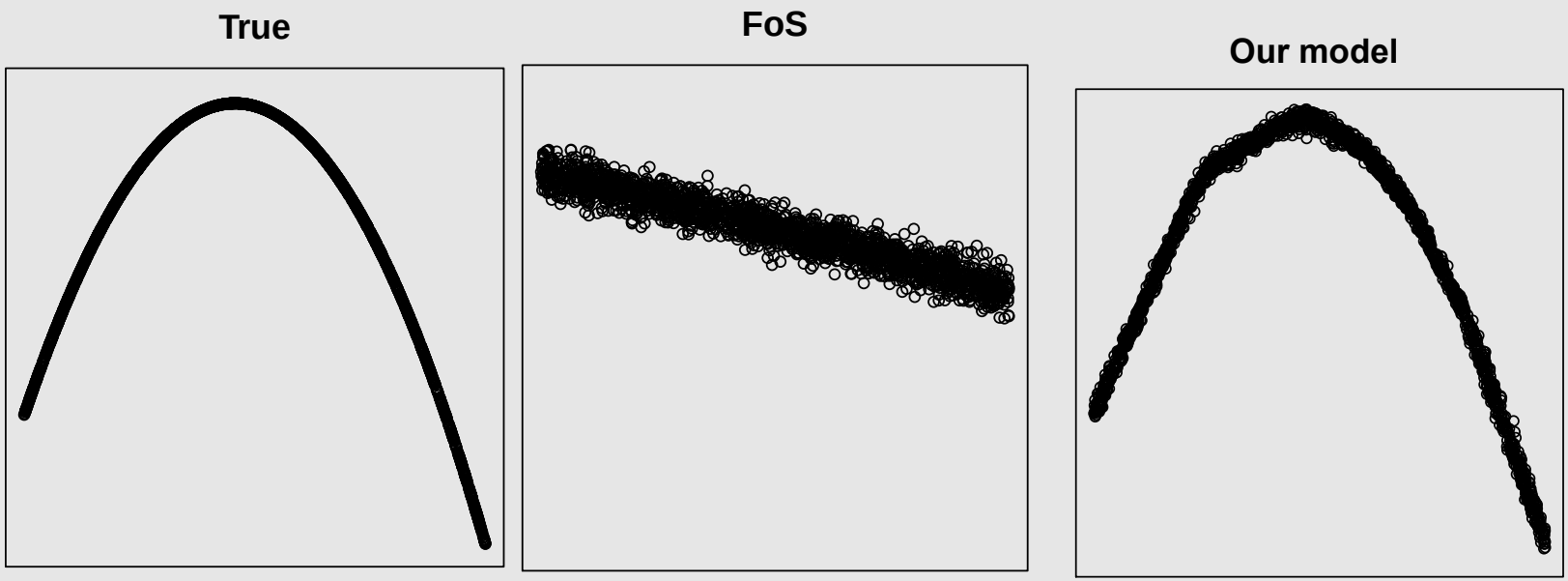

Figure: Basis coefficient as a function predictor X_2

Function on scalar regression : results

Table of MSEs of 20 random test sets. (Design 1)

Methods	FoS (Linear)	Our model (NN)
Mean	49.20	4.95
Std. Dev.	1.64	0.11
p-value	-	$<$2.2e-16

Function on scalar regression : results

Table of MSEs of 20 random test sets. (Design 2)

Methods	FoS (Linear)	Our model (NN)
Mean	4559.20	36.38
Std. Dev.	159.01	18.76
p-value	-	<2.2e-16 6

Function on scalar regression (linear design): results

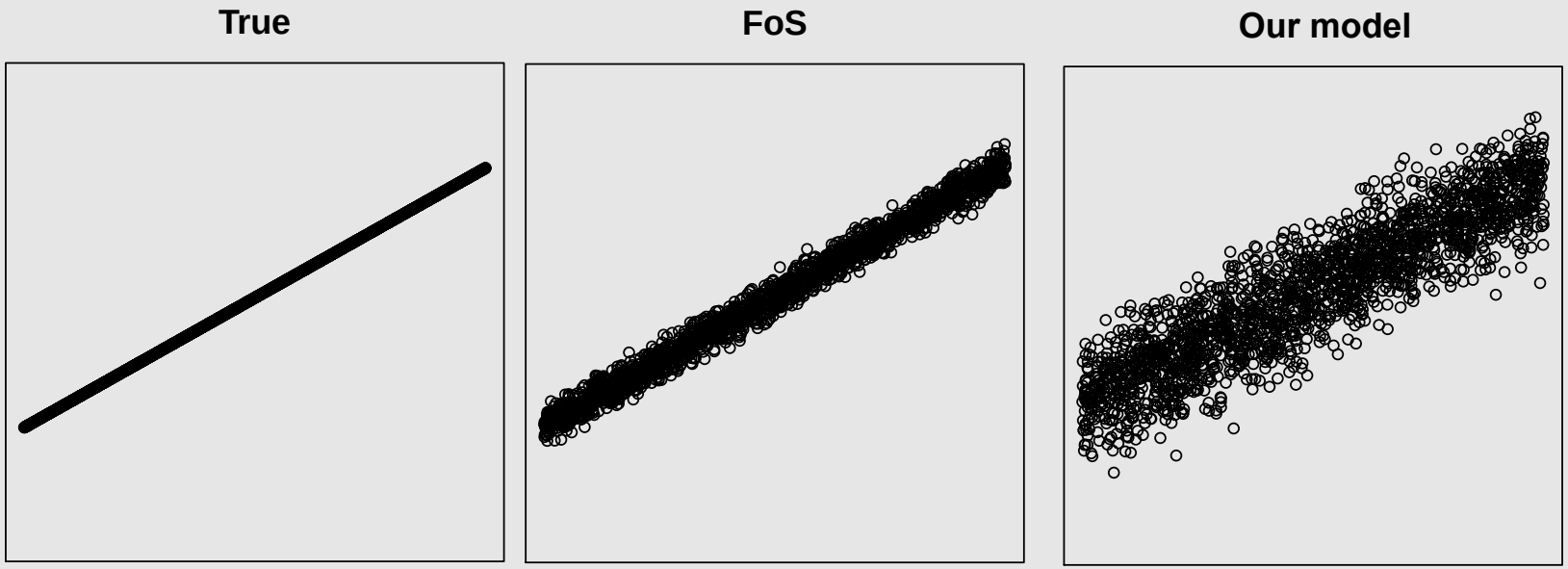

Figure: Basis coefficient as a function predictor X_2

Function on Scalar regression (linear design): results

Table of MSEs of 20 random test sets. (Linear design)

Methods	FoS (Linear)	Our model (NN)
Mean	4.01	4.07
Std. Dev.	0.05	0.05
p-value	-	7.8e-07

Functional neural networks: applications

- Predicting the hourly sugar level of a diabetic over a day given covariate.
- Using the hourly sugar level of a patient over a day to predict the evolution of weight over a year.
- All we need is data, the right data and lots of data.

Conclusion

- We designed layers to input or output functional data in modern neural network architectures.
- It allows any form of functional data to be now part of larger multi-modality deep learning models.
- It respects the assumptions of functional data.

Conclusion

- It adds a layer of complexity and hyper parameters to adjusts.
- The implementations are not user-friendly.

I would love to answer your questions.

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

Morris, J. S. . Functional regression. Annual Review of Statistics and Its Application, 2(1):321–359, 2015.

Wu, S., Beaulac, C. and Cao, J. Neural Networks for Scalar Input and Functional Output, Statistics and Computing, (33), 118, 2023.

Wu, S., Beaulac, C. and Cao, J. Functional Autoencoder for Smoothing and Representation Learning, Under Review, 2023.

Beaulac, C. and Larchevêque, V. Smooth Tunable Convolution: A novel input layer architecture for functional data, in preparation.